College of Computer Technology

Optical Character Recognition

Image processing, classification and learning using Neural Networks

Author: Kyle Truebody
Student Number: 2013620

ABSTRACT

This project presents study into the different components that make up an Optical Character Recognition system. The OCR system in this project is uses Neural Networks for pattern recognition. The Project build is implemented using Java, OpenCV and Encog Machine Learning API contained in a fully interactive graphical user interface.

ACKNOWLEDGEMENTS

After an intense 7 months, writing this note of thanks is the finishing touch to this project. it has been a period of intense learning for me and I have enjoyed every minute of it. The challenge has been a big driving force for me. But more importantly I would never have been able to accomplish it without the help of a few special people.

Firstly, I would like to thank Rory O'Connor for continuously pushing me forward and exploring alternative ways of understanding problems and solutions. As I reflect on the time spent explaining and examining problems and solutions with you, it greatly improved my ability to express the project goals. To Thiago and Caomahan Murphy, for letting me crash on your couch to save me from traveling on a bus 3 hours a day. I never would have had the time to do a project like this if was not for you guys. To all the lecturers and staff at College of Computer Technology. It has been an amazing journey; one I shall never forget. Keep up the fantastic work. Last and not least, Robyn and my family for your continuous support over the last year, for your wise counsel and sympathetic ears, you have all been invaluable.

Thank you very much everyone,

Kyle Truebody

Carlow, May 01, 2017.

Table of Contents

INTRODUCTION 1
METHODS & COMPONENTS OF OCR 1
PRE-PROCESSING 3
Pre-processing Techniques 4
Feature Extraction 10
Template Matching and Comparing Methods 12
Feature Detection Techniques 12
CLASSIFICATION 13
Distance Classifiers 13
Artificial Neural Networks 14
NEURAL NETWORKS 14
When to use Neural Networks 15
Neural Network Models 16
Neural Network Structure 16
Normalising Data 18
How Neural Networks Learn 18
POST PROCESSING 19
Grouping 19
Error Detection and Correction 19
LIMITATIONS OF OCR 19
SYSTEM ANALYSIS & DESIGN 20
DEPENDENCIES AND TECHNOLOGIES 20
Java 20
OpenCV 21
Encog API 22
SYSTEM ANALYSIS 22
Model-View-Controller 26
Wire frame Design 27
Class Diagrams 31
IMPLEMENTATION 35
TESTING AND EVALUATION 37
CONCLUSION 43
APPENDICES 45
APPENDIX A: PROJECT PLANNING 45
APPENDIX B: REFLECTIVE JOURNAL 46

APPENDIX C: CODE LISTINGS 50
REFERENCES 56

INTRODUCTION

The replication of human intelligence in machines has been long sort after dream. As our understanding of the human biological neural network expands, we can mimic our understanding of intelligence in machines better. We as human use our brains and sensory powers to act, react and proact rationally based on intelligence. The Oxford Dictionary defines the word intelligence as - '*The ability to acquire and apply knowledge and skills'.* This definition can be explained as a continuous enrichment process. As knowledge and skills are applied a result is generated which leads to the acquisitions of more information. The additional information is then used to adapt the existing knowledge. Being able to replicate intelligence in machines is difficult and the learning process is comparatively slower than humans. But AI, Artificial Intelligence, excels at repetitive, uniform and scalable tasks. Optical Character Recognition has been one of the most successful applications of Artificial intelligence and pattern recognition.

Optical Character Recognition belongs to a family of techniques used to perform automatic recognition. OCR, Optical Character Recognition, is a field of research in Pattern Recognition, Artificial Intelligence and Computer Vision. OCR provides a mechanical and electronic conversion of images of typed or hand written text into machine-encoded text. OCR tries to mimic the biological functions of the human body at a very basic level. As a person reads words on a page or screen. The combination of the eyes and brain are used to carry out OCR. The eyes receive optical information, light, colours and luminance that compose patterns and shapes. This information is converted into electrical signals and is passed to the brain. The brain deciphers and distinguishes patterns and shapes and classifies them using previously acquired knowledge through learning.

OCR technology has been used to develop many kinds of domain-specific OCR applications. OCR has revolutionized the document management process, enabling scanned documents to become more than just an image, by converting them into machine searchable documents. Businesses today are using OCR for document management, invoice processing, business card recognition, preserving meeting notes, importing form submissions and letter sorting for postal systems. OCR provides a means of handling large mundane data entry tasks that take up substantial amounts of time and reduces cost in processing them.

This project explores the use of machine vision methods and machine learning technologies and methodologies used for OCR Content represented in this project has been acquired from related OCR, Computer Vision and Machine Learning research papers. This knowledge has been applied to implement OCR software package with integrated Graphical User Interface.

METHODS & COMPONENTS OF OCR

The main principle behind OCR is teaching machines to understand and recognises patterns. Just like people machines need to be taught to be able to do a task. When a child learns to read, the method of Reinforcement Learning is applied. Reinforcement learning aims to strengthen behaviour whenever that specific behaviour is preceded with positive stimulus. As a child learns to recognise the letter '*a*', the teacher will reward them for

correctly recognising the character. Or if incorrectly recognised, will correct them and make them do the processes again. Reinforcement learning is the same concept applied to machine learning methods. The machine is rewarded when it correctly learns a pattern and is positively punished and corrected if incorrect.

Figure 1. Machines learn just like us. Well… almost!

Teaching the OCR software is done by providing it with examples of different characters to learn. The software then builds a description of each of the characters, this description is known as a prototype. Learning the description is done by iterating over each of the prototypes and extracting key features of each one. After each learning cycle, the results of each recognition attempt are compared to the known prototype description. The difference of the comparison is measured and determines if the software has learnt the pattern sufficiently. The learning cycle stops when the error of the comparison has reach predetermined threshold. After the software has finished learning, characters can be fed into the trained software. The input character is compared to all the learned prototypes and the prototype that best fits the description is communicated as the result.

A typical OCR system is comprised of several components. Each component describes a milestone in the operation pipeline. Figure 2 illustrates the flow of operation and their descriptions follow.

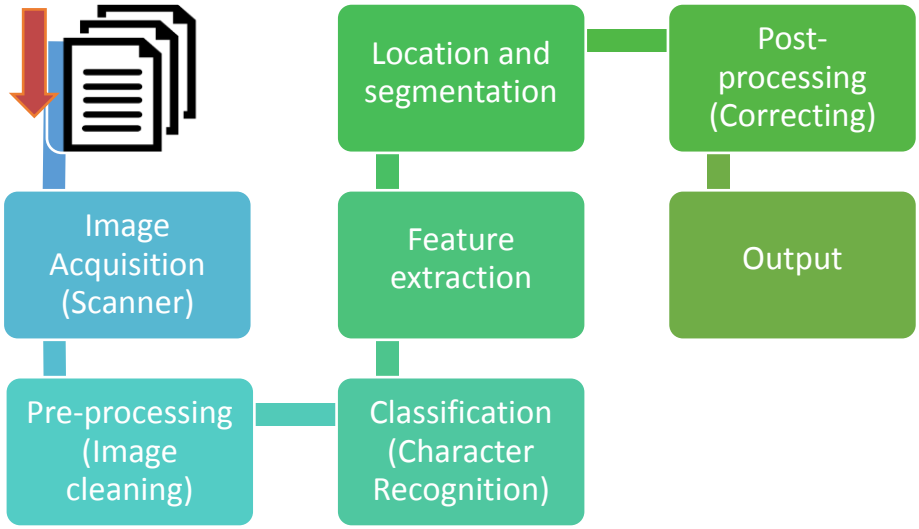


Figure 2. Typical OCR system operations

For machines to read like humans do we need to give them the ability to see. This can be done with digital imaging equipment like cameras or scanners. This is known as the image acquisition stage. The second step is to help the machine to understand the image. This can be done by cleaning the image of unwanted attributes. This stage is known as the pre-processing stage. The pre-processing stage removes or adds image details that will enhance features of the image that we want to recognise. The third step is to give the machine a means of locating lines of text, single words and single characters. This stage is known as the document analysis or location and segmentation stage. The fourth step is to take the single characters that have been segmented from the image and isolate its key features. This is the feature extraction stage. The features extracted are converted into a data structure that the machine can understand. The fifth step is to pass these key features to and compare them to the previously learnt character. This is called the classification stage. The last step is post processing. Post processing is the reconstruction of the resulting character recognition and applying any corrections to results. These methods described are explained in more detail below.

PRE-PROCESSING

Scanners use sensors to detect light levels and convert them into either grey scale or colour images. Colour and grey scale images are known as multi-level images, because they are made up of multiple colour channels like red, blue, and green, and multiple contrast values representing each colour. Multi-level images can be scanned in as bi-level images. A bi-level image is made up of only two colours with set contrast values, generally back and white. Bi-level images are used to ensure that a discrete boundary of patterns and the background are clearly distinguishable. It is better to scan images as multi-level images. This is because multi-level images can be converted into grey scale images and still retain information about pixel illumination, which retains more detail in the image. This is important as this information is required for further pre-processing steps, like edge detection and line detection, which require a high level of detail to be effective.

Figure 3. Respectively shows multi-level colour and grey, and bi-level image

Pre-processing for OCR is the improvement of image data that suppresses unwanted distortions that may have manifested during the image acquisition stage. The main objective of pre-processing is to is to decrease the variations that increase complexities of the image and make it more difficult to recognise characters. This is done by identifying and enhancing key features by normalising the image and removing global variations that could impede the pattern description. A set of image processing techniques is used to prepare the image prior to computation. Pre-processing system is structured in a pipeline

fashion, meaning that each stage in the process is reliant on the success of the previous stage, to provide an optimal outcome. These steps are in no means a set recipe that must be followed. Each problem requires its own solution. Automating the pre-processing stage is extremely difficult because of this.

The need for pre-processing is based on the following factors that can affect the accuracy of character recognition.

- Older or discoloured documents
- Watermarks and smudges
- Paper quality
- Scanner quality
- Scanner resolution
- Language complexities
- Fonts types
- Type of document

The importance of pre-processing stage for character recognition is to remedy some of these factors that may have occurred. Effective pre-processing algorithms make the OCR system more robust through image enhancement, noise removal, image thresholding, skew detection and correction, page segmentation, character segmentation and character normalisation.

Pre-processing Techniques

Pre-processing techniques in OCR are used on colour, grey scale or binary images. OCR systems mostly use grey scale and binary images. This is due to the computational cost of processing coloured images.

The first step in the pre-processing process is image enhancement. Image enhancement techniques are used to reduce the noise attenuation and correct the image contrast. The second step is thresholding (binary image conversion, bi-level image) to separate the background and the text. The third step is page segmentation to separate text areas and non-text areas. The fourth step is character segmentation, which is to isolate single characters from each other. The fifth step is morphological processing, which enhances the characters in cases where previous steps may have eroded or dilated parts of the character. The sixth and last step is to normalise the isolated characters in terms of size and focus so they conform to the predefined image samples. Some of the above techniques may be used at various stages of the process.

Image enhancement techniques are used to improve an images quality. The enhancements are used to optimize specific features that occur in the image and correct any artefact that may have occurred. These enhancements can be to correct noise, lighting, illumination and blurring.

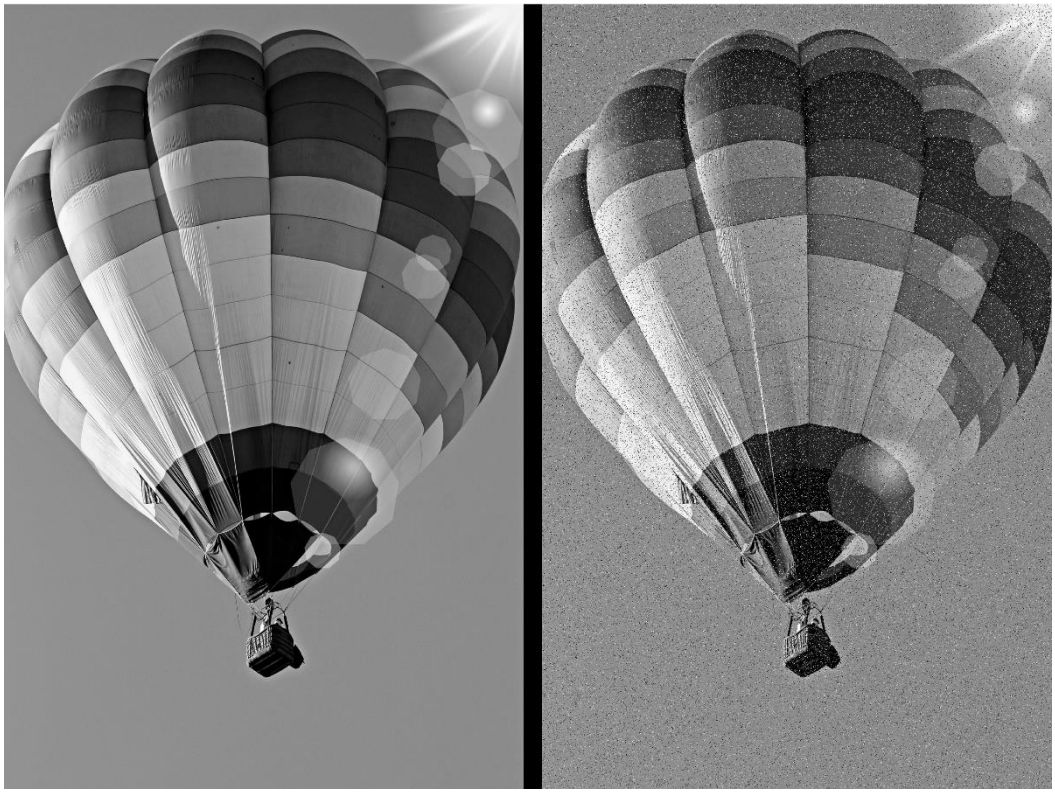


Figure 4. Left original image. Right, image effected by Noise (Salt and pepper noise).

Spatial Image Filters

Filtering is a technique used to modify or enhance images. Filters can be used to emphasize certain features or remove features. These features are determined by the pixel frequencies that occur in the image. Frequency meaning, the intensity of the pixel ranging from 0-255 in the case of a grey scale image. Therefore, filters are used to supress either high (image smoothing) or low frequency (edge detection) pixels. Spatial processing is classified into two groups, point processing and mask processing.

Point Processing

Point processing modifies the value of a pixel at a certain position in the image and applies a function to the pixel value and outputs the result of function to the output image at the pixel's original location as show in figure 1. Point processing is also known as a mapping function Some point processing techniques are contrast sketching, global thresholding, histogram equalisation.

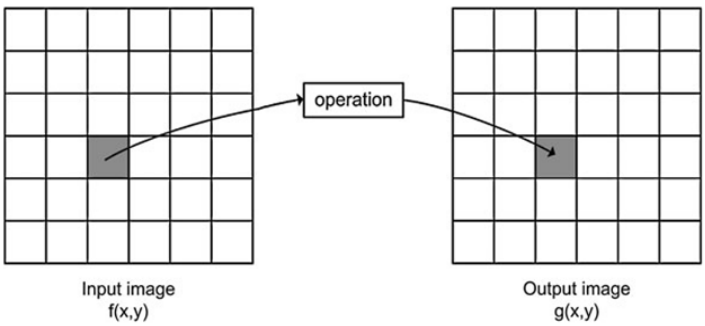


Figure 2: Depiction of point processing.

Contrast Sketching

The level of contrast in an image may vary because of poor illumination or low quality sensor on the acquisition device. The goal is to modify the dynamic range of the grey levels in the image. The range is defined to be the entire range of intensity values within the image, for example values will range between 0 to 255. Therefore, the maximum pixel value subtracted by the minimum pixel value will define the range. The desired level of contrast is calculated using in a linear mapping function to stretch the pixel values of low or high contrast image by extending the dynamic range across the image spectrum. Drawbacks of modifying the histogram of an image in this way can produce "graininess". Explained simply the darker pixels become darker and the lighter pixels become lighter.

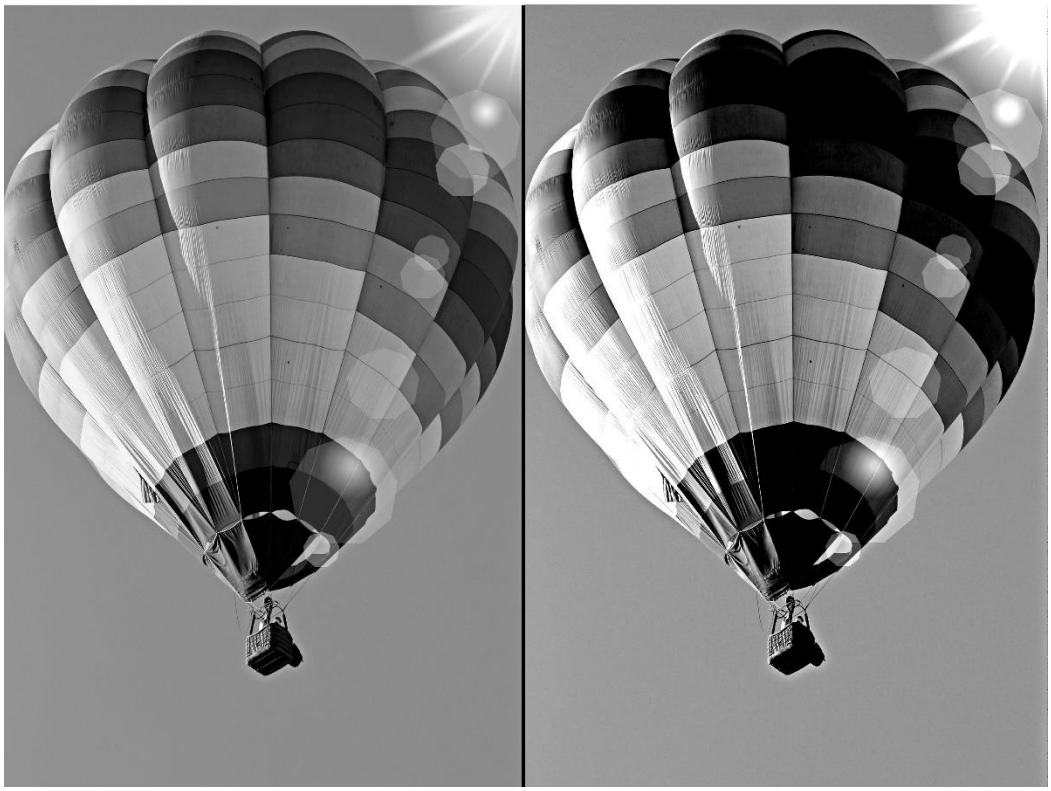


Figure 5. Left is the original image and on the right contrast sketching has been applied. Notice in the background of the image has become a little grainy.

Image Thresholding

The goal of thresholding is to covert an input image into a bi-level representation of the input image. The colours black and white are used because to represent the image as they are high contrast and can easily determine what is background information and what is foreground (text) information. This also adds uniformity to all image inputs when recognition stage of the OCR system is reached. Global threshold methods use one threshold value over an entire image. The threshold value can be determined either by the mean value of all pixel. The threshold is applied to the image using a linear mapping function, and if the corresponding pixel is less than the threshold value the pixel is assigned the new value of 0 (black). If the pixel value is above the threshold the corresponding pixel is assigned the value of 255 (white). Figure 3 illustrates an example of image thresholding result.

