UNIVERSITY OF
LINCOLN

A Study of an Optical Character Recognition (OCR) and Receipt Analysis System to Determine Best Image Pre-Processing Methods and Feasibility of OCR in Real Life Environments.

Benjamin Harry Terrill

(TER15622143)

A dissertation in partial fulfilment of the requirements for the degree of Bsc(Hons) Computer Science

School of Computer Science

University of Lincoln

2020

Table of Contents

1. Abstract

This project aims to present a summary of a successful design, development and testing of a trained Optical Character Recognition program that can identify both words and numbers printed on a receipt. The program will then store the information so the user can more easily track the data being input and to enable them to track expenses, compare different prices from shops or online ordering sites, and assist when claiming back tax expenses. The results obtained from the testing conducted will determine the feasibility of Optical Character Recognition systems for real-life scenarios. The feasibility can be determined based on a number of key tests, the first being an accuracy score returned by the system when comparing the results to a ground truth. The second test is a confidence score for each individual character. When conducted, these tests showed the system performed at an average of 80% for accuracy score, which would likely be acceptable for personal use where mistakes can be easily rectified. However, this result would be inappropriate for business use as there could still be a huge amount of data to correct.

2. Introduction

When purchasing items as an individual or as a business owner, you will get many receipts that contain a range of valuable information that can have many use cases. The most obvious piece of data on a receipt is a total amount which can be used to track expenses, compare prices from previous shops or be used when claiming back tax (VAT) on expenses. Tracking receipts is often difficult due to the amount that can build up. Additionally, storing them in a way that allows for the individual to easily look back through is a difficult and time consuming process. This is especially true when tracking receipts over long time periods, such as over many years. Many of these use cases can be improved by digitizing the information on the receipts rather than keeping all of them paper based. Normally, this would mean manually typing out the information on the receipt into a computer spreadsheet or database. However, this is not realistic for the following reasons. Firstly, this method would be time consuming and costly in terms of paying someone to do the work. Secondly, this method would have a high likelihood of user error causing incorrect data to be inputted into the system.

This project is designed to determine if the results from an Optical Character Recognition program can be used for real-life environments such

as personal and business use. This would mean that high enough accuracy scores would need to be produced with usable outputs.

The following report explains how the project design, development and testing were carried out. Using supporting literature a conclusion on Optical Character Recognitions practicality and cohesion with storage systems was made.

2a. Aims and Objectives

i. Aims

- Develop a tool using MATLAB and Optical Character Recognition (OCR) to analyse data from receipts or other sources (i.e. email receipts) and identify information about the items and cost.
- Store the data for the user to access and keep a log of money spent.

ii. Objectives

1. Literature; read and review literature to develop more context for the project.
2. Create a project plan that shows detailed timescales and milestones for completing the project.
    a. Gantt chart to keep each step on track week by week.
    b. Risk analysis to identify specific risks when developing the artefact and well as how each risk might be managed or mitigated.
3. Collet paper or email receipts.
4. Develop OCR program to read and identify important parts of source material and return them to be analysed.
    a. Read the name of items/cost/discount from the receipt and ignore unneeded data.
    b. Crop out unneeded parts of the image i.e. background noise.
5. Set up storage system for data.
    a. Store data using template in Microsoft Excel
6. Conduct qualitative and quantitative testing
    a. Quantitative – accuracy scores compared to ground truth
    b. Qualitative – output image of receipt before and after image pre-processing to evaluate quality of image.

3. Literature Review

To most people, receipts are something that are looked at once or often thrown away immediately after leaving a store and it is often overlooked how important the data on a receipt can be for many different circumstances. Many of the potential use cases of an automated receipt digitizer are outlined by Richter (2019), who wrote the blogpost about the many use cases of a Computer Vision system for digitizing receipts. For example, an individual could track their monthly expenses with ease using a system that stores the total amount spent or compare how much they spent at one store compared to another. Furthermore, this system would be helpful for someone who claims back business expenses or tax (VAT) and would be a better system to store data than having a drawer full of receipts. Richter (2019) also includes the basic steps needed to extract information from a receipt starting with simple image processing techniques that can drastically improve the accuracy of the Optical Character Recognition system. This is a key part of the program's success as a high accuracy is necessary for it to be worthwhile and not cause the user to have spent time fixing errors in the recognition. Furthermore, many of the practical uses discussed here are in line with the intended uses of my artefact as the negative effects of the OCR giving inaccurate results are minimal.

The use of automatic receipt analysis is increasing due to the high cost of manually entering the data on a receipt and the time it takes to complete. Having an automatic receipt analysis system would require an image processing system that is adaptable to real-life environments. This means using photos taken from a mobile phone of everyday receipts that can differ from different stores in different countries. Suponenkovs et al. (2017) wrote a paper about some of the challenges that apply when creating an image recognition system for real-life environments. Some of the issues include the limitations of current image processing technologies to identify handwriting opposed to printed text. Fortunately, this will not affect the system in most scenarios but can reduce the accuracy of the system significantly and cause incorrect data to be stored. Once the data has been processed it will need to be converted to a suitable format so it can be analysed by the user. Suponenkovs et al. (2017), discusses the positive impact this will have on the usability of the information after the system has been used and something as simple as a spreadsheet format will allow for easy reading and filtering of the information. For my artefact I plan on using a simple storage solution that most people will be able to use without training, such as a Microsoft Excel spreadsheet.

Character recognition on something like a receipt can be done using pre-built libraries, such as MATLAB's OCR function or python libraries for some of the best results. However, another common choice for an image processing system is the deep learning approach. Anh Duc Le et al. (2019) showed that this method has a higher F1 score for detection and recognition at 71.9% which includes the recognition of hand writing as well as printed text. This high accuracy score was achieved by employing a Connectionist Text Proposal Network (CTPN) for text detection by accurately localising test lines in natural images and AED for text recognition, as well as extensive image pre-processing to improve the image quality before Optical Character Recognition was applied. This shows the potential of machine learning techniques being applied to character recognition, but also shows that the significant improvements to Optical Character Recognition need to be made to achieve high enough accuracies for many practical uses and it can be much less consistent than tried and tested methods. Anh Duc Le et al. (2019) suggests this problem is largely due to the wide range of different styles of receipts and quality of the images provided. In most practical applications the image will likely be taken from a mobile phone and may not be the clearest image.

To create an Optical Character Recognition (OCR) system, it is important to first know how each step of the system works. This includes; pattern recognition, classification process, pre-processing, feature extraction and model estimation for the training data. For the test data the last step is replaced with the actual classification so the system can be tested, see below:

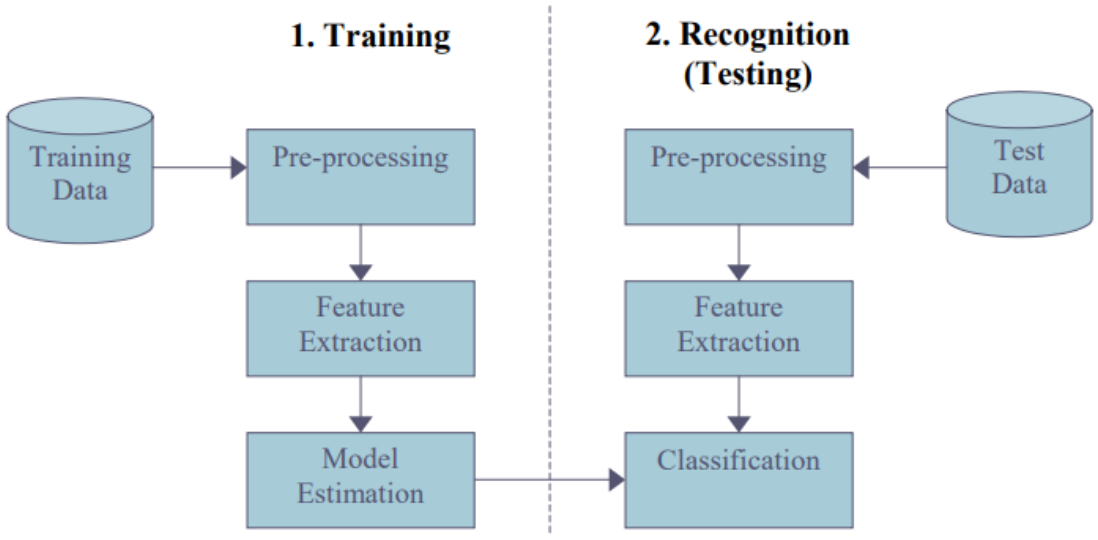

Figure 1: Pattern Classification Process

Hansen (2018) identifies the three most common types of pre-processing performed in OCR. The first is Binarization, which is the process of converting the image into grayscale or black and white to better identify foreground pixels. The second is Morphological Operator, which removes specks and holes in the characters, which can happen if the text is not printed perfectly. However, in small writing this can cause things like a semi-colon (;) to turn into an 'I' so has limited use in some scenarios. The final one is segmentation which can remedy any problems caused by Morphological Operator, this finds the gaps to identify the character correctly, for example, the letter I, or a colon (: or ;). Hansen (2018) states that "Segmentation is by far the most important aspect of the pre-processing stage". This is especially true in the case of printed text but when it comes to handwritten text, it is more complicated if the letters are joined up. Classification is the process of deciding the class membership of the patterns given. In the example of Optical Character Recognition, this would be the system categorising the feature patterns into a character class.

Following the collection of information from several receipts using the Optical Character Recognition, the information will then be stored for the user to be able to view and filter. A user could potentially have thousands of receipts which will contain information to be stored. Improvements in computer technology and automated data collection can make creating a large store of data both necessary and easier to implement. The Transportation Research Board (Chapter 4, 2013) proposed that the only viable way to store large amounts of data of different types was a database system. This is due to it being the most convenient way to maintain and utilize the data after it has been stored. Furthermore, it is safer to have it stored this way because it can have both on and off site back-ups created with ease and there is practically no limit to the size of the database due to its scalability. This has its obvious advantages over paper, being that it does not take up physical space and can be searched through easily, which is what I intend to replace with my artefact.

4. Methodology

4a. Project Management

The project management style should be decided before commencing with the project as it outlines how the work is to be completed. It starts with creating a project plan which contains the aims and objectives of the project, which can be accompanied by a schedule, in this case a Gantt chart. Depending on the project management methodology chosen for the completion of the project different directives are followed, for example in an Agile Project Management approach, the original design can be altered based on feedback and involves regular scrum meetings to discuss and adapt the system to better suit the requirements. In comparison, Waterfall Project Management uses a strict plan set out from the beginning where each section of the design and development stage has to be completed before moving on to the next.

