Imperial College London

MEng Individual Project

Imperial College London

Department of Electrical and Electronic Engineering

Assistive Technology to Study STEM Subjects for Visually Impaired People

---

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 24, 2020

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, electronic copies of my final year project report to both Blackboard and the EEE coursework submission system.

I affirm that the two copies of the report are identical.

I affirm that I have provided explicit references for all material in my Final Report which is not authored by me and represented as my own work.

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. These individuals face several unprecedented challenges, one these being education. Advances in hardware and software have paved the way for many solutions both simple and complex; from text enlargers to braille translation devices. However, these solutions are not adequate for students wishing to study more complex STEM subjects. Interpreting information with items such as graphs and tables is extremely difficult. This problem is further heightened, as much of the national curriculum still uses dated non-digital paper material; resources and equipment have also become exceedingly expensive and made it difficult for enough teachers to be properly trained. This project tackled this problem by developing a system that incorporated computer vision and machine learning technologies, to enable students to study and pursue a passion for STEM subjects. It is able to scan the literature of interest and extract all the information present within it. Graphs, tables, images, equations and text are all converted into speech or audio that can be interpreted and understood by the student. They can navigate a virtual interface using a joystick controller to access all this information. User testing proved that this system was effective at extracting and presenting the information to the user, enabling them to study and answers questions from the material. This simple intuitive virtual interface also made it quick and easy to learn and train students and teachers; all while being considerably cheaper than current solutions and methods.

Acknowledgements

I would first like to thank my project supervisor Dr Edward Stott, for his support and guidance throughout this project. His invaluable feedback and experience brought this project to fruition and kept on me on the right track at every turn, right from the beginning to the end of this project.

I would like to thank Jonathan Wong and Ivaxi Sheth for their fantastic company, support and encouragement these past four years at Imperial. They have always pushed me to grow as an engineer and to be fearlessly ambitious. We have formed a truly brilliant trio. I would also like to thank Akila Sekar for her unwavering support and care, she has kept me going through the roller-coaster of a ride that has been these last few years. I would also like to thank Advaith Sastry for his patience and kind ear throughout this project, he has stuck by me through the stressful periods of this year and project.

Lastly, I would like to thank my parents and brother, Dr Chikkanayakanahalli, Swarna and Athul Manjunatha. They have always been my biggest supporters and without them, my time at Imperial would never have been possible. They instilled in me, a real sense of drive and passion, which has truly fuelled my pursuit of following my passion for engineering. For this and for never letting me give up, I am eternally grateful.

To all my family and friends: I am incredibly thankful for all the truly unforgettable memories we have created together over the years and for the countless words of wisdom and continued support you have given me.

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 [10]. In 1988 the estimated number of severely impaired or blind people in the world was 37 million [11]. 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 care services [12]. Great efforts are being made into preventing many of these causes and in developing proper services for those without access to eye care. Despite this, there are still millions currently living with severe impairment 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 is often limited or restricted. There are many specialist schools with qualified and trained teachers to help develop specific curriculum's and provide support. 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 [13]. In truth, many of these children have great potential to pursue their passion in an academic field just as much as others [14]. Expensive equipment, lack of training for teachers and limited funding are some of the reasons why these children are hindered. STEM (Science, Technology, Engineering and Maths) subjects, in particular, are difficult to teach and to develop good resources for. This is large part due to the complex nature of these subjects and how they are presented through items such as graphs and equations. These are very visual representations of information and it is known to be very difficult to portray this by other means such as speech [15].

Many existing assistive technologies work well to make pre-prepared and digital material easy to interact and interpret. Technologies such as speech synthesizers, screen or text enlargers and braille translation software do this very well [13]. However, these methods are often limited to selected material and do not further analyse or convert any information, to make it easier to interpret or understand for the student. Furthermore, much larger and expensive equipment only remain usable within the classroom. Current methods to convert non-digital textbooks and papers into digital copies have only partially been successful. Many books converted into braille can be as expensive as $15,000 for the initial conversion and then $500 per textbook thereafter [16]. Text recognition software is often used to convert non-digital text into speech, however, it is primarily used in a primitive way. Converting the identified text directly to speech without any control or additional processing to make better use of that information for the user, such as forming summaries.

It is this lack of development in using current advancements in computer vision and machine learning technologies that motivates the research and development of this project. There is great potential to develop a system that can provide greater freedom for visually impaired and blind people. 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 STEM subjects, in and out of the classroom.

Chapter 2

Background

2.1 Visual Impairment and Blindness

Visual impairment occurs when the visual system and its functions are affected negatively, due to health conditions, these can form at birth or later on in life. 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 a severe visual impairment and one with less than 3/60 is considered to be blind [10]. 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 [17].

2.2 Teaching and Curriculum

QTVI or qualified teachers of children and young people with vision impairment are specialised teachers who educate those who are severely visually impaired from birth to the age of 16-25. Their roles range from direct at-home support for babies and parents to specialist skills training such as braille and independent living. Those who are QTVI must undergo a mandatory qualification to teach children and support their families [**?**].

There are a variety of development programs and structured curriculum guidelines for each major stage in a child's development process. DJVI or the development journal for babies and children with vision impairment deals with very early stages of growth. The majority of the 25,000 severely visually impaired children in England were born with this disability. This framework is essential for parents and support workers in aiding effective early age development. Similarly, there are national curriculum guidelines and support resources for primary, secondary and further education. In each of these, teaching methods and practices are discussed for each subject and the recommended resources and equipment that could be used in conjunction, are stated [**?**].

While these guides offer what is thought to be the best methods for teaching children with a vision impairment of all degrees, in practice, this is not always the case. Teachers are progressively finding that tools and resources from other methods, used for children with different disabilities such as dyslexia, can often be far more effective. While there are plenty of professionally qualified teachers, many have reported that regardless of this, they still feel they are often unequipped with the right knowledge, equipment and resources to maximise the potential of each student [13].

STEM subjects, in particular, pose several challenges for teachers; many visual concepts and mathematical models are extremely difficult to make available in an accessible format. These include images, graphs, diagrams, equations and alike. Current teaching methods and equipment are simply not as effective as they could be, particular efforts to convert and teaching this material in audio and braille forms. Research has highlighted this issue and states the need for further development and training [15].

In recent years, funding cuts and fewer people getting into teaching has resulted in a lack of adequate support and a reduced number of qualified teaching professionals. There is a strong need for new methods that can allow these students to tackle education more independently and through means that are more affordable and easier to train teachers and support works with.

2.3 Existing Solutions

There is 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.

2.3.1 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 implemented in, these devices are still expensive and many external video-based enlargers are even more expensive. The Acrobat HD Ultra LCD, for example, is priced at 1,945.00 [18], 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 [19].

(a) Acrobat HD Ultra LCD [18]　　　　　　　　　　　(b) AUMED IMAGE [19]

Figure 2.1: Enlarger/ Magnifier Examples

2.3.2 Text-to-Speech

Text-speech has been one of the most prevalent solutions in devices for making text information available to the blind. Its use is widespread everywhere from computers to accessibility help points at train station platforms. All of these methods use predefined text or they extract text from a source and convert that into speech to be played back to the user. With advancements in machine learning, 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 [20], can detect text from images but either presents very little control over the text or are simply cumbersome 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.

2.3.3 Speech-to-Text / Braille-to-Text

Speech-to-text is a 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. This can be used to perform basic tasks such as note-taking and emails or even basic commands for software assistants such as Siri, to ask 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 into braille.

A less common input method is braille. This is when a device takes braille input from a user and converts this to text. While this has some great benefits such as greater accuracy and increased skill proficiency for the user over time, they are all extremely expensive. For example, the BrailleNote Apex is priced at \$1,995 [21] and these devices tend to be quite bulky.

Figure 2.2: BrailleNote Apex [2]

2.3.4 Braille

Braille is a method by which characters are represented by an array of raised dots on a page. Touch reading is the method of using one's fingers to read the pattern of raised dots [22]. Braille solutions have seen 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 [16]. Braille devices that can convert text input from a computer to braille in real-time are extremely impressive but again are expensive and the same can be said about braille notetakers such as the BrailleNote Touch 18 Plus which is priced at \$4,195.00 [3].
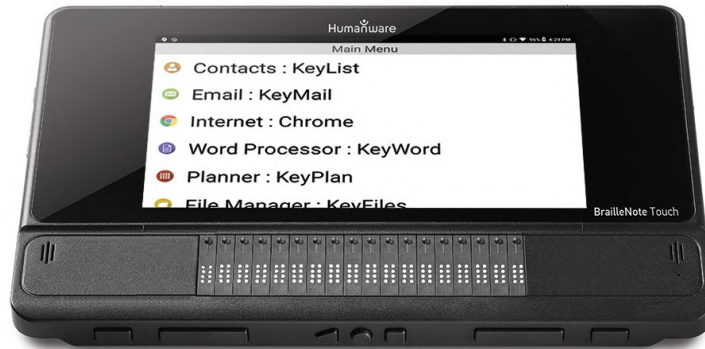
Figure 2.3: BrailleNote Touch 18 Plus [3]

2.3.5 OrCam

OrCam is a startup company based in Jerusalem, Israel, that has developed a wearable assistive device that enables users to read text-based information, recognise faces and identify products. The idea behind this product is to give visually impaired people the ability to explore the world around them. Using this device they can, for example, go shopping. The device, attached to the user's glasses will identify objects held within a certain distance such as a tin can. It will identify the can and read out the text printed on it to the user. It can also recognise a small selection of saved images of people that can help the user identify close friends and family within the devices field of vision. It can also interpret text on newspapers and documents and read that extracted information.

This device is one of the most advanced solutions that has been developed to date that does indeed make use of machine learning and computer vision technology. However, it's purpose was never to enable individuals to study, hence it lacks some necessary features and once again the price forms another drawback. They offer 3 devices, the OrCam MyEye, MyReader and Read; the cheapest of which starts at $2500 [4].

Figure 2.4: OrCam Device Range [4]

2.4   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 [23]. 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. [24]. Algorithms that can take advantage of that, have made digital image processing so prevalent in devices such as smartphones. In essence, it is the process of altering the information of an image through these algorithms; it shares a lot of similarities with signal processing techniques and the major concepts from 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 from 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.

2.5   Machine Learning

Machine learning is the ability of a system to learn from experience without the need to directly program its function. Often the system is given a large data set (training set/data) and it will 'learn' to extract patterns and algorithms [25]. Several main learning methods are commonly used in the technologies, services and tools that will be made use of in this project, these are discussed below.

Association Learning

This rule involves finding relationships between variables in data sets based on measures of interestingness. The algorithm seeks out complementary 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, whereby 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 of 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 many 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 2.5 gives a visual representation of how regression and classification deal with data.
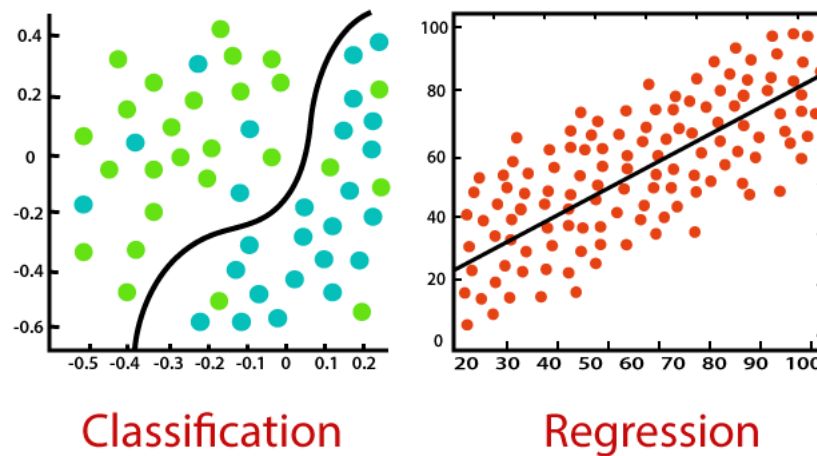
Figure 2.5: Visual Representation of Classification and Regression (Linear) [5]

Unsupervised learning

This is a method whereby 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 its algorithm based on the 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 the 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.

2.5.1 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 [26].

2.5.2 Deep Learning

The major distinguishing factor between deep learning networks and neural networks is the depth of the hidden layer; figure 2.6 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.
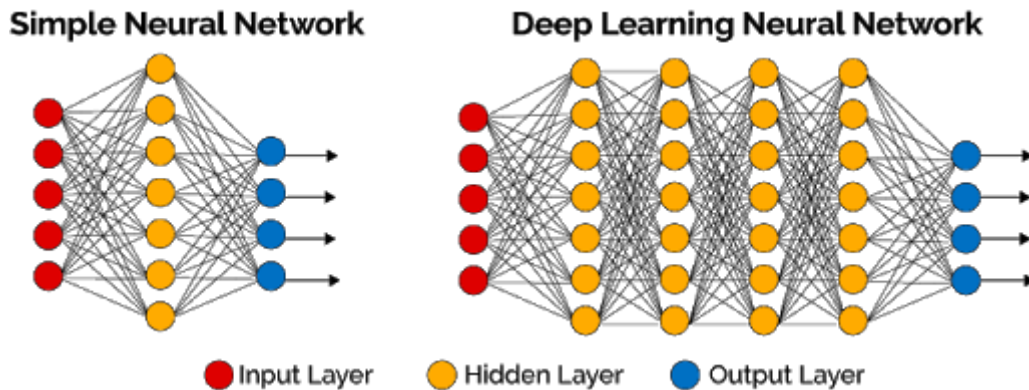
Figure 2.6: Illustration of a Simple Neural Network and a Deep Neural Network [6]

It is important to understand the basics behind the 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.

2.6 Optical Character Recognition

Optical character recognition is an age-old computer vision task that can be dated back as far as 1914 [27]. 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, multiple methods 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 is 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 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.

2.7 Speech

Speech is a complex communication medium, it can be broken down into several 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 the context of the audio, and so all of these can have varying definitions. 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 [28].

2.7.1 Recognition

Three models are used for speech recognition that matches 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 from 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' [29].

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

2.7.2 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 [7].

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 these 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 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 can identify features that are only computer recognisable which is one of the reasons why this method is better than Parametric TTS. It can extract features from a data-set, developing its 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 2.7. 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 until 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.
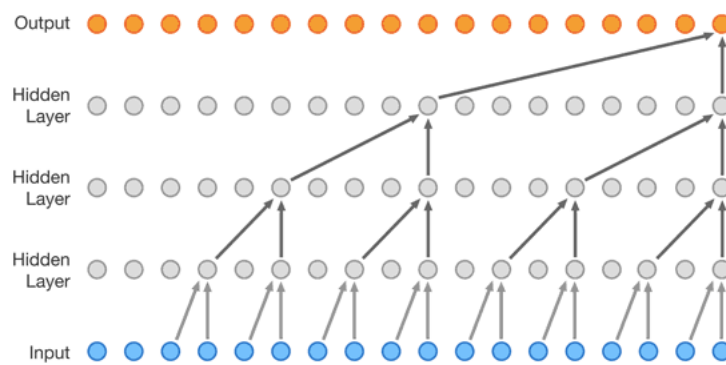
Figure 2.7: Illustration of the WaveNet Network [7]

2.8 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 to form a meaning of the text it is processing [30]. 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.

2.9 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 sounds 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 2.8 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 [8].
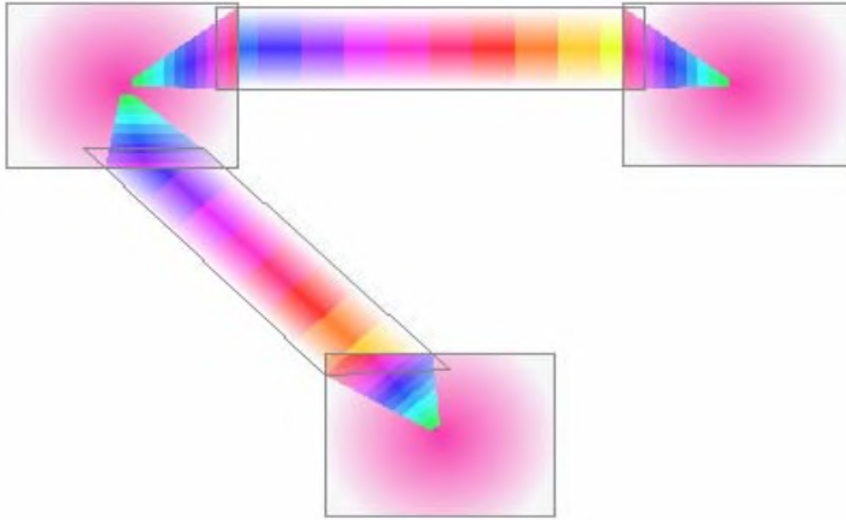
Figure 2.8: MIDI guidance of a 2 edged graph illustrated using HSB colour [8]

2.9.1 Sound Synthesis

Sound is simply the propagated vibration of a wave through a medium. Often that wave is known as being acoustic 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 can be used.

Chapter 3

Project Specification

3.1 Project Definition

The project can be easily specified after considering the background research that was conducted and by examining the type of material, students studying STEM subjects would come across. Figure 3.1 provides two examples of secondary level physics notes that discuss Ohm's Law and maths notes that discuss parabolas. Annotated are the five most common items found within these types of notes and textbooks.

(a) Maths - Parabola [31] (b) Physics - Ohm's Law [32]

Figure 3.1: Secondary School Level Maths and Physics Material Examples

The basis for this project is, therefore, to develop a system that at a minimum enables students to study these items from a given document:

- **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 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

Although various resources 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. Hence, **the system developed must be capable of using a camera feed to identify and scan the document of interest before digitising it.**

The items listed must then be extracted from the scanned document before being processed individually and making that information available through an interface. **This virtual audio interface must be controlled via a physical controller and the information extracted must also be stored for the student to access at a later date**, so that there is no need to scan the material again.

With limited features and functions in existing devices that allow these students to take notes and analyse the text as one would through highlighting or bookmarking sentences and pages. These should be implemented directly into the system, rather than separately. **Notes should be added by converting the student's speech into text and the controller should be used to identify key pages and sentences.**

Working on top of current solutions that simply extract the information within each item and present them in their most simple format. For example with text, most systems will simply read out whole paragraphs without creating summaries or breaking the text down further into sentences or words. With images, textual descriptions are only now becoming available with the first real-world example being "Get Image Descriptions" through the accessibility settings in Google Chrome [33]. **Hence, for each type of item that is extracted from the document, they must be processed and presented to the student in a way maximises their ability to properly and effectively understand and capture the information being portrayed.** As an example, graphs should be represented not only through audio, as seen in Section 2.9, but also a textual description of the function model it best represents.

It is also important to note that this system should be made use-able by normally sighted individuals, to ensure that it is easy to train teachers and students.

3.2 Software Specification

This project is almost entirely software-based and its structure is essential to ensuring that it operates efficiently, openly compatible to support future development. 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 Python3.6, which will enable quick, easy and well-documented access to a vast array of packages and libraries. The nature of this project also means that there are a significant number of individual components that must operate as one, integrating these will form the backbone of the project.

3.3 Hardware Specification

To use the Python programming language and to keep the costs as low as possible, the system must be capable of running on the latest Raspberry Pi 4 along with the Raspberry Pi Camera Module v2. This was determined for it's Linux based operating system compatibility and small form factor. The system must also utilise the Microsoft manufactured Xbox 360 Controller. With a 4GB Raspberry Pi and wired controller, the total initial hardware cost amounts to 74.

3.4 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 and footage from a smartphone were used 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 and tested with minor adjustments.

