Imperial College London

MEng Individual Project

Imperial College London

Department of Electrical and Electronic Engineering

# Developing Assistive Technology for Studying Science Literature for the Visually Impaired and Blind

*Author:*
Rishabh Manjunatha

*Supervisor:*
Dr Edward Stott

*Second Marker:*
Dr Thomas Clarke

A Final Report for the degree of

*MEng Electrical and Electronic Engineering with Management*

June 6, 2020

**Abstract**

The WHO reports with confidence that there are currently over 2 billion people in the world who have some form of visual impairment or blindness. Over 30 million have severe impairment and over 1 million of those are children under the age of 15. The challenges faced by these individuals is unprecedented, simply navigating and understanding the space around them, is a large enough problem in itself. Those who have severe visual impairment or total blindness rely on solutions such as guide dogs, technology and their remaining biological senses to tackle many of these challenges. These solutions become particularly interesting when education is considered.

Advances in hardware and software have paved the way for many solutions both simple and complex; from text enlargers to braille translation software and hardware. These solutions have their merits and drawbacks but they all share one common problem, they rely on the material they are making interpret-able, being digital. Studying paper based, non-digital material is a major challenge and one that this project aims to tackle. It will focus on enabling impaired students to study and pursue their passion for science.

The system developed will incorporate technologies that makes use of Machine and Deep Learning, digital image processing and heuristic methods to capture the paper material being studied. Analyse text, images, graphs, equations and tables within the page and make all of this information available via a user friendly audio and controller based UI. The evaluation of the system proved that for defined sets of material, the system functions admirably and does indeed enable an individual without vision to study science based material.

**Acknowledgements**

Special thanks to my supervisor Dr Edward Stott and second marker Dr Thomas Clarke for their continued support and insightful pointers. I would also like to thank my Father Dr Chikkanayakanahalli Manjunatha, my mother Swarna Manjunatha, Mrs Esther Perea, Akila Sekar and Advaith Sastry for their support during the rough patches of this project so far.

# Contents

# List of Figures

# List of Tables

# Listings

# Chapter 1

# Introduction and Motivation

The WHO (World Health Organization) released a world report on vision in 2019. It recorded with confidence that there are currently at least 2.2 billion people in the world with some form of visual impairment or blindness. Approximately 1 billion people have forms of impairment that were preventable or that have still not been addressed [**?**]. In 1988 the estimated number of severely impaired or blind people in the world, was 37 million [**?**]. Of those 37 million, approximately 1.4 million are children under the age of 15. The major causes of childhood blindness come from malnutrition, diarrhoea, intra-uterine infections and lack of available eye services [**?**].

Great efforts are being made into preventing many of the causes for blindness and in developing proper services for those without access to eye care. Despite this, there are still millions currently living with the condition and the number is continuing to grow. Those who are severely impaired or blind face several major challenges. Most importantly, being unable to properly navigate, understand and interact with the world.

Another major challenge is education. For those who are affected from birth or at a very young age, accessibility to educational resources are often limited or restricted. There are many specialist schools with qualified and trained teachers to help develop specific curriculum's for these children. However, many reports have shown that these teachers still do not believe they have the right resources or ability to effectively teach those who are blind or severely visually impaired [**?**]. In truth, many of these children have great potential to pursue their passion in an academic field just as much as others [**?**]. Expensive equipment, lack of training for teachers and limited funding are some of the reasons why these children are hindered. Scientific subjects such as physics or biology are particularly difficult to teach and to develop good resources for.

Many existing assistive technologies work well to make pre-prepared and digital material easily to interact with and interpret. Technologies such as speech synthesizers, screen or text enlargers and braille translation software do this very well [**?**]. However, these methods are often limited to selected material and only remain usable within the classroom. Current methods to convert textbooks and papers into digital copies have only partially been successful and many books converted into braille can be as expensive as $15,000 for the initial conversion and then $500 per textbook after that [**?**]. Text recognition software is often used to convert non-digital text into speech, however, it is primarily used in a primitive way. It converts the identified text directly to speech without any control or additional processing to make better use of that information for the user.

It is this lack of development in using current advancements in machine learning and image processing that motivates the research and development of this project. There is great potential to develop a system that can provide greater freedom for the visually impaired and blind. A system that will allow them to access and study non-digital paper material, be significantly cheaper than current methods and be designed to truly support their academic development in the sciences, in and out of the classroom.

# Chapter 2

# Background

## 2.1 Visual Impairment and Blindness

Visual impairment occurs when the visual system and it's functions are affected in a negative way, due to health conditions. The most widely accepted standard of measuring visual impairment, is visual acuity. This is a measurement method that tests ones ability to distinguish two high contrast points in space. In most cases this will be identifying various sizes of black and white text at a particular distance away from the eye. The measurement is represented via a fraction, the numerator measures in meters, the distance at which the healthiest eye was able to read a particular line of text. The denominator measures in meters the distance at which someone with normal vision, would be able to read the line. For example, 6/18 would mean that the individual was able to read at a distance of 6 meters, while a normal vision individual was able to read at 18 meters. The standard for normal vision is taken to be 6/6. Someone with a visual acuity of less than 6/60 is considered to have severe visual impairment and one with less than 3/60 is considered to be blind [**?**]. Individuals that fall into these categories, are certified as being severely sight impaired or blind. This means that they have a severely restricted field of vision or no vision at all, and are the main target users of this project [**?**].

## 2.2 Digital Image Processing

Without the immense computing power that has now become so readily available, digital image processing would not be possible. The handling of large images, was unimaginable just 20 years ago. The quantisation of a 1024 x 1024 image at 10 bits, yields an impressive 10 million bits of process-able data [**?**]. This alone illustrates the magnitude of the problem that existed for so long. Despite the criticism around the future of Moore's law, until now it has proven successful in paving the way for exceedingly powerful processors. [**?**]. Algorithms that can take advantage of that, have made digital image processing so prevalent in devices such as smartphones. The major concepts behind digital image processing that apply to this project are as follows:

- Enhancement - This is improving the quality of an image. Often parameters such as noise, sharpness and brightness are adjusted.
- Restoration - This is restoring an image that has been subject to effects such as blurring and distortion.
- Reconstruction - This is reconstructing an image from various 1D data sets, often at varying relative angles to the subject of the image.
- Analysis - This is analysing the data form the image and manipulating it such that it is interpret-able for machine based processing. This could be for further algorithmic processing, display representation or edge detection.

## 2.3 Machine Learning

Machine learning is the ability for a system to learn from experience without the need to directly program it's function. Often the system is given a large data set (training set/data) and it will 'learn' to extract patterns and algorithms [**?**]. There are several main learning methods that are commonly used.

### Association Learning

This rule involves finding relationships between variables in data sets based on measures of interestingness. The algorithm seeks out complimentary associations between variables and their data points and develops relationships based on the frequency and number of these associations.

### Classification

This is another learning method that assigns inputs to a class based on a rule it has learned. This is a form of supervised learning, where by the correct output is given for each input and the system will learn the classification rule. An example is shown below:

$$IF\ X > a\ AND\ Z > b\ THEN\ Y = True\ ELSE\ Y = FALSE \tag{2.1}$$

The major advantage of this type or learning is that is useful when making predictions. Given an input, the system can use the classification to predict the output.

### Regression

This is used to predict number based outputs. The function that relates the output to the input(s) and hidden parameters can be of may types such as linear and quadratic. An example is shown where Y being the output is linearly related to input X and hidden parameters a and b.

$$Y = aX + b \tag{2.2}$$

When the system is trained, the hidden parameters are best optimised with the input variables using an assumed model to obtain the output, this in turn allows the algorithm to predict outputs based on the function it develops. This learning method is also another form of supervised learning. Figure 2.1 gives a visual representation of how regression and classification deal with data.

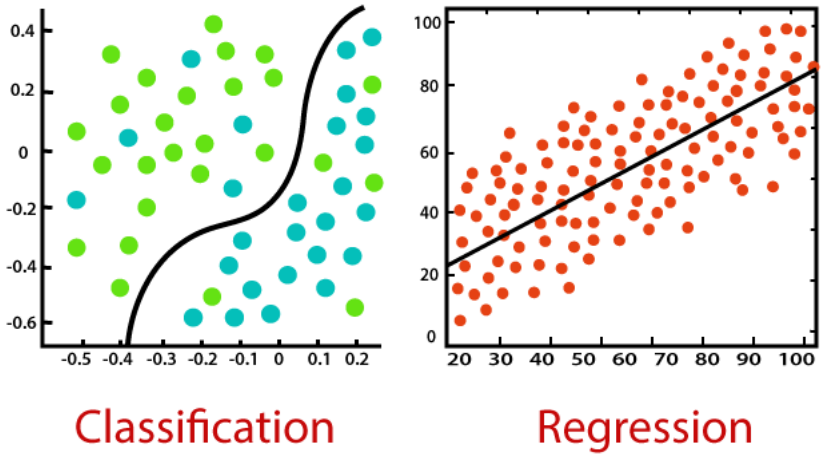



Figure 2.1: Visual Representation of Classification and Regression (Linear) [**?**]

#### Unsupervised learning

This is a method where by the systems acts to model the structure behind a data set. In this situation there is an input data set and no corresponding output. The system is left to find interesting information within the data to develop algorithms itself. To draw a clear comparison, supervised learning provides input data that is labelled and the algorithms have a clear method to predict the output from the input. Unsupervised learning provides un-labelled input data and the system will develop it's own algorithm based on information it forms from the data set. Semi-supervised learning incorporates a mixture of supervised and unsupervised learning, often the input data is mostly un-labelled.

#### Reinforcement Learning

This is based on the concept of outputs consisting of multiple actions. A correct sequence of actions to reach a desired goal is known as the policy. This type of learning is used to assess the effect of each action; good actions will be rewarded. The system uses this reward based learning to generate policies that it then uses make decisions. In many ways this method is similar to unsupervised learning where there are no given outputs and the system must learn from experience in dealing with previous data.

### 2.3.1 Neural Networks

Neural networks are groups of algorithms that are used to represent roughly how the human brain works and are generally used to recognise patterns. They are often used in systems where clustering and classification are required. Clustering is the process of putting together similarities found within data sets, it uses unsupervised learning and therefore, the data does need to have any labels for it to detect these similarities. Classification uses supervised learning and is dependent on the data having labels.

Neural networks are made from nodes, these nodes are simply representative of neurons in the brain. At each of these nodes, a calculation is made, they will take inputs with coefficients that either increase or decrease their value. These values then are summed together and placed through an activation function that decides if the resultant output should proceed to the next layer. Essentially they act like switches, allowing certain inputs to progress through the network. The hidden layer is a term used to describe the layer that acts on the inputs to generate the output, this includes the coefficients, summation and the activation function [**?**].

### 2.3.2 Deep Learning

The major distinguishing factor between deep learning networks and neural networks, is the depth of the hidden layer; figure 2.2 illustrates this difference. It is common for each layer in the deep network to be trained to work on distinct features. Each proceeding layers input operates on the output of the previous, this is known as feature hierarchy. The advantage is that the network can deal with an enormous volume of data and extremely complex features. It can develop this from unlabelled and unstructured data and unlike neural networks that require manual feature identification, they can do this automatically.

It is important to understand the basics behind machine and deep learning and their respective techniques as many of the tools and services that would be used for this project are based on these methods and are the sole reason why many of these technologies exist today.

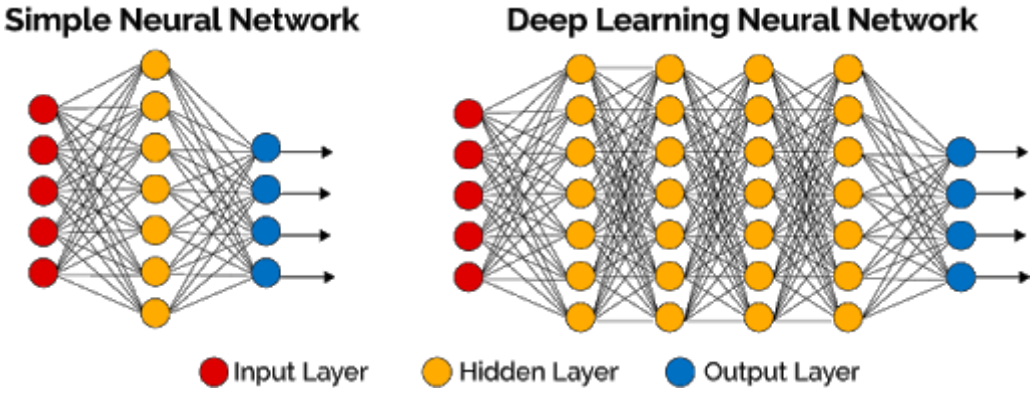



Figure 2.2: Illustration of a Simple Neural Network and a Deep Neural Network [**?**]

## 2.4 Optical Character Recognition

Optical character recognition is an age old computer vision task that can be dated back as far as 1914 [**?**]. With advancements in deep learning methods, OCR has become prevalent in many devices and applications. There are 6 major aspects of text that OCR needs to deal with, these include:

- Text Density - The volume of characters in a given area of an image.
- Structure - How the lines of text are structured in the page, they could be parallel or random depending on the content.
- Languages - Characters and their positions, this varies from language to language .
- Artefacts - Noise, reflections and any kind of interference or disturbance in the image.
- Location - The location of the text within the image.

The first step is therefore to identify the text within the image, taking into consideration the above mentioned attributes. Once the text has been singled out of the image, there are multiple methods that can be used to identify the characters.

#### Classic Computer Vision

Classical computer vision uses a three step process, first it applies filters to create a high contrast between the characters and the background of the image. The second is edge/contour detection that will recognise each of the characters in the image. The third step is classification, where each of the recognised characters are identified.

#### Deep Learning

Unlike classical computer vision methods, deep learning can be used in multiple ways to detect and recognise characters. One common method uses the EAST (Efficient Accurate Scene Text detector) approach for text detection and then uses an LSTM (Long Short Term Memory) to recognise characters. This method is very effective at detecting the features of text within in an image, it is the most common method used to identify text in images using bounding boxes to locate the position of the text. Optical character recognition is an essential part of this project in extracting text based information from the captured page.

## 2.5 Speech

Speech is a complex communication medium, it can be broken down into a number of different classes, the simplest of which are called phones. These are classes of similar sounds, within a continuous audio stream. Diphones are the segments between two phones, triphones and quinphones are classified in terms of context of the audio, and so all of these can have varying definitions. In

order to identify these phone types, senone detectors can be used, generally 4000 of these short sound detectors are used. One common method to identify words within audio streams, is fillers such as breathing, coughing and sounds like 'um' and 'uh'. These form utterances that are used to separate segments of audio between these pauses [**?**].

### 2.5.1 Recognition

There are three models that are used for speech recognition that match the sound samples from the audio source to their corresponding words. In many cases machine learning is used to develop these models; LSTM techniques are among the most common. The acoustic model uses the acoustic properties of senones to recognise words by considering the most probable feature for each phone and the context for each phone. The phonetic dictionary stores a mapping between words and phones. Often only a few pronunciation variations are noted for each word however this model is often more than enough in basic scenarios. Other than using a dictionary, this can also be implemented using a complex machine learning function, often some form of regression. The language model works to limit the number of searches needed to identify the next word by omitting words than are significantly unlikely to follow on form the previous. These three models are used together to form an engine to recognise words from speech.

The high level overview of the process will take the following steps. First the audio is pre-processed to convert the raw audio into numerical data that can be fed into a neural network. The output of the network is fed into a CTC (Connectionist Temporal Classification) loss calculator to calculate the probability of the sequence of characters. For example, samples taken at each timestamp may result in the characters 'CCCAAATTT' being recognised. The probability of the sequence matching the base word 'cat' at each time-step is represented in a matrix. The decoding stage will use tokens and the CTC matrix to determine the word. Repeated letters are collapsed into one character, blank characters (tokens) are placed between two repeating letters to prevent words like 'funny' from missing out the second 'n' [**?**].

This is another essential aspect of the project as this is the sole means by which the user is able to input information into the system such adding notes to pages.

### 2.5.2 Synthesis

Traditional text-to-speech systems use two methods, parametric TTS and concatenative TTS. Newer methods use deep learning models to achieve better results. To measure the effectiveness of these methods, intelligibility and naturalness are used. Intelligibility is the quality of the audio, meaning how clean and clear it is. The naturalness is the quality of the speech, in other words are there characteristics such as correct pronunciation and emotion [**?**].

#### Concatenative TTS

This method uses high quality audio clips that are concatenated together. This method is intelligible but not natural. This is simply because it is difficult to compile a large enough database with all the variations of emotion, tone and such for every single word.

#### Parametric TTS

This method takes a statistical approach, it generates speech by combining parameters such as frequency, magnitude and such, then processing these together. It will begin by extracting linguistic features from the text such as phonemes and then it will extract 'vcoder' features that represent the speech signal. A vcoder simply uses audio characteristics to manipulate other signals, in this cases it will extract features to generate the speech output. Although this method is not resource intensive and in theory, is a lot more able to generate better speech. In practice the speech is neither intelligible nor natural and this is because the parameters that are coded to generate the features for speech are not very good, and this may be because our understanding of what features are needed, may not actually be correct.

