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

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 Pre-processing" 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.

A unique feature of the research lies in its novel image pre-processing steps, specifically the masking of red and green colours 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 pre-processing 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 pre-processing 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

API	**A**pplication **P**rogramming **I**nterface
AWGN	**A**dditive **W**hite **G**aussian **N**oise
CNN	**C**onvolutional **N**eural **N**etwork
CRNN	**C**onvolutional **R**ecurrent **N**eural **N**etwork
CSV	**C**omma **S**eperated **V**alues
CTC	**C**onnectionist **T**emporal **C**lassification
DIQA	**D**ocument **I**mage **Q**uality **A**ssessment
ENG	**ENG**lish
FAHTA	**F**ast **A**utomatic **T**ext **A**lignment
HMM	**H**idden **M**arkov **M**odels
HP	**H**ewlett-**P**ackard
HSV	**H**ue, **S**aturation, and **V**alue
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
KNN	**K**-**N**earest **N**eighbour
LSTM	**L**ong **S**hort-**T**erm **M**emory
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
OCR	**O**ptical **C**haracter **R**ecognition
OpenCV	**Open** **S**ource **C**omputer **V**ision **L**ibrary
PDF	**P**ortable **D**ocument **F**ormat
PIL	**P**ython **I**maging **L**ibrary
PSM	**P**age **S**egmentation **M**ode
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
ReLU	**R**ectified **L**inear **U**nit
RGB	**R**ed **G**reen **B**lue
RNN	**R**ecurrent **N**eural **N**etwork

SSD	**S**ingle **S**hot **M**ultiBox **D**etector
SVHN	**S**treet **V**iew **H**ouse **N**umbers
SVTR	**S**ingle **V**isual model for **S**cene **T**ext **R**ecognition
SVM	**S**upport **V**ector **M**achine
UNLV	**U**niversity of **N**evada, **L**as **V**egas
YOLO	**Y**ou **O**nly **L**ook **O**nce

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Area of Interest

Optical Character Recognition (OCR) is a technology that converts different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data. In simpler terms, it enables computers to "read" and understand text contained within images. This capability has been instrumental in numerous applications, including digitizing ancient manuscripts, automating data entry processes, and even translating written text in real-time using mobile devices. With the advent of deep learning and advancements in computer vision, OCR technology has experienced significant improvements in accuracy and versatility.

Deep learning is a subfield of machine learning inspired by the structure and function of the brain, particularly neural networks. It utilizes multiple layers of artificial neural networks to analyse various factors of data, making it exceptionally powerful for tasks like image and speech recognition. As a cornerstone of modern artificial intelligence, deep learning models can automatically learn representations from data without manual feature extraction, leading to ground-breaking advancements in various fields, including computer vision, natural language processing, and robotics.

The area of interest for this literature review is the intersection of computer vision, 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.

TABLE 1.1: Nimbus Sensor Images

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 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 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, embedded often within crucial and expensive equipment, are accurate and indispensable. Given their extensive presence in industrial environments, many of these sensors have been in operation for long periods, making their upgrade neither straightforward nor essential. However, they 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 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 OCR on images of sensor readings by applying novel pre-processing 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 OCR Methods**

   The objective here is to develop a comprehensive literature review of various 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]

This literature review delves into research papers on Tesseract OCR, emphasizing its accuracy, performance across various document types, and specific applications. The chosen studies offer profound insights into Tesseract OCR's current landscape and future trajectory. By diving deep into these nuances, we seek to harness its transformative potential in our digital information era.

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 rectangular shapes instead of polygons, and the categories were 2-Wheelers, 3-Wheelers,

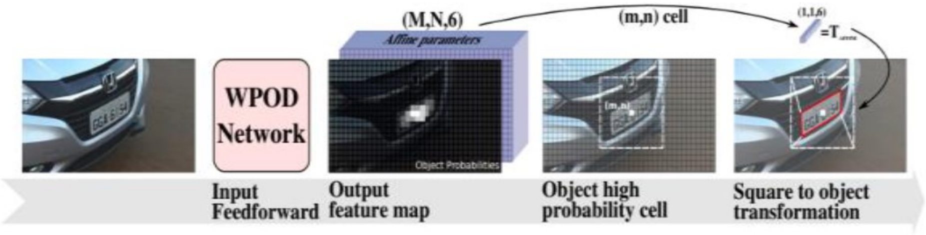


FIGURE 2.3: WpodNet Detection Method
[4]

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 ancient Tamil characters. To implement the optical character recognition techniques,

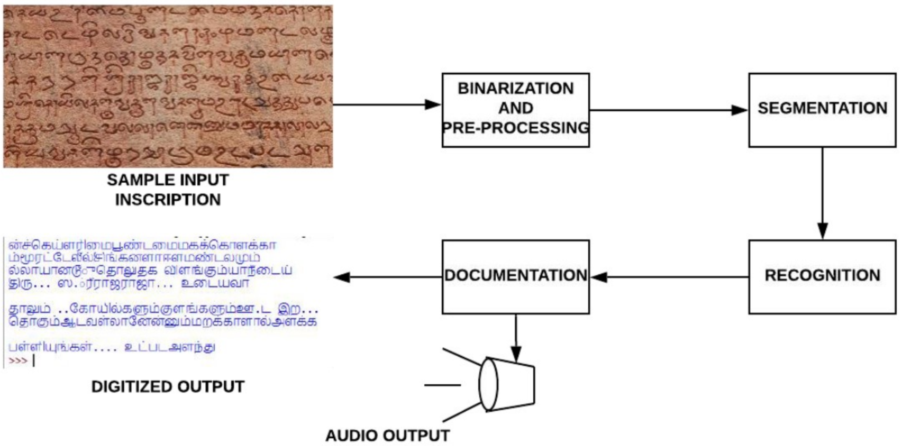


FIGURE 2.4: Giridhar's proposed architecture
[5]

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.

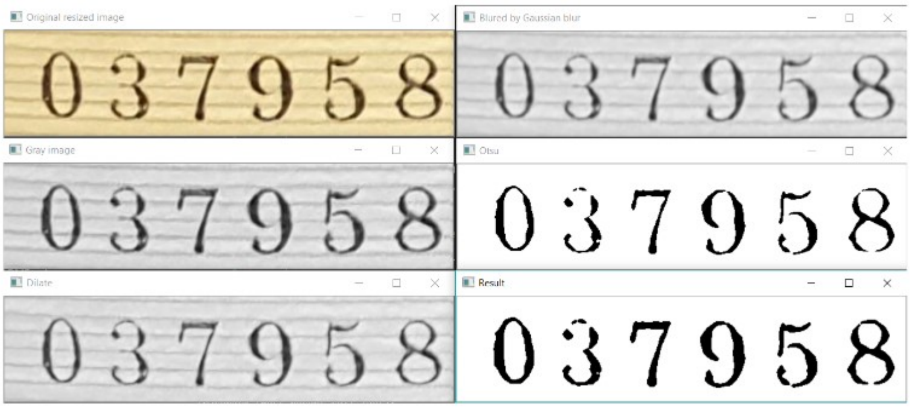


Figure 2.5: Numbers change through the pre-processing stage
[6]

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.

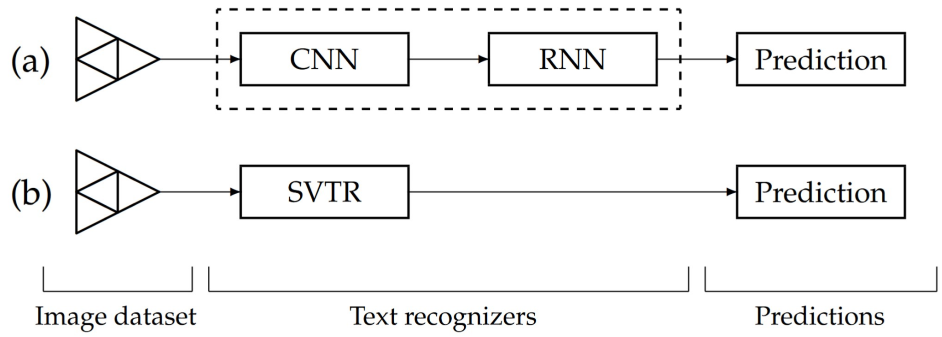


FIGURE 2.7: Biro et al.'s Modern text recognizers using (a) CRNN and (b) SVTR models
[9]

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.

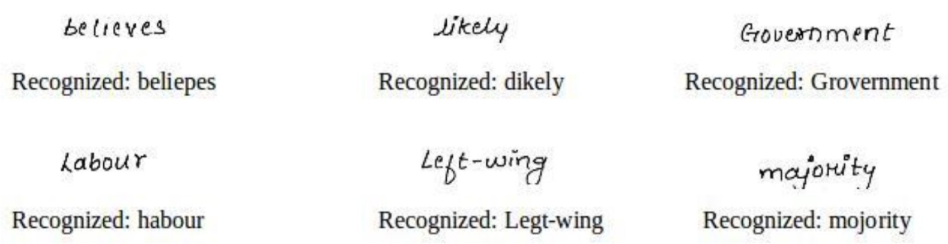


Figure 2.8: Shinde et al.'s Handwriting Results
[10]

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.

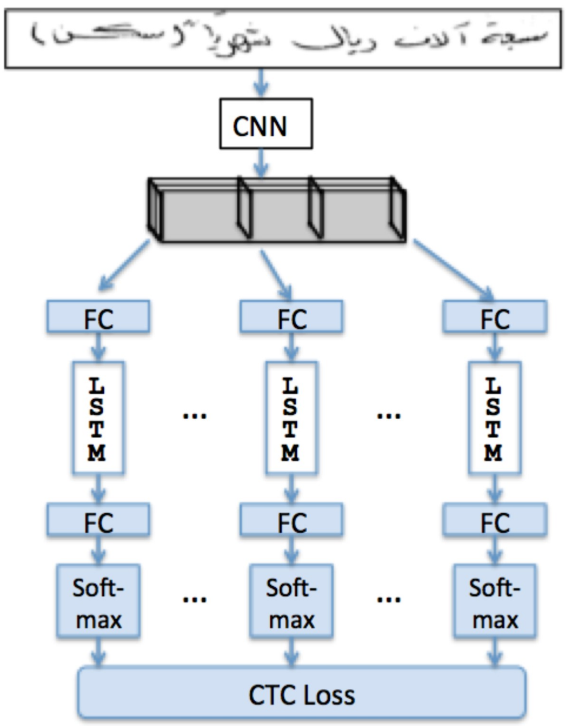


FIGURE 2.9: Rawl et al.'s End-to-End OCR Model: With CNN and LSTM layers [11]

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.

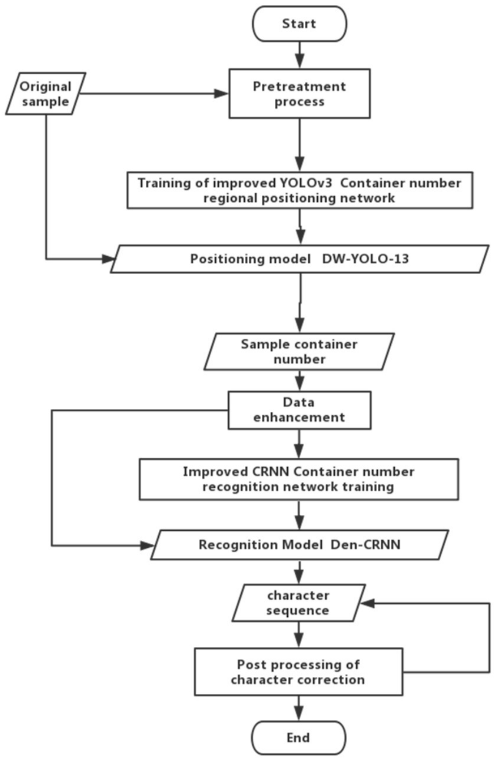


FIGURE 2.10: Feng et al.'s Algorithm Structure
[12]

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.

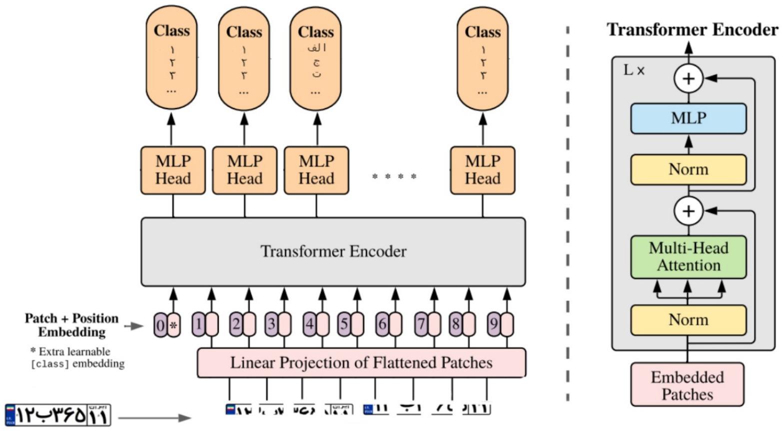


FIGURE 2.11: Azadbakht et al.'s MultiPath Visual Transformer OCR architecture [13]

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.

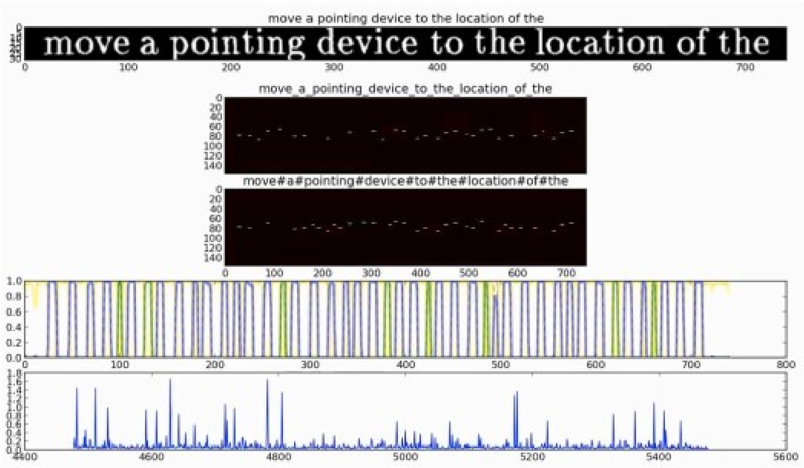


FIGURE 2.12: Greyscale image with background cleaning
[14]

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.