Chapter 4

Implementation

Implementing such a system poses many challenges, the largest of which is the inherent disparity in how the extraction of each item within a page, must be approached. Each will require different methods, different resources and tools and different forms of testing. The nature of this project, therefore, lends itself towards an iterative implementation process, where for each component, several solutions are considered and tested upon which the entire system is integrated. An overview of the various components that need to be implemented can be seen in Figure 4.1.
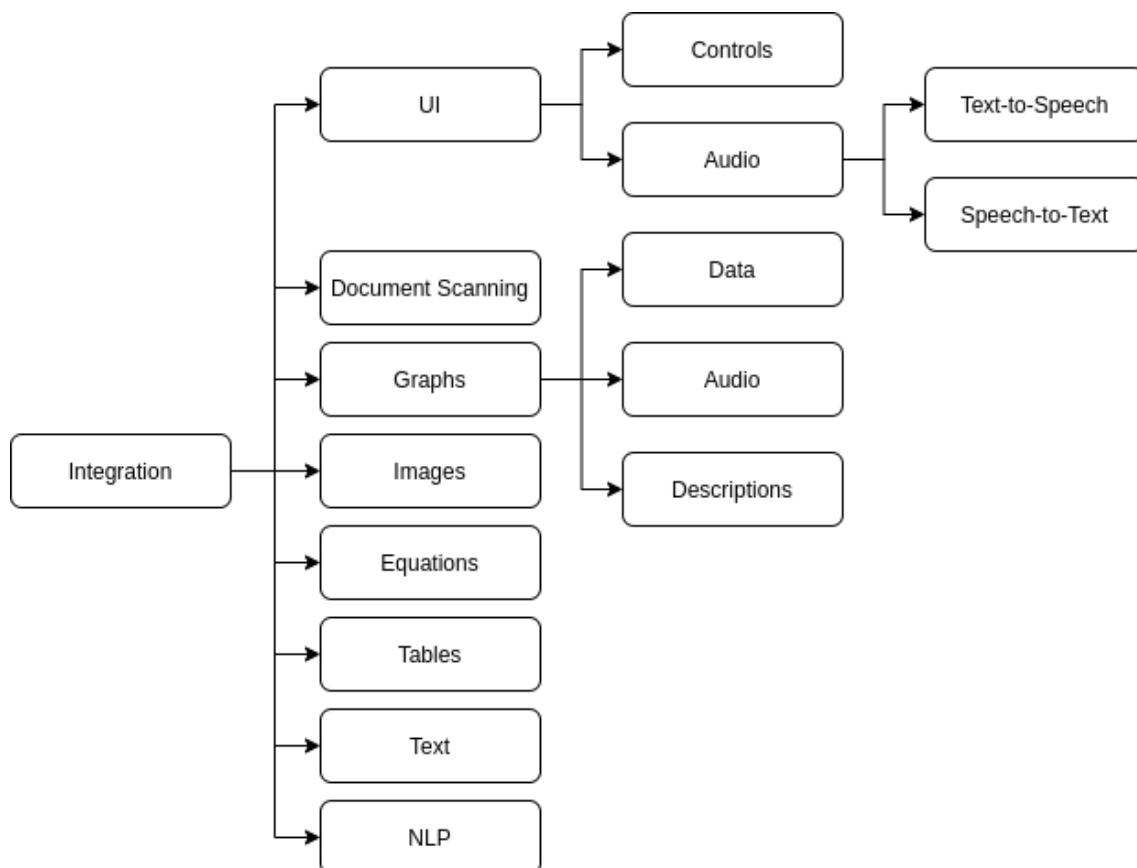
Figure 4.1: Implementation Overview

4.1 Document Scanning

To capture the page the student wishes to study, the system must be capable of performing what is essentially document scanning using a video feed from the camera. 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

To detect the page within the video feed, it is broken down into video frames that are essentially images. Several image processing steps need to be carried out 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 functions in the OpenCV library.

The first step is to apply a grayscale filter on the image using the function, **"COLOR_BGR2GRAY"**. 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 grayscale helps improve this. Secondly, working in grayscale reduces the number of computational resources required because of the reduced information density. Finally, other APIs and services that are to be used for extracting other 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 and all are used to reduce the noise in an image, hence increasing the signal to noise ratio. Median filtering is known for retaining good sharp edges and performs a good job if the image is of a particularly noisy. For this application, the images will not be incredibly noisy and hence the other 2 methods are favoured. Averaging filters work by taking the average of neighbouring pixels and assigning that to the pixel being smoothed. Gaussian filters will assign weightings to pixels based on distances and hence is, in general, better at reducing noise within images. It also has better frequency separation and Gaussian noise reduction properties. This is the method used that was chosen to be used in this application and is implemented using the **"GaussianBlur"** function.

The final step is to apply some form of edge detection. There are 2 main methods, the first is using 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 for the pixel values to be manually defined. Adaptive threshold is a method by which the thresholds are calculated based on smaller regions of an image rather than the entire image. Both perform admirably, but for this application Adaptive threshold was deemed more fitting. This is because it can better handle a larger variety of background environments by automatically adjusting the threshold. This was implemented using the "adaptiveThreshold" function with a binary conversion to identify the edges using black and white pixels.

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. After the binary conversion there remain only two values, '0' for black and '255' for white. OpenCV has a built-in function, **"findContours"**, 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 largest contour likely to be the document has been detected but conditions are set to ensure with some confidence, the contour is indeed the document. The first condition is that the area of the contour must be within a certain percentage range of the total area of the image. This serves 2 purposes, it ensures that smaller objects are rejected such as tables within pages and that the entire image itself is not detected as a page. Additionally, this ensures that the majority of the image resolution is being used to capture the document. This, in turn, allows for as much detail to be retained as possible. The second condition is time; the reason being that it was found that jittering was a very common issue. With a video feed operating at 24 frames per second, it was common to see that each frame would return a different contour. Setting a time-based threshold means that the contour must be consistent over a certain period in area to be captured. These two conditions proved to be very effective when set at 75 - 95% 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 image. The warp transform uses this information to warp the image to form the output. This will ensure that the contents of the image are not distorted after the transform by acting against it. The functions used to do this are **"getPerspectiveTransform"** and **"warpPerspective"**.

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 capabilities of the system in real-world situations, where it is unlikely that the student will be able to position the document correctly.

- Straight - The document is position parallel to the edges of the feed

- Slanted - The document was positioned at an angle to the edges of the feed

- Complex Background - The document is placed amongst other items 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 text, graphs and images. This is to test the systems ability to handle complex documents that may contain smaller contours from graphs and tables within them. Each video feed is approximately 15 seconds and it is expected that this component should reach accuracy levels above 90% in perfect conditions, as it follows a tried and tested method that has been used in countless other applications with similar results. The accuracy in this case is defined the number of successfully identified full documents: this means the entire page of interest must be extracted with no important areas being missed out. Some missed portions such as edges or corners are ignored as this does not contain any information and is subject to creases and folds which can make the page shape less like a perfect rectangle. An example of the output for each test case is shown in Figure 4.2 and 4.3 along with the key steps mentioned in the previous sections including Canny Threshold which was chosen not to be used.
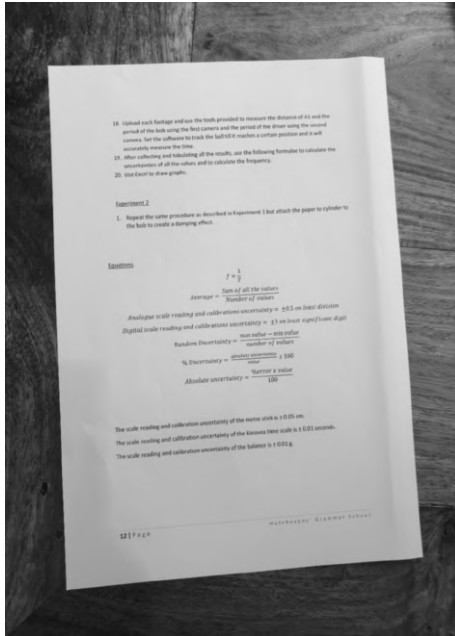
(a) Grayscale and Blurring
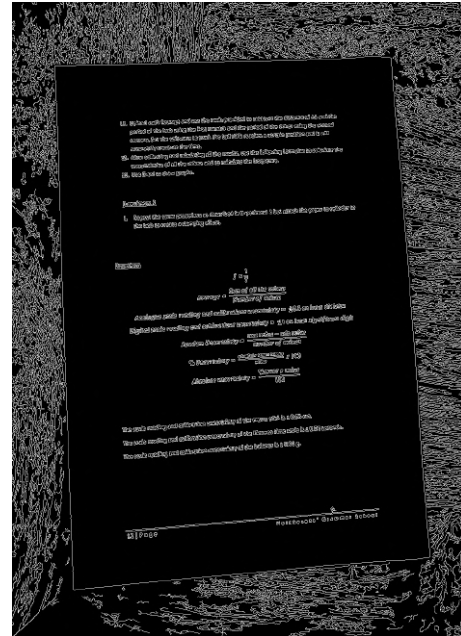
(b) Canny Threshold
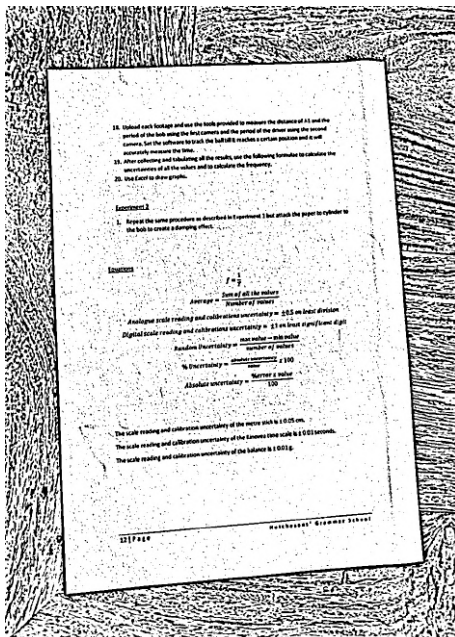
(c) Adaptive Threshold
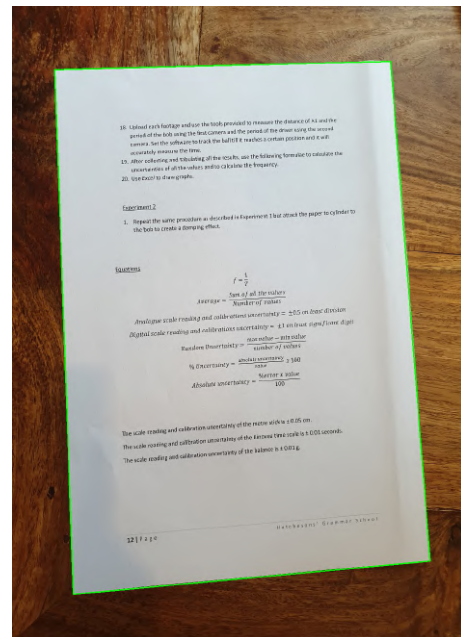
(d) Document Contour

Figure 4.2: Document Scanning Stages
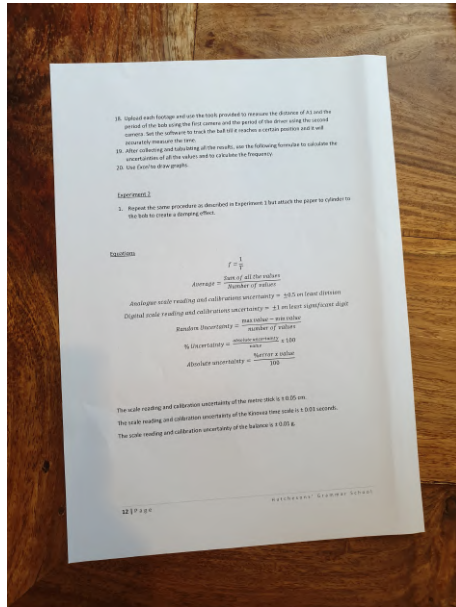
(a) Original
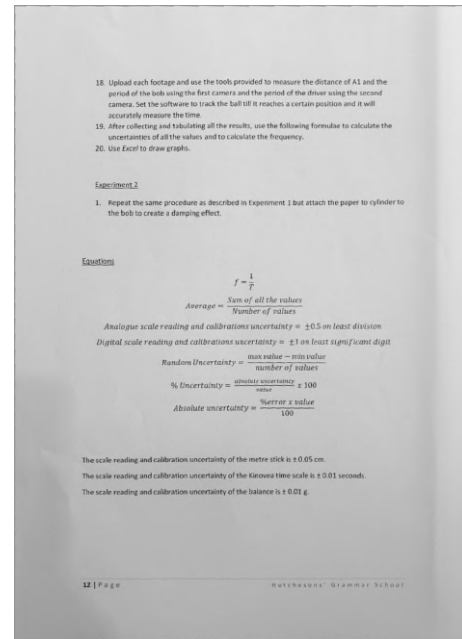
(b) Document

Figure 4.3: 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 expected. 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 those documents that it was not able to detect in both categories, a simple solution 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 after a false detection.

The final category, complex background, saw the biggest drop in performance, specifically in accuracy. From the example shown in figure 4.1, it's clear to see that with more complex items in the image, it's easier for the system to fail with what is essentially more 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 than recorded in these tests, this will be tested and evaluated in the usability testing section later.

With the document now scanned and ready to be processed, the following sections will now look at extracting blocks of items and methods of extracting information from those blocks.
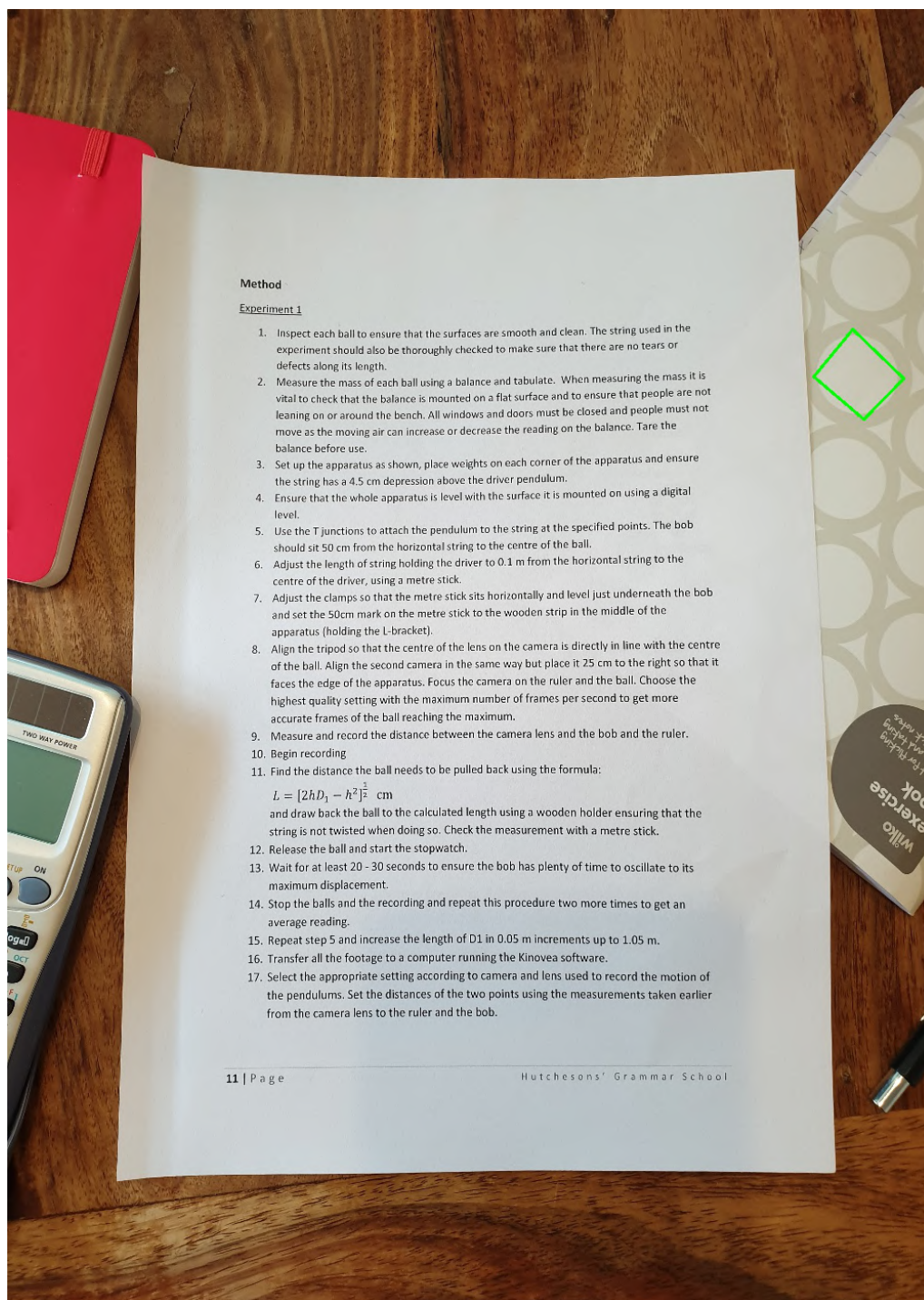
Figure 4.4: Complex Background Example Output

4.2 Block Extraction

The next major step is to breakdown the page into several blocks. Each block is defined as a group of information within a page. This includes images, graphs, tables, equations and text. This is a very important component of the system as many other processes are dependent on the successful extraction of these blocks, in particular, graphs and images, and text. Once again this component can be achieved by using OpenCV and it's myriad of functions.

4.2.1 Image Processing

The first step is to process the image to remove any shadows or illumination. This will provide a clearer image that can be processed more accurately. To do this successfully, it must be ensured that the information within the page is retained without much alteration. 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 those values over the information within the page which is typically dark grey or black and can be implemented using the **"dilate"** function.

As an added layer of text removal, the image can be smoothed using a blurring method. As explored previously in document scanning, 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. This can be implemented using the **"medianBlur"** function.

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 using **"absdiff"**. 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 can 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 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. **"normalize"** and **"THRESH_TRUNC"** can be used for these processes.

The image is then converted back to greyscale and inverted to obtain a black and white image with the text in white and background in black. This is performed using an Otsu threshold (**"THRESH_OTSU"**) 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 greyscale 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 4.5 and 4.6.
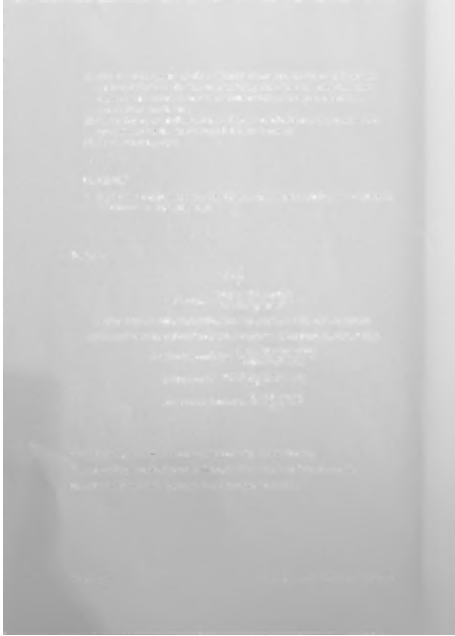
(a) Dilation

(b) Smoothing
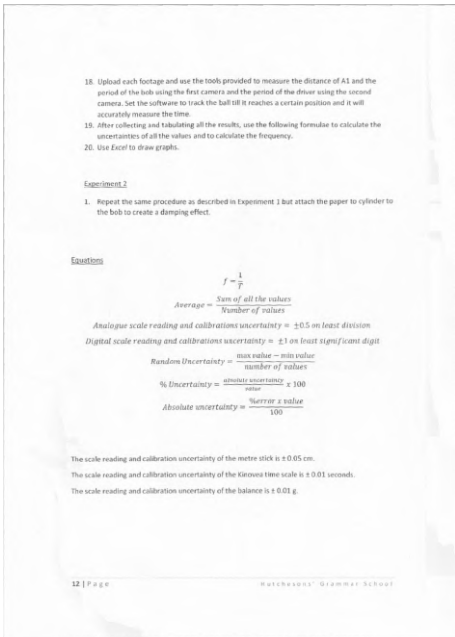
(c) Subtracted Background
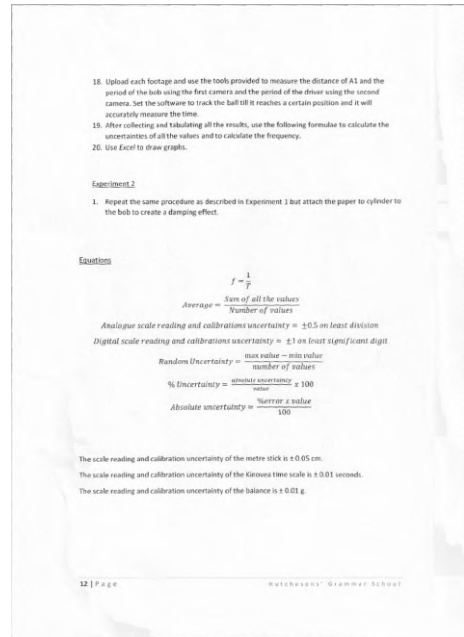
(d) Normalised and Truncated

Figure 4.5: Block Extraction Image Processing (1)

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.

(a) Otsu Threshold and Binary Inversion

Figure 4.6: Block Extraction Image Processing (2)

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 sizes tend to be 3x3 matrices. As the 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 initial testing against a variety of documents yielded 2 kernel sizes.

In both cases, the kernels are capable of capturing and obtaining all but one of either graph and image blocks. These two kernel sizes and the number of iterations that the dilation is performed on the image allows for both graphs and images to be obtained 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 needed 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 is 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 fewer words and text are 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 for 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 the most common forms found in reports, papers and notes. These parameters are left open for optimisation and the final configuration is stated in the results.

