

BABEȘ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

INTELLIGENT DOCUMENT PROCESSING FOR ROMANIAN MEDICAL CERTIFICATES

Team members

Lupu Eduard-Adrian, IS, 258-2, eduard.lupu@stud.ubbcluj.ro
Malancioiu Daniel-George, IS, 258-2, daniel.malancioiu@stud.ubbcluj.ro
Nichifor Dragos, IS, 258-2, dragos.nichifor@stud.ubbcluj.ro

Abstract

This project presents an intelligent system for automatic data extraction from Romanian medical leave certificates (*Certificat de Concediu Medical*). The motivation arises from the need to reduce manual transcription errors and accelerate administrative processing in healthcare and HR workflows.

The proposed solution integrates classical geometric computer vision techniques (**OpenCV** [1]) with a high-performance deep-learning OCR engine (**PaddleOCR**), controlled by a region-of-interest (**ROI**) mapping approach. Unlike traditional template matching, our system employs a "Candidate Voting" mechanism: for each field, it generates multiple image variants (grayscale, blue-ink enhanced, denoised) and selects the extraction result with the highest confidence score.

The entire pipeline is deployed as an interactive **Streamlit** web application, offering instant visual feedback, side-by-side alignment previews, and Excel export of structured results. Experimental validation on a dataset of anonymized certificates demonstrates robust extraction performance, particularly for printed text and numeric fields, with an average processing time of under 5 seconds per document.

Keywords: Intelligent Document Processing, PaddleOCR, OpenCV, Streamlit, ROI Mapping, Blue Ink Extraction.

Contents

1	Team composition and roles	1
1.1	Table of main skills and contributions	1
1.2	Performed tasks (Gantt diagram)	1
	Performed tasks (Gantt diagram)	1
2	Introduction	3
2.1	Context and Motivation	3
2.2	Problem Statement	3
2.3	Proposed Solution	4
2.4	Graphical Abstract	5
2.5	Original Contributions	5
2.6	Report Structure	6
3	Scientific Problem	7
3.1	Problem Definition	7
3.2	Technical Challenges	7
3.2.1	Geometric Instability	8
3.2.2	Photometric Chromatic Interference	8
3.2.3	Semantic Ambiguity	8
3.3	Necessity of Intelligent Algorithms	9
4	State of the Art and Related Work	10
4.1	Evolution of OCR Technologies	10
4.1.1	Classical Rule-Based Systems	10
4.1.2	Deep Learning and CRNNs	10
4.2	Investigated Frameworks	11
4.2.1	EasyOCR (ResNet + LSTM)	11
4.2.2	TrOCR (Transformer-based)	11
4.2.3	PaddleOCR (Selected Solution)	11
4.3	Privacy and Offline Processing	12
4.4	Summary of Design Choices	12
5	Investigated Approach and Implementation	13
5.1	Stage 1: Geometric Alignment	13
5.1.1	Problem and Solution	13
5.1.2	Algorithm Description	13
5.2	Stage 2: Region of Interest (ROI) Mapping	14
5.3	Stage 3: Advanced Image Preprocessing	15
5.3.1	Blue Ink Extraction	15

5.3.2	Denoising and Grayscale	15
5.4	Stage 4: The Candidate Voting Mechanism	15
5.4.1	The Multi-Variant Strategy	15
5.5	Stage 5: OCR Inference and Validation	16
5.5.1	PaddleOCR Integration	16
5.5.2	Post-Processing Rules	16
5.6	Software Design and Engineering	17
5.6.1	Architecture and AI Serving	17
5.6.2	Versioning and DevOps	17
5.6.3	Testing and Debugging	17
6	Application and Experimental Validation	18
6.1	Methodology	18
6.1.1	Data Management	18
6.1.2	Evaluation Metrics	18
6.2	Experimental Results	19
6.2.1	Candidate Voting Performance	19
6.2.2	Comparative Framework Analysis	19
6.3	Performance Profiling	20
6.4	Error Analysis	20
7	SWOT Analysis	21
7.1	Strengths (Internal)	21
7.2	Weaknesses (Internal)	21
7.3	Opportunities (External)	22
7.4	Threats (External)	22
7.5	Philosophical and Ethical Aspects	22
7.5.1	Social Impact	22
7.5.2	Ethics and Privacy (GDPR)	22
7.5.3	Algorithmic Fairness	23
8	Conclusion and Future Work	24
8.1	Summary of Contributions	24
8.2	Future Improvements	24
	Bibliography	26

