ABSTRACT

This project was initiated to explore and investigate the existing Cyprus ID format, including the structure and the various fields it contains, with a deep dive into the crafting of the distinctive features. The objective is to gain a thorough understanding of how these fields are structured and to then create an advanced, accurate, and fast OCR tool using Python. OCR technology allows for the digitization, reading, and interpretation of characters from physical documents. Successfully extracting and processing information from Cyprus IDs could significantly enhance the quality of services in various sectors, such as airports, police, and other industries. The development of an image-based tool requires an algorithm that can handle less-than-perfect character shapes and forms, necessitating the use of approximate matches—similar to an autocorrect function. Fields such as names and surnames need to align with a specific percentage of accuracy to meet the unit testing standards. This is achievable with confidence level scores that gauge the algorithm's accuracy in reflecting reality. Throughout the project, the challenges, issues, and potential areas for future enhancements are identified to better understand the present state of Cyprus IDs and to expand their current applications and use cases.

EXTRACT TEXT FROM A CYPRUS PERSONAL ID USING OCR

Angelos Ioannou

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

May, 2023

APPROVAL PAGE

Master of Science Thesis

EXTRACT TEXT FROM A CYPRUS PERSONAL ID USING OCR

Presented by

Angelos Ioannou

Research Supervisor

George Angelos Papadopoulos

Committee Member

Chris Mettouris

Committee Member

Alexis Yeratziotis

University of Cyprus

May, 2024

ACKNOWLEDGEMENTS

I would like to acknowledge and give my warmest thanks to my supervisor Dr. George Angelos Papadopoulos who made this work possible. His guidance and advice carried me through all the stages of writing my project. In addition, I would like to thank Dr. Savvas Savvides who has also guided and helped me throughout my entire journey. I would also like to give special thanks to my dear family and close friends for their continuous support, love and understanding when undertaking my research and writing my project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

1.1 Overview

Optical Character Recognition (OCR) technology stands as a pivotal innovation in the realm of digital information management, transforming printed and handwritten texts into machine-encoded formats with remarkable efficiency. This technology not only streamlines data entry processes, but also significantly reduces the margin for human error, thereby enhancing the overall accuracy and accessibility of digital data. As OCR technology continues to evolve, its application has become increasingly vital in the domain of identity verification—a critical function across numerous sectors including banking, security, and administrative services. In these contexts, the ability to accurately and swiftly recognize ID documents through OCR systems is paramount, ensuring seamless operations and bolstering security measures. This thesis delves into the specialized application of OCR for Cyprus ID recognition, exploring the unique challenges and opportunities presented by the multifaceted features of these identity documents, and aiming to elevate the standards of ID verification practices through a new python tool that was developed.

1.2 Aims and Objectives

1.2.1 Understanding OCR Technology

The first aim of this thesis is to delve into the intricacies of Optical Character Recognition (OCR) technology and then build upon the existing foundation that is already established. OCR stands as a cornerstone in the digital transformation of textual information, enabling the conversion of different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data. A comprehensive understanding of OCR involves exploring its historical evolution, the fundamental principles it operates on, the various algorithms and methodologies it employs, and its current technological standing. By dissecting the operational mechanisms of OCR systems, this paper aims to lay a foundational knowledge that will be pivotal in identifying areas of enhancement specifically tailored for the recognition of Cyprus Personal IDs.

1.2.2 Identifying Issues and Limitations

A critical aim of this research is to identify and analyze the prevalent issues, inconveniences, and errors inherent in current OCR technologies when applied to Cyprus IDs. These documents possess unique features and challenges, such as specific fonts, languages, layouts, and security elements, which may not be optimally processed by standard OCR systems. By examining these challenges in detail, the study intends to pinpoint the specific shortcomings of existing OCR solutions, such as inaccuracies in character recognition, difficulties with special characters or formats, and issues arising from the physical condition of the ID documents. This objective is crucial for paving the way toward targeted improvements in OCR technology for more efficient and error-free recognition of Cypriot IDs.

1.2.3 Optimization for Cyprus Personal IDs

Building on the identification of specific issues with current OCR applications to Cypriot IDs, the aim is to explore and develop optimizations that will enable faster, easier, and more accurate scanning of these documents. This involves investigating image preprocessing techniques, customized recognition algorithms, and existing machine learning models that are fine-tuned for the peculiarities of Cyprus Personal IDs. The goal is to enhance the OCR process in terms of speed, reliability, and efficiency, thereby reducing the error rate and improving the user experience in contexts where rapid and accurate document and image processing are essential.

1.2.4 Application Areas and Business Implications

Another significant aim is to explore the potential application areas of optimized OCR technology and its implications for businesses and other organizations, with a particular focus on airports. Airports are critical nodes in global travel and security infrastructure, where efficient and accurate document processing is paramount. The creation of a tool that is efficient, quick, and reliable could provide an immediate and effective solution for an industry consistently aiming to refine and streamline its methods. Therefore, by developing a sufficiently robust tool and sharing its core principles with industries like airport companies and others in the sector, further enhancements could be achieved.

1.2.5 Exploring the Capabilities of Google Tesseract

Finally, this thesis aims to conduct an in-depth exploration of Google Tesseract, an open-source OCR engine, to gauge its suitability and effectiveness in a variety of OCR applications, with a special emphasis on its application to Cyprus Personal IDs. Tesseract is known for its wide language support, adaptability, and extensive community-driven improvements. By testing its limits through rigorous experimentation, including its ability to handle the unique features of Cypriot IDs, this study intends to assess Tesseract's viability as a scalable and efficient OCR solution for diverse use cases. This will involve a comparative analysis with other OCR technologies, highlighting Tesseract's strengths, weaknesses, and potential areas for further development.

1.3 Structure of Thesis

This thesis is structured into eight chapters, each serving a specific purpose in exploring the application of Optical Character Recognition (OCR) technology for the recognition of Cyprus Personal IDs using the python tool. The chapters are organized in a logical progression, guiding the reader through the foundational concepts, objectives, methodologies, and findings of the research.

The introductory chapter lays the groundwork for the thesis by providing an overview of OCR technology and its significance in the context of identity verification. It highlights the vital role OCR plays in streamlining data entry processes and enhancing the accuracy and accessibility of digital data, emphasizing its importance in sectors such as banking, security, and administrative services. This chapter also outlines the specific aims and objectives of the research, encompassing the following key areas:

1) Understanding OCR Technology: Delving into the intricacies of OCR, including its historical evolution, fundamental principles, algorithms, and current technological standing.
2) Identifying Issues and Limitations: Analyzing the prevalent issues and limitations encountered when applying current OCR technologies to Cyprus Personal IDs, such as challenges with specific fonts, layouts, and security elements.
3) Optimization for Cyprus Personal IDs: Exploring and developing optimizations to enable faster, more accurate, and efficient scanning of Cyprus Personal IDs, including preprocessing techniques, customized recognition algorithms, and machine learning models.
4) Exploring the Capabilities of Google Tesseract: Conducting an in-depth exploration of the open-source OCR engine Google Tesseract, assessing its suitability and effectiveness for various OCR applications, with a special emphasis on Cyprus Personal IDs.

This second chapter provides a comprehensive review of existing literature and research pertaining to OCR technology, its applications, and the specific challenges associated with recognizing Cyprus Personal IDs. It serves as a foundation for understanding the current state of the art and identifying gaps or areas for further exploration. Chapter three (the methodology chapter) outlines the research approach and methods employed in the study. It details the data collection techniques, experimental design, and the specific tools and technologies utilized throughout the research process. This chapter also describes the evaluation metrics and criteria used to assess the performance and effectiveness of the proposed solutions. Chapter four presents the findings and results obtained from the research, including the outcomes of the experiments and analyses conducted. It encompasses the identification of issues and limitations in current OCR technologies when applied to Cyprus Personal IDs, as well as the optimizations and enhancements developed to address these challenges. The analysis section delves into the performance evaluation of the proposed solutions, examining factors such as accuracy, speed, and efficiency in recognizing Cyprus Personal IDs. Furthermore, it explores the capabilities and limitations of the Google Tesseract OCR engine in this specific context. The fifth chapter is building upon the results and analyses presented in the previous chapter, the discussion chapter provides an in-depth interpretation and evaluation of the findings. Chapter six delves into the practical aspects of implementing and testing the proposed OCR solutions. It provides an overview of the software tools and programming languages utilized throughout the project, detailing the specific libraries and components employed for OCR tasks. This chapter also explores the significance of unit testing in the context of OCR and outlines the testing methodologies employed to ensure the accuracy, reliability, and robustness of the developed solutions. It showcases examples of unit tests and their role in validating algorithms, handling errors, and ensuring the consistent performance of the OCR system. Chapter seven will evaluate the tool, highlighting the areas where it excelled and identifying those that need improvement. The assessment will utilize Cypriot IDs to effectively test and reveal the actual challenges encountered with real-life IDs. Additionally, the chapter will include graphs comparing different fields, illustrating the percentages of success and failure.

Chapter 2: Background

2.1 Image processing and OCR

2.1.1 Inception & Advancements of OCR

The inception and early applications of Optical Character Recognition (OCR) technology trace back to the mid-20th century, marking a significant milestone in the history of computing and information processing. OCR technology, designed to convert different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data, has revolutionized the way we manage and interact with textual information. A notable example is the invention by Dr. Emanuel Goldberg in the 1930s, who developed a machine that could read characters and convert them into telegraph code [1]. However, it was in the 1950s that OCR technology began to gain more practical applications, primarily in the business and banking sectors. The first major application was in the banking industry, where OCR was used to automate the processing of cheques. The "Electronic Reading Machine," developed by IBM in the late 1950s, was capable of reading characters on cheques, significantly reducing manual data entry and processing time [2]. The U.S. Postal Service was another early adopter of OCR technology. In the 1960s, the Postal Service began experimenting with OCR to automate the sorting of mail. This application required the OCR system to recognize and interpret handwritten and printed addresses on envelopes, a task that presented significant challenges due to the variability of handwriting styles and the quality of printed addresses [3]. In the academic realm, OCR technology found early applications in the digitization of printed materials. Libraries and universities began to use OCR systems to convert printed books and journals into digital formats, facilitating search and access to vast amounts of scholarly information. The Project Gutenberg, initiated by Michael Hart in 1971, is a prominent example of using OCR to digitize books, making them available in electronic format and accessible to a wider audience. The evolution of OCR technology from these early applications to its current state has been marked by significant advancements in computer vision, machine learning, and artificial intelligence. Today, OCR technology is not only more accurate and efficient but also capable of recognizing a wide array of fonts and styles, as well as overcoming challenges posed by poor quality documents and complex layouts.

2.1.2 Early Improvements in OCR Technology

Following the first applications of OCR, significant advancements were made to enhance its accuracy and efficiency. Early OCR systems were primarily hardware-based, with limited capabilities, and were often constrained by the quality of the printed material and the simplicity of the character sets they could recognize. The transition from hardware to software-based OCR systems marked a significant improvement, allowing for more complex algorithms and broader character recognition capabilities.

2.1.3 Integration with Digital Scanners

The late 20th century marked a significant era in the evolution of OCR technology, largely due to its integration with digital scanning devices. This fusion was more than just a technological convergence; it represented a paradigm shift in how text could be digitized and processed. Prior to this, OCR systems relied heavily on the quality of input from analog sources, which were often inconsistent and prone to errors. The advent of digital scanners brought about a revolution, allowing for the direct capture of printed text in a digital format, which significantly enhanced the quality of input data for OCR processing [4]. Digital scanners underwent rapid advancements during this period, with significant improvements in both hardware and software. The resolution of scanners increased dramatically, allowing for the capture of finer details in the printed material. This was crucial for OCR systems, as higher resolution images meant that even small or finely printed text could be captured with a high degree of clarity. Additionally, color scanning capabilities and improved dynamic range allowed OCR systems to distinguish text more effectively from complex backgrounds or colored fonts, further enhancing recognition accuracy. The integration of OCR technology with scanners also facilitated the development of more sophisticated document management systems. Documents could now be scanned, recognized, and archived in digital libraries, making them searchable and accessible in ways that were previously unimaginable. This integration laid the groundwork for the digital document workflows that are commonplace today, transforming industries such as legal, healthcare, and education by enabling efficient document handling, storage, and retrieval processes.

2.1.4 Advancements in Image Processing Algorithms

The evolution of image processing algorithms has been instrumental in the advancement of OCR technology. The late 20th and early 21st centuries saw significant developments in this area, with algorithms becoming increasingly sophisticated in dealing with a variety of challenges associated with digitizing printed text. Noise reduction, skew correction, and adaptive thresholding, as mentioned, became standard tools in the OCR preprocessing toolkit [5].

Noise reduction algorithms were developed to clean scanned images of random pixel variations or speckles that did not represent actual text, thereby reducing the potential for misrecognition. Skew correction algorithms addressed the issue of text alignment, automatically straightening text lines that were not perfectly horizontal due to scanning angles or page curvature. This was particularly important for ensuring accurate recognition of lines of text and their correct order in the document.

Adaptive thresholding represented a significant advancement in dealing with varying lighting conditions and print qualities. Traditional thresholding techniques, which converted grayscale images to black and white for analysis, struggled with documents that had non-uniform shading or were faded. Adaptive thresholding techniques, on the other hand, analyzed the image in smaller sections, adjusting the threshold dynamically to ensure that text was effectively distinguished from the background across the entire document.

2.1.5 The Impact of Machine Learning and AI on OCR Technology

The integration of machine learning (ML) and artificial intelligence (AI) into optical character recognition (OCR) systems heralded a transformative era for this technology. Traditionally, OCR systems relied on rule-based algorithms that required manual tuning and were limited by the specificity of their programmed instructions. The advent of ML and AI, particularly with the introduction of neural networks in the late 20th century, marked a paradigm shift in how OCR systems were developed and operated [6]. Neural networks, inspired by the biological neural networks that constitute animal brains, are a set of algorithms modeled loosely after the human brain, designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling, or clustering raw input. This revolutionary approach allowed OCR systems to process vast amounts of data, learning from each interaction, which significantly enhanced their recognition capabilities. One of the most significant contributions of ML and AI to OCR technology is the development of convolutional neural networks (CNNs). CNNs are particularly well-suited for image recognition tasks, making them ideal for OCR applications. They can take an image input, assign importance to various aspects/objects in the image, and differentiate one from the other. When applied to OCR, CNNs analyze pixels in scanned documents or images, learn from the structure and layout of the text, and effectively recognize and convert them into digital formats. This has dramatically improved the accuracy with which OCR systems can identify text, even when dealing with complex layouts, fonts, and backgrounds.

Moreover, the incorporation of AI and ML has enabled OCR systems to understand and interpret handwriting, a task that was previously fraught with high error rates due to the variability and complexity of human handwriting. Advanced machine learning models, trained on datasets comprising diverse handwriting samples, have become adept at recognizing a wide array of handwriting styles, thereby extending the utility of OCR beyond typed text to include handwritten notes, forms, and documents.

The impact of ML and AI on OCR technology extends to the ability to recognize and process text in low-resolution images or under suboptimal conditions. By employing sophisticated image enhancement techniques powered by AI, OCR systems can now extract text from images plagued by poor lighting, low contrast, or noise. This resilience to adverse conditions has expanded the range of applications for OCR technology, making it possible to extract text from photographs, screenshots, and other non-traditional sources. Furthermore, the adaptability of ML-driven OCR systems has facilitated their application across various languages and scripts. By training on diverse datasets, these systems can recognize and process text in multiple languages, including those with complex scripts such as Arabic, Chinese, and Cyrillic, thereby broadening the global applicability of OCR technology.

In essence, the integration of machine learning and artificial intelligence into OCR has not only enhanced its accuracy and efficiency but has also expanded its scope to encompass a wider range of text recognition tasks. This evolution reflects a broader trend in technology, where AI and ML are driving significant advancements across multiple domains. As OCR technology continues to evolve, the continued integration of AI and ML promises to unlock even greater capabilities, further solidifying OCR's role as a cornerstone of digital information management.

2.1.6 Language and Character Set Expansion

Early OCR systems were limited in their ability to recognize texts in languages other than English or with complex character sets. The development of Unicode and the improvement in OCR algorithms expanded the technology's capability to process texts in multiple languages, including those with non-Latin scripts such as Chinese, Arabic, and Cyrillic. This globalized the applicability of OCR, making it an invaluable tool in international contexts.

