

FORMS PROCESSING CHARACTER RECOGNITION METHODS

Rhysian Jones - 1809710

A report submitted to the University of Wales in partial fulfilment of the
requirements for the degree of Master of Software Engineering

Faculty of Architecture, Computing and Engineering
University of Wales Trinity Saint David



January 2020

DECLARATION:

I, Rhysian Jones, declare that I am the sole author of this project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: *RwJones* Date: 01/01/2020

Form of Consent

I, Rhysian Jones, hereby consent that my project, submitted in candidature for the MSc Software Engineering and Web Development degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: *RwJones* Date: 01/01/2020

ACKNOWLEDGEMENTS:

Project supervisor: Dr Michael Dacey

DVLA technical lead: Mr Gareth Edwards

FORMS PROCESSING CHARACTER RECOGNITION METHODS

RHYSIAN JONES – 1809710

1) ABSTRACT:

This work investigates different methods of feature extraction for character recognition. The aim of the project was to develop and compare two feature extraction algorithms. A range of literature was reviewed on topics pertaining to forms processing in general as well as more specific aspects of forms processing such as optical character recognition, segmentation and feature extraction. The information gathered by the literature review was used to design two feature extraction algorithms as well as the character recognition system that implements them. This developed system was used in an experiment to compare the accuracy of the two algorithms.

The data used to build the matching class consisted of roughly thirty to forty handwritten characters which were analysed and averaged and the test data consisted of twenty-two handwritten samples of the phrase “The quick brown fox jumps over the lazy dog”.

The results from the comparison of the two algorithms showed that the algorithm that did not utilise zones had a higher accuracy score, but neither algorithm was particularly effective. Further analysis of the results revealed some probable causes of the general inaccuracy of both algorithms, and this information was used to develop a third, combined, algorithm. Preliminary testing for this combined algorithm showed a vast improvement over its predecessors in the early stages.

CONTENTS

1) ABSTRACT:	3
2) INTRODUCTION:	6
3) LITERATURE REVIEW:	9
3.1) FORMS PROCESSING	9
3.2) RECOGNITION METHODS	10
3.3) EXISTING SOFTWARE	11
3.4) ALGORITHMS	11
3.5) EVALUATION OF SEGMENTATION ALGORITHMS	12
3.6) EVALUATION OF FEATURE DETECTION ALGORITHMS	15
3.7) MATCHING	17
3.8) MACHINE LEARNING	18
3.9) COVNET	20
3.10) CASCADED CLASSIFIERS	21
3.11) EVALUATION OF AVAILABLE SOFTWARE	21
3.12) PROPOSED OVERVIEW OF THE SOFTWARE	23
3.13) LITERATURE REVIEW CONCLUSION	24
4) PROJECT PLANNING:	25

5) RESEARCH METHODOLOGY:	26
5.1) PHILOSOPHY	26
5.2) APPROACH	28
5.3) RESEARCH PROCESS	29
5.4) RISKS AND LIMITATIONS	30
6) SOFTWARE DEVELOPMENT METHODOLOGY:	31
7) SOFTWARE DESIGN:	33
7.1) REQUIREMENTS	34
7.2) DESIGN	35
8) SOFTWARE IMPLEMENTATION:	36
8.1) TESTING	36
8.2) DEVELOPMENT	37
8.3) SOFTWARE OVERVIEW	40
9) EXPERIMENT:	41
10) RESULTS:	42
10.1) MAIN EXPERIMENT	42
10.2) COMBINED ALGORITHM	44
11) DISCUSSION:	46
11.1) ALGORITHM 2	46
11.2) ALGORITHM 1	48
11.3) COMBINED ALGORITHM DISCUSSION	49
12) CONCLUSION:	50
13) REFLECTION:	50
14) FURTHER WORK:	51
REFERENCES:	53
APPENDICES:	58
Appendix 1 - Raw results	58
Appendix 2 – Match data	59
Appendix 3 - Scores	59
Appendix 4 – T-test	60
Appendix 5 – Combined Algorithm Code	60

FIGURES

Figure 1 - Word Cloud Preliminary Research	9
Figure 2 - Levels of Recognition (N=102)[1]	10
Figure 3 - Benefits of ICR (N=25)[1]	10
Figure 4 - Binary Image. Red Lines = Cross Section Sequences[27]	13
Figure 5 - 100 Has Been Separated[27]	13
Figure 6 - 14° vs 8° Water Flow[28]	13
Figure 7 - Roger with Connected Nodes[30]	14
Figure 8 - Character Adaptive Mask[32]	14
Figure 9 - Row Segmentation[33]	15

Figure 10 - Character and Word Segmentation[33]	15
Figure 11 - Character Zones[35]	16
Figure 12 - Example of Topology Differentiating Characters[35].....	16
Figure 13 - Key Points[36].....	16
Figure 14 - Approximate Curve in Topological Representation[36].....	17
Figure 15 - Edge Walking[37].....	17
Figure 16 - The Green Hyperplane Separates The Two Groups By A Narrow Margin	19
Figure 17 - The Green Hyperplane Cuts Off Part Of The H While The Red Hyperplane Properly Separates The Characters	19
Figure 18 - Hard Margin.....	20
Figure 19 - Cascading Classifiers, Input And Three Stages	21
Figure 20 - Example Kanban Board.....	32
Figure 21 - Flow Diagram Of Overall System.....	35
Figure 22- Class Diagram	36
Figure 23 - Example of Methods	37
Figure 24- Example of Tests	37
Figure 25 - Example Test Data	41
Figure 26 - Example Results Table	42
Figure 27- Algorithm 1 Total Character Scores.....	43
Figure 28 - Algorithm 2 Total Character Scores.....	44
Figure 29 - Flow Diagram Of Combined Algorithm	45
Figure 30 - Combined Algorithm Results	46
Figure 31 - Raw Results Example A2	46
Figure 32 - Results Extract.....	46
Figure 33 - X and k comparison	47
Figure 34 - Results Extract.....	47
Figure 35 - Raw Results Example A1	48
Figure 36 - Results Extract.....	48
Figure 37 - Results Extract.....	48

2) INTRODUCTION:

This study is an investigation into existing methods of forms processing. A literature review will be carried out in order to investigate background information and any prior research which might aid in forming a plan for the final project. The aims of the research will be discussed along with the strategies to be implemented at each stage.

The area of study being investigated is forms processing. Forms processing is essential for any organisation that deals with large amounts of data being collected via physical documents. By facilitating the quick and accurate extraction of data from physical forms, the efficiency of data processing can easily be greatly improved.

Forms processing is a well-researched area with a decades-long history and many popular applications and is ever-changing and improving. There are many different methods for different applications and the differences in languages present a wide array of challenges (such as the difficulty in separating handwritten characters in languages which are written in cursive and with overlaps or languages containing accents). In a field with a large number of techniques and algorithms in place for many different circumstances, it is useful to compare some of these to assess their function and usefulness in a specific case. This will allow for a tailored system which should provide the best possible results for the organisation. The case being taken into account in this study will be the DVLA's (Driver and Vehicle Licensing Agency) forms processing. Some unique traits of this organisation are listed below:

- Language: English(primarily), Welsh
- Accessibility: All forms are available in large print
- Identifiers: All forms must include the DVLA and .gov icons.

These traits offer some challenge as while English processing has been widely studied, Welsh has not. While Welsh has no additional characters, it contains accents which don't appear in English and must be accounted for as well as the obvious challenge of different words if word recognition is in use. The accessibility options offer an additional challenge as the forms will not all be identical as they would be if no options were offered. The identifiers offer a helping hand. Since these always appear and always look the same, they can be used to line up the form. For example, if the logo is upside-down, the form is probably also upside-down. If the logo is slanted, the form is probably also slanted. This information can be used to perform corrections before any data is actually processed which should improve the accuracy of the process.

As the DVLA handles hundreds of thousands of forms and applications, any increase in efficiency or accuracy should make a difference to the work of the organisation.

Forms come in a variety of types. For example, you could be asked to fill in a form electronically and then print it out to be submitted which means that all fields will be typed, you

could be asked to colour in circles or you could be asked to fill in details using printed capitals. The different types of form fields can be categorised as follows: check marks, barcodes, magnetic ink, optical characters (typed), optical characters (handwritten, printed) and optical characters (handwritten, cursive).

Magnetic ink is used on cheques and uses two specific fonts consisting of only numbers and delimiters to display information such as bank account numbers. The characters are printed in magnetic ink which, when scanned, produces a distinct waveform which is used to identify the character. This system is efficient and effective but only suitable for identifiers and not user-entered fields.

Barcodes are similar in that they are used as identifiers but the main difference is that magnetic ink is human-readable while barcodes are only machine-readable. This means that barcodes are not suitable for user-entered fields.

Check marks are user-enterable and represent a choice between a set of distinct options. The indicated area (usually a box, circle or brackets) must be clearly marked by either colouring in the whole area or by entering the appropriate symbol such as a tick or cross. When the form is scanned, the fields that are marked are recorded while the empty fields are ignored. No information other than the choice of field can be captured which makes this a limited system.

Typed characters are processed using optical character recognition. The fields are scanned and the system can easily match the characters to characters of known fonts. This system is user-enterable and more informative than check marks but has the drawback of introducing extra steps to a process and being less user-friendly than other methods. If a computer is available to fill in the form then in most cases, the form could be sent electronically and would not need to be processed as the information could be read straight from the form into a database. This system is only useful in edge cases where forms cannot be sent electronically.

Handwritten printed characters are processed using optical character recognition much like typed characters are. The difference is that the handwritten characters cannot be directly matched with a stored character from a known font. This means that different methods of identifying a character must be explored. This is part of what will be investigated in this work.

Handwritten cursive characters are processed using intelligent character recognition. This is more complex than processing printed characters because the character need to be separated in order to identify them individually. There are many methods of doing this which will be investigated in this work.

In the DVLA, forms can contain printed and cursive fields so the focus of this work is the different methods of handwriting recognition. The processing of cursive text will be investigated in order to gain an understanding of the steps involved and the different algorithms available. This

understanding will be used to develop a basic system to compare two methods and compare their performance.

The main aim of this project is to investigate different techniques for recognising handwritten characters. This aim can be broken down into the following objectives:

- Investigate different techniques
- Develop two contrasting techniques
- Compare the two contrasting techniques

The final product should be a system for reading the text in images and information about the effectiveness of the system.

This introduction has outlined the context of this study as well as some key terms and overarching themes that will be discussed further in later sections.

The next section, the literature review, will be a thorough investigation of key themes as well as available methods and software. Existing algorithms will be evaluated for use in this project resulting in an outlined plan of action.

3) LITERATURE REVIEW:

The purpose of this literature review is to investigate forms processing and the methods and algorithms involved. The key themes already identified in preliminary research are: forms processing, optical character recognition and intelligent character recognition. These key terms will be used to inform the research carried out in this section.

By the end of this review, the proposed project should have a clear direction based on existing research.



Figure 1 - Word Cloud Preliminary Research

3.1) FORMS PROCESSING

Forms processing is the process of converting a physical form (for example, a student finance paper application) into an electronic format (for example, a database) for easier data handling[1]. The most basic method in terms of technology is manual data entry where a human will read the form and type the data into the appropriate fields. This takes a long time and is prone to human error such as typos[2].

Automated forms processing aims to reduce the time, effort, cost and errors associated with the process. A template can be used to identify where the data fields are located on the form [3]. The two types of form are fixed and semi/un-structured. In fixed forms, the data is always found in the same place on the form which makes processing easier. In semi-structured or un-structured forms, the data is not necessarily in the same place on the form every time [4], [5].

Forms can have different types of input: checkboxes, printed fields or handwritten fields (plus barcodes and magnetic ink). Each of these inputs requires a different processing method[6], [7]. The simplest being optical mark recognition (OMR) for reading coloured or checked fields. The next simplest method is optical character recognition (OCR) for reading printed characters[8]. Optical word recognition (OWR) is similar to OCR but it targets whole words rather than individual characters. The most complex forms processing techniques are intelligent character recognition (ICR) and intelligent word recognition (IWR)[9]. ICR and IWR are a version of OCR and OWR

which aim to parse handwritten characters and words[10], [11]. While OCR and OWR can match inputs to stored characters and fonts, ICR and IWR must take a more intelligent approach to processing as handwriting is far more unique and unpredictable than typed input[12].

3.2) RECOGNITION METHODS

As previously mentioned, different input types require different recognition methods. The AIIM white paper's[1] survey investigated the levels of recognition implemented by a selection of organisations. Their findings are below in Figure 1:

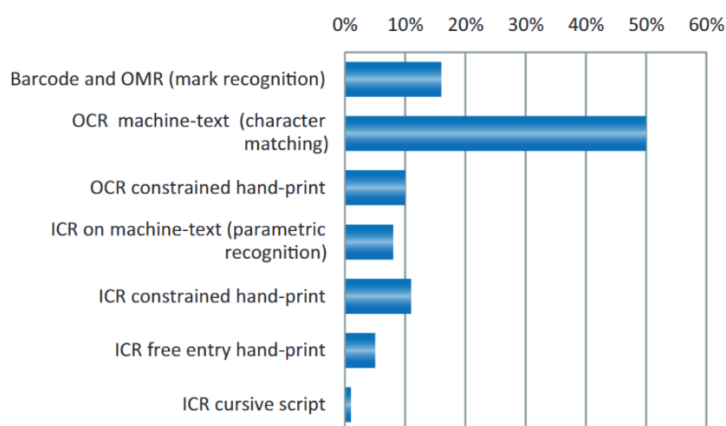


Figure 2 - Levels of Recognition (N=102)[1]

This shows that many companies are using some level of recognition with the simpler (and usually cheaper) methods being employed more often. OCR is being used to read machine-text in 50% of the organisations surveyed while hand-print is analysed using OCR and ICR in roughly 10%. This data can be used to infer that many companies are either constrained to offering only machine-text forms or hand-filled forms in which the handwritten data fields must be processed manually. The same study also found that a third of the companies offered forms with free-form unconstrained fields. For the companies that did use ICR handwriting recognition, the main benefits they found are below in Figure 3:

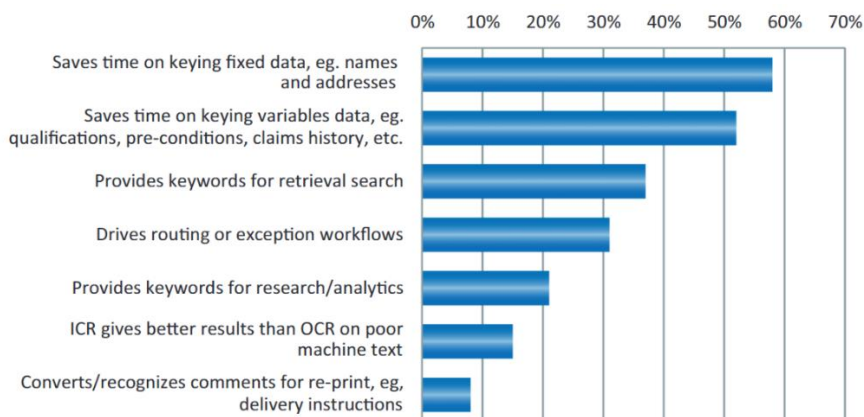


Figure 3 - Benefits of ICR (N=25)[1]

ICR is a process with many steps. Pre-processes such as de-skewing and monochrome conversion help to maximise the accuracy of the recognition. As far as algorithms are concerned, there are two main types commonly used in character recognition: matrix matching and feature extraction. Matrix matching involves isolating the character and matching it to a stored character. This approach is limited in its usefulness with handwritten characters as it compares a whole character to another whole character so it is most effective when dealing with typed text in known font. Feature extraction breaks down a character into features such as lines and loops which are then compared to character prototypes to decide which character is the most likely fit. This approach is better suited to processing handwritten text as it does not simply compare an image to another image and return a yes/no answer but instead identifies the features of the image to produce a match based on non-exact components. This allows for the greater variability of handwritten text to be accounted for.

3.3) EXISTING SOFTWARE

There are many commercial products available to help with forms processing. One popular example is Omnipage. This software is designed to read printed text but will also attempt to read handwritten text, resulting in nonsensical outputs[13].

The reCAPTCHA project offers a security feature for websites to ensure only legitimate, human users are able to access the site as opposed to bots. The service shows users images of words (and other images or sounds in more recent versions) that have been tested using OCR and found to be unreadable by software[14]. Users deciphering these words allow the project to collect data in order to improve ICR software. This is then used to process difficult-to-process books. Training character recognition algorithms improves their accuracy[15].

3.4) ALGORITHMS

When investigating existing research in the field of ICR and feature extraction, several themes recur: neural networks, segmentation, SVM (Support Vector Machine) and nearest neighbour along with a number of feature extraction algorithms including the use of wavelet transforms[16].

Artificial neural networks can be used to train the algorithm to improve its outcome[17], [18], [19], [20], [21]. SVMs can also be used for this purpose[22], [23] and distributed environments can be used to share work across nodes[24]. While these machine learning methods are often used and can improve the outcomes of a system, it is slightly beyond the scope of this project.

Nearest neighbour algorithms are simple machine learning algorithms used for classification or regression. They classify an object as belonging to a particular set based on the number of its neighbours that belong to that set. In character recognition, K-NN is used to classify extracted

features[25], [26]. K-NN is widely used because of its simplicity but leaves room for improvement as it is not particularly intelligent.

Segmentation is needed in forms processing in order to split an image into segments which allows characters to be processed individually. This process is essential in cases where characters touch or overlap. There are many methods of segmenting an image and the best choice often depends on the situation. Some examples include the use of a cross-sectional sequence graph[27] a water flow algorithm[28] or region of interest extraction[29] though many others exist[30], [31], [32], [33]. These methods are often developed in response to failings of other methods under specific circumstances such as when processing old texts with unfamiliar styles so comparisons are difficult to draw in a general sense.

In order to find the boundary of a field, cell extraction can be performed[3].