### Deep learning TTS

Deep learning models have proven to be extremely efficient at learning the features needed to generate intelligible and natural sounding speech from data. They are able to identify features that are only computer recognisable which is one of the reasons why this method is better than Parametric TTS. It is able to extract features from a data-set, developing its own direct understanding.

WaveNet is an example that uses stacks of convolutional layers with additional connections such as skip and residual, this can be seen in figure 2.3. Without applying conditions to the model, any input will produce a random output. However, if the model is trained to output audio that sounds like humans, it works remarkably well, the speech output is both intelligible and natural. Each generated audio sample is conditioned by all the previous samples and it is also used to generate the next sample based on previous samples. This is what is known as autoregression generation. The major downside to this method is that it is computationally very expensive.

SampleRNN works by using a hierarchy of recurrent layers. Each proceeding layer takes a smaller number of inputs from the previous, till the output produces a single sample. This method is similar to WaveNet, in that it is also an autoregressive generation model that is initially unconditioned. The advantage over WaveNet is that it is computationally faster while maintaining the intelligibility and naturalness.

Speech Synthesis is the sole form of communicating the information from the page to the user. It is therefore essential that the method used is fluid and as easily interpret-able as possible make the system natural and intuitive to use.

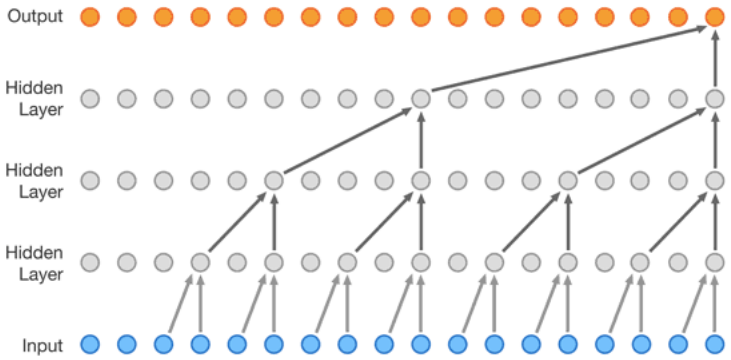



Figure 2.3: Illustration of the WaveNet Network [**?**]

## 2.6 Natural Language Processing

Natural language processing gives computers the ability to understand, interpret and manipulate spoken languages. It falls under the discipline of computational linguistics and has recently seen big advancements with the availability of big data and machine learning. It makes uses of several techniques to interpret the human language including statistical methods and machine learning. Regardless of the method, NLP essentially breaks the language down into smaller segments so that it can understand the relationships it has with other segments in order to form a meaning of the text it is processing [**?**]. The key capabilities of NLP are as follows:

- Content Categorisation - Documenting the contents of a piece of text to allow for searches and indexing.
- Topic discovery - Evaluating themes and meanings
- Contextual Extraction - Extracting information from text-based resources
- Sentiment Analysis - Identifying moods, sentiment and opinions
- Summarisation - Creating summaries for large pieces of text

It is evident how important natural language processing is in understanding, evaluating and processing language based information. NLP will be used in conjunction with OCR to process the extracted information and provide summaries and analysis on the literature.

## 2.7 Active Auditory Interfaces

The concept behind active auditory interfaces, is to use audio based cues to help the user navigate complex pieces of information. For visually impaired and blind people, understanding and navigating a graph is extremely difficult. Graphs contain an immense volume of information in a small area and representing that information in a non-standard format has been timidly explored. Current attempts have reprinted them as tactile graphs using special printers, drawn from the idea of braille. However this method is expensive and often difficult to work with. Passive auditory exploration does do a better job at representing that information, but provides limited user input and lumps all of the information together without any structure.

Using an active auditory interface, it is possible to break down the contents of a graph and represent each segment of information in the form of audio cues that are easy to navigate. Preliminary research has found that in general, people are very good at detecting variations in sound such as tones with changes in pitch and volume. In an experiment it was used to represent edges, curves and distances between points and the axis', and it was to found be extremely effective among those in the trial. Figure 2.4 illustrates what the sound was representing. Users were able to locate 4 points on a graph as quickly as 57 seconds and were able to retain a significant portion of the information they extracted navigating and understanding the graph [**?**].

### 2.7.1 Sound Synthesis

Sound is simply the propagated vibration of a wave through a medium. Often that wave is known as being acoustic in nature and has parameters such as the frequency and amplitude that affect the way it sounds. General synthesis techniques use what is known as fixed-waveform synthesis, this method generates a periodic signal based on predefined parameters for X amount of time: this is essentially the digital form of an oscillator [**?**]. This can be extended to more complex wave-forms that use frequency and amplitude modulation and waveform shapes such as square and saw-tooth. For the active auditory interface mentioned above, fixed-waveform synthesis can be used to generate the variations in pitch and volume for each axis.

For this project, active auditory interfaces will be essential in representing information extracted from graphs. The exact usage and method will differ, but the concept of breaking down the content and using audio cues with intuitive navigation should be used.

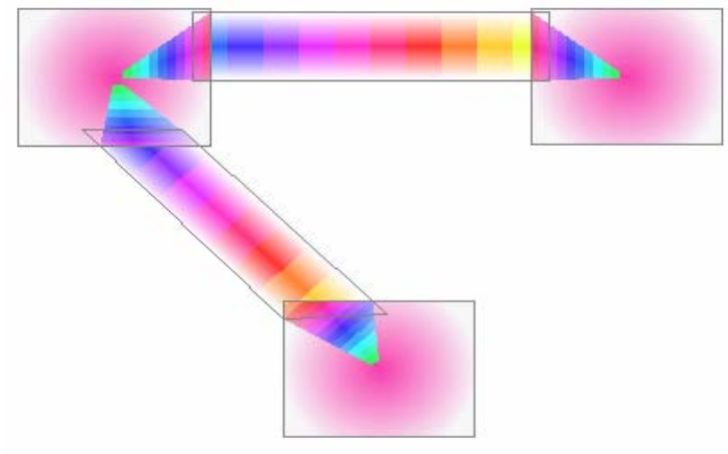


Figure 2.4: MIDI guidance of a 2 edged graph illustrated using HSB colour [**?**]

## 2.8 Existing Solutions

There are a range of existing solutions that are used by individuals and schools to help the impaired with not only accessing information but also studying it. The main categories amongst all of these are discussed below.

### 2.8.1 Enlargers/ Magnifiers

For individuals who are partially impaired, text and video enlargers are extremely useful. These solutions take digital text or a live camera feed and simply enlarge or magnify the input several times to make it easier to view. These have been implemented in many areas, the most common is within smartphones and computers. While these software solutions are free on the devices they are used on, many external video based enlargers are extremely expensive. The Acrobat HD Ultra for example is priced at 1,945.00 [**?**], it is fairly large and can only be used where it is placed, such as classrooms. Although smaller devices do exist such as the AUMED IMAGE, they are still expensive at 418.80.[**?**]


(a) Acrobat HD Ultra

(b) AUMED IMAGE

Figure 2.5: Enlarger/ Magnifier Examples

### 2.8.2 Text-to-Speech

Text-speech has been one of the most prevalent solutions in devices for making non-verbal information available to the blind. It's use is widespread everywhere from computers to accessibility help points at train station platforms. All of these methods use predefined text or extract text from a source and convert that into speech to be played back to the user. With the advancements in machine learning as mentioned before, speech synthesis has become more and more natural and intelligible making this a very effective solution.

This method often does not cost much other than the devices they are used in, however, the biggest problem is control. Many applications such as the KNFB reader [**?**], are able to detect text from images but either present very little control over the way the text is presented to the user or are difficult to use. Some applications allow the user to touch and highlight the text to be translated but this relies on the user being able to identify the right piece of text initially. This results in partial conversion and isn't as effective as dividing larger chunks of text into sentences or sections that can then be chosen by the user more easily.

### 2.8.3 Speech-to-Text / Braille-to-Text

Speech-to-text is the most common method used for data input, such as taking notes. Many devices or applications that do this, simply take whatever the user is saying and converts that into text which is stored. This can be used to perform basic tasks such as note taking, emails or creating grocery lists and even basic commands in smartphones such as asking for the weather. This technology has been easily and cheaply implemented into smartphones and PCs, however, accessing that information once stored isn't standardised. Some devices can transfer that information to a computer and then the user can use magnifiers to re-access that information. Other devices can convert that text back to speech and others can convert this to braille.

A less common input method, is braille. This is a device that takes braille input from a user and converts this to text. While this has some great benefits such as greater accuracy, they are all extremely expensive, for example the BrailleNote Apex is priced at $1,995 [**?**] and these devices tend to be quite bulky.


Figure 2.6: BrailleNote Apex

### 2.8.4 Braille

Braille is method by which characters are represented by an array of raised dots on a page. Touch reading is the method of using ones fingers to read the pattern of raised dots to read text. Braille solutions has seem multiple advancements in recent years, from braille textbooks and printers to braille note takers and converts. Braille textbooks and printers simply translate text into braille on a physical page. As previously mentioned, these methods have only partially been successful and many books converted into braille can be as expensive as $15,000 for the initial conversion and then $500 per textbook after that [**?**]. Braille devices that can convert text from a computer to braille in real-time are extremely impressive but again are expensive and the same can be said about braille note takers such as the BrailleNote Touch 18 Plus which is priced at $4,195.00 [**?**].

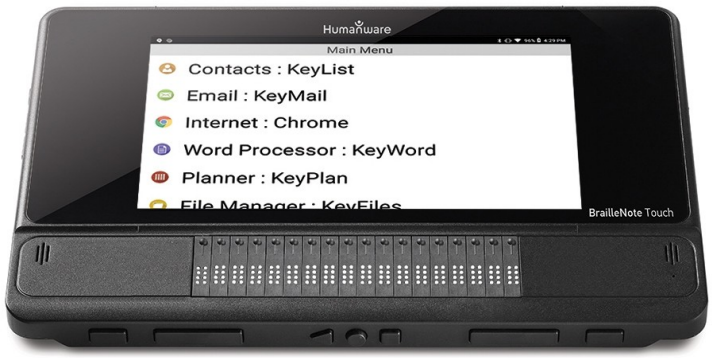



Figure 2.7: BrailleNote Touch 18 Plus

# Chapter 3

# Project Specification

## 3.1 Project Definition

Considering the background research, it then becomes easy to determine that this project can be defined clearly by looking at the major issues it currently faces and aims to resolve.

1. The first is that, although there are various resources that have been written from scratch in a digital format, many pieces of literature are or have been printed due to convenience and availability and this becomes more relevant the further back in time these resources were published. Despite the prevalence of computers and smartphones being used at school and universities, almost all fundamental aspects of the curriculum still use written papers notes and textbooks. The system must therefore be capable of identifying and capturing non-digital material in the form of a paper page through the use of a camera.

2. The second issue, is how existing assistive technologies extract and represent information such as graphs, images, tables, equations and text. Considering the latter, current solutions simply recognise and output word for word the text that is extracted from a page to speech. While this many be effective at interpreting small pieces of text, this method becomes unusable for larger pieces such as paragraphs or pages. There is no easy way to control how the extracted information is presented to the user. While text extraction is fairly simple, graph extraction is not, therefore this system must be capable of both, capturing and representing all of the above mentioned forms of information in an easy and effective manor.

3. The third issue, is that there are limited features and functions in existing devices that allow these students to take notes and analyse text as one would by underlining, highlighting or 'bolding' important information. This is an essential skill and tool that is necessary when studying any piece of literature, scientific or otherwise and should be implemented directly into the system rather than separately like many existing devices.

4. The fourth issue is that many of the devices that students currently use such as the "Braille N' Speak" or PEN based tablets, are incredibly expensive. The 'Perkins Classic Brailler', used for making braille notes, is currently priced at č744 including VAT [**?**]. This system must therefore be as cheap as possible to enable greater accessibility for individuals and schools.

Emphasis is drawn to making this system as easy and intuitive to use as possible. Without the aid of vision, touch and sound become the most important forms of communication. It is therefore essential above all that the virtual interface and input methods are robust and well structured. It is also important to note that this system should be made use-able by normal sighted individuals, to ensure that it is easy to train and teach.

## 3.2 Top Level Specification

The end goal of this project is to have a prototype piece of software that can at a minimum use a camera feed to capture the page to be studied and then process that page to extract the following:

- Text - All text present within the page
- Graphs - All graphs/plots within the page - This is limited to graphs that contain curves and excludes all others such as bar and pie charts.
- Images - All images (this is includes sections of the page that do not fall into any of the other categories)
- Equations - A defined set of equation types - This includes 1st to 3rd order polynomials, trigonometric functions, exponentials and logarithms
- Tables - All tables within the page

The extracted information should be made available through a virtual interface, that is navigated using a physical controller and speech, these methods are used for both input and output. The information must also be stored for the student to access at a later date, so they that they do not need to recapture previously studied material. The system must also use services and tools that will minimise the capital and running costs of the system.

## 3.3 Software Specification

This project is almost entirely software based and it's structure is essential to ensuring that it operates efficiently, is as openly compatible with hardware possibilities and has a smooth, predictable and intuitive interface. As this project takes the form of a prototype, gaining access to as many resources and tools is essential. As such, the system will be programmed using Python which will enable quick, easy and well documented access. The nature of this project also means that there are a significant number of individual extraction systems that must operate as one, these can be categorised exactly as the list above.

These must be accompanied by a control system that uses at a minimum a physical controller to navigate the interface and access all the information. The system must also store all the pages that the user studies in a format that can be accessed at a future point in time should the user wish to revisit previous material.

## 3.4 Hardware Specification

In order to enable the use of Python and to keep the costs as low as possible, the system must be able to run on the latest Raspberry Pi 4 along with the Raspberry Pi Camera Module v2. This would bring the total expected hardware costs to 58.

## 3.5 COVID 19 Adjustments

The impact of COVID 19 on this project is minimal. As this project is almost entirely software based, the only adjustment that has been made is running the system on a traditional PC and utilising the webcam rather than the proposed Raspberry Pi and Pi Camera. Though this does mean that the prototype won't be in it's intended final form, it does not hinder it's functionality and capabilities and can still be evaluated.

# Chapter 4

# Implementation

The nature of this project tends for an implementation process that is quite iterative. For each aspect of the system, the best tools, resources and services are tested against each other in order to find the optimal solution.

## 4.1 Document Scanning

In order to capture the page the student wishes to study, the system must be capable of performing what is essentially document scanning. The widely accepted and renowned way to achieve this is by using the OpenCV library. This is a computer vision and machine learning library that contains countless modules for a variety of applications, everything from video analysis to object detection. For this application, it can be used to identify and extract a document from a live video feed.

### 4.1.1 Image Processing

In order to detect the page within the image, there are several image processing steps that need to be carried out in order to identify certain features. The end goal is to have an image that makes edge detection easy so that contours within the image can be formed to find the page. These operations can all be achieved by using builtin function in the OpenCV library.

The first step is to apply a grey scale method to the image. There are many reasons why this is done, firstly, colour information is not particularly useful for performing other processing techniques. It can affect the signal to noise ratio and converting an image to greyscale helps improve this. Secondly, working in greyscale reduces the amount of computation resources required because of the reduced information density. Finally other APIs and services that are to be used for extracting information require the image to be in a grayscale format and this simply acts in favour of this. One additional benefit is that visualisation of the proceeding processing methods are clearer and makes fine tuning easier.

The second step is to apply a smoothing filter. The three most common methods are averaging, median and Gaussian. All of these methods are used to reduce the noise in an image. Median filtering is known for retaining good sharp edges and performs a good job at reducing noise if the image is very noisy. For this application, the images will not be incredibly noisy and hence the other 2 methods are favoured. Averaging filters will take the average of neighbouring pixels. Gaussian filters will assign weightings to pixels based on distance and hence is, in general, better at reducing noises within images. It has better frequency separation and Gaussian noise reduction properties. This is the method used in this application.

The final step is to apply some form of edge detection. There are 2 main methods, the first is using a Canny Threshold which uses multiple algorithms at different stages to detect single pixel edges within an image. This is a standard method that is used in many applications and requires the threshold values for the pixel values to be manually defined. Adaptive threshold is a method by which the threshold values are calculated based on smaller regions of an image rather than the

entire image. Both perform admirably, but for this application it was found that for the majority of pages tested, Adaptive threshold seemed to yield more consistent results and hence it was used.

### 4.1.2 Document Detection

Now that the image has been processed the next step is to find all the contours within the image, this is simply all the lines within the page that can be connected by pixels of the same value. OpenCV has a built in function that uses structural analysis and moments to find these contours. These contours are then fed into a loop that attempts to estimate the curve of the contours and tries to fit the largest of those contours into a rectangle.

At this point, the document has been detected but conditions are set to ensure with some confidence that the contour is indeed the document. The first condition is that the area of the contour must be of a certain percentage of the total area of the image. This serves 2 purposes, it ensures that smaller objects are rejected such as tables within pages and encourages the user to capture the document to fill the image as much as possible to retain detail. The second condition is time; the reason being that from testing, it was found that jittering was very common. With a video feed operating at 24 frames per second, it was possible that each frame would return a different contour. Setting a time based threshold means that the contour must be consistent over certain period in area in order to be captured. These two conditions proved to be very effective when set at 75% of the total area and when the contour is consistent for at least 3 seconds.