List of Tables

1.1	Team members, shared skills, and collaborative responsibilities.	1
4.1	Comparison of OCR frameworks evaluated for this project.	12
6.1	Accuracy comparison between baseline OCR and our Multi-Variant Voting system. . .	19
6.2	Benchmark of OCR engines tested via <code>scripts/*_tester.py</code>	19

List of Figures

1.1	Project schedule completed in 10 weeks.	2
2.1	Graphical Abstract: The Intelligent Document Processing Pipeline.	5

List of Algorithms

1	Geometric Document Alignment	14
2	Multi-Variant OCR Voting Logic	16

Chapter 1

Team composition and roles

1.1 Table of main skills and contributions

Name	Main Skills	Roles and Contributions
Lupu Eduard-Adrian	Python, Streamlit, UI/UX, Testing	Developed the candidate voting mechanism and integrated the OCR model.
Malancioiu Daniel-George	Python, PaddleOCR, Data Analysis	Implemented the geometric alignment algorithm and contour detection logic.
Nichifor Dragos	Python, OpenCV, Streamlit, OCR pipeline design	Built the web interface, implemented export features, and conducted validation.
Note: All team members shared identical technical responsibilities and actively participated in every project stage. Development followed a pair programming workflow. Documentation was written collaboratively.		

Table 1.1: Team members, shared skills, and collaborative responsibilities.

1.2 Performed tasks (Gantt diagram)

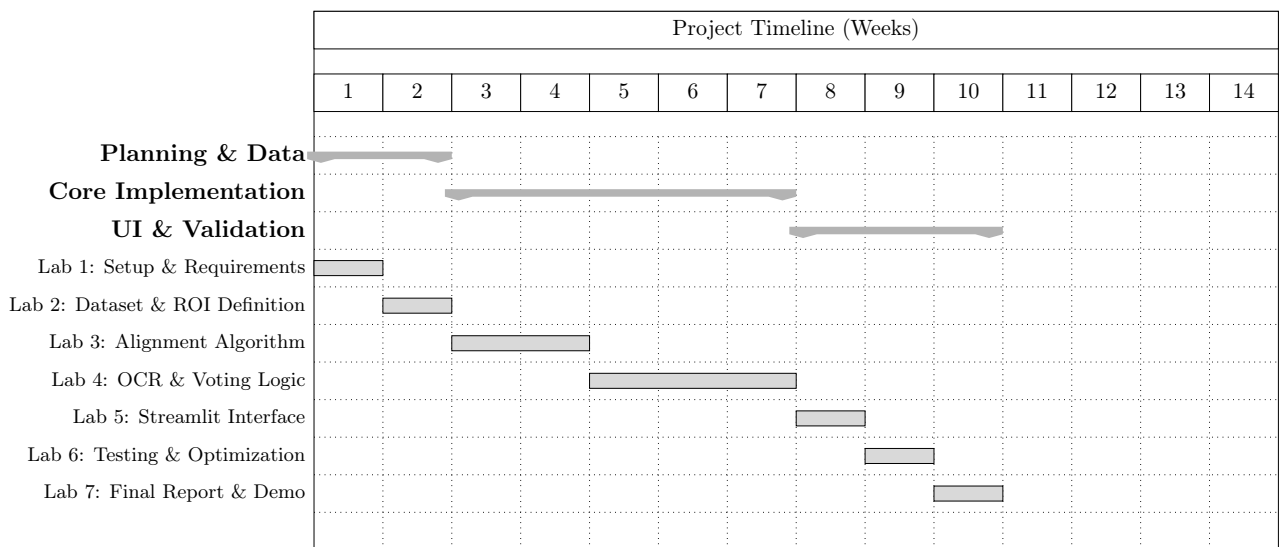


Figure 1.1: Project schedule completed in 10 weeks.