Feature extraction algorithms are used to break down a characters into features such as lines and loops[31], [34]. A horizontal energy profile can be used to classify the type of character being analysed which can provide insight into the orthography of the text[12]. Statistical feature extraction calculates the percentage of black pixels in a region while geometrical feature extraction calculates the angles of a character, topological feature extraction calculates the number of black-white transitions and directional feature extraction finds the direction of strokes[35]. Structural component extraction might use a planar graph to represent a character for analysis[36].

Feature detection can be summarised with the following categories:

- edge detection
- corner detection / interest points
- blob detection / region of interests

Skeletisation of the character must be carried out before any topological processing which requires the use of algorithms. Among many well-known algorithms, two possibilities for investigating a character's skeleton are edge walking and path finding[37] which can be used for segmentation as well.

Once a character has been extracted, fuzzy logic can be used to represent how well an image matches a character[10] and genetic algorithms can be used to mimic the process of Darwinian evolution to decide on a best match for a character[38].

3.5) EVALUATION OF SEGMENTATION ALGORITHMS

Dawoud[27] describes a method of using a cross section sequence graph for segmentation. This process involves boundary pixel extraction followed by the creation of the cross section sequence that analyses each pixel in relation to other pixels in order to find cross sections of regular sections of the image (Figure 4).

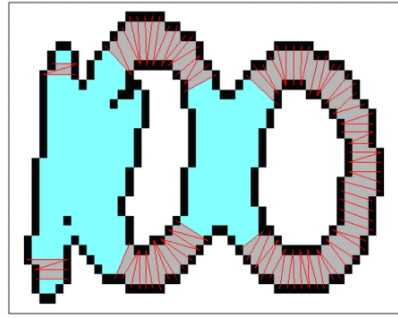


Figure 4 - Binary Image. Red Lines = Cross Section Sequences[27]

This iterative process is repeated until only regular sections are found. This means that the characters have been separated (Figure 5).

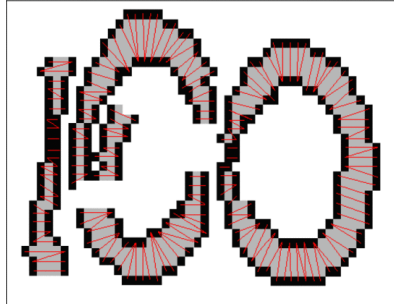


Figure 5 - 100 Has Been Separated[27]

This data can then be used to create the skeletons of the characters.

Brodic[28] describes the extended water flow algorithm. This builds upon the original water flow algorithm by introducing a slope direction variable. This algorithm simulates water flowing across the image and leaving certain areas dry. The difference between a shallow slope angle and a steeper one can be seen in the image below in Figure 6:

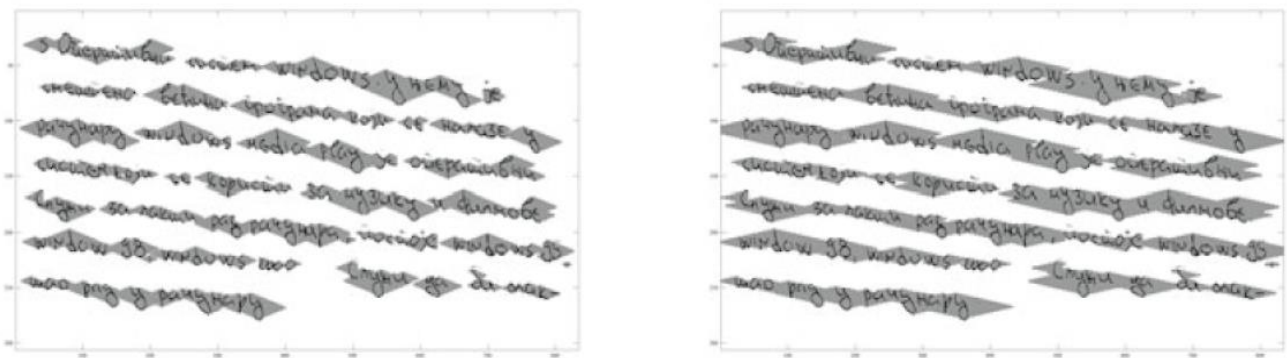


Figure 6 - 14° vs 8° Water Flow[28]

The steeper angle results in greater segmentation. The advantage of this algorithm is that it can identify lines of text rather than individual words. This should be used in conjunction with a character segmentation algorithm, not alone.

Nguyen[29] investigates handwriting symbol detection using colour differential before OCR. A heuristic filter is applied to remove noise and dilation is used to enhance the quality of the characters. The work makes use of a neural network for OCR which is beyond the scope of this project but the very basic detection of handwriting may still be applicable.

Ting[30] discusses connected character segmentation and points out that the top of a character (in English) carries more information than the bottom. The process outlined here involves skeletonisation and extraction of nodes and branches before segmentation. The segmentation process involves the algorithm attempting to match features to stored characters to form a test image which is then analysed. The focus is on the connection of pixels to find a path. In the case to joined nodes (when pixels appear in more than one character), the degree of membership is calculated to decide what belongs to each character. This is illustrated in Figure 7.

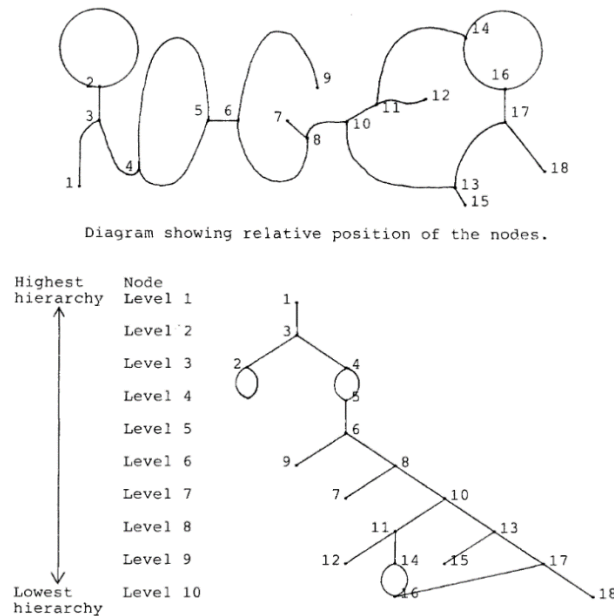


Figure 7 - Roger with Connected Nodes[30]

Pachpande[31] outlines the different layers of segmentation: structure, line, word and character. The work's word segmentation utilises a vertical projection histogram of each line. This histogram is analysed to find the width of gaps between and within words which is used to determine the number of words present.

Song[32] discusses forepart prediction as a segmentation method. The method uses forepart prediction to take the height of each line among other features and character-adaptive masking in order to be able to recognise a subset of characters as seen in Figure 8.

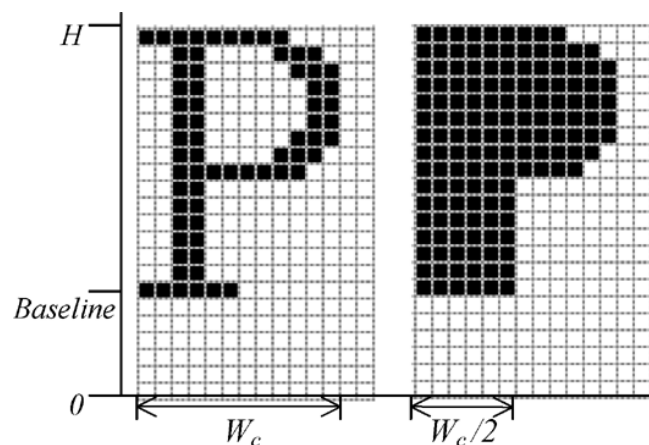


Figure 8 - Character Adaptive Mask[32]

Ashir[33] discusses a segmentation method using the white spaces between lines, words and characters. First, the document is processed along the vertical axis to find the rows and then these rows are processed along the horizontal axis to find words and characters as shown in Figure 9 and Figure 10.

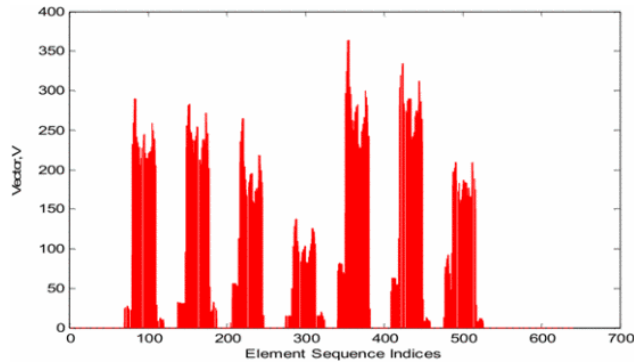


Figure 9 - Row Segmentation[33]

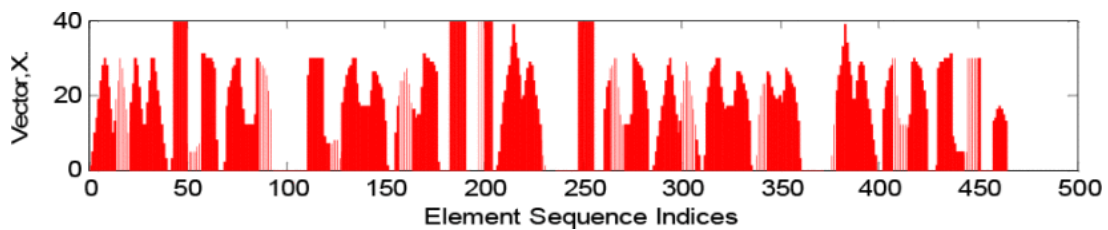


Figure 10 - Character and Word Segmentation[33]

Adaptive clustering is then used in cases where the width of the gap between characters is greater than average. This calculates the connectivity of pixels to extract blobs.

Of these researched methods, it is likely that the best suited for this project will be Ashir's work due to its relative simplicity. Pachpande's work similarly focuses on white space. This is subject to change after brief experimentation. It is possible that this method of segmentation will be wholly unsuitable in which case, either Ting or Dawoud's methods will most likely be explored.

3.6) EVALUATION OF FEATURE DETECTION ALGORITHMS

Cowell[34] describes a statistical approach to feature recognition. Rather than extracting features, the number of pixels in a character is used to match each character to template using probability. This approach seems very simple to the point that its usefulness is dubious. The work also utilises a confusion matrix which indicates the possibility of each character being confused with another.

Brodic and Amelio[12] discuss texture analysis. Run-length statistics is used to sequence the consecutive pixels along a line. This information is used to generate features such as what the work calls 'short run emphasis'. The features are used to determine the texture which is in turn used to analyse the text's orthography. As the work focuses on determining between two orthography styles rather than reading text, this process is not likely to be especially useful in the

case of this project. Sharma[35] outlines a feature extraction approach that utilises statistical, directional, geometrical and topological features to identify characters. Statistical feature extraction calculates the percentage of black pixels in each zone of each character. Thinning and normalisation are used to extract geometrical features; the centre of the skeleton of each character and the character is divided into zones and the pixels' distances from the centre and angles are calculated as seen in Figure 11.



Figure 11 - Character Zones[35]

Topological analysis is used to reduce possible confusion between characters. This involves analysing the number of transitions from black to white and vice versa both horizontally and vertically as seen in Figure 12.

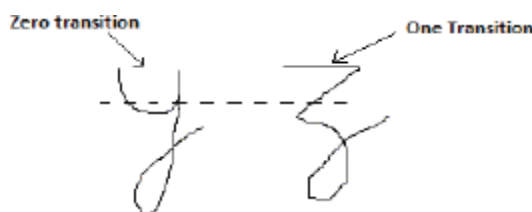


Figure 12 - Example of Topology Differentiating Characters[35]

A chain code histogram is used to analyse directional information; the contour of the character is traced to find external boundary pixels and their directions. Together, these features can be used to identify a character. This aspect of Sharma's work appears to be suitable for this project and will likely be one of the implemented methods.

Khaustov[36] discusses using a topological model to build a graphical representation of a character. This representation can be analysed to find the vertices which can be used as key points. After skeletonisation, the key points are selected (vertices and ends of lines) (Figure 13).

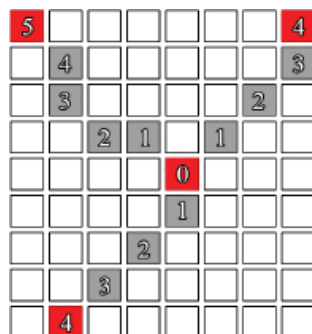


Figure 13 - Key Points[36]

The key points are used to find bends between them which are then used to create a topological representation of the character (Figure 14).

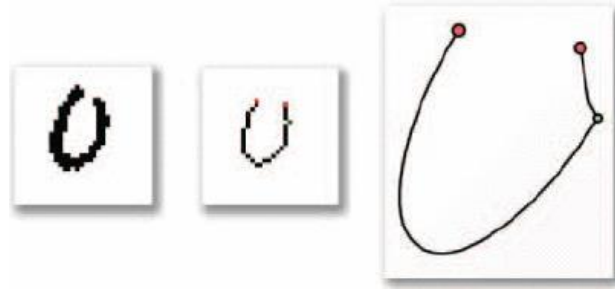


Figure 14 - Approximate Curve in Topological Representation[36]

A bipartite graph is used to connect key points. This is used in matching characters. This part of Khaustov's work appears to be suitable for this project and will possibly be one of the implemented methods.

Reed[37] investigates the creation of an OCR system that uses thinning to skeletonise characters, normalisation to scale the characters, templating to construct a representation of each known character which can be compared using edge filtering and edge walking for defining the characters as seen in Figure 15.

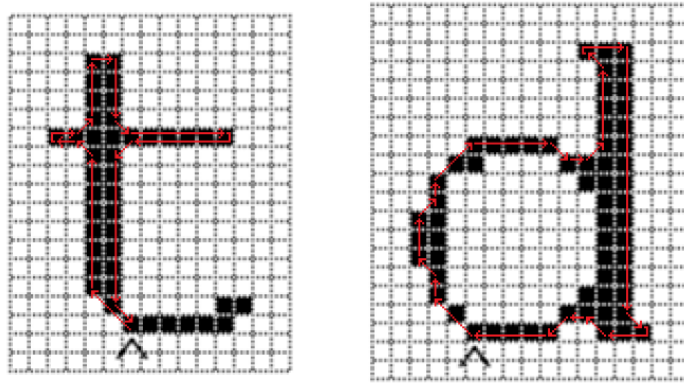


Figure 15 - Edge Walking[37]

Reed's work goes into great depth, investigating many techniques and analysing data. The bulk of the work is outside the scope of this project but the edge walking algorithm might be an interesting addition and will possibly appear as one of the implemented methods.

Of these researched methods, it is likely that the best suited for this project will be Sharma's compound feature extraction and either Reed's edge walking algorithm or Khaustov's topological method, possibly all three if time permits it. This is subject to change after brief experimentation and it is possible that a very simple homemade pixel analysis method will be implemented to test against instead.

3.7) MATCHING

As mentioned before, matrix matching and feature extraction are the two main types of character recognition strategies. Matrix matching simply matches printed characters to stored characters while feature extraction analyses features of each character and uses this information

to match features and provide a likely character match. Feature extraction is appropriate to use in the context of intelligent character recognition (handwritten form fields) as there is no exact character map database to match with for handwriting.

Different feature extraction algorithms will output a different list of attributes. For example, Khaustov's[36] algorithm will produce a list of key points and line angles. This means that the matching system for Khaustov's algorithm would need to match to a database containing the attributes for each character so that each processed character can be compared to all possible key point configurations and all possible line angle configurations in order to produce a most likely character match. As these characters are handwritten and thus variable, the matching process will need to utilise fuzzy logic to account for the fact that line angles will not be exact, etc.

Song[32] uses necessity-sufficiency matching which uses a feature-row-based character model to distinguish similar characters. This is used to match whole typed characters but some version of the algorithm could be used to match attribute lists or features.

Modern character recognition approaches apply machine learning algorithms to match characters to stored character masks. These algorithms can be trained using manually-matched input-output data pairs. This means that the system can figure out how the input and output images are related and use that information to classify unknown images. A large training set will likely produce a more successful algorithm.

3.8) MACHINE LEARNING

As briefly mentioned before, artificial neural networks and SVMs can be used to train algorithms to improve the outcome of the process. An artificial neural network is a system of nodes connected together to resemble the neural network of a brain. These networks can be used to learn to solve a problem in the optimal way mathematically. The system does not need any rules or information to be provided other than the sample set in order to solve the problem. An example of a neural network being used in image processing would be a system that identifies if an image contains a human face. The system would be trained using a set of images containing human faces and manually identified as such. The system would then use this training to classify other images without needing to know any extra information such as the fact that human faces are usually attached to human bodies or that they're likely to be a certain range above the ground.

The optimisation of the problem solving of these systems comes from the cost of an operation. An operation with lower cost will be favoured over one with higher cost. This will eventually result in a process that uses only the lowest cost operations; the optimal solution. For example, in order to add two numbers together, a system could take the first number and then

loop through the second number, adding one each time. This is an inefficient way to add numbers and an artificial neural network system will quickly discard it in favour of a more efficient method.

Supervised learning is a method of machine learning revolving around mapping an input to an output based on example pairs. Supervised learning algorithms analyse the training data and use it to build a function which is used to classify subsequent inputs. The algorithm must be able to form a generalised function in order to classify similar inputs as opposed to just matching exactly identical inputs and outputs which would not be very useful. Handwriting analysis is usually performed using supervised learning[17], [18], [19], [20], [21].

Unsupervised learning is not trained using input output pairs. It is used to identify previously unknown patterns. Cluster analysis is used to group data that is found to have similar traits. Each new data entry is compared to the data already in the cluster in order to group the data. This data is not labelled or categorised. It is used for pattern recognition.

Reinforcement learning is almost a cross between supervised and unsupervised learning. It does not require the input output pairs for training. The aim is to balance known and unknown ideas. It is especially useful when a system cannot be analysed without interacting with it.