2.1.7 Real-Time OCR and Mobile Applications

The advent of powerful mobile devices equipped with high-quality cameras led to the development of real-time OCR applications. These applications allowed users to capture text from any source using their mobile device's camera and instantly convert it into editable digital text. This advancement greatly expanded the use and accessibility of OCR technology, making it a tool for everyday tasks such as translating text on signs, digitizing printed documents, and more.

2.1.8 Cloud-Based OCR Services

The shift towards cloud computing saw the emergence of cloud-based OCR services, which offered scalable, high-accuracy OCR capabilities without the need for extensive local computational resources. These services leveraged the power of advanced server-based OCR algorithms, providing access to state-of-the-art OCR technology to a broader range of users and applications. [7]

2.1.9 OCR in Complex Document Analysis

Advancements in OCR technology have also enabled the processing of complex document layouts beyond simple text recognition. Modern OCR systems can now understand and extract information from documents with complex structures, such as tables, forms, and multi-column layouts, maintaining the original formatting and structure of the document in the digital output. [8]

2.1.10 Integration with Other Technologies

OCR technology has increasingly been integrated with other technologies to enhance its utility. For example, combining OCR with natural language processing (NLP) has enabled the extraction of not just text but meaningful information from documents, facilitating applications like automatic data entry, content indexing, and semantic analysis. [9]

2.1.11 Future Directions

Continued advancements in OCR technology focus on improving accuracy, speed, and the ability to understand context within the text. Emerging technologies like deep learning and computer vision continue to push the boundaries of what OCR can achieve, promising even more sophisticated and versatile OCR applications in the future. [10]

2.2 OCR Libraries

2.2.1 Tesseract

Tesseract OCR, an open-source Optical Character Recognition (OCR) engine, stands out as one of the most comprehensive and widely used OCR tools in the digital processing realm. Initially developed by Hewlett-Packard Laboratories in 1985 and later released as open-source software in 2005, Tesseract has evolved significantly, especially after Google adopted the project in 2006, enhancing its capabilities and extending its reach.

2.2.1.1 Historical Background and Evolution

Tesseract was conceived with the aim of providing a high-quality OCR tool that could convert images of text into a variety of digital text formats. Its early versions were primarily focused on English text recognition and were used internally by HP for tasks such as document scanning and management. The transition to an open-source model marked a pivotal point, enabling developers worldwide to contribute to its improvement and extend its functionalities. This collaborative effort led to the rapid evolution of Tesseract, incorporating advancements in OCR technology, and expanding its language support beyond English to include a multitude of languages and scripts. [4]

2.2.1.2 Core Features and Capabilities

One of Tesseract's distinguishing features is its ability to recognize more than 100 languages out of the box, making it an invaluable tool for global digital document processing tasks. It supports various image formats, including TIFF, JPEG, and PNG, and can handle multi-page TIFF documents, which are common in scanned document archives.

Tesseract's architecture is modular, comprising several components responsible for different stages of the OCR process, such as layout analysis, line finding, word recognition, and character classification. This modularity allows for targeted improvements and customization, enabling developers to adapt Tesseract for specific OCR tasks or integrate it with other software solutions. [4]

2.2.1.3 Advancements in Accuracy and Performance

The significant leap in Tesseract's capabilities came with the introduction of LSTM (Long Short-Term Memory) networks in version 4.0. This integration of deep learning techniques marked a substantial improvement in Tesseract's text recognition accuracy, particularly for complex documents and scripts. LSTM networks enhanced Tesseract's ability to understand contextual nuances and recognize text with higher precision, even in challenging conditions such as poor lighting, low resolution, or distorted text. [4]

2.2.1.4 Application in Diverse Domains

Tesseract's versatility allows for its application across various domains, from automating data entry and processing scanned documents to aiding in historical document digitization and accessibility projects. Its ability to be trained on new datasets makes it particularly useful for specialized OCR tasks, such as recognizing uncommon fonts or scripts in archival materials.

2.2.1.5 Integration and Extensibility

Developers can integrate Tesseract with other software tools and platforms, thanks to its API and support for various programming languages. This flexibility has led to the development of numerous applications and services that leverage Tesseract's OCR capabilities, from mobile apps that convert images to text to complex document management systems that automate data extraction and indexing.

2.2.1.6 Community Support and Development

The open-source nature of Tesseract has fostered a vibrant community of developers and users who contribute to its ongoing development, documentation, and support. This community-driven approach ensures that Tesseract stays at the forefront of OCR technology, continually adapting to new challenges and requirements in the field of digital text recognition.

2.2.2 ABBYY

Another very famous OCR library is ABBYY FineReader. ABBYY FineReader is a highly acclaimed OCR (Optical Character Recognition) software developed by ABBYY, a Russian company known for its document recognition and data capture software. FineReader is renowned for its exceptional accuracy in converting scanned documents, PDFs, and photographs into editable and searchable digital formats. It stands out for its advanced language support, handling more than 190 languages, and its ability to recognize and preserve the layout and formatting of the original document, including tables, headers, footers, and graphs.

ABBYY FineReader employs sophisticated AI and machine learning technologies to enhance its OCR capabilities, making it particularly effective in processing complex document layouts and recognizing text with high precision. This software is widely used in various professional fields, including legal, education, government, and business sectors, for tasks such as digitizing paper archives, automating document processing workflows, and facilitating document collaboration and management.

The software's versatility and robust feature set, including its ability to integrate with other enterprise solutions and its user-friendly interface, have contributed to its popularity and widespread adoption across different industries. [11]

2.2.3 OCRopus Overview

Another notable OCR library similar to Tesseract is OCRopus. OCRopus is an open-source document analysis and OCR system that uses state-of-the-art machine learning techniques for text recognition, primarily focusing on the processing of large collections of documents and books, especially those that are part of digital libraries.

OCRopus is designed with a modular architecture, allowing for the flexible integration of different OCR components, including layout analysis, character recognition, and post-processing modules. This modularity makes it highly adaptable to various text recognition tasks and document types.

Neural Network Models: One of the key features of OCRopus is its use of recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, for character and text line recognition. This approach enables OCRopus to achieve high accuracy levels, even on challenging documents with complex layouts or degraded text.

Language and Script Support: While OCRopus provides robust support for Latin script, its flexible architecture allows for the training and integration of models for other scripts and languages, making it a versatile tool for global OCR projects.

Community and Development: As an open-source project, OCRopus benefits from contributions from a global community of developers and researchers. Continuous improvements and updates are made to enhance its performance and extend its capabilities.

Applications: OCRopus is particularly well-suited for academic and research applications, including the digitization of historical texts and the processing of academic papers and technical documents. Its high accuracy and adaptability make it a valuable tool for libraries, archives, and research institutions aiming to digitize and make searchable large volumes of printed material. [12]

2.2.4 EasyOCR

EasyOCR is another tool designed for Optical Character Recognition (OCR), which allows the conversion of different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data.

EasyOCR is an open-source tool developed to make Optical Character Recognition (OCR) more accessible and efficient, particularly known for its ability to handle multiple languages and its ease of use. It employs advanced deep learning techniques, using Convolutional Neural Networks (CNNs) for the extraction of features from images and sequence models like Recurrent Neural Networks (RNNs) or transformers to interpret the sequence of characters within the text areas identified. This tool has emerged from a broader initiative aimed at democratizing access to powerful machine learning and computer vision technologies, making it a valuable asset for developers and researchers alike. EasyOCR's standout features include its high accuracy in text recognition, support for a wide range of languages, and a user-friendly design that facilitates easy integration into various projects. It is widely used in numerous applications, ranging from digitizing documents to automated data entry and license plate recognition. Despite its robust capabilities, the performance of EasyOCR can be influenced by the quality of input images and the complexity of text layouts, which underscores the ongoing efforts by the developer community to refine and expand its functionalities to meet evolving needs and challenges.

Development and Features: EasyOCR was explored for its performance with Latin characters under various conditions of image degradation, such as character-background intensity difference, Gaussian blur, and relative character size.

By examining the paper "Analysis of Optical Character Recognition using EasyOCR under Image Degradation" [13], the authors take a very intensive look at how this OCR can be proven very effective when faced with unoptimized and blurry images. The study found that EasyOCR is particularly effective in distinguishing between unique lowercase and uppercase characters but tends to favor uppercase for similar shapes like C, S, U, or Z. The OCR's performance varied significantly with changes in character-background intensity difference, with confidence scores ranging from 3% to 80%. It was noted that higher differences could cause confusion between characters like 'o' and '0', or 'i' and '1'. Additionally, while increased Gaussian blur generally hindered recognition, it actually improved recognition for certain letters like 'v'. The size of the image also had a significant impact on character detection, with failures occurring as sizes decreased to 40% to 30% of the original. These findings provide insights into EasyOCR's capabilities and limitations under various conditions of image degradation, highlighting its strengths in character distinction and the effects of image quality on its performance.

2.2.5 OCR Libraries Comparison

Feature / OCR Tool	Google Tesseract	ABBYY	OCRopus	EasyOCR
Languages Supported	Supports over 100 languages	Supports 200+ languages	Primarily focused on English, with support for other Latin-script languages	Supports 80+ languages including Latin, Cyrillic, and Asian languages
Ease of Use	Moderate; requires some setup and familiarity with OCR concepts	Moderate; user-friendly interfaces but can be complex due to extensive features	Moderate; aimed at researchers and requires understanding of OCR and machine learning concepts	High; designed for easy integration and use with minimal setup
Support for Python	Yes, through PyTesseract, a Python wrapper	Yes, through ABBYY Cloud OCR SDK or FineReader Engine (with Python wrappers or API calls)	Natively designed to be used with Python	Yes, natively designed to be used with Python
Performance	Excellent; highly dependent on pre-processing and training for custom needs	Good; known for high accuracy and extensive feature set	Good; performance can vary based on training data and specific use cases	Very Good; competitive performance, especially considering ease of use and speed

Table 1: Comparison between popular OCR libraries on languages supported, ease of use, support for Python and performance.

2.2.6 Why Tesseract was chosen

When evaluating Optical Character Recognition (OCR) tools for specific projects such as ID OCR, where accuracy, language support, and customization are paramount, Google Tesseract emerges as a superior choice among its peers, including ABBYY, OCRopus, and EasyOCR. Tesseract's longstanding reputation, open-source nature, and extensive language support, including over 100 languages, make it an exceptionally versatile tool suitable for diverse global applications, including the intricate task of ID OCR, which often involves capturing text from varied and complex backgrounds.

One of the standout features of Tesseract is its adaptability and the ability to train the engine for specific use cases. This is particularly beneficial for ID OCR, where the tool might need to recognize specialized fonts or formats. Tesseract allows for fine-tuning and custom training, which can significantly enhance accuracy when dealing with the unique challenges presented by ID documents, such as passports, driver's licenses, and national ID cards. This level of customization ensures that Tesseract can be precisely tailored to meet the nuanced requirements of ID OCR projects, thereby reducing errors and improving reliability.

Moreover, Tesseract's support for Python through PyTesseract, a Python wrapper, aligns perfectly with the current trends in software development, especially in data science and machine learning projects. Python's widespread adoption due to its simplicity and powerful libraries makes Tesseract easily integrable into existing workflows. This seamless integration facilitates the automation of ID OCR processes, enabling efficient processing of large volumes of ID documents without the need for extensive manual intervention.

Performance is another critical factor in the selection of an OCR tool for ID OCR projects. Tesseract's performance, particularly when optimized through pre-processing and custom training, is commendable. It has shown to be effective in recognizing text across various ID formats and backgrounds, which is crucial for ensuring that all pertinent information is accurately captured and processed. This reliability in performance underpins the trust many developers and organizations place in Tesseract for their OCR needs.

Furthermore, Tesseract's open-source model offers unparalleled transparency and flexibility, allowing developers to understand the inner workings of the OCR engine and contribute to its enhancement. This collaborative approach fosters continuous improvement and adaptation of the tool, ensuring it remains at the forefront of OCR technology. The active community around Tesseract also means that developers have access to a wealth of knowledge and support, which can be invaluable in troubleshooting and enhancing ID OCR projects.

While ABBYY, OCRopus, and EasyOCR each have their strengths, such as ABBYY's extensive language support and EasyOCR's user-friendly design, they fall short in offering the same level of customization, integration ease, and performance optimization that Tesseract does, especially in the context of complex OCR tasks like ID OCR. The ability to fine-tune Tesseract to specific project needs, combined with its robust performance and Python support, positions it as the superior choice for projects that demand high accuracy, flexibility, and efficiency. In conclusion, Tesseract's comprehensive language support, customization capabilities, Python integration, and reliable performance, coupled with its open-source nature, make it an unrivaled tool in the OCR landscape, particularly suited to the specialized demands of ID OCR projects. Its proven track record and adaptability ensure that it remains the go-to OCR solution for developers and organizations looking to harness the power of OCR technology to streamline and enhance their ID document processing workflows.

Chapter 3: Related Work

3.1 Designing a Real-Time-Based Optical Character Recognition to Detect ID Cards

The paper titled "Designing a Real-Time-Based Optical Character Recognition to Detect ID Cards" [14] presents a study focused on developing a real-time ID card detection system utilizing Optical Character Recognition (OCR) technology. This research aims to provide an administrative solution in Indonesia by facilitating the processing of identity cards using real-time OCR. Below is an overview of the study, including the software used, testing methodology, results, and effectiveness of the system.

3.1.1 Software Used and Application

The researchers employed a combination of software and technologies to design the real-time OCR system for ID card detection. The primary software used was Easy OCR, which is Pytorch-based, indicating its reliance on deep learning frameworks for OCR tasks. For ID card detection, they utilized TensorFlow's object detection API, specifically employing the SSD MobileNet V2 FPNLite 320x320 as the pre-trained model. This choice of software and models suggests a focus on achieving a balance between detection accuracy and computational efficiency, catering to real-time processing requirements.

3.1.2 Testing Methodology

The testing of the system involved the collection of ID card images using a webcam under various lighting conditions and orientations to mimic real-world scenarios. The images were then labeled using a labeling tool to prepare for the training process. With a dataset comprising only 20 photos, the researchers trained the TensorFlow object detection algorithm, iterating through 3000 training steps to refine the model's ability to accurately detect ID cards from the webcam feed.

3.1.3 Results

Upon completing the training process, the system achieved a loss of approximately 0.17 and an accuracy metric of 0.95. These results indicate a high level of precision in ID card detection, suggesting that the trained model could effectively distinguish and recognize ID cards from the webcam images.

3.1.4 Effectiveness

The study concludes that the real-time ID card detection tool using OCR operates effectively, as evidenced by the high accuracy metric obtained during testing. The system's ability to run well under various lighting conditions and orientations demonstrates its robustness and potential applicability in practical administrative contexts within Indonesia. The use of a webcam and commonly accessible software components like Easy OCR and TensorFlow makes the system approachable for real-world implementation, offering a promising solution for automating ID card processing tasks. Overall, this research showcases the potential of combining deep learning models with real-time OCR technologies to create efficient and accurate systems for ID card recognition, which can be particularly useful in streamlining administrative processes and enhancing data entry accuracy in various applications.

3.2 Citizen ID Card Detection Using Image Processing and Optical Character Recognition

The paper titled "Citizen ID Card Detection Using Image Processing and Optical Character Recognition" [15] delves into the creation of a system designed to automate the detection and recognition of Indonesian Electronic ID cards, which have been in widespread use since 2011. The study addresses challenges such as the difficulty in accurately detecting ID card fields and recognizing character data on the cards. Below is a detailed overview of the study, including the methodologies used, testing procedures, results obtained, and the overall effectiveness of the developed system.

3.2.1 Software Used and Application

The research team developed a technique that combines image processing and Optical Character Recognition (OCR) to detect and recognize data on electronic ID cards. The study does not specify the exact software used for OCR but mentions that the system was embedded in a website interface utilized by an automotive company, suggesting that the approach was designed to be integrated into existing digital platforms for real-time ID card processing.

Testing Methodology: The methodology involved the use of image processing techniques to detect the presence of ID cards within digital images, followed by the application of OCR technology to recognize and extract textual information from the detected ID card regions. The study likely involved the preprocessing of images to enhance the quality and readability of text on the ID cards, which is a common practice in OCR applications to improve recognition accuracy.

Results: The system achieved a remarkable accuracy rate of 98% in detecting ID cards using the proposed image processing and OCR techniques. This high level of accuracy indicates that the system was highly effective in identifying and processing ID card images, successfully overcoming challenges associated with field detection and character recognition.