### 4.1.3 Warp and Transform

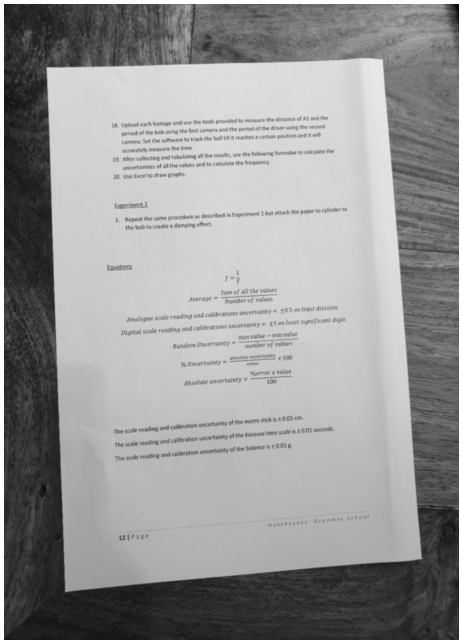
The final stage is to perform a warp and perspective transform to correct the slant and angle of the document within the image. The perspective transform determines the points of the contours and their relative position to the defined edges of the output. The warp perspective uses this information to warp the image to form the output. This will ensure that the contents of the image remain un-warped after the transform by acting against it.

### 4.1.4 Testing

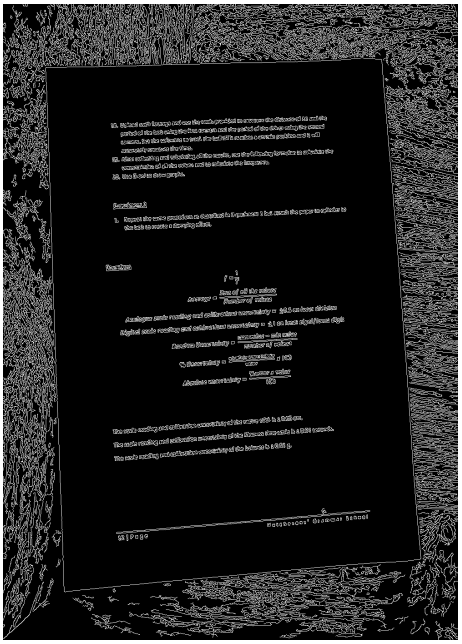
The testing was carried out in 3 categories each altering the position of the document within the image and the background. This was done to assess the systems capabilities under imperfect conditions.

- Straight - The document is position parallel to the edges of the feed
- Slanted - The document was position at an angle to the edges of the feed
- Complex Background - The document is placed amongst other item such as papers, sticky notes and stationary

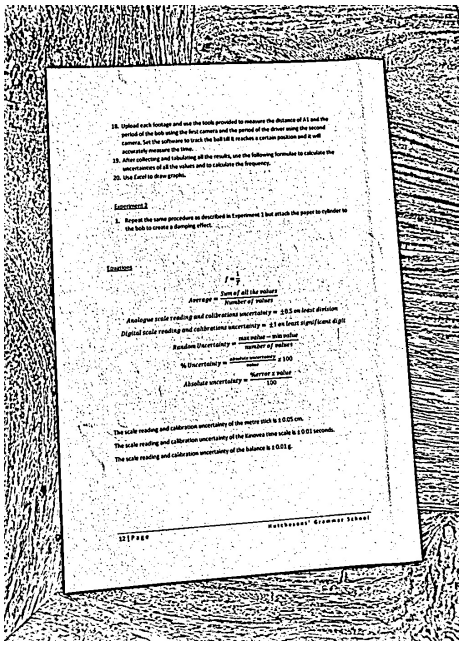
Within each category there are 25 test feeds each with different documents that contain different information such as varying text, graphs and images. This is to test the systems ability to handle complex documents that may contain smaller contours in graphs and tables within them. Each video feed is approximately 15 seconds. An example of the output for each test case is shown below along with the key steps mentioned in the previous sections.



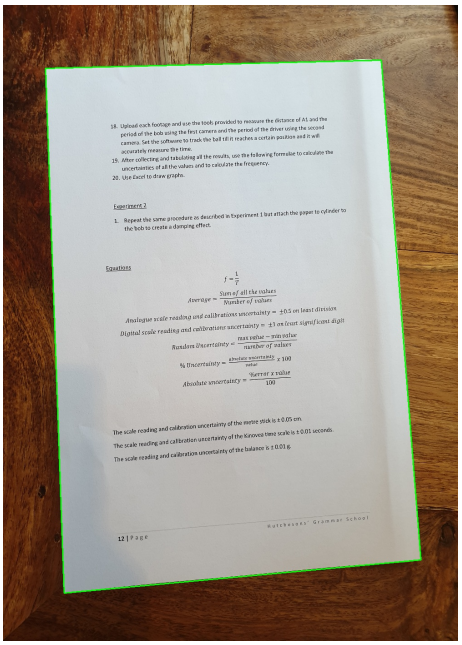
(a) Grayscale and Blurring



(b) Canny Threshold

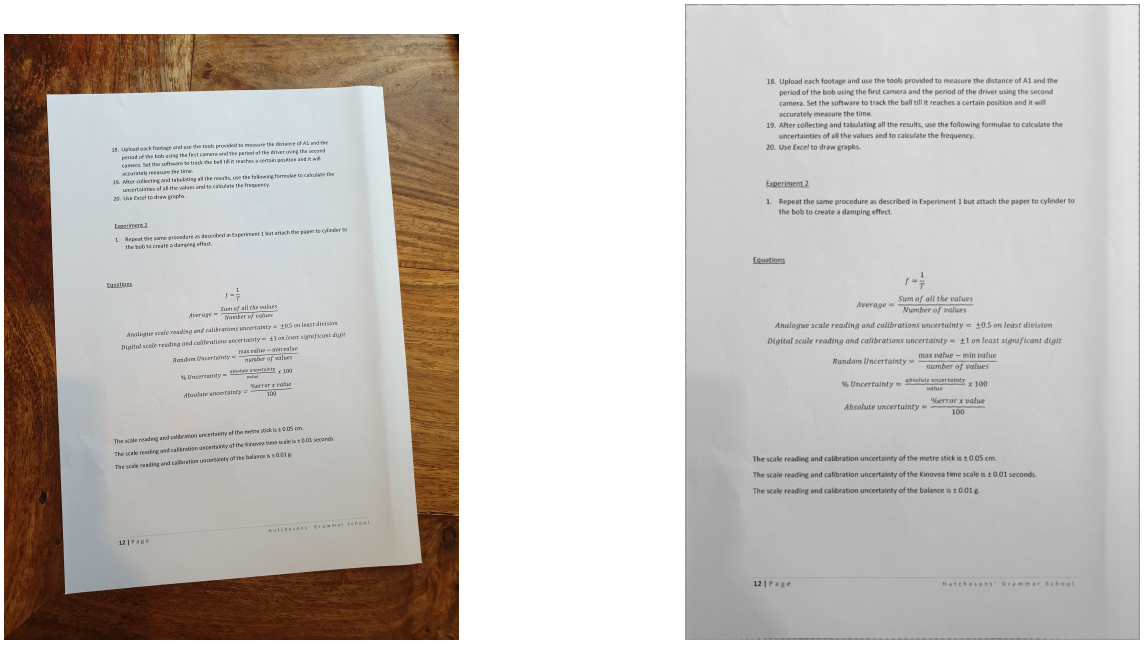


(c) Adaptive Threshold



(d) Document Contour

Figure 4.1: Document Scanning Stages



(a) Original

(b) Document

Figure 4.2: Document Scanning Input and Output

The results from the tests are as follows:

| | **Straight** | **Slanted** | **Complex Background** |
|---|---|---|---|
| **Accuracy** | 92% | 90% | 72% |
| **Time** | 5.3 | 5.4 | 6.4 |

Table 4.1: Document Scanning Test Results

It's clear to see that it performed as we would expect. With the document in near perfect conditions, it was detected very easily in the shortest amount of time. When slanted the performance dropped marginally. Considering the documents that it was not able to detect, a simple solution to this problem would be informing the user to move and readjust the camera to try and detect the page from a different height or angle after a certain period of false detection.

The final category, complex background, saw the biggest drop in performance, specifically in accuracy. From the example shown in Figure 4.3, it's clear to see that with more more complex items within the image, it's easier for the system to fail with what is essential additional noise. In addition to the solution mentioned above it would be beneficial to instruct the user to try and clear away any material that may be near the page that is being scanned. These measures will ensure a greater accuracy rate that than recorded in these tests.

It is important to know that these tests are working in an enclosed environment, with user feedback and controls, the user can be better instructed to position the camera to capture the document with greater success. This will be tested and evaluated later.

With the document now scanned and ready to be processed, the following components will now look at methods of extracted information from the document and making that available for the user to study.

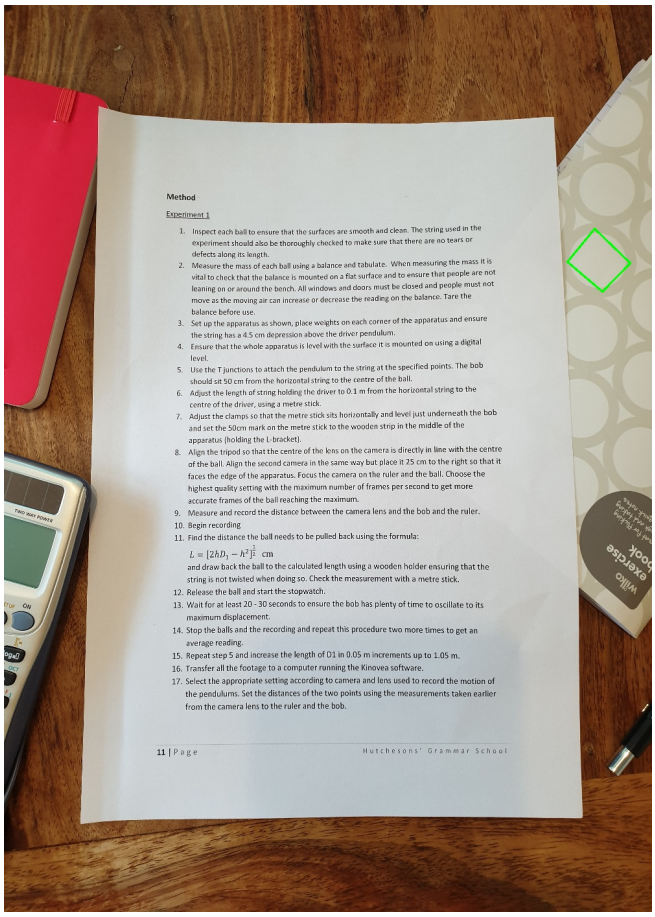



Figure 4.3: Complex Background Example Output

## 4.2 Block Extraction

Now that the document has been scanned, the next step is to breakdown the page into several blocks. Each block is defined as an area of the page with segmented information. These include, images, graphs, paragraphs, equations and text. The purpose of extracting these blocks is to make processing each type faster and easier. The text extracted in this however, will not be used, as will be shown later, this is by product of the extraction that can be easily ignored. This is a very important component of the system as many other processes are dependent on the successful extraction of these blocks.

### 4.2.1 Image Processing

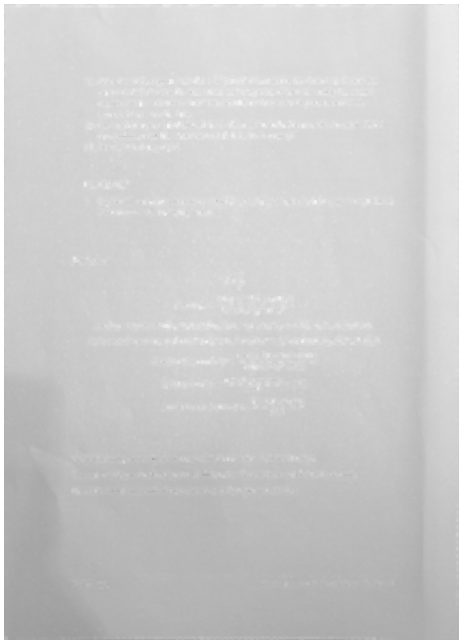
The first step is to process the image to remove any shadows or illumination from the image. This will enable better clearer image to be processed. In order to do this, it must be insured that the information within the page is retained. Dilation can be used to preserve this information, this is the process of convolving the image with a kernel that will essentially expand certain pixel values into their neighbouring pixels. Performing dilation on the background will expand the those pixel values over the information with in the page which is typically dark gray or black.

To further remove the text from the page to improve accuracy, the image can be smoothed using a blurring method. As explored previously, from the 3 most common smoothing methods, the method best applicable in this situation is the median filter. This will do a good job of removing noise, which in this case is the information within the page. As the dilation will have removed all sharp edges, the properties of the median filter that retain this will not affect the output which is what makes this most effective.

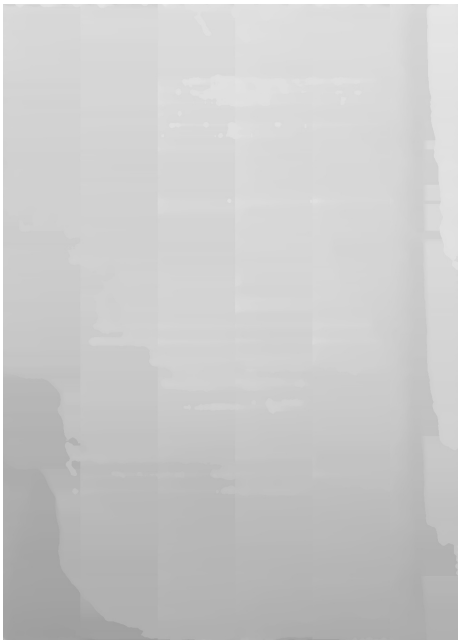
The image is now essentially comprised of just the background, that is all the shadows and highlights in the page that we wish to remove. The process is now as simple as subtracting this from the original image. This results in an image that is much clearer with the text and background better defined. This processing can leave patches of grey pixels, and does indeed reduce the dynamic range.

Normalising the image will redefine the range of pixel values that the image can take, this will return the dynamic range back to normal. Truncating the image will apply a new pixel value to the image above a certain threshold. This is set to remove grey pixels and convert them to white pixels.

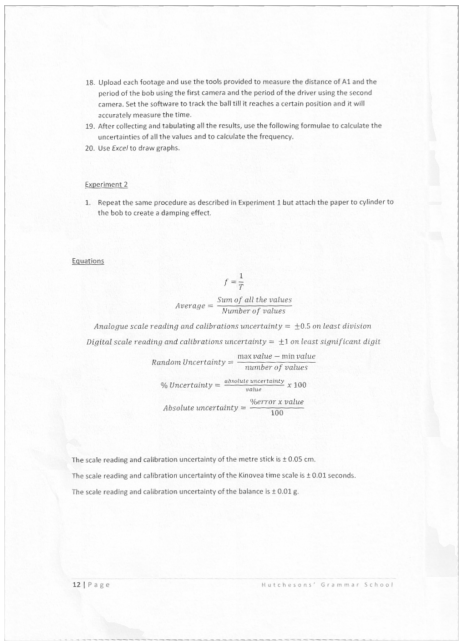
The image is then converted back to grayscale and inverted to obtain a black and white image with the text in white and background in black. This is performed using Otsu threshold and binary inversion. Otsu threshold is a method whereby an algorithm automatically calculates a threshold value that will separate in two, the pixel values with a grayscale image. The goal is to minimise the spread of pixel values on either side of the threshold. The binary inversion then uses this threshold value to convert the image into black and white. The output from each stage can be seen in Figure 3.



(a) Dilation



(b) Smoothing


18. Upload each footage and use the tools provided to measure the distance of A1 and the period of the bob using the first camera and the period of the driver using the second camera. Set the software to track the ball till it reaches a certain position and it will accurately measure the time.
19. After collecting and tabulating all the results, use the following formulae to calculate the uncertainties of all the values and to calculate the frequency.
20. Use *Excel* to draw graphs.

Experiment 2

1. Repeat the same procedure as described in Experiment 1 but attach the paper to cylinder to the bob to create a damping effect.

Equations

$$f = \frac{1}{T}$$

$$Average = \frac{Sum\ of\ all\ the\ values}{Number\ of\ values}$$

$$Analogue\ scale\ reading\ and\ calibrations\ uncertainty = \pm 0.5\ on\ least\ division$$

$$Digital\ scale\ reading\ and\ calibrations\ uncertainty = \pm 1\ on\ least\ significant\ digit$$

$$Random\ Uncertainty = \frac{max\ value - min\ value}{number\ of\ values}$$

$$\%\ Uncertainty = \frac{absolute\ uncertainty}{value} \times 100$$

$$Absolute\ uncertainty = \frac{\%error \times value}{100}$$

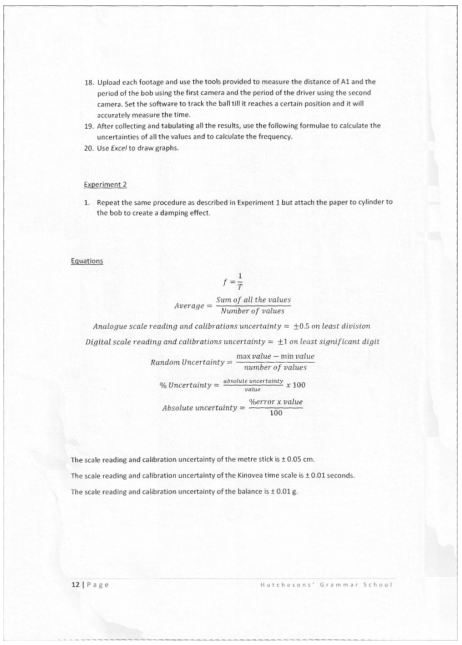
The scale reading and calibration uncertainty of the metre stick is ± 0.05 cm.