A support vector machine (SVM) is a supervised learning model[39]. They are used to classify data into two categories. Support vector clustering can be used when unsupervised learning is required. An example of when an SVM might be used is when classifying pixels as being in either one character or the next [22], [23]. This can be done by calculating a hyperplane with the maximum distance between the two groups of pixels. This is illustrated below in Figure 16:

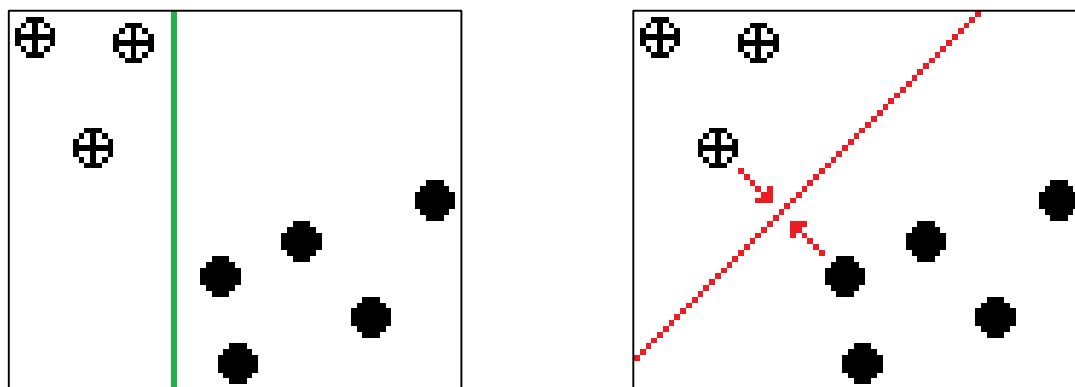


Figure 16 - The Green Hyperplane Separates The Two Groups By A Narrow Margin While The Red Hyperplane Separates The Groups By A Wide Margin

An example of where this specific method might be of use in OCR is seen in Figure 17:

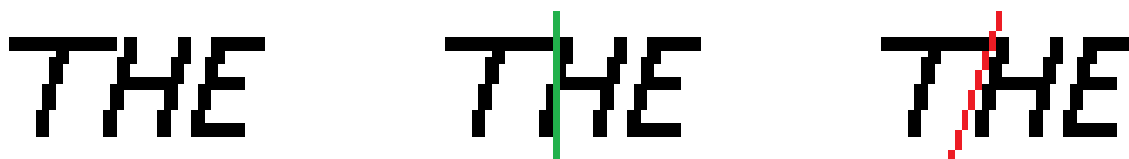


Figure 17 - The Green Hyperplane Cuts Off Part Of The H While The Red Hyperplane Properly Separates The Characters

The hyperplane with the largest separation between the closest points of each of the classes in the training data is the best. Figure 16 shows that multiple hyperplanes can achieve a separation between the classes with the green hyperplane being slightly less successful than the red hyperplane due to the distance between the nearest points each class being slightly smaller. The examples discussed here are linear SVMs. A linear SVM has a hard margin which describes the region between the classes as being bounded by two parallel lines with the margin lying halfway between them as illustrated below in Figure 18:

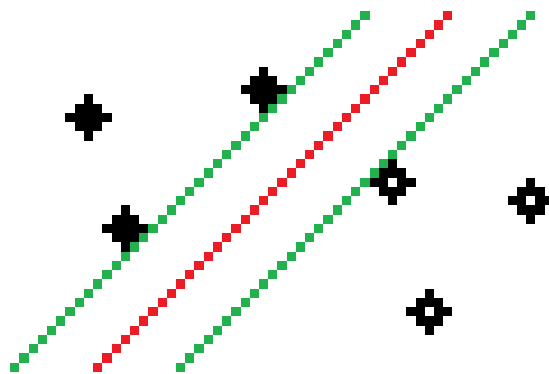


Figure 18 - Hard Margin

In contrast to a hard margin, a soft margin describes cases where the data cannot be separated by a linear hyperplane.

Statistical classification is another supervised learning technique used in handwriting recognition. K-nearest neighbour is one method of statistical classification that is used in this way[25], [26]. K-NN groups data according to the classification of the nearest neighbours. For example, if threshold distance $k=3$ and the nearest neighbours of object A belong to class X, then object A is assigned to class X as well. If two thirds of the nearest neighbours belong to class Y and one third belong to class X, object A is assigned to class Y. It is also possible to weight the votes so that closer neighbours' votes count for more than more distant neighbours'. In the context of character recognition, K-NN can be used to classify pixels as part of a character as it was used in Mithili and Kuwelkar's work on recognition of Devanagari characters [40].

3.9) COVNET

Neural networks can be designed to be modular in order to increase the ease of balancing the size of the system. An over-large system will require more training than necessary for the task while an undersized system would not learn how to solve the problem presented[41]. COVNET[41] is a cooperative coevolutionary model that develops subnetworks. This model uses cooperative coevolution to form modular neural networks with the modules learning to cooperate with each other. A model that builds itself in this way is referred to as structural learning. The network

assesses the suitability of its modules and deletes unsuitable nodes. Once the average fitness of the network stops increasing, the network stops evolving. This prevents over training. COVNET is one example of a viable option if an artificial neural network is to be used for character recognition. It is not suitable for real-time applications as the time taken to evolve the system may be quite large but in the case of forms processing, this should not be an issue.

3.10) CASCADED CLASSIFIERS

Cascading is a method of machine learning that uses multiple algorithms to classify data into a string of multiple classifiers[42]. Each classifier is used to provide more information to the previous classification. For example, in a system designed to detect cats in images, each stage should be able to identify every image of a cat in every image and return a percentage of false positives which are eliminated in later stages. Each stage analyses the image in more depth than the last, resulting in increasing processing time and an increasingly fine granularity. This is illustrated below in Figure 19:

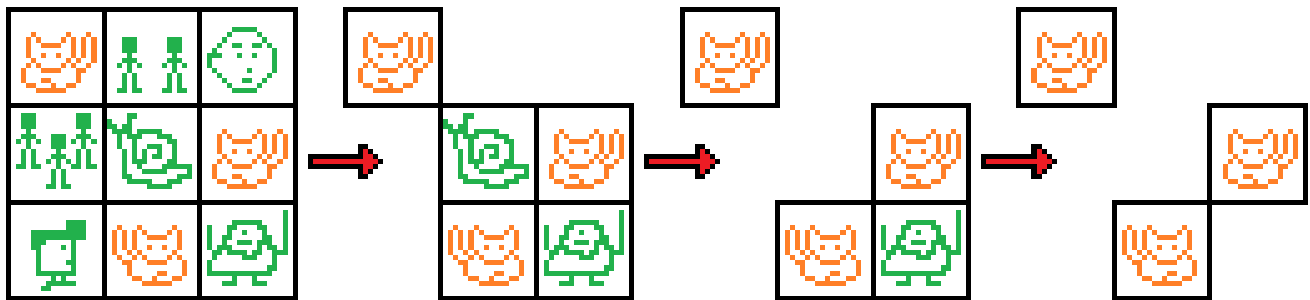


Figure 19 - Cascading Classifiers, Input And Three Stages

The illustration shows that the tiles that most obviously don't contain cats are eliminated immediately which reduces the computational power required for the rest of the process. Each stage then returns fewer false positives until, eventually (ideally), only the tiles containing cats remain.

The system works by training many simple learning algorithms and giving more weight to those that are correct more often in the training process. The process is used in facial recognition and object tracking.

3.11) EVALUATION OF AVAILABLE SOFTWARE

ImageJ[43], [44] (or Fiji) is an open-source image processing program developed in java. Its purpose is image processing and analysis for scientific purposes. Along with a wide array of built-in functions, ImageJ boasts a collection of thousands of downloadable plugins and macros as well as the ability to create custom scripts. This means that many common image processing techniques and tools can be easily invoked through short scripts as opposed to having to code them from scratch. An example of an ImageJ macro being called in a java method is shown below:

IJ.run("Enhance Contrast...", "saturated=0.3");

This code invokes the Enhance Contrast tool with a saturation level pre-selected which allows for the automation of processing as the user no longer needs to click on popup windows to select any options. For this project, ImageJ can be used to perform the pre-processing functions to make maximise the readability of the scanned images.

The analysis tools in ImageJ can be used to inform the development of the software. For example, the Analyse Line Graph tool is useful for determining where basic character segmentation will be most effective.

While ImageJ has available functions for segmentation, these are mostly for the processing of photographs to separate layers of a scene and not for text segmentation. The watershed segmentation tool is useful when segmenting thick handwriting where the thinnest section is the join between characters but does not work for word skeletons. It is possible that text segmentation scripts exist but it is likely that segmentation will need to be developed from scratch in order to adequately emulate the methods previously discussed.

Java with IntelliJ IDEA[45] will be used to develop the code for this project. IntelliJ community edition is free and offers IntelliSense which makes writing code easier and quicker. ImageJ is Java-based so Java programs can easily incorporate ImageJ scripts and macros.

OpenCV[46] is a programming library written in C and C++ but available in an array of languages including java. It includes a range of resources for image processing including a machine learning library with many features mentioned in this paper such as SVM, k-nearest neighbour algorithm, artificial neural network and cascading classifiers along with training methods. This library can help immensely when a system includes aspects of machine learning.

Noble[47] carried out a comparison between OpenCV's available feature detectors and feature matchers by experimenting with processing a series of images of the same sculpture captured from different angles. The work found that of the six feature detectors and two feature matchers tested, the SURF feature detector algorithm detected the most features and the BF feature matching algorithm matched the most but the optimum available combination was found to be the BRISK feature detector paired with the BF feature matcher.

In the context of character recognition, Vaidya *et al* [48] investigated handwritten character detection using deep neural networks. The work used OpenCV to process the images and Tensorflow (another machine learning programming library) to train the neural network. They used OpenCV to perform segmentation which was carried out first by line, then by word and finally by character. It is important to note that this work dealt with handwritten printed characters and not cursive. They were able to create a system that takes a photograph of handwritten printed English text and converts it to typed text.

Shelke and Apte[49] investigated the development of a real-time character recognition program for Marathi script which is a connected Indian script. Again, the work does not consider cursive but instead reads isolated characters. The matching is done by comparing characters to a template. OpenCV was used in the pre-processing stage to carry out de-skewing etc. OpenCV appears to have great potential in the field of character recognition to aid in development. It will likely be used in this project if it is decided that machine learning is to be used. It may also be used to carry out segmentation if found to be suitable after preliminary experimentation.

It is possible that a database may be required to store training data, character maps, character or feature information, etc. If this is found to be the case, either PostgreSQL or MongoDB will be used depending on what needs to be stored and ease of access. PostgreSQL is relational while MongoDB is document-oriented which will inform the eventual choice.

For this project, the minimum viable product (MVP) does not include the use of machine learning for segmentation, feature extraction or matching but with the vast majority of modern character recognition research utilising various machine learning techniques, it would be remiss not to attempt to investigate this. As such, while the MVP will not include machine learning, any further development will include it as a separate system for comparison with the systems developed as the MVP.

3.12) PROPOSED OVERVIEW OF THE SOFTWARE

Based on the research undertaken in this section, the proposed structure of the software can be outlined as follows:

- Input
- Pre-processing
- Segmentation
- Skeletisation
- Feature extraction
- Matching

The input stage will take in the image of the text that in this case, will consist of single handwritten words and short phrases. This will likely be carried out using ImageJ.

The pre-processing stage will involve de-skewing if necessary and conversion to black and white images and then a binary matrix[50].

Segmentation will be performed using one of the seven aforementioned methods. The exact method will be selected upon some further research and light experimentation but should remain the same in every tested program.

Skeletisation will be attempted using path finding but this is subject to change if it is found to be unsuitable during development.

Feature extraction will be carried out using a very basic edge detection method for the first program. For the second program, one of the aforementioned methods will be selected. This will be chosen based on further research and some light experimentation.

The matching stage will be carried out using fuzzy logic to match the characters to a collection of stored characters and features.

3.13) LITERATURE REVIEW CONCLUSION

This literature review has investigated prior research into forms processing, the stages involved and some methods and algorithms used. The research conducted here has been used to inform the direction of the project as a whole with candidates for segmentation and feature extraction methods being selected from the literature.

First, the general idea of forms processing was discussed, leading to an investigation into the types of forms and the different methods of processing them. Then existing software and projects on related topics were explored. The algorithms section of the literature review investigated existing algorithms and methods of segmentation and feature detection which were then evaluated in order to choose any appropriate for this project. The matching section discussed methods of matching the extracted characters to known characters and the machine learning section discussed the use of machine learning in modern forms processing with the following two sections describing two specific instances of this in more detail. The final sections discussed the software to be used in this and the probable course of the software development portion project based on what has been learned here.

The next section will outline the plan for the project in a more concrete way, building on the brief proposed plan put forward at the end of this review.

4) PROJECT PLANNING:

The resources required for this project are listed below:

- DVLA issued laptop.
- University access to Microsoft office and onedrive.
- IntelliJ community edition for programming
- ImageJ for image processing
- Database software (mongodb or postgresql) for storing character maps
- OpenCV for image processing and possible machine learning
- Online research materials (including university online library)
- University library access
- Possible additional image processing tools to be determined in design stages

The software development portion of this project can be broken down into its own list of tasks as seen below:

- DESIGN
- PRE-PROCESSING
- LINE SEGMENTATION
- WORD SEGMENTATION
- CHARACTER SEGMENTATION
- IMPLEMENTATION OF FEATURE EXTRACTION METHODS
- BASIC MATCHING SYSTEM
- POSSIBLE - MACHINE LEARNING MATCHING SYSTEM
- POSSIBLE - MACHINE LEARNING CHARACTER SEGMENTATION

The possible implementation of machine learning for segmentation and matching is dependent on time. If it is found to be feasible during development to implement these systems within the time constraint, this is the ideal outcome of development. The introduction of machine learning means that these systems must be trained which requires a large sample of known training data which will need to be collected.

The design stage of development will consist of bullet pointed lists similar to these task lists and some ULM diagrams where appropriate. A UML diagram will be used to illustrate the entire system as a whole, showing how different components interact. UML may also be used to illustrate the flow of data through the system and possibly to illustrate a selection of the classes developed.

The experiment portion can be broken down into its own list of tasks as seen below:

- COLLECT TEXT SAMPLES (KNOWN PHRASE IN CURSIVE)
- PASS TEXT SAMPLES INTO SYSTEM
- RETURN RESULTS FROM BOTH METHODS
- ANALYSE RESULTS, PRODUCING A CORRECTNESS PERCENTAGE
- COMPARE RESULTS OF TWO METHODS

The analysis listed here is a basic analysis of the results, only measuring how often each algorithm correctly identified the given characters. Later analysis will explore the possible reasons for incorrect or unsuccessful identifications.

This section has outlined a plan of action for the project going forward, including the software development steps anticipated. The next section will explore the research methodology used in the project including the philosophy, approach, strategies and limitations.

5) RESEARCH METHODOLOGY:

This section will discuss the available research methods in order to select an appropriate approach for this project. The project's aims will be outlined before discussing possible philosophies, approaches and strategies.

The aim of this project is to investigate different techniques and algorithms for recognising handwritten characters. Two methods will be developed into methods in a functioning system which will be tested to find the effectiveness of each method in a given situation.

By the end of this project, the aim is to have produced a program that uses two different approaches to character recognition to parse characters. An experiment will be carried out on this program to investigate the effectiveness of each of the two approaches. The final result should be a collection of information on the effectiveness and suitability of each approach and a conclusion as to which approach is most appropriate in the given case.

5.1) PHILOSOPHY

There are four research philosophies available: pragmatism, positivism, realism and interpretivism[51]. The two main philosophies are positivism and interpretivism which are essentially opposites.

Positivism deals with real, objective outcomes and experiments meant to observe the physical. This is a quantitative philosophy with the outcomes deriving from observable and

quantifiable results and some objective interpretation. Emphasis is placed on data collection and analysis and there should be no bias or special interests involved. One of the principles of positivistic research is that there is no “common sense”. This means that results must be allowed to go against what the researcher “knows” without the researcher introducing any bias. This makes positivism more of a scientific philosophy, appropriate in cases where the subject matter is closer to science than social science. In cases where research is dependent on subjective data such as user experience, this is not an appropriate philosophy. Positivistic research usually deals with large sample sizes and quantitative analysis.

Interpretivism, in contrast to positivism, deals more with subjective data. It is better suited than positivism for social science research. An example of an appropriate use of interpretivism is a study on the effect of a certain piece of software on the experience of a group of school pupils. This research might include researcher observations, individual or group interviews and surveys or questionnaires. These strategies can be used to gather subjective data and questions might be phrased as such:

“Do you enjoy science classes when you use the named software?”

“Do you think you are doing better or worse in class now that you use the software?”

“How would you rate your experience on a scale of 1 to 10?”

where the answers can take the form of free written or verbal fields or check boxes. This kind of data is subjective and should largely be analysed qualitatively. It is clear that a positivistic approach would make little sense in this case but interpretivism brings with it the danger of subjectivity. It is important to be aware of and minimise any researcher bias. Subjectivity also means that generalisation is less possible as the same study can be carried out with different samples and produce different results. It is important to take note of cultural factors.

Pragmatism and realism are less rigid than positivism and interpretivism. Pragmatism can involve both positivistic and interpretivistic techniques in a single study. The main part of a pragmatistic project is the research question. This determines the research approaches taken. It is possible to take a qualitative and a quantitative approach using pragmatism. This philosophy might be appropriate when a research question leads first to an objective data set (e.g. measurable outcomes such as exam results) and then to a subjective data set (e.g. experience of test takers).

Realism can take two forms: direct and critical. Direct realism deals with observations through human experience while critical realism takes the stance that human experience does not accurately portray reality. The main difference in research is that critical realism places importance on multi-level study to identify influences and relationships. Critical realism can deliver a more thorough view of the sample set and the research than direct realism. Realism can be quantitative or qualitative and is usually employed in more social science research. It might be appropriate in a study of social group dynamics.

Given this information and the aims of this project, it appears that the most appropriate philosophy in this case is positivism. This philosophy was chosen because the aim is to test different algorithms based on outcomes and not user experience. The lack of human participants means that this aligns best with a positivistic project dealing with objective, quantitative data rather than subjective experiences.

5.2) APPROACH

The available research approaches are deductive, inductive and abductive.