Histogram Equalization

A histogram simply plots the grey scale pixel intensity frequencies that occur from 0 to 255 (black to white). This method is used to increase the global contrast of an image, usually when pixel values are close in contrast. The intensities can be better distributed across a histogram allowing for areas of low local contrast to gain a higher contrast. This is done by spreading out the most frequent intensity values which is depicted in figure 6.

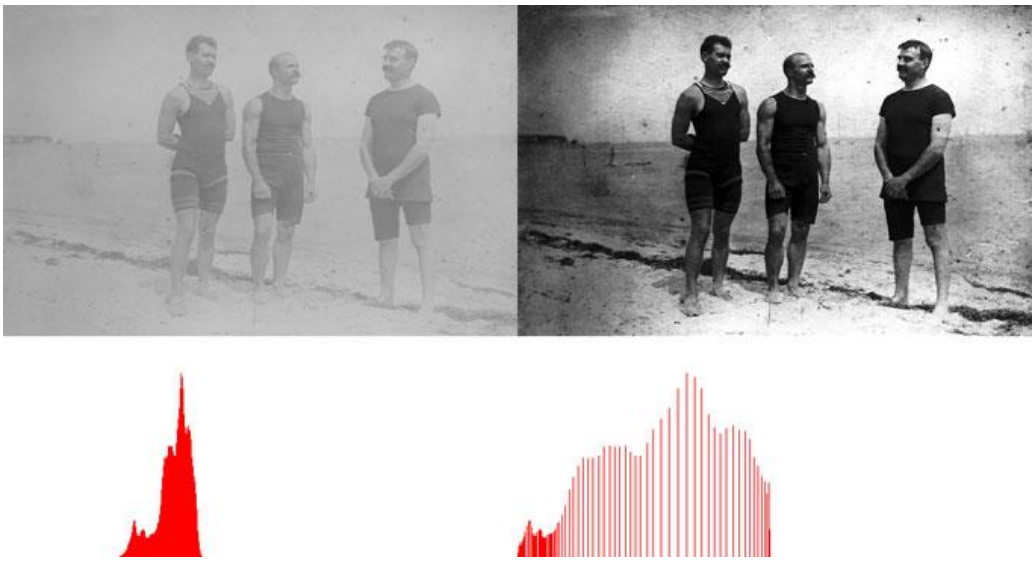


Figure 6: Histogram equalization depicting contrast change.

Mask Processing

Mask processing is a small matrix used to apply affects to an image. This small matrix is denoted as an odd number for example a 3X3,5X5,7X7 etc. Also, known as an image kernel, uses a technique called convolution or correlation matrix filtering to determine the most important portions in an image. Convolution is the treatment of a matrix by another matrix. So, in the case of image processing the first matrix is the pixel matrix of the full image and the kernel matrix is a smaller matrix that treats each pixel and it's neighbouring pixels. The origin pixel (usually the centre of the kernel) is the target pixel to be adjusted. To perform convolution the kernel should be flipped horizontally and vertically, then it is slid over the image and then multiply the corresponding elements and add the them

Figure 7. Example of how the kernel moves over an image, using the convolution treatment

together. This process is repeated until all values of the image have been processed. Correlation uses the same mechanics except the kernel is not rotated 180°. A kernel example shown in figure 4 represents a right Sobel kernel commonly used in edge detection and the sharpen kernel used for emphasizing finer details.

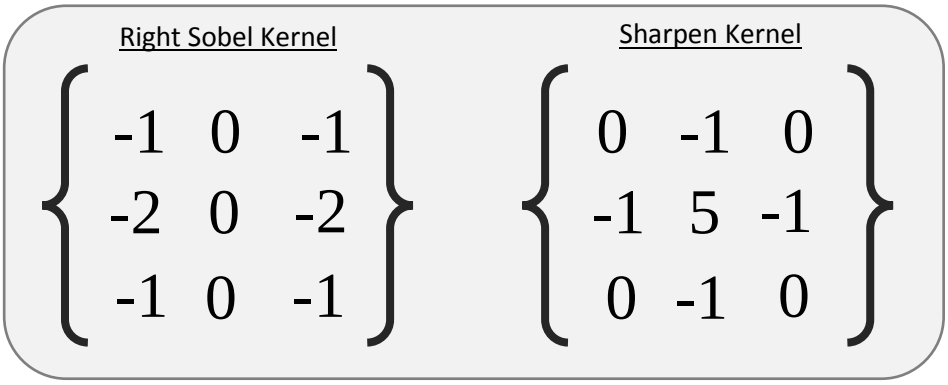


Figure 8: Right Sobel and Sharpen kernel examples

Low Pass Filters (Smoothing)

Low pass filters have a smoothing effect on images. This effect is accomplished by reducing the intensity variations between one pixel to the next. Low pass filtering is often used as an effective method for reducing noise in an image. The goal of the low pass filter is to replace each pixel value in the image with the mean value of its self and its neighbouring pixels. Mean filtering is generally used as a convolution filter, therefore having a kernel. In the case of mean filtering the larger the kernel size the more intense the smoothing effect will be.

Figure 9. Grainy (Noise) image on right. On the left is the result of low pass filter over the image.

High Pass Filtering (Sharpening)

High pass filtering accentuates the finer details in an image. This method has the opposite effect of the low pass filter by detecting the high contrast between pixels in a local area. The difference between low pass and high pass filtering is the contents of their kernels. High pass filters have negative and positive values weights. The result of the convolution over the image is pixels in an area with slowly varying values over an area will become close to zero. If the grey level in the image area varies rapidly then the result will be a

high value. The resulting image produced from the high pass filter is then applied back onto the original image to restore the grey level tonality. Figure 10 shows an example of 3X3 kernel of means filter (low pass filter) and high pass filter. Figure 11 shows the effects of high pass filter using the sharpening kernel in figure 10.

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

0	-1	0
-1	5	-1
0	-1	0

Figure 10: Means filter kernel and sharpening filter kernel respectively.

Figure 11: Before and after result of high pass filter

Median Filter

The median filter is a kind of low pass filter. The median filter does not use coevolution or correlation techniques. The pixel values that are covered by the kernel are sorted and the centre pixel (kernel origin) is replaced with the median of the pixel set. Median filters are effective in removing salt and pepper noise, which is a random occurrence of black and white pixels over the spectrum of the image.

Local Thresholding

Local thresholding techniques are used for with documents that have complex backgrounds or uneven background illumination. Instead of having a single threshold value, a local value is determined for a specific segment of the image allowing for the threshold to vary smoothly across the image.

Noise Removal

Scanned documents often contain noise that is a consequence of the quality of the imaging sensor. The sensors circuitry or other electrical interferences can produce noise. This makes is necessary to filter the noise to improve character recognition. Image noise is described as randomly varying brightness or colour information in images that gives an image a grainy texture. Depending of the type of noise, the image is effected to a different extent.

Skew Detection and Correction

Skew detection and correction methods are used to fix the rotation of the image to improve the chances of a more accurate document body alignment which will lead to a cleaner character segmentation. By calculating the deviation of the text lines the orientation of the document can be determined. Skew detection techniques can be classified into the following groups: Hough transform and horizontal projection analysis.

Hough transform is used to detect lines with in the image. The lines extracted from the Hough transform are then used to calculate the document deviation by using the corresponding text lines.

Horizontal projection analysis is the accumulation of pixels found across the same row in an image document. This method is only effective if the image is having zero skew angle. The pixel count is plotted onto a histogram. The peaks within the histogram will correspond to the text lines and the valleys will correspond to the non-text lines.

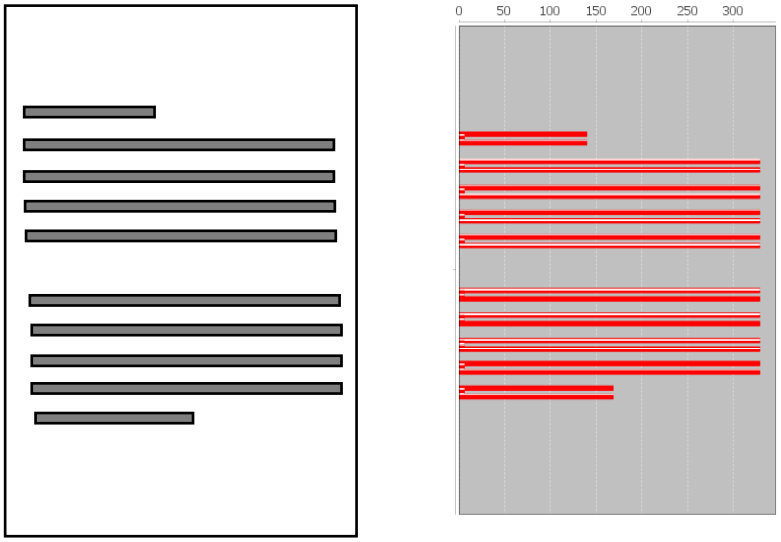


Figure 12. Example of horizontal projection. Left is the document and the right is the histogram of the pixel count along the y-axis.

Feature Extraction

The objective of feature extraction is to identify the essential characteristics of the symbols and characters in question. There are two main methods of feature extraction matrix mapping and feature identification.

Matrix mapping is a way of describing characters or symbols by its actual rastered image. A raster image is an image composed of individual pixels of varying colours. Feature Identification is a method extracting certain features of a character that characterize it while leaving out the unimportant attributes. Feature identification techniques can be categorized into three main groups. Structural analysis, distribution of points and transformation and series expansions. Different techniques can be evaluated on their sensitivity to noise and deformation and ease of implementation. This has been quantified by L. Eikvil, "Optical Character Recognition", p. 15, 1993. The Criteria for evaluation is as follows:

- **Robustness:**

    1. **Noise:** The sensitivity to disconnected line segments, bumps, gaps, salt and pepper etc.
    2. **Distortion:** The sensitivity to local variations like rounded corners, square corners, improper protrusions, dilation and shrinkages.
    3. **Style Variations:** The sensitivity to variations in font styles and different shapes to represent the character.
    4. **Translation:** Sensitivity to movement of the whole character or any of its components.
    5. **Rotation:** Sensitivity to change in orientation of characters.

- **Practical Use:**

    1. Speed of recognition.
    2. Complexity of implementation.
    3. Independence (the need for supporting techniques).

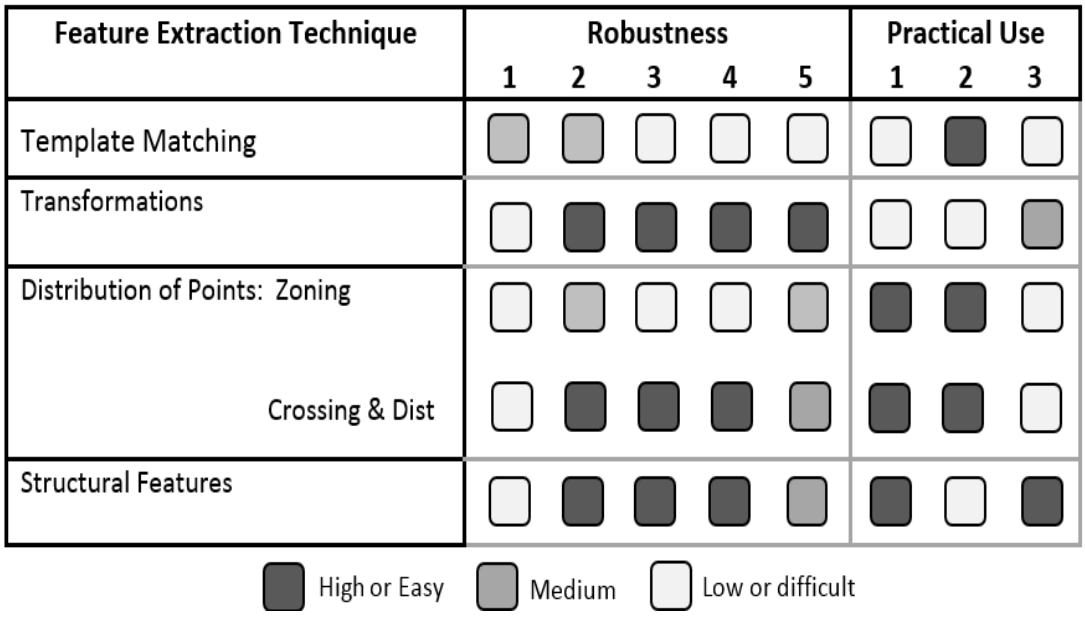


Figure 13: An evaluation of feature extraction techniques.

Template Matching and Comparing Methods

Template matching differs from feature extraction whereas no features are extracted. Instead the digitized image of the input character is directly matched and compared to a set of pre-existing characters. The difference between the prototype character and the input character is computed and the class that best matches the input character is assigned as the output. The method is easy to implement but is sensitive to noise and style variations. This technique has no way of handling rotated characters and if characters are touching it becomes difficult to determine a match. Template matching provide good recognition on monotype and uniform fonts and single column pages.

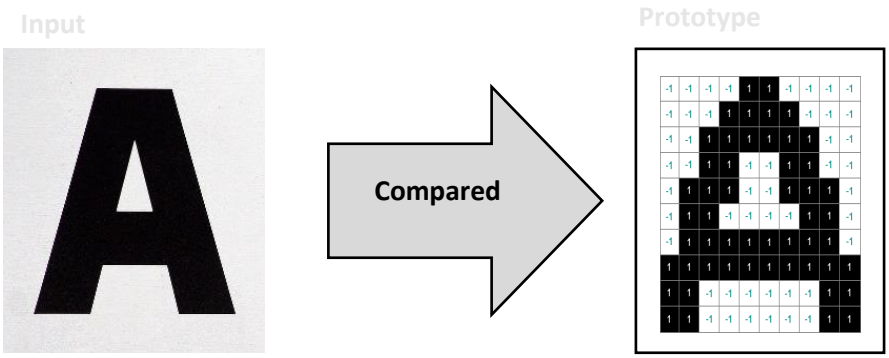


Figure 14: Depiction of template matching using bi-level images.

Feature Detection Techniques

In feature detection methods, meaningful measurements are calculated and extracted from the characters. These feature attributes are then compared with descriptions of previously obtained character classes acquired from training the system. The closest feature vector match determines the classification.

Zoning

The Zoning technique the isolated character image is divided in to smaller sections or zones. These sections can be overlapping or not over lapping regions of each other. The density of black and white pixels is calculated in each zone and is used as the descriptive features of the character.

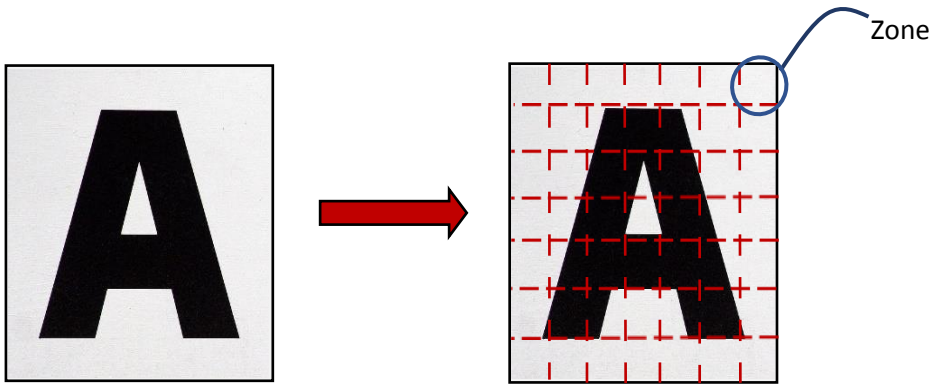


Figure 15: Depiction of character Zoning.

Crossing and Distances

Researchers have established a series of features that can be used to describe characters by the amounts of times the image is crossed by a vector in a certain direction. The distances or angles the character is crossed at is measured when a combination of attributes is triggered as true the best matched class is classified.

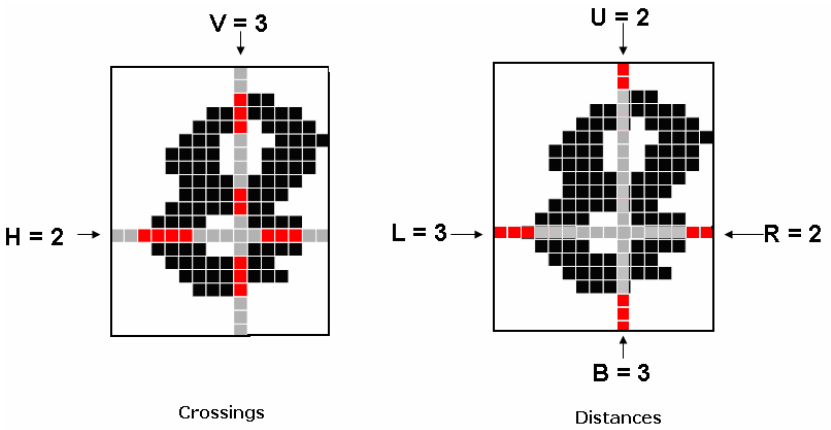


Figure 16: Crossing and Distance feature extraction.

Structural and Topological Features

The features considered in structural techniques are related to the geometry of the characters. Features are represented as convexities, concavities, line intersections, start and end points. These features try to describe the physical makeup of the characters. Structural techniques generally provide the most robust and reliable feature extraction and lends to a more accurate classification stage, yet implementation can be complex.

Figure 17: The basic concept behind structural and topological feature extraction.

CLASSIFICATION

Classification is the process by which the character feature space assignment is decided. The feature space is the agreed data structure and the features that will describe a prototype. The classification is usually done by comparing of feature vectors corresponding to the input characters with the representative of each character class. The principle approaches to the classification or decision making are distance classifiers, statistical classifiers and neural networks.

Distance Classifiers

Is the classification technique commonly used in conjunction with Template matching, also known as Matrix matching. The distance between input character vector and the obtained class prototype vector are compared using a correlation approach. Minimum

distance classifiers work well when the class descriptions are well separated. This can be defined by the distance between the means of each vector class. The larger the distance between classes will provide more accurate classification stage.

Artificial Neural Networks

Neural Networks are composed of interconnected elements called neurons. Neural networks can produce its own set of classification data by methods of training. Training sets are fed into the neural network and the resulting vector is compared with a desired output vector (in the case of supervised learning). The weights of the connected elements are adjusted using the error calculated from the output affecting the output of the next iteration. This is done until an acceptable level of error in the output is reach. Neural networks add a level of complexity to systems as it becomes difficult to predict the internal behaviours of the network. The advantages of neural networks are that they are flexible and adaptable and if implemented correctly can create large valuable data sets in very little time.

NEURAL NETWORKS

Neural networks models are inspired by the basic understanding of a biological neural network. The key element of the biological neural network is structured on is the idea of an enormous number of highly interconnected processing elements called neurons that work in unison to solve a specific problem. Artificial neural networks, just like the human brain learn by example. Learning in a biological system involves the adjustment of synaptic connections that connect the neurons.

In the human brain a typical neuron receives signals from other neurons via a host of fine structures call dendrites. The neuron sends out an electrical signal through the long stand called the Axon which splits into thousands of branches. At the end of each branch is a synapse. The synapse converts the electrical activity into an electrical effect that excites and stimulates the connected neuron. When a neuron receives input that is sufficiently larger than its internal input restraint, it sends an electrical signal down the axon. Leaning occurs when the effectiveness of a synapses is changed so that it influences the connected neuron differently.