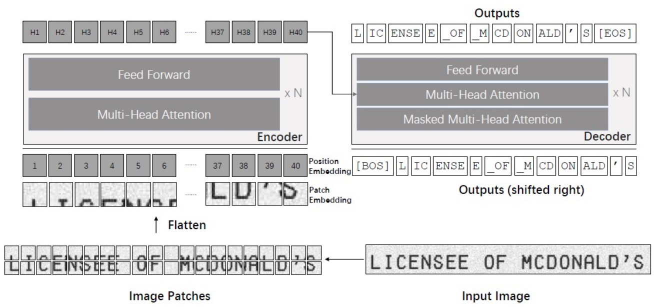


FIGURE 2.13: Li's Architecture of TrOCR [2]

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]

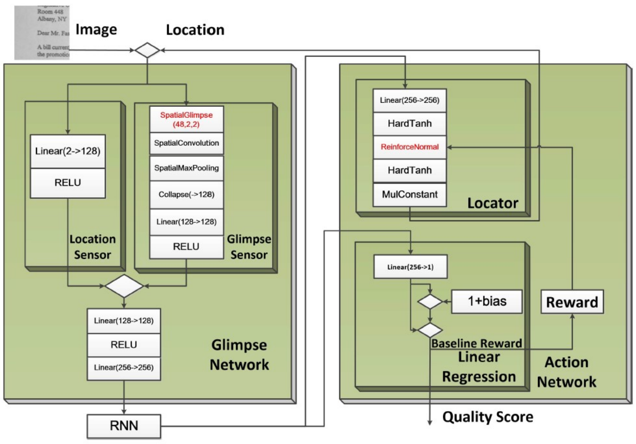


FIGURE 2.14: Architecture of Li's proposed network [15]

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]

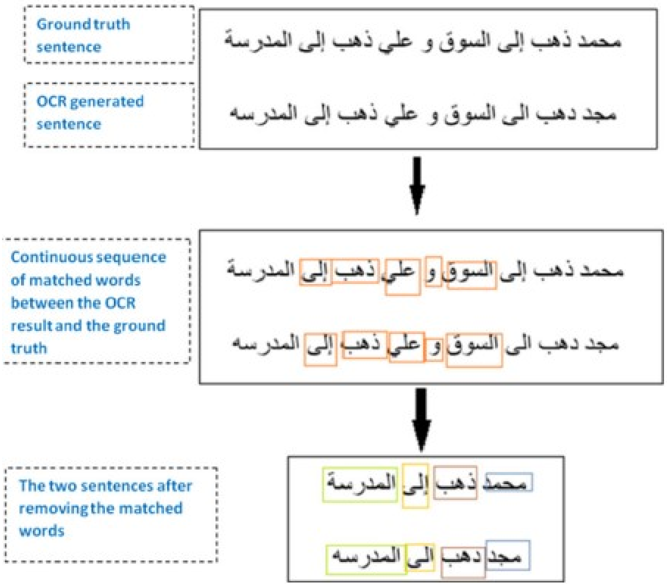


FIGURE 2.15: Example of applying the Rule Based FAHTA Algorithm [16]

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]

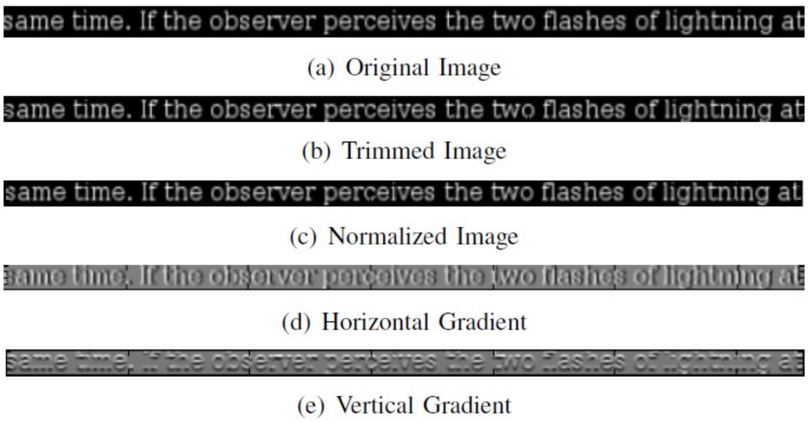
Joshua et al. describe the steps of their proposed system, which include image pre-processing, feature extraction, OCR, and SVM classification. They also explain how they collected and labelled their dataset of Nigerian vehicle plate numbers.

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.



(a) Original Image

(b) Trimmed Image

(c) Normalized Image

(d) Horizontal Gradient

(e) Vertical Gradient

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]

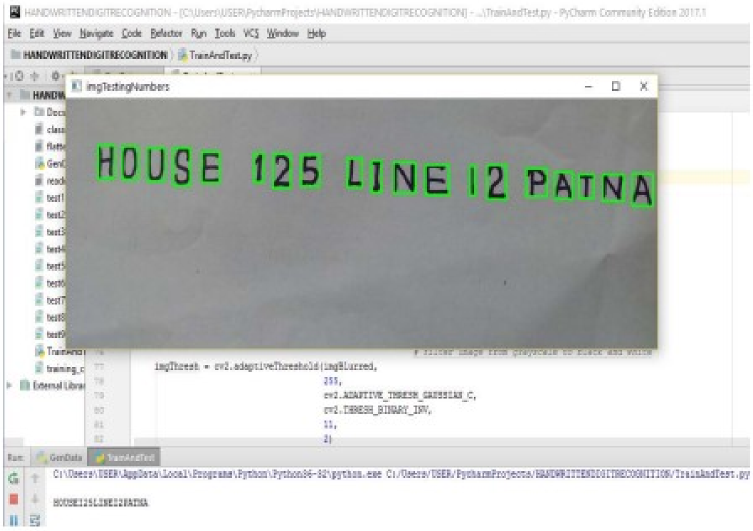


FIGURE 2.18: Characters and Digits recognised
[17]

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.

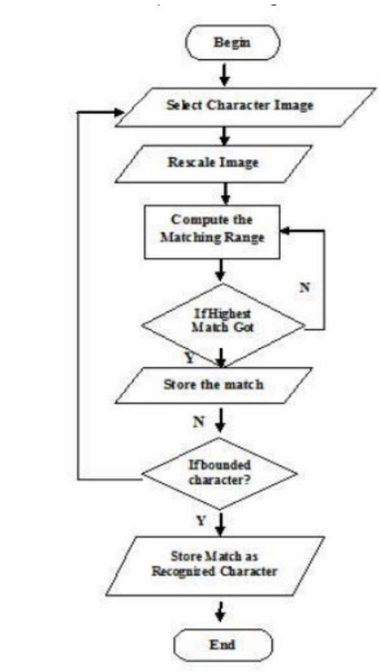


FIGURE 2.19: Flowchart of Hossain's TM OCR [20]

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. Prior to sophisticated models like Tesseract, rule-based OCR systems were common. While these rule-based systems were efficient for specific, well-defined cases, they often lacked the flexibility and generalizability needed for diverse and complex OCR tasks.

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 is derived from actual industrial meters and not from simulated data. It was supplied as a collection of image files distributed across eight distinct folders. Each folder corresponds to a unique sensor from which readings were taken. These real-world images provide a diversified dataset due to variations in sensor specifications and the conditions under which the readings were captured, underscoring their importance and value for genuine, practical applications.

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.

To streamline data management and facilitate the analysis process, a CSV file was compiled. This file serves as an inventory, cataloguing each image file name alongside its corresponding label, thereby establishing an efficient cross-referencing system for the ensuing data analysis phase.

To foster the training of the Convolutional Recurrent Neural Network (CRNN) models, several training databases were established. Each of these databases comprises 500,000 single-digit training images, laying a solid groundwork 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.

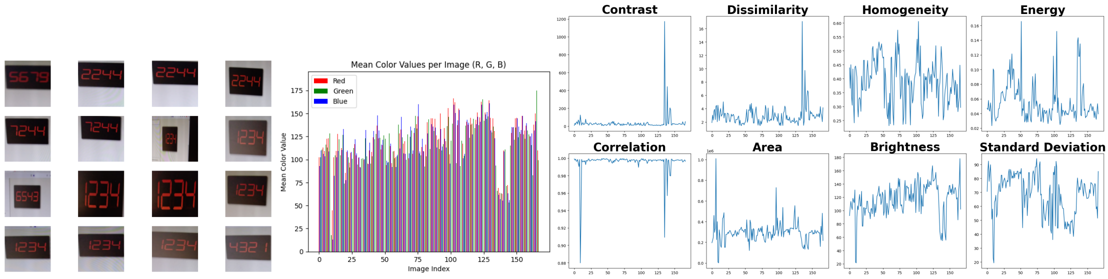
    (f) **Area:** The Area chart, in our context, represents the total area of contours detected in an image, providing information on the complexity of sensor readings.

    (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 Image Folder A. The images are of varying dimensions.

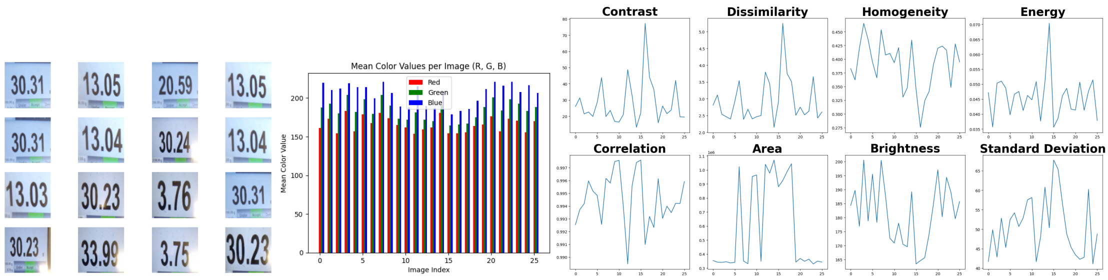


Montage RGB Data Analysis

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 Image Folder B. The images are of varying dimensions.

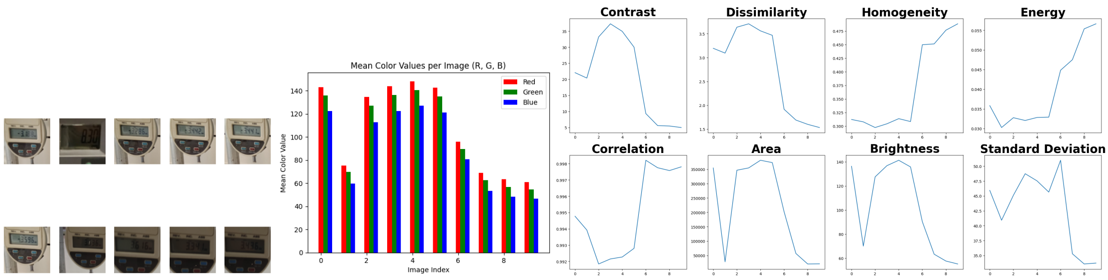


Montage RGB Data Analysis

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 Image Folder C. The images are of varying dimensions.

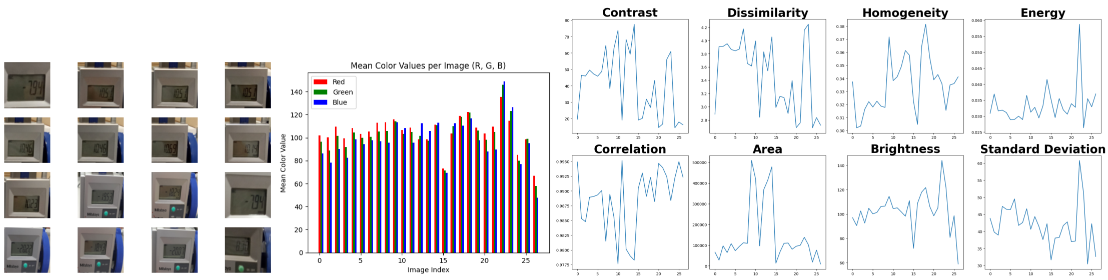


Montage RGB Data Analysis

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 Image Folder D. The images are of varying dimensions.

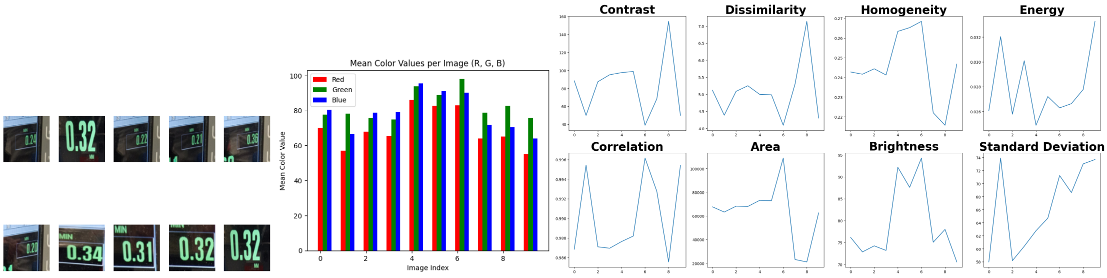


Montage RGB Data Analysis

FIGURE 3.4: Image Folder D Analysis

3.3.5 Image Folder E

There are 10 JPEG files totalling a size of 4.98mb in Image Folder E. The images are of varying dimensions.

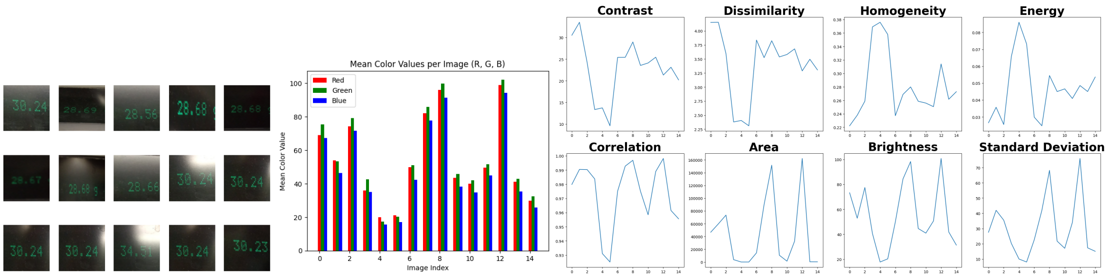


Montage RGB Data Analysis

FIGURE 3.5: Image Folder E Analysis

3.3.6 Image Folder F

There are 15 JPEG files totalling a size of 5.93mb in Image Folder F. The images are of varying dimensions.

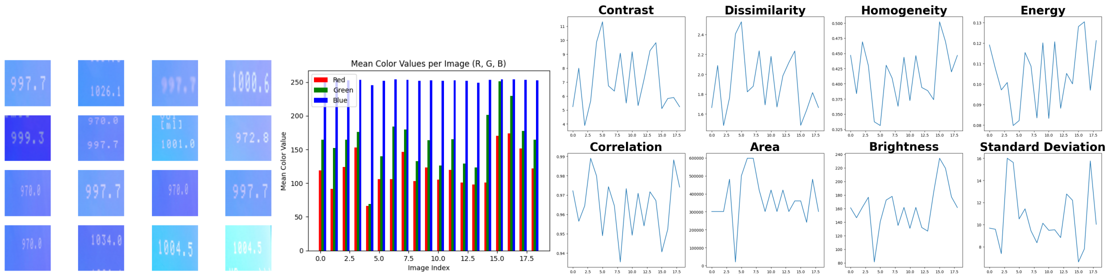


Montage RGB Data Analysis

FIGURE 3.6: Image Folder F Analysis

3.3.7 Image Folder G

There are 12 JPEG files totalling a size of 7.34mb in Image Folder G. The images are of varying dimensions.