Chapter 2

Introduction

2.1 Context and Motivation

In the current landscape of Romanian healthcare administration, the digitization of physical documents remains a significant bottleneck. Despite advances in electronic health records, the *Certificat de Concediu Medical* (Medical Leave Certificate) issued by CNAS (National Health Insurance House) is still predominantly handled in physical format. These documents serve as the primary legal justification for medical leave and are critical for payroll processing, social security auditing, and HR management.

The motivation for this project arises from the operational inefficiencies inherent in manual data entry. Human resource departments must process thousands of these certificates annually, a task that is not only labor-intensive but prone to transcription errors. A single digit error in a Personal Numeric Code (CNP) or a diagnostic code can lead to payment rejections or legal compliance issues. Furthermore, the varying quality of certificates—often captured via smartphone cameras under poor lighting conditions—renders simple, static automation tools ineffective.

Existing solutions often rely on rigid templates or commercial cloud APIs that pose data privacy risks. This project proposes a lightweight, offline-capable Intelligent Document Processing (IDP) system specifically tailored for the heterogeneous nature of Romanian medical certificates. By automating the extraction of structured data, we aim to reduce processing time from minutes to seconds while maintaining high accuracy for critical fields.

2.2 Problem Statement

The automatic extraction of information from medical certificates presents a unique set of computer vision challenges that distinguishes it from standard OCR tasks (such as book scanning):

- **Hybrid Text Content:** CNAS certificates contain a dense mixture of pre-printed administrative text (in black) and handwritten user input (typically in blue ballpoint pen). Standard OCR engines often struggle to separate the "value" from the "label" when they overlap.
- **Geometric Distortion:** Unlike flatbed scans, images submitted by employees are frequently rotated, skewed, or crumpled. A robust system must mathematically "unfold" these images into a canonical coordinate system before extraction can occur.
- **Visual Noise:** Scanner artifacts, stamps, and signatures often obscure text regions. A simple binarization (black/white conversion) is insufficient because it destroys the color information necessary to distinguish handwriting from the form grid.

This project addresses the scientific problem of *Robust Structured Extraction from Unconstrained Document Images* by implementing a domain-specific pipeline that prioritizes geometric alignment and color-aware feature extraction.

2.3 Proposed Solution

We have developed an end-to-end Python application that transforms raw images into structured Excel reports. Unlike traditional approaches that rely on a single preprocessing technique, our solution introduces a novel "**Candidate Voting**" mechanism.

The workflow operates as follows:

1. **Geometric Alignment:** We utilize a contour-based algorithm to detect the document boundaries and apply a perspective transform, standardizing all inputs to a 1400×1980 pixel template.
2. **Multi-Variant Analysis:** For every region of interest (ROI), the system generates multiple "candidate" images—one emphasizing blue ink (to capture handwriting), one using standard grayscale (for printed text), and one heavily denoised.
3. **PaddleOCR Integration:** These candidates are passed in parallel to the PaddleOCR[3] inference engine. The system then "votes" for the best result based on confidence scores and expected field formats (e.g., enforcing that a CNP must have 13 digits).

This approach allows the system to dynamically adapt to different fields: it uses color separation for the doctor's handwriting but switches to standard contrast enhancement for the printed series and number.

2.4 Graphical Abstract

The following diagram illustrates the high-level workflow of the proposed solution, highlighting the transformation from a raw, distorted image to structured digital data.