The scale reading and calibration uncertainty of the Kinovea time scale is ± 0.01 seconds.

The scale reading and calibration uncertainty of the balance is ± 0.01 g.



(c) Subtracted Background


18. Upload each footage and use the tools provided to measure the distance of A1 and the period of the bob using the first camera and the period of the driver using the second camera. Set the software to track the ball till it reaches a certain position and it will accurately measure the time.
19. After collecting and tabulating all the results, use the following formulae to calculate the uncertainties of all the values and to calculate the frequency.
20. Use *Excel* to draw graphs.

Experiment 2

1. Repeat the same procedure as described in Experiment 1 but attach the paper to cylinder to the bob to create a damping effect.

Equations

$$f = \frac{1}{T}$$

$$Average = \frac{Sum\ of\ all\ the\ values}{Number\ of\ values}$$

$$Analogue\ scale\ reading\ and\ calibrations\ uncertainty = \pm 0.5\ on\ least\ division$$

$$Digital\ scale\ reading\ and\ calibrations\ uncertainty = \pm 1\ on\ least\ significant\ digit$$

$$Random\ Uncertainty = \frac{max\ value - min\ value}{number\ of\ values}$$

$$\%\ Uncertainty = \frac{absolute\ uncertainty}{value} \times 100$$

$$Absolute\ uncertainty = \frac{\%error \times value}{100}$$

The scale reading and calibration uncertainty of the metre stick is ± 0.05 cm.

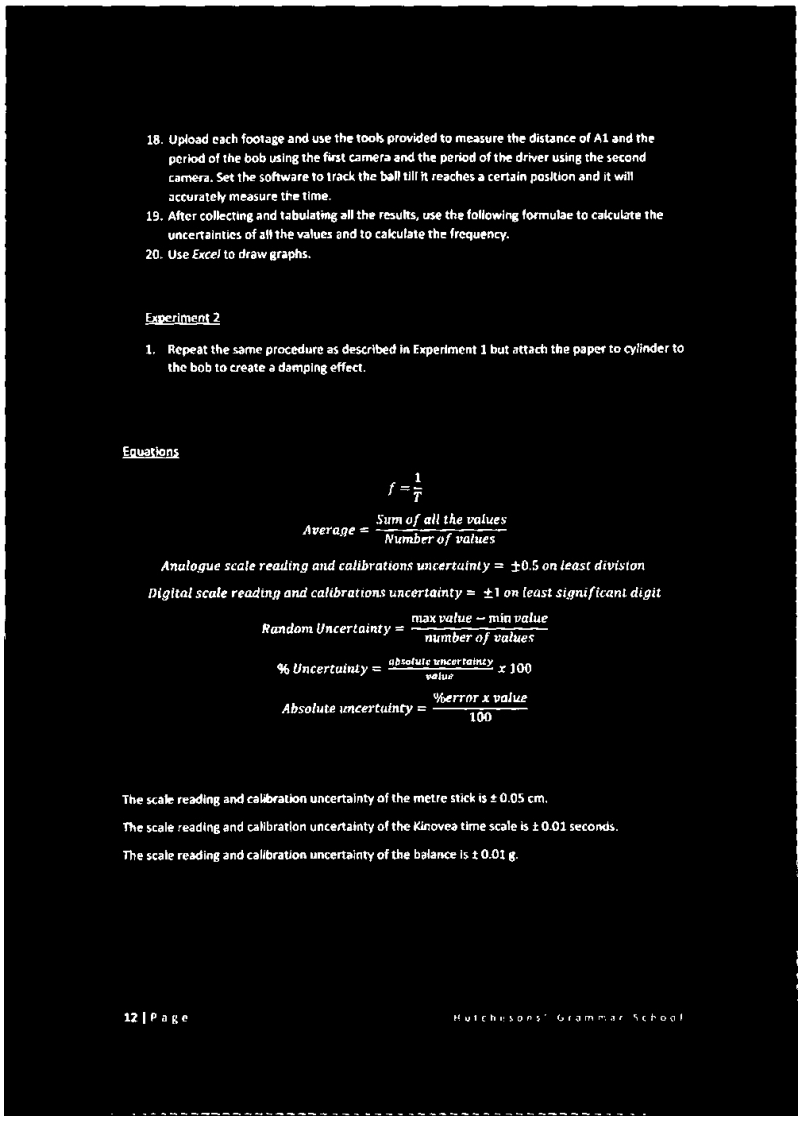
The scale reading and calibration uncertainty of the Kinovea time scale is ± 0.01 seconds.

The scale reading and calibration uncertainty of the balance is ± 0.01 g.

12 | Page Hutchesons' Grammar School

(d) Normalised and Truncated

Figure 4.4: Block Extraction Image Processing



18. Upload each footage and use the tools provided to measure the distance of A1 and the period of the bob using the first camera and the period of the driver using the second camera. Set the software to track the ball till it reaches a certain position and it will accurately measure the time.
19. After collecting and tabulating all the results, use the following formulae to calculate the uncertainties of all the values and to calculate the frequency.
20. Use *Excel* to draw graphs.

Experiment 2

1. Repeat the same procedure as described in Experiment 1 but attach the paper to cylinder to the bob to create a damping effect.

Equations

$$f = \frac{1}{T}$$

$$Average = \frac{Sum\ of\ all\ the\ values}{Number\ of\ values}$$

$$Analogue\ scale\ reading\ and\ calibrations\ uncertainty = \pm 0.5\ on\ least\ diviston$$

$$Digital\ scale\ reading\ and\ calibrations\ uncertainty = \pm 1\ on\ least\ significant\ digit$$

$$Random\ Uncertainty = \frac{\max value - \min value}{number\ of\ values}$$

$$\%\ Uncertainty = \frac{absolute\ uncertainty}{value} \times 100$$

$$Absolute\ uncertainty = \frac{\%error \times value}{100}$$

The scale reading and calibration uncertainty of the metre stick is ± 0.05 cm.

The scale reading and calibration uncertainty of the Kinovea time scale is ± 0.01 seconds.

The scale reading and calibration uncertainty of the balance is ± 0.01 g.



(a) Otsu Threshold and Binary Inversion

### 4.2.2 Block Detection

The most effective way of identifying blocks together is by grouping closely positioned entities within the page. Determining and grouping these can be achieved by applying the same dilation method used previously but in reverse. Instead of dilating the background, the foreground is dilated and expanded to overlap nearby entities. Before dilating, the kernel shape is set to be rectangular. This will aid in finding the contours and works with the standard assumption that text information with documents tends to cascade down the page in lines which can be approximated to long thin rectangles. The kernel size must also be set, common kernel sizes tend to be 3x3 matrices. As resolution of the images obtained is quite high, approximately 1000x1400, this size is far too small to join foreground entities together. Iterating through various values and testing against a variety of documents yielded 2 kernel sizes.

In both cases the kernels are capable of capturing and obtaining all but 2 types of blocks, graphs and images. These two kernel sizes and the number of iterations that the dilation is performed on the image allows us to obtain both graphs and images in 2 passes. For images, the optimal kernel size was larger and needed more dilation iterations. For graphs, the optimal kernel size was smaller and only need a single iteration. Understanding why these values are different comes down a simple observation. Images tend to have a variety of curves and small areas within a page that are entirely filled. Having a larger kernel allows for odd blank spaces with images to be filled and dilating for the second doubles down on this.

Graphs are more complicated, for data extraction, it is more beneficial if there are fewer words and text surrounding the plot area such as axis titles and keys. This smaller kernel with a single dilation pass is large enough to allow for the curve and axis to merge but small enough to exclude unnecessary items. It must be noted that while images tend to have similar structures, graphs do not. It is not easy to define a set parameter that will work every and all cases. It is understood that this is an inherent limitation. The values tested are based on the most common 'style' of graphs often generated from Excel, Matlab and alike that are widely found in reports and papers. These parameters are left accessible for adjustments.

The same contour finding method used for document scanning is used here. In the same way that the area of the contours was used as one of the conditions for correctly identifying the document. A similar conditional method is used. There are several considerations that need to be made first. As the document is almost perfectly fitting to the dimensions of the image, it is possible that OpenCV will detect the entire page as a block, this will occur if small areas around the edge of the document still contain some of the background. This can be avoided by comparing the horizontal and vertical lengths of the contours with the length and and height of the image. If the lengths are above 90% of the dimensions of the image then it is not possible for contour to be a block but rather a page or an invalid detection. This is because even with standard narrow margins in documents, there can be no block that has dimensions greater than that percentage of the document.

The second consideration is what conditional statements can be used to determine which of the blocks contain the image or the graph. There are many complex methods that can be used but they are require additional computation power such as per block OCR or machine learning. These methods are time and resource expensive and faster simpler methods are preferred. This problem can be looked at by considering the area, and the width divided by the height of each contour.

Conditional statements can be used to determine which blocks belong to which category. These conditions have been determined as such, the blocks can be broken into two categories, small and large; in general text and equations within a page will have a width that is much larger than the height. They will also tend to have small to medium areas. Smaller text items within a page such as page numbers and titles may not have a width that is much larger than it's height but it will have a very small area compared to other items within the page. Therefore these fall into the small category. The smallest possible image or graph is assumed to be equivalent to a quarter of the

area of the page within the largest margins. These tend to have almost equal width and height and large areas, In general however, these items tend to assume a large portion of the document and hence these fall into the large category.

It is quite clear, once again that this solution will not work for every and all cases. These are limitations that come with the inherent nature of documents that can take on a multitude of different forms. The idea is to design a system that works for set of document styles, that can be built upon as the foundations for a system that can tackle a wider range of material that will expand it's scope and abilities. With this in mind, testing this script will reveal based on simple statistics what the conditional values should be for the above category.

Separating these into two categories will perform 3 operations, in the first pass of the script it will extract all the blocks within the page, and it will extract any images. The second pass with the second set of kernel values will extract the graph. It is noted that within this, the table and equations will be extracted amongst the text blocks. As will be seen later, this poses no issues in separating them.

Once the graphs and images are detected, they are removed from the page by applying a white fill to the entire contour area. This will aid with OCR operations later in the system as it removes the errors that arise in detection text with in the graph such as axis values. The graphs and images are then saved to specific locations within the system to be processed further. The remaining blocks are cut and placed in a different folder to be processed for tables and equations.

### 4.2.3 Testing

A test base of 50 documents all containing a range of text, images, graphs, tables and equations were used. The aim of the test was to determine and adjust all the parameters within the script to reach an optimal success rate. The results and values obtained from testing can be seen in the following 3 tables. The results are relative to a fixed image definition of 1000 by 1414 pixels, this is a ratio for a standard A4 page taken to fit the maximum possible resolution process-able on this particular machine without introducing errors. For each test result, the blocks where broken down into the 3 categories, graphs, images and other. The correct output was compared with output from the script.

| | **Image Kernel Size** | **Image Iterations** | **Graph Kernel Size** | **Graph Iterations** |
|---|---|---|---|---|
| **Standard Default Parameters** | 3x3 | 1 | 3x3 | 1 |
| **Adjusted Optimal Parameters** | 45x15 | 2 | 23x10 | 1 |

Table 4.2: Block Detection Test Results - Kernels and Iterations

| | **W/H Threshold** | **Graph/ Image Min Area** | **Graph/ Image Max Area** |
|---|---|---|---|
| **Standard Default Parameters** | 2 | 10,000 Pixels | 1,000,000 Pixels |
| **Adjusted Optimal Parameters** | 2.2 | 100,000 Pixels | 7,000,000 Pixels |

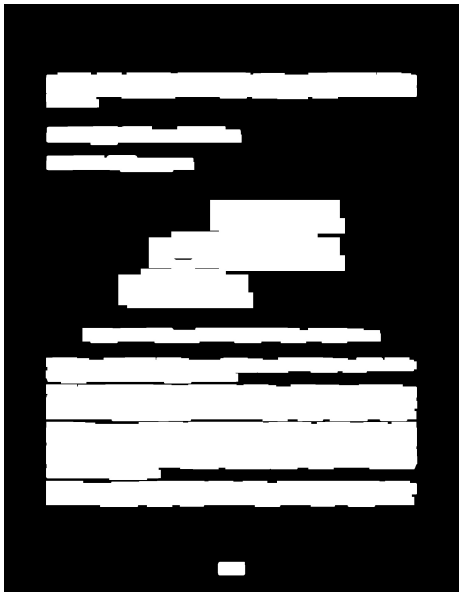
Table 4.3: Block Detection Test Results - W/H and Area Threshold and Limts

| | Graph Accuracy | Image Accuracy | General Block Accuracy |
|---|---|---|---|
| **Standard Default Parameters** | 13% | 24% | 34% |
| **Adjusted Optimal Parameters** | 67% | 73% | 79% |

Table 4.4: Block Detection Accuracy Test Results

The results of iterating through just shy of 100 test values has resulted in the a significant increase in the accuracy of the block extraction. Several observations can be made here, the first is that as expected and explained earlier, graphs are much harder to classify and extract and this is seen in the accuracy being less than that of images. The second is that although the accuracy is still far from 100% it is well into an acceptable region that can be used within this system. This simple method has proved to be quite effective and general block accuracy which includes graphs, images, tables, text and equations is very high.

Example outputs from the block detection can be seen in Figure 5. It illustrates the output from the dilation steps and the final block detection contours.



(a) Image Dilation

and changes done and implemented in the working prototype our project should be ready to transition from the prototype to the final Stage-Gate phase [3] - Full production & Market Launch.

## 3 Project Communication

### 3.1 Team Organogram

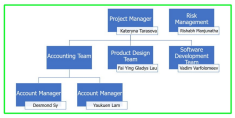



Figure 1: Team Organogram for the Angelo Pill Dispenser Project team

The team consists of four sub-teams: accounting team, product design team, software development team and risk management team.

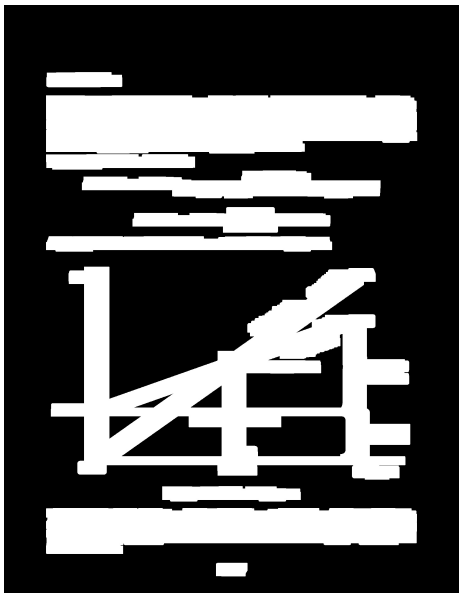
The project is divided up into parts for different sub-teams, and it is important that there is a central point for each sub-team to return to; conflicts can be identified while work from sub-teams is being put together, which also forms the responsibility of the project manager.

Project manager supervises that work for each sub-team is being carried out on time and to budget. Each sub-team reports to the project manager, apart from the independent risk management team. In order to identify problems at an early stage and ensure that mutual understanding of goals is achieved, report frequency is high and reports are usually very detailed for this project.

Risk management team is independent and act as the second line of defense. It detects the risks that the project is exposed to, evaluates and analyses such risks, and mitigates them.

6

(b) Block Detection



(c) Graph Dilation

### 5.7 Sales

In this section, a high-level break-even analysis will be conducted. From the above analysis, it was concluded that the final selling price per pill dispenser is at £200. The material costs per pill dispenser will be at £20. The majority of fixed costs come from wages, as shown in table 5 the total monthly salary is £160000. Combined with the cost of renting a manufacturing plant, the fixed cost is brought up to £200000.

By using the break-even formula:

$$Break-even\ Point = \frac{Fixed\ Costs}{Selling\ Price/product - Material\ Cost/product}$$

$$Break-even\ Point = \frac{200000}{200-20} = 1112\ units$$

A graphical illustration of the break-even metric is shown in figure 5.

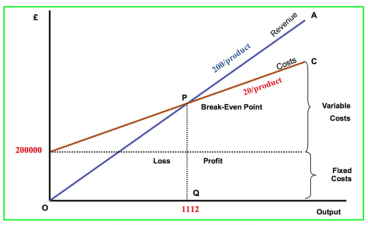



Figure 5: Break-even Diagram

The calculation shows that we need 1112 pill dispensers to break-even. Assuming that in the first year of business we sell an average of 1000 pill dispensers per month, the time to break-even is just slightly above a month. This means that we will start generating a profit after a month.