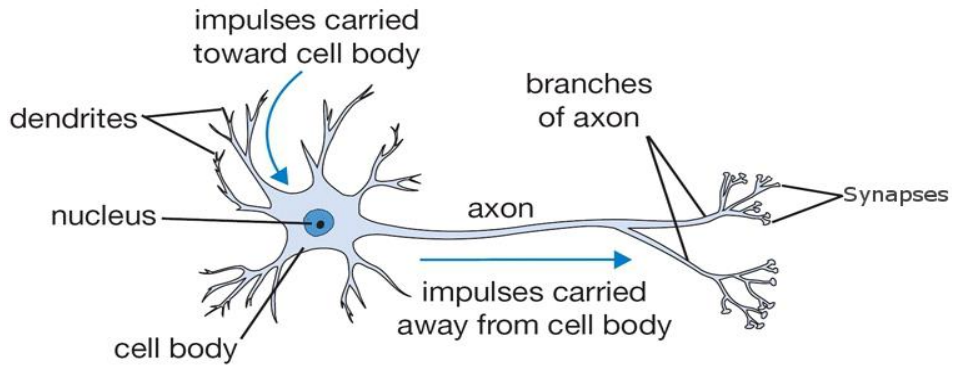


Figure 18: Diagram of human neuron

The essential features of the human neuron are extracted and simulated in as a program. Figure 17 depicts the structure of a simple neuron from an artificial neural network and its relevance to a biological neuron is explained.

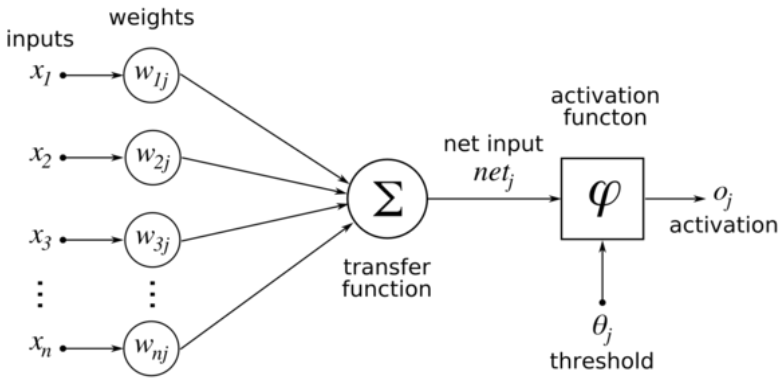


Figure 19: Diagram of ANN neuron structure.

The inputs nodes in *figure 17* act as the dendrites do in the biological neuron. The weights in act as the electrical signal strength. The summation of the weights is calculated and passed through an activation function which will normalizes the signal values. The Threshold value is the value that acts as the inhibitor value. After the activation function the we can imagine the activation acts as the Axon and communicates the activation value to the connected neurons.

Artificial Neural networks are general purpose tools unlike the human brain. Artificial neural networks are largely mathematical constructs. Neural networks accomplish pattern recognition very well and can learn large complex problems. At a high-level a neural network receives a pattern and then communicates a pattern as output. The neural network replies with a best matching pattern back if recognised.

When to use Neural Networks

Neural networks are described as universal approximators. This means that they are best used in a system that has a high tolerance to error. Neural networks have ability to derive meaning from complicated or fuzzy data and extract patterns and trends within the data. Neural networks are useful for solving problems that cannot be expressed as a series of steps. Josef Burger [2] lists the areas that neural networks work well.

- Capturing associations or discovering regularities within a set of patterns.
- Where the volume or the number of variable or diversity of the data is very great.
- The relationship between variables are vaguely understood.
- The relationships are difficult to describe with conventional approaches.

Neural networks can be applied to solve many different problems. Neural networks are great at predicting Stock Market trends, Loan applications decision approvals, Security using facial recognition and Medical diagnoses be analyzing MRI's and X-Rays images.

Neural Network Models

The methods that machine learning techniques use to evaluate and learn data are directly implicit to the data being analysed. The below section discusses data classification, regression analysis and clustering models. Each model is related to a learning method. Supervised learning is done by providing the desired labelled data of the input during the training. Unsupervised learning is done by examining key features of the data and tries to describe a hidden structure with in the data.

Data Classification

Data classifications attempts to derive a class or group that the input field falls into. These categories are determined using a training set. Data classification models usually use a supervised learning method. For data classification, the result is the identification of the class label. The networks are evaluated on how well they classify the known data. The ideal result is a neural network that can classify unknown data class. A class is usually a non-numeric data attribute and the memberships of a class must be well defined. Example of a class would be type of flower, let's say "rose" with in a set of flower images.

Regression Analysis

Regression analysis is used to calculate and estimate the relationship between variables. Regression analysis allows neural networks to train neural networks with input data that is numeric and can estimate the conditional expectation of a dependent variable. Regression analysis uses a supervised training method and groups the data under the specified class labels.

Clustering

Clustering analysis typically uses unsupervised leaning methods. Clustering is capable of analysing numerical data and labelled data. The difference of clustering analysis is that it does not need a class label to group data together. The data that is like other data are grouped into clusters. The number of clusters expected must be defined before training starts.

Neural Network Structure

Neural networks are made up of different layers. It is important to note that not all neural networks have some layering structure and the layers mentioned in this section do not have to be present to make up a neural network. Neural networks are structured to accept data items and output data items. Typically, neural networks are layered with the minimum of an input layer and an output layer.

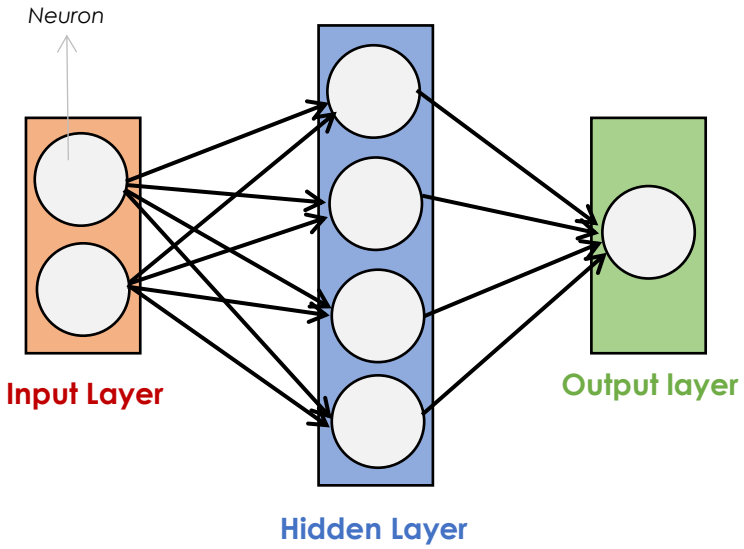


Figure 20: Visualisation of a typical neural network structure.

Input Layer

The first layer in a neural network is the input layer. The input layer contains a specific number of neurons. The neurons in the input layer all have comparable properties. The input layer generally has the same number of input neurons as there are attributes in the data set that the network will use for classification. Input neurons are passive neurons. Meaning that they do not modify the data in any way. If we consider an image with the dimensions 5 X 7 pixels, it would be need a total of 35 input neurons. Each x and y pixel coordinate of the image is an attribute of the image.

Hidden Layer

The hidden layer can take on a more complex structure within the network. The purpose of the hidden layer is to better produce the expected output. This is done by applying a function to the pervious layer that will adjust the connection weights of the neurons. The job of the hidden layer is to transform the input weights into something that the output layer can use. Hidden layers build up the complexity of the pattern recognition and the ability to which it can recognise. The challenge of implementing hidden layers is to avoid creating a structure that is too complex or too simple. If the structure is too complex the computational requirements increases and training time. If the structure is too simple the network runs the chance of not being able to learn the pattern accurately enough.

Output Layer

The output layer is the final layer in the network. The structure of the output layer is formatted very similarly to that of the data that was provided to the network. The real power of neural networks comes from their pattern recognition abilities. The neural network should be able to produce the desired output even if the input has been slightly distorted. The main challenge in determining the structure of the input data and attaching meaning to the output data. The regression model that was discussed earlier would typically produce a single output neuron that provides multiple numeric values. The classification model should produce one or more output neurons depending on how the

output has been encoded or normalised. The clustering model output neuron is equal to the number of total clusters defined in the training of the network.

Activation Functions

Activation functions are attached to the different layers and is applied by each neuron within the layer. The activation functions are used to model the probability of an actions potential for firing a neuron. The activation functions are usually a binary function that allows for a kind of on-off state. The activation function is used to scale the output of the neuron layers while keeping it within the normalized range.

Linear activation functions are primarily used for specific types of neural networks that have no activation function or they are used in the output neuron layer. This is to ensure that a constant derivative of 1 when producing the result. This allows the output of the function to be usable with backpropagation, which is a method of iterative training a neural network can use. This is discussed in the training section that follows. Below is the graphical representation of a linear activation

Activation sigmoid function is used when the expected output of the neural network needs to be a positive number. The sigmoid function can move negative numbers into a positive range.

The activation Tanh or the hyperbolic tangent function is used when the need for working with both positive and negative outputs arises. The Tangent function is the most common function used

Normalising Data

Normalisation refers to a series of sets that are used to adjust the data structure to ensure the alignment of input and output values are evenly distributed. Normalising data is done because of the need for universal data set structure. The universal data structure can be understood by the neural network when receiving input and communicating an output. This will allow for the comparison of corresponding normalized values from different data sets and produce outputs that can be interpreted into meaningful results. The normalisation of data will condition the neural network to look for relationships within the data sets by ensuring that the range of inputs and outputs remain within the specified range.

How Neural Networks Learn

For neural networks to learn they must be trained. The training process provides the neural network with an element of feedback to determine if it was successful. If we think of the way that children learn, they need to be told if what they are doing is right or wrong. For example, if someone is trying to through a ball through a hoop. When the person throws the ball, and notes how close they came to getting the ball through the hoop. The next time they would remember what went wrong and modify their movements accordingly to hopefully have a better result. So, feedback is used to compare the outcome that was desired with what happened and used to adjust the next try. The bigger the difference the more radically the movement would be altered.

Neural networks learn things the same way. Typically, a feedback process is used called back propagation. This involves comparing the output of the neural network produces with the desired output. The difference between them is used to modify the weights of the neuron connections working from the output layer through the hidden layers going backwards. In time the backpropagation causes the neural network to learn by reducing the difference between the actual and desired input until they come close to coinciding. The point at which the neural network stops learning an input is set by an error rate. The

error rate is the minimal level of error predetermined to satisfy backpropagation process.

POST PROCESSING

Post processing stage deals with the unrecognised characters found. There are two types of post processing techniques, Grouping and Error-detection and Correction.

Grouping

During the recognition stage, only single characters are identified. Each character needs to be associated to its respective neighbour to create a string of characters that makeup words. This association is known as grouping. Grouping of characters is based on the location of the symbol relative to its surrounding characters. If characters are sufficiently close enough to each other they are grouped together. Fonts that have fixed pitches are easier to group together as the position of each character is known. The distance between words is greater than the distance between characters and therefore it is possible to derive the beginning and end of a word.

Error Detection and Correction

After the grouping of characters is complete error-detection and correction can take place. Most OCR engines won't give 100% classification of single character recognition. Therefore, the makeup of words needs some mechanism to detect errors. On method is to use the possibility of characters appearing next to each other. By defining a syntax rule per the language specifics. For example, for the English language the probability of the letter "k" appearing after the letter "h" is extremely unlikely. Another approach is to implement the use of dictionaries. If the word is not found in the dictionary, then an error has been detected. The disadvantages of using the dictionary method are that they are time consuming and if a word is found in the dictionary it doesn't mean that no error is present.

LIMITATIONS OF OCR

The accuracy of an OCR system is dependent on the quality of the input image. The difficulties faced in varying documents can be defined by the following:

- Variations in font shape and size may not convert well and increase the range of difference between the same characters.
- Un-uniform characters, caused by noise, broken character lines and smudged prove difficult to deal with.
- Changes in spacing dew to orientation, change in font sizes with in the document.
- Defining text from graphics.
- Incorrect document boundaries. Documents that are warped and do not have a uniform structure need an extra level of processing.
- Languages. Different languages may contain characters not found in another, unless the OCR software is specifically trained on those characters.

SYSTEM ANALYSIS & DESIGN

The system analysis and design describes the system requirements, design details, architecture and processing and logic. The goals of the OCR software are to facilitate the processing of document images into machine encode text. The project is developed using the systematic software development life cycle.

The main principle of the project is to automate the recognition of patterns found in alphabetic and numerical character found in an image format file. Broken down the solution will need to provide a means of translating image data into an understandable structure the machine can process. The software will need to be able to be taught how to understand the content it is analysing and communicate a result.

The final product should provide the following functionalities.

- **Imaging process:** Allow for adaptation of the contrast, colour channels and noise removal. Determining what is background and foreground content.
- **Image handling:** Importing diverse types of file formats. Saving file formats. Copying images.
- **Document analysis:** Can identify text bodies and local and segment text and characters. Determine the difference between images and characters. Detect current transpose level of the document (skew detection).
- **Character recognition:** Create intuitive learning AI. Normalising data structures. Provide learning sets for AI. Saving created learning sets. Save the state of the AI. Provide a method of recognising characters. Handle unknown characters. Displaying the results of recognition.

DEPENDENCIES AND TECHNOLOGIES

The choices of technologies to be used within the project scope where determine according to their overall functionality, learning overhead, ease of use and available online support and materials.

Java

Java is a general-purpose programming language that is a class based object oriented programming language. Java was designed to have a few implementation dependencies as possible.

Java provides a lightweight user interface library called Swing. This will be used to build the user interface for the OCR software. Java also provides a robust automated garbage collector. The garbage collector is a memory management operation that runs behind the scenes in Java to ensure that memory that is no longer being used is freed for re-allocation and to lessen the overheads of RAM usage.

The choice for using Java is based on the level of documentation and list of global resources for the Java platform. Although the Java language can be long and in times a little more complex to implement than other languages. The fact that the automated memory management works so well and that the Native Just in Time compiler performance is rated very highly. Based on my current experience with the Java language this is the best fit for me and the project requirements.

OpenCV

OpenCV is a computer vision library written in C and C++. Originally developed by Intel, OpenCV is aimed mainly at real-time computer vision functions. OpenCV provides a common infrastructure to handle basic computer vision operations.

Even though OpenCV is written in C and C++, there are bindings available for Java. The Java bindings enable the OpenCV library to be used within a Java application. The OpenCV build path and the location of the .DLL files needs to be added to the Java application build configurations. This will allow Java OpenCV library to translate and utilizes the OpenCV objects and functions that are natively compiled in C/C++.

For this project, I used the OpenCV version 3.2.0. JRE referenced library configuration is as follows. The opencv-320.jar file is added to the build path libraries. Within the opencv320.jar libraries the native library location is set to point to the opencv.dll 64-bit file.

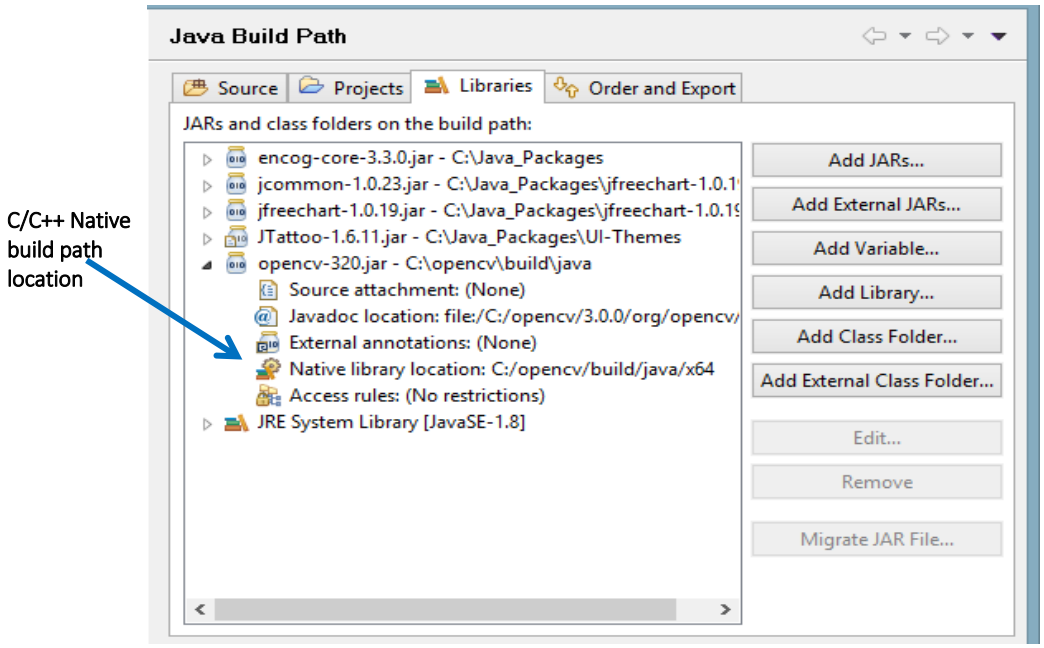


Figure 21: The OpenCV Java build path configurations.

For the OpenCV library and packages to be imported and utilized with in the code one more step needs to be applied. The Opencgv.dll library needs to be referenced in the code. This is done at the point of the applications start point and is done with the following script.

```
/**
 * Launch the application.
 */
public static void main(String[] args) {

    //Load the OpenCV native library
    System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

    //The rest of the code here...


}
```

Figure 22: Pointing to the native C library core in the Java source code.

OpenCV has well documented API and several streams active on online sources like Stack Overflow. Although most of the documentation is written for the C and C++ libraries, the Java implementation of functions and methods are done with similar syntax. The learning overhead for OpenCV is describe as quite steep due to the overlapping complexities of computer imaging. The benefits of OpenCV library is the ease of use. The syntax could be one line long where a complex and highly optimized algorithm is running and producing results. This takes a lot of the workload off running imaging algorithms and provides for more optimized code.

Encog API

Encog is a machine learning framework available in java created by Jeff Heaton, Encog is licenced under the Apache 2.0 licence. Encog strength lies behind it's Neural Network algorithms. There is a wide range of neural network classes that Encog supports.

Encog supports the creation of basic and complex neural networks and added support of normalising data classes, processing data and training neural networks. Encog tries to abstract some of the complexities behind neural networks to make implementation easier and more optimized.

Although there is not a lot of documentation and tutorials for Encog, there are a few published books that explain the Encog frameworks. The learning curve is very high for Encog, yet lower than other Java base neural network frameworks I encountered. Like Neuroph Studio which I had originally planned to use and encountered too many bugs and performance issues within the framework leading me to Encog Frameworks.

SYSTEM ANALYSIS

The following diagrams are to determine the order of process the system should follow. This represents the flow and the relationship between each process. Each process is dependent on the success of the previous process.

A high-level view for the proposed OCR system.

