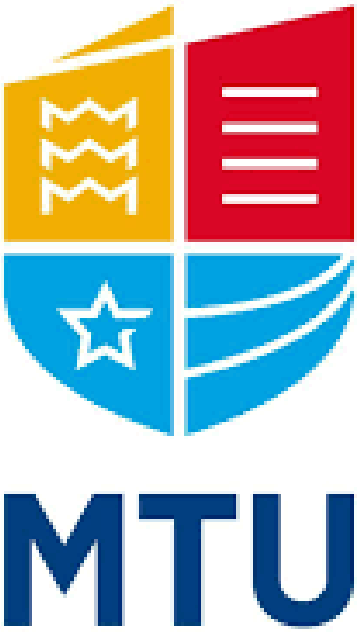


Enhancing Accuracy in Optical Character Recognition of Sensor Readings: A Comparative Study of Tesseract and CRNN Models with Emphasis on Image Preprocessing

by

Aidan Dennehy [R00145278]

For the module DATA9003 - Research Project as part of the
Master of Science in Data Science and Analytics, Department of Matematics

Supervisor: Dr Alex Vakaloudis

August 2023

Declaration of Authorship

I, Aidan Dennehy , declare that this thesis titled, "Enhancing Accuracy in Optical Character Recognition of Sensor Readings: A Comparative Study of Tesseract and CRNN Models with Emphasis on Image Preprocessing" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Munster Technological University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Munster Technological University

Abstract

Department of Matematics

Master of Science in Data Science and Analytics

by Aidan Dennehy [R00145278]

This research is primarily dedicated to the formulation of an innovative method for accurately interpreting sensor data obtained from digitized images. Confronting inherent challenges such as diminished contrast and subpar image quality, often associated with sensor readings, the study exploits Optical Character Recognition (OCR). This is accomplished employing two distinct techniques: Tesseract and Convolutional Recurrent Neural Network (CRNN) models.

An unique feature of the research lies in its novel image preprocessing steps, specifically the masking of red and green colors prior to conversion to grayscale. This process considerably augments the efficacy of OCR. Additionally, the study underlines the critical importance of correct font selection for each sensor to enhance reading accuracy.

The findings highlight the essential role of image quality and contrast in OCR, while presenting an innovative approach to image preprocessing for improved results. The potential implications of this research are extensive and could shape future undertakings in the fields of OCR and sensor digitization. The research underscores the vital aspects of image preprocessing and reveals how precise interventions can markedly improve sensor data interpretation from digitized images.

Acknowledgements

Acknowledgements here ...

Contents

List of Figures

Listings

List of Tables

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

For/Dedicated to/To my...

Chapter 1

Introduction

1.1 Area of Interest

The area of interest for this literature review is the intersection of computer vision, optical character recognition (OCR), and deep learning, with particular emphasis on the Tesseract OCR engine and Convolutional Recurrent Neural Networks (CRNNs). These technological advancements have revolutionized the way machines recognize and understand visual information, especially digits. Given their diverse and significant applications, ranging from digitizing written documents to aiding autonomous vehicle navigation, they hold vast potential for transforming many sectors. This research focuses on exploring the principles that underlie these tools, their performance in real-world applications, and the possibilities they offer for future development. This involves assessing the strengths of these systems, identifying their limitations, and suggesting potential areas of improvement. Moreover, it considers how these technologies are pushing the boundaries of OCR, paving the way for more sophisticated and versatile tools that can better navigate the complexities and variations in text size, font, and orientation often encountered in different visual scenes.

Optical Character Recognition (OCR) technology has seen substantial advancements in recent years, transforming the process of data extraction from visual mediums to digital formats. This technology, crucial in numerous fields ranging from document

TABLE 1.1: Nimbus Sensor Images

digitization to automated data entry systems. OCR holds specific importance when it comes to interpreting sensor readings, a key aspect of data-driven industries. The necessity for accurate, efficient, and automated reading of sensor-generated data has led to the investigation of various techniques and models within the OCR domain.

Two models which feature prominently emerged as potential solutions, namely Tesseract, an open-source OCR engine sponsored by Google, and Convolutional Recurrent Neural Network (CRNN), a combination of CNN, RNN, and Connectionist Temporal Classification that offers promising results in scene text recognition tasks.

In OCR applications, image pre-processing has a pivotal role. It prepares an image for further processing by reducing noise and unnecessary details and enhancing features that are important for later stages, thereby directly influencing the accuracy of the final output. Among various pre-processing techniques, the novel approach of red and green colour masking, followed by conversion to grayscale, has shown to significantly improve the accuracy of digit recognition.

While the core content of this thesis, including text and essential diagrams, adheres to the stipulated 60-page limit, the intrinsic visual nature of the research into Optical Character Recognition (OCR) methods necessitates the inclusion of numerous images, charts, and detailed reports. As such, the overall length of the document may extend beyond this limit. However, these additional pages are integral to fully understanding and appreciating the complexity and depth of the findings and analysis.

1.2 Motivation

The motivation behind this research stems from the challenges encountered in the manual and infrequent readings of environmental sensors in various operational settings such as factories. These sensors, while accurate and essential, lack a means for continuous data capture. Typically, an individual manually reads the sensor outputs at fixed intervals, which could range from hourly to daily. This method, while necessary, is prone to human error, potentially leading to inaccuracies in the recorded data and subsequent analysis reports. Furthermore, the infrequency of readings may result in delays in responding to critical sensor data, which could precipitate further issues. These complications could be mitigated with the implementation of Optical Character Recognition (OCR) technology. By enabling continuous, automated readings of these sensors, OCR has the potential to not only reduce errors but also ensure timely reaction to important sensor changes, optimizing the overall operation and efficiency of the systems.

1.3 Aims and Objectives

The primary aim of this research is to improve the efficiency and accuracy of Optical Character Recognition (OCR) on images of sensor readings by applying novel preprocessing steps and optimizing image capture settings. This project focuses on two OCR methods: Tesseract OCR and Convolutional Recurrent Neural Network (CRNN) models, both widely used for text recognition tasks.

1. **Objective 1: Systematic Literature Review of Optical Character Recognition (OCR) Methods**

   The objective here is to develop a comprehensive literature review of various Optical Character Recognition (OCR) methods. This review will explore the evolution, strengths, weaknesses, and applications of these techniques, as well as the advancements in this field, to provide a solid foundation for future OCR-related research and technology development.

2. **Objective 2: Data Capture**

   The aim of this objective is to capture a comprehensive and diverse dataset of images of sensor readings, in order to better understand and accurately reflect the multifaceted nature of the phenomena under investigation.

3. **Objective 2: Design and Implement Image Pre-processing Techniques**

   In an attempt to enhance the quality of the images and subsequently improve the OCR results, various image pre-processing methods will be introduced. A primary focus will be the implementation of colour masking (specifically for green and red) prior to the conversion to grayscale. This approach aims to make the images clearer and more conducive to OCR.

4. **Objective 3: Identify Optimal Image Capture Settings**

   In parallel with image pre-processing, the research will aim to identify the optimal parameters for image capture to further enhance OCR performance. The specific parameters of focus will include camera contrast, distance, and lighting.

5. **Objective 4: Compare and Evaluate the Effects of Pre-processing and Optimized Capture Settings on OCR Results**

   Once pre-processing measures and optimized image capture settings have been implemented, the images will undergo OCR using both Tesseract and CRNN models. This step aims to ascertain the joint impact of pre-processing and optimal capture parameters on the performance of OCR systems.

6. **Objective 5: Analyse and Report Findings**

   The final objective of the research is to analyse the findings and draw conclusions

on the effectiveness of the proposed pre-processing techniques and optimal capture parameters. This analysis aims to fill a gap in the literature, which currently lacks comprehensive studies on the potential benefits of image pre-processing and capture settings optimization for OCR of sensor readings.

In conclusion, this research seeks not only to enhance our understanding of how image pre-processing and capture optimization can improve OCR outcomes, but also to provide practical insights that could inform the future development of OCR systems.

1.4 Structure of the Thesis

This thesis is organized into five main chapters and appendices, each covering a specific aspect of the study:

1. **Chapter 1: Introduction**

   This chapter provides an overview of the research, outlining the area of interest and motivation behind the study. It also presents the aims and objectives that guide the research.

2. **Chapter 2: Literature Review**

   This chapter reviews previous research relevant to this study. It begins with a general introduction to the field, followed by specific sections on Tesseract OCR, CRNN OCR, and other OCR systems, examining their strengths, weaknesses, and applications.

3. **Chapter 3: Methodology**

   This chapter presents the research methodology, including the design and implementation of image pre-processing techniques and the methods used to identify optimal image capture settings. It also details how the Tesseract and CRNN OCR systems are applied in this research.

4. **Chapter 4: Results**

   This chapter presents the findings of the study. It includes an analysis of the OCR performance before and after the application of the pre-processing methods and optimized image capture settings.

5. **Chapter 5: Discussion and Conclusion**

   This final chapter discusses the implications of the research findings, drawing conclusions about the effectiveness of the proposed techniques for improving OCR performance. It also highlights potential areas for future research.

6. **Appendices**

   The appendices include additional information that is relevant to the research but

not essential to the main body of the thesis. This includes the full results of the OCR tests, and the full dataset of images used in the study.

Chapter 2

Literature Review

2.1 Introduction

As we stand on the precipice of a future moulded by artificial intelligence and machine learning, one domain that is experiencing considerable progress is Optical Character Recognition (OCR). In this dynamic and continuously evolving field, there are many techniques which have emerged among the significant game-changers, two of these are the Tesseract OCR engine and Convolutional Recurrent Neural Networks (CRNNs). Tesseract, initially developed by Hewlett-Packard and later adopted by Google, is a pioneering engine that converts images of text into machine-encoded text, offering utilities across numerous applications. On the other hand, CRNNs, a deep learning-based approach, combine the spatial feature extraction capabilities of Convolutional Neural Networks (CNNs) with the sequential data processing capacity of Recurrent Neural Networks (RNNs). These networks have set new benchmarks in the realm of scene text recognition, overcoming the challenges posed by variations in text sizes, fonts, and orientations. This literature review delves into the intricacies of these advanced tools, shedding light on their principles, applications, strengths, and potential areas for improvement, thereby enriching our understanding of current trends in OCR technology and pointing to the future possibilities.

This literature review explores the current state of OCR technologies, with a particular focus on Tesseract and CRNN models. It delves into various image pre-processing techniques, emphasizing the unique method of red and green colour masking before conversion to grayscale. Lastly, it investigates the role of font selection in enhancing OCR accuracy, thereby setting the context for the subsequent research.

While this review focuses on the capabilities of Tesseract OCR and Convolutional Recurrent Neural Networks (CRNNs) in the OCR domain, it's important to acknowledge that the OCR landscape is not limited to these technologies. Many other methods play equally significant roles in expanding the OCR frontiers and opening up new avenues for research and application. Long Short-Term Memory Networks (LSTMs), Transformers, attention-based OCR models, rule-based systems, Support Vector Machines (SVMs), Hidden Markov Models (HMMs), K-Nearest Neighbours (KNN), and template matching are some of these diverse methodologies that provide unique perspectives and solutions in the OCR realm. Each of these methods has its distinctive advantages, making them optimal for certain types of tasks, as well as its limitations, requiring continuous research and development for enhancement. However, the scope of this review will mainly revolve around Tesseract and CRNNs, while the mentioned methods provide an essential context for understanding the broader OCR ecosystem.

2.2 Tesseract OCR

Optical character recognition (OCR) is the process of converting images of text into machine-readable text. Tesseract is an open-source Optical Character Recognition (OCR) engine that is widely used for a variety of tasks, including document digitization, machine translation, and data entry. Tesseract was developed at HP between 1984 and 1994. Initially conceived as a PhD research project to improve OCR performance for HP's scanners, it outperformed contemporary commercial OCR engines but never became an HP product. Its development was mainly focused on improving rejection efficiency rather than base-level accuracy. Despite being shelved in 1994, Tesseract proved its prowess in the 1995 UNLV Annual Test of OCR Accuracy. HP made Tesseract open-source in 2005, hosted at Google Code. Tesseract's architecture assumes binary images as input and uses a step-by-step pipeline for processing, including a unique connected component analysis for detecting inverse text. Its recognition process is two-pass: initial recognition of words feeds an adaptive classifier which subsequently improves text recognition further down the page. A final phase resolves fuzzy spaces and checks alternative hypotheses for the x-height to locate small-cap text. [1]

FIGURE 2.1: Tesseract OCR
[2]

In this literature review, we will discuss five research papers that have been published on Tesseract OCR. These papers cover a wide range of topics, including the accuracy of Tesseract, the performance of Tesseract on different types of documents, and the use of Tesseract for specific applications. Each study presents unique findings, and together

they create a comprehensive overview of the current state and potential trajectory of Tesseract OCR. Through a detailed exploration of these works, we aim not only to comprehend the nuances of Tesseract OCR but also to contribute to the burgeoning discourse around its implications and opportunities in our increasingly digitized world

In the paper "Benchmarking Object Detection Algorithms for Optical Character Recognition of Odometer Mileage" Andersson et al. compare two state-of-the-art object detection models, Faster R-CNN and YOLO, with an open-source OCR model, Tesseract, for reading the odometer mileage from car dashboard images. They use a dataset of 2,389 images of car odometers and evaluate the models based on mean average precision, prediction accuracy, and Levenshtein distance.

FIGURE 2.2: Visualization of Boundary box predictions, and digit classification from YOLO-models. left - YOLO Singlebox-model. middle - YOLO Digitbox-model. right - YOLO Multibox-model.
[3]

They find that the object detection models outperform Tesseract on all metrics, and that Faster R-CNN has the highest mAP and accuracy, while YOLO has the lowest Levenshtein distance. [3]

Ahuja et al.'s paper "Detecting Vehicle Type and License Plate Number of different Vehicles on Images" discusses the development of a model that can locate a particular vehicle that the user is looking for depending on two factors 1. the Type of vehicle and the 2. License plate number of the car. The proposed system uses a unique mixture consisting of Mask R-CNN model for vehicle type detection, WpodNet and Tesseract for License Plate detection and Prediction of letters in it. [4]

The first stage of Ahuja et al.'s project involved annotating 2,650 images with a custom dataset using the open-source tool VGG Annotator. The images were annotated with

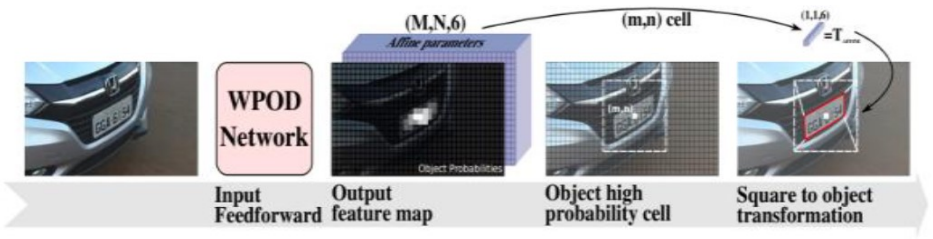


FIGURE 2.3: WpodNet Detection Method
[4]

rectangular shapes instead of polygons, and the categories were 2-Wheelers, 3-Wheelers, 4-Wheelers, and vehicles with more than four wheels. The trained Mask R-CNN model achieved an F1 score of 0.72399 on the training data and 0.68 on the test data.

The second stage was split into two parts: license plate detection and letter classification. The WpodNet model was used for license plate detection, achieving an accuracy rate of over 90%. The Tesseract model was then used to predict the letters on the license plate, enhancing speed and accuracy.

The third and final stage of the project involved the model accepting user input for vehicle type, license plate, and image. The model then compared this input with its output to determine if the searched vehicle had been identified.

The final hybrid model uses Mask R-CNN (F1 score: 0.72399 training, 0.68 testing), WpodNet, and pytesseract for car and license plate identification. This can aid in tracking stolen vehicles and assessing parking lot availability.

In an interesting paper, Giridhar et al.'s "A Novel Approach to OCR using Image Recognition based Classification for Ancient Tamil Inscriptions in Temples" describes a novel approach to OCR using image recognition based classification for ancient Tamil inscriptions in temples. The proposed work focuses on improving optical character recognition techniques for ancient Tamil script which was in use between the 7th and 12th centuries. A data set has been curated using cropped images of characters found on certain temple inscriptions, specific to this time period as a case study.

After using Otsu thresholding method for binarization of the image, a two-dimensional convolution neural network is defined and used to train, classify, and recognize the

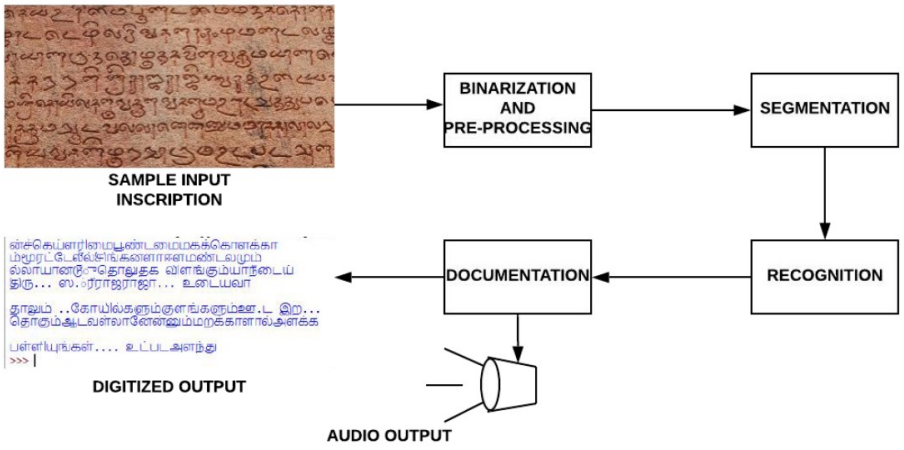


FIGURE 2.4: Giridhar's proposed architecture [5]

ancient Tamil characters. To implement the optical character recognition techniques, the neural network is linked to the Tesseract using the Pytesseract library in Python. As an added feature, this work also incorporates Google's text to speech voice engine to produce an audio output of the digitized text. Various samples for both modern and ancient Tamil were collected and passed through the system.

The author used Otsu Thresholding Method for binarization of the image and a Two-dimensional Convolution Neural Network to train, classify, and recognize the ancient Tamil characters. To implement the optical character recognition techniques, the neural network is linked to the Tesseract using the Pytesseract library in Python. As an added feature, this work also incorporates Google's text to speech voice engine to produce an audio output of the digitized text.