A deductive approach involves a hypothesis and an experiment to test it. A hypothesis is created and data is collected and analysed to either prove or disprove the hypothesis. For example, the hypothesis (or null hypothesis) might be that there is no difference between the number of male school pupils achieving a C or above in GCSE maths and the number of female pupils doing the same. The data collected would be the GCSE maths results of the two groups and this data would be analysed to either prove that there is no difference in numbers or to disprove the hypothesis in either direction (more male pupils achieving C or above or fewer male pupils achieving C or above). The conclusion drawn is only if the hypothesis is proven or disproven by the research. There is no speculation in this conclusion such as “female pupils achieved high grades in greater numbers because...”. This is a more quantitative approach.

An inductive approach explores a phenomenon. This approach is used to find patterns rather than confirm previously discovered patterns and can be used to form a hypothesis for later work. The observations are made first, before any hypothesis is discussed. Then the data is analysed to find trends and any trends can be used to form a theory as to why they exist. An example of an appropriate use of an inductive approach would be a study on how gender affects GCSE results. This is similar to the deductive example but in this case, the observations are made first and then trends are extrapolated. If the observation shows that boys achieve higher grades on average, this can later be explored in depth with research into possible causes of this phenomenon.

An abductive approach aims to explain a phenomenon. An example of its use might be a continuation of the previous example. If it is found that boys achieve higher grades on average than girls, the research question might be why is this the case? Qualitative and quantitative methods can be used to gather information in order to create a theory.

The approach taken in this project is deductive. A deductive approach involves a hypothesis and an experiment to test it rather than a theory to explain the research. This approach was chosen because there is no existing hypothesis as to which algorithm is best suited in this

case. The aim is to investigate the algorithms to find the best suited one which requires an experiment which is a deductive approach.

5.3) RESEARCH PROCESS

Given the positivist, deductive approach chosen, the strategies employed in this project are a literature review to research the background of the topic and plan out a course of action, software development to create some software based on the background research and an experiment to compare the outputs of the developed software. Three strategies were chosen in order to triangulate the research and these particular strategies were chosen because they allow for a sensible project flow:

- Literature review – investigate existing research
- Software development – implement the findings of the literature review to create programs to test
- Experiment – test the developed software in order to investigate the different outcomes of different algorithms

The selected methods and algorithms for character recognition will be developed in order to investigate their effectiveness in a given case. By the end of the project, at least two methods should be developed into functioning programs. The aim of the software development stage is to implement the findings of the literature review by developing some of the researched algorithms. The programs developed will be used in the experiment stage.

The developed software will be compared in an experiment. The experiment will entail testing each program by processing the same or similar information and gathering data on accuracy and efficiency. This data will then be compared to draw conclusions.

The accuracy of each algorithm will be tested by inputting known characters and measuring how many characters each algorithm correctly identifies. A set of at least twenty images containing the same characters (written by different people) will be processed and the accuracy of each algorithm will be averaged. Comparing this data should result in a conclusion on which algorithm is more accurate. This form of experiment should lead to reliable results as long as the software is always developed in the same way. The generalisability of this experiment covers other methods of character recognition: the experiment could be expanded to include additional algorithms.

The data being evaluated will be the results from the experiment. The success rate of each algorithm will be recorded and compared as well as any other useful information such as processing speed. The data will be analysed, compared and ranked. In order to produce data with

acceptable accuracy, each test will be performed multiple times and the results will be averaged. Reliability will also be tested by repeating the tests. The data produced by the experiment and analysed during evaluation will be presented with graphs and rank tables.

In the event that machine learning is used, the algorithm will need to be trained using training data (pairs of known inputs and outputs). This training data will be gathered by asking volunteers to write a set phrase in cursive and print. This will hopefully result in an algorithm with a higher success rate than if training data was not used or if training data only consisted of one person's handwriting.

5.4) RISKS AND LIMITATIONS

As this project may involve volunteer participants, any collected data must be treated appropriately. The data being handled will not contain any personal information aside from unlabelled handwriting samples or any government information aside from possible agency logos so there should be no concern over GDPR or privacy.

As a deductive approach is being taken, time is in short supply which will limit the possible scope of the project but risks are limited and prior research is abundant.

The project will be restricted to only using free resources or resources available through the university at no extra cost.

As this research is being undertaken as part of a partnership with the DVLA, a government organisation, it is possible that DVLA forms and processes might be discussed when seeking advice. None of the content of this project should contain any material retrieved from the DVLA and no DVLA processed should be detailed.

This research methodology section has discussed the research methodology of the project, including the philosophy, approach and strategies. The chosen philosophy is positivism with the approach being deductive. The strategies employed will be a literature review to investigate existing research, software development to implement the findings of the review and an experiment to test the different algorithms.

The next section will describe the software development methodology utilised by the project.

6) SOFTWARE DEVELOPMENT METHODOLOGY:

This section will briefly discuss some available software methodologies before selecting an appropriate choice for the project with the elements of the chosen approach laid out and justified.

It is important to set out how the development stage will be structured and carried out based on the requirements of the project. The most appropriate methodology will depend on many factors such as team size and structure, the needs of concerned parties, the existence of product owners or customers, available technologies, the planned release schedule, etc. In cases where a large team is used, it is often beneficial to use an agile methodology such as SCRUM[52]. Agile methodologies exist essentially to encourage team interactions and minimise the problems caused by a lack of communication. Agile teams should be self-organising and extremely communicative throughout the development process. Many agile methodologies assign team roles. For example, a SCRUM team is structured as follows:

- Product owner – in charge of business requirements, customer needs and maintaining and prioritising the product backlog.
- Scrum master – in charge of ensuring SCRUM is carried out within the team by scheduling resources and meetings.
- Development team – in charge of development and team skill sharing as well as planning each sprint.

In order to ensure good communication, SCRUM has a set of required meetings. These are the daily stand-up, the sprint review and the retrospective. Other meetings such as planning sessions or refinement can be added where necessary.

Each feature to be developed is split into two week sprints which consist of a number of user stories taking the form “As a ____ I want ____ so that ____”. This sprint system is a way of implementing both iterative and incremental development which allows for an approach that is constantly producing functional software but also allows for completed functions to be revisited. Iterative development usually focuses on chunks of work being released at deadlines while incremental development involves implementing functions as they are developed so that every release of the software is functional and is built upon the previous release.

Another example of an agile methodology is Extreme Programming (XP). This methodology focuses on paired programming, code review and code that is extremely clear[53]. Like SCRUM, frequent communication between the development team and stakeholders is paramount. Emphasis is placed on frequent releases of new features which is a key feature of agile methodologies. XP uses test-driven development which means that tests are carried out from the beginning of development and used to provide immediate feedback. This practice allows for bugs to be discovered and fixed at the earliest possible moment. The roles in XP are very similar to

SCRUM with the addition of the optional tracker position which is responsible for keeping track of progress.

Agile methodologies can have many benefits when implemented correctly and in the right context but it is important to take note of possible drawbacks associated as well as benefits. The most impactful criticism of agile in general is the potential for scope creep[54], [55]. This comes about because stakeholders are encouraged to provide input at every point during development which means that user stories can be added constantly. It is possible to spend a sprint or development cycle implementing a feature which is then completely changed or even removed by new user requirements.

In contrast to agile methodologies, many traditional software development methodologies exist. For example, Personal Software Process (PSP) is similar to agile except in the way documentation is used extensively and data is used to predict and refine the schedule of a project[56]. Another example would be the waterfall method which takes a linear approach to software development[57]. The steps involved in waterfall development are as follows:

- Requirements
- Design
- Implementation
- Testing
- Maintenance

From this, it can be seen that each stage follows on from the last with no revisits which means that if stakeholders want to add features or are unsatisfied with the final product, there is no opportunity for input. This linear, sequential development style should help to reduce the number of problems encountered in the later stages as they should be found in the requirements and design stages. Conversely, it is far more rigid and less stakeholder-friendly than agile methodologies.

Outside of agile and non-agile methodologies, tools and frameworks exist to ease development and decision making. One example is Kanban which is a system that is often used in conjunction with other methodologies[58]. The core tool involved is the Kanban board which lays out the tasks (or stories) to be completed in columns as in this example (Figure 20):

Backlog	Refining	In Progress	Reviewing	Testing	Done	Blocked
Story A		Defect ii		Defect iii		Story C
Story B		Spike 1				
Defect i						

Figure 20 - Example Kanban Board

This example board includes a 'Refining' column which signifies that a story or task is not yet ready to be developed in its current form and requires more information. It is better to start with a fully refined backlog but in some cases, this column is necessary.

In the case of this project, there are no stakeholders, a development team of one and the desired end product is not a fully functioning, market-ready piece of software but rather a small program for experimentation. Given these points and the information gathered on different development methodologies, the most suitable approach seems to be a hybrid of Kanban and a simple version of agile methodologies in general. The main things taken from agile would be the structure of user stories to build the backlog and the more open structure with opportunity to add requirements throughout the process along with the test-driven development practiced in XP. The user story structure is useful in laying out functional requirements and helps to ensure the whole system is planned out in small units of work.

Test-driven development ensures that the user stories will definitely be completed to the exact specifications because the tests are written to conform to the story and the code is written to pass the tests. This also ensures that user stories are fully understood before development takes place which reduces misunderstandings.

Kanban is useful for keeping track of progress and understanding what needs to be done at every point in development.

With these selected elements, the constructed approach can be outlined as such:

- Gather requirements and build backlog of user stories
- Design system and refine stories
- Write unit tests
- Develop stories

with the tasks being displayed on a Kanban board with the column headings 'Backlog', 'Writing Tests', 'Developing', 'Static Testing', 'Done' and 'Blocked'. This setup should allow for easy tracking of progress and management of tasks while allowing for continual improvements and remaining relevant to the one-dev context of the project.

7) SOFTWARE DESIGN:

This section will document the software development design process for a character recognition system that uses the two chosen methods. The requirements will be set out using the user story structure and the design will be outlined using diagrams.

7.1) REQUIREMENTS

The minimum viable product for this project is a system with two different algorithms for feature extraction which can be used to compare to a collection of stored features. Additional software features include segmentation (line, word and character) and a machine learning module.

In order to produce the minimum viable product, the following requirements must be met:

- a) I want to be able to input an image so that it can be processed
- b) I want to be able to process an image so that it is easier for the system to use
- c) I want to be able to turn the image into an array so that it can be transformed
- d) I want to be able to extract the text's lines so that each one is separate
- e) I want to be able to extract the text's words so that each one is separate
- f) I want to be able to extract the text's printed characters so that each one is separate
- g) (*NOT MVP*) I want to be able to extract the text's cursive characters so that each one is separate
- h) I want to be able to extract the characters' features using two different algorithms for matching
- i) I want to be able to match the extracted features to a collection of stored features so that they can be matched to possible characters
- j) (*NOT MVP*) I want to be able to use machine learning to improve the matching process
- k) I want the system to return a list of most likely character matches so that I can see if the text in the image was read correctly

Each of these stories can be further broken down into a small set of tasks. For example, story b) can be broken into the following subtasks:

- Spike: Investigate the best methods for pre-processing an image
- Follow on stories:
- Load image
 - Convert image to greyscale and save
 - Write scripts or macros to perform the ImageJ functions "Despeckle", "Enhance Contrast", "Smooth", "Make Binary", "Close-" and "Skeletonize" and save the resulting image
 - Write unit tests for all methods

It is likely that all stories will begin with a spike to investigate how best to fulfil the requirement.

7.2) DESIGN

From the requirements of the minimum viable product, an overview of the design of the proposed system can be illustrated using a dataflow diagram (Figure 21):

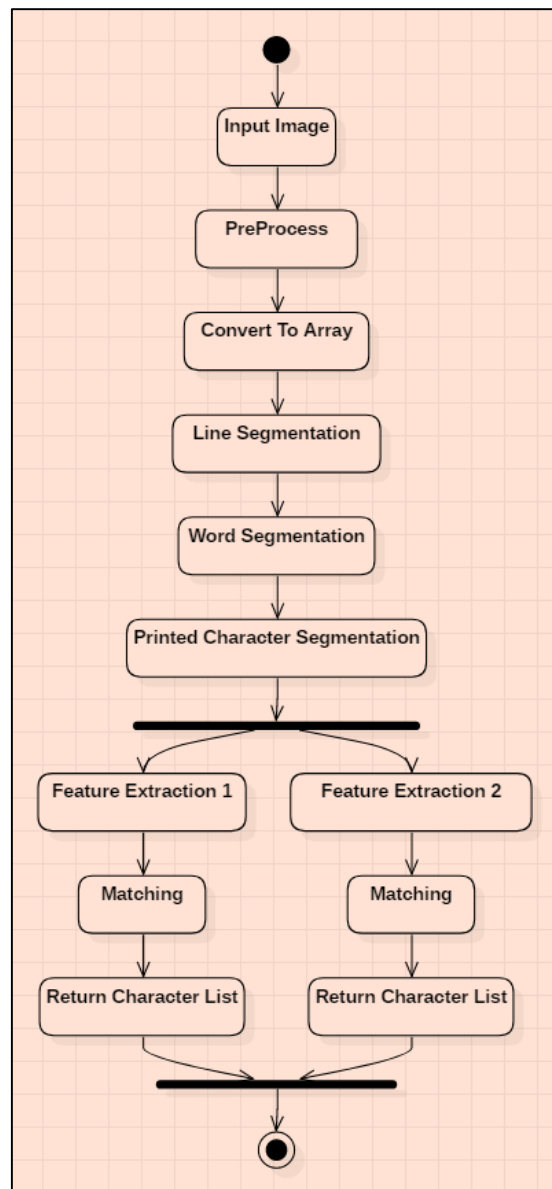


Figure 21 - Flow Diagram Of Overall System

It is likely that each of the activities shown in this diagram will be a class. A class diagram can be used to outline the classes of the proposed system as shown in Figure 22:

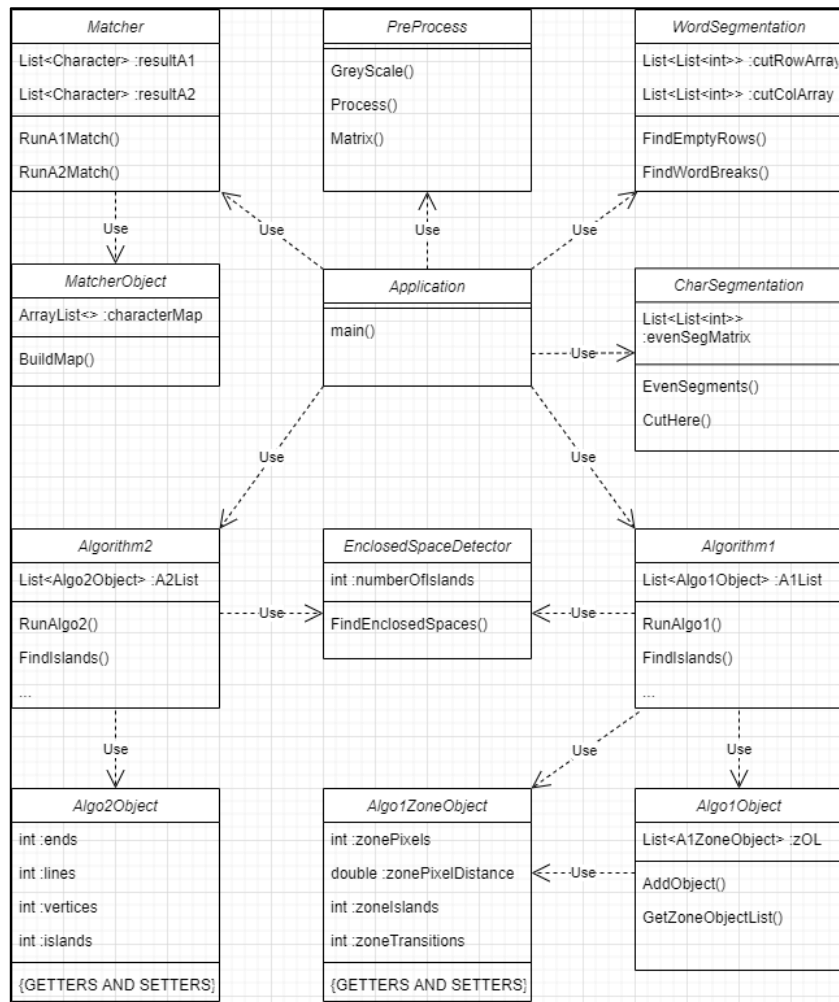


Figure 22- Class Diagram

Note that this diagram shows only two methods for each of the two algorithms. In reality, each of these algorithms will contain at least five methods; one for each value being collected and one to call the others. Algo2Object and Algo1ZoneObject have {GETTERS AND SETTERS} in their method fields to illustrate that each variable needs a getter and a setter.

8) SOFTWARE IMPLEMENTATION:

This section will document the software development process. Testing will be discussed and illustrated and then the development of the code will be discussed and illustrated.

8.1) TESTING

The testing framework being used in this project for unit testing is junit. Each method should be developed to pass a unit test. The tests should be written first wherever possible. Following the above dataflow diagram, each activity should be a class with a few methods inside. An example of this is the following (collapsed) code for some of the line segmentation methods and their respective unit tests (Figure 23 and Figure 24):

```

//FIND EMPTY ROWS:
public static List<List<int[]>> FindEmptyRows(List<int[][]> matrices)
{...}

//LOOP THROUGH ARRAYS:
public static void LoopThroughArrays(List<Integer> cutpoints, List<int[][]>
matrices, List<List<int[]>> cutRowArray)
{...}

//LOOP THROUGH ROWS:
public static void LoopThroughRows(List<Integer> cutpoints, int[][] array)
{...}

//CHECK FOR BREAKPOINTS:
public static List<Integer> CheckForBreaks(List<Integer> cutpoints,
int[][] array, int row)
{...}

```

Figure 23 - Example of Methods

```

@Test
void findEmptyRowsTest() {...}

@Test
void loopThroughArraysTest() {...}

@Test
void loopThroughRowsTest() {...}

@Test
void checkForBreaksTestGotOne() {...}

@Test
void checkForBreaksTestNotOne() {...}

```

Figure 24- Example of Tests

Here, the method findEmptyRows acts as the main class, calling the other methods. This means that findEmptyRowsTest in the test class is actually testing the whole line segmentation process end-to-end rather than one small part as the other tests do.