Effectiveness: The effectiveness of the system is demonstrated by its high accuracy rate and its successful implementation in a practical application within the automotive industry. By achieving 98% accuracy in ID card detection, the system proved to be a reliable solution for automating the processing of electronic ID cards, potentially streamlining administrative tasks and reducing manual errors in data entry. The integration of the system into a web interface further highlights its practical applicability and the potential for deployment in various digital platforms requiring ID card verification and data extraction. In summary, the research presented in "Citizen ID Card Detection Using Image Processing and Optical Character Recognition" showcases a successful application of image processing and OCR technologies in automating the detection and recognition of ID cards. The high accuracy and practical implementation of the system highlight its potential to enhance efficiency and accuracy in processes that require

ID card verification, making it a valuable tool for various industries and administrative applications.

3.2.2 An Overview And Applications Of Optical Character Recognition

Optical Character Recognition (OCR) involves the electronic or mechanical transformation of images from scanned or photographed text, whether it's from typed or printed sources, into text that computers can process. This technology finds extensive application in data entry from various paper-based sources like passports, invoices, bank statements, receipts, business cards, and postal mail, among others. By converting printed material into a digital format, OCR facilitates the editing, searching, compact storage, and online display of texts, in addition to enabling their use in computational processes such as text mining, machine translation, and text-to-speech applications. Positioned at the intersection of pattern recognition, artificial intelligence, and computer vision, OCR technology also encompasses the digital conversion of handwritten or printed text into machine-readable images, enhancing the accessibility and searchability of document content in digital forms. The significant body of research and literature on OCR underscores its foundational role and ongoing development in the field. This paper provides an overview of OCR technology, highlights seminal research contributions that have significantly advanced character recognition capabilities, and explores the diverse applications of OCR across various sectors, concluding with a synthesis of the key findings and contributions. [16]

3.2.3 Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study

The method of Optical Character Recognition (OCR) plays a crucial role in transforming printed texts into formats that can be edited electronically. OCR is a widely embraced technique across numerous fields due to its utility. The effectiveness of OCR technology often hinges on the methodologies employed for text preprocessing and segmentation. Challenges can arise in text extraction from images due to variations in text size, style, orientation, and the complexity of the image background. This paper initiates with an overview of the OCR methodology, delves into the historical development of the open-source OCR tool Tesseract, its structural design, and presents findings from experiments conducted using Tesseract on various types of images. The paper culminates in a comparative analysis between Tesseract and the commercial OCR software Transym OCR, specifically focusing on the task of extracting vehicle numbers from license plates. This comparison assesses both tools across different metrics to evaluate their performance in this specific application. [17]

3.2.4 Automatic OCR system for Colombian DNIs

This paper introduces a system designed for the optical character recognition of text on Colombian national identity cards. It incorporates an image acquisition algorithm that operates via Windows drivers, coupled with an OCR methodology that effectively removes the holographic background, which is a primary challenge in this context. The efficacy of this approach was evaluated using a dataset of 34 identity cards, achieving a 92.6% accuracy rate in recognizing characters. [18]

3.2.5 Image Preprocessing for Improving OCR Accuracy

Digital cameras serve as practical tools for capturing images due to their speed, flexibility, portability, non-intrusive nature, and affordability. Nonetheless, when used in OCR tasks, they encounter several challenges, including geometric distortions. This study focuses on the preprocessing phase that precedes text recognition, particularly concerning images taken with a digital camera. The significance of image preprocessing in OCR applications is underscored by experiments using FineReader 7.0 as the underlying recognition software, which affirms its critical role. [19]

3.2.6 Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)

The pervasive presence of handwritten documents in various aspects of human interactions underscores the significant practical value of Optical Character Recognition (OCR). OCR is a technological field that facilitates the conversion of different types of documents or images into data that is accessible, editable, and searchable. Over the past ten years, the application of artificial intelligence and machine learning methodologies has been explored extensively by researchers aiming to automate the analysis of both handwritten and printed documents for their conversion into digital formats. This review paper aims to consolidate the findings of research conducted on the OCR of handwritten documents and to suggest potential avenues for future investigation. Through a Systematic Literature Review (SLR), research papers focusing on handwritten OCR (and related subjects) published from 2000 to 2019 were gathered, synthesized, and examined. Established electronic databases were utilized according to a predefined review protocol, employing keyword searches as well as forward and backward reference searches to comprehensively identify relevant literature. Following a meticulous study selection process, 176 articles were chosen for inclusion in this SLR. The purpose of this review is to showcase the latest achievements and methodologies in OCR, while also identifying areas that require further research by pointing out existing gaps in the literature. [20]

Chapter 4: Cyprus ID Analysis

4.1 Overview

The introduction of electronic ID cards first took place in the early part of 2015. There were minor modifications made to them in 2020 to fully adhere to the Regulation (EU) 2019/1157, enacted by the European Parliament and the Council on 20 June 2019, aimed at enhancing the security of identity cards for Union citizens and the residence documents provided to Union citizens and their families who are utilizing their right to free movement. It’s mandatory for individuals who are twelve years old or older to obtain and hold a Civil Identity Card, while for those under twelve, it’s provisional. The Republic of Cyprus has issued two ID cards. Before 2015, IDs were not biometric and included the following fields: Cyprus ID (prior to 2015).

4.2 Layout

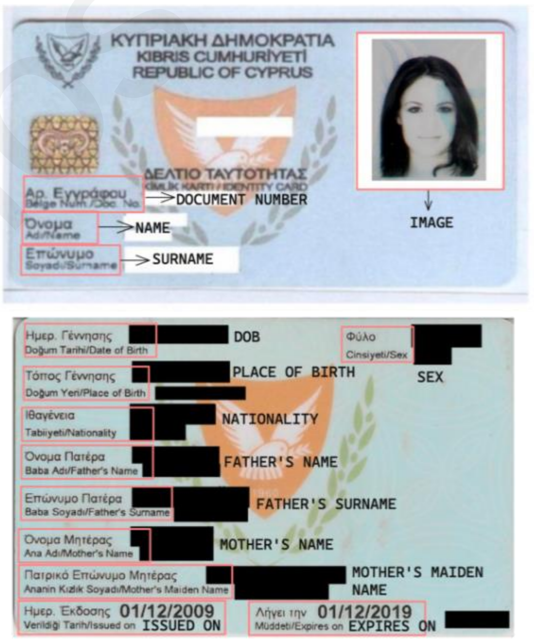

ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ
KIBRIS CUMHURİYETİ
REPUBLIC OF CYPRUS
ΔΕΛΤΙΟ ΤΑΥΤΟΤΗΤΑΣ
KİMLİK KARTI / IDENTITY CARD
Αρ. Εγγράφου Belge Num./Doc. No → DOCUMENT NUMBER
Όνομα Adı/Name → NAME
Επώνυμο Soyadı/Surname → SURNAME
IMAGE
Ημερ. Γέννησης Doğum Tarihi/Date of Birth [redacted] DOB
Φύλο Cinsiyeti/Sex [redacted] SEX
Τόπος Γέννησης Doğum Yeri/Place of Birth [redacted] PLACE OF BIRTH
Ιθαγένεια Tabiiyeti/Nationality [redacted] NATIONALITY
Όνομα Πατέρα Baba Adı/Father's Name [redacted] FATHER'S NAME
Επώνυμο Πατέρα Baba Soyadı/Father's Surname [redacted] FATHER'S SURNAME
Όνομα Μητέρας Ana Adı/Mother's Name [redacted] MOTHER'S NAME
Πατρικό Επώνυμο Μητέρας Ananin Kızlık Soyadı/Mother's Maiden Name [redacted] MOTHER'S MAIDEN NAME
Ημερ. Έκδοσης Verildiği Tarih/Issued on 01/12/2009 ISSUED ON
Λήγει την Müddeti/Expires on 01/12/2019 EXPIRES ON [redacted]

Image 1: Cyprus IDs prior to 2015 that were not biometric

- Document Number
- Name
- Surname
- Sex
- Date of Birth
- Place of Birth
- Nationality
- Father’s Name
- Father’s Surname
- Mother’s Name
- Issued on
- Expires on

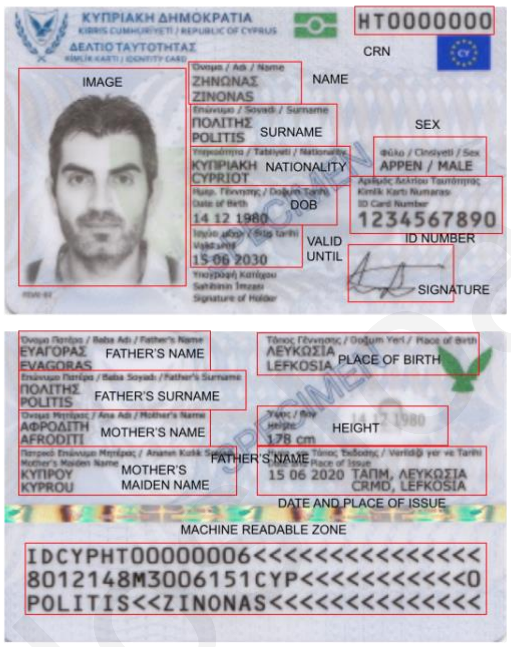


Image 2: Biometric Cyprus IDs that replaced the old ones in 2015

Cyprus Biometric ID card (2020) contains the following fields:

- CRN (Card Reference Number)
- Name
- Surname
- Place of Birth
- Date of Birth
- Valid Until
- Signature of Holder
- ID Card Number
- Father’s Name
- Father’s Surname
- Mother’s Name
- Mother’s Maiden Name
- Nationality
- Sex
- Height
- Date and Place of Issue

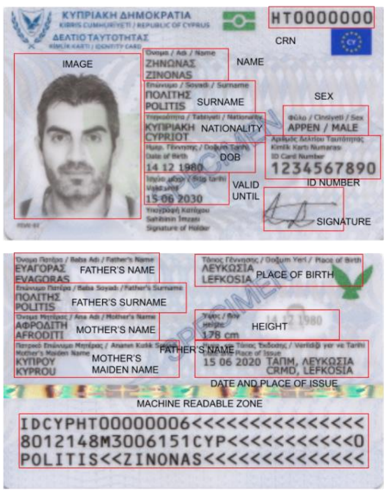
Moreover, the updated IDs feature a section designated as the Machine Readable Zone (MRZ). This comprises a three-line optical character recognition (OCR) code encapsulating fundamental details about the holder like their name, date of birth, and nationality. The MRZ facilitates swift and straightforward verification of the holder's identity by border control and other relevant authorities.

When comparing the IDs issued in 2020 with those from 2015, there are a few minor alterations. The ID card number has been shifted slightly upward, the text in all fields is now bolder and displayed in a larger font size, and a rectangle indicating Cyprus's membership in the European Union has been added right below the CRN.

The IDs feature text in three distinct languages. Every text field is defined in Greek, English and Turkish.

4.3 Linguistic Analysis

Investigating ID cards to enhance the development of a proficient and dependable OCR necessitates a comprehensive understanding of the varied character encoding protocols across different fields. While IDs exhibit variations in the fields they showcase, certain standards are commonly applicable to a degree. In this section, every field of the new Cypriot ID will be meticulously examined to gain deeper insights into the protocols involved in each, and to formulate algorithms that are optimally tailored to its requirements.



Front Side

Cypriot Emblem: Positioned at the very top left of the ID, the Cypriot Flag is showcased. Its bluish color symbolizes the country that the ID represents.

Name of Country: This field exhibits the name "Cypriot Republic of Cyprus" in Greek alphabet characters and, beneath it, the same is displayed in both Turkish and English (latin alphabet), utilizing a marginally smaller font size. All text is presented in uppercase letters.

Type of Identification: This segment confirms that the card serves as an identity card. Similar to the Name of the Country, the displayed information adheres to the exact same guidelines.

Biometric ID Symbol: All IDs categorized as biometric feature a symbol, which is a square with a circle in its center, to validate the ID's authenticity.

CRN: The Card Reference Number (CRN) is a distinct 9-digit identifier allocated to every Cypriot identity card. Serving as a crucial security element, it safeguards the holder's identity and mitigates fraudulent activities. The CRN is utilized for numerous purposes, such as:

- Recognizing the identity card within governmental databases
- Associating the identity card with additional documents, like passports and driving licenses
- To prevent fraud and counterfeiting

Each ID's CRN begins with the Latin letters "CR," followed by 7 unique digits in a string. CRN numbers are generated randomly, ensuring that every identity card possesses a distinct number, enhancing its resistance to forgery and counterfeiting. The CRN is produced through a cryptographic algorithm, intentionally designed to be exceedingly challenging to decipher.

Name, Surname, Place of Birth: These sections present strings, articulated in both the Greek language and English, utilizing uppercase letters. Every field reveals the information bilingually, in Greek and English. Furthermore, the sole field on the Cypriot ID that is exhibited in bold is the name, in both aforementioned languages.

Date of Birth, Valid Until: Rather than showcasing strings of information, these fields present evenly spaced numbers. Dates in both fields adhere to the DD MM YYYY format.

Signature of Holder: Positioned at the extreme bottom right of the ID is the holder’s signature, captured as a vector at the time the card was issued. The signature is notably larger than text in other fields, making it easily distinguishable on the card.

ID Card Number: Located directly above the holder’s signature, this field begins with four zeroes, followed by the ID's six-digit number, totaling 10 numeric digits. These digits are bolded and share the same font size as the CRN number.

Special Field: A small section on the bottom left side of the ID contains the first four digits of the holder’s surname, a hyphen, and then two numerical digits. The first digit is consistently a zero, while the purpose of the second is unclear. This field utilizes a notably small font size.

Holder’s Photograph: The left side of the card features the holder’s photograph in grayscale, capturing a portrait from the hair to the neck.

Cyprus Flag Iridescent: Adjacent to the holder’s photograph, the symbol of the Cyprus Republic is displayed with an iridescent effect, diverging from the grayscale used in other ID elements.

Back Side

Father’s Name, Father’s Surname, Mother’s Name, Mother’s Maiden Name, Nationality, Sex: These segments display strings, expressed in both Greek and English, using uppercase letters. Each field conveys the information in a bilingual manner, in both Greek and English.

Height: This section showcases the cardholder's height in centimeters at the time the ID was issued, represented with three digits followed by the "cm" unit. That value is a number.

Cyprus Emblem Dove: Positioned in the top right corner, the dove from the Cypriot emblem is depicted with a slightly greenish-blue hue that exhibits a glittering effect.

Oval With Portrait: Directly beneath the dove, an oval shape encases the card holder's portrait, presented with a holographic effect, and behind it, digits are displayed in the same format as the date of birth.

Date and Place of Issue: This section displays the card's issue date in the DD MM YYYY format, along with the place of issue in both Greek and English. Specifically, it shows the issuing department, followed by a comma (,) and the city.

MRZ (Machine Readable Zone): This section is designed for machine reading. In the first line, the Latin string characters “IDCYP” are displayed, followed by the CRN number. After the CRN, a string is displayed, the purpose of which is not specified, followed by a series of fifteen “<” symbols. In the second line, seven numbers (string) are shown, followed by a Latin character, then another seven numbers. After these, the characters “CYP” appear, followed by eleven “<” symbols, and finally, a numeric value in the last position of the line. In the last line, the holder’s surname appears in uppercase, followed by two “<<” symbols, the name in uppercase, and depending on the length of the name and surname, the remaining space is filled with “<” symbols.

4.4 Fixed Text on IDs

Name of Country: This field displays the "Cypriot Republic of Cyprus" in Greek, Turkish, and English, using uppercase letters. The consistent presentation of the country's name across IDs serves as a fixed text element.

Type of Identification: This segment clearly states the card's purpose as an identity card, using standardized wording in multiple languages to maintain consistency across all issued IDs.

Biometric ID Symbol: The presence of a biometric symbol (a square with a circle in its center) on IDs classified as biometric is a standardized feature indicating the card's authenticity and adherence to biometric data protocols.

CRN (Card Reference Number): The format "CR" followed by 7 digits is a fixed format used to uniquely identify each card. Although the numbers themselves vary, the "CR" prefix is a standard element.

Document Fields: Fields like "Name," "Surname," "Date of Birth," "Place of Birth," "Nationality," "Sex," "Father’s Name," "Father’s Surname," "Mother’s Name," and "Mother’s Maiden Name" are consistent across all IDs, serving as fixed text fields where only the specific details of the cardholder change.