17

(d) Block Detection

Figure 4.5: Block Extraction Dilation and Contour Examples

## 4.3 Graph Data Extraction

With the graphs now extracted from the document, the aim is to try obtain the data that lies within it. This task is a complex problem that has seen a lot of development in recent years. The key issues include, no reference points for the axis' or the scale, the large variety of graphs such as scatter, line and bar. There are also obstacles such as text on or surrounding the graph such as keys and data point titles. It's clear to see that the shear volume of variations in graph styles poses the greatest risk. Despite this, there are some tools and services that been developed that aim to tackle this problem.

All of these tools function in a similar manor. The image of graph is first pre-processed to remove as much text and background noise as possible. This will enable them to extract the data with better accuracy. Note that this also involves some of the tools, removing grid lines from the graph as well. They are then converted into black and white or greyscale images to better identify axis lines, curves and data points.

Put simply, the algorithms work by using the axis lines as a pixel reference point from which, foreground pixels are then measured against. For example, taking a horizontal curve would return the distance in pixels from the point on the line to the vertical pixel on the X axis. It is then measured against the Y axis. This is done for every foreground pixel that falls within the area of the graph inside the axis.

It is very important to understand that the relationship between the values of each data point and the axis can only be calculated by manually defining the reference between pixel distances and the values on the graph. For each of most commonly used tools, it is required that the 0,0 point and 2 other points, 1 on the X and Y axis are defined by the user.

For more complicated graphs with multiple curves and bars, pattern finding techniques are used to try and identify these graph types and varying methods of data extraction are used from that point on-wards that can group pixel values or separate values based on colour.

### 4.3.1 Exisiting Solutions

There are 5 leading solutions that are highly regarded and used by many professionals in academics. The primary use cases for this type of technology has been in research and development, hence the type of focused user groups.

#### Enguage Digitizer

Engauge Digitizer is a popular well developed, Linux, Windows and Mac based application that supports the conversion of variety of image formats to spreadsheets and data files. It has been under constant development since 2015 and is focused on extracting information from line graphs. Recent development has further focused on it's interface, accuracy and language support for text within the graph. Testing this application proved that it was highly capable but as mentioned before, required the user to define the axis points and although the source code has been made available on GitHub, there is no easy for this to be implemented in a headless manor into this system.

#### im2graph

im2graph is another very popular tool that is known for being particularly good at multi curve extraction from a single graph. It offers both a free and paid version of the application and is available on Linux and Windows. Once again through testing, this application proved to be very capable but having a copyright license and no repository to work with makes this application entirely unusable.

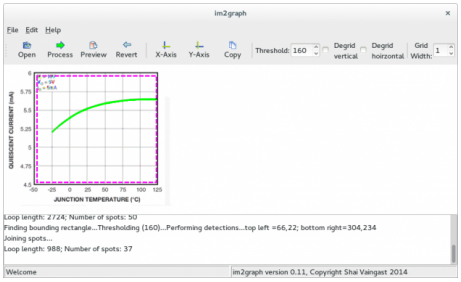
### GraphReader

GraphReader is a popular simple web-based application that offers the basics in data extraction. It supports png, jpg and gif image files and can digitise single curves with a graph. It also requires manual insertion of data points for axis but it also requires the user to draw or identify points on the curve using the cursor on the image to help improve with extraction. The interesting aspect of this application is that it can export the data as a JSON file, which is useful for other web based applications. While there are no source files available, the project is running with python tools in background. Using a web scrapper could yield good results but there is no indication even after emailing the creator, if this is allowed. The risk of IP blocking, makes this application less than ideal.

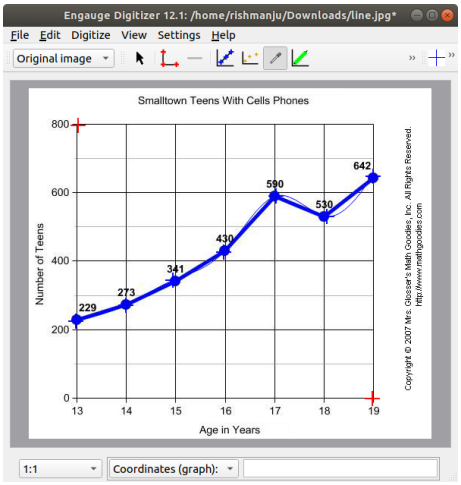
### WebPlotDigitizer

WebPlotDigitizer is the the leading tool used by professionals in a host of different fields and has even been presented at PLOTCON 2017. This is an incredibly powerful and diverse tool that can handle a multitude of graphs types, everything from bar charts and line graphs to ternary graphs and maps. The tool is available as a web application but also as standalone applications in Linux, Windows and Mac. All versions are entirely free and includes very well documented tutorials and full access to the GitHub repository. Testing with this application yielded the best and most accurate results from all of the above tools.

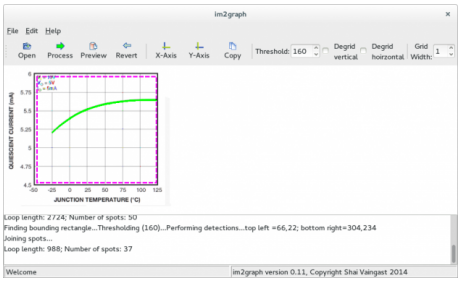
The major issue with using this version of the application is that it is not directly or easily compatible with python or any tools that could be used to convert the program. This is because it was designed in the first place to be a web based application. Thankfully, the developer has recently released a python packaged call PlotDigitizer that in it's initial form offers the basics in line graph data extraction. What's incredibly good about this is that for private use, the entire repository can be altered and modified to work with this system. This compatibility advantage is why this was chosen to be integrated into the system.



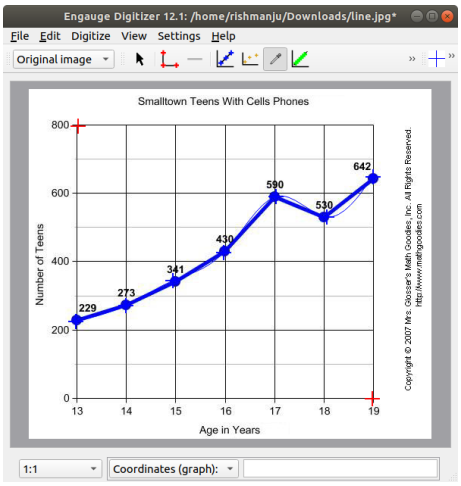

(a) im2graph Screenshot




(b) Enguage Digitizer Screenshot




(c) im2graph Screenshot




(d) Enguage Digitizer Screenshot

Figure 4.6: Existing Graph Extraction Solutions Examples

### 4.3.2 PlotDigitizer Adaption

The PlotDigitizer or PD package has 4 minimum requirements, the image file of the graph and 3 points that indicate the location of the axis' and the origin. Out of the box, the package takes arguments from the terminal and parses them into variables that are then used for the various functions. The package also launches an OpenCV viewer and click button event reader that asks the user to locate the 3 points they passed on the graph. Therefore there are several adjustments that need to be made in order to completely automate this program and integrate it into the system:

- An additional function needs to be written that can take input values from other scripts and either bypass the argument parser or directly feed the input string
- Pre-determined or estimated values for the 3 points are necessary
- A method for determining or estimating the location of those points needed to extract the graph needs to be implemented

The first issue can be tackled by generating a system argument array and then appending the necessary parameters and passing this directly to the main function in the script. The main function also needs to be adjusted to accept the argument as an input variable. This will allows the script to be called without the need for a terminal instance.

There are various methods that have been experimented with including using OCR and OpenCV to estimate the location of the cross point for the axis by looking at axis characters such as '0,0' or just '0'. It has also been used to try and estimate the location of the cross point of 2 black lines at 90which would indicate the corner point. This was also used to estimate the scale of the axis'. The issue with these methods is the variability and inconsistency in the style of the graphs. Even for defined style sets, the accuracy rate never rose above 17%.

Dividing this problem into 2 parts, simpler crude solutions where used based on the assumptions from the following tests. Using the output from the block extraction script, a test with 50 graphs using standard Excel and Matlab style layouts, with varying image sizes was used. It was found that on average, the kernel parameters used to extract the graphs would produce consistent contour detection that resulted in the X axis', on average being within 8% of the total height of the image from the bottom, this was determined based on pixels. The same can be said for the Y axis but with a percentage of 10%. Using these as a base assumption, the coordinate points on the graph can be estimated as a percentage of the total width and height with a buffer to ensure that points are within the graph area. The optimal percentage was found to be 10% and 12%. While this is a rather crude method, the results from further testing shown later are admirable. It is noted that this buffer will cut off a small portion of the graph but this is necessary to increase the accuracy and robustness of the system. This therefore allows us to estimate the lines of the axis' to place the 3 coordinates.

The last issue is determining what coordinate points to pass. The origin was set at '0,0' by default for graphs. As the main goal of this component is to extract the shape of the curve in the graph, the values for the axis are not particularly important as the curves are extracted on a pixel basis. Because of the nature of the project and the way PlotDigitizer works, the values were chosen based on reliability testing using the same test set as mentioned above. This resulted in setting the coordinate values to (10,0) and (0,10) with the '10' value on each axis relating to pixel value on the X axis that is 56% of the total width and 55% on the Y axis.

Placing these adjustments in a condition statement bypasses the inbuilt clickbutton procedure and the entire package now operates as intended.

```
def ask_user_to_locate_points(points, img):
    global coords_
    own = True

    # Adjustments for automatic coordinate positions
    if own ==True:
        r, c = img_.shape
        x = c * 0.1
        y = r * 0.12
        coords_.append((x, y))
        x = c * 0.56
        y = r * 0.1
        coords_.append((x, y))
        x = c * 0.1
        y = r * 0.55
        coords_.append((x, y))
    else:
        cv2.namedWindow( windowName_ )
        cv2.setMouseCallback( windowName_, click_points )

    while len(coords_) < len(points):
        i = len(coords_)
        p = points[i]
        pLeft = len(points) - len(coords_)
        show_frame( img, 'Please click on %s (%d left)' % (p, pLeft) )
        if len(coords_) == len(points):
            break
        key = cv2.waitKey(1) & 0xFF
        if key == 'q':
            break
    logging.info( "You clicked %s" % coords_ )

    return
```

Listing 4.1: Scipy Predefined Functions

### 4.3.3 Testing

The complete script was tested using the same data set as mentioned above. Of the 50 graphs that were tested, 48 graphs had their curve successfully extracted. an example of the out from script can be seen in Figure 4.7. One very important observation is made. While it is clear to see, that PlotDigitizer did manage to roughly extract the shape of the plot, there is also a lot of variability in the data. Particularly at the extremities of the data set and this is because of the residual interference from both the grid lines and illuminance and shadows that are still lingering in the image. The most effective way to reduce these errors and better estimate the shape of the curve is to smooth the data.


(a) Input Image

(b) Ouput Data

Figure 4.7: PlotDigitzer Adaption Input and Output

### 4.3.4 Smoothing

Smoothing the data will help in reducing the noise and eliminating outliers. Generating a more uniform data set will yield better results not only for determining their model functions but also for the audio conversion in the next section. There are various methods that are used for smoothing. The aim is to capture the most important data points and omit those that are not, essentially the goal is to increase the signal to noise ratio; the most common relative smoothing filters are explored below.

#### Moving Average Filter

The moving average filter is defined as a function that moves through the data set by calculating the sum of the previous inputs and dividing this by window size that is less than the total number of data points. The filter can be written as such:

$$MAF = \frac{1}{N} \sum_{i=0}^{N-1} x[n-1] \tag{4.1}$$

The main advantage of this simple filter is that it can be to any set of data, linear or non linear. The disadvantage is that this method works best for relatively large data sets. The beginning and end of the data can often be skewed and the resulting output is not always best for fitting function models too. While this method might be good for almost perfect data sets, the data extracted from the graph can be quite noisy and can in some cases have strong outliers. This type of filter does not respond well to these cases and as it is unlikely that many of the data points will be accurate this is not the preferred method.

#### Moving Triangular Filter

Moving triangular filters are essentially 2 moving average filters that are overlayed on top of eachother. This allows for greater weighting to be placed for close neighbouring points within the window rather than a uniform approach. The weighting can be thought of as representing a triangle, which is where it gets its name. The moving average filter is rectangular in shape because it applied an even weighting to all values within the window. The filter can be written as such:

$$MTF = \frac{1}{N} \sum_{i=0}^{N-1} MAF[n-1] \tag{4.2}$$

The main advantages of this filter lie in the reduced effects of aliasing compared to the averaging filter and it's very fast computational speed compared to other filters. Despite the advantages of weighting the data points, it still holds the same inherent flaw as the averaging filter in not being well suited to handle extreme outliers with data nad hence it is also not the preferred method.

**Gaussian Filter**

Gaussian filters operate in same way as the previous methods, the only difference being the shape of the filter representing a Gaussian impulse response. This method is particularly good at removing Gaussian noise and is a common filter used in a variety of applications, in this case a 1D filter is used and it can be represented by it's impulse response:

$$MGF = \sqrt{\frac{a}{\pi}} \cdot e^{-a \cdot x^2} \tag{4.3}$$

These filters essentially have the same advantages of the the triangular filter with the added benefit of being very effective at removing Gaussian noise. While there may be some guassian noise that has filtered into the data from the image processing steps prior, it isn't particularly relevant or useful in our application it will perform much the same as previous filter. For this reasons it also not the preferred method.

**Savitzky-Golay Filter**

Savgol Filters are known for being very good at preserving certain features within a data set, these include features such as relative maxima and minima and high frequency components. It works by using a polynomial regression technique, the simplest form can be thought of as having a bowl shaped window. This can be altered by defining the coefficients and the degree of the filter. The filter can be represented as such:

$$SVF = \frac{1}{h}[\sum_{i=\frac{np-1}{2}}^{\frac{np-1}{2}} a_i x_{n+1}] \tag{4.4}$$

The advantage of this type of filter is the preservation of important data points and the adjustability in the filter order. While it is slightly slower at computing, it more than makes up for it's ability to almost entirely eliminate strong outliers. As filter testing will show, this method is preferred purely due to it's impressive performance.

### 4.3.5 Filter Testing

All the filters are tested against a test base of 25 graphs of each of the following function models:

- 1st to 3rd order Polynomials - (Linear, Quadratic and Cubic)
- Exponentials
- Logarithms
- Trignometric functions - (Sine, Cosine and Tangent)

All filters were set to use the same window length, and the same order of degree. An example of the output of each filter against the same linear output from the graph data extraction section is shown in Figure 4.8. It is clear to see that the Savgol filter smoothed the data the best and in all cases for every data set, this was the same outcome. It is better at rejecting the the extreme outliers and does a much better job at smoothing the data into a more defined data set. This will be more effective later on when determining which model function best fits the data.

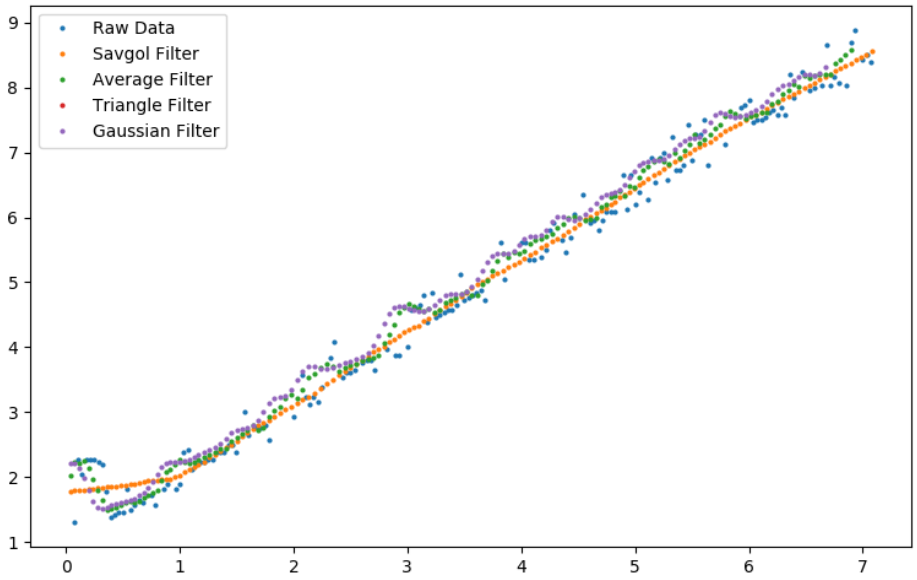



Figure 4.8: Linear Filter Testing Result

## 4.4 Graph Audio

In the research stage of this project the prospects of active auditory interfaces was explored and it was found to be very effective at communicating graph based information. This works by converting data points into audio signals. By assigning each axis to an audio feature such as frequency and amplitude, each data point can be mapped to a audio signal. In order to do this, an audio synthesizer must be utilised, this will convert our audio parameters into a sound wave that can saved and played back to the user. There are 2 very popular python packages called Wavebender and Chippy, both are audio synthesizers but have varying functionality. Neither are audio players, they simply generated the sound waves and can store them to be played by another application.

### 4.4.1 Wavebender

Wavebender offers a lot of flexibility and functionality, It can create audio samples and a select range of editing features such as dampening, slicing and individual channel synthesis. It's relatively light weight but can be rather cumbersome because of the range of features it offers. For every sample that is generated, there are many parameters to set and for certain audio samples looping through frequency ranges can be inefficient.