On average, the combined efficiency of the system for ancient Tamil script was 77.7%, although it varied based on the specific challenges encountered with different types of text and inscriptions. The study acknowledged that further work is needed to overcome these issues and increase the system's accuracy and efficiency.[5]

In Cakic et al's paper "The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing", they discuss the use of computer vision and optical character recognition (OCR) on mobile devices to read serial numbers from wine labels in order to enable

applications based on tracking and tracing of each individual wine bottle. The research focuses on the implementation and image processing that improved detection accuracy.

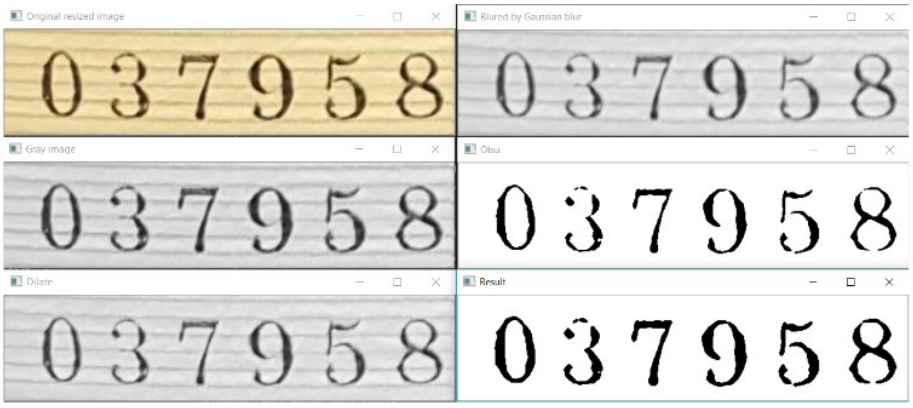


FIGURE 2.5: Numbers change through the pre-processing stage [6]

Cakic et al.'s research involved testing a script to read serial numbers from about 150 wine label images, some of which were of poor quality. The initial attempt, which didn't involve any image pre-processing, managed to correctly read the full serial numbers on around 62% of the images. After pre-processing the images, the success rate increased significantly to 87.5%. This suggests that the quality of the pre-processing and the images used in the test dataset significantly affects the accuracy of the results.

In 2019, Robby et al.'s paper "Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application" discusses the challenges and methods of recognising non-Latin scripts, especially Javanese, which have complex shapes and structures. They present their dataset collection, training, and testing process using Tesseract OCR tools. They report their results and analysis, showing that their model achieved the highest accuracy by combining single boundary box and separate boundary boxes for different parts of the characters.

FIGURE 2.6: Robby et al.'s Proposed Method [7]

The authors of this paper used a variety of methods to develop a Javanese character recognition system. First, they collected 5880 Javanese characters from various sources

and cropped and resized them to 32x32 pixels. Then, they applied several pre-processing steps to the images, such as binarization, noise removal, contrast enhancement, and skew correction. Next, they augmented the data by applying random rotations, translations, scaling, and shearing to the images. Finally, they used Tesseract, an open source OCR tool, to train different models with different boundary box methods.

The author's found that the best model was the one that combined single boundary box and separate boundary boxes for different parts of the characters, which achieved an accuracy of 97.50% and an error rate of 2.50% on the test set.

2.3 Convolutional Recurrent Neural Networks (CRNNs)

CRNN, an abbreviation for Convolutional Recurrent Neural Network, is a unique fusion of the advantages of convolutional neural networks (CNN) and recurrent neural networks (RNN), which are different kinds of neural network architectures.

CRNNs are usually applied in the classification and analysis of sequential data like text, speech, and images. Due to their ability to handle variable-length sequential data and recognize long-term dependencies, they are extremely useful for tasks that need to comprehend contextual and temporal information. They have displayed excellence in modelling and processing sequential data across diverse tasks, marking them as an efficient instrument in this domain.[8]

2.3.1 Working of CRNNs

The functionality of CRNNs is outlined as follows:

1. **Input:** The primary input to a CRNN is a sequence of data, which could be images or audio samples.
2. **Convolutional Layers:** The incoming sequence is channelled through convolutional layers, akin to those in CNNs. These layers extract features from the input and are particularly efficient for image-based inputs.
3. **Recurrent Layers:** The output from a convolutional layer is then sent through one or more recurrent layers. These layers preserve a hidden state that memorizes information from previous entries in the sequence, making them ideal for sequential data processing.
4. **Bridge between Convolutional and Recurrent Layers:** Usually, the output from a convolutional layer is sampled before it is introduced to a recurrent layer. This strategy helps to minimize the network's computational complexity while maintaining the core characteristics of the input.

5. **Output:** Finally, the output from the last recurrent layer is processed through a fully connected layer. This final layer produces a prediction for the input sequence, which could be a string of characters, words, or any other output related to the task.

In the article "Synthetized Multilanguage OCR Using CRNN and SVTR Models for Realtime Collaborative Tools" Biro et al. present a novel hybrid language vocabulary creation method that is utilized in the OCR training process in combination with convolutional recurrent neural networks (CRNNs) and a single visual model for scene text recognition within the patch-wise image tokenization framework (SVTR). The research used a dedicated Python-based data generator built on dedicated collaborative tool-based templates to cover and simulate the real-life variances of remote diagnosis and co-working collaborative sessions with high accuracy. The machine learning models recognized the multilanguage/hybrid texts with high accuracy.

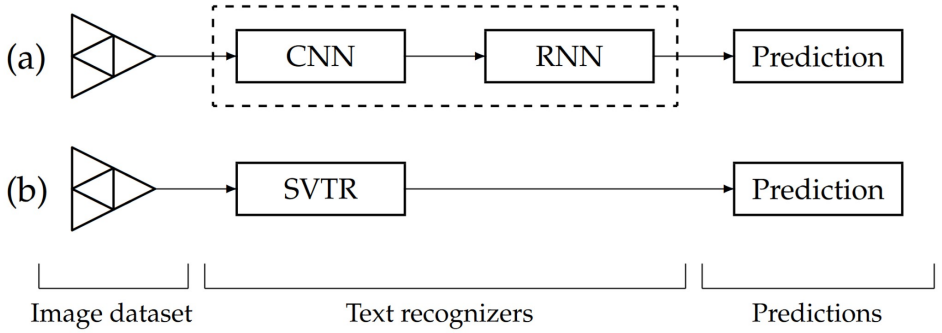


FIGURE 2.7: Biro et al.'s Modern text recognizers using (a) CRNN and (b) SVTR models
[9]

The highest precision scores achieved are 90.25%, 91.35%, and 93.89%. The study examines the feasibility of machine learning-supported OCR in a multilingual environment. The novelty of our method is that it provides a solution not only for different speaking languages but also for a mixture of technological languages, using artificially created vocabulary and a custom training data generation approach. The machine learning models for special multilingual, including languages with artificially made vocabulary, perform consistently with high accuracy. [9]

Shinde et al.'s paper "Using CRNN to Perform OCR over Forms" presents a structured process of locating input fields on the form, scanning the input data, processing the data and entering the data to the final database. The methods used in this system is a general method of performing OCR on human handwriting. The form filled by the user is to be scanned and sent as an input to the system. The model will then detect all the user input areas on the form as the main goal is to extract the user-entered information only. The system architecture consists of a Word Segmentation algorithm followed by a neural network architecture to perform optical character recognition on the words after segmenting them from the sentences. The convolutional layers are used to extract feature sequences from the input images. This output is passed on to the recurrent layers for making predictions for each frame of the feature sequence. Finally, the transcription layer or the CTC is used to translate the per-frame predictions by the recurrent layers into a label sequence.

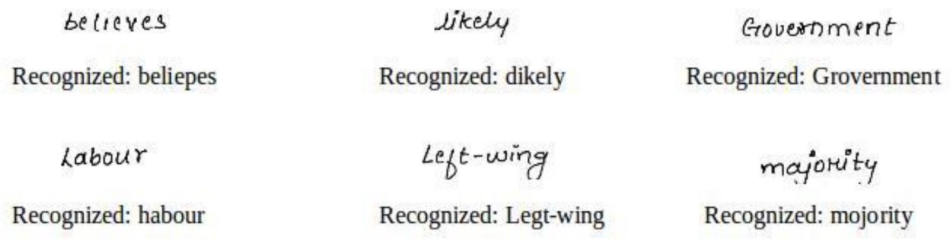


FIGURE 2.8: Shinde et al.'s Handwriting Results
[10]

The study found that handwriting styles that matched those of the IAM dataset (which feature larger spaces between words and smaller spaces within words) were correctly segmented and recognized. However, when users wrote words closely together (congested handwriting), the system had trouble distinguishing between words, as the spacing between words was similar to the spacing between letters.

The increased inter-word spacing improved the accuracy of word segmentation and recognition. After testing the system with 100 random words, it achieved an accuracy of 72.22%. Some errors were noted. For instance, the system mistook the letter "l" for "d" and "a" for "o". This suggests the system struggles with closely resembling words and varied handwriting styles.

The document proposes solutions to improve system accuracy, such as training the model on a wider variety of handwriting styles beyond those in the IAM dataset. Also, it suggests increasing the number of words in the training set relevant to the problem statement, like station names or sequences of numbers similar to mobile numbers, to enhance the probability of correct recognition.[10]

In 2018, Rawl et al.'s paper "How To Efficiently Increase Resolution in Neural OCR Models" discusses how modern CRNN OCR models require a fixed line height for all images. Increasing this input resolution improves recognition performance up to a point. However, doing so by simply increasing the line height of input images without changing the CRNN architecture has a large cost in memory and computation. The authors introduce a few very small convolutional and max pooling layers to a CRNN model to rapidly down sample high resolution images to a more manageable resolution before passing off to the "base" CRNN model. Doing this greatly improves recognition performance with a very modest increase in computation and memory requirements.

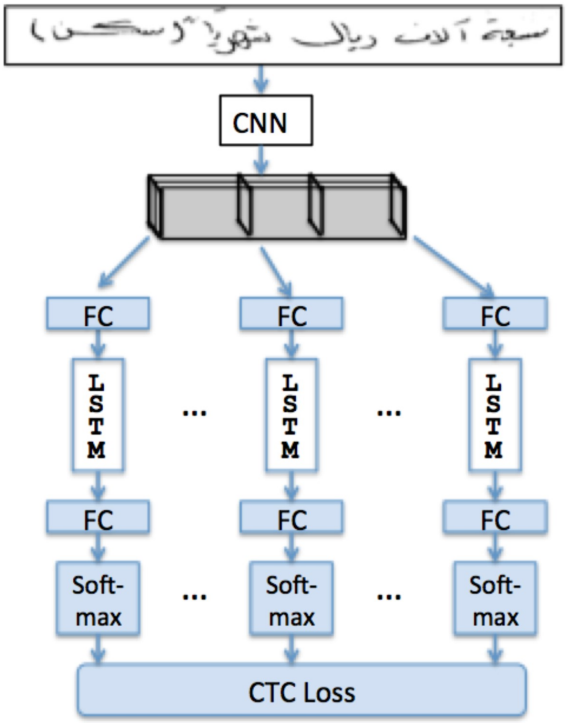


FIGURE 2.9: Rawl et al.'s End-to-End OCR Model: With CNN and LSTM layers [11]

They show a 33% relative improvement in WER, from 8.8% to 5.9% when increasing the

input resolution from 30px line height to 240px line height on OpenHART/MADCAT Arabic handwriting data. The authors report new state-of-the-art results for both Arabic handwriting recognition and English handwriting recognition. They do this by increasing the resolution of input images from a line height of 30px to 240px, an 8-fold increase.[11]

In Wuhan, Feng et al.'s paper "Port Container Number Recognition System Based on Improved YOLO and CRNN Algorithm" discusses the deep learning-based container number recognition system. The system is composed of two parts: object detection and character recognition. The system is designed to recognize container numbers from images captured in real-world scenarios.

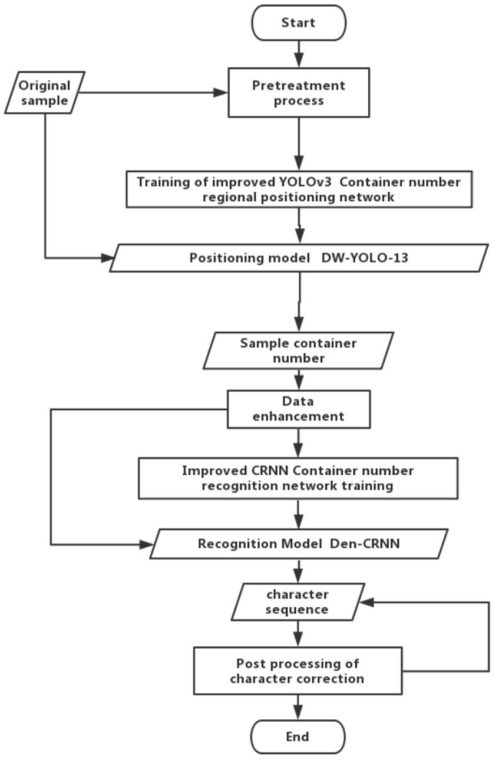


FIGURE 2.10: Feng et al.'s Algorithm Structure [12]

The object detection part uses YOLOv3 to detect the container number region and the character recognition part uses CRNN to recognize the characters in the region. The system was tested on a dataset of 2000 images and achieved an accuracy of 98.5%.[12]

In the paper "MultiPath ViT OCR: A Lightweight Visual Transformer-based License Plate Optical Character Recognition" by Azadbakht et al. present a new approach to

OCR of license plates using a lightweight model based on Visual Transformer architecture. The proposed model is lightweight and can be used on edge devices with limited computation power.

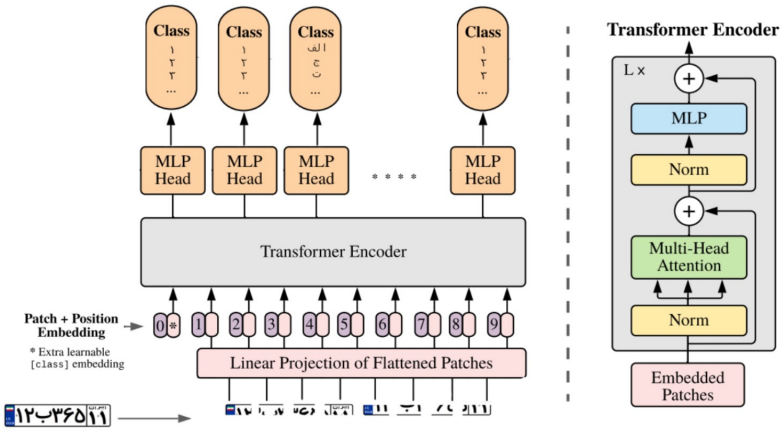


FIGURE 2.11: Azadbakht et al.'s MultiPath Visual Transformer OCR architecture [13]

It achieves 77.25% accuracy against CNN models with 75.18% accuracy and embedded OCR models in cameras with 60.37% accuracy on the LicenseNet test set. The authors gathered and annotated 1.3M images of license plates in various natural conditions from different points of view and different cameras and call this dataset as LicenseNet. The proposed model has 3.21 times fewer training parameters than previously proposed CNN-based models and achieves better accuracy with fewer parameters. The paper also explains the implementation details and training hyperparameters and compares the model's performance against the previously employed models.[13]

Other OCR Methods

2.4 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Networks (LSTMs) are a special kind of recurrent neural network capable of learning long-term dependencies, which makes them highly suitable for OCR tasks. They've been used successfully to decode sequences of characters from images.[14]

Breuel et al. in the paper "High-Performance OCR for Printed English and Fraktur using LSTM Networks" write about a novel application of bidirectional Long Short-Term Memory (LSTM) networks to the problem of machine-printed Latin and Fraktur recognition, without segmentation, language modelling or post-processing.

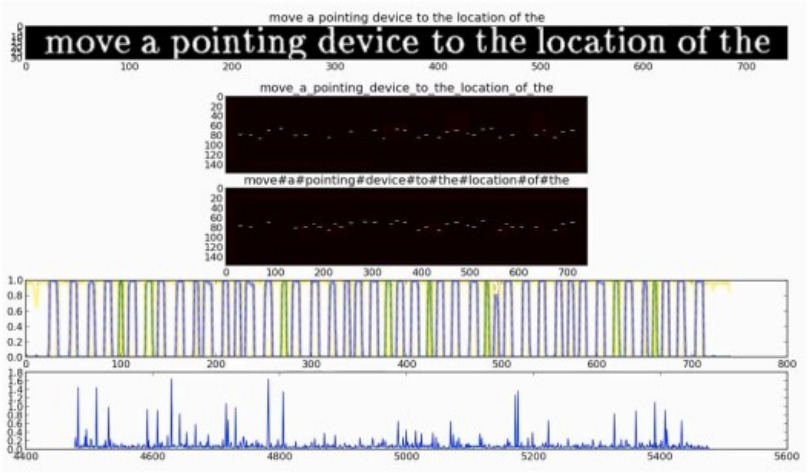


FIGURE 2.12: Greyscale image with background cleaning [14]

A pre-processing step for text-line normalisation that uses a dictionary of connected component shapes and associated baseline and x-height information to map the input text lines to a fixed size output image.

A comparison of the LSTM-based system with other OCR systems on printed English and Fraktur texts, showing that LSTM achieves very low error rates and generalizes well to unseen data.

2.5 Transformers

Originally developed for natural language processing tasks, Transformer models have been adapted for OCR. They treat the OCR problem as a sequence-to-sequence translation task, translating the input image into a sequence of characters.

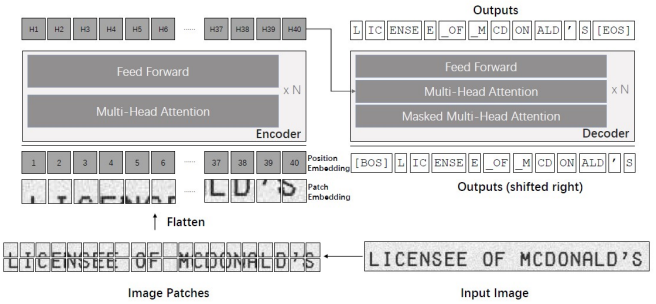


FIGURE 2.13: Li's Architecture of TrOCR
[2]

M.Li et al.'s "TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models" paper proposes an end-to-end text recognition approach with pre-trained image Transformer and text Transformer models, which leverages the Transformer architecture for both image understanding and word piece-level text generation. [2]

Transformer based OCR models have the advantage of being able to handle long sequences of text, which is useful for OCR tasks. However, they are computationally expensive and require large amounts of training data.