MRZ (Machine Readable Zone): The structure of the MRZ, including the format and positioning of the "IDCYP" prefix, CRN, and other details, follows international standards for machine-readable travel documents, making it a section with fixed formatting and text elements.

4.5 Data-types of fields

Alphanumeric Data

- CRN (Card Reference Number): This is a unique identifier consisting of a combination of letters and numbers. The "CR" prefix is followed by seven numeric digits, making it alphanumeric.
- ID Card Number: Typically starts with zeroes and is followed by a unique set of numbers, but the inclusion of leading zeroes makes this field alphanumeric as well.

Textual (String) Data

- Name, Surname, Place of Birth, Father's Name, Father's Surname, Mother's Name, Mother's Maiden Name, Nationality, Sex: These fields are purely textual, containing characters that represent names and other personal information. The data is presented in uppercase letters and includes both Greek and English languages, accommodating Cyprus's bilingual population.
- Type of Identification: This field also contains textual data, indicating the document's purpose as an identity card.

Numeric Data

- Date of Birth, Valid Until, Date and Place of Issue: These fields contain numeric data in the form of dates, represented in the DD MM YYYY format.
- Height: This field contains numeric data representing the cardholder's height in centimeters.

Biometric Data

- Signature of Holder: The signature is stored as an image but represents biometric data because it is unique to the individual and used for personal verification.

- Holder’s Photograph: This is also biometric data, presented as a grayscale image, and used for visual identification of the cardholder.

Graphical Data

- Cypriot Emblem, Cyprus Flag, Biometric ID Symbol: These fields contain graphical data in the form of symbols and emblems that are specific to the Cypriot identity and the ID card's security features.

Optical and Holographic Data

- Special Security Features (like the Cyprus Emblem Dove, Oval With Portrait): These features include optical and holographic elements that are part of the card's security measures, making duplication difficult.

Machine-Readable Data

- Machine Readable Zone (MRZ): This area contains a mix of alphanumeric data designed for electronic scanning and reading. It includes a series of characters that encode the cardholder's personal details, following an international standard format for machine-readable travel documents.

4.6 Location

Front Side

- Cypriot Emblem: Positioned at the top left corner, symbolizing national identity.
- Name of Country: Displayed at the top, usually centered or aligned with the emblem, indicating the issuing country in multiple languages.
- Type of Identification: Located near the top, often below or beside the country name, clearly stating the document's purpose.
- CRN (Card Reference Number): Typically placed in the upper portion, easily visible for quick reference.
- Name, Surname: These personal identification fields are prominently displayed, usually in the upper-middle section, to facilitate easy reading.

- Place of Birth, Date of Birth: Located centrally or just below the name and surname, providing essential personal details.
- Valid Until: Often found near the date of birth, indicating the ID's expiry date.
- Signature of Holder: Positioned towards the bottom, usually on the right side, to provide a personal authentication feature.
- ID Card Number: Located above or near the signature, ensuring easy visibility for identification purposes.
- Special Field: If present, this field is usually in a less prominent location, such as the bottom left, containing additional or supplementary information.

Back Side

- Father's Name, Father's Surname, Mother's Name, Mother's Maiden Name: These familial details are typically listed at the top or upper-middle section, providing background personal information.
- Nationality, Sex: Often located near the familial details, summarizing the individual's demographic information.
- Height: This detail could be positioned near personal information, providing a physical descriptor of the cardholder.
- Date and Place of Issue: Usually found in the middle or lower section, detailing when and where the ID was issued.
- Machine Readable Zone (MRZ): Positioned at the very bottom, spanning the width of the ID, designed for electronic reading devices.

Security Features

- Biometric ID Symbol: Located on the front, often near the top or side, indicating the presence of biometric data.
- Holder's Photograph: Typically on the front side, often on the left, providing a visual identification feature.
- Cyprus Flag Iridescent: Adjacent to the holder's photograph, adding a layer of security through a unique visual effect.

- Cyprus Emblem Dove, Oval With Portrait: These holographic or optical security elements are usually found on the back side, often in the upper section, enhancing the ID's security against forgery.

4.6 Type of Data

Alphanumeric Data:

- CRN (Card Reference Number): Comprises a combination of letters ("CR") and numbers, serving as a unique identifier for each ID card.
- ID Card Number: A mix of numbers, often starting with zeroes, followed by a unique set of digits for each card.
- Machine Readable Zone (MRZ): Contains alphanumeric characters, including the "IDCYP" prefix, the CRN, and other encoded information essential for electronic reading and verification.

Textual Data:

- Name, Surname, Place of Birth, Father's Name, Father's Surname, Mother's Name, Mother's Maiden Name, Nationality: These fields consist of text, capturing personal information in uppercase letters. The data is bilingual, presented in both the Greek and English languages, accommodating the linguistic diversity of Cyprus.
- Type of Identification: This field includes text indicating the card's function as an identity document, standardized across all cards.

Numerical Data:

- Date of Birth, Valid Until, Date and Place of Issue: These sections display dates in the DD MM YYYY format, providing clear numerical data related to the cardholder's birth and the card's validity period.
- Height: Presented in centimeters, this numerical data specifies the cardholder's height at the time of issuance.

Biometric Data:

- Signature of Holder: Captured as a digital image, the holder's signature is stored as biometric data, allowing for personal verification.
- Holder’s Photograph: A grayscale image from the hair to the neck, serving as a key biometric identifier for the cardholder.

Graphical Data:

- Cypriot Emblem and Cyprus Flag: These symbols are graphical elements that represent the country's identity, with the flag sometimes featuring an iridescent effect to enhance security.
- Biometric ID Symbol: A graphic symbol indicating the card's compliance with biometric data standards.
- Special Field: Contains a combination of letters and numbers in a specific format, possibly serving as an additional security feature or identifier.

Optical and Holographic Data:

- Cyprus Emblem Dove, Oval With Portrait: These features incorporate optical and holographic elements to enhance the card's security, making duplication more difficult.

4.7 Summary

In this chapter, we delved into the intricacies of the updated Cypriot identity cards, highlighting significant enhancements aimed at bolstering security and improving machine readability. A noteworthy addition to these IDs is the Machine Readable Zone (MRZ), which utilizes an optical character recognition (OCR) code in a three-line format, encapsulating essential holder details like name, date of birth, and nationality, thereby facilitating swift identity verification by authorities.

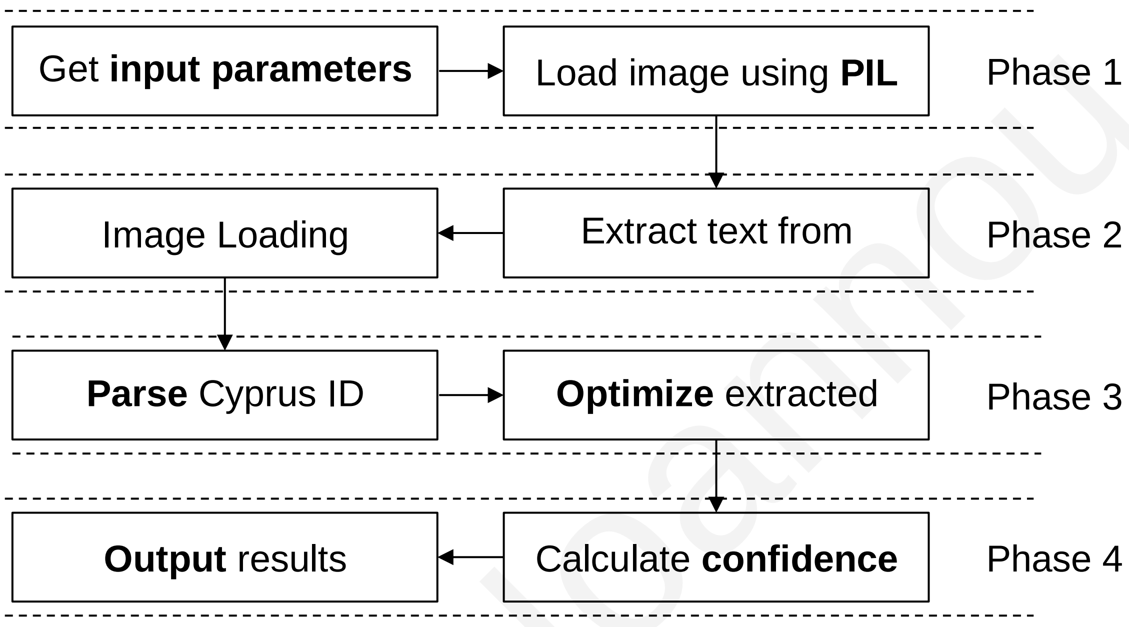
Comparative analysis of the IDs issued in 2020 versus those from 2015 reveals minor yet impactful modifications. These include a slight upward shift of the ID card number, enhanced readability through bolder and larger text, and a new rectangle symbolizing Cyprus's European Union membership, positioned below the Card Reference Number (CRN). The thesis underscores the multilingual presentation of ID text, with all fields defined in Greek, English, and Turkish, catering to Cyprus's linguistic diversity. This feature not only underscores Cyprus's commitment to inclusivity but also reflects the complexity involved in the development of a proficient OCR system capable of recognizing varied character encoding protocols across these languages. A comprehensive linguistic analysis is undertaken, dissecting every field of the new Cypriot ID to understand the encoding protocols and tailor algorithms for optimal OCR performance. The analysis spans across different data types - alphanumeric, textual, numeric, biometric, graphical, optical, and holographic data, each meticulously examined for their respective roles and security implications on the ID.

From a design and security perspective, the front side of the ID showcases the Cypriot emblem and flag, the name of the country in multiple languages, type of identification, and the biometric ID symbol, among others. The CRN, serving as a pivotal security element, is a unique 9-digit identifier enhancing fraud resistance. Personal details such as name, surname, and place of birth are presented bilingually in bold uppercase letters, while the holder's signature and photograph provide biometric validation. The back side continues with personal and familial details, also bilingually presented, and includes the cardholder's height, further personalizing the ID. Security features like the Cyprus Emblem Dove and the oval with the portrait exhibit optical and holographic enhancements against forgery. The MRZ, a critical element for machine readability, follows international standards for machine-readable travel documents. Its structured format includes encoded personal details, facilitating electronic scanning and verification processes. Throughout the analysis, the thesis articulates the importance of the fixed text and data-type fields across the IDs. It illustrates how these elements - from the CRN and ID card number to the holder's biometric data and the graphical symbols of Cypriot identity - play crucial roles in maintaining the integrity, security, and functionality of the identity cards.

In summary, the updated Cypriot identity cards represent a sophisticated blend of technological advancements and multilingual inclusivity, designed to meet contemporary security challenges while facilitating efficient identity verification processes. The detailed examination of the cards' features, from textual and biometric data to optical and holographic security measures, underscores the comprehensive approach taken to enhance their reliability and security.

Chapter 5: Cyprus ID OCR Tool Design

5.1 Algorithms



The tool operates in four distinct phases:

Phase 1 - User input and image loading: In the initial phase, the user uploads images they wish to analyze for data extraction. Upon receiving the images, the tool utilizes the PIL (Python Imaging Library), now known as Pillow, to import these images into the script. This library supports a myriad of image file formats, facilitating image manipulations such as opening, resizing, and processing, which are crucial before passing the images to the OCR tool for text recognition. This step ensures that images are appropriately prepped to optimize the accuracy and efficiency of the subsequent OCR phase.

Phase 2 - Text extraction and preparation: In this phase, the tool employs the Tesseract library to perform OCR and extract readable text from the images. The extracted text is then temporarily stored to allow for necessary modifications that enhance data accuracy. These adjustments help in correcting the text and ensuring it matches the correct information seamlessly. This preparatory step refines the text, setting the stage for further processing in the subsequent phase.

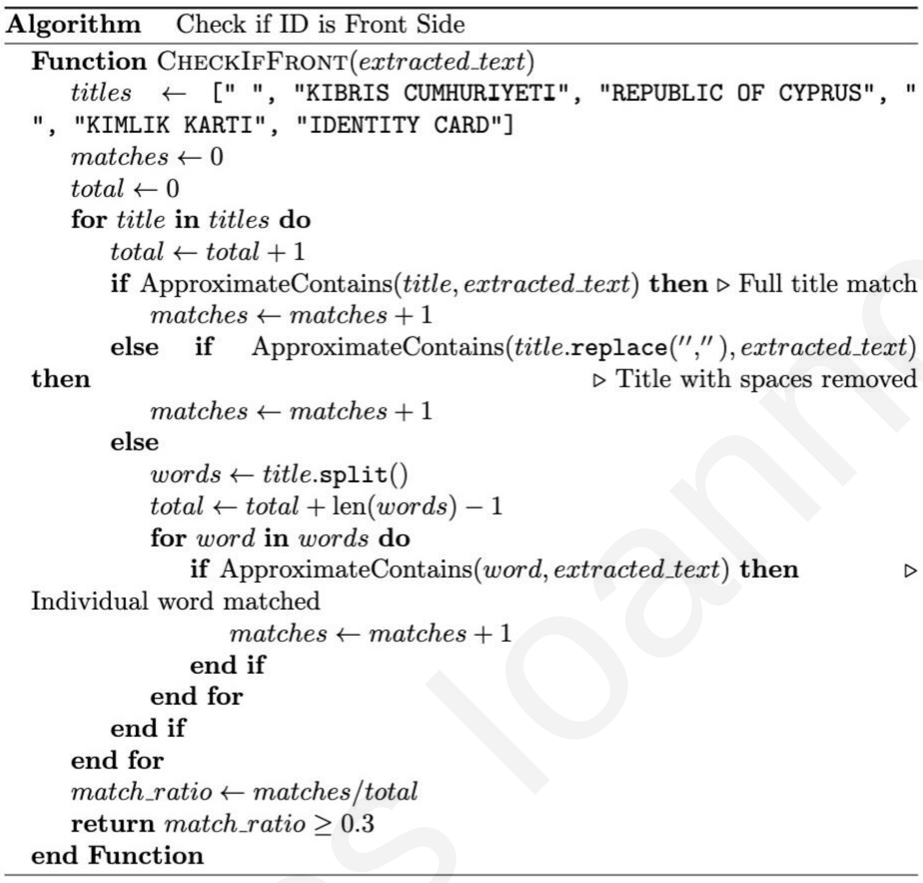
Phase 3 - Text Parsing and Optimizations: This stage involves precise identification and verification of personal details from Cyprus IDs, such as names, surnames, and places of birth, excluding dates and sex. The process utilizes two separate JSON files; the first contains first names in Greek, and the second encompasses surnames also in Greek. The tool employs these files to approximate and validate the names and surnames extracted in the previous phase, ensuring the data matches correctly. This verification is crucial for the accuracy and reliability of the information before it moves to the final processing step.

Phase 4 - Confidence Calculation and Output: In the final phase, the tool calculates the confidence level of the text matches established in the previous steps using advanced algorithms that assess the accuracy of name and surname identification from the processed IDs. This involves comparing the validated names and surnames against the entries in the JSON files to quantify how closely they match. Based on these calculations, the tool generates a confidence score for each field analyzed, which indicates the reliability of the information. The results, along with their confidence scores, are then compiled and presented in a structured output format, ready for application.

Algorithms

We utilize three distinct algorithms: the first extracts data from images provided by the user, the second determines whether the image depicts the front or back of a Cypriot ID, and the third identifies whether the ID follows a new or old design format. Although the process of extracting data from each specific field on the ID varies—since the OCR (Optical Character Recognition) searches for various indicators to identify and populate these fields—the fundamental method remains consistent across all data extraction algorithms. This consistency justifies why a single pseudocode can represent the operations of all the algorithms. For example, to identify the name field, the OCR looks for the keyword "name", and similarly, it searches for "surname" to find the surname field, "sex" for the sex field, etc., for all other fields. The pseudocode for these algorithms is provided below to illustrate the workings of the tool and the logic it uses to accurately locate the necessary keywords.