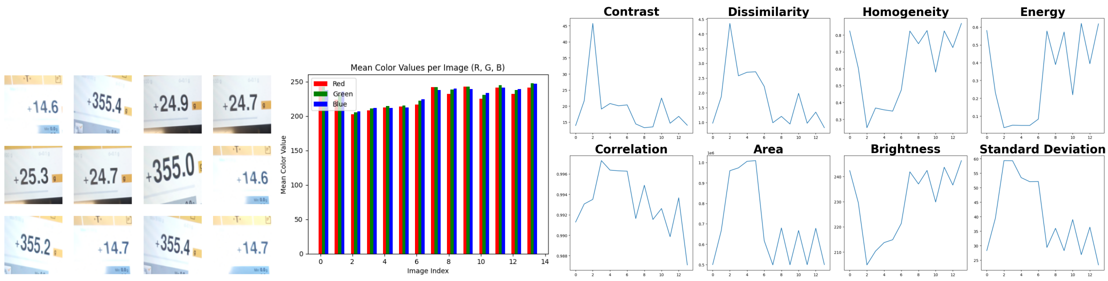


Montage RGB Data Analysis

FIGURE 3.7: Image Folder G Analysis

3.3.8 Image Folder H

There are 14 JPEG files totalling a size of 6.16mb in Image Folder H. The images are of varying dimensions.



Montage RGB Data Analysis

FIGURE 3.8: Image Folder H Analysis

3.4 First Sprint - Global Generic

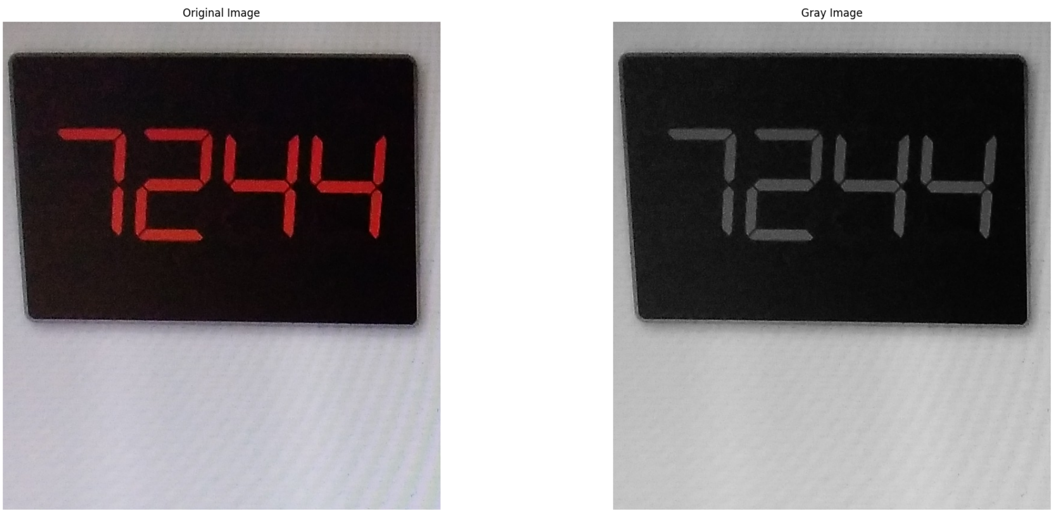
Objective: To successfully perform Optical Character Recognition (OCR) on the entire dataset of images using the pytesseract library, while implementing and evaluating the impact of image pre-processing techniques, specifically the conversion of images to greyscale, to optimize OCR accuracy.

Utilizing the pytesseract library's Python interface for the Tesseract OCR engine, this sprint delves into the practical execution of our objective, focusing on the nuances of image pre-processing.

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

Figure 3.9: 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 3.10: Greyscale Conversion

3.4.1 Conclusion for the First Sprint

In this sprint, we successfully implemented OCR on our dataset using the pytesseract library. The greyscale conversion pre-processing technique was crucial in optimizing the accuracy of the OCR. The specific configurations, as depicted in Fig 3.11, were adjusted to maximize results, with settings of PSM 6, 7, and particularly 13, yielding the most accurate readings. Detailed insights and implications of these findings will be further elaborated in the main Results and Conclusion chapters.

3.5 Second Sprint - Global Generic Analysis Resized

Objective for the Second Sprint: To enhance the accuracy of OCR results from the initial sprint by integrating advanced pre-processing techniques, including Otsu's thresholding, morphological closing, and image resizing. Furthermore, the sprint aims to utilize specialized Tesseract language files, adapting OCR to specific textual representations, thereby ensuring a more tailored and precise text recognition process across diverse image sets.

Building upon the foundation established in the first sprint, the second sprint introduced several pivotal enhancements. Otsu's thresholding and morphological closing were integrated into the pre-processing pipeline. Additionally, a novel resizing technique was employed alongside the inclusion of a seven-segment display language file, collectively augmenting the OCR system's accuracy. Each of these advancements deepens our exploration into the realm of OCR optimization.

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 3.11: 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 3.12: 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 3.13: 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

Conducting OCR on diverse image sets presents challenges due to their wide-ranging attributes, including differences in colour profiles, lighting conditions, text styles, and noise levels. Such variability underscores a key insight: a one-size-fits-all pre-processing pipeline for Tesseract may not be adequate. Embracing a more adaptive and tailored strategy is essential.

The power of OpenCV, an expansive open-source computer vision library, is harnessed in this sprint to provide granular image processing capabilities. By tailoring pre-processing steps for each specific image folder—whether it's grayscale conversion, Gaussian blurring, adaptive thresholding, or even morphological transformations—we're able to cater to the unique needs of each dataset. This meticulous adjustment, although intricate, was pivotal in enhancing OCR accuracy.

The process highlighted not just the intricacies of fine-tuning OCR techniques but also illuminated a key takeaway: the essence of flexibility. Global strategies, while appealing in their broad applicability, may not always produce the desired outcomes. It's often the detailed, customized methods that yield superior results, especially when dealing with diverse datasets.

3.6 Third Sprint - Analysis Tesseract Separate Folders

Objective for the Third Sprint: To further refine the Optical Character Recognition (OCR) capabilities across a variety of image sets by introducing specialized pre-processing techniques tailored to each folder's distinct properties. Techniques such as Red Mask, Green Mask, Deblurring, Thresholding, and Skewness Correction were employed. Additionally, the sprint aimed to address specific challenges presented by each image set, such as diverse colour profiles, skewness, and noise, ensuring the highest possible OCR accuracy.

The diverse attributes of the images in the dataset, ranging from colour profiles to skewness, posed intricate challenges for OCR using Tesseract. Instead of applying a one-size-fits-all approach, this sprint underscored the importance of adaptability. In the ensuing sections, we delve deeper into advanced pre-processing techniques, notably highlighting the Red Mask application via OpenCV. The focus remains on customizing processes for individual image subsets, showcasing the project's commitment to precision and tailored solutions.

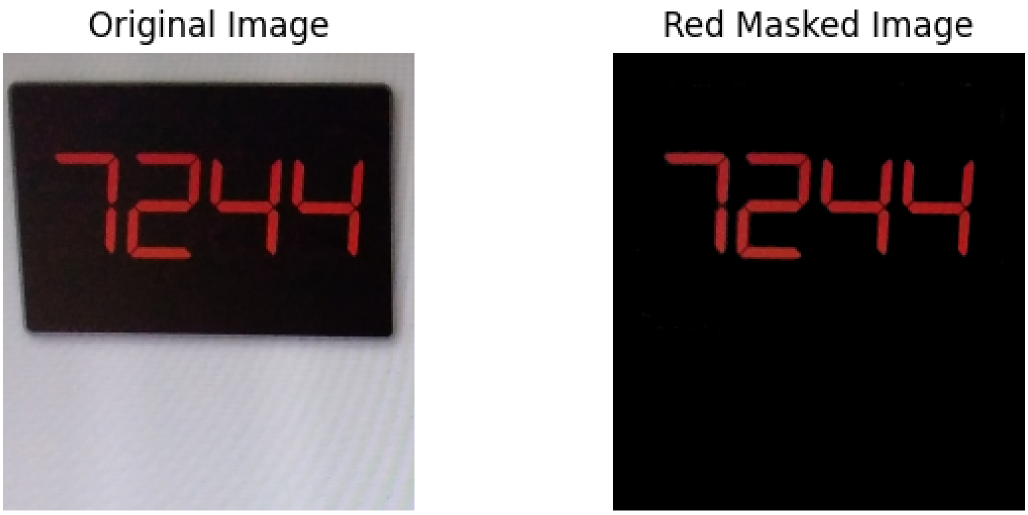
Method - Red Mask

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.

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



Original Image Red Mask Image

FIGURE 3.14: Red Mask

pixel values of 255 where the original image pixels are within the specified red range, and 0 otherwise.

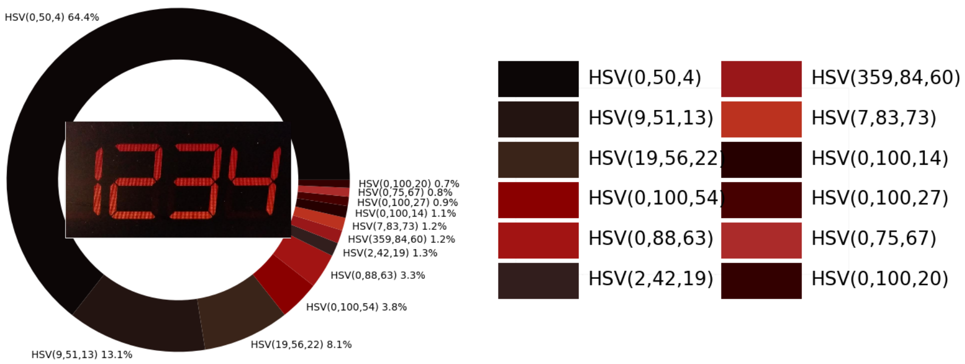


FIGURE 3.15: Colour Analysis

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 3.16: 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. After 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
30	0	['./sipaim...	678	104	NaN	NaN	678	578

FIGURE 3.17: 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 3.18: 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 3.19: 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 3.20: 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 3.21: 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 3.22: 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 3.23: 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.

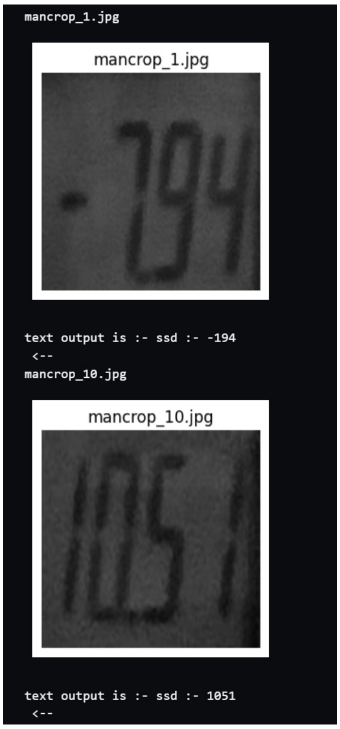


Figure 3.24: Image Folder D Sample Output

3.6.5 Image Folder E

FIGURE 3.25: Image Folder E 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 3.26: 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 E 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 Seven Segment Display (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.

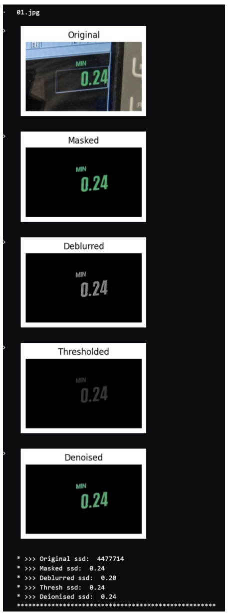


Figure 3.27: Image Folder E Sample Output

3.6.6 Image Folder F

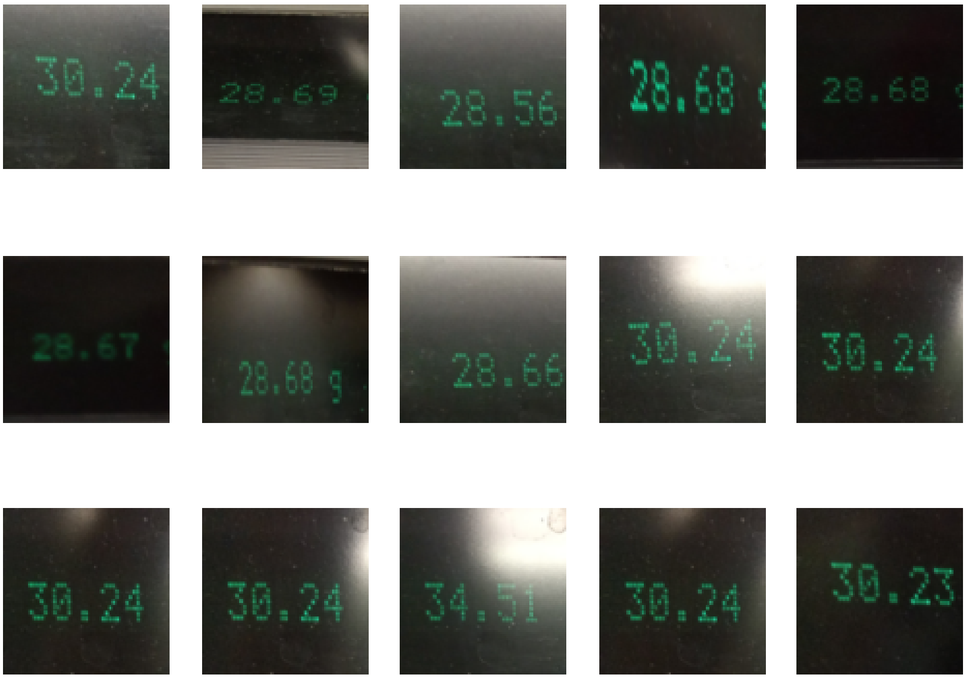


FIGURE 3.28: Image Folder F Montage

The methodology for Image Folder F 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.