The specific demands of this project incorporates constant feedback and an iterative design and development process to ensure any new requirements are met. Because of this the obvious choice when choosing a project management methodology would be an Agile one. More specifically an Extreme Programming (XP) which features three basic tenets, as explained above:

- Communication
- Simplicity
- Feedback

This allows the system to be improved with each iteration based on feedback from the supervisor.

4b. Project plan and Gantt chart

The project plan is a rough estimate for how long each section of the design and development of the project will take. This was created at the start of the project and many of the sections changed during development, this caused the project to deviate from this plan the further into the development cycle the project got. However, because the project was using agile software methodology, it was not expected to stick closely to this plan and it was simply used as a guideline to make sure the project didn't fall too far behind and to make sure the main objectives were achieved. Figure 2 shows the original project plan and figure 3 shows the Gantt chart for the project.

Title	T/M	Start	End		%
Assessment 1 - Aims and Objectives	T	15/10/2019	19/10/2019	4 days	
Assessment 1 - Ethical Approval Form (EA1)	T	19/10/2019	24/10/2019	4 days	
Assessment 1 - Literature Review	T	24/10/2019	30/10/2019	5 days	
Assessment 1 - Project Plan and Risk Assessment	T	30/10/2019	12/11/2019	10 days	
Assessment 1 - Review of Progress	T	12/11/2019	19/11/2019	6 days	
Assessment 1 - Poster	T	19/11/2019	27/11/2019	7 days	
Development - Collect resources	T	28/12/2019	05/01/2020	5 days	
Development - Create OCR program to identify receipt in image	T	06/01/2020	20/01/2020	11 days	
Development - Collect important data from image. Disregard rest	T	20/01/2020	24/01/2020	5 days	
Development - Create database system to store data	T	24/01/2020	30/01/2020	5 days	
Development - Integrate OCR and database system	T	30/01/2020	08/02/2020	7 days	
Report - Create report for project	T	11/11/2019	02/03/2020	81 days	
Final Assessment	T	02/04/2020	02/04/2020	1 day	

Figure 2: Project Plan Table

Due to the COVID-19 pandemic, the deadlines for the project and final assessment demonstration changed from 2nd April to 23rd April and the closure of the university affected the duration of some sections, most notably the testing of the artefact. This meant that after March the project plan was no longer used for the project.

Figure 3: Gantt Chart

4c. Risk Assessment

Before starting design and development of a project it is important to consider the risks involved that could affect the output of the project or affect how it is completed. Below is a risk assessment table containing some possible risks involved for this project specifically. It outlines the potential risk, the impact it is likely to have and the probability of it happening.

Table 1: Risk Assessment Table

No.	Description of Risk	Risk Category	Impact	Probability	Overall Risk Value
		Time Risk? Resource Risk? Quality Risk?	How big is the impact of the risk? 1 = Low 5 = High	How Likely? 1 = Low 5 = High	Impact x Probability
1	Potential that receipts written in handwriting won't be correctly read by the OCR.	Quality Risk	3	4	12
2	Some words and phrases on receipts are in short hand/abbreviations	Quality Risk	2	5	10
3	To test the program I will need large amounts of testing data.	Resource Risk	4	1	4
4	Change of product requirements	Time Risk	5	2	10

Based on these potential risks, here are some ways that they can be mitigated to reduce the chance of them negatively effecting the project.

1) Many receipts from stores contain shorthand and abbreviations that aren't always immediately obvious as to what they mean. This can make it difficult to sort the information buy category once it has been input. This can be helped by asking the user to confirm the type of item the first time it is entered.

2) I will need a large amount of receipts of all types to test the program and make sure it works as expected. However, I cannot collect enough receipts myself and verify each one. Fortunately, other similar tools use receipts so I can use there sources and compare results to test my program.

3) Throughout the development of the artefact, there is a chance that I will edit/add requirements for the program if it will improve the quality. This could come with increase work and cause problems with the time given. Using the project plan I have allowed for some additional time in case anything changes in this regard.

4d. Software Development

When starting a new project it is important to choose a software development methodology that suits the project type and the people that are a part of the project. This is because it will define how the work is done and provides the overall structure that will guide the development team throughout the project life cycle. There are many types of management methodologies to choose from, but the main three that are used in software engineering are; Agile, SCRUM and Waterfall. Each of these has its own advantages and disadvantages so it is vital that these are taken into account in order to make the best decision before starting the project.

One popular methodology for software development projects is Waterfall project management, this is a relatively simple approach that follows a distinct and thorough plan set out at the start of the project and each stage of the development must be completed before moving onto the next. Aston (2019) makes the argument that this methodology is mostly suited to short projects with a larger number of people working on them. This is due to the rigid plan that does not make room for changes to the requirements or take into account feedback. This would be unsuitable for the project but the use of a detailed plan can be taken from this methodology and applied to others.

The project will likely have changes causing it to deviate from the original project plan, which will be made to improve the success and clarity of the project. Due to this, it is likely that an Agile approach for the completion of the project has many significant advantages. Aston (2019) calls the Agile methodology a "flexible, iterative design and build process" and suggests that this may be the best approach for any software development projects that need to be able to adapt to changes in the product requirements. Furthermore, an Agile approach opens up opportunities to use many different types of methodologies. For example, SCRUM and Extreme Programming (XP) are two types of Agile methods that are suited to small teams with short development cycles. In this case you can interchange features of each methodology that best suit the project, such as the small programming teams of XP mixed with the daily/weekly SCRUM meetings. Gibbins (2017) states that "Agile project management incorporates feedback with every iteration", which is an advantage that can benefit this project as the ability to make changes to the development process after feedback from the supervisor can improve the artefact for the next iteration.

4e. Toolsets and Machine Environments

The project outlined will require two software parts, an Optical Character Recognition program and a data storage system, so at least two different software development tools will be needed as well as project management tools that will be used for planning, designing and testing the system.

To develop the Optical Character Recognition and character analysis program there are several software tool options available to produce the artefact which have several advantages and disadvantages to be taken into consideration. Listed below are examples of these tools:

1. MATLAB

   MATLAB is a proprietary programming language that contains a huge library of predefined functions, these provide solutions to many primary technical tasks. The advantage of this program is that the OCR system would not need to be created from scratch and trained with potentially thousands of characters. This allows the focus to be on developing the artefact and on improving the quality of the output using methods such as imaging pre-processing, feature extraction and model estimation.

JavaTpoint (2018) explains MATLABs advantages and disadvantages, listed below are some of the most important examples:

Advantages:

- **Platform Independence** which is supported by most computer systems e.g. Windows, Linux, Unix and MacOS. Which allows programs written in MATLAB to be used across all of the above systems
- **Speed** of both compiling and running programs which is comparable to Python.
- **Built in Features** such as debugger and integrated development environment make it optimal for faster prototyping of new applications.
- **Execution** does not require a compiler compared to other programming languages such as C or C++, which can increase the efficiency of the program.

Disadvantages:

- **Conversion -** MATLAB can be converted to C++ or Python, making it a versatile language that can be implemented into other languages, however, this can be difficult and requires a deep level of knowledge of the language.
- **Executable -** The inability to create development files such as executables, this then only allows the program to be run through the MATLAB software.
- **Costs -** A licence for MATLAB is also expensive whereas other programming languages are free to use for everybody.

2. Python

Python is a high level programming language that uses an object-oriented paradigm that is useful for both small and large scale projects. Similarly to MATLAB, Python contains a huge number of libraries such as Tesseract which is an OCR library built for Python.

Techvidvan Team (2019) listed many advantages and disadvantages of Python, examples are listed below

Advantages:

- **Interpreted language** which means it executes code one line at a time and stops if an error is found. This can make debugging an easier and faster process.

- **Dynamically typed** this allows the system to automatically assign data types to variables when the program is executed.
- **Open-Source** this means that Python is free to use and distribute.

Disadvantages:

- **Speed** can be an issue because Python executes code line by line which creates extra work for the program.
- **Memory Inefficiency** because Python stores all objects and data structures in a private heap which the user has no control over as it is controlled by the interpreter itself.
- **Runtime Errors** occur due to the use of automatically assigned datatypes because an integer variable may get changed to a string, this can be avoided with thorough testing.

Below is a table comparing a few important features of the above software tools:

Table 2: Features Table

	MATLAB	**Python**
Library Support	Y	Y
Interactive development environment	Y	N
Costs	From £30	Open-source
Graphical features	Y	N*
Interpreter	Y	N
Dynamic variables	N	Y

**Library required*

After consideration of the advantages and disadvantages of each of the above, MATLAB was chosen as the software tool for developing the artefact because of its speed in handling large datasets. This is important as the artefact could be using hundreds or even thousands of images and storing the results from each.

For the data storage part of the artefact, there are two main tools that can be used for the best results. The most commonly used of these options is a spreadsheet (.xlsx file) and the other is a database system. Both have their own unique features that give them advantages and disadvantages based on the scenario and the type of data being stored.

1. Spreadsheet

A spreadsheet is the more basic system and is essentially an electronic version of an accounting worksheet that allows for data to be stored in a tabular form digitally. Below are listed some of the advantages and disadvantages of the system:

Advantages:

- **Accessibility –** Spreadsheet software such as Microsoft Excel is much more accessible for the average person and more people have had experience of how to operate the software compared to more involved systems.
- **Features -** Microsoft Excel has software features such cell-formulas and editable templates built in, which makes organising the data a quick and easy process without having to have extensive knowledge on how to use the software.
- **Storage –** Has the capability of storing a single sheet of 1,048,576 * 16,384 cells. This is more than enough for the purpose of this artefact.
- **Integration –** It is easier to implement a spreadsheet into programs that are developed in Python or MATLAB.

Disadvantages:

- **Data integrity –** Microsoft Excel does not provide any data integrity features so the inclusion of invalid or incorrect data is more likely as the size of the dataset grows.
- **Scale ability –** spreadsheets do not scale well for huge datasets likely to be used in business. However, is suitable for personal use.

2. Database

A database is an organised collection of data stored electronically. A database management system (DBMS) is a piece of software that allows users to interact with the data. The following shows the advantages and disadvantages:

Advantages:

- **Storage –** Provides a much better scaling for larger amounts of data and is well suited for storage of data over long periods of time. However, for most use-cases (i.e. personal use) of the artefact, a spreadsheet would have more than enough capacity so the better scaling of a database would be unnecessary.
- **Data integrity –** A database also features improved data integrity and data redundancy features. For example, data of different types cannot be entered in the same field. This can help reduce the amount of erroneous data being accepted into the system and reduce the overall amount of data being stored.
- **Sharing information –** Allows multiple users at one time. This enables more than one person to add or update information at the same time.