5.1.1 Detect Front or Rear Side

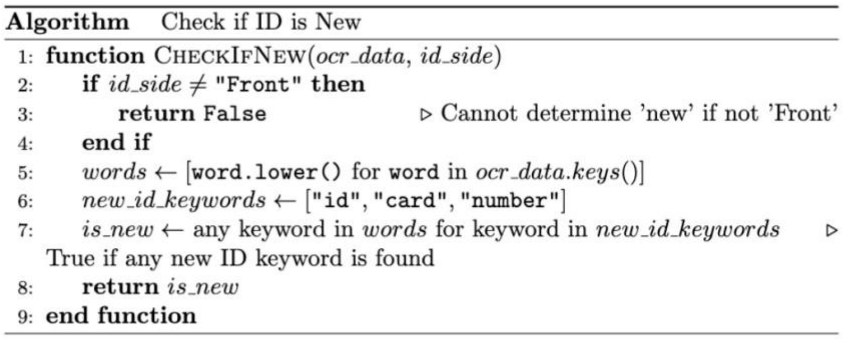


```
Algorithm  Check if ID is Front Side
  Function CheckIfFront($extracted\_text$)
    $titles \leftarrow$ [" ", "KIBRIS CUMHURIYETI", "REPUBLIC OF CYPRUS", "
  ", "KIMLIK KARTI", "IDENTITY CARD"]
    $matches \leftarrow 0$
    $total \leftarrow 0$
    for $title$ in $titles$ do
      $total \leftarrow total + 1$
      if ApproximateContains($title, extracted\_text$) then ▷ Full title match
        $matches \leftarrow matches + 1$
      else if ApproximateContains($title$.replace('',''), $extracted\_text$)
  then                                             ▷ Title with spaces removed
        $matches \leftarrow matches + 1$
      else
        $words \leftarrow title$.split()
        $total \leftarrow total + \text{len}(words) - 1$
        for $word$ in $words$ do
          if ApproximateContains($word, extracted\_text$) then             ▷
  Individual word matched
            $matches \leftarrow matches + 1$
          end if
        end for
      end if
    end for
    $match\_ratio \leftarrow matches/total$
    return $match\_ratio \geq 0.3$
  end Function
```

Image 3: The algorithm to identify the front and rear side of an ID.

If the algorithm determines the image fails this verification, it will then automatically classify it as the old ID.

5.1.2 Detect New or Old ID



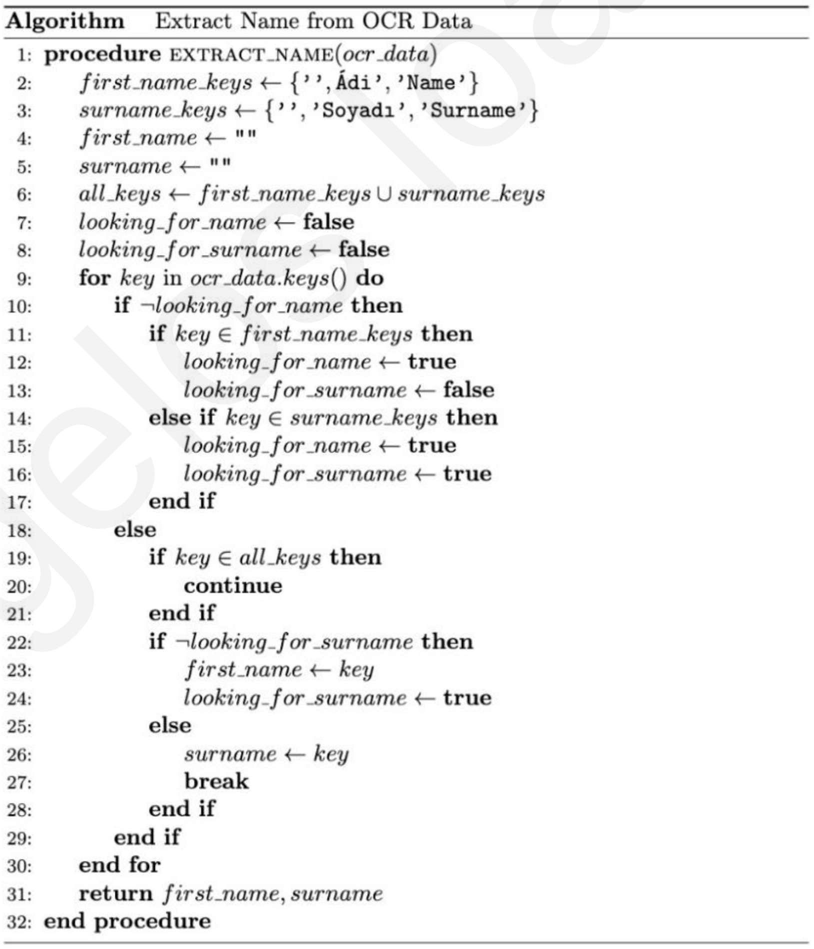
Algorithm Check if ID is New

```
function CHECKIFNEW($ocr\_data$, $id\_side$)
    if $id\_side \neq$ "Front" then
        return False                    ▷ Cannot determine 'new' if not 'Front'
    end if
    $words \leftarrow$ [word.lower() for word in $ocr\_data.keys()$]
    $new\_id\_keywords \leftarrow$ ["id", "card", "number"]
    $is\_new \leftarrow$ any keyword in $words$ for keyword in $new\_id\_keywords$    ▷ True if any new ID keyword is found
    return $is\_new$
end function
```

Image 4: The algorithm to detect if this is a new or old ID.

Similar to algorithm 1, if it identifies that it is not the new ID, it will automatically be categorized as the old ID.

5.1.3 Parse Information



Algorithm Extract Name from OCR Data

```
procedure EXTRACT_NAME($ocr\_data$)
    $first\_name\_keys \leftarrow$ {'', Ádi', 'Name'}
    $surname\_keys \leftarrow$ {'', 'Soyadı', 'Surname'}
    $first\_name \leftarrow$ ""
    $surname \leftarrow$ ""
    $all\_keys \leftarrow first\_name\_keys \cup surname\_keys$
    $looking\_for\_name \leftarrow$ false
    $looking\_for\_surname \leftarrow$ false
    for $key$ in $ocr\_data.keys()$ do
        if $\neg looking\_for\_name$ then
            if $key \in first\_name\_keys$ then
                $looking\_for\_name \leftarrow$ true
                $looking\_for\_surname \leftarrow$ false
            else if $key \in surname\_keys$ then
                $looking\_for\_name \leftarrow$ true
                $looking\_for\_surname \leftarrow$ true
            end if
        else
            if $key \in all\_keys$ then
                continue
            end if
            if $\neg looking\_for\_surname$ then
                $first\_name \leftarrow key$
                $looking\_for\_surname \leftarrow$ true
            else
                $surname \leftarrow key$
                break
            end if
        end if
    end for
    return $first\_name, surname$
end procedure
```

Image 5: The algorithm that parses the information in the OCR tool.

This algorithm is a procedure to extract a person's first name and surname from OCR data. It starts by defining keyword sets for identifying first names and surnames. Then, it iterates through each key in the OCR data: if it finds a key matching the first name keywords, it starts looking for the surname; if it finds a key matching the surname keywords, it captures the next key as the surname. The algorithm stops searching after finding the surname, returning both the first name and surname.

5.2 Tool Layout

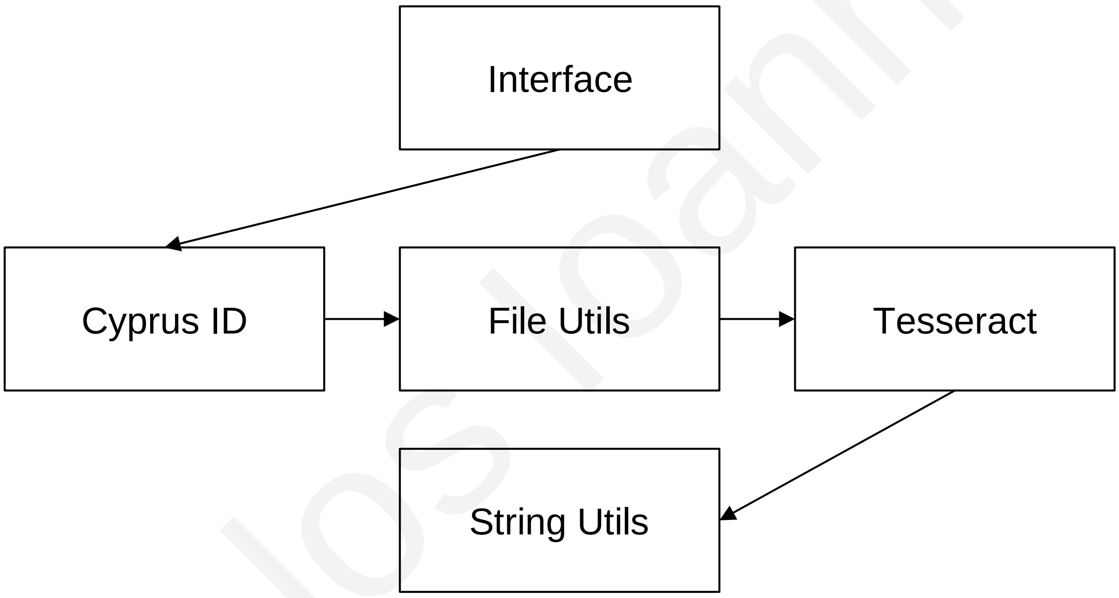


Image 6: the layout of how the codebase is connected for the python tool.

5.2.1 Cyprus ID

The `cyprus_id.py` file defines a set of classes and enumerations that provide a structured representation and functionality for handling Cyprus personal identification (ID) cards. It serves as a crucial component in the overall system architecture of the project, facilitating the processing and manipulation of ID card data.

Firstly, the file introduces two enumeration classes, `IdVersion` and `IdSide`. The `IdVersion` class represents the version of the Cyprus ID card, with two possible values: "Old" and "New". Similarly, the `IdSide` class encapsulates the notion of the card's side, offering three options:

"Front", "Rear", and "Both". These enumerations ensure consistency and type safety when working with ID card versions and sides throughout the codebase.

The `IdField` class is a fundamental building block in the file's design. It represents a single field within a Cyprus ID card, such as the first name, last name, date of birth, or any other relevant piece of information. Each `IdField` instance has three key attributes: `header` (the field's caption), `value` (the field's actual value), and `confidence_level` (a measure of confidence in the value's accuracy). Additionally, the `IdField` class provides a `merge` method, which allows for the merging of two `IdField` instances. This functionality is particularly useful when combining or updating ID card data from multiple sources, ensuring that the most accurate and up-to-date information is retained.

At the core of the `cyprus_id.py` file lies the `CyprusId` class, which represents a complete Cyprus personal ID card. This class encapsulates various attributes, each of which is an instance of the `IdField` class. These attributes cover a wide range of ID card fields, including the ID number, document number, first and last names, date of birth, nationality, gender, height, and more. The `CyprusId` class offers methods such as `set_version` and `set_side`, which allow for setting the ID card's version and side, respectively. Furthermore, the `merge` method enables the merging of two `CyprusId` instances by intelligently combining their respective `IdField` instances.

The `__repr__` method within the `CyprusId` class provides a formatted string representation of the ID card instance. This representation can be printed or used for other purposes, such as logging or displaying the ID card information in a human-readable format.

In the context of the project, the `cyprus_id.py` file plays a pivotal role in the overall system architecture by providing a structured and efficient way to represent and manipulate Cyprus ID card data. It serves as a foundational component, enabling various operations and functionalities related to ID card processing, validation, merging, and presentation.

5.2.2 Tesseract

The `ocr_utils.py` file contains several functions that assist in extracting information from images of Cyprus identification cards using Optical Character Recognition (OCR) techniques. The `preprocess_image` function preprocesses an image by converting it to grayscale and improving its contrast, which can enhance OCR performance. The `extract_data` function extracts text data from an image using Pytesseract OCR engine, supporting image preprocessing and specifying languages for better recognition.

The file includes functions like `check_if_front` and `check_if_rear` to determine whether the given OCR data corresponds to the front or rear side of an ID card by checking for specific keywords. Similarly, `check_if_new` and `check_if_old` attempt to identify whether the ID card is a new or old version based on the OCR data and the detected side. The `detect_version_and_side` function combines these functions to determine the version and side of the ID card based on the OCR data.

The `extract_name` function extracts the first name and surname from the OCR data by looking for specific identifiers and performing approximate matching with a list of Greek names. Several other functions like `extract_place_of_birth`, `extract_date_of_birth`, `extract_valid_until`, `extract_id_card_number`, `extract_doc_number`, `extract_father_first_name`, `extract_father_surname`, `extract_sex`, `extract_nationality`, `extract_date_and_place_of_issue`, and `extract_height` use regular expressions and other techniques to extract specific information from the OCR data, such as place of birth, date of birth, valid until date, ID card number, document number, father's name, sex, nationality, date and place of issue, and height.

The `parse_ocr_output_to_fields_dict` function consolidates the extracted information into a dictionary, where the keys represent different fields of the ID card, and the values are the corresponding extracted data. The `ocr_utils.py` file plays a crucial role in the overall system architecture by providing a set of utilities for extracting various pieces of information from Cyprus ID card images using OCR techniques, which can be integrated into a larger pipeline or system for processing and analyzing ID card data.

5.2.3 File Utils

The `file_utils.py` file contains two functions that assist in handling and processing image files within a given file or directory path.

The first function, `is_image(file_path)`, is responsible for determining whether a given file is an image or not. It takes a `file_path` argument, which represents the path to the file you want to check. The function utilizes the `filetype` library, which provides a convenient way to guess the MIME type of file based on its contents. By checking if the guessed MIME type starts with 'image', the function can reliably determine if the file is an image or not. This function returns a boolean value, `True` if the file is an image, and `False` otherwise.

The second function, `get_image_paths(input_path)`, is designed to retrieve a list of image file paths within a given file or directory path. It takes an `input_path` argument, which can be either a file path or a directory path. The function first checks if the provided `input_path` exists; if it doesn't, it returns an empty list. If the `input_path` is a file, the function checks if it's an image using the `is_image` function. If the file is an image, its path is added to the `image_paths` list, which is returned at the end of the function. If the `input_path` is a directory, the function lists all files within that directory using `os.listdir`. For each file, it constructs the full file path by joining the `input_path` and the file name using `os.path.join`. It then checks if the file is an image using the `is_image` function. If the file is an image, its path is added to the `image_paths` list. Finally, the function returns the `image_paths` list containing the paths of all image files found within the given `input_path`.

The `file_utils.py` file can be utilized to handle and process image data. For example, you might need to gather all image files from a specific directory or check if a particular file is an image before performing further operations on it.

5.2.4 String Utils

The provided code defines several functions and a custom enumeration class called `PrintStyle` in a Python module. This module serves as a utility for string manipulation, approximate string matching, and date validation.

Firstly, the `PrintStyle` enumeration class is defined, providing a way to associate constant values with named styles. It defines four different print styles: `INFO`, `WARNING`, `ERROR`, and `SUCCESS`. Enumerations improve code readability and maintainability by representing a set of related constant values.

The `print_string` function takes a string `s` and an optional `style` parameter, which defaults to `PrintStyle.INFO`. Depending on the specified `style`, the function prints the given string `s` with different color formatting using the `termcolor` library. If the `style` is `PrintStyle.INFO`, it prints the string in the default color. If the `style` is `PrintStyle.WARNING`, it prints the string in yellow. If the `style` is `PrintStyle.ERROR`, it prints the string in red. If the `style` is `PrintStyle.SUCCESS`, it prints the string in bold green.

The `match_ratio` function calculates the similarity ratio between two strings using the `fuzz.ratio` function from the `fuzzywuzzy` library. It takes two string arguments, `str1` and `str2`, and returns an integer value representing the similarity ratio between them, ranging from 0 to 100.

The `approximate_match` function checks if two strings have a similarity ratio greater than or equal to a specified threshold. It takes two string arguments, `str1` and `str2`, and an optional `ratio` parameter with a default value of 90. It returns `True` if the similarity ratio between the two strings is equal to or greater than the specified `ratio`, and `False` otherwise.

The `approximate_str_contains` function checks if a substring approximately matches another string based on the partial ratio. It takes two string arguments, `str1` (the substring) and `str2` (the string), and an optional `ratio` parameter with a default value of 90. It returns `True` if the partial similarity ratio between `str1` and `str2` is equal to or greater than the specified `ratio`, and `False` otherwise.

The `approximate_index` function finds the index of the first string in a list that approximately matches a given string. It takes three arguments: `str1` (the string to find), `lst` (the list of strings to search), and an optional `ratio` parameter with a default value of 90. It returns the index of the first matching string in the list if the similarity ratio is equal to or greater than the specified `ratio`; otherwise, it returns -1.