However, for simple audio samples such as square waves, it can be easily implemented. We define the channel parameters, these are frequency and amplitude, in this example, they have been set to 500 and 0.5 respectively. The samples are then produced by setting the sampling rate, duration and the number of channels. This is then temporarily written to be played using an audio player of choice.

```
channels = ((square_wave(500, amplitude=0.5),),)

samples = compute_samples(channels, 44100 * 2 * 1)
write_wavefile(stdout, samples, 44100 * 2 * 1, nchannels=1)
```

Listing 4.2: Wavebender Square Wave Example

### 4.4.2 Chippy

Chippy is package that was developed under the inspiration of wavebender. It's a very lightweight and simple package that offers the basics but has greater compatibility and stability. It supports sine, sawtooth, triangular and square. It is a pure python package that has generators for each of the supported waves and has excellent support for direct or written audio use.

Though it may not have as wide a feature set in terms of dampening or channel support, for this application it can still serve the same purpose without compromising the execution. It's advantages lie in the inherent simplicity and integration support with other packages that will be necessary. It can be implemented as simply as calling the wave function and defining the same parameters as in Wavebender except for the number of channels.

```
synth = chippy.Synthesizer(framerate=44100)

sine_wave = synth.sine_pcm(length=2, frequency=5000, amplitude=0.5)
synth.save_wave(sine_wave, "sin.wav")
```

Listing 4.3: Chippy Sine Wave Example

When integrating the two into the final system it was found that while both performed their functions just as expected, there were compatibility issues. These originate from creating temporary files that the audio is stored within. Wavebender had many issues writing the audio to any form of temporary files other than outputting the audio as a real-time stream. While this is useful for direct synthesis it's not particularly useful when generating an entire audio sequence for all data points. For this reason alone, Chippy was used over Wavebender with the Pygame library that supports Bytes format temporary files and written saved files.

### 4.4.3 Data to Audio Conversion

Representing the data through audio, requires forming a relationship between the data points and the frequency and amplitude. The range of audio values that can be converted to, must be predetermined. The amplitude will simply range from 0 to 1. The frequency can be predefined, from rough testing it was observed that there are several factors that affect this value.

The first is the frequency response range of the speakers that the audio is being played through. The speakers present in the laptop that is being used cannot output very low frequencies below 100Hz and there are certain frequency values and short ranges that are inaudible. These issues changed between external speakers and headphones. The second issue was related to the frequency range defined for the number of data points. The larger the number of data points, the larger the frequency range needs to be. This is to ensure that the differences within the data points are audible, the opposite is true for a smaller range of data points. The range needs to be smaller to prevent harsh changes in the pitch that can be uncomfortable to listen to.

It's important to note that the results from the data extraction recreate the graph relative a pixel based axis. Therefore, the optimal relationship for range of graph trends was found to be setting 20Hz per pixel difference. As the average maximum number of data points us roughly 300, this puts the upper limit at around 6000Hz above the lower limit. The lower limit was set based on the speakers that were being used.

While this relationship, produced clear variations and minimised the harshness of the differing data points. It is noted that audio is highly personal and for each individual, preferences will vary and an option kept open for the user to manually determine the frequency range that they wish to use. The exact relationship is determined by calculating the end to end range of the values in the data. This is as simple as deducting the maximum value from the minimum value and works for both negative and positive values.

The assignment of frequency and amplitude to each axis was based on the average shape of the graph image. This refers to the pixel length of the images and if they represent a square or rectangle. In the case of a square shaped graph image, it does not matter what each is set too. As the majority of images are square or a horizontal rectangle with the longest edge falling parallel to the bottom and top of the page, the X-axis was set to represent the frequency. This provides the greatest possible range for the longest stretch of the graph. Once again this can be flipped through a user based option, but by default it is set as such.

The conversion is then as simple as converting each data point into a frequency and amplitude parameter and synthesising each data point defining the length of each sound sample, this is set at 0.1 seconds by default but can also be changed by the user. There are 2 functions that can be called, one that runs through every single data point and joins the samples to create a master audio file. This can be played back to represent the entire graph. The second function can be called to convert individual data points. This gives the user flexibility around how they wish to study the graph. They can scrub through the points manually and study with greater control and detail, or they can listen to complete conversion and get a general idea.

### 4.4.4 Testing

This component was tested with 100 defined data sets that represented some form of model function. For example, polynomials, exponentials and trigonometric functions. For each test, the entire data set was converted, these were then played back alongside a plot of the original data to ensure that they had been converted properly to represent the model exactly. In all 100 tests, the conversions were entirely successful.

## 4.5 Graph Description

In addition to the auditory interface and feedback of the graph data, there is a need for a textual description to support that audio representation. The descriptions of the graph are proposed to return what model of function the curve in the graph best fits too. These functions are defined as follows:

- 1st to 3rd order Polynomials - (Linear, Quadratic and Cubic)
- Exponentials
- Logarithms
- Trignometric functions - (Sine, Cosine and Tangent)

This is a fantastic addition the system as it will act as a secondary check for the user who is studying the paper to ensure their assumptions from analysing the audio are indeed correct. this will also enable them to extract greater detail from the graph if they know better what the shape is. In order to achieve this, each of the above mentioned models are first applied to the extracted graph data. Their corresponding coefficient of determination or 'R squared' value is calculated to determine which model/curve best fits the data. This is then matched to a predefined description that is fed back to the user.

### 4.5.1 Coefficient of Determination

The coefficient of determination is defined in statistics as a measure of error relative to the variance in a relationship between variables and their data points. In Figure 4.5, a linear curve has been fitted against a noisy data set. The distance between each point and the curve, squared, is in simple terms the error that we wish to minimise. In many methods, the curve that is being fit to the data uses a least square regression model that minimises the sum of the squares residuals. This is essence part of how this measure is also calculated. Using the following definitions, it is possible to calculate the coefficient of determination between data points and a function model:

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.5}$$

$$SSE_{total} = \sum (y_i - \overline{y})^2 \tag{4.6}$$

$$SSE_{residual} = \sum (y_i - \overline{f})^2 \tag{4.7}$$

$$R^2 = 1 - \frac{SSE_{residual}}{SSE_{total}} \tag{4.8}$$

Equation 4.1 defines the mean of the set of data, 4.2 defines the sum of the total square. This in measures the inherent variability within the data itself. 4.3 defines the sum of the total square error. This measures the variability between the data and the function that that is being fitted. 4.4 gives the adjusted ratio between the two is the coefficient of determination. It states that the close this coefficient value is to 1 the more closely the function matches the data. This is what will be used to determine which function model best fits the data that is extracted from the graph.

### 4.5.2 Numpy - polyfit

Numpy has a polynomial fitting function that can take a set of data and the degree of the polynomial being fitted and return the coefficients of the function that best fits that data. This method works on the least squares method that as mentioned previously acts to minimise the error. When working with polynomials, this method works very well.

The major drawback from this function is that it uses linear regression to fit the functions to the data. This causes problems when it is used to fit models such as logarithms and exponentials

as it will treat them as if they were linear. This will cause the smaller values in the data to be emphasised that can then cause major deviations for higher values. The standard work around for this is to add weights to each of the data points that will compensate. This is often known as weighted least square method. While this does perform an admirable job, it still lends to small errors making it not as efficient as other methods. An example of how a 3rd order polynomial and a logarithm would be called with polyfit can be see in listing 4.

```
# Cubic / 3rd Order Polynomial
cubic = poly.polyfit(x, y, 3, full=True)

# Exponential (with weighting)
loga = poly.polyfit(x, numpy.log(y), 1, w=numpy.sqrt(y))
```

Listing 4.4: Scipy Predefined Functions

### 4.5.3 Scipy - curve_fit

Scipy has a curve fitting function that can take a predefined equation and return the parameters for those equations back with the co-variance matrix. The advantage of this method over the polyfit is that this can be used to fit almost any function without the need for any transformations or weightings. The function acts as a wrapper around the Scipy least square optimisation function. This operation is a common method used with regression to minimise the sum of squares in the residual error when fitting systems to a series of equations.

In order to define the fitting equations we pass them into Scipy as functions that return the result of functions equation given the input data that represents the X values:

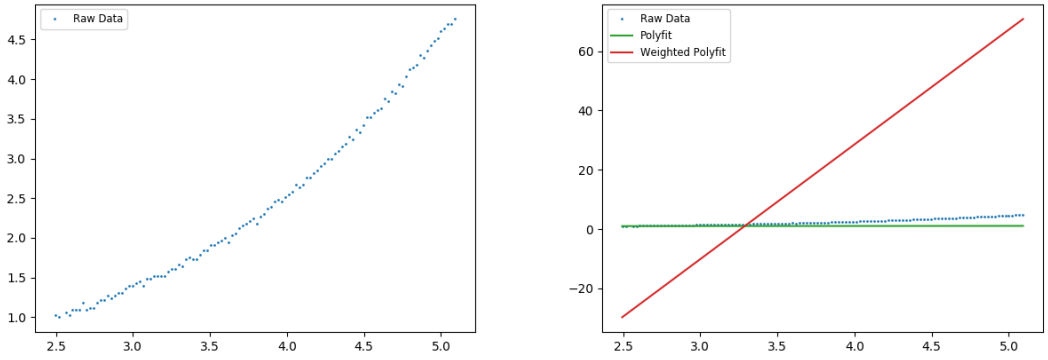
```
def ord1(x, a, b):
    return (a * x) + b

def ord2(x, a, b, c):
    return (a * (x**2)) + (b * x) + c

def ord3(x, a, b, c, d):
    return (a * (x**3)) + (b * (x**2)) + (c * x) + d

def expo(x, a, b, c):
    return a * np.exp(-b * x) + c

def loga(x, a, b):
  return a * np.log(x) + b

def sinfn(x, a, b):
    return a * np.sin(b * x)

def cosfn(x, a, b):
    return a * np.cos(b * x)

def tanfn(x, a, b):
    return a * np.tan(b * x)
```

Listing 4.5: Scipy Predefined Functions

### 4.5.4 Testing

Determining which method will be best is as simple as plotting one graph for every function that compares the standard polyfit, polyfit with weighting and the curvefit method. Whichever better fits the data will be the the most accurate. Noting that curvefit already as a theoretical advantage, this is a practical test to ensure that our assumptions are indeed correct. The $R\hat{2}$ value is used here to quantitatively measure which methods are more accurate.

While testing these methods it became very clear as soon as the exponential function was fitted that the polyfit method would not work. In every case and trial, it failed to plot a curve that matched in any the data was used. It is in important to note that at the time of writing this report Numpy is now strongly recommending that users integrate the polyfit function through the polynomial library. Despite using this and considering various transformations for both exponentials and logarithms, it causes a whole host of errors and results that cannot be considered by any means. An example of what is returned when it is called to fit an exponential curve to an exponential set of data points is shown in Figure 4.7. It is important to note that this is after forcing the function to limit the number of self calls to return it's best attempt before failing.



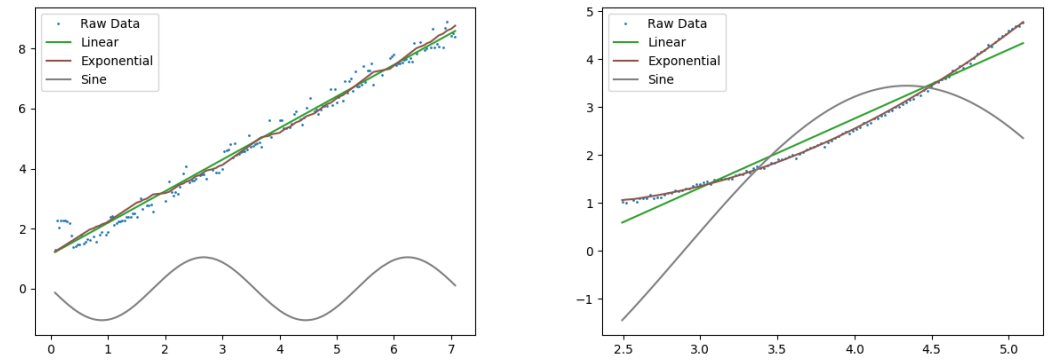

(a) Exponential Data

(b) Polyfit and Weighted Polyfit

Figure 4.9: Numply Polyfit Example - Limited iterations

It is clear to see that the results from this are unacceptable. Although this may work very well for polynomial functions, anything beyond this causes too many errors and for this reason, it will not be used for this system.

The curvefit function faired much better in every single test case. Even with very noisy data, the function was able to try each and every model and returned valid determination values and results. To better visualise the data, a single curve is plotted from each model category for both a linear data set and the above exponential data set.




(a) Linear Data

(b) Exponential Data

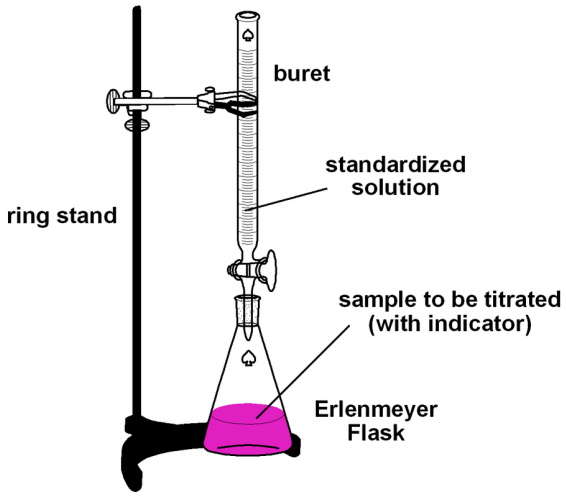
Figure 4.10: Scipy curve_fit - Results Example

From a total of 160 data sets, equally distributed to represent each model function, 85% of those sets were correctly identified. The remaining 15% were all related to polynomials. In some cases, the script would return the wrong order of polynomial that it best fit to the data. This is inevitable as variations in the data even after smoothing can led to higher order polynomials fitting better to the data. In all cases the determination value was above 0.95. Knowing that this would only occur if higher order polynomials fit better, we can apply a simple condition that chooses the lowest order above 0.95 as the best fit. With this condition in place, all data sets were correctly identified. This is a fantasic result and concludes that the Scipy curvefit method is the choice to move ahead with and will provide the user with the best possible chance of understanding in greater detail the graph that is extracted.

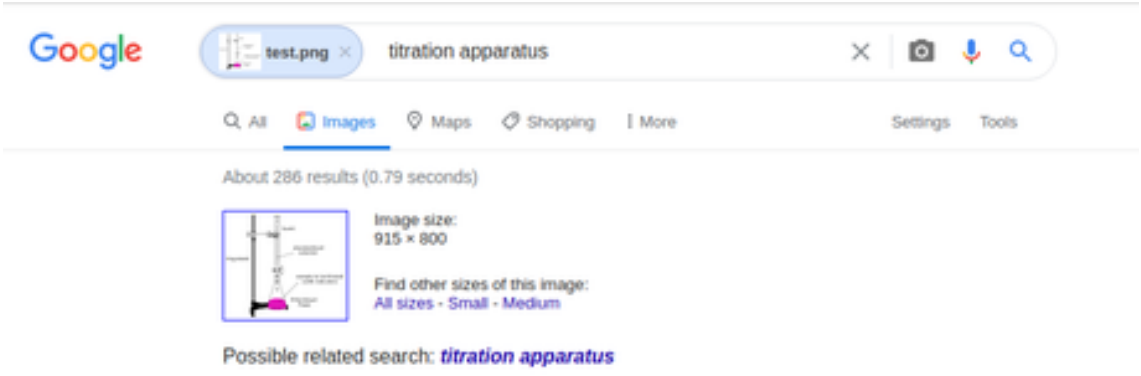
## 4.6 Image Description

Textual descriptions for images is a complex and ongoing problem that is being solved using various methods. Many attempts to create descriptions involve using machine learning to train a defined set of images with predefined labels, contents and entities. This method works well for systems that look for define specific topics or categories, such as images of animals or fruit and vegetables. As we require textual descriptions for images within scientific literature, the boundaries are not easy to define and training a data-set would be infeasible and would only work for specific narrow topics. The images could range from atomic structures in Chemistry to apparatus setups in physics.

The approach chosen to tackle this problem based on finding the largest database of images that could be used to draw comparisons against. This would be Google Images; with a database in excess of 10 billion images. It is the largest single resource that can be utilised. The key feature that can be utilised is Google Reverse Search, this enables users to submit images and in return will find other images and sources that closely match. An example of the results returned from a search of a titration apparatus image can be seen in Figure 4.4.




(a) Image




(b) Results

Figure 4.11: Google Images Search Example

The search result correctly identified that image as being related to titration apparatus. While it is recognised this method will not be able to provide specific information or detailed descriptions, it does give some general information about the image. In order to implement this into the system, a web scrapper must be used to send a request to Google Images and extract from the resultant URL the information that describes the image. This can be achieved by accessing the HTML parser of the website and search for the class ID that contains the search text.