CRNNs are more suitable for this project because they are faster and require less training data and are better at handling spatial information

2.6 Attention-based OCR models

Attention mechanisms allow models to focus on different parts of the input image while predicting each character in the output sequence, similar to how humans read. This can improve accuracy, especially on more complex images.

Li et al.'s "Attention Based RNN Model for Document Image Quality Assessment" paper proposes a novel method for document image quality assessment (DIQA). The method integrates convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to capture spatial features and attention mechanisms. It also uses reinforcement learning to train a locator network that selects the optimal regions for quality evaluation.

The CNNs are used to extract spatial features from the document images. The RNNs are used to capture the temporal dependencies between the features. The locator network is used to select the optimal regions for quality evaluation. The regions are selected based on the attention mechanism, which identifies the most important regions in the document images. [15]

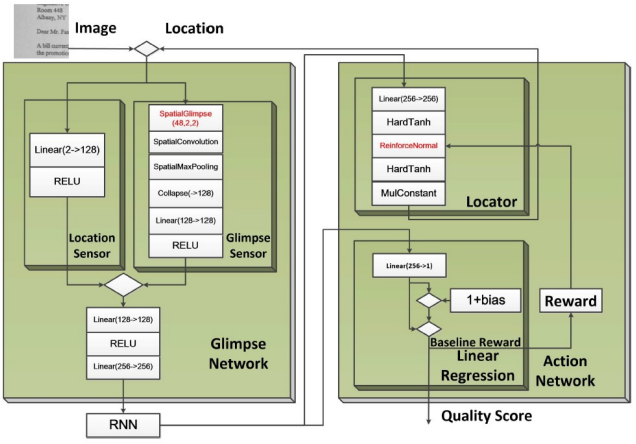


FIGURE 2.14: Architecture of Li's proposed network [15]

RNNs are good at handling sequential information but are poor at handling spatial information. CRNN's are more complex and combine the strengths of CNNs and RNNs which is more suitable for the this paper's OCR task.

2.7 Rule-based systems

These were some of the earliest methods for OCR and use specific rules for identifying characters based on their shape, size, and relative position. They are now less commonly used due to their limitations with complex and diverse inputs.

Doush et al.'s paper "A novel Arabic OCR post-processing using rule-based and word context techniques" developed a rule-based OCR system for Arabic text that uses a combination of horizontal and vertical projections to segment characters and then classifies them based on their shape and relative position. [16]

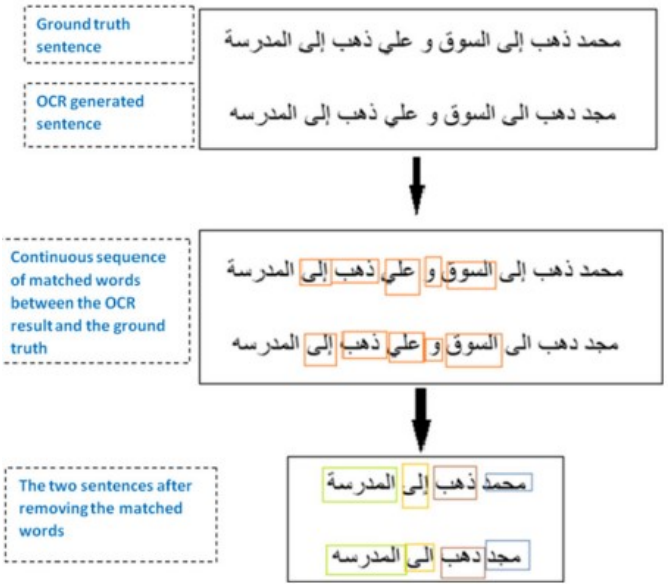


FIGURE 2.15: Example of applying the Rule Based FAHTA Algorithm [16]

The FAHTA algorithm is a novel alignment technique that is used to match the ground truth text with the OCR misrecognized text. The paper also says that the FAHTA algorithm is fast, accurate, and can handle different types of OCR errors, such as over-segmentation, under-segmentation, and merging words. The paper claims that the FAHTA algorithm can be used for other languages as well.

For the purposes of this project, the rule-based system is not suitable because it requires a large number of rules to be defined for each character, which is not feasible for the large range of digit fonts.

2.8 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are used for character recognition in OCR due to their effective high-dimensional mapping and classification abilities. They work best when text is clearly segmented. In the their paper "Development of an Image Processing Techniques for Vehicle Classification Using OCR and SVM", Joshua et al. used SVMs to classify characters in a license plate image and achieved an accuracy of 98.3% using a local dataset of 10,000 images.[17]

FIGURE 2.16: Greyscale image with background cleaning [17]

Joshua et al. describe the steps of their proposed system, which include image pre-processing, feature extraction, OCR, and SVM classification. They also explain how they collected and labelled their dataset of Nigerian vehicle plate numbers.

2.9 Hidden Markov Models (HMMs)

HMMs have been used in OCR for recognizing sequential data. HMMs are statistical models that assume an underlying process to be a Markov process with hidden states.

In Rashid et al.'s "An evaluation of HMM-based Techniques for the Recognition of Screen Rendered Text" paper, they evaluate Hidden Markov Model (HMM) techniques for optical character recognition (OCR) of low resolution text from screen images and compares them with other OCR systems.

The paper uses two data sets of screen rendered characters and text-lines, and extracts two types of features from them: grey scale raw pixel features and gradient based grey level intensity features.

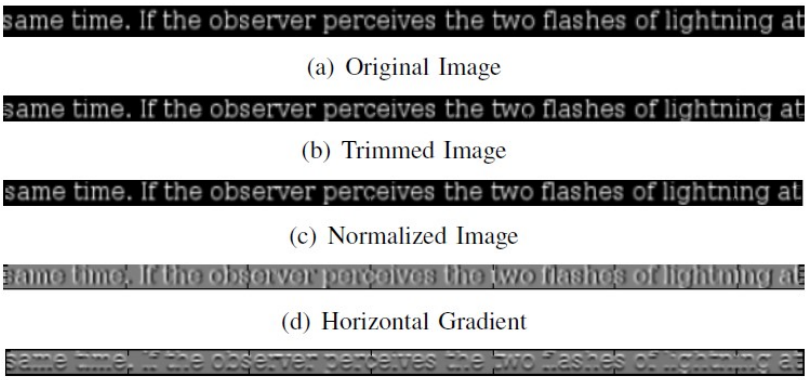


FIGURE 2.17: Rashid's Extraction steps from screen rendered text-lines [18]

The paper reports the character recognition accuracy of the HMM-based methods and other OCR engines on the two data sets. It shows that the HMM-based methods reach the performance of other methods on screen rendered text and achieve above 98% accuracy.[18]

HMMs are a good choice for tasks where simplicity and interpretability are important. CRNNs are a good choice for tasks where accuracy is more important, and where the sequences are long or complex.

2.10 K-Nearest Neighbours (KNN)

KNN is a simple, instance-based learning algorithm used for OCR, particularly for isolated character recognition. Hazra et al. develop an optical character recognition (OCR) system that uses a custom image to train a k-nearest neighbour (KNN) classifier. They claim that their system can recognize handwritten or printed text in any language by changing the training image and labels. [19]

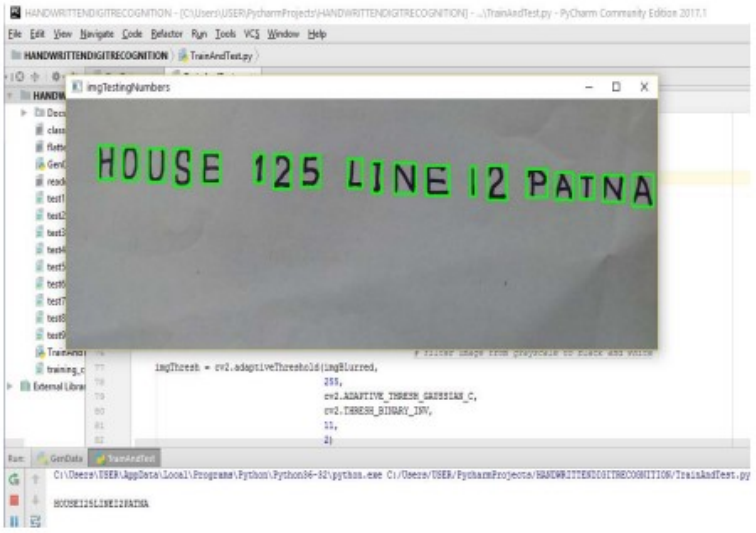


FIGURE 2.18: Characters and Digits recognised [17]

Hazra et al. explain the steps of their algorithm, which include image processing, feature extraction, and KNN classification. They also discuss the advantages of KNN over other classification methods, such as ease of interpretation, low computation time, and high predictive power. In this paper the authors started with clear images of known fonts, which is not the case in this project.

2.11 Template Matching

Template Matching is a technique used to locate small-parts of the bigger image which match a template image. This can be useful in OCR when the set of possible characters is known and limited. In Hossain et al.'s "Optical Character Recognition based on Template Matching" paper, they use template matching to recognize characters in a license plate image.

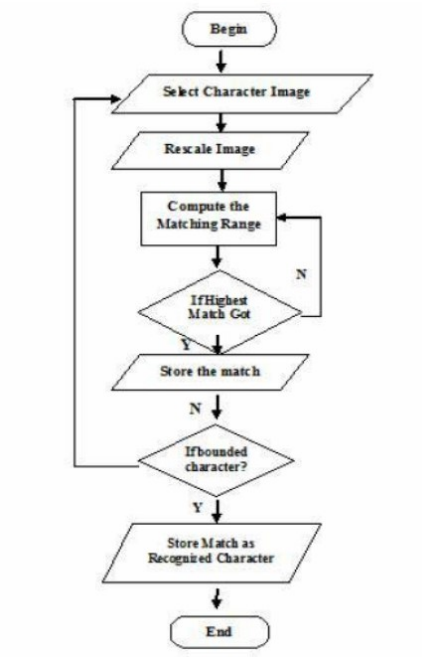


FIGURE 2.19: Flowchart of Hossain's TM OCR [20]

Their system prototype was tested on different text images with different fonts and sizes. The accuracy was calculated based on the character recognition accuracy. Their results show that Calibri and Verdana fonts had the highest accuracy, while Cambria and Times New Roman fonts had the lowest accuracy. The accuracy can be improved by training the system with more fonts and features. [20]

2.12 Conclusion

Tesseract is an open-source optical character recognition (OCR) engine with a wide range of capabilities. It has been used to digitize ancient Tamil inscriptions, identify vehicles and their license plates, and recognize serial numbers on wine labels.

In this chapter, we have explored the capabilities and applications of Tesseract. We have seen that Tesseract can be used as a stand-alone tool, but it can also be combined with other technologies to improve its performance. For example, Tesseract can be integrated with object detection models to improve its accuracy at reading odometer mileage from car dashboard images.

We have also seen that Tesseract can be used in a variety of domains. For example, it has been used to preserve historical documents, aid linguistic research, and locate specific vehicles.

Overall, the potential of Tesseract OCR is vast. Its performance can be improved through integration with other models or pre-processing steps, and its applications extend to a variety of domains. The challenge lies in further refining the OCR technology, continuing to expand its range of uses, and exploring its potential for even more innovative applications in our increasingly digitized world.

Convolutional recurrent neural networks (CRNNs) are a powerful tool for processing sequential data and extracting important features. They have been shown to be effective in a variety of tasks, including optical character recognition (OCR), natural language processing, and image recognition.

CRNNs are able to handle variable-length sequential data and recognize long-term dependencies. This makes them ideal for tasks that require understanding of contextual and temporal information. For example, CRNNs can be used to recognize text in images, even if the text is of different languages or handwriting styles.

There have been many recent advances in CRNN research. For example, Biro et al. developed a CRNN that can recognize multilingual and hybrid language texts in OCR.

Shinde et al. used CRNNs to recognize handwriting on forms, and Feng et al. used CRNNs to recognize container numbers.

These are just a few examples of the many applications of CRNNs. As research in this area continues, we can expect to see even more powerful and efficient CRNN models being developed. These models will have a wide range of applications, and will help us to better understand the world around us.

With the comprehensive exploration and evaluation of various Optical Character Recognition (OCR) methods presented in this section, we have successfully achieved our first objective: 'Systematic Literature Review of Optical Character Recognition (OCR) Methods'. This thorough review provides a solid foundation for our future work, offering clear insights into the advancements, strengths, and challenges in this field.

Chapter 3

Methodology

3.1 Introduction

This chapter provides a comprehensive and detailed explanation of the techniques and procedures adopted during the course of the research. It serves to provide an in-depth account of the methods used in this study, thereby ensuring the research's transparency and reproducibility.

This research aims to enhance the performance of Optical Character Recognition (OCR) systems - specifically Tesseract OCR and Convolutional Recurrent Neural Network (CRNN) models - on images of sensor readings. To accomplish this, a systematic approach is adopted involving an initial global run of the OCR systems on the raw image datasets, followed by the application of specific image pre-processing techniques and subsequent localized OCR applications.

The purpose of these processes is to establish a baseline of OCR performance, then test the hypothesis that image pre-processing can enhance OCR results. The pre-processing, focused on applying colour masks before converting the images to grayscale, aims to increase the clarity of the images, thereby increasing the efficiency of the OCR processes.

This chapter outlines each of these processes in detail, thereby providing a clear roadmap of the research methodology adopted in this study. From the initial assessment of the

OCR systems' performance to the implementation of the pre-processing techniques, this chapter serves as a guide to understanding the practical steps taken during this research project.

The subsequent sections provide further detail on the data being used, the OCR systems of focus, the pre-processing techniques applied, and the methods of evaluation. The goal of this chapter is to present a detailed and comprehensive account of the methodology that underpins this research.

—

3.2 Data Collection

The dataset used in this research was supplied as a collection of image files distributed across ten distinct folders. Each folder corresponds to a unique sensor from which readings were taken. These images provide a diversified dataset due to variations in sensor specifications and the conditions under which the readings were captured.

Upon receiving the data, an initial examination was carried out to ensure the integrity and completeness of the files. The image files were found to be in good condition, readable, and ready for further processing and analysis.

In order to streamline the data management and analysis process, a CSV file was created. This file serves as an inventory, containing each image file name and its corresponding label, thereby facilitating an efficient cross-referencing system for the data analysis phase.

To facilitate the training of the Convolutional Recurrent Neural Network (CRNN) models, multiple training databases were created. Each of these databases consists of 500,000 single digit training images, thereby providing a robust foundation for the machine learning tasks.

3.3 Data Analysis

For each folder, there are three charts that provide an initial statistical data analysis of the images. These charts are as follows:

1. **Montage:** A simple representation of the images in the folder, arranged in a grid format. This provides a visual overview of the images in the folder, thereby facilitating a quick assessment of the data.

2. **RGB Histogram:** This chart shows the distribution of pixel intensities for the Red, Green, and Blue channels separately in each image.

   (a) *Axes:* The X-axis represents the possible pixel intensity values (ranging from 0 to 255 for an 8-bit image), and the Y-axis represents the number of pixels in the image with that intensity value.

   (b) *Colour Lines:* The Red line shows the distribution of red pixel intensities, the Green line shows the distribution of green pixel intensities, and the Blue line shows the distribution of blue pixel intensities.

   (c) *Interpretation:* Peaks in the graph represent the colours that are most present in the image. For instance, a high peak in the red line around the value 200 would indicate that the image has many pixels with high red intensity, suggesting the image may visually appear reddish.

   (d) *Colour Composition:* The overall shape of these colour distributions can provide an idea about the colour composition of the images.

   (e) *Utility:* The RGB Histogram aids in understanding the dominant colours in the image, the contrast, and the brightness. Variations in these histograms across the image set might be related to different sensor readings or variations in image capture settings.

3. **Data Analysis:** Eight metrics have been defined to quantify various properties of an image. Each of these metrics provides insight into different aspects of the image, allowing for a detailed analysis and comparison of images. These metrics are as follows:

   (a) **Contrast:** The Contrast chart visualizes the degree of local variation in an image, which can be associated with the details or changes in sensor readings.

   (b) **Dissimilarity:** Dissimilarity, like contrast, measures local variations, offering additional information about changes in the image.

   (c) **Homogeneity:** The Homogeneity chart shows the closeness of the distribution of elements in an image to its diagonal, providing insight into the uniformity or variation in sensor readings.

   (d) **Energy:** The Energy chart encapsulates the sum of squared elements in the image, which can suggest patterns or randomness in sensor readings.

   (e) **Correlation:** The Correlation chart illustrates the joint probability occurrence of specific pixel pairs, thereby hinting at the predictability or scatter of sensor readings.

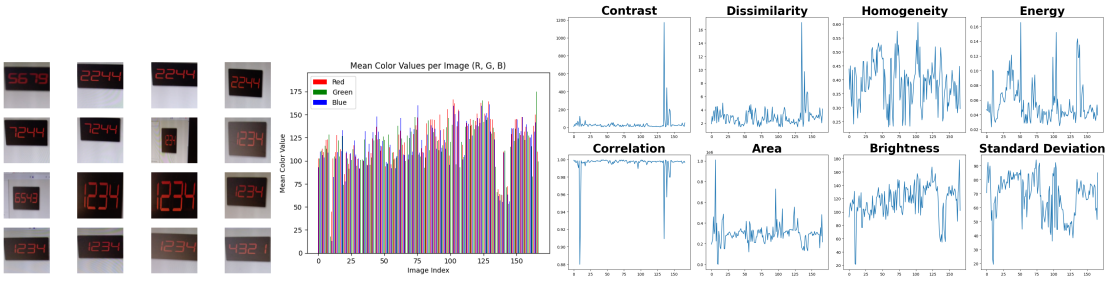
   (f) **Area:** The Area chart, in our context, represents the total area of contours detected in an image, providing information on the complexity of sensor readings.

   (g) **Brightness:** The Brightness chart displays the average lightness or darkness of each image, which might be influenced by different environmental conditions or sensor settings.

   (h) **Standard Deviation:** The Standard Deviation chart shows the variability in pixel intensities within each image, helping infer the contrast, detail, and complexity of sensor readings.

3.3.1 Image Folder A

There are 167 JPEG files totalling a size of 15.78mb in this folder. The images are of varying dimensions.