The `approximate_contains` function checks if a string approximately matches any string in a list. It takes two arguments: `str1` (the string to check) and `lst` (the list of strings to search), and an optional `ratio` parameter with a default value of 90. It returns `True` if the string approximately matches any string in the list based on the specified `ratio`, and `False` otherwise.
The `is_valid_date` function checks if a given string can be interpreted as a valid date. It takes two arguments: `date_str` (the string to check) and an optional `fuzzy` parameter with a default value of `True`. The function allows two date formats: "DD MM YYYY" and "DD/MM/YYYY". It performs various checks on the string, such as checking for alphabetic characters, the presence of '/' and ' ' characters, and attempts to parse the string using the `dateutil.parser.parse` function. If the string can be successfully parsed as a date, it returns `True`; otherwise, it returns `False`.
This module provides utilities for string manipulation, approximate string matching, and date validation. These functions can be useful in various scenarios, such as data cleaning, text processing, or user input validation.

Chapter 6: Implementation and Testing

6.1 Introduction

Throughout the project's duration, all scripts were executed using Python 3.9.7. Python 3.9, released in 2020 and supported until 2025, is a stable and efficient version of the programming language. Its robust performance and rigidity were well-suited for the project's requirements, including optical character recognition (OCR) and background processing tasks. However, as the project evolves and becomes more intricate, necessitating additional functionalities and dependencies, an upgrade to a more recent Python version might be warranted. While new versions typically introduce performance and security enhancements, it is crucial to note that when packaging the software for deployment on other devices, the specific Python version and all associated libraries required by the scripts will be included as dependencies.

6.1.1 Python Libraries

json: This library provides functionality for encoding and decoding JSON (JavaScript Object Notation) data, which is a lightweight data-interchange format. In the context of OCR for IDs, this library could be used for parsing or generating JSON data structures, potentially for storing or transmitting the extracted information from the IDs.

pytesseract: This is a Python wrapper for Google's Tesseract-OCR Engine, which is a powerful open-source OCR engine. pytesseract allows you to use Tesseract's OCR capabilities directly from Python scripts, making it an essential library for performing OCR on ID images.

PIL (Python Imaging Library): PIL is a popular library for opening, manipulating, and saving various image file formats. In the context of OCR for IDs, PIL can be used for preprocessing the input images, such as resizing, converting color modes, or applying filters, to enhance the quality and accuracy of the OCR process.

pytesseract.Output: This module from the pytesseract library provides additional output information beyond just the recognized text. It can return data like bounding boxes for individual characters or words, allowing for more advanced analysis and processing of the OCR results when dealing with IDs.

re: The re library provides support for regular expressions in Python. Regular expressions can be invaluable for post-processing the OCR output, such as validating or extracting specific patterns from the recognized text, which can be particularly useful when parsing structured information like ID numbers or expiration dates.

6.1.2 Tesseract

The version of Tesseract used was 0.3.10, which played a pivotal role in facilitating the OCR process. To carry out the character recognition, several settings needed to be configured. These settings were incorporated into a function named `extact_data`:

The extract_data function is designed to perform optical character recognition (OCR) on an image file and return the recognized text along with its corresponding confidence levels. It takes three parameters:

1. image_path: This is the file path to the image file that needs to be processed for OCR.

2. preprocessing: This is a boolean flag that determines whether image preprocessing should be applied before performing OCR. Image preprocessing can involve operations like resizing, converting color modes, or applying filters to enhance the quality and accuracy of the OCR process.

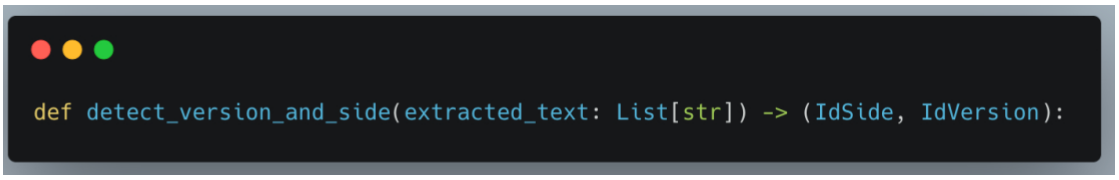
3. languages: This is a string representing the languages to be considered by the OCR engine when recognizing text in the image. For example, if you pass "eng+ell+tur", it will expect the image to contain text in English, Greek, and Turkish. The function starts by opening the image file using the Image.open method from the Python Imaging Library (PIL). This loads the image data into memory. If the preprocessing flag is set to True, the function calls a custom preprocess_image function and passes the loaded image to it. This custom function presumably performs the necessary preprocessing operations on the image, and the preprocessed image is then assigned back to the image variable. Next, the function uses the pytesseract library to perform OCR on the image (either the original or preprocessed image, depending on whether preprocessing was applied). The pytesseract.image_to_data function is called, passing the image object, the specified languages, and an output_type argument set to Output.DICT. This function performs the OCR process and returns a dictionary containing various information about the recognized text, such as the text itself, bounding boxes, confidence levels, etc. The function then creates an empty dictionary called ocr_text_conf, which will store the recognized text and its corresponding confidence level. It iterates over two lists from the OCR output dictionary: the list of confidence levels (ocr_data["conf"]) and the list of recognized text (ocr_data["text"]). The zip function is used to pair each confidence level with its corresponding text. For each pair of confidence level and text, the code checks if the text is not empty. If the text is not empty, it adds a key-value pair to the ocr_text_conf dictionary, where the key is the recognized text, and the value is the corresponding confidence level. Finally, the function returns the ocr_text_conf dictionary, which contains the recognized text from the image file and their associated confidence levels, excluding any empty text entries.

6.2 Testing

6.2.1 Python Function Annotations

Unlike statically typed languages like C++ or Java, Python is dynamically typed, allowing variables to change their type during runtime. Function annotations in Python provide a means to add metadata about the expected types of function parameters and return values. While Python itself does not enforce these annotations, they play a crucial role in enhancing code quality, readability, and maintainability. Annotations serve as self-documenting code, providing valuable insights into the expected types of function parameters and return values. By annotating functions with parameter types and return types, developers can better understand the intended usage of functions without needing to dive into the implementation details. This improves code readability and comprehension, especially in large codebases where understanding the behavior of functions may be challenging. Although Python is dynamically typed, recent developments such as the introduction of type hinting and static type checkers like MyPy allow developers to perform static type checking on their codebases. By annotating functions with type hints, developers can leverage static type checkers to catch type-related errors and inconsistencies early in the development process. This helps prevent common runtime errors and improves code robustness. Function annotations enable Integrated Development Environments (IDEs) to provide better support and autocompletion features. IDEs can use function annotations to offer context-aware suggestions and type hints, improving the developer experience and productivity. Additionally, IDEs can leverage annotations to perform type inference and provide more accurate code analysis and error detection.

Below, a code block of a function annotation of our code will be shown as well as how it works:



```
def detect_version_and_side(extracted_text: List[str]) -> (IdSide, IdVersion):
```

This function is annotated to indicate that it takes one parameter and returns a tuple consisting of two elements. The parameter extracted_text is expected to be a list of strings (List[str]), which would typically contain the text that has been extracted from an image or document using OCR (Optical Character Recognition) or a similar method. The return type is a tuple with two elements: IdSide, which is an enumeration that represents the side of the ID (e.g., front or back) and IdVersion, which is also an enumeration that indicates the version of the ID (e.g., old or new design).

6.2.2 Python Unit Testing

Unit Testing

Unit testing is a software testing method where individual units or components of a software are tested independently to ascertain if they operate as expected. A "unit" can be as small as a function or as large as a complex class in an object-oriented programming context. The primary goal of unit testing is to isolate each part of the program and show that the individual parts are correct in terms of requirements and functionality.

Why Unit Testing is Useful

The utility of unit testing lies in its ability to help developers catch errors early in the development cycle. By testing the smallest testable parts of an application, separately and independently, it becomes much easier to pinpoint where a problem may be occurring. This approach leads to several benefits: it simplifies the debugging process, enhances the quality of the code, encourages changes and simplification of code, and provides documentation of the system's units. Furthermore, unit tests ensure that code changes in the future do not break existing functionality, a concept known as regression testing. Another critical advantage of unit testing is that it facilitates Test-Driven Development (TDD). TDD is an advanced software development process where the developer writes a test before writing a code snippet to fulfill that test's requirement. The cycle of TDD involves writing a test, running it to see it fail (since the code isn't written yet), writing the code, running the tests again to see them pass, and then refactoring the code with confidence that it's still functioning correctly as the tests will confirm.

Unit Testing For OCR

Considering the necessity for rigorous testing in this project, it is imperative to employ unit testing for a more effective evaluation of the tool. The following outlines the advantages of utilizing unit testing in Python for the tool:

Accuracy and Reliability: OCR systems must accurately interpret and convert different types of images into machine-encoded text. By employing unit tests, the accuracy of the OCR algorithms can be verified on a variety of test images under controlled conditions. This helps to confirm that the system reliably extracts text data such as names, ID numbers, and dates from Cypriot IDs.

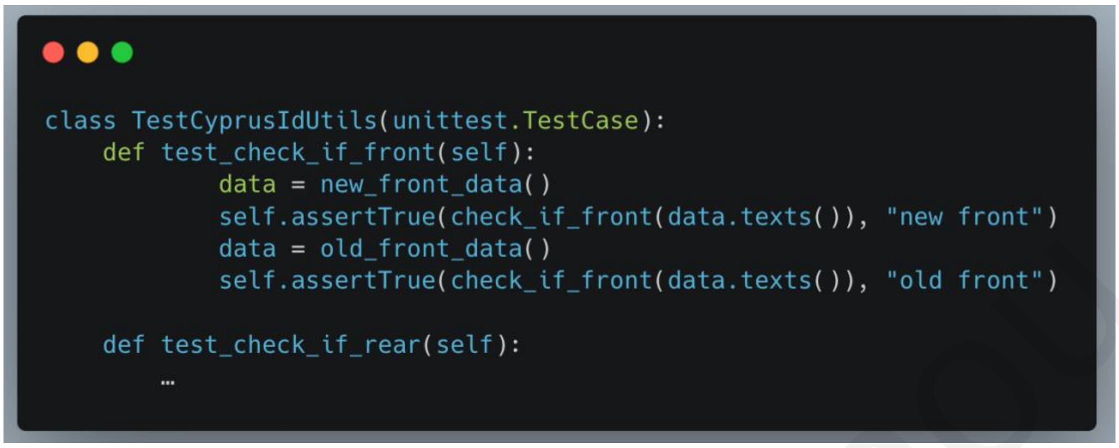
Algorithm Validation: With unit testing, each algorithmic component, such as those for detecting text regions, character recognition, and format validation, can be individually tested. For example, a test for the algorithm might be written that determines if an ID is old or new, ensuring it consistently identifies the correct version despite variations in image quality or layout.

Error Handling: Unit tests can be used to ensure that the OCR system gracefully handles common errors, such as blurred or skewed images, and can still operate effectively in less-than-ideal conditions. Testing for error conditions will help to improve the robustness of the OCR application.

Python Unit Testing

In Python, unit tests are usually expressed using a built-in module named unittest. This module comes with a wide variety of tools to construct and run tests, and it's part of Python's standard library. The unittest framework provides a TestCase class that can be extended to create test cases.

Below is an example of a unit test:


```
class TestCyprusIdUtils(unittest.TestCase):
    def test_check_if_front(self):
            data = new_front_data()
            self.assertTrue(check_if_front(data.texts()), "new front")
            data = old_front_data()
            self.assertTrue(check_if_front(data.texts()), "old front")

    def test_check_if_rear(self):
        ...
```

Method	**Checks that**	**Explanation**
assertEqual(a,b)	a == b	Asserts that the two values a and b are equal
assertNotEqual(a,b)	a != b	Asserts that the two values a and b are not equal
assertTrue(x)	bool(x) is True	Asserts that the expression x evaluates to True
assertFalse(x)	bool(x) is False	Asserts that the expression x evaluates to False

Table 2: Different methods that are used in unit testing.

In the code block above, the **test_check_if_front** method is designed to validate the check_if_front function. The assertTrue method is used twice to assert that check_if_front returns True for both new and old ID data, which would indicate that the data indeed corresponds to the front side of the IDs. The string messages "new front" and "old front" are optional messages that will be displayed if the assert fails, helping to identify which case failed during testing.

So, when test_check_if_front is executed:

- It first calls new_front_data() to simulate the OCR data extraction from a new version of a Cyprus ID's front side.

- Then, it passes this data to the check_if_front function and uses assertTrue to assert that the result should be True. If check_if_front doesn't return True, the test fails, and "new front" is printed as the failure message.
- Next, it does the same for old_front_data(), simulating an older version of the ID.

The use of assertTrue in this context is based on the expectation that check_if_front(data.texts()) should evaluate to True if the data provided is indeed from the front of a Cyprus ID. If the function is working correctly, the unit test will pass without any error messages. If it's not, the unit test will fail, and the associated message will help in quickly understanding what went wrong.

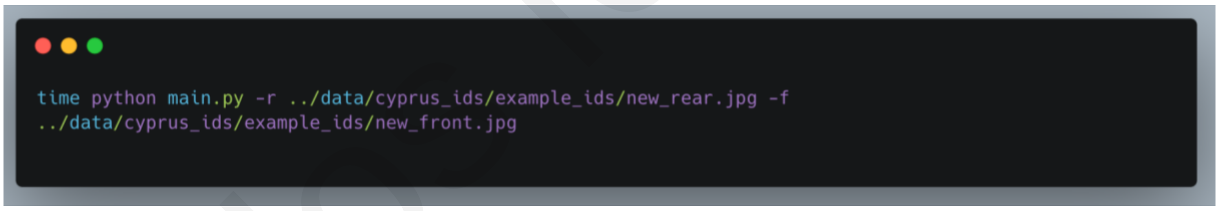
Chapter 7: Evaluation

7.1 Evaluation Setup

For the evaluation of the tool, a 2021 MacBook Air is utilized for operating the necessary actions. This system includes 8GB of unified memory and 512GB of SSD storage. It operates on an M1 ARM processor, marking Apple's initial transition from traditional Intel CPUs to its own ARM-based chips. This shift in processor architecture necessitates the use of an ARM-compatible version of Visual Studio, specifically version 1.87.2, to accommodate the differences from x86 architectures.

7.2 Execution Time

To measure the performance of the system that it is being tested at and the efficiency of the tool, some performance tests have to be run. Using a command of:



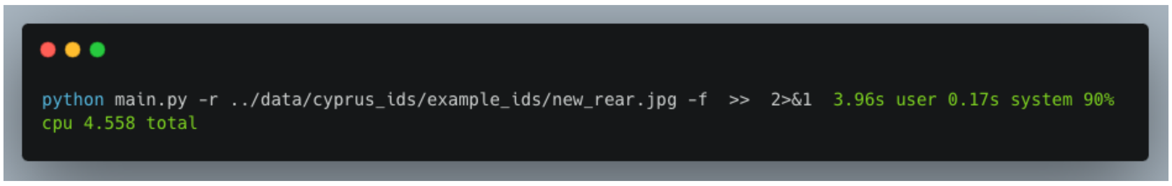
```
time python main.py -r ../data/cyprus_ids/example_ids/new_rear.jpg -f
../data/cyprus_ids/example_ids/new_front.jpg
```

In the code block above, -r is used to indicate that the image is the rear side of the ID and -f for the front side of the ID.

This command will calculate these parameters:

- User Time (in seconds)
- System Time (in seconds)
- CPU Usage (%)
- Total Time (in seconds)

With that command, we will get this response:


```
python main.py -r ../data/cyprus_ids/example_ids/new_rear.jpg -f  >>  2>&1  3.96s user 0.17s system 90%
cpu 4.558 total
```

From that response, we conclude that:

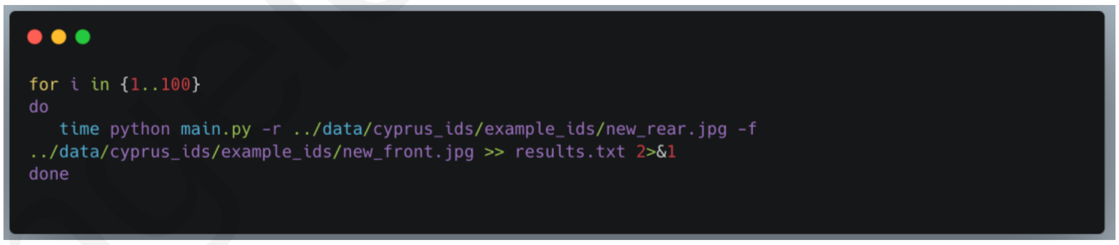
3.96s User: This time shows that the script was actively processing the commands for 3.96 seconds.