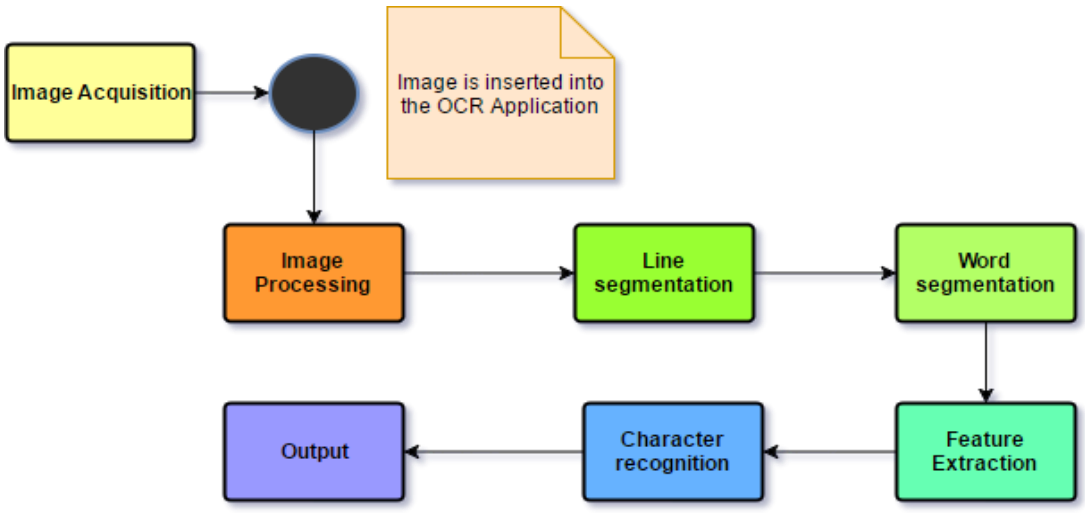


Figure 23: High-level process pipeline for OCR system.

Proposed image processing pipeline.

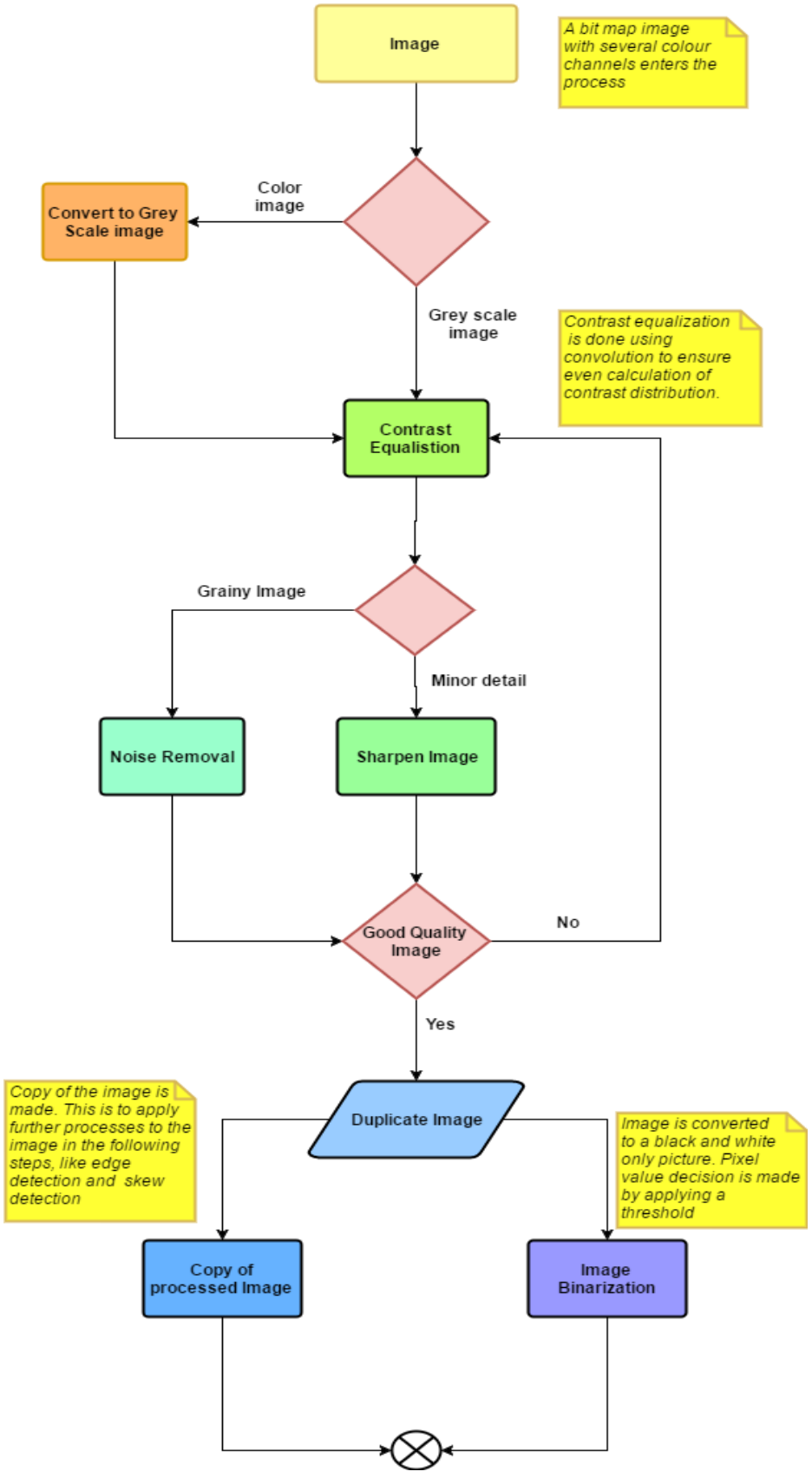


Figure 24: Flow diagram of the image processing pipeline

Proposed text and character segmentation process.

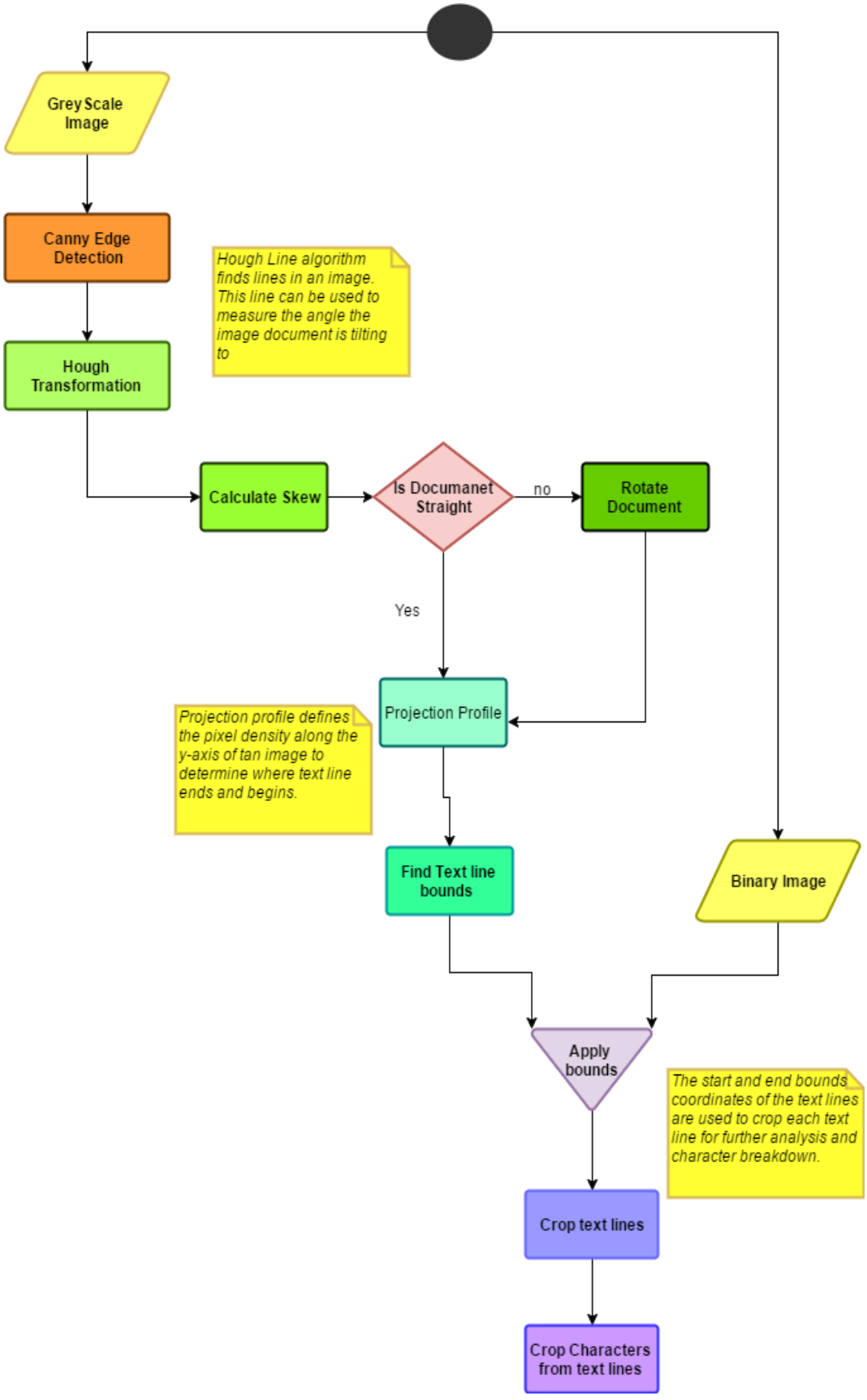


Figure 25: Flow diagram of the character location and segmentation process.

Proposed neural network creation and training.

Figure 26: The flow diagram for the neural network

Model-View-Controller

The choice for the software architecture is based on the MVC pattern. The MVC pattern works well when implementing user interfaces by dividing the application into three interconnected parts. This allows for the internal representations of the application to be separated into the ways information is presented to and accepted from the user. By decoupling these three major components will allow for more efficient code reuse and parallel development. The components of the MVC architecture are as follows.

Model:

The model is the central component and expresses the behaviour in terms of the area examined. The model manages the data, logic and rules of the application. The following objects are described and will be implemented as model components within the OCR software.

- The Neural Network:
- Training Sets:
- Images:
- Document analysis:
- Saving Objects:
- Imaging Utilities:

View:

The view is the output representation of the information. Multiple views of the same information is possible. The view will represent and display the changes made to the user interface communicated through the controller to the model. The following objects will be implemented as view components in the OCR software.

- The user interface.
- The image display.
- Image controller tools.
- Neural network interface.

Controller:

The controller is the mechanism that accepts input and converts it into commands for the model or the view. The controller is the means of communication with the model and the view components of the applications and adjusts accordingly to each of them. The following objects will be represented by the controller components within the OCR software.

- Menu bar controllers
- Buttons.
- Primary actions.
- Imaging tools.
- Preview actions

Wire frame Design

The wire frame design provides a visual guideline to suggest the structure of the user interface for the OCR software. These designs served as a blue print to the construction and to help define functionality of the view component.

Essential elements and requirements:

- Menu bar:
    - Facilitate primary actions
        - Opening images
        - Saving images
        - Creating new projects
    - Provide other ways of carrying out an action
        - Image processing functionality
        - Neural network configuration
        - Training set creation
    - Help option
- Imaging Tools:
    - Provide tools to apply imaging actions
    - Way of reverting changes made to images
- Neural network configuration
    - Allow for neural network creation
    - Load existing neural networks
    - Training neural networks
    - Loading existing training sets
    - Creating training sets
- List of images loaded into the application
    - Opening images action
    - Saving images action
    - Copying images action
    - Deleting images action
    - Selecting and displaying images for a list
- Display panel
    - Display selected images
    - Allow for zooming in and out

After consideration of the required elements the following structure was created. The general design is broken down further to plan for detailed layout management and frame and panel layering.

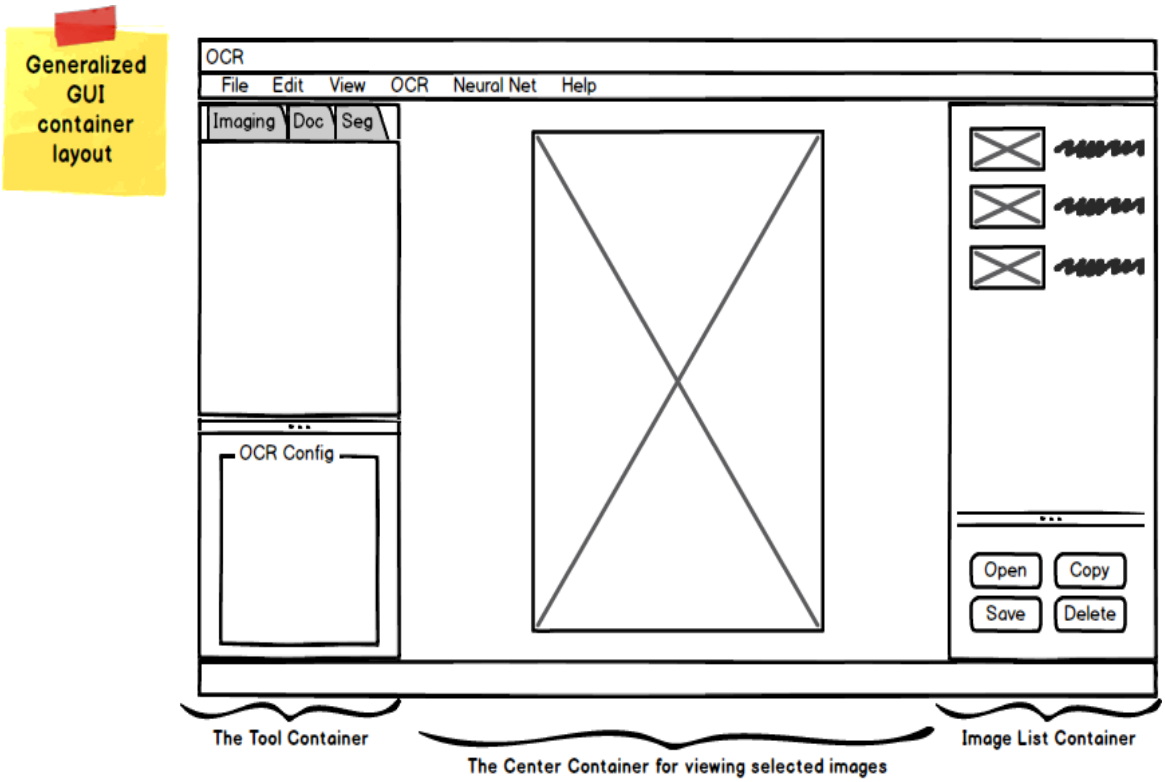


Figure 27: The proposed GUI design.

The following wire frame is a breakdown of the underlying layout. Each panel is sized by calculating the size of the native resolution of the device and setting a set parameter to divide it by. This is to help contain the panel resizing correctly due to change of native screen resolution.

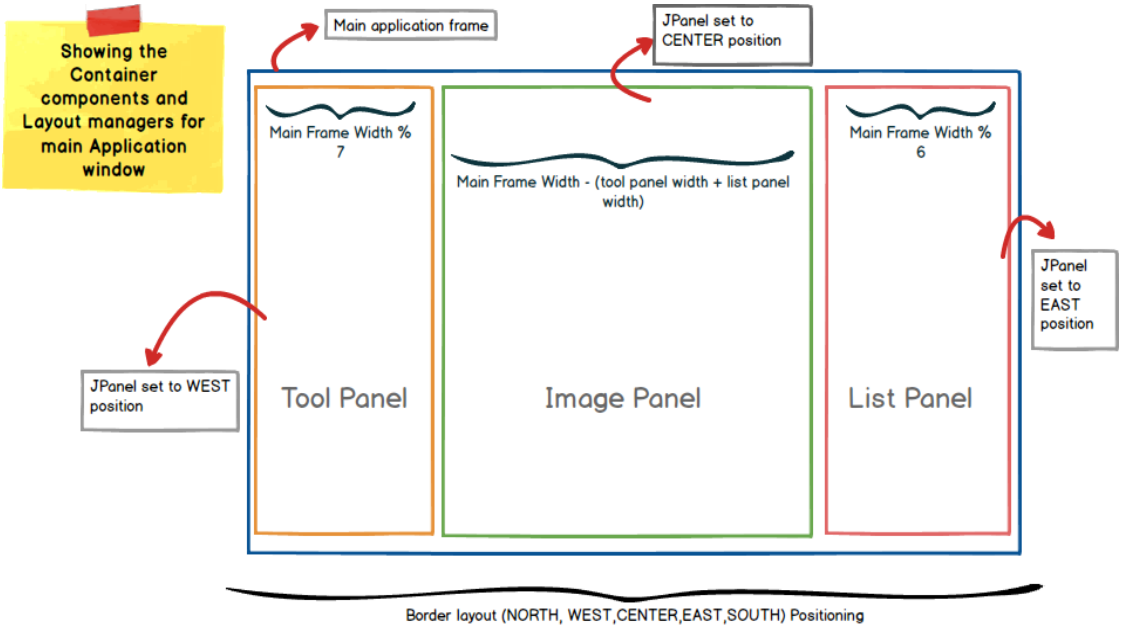


Figure 28: Root frame layout.

The following diagram describes the layout and components within the centre panel that will be used to display the selected image.

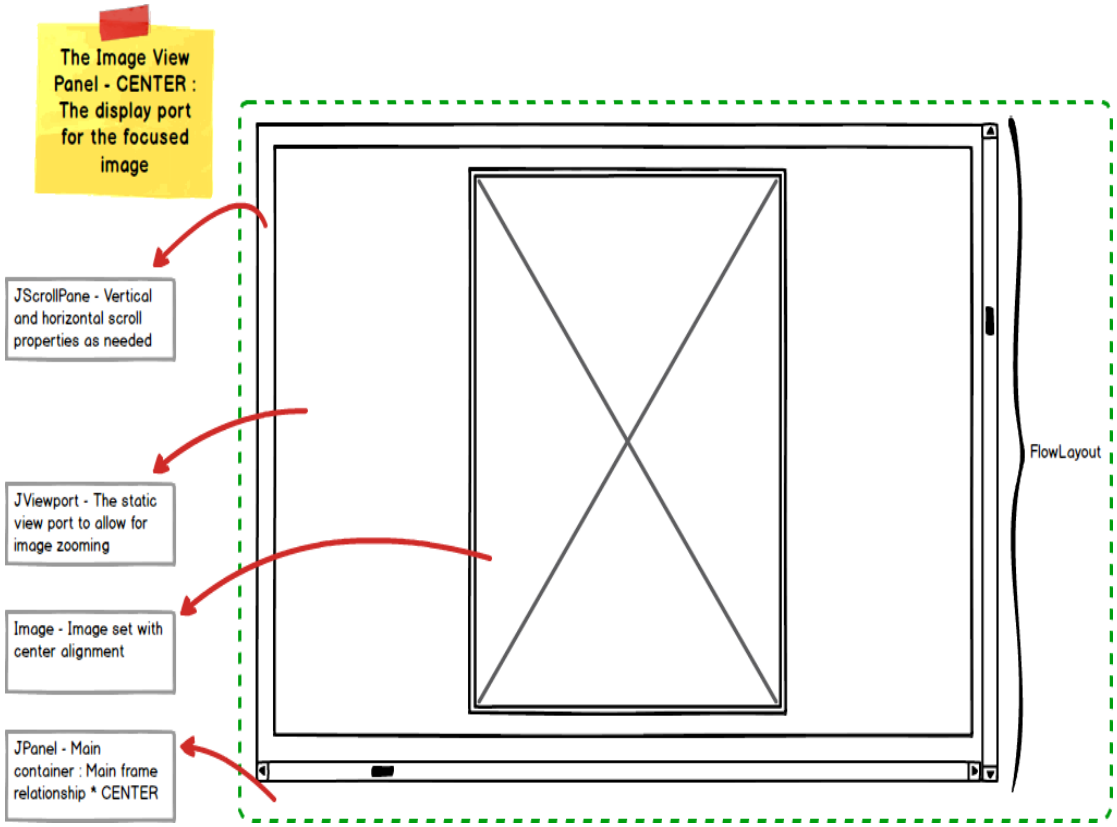


Figure 29: Centre panel layout and components.