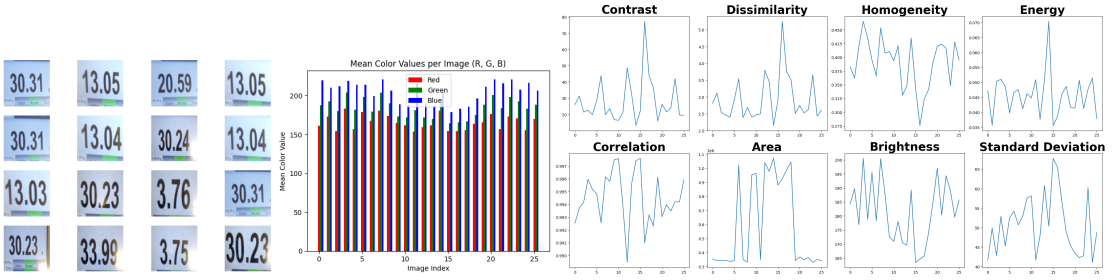


Montage RGB Data Analysis

FIGURE 3.1: Image Folder A Analysis

3.3.2 Image Folder B

There are 26 JPEG files totalling a size of 77.88mb in this folder. The images are of varying dimensions.

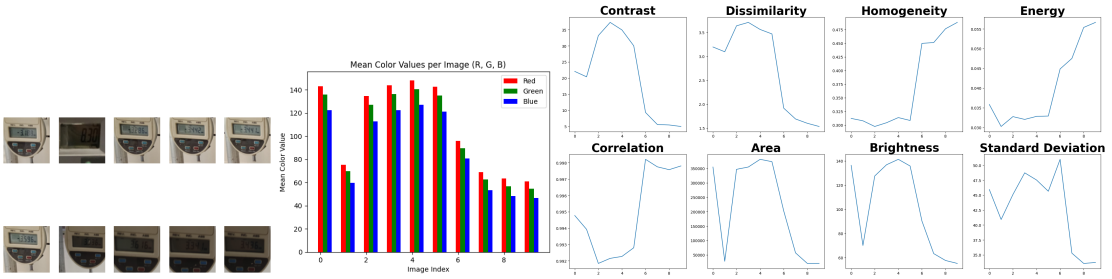


Montage RGB Data Analysis

FIGURE 3.2: Image Folder B Analysis

3.3.3 Image Folder C

There are 10 JPEG files totalling a size of 4.52mb in this folder. The images are of varying dimensions.

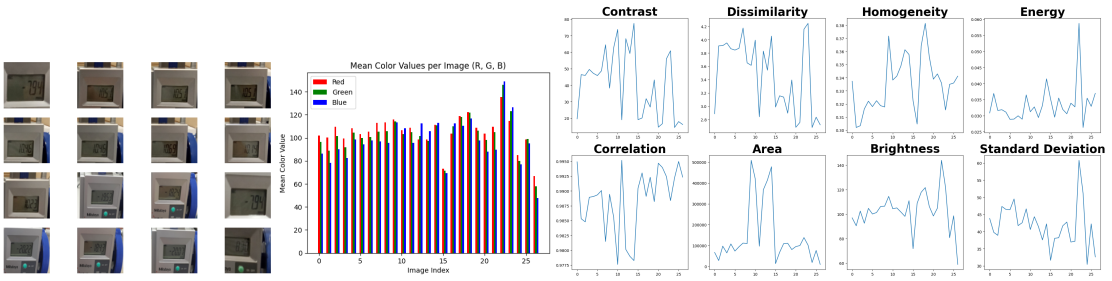


Montage RGB Data Analysis

FIGURE 3.3: Image Folder C Analysis

3.3.4 Image Folder D

There are 27 JPEG files totalling a size of 6.96mb in this folder. The images are of varying dimensions.

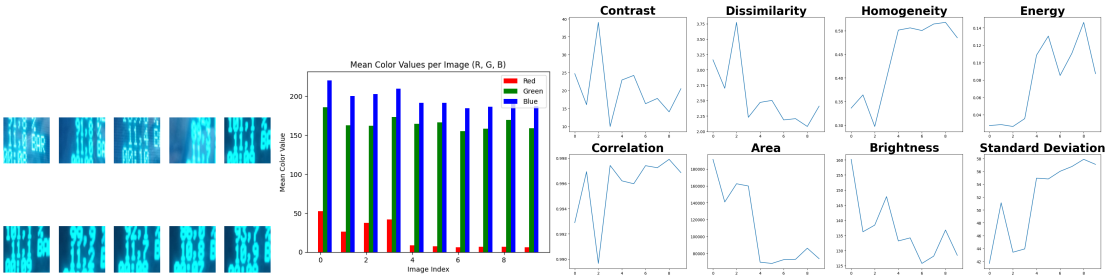


Montage RGB Data Analysis

FIGURE 3.4: Image Folder D Analysis

3.3.5 Image Folder E

There are 10 JPEG files totalling a size of 1.79mb in this folder. The images are of varying dimensions.

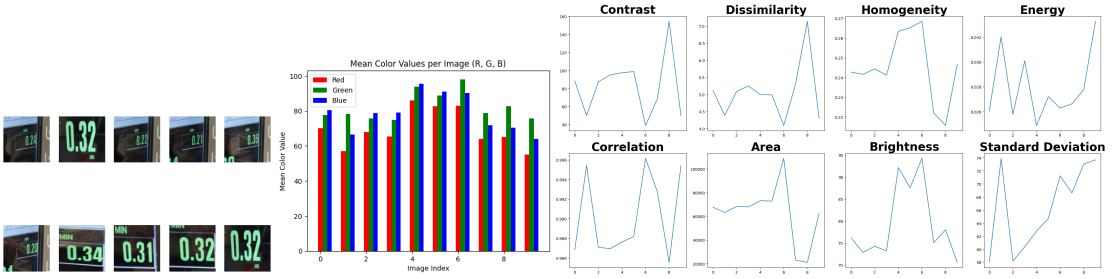


Montage RGB Data Analysis

FIGURE 3.5: Image Folder E Analysis

3.3.6 Image Folder F

There are 10 JPEG files totalling a size of 4.98mb in this folder. The images are of varying dimensions.

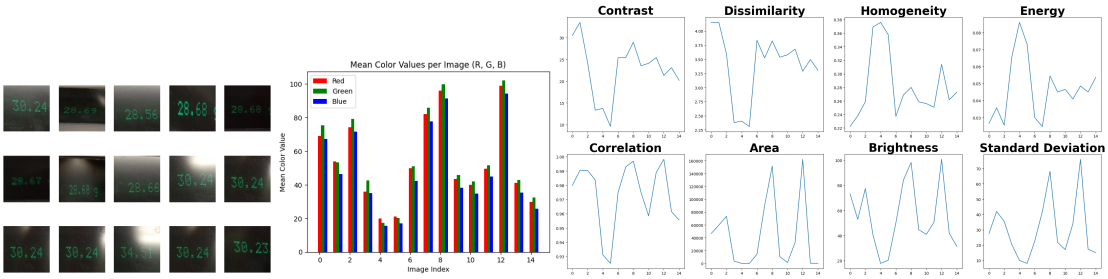


Montage RGB Data Analysis

FIGURE 3.6: Image Folder F Analysis

3.3.7 Image Folder G

There are 15 JPEG files totalling a size of 5.93mb in this folder. The images are of varying dimensions.

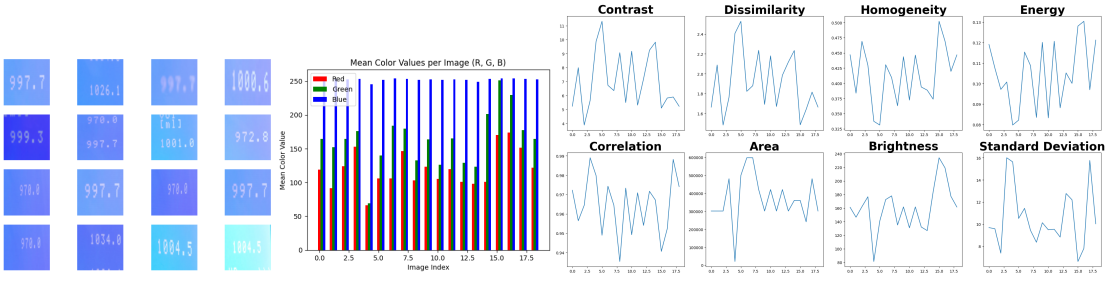


Montage RGB Data Analysis

FIGURE 3.7: Image Folder G Analysis

3.3.8 Image Folder H

There are 12 JPEG files totalling a size of 7.34mb in this folder. The images are of varying dimensions.

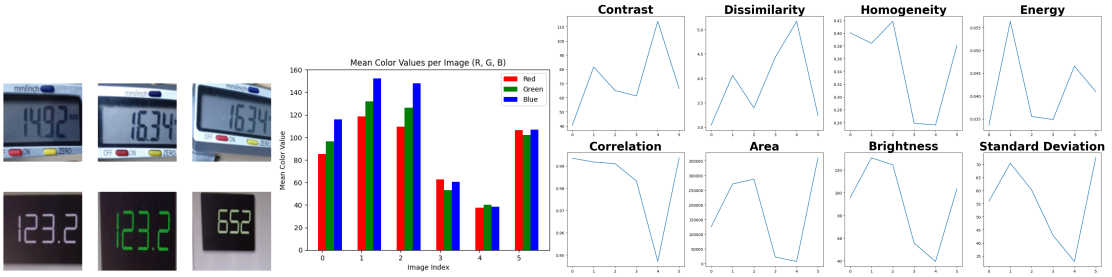


Montage RGB Data Analysis

FIGURE 3.8: Image Folder H Analysis

3.3.9 Image Folder I

There are 6 JPEG files totalling a size of 3.36mb in this folder. The images are of varying dimensions.

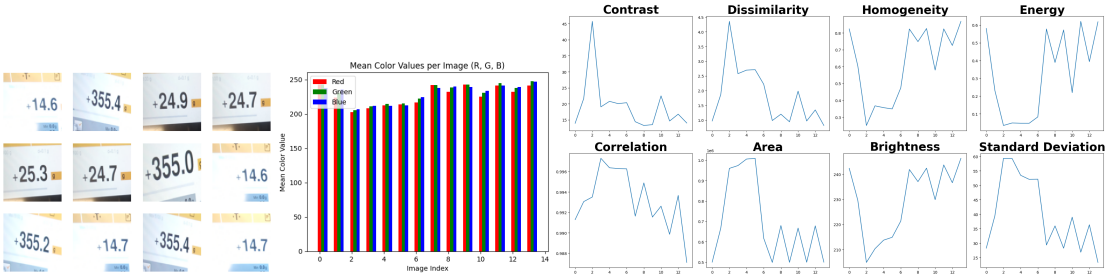


Montage RGB Data Analysis

FIGURE 3.9: Image Folder I Analysis

3.3.10 Image Folder J

There are 14 JPEG files totalling a size of 6.16mb in this folder. The images are of varying dimensions.



Montage RGB Data Analysis

FIGURE 3.10: Image Folder J Analysis

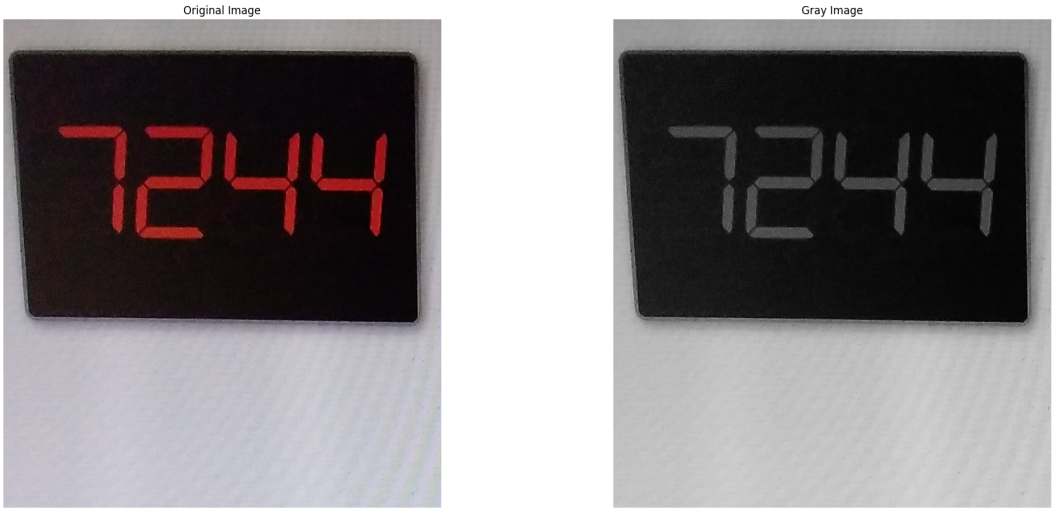
3.4 First Sprint - Global Generic

The first run performs Optical Character Recognition (OCR) on all of the images using the pytesseract library, a Python interface for the Tesseract OCR engine.

```
config_tesseract = '--tessdata-dir ./ttesseract_langs --psm 13 tessedit_char_whitelist=0123456789'
```

FIGURE 3.11: PyTesseract Config Settings

The pre-processing here involves turning the image to greyscale. Converting a colour image to greyscale is a process of condensing the three colour channels (red, green, and blue) into a single channel that represents the image's brightness. This is done by applying specific weights to each channel, which mimic the way the human eye perceives colour. The weights used are 0.2989 for red, 0.5870 for green, and 0.1140 for blue. This process reduces the amount of data required to represent the image, which can simplify many image processing tasks. [21]



Original Image Greyscale Image

FIGURE 3.12: Greyscale Conversion

OCR is performed on the images using the configuration specified in Fig 3.11 above. Several variations of this configuration were tested and settings of PSM 6 and 7 were found to provide the good results, however 13 appears to provide the best results overall.

These results will be discussed in the Results chapter.

3.5 Second Sprint - Global Generic Analysis Resized

The second sprint supplemented the work carried out in the first sprint by adding Otsu's thresholding and morphological closing to the pre-processing steps. It also included a new resizing technique and added a seven segment display language file, both of which further improved the accuracy of the OCR system. These enhancements advanced the first sprint, each warranting a more detailed exploration to fully appreciate their contribution to the improved performance.

Method - Otsu's Thresholding

Otsu's method is a global thresholding technique used in image processing. It is named after its inventor, Nobuyuki Otsu, and works by minimizing the intraclass variance, which is a measure of how similar the pixels within each class are. The optimal threshold value is the one that produces the two classes with the lowest intraclass variance. [22]

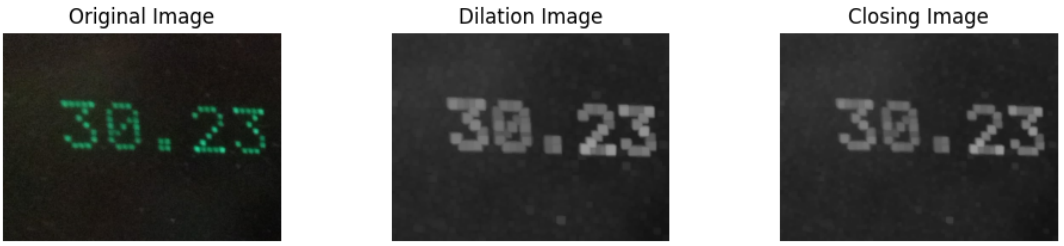
Original Image Otsu's Thresholding

FIGURE 3.13: Otsu's Thresholding

Once the optimal threshold value has been determined, the image can be binarized, which means converting it to a black and white image. In this sprint, the pixels with values below the threshold will be set to black, and the pixels with values above the threshold will be set to white.

Method - Morphological Closing

Morphological closing is an image processing operation that is used to close small holes in the foreground of an image. In OpenCV, closing is performed by first applying a dilation operation, which grows or thickens objects in the image, followed by an erosion operation, which shrinks objects in the image. The size and shape of the area affected by each operation depends on the structuring element used. The overall effect of the closing operation is that small holes within an object, thin lines or gaps between objects, and small black points on the object are eliminated, while keeping the size and shape of the object roughly the same as before the operation. This operation is particularly useful in many image processing tasks, such as noise reduction and separation of touching objects. [23]



Original Dilated Closing

FIGURE 3.14: Morphological Closing

Method - Tesseract Language Files

The lang parameter that can be passed into Tesseract specifies the language of the text to be recognized. Tesseract is capable of recognizing text in multiple languages. To utilize this functionality, the appropriate language data files must be downloaded and installed. These files can be found on the Tesseract GitHub page. [24]

```
# Read text from image using English character training data
text_eng = pytesseract.image_to_string(image, lang="eng", config=config_tesseract)
```

FIGURE 3.15: PyTesseract Config Settings Language

The language files used in this research are as follows:

- eng.traineddata - English

- ssd.traineddata - Seven Segment Display

Method - Resizing

To improve the results of optical character recognition (OCR) using Tesseract, image resizing during pre-processing was explored. Tesseract's performance can be sensitive to the scale of the image, as the size of the text can greatly impact the OCR engine's ability to accurately recognize characters. Therefore, resizing images to various scales became an essential part of the pre-processing pipeline.

Resizing is performed using OpenCV's resize() function, which allows images to be scaled up or down. By altering the resolution, the text in the images is effectively manipulated to appear larger or smaller. It is worth noting that while upscaling can sometimes help in capturing more detail and thereby improving OCR accuracy, it also increases computational load. Conversely, downscaling an image reduces the computational burden but might cause loss of important details that can negatively affect OCR performance. [25]

The Python script was redesigned to run the control function in a loop incrementally adjusting the image resize parameter from 50 pixels to 749 pixels and evaluating the subsequent influence on Optical Character Recognition (OCR) performance. The aim was to find an ideal image size that would not make the text too small or too large, thus optimizing the recognition potential of Tesseract.

3.5.1 Second Sprint Conclusion

The results for the Second Sprint will be discussed in the Results chapter.

The challenge of performing OCR on multiple sets of images is that the images vary widely in terms of colour profiles, lighting conditions, text styles, noise levels, and other factors. This means that with using Tesseract, a single generic pre-processing pipeline is not sufficient. Instead, a more nuanced and flexible approach is required.

OpenCV, an open-source computer vision library, provides a wide range of image processing functionalities. This allows us to tailor the pre-processing steps to each individual folder of images. For example, some folders may require grayscale conversion, Gaussian blurring, or adaptive thresholding. Others may benefit from morphological transformations such as dilation and erosion, or image resizing and rotation.

Tuning and applying diverse techniques to different image folders may present as a complex task. However, pursuing this course is essential for achieving optimal OCR results. This approach led to overcoming the issue of a non-functional global run and significantly improving the accuracy of OCR outcomes.

The task underlined the importance of adopting an adaptive and individualized methodology when handling diverse datasets. A uniform global strategy may not always be feasible, and it often results in suboptimal outcomes. Instead, it proves beneficial to tailor the approach to the specific characteristics of each dataset.