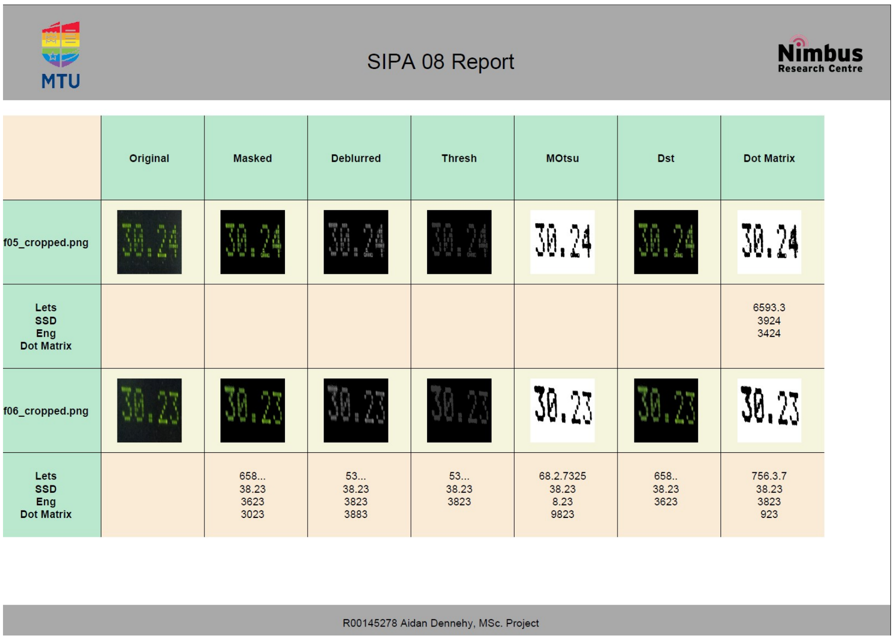
3. **Grayscale**: Convert the masked image to grayscale. This step simplifies the image and is a common requirement for many image processing algorithms.

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.


MTU

SIPA 08 Report

Nimbus
Research Centre

	Original	Masked	Deblurred	Thresh	MOtsu	Dst	Dot Matrix
f05_cropped.png							
Lets SSD Eng Dot Matrix							6593.3 3924 3424
f06_cropped.png							
Lets SSD Eng Dot Matrix		658... 38.23 3623 3023	53... 38.23 3823 3883	53... 38.23 3823	68.2.7325 38.23 8.23 9823	658.. 38.23 3623	756.3.7 38.23 3823 923

R00145278 Aidan Dennehy, MSc. Project

FIGURE 3.29: Image Folder F Sample Output

3.6.7 Image Folder G

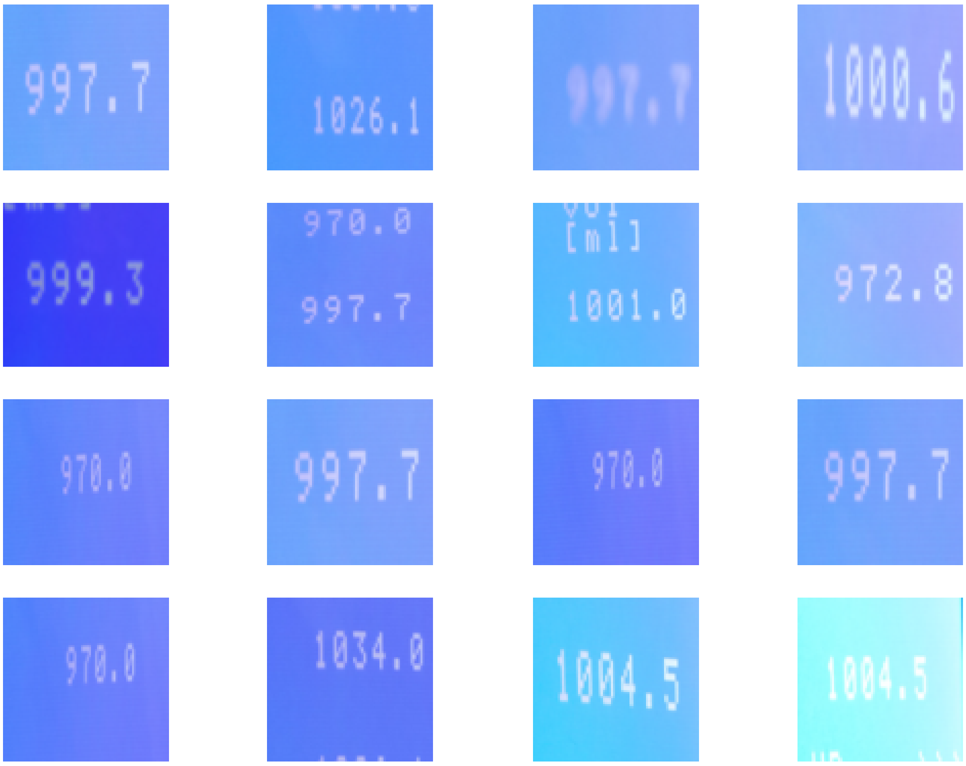


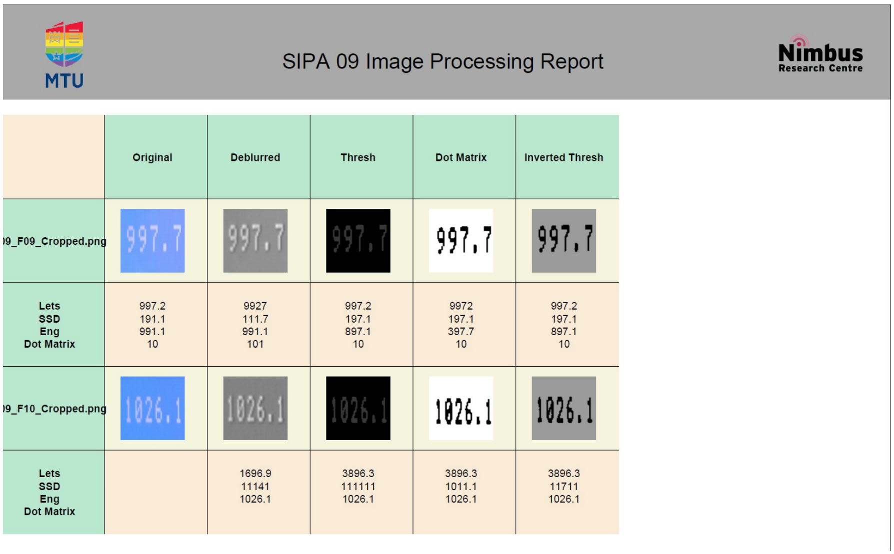
FIGURE 3.30: Image Folder G Montage

The methodology for Image Folder G 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.



SIPA 09 Image Processing Report

	Original	Deblurred	Thresh	Dot Matrix	Inverted Thresh
9_F09_Cropped.png	997.7	997.7	997.7	997.7	997.7
Lets SSD Eng Dot Matrix	997.2 191.1 991.1 10	9927 111.7 991.1 101	997.2 197.1 897.1 10	9972 197.1 397.7 10	997.2 197.1 897.1 10
9_F10_Cropped.png	1026.1	1026.1	1026.1	1026.1	1026.1
Lets SSD Eng Dot Matrix		1696.9 11141 1026.1	3896.3 111111 1026.1	3896.3 1011.1 1026.1	3896.3 11711 1026.1

FIGURE 3.31: Image Folder G Sample Output

3.6.8 Image Folder H

FIGURE 3.32: Image Folder H 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 H 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.

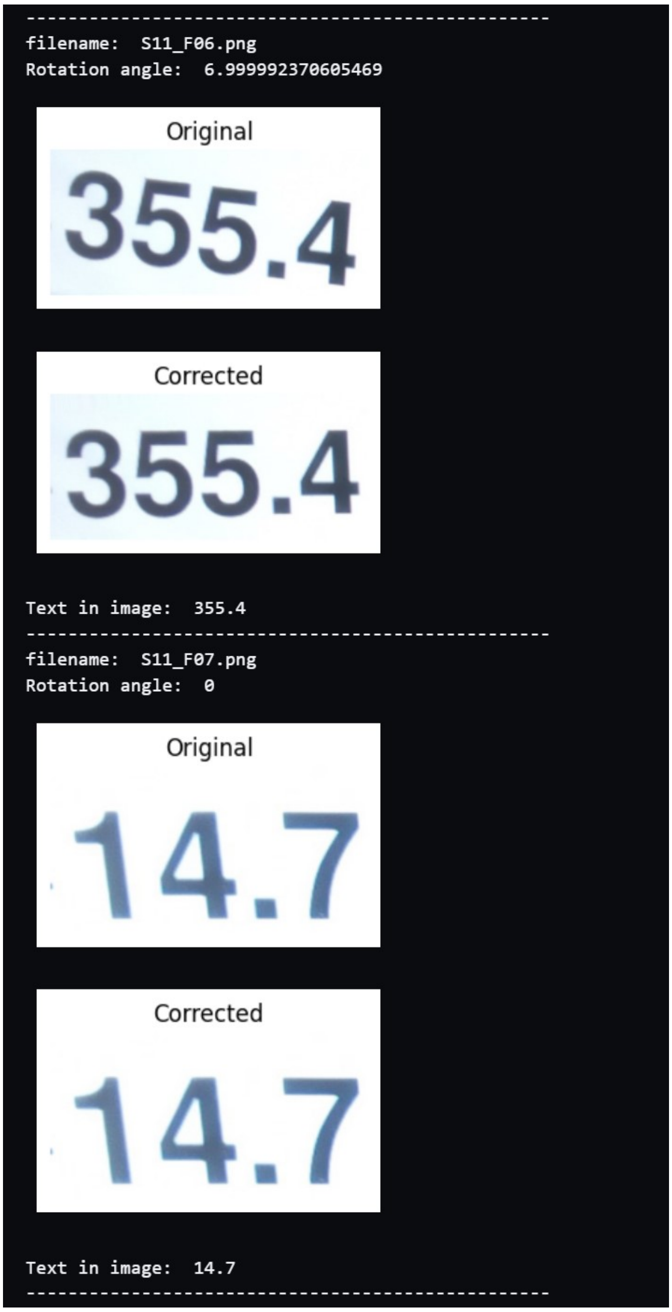


FIGURE 3.33: Image Folder H Sample Output

3.6.9 Third Sprint Conclusion

In the Third Sprint, the project delved deeper into the nuances of OCR by catering to specific challenges presented by each image set. Addressing the multifaceted nature of the images, individualized pre-processing techniques were employed for each folder. From leveraging colour masks like the Red and Green Masks to correct skewness using the Hough Transform, the methodologies were designed for optimized results. Notably, the sprint showcased the importance of adaptability and specificity, affirming that while global strategies offer a foundational approach, it's the fine-tuned, tailored interventions that ensure precision and excellence. The outcome of this sprint was a testament to the project's evolving capabilities, highlighting that with each iteration, the OCR process becomes increasingly robust and reliable.

3.7 Fourth Sprint - CRNN Methodology

Objective for the Fourth Sprint: To harness the combined strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) by implementing and fine-tuning a Convolutional Recurrent Neural Network (CRNN) methodology for Optical Character Recognition (OCR). The sprint encompasses the creation of a bespoke digit dataset tailored to specific image fonts, model architecture definition, data preparation and pre-processing, culminating in accurate digit predictions. The overarching goal is to achieve superior OCR accuracy by capitalizing on the spatial and sequential processing capabilities of the CRNN.

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.

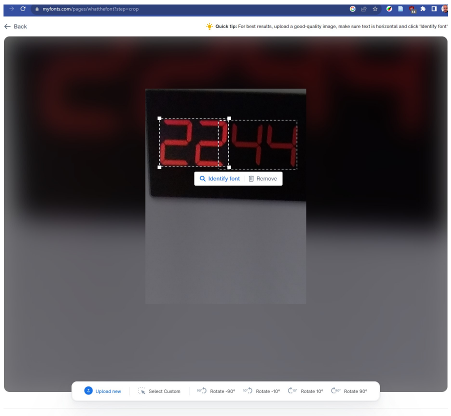


Figure 3.34: Image Font Identification

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

Several functions were developed to cater to the unique characteristics of the image folder and its segmentation requirements. These functions transform the image to grayscale, streamlining the process for subsequent contour detection. Contours falling below a certain area threshold are filtered out, ensuring only meaningful segments are retained. The accepted contours are organized based on their x-coordinates, which guarantees a sequential extraction of digits. For each pertinent contour, a bounding box is drawn, and segments meeting the area criteria are isolated as individual digits. The collective output of these functions comprises both the valid contours and the segmented digits, equipping the system for any further analysis or operations.

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.

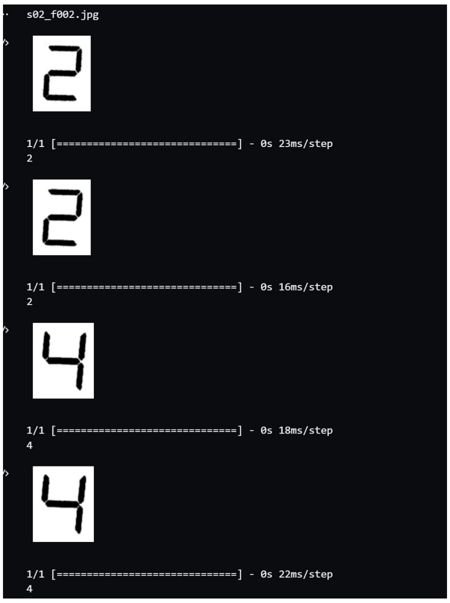


FIGURE 3.35: Image Prediction via CRNN

3.7.9 Fourth Sprint Conclusion

Throughout the fourth sprint, the focus has been the integration and fine-tuning of Convolutional Recurrent Neural Network (CRNN) methodologies for Optical Character Recognition (OCR). This hybrid model, which synergizes the spatial prowess of CNNs with the sequential expertise of RNNs, was found to be significantly adept at recognizing digits, especially when trained on custom datasets tailored to specific fonts in the images.

Our comprehensive exploration began with generating an expansive and varied training dataset. Emphasis was placed on the quality and diversity of the dataset, ensuring the CRNN model was exposed to an array of digit representations, mimicking real-world nuances. The meticulous process of defining the CRNN model architecture was undertaken, capitalizing on the complementary strengths of CNNs and RNNs. This was followed by data preparation steps, involving loading, normalization, and one-hot encoding to ensure data compatibility with the model.

The segment on Image Folder Pre-processing underlined the importance of accurate segmentation. The methods employed not only ensured efficient digit extraction but also catered to the unique characteristics of each image set. The sprint culminated in the prediction phase, where the trained CRNN model was leveraged to discern digit identities with high accuracy.

In essence, this sprint has showcased the efficacy of CRNNs in the realm of OCR, demonstrating its potential in handling both spatial and sequential intricacies inherent in the task. The detailed breakdown, from data generation to prediction, serves as a testament to the rigorous approach adopted in this phase, solidifying the foundation for subsequent sprints and investigations.