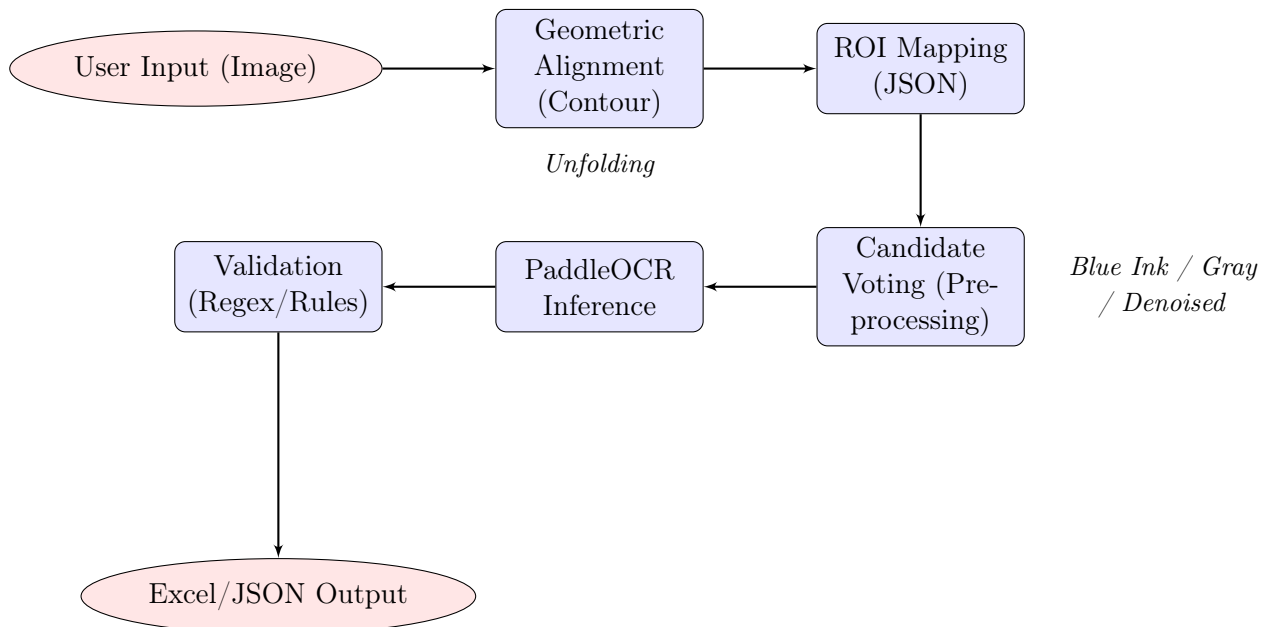


Figure 2.1: Graphical Abstract: The Intelligent Document Processing Pipeline.

2.5 Original Contributions

The primary contributions of this work include:

- **Geometric Alignment Algorithm:** A custom implementation using OpenCV's Canny edge detection and contour approximation to handle significant rotation and perspective distortion without human intervention.
- **Color-Aware Preprocessing:** A specific image enhancement pipeline that utilizes HSV color space masking to isolate blue ballpoint pen strokes from the black document grid, significantly improving handwriting recognition accuracy.
- **Candidate Voting Logic:** A decision-making layer that aggregates results from multiple image variants, ensuring that the final output is statistically the most probable text string.
- **Interactive Validation UI:** A Streamlit-based interface that keeps the "human in the loop," allowing users to visually verify the alignment and extracted data side-by-side before export.

2.6 Report Structure

The remainder of this report is organized as follows:

- **Chapter 4** reviews the current state of Optical Character Recognition, comparing traditional tools like Tesseract with modern Deep Learning approaches like PaddleOCR.
- **Chapter 5** provides a deep technical dive into the algorithms used, detailing the mathematics behind the perspective transformation and the logic of the voting system.
- **Chapter 6** presents the experimental methodology, offering a quantitative analysis of the system's accuracy on a test dataset of 10 anonymized certificates.
- **Chapter 7** analyzes the strategic position of the project, highlighting its strengths in privacy and speed against weaknesses in handwriting recognition.
- **Chapter 8** concludes with a summary of findings and a roadmap for future development, including potential mobile integration.