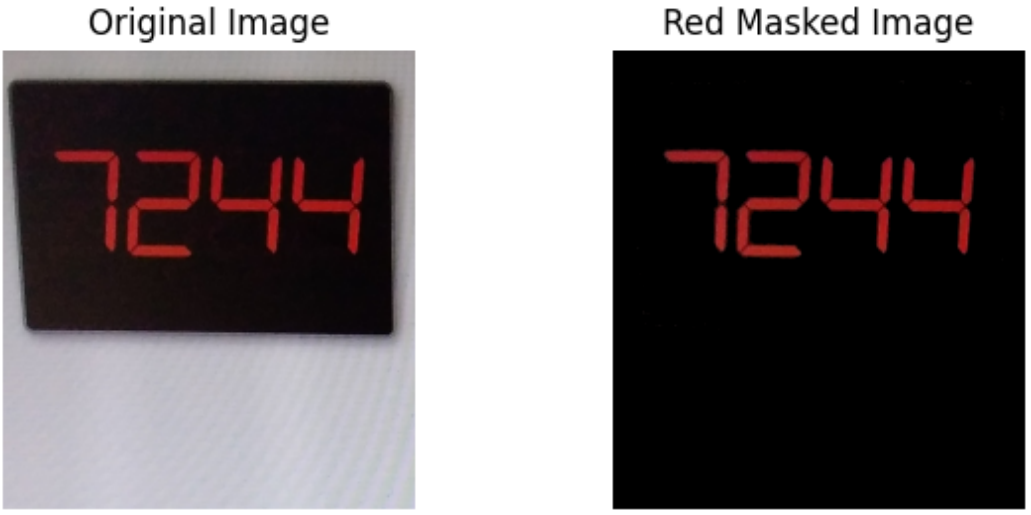
3.6 Analysis Tesseract Separate Folders

Navigating the variety of image properties in the dataset presented a unique challenge when utilizing Tesseract for the execution of Optical Character Recognition (OCR). This situation called for an adaptive approach, rather than a generic solution. The upcoming section will expand on pre-processing techniques beyond those already discussed, including the application of a red mask via OpenCV. The individual tailoring of steps to each subset of images and the inclusion of the red mask technique will be explored.

Method - Red Mask

The Red Mask function operates on an input image to isolate and return only the red pixels in that image.

The input image should be a numpy ndarray representing the original image in BGR format. To process this, the function first converts the image into the HSV (Hue, Saturation, Value) colour space using OpenCV's cvtColor function.



Original Image Red Mask Image

FIGURE 3.16: Red Mask

Because the hue component of red colour in HSV space spans both ends of the hue spectrum, two ranges are defined to capture the entire red hue — the lower range (0-10)

and the higher range (170-180). These ranges are combined with saturation and value thresholds to define what is considered a red pixel in the image.

The function then creates two binary masks for these ranges, using OpenCV's inRange function, which applies these boundaries on the HSV image. The resulting masks have pixel values of 255 where the original image pixels are within the specified red range, and 0 otherwise.

These two masks are added together to form a comprehensive mask of red pixels. The function then applies this mask onto a copy of the original image, setting all the pixels where the mask equals 0 to also be 0 in the output image. This leaves only the red pixels visible in the output image. Hence, the function returns an image emphasizing the red components of the original input.

3.6.1 Image Folder A

FIGURE 3.17: Image Folder A Montage

The methodology for processing images in Folder A involved the following steps:

1. An initial exploratory analysis was conducted where the images were resized to sizes ranging from 50 to 649 pixels in both height and width. This was done to empirically determine the optimal size that yields the best results in subsequent processing and text extraction steps.

2. TAfter analysing the results, an image size of 104 pixels was found to be the optimal size and was used for further processing of the images.

3. A Red Mask was applied. The significance and process of this technique has been previously explained in the Introduction.

4. The images were then converted to grayscale to reduce computational complexity and focus on intensity values.

5. The OTSU method was applied to further process the images. The choice of this method is discussed in the Introduction.

6. Text was extracted using the ENG and SSD Tesseract training libraries, which were chosen for their proven efficiency and accuracy in optical character recognition tasks.

7. The process was repeated with the additional step of using morphological operations of Closing and Dilation, in order to remove noise and enhance the accuracy of text extraction.

8. The results from each step and the final output were saved to a CSV file for further analysis and interpretation.

	index	image...	seen_data_numeric	size_used	*closing_ssd*	closing_eng	mro_ssd	mro_e...
8	0	['./sipaim...	6543	104	NaN	NaN	17	NaN
9	0	['./sipaim...	1234	104	1634	NaN	1	NaN
10	0	['./sipaim...	1234	104	81	3	111	NaN
11	0	['./sipaim...	1234	104	190	NaN	nan	NaN
12	0	['./sipaim...	1234	104	NaN	NaN	1	3
13	0	['./sipaim...	1234	104	NaN	NaN	11234	NaN
14	0	['./sipaim...	1234	104	NaN	NaN	nan	NaN
15	0	['./sipaim...	4321	104	19	NaN	nan	NaN
16	0	['./sipaim...	4321	104	NaN	NaN	131	NaN
17	0	['./sipaim...	4321	104	NaN	NaN	4211	NaN
18	0	['./sipaim...	4321	104	NaN	NaN	611	NaN
19	0	['./sipaim...	672	104	88	2	672	612
20	0	['./sipaim...	674	104	NaN	NaN	674	NaN
21	0	['./sipaim...	675	104	NaN	NaN	675	515
22	0	['./sipaim...	678	104	NaN	NaN	678	578
23	0	['./sipaim...	678	104	NaN	NaN	13715	678
24	0	['./sipaim...	678	104	NaN	NaN	12151	NaN
25	0	['./sipaim...	678	104	NaN	NaN	678	678
26	0	['./sipaim...	678	104	NaN	NaN	678	578
27	0	['./sipaim...	678	104	NaN	NaN	678	NaN
28	0	['./sipaim...	678	104	NaN	NaN	678	NaN
29	0	['./sipaim...	678	104	NaN	NaN	678	NaN

FIGURE 3.18: Image Folder A Sample Output

The figure above presents a sample of the output generated from the Image Folder A analysis. This analysis file is discussed in the Results section of this document.

3.6.2 Image Folder B

FIGURE 3.19: Image Folder B Montage

The methodology for processing images in Folders A and B involved the following steps:

1. An initial exploratory analysis was conducted where the images were resized to sizes ranging from 50 to 649 pixels in both height and width. This was done to empirically determine the optimal size that yields the best results in subsequent processing and text extraction steps.

2. After analysing the results, an image size of 550 pixels was found to be the optimal size and was used for further processing of the images.

3. An initial pre-processing function of thresholding (thresh) was applied. This method helps in separating an object from its background and improves the overall contrast of the images.

4. Following the thresholding step, the images underwent morphological closing operations. This technique, which involves dilation followed by erosion, is used to close small holes in the object, making the images cleaner for further processing.

5. The OTSU method was applied as a final pre-processing step to binarize the images. This adaptive thresholding method maximizes inter-class variance and improves the precision of text extraction, as discussed in the Introduction.

6. Text was extracted using the ENG and SSD Tesseract training libraries, chosen for their proven efficiency and accuracy in optical character recognition tasks.

7. The results from each pre-processing step and the final output were saved to a CSV file for further analysis and interpretation.

	index	image...	seen_...	size_used	closin...	closin...	mro_ssd	mro_e...	thresh...	thresh...
				550						
13000	0	[':\\sipai...	30.31	550	31	31	3011410...	3031.0	31	31
13001	0	[':\\sipai...	13.05	550	1303	13.05	['131051...	1305309.0	13.05	1303
13002	0	[':\\sipai...	20.59	550	0	5	1106000...	2059	5	0
13003	0	[':\\sipai...	13.05	550	13.08	13.05	13119	1305	13.05	13.08
13004	0	[':\\sipai...	30.31	550	30.9	303	11	30.31	303	30.9
13005	0	[':\\sipai...	13.04	550	13.04	13.04	1311193...	13083	13.04	13.04
13006	0	[':\\sipai...	30.24	550	0	2	['111145...	3012451...	2	0
13007	0	[':\\sipai...	13.04	550	13.04	13.04	13.04	13.04	13.04	13.04
13008	0	[':\\sipai...	13.03	550	13.09	13.03	13.09	13.03	13.03	13.09
13009	0	[':\\sipai...	30.23	550	14	30.23	110	NaN	30.23	14
13010	0	[':\\sipai...	3.76	550	410	0.16	331111011	11	0.16	410
13011	0	[':\\sipai...	30.31	550	31	31	['000000...	3031	31	31
13012	0	[':\\sipai...	30.23	550	11	0	['111111...	['302319...	0	11
13013	0	[':\\sipai...	33.99	550	0	3	3.1	30	3	0
13014	0	[':\\sipai...	3.75	550	41	413	41	318	413	41
13015	0	[':\\sipai...	30.23	550	10.01	3023	10.01	30.22	3023	10.01
13016	0	[':\\sipai...	30.24	550	1	3.0	41.44	24	3.0	1
13017	0	[':\\sipai...	33.99	550	4	89	3404910...	339910000	89	4
13018	0	[':\\sipai...	3.75	550	317	30	31100.0	3175	30	317
13019	0	[':\\sipai...	13.04	550	13.04	1304	13.04	1304	1304	13.04
13020	0	[':\\sipai...	13.05	550	NaN	NaN	13.05914...	['1305....']	NaN	NaN
13021	0	[':\\sipai...	30.3	550	0.0	30	['000010...	30303	30	0.0
13022	0	[':\\sipai...	13.04	550	104	4	['000011...	['1304...']	4	104
13023	0	[':\\sipai...	13.04	550	15.04	13.04	1910709...	['.']	13.04	15.04
13024	0	[':\\sipai...	30.31	550	1	31	1830031...	0.3031	31	1
13025	0	[':\\sipai...	13.04	550	114	134	13.04	1304	134	114

FIGURE 3.20: Image Folder B Sample Output

3.6.3 Image Folder C

Two new methods were introduced for working with Image Folder C.

Method - Denoise

The *cv2.fastNlMeansDenoisingColored* function in OpenCV is a fast algorithm for denoising colour images. It works by converting the image to CIELAB colour space and then denoising the L and AB components separately. The denoising is done using a non-local means algorithm, which works by finding similar patches in the image and averaging them together. The parameters of the algorithm can be adjusted to control the amount of denoising. [26]

The *cv2.fastNlMeansDenoisingColoured* function takes the following parameters:

- **src**: The input image.
- **dst**: The output image.
- **h**: The parameter that controls the amount of denoising for the L component.
- **hColour**: The parameter that controls the amount of denoising for the AB components.
- **templateWindowSize**: The size of the template patch used for denoising.
- **searchWindowSize**: The size of the search window used for denoising.

The `cv2.fastNlMeansDenoisingColoured` function is a fast and effective way to denoise colour images. It is particularly well-suited for images that have been corrupted by Gaussian noise.[26]

Method - Weiner Filter

The Wiener filter is a linear filter that is used to denoise signals that have been corrupted by additive white Gaussian noise (AWGN). The Wiener filter is optimal in the sense that it minimizes the mean-squared error between the denoised signal and the original signal.

The Wiener filter is defined as follows:

$$w(f) = K \cdot R_x x^{-1} \cdot R_x n \tag{3.1}$$

FIGURE 3.21: Weiner Filter Equation
[27]

The Wiener filter can mitigate the effects of sunlight, treated as additive white Gaussian noise, on an image. It estimates the sunlight's power spectrum, applying its inverse to the image, thus reducing sunlight while preserving the original signal. The Wiener filter can be implemented through:

- Frequency domain: Adjusting the image using the inverse of the sunlight's power spectrum.
- Time domain: Utilizing a recursive algorithm.

However, it may produce ringing artifacts and requires precise knowledge of the sunlight's power spectrum for effective results.

FIGURE 3.22: Image Folder C Montage

The methodology for Image Folder C was developed to maximize the readability and accuracy of the extracted text from the images. The following steps were taken:

FIGURE 3.23: Image Folder C Sample Output

1. **Manual Cropping**: Images were manually cropped to isolate the text. This process was necessary to ensure that only the relevant portions of the images were analysed. It is important to note that this manual step could be avoided with more precise camera positioning during the initial image capture phase.

2. **Deblurring**: A deblurring operation was performed on the images using the Wiener filter. This step was necessary to reduce the blur caused by linear motion or unfocused optics.

3. **Thresholding**: The images underwent a thresholding operation. This process was required to help separate the text (foreground) from the background, improving the contrast and readability of the text.

4. **Denoising**: The images were denoised to reduce the noise present. This step was essential to further enhance the clarity and readability of the text.

5. **Text Extraction and Accuracy Assessment**: Text was extracted from the processed images. To evaluate the accuracy of the extraction, the extracted text was compared with predefined labels. The text that matched the predefined label the closest was considered to be the most accurate.

3.6.4 Image Folder D

FIGURE 3.24: Image Folder D Montage

The methodology for Image Folder D involved a series of systematic steps to accurately read the text within the images and output the analysis to a CSV file. The procedure followed is detailed below:

1. **Image Cropping:** The text within the images was manually cropped. While this method was chosen for its simplicity and effectiveness, the necessity of this step could be mitigated in the future by improving the camera positioning to automatically focus on the text.

2. **Grayscale Conversion:** The cropped images were then converted to grayscale. This step is crucial as it simplifies the image, reduces computational complexity, and is preferred for most image processing tasks such as the OCR (Optical Character Recognition) used in this project.

3. **Median Blurring:** The grayscale images underwent a median blur process. This step helps in reducing noise within the images, thereby enhancing the efficiency of the OCR.

4. **Text Recognition:** The processed images were then used to read text using Optical Character Recognition (OCR) with English language (ENG) and Seven Segment Display (SSD) configuration files. The SSD configuration, an algorithm for object detection, aids in accurately identifying and locating the text within the image.

5. **Data Export:** Finally, the text recognized from the images was analysed, and the output was exported to a CSV file for further analysis and record-keeping.

FIGURE 3.25: Image Folder D Sample Output

3.6.5 Image Folder E

FIGURE 3.26: Image Folder E Montage

In this project, Image Folder E was not included in the data set and thus, no processing or analysis was conducted on it.

3.6.6 Image Folder F

FIGURE 3.27: Image Folder F Montage

In this section, a function similar to the previously discussed Red Mask technique is introduced and utilized. This new function, termed the 'Green Mask', operates under similar principles but targets green pixel values instead of red.

Method - Green Mask

The Green Mask function isolates and returns only the green pixels in an input image. The input image is a numpy ndarray representing the original image in BGR format. The function first converts the image into the HSV (Hue, Saturation, Value) colour space using OpenCV's cvtu function.

In the HSV colour space, green occupies a certain section of the hue spectrum. A specific range is defined to capture the green hue, typically around 36-70. This range, along with specific thresholds for saturation and value, defines what is considered a green pixel in the image.

The function then creates a binary mask for this range using OpenCV's inRange function. This function applies the boundaries of the defined range to the HSV image,

Original Image Green Mask Image

FIGURE 3.28: Green Mask

resulting in a mask with pixel values of 255 where the original image pixels are within the specified green range, and 0 otherwise.

The function then applies this mask onto a copy of the original image, setting all the pixels where the mask equals 0 to also be 0 in the output image. This leaves only the green pixels visible in the output image. Therefore, the function returns an image emphasizing the green components of the original input.

The methodology for Image Folder F utilized a sequential process to efficiently read text within the images and subsequently output the analysis to a CSV file. The steps involved in the process are as follows:

1. **Image Cropping:** The text within the images was manually cropped. This task, while manually intensive, could be avoided in future iterations by optimizing camera positioning to directly focus on the text.

2. **Green Mask Application:** A green mask was applied to the images to isolate specific features or areas of interest in the image, enhancing the subsequent image processing steps.

3. **Grayscale Conversion:** The masked images were then converted to grayscale. This step reduces computational complexity and is a standard pre-processing step in many image processing workflows, including OCR (Optical Character Recognition).

4. **Deblurring:** A deblurring operation was performed on the grayscale images to enhance the clarity and legibility of the text in the images.

5. **Thresholding:** Thresholding was applied to the deblurred images, converting them into a binary format. This step helps in separating the text (foreground) from the background.

6. **Denoising:** The binary images underwent a denoising process to further reduce noise and improve the effectiveness of the subsequent OCR process.

7. **Text Recognition:** The denoised images were then used to read text using Optical Character Recognition (OCR) with English language (ENG) and Single Shot MultiBox Detector (SSD) configuration files. SSD configuration, a method for object detection, is used to accurately identify and locate the text within the images.

8. **Data Export:** The recognized text from the images was analysed, and the output was exported to a CSV file for further analysis and record-keeping.

Every step was conducted with precision to ensure the accuracy of the results and the effectiveness of the method employed.

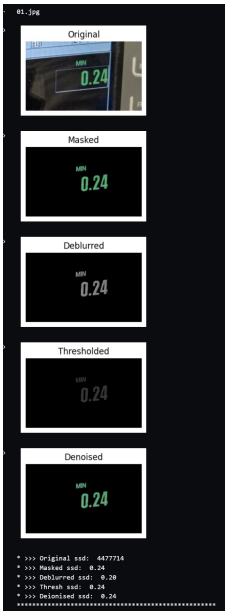


FIGURE 3.29: Image Folder F Sample Output

3.6.7 Image Folder G

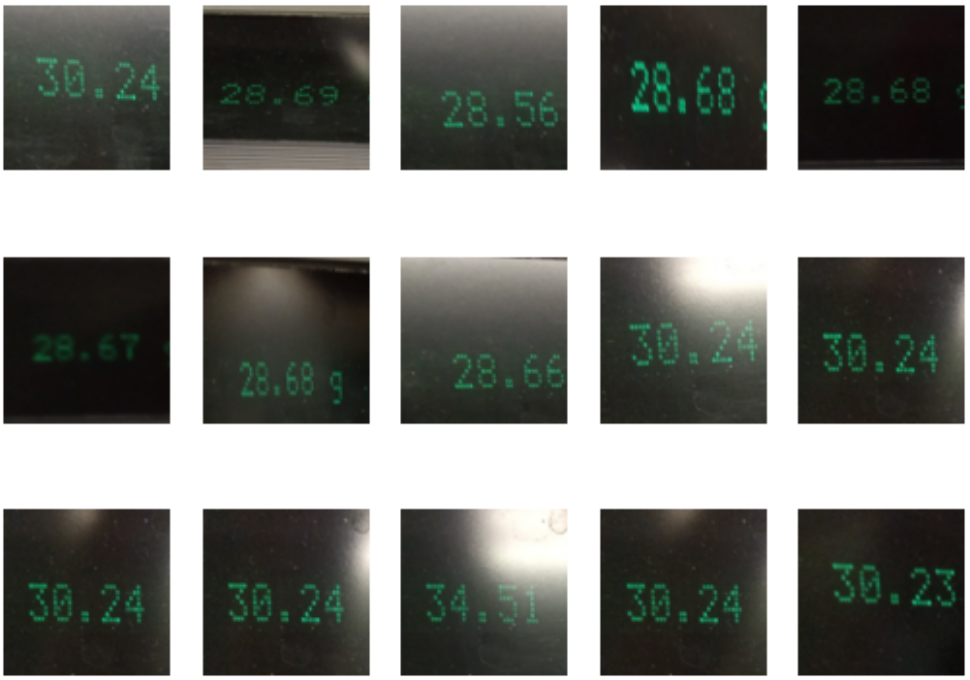


FIGURE 3.30: Image Folder G Montage

The methodology for Image Folder G utilized a sequential process to efficiently read text within the images and subsequently output the analysis to a PDF file. The steps involved in the process are as follows:

1. **Image Cropping:** The text within the images was manually cropped. This task, while manually intensive, could be avoided in future iterations by optimizing camera positioning to directly focus on the text.