3.8 Methodology Chapter Conclusion

This chapter details the research methodology, emphasizing dedicated efforts to enhance the performance of Optical Character Recognition (OCR) systems. Specifically, the focus on Tesseract OCR and the Convolutional Recurrent Neural Network (CRNN) models led to a comprehensive evaluation of OCR systems using unprocessed image datasets.

Data capture, as outlined in Objective 2, equipped the study with a diverse set of sensor reading images, reflecting the multifaceted nature of the subject under investigation.

Objective 3 guided the exploration of image pre-processing techniques, prominently featuring colour masking before grayscale conversion. This method aimed to sharpen image clarity, setting the stage for better OCR results.

In alignment with Objective 4, the research delved into pinpointing optimal image capture settings. This exploration, combined with pre-processing, framed a strategy to enhance OCR precision and efficiency.

In essence, this chapter stands as a thorough guide, mapping out the methodology and addressing the initial four objectives of the research. It spans from initial OCR system assessments to the details of refined pre-processing methods and the nuances of optimal image capture settings.

Chapter 4

Results

4.1 Introduction

This chapter presents the results from our earlier systematic approach. By analysing the OCR performance on pre-processed image datasets, we aim to confirm that image pre-processing improves OCR effectiveness on sensor reading images. Each section discusses the results, highlighting the impact of the methodologies and paving the way for further discussion.

4.2 First Sprint - Global Generic

TABLE 4.1: OCR Performance First Sprint

Folder	Total Count	Tesseract	
		Read	Not Read
A	165	0	165
B	26	5	21
C	10	0	10
D	27	0	27
E	10	0	10
F	15	0	15
G	19	3	16
H	14	3	11

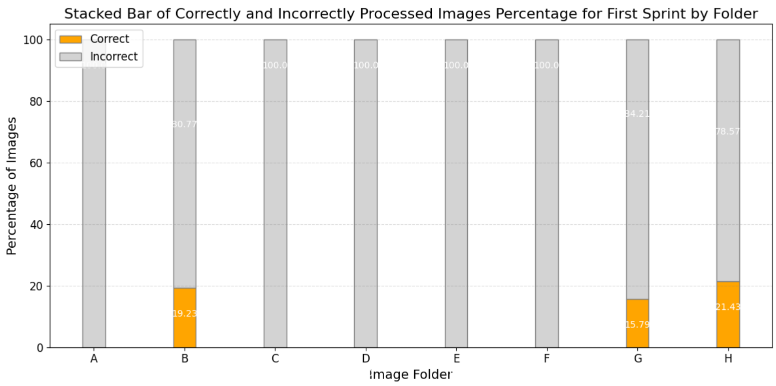


FIGURE 4.1: Tesseract First Sprint Across Folders

Observations: The chart demonstrates that most images across all folders were processed inaccurately. While Folders B, G, and H had some images processed correctly—with Folder H leading in accuracy—Folders A, C, D, E, and F recorded a complete lack of successful processing. This emphasizes the need for refining the image pre-processing algorithms.

4.3 Second Sprint - Global Generic Analysis Resized

TABLE 4.2: OCR Performance for Second Sprint

Folder	Total Count	Tesseract	
		Read	Not Read
A	165	1	164
B	26	12	14
C	10	0	10
D	27	0	27
E	10	2	8
F	15	2	13
G	19	5	14
H	14	7	7

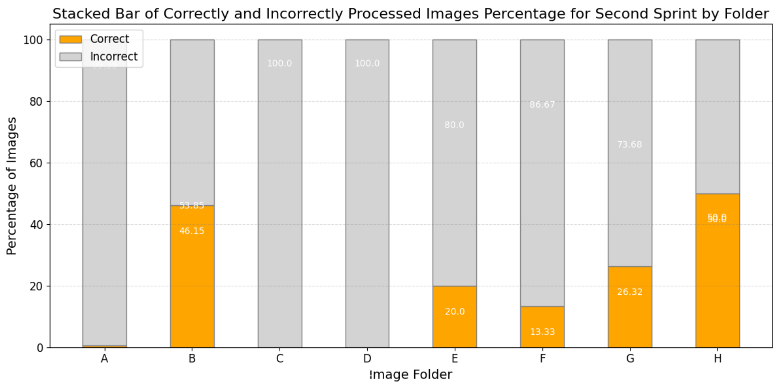


FIGURE 4.2: Tesseract Second Sprint Across Folders

Observations:The second sprint shows improvement in image processing across more folders, with larger orange sections. Folders B, E, F, G and H have a higher percentage of correctly processed images. Folders A, C, and D still have 100% or near, incorrectly processed images, indicating that the image pre-processing algorithms needs to be improved further to achieve better results.

4.4 Third Sprint - Analysis Tesseract Separate Folders

TABLE 4.3: OCR Performance for Different Folders

Folder	Total Count	Tesseract	
		Read	Not Read
A	165	70	90
B	26	12	14
C	10	3	7
D	27	16	11
E	10	10	0
F	15	1	14
G	19	12	7
H	14	13	1

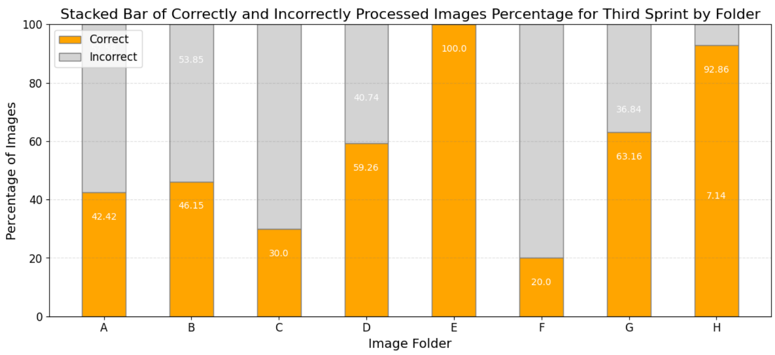


FIGURE 4.3: Tesseract Third Sprint Across Folders

Observations: Most folders now show a high percentage of correctly processed images. Folders E, and F are particularly successful, with close to or even 100% correctness. Folders G and I still have a proportion of incorrectly processed images, but their performance has improved significantly from the earlier sprints. Folder H has also seen improvement, but its rate of correctly processed images is slightly lower than the others.

4.5 Visualisation of Tesseract Sprint Performance

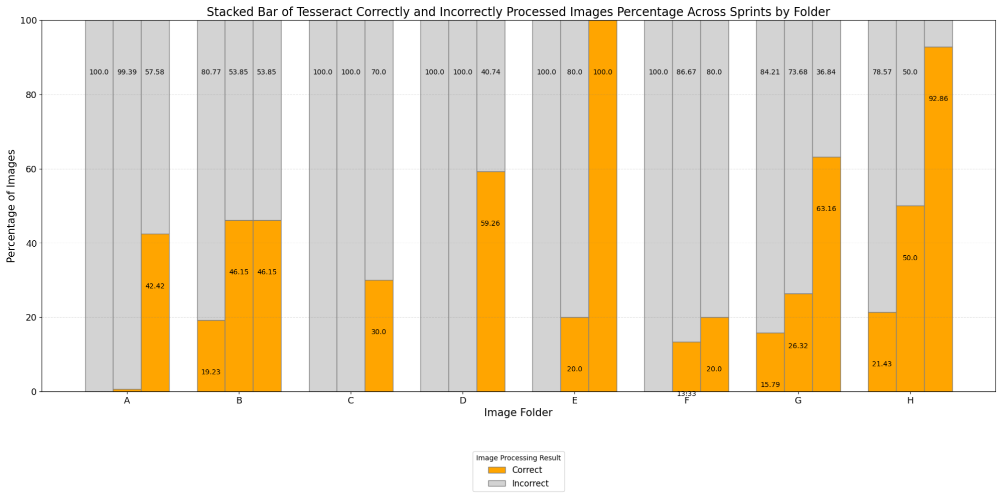


FIGURE 4.4: Tesseract Sprint Results Across Sprints

4.5.1 Analysis of Image Processing Accuracy Across Sprints

1. Overall Trend

There is a noticeable improvement in the accuracy of image processing from the "First Sprint" to the "Third Sprint". This is evident from the increasing size of the darker segments of the bars (representing correctly processed images) across most image folders from one sprint to the next.

2. Sprint-wise Observations

- **First Sprint**: Image processing accuracy is relatively lower, especially for folders A, B, D, and H. Folder G shows a significant number of incorrectly processed images, making it the folder with the highest inaccuracy in this sprint.
- **Second Sprint**: Accuracy has improved across almost all folders, especially for folders A, B, and D. The issues faced during the first sprint for these folders have been addressed. Folder G, however, still has a low accuracy percentage and scope for improvement.

- **Third Sprint**: The accuracy has further improved across all folders. Especially for Folder G, which had a significant number of inaccuracies in the first two sprints, the accuracy has noticeably increased. Folders E, F, and C seem to have almost perfect accuracy in this sprint.

3. Folder-wise Observations

- **Folder G**: This folder posed the most significant challenge throughout the sprints. However, by the third sprint, the accuracy improved substantially, indicating that the issues were identified and rectified over time.

- **Folders E, F, and C**: These folders consistently showed high accuracy across all sprints, suggesting that the image processing system was well-tuned for images from these folders from the start.

- **Folder H**: The accuracy for image processing in Folder H has consistently improved across the sprints. Starting with a relatively lower accuracy in the "First Sprint", there's a noticeable improvement in the "Second Sprint", and by the "Third Sprint", the accuracy is near-perfect.

4.6 Fourth Sprint - Analysis CRNN Separate Folders

TABLE 4.4: OCR Performance for CRNN Separate Folders

Folder	**Total Count**	**CRNN**		
		Read	**Not Read**	**No Contours**
A	165	46	9	105
B	26	17	9	0
C	10	1	8	1
D	27	1	4	6
E	10	4	6	0
F	15	0	6	8
G	19	17	2	0
H	14	6	7	1

Of the 286 images 92 were read making a 32.16% read count.

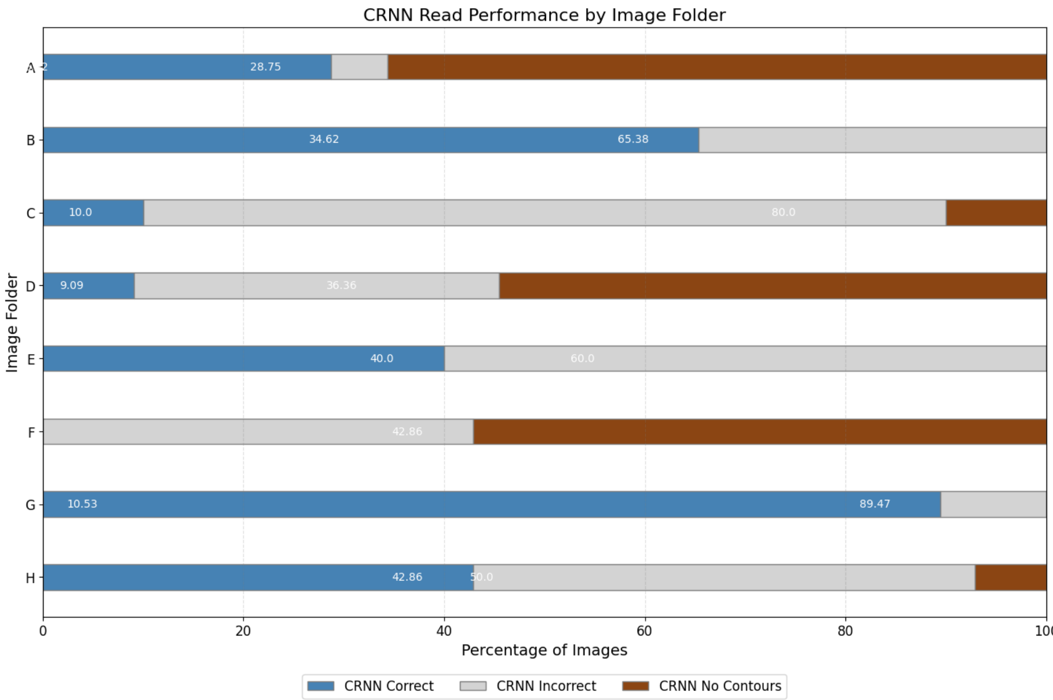


FIGURE 4.5: CRNN Results

4.7 Tesseract and CRNN Combined

TABLE 4.5: OCR Performance for Different Folders

Folder	Total Count	Tesseract		CRNN		
		Read	Not Read	Read	Not Read	No Contours
A	165	70	90	46	9	105
B	26	12	14	17	9	0
C	10	3	7	1	8	1
D	27	16	11	1	4	6
E	10	10	0	4	6	0
F	15	1	14	0	6	8
G	19	12	7	17	2	0
H	14	13	1	6	7	1

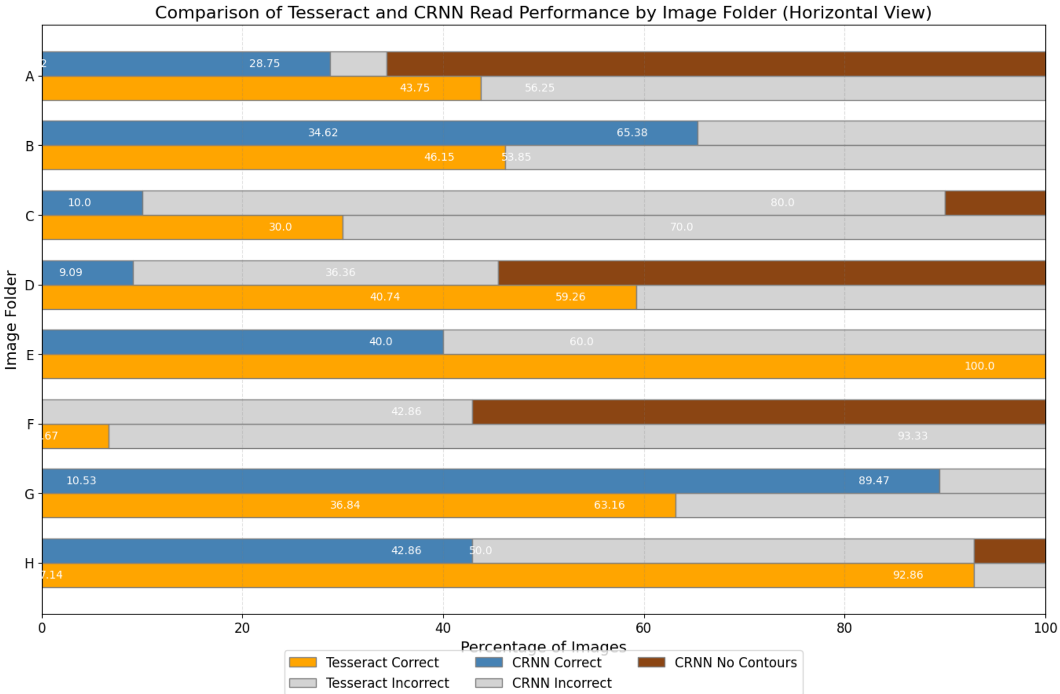


FIGURE 4.6: Tesseract Versus CRNN

4.7.1 Analysis of Tesseract and CRNN Performance

1. **Overall Performance**:

   - **Tesseract**: Tesseract showcases consistent performance across most image folders. Notably, folders E and H both achieve an accuracy above 90%.
   - **CRNN**: CRNN's performance varies extensively across the folders. In Folder A, its read percentage is only 28.75%, indicating a significant area for improvement.