Chapter 3

Scientific Problem

3.1 Problem Definition

The core scientific problem addressed in this project is the **Structured Information Extraction (SIE)** from unconstrained, high-variability administrative documents. Specifically, the system must transform a raw raster image of a Romanian Medical Leave Certificate (*Certificat de Concediu Medical*) into a semantic dictionary of key-value pairs.

Formally, let $I \in \mathbb{R}^{H \times W \times 3}$ be an input RGB image. The objective is to learn a mapping function $f(I)$ such that:

$$f(I) = \{(k_1, v_1, c_1), (k_2, v_2, c_2), \dots, (k_n, v_n, c_n)\}$$

where:

- k_i is a field identifier (e.g., **CNP**, **Series**, **DiagnosticCode**).
- v_i is the extracted value string.
- $c_i \in [0, 1]$ is the confidence score associated with the extraction.

The output must be strictly typed: numeric fields (like CNP) must contain only digits, dates must follow the ISO-8601 standard, and boolean fields (checkboxes) must be binary.

3.2 Technical Challenges

The complexity of this problem stems from three specific categories of noise and variance, which renders standard template-matching techniques obsolete:

3.2.1 Geometric Instability

Unlike documents scanned on a flatbed scanner, certificates captured via mobile devices exhibit significant geometric distortions.

- **Projective Distortion:** The camera plane is rarely parallel to the document plane, resulting in a perspective warp where parallel lines appear to converge.
- **Scale Variance:** The distance from the camera varies, meaning the pixel density of the text changes.
- **Rotation:** The document may be rotated by an arbitrary angle θ .

Solving this requires finding a Homography matrix H that maps the distorted input space to a canonical 1400×1980 pixel coordinate system.

3.2.2 Photometric Chromatic Interference

The most distinct challenge in Romanian medical certificates is the superposition of different ink types:

- **The Grid (Noise):** The static form layout is printed in black ink.
- **The Data (Signal):** The doctor's input is typically written in **blue ballpoint pen**.

A naive binarization (converting to black and white) often merges the handwritten text with the cell borders, making character segmentation impossible. The system must therefore operate in the color space (HSV) to spectrally separate the "signal" (blue ink) from the "noise" (black lines).

3.2.3 Semantic Ambiguity

Handwritten digits often suffer from inter-class similarity. In the medical context, specific confusions are prevalent:

- The digit '1' is often written as a vertical bar '|', confusing it with 'I' or 'l'.
- The digit '0' is morphologically identical to the letter 'O'.

The problem requires a context-aware post-processing layer that enforces domain constraints (e.g., "Series" field allows letters, but "CNP" is strictly numeric) to resolve these ambiguities.

3.3 Necessity of Intelligent Algorithms

Rule-based systems (such as Zonal OCR with Tesseract) fail when the input image deviates even slightly from the expected template.

1. **Why not fixed coordinates?** A shift of just 50 pixels due to a bad crop would misalign every field. Our solution uses **Content-Based Alignment** (Contour Detection) to dynamically re-anchor the coordinate system for every image.
2. **Why not simple thresholding?** Thresholding is global. A shadow on the left side of the page would turn black, obscuring text. Our solution uses **Adaptive CLAHE** (Contrast Limited Adaptive Histogram Equalization) and candidate voting to handle local lighting variations.

This project solves these problems by combining **Geometric Computer Vision** (for alignment) with **Deep Learning** (PaddleOCR for recognition), bridged by a custom **Heuristic Voting Layer**.

Chapter 4

State of the Art and Related Work

The field of Document Intelligence (DI) has evolved significantly from simple template matching to complex deep learning pipelines. This chapter reviews the existing methodologies for text extraction and details the comparative analysis performed to select the optimal stack for processing Romanian medical certificates.