The following diagram describes the tool panel on the left side of the root layout.

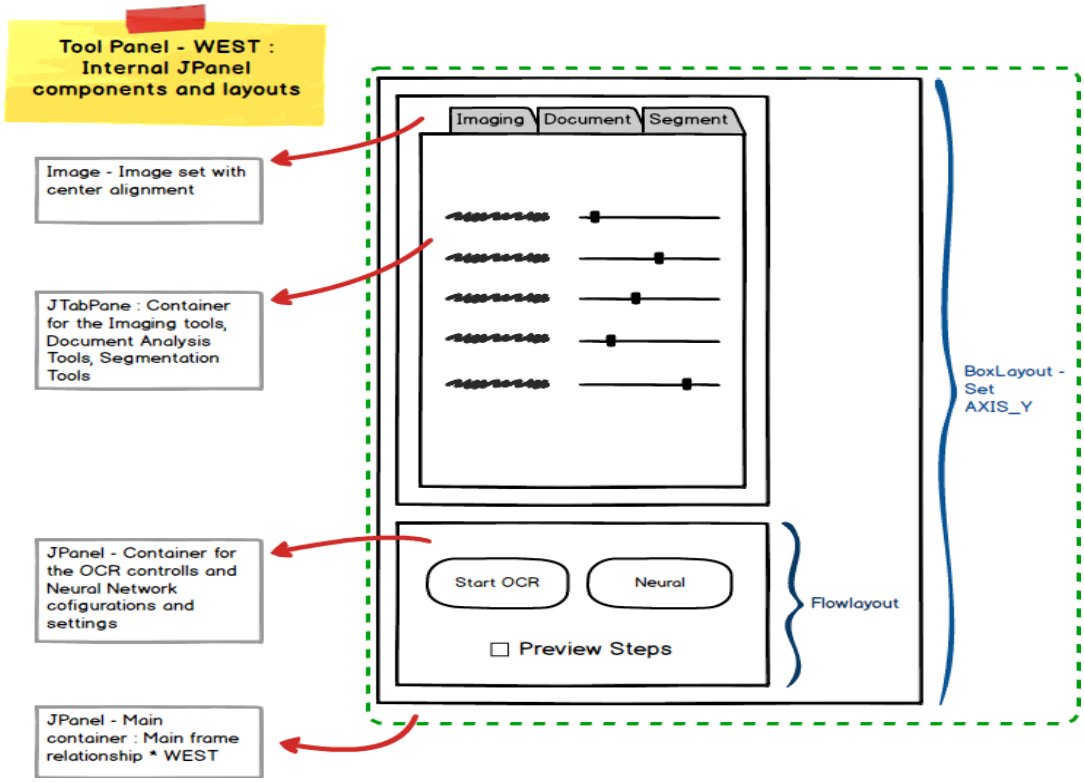


Figure 30: The layout and components of the Left panel.

The following diagram describes the open image list panel.

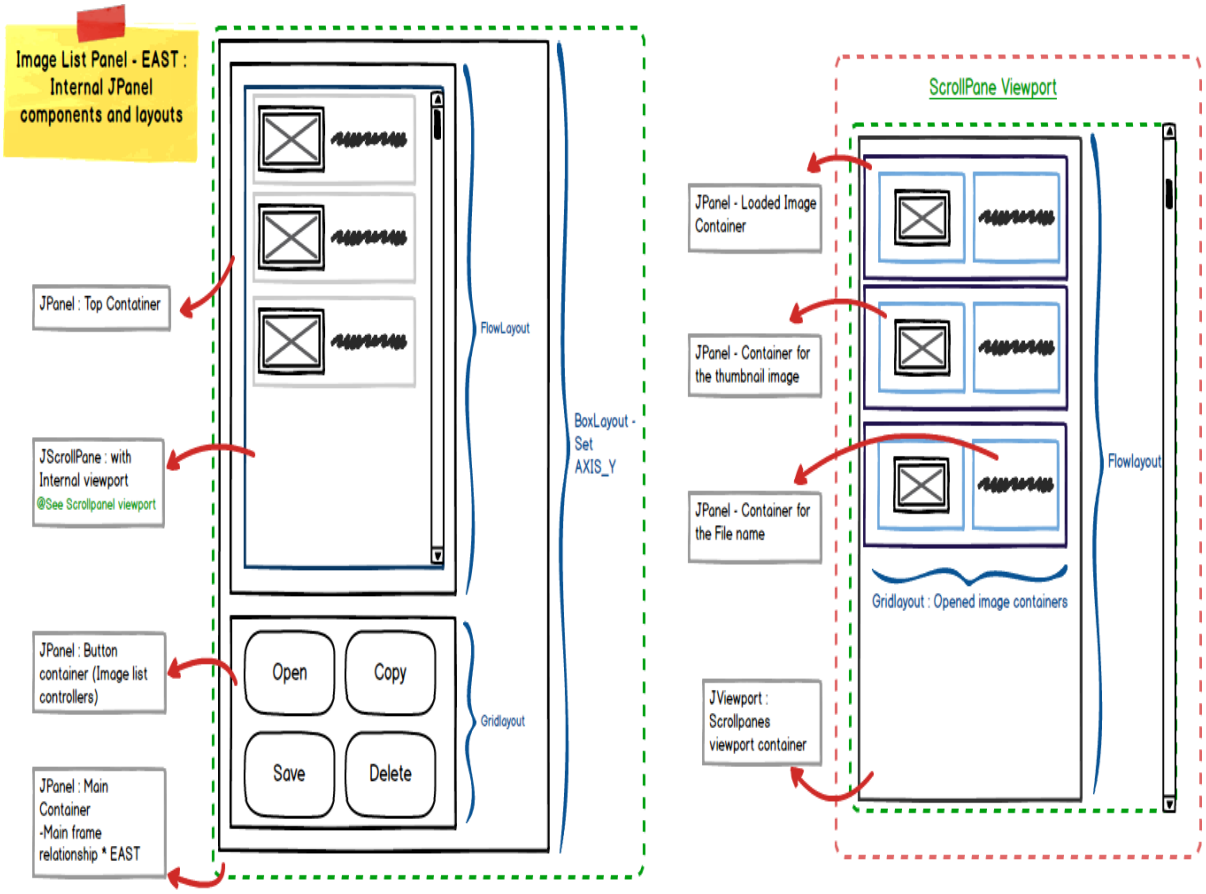
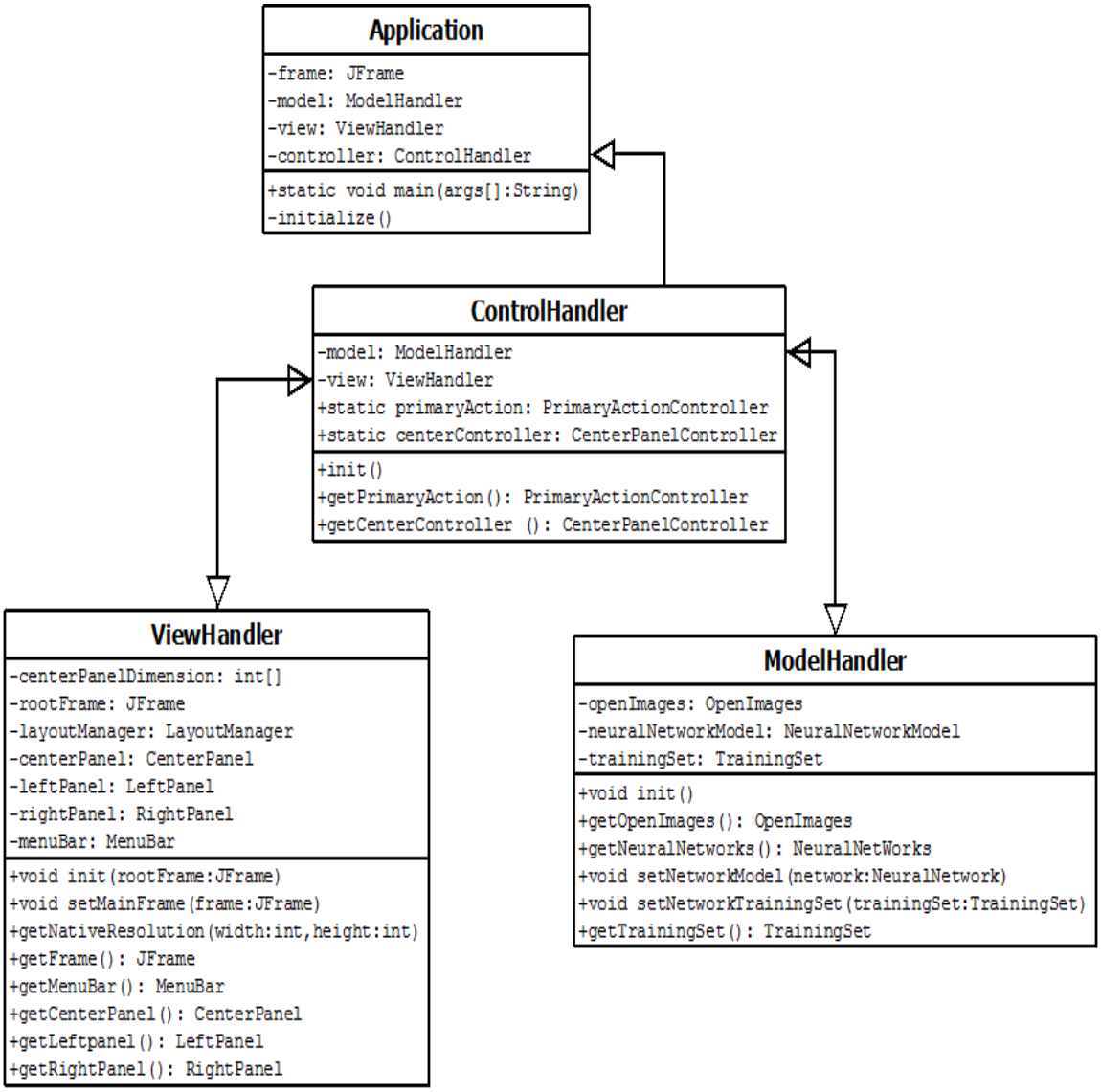


Figure 31: The layout and components of the right-side panel.

Class Diagrams

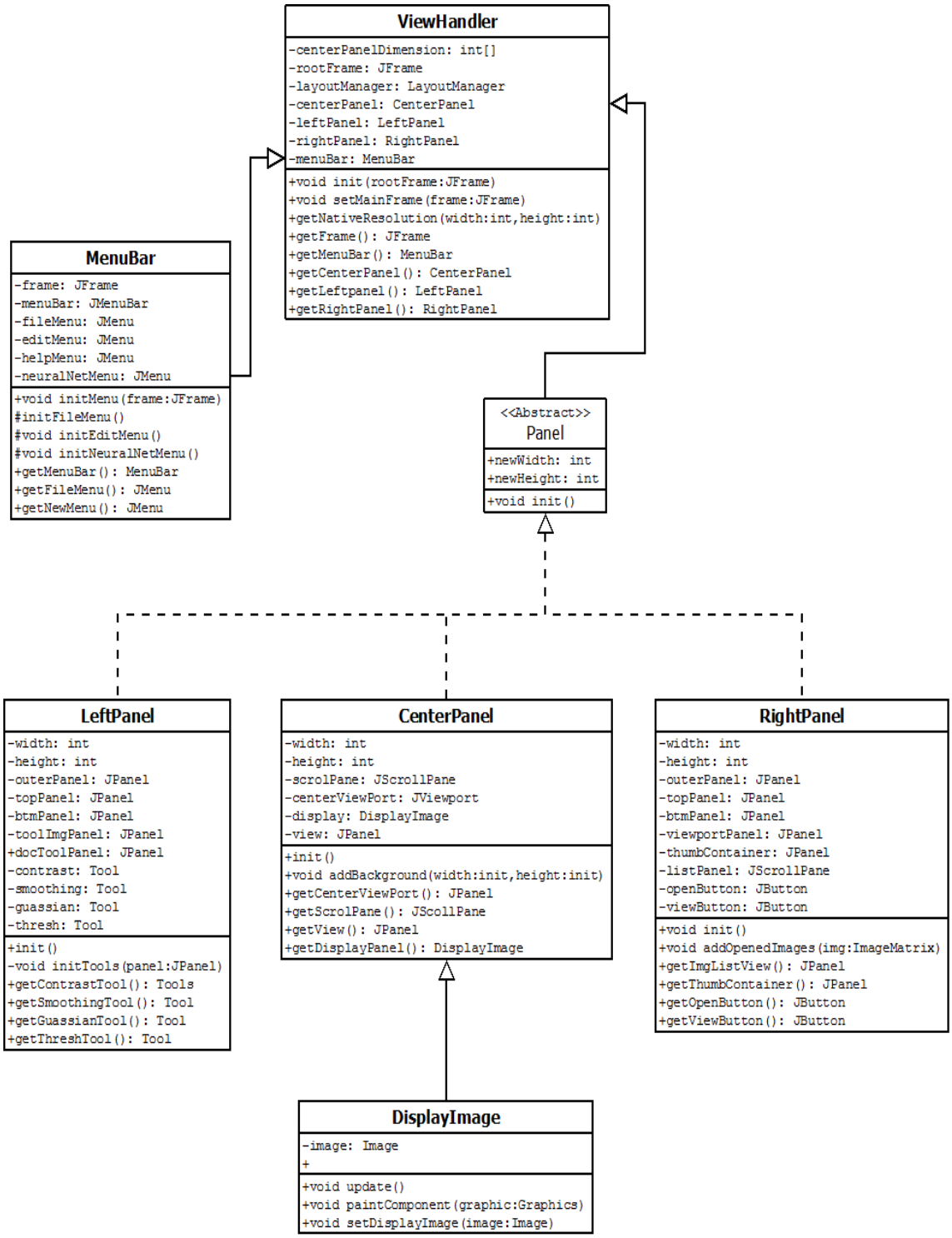
Top level MVC application structure:

The class diagram below represents the MVC structure and relationship between each of the MVC components. Each component is represented by a simplified interface allowing all the related child components to be initialized and handled by a central class, denoted as a handler. This structure will allow for the creation of each object and its internally initialized objects to retain their current state in relation to the process of the application. Each class can be accessed via these relationships shared whilst still encapsulating the attributes and methods.



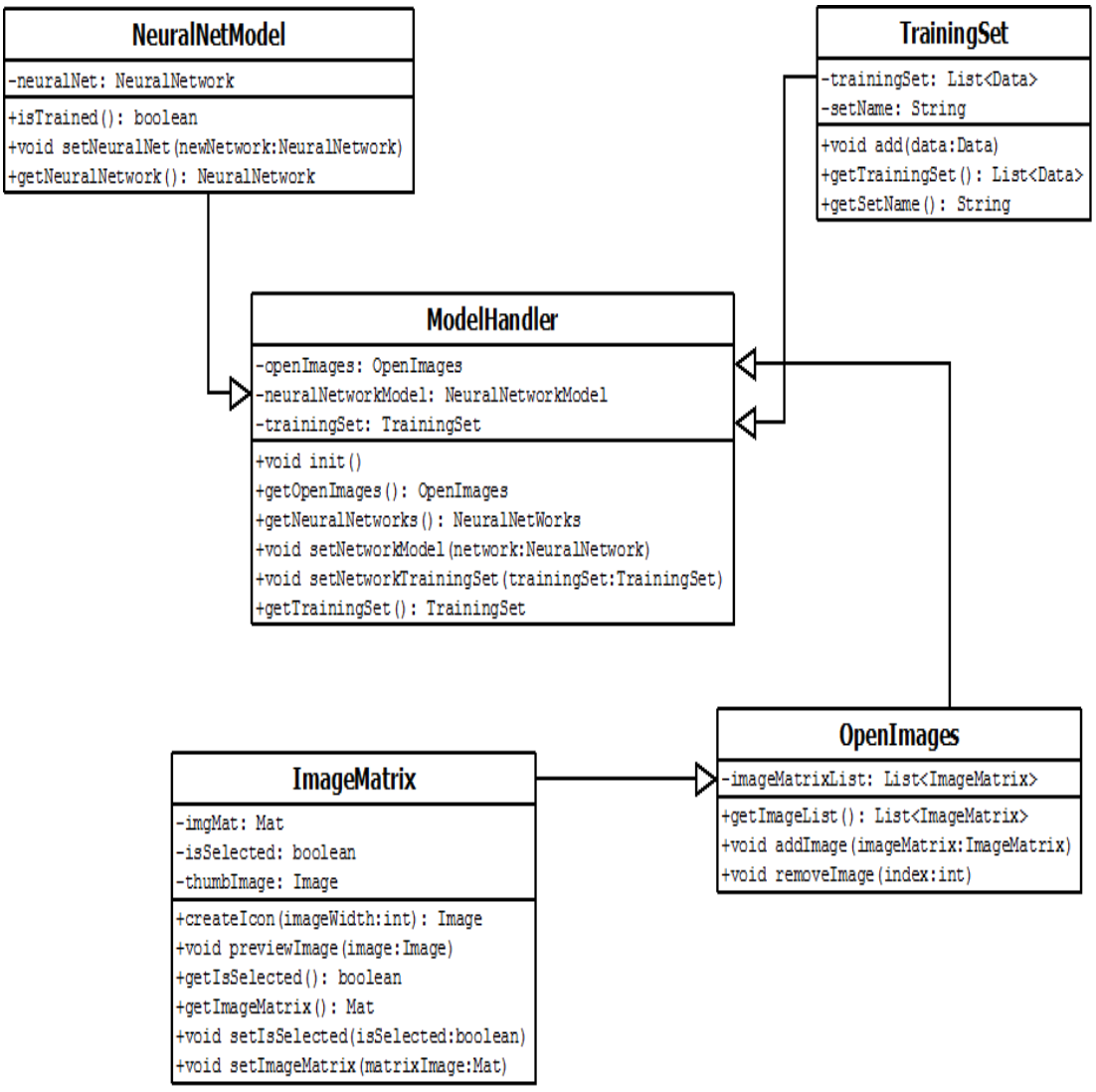
View Class handler:

The view is built by adding each component of the UI to a layout manager. Each object applies its own methods and internal layout to the main content frame via the 'View Handler' class. A basic factory pattern called 'Panel' is used to extent attributes shared by the main UI panels that span the width of the interface. This allows for each of the extended 'Panel' objects to provide a common interface and provide individual design, layout and components.