2. **Image Folder Specific Insights**:

   - **Folder A**: Tesseract significantly outperforms CRNN.
   - **Folder B**: CRNN outperforms Tesseract.
   - **Folder C**: Tesseract has a distinct advantage over CRNN albeit the performance both is poor.
   - **Folder D**: Tesseract performs better, while CRNN struggles, especially with contour detection.
   - **Folder E**: Tesseract achieves an impressive accuracy rate of 100%, surpassing CRNN.
   - **Folder F**: Tesseract and CRNN perform poorly, notably in contour detection.
   - **Folder G**: CRNN outperforms Tesseract.
   - **Folder H**: Tesseract achieves another outstanding accuracy rate of over 90%, significantly outperforming CRNN.

3. **CRNN's Contour Detection**:

   - A significant challenge for CRNN is its inability to detect contours, especially evident in folders A and D. This indicates a potential area for improvement or adjustment in the CRNN model for specific image types.

4. **Comparison**:

   - Tesseract has outperformed CRNN in all folders apart from B and G. On the other hand, while CRNN exhibits potential in some folders, its variable performance and specific challenges in others highlight areas that need addressing.

In conclusion, Tesseract's performance is higher, especially in folders E and H, makes it a preferable choice in its current state. CRNN, with its present inconsistencies, requires further tuning or modifications for enhanced reliability.

Chapter 5

Discussion and Conclusion

5.1 Introduction

The motivation for this project was the prevalent challenges faced in manual and infrequent environmental sensor readings in operational settings, and the potential of Optical Character Recognition (OCR) technology to enhance continuous data capture, reduce errors, and optimize system efficiency.

The research aims to enhance the efficiency and accuracy of Optical Character Recognition (OCR) on sensor reading images through novel pre-processing and optimized image capture settings, with a focus on Tesseract OCR and CRNN models. The specific objectives are as follows:

- Conducting a comprehensive literature review on OCR methods.
- Capturing a diverse dataset of sensor reading images.
- Designing and implementing image pre-processing techniques, with an emphasis on colour masking.
- Identifying optimal image capture parameters such as contrast, distance, and lighting.

- Comparing the impact of pre-processing and optimized capture settings on Tesseract and CRNN OCR results.
- Analysing and reporting the findings to fill the current gap in literature concerning pre-processing and capture optimization for OCR of sensor readings.

5.2 Discussion of Findings

5.2.1 Performance of Tesseract OCR on Raw Image Datasets

The research study involved the analysis of 286 sensor digit reading images distributed across 8 distinct folders. Each folder encapsulated images that presented unique challenges, including skewness, sun reflection, and suboptimal image quality. The underlying distinctiveness of each folder was the type of digit font used in the sensor readings.

For the optical character recognition (OCR) process, Tesseract version 5.2.0.20220712 was chosen, leveraging pretrained models tailored for English and Seven Segment Display. The configurations `--psm 13` and `tessedit_char_whitelist=123456789` were employed to optimize the recognition.

5.2.2 OCR Sprints and Strategies

5.2.2.1 First Sprint

The initial strategy was straightforward, converting the images to grayscale and then subjecting them to Tesseract. The outcome of this sprint yielded an unsatisfactory accuracy rate of 3.85%, indicating a need for a more nuanced pre-processing approach.

5.2.2.2 Second Sprint

Recognizing the limitations of the initial approach, this phase introduced OTSU thresholding and morphological closing. A systematic examination against an expansive range of sizes was also undertaken to pinpoint the optimal size for processing.

5.2.2.3 Third Sprint

This phase saw the tailoring of pre-processing techniques to the unique challenges posed by each folder:

- **Folder A**: A red mask was developed to isolate and return only the red colour. Subsequent steps included grayscale conversion, OTSU thresholding, morphological closing, and dilation. A success rate of 47.42% was achieved.
- **Folder B**: The images underwent thresholding, morphological closing, erosion, and finally, OTSU thresholding. A success rate of 46.15% was achieved.
- **Folder C**: Denoising and Weiner filtering were the primary pre-processing techniques applied. A success rate of 30.00% was achieved.
- **Folder D**: The pre-processing involved cropping followed by median blurring. A success rate of 59.26% was achieved.
- **Folder E**: A green mask was employed to isolate green elements. This was followed by deblurring, thresholding, and denoising. A success rate of 100.00% was achieved.
- **Folder F**: The images in this folder were subjected to a green mask, deblurring, and thresholding. A success rate of 20.00% was achieved.
- **Folder G**: Cropping, inversion, and thresholding formed the pre-processing sequence. A success rate of 61.16% was achieved.
- **Folder H**: The images underwent cropping and skewness correction. A success rate of 92.86% was achieved.

All folders, regardless of the additional pre-processing steps, had grayscale conversion as a standard step.

5.2.3 Performance and Results

Detailed accuracy metrics, visualized through tables and charts, are elucidated in the results section. These will offer a comprehensive view of the incremental improvements achieved across the three sprints and the specific success rates associated with each folder. In this project Tesseract OCR was able to achieve an overall accuracy higher than the CRNN model.

Unreadable/Partially read Images: Certain images were found to be unreadable or challenging to process due to various reasons (e.g., poor quality, distortion, lighting, shadows etc.). The optimal pre-processing methods tailored for each folder within the dataset were unable to sufficiently improve the readability of these images. These images were not excluded from the final analysis and have directly contributed to the observed results, leading to lower accuracy metrics.

FIGURE 5.1: Selection of Unread Images

Recognizing this challenge is crucial, as it emphasizes the importance of image quality in achieving optimal outcomes. Future research might focus on developing methods to enhance the readability of such images, exploring pre-processing techniques, and optimizing camera positioning under ideal conditions. Proper testing of these placements can also play a pivotal role in mitigating the impact of image quality on results.

5.2.4 Fourth Sprint - CRNN Model Analysis

In addition to Tesseract, the study also evaluated the performance of Convolutional Recurrent Neural Network (CRNN) models as an OCR tool. At least one CRNN model was developed per image folder. The models were trained using fonts that closely resemble those in the image dataset pertaining to that folder. The CRNN models underperformed in comparison to Tesseract with this author attributing this to the following reasons:

1. **Partial Accuracy in Sensor Readings:** Some models were able to make accurate predictions of specific digits, such as 2, 3, 4, 5, 7, and 9. However, it sometimes misinterpreted the digit 8 as 6. This would return an "incorrect" prediction, even though the model was able to correctly identify the majority of the digits in the sensor reading.

2. **Challenges in Digit Segmentation during Pre-processing:** A pivotal step in CRNN pre-processing is the segmentation of individual digits for prediction. Especially in noisy images, this step becomes arduous. Noise can obfuscate the model's ability to discern and correctly segment digits, influencing overall prediction accuracy.

3. **Overfitting:** With their deep architectures, CRNNs can be prone to overfitting, especially when trained on a limited dataset or if the font type selected was not identical to the digits on the sensor readings.

4. **Insufficient Training Data:** CRNNs often require a significant amount of labelled training data for optimal performance. The limited dataset lead to poor generalization in some instances.

5. **Complexity:** The interplay between CNN and RNN layers can make CRNNs computationally intensive, affecting training and fine-tuning durations.

6. **Alignment Issues:** In tasks like Optical Character Recognition (OCR), there can be misalignment between the predicted sequence and the ground truth, leading to discrepancies in results.

7. **Lack of Invariance:** If not trained on a diverse set of data, CRNNs might not handle variations like rotations, scaling's, or other transformations effectively. Variations were introduced into the training data, however it is likely that these variations were not sufficient to ensure invariance.

5.2.5 Image Pre-processing Techniques

Image pre-processing is instrumental in bridging the gap between raw data and actionable insights in the domain of machine learning. By refining the input data, these techniques set the stage for models to deliver optimal performance. Here, we dissect the significance and impact of various pre-processing techniques employed across different image folders:

- **Grayscale Conversion:** This step, indispensable for Tesseract, simplifies the image by eliminating colour variations by converting three colour channels (RGB) to one single channel, allowing the algorithm to focus solely on structural features.

- **Resizing:** The dimensions of an image play a pivotal role in determining Tesseract's performance. By systematically testing a range of sizes, the optimal dimensions that yielded the highest recognition accuracy were identified.

- **Thresholding Techniques:** By binarizing the image, thresholding techniques accentuate the contrast between the text and the background, facilitating clearer text recognition.

- **Denoising:** Real-world images often come with their fair share of noise. Denoising techniques were employed to cleanse the images, making them more palatable for the subsequent processing steps.

- **Deblurring:** Motion blurs or out-of-focus images can obfuscate details. Deblurring techniques were harnessed to sharpen the images, ensuring that the text details remained discernible.
- **Red Mask** A tailored red mask was developed for Image Folder A, elevating the model's accuracy from an initial 1% to a commendable 47.42%.
- **Green Mask:** The introduction of a green mask filter significantly bolstered the accuracy of predictions. In the case of Image Folder E, while the second iteration registered a 20% accuracy, the application of the green mask in a later sprint escalated this figure to 100%. Similarly, the accuracy for Image Folder F experienced an uptick upon the integration of the green mask filter.

The key takeaway from these experiments is that meticulous and tailored pre-processing can have a significant impact on performance. By accounting for the unique characteristics of each dataset, these techniques can not only improve performance but also reveal the potential of data refinement to transform results.

5.2.6 Comparative Analysis using CRNN Models

The results show that the effectiveness of Tesseract and CRNN depends on the type of images and the pre-processing techniques used. Tesseract performed better in some folders, while CRNN performed better in others. This variability highlights the importance of understanding the dataset's characteristics and tailoring pre-processing techniques accordingly. Future research could focus on improving these techniques or developing hybrid approaches that combine the strengths of both methods.

5.3 Challenges and Limitations

The study, while providing valuable insights, was not without its challenges and limitations. These constraints, detailed below, offer avenues for future research and refinement.

1. **Data Variability:** The disparate characteristics across different image folders influenced the performance of both Tesseract and CRNN. Achieving consistent accuracy across folders with varying image quality and content was challenging.

2. **Image Quality:** Some images, inherently noisy, blurred, or of low resolution, impeded the recognition algorithms' accuracy.

3. **Segmentation Issues:** Proper digit segmentation, especially in the presence of image noise, posed significant challenges. Noise often obfuscated the differentiation and correct segmentation of digits.

4. **Misinterpretation of Digits:** Specific digits, like 8 being recognized as 6, were consistently misinterpreted, indicating potential pitfalls in the recognition process.

5. **Pre-processing Limitations:** The efficacy of pre-processing techniques varied. Techniques like the green mask might excel for one folder but falter for another.

6. **Computational Constraints:** The deep architectures of CRNNs demanded substantial computational resources, constraining extensive hyperparameter tuning or exploration of intricate models.

7. **External Factors:** Aspects like lighting conditions, shadows, and potential background interference during the initial image capture could have affected recognition accuracy.

8. **Overfitting:** The risk of overfitting, especially pertinent given the complexity of CRNNs, was a concern, particularly as the training dataset lacked diversity.

9. **Training Data Limitations:** A non-comprehensive training dataset, missing certain image variations, compromised the model's generalization capabilities.

10. **Algorithm Limitations:** Both Tesseract and CRNN, while robust, come with inherent limitations. Tesseract had issues with specific fonts or styles, whereas CRNN faces challenges with sequence length variability. This lead to the need for tailored pre-processing techniques such as digit segmentation.

5.3.1 Analysis of OpenCV

OpenCV is a powerful and versatile tool for computer vision tasks. It has a wide range of functionalities, from basic image pre-processing to advanced techniques. This makes it a good choice for a wide variety of projects.

However, OpenCV's extensive feature set was a drawback at the beginning of this project. The learning curve was steep, and it was difficult to know where to start. Additionally, OpenCV is not specifically designed for deep learning-based vision applications. For these tasks, specialized frameworks like **TensorFlow** or **PyTorch** may be a better choice. OpenCV was chosen for this project

5.3.1.1 Detailed Breakdown of OpenCV's Strengths and Weaknesses

Strengths:

- *Wide range of functionalities*: OpenCV has a wide range of functions for image processing, object detection, and machine learning. This makes it a good choice for a variety of projects.
- *Seamless integration with other libraries*: OpenCV can be seamlessly integrated with other popular libraries, such as **NumPy** and **Python Imaging Library (PIL)**. This makes it easy to build complex applications.
- *Performance*: OpenCV is written in C/C++, which makes it fast and efficient.
- *Open source*: OpenCV is open source, which means it is free to use and modify.

Weaknesses:

- *Steep learning curve*: OpenCV's extensive feature set can make it difficult to learn.
- *Not specifically designed for deep learning*: OpenCV is not specifically designed for deep learning-based vision applications. For these tasks, specialized frameworks like **TensorFlow** or **PyTorch** may be a better choice.
- *Documentation*: OpenCV's documentation can be outdated and difficult to understand.

5.3.2 Analysis of Tesseract

Tesseract is a powerful and versatile open-source OCR tool. It has been evolving for decades and is now one of the leading OCR tools available. During this project, Tesseract yielded optimal results only when specific pre-processing steps were applied to the images; it did not excel as a generic tool without tailored interventions.

However, like all tools, Tesseract has its strengths and weaknesses. It is most accurate when the input images are of high quality and the document layout is simple. In cases where the images are low quality or the document layout is complex, Tesseract was not be able to accurately recognize the text.

5.4 Implications and Applications

The results of this research not only provide insights into the intricacies of optical character recognition but also pave the way for tangible advancements in various application domains.

5.4.1 Contributions to the Field of OCR

The comparison of Tesseract and CRNN highlights their respective strengths and limitations, guiding researchers in tool selection based on dataset characteristics. The challenges identified can also direct future OCR improvements.

5.4.2 Reading Sensor Data

The ability to accurately read sensor data has vast implications. Automated and accurate sensor data readings can lead to:

- Improved monitoring and reporting in industrial settings.
- Enhanced data integrity, minimizing human-induced errors.
- Efficient real-time tracking, leading to timely decision-making.
- Cost savings by reducing manual monitoring efforts.