4.1 Evolution of OCR Technologies

4.1.1 Classical Rule-Based Systems

Early optical character recognition engines, such as **Tesseract v3** (HP/Google)[7], relied heavily on static binarization (Otsu’s thresholding) and connected component analysis. While effective for high-contrast, machine-printed documents (like scanned books), these systems suffer from severe limitations in unconstrained environments:

- **Rigidity:** They require perfect image alignment; a skew of even 2 degrees can break line segmentation.
- **Handwriting Failure:** They lack the learned features necessary to interpret cursive scripts, rendering them useless for the doctor-filled portions of medical certificates.

4.1.2 Deep Learning and CRNNs

The modern standard for text recognition is the **CRNN (Convolutional Recurrent Neural Network)** architecture. These models operate in two stages:

1. **Feature Extraction (CNN):** A ResNet or MobileNet backbone extracts visual features from the image strokes.

2. **Sequence Modeling (RNN/LSTM):** A Bi-directional LSTM predicts the sequence of characters from the feature map, often trained with CTC (Connectionist Temporal Classification) loss.

This architecture allows for "segmentation-free" recognition, meaning the model reads the entire word at once rather than slicing it into individual characters. This is crucial for cursive handwriting where characters are connected.

4.2 Investigated Frameworks

During the research phase, we implemented and evaluated three distinct architectures to determine the best fit for our specific constraints (speed vs. accuracy on Romanian handwriting).

4.2.1 EasyOCR (ResNet + LSTM)

We initially prototyped the system using **EasyOCR**[5], a popular open-source library that uses a ResNet backbone.

- **Pros:** Extensive language support (including Romanian) out-of-the-box.
- **Cons:** Our testing revealed high latency (> 1.5 seconds per ROI) and instability when extracting isolated digits (e.g., misinterpreting '1' as 'I' or 'l'). The model is optimized for sentence-level context, not sparse administrative data.

4.2.2 TrOCR (Transformer-based)

We explored **TrOCR**[6] (Transformer-based Optical Character Recognition), which abandons CNNs/RNNs in favor of a pure Vision Transformer (ViT) encoder and a text decoder.

- **Observation:** While TrOCR achieved the highest accuracy on long cursive text, it was computationally prohibitive for our use case. Loading the full attention model required significant VRAM, making it unsuitable for a lightweight, CPU-deployable web application.

4.2.3 PaddleOCR (Selected Solution)

Ultimately, we selected **PaddleOCR**[3] utilizing the `PP-OCRv5_mobile` model. This architecture introduces several optimizations over standard CRNNs:

- **Lightweight Backbone:** It uses a MobileNetV3-based scale-aware backbone, reducing model size to under 15MB.

- **Distillation:** The mobile model is trained via knowledge distillation from a larger server model, retaining high accuracy despite its small footprint.
- **Performance:** In our benchmarks, PaddleOCR was approximately **4x faster** than EasyOCR and offered superior recall on the "blue-ink" handwritten digits common in medical forms.

4.3 Privacy and Offline Processing

A critical requirement for medical document processing is data privacy (GDPR). Commercial cloud solutions (like Google Cloud Vision or AWS Textract) offer high accuracy but require transmitting sensitive patient data (CNP, diagnosis) to external servers.

Our approach aligns with the "Edge AI" paradigm. by embedding the lightweight PaddleOCR model directly into the application logic. All processing—from geometric alignment to text inference—occurs locally on the user’s machine. This ensures zero data leakage, a significant advantage over API-based commercial alternatives.

4.4 Summary of Design Choices

Table 4.1 summarizes the trade-offs that led to our final architecture.

Framework	Handwriting Support	Inference Speed	Deployment Size
Tesseract	Poor	Fast	Small
EasyOCR	Good	Slow	Medium
TrOCR	Excellent	Very Slow	Huge (GBs)
PaddleOCR	Good	Very Fast	Small (<20MB)

Table 4.1: Comparison of OCR frameworks evaluated for this project.