Disadvantages:

- **Complexity -** the use of a database system can be impractical because they can be complex, difficult to implement, time consuming to design, and training is often needed to use the system.
- **Costs –** costs increase as the amount of information stored is increased.
- **Maintenance –** overtime maintenance will be required which will incur costs and it will require someone trained in maintaining the database.

The above shows that benefits of a database are not needed for this artefact and a spreadsheet system using Microsoft Excel is the best option due to it being more user-friendly and more suitable for the most likely use-cases of the system.

Below is a table comparing a few important features of the above software tools:

Table 3: Features Table

	Excel	Database
Self-service (can be built by non-IT trained)	Y	N
Interactive visualisation	Y	N
Costs	Licence Fee	Various
Highly formatted	Y	Y
Scalable to large datasets	N	Y
Security	N	Y
Long-term support	Y	Y*

*Support is dependent

4f. GIT Version Control

For any software project it is important to keep different versions of the artefact saved in case there is an issue and you need to revert back to a previous version. For this project a GitHub repository was used to track changes made and keep an organised system of when and what was changed. Furthermore, it is important to keep an online back-up of any files for the project in case of unforeseen circumstances that may cause loss of files a physical machine being used for the project.

5. Research Methods

The two research methods used are qualitative and quantitative, the former being a source of non-statistical data often made up of views and opinions, this can be collected using questionnaires, interviews and focus groups. For this project observation is a more appropriate means to collect qualitative data. SkillsYouNeed (2011) outlines several ways qualitative data can be collected from a wide variety of sources and have a large variance in scope, therefore there are many ways to analyse the data. For example; grounded analysis which allows the data to 'speak for itself' from any point which has themes emerging from discussions, using this helps to come to a conclusion on the data to be analysed.

Quantitative data is made up of scientific data and is number based. It can be statistics or any quantifiable data. This can be collected by comparing the training and testing data sets to get an accuracy score. Analysing quantitative data can also be performed through several methods including summarising the data by creating graphs and charts. Another method is measuring skew, which is how a symmetrical training data set compares with the testing data set.

For the artefact a combination of both qualitative and quantitative research methods were appropriate to allow improved valuation and testing of the results. Quantitative research will be used for the artefact to obtain objective and observable data in the form accuracy scores and character confidence scores. These statistics take the form of interval data, which will be represented by a line graph that will clearly display the results of the Optical Character Recognition (OCR) system. The independent variable in this will be the accuracy scores obtained. Below is a sample of the first ten results. A full table and graph of the results are shown in Appendix 2 and 3.

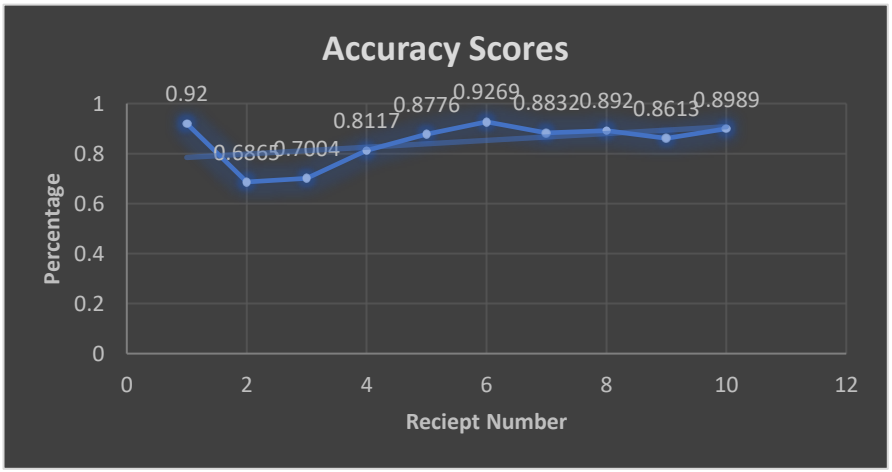

Figure 4: Sample graph of first 10 results

Qualitative research methods for the project involves displaying the detection results using output images. This allows for the images to be seen before and after pre-processing is completed. It will also show the character confidence scores that the Optical Character Recognition retrieves from reading the characters in the image.

Doing this allows for the comparison of different pre-processing methods for achieving the best results. From testing it was found that simply converting the image to grayscale and then imbinarizing meant that the results were consistent across a variety of images with different quality and clarity. This is important because an end user would then be able to use a majority of images for analysis, even if they have a low quality camera.

Figures 5 and 6 below show an example of the image pre-processing and the word boxes of found words. This a successful test showing 92% of words were found and identified correctly:

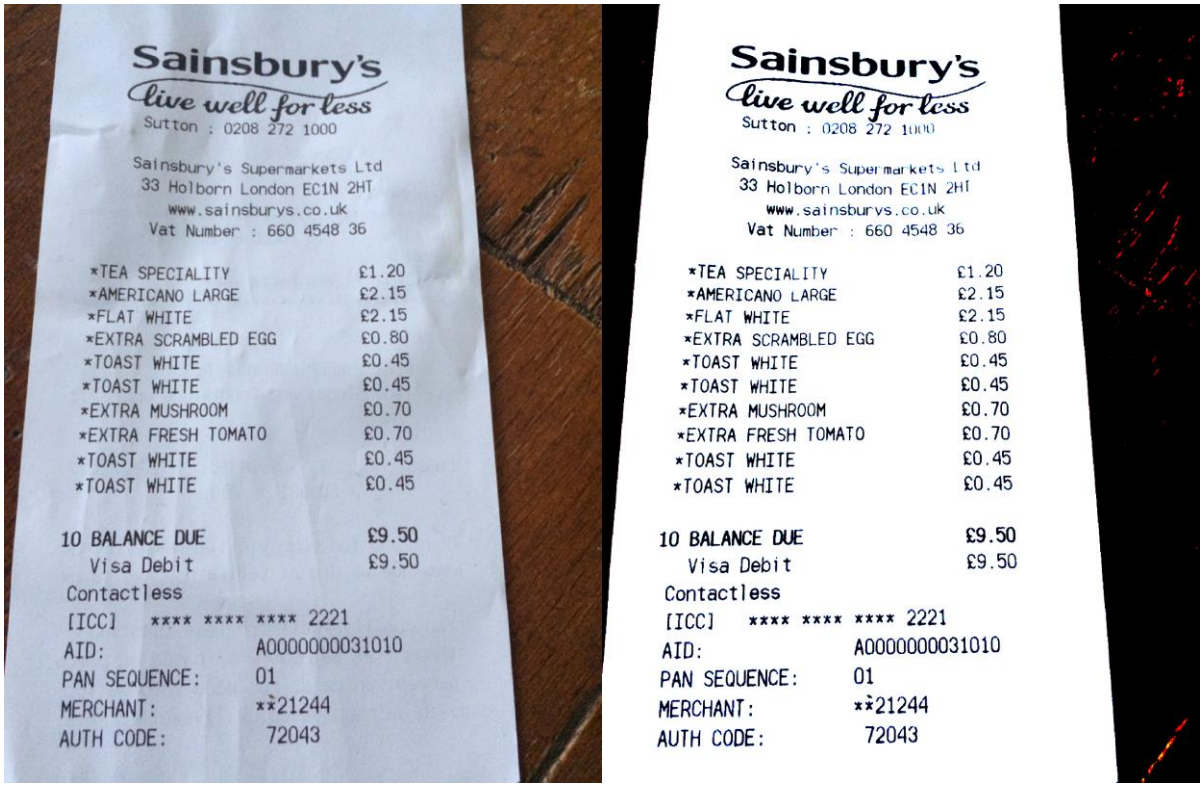

Figure 5: Receipt before and after imbinarization

Figure 6: Image annotated with word boxes

Unfortunately, in some instances the images are too low quality, there are several reasons for this such as:

- Low resolution
- Extremes in brightness or contrast
- Size image within photograph (picks up background noise)
- Damages to receipt

These examples reduce the OCR's ability to find characters in the image and can lead to poor results as shown in the illustrations below. Figure 7 demonstrate that images with dark shadows or badly creased negatively affect the imbinarization and lead to a lower ability to detect words as shown in figure 8:

Figure 7: Image before and after pre-processing

Figure 8: Image annotated with word boxes

6. Design, Development and Evaluation

6a. Requirements Gathering

Requirements gathering is part of the planning process in the software development cycle and determines the requirements that need to be met to complete the project objectives. The first step is to identify the user and document these needs and requirements. These can then be used to create project aims and objectives which can then be followed throughout the development of the project. Rastogi (2018) describes the importance of requirements gathering based on statistics of what causes a project failure. "As per PMI, about 70% project's failure is attributed to requirement collection. Also, this failure ranges from 50% to 70% depending on industry and type of project" (Rastogi 2018)

For this project the requirements for the system were gathered by consulting relevant literature to identify the current strengths and weaknesses of other OCR projects. This information was then used to construct objectives for how accurate and successful the system needed to be. Furthermore, discussion with the project supervisor assisted in identifying objectives specific to the use of the system by the user. For example, the addition of a separate data store to allow the user easy access to the information. Throughout the project an agile management methodology was used, which meant constant communication and feedback on the system throughout development. This feedback was often used to alter the requirements of the project to better suit the needs of the user and improve the final artefact.

The original requirements for this project are outlined in section 2a. It is important to ensure that these requirements are SMART (specific, measurable, agreed upon, realistic and time-based). This meant keeping the objectives clear in a way that they were measurable for when completed and fit into the project plan and project Gantt chart, so that the project stayed on schedule.

6b. Design

The following section encompasses the design section of the software development cycle. It is important to design the software before development and implementation so the program follows the requirements.

i. Optical Character Recognition and Image Pre-Processing

Optical Character Recognition (OCR), sometimes called Optical Character Reader is the conversion of text whether that be handwritten, typed or machine text. The image could be a photo, document or scanned image containing text.

For this project MATLAB's, OCR function will be used, this function has been trained using thousands of character images already and can be trained during development to be more accurate, especially for specific characters such as £ or $.