5.4.3 Real-world Applications and Benefits

Beyond sensor data reading, the findings of this research can be extrapolated to various real-world scenarios:

- **Healthcare:** Automated reading of medical tests, patient data, or prescription labels.
- **Logistics:** Recognizing and sorting packages based on labels or addresses.

Each of these applications not only enhances efficiency but also contributes to a more streamlined and error-free operational environment.

5.5 Conclusions

This research embarked on a journey to delve deep into the intricacies of optical character recognition, specifically comparing the prowess of Tesseract and CRNN. Through experimentation and analysis across diverse image folders, several pivotal insights were unearthed:

- The efficacy of both Tesseract and CRNN is heavily influenced by the nature of the images and the pre-processing techniques employed. While Tesseract showcased superior performance in certain folders, CRNN emerged as the preferred choice in others.
- Image quality, manifested through challenges like light reflection, blurriness, and skewness, plays a cardinal role in determining recognition accuracy. Addressing these challenges through tailored camera recommendations and advanced pre-processing can significantly bolster recognition rates.
- The study underscored the transformative potential of bespoke data refinement. Treating each dataset with its unique idiosyncrasies in mind can pave the way for substantial improvements in recognition accuracy.

This author believes this research holds some significance. As the digital world continues to expand, the demand for efficient and accurate OCR systems is escalating. By highlighting the strengths, limitations, and potential areas of improvement for both Tesseract and CRNN, this study provides a roadmap for future endeavours in the field. The insights gleaned not only contribute to the academic corpus of OCR but also have tangible implications for real-world applications, from reading sensor data in industrial setups to streamlining operations in sectors like healthcare, retail, and logistics.

In conclusion, while there remain avenues for further exploration and refinement, this research stands as a testament to the strides made in enhancing OCR performance and the promising horizons that lie ahead.

5.6 Recommendations and Future Work

5.6.1 Camera Recommendations for Enhanced Image Capture

The quality of captured images plays an important role in determining the efficacy of subsequent processing and analysis. To address challenges like light reflection, blurriness, and skewness, the following camera recommendations are proposed:

- **Optimal Lighting:** Ensure a uniform lighting environment during image capture. Avoid placing sensors under direct light to prevent reflections. The use of diffused lighting or soft-boxes can help mitigate harsh glares and reflections.
- **Camera Stability:** To minimize image blurriness, employ tripods or stable platforms for camera mounting, ensuring a steady capture process.
- **Adjust Focus:** Regular camera focus calibration is crucial. Cameras equipped with an adaptive auto-focus feature can be particularly advantageous for varying capture distances.
- **Angle Consistency:** To curtail skewed image captures, position the camera perpendicular to the sensor. Adjustable mounts or stands can facilitate precise angle alignments.
- **Define Region of Interest (ROI):** Utilize cameras that allow for ROI definition. By delineating the ROI, cameras can prioritize and focus on the specified area, especially beneficial in environments with potential visual noise or distractions.
- **Image Resolution:** Opt for high-resolution cameras to discern finer details. Even if resizing becomes necessary during post-processing, higher resolution images offer a more detailed foundation.

- **Lens Choice:** Lenses with anti-reflective coatings can further diminish glare and undesired reflections.
- **Background Contrast:** A contrasting background relative to the sensor can significantly aid in distinguishing the sensor during subsequent image processing tasks.

Adhering to these recommendations can markedly enhance image capture quality, paving the way for more accurate and efficient data analysis. Such guidelines serve as a bedrock for future research, underscoring the importance of pristine initial data capture.

5.6.2 Future Work

As the field of OCR continues its evolutionary trajectory, myriad promising avenues beckon exploration. Highlighted below are some of the compelling avenues for future research:

- **Exploration of Other Neural Network Architectures:** Delving into architectures beyond the conventional, like Transformer-based models or Capsule Networks, holds promise. Recognized for their efficacy in various sequence recognition tasks, adapting these architectures for OCR could herald significant advancements.
- **Real-world Deployment and Scalability:** The litmus test for any OCR system lies in real-world deployment. This necessitates a deep dive into factors encompassing scalability, computational efficiency, and a design that is user-centric.
- **Integration with Augmented Reality:** Melding OCR with augmented reality (AR) can usher in real-time text recognition within a live augmented environment, potentially enhancing user experiences and diversifying OCR applications.
- **Semi-supervised and Unsupervised Learning:** Given the labour-intensive annotation requirements of data, semi-supervised and unsupervised learning paradigms could be pivotal. Harnessing vast unlabelled data pools can expedite the journey towards more refined OCR systems.

- **Feedback Loop for Continuous Learning:** Infusing OCR systems with a user feedback loop can be transformative. This mechanism, allowing users to rectify inaccuracies, ensures the system's continuous learning and refinement.

- **Security and Privacy:** With OCR systems processing increasingly sensitive data, underpinning them with robust security and privacy measures becomes non-negotiable. This calls for advanced encryption methodologies and cutting-edge anonymization techniques.

- **Customizable OCR Systems:** Tailoring OCR systems to individual user needs and diverse applications can ensure their broader relevance. Systems, when designed with customization or fine-tuning capabilities, can cater to a gamut of specific requirements.

These facets merely scratch the surface of the vast expanse of potential research directions in OCR. Fuelled by relentless innovation, the horizon looks promising with OCR systems poised to be more precise, agile, and adaptable, catalysing a plethora of novel applications.

The journey of this research is a reflection of the evolving nature of OCR, and while conclusions have been drawn, it also marks the beginning of numerous possibilities and questions yet to be explored.

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

Appendix Introduction

The appendix of this thesis provides a more detailed look at the technical aspects of the research presented in the main body. This includes supporting charts, visualizations, and code snippets that explain the specific pre-processing steps, Optical Character Recognition (OCR) techniques, and font selection choices that were made.

The materials in this appendix are not essential for understanding the main findings of the thesis, but they are important for readers who want a deeper understanding of the methods and processes used. This is especially true for the novel pre-processing steps and the role of image quality and contrast in OCR.

By navigating through this appendix, enthusiasts and specialists alike will gain a deeper understanding of the intricacies that have contributed to the efficacy of the OCR approach undertaken in this Thesis.

A.2 Analysis Tesseract Separate Folders

A.2.1 Image Folder A Best Image Size Analysis

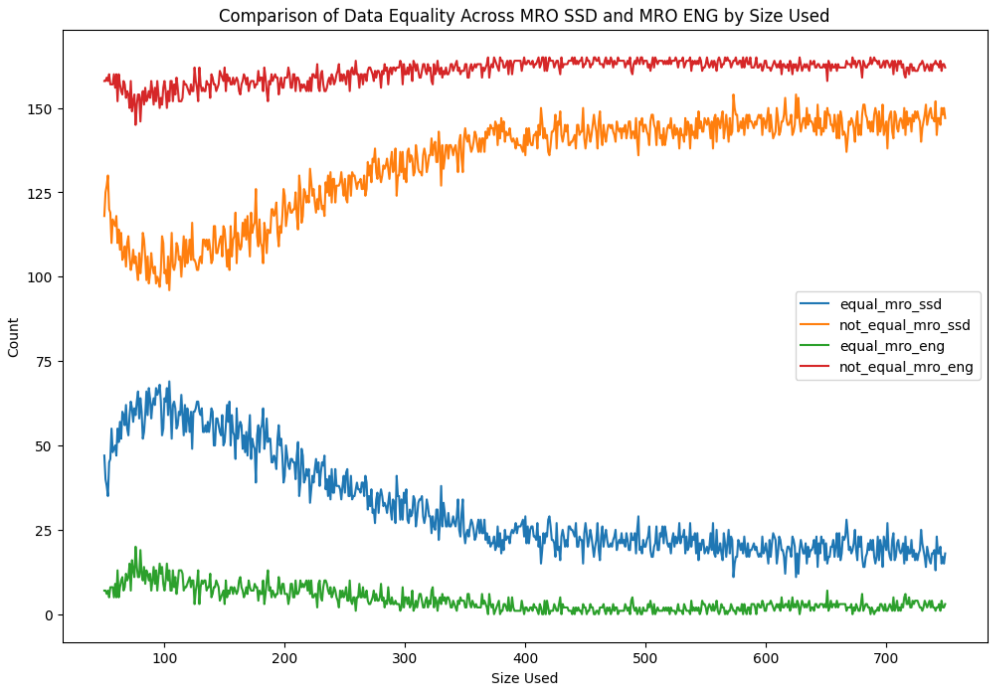


FIGURE A.1: Image Folder A Count Analysis

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.2.2 Image Folder A Contrast Analysis on MRO SSD

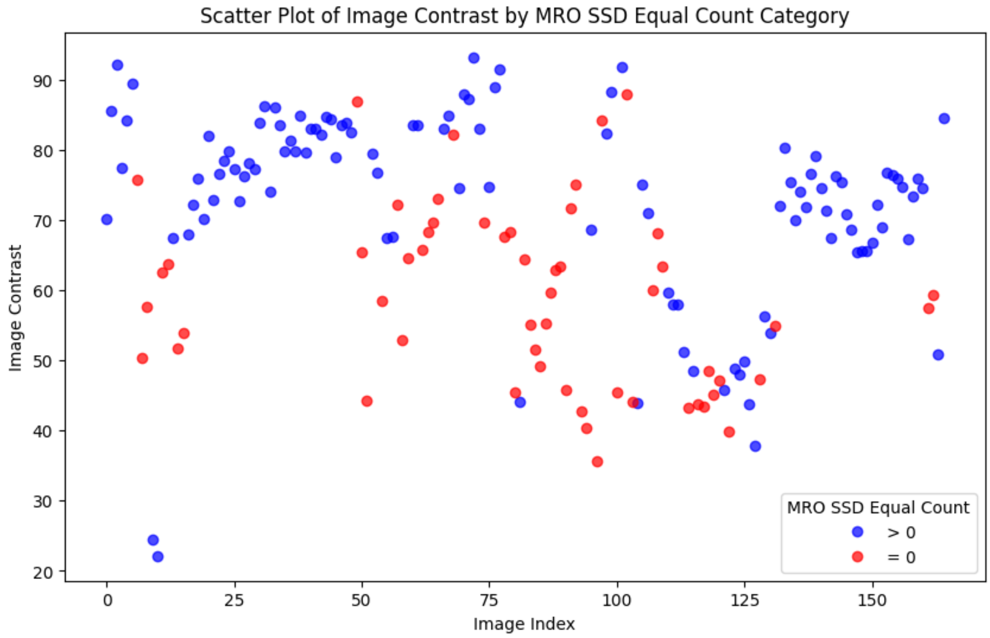


FIGURE A.2: Image Folder A Contrast Analysis on MRO SSD

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.2.3 Image Folder A Brightness Analysis on MRO SSD

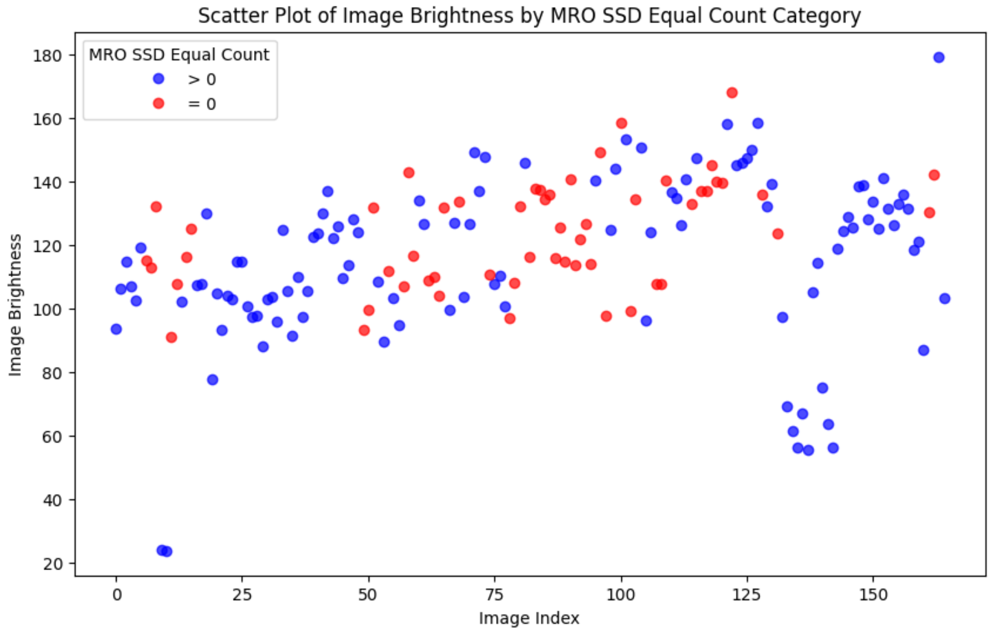


FIGURE A.3: Image Folder A Brightness Analysis on MRO SSD

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.2.4 Image Folder A Standard Deviation Analysis on MRO SSD

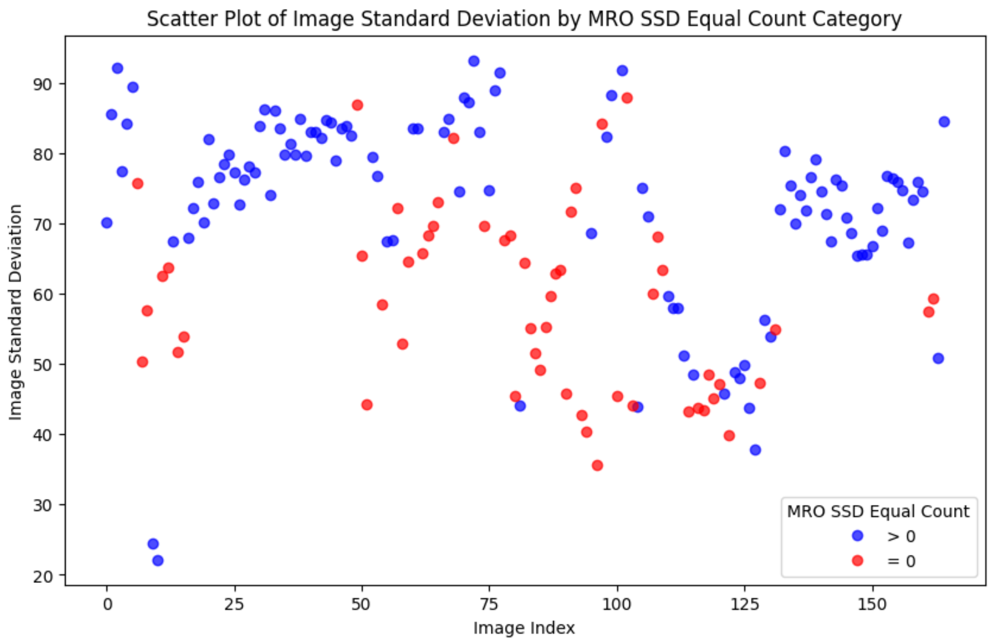


FIGURE A.4: Image Folder A Standard Deviation Analysis on MRO SSD

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.2.5 Image Folder A Homogeneity Analysis on MRO SSD