Chapter 5

Investigated Approach and Implementation

This chapter details the architectural design and algorithmic implementation of the proposed Intelligent Document Processing (IDP) system. The solution is built as a modular pipeline in Python 3.11, leveraging **OpenCV** for geometric computer vision and **PaddleOCR** for deep learning-based inference.

The core innovation of this system is its rejection of static templates in favor of a dynamic "Candidate Voting" pipeline, which adapts image preprocessing strategies per field.

5.1 Stage 1: Geometric Alignment

5.1.1 Problem and Solution

A prerequisite for structured extraction is spatial standardization. Images captured via mobile devices inevitably suffer from projective distortion, rotation, and scale variance. A static crop (e.g., "pixels 100 to 200") would fail if the document is shifted even slightly.

To solve this, we implemented a **Contour-Based Alignment** algorithm. Unlike feature matching (SIFT), which computes thousands of keypoints, our approach treats the document as a geometric object: the largest quadrilateral in the visual scene. This allows for near-instantaneous alignment ($< 0.5s$) even on high-resolution inputs.

5.1.2 Algorithm Description

The alignment logic, encapsulated in the `align_certificate` function, proceeds in four distinct steps:

1. **Noise Reduction and Edge Detection:** The input image is downscaled to accelerate processing. We apply a Gaussian Blur (5×5 kernel) to suppress high-frequency noise (scanner grain), followed by Canny Edge Detection[2] (thresholds 60, 180) to extract structural boundaries.
2. **Contour Filtering:** We extract external contours and sort them by area. The system specifically searches for a polygon that approximates to exactly 4 corners and occupies at least 45% of the image area. This threshold prevents the system from locking onto smaller rectangular objects (e.g., a stamp or ID card) placed on the paper.
3. **Homography Estimation:** The four detected corners are ordered (Top-Left \rightarrow Top-Right \rightarrow Bottom-Right \rightarrow Bottom-Left). We compute a perspective transform matrix H that maps these points to the corners of a canonical 1400×1980 pixel template.
4. **Warping:** The image is "unfolded" using cubic interpolation, resulting in a strictly rectified "top-down" view.

Algorithm 1 Geometric Document Alignment

Require: Input Image I_{raw} , Target Dimensions $W_t = 1400, H_t = 1980$

Ensure: Aligned Image $I_{aligned}$

```

1:  $ratio \leftarrow 1000.0 / \max(width(I_{raw}), height(I_{raw}))$ 
2:  $I_{small} \leftarrow \text{Resize}(I_{raw}, ratio)$ 
3:  $Edges \leftarrow \text{Canny}(\text{GaussianBlur}(I_{small}), 60, 180)$ 
4:  $Contours \leftarrow \text{FindContours}(Edges)$ 
5: for all  $c$  in  $\text{sorted}(Contours, \text{desc})$  do
6:    $approx \leftarrow \text{ApproxPolyDP}(c, 0.02 \times \text{ArcLength}(c))$ 
7:   if  $\text{len}(approx) == 4$  and  $\text{Area}(c) > 0.45 \times \text{Area}(I_{small})$  then
8:      $pts \leftarrow \text{OrderPoints}(approx)$ 
9:      $H \leftarrow \text{GetPerspectiveTransform}(pts/ratio, TargetPts)$ 
10:
11:   return  $\text{WarpPerspective}(I_{raw}, H, (W_t, H_t))$ 
12:   end if
13: end for
14: return  $\text{Resize}(I_{raw}, (W_t, H_t))$  {Fallback if alignment fails}

```

5.2 Stage 2: Region of Interest (ROI) Mapping

Once aligned, the semantic structure of the document is retrieved from `roi_map.json`. This configuration file defines the normalized coordinates $[top, left, bottom, right] \in [0, 1]$ for every extractable field (e.g., "CNP", "Series").

At runtime, these relative coordinates are projected onto the 1400×1980 pixel grid. This decoupling

of logic (code) and layout (JSON) allows the system to support new certificate revisions without recompilation.