The other classes or activities should follow the same pattern wherever possible.

8.2) DEVELOPMENT

Building on the dataflow diagram (Figure 21) and the class diagram (Figure 22) in the design section and the user stories in the requirements section, the unit tests were written first. Once the unit tests were completed the code could begin to be developed.

The first two user stories listed involve the image input. The images of the test data must be input into the system in a way that allows mathematical operations to be performed on them. It was decided that converting the image into an array would be the most appropriate way of allowing these mathematical operations. Before this, the images must be pre-processed. This should increase the readability of the text in the image so that the algorithms are more likely to be able to identify the characters.

The first component to be developed was the pre-processing class which takes in the input image and converts it to greyscale before performing imageJ functions to increase usability of the image. It then saves the image as a binary matrix. The imageJ library was used to perform all of the actual pre-processing functions such as despeckling. The spike for investigating how to carry out this user story was carried out by using the imageJ desktop application to experiment with using different functions on sample images.

The second component to begin to be developed was the main application. The aim of this class was to call the other methods and also to perform utility functions involving reading and deleting files.

At this point in development, the system was able to take in an image file, perform a set of pre-processing scripts on it, convert the image into an array and clean up the file system (delete the image copies made during pre-processing).

The system needs to be able to separate characters in order to analyse each one separately. The first step towards this is to identify the lines of text so the next component to be developed was line segmentation. This set of methods identifies the line breaks and separates the input array into separate lines.

After line segmentation, the word segmentation was developed. This takes the separated lines and identifies the gaps which in turn identifies the word breaks. In the case of printed (non-cursive) characters, characters are separated during this step.

The next component attempted was the cursive character segmentation. This is a non-MVP story but was outlined at this stage in development for further refinement at a later date. These functions use an idea from Salvi's work[59] which splits the word into even segments and analyses these segments to find which segments belong to each character within the word.

Once MVP segmentation was implemented, the characters had been separated so the next stage was the development of the recognition algorithms. The two algorithms were developed as two classes, each containing the methods for extracting various features. The first algorithm was based on Sharma's work[35] and involves splitting the character into zones and analysing the zones. The second algorithm was based on Khaustov's work[36] which analyses the character as a whole. Both of these algorithms use a method that finds enclosed spaces within a character or zone. This method was based on a solution to an online tutorial problem[60] and adapted to suit the needs of this work. The implementation of the two algorithms is the main function of the system as these are the subjects of the experiment. The algorithms both return objects containing the character features.

Having returned these character feature objects, the next stage was to match this information to stored average information for characters. The matching method takes in the list of objects

returned from the two feature extraction methods, gets the information from the fields and then compares this information to information stored in the matcher object. This matcher object contains averaged values for all fields as well as the corresponding character and was made by passing a collection of images of each character through the two algorithms and averaging the results for each field. The matcher allows the algorithms to be functional in a meaningful way and allows for testing and analysis.

The results from the zone-based algorithm (based on Sharma's work) are matched based on zone averages because the difference in the number of zones a character can be split into is based on image size which can lead to inconsistency.

This method returns the most likely character match for each input character or, if no match is found, a '?' is returned.

At this stage in development, the system can take in an image, pre-process and convert it, segment it (excluding cursive segmentation), pass it through the two algorithms, return its character information and match that information against stored character data. The output of the system is two strings for the primary matches produced by each algorithm and two for the secondary matches.

The last user story listed is:

- *(NOT MVP) I want to be able to use machine learning to improve the matching process*

This story was not developed and is instead a starting point for future work. During the initial spike for this story to investigate how to implement machine learning, it was decided that as most machine learning character recognition methods use character masks to segment the sentence, it would not be fully appropriate to make machine learning a core part of this project. Machine learning could be used to build a separate system for comparison but it is not easily used specifically for feature extraction alone which would mean that the experiment would not be a fair test between feature extraction algorithms if used in this case.

This section has discussed the development of the system containing the two algorithms, outlining the order of development and the user stories covered. The next subsection will highlight the main aspects of the software as well as some of the difficulties encountered and any particular successes.

8.3) SOFTWARE OVERVIEW

The purpose of this subsection is to highlight the major components of the developed system and to point out the difficulties and successes experienced during development.

The main aspect of this project is the two feature extraction algorithms but the other components needed in order to use the algorithms were at times more difficult to work with. For example, the pre-processing functions use ImageJ scripting which required learning how to use ImageJ on top of figuring out the best functions to use. While time-consuming, the decision to use ImageJ appears to have paid off as the images being processed are far more useable which means that the resulting matrices are easier to work with.

Perhaps the biggest difficulty came not from the two algorithms but from the segmentation methods. The basic line segmentation implemented was relatively simple to develop, as was the word segmentation which uses the same principle but the cursive character segmentation proved to be very difficult to get working to an acceptable level. The basic idea of splitting the word into even segments and breaking the word at the least populated segments seemed to be a workable plan but the actual implementation proved to be too complicated to be worth pursuing for this project. It was partially developed but not implemented and left to be part of further work.

The major challenges when it comes to the two algorithms were the coding of the island detector (does a character contain an enclosed space?) and how to handle the zones from the first algorithm. The island detector uses an iterative depth-first search to scan through the matrix elements to find connected elements. This method was adapted from an online tutorial which was finding the area of islands of 1s in a sea of 0s. This project takes the same basic idea and uses it to find an island of 0s within a boundary of 1s and unconnected to the edge of the matrix. This was found to work very well for discerning whether a character contains an enclosed space. The zones in the first algorithm pose a greater problem. Originally, because the way the zones are applied depended partially on the size of the matrix, it was possible to pass multiple versions of the same character through and return a different number of zones each time. This was minimised by passing similarly-sized images in to the system but the problem still persisted and rendered the results of the algorithm far less useful than they otherwise could have been. The data for each zone is averaged and then the data for all zones is averaged and this is what is compared. This would yield useful information if the number of zones was static so rather than going with the original design of dynamically allocating zones, the code was changed to split every character into the same number of zones.

The most successful aspect of the development of this system was the two feature extraction algorithms. This is almost certainly due to the heavy use of test-driven development during this stage in development.

9) EXPERIMENT:

The previous section discussed the software development aspect of the project. This section will discuss the experimental aspect. The structure of the experiment will be outlined as well as how the data will be processed.

The aim of the experiment is to discern whether there is a difference in accuracy between the two feature extraction algorithms when they receive the same input. The null hypothesis is that there is no difference in the average accuracy: $H_0: \mu_1 = \mu_2$

The experiment can be outlined by the following steps:

- 1) Gather test data consisting of a sample of the phrase “The quick brown fox jumps over the lazy dog” printed and handwritten by different people.
- 2) Pass these images through the system and record the output.
- 3) Calculate the accuracy of each algorithm for each image, assigning a percentage based on the number of times the correct character appears in the algorithm’s output as either the primary or secondary match.
- 4) Calculate the average accuracy for each algorithm.

The character information to be matched against was built by having volunteers write each character on a plain background. Roughly thirty to forty samples of each character were collected. In cases where characters had distinguishable variants, these variants were separated into their own characters e.g. z and **z**. The collected samples were separated into a single image containing all samples of a single character and these images were then passed through the system. The feature extraction values were calculated and averaged and these averages were used to build the matcher object data which is compared against.

The test data was gathered by having volunteers write the test phrase in printed characters on a plain background. The use of individual character or word text fields was considered but deemed unnecessary as the system is able to segment adequately given the text is not connected. An example of this test data is shown below in Figure 25:

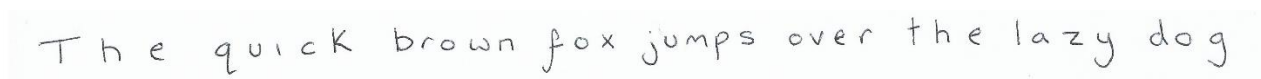


Figure 25 - Example Test Data

Twenty-two test samples were collected and each one was run through the system.

The results from the experiment were recorded in a table, an example of which is shown below in Figure 26:

Character:			T	H	E	Q	U	I	C	K	B	R	O	W	N	F	O	X	...	Score
Match Score	Sample	Algo																		
	S1	A1	1	1	0	1	1	2	1	1	0	0	1							18
		A2	0	0	0	0	0	1	0	0	0	0	1	0						7
	S2	A1	2	0	1	0	0	2	2	1	0	0	1	0	0					15
		A2	2	1	0	0	2	0	1	1	0	0	0	1	0	0				8

Figure 26 - Example Results Table

Each algorithm will get a match score of either 0, 1 or 2 for each character. A score of 0 means the correct character did not appear in the match list, 1 means it appeared as the secondary match and 2 means it appeared as the primary match. These scores will be summed to get the overall score for the algorithm for that sample. These overall scores will be averaged over all samples to get the average accuracy score for algorithm 1 and algorithm 2 which can then be compared. As well as this score table, the returned character lists will be recorded in order to analyse any trends or interesting matches.

10) RESULTS:

The previous section discussed the setup of the experiment and how data collection should be carried out. This section will detail the data collection and handling. The raw data will be discussed and then the processing of this data will be outlined. Finally, the results will be gathered from the processed data.

10.1) MAIN EXPERIMENT

The raw results can be seen in [appendix 1](#). These results show the output of the two algorithms for each sample. Where a '?' appears, the algorithm failed to find a match for that character. It is important to note that a small number of the samples yielded a result that was a different length than expected. This is because the segmentation method either detected a gap in a character where none should be or failed to detect a gap where characters overlapped. This was accounted for in the next step of data handling by analysing the data, comparing it to the input image, deciding where the mistake occurred and shifting the results in the appropriate direction so the rest of the result can still be analysed.

[Appendix 2](#) shows the match data illustrated in Figure 25 in the previous section. This data was gathered by taking the raw results and matching them to the expected output:

T h e q u i c k b r o w n f o x j u m p s o v e r t h e l a z y d o g

If the expected character appeared as the primary result, the match score was recorded as 2. If the expected character appeared as the secondary result, the match score was recorded as 1. If the expected character did not appear in either of the result sets, the match score was recorded as 0. Some of the input images contained uppercase characters so it was decided that these results would be case-insensitive. This means that if the expected character was 'j' and the result was 'J', this was recorded as a match.

The findings of these first two steps of data handling give rise to interesting implications which will be discussed in the next section.

These results show quite a wide range with the lowest score being recorded as 0 and the highest being 21. At this stage it is becoming apparent that there is a marked difference between the outputs of the two algorithms.

Appendix 3 shows the overall results for each of the two algorithms and the average score. The maximum possible score is 70 if every character is matched correctly in the primary result. The average score for algorithm 1 was 5.6 while the average score for algorithm 2 was 16.3. At this point, the null hypothesis can be tested.

Figure 27 shows the total character scores for algorithm 1. From this graph, it can be seen that the characters that were never correctly matched were 'T', 'Q', 'P' and 'Y'. All of the other characters were correctly identified (either as the primary match or the secondary) at least once. 'J' was successfully identified most often but it only had a score of 13 which means that the overall rates of successful identification were low.

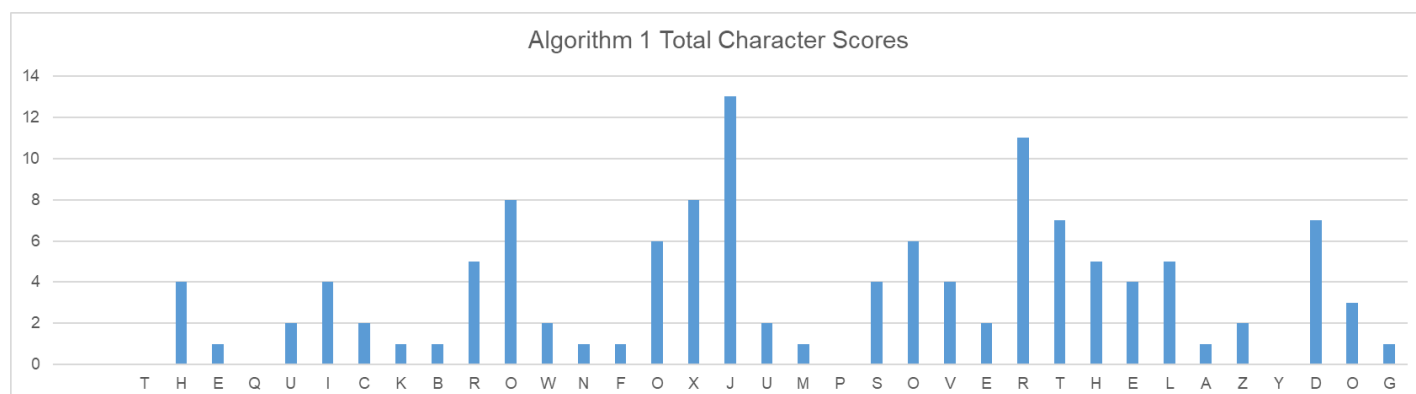


Figure 27- Algorithm 1 Total Character Scores

Figure 28 shows the total character scores for algorithm 2. This graph shows that the characters that were never correctly matched were 'H', 'Q', 'U', 'I', 'B', 'F', 'M', 'V', 'R', 'H', 'L', 'Z', 'Y', and 'G'. It is clear that algorithm 1 correctly matched more characters (algorithm 2 never matched 40% of the input characters) but the rates of successful identification are far higher for algorithm 2; the highest total character score is 40.

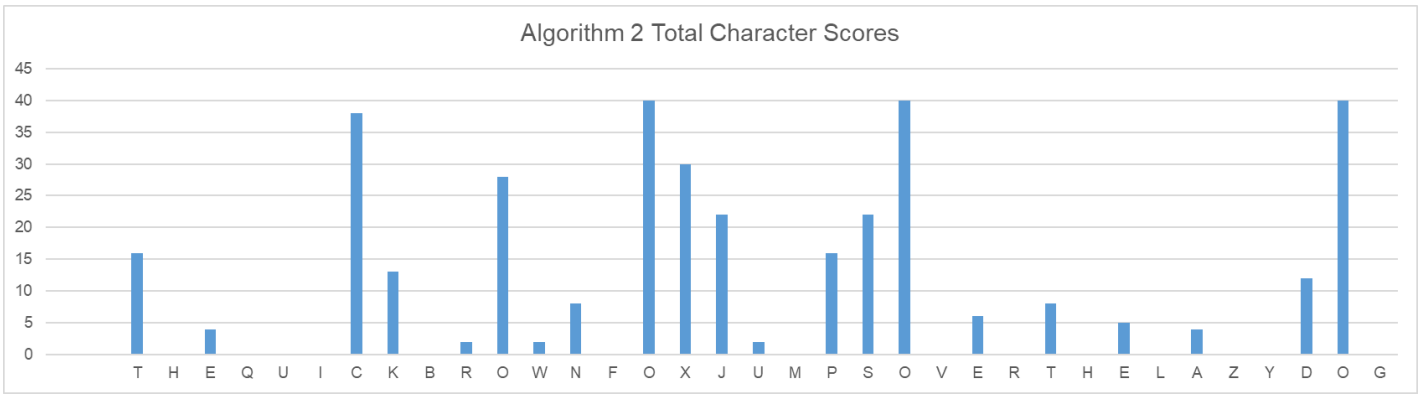


Figure 28 - Algorithm 2 Total Character Scores

Comparing the two graphs of total character scores gives an indication that algorithm 2 might have a higher accuracy score than algorithm 1 but only because it is good at identifying a small subset of the characters. This will be discussed further in the discussion section.

The two samples compared are independent samples from two different populations (the two feature extraction algorithms) so they can be compared using an unpaired t-test[61]. This test takes the average values of two sample populations and determines whether they are statistically different. In this case, the null hypothesis is $H_0: \mu_1 = \mu_2$ so if the means of the scores from the two algorithms are statistically different, the null hypothesis is disproved.

The variances of the samples were calculated in order to determine the appropriate t-test. The variance for algorithm 1 was found to be 10.6 and the variance for algorithm 2 was found to be 10.0. This means that the t-test can be carried out assuming equal variances.

The results of the t-test can be seen in appendix 4. These results show that t-stat is -10.7 and t-crit is 2.0. According to the t-test, in order to qualify as a statistically significant difference, one of the following criteria must be met:

$$t\text{-stat} < -t\text{-crit} \quad \text{or} \quad t\text{-stat} > t\text{-crit}$$

In this case -10.7 is less than -2.0 so this difference is statistically significant. This disproves the null hypothesis $H_0: \mu_1 = \mu_2$ and proves instead that $\mu_1 < \mu_2$.

The end result of this experiment is that the average accuracy score of the first feature detection algorithm is less than that of the second.

10.2) COMBINED ALGORITHM

This subsection will discuss the creation of a third algorithm based on both algorithm 1 and algorithm 2. The results of the main experiment show that there is a significant difference in the accuracy of the two algorithms but neither are particularly accurate. From analysing the results of algorithm 1 and algorithm 2, many possible directions for further investigation have become

apparent. Both algorithms could be investigated further individually and improved vastly but one interesting investigation would be to combine the two. This would result in a single feature extraction algorithm with the following structure (Figure 29):

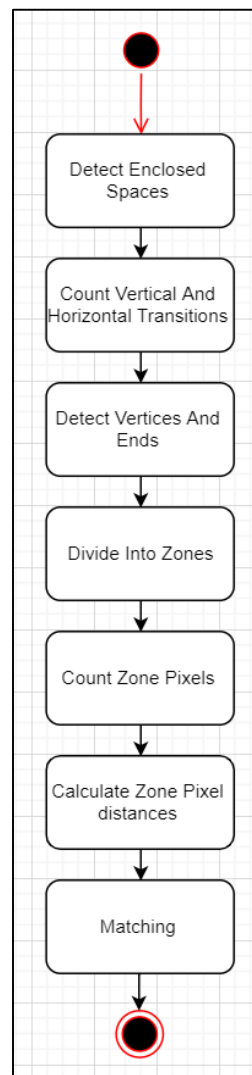


Figure 29 - Flow Diagram Of Combined Algorithm

The matching method here would match the values of the enclosed spaces, transitions and vertices and ends for the whole character and then match the zone values for the remaining features. This means that where the current matcher data for each character contains nine values (four for each algorithm plus the character), the new combined algorithm matcher data would contain fourteen: enclosed spaces, vertical transitions, horizontal transitions, vertices, ends, zone 1 pixels, zone 1 pixel distance, zone 2 pixels, zone 2 pixel distance, zone 3 pixels, zone 3 pixel distance, zone 4 pixels, zone 4 pixel distance, character. This is assuming that the number of zones is always four.