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. Several considerations need to be made. As the document is almost perfectly fitting to the dimensions of the image, OpenCV may detect the entire page as a block, this will occur if small areas around the edge of the document still contain some of the backgrounds. This can be avoided by comparing the horizontal and vertical lengths of the contours with the length and height of the image. If the lengths are above 90% of the dimensions of the image, then the contour can't 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. Many complex methods can be used but they all require additional computational power such as per block OCR. These methods are time and resource expensive and faster simpler methods are preferred to run on small devices such as the Raspberry Pi. 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. 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 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 a 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 its 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.

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 in OCR operations later in the system as it removes the errors that arise in the detection of text in graphs and images such as axis values and labels. The graphs and images are then saved to specific locations within the system to be processed further.

4.2.3 Testing

A test base of 100 documents all containing a range of text, images, graphs, tables and equations were used. The test aimed 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 without introducing errors. For each test, the blocks were broken down into the 3 categories, graphs, images and other. The accuracy was measured as the number of correctly identified and extracted blocks. This is determined by comparing the actual output against the expected output for each document. Incorrect results would include, wrong block identification and blocks that miss or crop out important information.

	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

	Average Graph Accuracy	Average Image Accuracy	Average 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 pages resulted in the 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 as a prototype. 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 block detection can be seen in Figure 4.7. It illustrates the output from the dilation steps and the final block detection contours.

(a) Image Dilation

(b) Block Detection

(c) Graph Dilation

(d) Block Detection

Figure 4.7: Block Extraction Dilation and Contour Examples

4.3 Graph Data Extraction

With the graphs now extracted from the document, the aim is to try and 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 sheer volume of variations in graph styles and formats pose the greatest challenge. Despite this, there are some tools and services that been developed that aim to tackle this problem.

All of these tools function similarly. The image of the 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 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 the 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 group.

Enguage Digitizer

Engauge Digitizer is a popular well developed, Linux, Windows and Mac-based application that supports the conversion of a 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 its 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 manner 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 the 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 the 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 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 include 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 its 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.
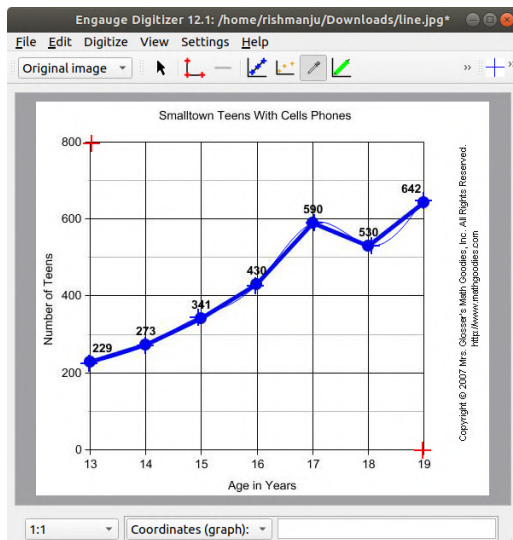
(a) Enguage Digitizer Screenshot
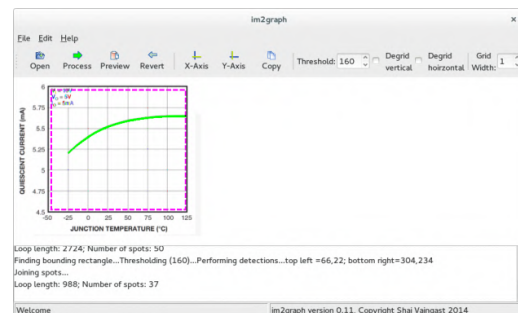
(b) im2graph Screenshot
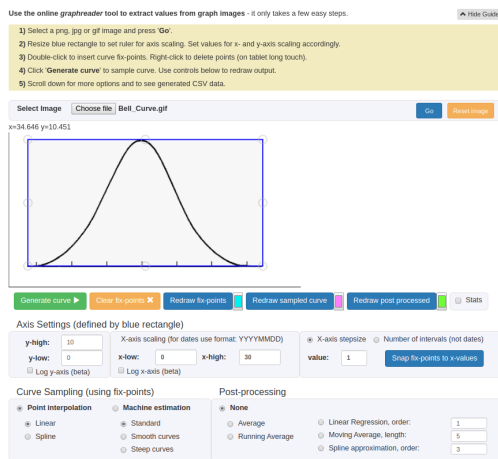
(c) GraphReader Screenshot
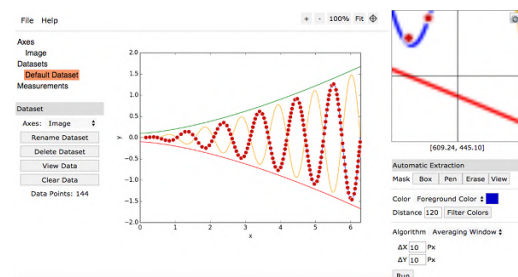
(d) WebPlotDigitizer Screenshot

Figure 4.8: Existing Graph Extraction Solutions Examples

4.3.2 PlotDigitizer Adaption

The PlotDigitizer or PD package has four minimum requirements, the image file of the graph and three 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 several adjustments need to be made 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 an 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.

Various methods 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 two black linear 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. In particular the noise that is introduced when grid lines and keys are present in, on or around the plot. Even for defined style sets, the accuracy rate never rose above 17%; an example of the typical output from PlotDigitzier when these methods are used to estimate the corner point is shown in Figure 4.9.

(a) Input Image

(b) Ouput Data
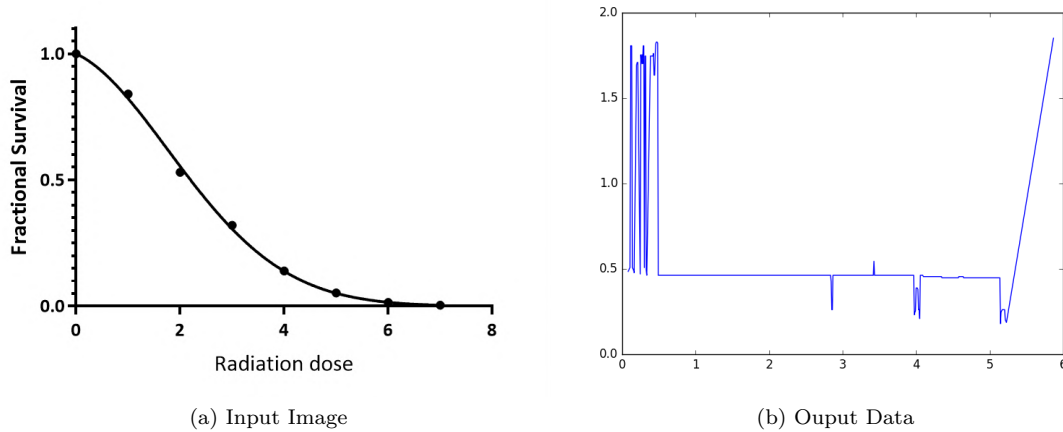
Figure 4.9: Failed (0,0) Point Estimation - PlotDigitizer Output Example

40

Dividing this problem into 2 parts, simpler crude solutions were 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 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. Bypassing the inbuilt "clickbutton" procedure and with this method enables the entire package to operate as intended, this implementation has been shown in Listing 4.1.

```python
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: PlotDigitizer Coorindate Implementation

4.3.3 Testing

The completed script was tested using the same data set as mentioned above, the results of which can be seen in Table 4.5. The accuracy is measured as the successful extraction of the shape of the curve in the graph, it is noted that in all cases the data is likely to be very noisy as is discussed later. This is disregarded for the time being and the shape, which is the most important part of the graph extraction is solely considered.

Average Accuracy	Average Processing Time
96%	0.23 Seconds

Table 4.5: Graph Extraction Test Results

Of the 50 graphs that were tested, 48 graphs had their curve successfully extracted and in less than 0.3 seconds. An example of the output from the script given a Sinewave can be seen in Figure 4.10. One very important observation is made, that while PlotDigitizer managed to roughly extract the shape of the curve, there is also a lot of variability in the data. This is because of the residual interference from both the grid and axis 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) Sinewave Input Image   (b) Ouput Data
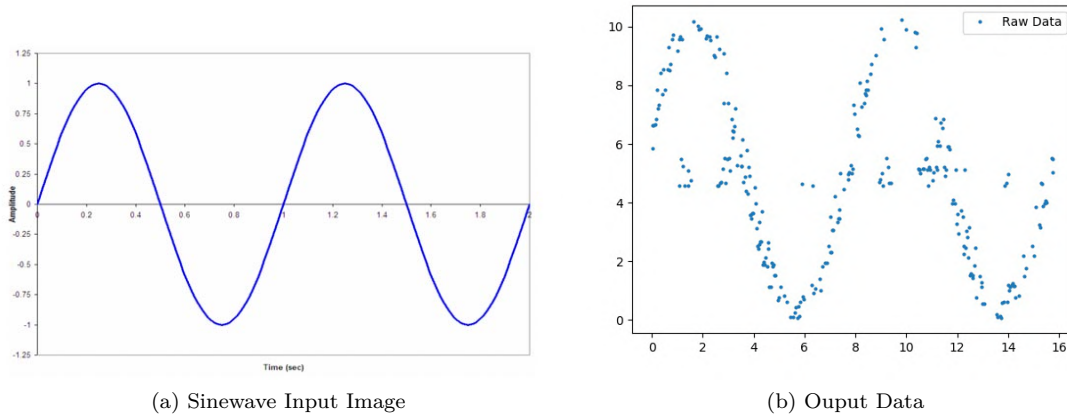
Figure 4.10: PlotDigitzier Adaption Sinewave Input and Output Example

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. Various methods are used for smoothing, the aim amongst all 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 each other. 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 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 is not being well suited to handle extreme outliers with data and hence it is also not the preferred method.

Gaussian Filter

Gaussian filters operate in the 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 its 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 triangular filter with the added benefit of being very effective at removing Gaussian noise. While there may be some Gaussian 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 the previous filter. For these reasons it also not the preferred method.

Savitzky-Golay Filter

Savitzky-Golay 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:

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

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 its ability to almost eliminate strong outliers. As filter testing will show, this method is preferred purely due to its impressive performance.

4.3.5 Filter Testing

All the filters are tested against a test base of 50 graphs, the same in the graph extraction tests. The graphs contain of each of the following function models:

- 1st to 3rd order Polynomials - (Linear, Quadratic and Cubic)
- Exponentials
- Logarithms
- Trigonometric 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 curve output from the graph data extraction section is shown in Figure 4.11. It is clear to see that the **Savitzky-Golay filter smoothed the data the best and in all cases for every function model, this was the same outcome**. It is better at rejecting 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. It can, however, be seen that for extremely noisy sections of the curve, in this case, the beginning, none of the filters can entirely smooth the curve or correct the data. This is to expected and is accepted as a drawback of imperfect data extraction from the graph image.
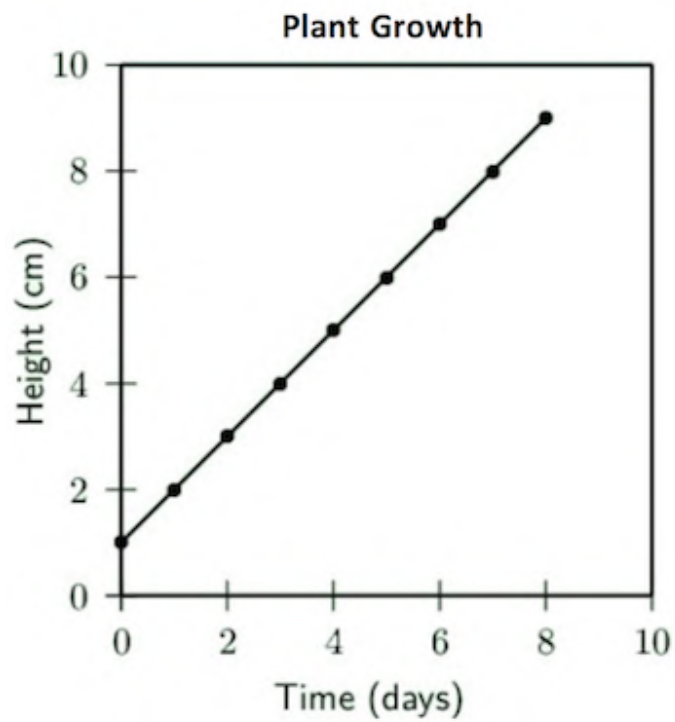
(a) Input Image
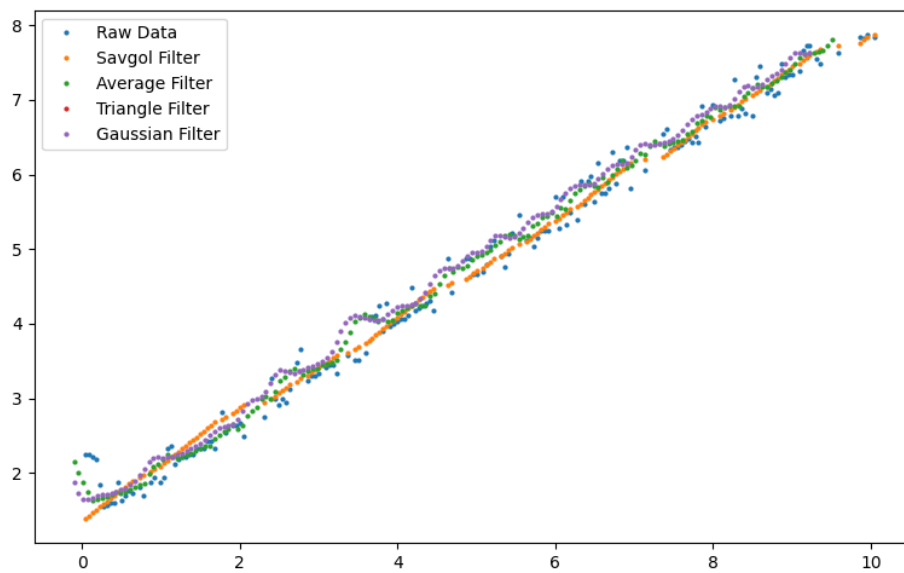
(b) Filter Outputs

Figure 4.11: Filter Testing Linear Curve Example

4.4 Graph Audio

In the research stage of this project, the prospects of active auditory interfaces were 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 converted to an identifiable audio signal. To do this, an audio synthesizer must be utilised, this will convert our audio parameters into a sound wave that can be 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. It must be noted that neither are audio players and the generated audio files must be played backed using a different package.

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 lightweight 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.

```
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 a 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. This will enable the generated audio files to be played through the speakers.

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 several factors 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 to a pixel-based axis. Therefore, the optimal relationship for a 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 at 60Hz.

While this relationship produced clear variations and minimised the harshness of the different 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 0.2 seconds by default. Two functions can be called, one that runs through every single data point and joins the samples to create a master audio file. each data point is faded into the next by 0.1 seconds to produce a smooth audio track. 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 the complete conversion and get a general idea.

4.4.4 Testing

This component was tested with 100 defined data sets that represented the following forms of model functions:

- 1st to 3rd order Polynomials - (Linear, Quadratic and Cubic)

- Exponentials

- Logarithms

- Trignometric functions - (Sine, Cosine and Tangent)

For each test, the entire data set was converted, these were then played back alongside a view of the original data to ensure that they had been converted properly to represent the model exactly. The successful conversions for each model are shown in Table 4.6.

	Linear	Quadratic	Cubic	Exponential
Successful Conversion	Yes	Yes	Yes	Yes
	Logarithm	**Sine**	**Cosine**	**Tangent**
Successful Conversion	Yes	Yes	Yes	Yes

Table 4.6: Graph to Audio Conversion Test Results

As can be seen, all model functions were successfully converted. Describing what the audio output sounds like can be done by looking at some examples. Figure 4.12a illustrates a parabola, the audio conversion of this graph would, therefore, sound like the volume would be rapidly decreasing as the frequency increases linearly and then rapidly increasing again after passing through the trough. Figure 4.12b illustrates a cosine wave, the audio conversion of this graph would, therefore, sound like the volume would be periodically decreasing and increasing with the peak and troughs of the curve. With the frequency of the graph increasing linearly as the pointer moves along the curve.
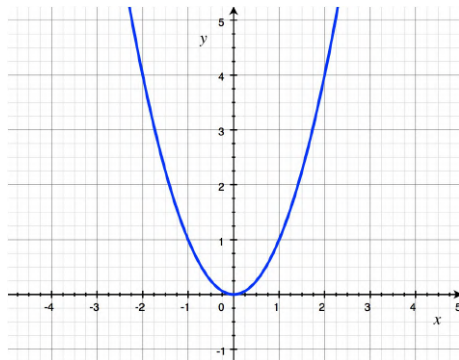
(a) Parabola Curve
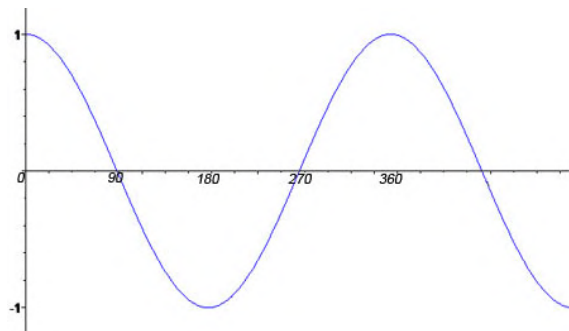
(b) Cosine Curve

Figure 4.12: Parabola/Cosine Curve to Audio Conversion Examples

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

- Trigonometric functions - (Sine, Cosine and Tangent)

This is a fantastic addition to 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 what the shape is. To achieve this, each of the above-mentioned models is first fitted 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 the example shown in Figure 4.13, a linear curve has been fitted against a noisy data set. The distance between each point and the line, 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 in essence part of how this measure is also calculated.
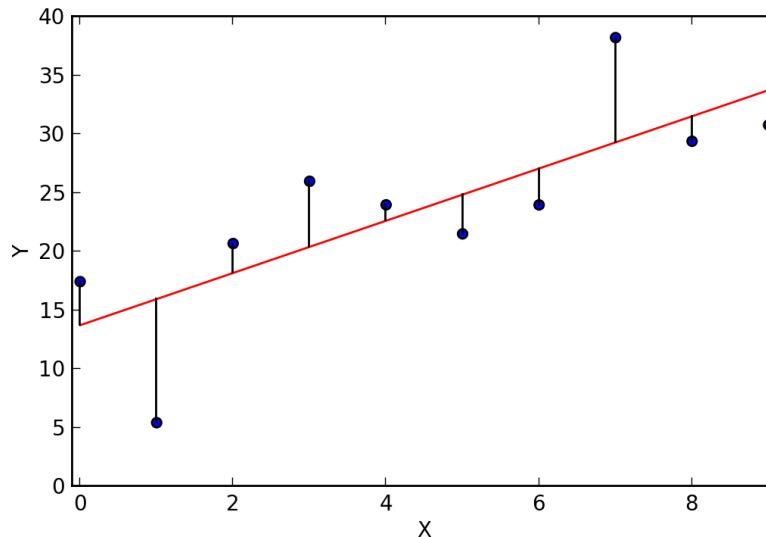
Figure 4.13: Residuals Example Graph

49

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 4.5 defines the mean of the set of data, 4.6 defines the sum of the total square. This measures the inherent variability within the data itself. 4.7 defines the sum of the total square error. This measures the variability between the data and the function that that is being fitted. 4.8 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 workaround for this is to add weights to each of the data points that will compensate. This is often known as the 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 seen in Listing 4.4.

```
# 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: Numpy - polyfit Implementation Example

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 covariance 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.

To define the fitting equations we pass them into Scipy as functions that return the result given the input data that represents the X values. These equation functions can be seen in Listing 4.5.

```python
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 curve_fit method. Whichever better fits the data will be the most accurate. Noting that curve_fit already as a theoretical advantage, this is a practical test to ensure that our assumptions are indeed correct. The $\hat{R2}$ value is used here to quantitatively measure which methods are more accurate.

Through rough testing, it became very clear as soon as the exponential function trialled, that the polyfit method would not work. In every case and trial, it failed to plot a curve that matched any of the data that was used. It is 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 caused 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 4.14. It is important to note that this is after forcing the function to limit the number of self calls to return the best attempt before failing.