2. **Mask Green**: Apply a green colour mask to images. This step isolates the green parts of the image for further processing.

3. **Grayscale**: Convert the masked image to grayscale. This step simplifies the image and is a common requirement for many image processing algorithms.

4. **Deblur**: Deblur the grayscale image. This step sharpens the image, enhancing details for the OCR.

5. **Threshold**: Apply a thresholding technique on the deblurred image. This step separates the image into foreground and background, aiding in the recognition of text.

6. **OCR**: Use Tesseract OCR on the processed images. This step recognizes text in various languages and fonts, including English (ENG), SSD, and Dot Matrix.

7. **PDF Generation**: Use the ReportLab library in Python to create a PDF of the OCR results. This step provides a convenient way to view and share the results.

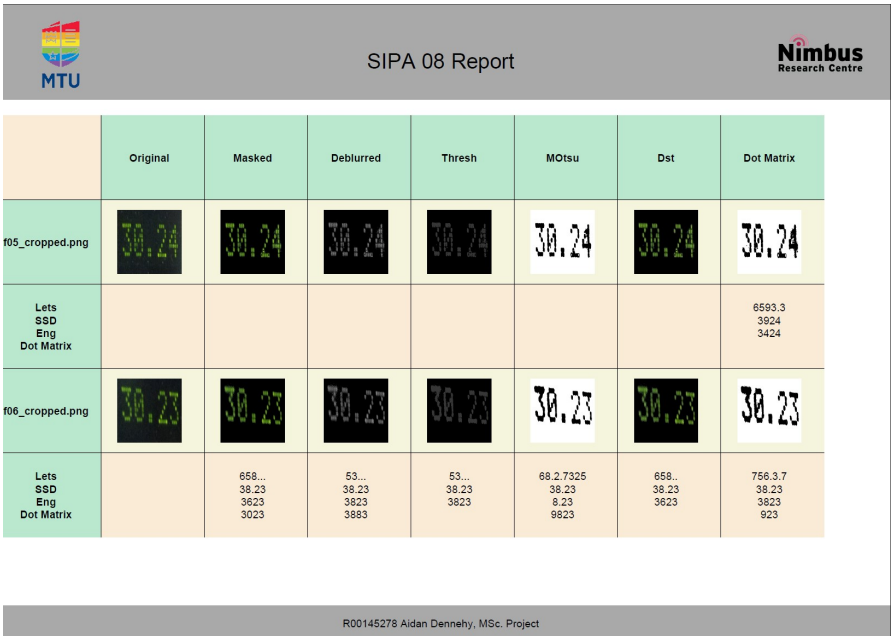


FIGURE 3.31: Image Folder G Sample Output

3.6.8 Image Folder H

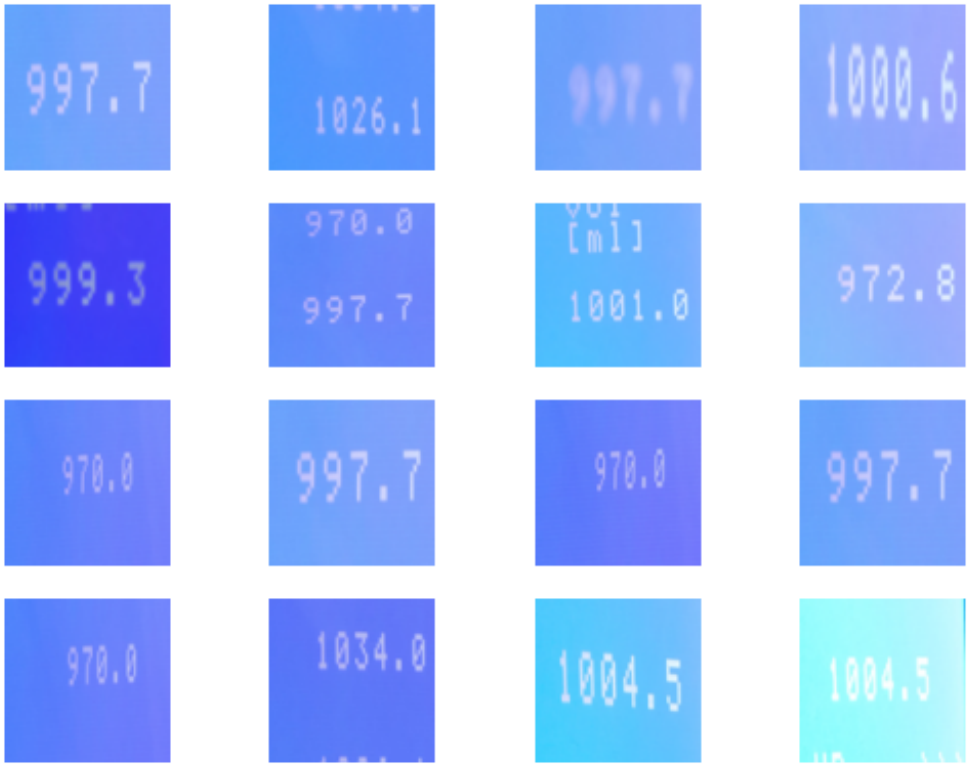


FIGURE 3.32: Image Folder H Montage

The methodology for Image Folder H used a sequential process to read text within the images efficiently and output the analysis to a PDF file. The steps involved in the process were as follows:

1. **Manual Crop**: Manually crop the image to focus on the area of interest and remove any irrelevant parts.
2. **Convert to Grayscale**: Convert the cropped image to grayscale. This simplifies the image, reducing the amount of information to process, and is a common requirement for many image processing algorithms.
3. **Invert Image**: Invert the grayscale image. This step switches the light and dark areas of the image, which can sometimes help in improving OCR results.
4. **Threshold Grayscale Image**: Apply a thresholding technique to the inverted grayscale image. This step separates the image into foreground and background, aiding in the recognition of text.

5. **OCR Using Tesseract**: Use Tesseract OCR on the threshold image. Run Tesseract with multiple configurations, using the SSD, LetsGoDigital, DotMatrix, and ENG language models, to find the best result.

6. **PDF Generation**: Use the ReportLab library in Python to create a PDF of the OCR results. This step provides a convenient way to view and share the results.

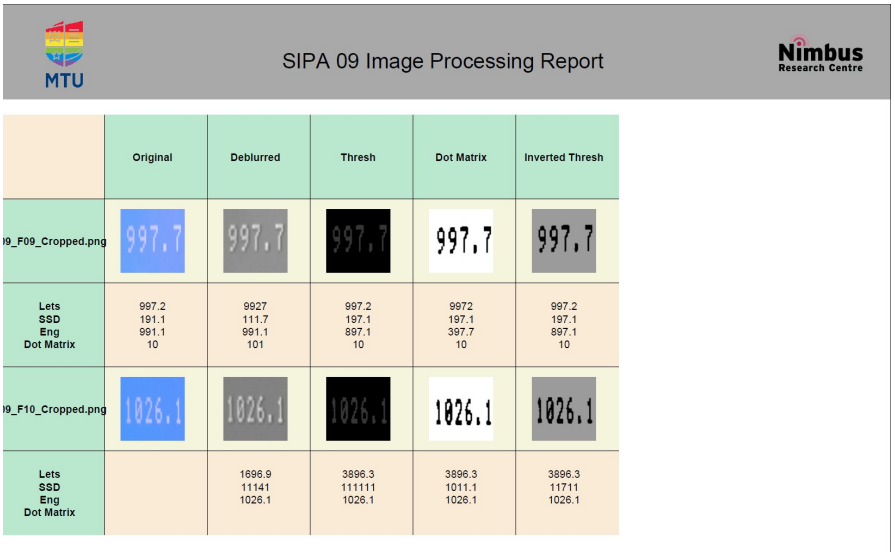


FIGURE 3.33: Image Folder H Sample Output

3.6.9 Image Folder I

FIGURE 3.34: Image Folder I Montage

Image Folder I houses images that have already been processed through the workflows of Folders A, C, and D.

3.6.10 Image Folder K

FIGURE 3.35: Image Folder K Montage

Black text on a white background is a good scenario for OCR. The main problem with these images above is skewness.

A function was developed to estimates the rotation angle of an image by detecting lines using the Hough Transform. [28] The function first converts the image to grayscale and applies the Canny edge detector to find edges. It then uses the Hough Transform to detect lines in the image. If no lines are detected, it returns 0. Otherwise, it calculates the angles of the detected lines, and computes and returns the median of these angles as the estimated rotation angle.

The methodology for processing images in Folder K involved the following steps:

1. **Manual Crop**: Manually crop the image to focus on the area of interest and remove any irrelevant parts.
2. **Correct Skewness**: Correct the skewness of the image using the Hough Transform.

3. **OCR Using Tesseract**: Use Tesseract OCR on the corrected image using the ENG model.

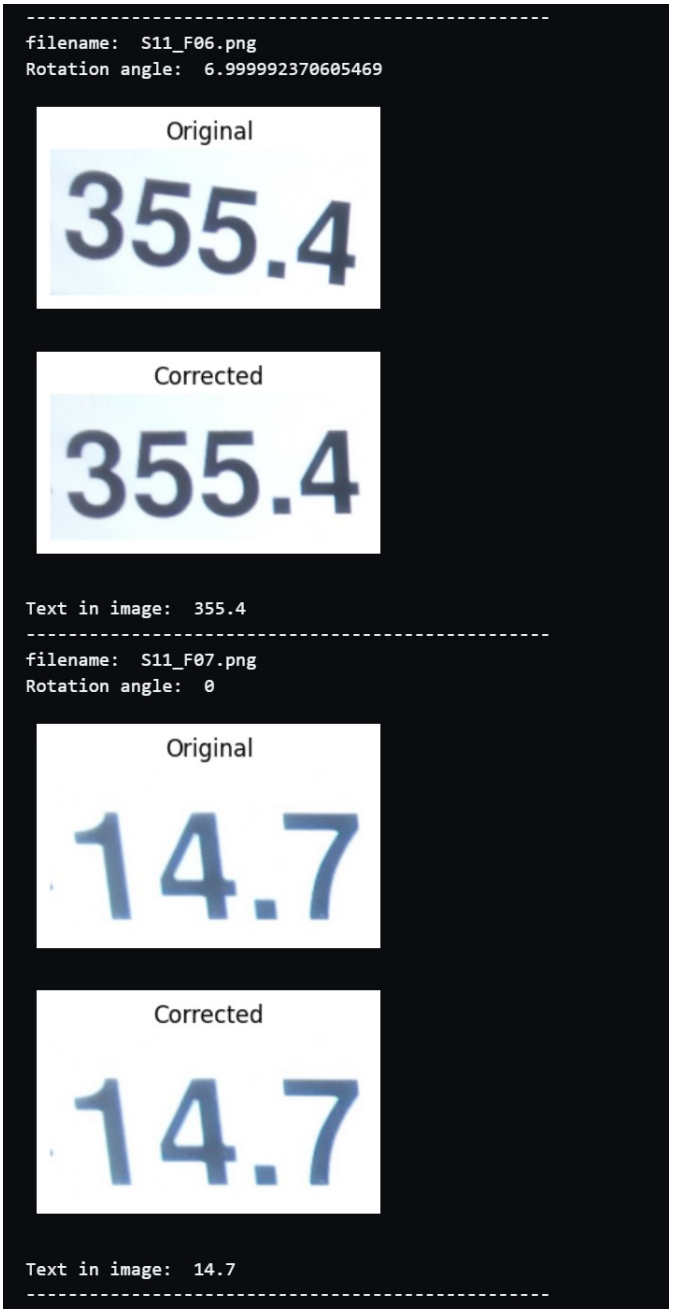


FIGURE 3.36: Image Folder K Sample Output

3.7 CRNN Methodology

3.7.1 Introduction

In this subsection, the specifics of utilizing Convolutional Recurrent Neural Networks (CRNNs) for the task are discussed. The CRNN, a hybrid model, harnesses the spatial feature extraction capabilities of CNNs with the sequence modelling prowess of RNNs. The subsequent sections guide through the systematic process: from building the training databases to defining the architecture of the CRNN model, followed by data preparation through loading, normalization, and one-hot encoding. Finally, the model's compilation and the prediction of numbers are addressed.

3.7.2 Building the Training Databases

While initial experiments with the CRNN utilized the MNIST [29] and SVHN [30] datasets for training, results indicated that a custom digits training database, tailored to the specific font present in the images, yielded the most optimal performance.

In this research, a Python-based approach was employed to generate random images of individual digits using a specified font. The Python Imaging Library (PIL) was used to perform image manipulations. The process begins by determining the desired font.

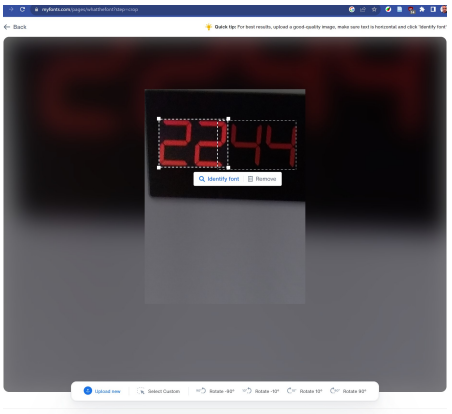


FIGURE 3.37: Image Font Identification

The process involved using font identification websites such as *www.myfonts.com*[31], but generally the best success was attained by looking for similar fonts on Google Fonts [32]. The font was then downloaded and installed on the local machine.

The primary function is tasked with producing a designated number of digit images and saving them in a predetermined directory. Each digit image file is labelled with a randomly generated alphanumeric name, followed by an underscore and the digit it represents.

To enhance the diversity of the dataset, digits are rendered in varying font sizes. Additionally, there's a 50% chance that any given image will undergo a slight rotation within a specified range. This introduces a semblance of natural variability that might be found in real-world digit representations, mimicking minor skews or rotations.

After the image creation, all the generated image file names, along with their respective digits, are documented in a structured format. This record, which acts as a catalogue, is then appended to a designated CSV file. The CSV provides a ready reference, allowing researchers to quickly correlate an image file to its corresponding digit without visual interpretation.

This methodology ensures a vast and varied dataset, essential for robust machine learning training or any analysis requiring a diverse representation of numerical digits.

3.7.3 Defining the CRNN Model

To instantiate the model in our experiment, we set the `input_shape` to be (32, 32, 1), as our pre-processed images are 32x32 pixels with 1 channel (grayscale). The number of output classes, `num_classes`, is set to 10 to represent the digits from 0 to 9.

```
input_shape = (32, 32, 1)
num_classes = 10
crnn_model = create_crnn_model(input_shape, num_classes)
```

The structure of the instantiated model is as follows:

- **Input Layer**: The model takes an input of shape `input_shape` (32, 32, 1) in our case.
- **Convolutional Layers**: The first part of our model comprises three convolutional blocks. Each block consists of:
  - **Convolutional Layer**: Uses a varying number of filters, starting from 32, then 64, and finally 128, all with a kernel size of 3x3 and 'same' padding.
  - **Batch Normalization**: Normalizes the outputs of the convolutional layers.
  - **ReLU Activation**: Introduces non-linearity, enabling the model to learn complex patterns.
  - **Max Pooling**: Reduces the spatial dimensions of the output.
  - **Dropout**: Applied with a rate of 0.25 to prevent overfitting.
- **Reshaping Layer**: The output of the convolutional layers is reshaped to a target shape of (4, 4*128) for the LSTM layer.
- **Recurrent Layers**: A Bidirectional LSTM layer with 256 units processes the reshaped sequences, followed by a TimeDistributed Dense layer with ReLU activation.
- **Flattening and Dropout**: The output from the TimeDistributed layer is flattened and passed through a Dropout layer with a rate of 0.5.
- **Output Layer**: A Dense layer with `num_classes` units (10 in our case) and a softmax activation function classifies the images. This layer also uses L1 and L2 regularization.

This model structure is chosen because it capitalizes on the strengths of both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Specifically, the CNN layers are adept at processing spatial information and extracting important features from the input images, making them ideal for image recognition tasks. Following this, the sequential patterns in these extracted features are processed by an RNN layer (specifically an LSTM), which is proficient in learning from temporal or sequential

data. This combination, often referred to as a Convolutional Recurrent Neural Network (CRNN), can be particularly effective in tasks like digit recognition, where recognizing both spatial features (e.g., shape of the digits) and sequential dependencies (e.g., order of strokes in handwritten digits) can be beneficial.

3.7.4 Loading the Training Data

In the methodology, the dataset generated of 500k seven segment digits is loaded. Each image corresponds to a specific digit that is labelled in a corresponding csv file. A load function is utilized to read and process these images. The images are accessed with their respective file names, read into memory as grayscale images using the OpenCV library.

Following this, each image is resized to a uniform size of 32x32 pixels, ensuring that the input to our model remains consistent. The images are then expanded along the last axis to create an additional dimension, a standard pre-processing step required by the Convolutional Neural Networks (CNNs). This dimension effectively represents the colour channels of the image, which in our case is one, due to the use of grayscale images. The processed images are then stored in a list, which is converted into a numpy array for efficient numerical operations.