Preliminary results for this work are shown in Figure 30 while the code is shown in [appendix 5](#). The preliminary results for this algorithm appear to show a vast improvement on the previous algorithms. However, due to the incomplete nature of this testing, this is not sufficient to say that this algorithm is better than the others. In order to definitively prove this, complete testing must be

carried out including building the match data based on average values for a set of collected character samples. This is further discussed in the further work section.

This algorithm has been developed and undergone preliminary testing. The results of this testing are shown below in Figure 30.

A3:
 PRIMARY : [F, H, E, Q, U, I, C, K, B, R, O, W, N, F, O, X, J, U, M, P, S, O, V, E, R, F, H, E, L, A, Z, Y, D, O, G]
 SECONDARY: [T, N, C, D, V, T, U, C, E, K, U, B, H, T, Q, C, U, G, W, O, N, Q, U, C, K, S, N, C, I, X, X, V, O, Q, X]

Figure 30 - Combined Algorithm Results

From this, it appears that the combined algorithm is far more accurate than the previous two with only the 'T' being miscategorised as an 'F' in the primary match data. This would mean an accuracy score of 67/70, over 40 points higher than the best of the other algorithms. However, this testing is incomplete with an inadequate sample size and only capital letters from one font being tested. This means that this result cannot be compared to the previous results and it is not possible to say that this algorithm is definitively better than the others at this stage.

11) DISCUSSION:

The previous section discussed the data gathered and the way it was handled as well as the overall result of the experiment. The finding was that the first algorithm has a significantly lower accuracy than the second. This section will discuss the possible reasons for this result and outline some interesting results from the initial tests.

11.1) ALGORITHM 2

From the initial result ([appendix 1](#)) some interesting matches are evident. For example, the extract below (Figure 31) shows the results for sample 4.

S4		PRIMARY : [s, F, c, g, 1, i, r, 9, ?, i, j, j, j, 8, L, 1, 7, 1, 1, 4, L, j, i, V, i, s, M, s, 1, Y, z, 0, E, j, ?]
	1	SECONDARY: [c, p, a, b, v, i, v, J, ?, h, i, i, i, 5, n, v, T, i, v, f, v, i, h, o, r, c, E, l, v, L, s, ?, 4, i, ?]
		PRIMARY : [X, T, O, d, c, c, c, X, P, c, o, c, c, ?, c, X, j, c, c, O, c, o, c, O, c, X, T, P, c, P, c, 5, A, o, d]
	2	SECONDARY: [k, E, ?, d, S, S, S, k, p, S, ?, S, S, ?, S, k, ?, S, S, ?, S, ?, S, ?, S, k, E, p, S, p, S, n, A, ?, d]

Figure 31 - Raw Results Example A2

This sample was one of the worst scoring but the results for the second algorithm are often wrong in an interesting way. To compare some interesting results to the expected values see Figure 32:

Expected	q	u	i	c	k	b	r	o	w	n	f	o	x	j	u	m	p	s	o	v	e	r	d	o	g
Actual 1st	d	c	c	c	X	P	c	o	c	c	?	c	X	j	c	c	0	c	o	c	0	c	A	o	d
2nd	s	S	S	S	k	p	s	?	S	S	?	S	k	?	S	S	?	S	?	S	?	S	A	?	d

Figure 32 - Results Extract

The correct matches have been shaded in green. These matches are interesting because when one match is correct, the second match shows a character that can be easily mistaken for that match. For example, the 'X' can be mistaken for a 'k' and the 'c' can be mistaken for an 'S'. This is consistent with the way the second algorithm extracts features. The image below (Figure 33) illustrates the similarities between 'X' and 'k':

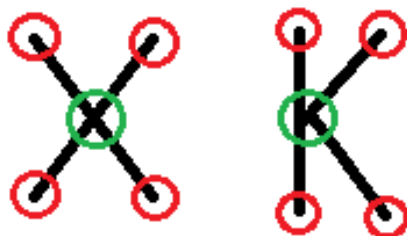


Figure 33 - X and k comparison

The ends have been circled in red and the vertices have been circled in green. Both 'X' and 'k' have four ends and one vertex. The number of lines is calculated by the algorithm to be two in both characters. According to algorithm 2, 'X' and 'k' are the same character. The only reason more characters such as 'x' and 'K' are not also included is because of the way the matcher data was collected and averaged which means that small variations in the way these characters are written causes the averaged data to be slightly different.

In the case of characters that were mismatched such as the 'g' in 'dog' being categorised as 'd', the same problem is the cause. Both 'd' and 'g' have one (or occasionally two) end, two lines and one enclosed space. The same applies to 'p', 'b', 'q' and cursive 'o'. The second algorithm has no real way of distinguishing these characters. If the sample results are compared now using this information, the result is shown in Figure 34:

Expected	q	u	i	c	k	b	r	o	w	n	f	o	x	j	u	m	p	s	o	v	e	r	d	o	g
Actual 1st	d	c	c	c	X	P	c	o	c	c	?	c	X	j	c	c	0	c	o	c	0	c	A	o	d
2nd	s	S	S	S	k	p	s	?	S	S	?	S	k	?	S	S	?	S	?	S	?	S	A	?	d

Figure 34 - Results Extract

Now, in addition to the correct matches being shaded in green, the incorrect but identical matches are shaded in orange. This shows that the algorithm could have a much greater accuracy if additional features were detected which could distinguish these characters.

The inaccuracy of the second algorithm is very interesting in that it very easily exposes the flaw in the feature extraction method. It is likely that further work focusing on extracting additional features would result in greatly improved accuracy.

11.2) ALGORITHM 1

The inaccuracy of the first algorithm is less easily explained than the second algorithm. From initial design, it was expected that the first algorithm would be more accurate than the second because of the use of zones but this was evidently not the case. Analysis of the best results for algorithm 1 show some interesting results as seen in Figure 35 and Figure 36:

SS	1	PRIMARY : [J, 0, y, 0, V, h, V, 7, 5, R, U, K, H, 7, F, n, O, O, 9, 4, b, b, T, E, R, 7, M, e, x, G, q, n, 8, z, k] SECONDARY: [I, Z, b, N, o, b, c, K, B, d, o, q, A, ?, m, l, ?, h, k, M, a, 6, t, A, R, ?, G, M, n, k, p, c, 0, o, e]
	2	PRIMARY : [T, X, 5, X, c, ?, c, X, o, d, c, c, 5, T, O, X, 5, c, c, O, c, o, c, T, d, T, X, T, c, A, c, 5, R, o, c] SECONDARY: [E, k, n, k, S, ?, S, k, ?, d, S, S, n, E, ?, k, n, S, S, ?, S, ?, S, E, d, E, k, E, S, A, S, n, ?, ?, S]

Figure 35 - Raw Results Example A1

Expected	T	h	e	q	u	i	c	k	b	r	o	w	n	f	o	x	j	u	m	p	s	o	v	e	r
Actual 1st	J	0	y	0	V	h	V	7	5	R	U	K	H	7	F	n	O	O	9	4	b	b	T	E	R
2nd	I	Z	b	N	o	b	c	K	B	d	o	q	A	?	m	l	?	h	k	M	a	6	t	A	R

Figure 36 - Results Extract

Again, the correct matches have been shaded in green. While the use of zones for the first algorithm was expected to yield more specific information about the characters, it is also the most likely culprit for causing the inaccuracy. Or rather, the handling of the data gathered from these zones is the problem. Algorithm 1 first splits the character into zones, then calculates the pixels, distances, transitions and islands within each zone. This is valid data as it could be used to distinguish between a character with pixels in the top right quadrant but not in the bottom left (e.g. 'q') and one with pixels in the bottom left quadrant but not the top right (e.g. 'b'). Unfortunately, some of these parameters are less useful than others. For example, finding the enclosed spaces in the top right quadrant is far less useful than simply finding the enclosed spaces in the whole character as the second algorithm does. There is no good reason to mistake a 'j' or a 'u' for an 'O'. Much like the discussion of the second algorithm, the results extract can be compared to easily confused matches to better understand how these mismatches happen (Figure 37):

Expected	T	h	e	q	u	i	c	k	b	r	o	w	n	f	o	x	j	u	m	p	s	o	v	e	r
Actual 1st	J	0	y	0	V	h	V	7	5	R	U	K	H	7	F	n	O	O	9	4	b	b	T	E	R
2nd	I	Z	b	N	o	b	c	K	B	d	o	q	A	?	m	l	?	h	k	M	a	6	t	A	R

Figure 37 - Results Extract

Some of the more obvious mismatches have been shaded in orange. These have been selected to highlight to way that the white-black transitions only being calculated horizontally across the character gives rise to simple mismatches. A 'J' will commonly have one white-black transition when scanning from one side to the other. A 'T' (the expected match) will also have one transition, as will an 'l'.

The confusion between 'u' and 'V' is easy to see but mistaking a 'u' for an 'o' happens because of the lack of vertical transition detection and the misuse of island detection. If enclosed spaces were detected for the whole character, this mismatch would not happen.

Many of the other mismatches are a result of the averaged pixel numbers and distances being similar. It appears from analysing these results that this is not enough information to adequately categorise a character.

11.3) COMBINED ALGORITHM DISCUSSION

The combined algorithm addresses some of the aforementioned issues with algorithms 1 and 2. The vertical and horizontal white-black transition detection and the whole-character enclosed space detection focus on the stated failings of algorithm 1. The addition of zone analysis addresses the main failing of algorithm 2; lack of distinguishing character features detected.

The preliminary results (Figure 30), indicate that the combined algorithm shows promise. It appears that the combined algorithm could be a vast improvement over its predecessors which also implies that the analysis of the results was correct. If the combined algorithm, developed to address the issues with algorithm 1 and 2, is an improvement, then the suspected causes of algorithm 1 and 2's inaccuracy are likely the actual causes. Not only does this mean that the combined algorithm could be fully tested and may prove to be a sufficiently accurate feature extraction algorithm, it also means that improvements could be made to algorithm 1 and 2 focussing on these identified failings.

This section has investigated the possible causes of some of the unexpected results as well as discussing the results of the combined algorithm which was posed as a possible improvement. With these causes outlined, possible solutions can be posed as further work. This will take place in the further work section. The next section will conclude the project as it stands before further work is considered.

12) CONCLUSION:

This project has investigated two methods of feature extraction by building a system containing two feature extraction algorithms: one based on Sharma's work (algorithm 1) and the other based on Khaustov's work (algorithm 2). The aim was to compare the accuracy of the two algorithms and gain insight into feature extraction as a process. It was found that algorithm 2 was far more accurate than algorithm 1 but both left great room for improvement. When these results were analysed it was found that the likely reason for the inaccuracy of algorithm 2 was a lack of defining features extracted which can be easily improved upon. The inaccuracy of algorithm 1 was found to be more likely due to an ineffective order of functions which requires further research.

A third, combined, algorithm was created in order to address some of the shortcomings of the two original algorithms. This combined algorithm was preliminarily tested and appears to be a vast improvement on its predecessors but further experimentation is required in order to confirm this.

13) REFLECTION:

In this section, I will reflect on the project as a whole, outlining the highlights and disappointments or failings in my opinion.

At the beginning of this experiment, I had limited experience with java coding and testing and this is apparent in the varying quality of the different classes in the code. The first classes to be developed (the application class and the pre-processing class) are lacking in reusability and contain sections of code that could easily be abstracted out into their own methods to increase readability. The later classes (such as the algorithm classes) show a vast improvement. These classes were developed using test-driven development so the code contains many methods that each carry out a specific function. I found that focusing on code testability was the main thing that helped to improve my coding.

As mentioned before, the character segmentation class could not be implemented. This was the biggest disappointment of the project. The research and planning had been completed, the code had been developed but I was ultimately unable to develop it to an adequate level. Having already spent longer than planned working on this single aspect of the development process and having not yet developed the algorithms, I made the decision to shelve character segmentation for possible revisiting at a later date. I think that my lack of programming skill caused this setback and that if I were to start the character segmentation class from scratch now, I would have a less difficult time developing it.

The things I am proudest of in this project are the algorithms. The unit tests were very useful in ensuring that I understood exactly what each of my methods was doing. Though Sharma and Khaustov's works were simplified for this project, I feel that they provided a good base to work from and allowed me to explore the different ways a character can be identified.

Above all, I think the main things I gained from this project are as follows:

- 1) Java coding experience
- 2) Understanding of the value of unit testing
- 3) A far deeper understanding of forms processing

Having completed this project, my understanding of character recognition, feature extraction and forms processing has gone from basically nothing to the point where I have developed a system using some of the features. Even for the aspects that I did not use in the development of this system (such as machine learning), researching how existing systems work has given me some understanding of their value and function. I have gained a wide range of knowledge from researching forms processing and from experimenting with it and the further work section emphasises that this is an ongoing process; there are still many improvements to be made.

14) FURTHER WORK:

As mentioned in the discussion section above, the two algorithms are not particularly accurate and leave substantial room for improvement. The main shortcoming of algorithm 2 was identified as a lack of specificity which could be solved by extracting additional features. The main shortcoming of algorithm 1 was identified as the order of the methods. This section will outline some possible further work and briefly explore some of the more readily implemented improvements.

As part of the literature review, it was stated that most character recognition systems use some form of machine learning but this was deemed to be unnecessary for the scope of this project. As part of further work, this would be a useful avenue to explore. Where most systems use machine learning and masks to segment and match characters, this further work could explore using machine learning in conjunction with the feature extraction algorithms in order to improve the matching process. Matching currently works by comparing averages but machine learning could be used to build the matcher object data and replace the threshold comparison in the matching class.

One of the major pitfalls of this project was the failure to implement an adequate character segmentation method. The current (unused) version results in looped sections of the input matrix being added at the end of the characters. The defects in this method should be fixable with some effort and testing so the fixing and implementation of the character segmentation method should form part of further work. Once implemented, this will allow the system to analyse cursive handwriting, introducing new avenues of research.

Many of the works mentioned in the literature review discussed methods of pre-processing the input images. Much of this was deemed to be outside the scope of this project and any processing beyond the implemented imageJ scripts used in the pre-processing class was done manually. For example, the input images can possibly be wildly different sizes which could change the outcome of the experiment. This means that further work on the pre-processing stage such as resizing images and de-skewing the text would result in a more robust system.

As for the aforementioned shortcomings of the two algorithms, a large amount of further work could be done to improve these. For one, algorithm 2 could be investigated in greater depth to find a suitable extra feature to extract which would result in less character confusion. Algorithm 1 could be reworked to extract enclosed spaces from the whole character while using zones for other features. It could also be altered so that rather than comparing average zone values, the positions of the zones are taken into account. The average percentage zone pixel coverage could also be a more useful metric than the average number of zone pixels.

The most pressing item of further investigation is the full analysis of the combined algorithm. Preliminary results appear promising but it is vital to fully test this algorithm in the same way that the other two were tested in order to acquire a fair and accurate result. The matcher data for the combined algorithm must be created by averaging the results of a set of collected data (preferably, this should be done using the same collected sample as was used by the other algorithms), the experiment must be carried out multiple times for multiple samples (again, this can be done using the existing collected samples) and the results should then be analysed in the same manner as the previous experiment, comparing the accuracy scores of the algorithms. Only after this full investigation can it be said that the combined algorithm was or was not a significant improvement on the individual algorithms.

REFERENCES:

- [1] M. D, "Forms Processing - User Experiences of Text and Handwriting Recognition (OCR/ICR)," 2012. [Online]. Available: www.aiim.org/pdfdocuments/MIWP_Forms-Processing_2012.pdf.
- [2] X. Lu, S. Kataria, W. Brouwer, J. Wang, P. Mitra, and C. L. Giles, "Automated Analysis of Images in Documents fir Intelligent Document Search," 2009.
- [3] R. D. Felipe and L. A. P. Neves, "Pre-Printed and Hand-Filled Table-Form Analysis Aiming Cell Extraction," in *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, 2008, pp. 439–443.
- [4] S. Golubev, "'Construction of Model of Structured Documents Based on Machine Learning', in Pattern recognition and machine intelligence : 4th international conference, PReMI 2011, Moscow, Russia, June 27 - July 1, 2011 : proceedings," 2011, p. 424.
- [5] "CAPTURING SEMI-STRUCTURED FORMS AND DOCUMENTS: CHALLENGES AND AVAILABLE TECHNOLOGIES," 2005. [Online]. Available: https://c.ymcdn.com/sites/www.bfma.org/resource/resmgr/articles/05_04.pdf. [Accessed: 23-May-2019].
- [6] S. Mozaffari and P. Bahar, "Farsi/Arabic Handwritten from Machine-Printed Words Discrimination," in *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 698–703.
- [7] S. N. Srihari, Yong-Chul Shin, V. Ramanaprasad, and Dar-Shyang Lee, "Name and Address Block Reader system for tax form processing," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 5–10.
- [8] Zhi-Hong Zhao, Xue-Dong Tian, and Bao-Lan Guo, "A study on printed form processing and reconstruction," in *Proceedings. International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 1730–1732.
- [9] S. Mandal, S. Sur, A. Dan, and P. Bhowmick, "Handwritten Bangla character recognition in machine-printed forms using gradient information and Haar wavelet," in *2011 International Conference on Image Information Processing*, 2011, pp. 1–6.
- [10] P. Mahasukhon, H. Mousavinezhad, and J.-Y. Song, "Hand-printed English character recognition based on Fuzzy theory," in *2012 IEEE International Conference on Electro/Information Technology*, 2012, pp. 1–4.
- [11] F. Yasmin, S. M. M. Hossain, and M. S. Arefin, "APSD: A framework for automated processing of survey documents," in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2017, pp. 411–416.
- [12] D. Brodic and A. Amelio, "Classification of the hand-printed and printed Medieval Glagolitic documents using differentiation in orthography," in *2017 40th International Convention on*

Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017, pp. 1110–1115.