It is clear to see that this is 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 curve_fit function faired much better in every single test case. Even with very noisy data, the function was able to try 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, these can be seen in Figure 4.15.
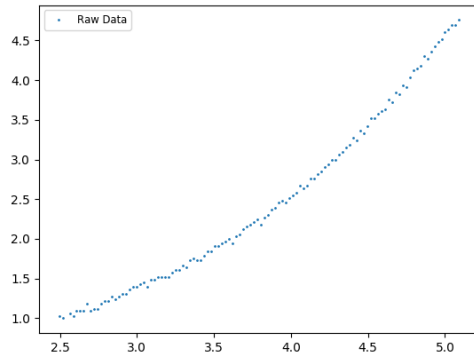
(a) Exponential Data
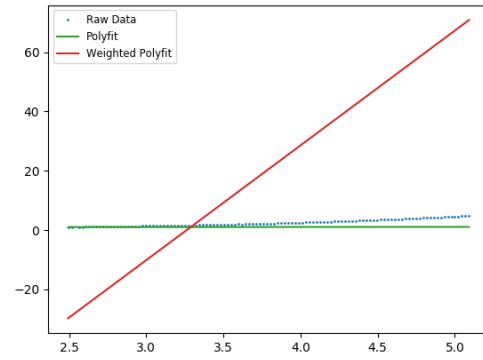
(b) Polyfit and Weighted Polyfit

Figure 4.14: Numply Polyfit Example - Limited iterations
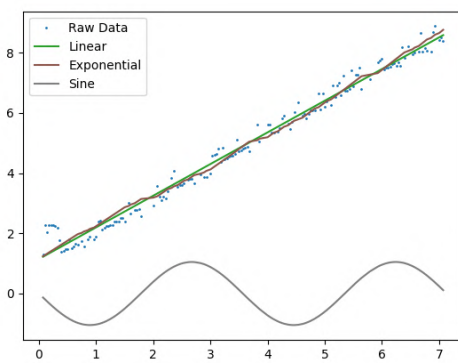
(a) Linear Data
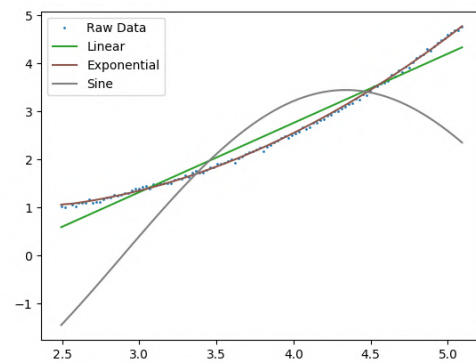
(b) Exponential Data

Figure 4.15: Scipy curve_fit - Results Example

A test base of 160 data sets, equally distributed to represent each model function, was tested with the curve_fit method. The success of each data set was determined by the correct identification of the function model. The results from those tests are shown in Table 4.7.

	Linear	Quadratic	Cubic	Exponential
Average Accuracy	84%	82%	100%	100%
	Logarithm	**Sine**	**Cosine**	**Tangent**
Average Accuracy	100%	100%	100%	100%

Table 4.7: Curve_fit Test Results

From the tests, 96% were correctly identified. The remaining 4% were all related to the second and first-order polynomials. In some cases, the script would return the wrong order of polynomial that it fit the data. This is inevitable as variations in the data even after smoothing can lead 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 fantastic result and concludes that the Scipy curve_fit 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 defined specific topics or categories, such as images of animals or fruit and vegetables. As we require textual descriptions for images surrounding STEM subjects, the boundaries are not easy to define and training a data-set large enough to produce adequate results 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.

4.6.1   Object Detection

One alternative method would be to try and detect objects within the image to create predefined descriptions around those objects. Google's Vision API comes with several pre-trained models for common image labels. While they do not cover labels and entities specific to science, it was still trialled to determine if it could provide any useful information based on the contents of the image. The implementation is very simple and is shown in Listing 4.6.

```python
def localize_objects(content):

    image = vision.types.Image(content=content)

    objects = client.object_localization(
        image=image).localized_object_annotations

    print('Number of objects found: {}'.format(len(objects)))
    for object_ in objects:
        print('\n{} (confidence: {})'.format(object_.name, object_.score))
```

Listing 4.6: Google Vision API Object Detection

Given the input image of a titration apparatus as seen in Figure 4.16a, the results received from this API were disappointing but as expected. Without any trained models specific to the objects and entities found in the types images present in STEM material, it was unable to even roughly identify the image. In this case, the API returned "Footwear" and "Luggage Handbags" as recognised objects with confidence levels of 0.5 and 0.7. It's apparent that this method cannot be used and although this has the potential to provide promising results if trained properly, this simply is not feasible.

(a) Input Titration Image   (b) Google Reverse Images Result
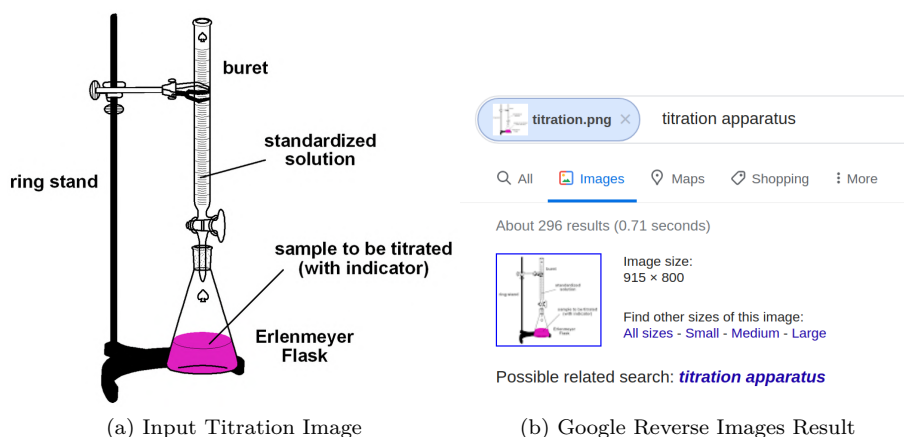
Figure 4.16: Image Descriptions Example

54

4.6.2 Google Images Reverse Search

The approach chosen to tackle this problem is based on utilising the largest database of images that could be used to draw image comparisons. This would, of course, be Google Images which has a database of over 10 billion images. The key feature with Google Images that can be used 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 the titration apparatus image can be seen in Figure 4.16b.

As can be seen, 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, however, give some general information about the image. In the background research for this project, Google's "Get Image Descriptions" was discussed, unfortunately this cannot be used as this is a browser integrated feature which cannot be adapted to use in this project. To implement the reverse search method into this system, a web scrapper must be used to send a request to Google Images and extract from the resultant URL the information description. This can be achieved by using an HTML parser of the website to search for the class ID that contains the information needed.

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 a 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 scrape the information desired from the website. Put together, this can be implemented as seen in Listing 4.7.

```python
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 4.7: Google Images Web Scrapper

It must be noted that this method is only 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 may happen however is that the HTML structure could change over time. If this is the case, then adjustments must be made to the class iterator to find the correct location within the HTML code to retrieve the information.

4.6.3 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 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 and their level of detail on a scale of one to ten, intermediate score meanings are shown in Table 4.8.

	Match Scoring	Detail Scoring
0	Description does not match image in anyway	Description provides no detail
5	Description matches the general topic of the image	Description provides broad detail
10	Description matches the specific topic of the image	Description provides specific detail

Table 4.8: Image Descriptions Results Scaling Key

Average Match Score	Average Detail Score	Number of Null Descriptions
8	3	0

Table 4.9: Image Description Results

Of the 50 images that were tested, there were no images that returned a null or invalid description. On average the match results scored a n eight out of ten and a three out of ten for detail. Although the results were quite accurate at identifying the contents of the image, the detail level is quite low. Being the only available approach that is feasible, the results are still promising for this prototype and will provide at least some additional information to the user that they would otherwise not be able to receive. As such, 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:

- 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 how they breakdown the text they extract. Google cloud vision will be used to test cloud-based solutions, for its superior output structure and its low cost compared with other cloud services. The first 1000 requests per month are free and thereafter 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 the 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 can also 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 threefold; language, layout analysis and engine mode. Calling the correct language is as simple as defining it from the list of trained language sets. By default, Tesseract will attempt to recognise English but there are 125 to choose from. 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 4.8: 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 [34]. The major differences in performance are in the form 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.

```python
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 4.9: 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 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 4.10 to extract each section into an array:

```python
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 4.10: 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 is measured as the number of correct characters per page

- The processing time - This involves comparing the time it takes for each method to identify and extract same piece of text from the document

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 4.10 and an example of the words detected by each method are shown in Figure 4.17.

	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 4.10: Text Extraction Test Results

4

The Principles of Science and
Interpreting Scientific Data

Scientific method refers to the body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge. It is based on gathering observable, empirical and measurable evidence subject to specific principles of reasoning.

Isaac Newton [1687, 1713, 1726]
"Rules for the study of natural philosophy,"
Philosophiae Naturalis Principia Mathematica

Forensic science actually is a broad array of disciplines, as will be seen in the next chapter. Each has its own methods and practices, as well as its strengths and weaknesses. In particular, each varies in its level of scientific development and in the degree to which it follows the principles of scientific investigation. Adherence to scientific principles is important for concrete reasons: they enable the reliable inference of knowledge from uncertain information—exactly the challenge faced by forensic scientists. Thus, the reliability of forensic science methods is greatly enhanced when those principles are followed. As Chapter 6 observes, the law's admission of and reliance on forensic evidence in criminal trials depends critically on (1) the extent to which a forensic science discipline is founded on a reliable scientific methodology, leading to accurate analyses of evidence and proper reports of findings and (2) the extent to which practitioners in those forensic science disciplines that rely on human interpretation adopt procedures and performance standards that guard against bias and error. This chapter discusses the ways in which science more generally addresses those goals.

III

(a) Tesseract Text Detection   (b) Cloud Vision Text Detection

Figure 4.17: OCR Detected Words Example Contour Output

As can be seen from the results, the Cloud Vision API has a large efficiency advantage as it can take advantage of powerful 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 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 the example shown in Figure 4.17. Tesseract was able to identify all the words on the page except the chapter number '4' which was not recognised as an entity. Vision API did recognise the number and all the words on the page but it was not able to identify them. In this case, it completely missed out the first paragraph. It is this behaviour that leads to it's lower performance and is the reason why Tesseract was chosen for this application; it's superior accuracy outweighs the inefficiency. The extracted text from the example shown in Figure 4.17, can be seen in Appendix 8.4.

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 [35]. 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 on 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 therefore set fairly low at 0.48 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 rare cases, after an error was rectified and the text was passed through again to search for more errors, the API would return a non-existent correction. It would always be located one character ahead in the string from the previous error and would occur indefinitely unless it was ignored. Unfortunately at the time of writing this report, there are no clear explanations as to why this occurs or any fixes. Various attempts were made by resetting the API input and multiple full and segmented

passes with no avail. The only viable solution as mentioned, was to ignore these false corrections. Fortunately these circumstances were quite rare, passing these sentences with whatever few errors they may have is deemed acceptable as the accuracy from Tesseract itself is already quite high.

```python
def correct(text):

    corrected = []

    for part in text:
        part = re.sub(' {2,}', ' ', part)
        part = re.sub(r'\(.*?\)', lambda x: ''.join(x.group(0).split()), part)
        part = re.sub(r'\s([.,!?:](?:\s|$))', r'\1', part)
        check = api.check(part, api_url='https://languagetool.org/api/v2/', lang='en-GB')

        while len(check.get('matches')) < 0:

            wrn = check.get('warnings')
            err = check.get('matches')[0]['message']
            conf = check.get('language').get('checkectedLanguage').get('confidence')

            if conf >= 0.48:

                chg = check.get('matches')[0]['replacements'][0]['value']
                offset = check.get('matches')[0]['context']['offset']
                length = check.get('matches')[0]['context']['length']

                if chg == "":
                    break
                else:
                    p1 = part[ : offset]
                    p2 = part[offset + length : ]
                    part = p1 + chg + p2

                check = api.check(part, api_url='https://languagetool.org/api/v2/',
                    lang='en-GB')

        corrected.append(part)

    return corrected
```

Listing 4.11: PyLanguageTool Auto-correct Implementation

This autocorrect function was evaluated with the output from the OCR tests, the extracted text was corrected and matched with the original text to see how closely the text now matched the original. The results are summarised in Table 4.11 and an example of the correction input and output text are shown below the table.

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

Table 4.11: PyLanguageTool Auto-correct Test Results

Auto-correct PyLanguageTool Input Example:

In mammals the heart at double pump. The right side pumps deoxgenated blood to the lungs and the left side pumps oxygenated blood to the rest of the body tissues. This means that an red blood celll will travel through the heart twice during a complete circulation around the body.

Auto-correct PyLanguageTool Output Example:

In mammals the heart is a double pump. The right side pumps deoxygenated blood to the lungs and the left side pumps oxygenated blood to the rest of the body tissues. This means that a red blood cell will travel through the heart twice during a complete circulation around the body.

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 will increase the fluidity of the text-to-speech output.

4.8 NLP

Natural Language Processing has been pivotal in recent years for many software technologies that are now being taken for granted, such as auto-correct, email spam filters and software assistants such as Alexa. In this project, it is been utilised to extract more detail surrounding the information with the page by looking for interesting content and entities and forming summaries for the student.

4.8.1 Analysis

There are four key branches of NLP analysis, sentimental, syntactic, entity and content. Sentimental analysis aims at determining the prevailing emotion that is present within a piece of text. On a scale of 0 to 1 negative, neutral and positive sentiment can be defined within predefined ranges. For this project the, knowing the emotion of the information is not particularly useful or important. STEM subjects in general are not written to illustrates these emotions and hence this type of analysis is omitted.

Syntactic analysis creates tokens from sentences and words within the text and looks specifically at the syntax of each token. It provides linguistic information such as grammatical genders and person, for example, if particular words are referring to the male, female or neutral genders or the text is written in the 1st, 2nd or 3rd person. Again for the type of information that the students using this system would study, this is once again not particularly useful and is omitted.

Entity and content analysis are very similar, entity looks at identifying nouns and defining what they are in relation to. For example, proper nouns could be referring to famous landmarks of figures and common nouns could be referring to entities such as restaurants or stores. Content analysis acts more generally, trying to classify what whole chunks of text are about. The classifications must be predefined but can include topics such as healthcare and finance. This is the most relevant to this project and is what will be tested to see if it can provide any useful information.

In order to test this method, the Google Natural Language API is used. There are several cloud-based API that can be used but they all perform very similarly and as Google's other API's have already been used in other components of this system, it was logical to use a service that was under the same company. The implementation for both entity and content analysis are very simple and can be seen in Listing 4.12.

```python
def content_analysis(text_content):

    client = language_v1.LanguageServiceClient()

    # Define the input text type
    type_ = enums.Document.Type.PLAIN_TEXT

    # Define language and configuration
    language = "en"
    document = {"content": text_content, "type": type_, "language": language}

    # Send the API request
    response = client.classify_text(document)

    # Loop through classified contents returned from the API
    for category in response.categories:
        print(u"Category name: {}".format(category.name))
        print(u"Confidence: {}".format(category.confidence))

    return response
```

```python
def sample_analyze_entities(text_content):

    client = language_v1.LanguageServiceClient()

    # Define the input text type
    type_ = enums.Document.Type.PLAIN_TEXT

    # Define language and configuration
    language = "en"
    document = {"content": text_content, "type": type_, "language": language}
    encoding_type = enums.EncodingType.UTF8

    # Send the API request
    response = client.analyze_entities(document, encoding_type=encoding_type)

    # Loop through entitites returned from the API
    for entity in response.entities:
        print(u"Representative name for the entity: {}".format(entity.name))
        # Get entity type
        print(u"Entity type: {}".format(enums.Entity.Type(entity.type).name))
        # Get the salience score associated with the entity
        print(u"Salience score: {}".format(entity.salience))
        for metadata_name, metadata_value in entity.metadata.items():
            print(u"{}: {}".format(metadata_name, metadata_value))

        # Similar mentions within websites
        for mention in entity.mentions:
            print(u"Mention text: {}".format(mention.text.content))
            print(
                u"Mention type: {}".format(enums.EntityMention.Type(mention.type).name)
            )

    return response
```

Listing 4.12: NLP - Entity and Content Analysis Implementation

4.8.2 Testing

A total of 50 pieces of relevant text were passed through both functions and the outputs recorded. For content analysis, the topic with the largest confidence level was recorded. For entity analysis, the top five results with the highest salience levels were recorded. The accuracy was then determined based on whether or not the results correctly identified the topic of the text. The output was also rated on a scale of zero to ten based on how useful the information was. The usefulness is determined based on what additional information can be extracted from the text that isn't already present in the material. The scaling can be seen in Table 4.12, the results can be seen in Table 4.13 and an example of the type of text used as the input is shown after that.

	Usefulness Scoring
0	Analysis provides no useful information
5	Analysis provides some useful information
10	Analysis provides significant useful information

Table 4.12: NLP Analysis Scaling

	Average Topic Accuracy	Usefulness
Entity Analysis	24%	1
Content Analysis	89%	3

Table 4.13: Entity and Content Analysis Test Results

NLP Analysis Input Example:

Optical isomers are asymmetric molecules that are nonsuperimposable mirror images of each other. They can be described as chiral molecules or enantiomers. Optical isomerism occurs in substances in which four different substituent groups are arranged around a central carbon atom called the chiral centre. Optical isomers, in general, have identical physical and chemical properties, except when they are in a chiral environment. However, they have an opposite and equal effect on the direction of rotation of plane-polarised light, and are therefore said to be optically active. Racemic mixtures contain equal amounts of both enantiomers, and are optically inactive. In biological systems, only one optical isomer of each asymmetric organic compound is usually present [36].

As can be seen from the results, entity analysis did not perform very well. It was very rarely able to identify the topic of the text and because of this, it either classified the entities as "OTHER" or "COMMON". This provides no information at all surrounding the text and for this reason alone it will not be used.

Content analysis performed much better, it was able to correctly identify the topic of the majority of input texts but unfortunately did not provide any additional useful information. In the majority of cases, it was only able to provide general topic titles rather than specific classifications. In the example input text shown above, the output from the content analysis was "/Science/Chemistry" with a confidence level of 0.96. Although it may not be very useful, it can be used to briefly introduce the topic of the page the student is studying before they access the information within it to provide some material awareness.

4.8.3   Summarization

Automated text summarization is a form of natural language processing that can be divided into two major categories, extractive and abstractive summarization. Extractive summarization involves using sentences within the original piece of text to form a shorter summary. Abstractive summarisation typically used machine and deep learning techniques to form entirely new sentences for the summary [37]. The issue with this method is the need for a data-set. As STEM covers such a wide array of subjects, this once again isn't quite feasible.

For the purposes of this project, extractive summarization will be used to provide a general description of the page being studied by the student. Paired with the brief introduction that can be formed from content analysis, this should provide the student with an alternative method of quickly studying the page in question, without having to go through everything.

The most effective implementation method is known as TextRank, this is based on the same algorithm used in Google's search, called PageRank. It works by breaking down the text into sentences and forming vectors based on the words embedded within each one. A matrix is then formed based on the similarities between these sentences and a graph is formed. From the graph, sentences are ranked and the top entries are used to form the summary [38].

The leading TextRank python package is by Gensim, almost all wrappers and alternative packages are either directly using the "gensim.summarization" library or as based on the same exact method with slightly different graph-based ranking. This can be implemented very easily as is shown in Listing 4.13.

```
from gensim.summarization.summarizer import summarize

def summ(text):

    # Summarization (Reduction based on the number of words)
    summ_w5 = summarize(text, word_count = 50)
    summ_w7 = summarize(text, word_count = 75)
    summ_w10 = summarize(text, word_count = 100)

    return summ_w5, summ_w7, summ_w10
```

Listing 4.13: Gensim Text Summarization Implementation

4.8.4 Testing

Three summarising outputs can be compared to determine which will yield a better result, they compress the text based on a word count threshold of 50, 75 and 100 words. This function was tested with the same 50 pieces of text that were used in the analysis testing in Section 4.8.2. The results were based on the quality of the summary on a scale of zero to ten. The higher the rating, the more the summary was able to capture **only** important information and not anything unnecessary.

	Effectiveness Scoring
0	Summary includes no key points
5	Summary includes some key points
10	Summary includes all key points

Table 4.14: Summarization Effectiveness Scaling

	50 Words	100 Words	150 Words
Effective Summarization	6	7	4

Table 4.15: Gensim TextRank Test Results

From the results it's clear to see that, both 50 and 100 words performed admirably. They provided a summary that included most of the relevant information within the pieces of text. Considering that most pages contain between 200 to 400 words, averaging 250, it's easy to see how 150 words started to include more irrelevant information. For this reason, a summary of threshold of 100 words was chosen for this system.

4.9 Equation Extraction

Equation extraction from images has seen a lot of development in recent years. Many applications and services are offering methods of converting handwritten equations from whiteboards, notebooks and even images. These all have trained OCR engines that are very good at identifying mathematical characters. For this application, the standard Tesseract Engine was compared against the leading Python-compatible Equation OCR engine Mathpix.

4.9.1 Tesseract

Extracting equations using Tesseract is as simple as using the same implementation process as seen in Section 4.7.1. The blocks extracted are fed into the function one by one and the contents are extracted. The disadvantage with this method is of course that, it has not been specifically trained to extract mathematical characters and it also cannot automatically detect which blocks have equations within them. This must be carried out separately by searching for operators within the output to identify the equations.

4.9.2 Mathpix

Mathpix is an online API service that offers 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 conversions 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 4.14.