To validate the model's performance and generalization capabilities, the dataset is divided into a training and testing set, using an 80-20 split. This split is achieved by using the *train_test_split* function from the Scikit-learn library. The resulting subsets are X_train and y_train for the training set, and X_test and y_test for the testing set, where 'X' represents the images and 'y' the corresponding labels. The *random_state* parameter is arbitrarily fixed at 42, providing reproducibility in the generation of the training and testing splits. This approach facilitates the reservation of a significant portion of the dataset for model training while ensuring a distinct set remains for evaluating the model's ability to generalize to unseen data.

3.7.5 Normalisation and One-Hot Encoding

The datasets are normalised and one-hot encoded to facilitate the training process. Normalisation is achieved by dividing the pixel values by the maximum grayscale value, which is 255. This transformation reduces the scale of input values, facilitating the convergence of our model during the training process. One-hot encoding is performed using the *to_categorical* function from the TensorFlow Keras utility module. This function converts the training and test labels (y_train and y_test) into one-hot vectors, a format required by the model algorithms when dealing with categorical targets. Each digit from 0-9 is represented as a 10-element vector with a single '1' in the position representing the digit and '0's elsewhere.

3.7.6 Model Compilation

The model is compiled using the *compile* function from the TensorFlow Keras API. The model employs the Adam optimization algorithm, a popular choice due to its efficient memory usage and capability to handle large datasets and parameters. The *categorical_crossentropy* is set as the loss function, which is suitable for multi-class classification tasks. The 'accuracy' of model is tracked as a metric during the training process.

The batch size, set at 64, defines the number of samples that will be passed through the network at one time. This number represents a balance between computational efficiency and the stochastic nature of the learning process. The number of epochs, set at 25, represents the number of times the entire training dataset will be passed forward and backward through the neural network.

The training process starts with the fit function. Here, X_train and y_train are the training images and labels respectively. The training proceeds with a batch size of 64 and for 25 epochs as defined earlier. The model's performance is evaluated on the validation dataset, X_test and y_test, after each epoch, providing a view on the model's ability to generalize from the training data to unseen data. The output from this function, including the loss and accuracy of the model after each epoch, is saved to the history variable for potential later analysis.

3.7.7 Image Folder Pre-processing

A function was developed to segment the digits within the images, within this, the image is transformed to grayscale, simplifying the subsequent process of contour detection. Identified contours with areas below a specified threshold are disregarded to ensure meaningful segmentation. The retained contours are sorted based on their x-coordinates, ensuring sequential extraction of digits. For each valid contour, a bounding box is established, and segments surpassing the area criteria are extracted as individual digits. The function ultimately returns both these valid contours and the extracted segments, facilitating further analysis or processing.

3.7.8 Prediction of Numbers

Each segmented is resized to the target size, the pixel values are normalized to the range [0, 1], and dimensions adjusted to be compatible the CRNN model input expectations, specifically by adding channel and batch axes.

After pre-processing the binary digit image, the model predicts its value, and the digit's identity is determined by selecting the index with the highest prediction score.

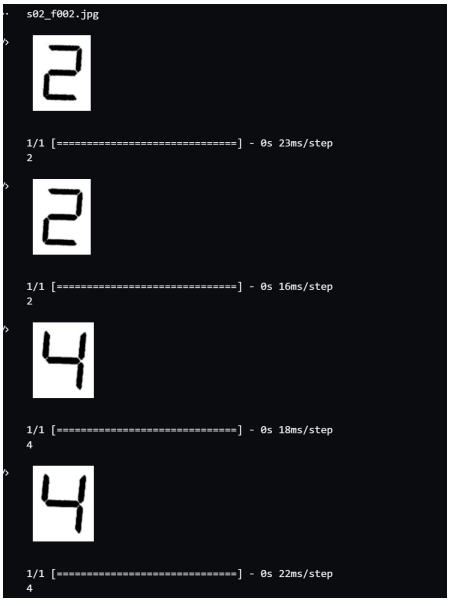


FIGURE 3.38: Image Prediction via CRNN

3.7.9 Conclusion

In this chapter, an exploration of the research methodology has been presented. The primary objective being the enhancement of the performance of Optical Character Recognition (OCR) systems, with a specific focus on Tesseract OCR and Convolutional Recurrent Neural Network (CRNN) models. A systematic approach was employed, initiating a global run of OCR systems on raw image datasets. This highlighted the pivotal role of image pre-processing in augmenting OCR results. The importance of pre-processing, especially the application of colour masks before grayscale conversion, was underscored as a method to improve image clarity and subsequently the efficiency of the OCR processes. Serving as a detailed roadmap, the chapter elucidated each phase of the research methodology, from the initial evaluation of OCR systems to the intricate details of the pre-processing techniques.

Chapter 4

Results

4.1 Introduction

This chapter delves into the empirical findings derived from the systematic approach delineated earlier. Using the raw image datasets subjected to various pre-processing techniques, the performance of the OCR systems will be analysed and presented. The results aim to validate the hypothesis that image pre-processing can significantly enhance the efficacy of OCR systems on sensor reading images. Each section will provide a detailed account of the outcomes, offering insights into the effectiveness of the applied methodologies and setting the stage for the subsequent discussion and analysis.

4.2 First Sprint - Global Generic

4.3 Second Sprint - Global Generic Analysis Resized

4.4 Analysis Tesseract Separate Folders

Table 4.1: OCR Performance for Different Folders

Folder	**Total Count**	**Tesseract**		**CRNN**		
		Read	**Not Read**	**Read**	**Not Read**	**No Contours**
A	160	70	90	46	9	105
B	-	-	-	-	-	-
C	-	-	-	-	-	-

4.4.1 Sipa 2

4.4.2 Sipa 3

4.4.3 Sipa 4

4.4.4 Sipa 5

4.4.5 Sipa 6

4.4.6 Sipa 7

4.4.7 Sipa 8

4.4.8 Sipa 9

4.4.9 Sipa 10

4.4.10 Sipa 11

Chapter 5

Discussion and Conclusion

5.1 Discussion

5.2 Conclusion

Bibliography

[1] R. Smith, "An Overview of the Tesseract OCR Engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. IEEE, pp. 629–633. [Online]. Available: http://ieeexplore.ieee.org/document/4376991/

[2] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models," vol. 37, no. 11, pp. 13 094–13 102. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26538

[3] E. Andersson, "Benchmarking Object Detection Algorithms for Optical Character Recognition of Odometer Mileage." [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1688125&dswid=462

[4] A. Ahuja and A. Chaudhuri, "Detecting Vehicle Type and License Plate Number of different Vehicles on Images."

[5] L. Giridhar, A. Dharani, and V. Guruviah, "A Novel Approach to OCR using Image Recognition based Classification for Ancient Tamil Inscriptions in Temples," p. 8. [Online]. Available: https://arxiv.org/abs/1907.04917

[6] S. Cakic, T. Popovic, S. Sandi, S. Krco, and A. Gazivoda, "The Use of Tesseract OCR Number Recognition for Food Tracking and Tracing," in *2020 24th International Conference on Information Technology (IT)*. IEEE, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9070558/

[7] G. A. Robby, A. Tandra, I. Susanto, J. Harefa, and A. Chowanda, "Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application," vol. 157, pp. 499–505. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050919311640

[8] P. Verma and G. M. Foomani, "Improvement in OCR Technologies in Postal Industry Using CNN-RNN Architecture: Literature Review," vol. 12, no. 5. [Online]. Available: http://www.ijmlc.org/index.php?m=content&c=index&a=show&catid=125&id=1291

[9] A. Biró, A. I. Cuesta-Vargas, J. Martín-Martín, L. Szilágyi, and S. M. Szilágyi, "Synthetized Multilanguage OCR Using CRNN and SVTR Models for Realtime Collaborative Tools," vol. 13, no. 7, p. 4419. [Online]. Available: https://www.mdpi.com/2076-3417/13/7/4419

[10] S. Shinde, T. Saraiya, J. Jain, and C. Narvekar, "Using CRNN to Perform OCR over Forms," vol. 9, no. 3.

[11] S. Rawls, H. Cao, J. Mathai, and P. Natarajan, "How To Efficiently Increase Resolution in Neural OCR Models," in *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*. IEEE, pp. 140–144. [Online]. Available: https://ieeexplore.ieee.org/document/8480182/

[12] X. Feng, Z. Wang, and T. Liu, "Port Container Number Recognition System Based on Improved YOLO and CRNN Algorithm," in *2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*. IEEE, pp. 72–77. [Online]. Available: https://ieeexplore.ieee.org/document/9221498/

[13] A. Azadbakht, S. R. Kheradpisheh, and H. Farahani, "MultiPath ViT OCR: A Lightweight Visual Transformer-based License Plate Optical Character Recognition," in *2022 12th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, pp. 092–095. [Online]. Available: https://ieeexplore.ieee.org/document/9960026/

[14] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-Performance OCR for Printed English and Fraktur Using LSTM Networks," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, pp. 683–687. [Online]. Available: http://ieeexplore.ieee.org/document/6628705/

[15] P. Li, L. Peng, J. Cai, X. Ding, and S. Ge, "Attention Based RNN Model for Document Image Quality Assessment," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 819–825. [Online]. Available: http://ieeexplore.ieee.org/document/8270070/

[16] I. A. Doush, F. Alkhateeb, and A. H. Gharaibeh, "A novel Arabic OCR post-processing using rule-based and word context techniques," vol. 21, no. 1-2, pp. 77–89. [Online]. Available: http://link.springer.com/10.1007/s10032-018-0297-y

[17] I. O. Joshua, M. O. Arowolo, M. O. Adebiyi, O. R. Oluwaseun, and K. A. Gbolagade, "Development of an Image Processing Techniques for Vehicle Classification Using OCR and SVM," in *2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*. IEEE, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/10124622/

[18] S. F. Rashid, F. Shafait, and T. M. Breuel, "An Evaluation of HMM-Based Techniques for the Recognition of Screen Rendered Text," in *2011 International Conference on Document Analysis and Recognition*. IEEE, pp. 1260–1264. [Online]. Available: http://ieeexplore.ieee.org/document/6065512/

[19] T. K. Hazra, D. P. Singh, and N. Daga, "Optical character recognition using KNN on custom image dataset," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*. IEEE, pp. 110–114. [Online]. Available: http://ieeexplore.ieee.org/document/8079572/

[20] M. A. Hossain and S. Afrin, "Optical Character Recognition based on Template Matching," pp. 31–35. [Online]. Available: https://globaljournals.org/GJCST_Volume19/4-Optical-Character-Recognition.pdf

[21] M. Ĉadík, "Perceptual Evaluation of Color-to-Grayscale Image Conversions," vol. 27, no. 7, pp. 1745–1754. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01319.x

[22] R. G. Garcia, R. C. M. Balaba, E. M. V. Ebro, and R. C. D. Ilagan, "Detection and Classification of Pathogens in Gram-Stained Dairy Cow Milk Using Otsu Method," in *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pp. 1–6.

[23] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," vol. PAMI-9, no. 4, pp. 532–550.

[24] "Tessdata," tesseract-ocr. [Online]. Available: https://github.com/tesseract-ocr/tessdata

[25] A. Das, M. Wasif Ansari, and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," in *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1–5.

[26] OpenCV: Denoising. [Online]. Available: https://docs.opencv.org/4.x/d1/d79/group__photo__denoise.html

[27] J. Priyadarshini and G. CharlynPushpaLatha, "Comparative Performance Analysis for Maximum Segmented Accuracy in Voice Stammer using Wiener Filter and Gaussian Filter Recognition," in *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp. 680–684.

[28] P. Mukhopadhyay and B. B. Chaudhuri, "A survey of Hough Transform," vol. 48, no. 3, pp. 993–1010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314003446

[29] Mnist — TensorFlow Datasets. [Online]. Available: https://www.tensorflow.org/datasets/catalog/mnist

[30] The Street View House Numbers (SVHN) Dataset. [Online]. Available: http://ufldl.stanford.edu/housenumbers/

[31] WhatTheFont — MyFonts. [Online]. Available: https://www.myfonts.com/pages/whatthefont?step=crop

[32] Browse Fonts - Google Fonts. [Online]. Available: https://fonts.google.com/

Appendix A

Appendix A

A.1 Introduction

A.2 Analysis Tesseract Separate Folders

FIGURE A.1: Image Folder A Count Analysis

The chart presented primarily emphasizes the blue line, which signifies the count of Masked Red OTSU. This line is of particular importance. At size 104, we observe the peak read count.

This establishes the dimensions to which the images will be resized for each of the subsequent directories.

A.2.1 Sipa 2 Contrast Analysis on MRO SSD

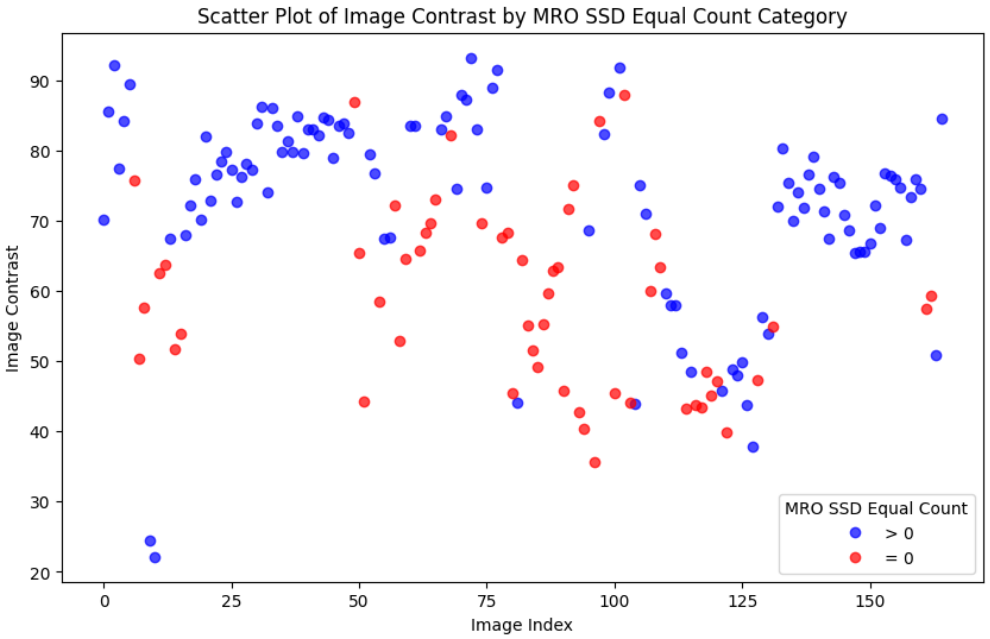


FIGURE A.2: Sipa 2 Contrast Analysis on MRO SSD

The scatter plot of image contrast shows that there is a wide range of contrast levels for both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0). This means that the contrast of images in both categories is highly variable. Additionally, there is significant overlap in the contrast values of both categories, suggesting that the contrast of an image may not be a strong indicator of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

There are a few outliers in the MRO SSD EQUAL COUNT greater than 0 category with particularly high contrast levels. However, these outliers do not significantly change the overall pattern of the scatter plot.

In summary, the scatter plot of image contrast does not provide any clear evidence that the contrast of an image is a good predictor of the MRO SSD EQUAL COUNT value.

A.2.2 Sipa 2 Brightness Analysis on MRO SSD

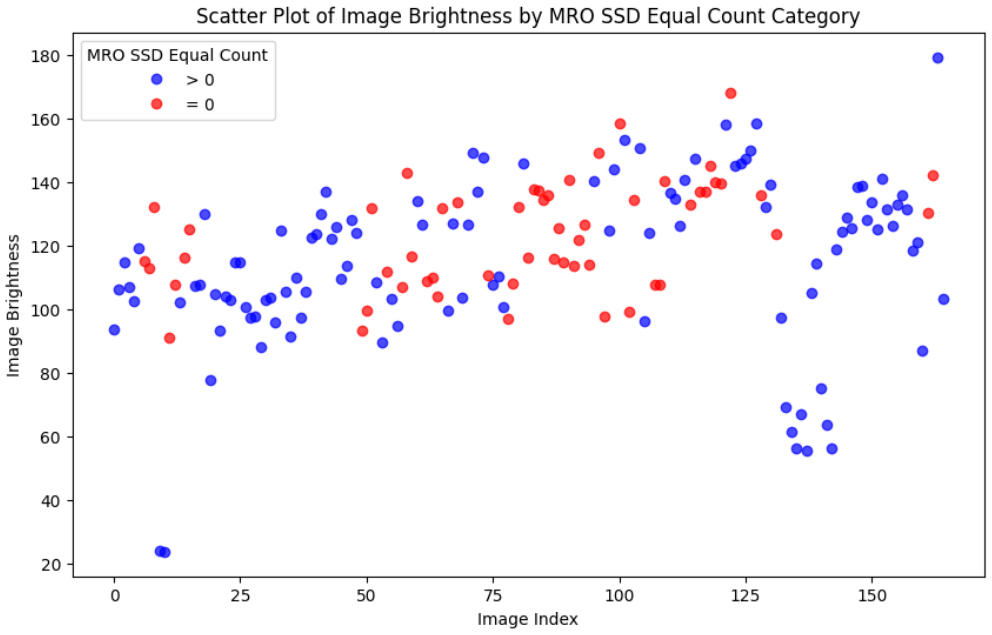


FIGURE A.3: Sipa 2 Brightness Analysis on MRO SSD

The scatter plot of image brightness shows that both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0) have a wide range of brightness values. This means that the brightness of images in both categories is highly variable. Additionally, there is no clear pattern or correlation between the MRO SSD EQUAL COUNT category and the brightness of the images. This suggests that the brightness of an image may not be a good predictor of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

There are a few outliers in the MRO SSD EQUAL COUNT greater than 0 category with particularly high brightness levels. However, these outliers do not significantly change the overall pattern of the scatter plot.

In summary, the scatter plot of image brightness does not provide any clear evidence that the brightness of an image is a good predictor of the MRO SSD EQUAL COUNT value.