- [13] K.-H. Steinke, "Improvement of Omnipage 18's efficiency," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2012, pp. 434–438.
- [14] "reCAPTCHA: Easy on Humans, Hard on Bots." [Online]. Available: <https://www.google.com/recaptcha/intro/v3.html>. [Accessed: 26-May-2019].
- [15] S. Gunter and H. Bunke, "Optimizing the number of states, training iterations and Gaussians in an HMM-based handwritten word recognizer," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, vol. 1, pp. 472–476.
- [16] A. M. Sabu and A. S. Das, "A Survey on various Optical Character Recognition Techniques," in *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, 2018, pp. 152–155.
- [17] A. M. El-Gabaly, P. Liang, M. Barth, and E. Tuncel, "UNIVERSITY OF CALIFORNIA RIVERSIDE Expectation and Knowledge Guided Neural Networks for Image Recognition," 2016.
- [18] S. Afroge, B. Ahmed, and F. Mahmud, "Optical character recognition using back propagation neural network," in *2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, 2016, pp. 1–4.
- [19] T. C. Wei, U. U. Sheikh, and A. A.-H. A. Rahman, "Improved optical character recognition with deep neural network," in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, 2018, pp. 245–249.
- [20] M. M. Farhad, S. M. N. Hossain, A. S. Khan, and A. Islam, "An efficient Optical Character Recognition algorithm using artificial neural network by curvature properties of characters," in *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, 2014, pp. 1–5.
- [21] B. Vani, M. S. Beaulah, and R. Deepalakshmi, "High accuracy optical character recognition algorithms using learning array of ANN," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, 2014, pp. 1474–1479.
- [22] K. Tsegay Fitsum and Y. Patel, "Optical Character Recognition for Tigrigna Printed Documents Using HOG and SVM," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 1489–1494.
- [23] J. Xie, "Optical Character Recognition Based on Least Square Support Vector Machine," in *2009 Third International Symposium on Intelligent Information Technology Application*, 2009, pp. 626–629.
- [24] D. Pop, B. Irimie, and D. Petcu, "Distributed Optical Character Recognition for Old Romanian Prints," in *2017 19th International Symposium on Symbolic and Numeric*

Algorithms for Scientific Computing (SYNASC), 2017, pp. 416–418.

- [25] T. K. Hazra, D. P. Singh, and N. Daga, "Optical character recognition using KNN on custom image dataset," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, 2017, pp. 110–114.
- [26] P. Romulus, Y. Maraden, P. D. Purnamasari, and A. A. P. Ratna, "An analysis of optical character recognition implementation for ancient Batak characters using K-nearest neighbors principle," in *2015 International Conference on Quality in Research (QiR)*, 2015, pp. 47–50.
- [27] A. Dawoud, "Iterative Cross Section Sequence Graph for Handwritten Character Segmentation," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2150–2154, Aug. 2007.
- [28] D. Brodic, "Advantages of the Extended Water Flow Algorithm for Handwritten Text Segmentation," in *Pattern Recognition and Machine Learning*, 2011, p. 418.
- [29] N. Q. Nguyen, Q. T. Nguyen, H. V. D. Nguyen, T. D. Chau, and T. A. Tran, "Automatic Extract Handwriting Marked Regions in Business Document Images," in *2018 International Conference on Advanced Computing and Applications (ACOMP)*, 2018, pp. 112–116.
- [30] V.-C. R. Ting, "Separation and recognition of connected handprinted capital English characters," 1986.
- [31] S. Pachpande and A. Chaudhari, "Implementation of devanagri character recognition system through pattern recognition techniques," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 2017, pp. 717–722.
- [32] J. Song, Z. Li, M. R. Lyu, and S. Cai, "Recognition of Merged Characters Based on Forepart Prediction, Necessity-Sufficiency Matching, and Character-Adaptive Masking," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 35, no. 1, pp. 2–11, Feb. 2005.
- [33] A. M. Ashir and G. S. Shehu, "Adaptive clustering algorithm for optical character recognition," in *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2015, p. SG-13-SG-16.
- [34] J. Cowell and F. Hussain, "Amharic character recognition using a fast signature based algorithm," in *Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003.*, pp. 384–389.
- [35] N. Sharma, B. Kumar, and V. Singh, "Recognition of off-line hand printed English Characters, Numerals and Special Symbols," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, 2014, pp. 640–645.
- [36] P. A. Khaustov, V. G. Spitsyn, and E. I. Maksimova, "Algorithm for optical handwritten characters recognition based on structural components extraction," in *2016 11th International Forum on Strategic Technology (IFOST)*, 2016, pp. 355–358.

- [37] J. Reed, "An Optical Character Recognition Engine for Graphical Processing Units," *Theses Diss. Sci.*, Jan. 2016.
- [38] L. Niharmine, B. Outtaj, and A. Azouaoui, "Tifinagh handwritten character recognition using genetic algorithms," in *2018 International Conference on Advanced Communication Technologies and Networking (CommNet)*, 2018, pp. 1–6.
- [39] D. Decoste and B. Schölkopf, "Training Invariant Support Vector Machines," *Mach. Learn.*, vol. 46, no. 1/3, pp. 161–190, 2002.
- [40] M. M. N. Mayekar and M. S. Kuwelkar, "Implementation of machine learning algorithm for character recognition on GPU," in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, 2017, pp. 470–474.
- [41] N. Garcia-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "COVNET: a cooperative coevolutionary model for evolving artificial neural networks," *IEEE Trans. Neural Networks*, vol. 14, no. 3, pp. 575–596, May 2003.
- [42] G. Lu and R. Guo, "Cascaded classifier for improving traffic classification accuracy," *IET Commun.*, vol. 11, no. 11, pp. 1751–1758, Aug. 2017.
- [43] "ImageJ." [Online]. Available: <https://imagej.nih.gov/ij/>. [Accessed: 31-Jul-2019].
- [44] "ImageJ." [Online]. Available: <https://imagej.net/Welcome>. [Accessed: 31-Jul-2019].
- [45] "IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains." [Online]. Available: <https://www.jetbrains.com/idea/>. [Accessed: 31-Jul-2019].
- [46] "OpenCV." [Online]. Available: <https://opencv.org/>. [Accessed: 06-Aug-2019].
- [47] F. K. Noble, "Comparison of OpenCV's feature detectors and feature matchers," in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2016, pp. 1–6.
- [48] R. Vaidya, D. Trivedi, S. Satra, and P. M. Pimpale, "Handwritten Character Recognition Using Deep-Learning," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 772–775.
- [49] S. Shelke and S. Apte, "Real-time character reading system for marathi script using raspberry Pi," in *3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016)*, 2016.
- [50] A. Nectoux, "Matrices and Digital Images | Klein Project Blog," 2015. [Online]. Available: <http://blog.kleinproject.org/?p=588>. [Accessed: 24-May-2019].
- [51] J. Dudovskiy, "Research Philosophy - Research Methodology." [Online]. Available: <https://research-methodology.net/research-philosophy/>. [Accessed: 20-Aug-2019].
- [52] C. Drumond, "Scrum - what it is, how it works, and why it's awesome." [Online]. Available: <https://www.atlassian.com/agile/scrum>. [Accessed: 21-Aug-2019].
- [53] M. R. Borth and H. Y. Shishido, "A Comparative Analysis of Two Software Development

- Methodologies: Rational Unified Process and Extreme Programming,” *Rev. Vértices*, vol. 15, no. 3, pp. 143–157, Jan. 2013.
- [54] F. Buschmann, “Learning from Failure, Part 1: Scoping and Requirements Woes,” *IEEE Softw.*, vol. 26, no. 6, pp. 68–69, Nov. 2009.
- [55] S. Puhl and R. Fahney, “How to assign cost to ‘avoidable requirements creep’: A step towards the waterfall’s agilization,” in *2011 IEEE 19th International Requirements Engineering Conference*, 2011, pp. 307–312.
- [56] M. Pomeroy-Huff, R. Cannon, T. Chick, J. Mullaney, and W. Nichols, “The Personal Software Process (PSP) Body of Knowledge, Version 2.0,” 2009.
- [57] A. Adel and B. Abdullah, “A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 12, no. 1, pp. 106–111, 2015.
- [58] R. Piplani and A. W. H. Ang, “Performance comparison of multiple product kanban control systems,” *Int. J. Prod. Res.*, vol. 56, no. 3, pp. 1299–1312, Feb. 2018.
- [59] D. Salvi, “Document Image Analysis Techniques for Handwritten Text Segmentation, Document Image Rectification and Digital Collation.,” University of South Carolina - Columbia, 2014.
- [60] “Max Area of Island - LeetCode.” [Online]. Available: <https://leetcode.com/problems/max-area-of-island/solution/>. [Accessed: 12-Nov-2019].
- [61] “THE PROBABLE ERROR OF A MEAN,” *Biometrika*, vol. 6, no. 1, pp. 1–25, Mar. 1908.

APPENDICES:

Appendix 1 - Raw results

Sample	Algo	Result
S1	1	PRIMARY : [R, F, x, 0, L, ?, 1, f, S, r, X, X, r, R, 1, ?, v, s, 1, 7, ?, 1, X, r, n, 7, h, U, X, 7, v, 1, 0, h, j, ?]
		SECONDARY: [f, p, c, w, v, ?, e, R, j, n, v, V, n, f, v, ?, r, c, c, 4, ?, Y, V, n, c, ?, b, w, V, 4, 1, v, j, b, i, ?]
		2 SECONDARY: [E, n, p, 9, E, S, S, k, p, S, ?, S, S, S, ?, S, ?, ?, E, n, p, S, ?, S, p, S, k, n, p, S, p, n, n, d, ?, ?]
S2	1	PRIMARY : [R, S, ?, ?, O, ?, ?, O, S, B, 4, A, R, ?, W, U, S, h, A, f, q, O, j, G, 0, O, E, A, S, 4, 4, n, 8, I, R]
		SECONDARY: [?, B, ?, ?, j, ?, ?, j, 3, f, P, y, ?, ?, S, o, B, b, f, R, p, j, h, k, y, j, b, d, 7, E, F, c, 3, k, f]
		2 SECONDARY: [n, ?, E, d, S, ?, S, ?, ?, d, ?, S, n, n, ?, k, n, S, n, 9, S, ?, S, ?, A, E, ?, E, S, ?, S, E, 9, ?, S]
S3	1	PRIMARY : [E, q, o, O, L, 1, r, M, O, i, j, j, j, S, X, 1, T, 1, u, ?, s, X, 1, j, i, B, L, 1, n, 1, O, M, 1, ?]
		SECONDARY: [A, p, h, j, x, ?, 1, P, j, r, j, j, i, B, L, V, s, c, s, ?, 1, n, a, i, i, f, n, c, c, c, ?, P, c, ?]
		2 SECONDARY: [n, n, ?, p, E, S, S, k, p, S, S, S, E, ?, ?, k, ?, E, E, p, S, ?, S, ?, S, ?, ?, S, n, ?, ?, p]
S4	1	PRIMARY : [S, F, c, g, 1, i, r, 9, ?, i, j, j, j, 8, L, 1, 7, 1, 1, 4, L, j, i, V, i, s, M, s, 1, Y, z, O, E, j, ?]
		SECONDARY: [c, p, a, b, v, i, v, j, ?, h, i, i, i, S, n, v, T, i, v, f, v, i, h, o, r, c, E, 1, v, L, s, ?, 4, i, ?]
		2 SECONDARY: [k, E, ?, d, S, S, S, k, p, S, ?, S, S, ?, S, k, ?, S, S, ?, S, ?, S, ?, S, k, E, p, S, p, S, n, A, ?, d]
S5	1	PRIMARY : [J, 0, y, 0, V, h, V, 7, S, R, U, K, H, 7, F, n, O, 9, 4, b, b, T, E, R, 7, M, e, x, G, q, n, 8, z, k]
		SECONDARY: [I, z, b, N, o, b, c, K, B, d, o, q, A, ?, m, 1, ?, h, k, M, a, 6, t, A, R, ?, G, M, n, k, p, c, 0, e]
		2 SECONDARY: [E, k, n, k, S, ?, S, k, ?, d, S, S, n, E, ?, k, n, S, S, ?, S, E, d, E, k, E, S, A, S, n, ?, ?, S]
S6	1	PRIMARY : [Q, F, j, O, 1, r, j, ?, E, i, L, X, v, 3, L, 1, o, 1, j, 9, j, j, i, j, 1, V, o, j, i, j, j, j, j, ?]
		SECONDARY: [g, p, O, j, v, 1, i, ?, A, c, v, V, 1, D, n, c, h, c, i, h, i, j, j, i, h, c, c, h, i, i, i, j, b, j, ?]
		2 SECONDARY: [E, E, ?, ?, S, S, S, ?, ?, S, ?, S, S, k, ?, k, ?, S, S, ?, S, ?, S, E, n, ?, E, ?, S, ?, n, E, k, S, ?]
S7	1	PRIMARY : [S, ?, ?, ?, U, ?, j, 9, S, E, K, e, h, 7, O, X, ?, S, F, 7, j, O, X, w, A, s, I, f, X, k, u, 1, Q, 2, q]
		SECONDARY: [I, ?, ?, h, ?, O, h, 3, A, a, b, 2, j, n, ?, 1, m, s, i, j, V, k, d, c, k, d, V, a, c, s, f, d, k, p]
		2 SECONDARY: [n, ?, n, 9, S, k, S, N, ?, 9, ?, E, E, E, ?, k, n, S, ?, ?, S, ?, n, E, d, n, k, k, S, ?, S, n, 9, S, S]
S8	1	PRIMARY : [O, w, 7, k, 1, r, 1, 6, Z, i, i, o, L, 9, 1, 1, j, V, u, 6, j, s, j, s, r, s, U, j, j, j, j, 0, 0]
		SECONDARY: [j, h, s, e, a, 1, e, A, y, j, h, h, x, h, x, v, h, n, s, H, ?, 1, ?, 7, 1, c, o, h, i, i, i, O, j, ?]
		2 SECONDARY: [E, n, ?, p, E, S, S, ?, ?, S, S, S, ?, ?, E, E, p, S, ?, S, k, E, p, S, 9, k, n, d, p, p]
S9	1	PRIMARY : [h, f, 1, k, 1, 1, 1, S, E, i, j, V, 1, 4, u, v, j, j, ?, O, ?, ?, j, i, u, q, L, 1, c, V, B, 6, 1, B]
		SECONDARY: [b, p, a, e, Y, ?, ?, j, 4, h, i, o, v, E, t, 1, i, i, ?, j, ?, ?, j, h, t, p, v, a, c, f, G, L, g]
		2 SECONDARY: [k, E, ?, d, n, S, S, ?, d, n, S, ?, n, k, ?, ?, n, S, p, S, n, ?, S, ?, E, ?, S, p, n, ?, d, ?, ?]
S10	1	PRIMARY : [?, p, ?, e, ?, ?, j, B, A, i, L, w, L, 8, T, X, O, 7, j, O, s, ?, j, i, z, 9, j, o, j, j, 8, y, j, 3]
		SECONDARY: [?, M, ?, d, ?, ?, i, f, y, i, o, h, v, 0, z, n, j, f, h, j, n, ?, ?, j, j, y, j, h, h, h, i, 3, b, i, D]
		2 SECONDARY: [?, s, j, j, T, c, c, X, O, c, o, c, c, ?, O, X, j, S, 5, P, c, o, 5, d, c, ?, T, O, 5, d, j, S, R, o, P]
S11	1	PRIMARY : [O, w, 7, k, 1, 1, 1, S, E, i, j, V, 1, 4, u, v, j, j, ?, O, ?, ?, j, i, u, q, L, 1, c, V, B, 6, 1, B]
		SECONDARY: [j, 4, j, b, V, 1, h, ?, f, i, L, u, 1, W, z, n, j, s, ?, j, F, ?, j, i, i, 1, a, q, v, V, v, j, M, v, L, ?]
		2 SECONDARY: [n, n, d, p, E, S, S, k, d, E, ?, ?, n, ?, j, k, ?, E, k, ?, S, ?, n, E, n, S, k, n, ?, S, p, S, E, d, ?, p]
S12	1	PRIMARY : [E, n, p, 9, E, S, S, k, p, S, S, S, E, ?, ?, k, ?, E, E, ?, f, O, 0, 5, 7, m, E, V, 4, 7, ?, 8, U, U]
		SECONDARY: [f, O, b, B, e, f, p, A, ?, 0, j, j, 0, j, y, w, ?, a, 4, A, ?, ?, j, b, B, s, a, 4, c, E, F, ?, 0, 2, K]
		2 SECONDARY: [n, ?, n, d, S, ?, S, N, ?, d, ?, S, n, E, j, k, E, S, n, ?, S, S, n, ?, S, n, k, E, S, A, S, n, ?, S]
S13	1	PRIMARY : [w, h, s, I, v, r, 1, j, P, ?, ?, L, L, O, 1, 1, O, 1, j, O, ?, 7, 1, ?, O, 1, ?, h, j, X, j, j, O, j, ?]
		SECONDARY: [a, b, c, b, 1, 1, a, I, e, ?, ?, n, o, j, v, v, j, v, i, j, ?, f, a, ?, j, c, ?, b, i, V, i, i, ?, j, j, ?]
		2 SECONDARY: [k, n, ?, d, S, S, k, p, S, S, S, S, ?, j, k, ?, S, S, ?, S, n, ?, S, k, n, ?, S, p, S, E, j, ?, ?]
S14	1	PRIMARY : [X, S, O, d, c, c, c, X, P, c, c, c, c, ?, o, X, j, c, c, O, c, j, c, S, O, c, X, S, O, c, P, c, T, o, O]
		SECONDARY: [k, n, ?, d, S, S, k, p, S, S, S, S, ?, j, k, ?, S, S, ?, S, n, ?, S, k, n, ?, S, p, S, E, j, ?, ?]
		2 SECONDARY: [c, S, T, O, O, c, c, c, X, O, c, o, S, T, X, o, X, c, c, T, O, c, o, S, O, c, j, S, ?, c, P, X, S, 9, o, 9]
S15	1	PRIMARY : [S, n, E, ?, ?, S, S, S, k, ?, S, ?, n, E, k, ?, ?, k, S, S, E, ?, S, ?, n, ?, S, ?, p, k, n, 9, ?, ?]
		SECONDARY: [?, n, ?, p, E, S, S, k, p, S, S, S, S, ?, j, k, E, S, S, ?, S, n, S, N, n, d, S, ?, n, n, p, ?]
		2 SECONDARY: [O, U, 1, g, L, 1, r, ?, e, i, 1, L, v, 3, n, 1, O, j, O, b, j, j, j, j, r, q, q, L, X, X, L, B, H, L, B]
S16	1	PRIMARY : [j, K, c, b, v, ?, n, ?, a, i, v, v, 1, D, c, c, j, ?, j, S, j, i, i, i, i, p, p, v, v, V, v, f, G, n, f]
		SECONDARY: [j, T, 9, 9, S, c, c, X, 9, c, o, c, c, X, o, j, j, S, 5, O, c, o, c, P, S, j, T, 9, c, o, V, d, o, O]
		2 SECONDARY: [?, e, 9, 9, n, S, S, k, 9, S, S, S, k, ?, ?, n, n, ?, S, ?, S, p, n, ?, E, 9, S, ?, S, E, d, ?, ?]
S17	1	PRIMARY : [9, O, O, w, ?, ?, h, 8, D, h, 0, k, K, k, t, E, M, E, w, j, 4, 7, ?, E, 7, M, k, X, h, U, n, d, q, T]
		SECONDARY: [W, j, j, j, h, ?, ?, b, 3, g, b, y, e, q, a, s, A, e, A, h, I, E, 1, ?, f, F, e, e, L, b, o, 1, b, p, s]
		2 SECONDARY: [S, X, T, ?, T, j, c, X, ?, 9, o, j, N, T, o, j, S, 5, T, P, c, o, c, S, d, T, ?, T, c, ?, S, 5, ?, o, c]
S18	1	PRIMARY : [n, k, E, ?, E, ?, S, k, ?, 9, ?, ?, N, E, ?, ?, n, n, E, p, S, ?, S, n, d, E, ?, E, S, ?, n, n, ?, S, S]
		SECONDARY: [?, n, ?, p, E, S, S, k, p, S, S, S, S, ?, j, k, E, S, S, ?, S, n, S, N, n, d, S, ?, n, n, p, ?]
		2 SECONDARY: [O, U, 1, g, L, 1, r, ?, e, i, 1, L, v, 3, n, 1, O, j, O, b, j, j, j, j, r, q, q, L, X, X, L, B, H, L, B]
S19	1	PRIMARY : [j, K, c, b, v, ?, n, ?, a, i, v, v, 1, D, c, c, j, ?, j, S, j, i, i, i, i, p, p, v, v, V, v, f, G, n, f]
		SECONDARY: [j, T, 9, 9, S, c, c, X, 9, c, o, c, c, X, o, j, j, S, 5, O, c, o, c, P, S, j, T, 9, c, o, V, d, o, O]
		2 SECONDARY: [?, e, 9, 9, n, S, S, k, 9, S, S, S, k, ?, ?, n, n, ?, S, ?, S, p, n, ?, E, 9, S, ?, S, E, d, ?, ?]
S20	1	PRIMARY : [9, O, O, w, ?, ?, h, 8, D, h, 0, k, K, k, t, E, M, E, w, j, 4, 7, ?, E, 7, M, k, X, h, U, n, d, q, T]
		SECONDARY: [W, j, j, j, h, ?, ?, b, 3, g, b, y, e, q, a, s, A, e, A, h, I, E, 1, ?, f, F, e, e, L, b, o, 1, b, p, s]
		2 SECONDARY: [S, X, T, ?, T, j, c, X, ?, 9, o, j, N, T, o, j, S, 5, T, P, c, o, c, S, d, T, ?, T, c, ?, S, 5, ?, o, c]
S21	1	PRIMARY : [n, k, E, ?, E, ?, S, k, ?, 9, ?, ?, N, E, ?, ?, n, n, E, p, S, ?, S, n, d, E, ?, E, S, ?, n, n, ?, S, S]
		SECONDARY: [?, n, ?, p, E, S, S, k, p, S, S, S, S, ?, j, k, E, S, S, ?, S, n, S, N, n, d, S, ?, n, n, p, ?]
		2 SECONDARY: [O, U, 1, g, L, 1, r, ?, e, i, 1, L, v, 3, n, 1, O, j, O, b, j, j, j, j, r, q, q, L, X, X, L, B, H, L, B]
S22	1	PRIMARY : [h, j, b, b, h, V, c, T, e, j, F, h, j, c, h, V, k, c, p, h, z, s, ?, j, h, c, h, e, V, p, t, f, d, F, 4]
		SECONDARY: [T, X, S, O, c, j, c, ?, R, o, T, ?, T, o, j, T, c, c, P, c, o, c, T, d, T, X, S, c, d, c, S, 9, o, c]
		2 SECONDARY: [E, k, n, ?, S, ?, S, ?, ?, ?, E, ?, E, ?, ?, E, S, S, p, S, ?, S, E, d, E, k, n, S, d, S, n, 9, ?, S]
S23	1	PRIMARY : [?, s, 7, D, 1, ?, ?, E, f, ?, j, O, L, B, s, L, O, ?, j, O, ?, j, j, O, r, k, h, j, v, o, X, ?, f, L, ?]
		SECONDARY: [?, 7, 4, g, n, ?, ?, A, ?, ?, i, h, n, g, n, 1, j, ?, h, j, ?, j, j, h, i, e, b, h, o, c, V, ?, d, n, ?]
		2 SECONDARY: [T, S, P, d, c, c, T, X, P, c, o, T, c, X, o, X, j, c, 5, O, c, o, c, O, T, ?, S, P, c, P, c, T, 9, o, O]