```python
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 4.14: Mathpix Equation Extraction Implementation

As there are no other packages or services that support python and that can perform as effectively as this. The cost of purchasing a key for the 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.

4.9.3 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 limited in nature due to the manual conversion from latex to SSML to speech and considering the complexity of the equations often seen by the majority of primary and secondary school students. This will be explained further in the Text-to-Speech section. For each

of the above function types, 25 assorted documents with 60 simple to complex equations are used to test the speed and accuracy, in rejecting blocks that do not contain equations and correctly converting those that do. The results from the tests can be seen in Table 4.16 and they show the accuracy which is the number of correctly identified characters in the equation and the average processing time.

	AverageTime	Block Accuracy	Equation Accuracy
Tesseract	1.6 Seconds	87%	9%
Mathpix	0.9 Seconds	92%	98%

Table 4.16: Equation Extraction Test Results

As can be seen from the results, the Mathpix API 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 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. Tesseract on the other hand performed very poorly, unable to identify many characters, it introduced several errors and missed out large sections of equations. This was unfortunately worse than what was expected and when compared with Mathpix, is clearly written out. Mathpix was chosen as the preferred method.

An example of a Fourier Transform function that contains a sum and an exponential with superscripts and subscripts is shown in Figure 4.18 and the output from Mathpix can be seen in equation 4.9 and the output from Tesseract can be seen in 4.10. Converting this output into speech is handled and discussed in Section 4.12.4.

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

Figure 4.18: Mathpix Input Example

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

$$
\begin{aligned}
&N-1 \\
&DYfayePs \\
&= 0 \\
&flePS \\
&n0, 1, 2, 14, 15 \\
&15k \\
&.kF2k.14k. \\
&14 + eI2776 + e276 + e32016 + e320 \\
&16
\end{aligned}
\tag{4.10}
$$

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 all tables present within the page.

4.10.1 OpenCV and Tesseract

This method involves preprocessing the blocks by applying, in the same manner as was done for block extraction, 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 a 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 using 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 incredibly robust and accurate and therefore does not need the page to be broken down into blocks.

The standard Camelot python package only supports 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 4.15.

```python
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:
        print(check_usage(api_key))

    return
```

Listing 4.15: CamelotPro Table 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 were passed directly into the API. The results from the test can be seen in Table 4.17 and 4.18, they show the accuracy which is the number of correctly identified cells in the the table and the average processing time..

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

Table 4.17: 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 4.18: Table Extraction Average Time Test Results

The results from the test proved to be rather interesting, the CamelotPro 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 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 not effective and hence CamelotPro is used. Example outputs for the table in Figure 4.19 can be seen in Tables 4.20 and 4.19.

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 4.19: 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.

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 4.19: OpenCV and Tesseract Output

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 4.20: CamelotPro Output

4.11 Text-to-Speech

The primary form of communication for this system will be through the sense of sound. All of the information extracted from the document, other than the graph audio, will be in the form of text which can be converted to speech using Text-to-Speech software. There has been a lot of development in this space for solutions that offer the most natural and intelligible sounding conversions. Generally speaking, traditional solutions that do not employ machine learning are more unintelligible and less natural sounding. To this end, ESpeak a popular offline traditional TTS package was tested against a state-of-the-art cloud-based service, Google Text to Speech to test this assumption.

4.11.1 ESpeak

ESpeak is an open-source offline TTS package that uses 'formant synthesis', This method of synthesis uses the spectral shape of an acoustic resonance that is produced by the human vocal tract. This method does not use any form of machine or deep learning and is a purely heuristic way of generating speech.

The advantages of this are that it is entirely offline, it is incredibly lightweight and therefore very fast at performing conversions and produced clear sounding speech. It supports a range of languages but performance is not consistent between them all. Implementing this in Python can be done using the Pyttsxs3 package that is a wrapper for a range of TTS engines such as Espeak, NSSS and Sapi5. It offers basic configuration parameters such as speech rate, voice and volume. It also comes with a built-in audio player that can playback the speech without the need to save the audio file. Listing 4.16 illustrates how simple this can be implemented.

```python
def espeak(text)

    # Define the TTS engine to be used
    pyttsx3.init(driverName='espeak')
    engine = pyttsx3.init()

    # Set engine configuration parameters
    engine.setProperty('voice', 'english')
    engine.setProperty('rate', 120)

    # Convert text to speech and playback
    engine.say(text)
    engine.runAndWait()

    return
```

Listing 4.16: Pyttsxs3 Implementation with ESpeak

4.11.2 Google Text to Speech

Google Text to Speech has been thoroughly developed using modern machine and deep learning techniques to generate larger, more realistic human-sounding synthesisers. It has the advantage of tapping into its incredibly large resource for test data and as a result, their TTS engine is one of the best that services that exist today. It supports more than 30 languages and 180 voices but more importantly, it offers access to DeepMinds fantastic Wavenet technology that has generated some of the most natural sounding voices of any engine.

Implementing this in python is relatively simple, similar to Pyttsxs3, the configuration parameters are first set. These include the language code, voice, and encoding format for saving the audio file. This does not come with a built-in audio player, hence, Pygame is once again used to playback the audio files that are produced. It can be played by writing the output to temporary BytesIO files or WAV files.

```
def g_tts(text, filename):

    # Configure the clinet configuration parameters
    language_code = 'en-GB'
    text_input = types.SynthesisInput(text=text)
    voice_params = types.VoiceSelectionParams(
        language_code=language_code,
        name='Wavenet-A')
    audio_config = types.AudioConfig(
        audio_encoding=enums.AudioEncoding.LINEAR16)


    # Instantiate the Cloud Client
    client = texttospeech.TextToSpeechClient()
    response = client.synthesize_speech(text_input, voice_params, audio_config)


    # Temporarily save the output to an Wav file
    with open(filename, 'wb') as out:
        out.write(response.audio_content)

    return
```

Listing 4.17: Google Text to Speech Implementation

4.11.3 Testing

Testing these two methods was very simple, given a range of input text the output from each engine can be played back to compare two key properties. The naturalness and the intelligibility of the speech. The tests involved using 100 sentences of length 10-20 words that contained a range of topics surrounding STEM subjects. This was done to test how the system would sound concerning the content that would be studied by the students. The properties were rated on a scale of zero to ten and their intermediate meanings can be seen in Table 4.21. The Google Text to Speech method was run with both the standard and Wavenet voices and the results can be seen in Table 4.22.

	Naturalness Scoring	Intelligibility Scoring
0	Speech does not sound natural	Speech is not intelligible
5	Speech sounds partly natural	Speech is partly intelligible
10	Speech is as natural as humans	Speech is entirely intelligible

Table 4.21: Text-to-Speech Naturalness and Intelligibility Scale

	Successful Conversion Rate	Naturalness	Intelligibility
ESpeak	100%	2	5
Google Text To Speech (Standard)	100%	7	8
Google Text To Speech (Wavenet)	100%	8	8

Table 4.22: Text-to-Speech Test Results

ESpeak produced 'electronic' sounding speech, there was little variation in the tone and although the sound was clear it was not always intelligible. Often there were sharp variations in the volume when pronouncing words and this made it sound even more unnatural. Despite this and all methods having perfect conversion rates, as seen from the results, Espeak performed the poorest.

The Google TTS engine was tested with the standard and Wavenet voices. Both were very natural sounding and very intelligible. There were variations in tones for capital letters and appropriate pauses in and around parenthesis. The Wavenet voice was slightly more natural sounding, mostly due to the smoothness between the synthesis of different words. This was reflected in the results and therefore became the preferred method and voice for all text-to-speech conversions.

4.11.4 SSML

Speech Synthesis Markup Language or SSML is a markup language based on XML used for interactive speech synthesis. Allows bodies of text to be marked to perform certain operations such as additional pauses between sentences and phrases or enforcing a certain variation in how text is pronounced such as numbers; 'one' or 'first'. It is incredibly useful in this application for converting the extracted equations to speech.

Two main issues need to be tackled, the first is that the equation is converted into a Latex format. This means the equations will contain brackets and Latex references such as 'sum' of 'omega'. The second issue is that not all mathematical operators are recognised by Google Text to Speech. These need to be translated into a text format that can be converted to speech. As mentioned prior, the list identifies the standard set of equations that this system will support.

- Additon, Subtraction, Multiplication and Division

- Polynomial Functions

- Exponential and Logarithmic Functions

- Trignometric Functions

- Integrals and Differentials

Passing a simple equation such as {a x (b + 49) = [x + y]} to test how Google TTS handles each type of bracket reveals that it does not hinder or alter the way it is converted. This is a positive result as this means that the brackets do not need to be removed or replaced. For the remaining operators, they can be replaced by formatting the latex equation and replacing them with SSML elements.

The formatting first begins by replacing special characters such as '<' to an alternative such as '<−'. This is necessary as SSML is based on XML and requires characters such '<' to function correctly. This formatting step is perfectly acceptable as Latex does not contain any specific operators that refer to functions. It's the only definition is less than or greater than. The equation must also be formatted to include the '<speak>' and '</speak>' start, stop markers.

The second step is passing the text through a filter that replaces all the operators with an appropriate textual description that TTS will be able to convert. For example, equation 4.11 is represented in latex as is asigned to the variable, 'text', in Listing 4.6. This would ordinarily be translated into speech as "int x carrot dx". This is of course, incorrect, by running the function as seen in Listing 4.16, this would be correctly converted to "the integral of x to the power of 2 dx".

$$\int x^2 dx \tag{4.11}$$

```
def latex_to_ssml(inputfile):

    text = '{ \int x^2 dx }'

    # Replace confusing characters such as < with <-- to prevent errors
    escaped_lines = html.escape(text)

    # Convert Latex to SSML
    convert = convert.replace('\int', ' the integral of ')
    convert = convert.replace('^', ' to the power of <break time="0.5s"/> ')

    ssml = '<speak>{}</speak>'.format(convert)

    return ssml
```

Listing 4.18: Google Text to Speech with SSML for equations Implementation Example

The reason for using SSML rather than simply replacing all the operators and characters is that this offers greater flexibility when converting more complex functions. Adding additional pauses or varying pronunciation can aid in producing a more natural-sounding conversion from the equation to speech. This can be seen in Listing 4.18, with the insertion of a pause which does indeed produce a more natural-sounding conversion. A full list of the replacements that have been incorporated can be seen in Appendix 8.3.

The same data set that was used for testing the Tesseract and Mathpix equation extraction in Section 4.9.3, was used here. A total of 25 documents with 60 assorted simple to complex equations that had been converted into Latex were converted into SSML and converted from Text to Speech using SpeechRecognition. The successful conversion rate and naturalness of the audio, rated on a scale from zero to ten just as before, were recorded in Table 4.23.

	Successful Conversion Rate	Naturalness
Latex Equation to SSML to Speech	100%	8

Table 4.23: SSML Test Results

As can be seen from the results, the conversion rate was perfect and although the naturalness does not appear to have changed, it did increase marginally. The major advantage with this, is that as text-to-speech engines get better and better, the SSML conversion can be adjusted to continually improve the speech along with that.

4.12   Controls

Arguably the most important aspects 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 and instructions 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 a 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.

4.12.1   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 4.21.

```
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 4.19: 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, the button hardware can often register a single press or movement as multiple events.

4.12.2 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 errors produced in the xpad method with multipe event triggers . xboxdrv also supports builtin callback functions which will be essential when integrating controls within the system for each 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, vibration, LED light support and shoulder buttons; at the time of writing this report, this is missing from the xpad driver.

Implementing callback functions is as simple as starting the driver and setting the callbacks wherever needed, an example is shown in Listing 4.22.

```python
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 4.20: xboxdrv and Pygame Implementation

As this method provides shoulder button support and prevented the erros found in the xpad driver, this was chosen to be used in the system. Initial testing ensured that this method produced the correct output for every single button, trigger and joystick position. Further testing with the whole system is carried out in Section 5.3.

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.

Two main methods can be used to implement this within Python, the first is using the SpeechRecognition package that acts as a wrapper for several 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)

As many of the other services used in this project have been by Google, it was logical to try and keep as much of any additional services within the Google environment. Furthermore, as offline speech recognition software is renowned for being less accurate than cloud-based services and requires more local computational resources. These methods were not used. For example, the CMU Sphinx method has an error rate of approximately 15%, whereas, Google Cloud Speech API has an error rate of approximately 5% which is significantly lower and borders with the Human accuracy threshold. For this reason, offline methods were not used and the two Google services were tested.

The main difference between the two services is their revisions. Google Speech Recognition is a beta version of the latest Google Cloud Speech API. Both are free to use with the Cloud API charging only if the request contains an audio clip that is more than 60 minutes. For this project, both are essentially free and cost the user nothing.

As the performances of both are known to be almost identical, the implementation methods are being tested to determine which will better suit this system.

4.13.1   SpeechRecognition

As this system will be using a microphone to record what the user is saying, SpeechRecognition requires that Pyaudio is used for this. This is another Python package that makes available built in or external microphones to be used as input devices. Once this has been installed, implementing this method is as simple as initialising the the recogniser and, adjusting for ambient noise and passing the microphone feed. The function will then return, translated blocks of audio. This is particularly useful as a string of words can be said for several seconds and the entire stream is translated as if it were a single audio file. Listing 4.19 shows how this is implemented.

```python
def understand()

    # Initialise the recogniser
    r = sr.Recognizer()

    try:
        with sr.Microphone() as source2:

            # Adjust for background noise
                r.adjust_for_ambient_noise(source2, duration=0.2)

                # Listen
                audio2 = r.listen(source2)

                # Pass to Google Speech Recongition
                Text = r.recognize_google(audio2)
                Text = Text.lower()

                print(Text)

    except sr.RequestError as e:
        print("Could not request results; {0}".format(e))

    except sr.UnknownValueError:
        print("unknown error occured")
    return
```

Listing 4.21: SpeechRecognition Implementation with Google Speech Recognition

4.13.2   Google Cloud Speech API

Google Cloud Speech API is more complicated to implement as there are no wrappers but the process is much the same. The first step is to create a microphone class that collects audio chunks from the microphone stream and places them into a buffer that is then used as the input. As this API works to translate in real-time each word rather than a sentence as in the SpeechRecongition Method, the buffer is set up to always contain a chunk of data so that the cloud client does not stop until it is flagged to do so.

The recognition must now be configured, this involves setting the sampling rate, encoder, language and an interesting option called 'interim_results'. This returns results of the words being converted as they are being recognised and the final result is stringed together. With the configuration complete, the API can be called to start converting the microphone stream into text.

As the script for this is quite long, a short snippet of the implementation, where the actual API is called can be seen in Listing 4.20.

```python
def listen_print_loop(responses):

    for response in responses:
        if not response.results:
            continue

        result = response.results[0]
        if not result.alternatives:
            continue

        # Get the top result for the conversion
        transcript = result.alternatives[0].transcript

        if not result.is_final:
            sys.stdout.write(transcript + overwrite_chars + '\r')
            sys.stdout.flush()

        elif leave == True:
            break

    return transcript
```

Listing 4.22: Google Cloud Speech API Implementation

4.13.3 Testing

The two methods were tested against 50 predefined scripts that represented the type of notes that would be written by a student studying STEM subjects. The average length was 40 words and a portion of the tests also included basic mathematical operators such as addition and multiplication. The accuracy, based on the number of correctly identified characters and average times were recorded for each and presented in Table 4.24 and an example of the type of notes stored is shown below that.

	Average Accuracy	Average Processing Time
SpeechRecognition	98%	2.65
Google Cloud Speech API	98%	2.73

Table 4.24: Speech-to-Text Test Results

Note Input Example:

> *Try to remember that the arteries carry blood away from the heart whereas veins carry blood towards the heart.*

The results showed, identical performance between the two methods. Each performed with 98% accuracy through all tests and both took roughly the same average time of approximately 2.7 seconds after the user finished speaking. The cases that did not reach 100% either identified incorrect words or missed out words but did not result in a saved not that no longer made sense. For this reason, with this accuracy rate, the errors were accepted and the solution to prevent this is to advise the user to speak loud and clearly. As a result, this means that both Google text-to-speech services operate almost identically as expected. The determining factor then came down to implementation, as the SpeechRecognition method is much simpler, this was the chosen method alongside the Google Speech Recognition engine which does not require cloud Key, further making this method even easier to integrate.

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 on two criteria. The first is compatibility and the second is the 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 its wide compatibility with python packages that process images. OpenCV, Mathpix and CamelotPro all make 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 compresses and removes a lot of the information from images to reduce the file size, it was found through testing that this did not pose much of an issue and the incompatibility and errors produced from using formats such as PNG and GIF made it the most viable option.

The second is WAV files for audio. This is chosen simply because of its widespread adoption in almost all audio synthesis programs and it's compatibility with Pygame. There are known issues with not all MPEG files being recognisable by Pygame, the WAV format however works flawlessly. It can be easily be recognised and played through almost every single audio player across a variety of systems and has. The Graph audio script in this system generates temporary files in the WAV format, using the Pydub "audiosegment" function.

The third is CSV files for graph data and table data storage. This format was once again chosen for its compatibility with Python through the CSV library and also for its 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, settings and API keys. This format was used for its 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 is 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 4.23.

```json
{
    "page": [
        {
            "bookmarked": "",
            "topic" : "",
            "description": "",
            "notes": [
            ],
            "image_results": [

            ],
            "graph_results": [

            ],
            "equations": [

            ],
            "text": [
                {
                    "full_text": "",
                    "sentences": [
                        {
                            "text": "",
                            "highlighted": ""
                        }
                    ]
                }
            ],
            "misc": [

            ]
        }
    ],
    "settings" : [
        {
            "window" : "",
            "gender" : "",
            "speaking_rate" : "",
            "camera" : ""
        }

    ],
    "keys": [
        {
            "mathpix_id": "",
            "mathpix_key": "",
            "camelotpro": ""
        }
    ]
}
```

Listing 4.23: Data Storage 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 problems is still unclear but placing the scripts in one location and using threads and processes fixed all of these issues. This directory also contains a sub-directory for all the UI navigation audio files.
The 'library' directory stores the audio, 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 images. When each new document is scanned, the temporary files are wiped after they have been stored by chosen by the user. Figure 4.20 illustrates what the file structure looks like.
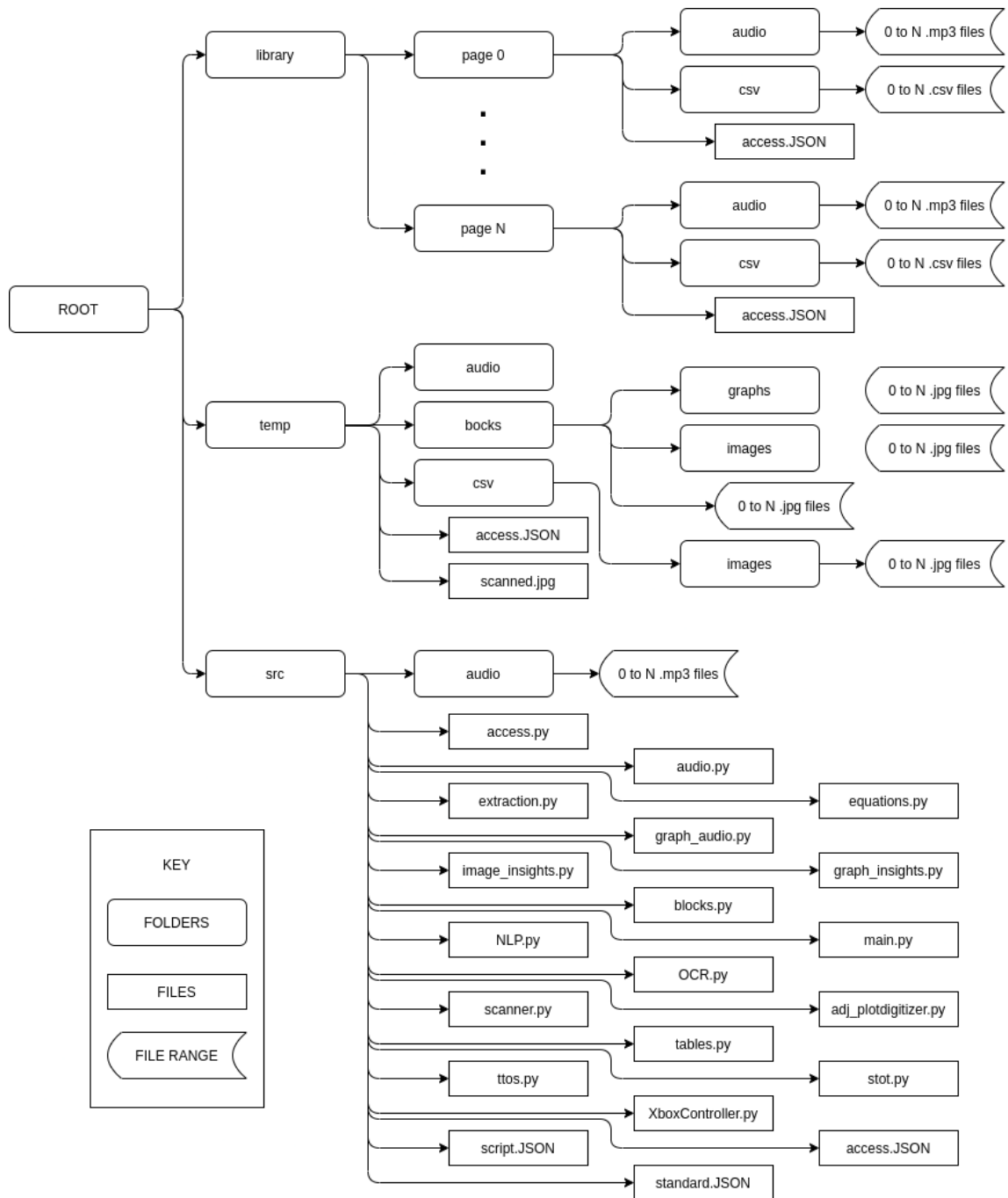
Figure 4.20: File Structure

4.15 System Integration

With each of the individual components, developed, implemented and tested, the final task is to integrate all of these components into one complete system. There is one main hardware adjustment that must be stated. As this will be run on a PC and not a Raspberry Pi as was originally intended. Starting the system is not taken into consideration in this project. The program will be run by manually executing the main python file and from then on, the system should operate as expected. If the system was to be run on a standalone device, it would be set up to automatically boot up and run the main script at launch to start the program. A full system overview can be be seen in Figure 4.21.