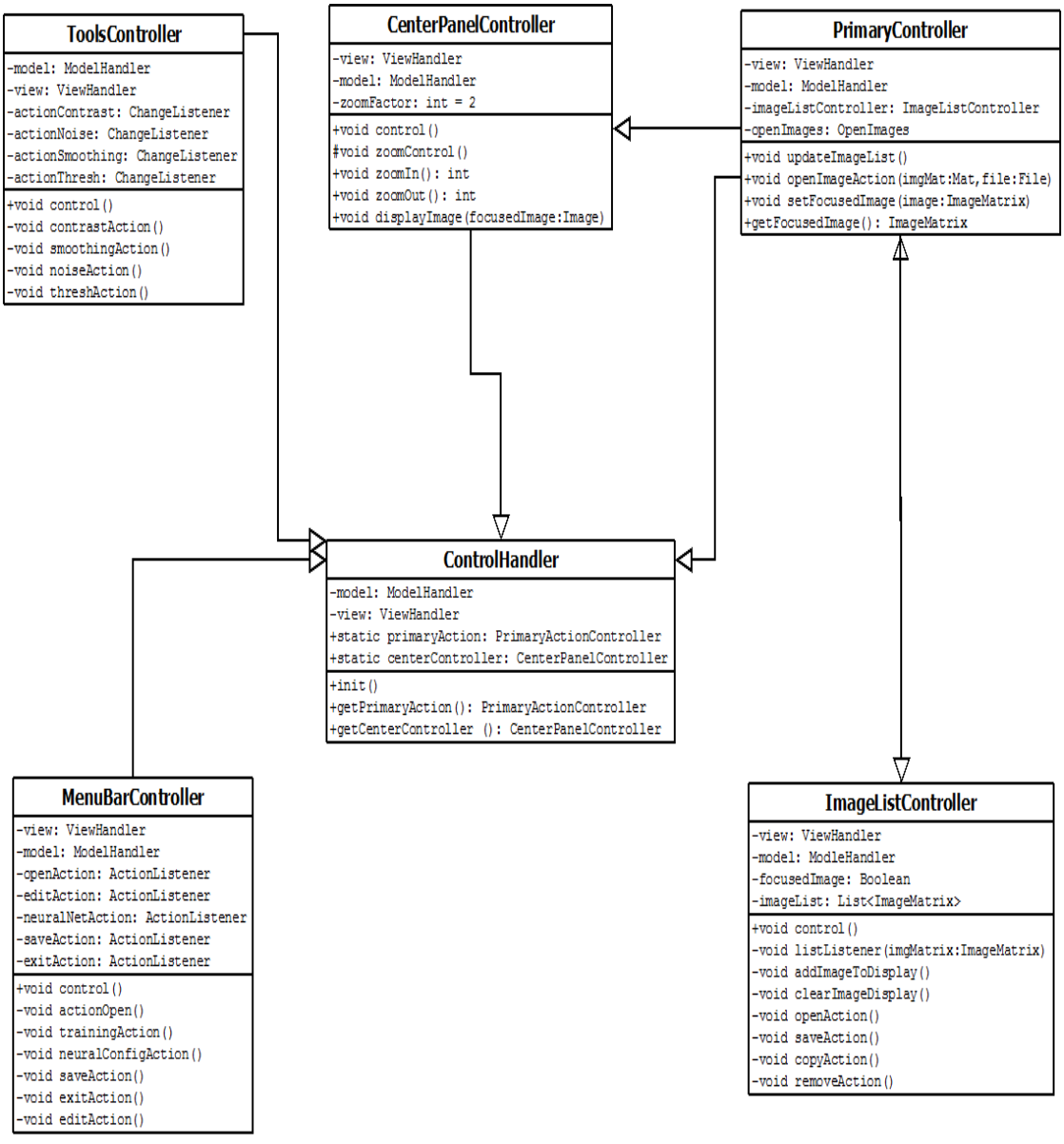
Model Class Handler:

The 'ModelHandler' will provide a common interface that each state of a model object can be set or gotten through. This will allow for the retention of the current state of each model object. The 'NeuralNetModel' is used to set a neural network to be used. The neural network model structure will allow for various kinds of neural networks to be used. The training set state will retain the normalised training set that will be feed into the neural network. The 'OpenImage' class is a list of the images loaded into the application. These images are displayed in the views 'RightPanel'. The 'ImageMatrix' class stores the images in different formats and provides the common attributes each image object has.



Control Class Handler:

The control handler applies a central interface to retain and add the UI action controls. The control elements are broken down into the respective panels the action is being triggered from. The exception to the design is the 'PrimaryController', which is a class that provides the actions that are applied over serval classes because of the widespread use of the actions.



IMPLEMENTATION

In order to minimise the complexity and to keep the flow of the program constant Modular programming approach was applied. By following this standard, it helped encourage me to use subprograms and organize the code better.

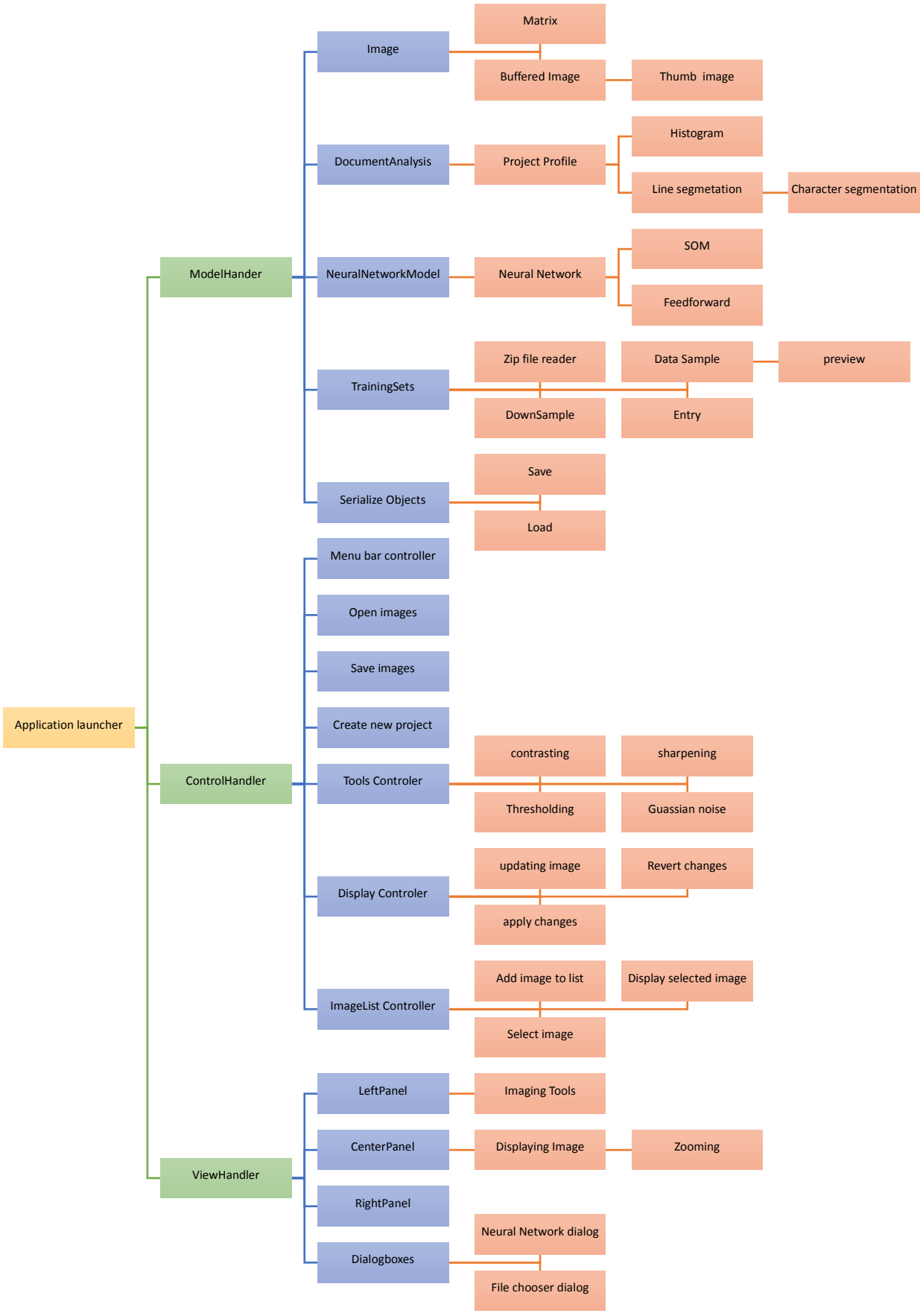
In a modular programming approach the code is broken down into smaller groups of instructions known as modules or subprograms. This made implementing various aspects of the software easier and safer. The reason being that I could build and testing each module separately from each other. This made it easier to understand how the logic was functioning and simpler to isolate bugs and errors. Figure 17 depicts the module structure. The structure shows the hierarchical relationship of the modules. The relationship and interactions is not representative of the structure. Each module correlates to a class within the software.

With regards to the neural network structures I had planned to implement two different structures. A Self Organising Map, SOM and a Feed Forward network. I only manage to get the SOM network implemented in the end. A SOM neural network is a very basic structured network. It only has two layers. An input layer and an output layer. At a basic level a SOM groups and clusters similar inputs together during the learning stage. During the recognition stage the group id of the cluster that best fits the input value is mapped to a set of known outcomes and produces the result.

Implementation choices:

Component	Method	Using
Feature Extraction	Template Matching	Self-employed solution
Neural Network	Self-Organising Map	Encog API
Document Skew Detection	Hough Line Transformation	OpenCV API
Document analysis	Horizontal Profile Projection using histogram data	Self-employed Solution
Contrast Manipulation	• Histogram Equalisation • Contrast Sketching	OpenCV API
Low Pass filtering	Gaussian Algorithm	OpenCV API
Edge Detection	Canny Algorithm	OpenCV API
Binarization - (Bi-level conversion)	Local Threshold Algorithm	OpenCV API

Application Modules



TESTING AND EVALUATION

Testing each module of was done outside of the full application to help isolate and understand what was happening with in the code. The following section captures the testing of each of a few of the modules including the user interface.

This is an almost completed view of the application user interface.

Figure 32: The User interface.

The user interface demonstrating the image list on the right and the selected image open in the display panel in the centre.

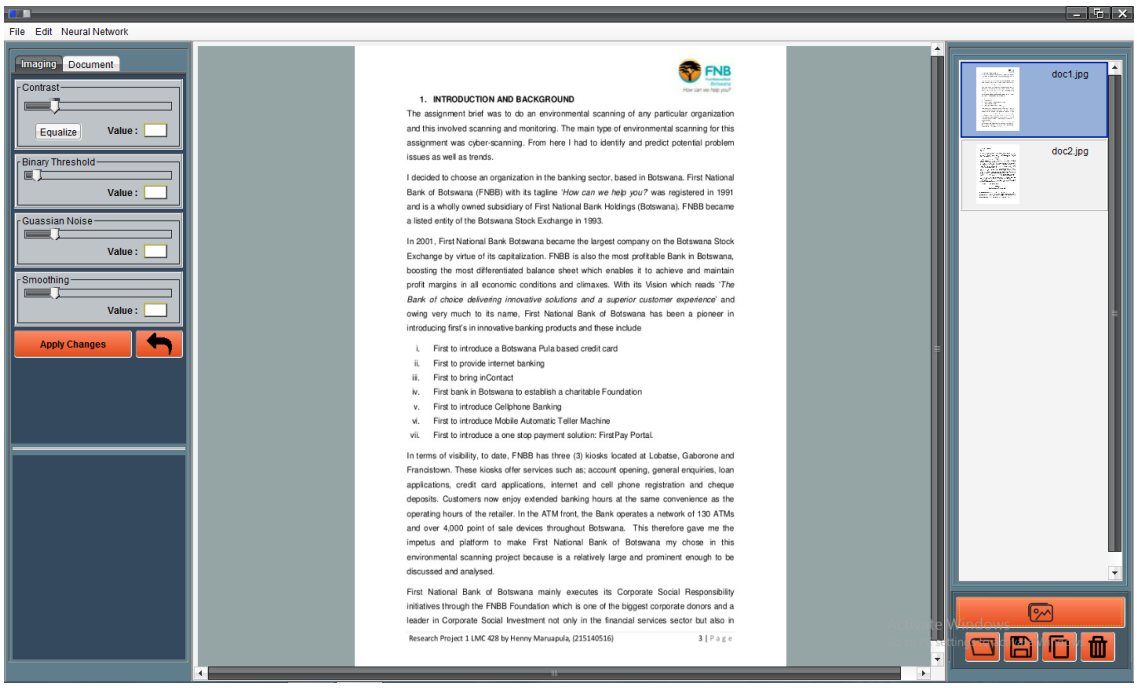


Figure 33: The interface in action.

Based on the image loaded in the previous image. The contrast sketching functionality has been applied using the contrast slider in the left panel. This method of contrast sketching work just as intended.

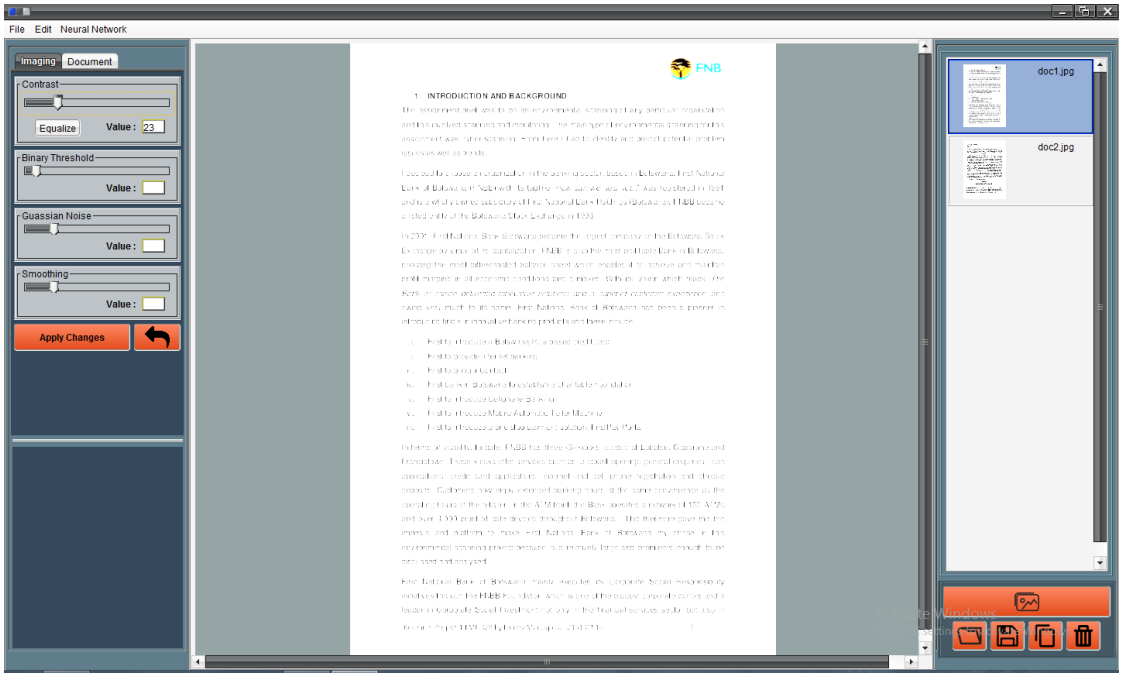


Figure 34: Contrast sketching action applied to the image.

The next example is of the contrast equalisation. This did not work as intended. I believe the reason is that the algorithm is being applied to the entire image. To improve the result of the figure 33, I would have like to have used the contrast equalisation as a conventional method. This would rather get the average contrast difference over the 3 by 3 kernel size and have a more even contrast distribution.

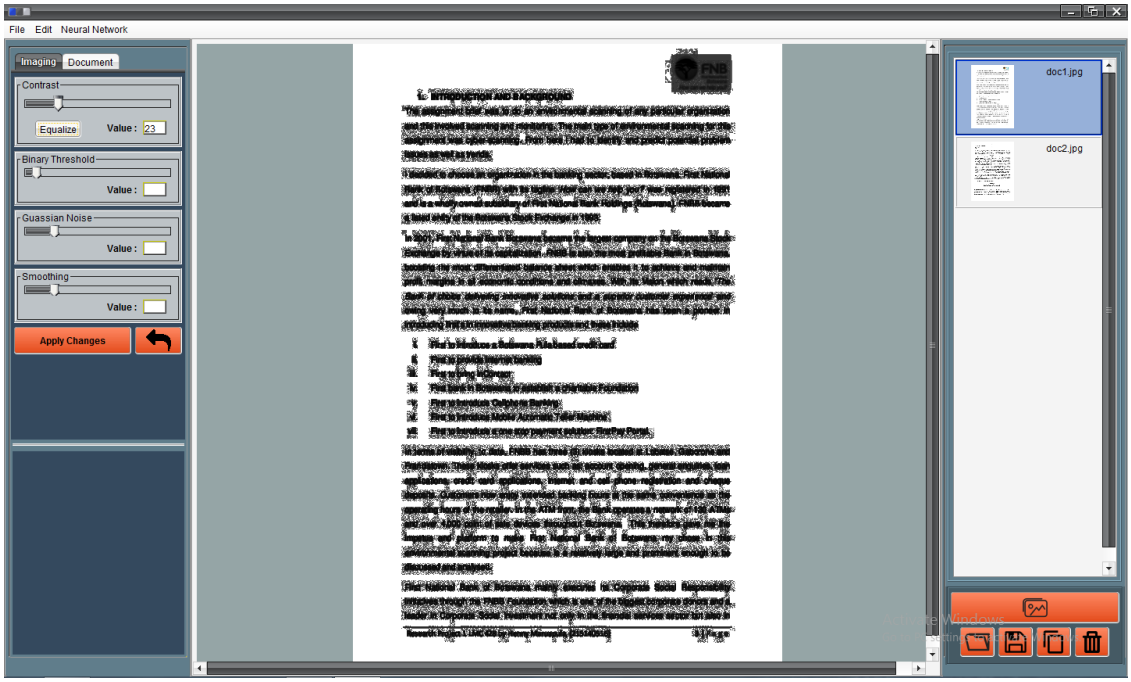


Figure 35: Contrast equalisation attempt.

Testing the creation of the training sets is shown below. The method for feature extraction is matrix matching. Each character has been converted from a .png image. The characters have been down sample well and can still make out the details of each character.