OCR works by converting the image into an array containing 1's and 0's. The 1's represent the character outline in the image, see figure 9:

Figure 9: Array example for character recognition

The program then learns the pattern of characters using the training data and can then detect the characters on an unseen image. It will then convert the new characters to an array and compare them to what it has learned to determine the closest match. The system will need to be retrained if the user is using a particularly unusual font because some letters look very different dependent on the font.

Image pre-processing methods such as converting to grayscale can make the array pattern standout and allow for easier recognition due to the high contrast between the text and the background.

To teach the OCR to detect special characters commonly seen on receipts, most notable currency symbols (£, $, €) and percentage symbol (%). This has to be done manually using MATLAB's OCR trainer. Using this you are able to see incorrect characters and correct them so that trainer can learn.

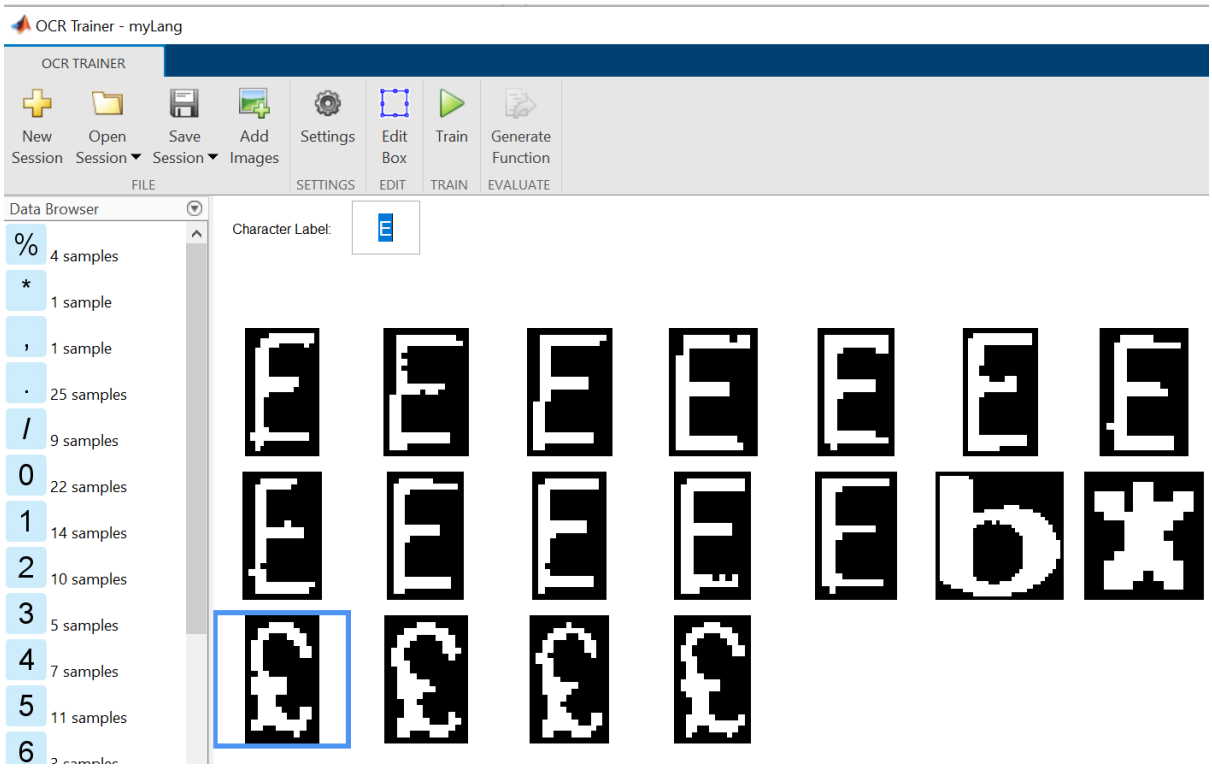

Figure 10: OCR trainer

From the example above, it can be seen that the OCR was detecting £ symbol as an E and was corrected to improve future recognition.

Character confidence scores are produced by the OCR system to present how confident it is that the character has been recognised correctly. Having this data shows the strengths and weaknesses based on which characters are confidently recognised. A low score does not necessarily mean it was incorrect but there is a higher chance of the character being read incorrectly. This data can also be used to measure precision.

The following image illustrates the confidence scores for whole words and costs on a receipt image. The words with a score lower than 0.6 on average have a higher chance of returning a false positive.

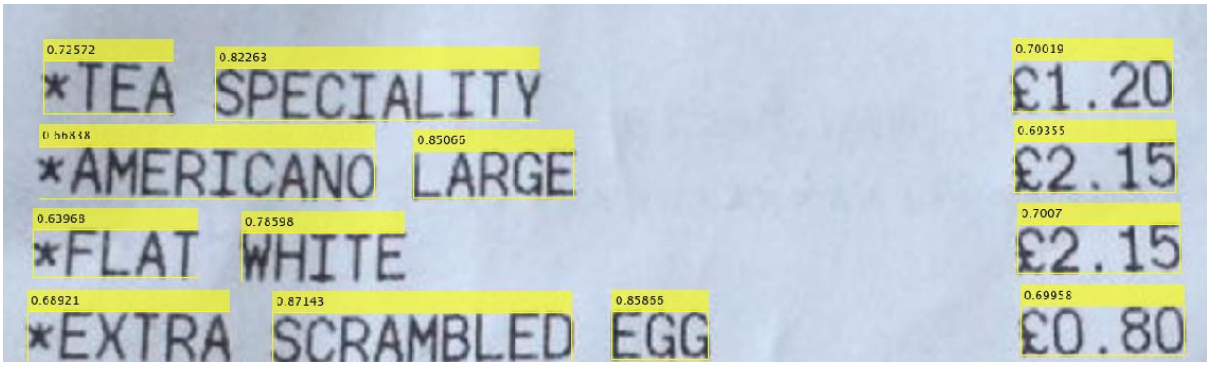

Figure 11: Character confidence screenshot

To improve the OCR's ability to read characters image pre-processing steps can be carried out. For this project a wide range of different images can be uploaded so no one method will be perfect for all of the images, so a few of the most consistent will be used for this project. Firstly, the image will be converted to grayscale, this allows for the image to be imbinarized, this then interprets an image as a volumetric grayscale image. Next, the image is imdilated, this is the process of dilating the greyscale image. These steps are the most consistent for making characters in the image more visible compared to the background which can improve readability for the OCR system.

These methods, however, can have a negative effect on some images which have dark shadows, poor lighting or thick creases and make readability worse. Despite this, this is still the best method of pre-processing for a majority of images. Detailed method of implementation will be discussed next section.

ii. Data Storage

As discussed in section 4e. Toolsets and Machine Environments, for this project a Microsoft Excel spreadsheet will be used to store the information gathered by the OCR system. To do this a .xlsx file template will be used, which will format the information in a readable way for the user. It will also allow for easy comparison between the ground truth and the test data gathered to get more accurate results.

The template needs to be formatted to enable specific cells to be used for different types data, this included cell sizing, splitting text, numbers and currency, data filters and data validations. The spreadsheet is then more user friendly and ready for use.

A ground truth Excel spreadsheet was also created which contains the correct characters in each cell that are in each image. This will be used for comparison in MATLAB to get the accuracy score.

6c. Development

The next stage of the software development cycle is the implementation/development of the artefact. In this stage the process of how the software was created from a development standpoint is outlined.

The first section that needed to be developed was image uploading and pre-processing. This needs done to be before Optical Character Recognition can be performed to get the best results. As mentioned previously, three steps need to be carried out for pre-processing. The first is converting the image to grayscale. The text will be black anyway so removing colour can improve readability of the writing as shown below:

Figure 12: Grayscale image

Next, the image is imbinarized. This has the benefit of making black text much more clear and can remove things such as creases in the paper of the receipt as shown below, this gives the text a bolder appearance which can drastically improve the OCR accuracy score. However, on some images it can have a negative effect if the image contains dark shadows or if these images is low resolution (see figure 13).

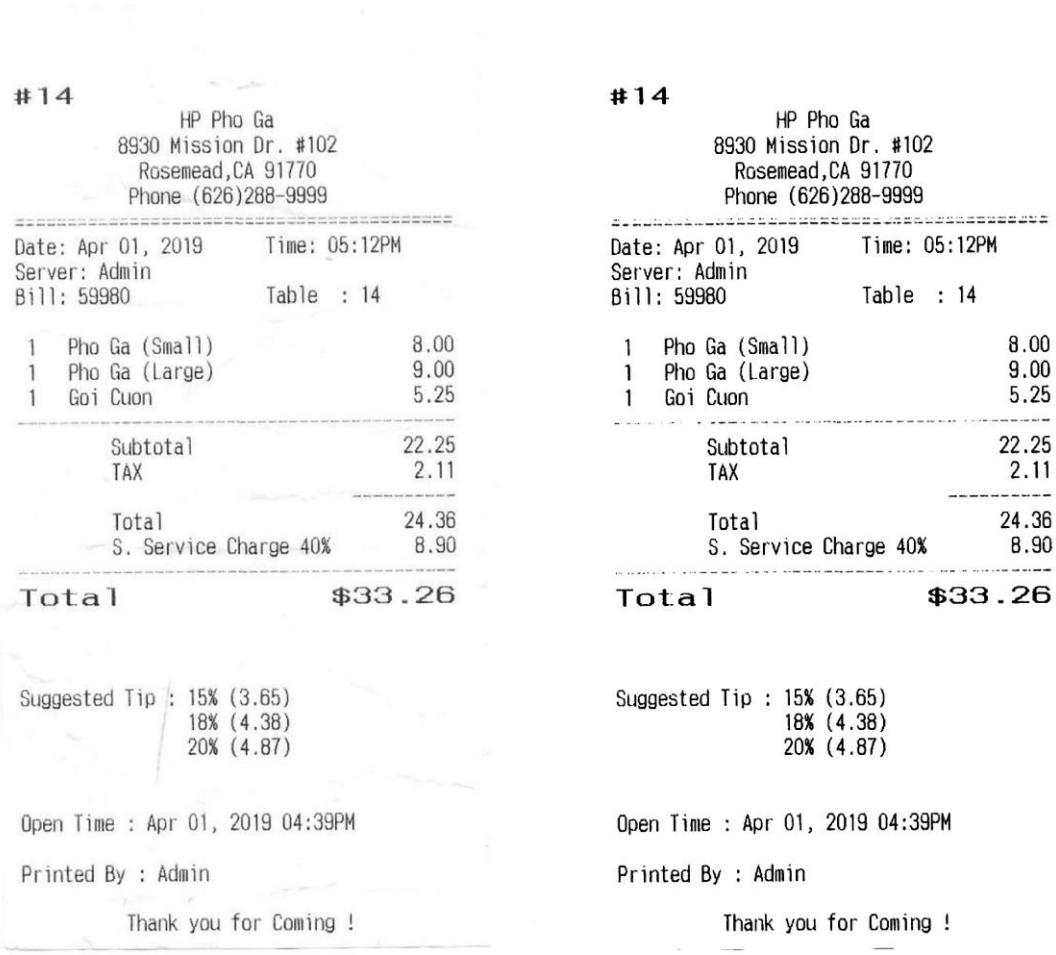

Figure 13: Imbinarized image

The final pre-processing step is to imdilate the grayscale image. This returns as a strel, a morphological structuring element, which stores the image in a multi-dimensional array containing binary values (1's and 0's). This allows the OCR to easily detect patterns in the array that match character patterns it has stored from training.