For each component in the system, specific call back functions are determined to provide unique controls that best suit the necessary actions. At points in the system where the user can input information, the controller can be used to activate the microphone that will listen and take in what they are saying to be processed. At all points in the system, predefined scripts are run to provide instructions and information on the available options and controls. The full list of the scripts written, can be seen in Appendix 8.2.
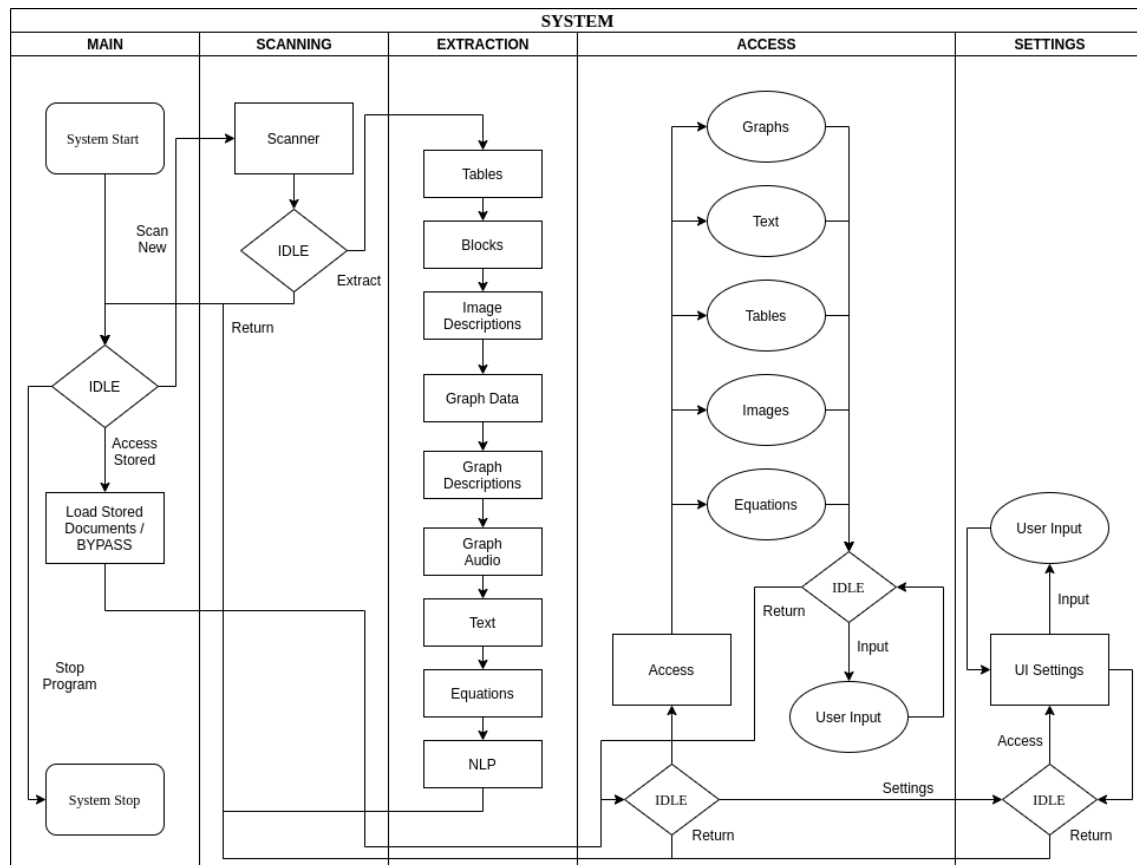
Figure 4.21: System Overview

4.15.1 Main Menu

The system can be broken down into five key segments, the main menu, scanning, processing, accessing and adjusting settings. Once the system starts up, the main menu is presented to the user. A script is used to introduce the user to the system, to communicate the controls and to wait for the user to decide what action to take. The user can then decide using the controller to scan a new document or to load existing documents that they wish to study again. They may also choose to shut the program down from this menu. This is a particularly important navigation point as all the proceeding segments will link back to this main menu either after they have run their functions or when the user decides to scan or access more documents or stop entirely.

4.15.2 Scanning

If the user decides to scan a new document, they are once again instructed to hold the device/camera above the page that is to be scanned and to hold steady while the page is detected. As the script begins to detect the page, further instructions are given to the user, to move closer to the page if necessary or to readjust the camera if the page has not been detected after a certain period. The system will then ask the user if they wish to extract the information from the page that was detected. They can also decide to return to the main menu, this will enable them to re-scan the page or to load stored pages.

4.15.3 Extraction

Should the user decide to extract the information from the page, the system will begin to obtain all tables, graphs, images, equations and text. It will also process this information and store it, ready to be accessed by the user. While this is running, the system will give halfway and finished updates to the user. Upon completion, the system will return to the main menu for the user to decide once again if they wish to store the page they scanned and extracted or to start a new scan.

4.15.4 Accessing Information

Every time the user wishes to access information, it will either permanently store the page that was just scanned or filter through the library and allow the user to decide what they would like to access. This is done by voicing the page identities and bookmark flag set by the user, or in the case of a new scan, the user is asked to set that ID using speech-to-text and to decide if they wish to bookmark the page. Accessing the information is as simple as the user defining what they wish to access and for some items how. Using the controller they can decide between pages and then between each type of item. Each item is then presented in a particular format that best communicates that information to the user.

Text

As seen in section 4.14, the text extracted from the page is broken down into sections and sentences which contain the highlight flag. When the user accesses this information they are presented with four options, the first is highlighted information. This will enable the use of the DPad to run through all the highlighted sentences within the page. The second is accessing sections of the page and the third is accessing sentences in each sentence. This provides a lot of flexibility, in what and how much information is presented to the user. They can also highlight or un-highlight sentences using the controller as they are studying the material. They will also be presented with the option to hear the summary of the page that has been generated.

Graphs

There are 3 types of information that the user can access related to the graphs, these are a full audio conversion, a by data point conversion and the graph description. The graph description is simply voiced to the user. The full conversion is also a simple playback of audio file that was generated. The by point conversion gives the user the ability to study the graph by listening to each

data point individually. Using the DPad, they can move from point to point which is converted and played back to the user. Note that each data point is based on window size, this ensures a greater variation between points than would normally be detectable between each original points. This window size can be changed by the user.

Tables

When accessing the information extracted by tables, the user can use the DPad on the controller to move down or across rows and columns. As mentioned in section 4.10, the system will assume that the first row contains the headings. It will then read out each heading for each row of data per cell. The user can then move through the rows and switch the heading assumptions if they identify that it was originally incorrect.

Images

When accessing the image descriptions, the system will communicate the number of images extracted. The user can use the DPad to run through all the image descriptions that were extracted from the page, these are converted to speech and communicated to the user.

Equations

The equations follow the same structure as images with the additional step of converting the equations to SSML before communicating them to the user.

Notes

For each of the above items, the user can add notes, by voicing their notes, text-to-speech is used to add notes to the JSON file while accessing each of the items. They are tagged with the item title and the item number if there are more than one images for example.

4.15.5   Settings

From the access menu, the user can adjust certain settings related to the communication of information. Using their voice, they can adjust settings based on instructions provided by the system. The settings that can be altered are as follows:

- Graph Audio Window size - Between 1 and 10 (the default is set at 2)

- Voice Gender - Male or Female (the default is female)

- Speaking Rate - Between 0.25 and 4.0 (the default is set at 1.0)

Chapter 5

Testing

During the implementation stage, individual testing was carried out to determine from a range of solutions, the absolute best. The results and chosen methods were discussed and determined in each of those sections. The purpose here is to carry out system-wide tests that will measure the overall performance with all components integrated. This will provide real insight into how well the system functions as a whole.

5.1 Adjustments and Procedures

Before testing, certain adjustments and procedures to the software and hardware were made based on observations made during the implementation stage. These are as follows:

- The camera feed for the system is coming from a built-in webcam in a PC. This created several issues, the first is that the camera is fixed to the screen of the PC. This makes it impossible to test how the user would hold the device over top a document. The second is that the resolution is 0.3 Megapixels, this is incredibly low. While it was possible to identify documents using this as the input feed, with a certain degree of success. The detail in the scanned document was far too low for any information to be recognisable or even extract-able. The workaround for these problems was to pre-record test footage from a smartphone camera, that would mimic how the user would use the device to capture and scan a document. The camera from the smartphone had a 12 Megapixel Image stabilised camera that was more than sufficient. The video frames from this camera were then resized to a resolution of 3200x2400, this approximately matches the resolution from the Raspberry Pi Camera Module v2.

- The playback of the graph audio through the internal speakers resulted in several issues. The first is the hardware limitation in the speakers' ability to produce equal sound at all frequencies. In the range that is being used certain 'blank' points in the frequency range mean that the speakers simply could not produce the sound that accurately represented the data points extracted from the graph. This issue was resolved by using high-quality Bose QC35II headphones, that could represent all frequencies in the range that was being used. The second issue was related to Linux audio drivers that on occasion, would not release and accept new playback requests. The solution was to insert a short time delay between very close requests that would extend beyond the timeout threshold. This ensured that the audio buffer was cleared and the new request would be processed correctly. The time delay ranged from 0.2 to 0.5 seconds.

- On occasion while the system or individual components where being tested, large calls to the Pygame library causes the system to freeze. The entire PC would need to be restarted and the system launched again. The real cause of this issue, at the time of writing this report, is still unknown. What was found to be an effective solution was restarting the system, every 50 test runs to disconnect the library from the controller and drivers. This seemingly prevented the library itself from crashing and hence the PC.

5.2 System Testing

System testing aims to assess the performance of the whole system. Specifically, the combined performance when all components have been fully integrated. This will provide insightful information that will aid in evaluating each of the chosen methods. This excludes software components that are related to the controls and audio of the system as this will be tested in Section 5.3.

5.2.1 Approach

The system is placed into a testing script that cycles through a range of pre-recorded footage. The script plugs the footage directly into the system as the camera feed. The footage then goes through the following scanning and extraction stages.

For each component, the final output and processing time is saved and recorded. All errors that occur during the tests are saved to a log file for review. Hence, the average accuracy rate can be calculated by comparing the actual output with the expected output. The average processing time can also be calculated for each component and collectively as a whole. These will provide a quantitative measure of how the entire system performs and where any of the most common errors occur within the system.

5.2.2 Test Data

The test data comprised of 100 pre-recorded videos of length 20 seconds and they contained a variety of documents surrounding STEM in straight and slanted positions. Every document contained at least text and comprised of an assortment of up to 3 of each: graphs, tables, images and equations. This was created to test the system's ability to handle the most common contents within documents. They were also all single column with normal margins and in colour. Font sizes and structures all varied and all documents were written in word documents or Latex. The tables vary in heading orientation and size. The graphs varied amongst the defined range of function models as stated in Section 4.3.5. The equations also varied amongst the defined range of equation types as stated in Section 4.9.1. Four examples of the type of documents that were tested can be seen in Figure 5.1.

(a) Document with Image and Text

(b) Document with Table and Text

(c) Document with Equation and Text

(d) Document with Graph and Text

Figure 5.1: System Testing Document Examples

5.2.3 Results

The average processing time and accuracy from the tests are presented in Tables 5.1 and 5.2.

Average Processing		Time (Seconds)	
Document Scanning	**Block Extraction**	**Table Extraction**	**Image Descriptions**
6.1	0.72	1	4.97
Graph Extraction	**Graph Descriptions**	**Graph Audio**	**Text Extraction**
0.3	~0	0.04	2.62
	Equation Extraction	**NLP**	
	6.68	2.1	

Table 5.1: System Testing - Average Processing Time

Average		Accuracy (%)	
Document Scanning	**Block Extraction**	**Table Extraction**	**Image Descriptions**
83	77	100	100
Graph Extraction	**Graph Descriptions**	**Graph Audio**	**Text Extraction**
93	100	100	89
	Equation Extraction	**NLP**	
	95	87%	

Table 5.2: System Testing - Average Accuracy

Document Scanning

From this, several observations can be made; first, the document scan took on average 6.1 seconds. This means that roughly 3 seconds were required to identify the document before the 3-second conditional statement verified and captured the page. The accuracy was found to be high at 88%. The majority of the failed scans came from the system being unable to identify a document within the video, as theses tests had no user input to aid in adjusting the camera if the document was not detected. This error rate is mostly negligible. 2 scans failed to capture the entirety of the document, a little over half the page was identified in both cases. If the negligible cases are removed, this brings the accuracy up to 97% which is what we expect to see when the system is under control of the user.

Block Extraction

Block extraction took 0.72 seconds with an accuracy rate of 77%, this is slightly lower than what was expected but not low enough that resulted from any major problems. Other than the known error rate in detecting images and graphs, occasional issues were detected when page format places, paragraphs and titles very close together and this resulted in blocks joining to form larger ones rather than being separate. This did not affect the extraction of information but does impact how the information is structured. Overall the performance here was as expected.

Table Extraction

Table extraction took on average 1 second and with 100% accuracy. This performed as expected with absolutely no issues.

Image Descriptions

Once again, the web scrapping method used for grabbing image descriptions worked perfectly and produced results that were fairly descriptive of images in the document. The processing times, however, although averaging to roughly 5 seconds, were very largely spread. The more complex and detailed the image was, the longer it took to find a description. As the images in the documents were all different this was the most time variable aspect of the system. The short time was recorded at just 1.12 seconds with the longest at 8.75 seconds.

Graph Extraction

This was performed very quickly at 0.3 seconds and achieved an extraction accuracy that was roughly what was expected at 94%. In all cases, there were varying levels of noise and the Savgol filter performed admirably to smooth the data. Overall this component performed very well.

Graph Descriptions

All graph descriptions were accurate and the processing time was so low it to was less than 0.5 milliseconds in all cases.

Graph Audio

With the smoothed data, the conversion to audio worked flawlessly for all graph and the average processing time was the most consistent amongst all components at 0.04 seconds.

Text Extraction

Text extraction achieved an accuracy rate of 89% with the majority of the error coming from false or failed detections in header and footer information such as chapter titles and page numbers. As these items are not particularly important as each page is processed individually, it can be concluded that this component performed very well.

Equation Extraction

Equation extraction took the longest to process with an average of roughly six and a half seconds. This is entirely dependent on the number of blocks extracted from the page, the greater the number of blocks the longer the processing time. The accuracy rate was, once again, as expected at 98%. There were 2 cases where the wrong block was processed, these blocks did not contain words or equations but rather a string of digits which is why it slipped through the system. As this was content-specific and not a feature of the document that can be processed, these errors were ignored.

NLP

Content analysis performed as expected with an average accuracy rate of 87%. The average time of roughly two seconds for both summarization and content analysis is also acceptable and overall this component also performed well.

Overall System

In total, the average total processing time came to 24.53 seconds from the point of starting the document scanner to the completion of extraction. The accuracy is first bounded by the scanning process at 97% with user adjustments. The accuracy for each component thereafter is compounded on top of this. At most the accuracy remains at 97%, and a minimum 75%. This is quite a large spread and is due to the varying accuracy levels between each component. It is clear to see that to improve this system, all components must operate with an equal level of minimum error. However, there were no system failures at any point during the testing process. All the information was correctly stored in their corresponding directories and files. There were also no additional issues that were recorded other than those mentioned above. The processing aspects of this system, therefore, performed very well.

5.3 Usability Testing

Usability testing involves getting prospective users to test the system for it's intended purpose. From this, problems in the design, potential improvement, user information and overall effectiveness can be determined. This means testing all of the user interact-able aspects of the system, such as the user-interface, system to user communication and the representation of information.

5.3.1 Approach

In light of the current Corona-virus situation at the time of writing this report, testing the system with visually impaired students is unfortunately not possible. To best represent how they would use the system, three normal sighted individuals were blindfolded before being asked to start.

For each test user, they were initially briefed with instructions on how to use the system and given up to one hour, blindfolded, to familiarise themselves with the interface. They were then presented with one document at a time to study, and five questions, varying in detail and complexity to answer. The questions were related to information within the document and were designed for the student to make use of each component within the system. They were also designed to test how well the user was able to study and retain information from the page and with how much detail. They were then given up to 45 minutes to answer all the questions.

From each test, the length of time taken to familiarise themselves with the system, answer each question and the number of correct answers and any notes made were recorded. Observations on how the user navigated and made use of the system were also noted. The operation of the controller, speech input and structure of the user interface were also observed to identify any issues or inefficiencies that could be improved upon.

5.3.2 Test Data and Questions

The users were tested against a collection of 10 documents and all were around STEM subjects at the secondary school level. Each was tested before to ensure that they could be perfectly scanned and processed. This was done to eliminate any bias from the system not being capable of processing any of the documents correctly. Three simple and three difficult questions were asked of the user; an example of one document that was used can be seen in Figure 5.2 and the questions asked can be seen below:

1. What is this document about?

2. What does pH measure?

3. What example is given for a pH of 9?

4. Are buffers used to constantly change the pH value of a solution?

5. Blood plasma has what buffers??

6. What is the colorimeter used to measure in Bradford protein assay?

pH of solutions

pH is a measure of the acidity or basicity of a solution. Acidic solutions have a pH of less than 7 and basic (or alkaline) solutions have a pH of greater than 7. A solution with a pH of 7 is said to be neutral. pH is a logarithmic measure of hydrogen ion concentration.

$$\text{pH} = -\log\left[\text{H}^+\right]$$

The pH of a solution can be measured using an electronic meter or an indicator.

pH	Examples of solutions
0	Battery acid, strong hydrofluoric acid
1	Hydrochloric acid secreted by stomach lining
2	Lemon juice, gastric acid, vinegar
3	Grapefruit juice, orange juice, soda
4	Tomato juice, acid rain
5	Soft drinking water, black coffee
6	Urine, saliva
7	'Pure' water
8	Sea water
9	Baking soda
10	Great Salt Lake, milk of magnesia
11	Ammonia solution
12	Soapy water
13	Bleach, oven cleaner
14	Liquid drain cleaner

pH scale

A pH buffer is a solution whose pH changes very little when a small amount of acid or base is added to it. Buffers work by allowing the addition of hydrogen or hydroxide ions without affecting the pH of the solution. Buffer solutions are used as a means of keeping pH at a nearly constant value. A buffer of carbonic acid and bicarbonate is present in blood plasma to maintain a pH between 7.35 and 7.45.