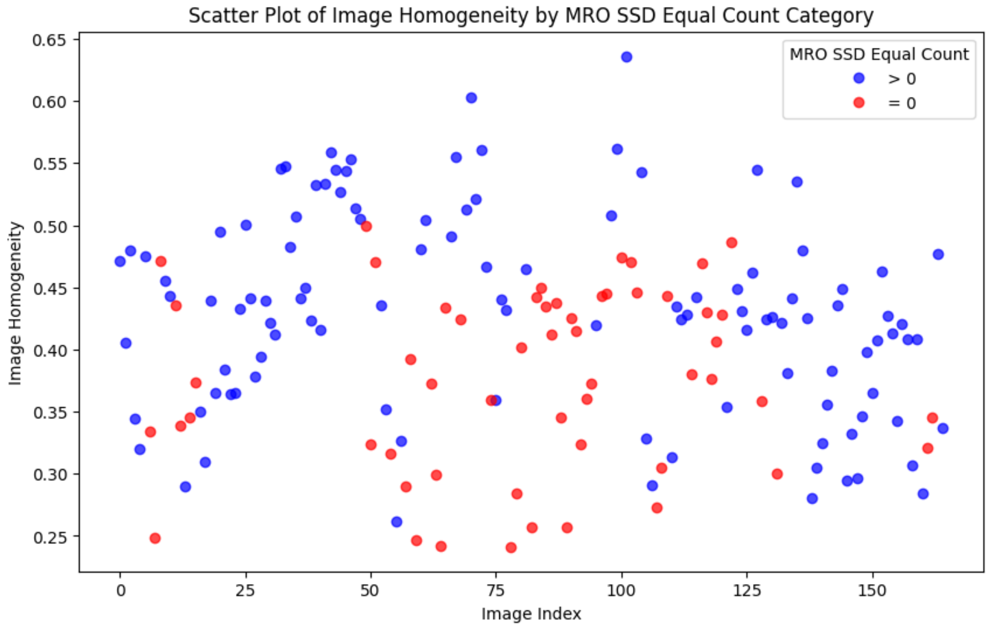


FIGURE A.5: Image Folder A Homogeneity Analysis on MRO SSD

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.2.6 Image Folder A Sharpness Analysis on MRO SSD

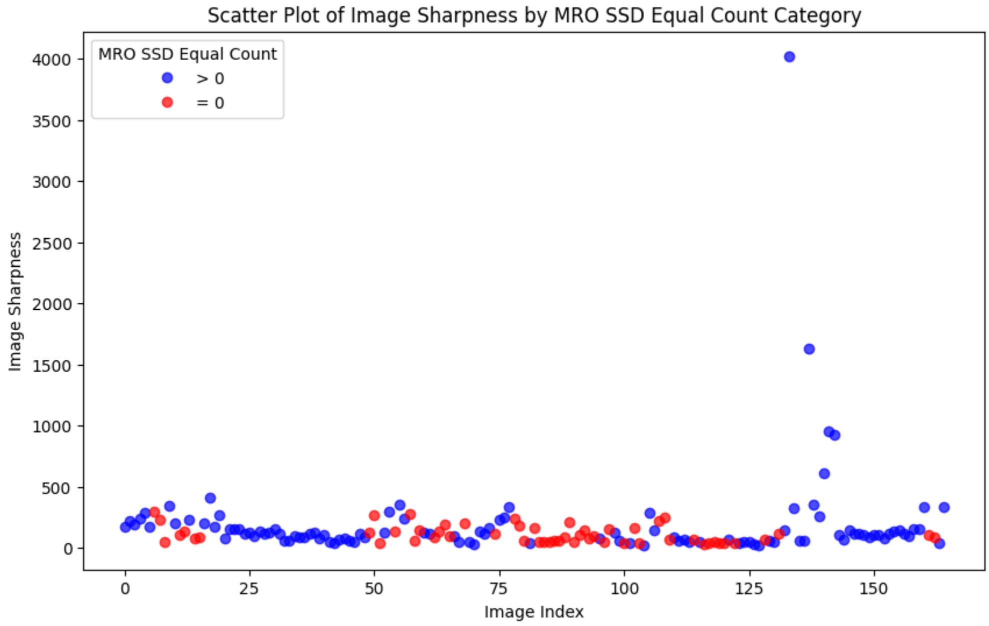


Figure A.6: Image Folder A Sharpness Analysis on MRO SSD

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.2.7 Image Folder A Dissimilarity Analysis on MRO SSD

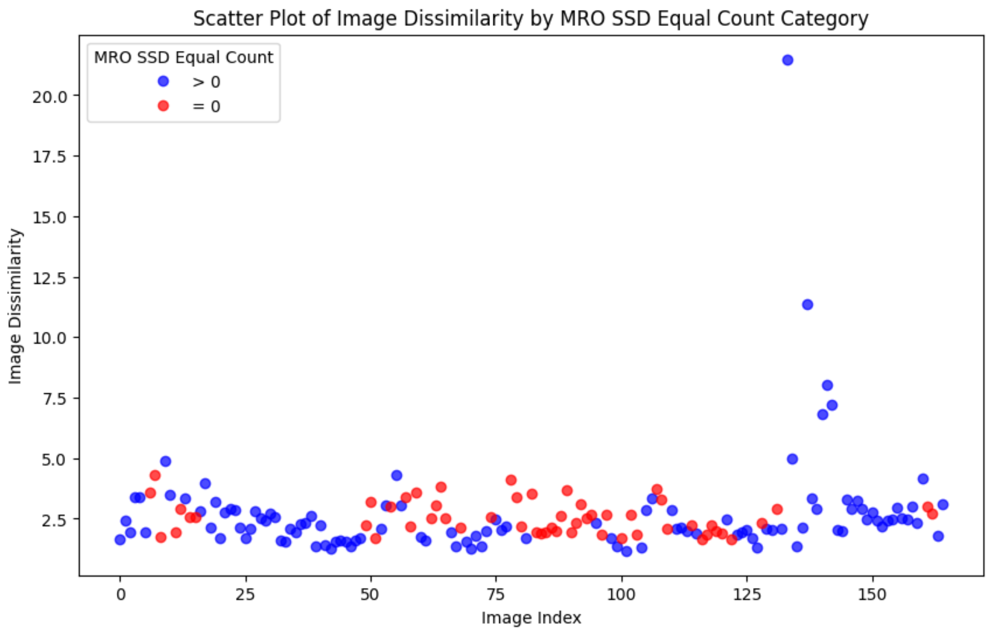


FIGURE A.7: Image Folder A Dissimilarity Analysis on MRO SSD

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.2.8 Image Folder A Area Analysis on MRO SSD

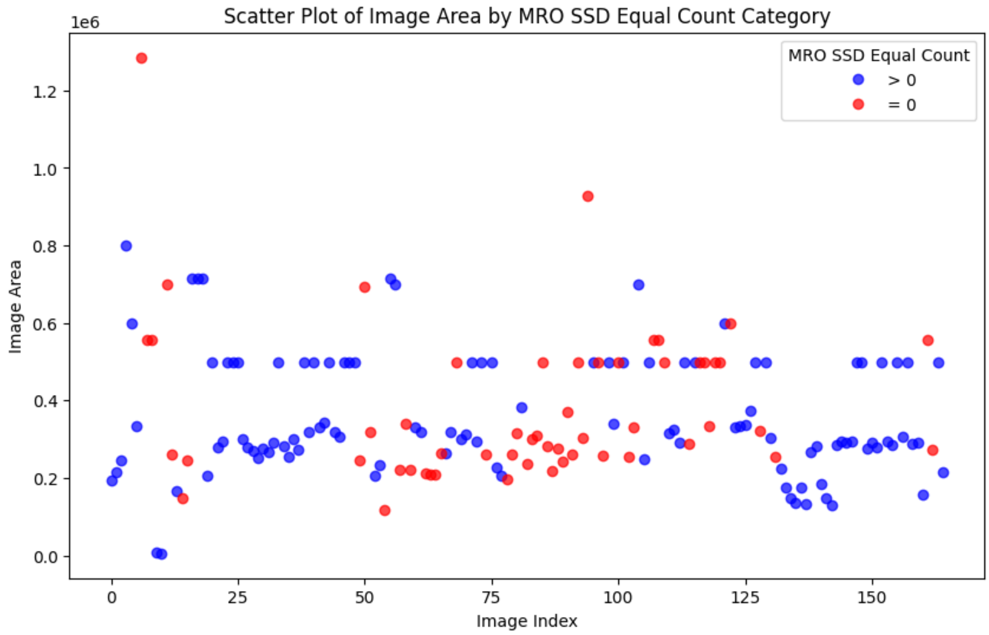


FIGURE A.8: Image Folder A Area Analysis on MRO SSD

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.2.9 Image Folder A Correlation Analysis on MRO SSD

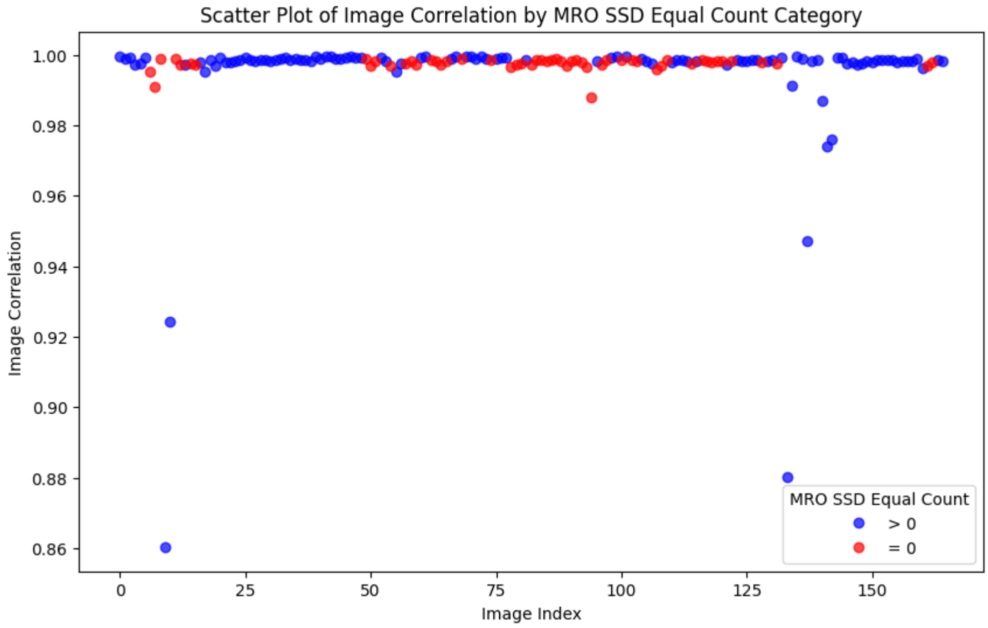


FIGURE A.9: Image Folder A Correlation Analysis on MRO SSD

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.2.10 Image Folder A Energy Analysis on MRO SSD

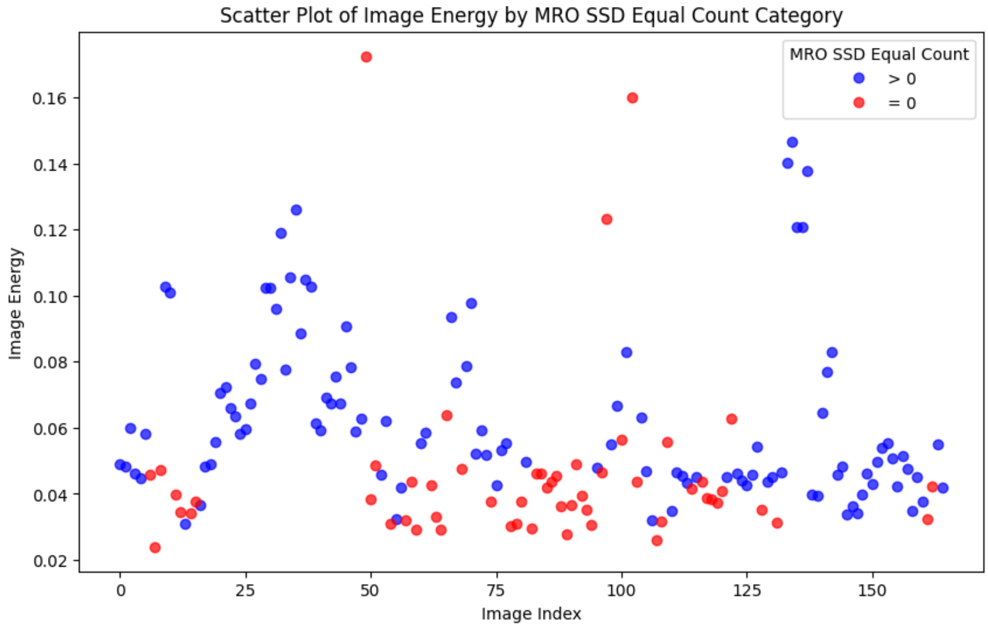


FIGURE A.10: Image Folder A Energy Analysis on MRO SSD

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.

A.2.11 Mask Red Function

```
def mask_red(img):
    """
    Returns an image with only the red pixels from the input image.

    Parameters:
    img (numpy.ndarray): Input image in BGR format.

    Returns:
    numpy.ndarray: Output image with only the red pixels from the input image
.

    """
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # define lower and upper red ranges in HSV
    lower_red1 = np.array([0, 50, 50])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([170, 50, 50])
    upper_red2 = np.array([180, 255, 255])

    # create masks with the specified red ranges
    mask1 = cv2.inRange(img_hsv, lower_red1, upper_red1)
    mask2 = cv2.inRange(img_hsv, lower_red2, upper_red2)

    # join the masks
    mask = mask1 + mask2

    # set the output image to zero everywhere except the red mask
    output_img = img.copy()
    output_img[np.where(mask == 0)] = 0

    return output_img
```

Listing A.1: Mask Red Function

A.2.12 Mask Green Function

```
def mask_green(img):
"""
Returns an image with only the green pixels from the input image.

Parameters:
img (numpy.ndarray): Input image in BGR format.

Returns:
numpy.ndarray: Output image with only the green pixels from the input image.

"""
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# define range of green color in HSV
lower_green = np.array([40, 50, 50])
upper_green = np.array([80, 255, 255])

# create a mask with the specified green range
mask = cv2.inRange(img_hsv, lower_green, upper_green)

# apply the mask to the image to get the output image
output_img = cv2.bitwise_and(img, img, mask=mask)

return output_img
```

Listing A.2: Mask Green Function