0.17s System: This time was spent by the system to support the script, like reading the necessary files.

90% CPU: This indicates that the CPU was actively used 90% of the time while the script was running, showing good CPU engagement.

4.558 Total: This is the overall time from start to finish for the script’s run, including both the active processing and any waiting time.

To effectively monitor system performance, it's necessary to execute the tool repeatedly to conduct thorough testing. Utilizing a Bash loop to run the script 100 times allows for subsequent data extraction and analysis. This data can then be evaluated by calculating the average, maximum, and minimum values.


```
for i in {1..100}
do
   time python main.py -r ../data/cyprus_ids/example_ids/new_rear.jpg -f
../data/cyprus_ids/example_ids/new_front.jpg >> results.txt 2>&1
done
```

Running it 100 times, the following results arise:

Category	**Average**	**Maximum**	**Minimum**
User Time (s)	3.95s	3.97s	3.92s
System Time (s)	0.128s	0.17s	0.12s

CPU Usage (%)	97.4%	99%	90%
Total Time (s)	4.163s	4.558s	4.074s

Table 3: Running the script 100 times to benchmark the performance in user time, system time, CPU usage and total time.

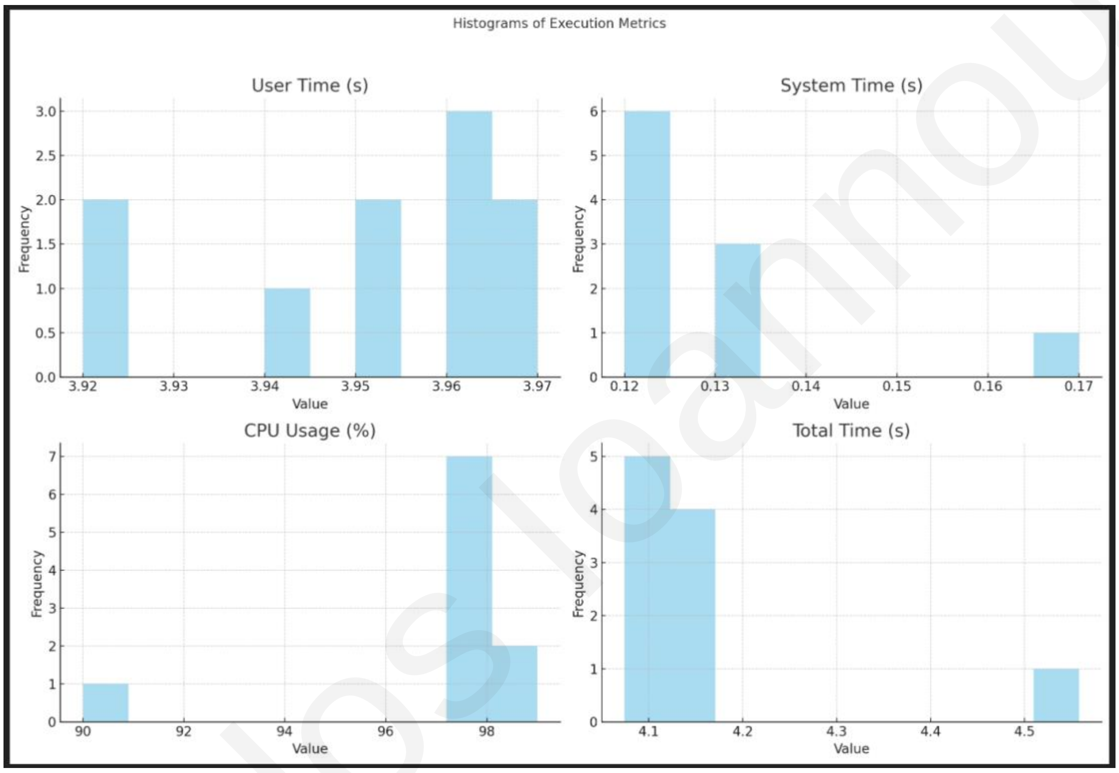


Figure 1:Visual representation of the user time, system time, CPU usage and total time in bar charts.

Running a single side of an ID, 100 times, provides these results:

Category	**Average**	**Maximum**	**Minimum**
User Time (s)	0.825	0.860	0.810
System Time (s)	0.056	0.100	0.050
CPU Usage (%)	93.87	96.00	24.00

Total Time (s)	0.977	3.883	0.897

Table 4: Running the script 100 times to benchmark the performance for a single image in user time, system time, CPU usage and total time.

Comparing the two tables, the table of a single image and the table that is using both side of the ID:

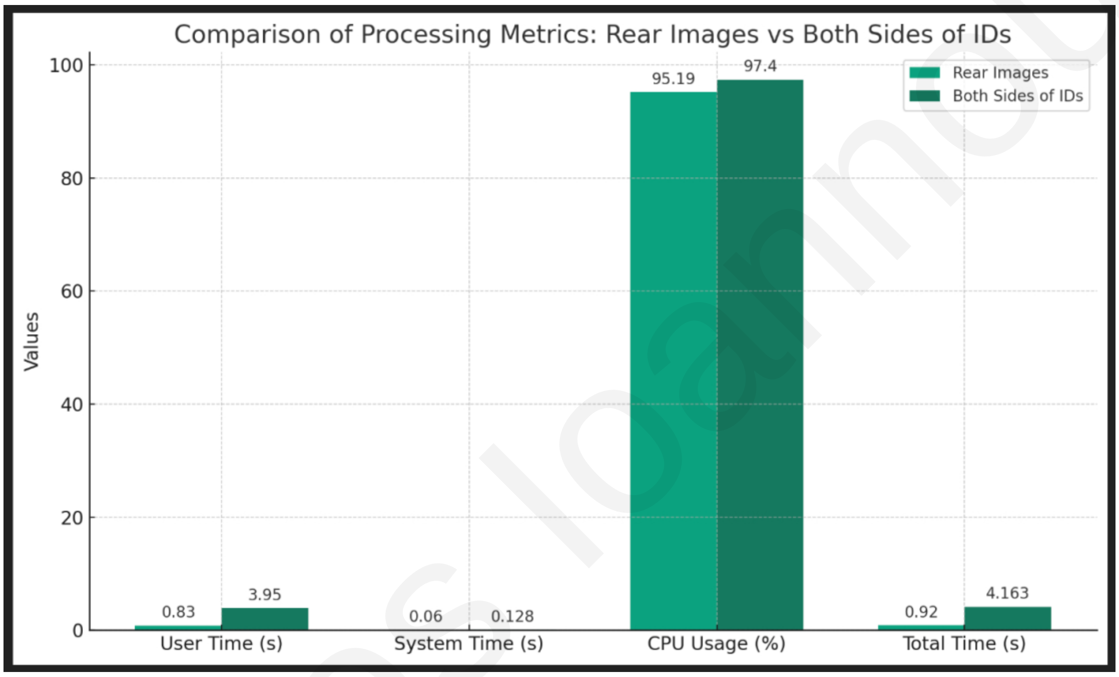


Figure 2:The comparison of processing metrics between rear images and both sides of IDs

By comparing the two tables for processing metrics of rear images of IDs and both sides of IDs, we can draw several insights:

1) **Increased Processing Time:** Processing both sides of IDs generally takes significantly longer in terms of user time and total time compared to processing just the rear images.
2) **Higher System Time:** The system time, which could involve operating system tasks necessary to support the application, is also higher when processing both sides of IDs.
3) **CPU Utilization:** CPU usage is consistently higher when processing both sides of IDs. This heightened CPU usage points to increased computational demand, which is present due to additional processing steps like more intensive image analysis and data extraction processes.

4) **Variability:** The range between the maximum and minimum values in each category (user time, system time, total time) is wider when processing both sides of IDs. This increased variability suggests that the complexity of processing varies more significantly across different ID scans when both sides are involved, due to variations in image content and quality.

7.3 Confidence Level

The confidence level is a metric used to determine the accuracy of Tesseract OCR in identifying characters during scans. A higher confidence level implies a stronger probability that the character or word scanned is correctly identified. Lower confidence levels may indicate visual issues hindering clear character recognition, or that the character is absent from the training database. However, occurrences like these are infrequent since Tesseract OCR effectively handles Greek, English, and Turkish characters.

To determine the confidence level for each field, a series of tests are conducted:

- Average confidence level per field for both front and back of IDs
- Overall average confidence level for the entirety of both sides of IDs
- Assessments of several IDs are conducted, ensuring the privacy and anonymity of the ID providers, and the findings are outlined below:

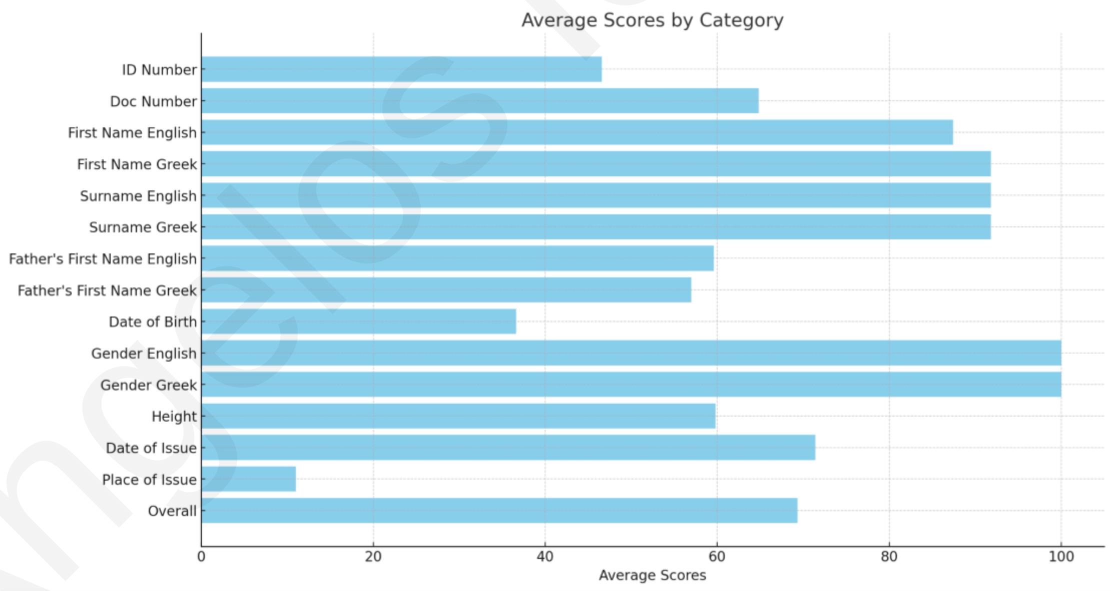


Figure 3: The comparison of the average scores for each ID field in confidence score

It is evident that some fields consistently show very high confidence levels, whereas others are notably low. A comparison will be made between the fields with the highest and lowest scores (Gender versus Place of Issue).

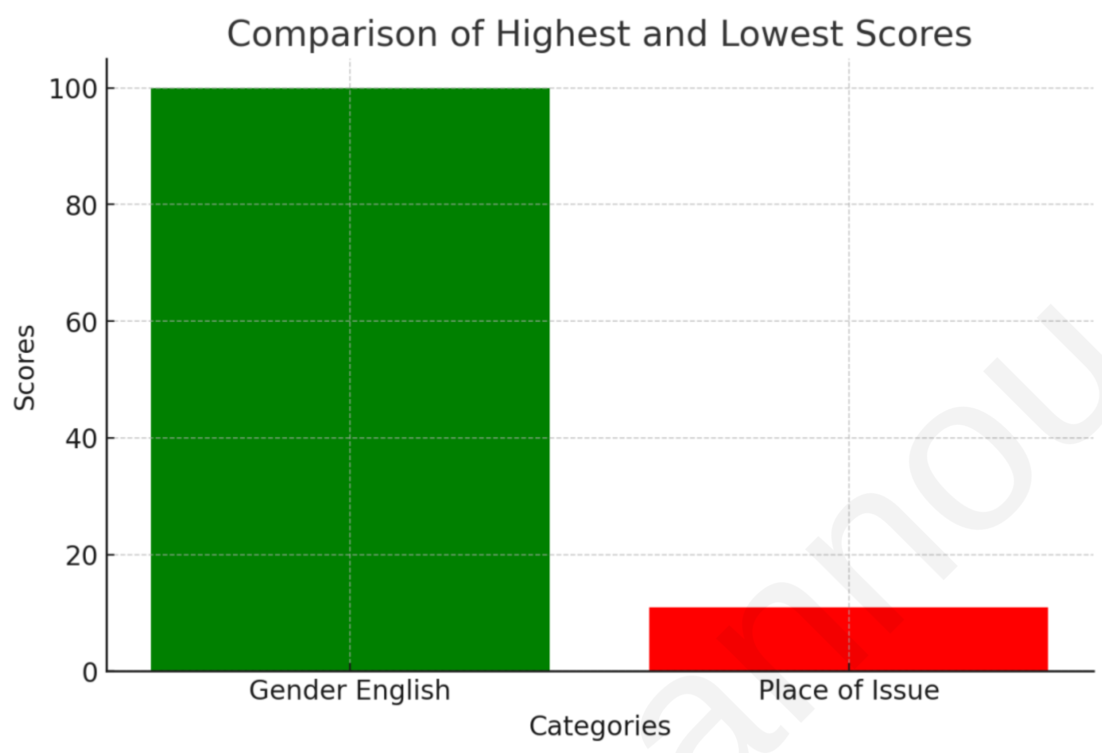


Figure 4: The comparison between the gender field and the place of issue field

Further analysis and tests on the scanned IDs reveal that the 'Place of Issue' field fails to be recognized correctly in the provided IDs. Conversely, the 'Gender' field achieved perfect scores in all assessments. This discrepancy indicates that enhancements are crucial for the 'Place of Issue' field to boost the tool's accuracy and reliability.

This need for improvement is underscored by a subsequent comparison of the three best and three worst-performing fields:

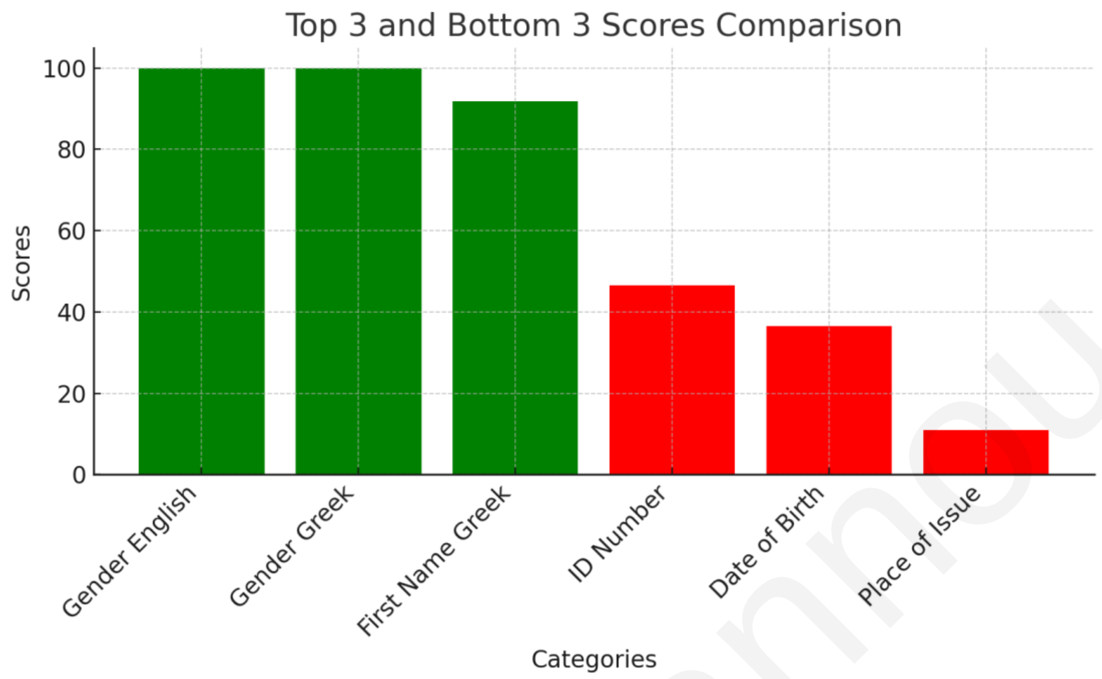


Figure 5: The confidence level of the 3 highest performing fields and the 3 lowest performing fields

The analysis of these fields reveals significant inconsistencies. The 'Date of Birth' and 'Place of Issue' fields, typically found on the back of an ID, and the 'ID Number' on the front, demonstrate the necessity for further research and development of more refined and precise algorithms to address these issues in future updates.