Determining an unknown concentration

It can often be important to determine unknown concentrations of solutions in a laboratory. A **standard curve** is one method that is used to determine the concentration of a solution. A series of 'standards' of known concentration are measured and graphed. This graph can then be used to determine the concentration of an unknown sample.

One such example is the Bradford protein assay. In this test, the depth of colour produced by coomassie brilliant blue dye changes depending on the concentration of protein in the sample; this can be measured using a colorimeter.

Figure 5.2: Usability Testing Document Example [9]

5.3.3 Results and Observations

Average Time to Familiarise	Average Time to Scan and Read	Average Time to Answer Simple Q's
26 Minutes	26 Minutes	6 Minutes
Average Time to Answer Difficult Q's	**Number of Correct Answers out of 180**	**Average Total Time to Study Page**
6 Minutes	180	33 Minutes

Table 5.3: Usability Test Results

The user behaviour must first be analysed to understand how the users approached and answered the questions. Each of them took similar steps. First, they scanned and extracted the information from the document that was placed in front of them. They then proceeded to go through all the information that was present on the page before looking specifically to answer each question. This scan and reading phase took roughly 10 minutes.

Working from question one to six, each user was able to immediately answer simple questions and for more difficult questions, users would revisit sections to better understand the material before answering. As a result, for the simple questions such as question one, users were able to answer on average within 2 minutes of completing the scan and reading phase. For more detailed and complex questions such as question five, users took on average 6 minutes. Overall, approximately 30 minutes was taken to scan, study and answer all five questions. Out of the 60 questions posed to all three users, all were answered correctly.

Out of the ten documents that were tested, two documents contained a question that asked the user to summarise the contents of a particular paragraph and save this in the notes for the page. A further two documents asked the user to double the window size for the graph to audio conversion. This was done to test the input functionality of the system for taking notes and adjusting settings. All three users were successfully able to adjust the window length setting and add their notes with 100% speech-to-text accuracy and found it very easy to do so.

Users reported that navigating the user interface using the controller was quite intuitive and that with repetitive use it would be very easy to memorise and use the system without the need for the instructional cues that are currently built-in. This was reflected in the average time taken to familiarise themselves with the whole system, despite been given up to one hour, almost all users took less than half that time. They also stated that while the Wavenet voice was very clear and easy to understand it did not of course flow as naturally as normal human speech. This is not an issue that can be solved with this project or system but rather through future wider development in text-to-speech algorithms and synthesizers.

One issue that was raised by all users was the delay in the speech that was often experienced between button presses. In all cases, as the script is converted into speech in real-time, it can often lead to delays if the internet connection is slow or if the script is particularly long. This is certainly something that needs to be improved to increase the responsiveness of the system.

Chapter 6

Evaluation

The evaluation of this project must take into consideration that, the core aspect of this system is the integration of many different technologies tackling smaller goals, to achieve a much larger one. In this case, the overall goal is to give visually impaired students the ability to study STEM subjects. Therefore each component that provides the necessary features and functionality to enable this must be evaluated separately and together as a whole to understand how well this was tackled.

6.1   Software

The software met all of the project specifications laid out at the beginning of the project. As seen from the results, the system was capable of capturing a document from a camera feed and processing each of the following items and providing an interface for the user to access that information:

- 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 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

Through an iterative implementation process that explored several possible solutions, each component achieved varying levels of success. For some, the chosen solutions proved to be very effective and in others, it was evident that improvements could be made in several areas.

Document scanning was one of the most successful components of the system, with accuracy rates above 90%. It is what enables this system to capture the document of interest to break it down. It would be highly beneficial for this to be tested with the Raspberry Pi and Pi Camera, to determine how well the instructions to re-position the camera to capture the documents improves the current accuracy rate. Overall, this component performed very well and with further hardware testing, this can be tuned to reach near-perfect accuracy.

Block detection used essentially used the same method as document scanning, the major drop in performance here was due to the identification and separation of graphs and images. The simple method incorporated improved the performance significantly but requires further work. To better separate these two items, work into training object detection models could see a significant increase in the accuracy level. The data-set needed to train this, however, would likely be quite difficult to build. One approach would be to define a specific publisher of notes or textbooks and

form a data-set around the types of images and graphs that they would include in all their material.

Extracting graph data using PlotDigitizer proved to be fairly successful despite using a rather simple method of estimating the location of the axis'. Better estimating the location could come from further experimentation in image processing and OCR analysis, however, it can be concluded that this is likely to be a losing battle simply because of the disparity in the styles of graphs. Again, classifying these styles in some form could lead to a better more focused solution that can estimate the centre of the graph more effectively. However, considering this project is aimed at developing a prototype, this component still performed very well for the graph sets is defined. Once the data was smoothed it ensured that converting the data into audio and determining it's a model function giving the user the best possible chance of accurately understanding the information within it.

The graph to audio conversions also performed exceedingly well, though there was no doubt that this would bring up any major issues. Sound synthesizers have been tried and tested in all forms for many years and in this project worked flawlessly. The graph descriptions also worked very well, with the adjustments made for the 1st and 2nd order polynomials, there were no cases where this failed. Together, these two components provided students with the chance to better interpret and explore graphs. With current methods attempting to convert graph shapes on to braille paper and teachers constantly trying new descriptive methods to help students understand graphs. This solution offers a different approach that proved to be very effective and in all tests, the users were able to identify the shapes and even extract more detailed information about the trend. Overall these components performed very well but with clear improvement points that can be looked at in future work.

The image description method provided simple explanations of the images within the page but with minimal detail. While this met the original specification, it's effectiveness is limited by the ability of Google Images Search and more importantly the scope of training a large enough data set to use machine learning methods. This aspect of the project could use a lot of work in developing this functionality further to provide better information for the user.

The decision to use Tesseract over Google Vision API was based on the higher accuracy rate that could be achieved. Although the processing time increases quite rapidly with the number of characters, the time taken on average to process a document was quick enough to justify this disadvantage. Given that this OCR engine was used as is with no further training, this performed admirably and in the usability tests, proved to extract and decompose the information in the page effectively, giving the user a good level of control.

The specification to extract data from graphs within the page was met through the use of the PlotDigitizer python package. It was indeed able to identify curves within an XY graph and grab the data points that could be used to represent the shape. This was then converted to provide the user with an audio interface to interact with the data and a description of the function model. This adaption of an existing tool enabled the specification to be met but only to a specific range styled graphs. Without the user determining the exact location of the axis, only a best rough estimate can be achieved that sacrifices some of the information in the graph. Smoothing the data was largely successful and the audio conversion was also very effective. Users reported that this could be improved by creating a constant stream of audio based on the location along the curve. This would enable them to smoothly pass through all the points, forwards and backwards, rather than moving between each point. Essentially implementing a smooth live scrubbing feature that can be found in video players.

Natural language processing was not as effective as was initially hoped, specifically concerning the core analysis. Only content analysis saw a marginally useful output that could be used to introduce the page. This area could use significant work into training more labels specific to the STEM subjects to gain better insight into the text and provide more informative points about the page. In its current form, it can provide only generalised topic identification. The Gensim TextRank summarization method used to provide an overview description of the document being

studied performed quite well. At the 100-word threshold limit, it was able to provide concise and informative summaries that gave the user a quick way to evaluate the contents of the page. For this purpose, these components performed very well but improvements can, of course, be made by training deep learning models to approach summarization with the abstractive method. Once again this would require specific training surrounding STEM subjects and while out of the scope of this project, can be developing in future work.

CamelotPro and Mathpix were chosen for extracting tables and equations respectively and both performed near perfectly in the final system. These method choices not only met the specification laid out but exceeded initial performance expectations. While equation extraction was the most time-consuming component, this is forgiven for its high accuracy rate. Reducing this processing time would require identifying which of the blocks contained the equation to reduce the number of API requests. This is not an easy problem as equations cannot be easily classified. They share almost identical properties to the text within a page. One solution would be to classify them based on certain types of documents such as Latex reports. Here the equation identifier, often a number, is placed on the right side margin and could be used to find the placement of the equation within the page. This, of course, is a very focused solution, further research and experimenting would be required to find a much more robust solution.

Text-to-speech performed extremely well with Googles API combined with the Wavenet voices that were developed by DeepMind. There were no major issues when converted text or equations into speech using SSML. The only issue that was encountered came from processing delays and in most cases, this was due to long pieces of text that are first entirely converted and then played back. While this reduced the responsiveness of the system it did not affect the student's ability to use it. This can be improved by breaking the text down further and processing it in chunks or even having prerecorded scripts pre-loaded into the system without needing to process them again and again. This would increase the overall size file size but could prove to be more responsive as it would only need to be played back.

Speech-to-text also performed very well, in all user tests there were no issues or mistakes. Notes and setting changes were all saved and stored correctly. The only observation made from this is the processing time, the longer the length of the note, the longer it can take to identify and convert the speech to text. However, not much can be done about this as the processing is carried out in the cloud by Google.

Overall the various components integrated very well together with no compatibility issues. The controller callbacks operated very well, enabling the user to move throughout the system to access and use all the features. The users also reported that it was very intuitive and easy to learn because of its stark similarities with gaming console interfaces, just without a screen. The audio cues and speech feedback provided clear instructions and users reported that they could very quickly familiarise themselves and use the system quickly, with the builtin instructions. These also made it very easy to teach without additional external input, meaning that individuals such as non-specialised teachers or parents, could use and train children and students without any expensive resources or specific training.

Although all the specifications were met, it's clear to see that they were met with varying degrees of success. It is therefore pivotal that any future work carried out on this project should aim to equalise the success of each of these components first to produce a more robust, equally performing system before focusing on developing a more feature-rich system.

6.2 Hardware

The project only met a single hardware specification due to the unavailability of the Raspberry Pi and Pi Camera modules. Only the Microsoft Xbox 360 controller could be used and tested. The controller proved to be more than sufficient in acting as an input device. The variety of input methods such as the buttons, triggers and joystick provided great flexibility in how the user could interact with the information. The xboxdrv driver used to enable these functions performed flawlessly and the callback function implementation made it very simple and easy to incorporate them. With only half of the inputs on the controller currently being used, it also leaves plenty of room for additional controls and functionality to be integrated with this.

The PC that the system was running on for this project does not represent the same specification as the Raspberry Pi 4 and therefore it is difficult to estimate or conclude how this will perform on that device. It is highly unlikely that there would be any compatibility issues but rather the processing times could be slightly or significantly longer. It would be highly beneficial to test and evaluate the system on not just this device but a range of possible hardware platforms that could provide a small, cheap foundation to deploy this system as a complete device.

While the camera module could not be tested, a smartphone camera, resized to have the same resolution was used. This provided a rough representation of what the camera feed could like in terms of quality from an 8 Megapixel sensor. Throughout the entirety of the project, this posed absolutely no issues and even reducing the resolution further posed only caused issues with noise at very low levels. It is, therefore, safe to assume that the Raspberry Pi Camera v2 would indeed work, however, it is once again clear that it would be highly beneficial to test the system with the actual module to verify this assumption.

6.3 Cost

There are two main sources of cost in this system, the first is in the hardware and the second is in the software. As discussed in the project specification, the base cost for the intended hardware was 74. This still stands true and no changes have been made to this.

Certain software tools and services that are being used have rolling costs that charge a certain amount per API requests a month. These costs are illustrated in Table 6.1; for US Dollar prices, they were converted to British Pounds. At the time of writing this report, the conversion rate is $1 to 0.78.

Service	Free	Price per Month ()	Requests per Month
Mathpix	1,000 Requests	0.00312	up to 100,000 Requests
CamelotPro ExtractTable	25 Images	22.54	per additional 1,000 Images
Google Text to Speech Standard Voice	4 Million (Characters)	3.13	per additional 1,000,000 Characters
Google Text to Speech Wavenet Voice	1 Million (Characters)	12.48	per additional 1,000,000 Character)
Google Natural Language	5,000 Requests	1.56	up to 250,000 Requests

Table 6.1: Software Services Price List

Mathpix is by far the cheapest service used per request with unlimited at a price of much less than 0.01 per month. Considering that the performance of this service was near enough perfect, the low cost is an additional fantastic benefit. CamelotPro runs on the ExtractTable service that is by far the most expensive component. At more than 22 per 1,000 images, this is a hefty price that would be paid by students or schools. However, the price does reflect on how the service

performs with 100% accuracy in the tests performed. It would, however, be beneficial, if further research could be done to try and improve local processing methods or other less accurate but cheaper options. This could bring out other alternatives that could rival the cost.

The Google Text to Speech engine with the standard voice is incredibly cheap, offering 4 million characters per month free. Charging just above 3 per month for every additional 1,000,000 characters. The Wavenet voice is four times as expensive at over 12 and offers fewer free characters at 1,000,000. As the Wavenet voices offer only a marginal improvement in the naturalness of the speech, the costs savings outweigh this slight advantage and the standard voices are used by default but is left as an option that can be chosen by the user.

Google natural language is by far the cheapest item in this list at only 1.50 per month for up to 250,000 requests. As content analysis does not provide particularly useful information, the 5,000 free requests are easily justifiable but additional requests, even at such a low price are questionable. This is once again left for the user to decide if they wish to continue incorporating the brief introduction beyond the initial free stage.

If an assumption is made that a student studies on average 20 pages a day, 5 days a week for a month, this would put the total page count at approximately, 400 pages per month. The average number of words per page through testing documents surrounding STEM material was found to be roughly 300 and the average number of characters in the English language per word is 4.5 [39]. This puts the monthly character count at 540,000. The ongoing monthly cost of studying 400 pages per month would be approximately 26.44. The contributing costs coming from CamelotPro and Mathpix. As the character numbers and requests fall into the free-range, regardless of whether or not the user chooses the Wavenet voice or content analysis, the price remains the same. This brings the total cost in the first month of purchase to 100.44 with a running monthly cost of 26.44.

This is considerably cheaper than the existing solutions that were discussed in the background. Furthermore, with the hardware, CamelotPro and Google services, it is possible to negotiate even cheaper prices if schools purchased these in bulk and used more of each service. The biggest drawback in the pricing is that at some point with extended use, the ongoing costs could exceed the price of similar solutions that have a much higher capital cost but no ongoing cost. Considering the OrCam MyEye which is priced at approximately 2000, the system would need to be run for a little over 6 years for the costs to match. However, the advantages lie in the low start-up costs that make this more affordable right from the beginning. Schools could purchase far more of these devices at once than they could devices such as the MyEye or the BrailleNote Apex. Hence making this far more feasible for schools to purchase for several students.

Chapter 7

Future Work and Conclusion

Due to the nature of this project being a prototype of a system that could be launched as a standalone device for individuals and the education sector. There is a myriad of possibilities as to how this can be improved and put in front of the thousands of visually impaired students that would greatly benefit from this. Suggestions and minor improvements were discussed in the evaluation but the most important improvements that could be made to greatly improve the success factor of this system are presented here.

7.1 Software

As can be made out of the evaluation, many of the components within this system would benefit from models being specifically trained to work with STEM subjects. These would include components such as graph and image detection from block extraction, image classification/ object detection for image descriptions and natural language processing for content and entity analysis as well as abstractive text summarization. Each of these would, of course, involve different approaches but the core method behind them all is the same. Forming a large enough data set for each type and training the relevant models to correctly identify labels, create and process tokens and classify items. This is a lengthy and costly procedure for multiple reasons, the first is that with STEM there are at least 6 major categories, Physics, Chemistry, Biology, Technology, Engineering and Maths. However, each can be broken down even further and still be left with far too many avenues to focus on. It is this large scope and expensive cloud and local computing services that pushed this development outside of the scope of this project. For future work, one particular subject could be chosen to research and develop the system further to see if it would be possible to incorporate these trained models to improve the performance.

One key process that would greatly benefit from further research and development is the automatic identification of graph centre points and graphs scales. Thus far, a very simple method is used to estimate the centre point and the graph axis scales have been disregarded as the shape of the curve is the main interest. Finding a reliable and robust method of determining the (0,0) point on the graph would allow more accurate extraction of the entire curve. Finding a way to determine the axis scales, would enable accurate data to be extracted. This becomes particularly important when logarithmic scales are used.

Centralising the scanned documents and making them readily available for all users could also see some major advantages. Work into forming a cloud database of scanned and referenced documents would enable students to study a whole host of material that may not have access to. This also would enable multiple scans of the same document to be cross-examined and checked to store the most accurate and up to date version. Should this be successful there is even more potential for digital material to be plugged directly into the system. This would enable teachers to prepare specific documents for classes but allow students to use this system to access that information. This could be done by simply extracting the metadata from, for example, a PDF, and enabling the student to use the same controller and audio input and output methods to study the material.

One aspect of this project that could not be carried out, was working with visually impaired students. This would have provided a better understanding of how well the system would function in the real world and what changes or improvements could be made before launching a final product. This is certainly something that should be carried in future work, getting in contact with charities and organisations that would be interested in helping this system to bring it to market.

In addition to this system being deployed as a standalone device with the Raspberry Pi acting as the hardware base, there is potential for this system to reach a wider user base. For those who are partially impaired and not severely impaired or blind, this could be converted into a mobile application. For these users, using smartphones is possible through accessibility features built into the device itself, such as speech feedback and magnifiers. Making use of the cameras that are built into modern-day smartphones, the system could easily be converted into an Android and Apple application through the use of software packages such as Kivy [40].

7.2 Hardware

As most of the hardware components in this project could not be tested, the first major future step would be to run this system on a Raspberry Pi 4 and Pi Camera. This will provide insight into how this system could be optimised and deployed on this hardware or if alternative solutions need to be considered. One such situation could arise if the Raspberry Pi is found not to be powerful enough. In this case, all heavy processing tasks could be run on a cloud service such as Google Cloud Services, this would elevate the load from the device but would make having an internet connection essential. The end goal is to have a device that is small, cheap, ergonomic and responsive to make it easy for users without vision to hold and scan documents quickly and efficiently.

The controller could also be explored, at this point, the Xbox 360 controller forms a fantastic baseline for what type of controls are needed and how many. It also provides a good understanding of how the user interface can be optimised both in software and hardware. This opens up the possibility of developing a custom controller, that takes the necessary physical input buttons and triggers and puts them into a form factor that is smaller and more ergonomic. This would make extended periods use more comfortable than the standard controller. There could even be the potential for the entire device to be built into the controller itself, creating an all in one device that could be used in more environments due to its portability.

7.3 Conclusion

This project set out to develop an integrated system of existing computer vision technologies and machine learning-based services to enable visually impaired and blind people to study STEM subjects. Particularly, paper-based material that has not or cannot be converted into a digital format that can be used by many existing assistive technologies such as Text Enlargers and Text-to-Braille devices. The success of this project came from the integration, that not only enabled users to study a wide array of material but also at a cost much less than the majority of similar advanced solutions such as the OrCam MyEye. Despite this, as seen in the results in Section 5, not all aspects of the system achieved their goal to the same uniform level and improvements in areas such as block detection and NLP could be improved with further development. As a prototype, however, it proved that such a system can be effective and with further development can be deployed into schools and made available for individuals to purchase.