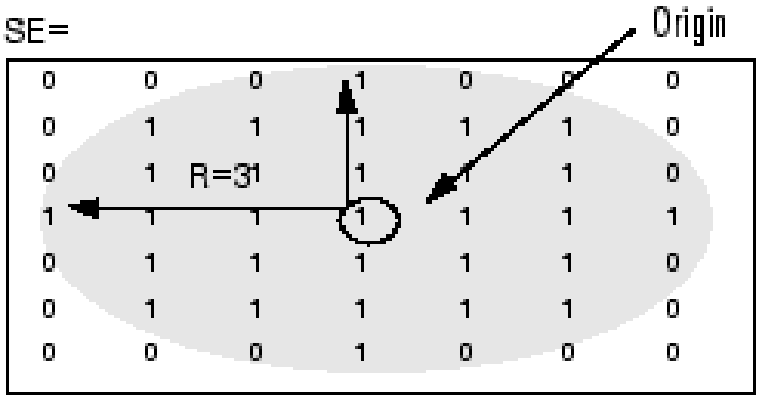

Figure 14: Strel element example

The array as one's representing black and zero's representing whites. As it is a grayscale image no other colours are present.

i. Optical Character Recognition

Once the images have been pre-processed OCR can be carried out. For this project MATLAB OCR function has been used in conjunction with OCR trainer that was discussed previously. The OCR trainer takes the uploaded images and returns the characters it has detected and if a character was read incorrectly it can be corrected to improve accuracy in the future, see figure 10. This was especially important for currency symbols that had very low accuracy scores beforehand. After training this increased from 5% accuracy to ~25% accuracy for £ symbols. Whilst still a low score this is a significant improvement though unfortunately it sometimes returns as a 3.

Once, OCR has been done on the image, Word Bounding Boxes are used to group up the characters into words. This is necessary to store information in full words instead of one character at a time, see figure 15.

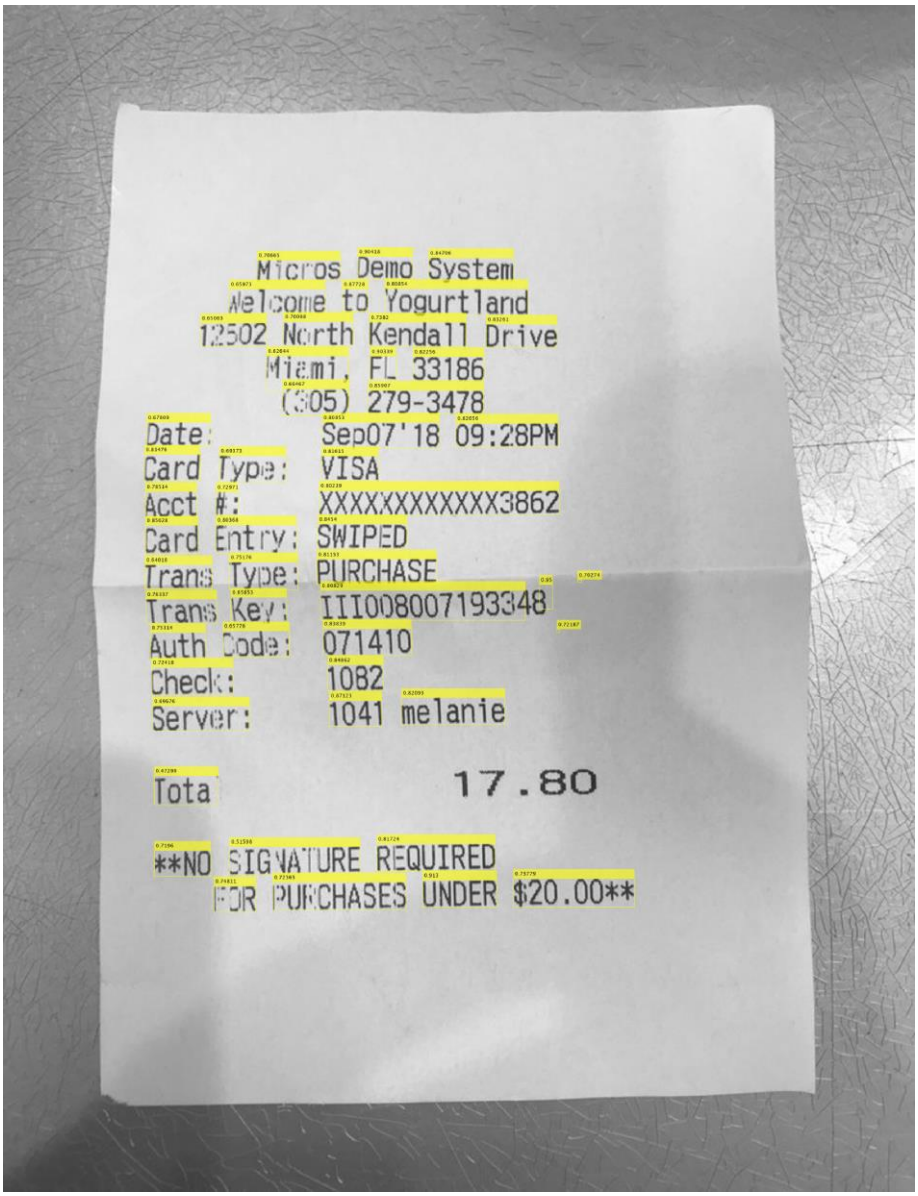

Figure 15: Word Bounding Boxes

ii. Data Storage

The information gathered by the OCR needs to be stored in a spreadsheet, reasons for this have been discussed in section 4e. Firstly, a master template needs to be created so that when the information is output it is formatted automatically. This is also to allow for easy comparison between test data and the ground truth for testing purposes.

Figure 16 shown on the right, is an example of the ground truth for one of the receipts used in testing in a Microsoft Excel Spreadsheet using the template. As you can see numbers and currency are separated and each cell contains one word. Some of the key formatting that was required are:

- Expanding cells to reduce truncated data.
- Usage of table formatting to ensure information is clear and readable.
- Automated currency symbol added to prices.

¢E:ag3§11az
CRADLEY
HEATH
345
6779890
NAPPIES
£3
0.75
NAPPIES
£3
0.75
NAPPIES
£3.75
DOUGHNUTS
£0.65
RED
WINE
X
£4.05
SUB-TOTAL
£15.95
MULTIBUY
SAVINGS
NAPPIES
3
FOR
£10
-1.25
TOTAL
SAVINGS
-1.25
TOTAL
TO
PAY
14.7
CASH
‘£20
0
CHANGE
DUE
£5.30

Figure 16: Ground Truth - sample

Data from the OCR is originally stored in cell arrays, each cell array contains another cell array with the characters from that specific image. Shown in figures 17 and 18 is an example of the cell array's being used to store OCR data. The amount of columns in the array is based how many words and numbers there are in that image.

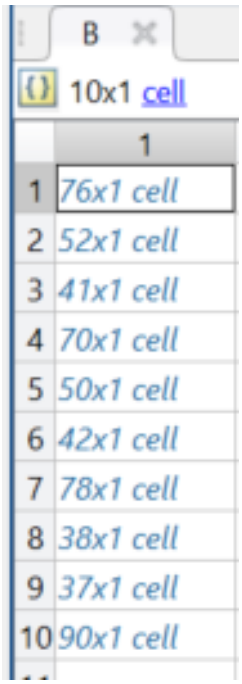

Figure 17: Cell array of cell arrays

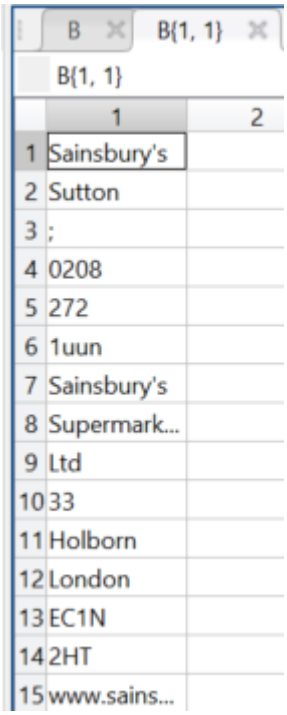

Figure 18: Array of OCR data

The information in these arrays needs to be output to a spreadsheet to make it easier for the user to read and use the information input. This is done in MATLAB using the code snippet in figure 19. To do this the amount of rows required for each receipt needs to be found using 'cellfun' function. Following this the information from each receipt can be written to .xlsx file. Again shown in figure 19, is the for loop used to scan through the total number of images and write them one image at a time to the file of chosen name.

For the purposes of testing each column of the output excel file needs to have the same number of rows because if the raw data had more rows that the ground truth it would negatively affect the accuracy of the accuracy scores. To alleviate this issue line 82 show in the image below was used to create a minimum of 100 rows for each column. More rows can be added if need be.

```
71    %Find number or rows in each cell array%
72 -  RowSizes = cellfun('length', B);
73 -  maxlen = max(RowSizes);
74
75 -  B1 = vertcat(B{:});
76
77    %Writing the data to a xlsx file. Each reciept is wirtten on new columns%
78    %To do this each column has to have the same number of rows%
79    %Fill unneeded rows with NaN value%
80 -  w = 'A':'Z';
81 -  for k = 1:numImages
82 -    range=[w(k) '1:' w(k) num2str(100)];
83 -    x=B{k, 1};
84 -    xlswrite(filenameWrite,x,range);
85 -  end
86
87
88    %Creating two tables too store the OCR data and the ground truth%
89 -  T = readtable(filenameWrite)
90 -  T2 = readtable(filename_groundtruth)
91
```

Figure 19: Code snippet output to Excel

iii. Development of Testing System

As discussed in section 5, two types of test have been carried out. The most important being quantitative which would show the accuracy score of the system. To get an accuracy score all that is required is to compare the raw data output by the Optical Character Recognition with the ground truth. Once the total number of words found correctly are added up, this is divided by the length in rows of data to get an accuracy as a percentage. Once this has been completed for all the images an average can be found across all the testing.