An important note to make here is that the open source, chromium software by Google is being used here to create a headless search request. This allows for browser like search to be performed without launching or using the typical browser interface or application. The Python Selenium package is being used here to send the initial request and return the desired URL. The BeautifulSoup4 package is being used to scrap the information desired from the website. Put together this can be implemented as seen in Listing 4.1.

```
def get(file_path):

    searchUrl = 'http://www.google.hr/searchbyimage/upload'
    multipart = {'encoded_image': (filePath, open(filePath, 'rb')), 'image_content': ''}
    response = requests.post(searchUrl, files=multipart, allow_redirects=False)
    fetchUrl = response.headers['Location']

    # Open this new URL in a headless browser so that we can access the results
    options = Options()
    options.add_argument('--headless')
    options.add_argument('--disable-gpu')
    driver = webdriver.Chrome("/usr/bin/chromedriver", chrome_options=options)
    driver.get(fetchUrl)

    # Extract the information we need
    content = driver.page_source
    soup = BeautifulSoup(content, features="html.parser")
    for a in soup.findAll('div', attrs={'class':'r5a77d'}):
        extract=a.find('a', href=True, attrs={'class':'fKDtNb'})
        results.append(extract.text)

    return results
```

Listing 4.6: Google Images Web Scrapper

It must be noted that this method is possible because Google Images allows for web scrapping. Hence the traditional fears surrounding this method such as IP blocking, or HoneyPot traps are not relevant here. What is possible however is that the HTML structure could change overtime. If this is the case, then adjustments must be made to class iterator to find the correct location within the HTML code to retrieve the information.

### 4.6.1 Testing

Testing this component of the system is not as simple as running it through a set of images and matching it to a defined description or the search title that it was obtained from. It is highly unlikely that it will return the exact same result. Instead it will return a generalised description of what the image contains. This therefore means that for every image that is tested, it must be manually compared against the original image and description. The results were based on how closely descriptions matched their image on scale of 1 to 5.

Of the 50 images that were tested, there were no images that returned a null or invalid description. On average the results scored a 4 out of 5 in terms of the accuracy of the descriptions. Being the only available approach that is feasible, the results are promising and should provide at least some addition information to the user that they would otherwise not be able to receive. As such, for this prototype system, this is accepted and integrated into the system.

## 4.7 Text Extraction

Text extraction is carried out by using optical character recognition, this captures all the recognisable text on the page and returns a string. As the majority of most pages contain a large portion of textual information, it is essential that from the methods being tested here are robust and accurate. The inherent capabilities of OCR has lead to a number organisations and firms developing their own packages and services that are available to use such as:

- OpenCV and Tesseract
- Google Cloud Vision
- Amazon Rekognition
- Microsoft Azure ComputerVision OCR
- Cloudmersive

It is important to note that almost all of these cloud based APIs function identically and have similar performance. The only major differentiating factors are the way in which they breakdown the text they extract. Google cloud vision will be used to test cloud based solutions, for it's superior output structure and for it's low cost. The first 1000 requests per month are free and there after it is only $1.50 per 1000 requests. It is also important to note that this API comes with a whole host of other additional functions such as handwritten text detection and web detection that can search through google images for entities it recognises within text.

This will be evaluated against Tesseract, an offline open source OCR package that was collaborated with Google. The major advantages of this are that it is completely free and easy to implement. Being offline, it also has the ability to operate without an internet connection. It is important to note that the Tesseract package will work on any image that is presented to it, OpenCV is used in conjunction with that to identify and make visible the blocks, words and characters it recognises in the form of bounding boxes. The Cloud Vision API uses a PIL package to identify the items it recognises but both perform the same function.

### 4.7.1 Tesseract

Implementing Tesseract with python is as simple as installing the package, importing the library, setting the configuration parameters and calling the function. The configuration parameters are three fold; language, layout analysis and engine mode.

Calling the correct language is as simple as defining it from the list of trained language datasets. By default, Tesseract will attempt to recognise English but there are 125 to choose from making it widely adaptable. The layout analysis allows you to define relative to the page structure how the text should be extracted, there are 13 such options:

```
0 = Orientation and script detection (OSD) only.
1 = Automatic page segmentation with OSD.
2 = Automatic page segmentation, but no OSD, or OCR. (not implemented)
3 = Fully automatic page segmentation, but no OSD. (Default)
4 = Assume a single column of text of variable sizes.
5 = Assume a single uniform block of vertically aligned text.
6 = Assume a single uniform block of text.
7 = Treat the image as a single text line.
8 = Treat the image as a single word.
9 = Treat the image as a single word in a circle.
10 = Treat the image as a single character.
11 = Sparse text. Find as much text as possible in no particular order.
12 = Sparse text with OSD.
13 = Raw line. Treat the image as a single text line,
     bypassing hacks that are Tesseract-specific.
```

Listing 4.7: Tesseract Layout Options [**?**]

By default the system is set to automatically segment the page and perform OCR without identifying the orientation, this is option 3. Should the orientation be necessary, the image can be processed for a second time with option 0. The final parameter is the OCR engine mode, this determines which algorithm Tesseract will use to identify the text. The latest version, Tesseract 4, supports a new LSTM neural network algorithm that focuses on better recognition of lines of text. This is configuration option 1; option 0 is the legacy algorithm that was supported by Tesseract 3 [**?**]. The major differences in performance are in the from of accuracy and speed. Version 3 was very accurate but slow, whereas, version 4 was must faster but slightly less accurate. Both versions will be tested but it is recognised that it is widely accepted the LSTM algorithm is the new standard.

The output will return an array with each element representing the text extracted from each block identified within the page, such as paragraphs or titles. It is also important to note that implementing this into a python script requires an additional package such as PIL or OpenCV to open and read the image to be processed first so that it is in the correct format.

```
def tess_detect(img):

    # -l eng       English language
    # --oem 1      LSTM OCR Engine
    # --psm 3      Layout analysis
    config = ('-l eng --oem 1 --psm 3')

    # Calling the Tesseract OCR function
    text = pytesseract.image_to_string(img, config=config)

    return text
```

Listing 4.8: Tesseract OCR Implementation (English/LSTM/Auto-Segmentation)

### 4.7.2 Google Cloud Vision

Implementing Google Cloud Vision is even simpler than Tesseract, the API performs auto segmentation and by default automatically recognises the language of the text. Where this method has an advantage is the output structure, the service returns a nested array that breaks the API request into, pages, blocks, paragraphs, words and even symbols/characters. The disadvantage is that the actual text that is extracted is by symbol and in order to reconstruct the sentences and paragraphs, you must iterate through the output and join these characters together. While this does add an extra step, it doesn't require much additional computation power and with the advantage of being cloud based. The losses in these steps are more than compensated for in the speed of the text extraction.

As such, it can be implemented as seen in Listing 4.9 to extract each section into an array:

```
def vision_detect(image, feature):

    # Calling the Vision OCR function
    response = client.document_text_detection(image=image)
    document = response.full_text_annotation

    bounds = []
    section = []

    # Extract text and bounding boxes from the response
    for page in document.pages:
        for block in page.blocks:
            for paragraph in block.paragraphs:
                words = []
                for word in paragraph.words:
                    print(word)
                    text = ''.join([symbol.text for symbol in word.symbols])
                    words.append(text)
                    for symbol in word.symbols:
                        if (feature == FeatureType.SYMBOL):
                    if (feature == FeatureType.WORD):
                para = " ".join(words)
                if (feature == FeatureType.PARA):
            section.append(para)
            if (feature == FeatureType.BLOCK):

    return bounds, section
```

Listing 4.9: Google Cloud Vision OCR Implementation

### 4.7.3 Testing

Both methods are evaluated based on accuracy, deconstruction and efficiency:

- The accuracy of the text that is extracted - This involves comparing the number of correctly identified words
- Deconstruction capabilities - This refers to the way each method classifies chunks of text within a page and how it breaks that down into words and or characters
- The efficiency - This involves comparing the time it takes for each method to identify the same piece of text

The test base comprised of 100 documents, 50 with an average of 450 words each and 50 with an average of 200 words. The first set is significantly higher than what would be found in a normal document. As both methods are highly regarded and have been improved over many years. The goal behind these more dense documents is separate the two in terms of efficiency, the second set with fewer words, is more representative of the number of words found in documents with images, graphs and texts. This set is designed to test the accuracy level that is more relevant to this application; the results can be seen in Table 1 and an example of the words detected by each method are shown in Figure 6.

| | **Accuracy (450 Words)** | **Average Time (450 Words)** | **Accuracy (200 Words)** | **Average Time (200 Words)** |
|---|---|---|---|---|
| **Tesseract** | 86% | 5.1 Seconds | 91% | 2.2 Seconds |
| **Google Cloud Vision** | 65% | 1.7 Seconds | 73% | 0.9 Seconds |

Table 4.5: Text Extraction Test Results


(a) Tesseract Text Detection


(b) Cloud Vision Text Detection

Figure 4.12: OCR Detected Words Contour Output

As can be seen from the results, the Cloud Vision API has a large efficiency advantage as it can take advantage of power cloud based processing. The time difference between 200 and 450 words was less than 0.3 seconds. Tesseract saw a much larger difference of 3.3 seconds. As expected the density of words within the page severely impacts the efficiency of Tesseract. At 2.2 seconds this is still quite far off from cloud based solutions but is more than acceptable for this application as the major deciding factor is the accuracy. It's clear to see that Tesseract is significantly more accurate than Vision API. This is not an unusual result, it is well known that many cloud based APIs cannot match the performance of offline packages. They are better suited for specialised use cases and for integration into bigger commercial applications such as mobile phone applications.

The main observation made from the results is that the difference in accuracy comes from the OCR algorithm. In example shown in Figure 6. Tesseract was able to identify all the words in the page except the chapter number '4' which was not recognised as an entity. Vision API did recognise the number and all the words in the page but it was not actually able to identify them. In this case, it completely missed out the first paragraph. It is this behaviour that lends to it's less performance and is the reason why Tesseract was chosen for this application. It's superior accuracy outweighs loss in efficiency.

### 4.7.4 Auto-Correct

Based on the results from the evaluation it is clear to see that the accuracy will never reach 100% every single time. While the most common faults will be missing words, the second most common fault is incorrect spelling and grammar. This is particularly important considering that this will be converted to speech. Any errors can result in unnatural, inconsistent speech that does not flow easily, making it harder to understand and interpret the information being presented. The solution to this is to pass the extracted text through an auto-correct function that iterates through any errors and rectifies them. There are many packages in python that can be used for this, PyLanguageTool, PySpellChecker, Autocorrect, TextBlob and such. However, many of these offer only spell checking and not grammar checking. PyLanguageTool is one of the only packages that can perform both of these and is the most powerful. Every request that is sent goes through the LanguageTool server [**?**]. This is particularly advantageous because this service is used by many companies and is regularly updated and maintained. This provides a layer of certainty for the support surrounding for this package

This package is incredibly powerful, it offers a lot of flexibility in how the text to be corrected should be processed. Options range from custom words and rules, tiered correction suggestions and a whole host of languages. This API works by submitting a request and a JSON structured result is returned with all the information it provides including the suggested corrections. The most important keys are the confidence level for the error that is detected and the suggested replacement words or grammar. The replacement suggestions are listed from the most to the least likely; applying a condition for the confidence level and replacing the error with the first item in the list yields the best implementation for this system. This is used to iterate through every sentence in the page rather than words or characters to increase efficiency.

Initial testing revealed several hurdles, the first was the confidence level. Even for simple, very obvious and clear spelling mistakes, the confidence level never rose above approximately 0.65. Regardless of the complexity of the error, spelling or grammar, the error level consistently placed between 0.5 and 0.65. The condition was therefor set fairly low at 0.48 in order to capture as many errors as possible. Furthermore, not all errors had suggestions, for each of these cases the words were ignored. The most pressing issue was randomised error detection after a correction. In a few cases, after an error was rectified and the text was pass through again to search for more errors, the API would return a non-existent result indefinitely unless it was passed. This issue was resolved by setting a buffer variable that stores the text from 3 iterations prior. If the API returned the same error 3 times it would be rejected and the buffer text would be passed through instead. 3 iterations was chosen to prevent cases such as " that, that", where sentences with double words are present. With these improvements, individual tests saw an accuracy rate of 93%.

Once finalised this was then evaluated with the output from the OCR tests, the extracted text is corrected and matched with the original text to see how closely the text improves to match the original. The results are summarised below:

| **Extracted Text Accuracy (Tesseract)** | **Corrected Text Accuracy** | **Average Processing Time (per page)** |
|---|---|---|
| 78% | 86% | 1.043 seconds |

There are several observations to be made from these results, the first is that it does indeed increase the accuracy of the data by, on average, 8% for an error rate of 22%. The processing time is also incredibly short for a lengthy page with 450 words. It is important to note that this is an aggregated result. The accuracy is relative to OCR processing, as seen previously, this inherently introduces errors that are independent of this auto-correct functionality. While in theory this function could correct a piece of text to match exactly, this will never be the case if the OCR results introduce errors such as missing words or even paragraphs. Regardless, this performs admirably and aids in the fluidity of the text-to-speech methods that will be used.

## 4.8 NLP

## 4.9 Equation Extraction

Extracting equations using Tesseract OCR is possible, and for basic equations this will return the correct results. However for more complicated equations that include for example, integrals, differentials, or sums; standard OCR engines will not work. What is necessary is using a service that has been specifically trained to detect equations from images and return the result.

Mathpix is a online API service that offers exactly that, it has a specifically trained OCR engine, to recognise a whole host of equation characters and it can convert these directly into a latex format. The service comes with a python package that requires an ID and a Key in order use. The service offers 100 free conversion per month for students and unlimited conversion for $4.99 a month. It is incredibly robust and accurate and it can be implemented in just a few lines as seen in Listing 10.

```
def process(filePath):

    store = ""

    mathpix = MathPix(app_id=idd, app_key=key)
    ocr = mathpix.process_image(image_path=filePath)
    store = str(ocr.latex)

    return store
```

Listing 4.10: Mathpix Equation Extraction Implementation

As there are no other packages or services that support python that can perform as effectively as this. The cost of purchasing a key for unlimited conversion is deemed acceptable. This function can therefore be called for each of the remaining blocks that are extracted that do not fall into the image or graph categories to extract every and all equations that may be present within the page.

### 4.9.1 Testing

The tests include a variety of equations types ranging in complexity that include:

- Additon, Subtraction, Multiplication and Division
- Polynomial Functions
- Exponential and Logarithmic Functions
- Trignometric Functions
- Integrals and Differentials

These test cases are limit in nature due to the manual conversion from latex to SSML to speech. This will be explained further in the Text-to-Speech section. For each of the above function types 25 assorted documents with simple to complex equations are used to test the speed and accuracy of the API, in rejecting blocks that do not contain equations and correctly converting those that do.The results from the tests can be seen in Table 11.

| | **AverageTime** | **Block Accuracy** | **Equation Accuracy** |
|---|---|---|---|
| **Mathpix** | 0.9 Seconds | 92% | 98% |

Table 4.6: Equation Extraction Test Results

As can be seen from the results, the API's performance is fantastic. For the few cases where the API returned a string for a block that was not an equation, filtering through the string and searching for complete words can completely eliminate every and all of these false cases. There were no cases where Mathpix missed an equation and incredibly few where the API returned an incorrect equation. In each of those cases, the API missed a few characters but did not miss the entire equation.

An example of a Fourier Transform function that contains a sum and an exponential with superscripts and subscripts is shown in Figure 4.13 and the output from Mathpix can be seen in equation 4.9.

$$
\begin{aligned}
F(k) &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi \frac{kn}{N}} \\
&= \sum_{n \in \{0,1,2,14,15\}} f(n) e^{-j2\pi \frac{kn}{N}} \\
&= 1 + e^{-j2\pi \frac{k}{16}} + e^{-j2\pi \frac{2k}{16}} + e^{-j2\pi \frac{14k}{16}} + e^{-j2\pi \frac{15k}{16}}
\end{aligned}
$$

Figure 4.13: Mathpix Input Example

$$
\begin{aligned}
&F(k) = \textstyle\sum_{n=0}^{N-1} f(n) e^{-j2\pi \frac{kn}{N}} \\
&= \textstyle\sum_{n \in \{0,1,2,14,15\}} f(n) e^{-j2\pi \frac{kn}{N}} \\
&= 1 + e^{-j2\pi \frac{k}{16}} + e^{-j2\pi \frac{2k}{16}} + e^{-j2\pi \frac{14k}{16}} + e^{-j2\pi \frac{15k}{16}}
\end{aligned}
\tag{4.9}
$$

## 4.10 Table Extraction

Table extraction can be achieved using two different methods, the first would be to utilise OpenCV and Tesseract to search through extracted blocks and identify a table structure which can be then be processed using OCR. The second method would be to use a service similar to Mathpix called Camelot. In essence it uses the same computer vision technology found in OpenCV and Tesseract but it has been specifically trained to be very good at extracting tables from PDFs. The main aim of this component of the system is to extract every and all tables present within the page.

### 4.10.1 OpenCV and Tesseract

This method involves pre-processing the the blocks by applying, in the same manor as was done for block extraction, a conversion to grayscale and then a conversion to black and white using Otsu threshold. The image is then dilated using the same rectangular morph kernel of size 8x8.

The contours within the image are then identified and passed through a conditional statement. This statement checks for bounding boxes that fall between a certain height range. This range is determined based on the font size. By default, standard font sizes fall between 10 and 12pts and this is what is used in this application.