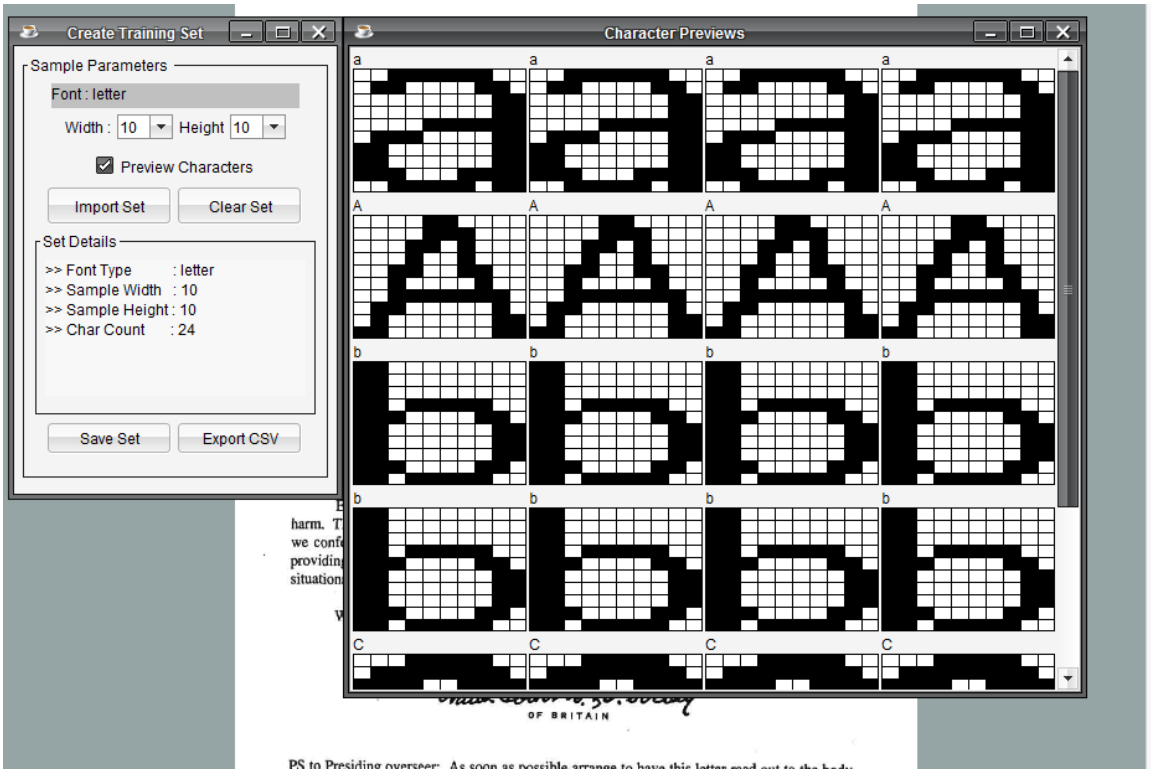
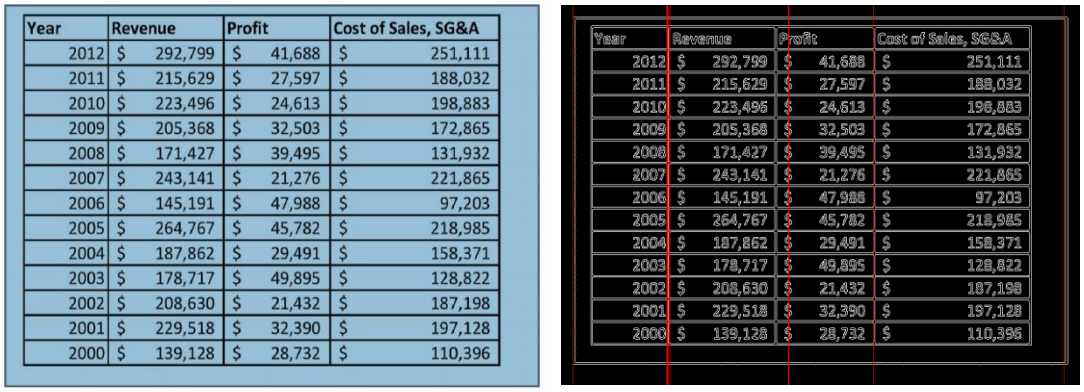


Figure 36: Testing the creation of training sets. Loaded in from a zip file.

These are the copies of the images used to create the prototypes in the preview above. The letters of the right show how the images are down sampled to crop out any unwanted area.

Figure 37: Characters used to create the prototypes.

The following tests were conducted to see if it was possible to find the lines that made up an excel table. The idea was to be able to extract text relevant to each cell within the table. Unfortunately, the Hough line transformation algorithm needs a bit more work and debugging to be usable. These are the results of the mentioned test.



Year	Revenue	Profit	Cost of Sales, SG&A
2012	$ 292,799	$ 41,688	$ 251,111
2011	$ 215,629	$ 27,597	$ 188,032
2010	$ 223,496	$ 24,613	$ 198,883
2009	$ 205,368	$ 32,503	$ 172,865
2008	$ 171,427	$ 39,495	$ 131,932
2007	$ 243,141	$ 21,276	$ 221,865
2006	$ 145,191	$ 47,988	$ 97,203
2005	$ 264,767	$ 45,782	$ 218,985
2004	$ 187,862	$ 29,491	$ 158,371
2003	$ 178,717	$ 49,895	$ 128,822
2002	$ 208,630	$ 21,432	$ 187,198
2001	$ 229,518	$ 32,390	$ 197,128
2000	$ 139,128	$ 28,732	$ 110,396

Figure 38: Hough line transformation test.

As we can see the algorithm is correctly identifying lines found in the image, but seems to get stuck at finding the same line. If we look carefully at the second red line, it is a group of lines. This test lead to the test of the Hough transformation in the use of document skew detection. The algorithm could identify at least one line within a document that it is able to use to adjust the level of the image.

2

Signature-Based Retrieval of Scanned Documents Using Conditional Random Fields

Harish Srinivasan and Sargur Srihari

Summary. In searching a large repository of scanned documents, a task of interest is that of retrieving documents from a database using a signature image as a query. This chapter presents a signature retrieval strategy using document indexing and retrieval. Indexing is done using (i) a model based on Conditional Random Fields (CRF) to label extracted segments of scanned documents as Machine-Print, Signature and Noise, (ii) a technique using support vector machine to remove noise and printed text overlapping the signature images and (iii) a global shape-based feature extractor that is computed for each signature image. The documents are first segmented into patches using a region growing algorithm and the CRF based model is used to infer the labels of each of these patches. The robustness of the method is due to the inherent nature of modeling neighboring spatial dependencies in the labels as well as the observed data using CRF. The model parameters are learnt using conjugate gradient descent with line search optimization to maximize pseudo-likelihood estimates and the inference of labels is done by computing the probability of the labels under the model with Gibbs sampling. A further post processing of the labeled patches yields signature regions which are used to index the documents. Retrieval is performed using a matching algorithm to compare the query with the indexed documents. Signature matching is based on a normalized correlation similarity measure using global shape-based binary feature vectors. The end-to-end system is a content-based image retrieval system designed for signatures.

Introduction

Retrieving relevant documents from a repository of scanned documents has many applications including the legal and forensic domains. In particular documents containing handwriting have a potentially useful role in counter-terrorism operations, e.g., retrieving forms filled out by certain applicants for opening post-office boxes, identifying envelopes of interest in the mail stream, etc. In searching complex documents, a task of relevance is relating the signature in a given document to the closest matches within a database of documents; this is the signature retrieval task which is addressed in this chapter.

S. Argamon, N. Howard (eds.), *Computational Methods for Counterterrorism*, DOI 10.1007/978-3-642-01141-2_2,
© Springer-Verlag Berlin Heidelberg 2009

17

Figure 39: Result of the Hough line transformation, To find out how much to adjust image skew.

The document analysis was done using horizontal projection profiling. The following image shows the analysis segmenting text line

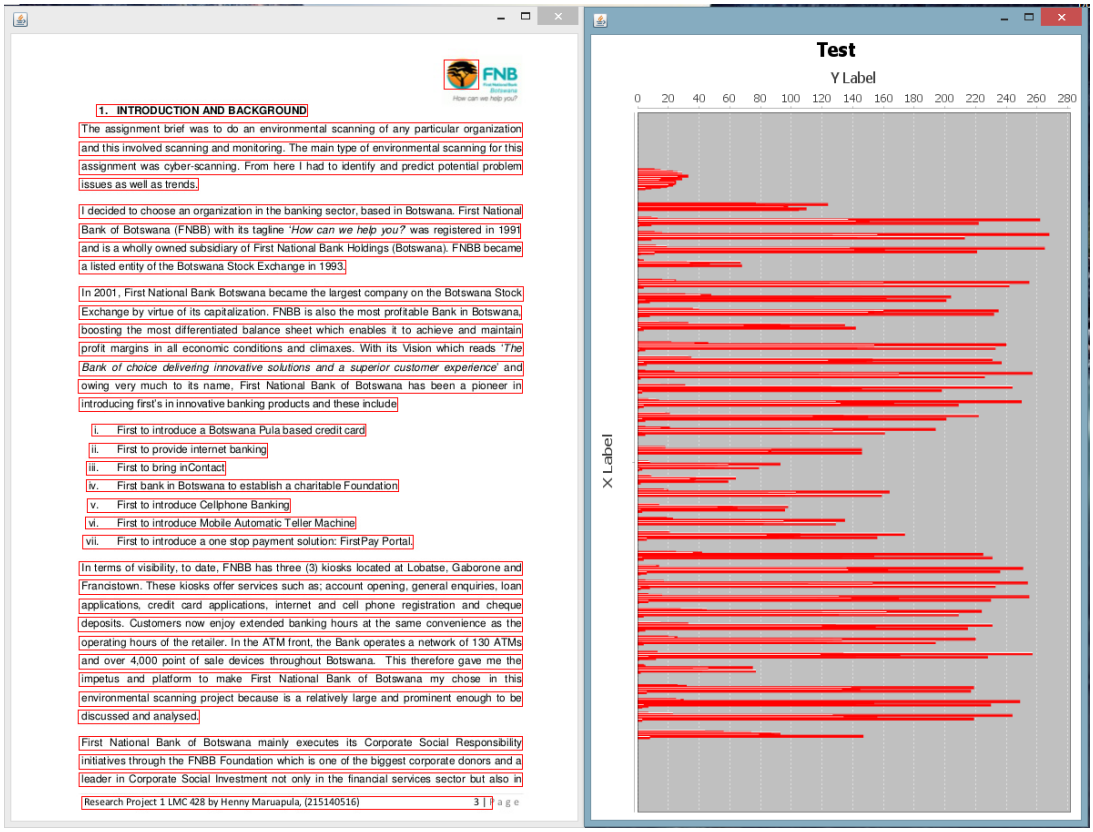


Figure 40: Project profile test.

Testing the neural network was done outside of the complete application. The network was trained on a set, A to Z characters upper and lower case of the Calibri font. To test that the network was successfully recognising characters an unknown and non- uniform character of the same font was fed into the neural network.

The first attempts varied in success. Most of the time letters where being correctly recognised. And other times, the same character where not being recognised. I started to notice that the letters that the network was having trouble recognising the difference between the letters '*E*' and '*F*' and '*H*', '*a*' and '*o*' and '*c*', '*i*' and '*l*' and '*t*'. I figured out through trial and error that the neural network needed a larger and more varying data to learn from. This did solve the problem and nearly all of the time the network was able to recognise the characters. Figure 37 shows the output console of the neural network running with in an IDE. The network was asked to recognise the letter 'r' and was successful. And the neuron that won was neuron 43.

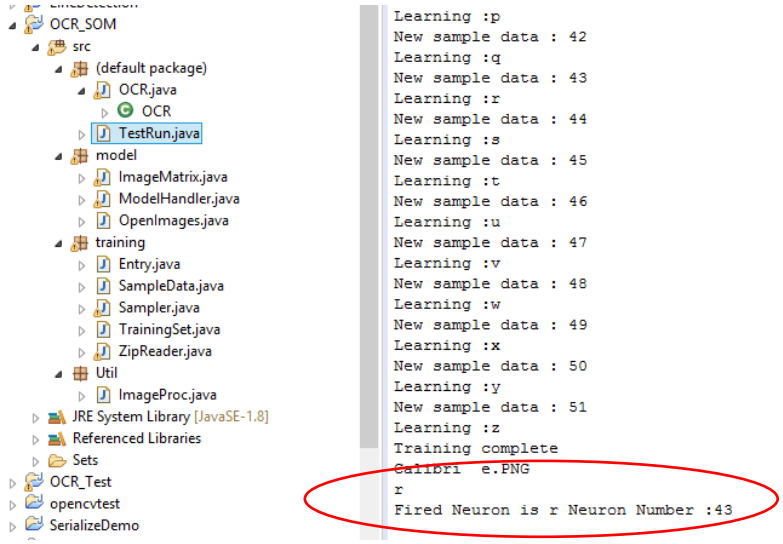


Figure 41: Console log of the neural network recognition test.

CONCLUSION

In this project, I studied several methods used for Optical Character recognition. These methods were the applied to a newly implemented software shell to replicate these techniques. The results are partially successful. The application tries to visualise each step of the process. This is to help understand what is happening under the hood and make it apparent to the user what is happening by previewing the results and process. When making the implementation choices the decision to use techniques and methods that took the simplest implementation and often the results showed this. This journey was to self-learn, about neural networks and machine learning, and apply some of my previous knowledge in computer vision.

Template matching method used as the feature extraction method possesses many difficulties when looking to scale. Template matching is a very simple technique that does not handle poorly processed images well. Therefore, images with varying impairments like noise, poor illumination or poorly scanned images would make it near impossible to prepare accurately for an OCR system. In the future, I would like to further the studies on Topological and Structural feature extraction. By bettering the methods of extracting curves, dips, peaks and valleys with in features really is the only feature extraction method that I believe show any likeness to intelligence. The implementation of the Self-Organising Map neural network, while it works fine for this application. It too does not have to potential to scale very well. SOM networks need a greater variance in classes to be effective. For example, if we had to add more font types to the training set of the SOM the network would still converge, but when it comes to recognition the network would battle to recognise patterns. As the patterns would be so like each other that they would eventually converge, because SOM networks cluster and group similar types. However, they do use very little computing power. Whereas the more optimal answer would have been to implement a Feed Forward neural network (MPL) as they can facilitate multiple hidden layers and can apply deep learning methodologies.

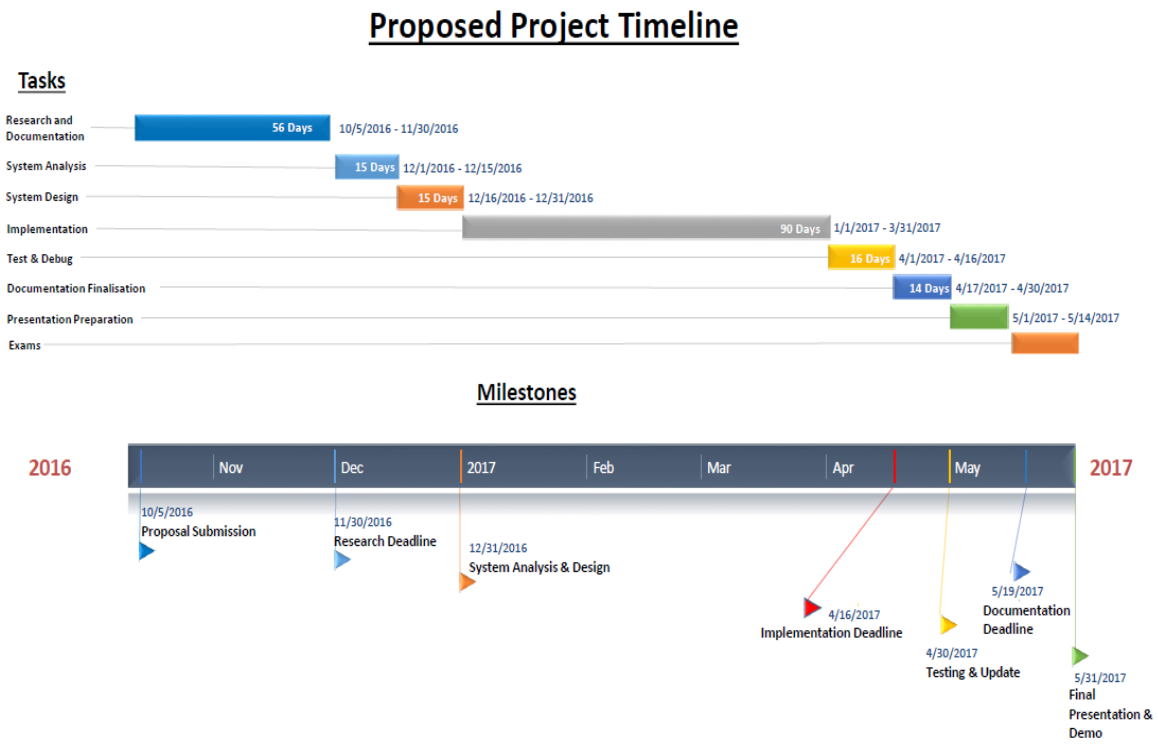
With a significant improvement in the development and understanding of Neural Networks, giving machines the ability to learn and act rationally, is becoming a reality. With research into Deep Neural Networks we are only starting to understand the potential of machine learning and getting more insight into how the human brain functions. Although OCR is widely used in commercial businesses and has been successfully deployed. The applications for neural networks are expanding rapidly, with machines predicting future business trend, playing a game of chess better than a person, diagnosing medical problems, controlling traffic lights, self-driving cars, statistical analysis, the list goes on. The future in computing applications is in AI.

APPENDICES

APPENDIX A: PROJECT PLANNING

In preparation of the project planning I broke down the constant requirements that I knew would have to be delivered regardless of the project choice. These requirements are as follows. Research and draft documentation, System analysis, System design, Implementation, Testing and debugging, Final documentation and Presentation preparation. I also took into consideration the pressure of exams to evenly distribute time spent on each area.

After I had made my choice to peruse OCR as my subject I was able to establish a more detailed time line. I devised a poker planning method (for one) and this was the result. I prescribed myself two months to gather my research and to undertake the task of learning the subject. I decided that a full month would be enough time to analyse and design the application. I extended the implementation time to 3 months with the understanding that testing and debugging some of the modules would happen within this period. The testing and updating phase would last half a month although I would have like to have used more time to apply here, it just wasn't feasible. I allowed the other half of the month to finalise the project documentation. The resulting timeline was created with regards to the poker planning.



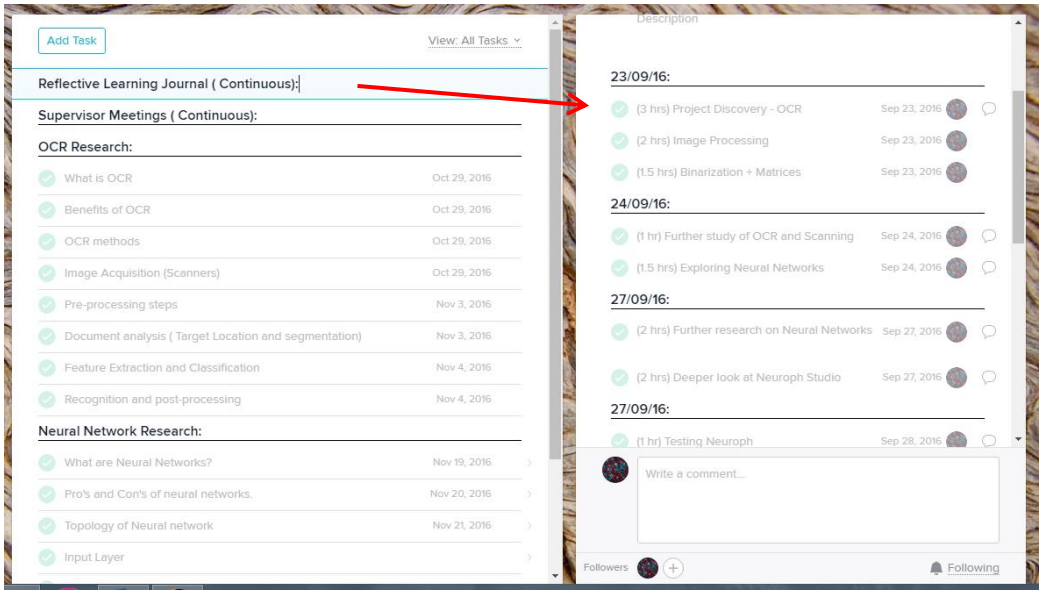
Throughout the project, I managed to stay up to date for most of the schedule. Implementation did only start in the beginning of February, but went smoothly and was able to make up most of the lost ground.