Figure 20 contains a code snippet of the project that shows how the accuracy score for each image in calculated. The raw data and the ground truth data are then put into tables called 'T1' and 'T2'. The length (number of rows) in each column is found and incremented by 1 to take into account the fact that it starts at zero. Then the 'setdiff' function in MATLAB is used to compare the two tables 1 row and column at a time, if the cells match they are stored in the variable 'Diff'. The accuracy scores are then added to a list and output for the user to see. A graph of the first ten results can be seen in figure 4.

```
96    %Create an array that will store the Accuracy score for each image%
97    %Filled with 0's to start%
98    AccuracyScoreList = zeros(25,1);
99
100   %Loop to find the difference between OCR data and ground truth for each image%
101   for i = 1:numImages
102       Diff = setdiff(T(:, i), T2(:, i));
103       result_length = size(T, i) + 1;
104       ground_truth_length = size(T2, i) + 1;
105       total_correct = 75 - size(Diff, 1);
106
107       %Adds accuracy score to list%
108       AccuracyScoreList(i) = total_correct/75;
109   end
110
111   AccuracyScoreList
```

Figure 20: Code snippet accuracy score

iv. Confidence Values

MATLAB's OCR function also returns confidence values which is how confident the OCR is that it identified the characters correctly. For example if a confidence score is >80 then it highly likely to be detected correctly and if it is a low score, <20, then it is highly likely it will be detected incorrectly. These confidence values are then displayed above the Word Bounding Boxes to give a visual representation of which parts are detected easier than others.

```
highConfidenceIdx = example.CharacterConfidences > 0.6;

% Find characters with low confidence.
lowConfidenceIdx = example.CharacterConfidences < 0.5;


% Get the bounding box locations of the low confidence characters.
lowConfBBoxes = example.CharacterBoundingBoxes(lowConfidenceIdx, :);

% Get confidence values.
lowConfVal = example.CharacterConfidences(lowConfidenceIdx);

% Annotate image with character confidences.
str       = sprintf('confidence = %f', lowConfVal);
Ilowconf = insertObjectAnnotation(colourReciept,'rectangle',lowConfBBoxes,str);

%figure; imshow(Ilowconf);
```

Figure 21: Confidence IDX code snippet

The code snippet shown in figure 21, is how the program detects and stores the confidence values. Firstly, two variables are created called 'highConfidenceIdx' and 'lowConfidenceIdx' that will store high and low confidence values respectively. In this case high confidence is considered to be above 0.6 and any lower is low confidence. After finding the confidence values they can be displayed along with Word Bounding Boxes to the user. The example in figure 21 shows the low confidence values and how they are displayed.

For testing purposes, an average confidence value is found and used to determine how well the OCR believes it has identified the characters in the image. Shown in figure 22 is a code snippet used to calculate the total number of words with a high confidence score and return an average for each individual image and an overall average. Furthermore, comparing these scores to the accuracy score of each word can help identify if the confidence values have any meaning, for example if confidence values are high but accuracy shows that it was still incorrect would suggest that the values have little meaning.

```
45 -        highConfidenceIdx = temp.CharacterConfidences >= 0.6;
46 -        lowConfidenceIdx = temp.CharacterConfidences < 0.6;
47
48 -        len = length(highConfidenceIdx);
49
50 -        for q = 1:len
51 -            if highConfidenceIdx(q) == 1
52 -                highConfidence = highConfidence + 1;
53 -            else
54 -                lowConfidence = lowConfidence + 1;
55 -            end
56 -            ConfidenceScoreList(i) = highConfidence;
57 -        end
58
59
60 -        ConfidencePercentageHigh = highConfidence/len;
61 -        ConfidencePercentageHigh = ConfidencePercentageHigh * 100;
62
63 -        ConfidencePercentageLow = 100 - ConfidencePercentageHigh;
64
65 -        ConfidenceAverageListHigh(i) = ConfidencePercentageHigh;
66 -        ConfidenceAverageListLow(i) = ConfidencePercentageLow;
67
```

Figure 22: Confidence Value code snippet

Confidence values are also linked with qualitative testing as they can be overlapped onto receipt images to get a visual representation of which

characters and words have high or low confidence scores. An example of this is shown in figure 15.

6d. Testing

Testing and integration is the fifth step in the software development cycle. Testing is vital to ensure that the software is working as intended as well as gathering data if it is a research project. For this project testing will be completed for both of these reasons.

For quantitative testing accuracy scores and confidence scores were used, each provided data on the success of the artefact. When looking at the confidence scores it is important to compare them to the accuracy for a specific image to get a picture of how the OCR thinks it is performing. If both accuracy and confidence scores are similar that would suggest that this system knows which characters or words it is likely to identify a true positives and true negatives. Whilst this does not improve the accuracy of the OCR it does inform us whether or not the predictions are likely to be correct. In table 4, shown below, are the scores as percentages for the first 10 images.

Table 4: Testing Scores Samples

Image No.	Accuracy Score (%)	High Confidence (%)	Low Confidence (%)
1	0.9200	82.9703	17.0297
2	0.6865	83.0028	16.9972
3	0.7004	78.3133	21.6867
4	0.8117	79.4582	20.5418
5	0.8776	81.9048	18.0952
6	0.9269	69.4561	30.5439
7	0.8832	79.2271	20.7729
8	0.8920	78.8546	21.1454
9	0.8613	82.9268	17.0732
10	0.8989	77.0909	22.9091

As shown the standard deviation between the accuracy scores and the high confidence scores is relatively low with the biggest difference being 26% and the lowest being 3%. So if the confidence score is higher than the accuracy score it means the OCR was over confident and reported more false positive character identification. If the confidence score is lower than the accuracy score it means the system was more accurate at identifying characters in the image than predicted. As the above table shows the confidence score is rarely higher that the accuracy score, this proves that the OCR is giving low confidence scores (<0.6) for characters it is able to

correctly recognise. It is important to also note that is some instances (for example image 19 in testing table appendix 2), the accuracy on words was relatively high, but most words were not detected at all. This gave an initial reading of high accuracy but when looking closer, it can be seen that the OCR did not see any characters there at all. This happens mostly when taking images of receipts that are heavily faded. A full testing table can be found in appendix 2.

6e. Operation and Maintenance

The final stage of the software development cycle is operation and maintenance. This section will continue for the lifetime of the software and is important to make sure the software works as intended at all times. It is also important that the end-user fully understands how to operate the software to get the best results.

Operation of the system requires basic knowledge of Microsoft Excel spreadsheets and the ability to upload images of the receipts. This software gives the user the ability to upload any of their own receipt images and the OCR will extract the text information required from the image automatically and output the data to the spreadsheet. From here it is to the user's discretion what they do with the information that has been extracted. The artefact uses preformatted templates to make operation of the software easy for the user so they do not require extensive knowledge or training.

Whilst using the software basic maintenance needs to be carried out, this includes updates to the software it is run on such as MATLAB and Microsoft Excel. Common maintenance also includes fixing any defects that negatively affect the running of the system.

7. Projection Conclusion

The research question posed in the introduction for this project was to determine if the results from an Optical Character Recognition (OCR) program can be used for real-life purposes such as personal or business use. We know that in perfect scenarios with high quality images and clear text an OCR program can achieve high accuracy scores but it is important to test the system using more realistic environments that the average person may have. This meant using a range of receipt images of different quality taken by the average person and using image processing to extract the best results.

After testing dozens of receipt images the accuracy scores of the OCR system were found and showed that images that are taken clearly and carefully can achieve an accuracy in the range of 80-95%, while poor quality images with low lighting achieve 60-70% accuracy even after image pre-processing. These results show that inconsistent image quality plays a huge effect on the OCR's ability to identify characters. This was also effected by a change to the original artefact design, specifically relating to image pre-processing. Originally, it was planned to have different methods of image pre-processing on each image but it was found that many methods such as eroding and reconstructing reduced the OCR accuracy more often than it improved it. Additionally, there are some anomalies such as receipt image number 12 where the OCR was not inaccurate but instead did not detect any characters within the image. This is shown in testing table in Appendix 2.

These results answered the question posed in this report. It is clear that inconsistency in the OCR results would make a system like this impractical for large datasets. This is because an error rate of 10-20% could mean thousands of errors that would need to be corrected, this would be time consuming and costly. However, when using OCR for small datasets it would be less of a problem because 20% could mean less than 100 total errors, meaning that the system is still suitable for personal use as some errors will not be too costly for the end task being carried out. Furthermore, this also extends to the storage of the information extracted by the OCR, storing data from thousands of images is doable in a spreadsheet but it is more suited to smaller datasets.

8. Reflective Analysis

In this section I will discuss the process of completing the project with a focus on which sections went well, which went badly and what I would do differently if given 20:20 hindsight.

A big part of this artefact was image pre-processing to improve the Optical Character Recognitions (OCR) success. Many pre-processing techniques proved to be unsuccessful and in some circumstances reduced the quality of the image rather than improving. A solution to this was found by using some more basic image pre-processing such as Binarization of the images, however, finding this solution set the project back and reduced the amount of time available to work on and improve the other parts of the artefact. Furthermore, gathering large amounts of receipts to use for testing was difficult and time consuming thus limiting the amount of testing that could be carried out on the system. Another issue that arose was the outputting of the data all together rather than separately. I solved this problem in MATLAB by converting data to tables first but this caused it to vary from the original design and the Microsoft Excel template design also had to be altered.

Despite these setbacks the OCR accuracy had a very high average accuracy score, consistently getting about 75% of the words in an image correct. The tests also worked flawlessly and allowed for easy comparison between the ground truth and the testing data.

The artefact also contains some bugs where I have not be able to identify the source, for example one image (12) failed to detect the words despite the use of high quality image with clear text. In this case it was not that the OCR was incorrectly characterising the information but instead it could not see any text in the image.