A.2.3 Sipa 2 Standard Deviation Analysis on MRO SSD

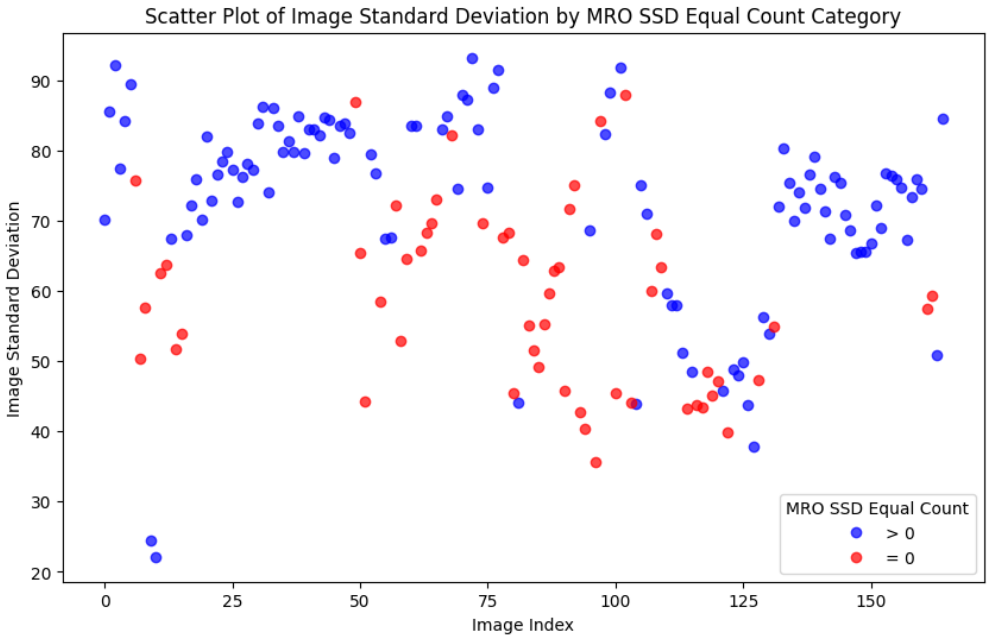


FIGURE A.4: Sipa 2 Standard Deviation Analysis on MRO SSD

The scatter plot of image standard deviation shows that there is a wide range of standard deviations for both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0). This means that the spread of pixel values within the images is highly variable. Additionally, there is significant overlap in the standard deviation values of both categories, suggesting that the standard deviation of pixel values within an image may not be a strong indicator of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

A.2.4 Sipa 2 Homogeneity Analysis on MRO SSD

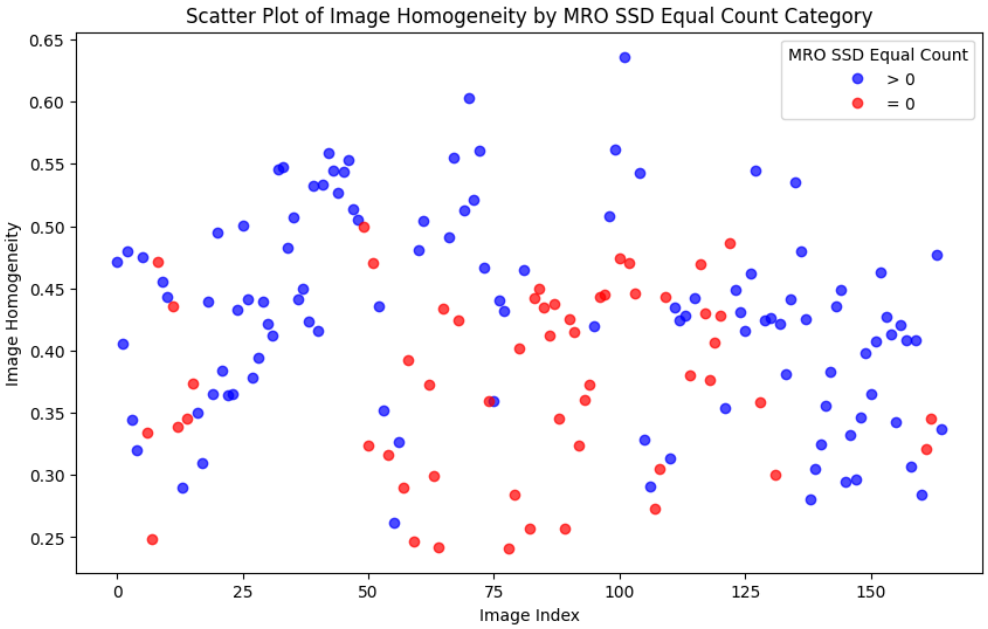


FIGURE A.5: Sipa 2 Homogeneity Analysis on MRO SSD

The scatter plot of image Homogeneity shows that there is a wide range of homogeneity levels for both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0). This signifies that the homogeneity of images in both categories is highly variable. Additionally, there is significant overlap in the homogeneity values of both categories, suggesting that the homogeneity of an image may not be a strong indicator of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

In summary, the scatter plot of image homogeneity does not provide any clear evidence that the homogeneity of an image is a good predictor of the MRO SSD EQUAL COUNT value.

A.2.5 Sipa 2 Sharpness Analysis on MRO SSD

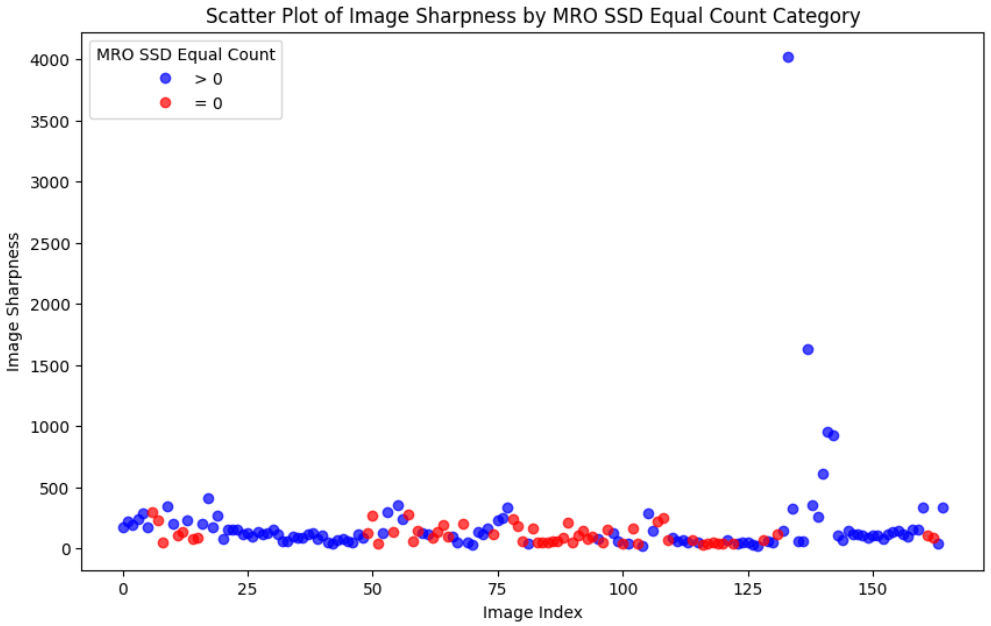


FIGURE A.6: Sipa 2 Sharpness Analysis on MRO SSD

The scatter plot of image Sharpness shows that there is a narrow range of sharpness levels for both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0). This signifies that the sharpness of images in both categories is reasonably uniform. Additionally, there is significant overlap in the sharpness values of both categories, suggesting that the sharpness of an image may not be a strong indicator of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

In summary, the scatter plot of image sharpness does not provide any clear evidence that the sharpness of an image is a good predictor of the MRO SSD EQUAL COUNT value.

A.2.6 Sipa 2 Dissimilarity Analysis on MRO SSD

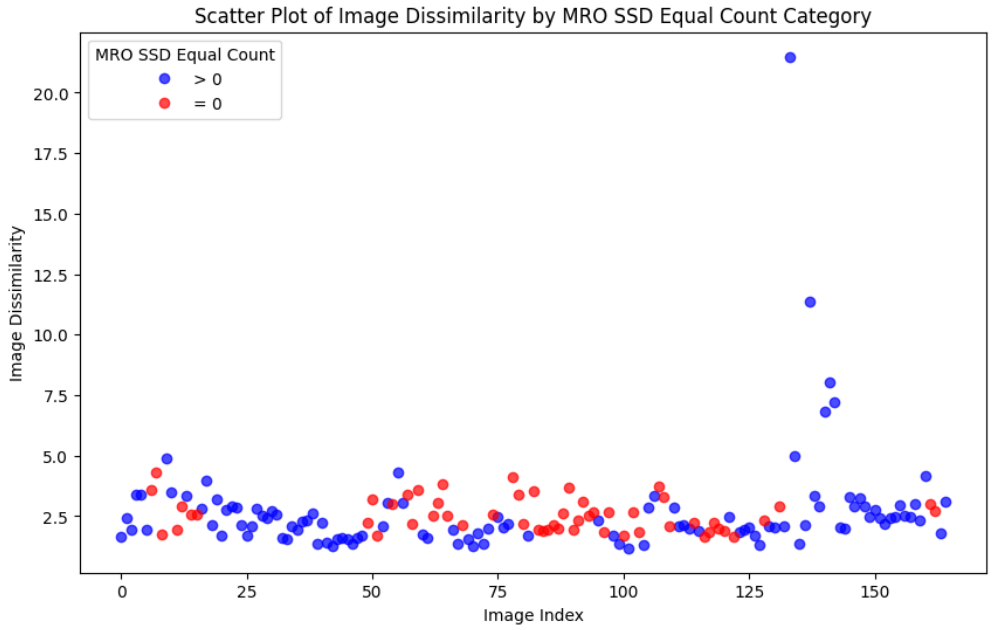


FIGURE A.7: Sipa 2 Dissimilarity Analysis on MRO SSD

The scatter plot of image Dissimilarity shows that there is a narrow range of dissimilarity levels for both categories of images (MRO SSD EQUAL COUNT greater than 0 and equal to 0). This signifies that the dissimilarity of images in both categories is reasonably uniform. Additionally, there is significant overlap in the dissimilarity values of both categories, suggesting that the dissimilarity of an image may not be a strong indicator of whether the MRO SSD EQUAL COUNT is greater than 0 or not.

In summary, the scatter plot of image dissimilarity does not provide any clear evidence that the dissimilarity of an image is a good predictor of the MRO SSD EQUAL COUNT value.

A.2.7 Sipa 2 Area Analysis on MRO SSD

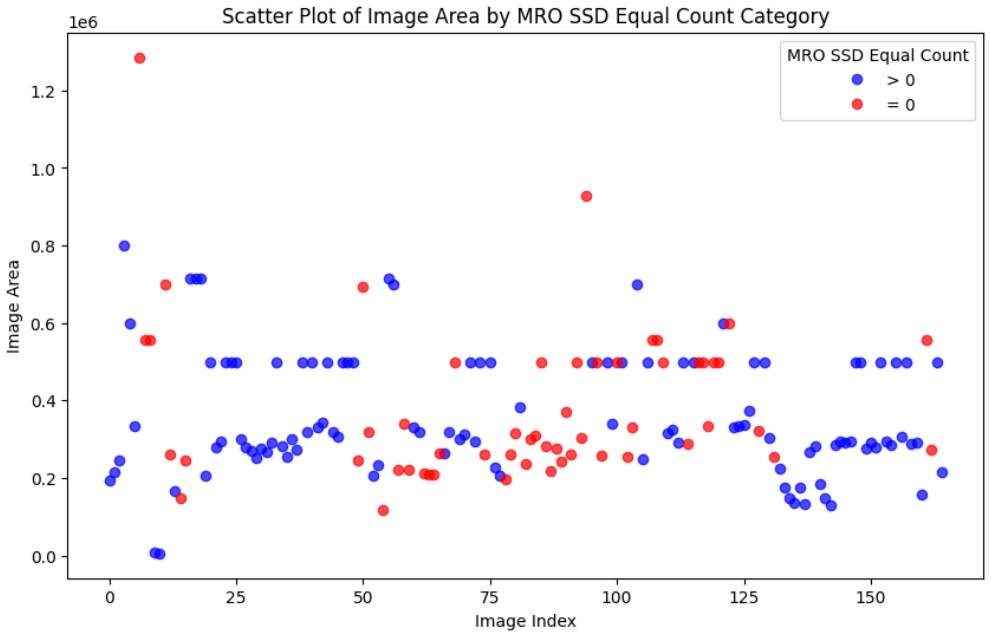


FIGURE A.8: Sipa 2 Area Analysis on MRO SSD

The distribution of image area values for both categories (¿ 0 and = 0) is wide. However, the distribution appears to be slightly different for the two categories, with a higher density of points at lower image area values for the = 0 category.

There is a significant density of points near the lower end of the image area values for both categories. The = 0 category seems to have a higher density of points at lower image area values.

There is considerable overlap between the two categories, especially at lower image area values. This suggests that while there may be a trend, the MRO SSD EQUAL COUNT value is not a definitive predictor of the image area value.

There appear to be some outliers in the ¿0 category with high image area values. This suggests that there may be some instances where MRO SSD EQUAL COUNT is greater than 0 and the image area is significantly larger than the typical values.

In summary, this scatter plot suggests that there might be a slight tendency for instances with MRO SSD EQUAL COUNT of 0 to have smaller image areas. However, the relationship is not clear-cut, as there is significant overlap between the two categories.

A.2.8 Sipa 2 Correlation Analysis on MRO SSD

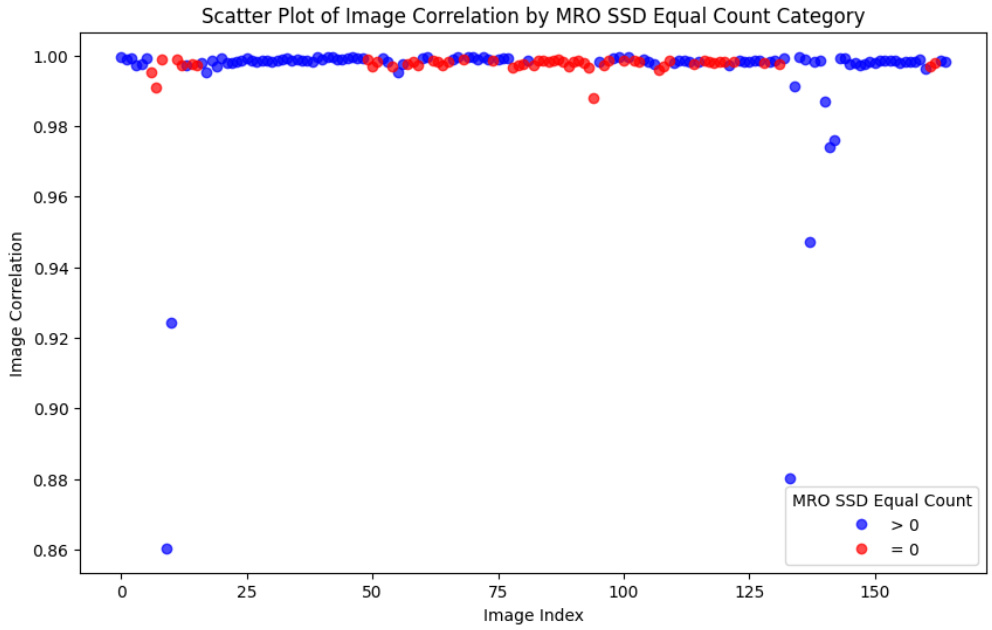


FIGURE A.9: Sipa 2 Correlation Analysis on MRO SSD

The distribution of image correlation values for both categories (¿ 0 and = 0) is varied, similar to the previous plot.

The density of points for both categories appears to be quite uniform across the range of image correlation values. There is perhaps a slightly higher density at lower correlation values for the = 0 category.

There is significant overlap between the two categories across the entire range of image correlation values. This suggests that the MRO SSD EQUAL COUNT value does not strongly distinguish between different levels of image correlation.

There do not seem to be any clear outliers in this plot, unlike the previous plot for Image Energy.

In summary, while there might be a slight tendency for instances with MRO SSD EQUAL COUNT of 0 to have lower image correlation, the relationship is not clear-cut. There is substantial overlap between the two categories, indicating that MRO SSD EQUAL COUNT may not be a strong predictor for image correlation.

A.2.9 Sipa 2 Energy Analysis on MRO SSD

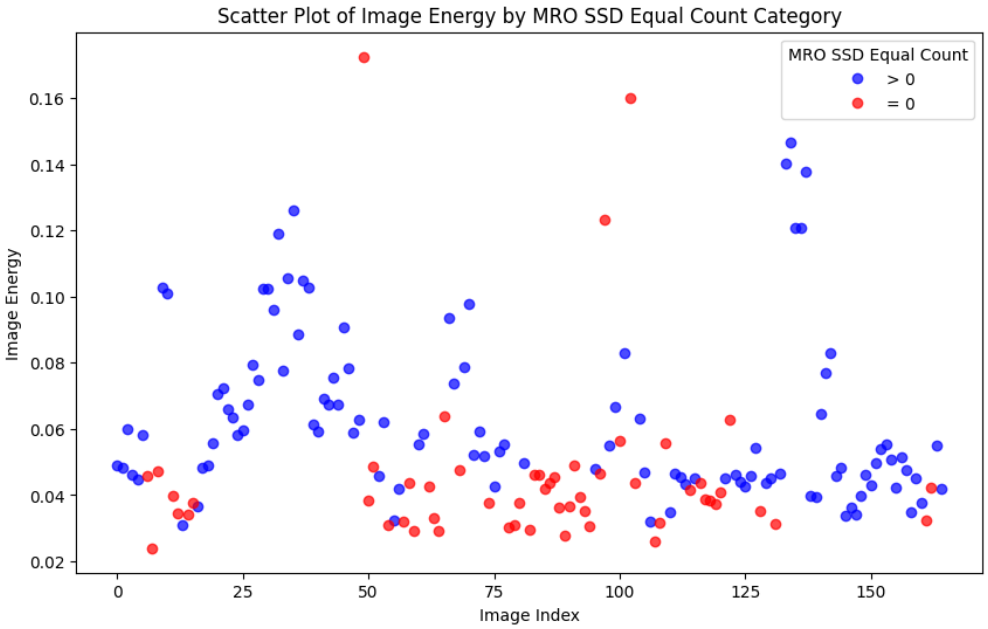


FIGURE A.10: Sipa 2 Energy Analysis on MRO SSD

The distribution of Image Energy values for both categories (¿ 0 and = 0) is quite scattered. This suggests that the distribution of energy values is varied for both cases.

There appears to be a higher density of red points (= 0 category) towards the lower Image Energy values. This suggests that when MRO SSD EQUAL COUNT is zero, the image energy tends to be lower.

There are some apparent outliers in the blue category (gt 0), with significantly higher Image Energy values. This suggests that there might be a few instances with MRO SSD EQUAL COUNT greater than 0 that have unusually high image energy.

There is considerable overlap between the two categories. This indicates that while there might be a general trend (as indicated in points 2 and 3), the MRO SSD EQUAL COUNT value is not a definitive predictor of the Image Energy value.

```
    def hello_world():
    print("Hello, world!")
```

LISTING A.1: Python example

the above code is a python code snippet.