Appendix 2 – Match data

Character	Sample	Algo	T	H	E	O	U	I	C	S	B	R	D	W	N	F	O	X	J	U	M	P	S	O	V	B	R	T	H	S	L	A	Z	Y	O	O	O	Score		
Match Score	01	A1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	6		
		A2	2	0	0	0	0	0	0	2	1	0	0	2	0	0	0	2	0	2	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	2	2	0	20	
	02	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
		A2	0	0	1	0	0	0	2	0	0	0	0	2	0	1	0	2	2	0	0	0	0	1	2	0	0	0	0	2	0	1	0	0	0	0	2	0	18	
	03	A1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	
		A2	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	2	2	2	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	16	
	04	A1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	5		
		A2	0	0	0	0	0	0	0	2	1	0	0	2	0	0	0	0	2	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	14	
	05	A1	0	0	0	0	0	0	0	1	1	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	2	0	0	0	0	1	0	13	
		A2	2	0	0	0	0	0	0	2	1	0	0	0	0	1	0	2	2	0	0	0	0	1	2	0	1	0	2	0	1	0	2	0	0	0	2	0	21	
	06	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		A2	2	0	0	0	0	0	0	2	0	0	0	2	0	0	0	2	2	2	0	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	16	
	07	A1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2	2	0	0	1	0	2	1	0	0	0	0	0	0	0	0	0	0	1	0	0	11
		A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	10	
	08	A1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	8	
		A2	2	0	0	0	0	0	0	2	0	0	0	2	0	0	0	0	2	0	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	0	2	2	0	17
	09	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	5	
		A2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	2	0	2	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	2	2	0	16
	10	A1	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	2	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	8	
		A2	0	0	0	0	0	0	0	2	1	0	0	0	0	2	0	0	0	2	2	0	2	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	18	
	11	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3	
		A2	0	0	0	0	0	0	2	1	0	0	2	0	1	0	2	2	2	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	2	2	0	20
12	A1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2			
	A2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	0	0	1	0	2	0	13				
13	A1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	5		
	A2	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	2	2	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	12		
14	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	8		
	A2	0	0	1	0	0	0	0	2	0	0	0	0	0	0	1	0	2	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	13		
15	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3			
	A2	0	0	0	0	0	0	2	1	0	0	0	2	0	0	0	0	2	0	2	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	2	0	16		
16	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	4			
	A2	0	0	1	0	0	0	0	2	1	0	0	2	0	0	0	2	0	0	0	0	2	1	2	0	0	0	0	2	0	1	0	0	0	0	2	0	20		
17	A1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	4			
	A2	2	0	1	0	0	0	0	2	1	0	0	2	0	0	0	2	2	0	0	0	0	1	2	0	1	0	0	0	1	0	0	0	0	2	2	0	21		
18	A1	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	5			
	A2	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	2	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	12			
19	A1	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	2	0	0	2	0	0	0	2	0	0	0	0	0	0	0	2	0	11		
	A2	2	0	0	0	0	0	0	2	0	0	0	2	0	0	0	0	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	2	13			
20	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4			
	A2	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	2	2	2	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	0	16			
21	A1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	2	1	1	0	0	0	0	1	0	10			
	A2	2	0	0	0	0	0	0	2	0	0	0	2	2	0	0	0	2	0	0	0	0	2	1	2	0	1	0	2	0	0	0	0	0	0	2	20			
22	A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	2	0	2	0	0	0	0	0	1	0	7			
	A2	2	0	0	0	0	0	0	1	0	0	2	0	0	0	0	2	2	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	2	0	16		

Appendix 3 - Scores

Algo	Scores																																	Ave
A1	6	2	3	5	13	0	11	5	5	8	3	2	5	8	3	4	4	5	11	4	10	7											5.636364	
A2	20	18	16	14	21	16	10	17	16	18	20	13	12	13	16	20	21	12	13	16	20	16											16.27273	

Appendix 4 – T-test

t-Test: Two-Sample Assuming Equal Variances		
	A1	A2
Mean	5.636363636	16.27272727
Variance	11.0995671	10.49350649
Observations	22	22
Pooled Variance	10.7965368	
Hypothesized Mean Difference	0	
df	42	
t Stat	-10.73611821	
P(T<=t) one-tail	6.45128E-14	
t Critical one-tail	1.681952357	
P(T<=t) two-tail	1.29026E-13	
t Critical two-tail	2.018081703	

Appendix 5 – Combined Algorithm Code

```
package FurtherWork;

import Application.Algorithm1;
import Application.Algorithm2;
import java.util.ArrayList;
import java.util.List;

public class Algorithm3 {
    //THIS IS THE COMBINED FEATURE EXTRACTION TECHNIQUE
    // BASED ON BOTH KHAUSTOV AND SHARMA'S WORK
    //IT CAN BE SUMMARISED AS:
    // -DETECT ISLANDS
    // -COUNT HORIZONTAL TRANSITIONS
    // -COUNT VERTICAL TRANSITIONS
    // -FIND VERTICES
    // -FIND ENDS
    // -RECORD WHOLE CHARACTER INFO
    // -DIVIDE INTO ZONES
    // -COUNT ZONE PIXELS
    // -CALCULATE ZONE PIXEL DISTANCES
    // -RECORD ZONE INFO
    // INFO:
    // enclosed spaces, vertical transitions,
    // horizontal transitions, vertices, ends,
    // z1 pixels, z1 pixel distance,
    // z2 pixels, z2 pixel distance,
    // z3 pixels, z3 pixel distance,
    // z4 pixels, z4 pixel distance,
    // character
    //-----

    //RUN COMBINED ALGORITHM 3:
    public static List<A3Object> RunAlgorithm3(List<List<int[]>> characters) {
        //VARIABLES:
        List<A3Object> a3objectList = new ArrayList<>();
    }
}
```

```

A3Object a3Object = new A3Object();
List<List<List<Integer>>> zoneList;

//FLIP BACK CHARACTERS:
Algorithm1.FlipBackUpright(characters);

//LOOP THROUGH CHARACTERS:
for (int currentChar = 0; currentChar < characters.size(); currentChar++) {
    a3Object = new A3Object();
    a3Object.setEnds(Algorithm2.FindEnds(characters.get(currentChar)));

    a3Object.setVertices(Algorithm2.FindVertices(characters.get(currentChar)));
    a3Object.setIslands(Algorithm2.FindIslands(characters.get(currentChar)));
    a3Object.setHorTrans(CountHorTransitions(characters.get(currentChar)));
    a3Object.setVertTrans(CountVertTransitions(characters.get(currentChar)));

    //GET ZONES
    zoneList = new ArrayList<>();
    zoneList = DivideIntoZones(characters.get(currentChar));
    //LOOP THROUGH ZONES
    for (int currentZone = 0; currentZone < zoneList.size(); currentZone++) {
        switch (currentZone) {
            case 0:
                a3Object.setZ1Pix(
                    Algorithm1.CountZonePixels(
                        zoneList.get(currentZone)));

                a3Object.setZ1PixDist(
                    Algorithm1.FindAveragePixelDistance(
                        zoneList.get(currentZone)));

                break;
            case 1:
                a3Object.setZ2Pix(
                    Algorithm1.CountZonePixels(
                        zoneList.get(currentZone)));

                a3Object.setZ2PixDist(
                    Algorithm1.FindAveragePixelDistance(
                        zoneList.get(currentZone)));

                break;
            case 2:
                a3Object.setZ3Pix(
                    Algorithm1.CountZonePixels(
                        zoneList.get(currentZone)));

                a3Object.setZ3PixDist(
                    Algorithm1.FindAveragePixelDistance(
                        zoneList.get(currentZone)));

                break;
            case 3:
                a3Object.setZ4Pix(
                    Algorithm1.CountZonePixels(
                        zoneList.get(currentZone)));

                a3Object.setZ4PixDist(
                    Algorithm1.FindAveragePixelDistance(
                        zoneList.get(currentZone)));

                break;
        }
        a3ObjectList.add(a3Object);
    }
}
return a3ObjectList;

```

```

}

//A1-DIVIDE INTO ZONES:
public static List<List<List<Integer>>> DivideIntoZones(List<int[]> singleCharacter) {
    //VARIABLES:
    List<List<List<Integer>>> zoneList = new ArrayList<>();
    List<List<Integer>> zone;
    int rowDenominator = 2;
    int colDenominator = 2;
    int rows = singleCharacter.size();
    int columns = singleCharacter.get(0).length;

    //FIND DENOMINATORS:
    if (rows % 2 != 0) {
        int[] emptyRow = new int[columns];
        singleCharacter.add(emptyRow);
        rows += 1;
    }
    if (columns % 2 != 0) {
        //LOOP ROWS
        for (int i = 0; i < rows; i++) {
            int[] evenRow = new int[singleCharacter.get(i).length + 1];
            for (int j = 0; j < evenRow.length - 1; j++) {
                evenRow[j] = singleCharacter.get(i)[j];
                evenRow[evenRow.length - 1] = 0;
            }
            singleCharacter.set(i, evenRow);
        }
    }

    //POPULATE ZONES:
    for (int k = 1; k < rowDenominator + 1; k++) {
        for (int l = 1; l < colDenominator + 1; l++) {
            int rowStart = (singleCharacter.size() / rowDenominator) * (k - 1);
            int rowEnd = (singleCharacter.size() / rowDenominator) * (k) - 1;
            int colStart = (singleCharacter.get(0).length / colDenominator) * (l - 1);
            int colEnd = (singleCharacter.get(0).length / colDenominator) * (l) - 1;
            zone = new ArrayList<>();

            for (int m = rowStart; m < rowEnd + 1; m++) {
                List<Integer> zoneRow = new ArrayList<>();
                for (int n = colStart; n < colEnd + 1; n++) {
                    zoneRow.add(singleCharacter.get(m)[n]);
                }
                zone.add(zoneRow);
            }
            zoneList.add(zone);
        }
    }
    return zoneList;
}

//A1/A2-COUNT HORIZONTAL WHITE-BLACK TRANSITIONS:
public static int CountHorTransitions(List<int[]> singleCharacter) {
    //VARIABLES:
    int whiteBlackTransitions = 0;

    //GET TRANSITIONS FOR EACH COLUMN:
    for (int i = 0; i < singleCharacter.size(); i++) {
        for (int j = 0; j < singleCharacter.get(i).length - 1; j++) {
            if (singleCharacter.get(i)[j] == 1 && singleCharacter.get(i)[j + 1] == 0) {
                whiteBlackTransitions++;
            }
        }
    }
}

```

```

    }
}
whiteBlackTransitions =
    (int) Math.round((double) whiteBlackTransitions / singleCharacter.size());
return whiteBlackTransitions;
}

//A1/A2-COUNT VERTICAL WHITE-BLACK TRANSITIONS:
public static int CountVertTransitions(List<int[]> singleCharacter) {
    //VARIABLES:
    int whiteBlackTransitions = 0;

    //GET TRANSITIONS FOR EACH ROW:
    for (int i = 0; i < singleCharacter.size() - 1; i++) {
        for (int j = 0; j < singleCharacter.get(i).length /*- 1*/; j++) {
            if (singleCharacter.get(i)[j] == 1 && singleCharacter.get(i + 1)[j] == 0) {
                whiteBlackTransitions++;
            }
        }
    }
    whiteBlackTransitions =
        (int) Math.round((double) whiteBlackTransitions / singleCharacter.get(0).length);
    return whiteBlackTransitions;
}

//A1/A2-CONVERT FROM LIST OF LISTS TO LIST OF ARRAYS:
public static List<int[]> ConvertToArray(List<List<Integer>> zone) {
    //VARIABLES:
    List<int[]> zoneAsArrays = new ArrayList<>();

    for (int i = 0; i < zone.size(); i++) {
        int[] rowArray = new int[zone.get(i).size()];
        for (int j = 0; j < zone.get(i).size(); j++) {
            rowArray[j] = zone.get(i).get(j);
        }
        zoneAsArrays.add(rowArray);
    }
    return zoneAsArrays;
}
}

```