If time had not been a constraint there are a few features I would have added to the artefact to improve it. The main one being the ability to use different image pre-processing techniques on each image, getting results from the OCR and reporting the best results. This is because some pre-processing works well on a small number of images but not on many others. Having the ability to use the best methods for each unique image rather than the same methods for every image would improve the average accuracy score across all images. Given more time I would also take the opportunity to gather more receipts and test the artefact more thoroughly, though due to the uncontrollable circumstances surrounding the Coronavirus outbreak collection of test material was impeded. After testing there was still some bugs in the artefact that effects a small number of receipts, these include outputting the text out of order on images with

higher number of characters and one image where the OCR is unable to detect any characters before and after image pre-processing. However, due to these time constraints and the fact they only effected a very small number of images, they were not a priority when developing the artefact.

For the duration of the project an Agile approach to project management was taken, in theory this meant regular meetings with the end-user or in this case the project supervisor. In reality this was done but due to unforeseen circumstances I was unable to stick with the Agile methodology at all times. Even so, Agile allowed me to improve the artefact during development by updating requirements and adding new features.

9. References

Richter, M. (2019). Digitize your Receipts using Computer Vision – inovex Blog. [online] inovex-Blog. Available at: https://www.inovex.de/blog/digitize-receipts-computer-vision/ [Accessed 3 Dec. 2019].

Anh Duc Le, Dung Van Pham, Tuan Anh Nguyen. (2019). [online] Available at: https://arxiv.org/pdf/1905.12817.pdf [Accessed 3 Dec. 2019].

Artjoms Suponenkovs, Aleksandrs Sisojevs, Guntis Mosāns, Jānis Kampars, Krišjānis Pinka, Jānis Grabis, Audris Locmelis, Romāns Taranovs. (2017). Application of image recognition and machine learning technologies for payment data processing review and challenges - IEEE Conference Publication. [online] Available at: https://ieeexplore.ieee.org/document/8270536 [Accessed 3 Dec. 2019].

Hansen, J. (2018). A Matlab Project in Optical Character Recognition (OCR). [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.657&rep=rep1&type=pdf [Accessed 3 Dec. 2019].

Aston, B., 2019. 9 Of The Most Popular Project Management Methodologies Made Simple - The Digital Project Manager. [online] The Digital Project Manager. Available at: <https://thedigitalprojectmanager.com/project-management-methodologies-made-simple/#scrum> [Accessed 7 April 2020].

Mark Gibbins, S., 2017. [online] Repository.cardiffmet.ac.uk. Available at: <https://repository.cardiffmet.ac.uk/bitstream/handle/10369/8747/GIBBINS%2C%20Samuel%20Dissertation.pdf?sequence=1&isAllowed=y> [Accessed 7 April 2020].

www.javatpoint.com. 2018. Advantages And Disadvantages Of MATLAB - Javatpoint. [online] Available at: <https://www.javatpoint.com/advantages-and-disadvantages-of-matlab> [Accessed 7 April 2020].

Team, T., 2019. Python Advantages And Disadvantages - Step In The Right Direction - Techvidvan. [online] TechVidvan. Available at: <https://techvidvan.com/tutorials/python-advantages-and-disadvantages/> [Accessed 7 April 2020].

2011, (., 2020. Quantitative And Qualitative Research Methods | Skillsyouneed. [online] Skillsyouneed.com. Available at: <https://www.skillsyouneed.com/learn/quantitative-and-qualitative.html> [Accessed 7 April 2020].

Rastogi, A., 2018. Greycampus. [online] Greycampus.com. Available at: <https://www.greycampus.com/blog/project-management/project-management-how-to-collect-requirements-for-your-project-effectively> [Accessed 7 April 2020].

Cs.princeton.edu. (2019). [online] Available at: https://www.cs.princeton.edu/courses/archive/spring18/cos598B/public/projects/System/COS598B_spr2018_ReceiptParsing.pdf [Accessed 3 Dec. 2019].

Yue, A., 2020. [online] Cs.princeton.edu. Available at: <https://www.cs.princeton.edu/courses/archive/spring18/cos598B/public/projects/System/COS598B_spr2018_ReceiptParsing.pdf> [Accessed 7 April 2020].

OpenLearn. 2020. *Understanding Different Research Perspectives*. [online] Available at: <https://www.open.edu/openlearn/money-management/understanding-different-research-perspectives/content-section-8> [Accessed 7 April 2020].

2020. [online] Available at: <https://www.researchgate.net/profile/Samer_Sarsam/post/What_is_the_Matlab_OCR_Feature_Extraction_and_Classification_Algorithms/attachment/59d6408f79197b807799cbed/AS:431507207659522@1479890743719/download/10.1.1.466.657.pdf> [Accessed 7 April 2020].

Appendix

Appendix 1

Artefact Code

```
%Project%
%Author: Benjamin Terrill%
%Students ID: 15622143%
%Date: 31/10/2019%

%Import Data%

numImages = 10;
A = cell(numImages, 1);
B = cell(numImages, 1);

%Creating variables for the names of files that need to be writen to or loaded in%

filenameWrite = "RawData.xlsx";
filename_template = 'template.xlsx';
filename_groundtruth = 'GroundTruth.xlsx';
%filename_groundtruth = 'testTruth.xlsx';
copyfile(filename_template, filenameWrite);

%imagefiles = imread('images/im1.jpg');
ConfidenceScoreList = zeros(10,1);
ConfidenceAverageListHigh = zeros(10,1);
ConfidenceAverageListLow = zeros(10,1);
highConfidence = 0;
lowConfidence = 0;

for i = 1:numImages
    %Loading in the next image%
    filename = "images/im (" + i + ").jpg";
    loadim = imread(filename);
    colourReciept = loadim;
    colourReciept = imresize (colourReciept, 3);

    %Detecting text in the image and image pre-processing%
    greyReciept = rgb2gray(colourReciept);
    BWReciept = imbinarize(colourReciept);
    BWReciept = imdilate(BWReciept, strel('square',1));


    %Storing the OCR results%
    A{i} = ocr(BWReciept);
    B{i} = A{i, 1}.Words;

    temp = ocr(BWReciept);
    % Find characters with low and high confidence.
    highConfidenceIdx = temp.CharacterConfidences >= 0.6;
    lowConfidenceIdx = temp.CharacterConfidences < 0.6;

    len = length(highConfidenceIdx);

    for q = 1:len
        if highConfidenceIdx(q) == 1
            highConfidence = highConfidence + 1;
        else
            lowConfidence = lowConfidence + 1;
```

```
        end
        ConfidenceScoreList(i) = highConfidence;
    end


    ConfidencePercentageHigh = highConfidence/len;
    ConfidencePercentageHigh = ConfidencePercentageHigh * 100;

    ConfidencePercentageLow = 100 - ConfidencePercentageHigh;

    ConfidenceAverageListHigh(i) = ConfidencePercentageHigh;
    ConfidenceAverageListLow(i) = ConfidencePercentageLow;

    highConfidence = 0;


    %Displaying example image after image pre-processing%
    if i == 2
        %figure; imshowpair(colourReciept, BWReciept,'montage');

        example = ocr(BWReciept)

        wordBox = example.WordBoundingBoxes(:,:);     %Creating a box around a word
 in image%
        figure;
        wordDisplay = insertObjectAnnotation(greyReciept, 'rectangle', wordBox, exa
mple.WordConfidences); %Displays the character confidences next to each word%
        imshow(wordDisplay);


        highConfidenceIdxtest = example.CharacterConfidences > 0.6;

        % Find characters with low confidence.
        lowConfidenceIdxtest = example.CharacterConfidences < 0.5;


        % Get the bounding box locations of the low confidence characters.
        lowConfBBoxes = example.CharacterBoundingBoxes(lowConfidenceIdxtest, :);

        % Get confidence values.
        lowConfVal = example.CharacterConfidences(lowConfidenceIdxtest);

        % Annotate image with character confidences.
        str      = sprintf('confidence = %f', lowConfVal);
        Ilowconf = insertObjectAnnotation(colourReciept,'rectangle',lowConfBBoxes,s
tr);

        %figure; imshow(Ilowconf);
           end
       end


       %Find number or rows in each cell array%
       RowSizes = cellfun('length', B);
       maxlen = max(RowSizes);

       B1 = vertcat(B{:});

       %Writing the data to a xlsx file. Each reciept is wirtten on new columns%
       %To do this each column has to have the same number of rows%
       %Fill unneeded rows with NaN value%
       w = 'A':'Z';
       for k = 1:numImages
         range=[w(k) '1:' w(k) num2str(100)];
         x=B{k, 1};
```

```
117.          xlswrite(filenameWrite,x,range);
118.        end
119.
120.
121.        %Creating two tables too store the OCR data and the ground truth%
122.        T = readtable(filenameWrite)
123.        T2 = readtable(filename_groundtruth)
124.
125.
126.        %Compare the two tables%
127.        %Diff = setdiff(T(:, 2), T2(:, 2))
128.
129.        %Create an array that will store the Accuracy score for each image%
130.        %Filled with 0's to start%
131.        AccuracyScoreList = zeros(25,1);
132.
133.        %Loop to find the difference between OCR data and ground truth for each imag
    e%
134.        for i = 1:numImages
135.            Diff = setdiff(T(:, i), T2(:, i));
136.            result_length = size(A{i, 1}.Words);
137.            ground_truth_length = size(T2, i) + 1;
138.            total_correct = result_length - size(Diff, 1);
139.
140.            %Adds accuracy score to list%
141.            AccuracyScoreList(i) = total_correct/result_length;
142.        end
143.
144.        AccuracyScoreList
145.        ConfidenceAverageListHigh
146.        ConfidenceAverageListLow
```

Appendix 2

Testing Table

Image No.	Accuracy Score (%)	High Confidence (%)	Low Confidence (%)
1	0.9200	82.9703	17.0297
2	0.6865	83.0028	16.9972
3	0.7004	78.3133	21.6867
4	0.8117	79.4582	20.5418
5	0.8776	81.9048	18.0952
6	0.9269	69.4561	30.5439
7	0.8832	79.2271	20.7729
8	0.8920	78.8546	21.1454
9	0.8613	82.9268	17.0732
10	0.8989	77.0909	22.9091
11	0.7798	78.2278	21.7722
12	0.0000	70.4057	29.5943
13	0.8405	79.2013	20.7987
14	0.8988	76.7391	23.2609
15	0.7369	76.2208	23.7792
16	0.6987	79.3103	20.6897
17	0.7120	80.8989	19.1011
18	0.9558	76.3780	23.6220
19	0.6504	76.7974	23.2026
20	0.8923	75.6024	24.3976
21	0.7417	72.8785	27.1215
22	0.6861	75.2621	24.7379
23	0.5934	76.1488	23.8512
24	0.9375	79.2079	20.7921