Chapter 8

Appendices

8.1 Repository

GitHub : https://github.com/rish21/ATFS

8.2 User Interface Script

```
{
    "main_001": "Hello, please wait one moment while the system starts up",
    "main_002": "Press A to begin scanning a new page, or press Y to access stored
        material",
    "main_003": "To start extracting the page press X",
    "main_004": "The system is now shutting down",
    "main_005": "The information in the document will now be extracted",
    "main_006": "Please try and scan a document first before extracting information",
    "main_007": "Please try scan and extract a document first, before accessing the new
        document",
    "main_008": "You are now at the main menu",
    "main_009": "To access the document just scanned and extracted, press B, otherwise,
        press Y to access stored documents",
    "main_010": "The document was not scanned, please try again by pressing A",
    "main_011": "The information was not extracted, please try again by pressing X",
    "scanner_001": "The page has been found, please hold steady",
    "scanner_002": "The page has been captured successfully",
    "scanner_003": "Try to bring the device closer to the page",
    "scanner_004": "Try to hold the device steady",
    "scanner_005": "The page could not be found, please try again",
    "extraction_001": "Please wait while the information is extracted, this can take up
        to 30 seconds",
    "extraction_002": "There are roughly 15 seconds remaining",
    "extraction_003": "The document information has been successfully extracted",
    "extraction_004": "The document information failed to extract successfully, please
        try and re-scan the document or rerun the extraction",
    "access_001": "If you would like to access, the images, please press Y",
    "access_002": "If you would like to access, the text, please press Right Shoulder
        Button",
    "access_003": "If you would like to access, the graph, please press X",
    "access_004": "If you would like to access, the equations, please press B",
    "access_005": "If you would like to access, the tables, please press A",
    "access_006": "You have reached the upper limit. There are no more items, press back
        to return to the access menu or use the Dpad to continue",
    "access_007": "You have reached the lower limit. There are no more items, press back
        to return to the access menu or use the Dpad to continue",
    "access_008": "You are now at the access menu",
```

```
    "access_009": "The rows and columns have been swapped",
    "access_010": "Please use the left and right directions on the Dpad to select
        between, graph descriptions, full graph audio and by point graph audio. Please
        press X again once you have decided.",
    "access_011": "Graph descriptions",
    "access_012": "Full graph audio",
    "access_013": "By points graph audio",
    "access_014": "Selected",
    "access_015": "Please use the left and right directions on the Dpad to select
        between, adding a new note and accessing saved notes. Please press XBOX button
        again once you have decided.",
    "access_016": "If you would like to access or add notes, please press the XBOX
        button",
    "access_017": "Add a new note",
    "access_018": "Saved notes",
    "access_019": "To add a new note, press the XBOX button again and start speaking.
        Once you have finished speaking, please wait a few seconds while the note is
        saved",
    "access_020": "Your note has been saved, please press the XBOX button to save more
        notes or press back to return the access menu",
    "access_021": "Full block of text",
    "access_022": "Individual sentences",
    "access_023": "This sentence is currently not highlighted. If you would like to
        highlight this sentence, please press the Left shoulder button, otherwise hit
        back",
    "access_024": "This sentence has been previously highlighted",
    "access_025": "This sentence has been highlighted",
    "access_026": "You are now at the settings menu",
    "access_027": "If you would like to change the window length, please press A",
    "access_028": "To change the window length, please say a number between 1 and 10. The
        large the window length, the fewer the number of data points. It is recommended
        that it be kept between 2 and 4.",
    "access_029": "The setting has been changed",
    "access_030": "Please try again",
    "access_031": "If you would like to change the gender of the voice, please press X",
    "access_032": "To change the gender, please say male or female",
    "access_033": "If you would like to change the speech rate, please press B",
    "access_034": "To change the speech rate, please say a number between 0.25 and 4. It
        is recommended that it be kept between 1.",
    "access_035": "If you would like to access or store notes, please press the Xbox
        button"
}
```

Listing 8.1: User Interface Script

8.3 SSML Conversion List

```python
# Greek Letters
convert = convert.replace('\varepsilon', ' epsilon ')
convert = convert.replace('\vartheta', ' theta ')
convert = convert.replace('\mu', ' mew ')
convert = convert.replace('\nu', ' new ')
convert = convert.replace('\\xi', ' sigh ')
convert = convert.replace('\varrho', ' rho ')
convert = convert.replace('\varphi', ' phi ')


# Other Symbols
convert = convert.replace('\infty', ' infinity ')
convert = convert.replace('\Re', ' real ')
convert = convert.replace('\nabla', ' differential ')
convert = convert.replace('\mu', ' mew ')
convert = convert.replace('\neg', ' negative ')
convert = convert.replace('\Im', ' imaginary ')
convert = convert.replace('\nexists', ' does not exist ')
convert = convert.replace('\varnothing', ' nothing ')
convert = convert.replace('\cdots', ' and so on ')
convert = convert.replace('\surd', ' square root of <break time="0.5s"/> ')
convert = convert.replace('\angle', ' an angle of ')


# Operations
convert = convert.replace('\div', ' divided by ')
convert = convert.replace('\cup', ' union ')
convert = convert.replace('\cap', ' intersection ')
convert = convert.replace('\subset', ' is a proper subset of ')
convert = convert.replace('\not\subset', ' is not a proper subset of ')
convert = convert.replace('\subseteq', ' is a subset of ')
convert = convert.replace('\nsubseteq', ' is not a subset of ')
convert = convert.replace('\supset', ' is a proper super of ')
convert = convert.replace('\not\supset', ' is not a proper super of ')
convert = convert.replace('\supseteq', ' is a super of ')
convert = convert.replace('\nsupseteq', ' is not a super of ')

convert = convert.replace('\neq', ' is not equal to ')
convert = convert.replace('\ne', ' is not equal to ')
convert = convert.replace('\nless', ' is not less than ')
convert = convert.replace('\leqslant', ' is less than or equal to ')
convert = convert.replace('\nleq', ' is neither less than or equal to ')
convert = convert.replace('\nleqslant', ' is neither less than or equal to ')
convert = convert.replace('\geq', ' greater than or equal to ')
convert = convert.replace('\gtr', ' is not greater than ')
convert = convert.replace('\ngtr', ' is not greater than ')
convert = convert.replace('\geqslant', ' is greater than or equal to ')
convert = convert.replace('\ngeq', ' is neither greater than or equal to ')
convert = convert.replace('\ngeqslant', ' is neither greater than or equal to ')

convert = convert.replace('\sinh', ' sine ')
convert = convert.replace('\cosh', ' cosine ')
convert = convert.replace('\tanh', ' cosine ')

convert = convert.replace('\int', ' the integral of <break time="0.5s"/> ')
convert = convert.replace('\sum', ' the sum of <break time="0.5s"/> ')
convert = convert.replace('\prod', ' the product of <break time="0.5s"/> ')
convert = convert.replace('\lim_', ' with a lower limit of <break time="0.5s"/> ')
convert = convert.replace('_{', ' with a lower limit of <break time="0.5s"/> ')
convert = convert.replace('^', ' to the power of ')
```

```
convert = convert.replace('\in', ' belongs to <break time="0.5s"/> ')
convert = convert.replace('\perp', ' is perpendicular to ')
convert = convert.replace('\notin', ' does not belong to <break time="0.5s"/> ')
convert = convert.replace('\simeq', ' is similarly equal to <break time="0.5s"/> ')
convert = convert.replace('\sim', ' is similar to <break time="0.5s"/> ')
convert = convert.replace('\approx', ' is approximately equal to ')
convert = convert.replace('\equiv', ' is equivalent to <break time="0.5s"/> ')
convert = convert.replace('\cong', ' is congruent to <break time="0.5s"/> ')
convert = convert.replace('\propto', ' is proportional to <break time="0.5s"/> ')
```

Listing 8.2: SSML Conversion List

8.4 Text Extraction Example Outputs [1]

8.4.1 Tesseract

The Principles of Science and Interpreting Scientific Data

Scientific method refers to the body of techniques for investigating phe- nomena, acquiring new knowledge, or correcting and integrating previous knowledge. It is based on gathering observable, empirical and measurable evidence subject to specific principles of reasoning.

Isaac Newton (1687, 1713, 1726) Rules for the study of natural philosophy, Philosophiae Naturalis Principia Mathematica

Forensic science actually is a broad array of disciplines, as will be seen in the next chapter. Each has its own methods and practices, as well as its strengths and weaknesses. In particular, each varies in its level of scientific development and in the degree to which it follows the principles of scientific investigation. Adherence to scientific principles is important for concrete reasons: they enable the reliable inference of knowledge from uncertain informationexactly the challenge faced by forensic scientists. Thus, the reliability of forensic science methods is greatly enhanced when those principles are followed. As Chapter 3 observes, the laws admission of and reliance on forensic evidence in criminal trials depends critically on (1) the extent to which a forensic science discipline is founded on a reliable scientific methodology, leading to accurate analyses of evidence and proper reports of findings and (2) the extent to which practitioners in those foren- sic science disciplines that rely on human interpretation adopt procedures and performance standards that guard against bias and error. This chapter discusses the ways in which science more generally addresses those goals.

111

8.4.2 Google Vision API

4

The Principles of Science and Interpreting Scientific Data

Isaac Newton (1687 , 1713 , 1726) Rules for the study of natural philosophy , " Philosophiae Naturalis Principia Mathematica

Forensic science actually is a broad array of disciplines , as will be seen in the next chapter . Each has its own methods and practices , as well as its strengths and weaknesses . In particular , each varies in its level of scientific development and in the degree to which it follows the principles of scientific investigation . Adherence to scientific principles is important for concrete reasons : they enable the reliable inference of knowledge from uncertain information - exactly the challenge faced by forensic scientists . Thus , the reliability of forensic science methods is greatly enhanced when those principles are followed . As Chapter 3 observes , the law's admission of and reliance on forensic evidence in criminal trials depends critically on (1) the extent to which a forensic science discipline is founded on a reliable scientific methodology , leading to accurate analyses of evidence and proper reports of findings and (2) the extent to which practitioners in those foren sic science disciplines that rely on human interpretation adopt procedures and performance standards that guard against bias and error . This chapter discusses the ways in which science more generally addresses those goals ."

111

8.4.3 Usability Testing Example Raw Output

access.JSON

```
{
    "page": [
        {
            "bookmarked": "",
            "topic": " Science",
            "description": "LABORATORY TECHNIQUES FOR BIOLOGISTS PH of solutions pH is a
                measure of the acidity or basicity of a solution.\nA standard curve is one
                method that is used to determine the concentration of a solution.\nA
                series of 'standards' of known concentration are measured and graphed.",
            "notes": [],
            "image_results": [],
            "graph_results": [],
            "equations": [
                {
                    "0": "pH = - { log } [ H ^ { + } ]"
                }
            ],
            "text": [
                {
                    "full_text": "TOPIC 1. LABORATORY TECHNIQUES FOR BIOLOGISTS",
                    "sentences": [
                        {
                            "text": "TOPIC 1.",
                            "highlighted": ""
                        },
                        {
                            "text": "LABORATORY TECHNIQUES FOR BIOLOGISTS",
                            "highlighted": ""
                        }
                    ]
                },
                {
                    "full_text": " ",
                    "sentences": [
                        {
                            "text": " ",
                            "highlighted": ""
                        }
                    ]
                },
                {
                    "full_text": "PH of solutions",
                    "sentences": [
                        {
                            "text": "PH of solutions",
                            "highlighted": ""
                        }
                    ]
                },
                {
                    "full_text": "pH is a measure of the acidity or basicity of a solution.
                        Acidic solutions have a pH of less than 7 and basic (or alkaline)
                        solutions have a pH of greater than 7. A solution with a pH of 7 is
                        said to be neutral. pH is a logarithmic measure of hydrogen ion
                        concentration.",
                    "sentences": [
                        {
```

```
                "text": "pH is a measure of the acidity or basicity of a
                    solution.",
                "highlighted": ""
        },
        {

                "text": "Acidic solutions have a pH of less than 7 and basic (or
                    alkaline) solutions have a pH of greater than 7.",
                "highlighted": ""
        },
        {
                "text": "A solution with a pH of 7 is said to be neutral.",
                "highlighted": ""
        },
        {

                "text": "pH is a logarithmic measure of hydrogen ion
                    concentration.",
                "highlighted": ""
        }
    ]
},
{

    "full_text": "The pH of a solution can be measured using an electronic
        meter or an indicator.",
    "sentences": [
        {
                "text": "The pH of a solution can be measured using an
                    electronic meter or an indicator.",
                "highlighted": ""
        }
    ]
},
{

    "full_text": "pH Examples of solutions 0 Battery acid, strong
        hydrofluoric acid 1 Hydrochloric acid secreted by stomach lining 2
        Lemon juice, gastric acid, vinegar",
    "sentences": [
        {
                "text": "pH Examples of solutions 0 Battery acid, strong
                    hydrofluoric acid 1 Hydrochloric acid secreted by stomach
                    lining 2 Lemon juice, gastric acid, vinegar",
                "highlighted": ""
        }
    ]
},
{

    "full_text": "A pH buffer is a solution whose pH changes very little
        when a small amount of acid or base is added to it. Buffers work by
        allowing the addition of hydrogen or hydroxide ions without
        affecting the pH of the solution. Buffer solutions are used as a
        means of keeping pH at a nearly constant value. A buffer of
        carbonic acid and bicarbonate is present in blood plasma to
        maintain a pH between 7.35 and 7.45.",
    "sentences": [
        {
                "text": "A pH buffer is a solution whose pH changes very little
                    when a small amount of acid or base is added to it.",
                "highlighted": ""
        },
        {

                "text": "Buffers work by allowing the addition of hydrogen or
                    hydroxide ions without affecting the pH of the solution.",
```

```json
            "highlighted": ""
        },
        {

            "text": "Buffer solutions are used as a means of keeping pH at a
                nearly constant value.",
            "highlighted": ""
        },
        {

            "text": "A buffer of carbonic acid and bicarbonate is present in
                blood plasma to maintain a pH between 7.35 and 7.45.",
            "highlighted": ""
        }
    ]
},
{
    "full_text": "Determining an unknown concentration",
    "sentences": [
        {

            "text": "Determining an unknown concentration",
            "highlighted": ""
        }
    ]
},
{

    "full_text": "It can often be important to determine unknown
        concentrations of solutions in a laboratory. A standard curve is
        one method that is used to determine the concentration of a
        solution. A series of 'standards' of known concentration are
        measured and graphed. This graph can then be used to determine the
        concentration of an unknown sample.",
    "sentences": [
        {

            "text": "It can often be important to determine unknown
                concentrations of solutions in a laboratory.",
            "highlighted": ""
        },
        {

            "text": "A standard curve is one method that is used to
                determine the concentration of a solution.",
            "highlighted": ""
        },
        {

            "text": "A series of 'standards' of known concentration are
                measured and graphed.",
            "highlighted": ""
        },
        {

            "text": "This graph can then be used to determine the
                concentration of an unknown sample.",
            "highlighted": ""
        }
    ]
},
{

    "full_text": "One such example is the Bradford protein assay. In this
        test, the depth of colour produced by coomassie brilliant blue dye
        changes depending on the concentration of protein in the sample;
        this can be measured using a colorimeter.",
    "sentences": [
        {

            "text": "One such example is the Bradford protein assay.",
```

```
                    "highlighted": ""
                },
                {

                    "text": "In this test, the depth of colour produced by coomassie
                        brilliant blue dye changes depending on the concentration
                        of protein in the sample; this can be measured using a
                        colorimeter.",
                    "highlighted": ""
                }
            ]
        },
        {
            "full_text": "\u00a9 HERIOT-WATT UNIVERSITY",
            "sentences": [
                {
                    "text": "\u00a9 HERIOT-WATT UNIVERSITY",
                    "highlighted": ""
                }
            ]
        }
    ],
    "misc": [
        {
            "0": 2
        }
    ]
    }
],
"settings": [
    {
        "window": 2,
        "gender": "female",
        "speaking_rate": 1.1
    }
]
}
```

Listing 8.3: Generated access.JSON

Table Extraction Output

pH	Examples of solutions
O	Battery acid, strong hydrofluoric acid
1	Hydrochloric acid secreted by stomach lining
2	Lemon juice, gastric acid, vinegar
3	Grapefruit juice, orange juice, soda
4	Tomato juice, acid rain
5	Soft drinking water, black coffee
6	Urine, saliva
7	'Pure' water
8	Sea water
9	Baking soda
10	Great Salt Lake, milk of magnesia
11	Ammonia solution
12	Soapy water
13	Bleach, oven cleaner
14	Liquid drain cleaner

Bibliography

[1] T. N. Academies, "Strengthening forensic science in the united states: A path forward (2009)," 2020. [Online]. Available: https://bit.ly/2CDiOlJ

[2] Humanware, "Braillenote apex," 2020. [Online]. Available: https://bit.ly/2YYD6O4

[3] ——, "Braillenote touch 18 plus braille note taker/tablet," 2020. [Online]. Available: https://bit.ly/2VutSbB

[4] ORCAM, "Orcam device comparison," 2020. [Online]. Available: https://bit.ly/3fQfzpp

[5] J. T. Point, "Regression vs. classification in machine learning." [Online]. Available: https://bit.ly/2BFho9F

[6] J. Williams, "Deep learning in digital pathology," 2018. [Online]. Available: https://bit.ly/37VjSNk

[7] U. Saxena, "Speech synthesis techniques using deep neural networks," 2017. [Online]. Available: https://bit.ly/2NwkXlb

[8] R. F. Cohen, A. Meacham, and J. Skaff, "Teaching graphs to visually impaired students using an active auditory interface," pp. 279–282, 2006. [Online]. Available: https://bit.ly/3ewh0sG

[9] B. Smith and D. Campbell, "Cfe advanced higher biology unit 1: Cells and proteins," 2020. [Online]. Available: https://bit.ly/2YtKGRJ

[10] W. H. Organization, "World report on vision," pp. 1–28, 2019. [Online]. Available: https://bit.ly/3i47QWP

[11] C. E. H. Journal, "Changing patterns in global blindness: 19882008," 2008. [Online]. Available: https://bit.ly/3hXLwhm

[12] A. Foster, "Childhood blindness," pp. S27–S36, 1988. [Online]. Available: https://go.nature.com/31d41Z6

[13] G. H. Abner and E. A. Lahm, "Implementation of assistive technology with students who are visually impaired: Teachers'readiness," pp. 98–105, 2002. [Online]. Available: https://bit.ly/2Zj16f9

[14] D. D. Kumar and G. P. Stefanich, "Science for students with visual impairments: Teaching suggestions and policy implications for secondary educators," 2001. [Online]. Available: https://bit.ly/2YvgWE5

[15] RNIB, "Teaching stem subjects to blind and partially sighted learner," 2020. [Online]. Available: https://bit.ly/2Z3aROm

[16] M. Contributor, "Braille versions of textbooks help blind college students succeed," 2017. [Online]. Available: https://bit.ly/2CDgfA7

[17] RNIB, "The criteria for certification." [Online]. Available: https://bit.ly/2CwH34R

[18] E. Vision, "Acrobat hd ultra lcd," 2020. [Online]. Available: https://bit.ly/3hWIsCa

[19] PAMTRAD, "Aumed image hand held magnifier," 2020. [Online]. Available: https://bit.ly/2Yrs7Oa

[20] KNFB, "Knfb reader," 2020. [Online]. Available: https://bit.ly/2CD88DJ

[21] J. Meddaugh, "Braillenote apex," 2020. [Online]. Available: https://bit.ly/31dHVpE

[22] BrailleWorks, "History of braille," 2020. [Online]. Available: https://bit.ly/2AY0bbQ

[23] M. P. Ekstrom, "Digital image processing techniques," pp. 1–47, 1984. [Online]. Available: https://bit.ly/2Z1HKe8

[24] R. R. Schaller, "Moores law: past, present, and future," pp. 53–59, 1997. [Online]. Available: https://bit.ly/2YvhKZq

[25] E. Alpaydin, "Introduction to machine learning," 2014. [Online]. Available: https://bit.ly/2VeZvFQ

[26] pathmind, "Regression vs. classification in machine learning." [Online]. Available: https://bit.ly/3hXMmL2

[27] G. Shperber, "A gentle introduction to ocr," 2018. [Online]. Available: https://bit.ly/37WStKP

[28] CMUSphinx, "Basic concepts of speech recognition," 2018. [Online]. Available: https://bit.ly/3fZi8FP

[29] A. Nandan, "Hello world in speech recognition," 2019. [Online]. Available: https://bit.ly/3hZW1R6

[30] SAS, "Natural language processing (nlp)." [Online]. Available: https://bit.ly/3fUfYaa

[31] "Ohm's law," 2020. [Online]. Available: https://bit.ly/2VeJwaD

[32] "Graph of a parabola," 2020. [Online]. Available: https://bit.ly/3hWiIpj

[33] Google, "Get image descriptions on chrome," 2020. [Online]. Available: https://bit.ly/2BvBkvL

[34] tesseract ocr, "Tesseract(1) manual page," 2020. [Online]. Available: https://bit.ly/3hYVQ8U

[35] pylanguagetool, "Python api wrapper for the languagetool rest api," 2020. [Online]. Available: https://bit.ly/2AVD41C

[36] SQA, "Advanced higher chemistry course/unit support notes," 2020. [Online]. Available: https://bit.ly/3hXNEWo

[37] M. A. S. S. E. D. T. J. B. G. Mehdi Allahyari, Seyedamin Pouriyeh and K. Kochut, "Text summarization techniques: A brief survey," 2020. [Online]. Available: https://bit.ly/3dpbKpu

[38] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts," 2020. [Online]. Available: https://bit.ly/3dvZmUy

[39] T. C. H. Connecticut, "Percentages of letter frequencies per 1000 words," 2020. [Online]. Available: https://bit.ly/3fU6Wdw

[40] Kivy, "Python kivy framework," 2020. [Online]. Available: https://bit.ly/2CDiKlZ