APPENDIX B: REFLECTIVE JOURNAL

For the purpose, of task tracking and recording a reflective journal, Asana task management tool was used. As the project developed from the research process the task were added. This allowed me to track my progress and for my project supervisor access and monitor progress made throughout the activity. The Asana project may be viewed using the following link:

https://app.asana.com/-/share?s=189939322551905-IATQ7RCbfg1n8PgOdSfBRaJFoXchRe2AAoFLEYekQ0W-59678617231603

There is a recording of the time spend on sections of research. This is shown in figure 19 by the red arrow. My research sources consisted primarily of research journals and papers published by universities all over the world.



The Asana project task list is broken down into several categories. These categories where designed to follow the proposed project timeline. The full contents of the Asana project are as follows:

- **Reflective Learning Journal:** Describes the time spent on each research aspect of the project.
- **Supervisor meetings:** A recording of meetings held with project supervisor
- **Optical Character Recognition Research.**
    1. **What is OCR:** Understand what OCR is at a high level.
    2. **Benefits of OCR:** Determine the practical uses of OCR
    3. **Methods of OCR:** What ways OCR is implemented, Core functionality
    4. **Image Acquisition using scanners:** Understanding best settings for image acquisition
    5. **Pre-processing steps or image cleaning:** Imaging sets taken to prepare images for OCR

6. **Document analysis:** Understanding different methods of projecting document structures
7. **Feature extraction methods and Classification:** Understanding the different methods of extracting meaning from character images.
8. **Recognition and post processing:** Types of recognition steps and models and improving results accuracy.

- **Neural Network research**

1. **What are neural networks:** Understanding what neural networks are at a basic level
2. **Pros and cons of neural networks:** Applications of neural networks.
3. **Topology of neural networks:** The different structures neural networks take on
4. **Input layer:** What is the input layer, it's importance and role
5. **Output layer:** What is the output layer, it's importance and role
6. **Hidden layers:** What is the hidden layer, it's importance and role and structure
7. **Role of the activation function:** What the activation functions do and the maths behind each function. And when to use them.
8. **Training a neural network:** How neural networks learn. The methods that they lean
9. **Normalising data sets for neural networks:** Why we need to normalise the data passing into and out of neural networks.
10. **Maths behind neural networks:** More understanding of the maths behind NN, to try to negate the idea of the internals being a black (magic) box.

- **Image Processing Research**

1. **Machine vision:** What is machine vision
2. **Grey scale imaging:** What and how to convert image to grey scale
3. **Thresholding/ bi-level images/ image binarization:** Converting image pixel values to two values only. ON and OFF
    - **Global Thresholding**
    - **Local Thresholding**
4. **Image Contrast:** Understanding how contrasting works
5. **Edge detection:** What is edge detection and how it works
6. **Hough line transformation:** Trying to understand how Hough line algorithm works. What it does where it can be used.
7. **Image transforming:** How to rotate, resize and manipulate image skew.

8. **Histogram equalisation:** How histogram equalisation works. Where it can be applied.
9. **Masking and filtering with correlation methods:** Understanding filters and correlation.
10. **Masking and filtering with convolution:** Understanding filters with Convolution.
11. **Image noise reduction:** Removing noise from images.
12. **Image segmentation (Global and Local):** how to create sub-matrix of images

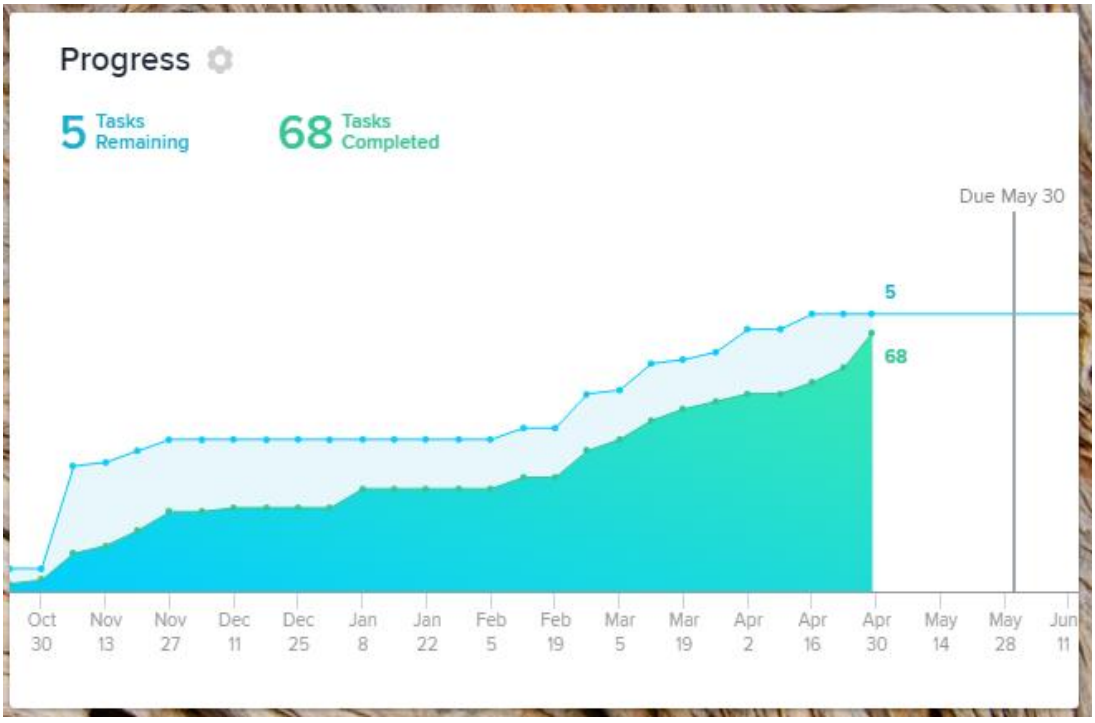
The following is a list of tasks that where to track progress of implementations of the OCR Software. The time stamps of completion can be viewed on Asana.

- **Other pre-processing functions.**
    1. **Document analysis**
    2. **Zoning**
    3. **Character isolation**
- **Technologies and dependencies.**
    1. **Java SE**
    2. **OpenCV**
    3. **Neuroph**
    4. **Encog**
    5. **Scanner recommendations**
- **Software design and Analysis**
    1. **Model, View, Controller**
    2. **System schematics**
    3. **GUI Wireframe design**
    4. **Class Diagrams**
    5. **Flow Diagrams**
    - **Development and Code Implementation.**
    1. **GitHub repository**
    2. **GUI interface**
    3. **GUI dialog box interfaces**
    4. **Controller for basic operations**
    5. **Imaging Tools**

6. **Imaging Utilities class**
7. **Displaying images**
8. **Dealing with unknown images**

- **Neural Network Implementation.**
  1. **Creating data sets**
  2. **Serializing Training sets**
  3. **Exporting to .CSV**
  4. **Create Self Organising Map Network**
  5. **Finalizes SOM**
  6. **Create feedforward neural network**
- **Documentation**
  1. **Table of contents**
  2. **First Draft**
  3. **Revised Draft**
  4. **Final Draft**

The following graph shows the time line and completion of task throughout the running of the project.



APPENDIX C: CODE LISTINGS

The following section reviews some of the code implemented. The full source code can be found on GitHub by following the link to the repository.

https://github.com/KyleT082/OCR_GUIDesign/tree/master/src/kgt/dev/ocr_gui

The following code listing is the implementation of the document analysis class. In this class, the pixels of the image are counted along the Y-Axis of the image and saved in an array. This array is later used to find the start and end bounds of text lines found within the document body.

```
public class DocumentAnalysis {

	private Mat mat,dst;
	private String docName;
	private double[] histData;
	private List<int[]> lineBins;

	/**
	 * CONSTRUCTOR
	 *
	 * @param newMat
	 */
	public DocumentAnalysis(Mat newMat, String name){
		this.mat = newMat;
		this.docName = name;
	}

	 /**
	 * Calculate the pixel density along the y axis
	 */
	public void calculateHist(){

		histData = new double[this.mat.rows()];
		int count;
		int total;
		Mat colorCon = new Mat();
		dst = new Mat();

		Imgproc.cvtColor(this.mat, colorCon,Imgproc.COLOR_RGB2GRAY);
		Imgproc.threshold(colorCon,
				dst,123,255,Imgproc.THRESH_BINARY);

		for(int x = 0; x < dst.rows();x++){
			count = 0;
			for(int y = 0; y < dst.cols();y++){

				double[] index = dst.get(x, y);
				if(index[0] == 0.0){
					count++;
				}
			}
			histData[x] = count;
		}
	}
```

```
/**
   * Calculate the start and end point of each word line along the Y-Axis
 */
public void calcLineProjection(){

	lineBins = new ArrayList<int[]>();

	double last = 0.0;
	int x,y;

	//Finds the starting point of a text line along the Y-Axis
	for(x= 0; x < histData.length; x++){
		int[] binDim = new int[4];

		if(histData[x] > 0 && last == 0.0){
			binDim[1] = x;
			lineBins.add(binDim);
		}
		last = histData[x];
	}

	//finds the end point of the text line along it's Y_Axis
	int step = 0;
	for(y = 0; y < histData.length;y++){

		if(histData[y] == 0 && last > 0.0){
			lineBins.get(y)[3] = y + 2;
			step++;
		}
		last = histData[y];
	}
	//Finds end point and the start point along the X_Axis
	for(int q = 0; q < lineBins.size(); q++){
		int[] bins = lineBins.get(q);

		int temp;
		int start_X = this.dst.cols();
		int end_X = 0;

		for(y = bins[1]; y < bins[3];y++){
			for(x = 0; x < this.dst.cols(); x++){

				double[] index = this.dst.get(y, x);
				if(index[0] == 0.0){
					temp = x;
					if(temp < start_X){
						start_X = temp;
						lineBins.get(q)[0] = start_X;
					}
				}
			}
		}
		//ending point
		for(y = bins[1]; y < bins[3]; y++){
			for(x = this.dst.cols() - 1; x > 0; x--){

				double[] index = this.dst.get(y,x);
				if(index[0] == 0.0){
					temp = x;
					if(temp > end_X){
						end_X = temp;
						lineBins.get(q)[2] = end_X;
					}
				}
			}
		}
	}
}
```

The Following code snippet is the Self Organising Map Neural network. The SOM can be created and the parameters added atomically depending on the type and size of the training set added to the neural network. Or the parameters can be manually applied if the structure and parameters of the training set are already known. This class can be found at the following package directory *kgt.dev.ocr_gui.neuralnet.*

```
public class SOM_Net extends NeuralNets{

	private static final long serialVersionUID = 7966537469301647951L;

	private SOM network;

	private String message;

	/**
	 * CONSTRUCTOR
	 *
	 * Used for automated creation of the SOM network
	 * based on the training set
	 *
	 * @param newTs - training set to add to the self organizing map
	 */
	public SOM_Net(TrainingSet newTs){
		super(newTs);
		this.setTrained(false);
	}

	/**
	 * CONSTRUCTOR
	 *
	 * Used for manual creation of the SOM
	 *
	 * @param sample_width
	 * @param sample_height
	 */
	public SOM_Net(){
		this.setTrained(false);
	}

	/**
	 * Set the sample width and height to establish input and output
	 * neuron count.
	 */
	public void setSampleDim(){

		if(this.trainSet.getTrainingSet().isEmpty()){
			throw new IndexOutOfBoundsException();
		}else{
			this.SET_SIZE = this.trainSet.getTrainingSet().size();
			this.SAMPLE_WIDTH = this.trainSet.getTrainingSet().get(0).getWidth();
			this.SAMPLE_HEIGHT =
			this.trainSet.getTrainingSet().get(0).getHeight();
		}
	}
```

This code shows the training method used by the SOM network. The Network is created within the training method to ensure that the Neural Network is trained before it can be use. The number of input neurons is equal to the width x height of the images sized used.

```
@Override
public void train(){

            try{
                  int inputNeurons = this.SAMPLE_WIDTH *
                  this.SAMPLE_HEIGHT;
                  int outputNeurons = this.SET_SIZE;

                  final MLDataSet dataSet = new BasicMLDataSet();

                  for(int e = 0; e < this.SET_SIZE;e++){

                        final MLData data = new BasicMLData(inputNeurons);

                        int index =0;
                        final SampleData ds =
                        this.trainSet.getTrainingSet().get(e);

                        for(int x = 0; x < ds.getWidth();x++){
                              for(int y = 0; y < ds.getHeight();y++){

                                    if(ds.getBiPolarData(x,y)){
                                          data.setData(index++, 1.0);
                                    }else{
                                          data.setData(index++, -1.0);
                                    }
                              }
                        }
                        dataSet.add(new BasicMLDataPair(data,null));
                  }

                  this.network = new SOM(inputNeurons,outputNeurons);
                  this.network.reset();

                  SOMClusterCopyTraining train = new
                         SOMClusterCopyTraining(this.network,dataSet);
                  train.iteration();
                  this.setTrained(true);
            }catch(Exception e){
                  setMessage("Error Training the Self Orginising Map");
            }
      }
```

This *mapNeuron* class creates an array used to associate the output neuron number to its relevant character it is learnt to recognise.

```
/**
  * Associates neuron to a character
  *
  * @return - neuron to char map
  */
 public char[] mapNeurons(){
       final char neuronMap[] = new char[SET_SIZE];

       for(int n = 0;n< neuronMap.length;n++){
             neuronMap[n] = '*';
       }

       for(int i =0; i < SET_SIZE; i++){
             MLData DataIn = new BasicMLData(SAMPLE_WIDTH * SAMPLE_HEIGHT);

             int index = 0;
             SampleData ds = this.trainSet.getTrainingSet().get(i);

             for(int x = 0; x < ds.getWidth(); x++){
                   for(int y = 0; y < ds.getHeight(); y++){

                         if(ds.getBiPolarData(x, y)){
                               DataIn.setData(index++, 1.0);
                         }else{
                               DataIn.setData(index++, -1.0);
                         }
                   }
                   int winner = this.network.classify(DataIn);
                   neuronMap[winner] = ds.getSymbol();
             }
       }
       return neuronMap;
 }
```

The following code is the recognition method. The acquired character is feed into the SOM and the winning neuron fires. The Neuron that fires is then used to refer to the neuron map to determine what to output.

```
/**
  * Check which neuron is fired to determine the closest match.
  *
  * @param test - the character in question
  * @return - the winning character
  */
 @Override
 public char recognise(SampleData test){
       char best = ' ';
       try{
             MLData input = new BasicMLData(SAMPLE_WIDTH * SAMPLE_HEIGHT);
             int index = 0;
             SampleData ds = test;

             for(int x = 0; x < ds.getWidth(); x++){
                   for(int y = 0; y < ds.getHeight(); y++){

                         if(ds.getBiPolarData(x, y)){
                               input.setData(index++, 1.0);
                         }else{
                               input.setData(index++, -1.0);
                         }
                   }
             }

             int winner = this.network.classify(input);
             char map[] = mapNeurons();

             best = map[winner];

       }catch(NullPointerException non){
             this.setMessage("The network needs to be trained first!");
       }
       return best;
 }
```

```
/**
  * Adds a training set for manual creation of SOM
  *
  * @param newTs - the SampleData set to train with
  */
 public void addTrainingSet(TrainingSet newTs){

       this.trainSet = newTs;
       if(newTs.getTrainingSet().isEmpty()){
             this.setMessage("Error, there is no data in the training
             set!");
             throw new NullPointerException();
       }else{
             this.setTrained(false);
             this.SET_SIZE = newTs.getTrainingSet().size();
       }
 }
```

REFERENCES

- P. Vithlani, "Pre-processing Techniques in Character Recognition", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 11, 2014.
- N. VENKATA RAO, D. SASTRY, A. CHAKRAVARTHY and K. P, "OPTICAL CHARACTER RECOGNITION TECHNIQUE ALGORITHMS", *Journal of Theoretical and Applied Information Technology*, vol. 83, no. 2, 2016.
- V. Shrivastava and N. Sharma, "ARTIFICIAL NEURAL NETWORK BASED OPTICAL CHARACTER RECOGNITION", An International Journal (SIPIJ), vol. 3, no. 5, 2012.
- F. Shafaita, D. Keysersa and T. Breuelb, "Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images".
- S. Roy, A. Chatterje, R. Pandit and K. Goswami, "Printed Text Character Analysis Version-I: Optical Character Recognition with the new User Training Mechanism", International Journal of Advanced Computer Research, vol. -4-2, no. -15, 2014.
- R. Maini and D. Aggarwal, "Study and Comparison of Various Image Edge Detection Techniques".
- Khushbu and S. Mehta, "Image Pre-processing on Character Recognition using Neural Networks", International Journal of Computer Applications, vol. 82, no. 13, 2013.
- M. Kasthuri and V. Shanthi, "Noise Reduction and Pre-processing techniques in Handwritten Character Recognition using Neural Networks", International Journal of Computing Science and Communication Technologies, vol. 6, no. 2, 2014.
- T. Gupta, C. Ahuja and S. Aich, "Optical Character Recognition", International Journal of Emerging Technology and Advanced Engineering, vol. 4, no. 9, 2017.
- M. Gupta, N. Jacobson and E. Garcia, "OCR binarization and image pre-processing for searching historical documents", *The Journal of Pattern Recognition Society*, vol. 40, no. 389-397, 2005.
- D. GUPTA and L. NAIR, "IMPROVING OCR BY EFFECTIVE PRE-PROCESSING AND SEGMENTATION FOR DEVANAGIRI SCRIPT:A QUANTIFIED STUDY", *Journal of Theoretical and Applied Information Technology*, vol. 52, no. 2, 2013.
- A. El Harraj and N. aissouni, "OCR ACCURACY IMPROVEMENT ON DOCUMENT IMAGES THROUGH A NOVEL PRE-PROCESSING APPROACH", An International Journal (SIPIJ), vol. 6, no. 4, 2015.
- S. Chopra, A. Ghadge, O. Padwal, K. Punjabi and P. Gurjar, "Optical Character Recognition", International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, no. 1, 2014.
- "A Study on Preprocessing Techniques for the Character Recognition", International Journal of Open Information Technologies, vol. 2, no. 12, 2014.

- Y. Alginahi, "Preprocessing Techniques in Character Recognition", Taibah University, Kingdom of Saudi Arabia, 2017.
- L. De Russis, "Introduction to OpenCV", 2013.
- D. Baggio, OpenCV 3.0 Computer Vision with Java, 1st ed. Packt Publishing, 2015.
- D. Belavkin, "Lecture 11: Feed-Forward Neural Networks", 2017.
- W. Bieniecki, S. Grabowski and W. Rozenberg, "Image Preprocessing for Improving OCR Accuracy", MEMSTECH, 2007.
- L. Eikvil, "Optical Character Recognition", Norsk Regnesentral, 1993.