Appendix 3

Graphs

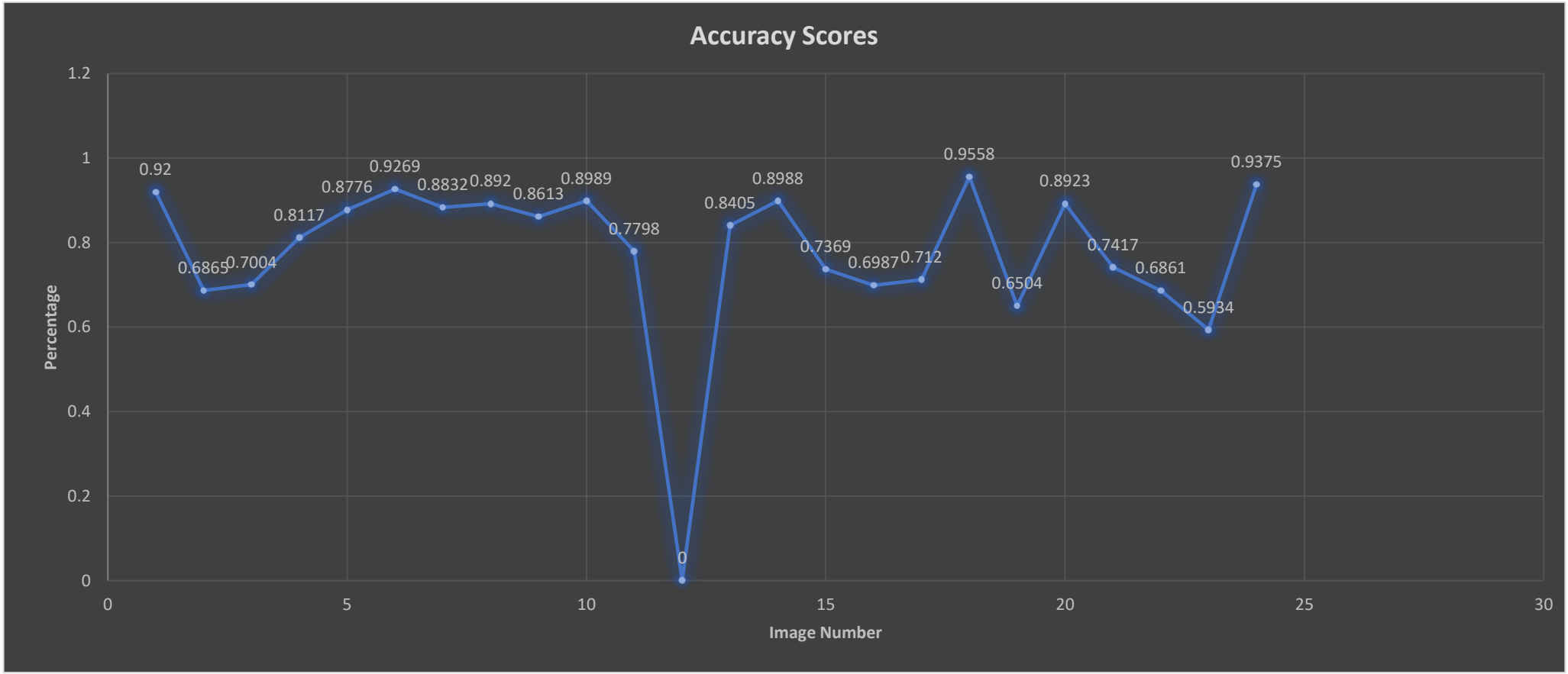

Figure 23: Accuracy Scores

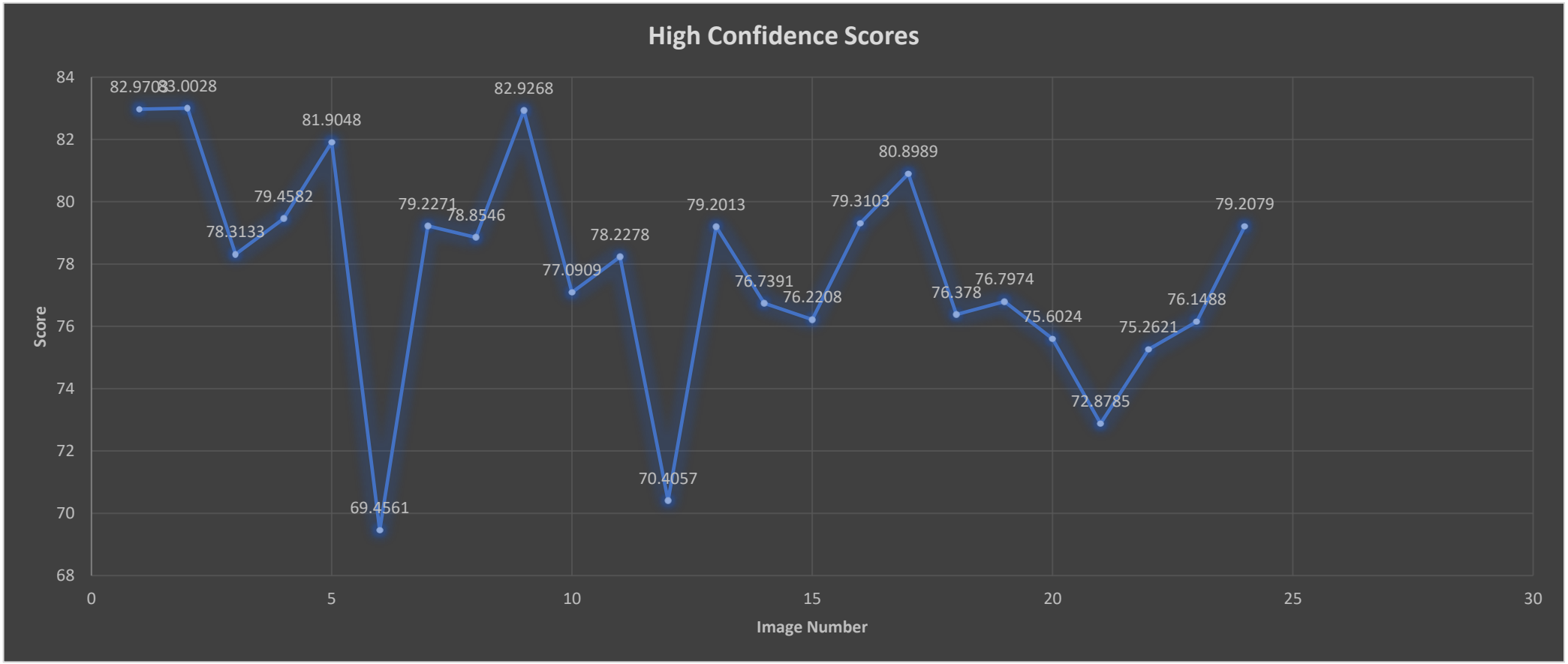

Figure 24: High Confidence Scores

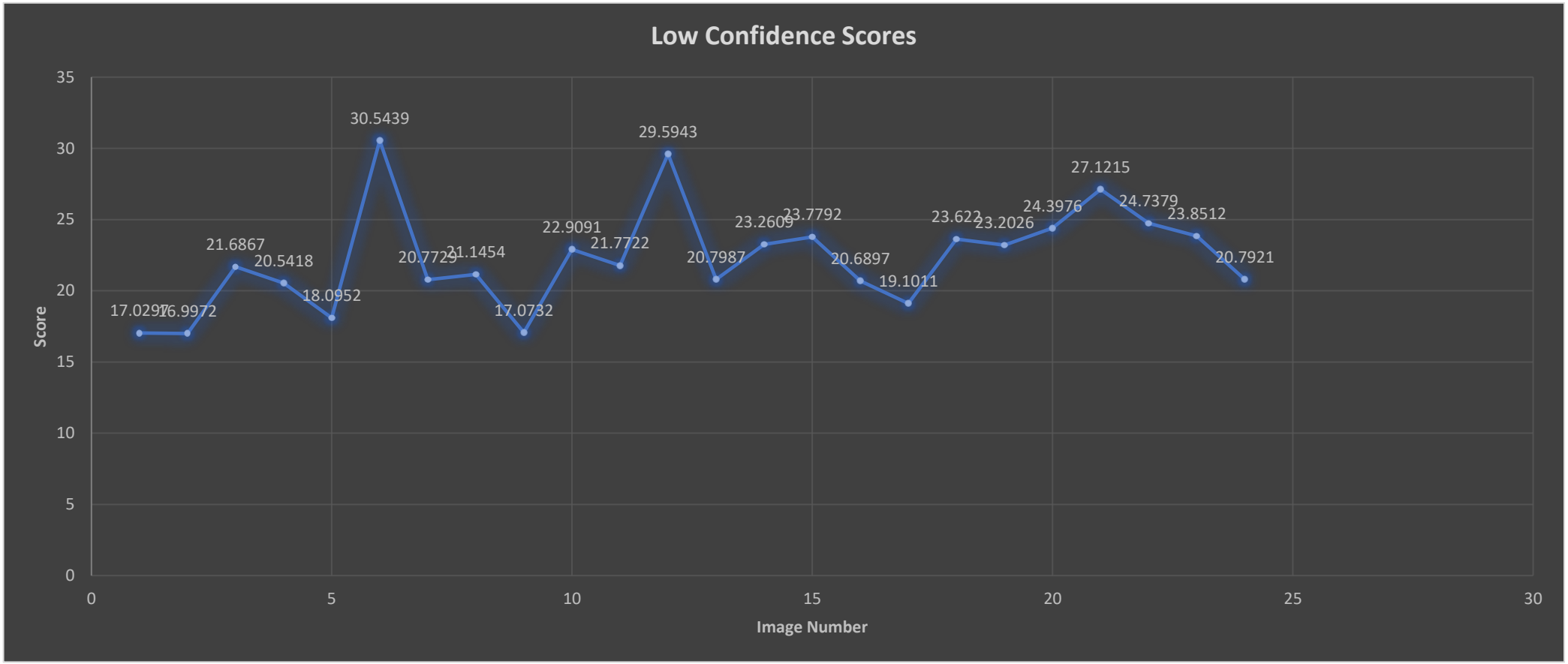

Figure 25: Low Confidence Scores