7.4 Summary

In summary, we evaluate the performance of Tesseract OCR in recognizing various fields across ID documents. Tesseract OCR, known for its proficiency in processing Greek, English, and Turkish characters, employs a confidence level metric to gauge its accuracy in character recognition during scans. A high confidence level signifies accurate character identification, while a low score may indicate potential visual obstacles or missing characters from the OCR's training database. However, such challenges are generally rare due to Tesseract's effective processing capabilities. To assess the OCR's accuracy comprehensively, multiple tests were conducted on both the front and back of ID documents. These evaluations included measuring the average confidence level per field and the overall average for all fields combined. Despite maintaining strict confidentiality and anonymity in handling ID samples, our analysis led to several insights.

Our findings reveal a distinct discrepancy in confidence levels among the fields analyzed. Notably, the 'Gender' field consistently achieved perfect scores, highlighting its reliable recognition. In stark contrast, the 'Place of Issue' field significantly underperformed, indicating a failure in its accurate detection across the sampled IDs. This gap underscores a critical need for targeted improvements in the OCR's algorithm, particularly in fields that are currently poorly recognized. A further comparison of the top three and bottom three performing fields emphasized the existing flaws. The 'Date of Birth' and 'Place of Issue', typically located on the rear, and the 'ID Number' on the front, all require additional scrutiny. These results advocate for the development of more sophisticated and accurate algorithms to enhance recognition accuracy, ensuring Tesseract OCR can reliably interpret key document information in diverse contexts. This summary serves as a pivotal reference for ongoing enhancements and underscores the necessity for continuous development in OCR technology.

Chapter 8: Conclusions & Future Work

8.1 Difficulties Faced

Throughout the project, numerous challenges were encountered. A primary and significant challenge was devising a logic for the OCR operation, including determining the relevant keywords and their locations. Crafting detailed, step-by-step instructions to replicate the human ability to effortlessly discern whether an ID is old or new, and its orientation, proved to be time-consuming. Similarly, the task of developing an OCR tool capable of accurately capturing and allocating data into appropriate fields presented difficulties. This is because the tool must precisely identify each field's location, as well as its font, format, and presentation style, to ensure the correct logic is applied for accurately reading and assigning data.

Furthermore, beyond the technical intricacies of programming the OCR to recognize and process various elements, there's the added complexity of ensuring it adapts to inconsistencies in ID designs and layouts. Variations in lighting, angles of capture, and physical condition of the IDs further complicate the tool's ability to accurately interpret data. Achieving a level of robustness where the tool can reliably operate under diverse and less-than-ideal conditions requires extensive testing and iterative refinement. This iterative process is critical for enhancing the tool's precision and reliability, making it adept at handling real-world variability in ID documents. Additionally, the effort to maintain user privacy and data security while processing sensitive information introduces another layer of complexity, underscoring the importance of implementing stringent data protection measures.

Moreover, the continuous evolution of ID formats presents an ongoing challenge. As governments update ID designs to incorporate advanced security features, the OCR tool must be regularly updated to remain effective. This necessitates a flexible and scalable architecture capable of accommodating new patterns, features, and technologies without extensive overhauls. Engaging in constant research and development is essential to stay ahead of these changes, requiring a commitment to learning and adaptation that extends beyond the initial deployment of the tool.

In summary, developing an OCR tool for ID recognition is a multifaceted endeavor that extends beyond the technical challenges of accurate data extraction. It involves a holistic approach that considers the tool's adaptability to new ID formats, its usability by a broad audience, and the imperative of safeguarding personal data. These considerations are critical for creating a tool that is not only technically proficient but also practical and secure for everyday use.

8.2 Knowledge Acquired

Embarking on this project not only provided an opportunity to tackle a series of complex problems but also served as a rich learning experience that extended across various domains of software development, data processing, and project management. One of the most significant areas of knowledge acquisition was in the realm of programming, particularly in Python, facilitated by the use of Visual Studio Code as the primary development environment. The project necessitated a deep dive into Python's ecosystem, leveraging its powerful libraries and frameworks to work with a sophisticated Optical Character Recognition (OCR) tool. Among these, the Tesseract Python library stood out as a critical component, offering extensive functionalities for image processing and text extraction. The exploration of Tesseract, alongside a host of other Python libraries, significantly enhanced my understanding of Python’s capabilities, particularly in handling complex data processing tasks.

The project's demands led to a substantial improvement in my ability to develop and implement logic. Crafting algorithms that could accurately interpret and classify information from varied ID documents required not only technical knowledge but also a nuanced understanding of the problem domain. This experience sharpened my analytical skills, enabling me to construct detailed, step-by-step processes that the OCR tool could reliably execute. It was a lesson in translating human cognitive processes—such as identifying an ID’s orientation or determining its age—into programmable logic that a computer could follow, a task that proved both challenging and immensely rewarding.

Learning to use Git and GitHub was another pivotal aspect of the project. These tools are indispensable for modern software development, offering robust solutions for version control and collaboration. Through Git, I mastered the art of managing code changes, ensuring that every modification, whether a new feature or a bug fix, was tracked and documented. GitHub served as the project's repository, a central hub where the fruits of my labor were stored, shared, and safeguarded. This experience was not just about learning the commands and operations necessary to commit and push code; it was an education in the best practices of collaborative development, understanding how to structure a project in a way that facilitates teamwork and transparency.

The project was also a profound lesson in problem-solving and innovation. From the outset, it was clear that developing an effective OCR tool would involve navigating a maze of unforeseen challenges. These ranged from technical hurdles, like adapting the tool to recognize and process text across different ID formats, to more abstract problems. Each obstacle demanded a solution, often requiring creative approaches that were not apparent at the beginning. This process of identifying issues, hypothesizing solutions, and iterating based on results became a cycle of learning in itself. It taught me the value of resilience and adaptability, qualities that are crucial for any software developer facing the ever-evolving landscape of technology.

Furthermore, the project was an education in the importance of thorough testing and quality assurance. Ensuring that the OCR tool performed reliably under a variety of conditions meant adopting a rigorous testing framework, simulating as many different scenarios as possible to uncover and correct any deficiencies. This aspect of the project highlighted the importance of attention to detail and the need for a methodical approach to software development, where every potential user interaction is considered, and every outcome is meticulously evaluated.

In addition to the technical skills and knowledge gained, the project was a lesson in project management and planning. Balancing the various components of the work—coding, testing, debugging, and documentation—required careful organization and time management. It emphasized the importance of setting realistic goals and milestones, prioritizing tasks, and adjusting plans as the project evolved. This experience was invaluable in understanding how to manage complex projects efficiently, ensuring that progress is steady and aligned with the project's objectives.

Lastly, the project underscored the significance of continuous learning and curiosity. The field of software development is dynamic, with new tools, languages, and methodologies emerging regularly. Keeping abreast of these developments and being willing to explore and adopt new techniques is essential for staying relevant and effective as a developer. This project was a testament to the power of learning, demonstrating how tackling real-world problems can lead to the acquisition of a wealth of knowledge and skills.

In conclusion, this project was more than just an exercise in developing a functional OCR tool; it was a comprehensive learning journey that touched on various aspects of software development and project management. From enhancing my proficiency in Python and familiarizing myself with essential tools like Git and GitHub to refining my problem-solving skills and understanding the intricacies of project planning, the knowledge and experience gained from this endeavor are invaluable. They have laid a solid foundation for my future projects, equipping me with the skills, insights, and confidence to tackle even more challenging problems in the field of software development.

8.3 Future Work

The journey through the development of the Optical Character Recognition (OCR) tool has laid a solid foundation for what has been a markedly rewarding endeavor. However, the nature of technological evolution and the ever-expanding scope of user requirements compel us to look forward, envisioning enhancements that not only refine the tool’s capabilities but also broaden its applicability and accessibility. In this "Future Works" section, we delineate a roadmap for the advancement of our OCR tool, focusing on several pivotal areas aimed at maximizing its efficacy and user experience.

8.3.1 Sophisticated Logic for Enhanced Accuracy

One of the primary objectives on our horizon is to develop a more sophisticated logic for our algorithms. The motivation behind this enhancement is to ensure that, regardless of the lighting conditions under which images are provided by the user, our OCR tool will still deliver accurate results. Achieving this necessitates a multifaceted approach, including but not limited to, advanced image preprocessing techniques to normalize lighting conditions, contrast adjustment algorithms, and perhaps incorporating machine learning models trained to recognize text across diverse illumination scenarios. This endeavor will involve extensive research and development, experimenting with different algorithms to find the optimal balance between preprocessing overhead and accuracy improvement.

8.3.2 Performance Optimization for Speed

In an era where speed is of the essence, making our OCR tool run faster is another critical area of focus. This could involve exploring more efficient methodologies for text recognition, such as optimizing our current algorithms for parallel processing, investigating the use of more powerful OCR libraries, or leveraging hardware acceleration where possible. Additionally, refining our codebase to eliminate bottlenecks and streamline data processing workflows will be key. Performance profiling tools will play a crucial role in this process, helping us identify and address areas where improvements can be made to enhance the tool's operational speed without compromising the quality of its output.

8.3.3 Comprehensive Testing for Uncharted Parameters

The diversity of human names and surnames presents a unique challenge for any OCR tool, especially when it encounters unusual or less common names. Future work will include a rigorous testing and evaluation phase to ensure our tool's algorithms are robust enough to handle a wide array of names and surnames from different cultures and languages. This will likely involve expanding our dataset to include a more diverse set of names, enhancing our tool's ability to accurately recognize and process text that falls outside the parameters of what has been previously encountered. Collaboration with linguists and cultural experts may also provide valuable insights into improving our tool's name recognition capabilities.

8.3.4 User-Friendly Interface and Web Application Development

To make our OCR tool more accessible and user-friendly, we envisage the development of a web application that can offer OCR capabilities in real-time. This involves designing an intuitive user interface that simplifies the process of uploading and processing images for OCR. The interface should be clean, minimalistic, and guide the user through each step of the process, providing helpful feedback along the way. Additionally, real-time OCR processing capabilities would significantly enhance the user experience, offering instant results and the ability to adjust or correct any inaccuracies immediately. This real-time functionality could be particularly beneficial in scenarios where quick turnaround times are critical, such as processing identity documents or forms in a business context.

8.3.5 Embracing the Future with Adaptability and Innovation

As we chart the course for future enhancements to our OCR tool, the underlying principles guiding our efforts will be adaptability and innovation. The landscape of digital technology is perpetually in flux, with new challenges and opportunities emerging at a rapid pace. Staying ahead requires not only a commitment to continuous improvement, but also a willingness to rethink and reinvent our approaches in response to new developments. The journey thus provided valuable insights and a solid foundation upon which to build. However, the path ahead is filled with possibilities that extend far beyond the current capabilities of our tool. By focusing on enhancing accuracy under varying conditions, optimizing performance, broadening our testing parameters to encompass a wider range of text scenarios, and developing a more user-friendly and accessible platform, we are poised to not only meet the evolving needs of our users but also redefine the standards of excellence in OCR technology.

In conclusion, the future of our OCR tool is one of ambitious expansion and refinement. The roadmap outlined in this section underscores our commitment to pushing the boundaries of what is possible, driven by a relentless pursuit of accuracy, efficiency, and user-centric design. Through diligent research, innovative engineering, and a deep understanding of the users' needs, a new advanced OCR tool that not only meets but exceeds the expectations of those who rely on it, could be developed for now, and in the years to come.

Bibliography

[1] C. S. Smith, “What Is OCR (Optical Character Recognition) Technology?,” Forbes. Accessed: May 12, 2024. [Online]. Available: https://www.forbes.com/sites/technology/article/what-is-ocr-technology/

[2] C. J. Bashe, W. Buchholz, and N. Rochester, “The IBM Type 702, An Electronic Data Processing Machine for Business,” *J. ACM*, vol. 1, no. 4, pp. 149–169, Oct. 1954, doi: 10.1145/320783.320784.

[3] A. Vachon, L. Ordonez, and J. R. Fonseca Cacho, “Global Postal Automation,” in *Intelligent Systems and Applications*, K. Arai, Ed., Cham: Springer International Publishing, 2022, pp. 135–154. doi: 10.1007/978-3-030-82199-9_10.

[4] R. Smith, “An Overview of the Tesseract OCR Engine,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Sep. 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[5] Ø. Due Trier, A. K. Jain, and T. Taxt, “Feature extraction methods for character recognition-A survey,” *Pattern Recognit.*, vol. 29, no. 4, pp. 641–662, Apr. 1996, doi: 10.1016/0031-3203(95)00118-2.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[7] S. Rice, J. Kanai, and T. Nartker, “An Evaluation of OCR Accuracy,” Jan. 1993.

[8] S. Mori, C. Y. Suen, and K. Yamamoto, “Historical review of OCR research and development,” *Proc. IEEE*, vol. 80, no. 7, pp. 1029–1058, Jul. 1992, doi: 10.1109/5.156468.

[9] H.-Y. Ma, H.-H. Huang, and C.-L. Liu, “Reading between the Lines: Image-Based Order Detection in OCR for Chinese Historical Documents,” *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 21, Art. no. 21, Mar. 2024, doi: 10.1609/aaai.v38i21.30572.

[10] I. J. Goodfellow *et al.*, “Generative Adversarial Networks.” arXiv, Jun. 10, 2014. doi: 10.48550/arXiv.1406.2661.

[11] “Abby OCR Reader,” ABBYY. Accessed: May 12, 2024. [Online]. Available: https://www.abbyy.com/ocr-sdk/

[12] “OCRopus,” GitHub. Accessed: May 12, 2024. [Online]. Available: https://github.com/ocropus

[13] M. A. M. Salehudin *et al.*, “Analysis of Optical Character Recognition using EasyOCR under Image Degradation,” *J. Phys. Conf. Ser.*, vol. 2641, no. 1, p. 012001, Nov. 2023, doi: 10.1088/1742-6596/2641/1/012001.

[14] “[PDF] Designing a Real-Time-Based Optical Character Recognition to Detect ID Cards | Semantic Scholar.” Accessed: May 12, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Designing-a-Real-Time-Based-Optical-Character-to-ID-Iskandar-Kesuma/4cb5fc12437cd1c271ef183804c4b575e8990716

[15] W. Satyawan *et al.*, “Citizen Id Card Detection using Image Processing and Optical Character Recognition,” *J. Phys. Conf. Ser.*, vol. 1235, no. 1, p. 012049, Jun. 2019, doi: 10.1088/1742-6596/1235/1/012049.

[16] Dr. Y. Perwej, S. A. Hannan, A. Asif, and A. Mane, “An Overview and Applications of Optical Character Recognition,” *Int. J. Adv. Res. Sci. Eng. IJARSE*, vol. Vol. 3, p. Pages 261-274, Jun. 2014.

[17] C. Patel, A. Patel, and D. Patel, “Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study,” *Int. J. Comput. Appl.*, vol. 55, pp. 50–56, Oct. 2012, doi: 10.5120/8794-2784.

[18] C. Isaza, J. Vargas, C. Gaviria, and L. Hernández, “Automatic OCR system for colombian DNIs,” in *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, Sep. 2012, pp. 295–300. doi: 10.1109/STSIVA.2012.6340598.

[19] W. Bieniecki, S. Grabowski, and W. Rozenberg, “Image Preprocessing for Improving OCR Accuracy,” Jun. 2007, pp. 75–80. doi: 10.1109/MEMSTECH.2007.4283429.

[20] J. Memon, M. Sami, R. A. Khan, and M. Uddin, “Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR),” *IEEE Access*, vol. 8, pp. 142642–142668, 2020, doi: 10.1109/ACCESS.2020.3012542.