The next step involves identifying a column and row pattern. This can be done by grouping bounding boxes together and calculating the number of adjacent boxes to identify columns and rows of more than 1. The identified boxes are then arranged based on their width and heights as the OpenCV contour function will find boxes based on confidence levels and not by page structure. By forming this matrix of contours the extreme corners of the table can then be extracted, this being the top left and bottom right.

Using these coordinates, the table can be extracted and passed through another function which will look for vertical and horizontal lines in the foreground that will align parallel to the edges of the image. This will divide the table into cells and each cell is extracted and passed through Tesseract to identify the text and save this to a matrix with the same dimensions as the table.

### 4.10.2 CamelotPro

Camelot has a python package that can be easily installed and used after registering for an Key. They offer 25 free credits and negotiable pricing thereafter but the standard rate is 50 credits for $2. While this is fairly expensive. The advantage of usng this service is that, instead of filtering through each block, the entire document can be passed through to the API and it will extract all the tables within the page for a single credit. This works because the service is incredible robust and accurate and therefore does not need the page to be broken down into blocks.

The standard Camelot python package only suports PDFs. CamelotPro is a wrapper for Camelot that includes additional functionality such as processing images of types PNG and JPEG. It also offers a simpler cleaner implementation technique, it can be coded in a few lines as seen in Listing 10.

```
def get(info, file_path, out_file):
    # Request for the tables from the page to be extracted and stored
    pro_tables = read_pdf(file_path, flavor="CamelotPro", pro_kwargs={'api_key': api_key})
    pro_tables.export(out_file + '.csv', f='csv')

    # Print out request information
    if info == True:
        #pro_tables.JobStatus
        #pro_tables
        print(check_usage(api_key))

    return
```

Listing 4.11: Mathpix Equation Extraction Implementation

### 4.10.3 Testing

A test base of 60 documents where used, divided into 3 groups each containing, 1,2 and 3 tables of varying size and information. For the OpenCV and Tesseract method, the scanned documents were passed through block extraction and block outputs were used as inputs. For the CamelotPro method, the scanned documents where passed directly into the API. The results from the test can be seen in Table 12 and they show the accuracy rate and processing time for each group.

| | **1 Table Accuracy** | **2 Tables Accuracy** | **3 Tables Accuracy** |
|---|---|---|---|
| **OpenCV and Tesseract** | 38% | 17% | 13% |
| **CamelotPro** | 100% | 100% | 100% |

Table 4.7: Table Extraction Accuracy Test Results

| | **1 Table Average Time** | **2 Tables Average Time** | **3 Tables Average Time** |
|---|---|---|---|
| **OpenCV and Tesseract** | 0.9 Seconds | 1.3 Seconds | 3 Seconds |
| **CamelotPro** | 2 Seconds | 2.5 Seconds | 2.8 Seconds |

Table 4.8: Table Extraction Average Time Test Results

The results from the test proved to be rather interesting, the Camelot API worked flawlessly in all 3 cases and while it did take longer to process than the OpenCV method, this is more than acceptable considering it produced consistently perfect results. The OpenCV method did not fair well. In the single table tests, it managed to detect the tables in the majority of documents but failed when Tesseract was used to reconstruct the and fill the table. Often returning random characters that seem to be coming from mistaking the lines in the table as 'I' and 'L'. As more tables in the blocks appeared, the accuracy fell even further as the error rate was doubled and tripped. Despite being much faster, this method is clearly not effective and hence Camelot and it's wrapper CamelotPro are used. Example outputs for the table in Figure 14.4 can be seen in Tables 4.1 and 4.2

| **Material** | **Quantity** | **Price of Each (£)** |
|---|---|---|
| Stepper Motors | 1 | 34.90 |
| Bipolar Motors | 1 | 15.99 |
| Infrared Sensors | 2 | 4.87 |
| Load Sensors | 6 | 9.04 |
| Vacuum Pump | 1 | 13.69 |
| Suction Cup | 1 | 4.99 |
| Raspberry Pi | 1 | 27.59 |
| Wires | 2 | 1.41 |
| Power Bank | 1 | 1.20 |
| Batteries | 2 | 1.49 |
| **Total** | | **168.14** |

Figure 4.14: Table Extraction Input Example

An important note to be made is that there is no inherent way to detect whether the first column or row contains the table headings. The workaround for this is to assume that the first row contains the headings and to play this back to the user when cascading down the rows. If the user notices that this assumption is indeed incorrect, then the system will swap the columns for rows and assume that the columns contain the headings.

| Material | Quantity | Price of Each (E) |
|---|---|---|
| Stepper Motors | 1 | 34.90 |
| Bipolar Motors | 1 | 15.99 |
| Infrared Sensors | 2 | 4.87 |
| Load Sensors | 6 | 9.04 |
| Vacuum Pump | 1 | 13.69 |
| Suction Cup | 1 | 4.99 |
| Raspberry Pi | 1 | 27.59 |
| Wires | 2 | 1.41 |
| Power Bank | 1 | 1.20 |
| Batteries | 2 | 1.49 |
| Total | | 168.14 |

Table 4.9: CamelotPro Output

| Matfef | QuaNtity | Price of Eac |
|---|---|---|
| Stepgga Motossf | 1 | 34.90 |
| BIpOlar f | | 15.99 |
| INfafar Sensor | | 4.8887 |
| Load Sensors | 6 | 9.04 |
| Vaaubs PUMp | | 13.69 |
| Raspberry Pi | 1 | 227..59 |
| PowER basfn | 1 | 1.20 |
| | 2 | |
| Tt | | 168.14 |

Table 4.10: OpenCV and Tesseract Output

## 4.11 Controls

Arguably the most important aspect of this project is how the user will control the system to access all the information and how that information is communicated. Considering that the target users are severely visually impaired, touch and sound are the most important senses after vision for completing tasks such as studying. When considering touch, using physical controllers as input devices for the system is crucial. This gives the user a tangible reference point when correlating audio cues with actions.

The most common and widely compatibility controller is the Xbox 360 Controller manufactured and designed by Microsoft. This controller contains three excellent forms of input modes, buttons, variable triggers and joysticks. The controller contains a vibration motor, that can be used as form of feedback or output to the user.

There are two main python packages for using Xbox controllers within Linux. Both perform the same function of making the event signals produced by the controllers interpret-able and usable within Python. Where they differ is the driver packages they used to interpret and process those signals.

### 4.11.1 xpad kernel driver

This package makes use of an Xbox Controller specific driver that has been developed for Linux called xpad. It works by assigning functions to controller event types by processing each signal sequentially. The basic events for buttons are 'pressed' and 'released', for triggers the events are 'triggered' with a value of the position of the trigger. The joysticks events state the direction of joystick relative to the axis.

It can be coded in a few lines as seen in Listing 4.12.

```
def main():
    try:
        with Xbox360Controller(0, axis_threshold=0.2) as controller:
            # Button A events
            controller.button_a.when_released = on_button_released

            # Left Joystick axis move event
            controller.axis_l.when_moved = on_axis_moved

            # Left Trigger move event
            controller.button_trigger_l.when_moved = on_axis_moved

            signal.pause()
    except KeyboardInterrupt:
        pass
```

Listing 4.12: xpad kernel driver Implementation

The major drawback with this package is that it operates on a per signal basis. The 'signal.pause()' function is necessary for the script to process each event and this brings about two issues. The first is that the system can only function within a loop where the controller is the sole means of executing functions. The second is that because it calls on functions or tasks for every single event that is detected, inaccuracies in the button hardware can often register a single press or movement as multiple events. There is no functionality whereby the latest signal is used.

### 4.11.2 xboxdrv and Pygame

xboxdrv is an alternative driver for the Xbox controller that has a much wider array of features and functions but uses the same event system as xpad. It operates based on the latest signal produced by the controller which immediately eliminates the error found in the xpad method. xboxdrv also supports builtin callback functions which will be essential when integrating controls within the system for each individual component. Furthermore, because it was developed with the Pygame

package, to give the controller greater functionality for game developers. It offers improved compatibility and addition features such as axis sensitivity controls, shoulder button support (at the time of writing this report, this is missing from the xpad driver) and vibration and LED light support. Therefore this is preferred driver and package that will be used to control and integrate the entire system

Implementing callback functions is as simple as starting the driver and setting the callbacks wherever needed, an example is shown in Listing 4.13.

```
def CallBack(controlId, value):

    # Run the scanner if the start button is pressed and released
    if controlId == 13 and value == 0:
        scanner.run()

    # Stop the program if the stop button is pressed and released
    if controlId == 14 and value == 0:
        global end
        end = True

    return

def main():
    xboxCont = XboxController.XboxController(
            controllerCallBack = CallBack,
            joystickNo = 0,
            deadzone = 0.1,
            scale = 1,
            invertYAxis = False)

    xboxCont.start()

    while end == False:
        pass

    xboxCont.stop()

    return
```

Listing 4.13: xboxdrv and Pygame Implementation

## 4.12 Text-to-Speech

## 4.13 Speech-to-Text

With the use of a physical controller giving the user an input method to navigate through the system, another method must be used to input data into the system. Without the use of a keyboard, this problem turns to use of speech. Machine and deep learning techniques have paved the way for very efficient and accurate speech-to-text software. These can be used in particular to facilitate the input of notes for sections within the page.

There are two main methods that can be used to implement this within Python, the first is using the SpeechRecognition package that acts as a wrapper for a number of leading Speech-to-Text services and the second is to directly use each of those services. These include:

- CMU Sphinx (offline)
- Google Speech Recognition (Original)
- Google Cloud Speech API (Latest)
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text
- Snowboy Hotword Detection (offline)

## 4.14 Data/File Structure and Format

When designing the file structure for this system, several considerations need to be taken into account. These are, storing all the important information from each scanned document for the user to access at a later date, a directory for temporary files to be read and written to and a directory for all the source files with access to the necessary standard data templates and API Keys. Before designing the structure, the data formats for each file type are defined based two criteria. The first is compatibility and the second is ease of implementation within Python.

### 4.14.1 Data/File Formats

There are 4 main file formats used, the first is JPEG for all images. This is used because of it's wide compatibility with python packages that process images. OpenCV, Mathpix and CamelotPro all makes use of JPEG files. PNG files were not used because of image transparency issues that can often confuse and create errors with passing through image processing methods or packages. Although JPEG compress and removes a lot of the information from images to reduce the file size, it was found through testing that this did not not pose much of an issue and the incompatibilty and errors produced from using formats such as PNG and GIF made it the most viable option .

The second is MP3 files for audio. This is chosen simply because of it's widespread adoption in almost all audio systems. It can be easily be recognised and played through almost every single audio player across a variety of systems. While the Graph audio script in this system generates temporary files in the WAV format, using the Pydub Audiosegment, these can be easily converted to MP3 files with a single line of code. As the WAV files are never played back by an audio player, the temporary read and write operations to these poses no risk.

The third is CSV files for graph data and table data storage. This format was once again chosen for it's capatibilty with Python through the CSV library and also for it's easy conversion to Excel and ODS formats in Linux and Windows for visualising the data when Matplot tools were not convenient.

The final format is JSON for storing all the remaining information from extraction. This format was used for it's serialisation structure. The Keys can be set to identifying categories of information and these value pairs can then be set to store a wide array of data. In this application it used to store the following:

- User defined page description or identity
- If the user has bookmarked the page
- Image descriptions
- Graph descriptions
- Equations
- A breakdown of the blocks of text within the page
- If the user highlighted any sentences
- If the user stored any notes for each block
- UI script

An additional JSON file is used to store the Script for the system separate from the rest of the data extracted from the document. The template JSON structure used for storing this information for each page can be seen in Listing 4.14.

```
{
    "page": [
        {
            "identity": "",
            "bookmarked": "",
            "image_results": [
                {
                    "0": ""
                }
            ],
            "graph_results": [
                {
                    "0": ""
                }
            ],
            "equations": [
                {
                    "0": ""
                }
            ],
            "sections": [
                {
                    "full_text": "",
                    "description": "",
                    "notes": "",
                    "nlp": [
                        {
                            "0": ""
                        }
                    ],
                    "sentences": [
                        {
                            "text": "",
                            "highlighted": ""
                        }
                    ]
            }

            ]
        }
    ]
}
```

Listing 4.14: access.JSON template

### 4.14.2 File Structure

The file structure is divided into 3 main directories. The 'src' directory contains all the Python scripts and the script and the data template JSON files. It is important to note that all of these individual scripts were placed into one folder by design rather than separating them into sections. When testing the system as a whole it was found that certain packages such as OpenCV and Mathpix, ran into issues when the main script that was importing those functions from their separate directories. The functions would execute extremely slowly or even fail. The exact issue that caused these event is still unclear but placing the scripts in one location and using sub-processes to create threads for certain functions fixed all of these issues. This directory also contains a sub-directory for all the UI navigation audio files.

The 'library' directory stores the CSV and JSON files generated when each document is scanned and is chosen by the user to be stored. The files are placed into a numbered folder and the page descriptions within the 'access.JSON' files are used to identify each page when the user wishes to access them again.

The 'temp' directory stores all of the temporary files that generating from each component. The directory is divided into a sub-directories for audio, blocks and csv files, with blocks broken down further into graphs and and images. When each new document is scanned, the temporary files are wiped after they have been stored by chosen by the user.

Figure 4.15 illustrates what the file structure looks like.

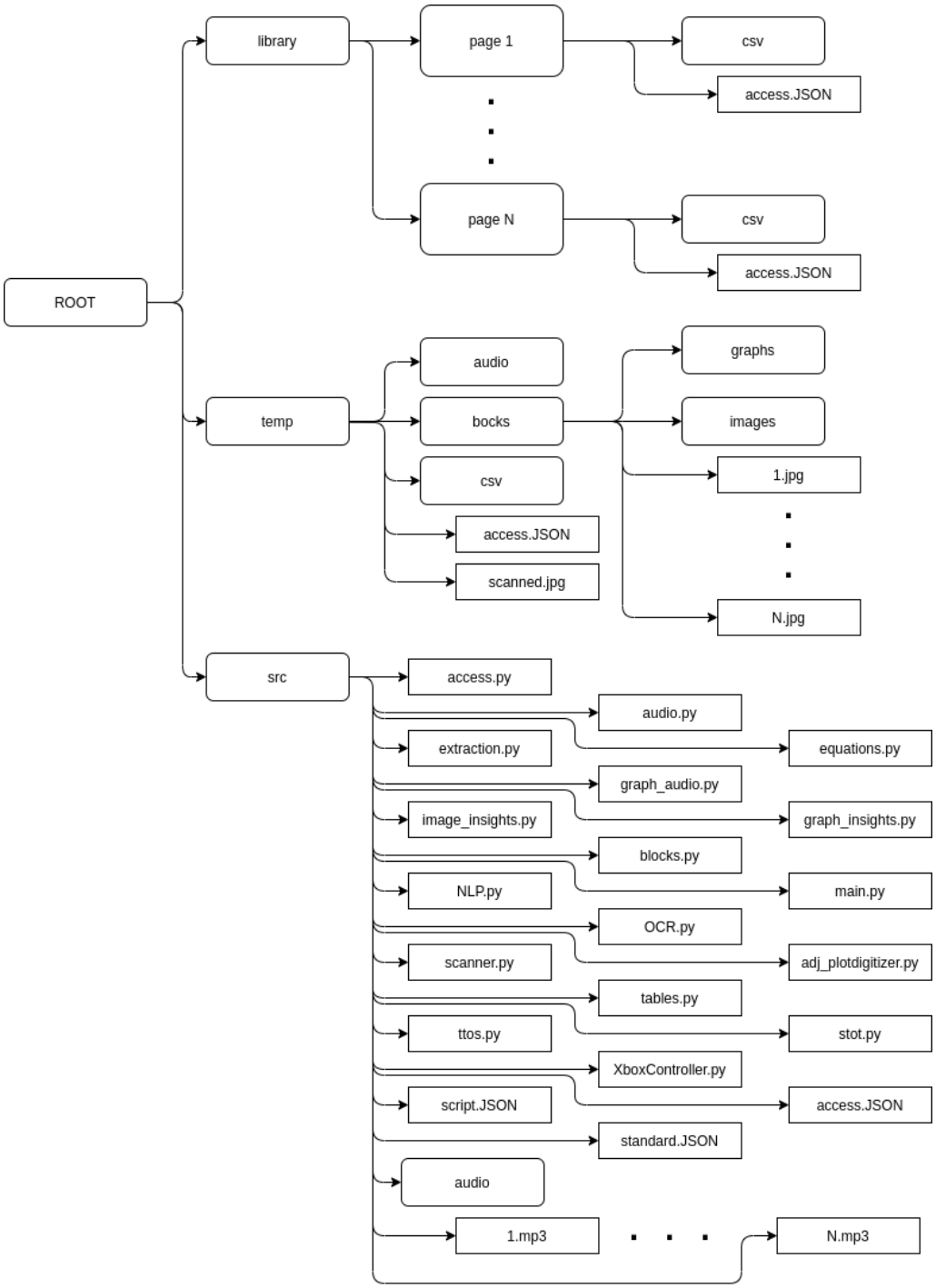



Figure 4.15: File Structure

## 4.15 System Integration