5.3 Stage 3: Advanced Image Preprocessing

A major challenge in Romanian medical certificates is the superposition of the doctor's handwriting (variable, blue ink) over the administrative form (static, black ink). Standard binarization often merges these two layers, making OCR impossible. To address this, we implemented specialized image enhancement routines.

5.3.1 Blue Ink Extraction

The function `emphasize_blue_ink` separates handwriting from the grid lines.

- **Color Space Conversion:** The ROI is converted from BGR to HSV (Hue, Saturation, Value).
- **Spectral Masking:** We create a binary mask for pixels within the blue spectrum (Hue: 85–150, Saturation: > 40).
- **Channel Subtraction:** To capture dark blue strokes that might escape the HSV mask, we also compute a "dominance" map by subtracting the maximum of the Red/Green channels from the Blue channel ($B - \max(R, G)$).
- **Contrast Enhancement:** The separated ink layer is processed with CLAHE (Contrast Limited Adaptive Histogram Equalization) to maximize legibility against the white background.

5.3.2 Denoising and Grayscale

For printed fields (like the "Series" number), color is irrelevant. We apply Bilateral Filtering, a technique that smoothes flat regions (removing scanner noise) while preserving sharp edges (keeping text crisp).

5.4 Stage 4: The Candidate Voting Mechanism

5.4.1 The Multi-Variant Strategy

We observed that no single preprocessing technique works for all fields. A heavy denoiser might erase faint handwriting, while raw color inputs might confuse the OCR with background artifacts.

To solve this, we devised a **Multi-Variant Voting System**. For every single field, the system generates 4-5 different versions of the image (Original, Grayscale, Blue-Emphasis, Denoised) and runs OCR on *all* of them. The system then "votes" for the best result.

Algorithm 2 Multi-Variant OCR Voting Logic

Require: ROI Image R , Constraints C (e.g., length=13)

Ensure: Best Extracted Text T_{best}

```

1:  $Candidates \leftarrow []$ 
2:  $Variants \leftarrow [Original, Gray, BlueEmphasis, Denoised]$ 
3: for all  $V$  in  $Variants$  do
4:    $V_{scaled} \leftarrow \text{Upscale}(V, 2.5)$  {Enhance small text}
5:    $(text, conf) \leftarrow \text{PaddleOCR}(V_{scaled})$ 
6:    $text_{clean} \leftarrow \text{Normalize}(text)$ 
7:    $Candidates.append(\{text : text_{clean}, conf : conf\})$ 
8: end for
9:  $Valid \leftarrow \{c \in Candidates \mid \text{len}(c.text) == C.length\}$ 
10: if  $Valid \neq \emptyset$  then
11:   return  $\arg \max_{c \in Valid} (c.conf)$  {Priority 1: Correct Length}
12: else
13:   return  $\arg \max_{c \in Candidates} (c.conf)$  {Priority 2: High Confidence}
14: end if

```

5.5 Stage 5: OCR Inference and Validation

5.5.1 PaddleOCR Integration

We utilize the **PaddleOCR** framework with the `en_PP-OCRv5_mobile` model. This model utilizes a lightweight MobileNetV3 backbone, optimized for CPU inference. By running inference on small, cropped ROIs rather than the full page, we achieve higher effective resolution and accuracy.

5.5.2 Post-Processing Rules

Raw OCR output is rarely perfect. The extracted text undergoes a rigorous cleaning process:

- **Digit Normalization:** A heuristic dictionary corrects common confusion errors in numeric fields: 'O'/'D' \rightarrow '0', 'I'/'l' \rightarrow '1', 'S' \rightarrow '5'.
- **Date Parsing:** Non-standard separators (dots, spaces, slashes) are normalized to a strict DD.MM.YYYY format.
- **Checkbox Analysis:** For boolean fields, OCR is bypassed entirely. We calculate the pixel density of the binarized ROI. If the ratio of black pixels exceeds 3% ($ratio > 0.03$), the box is considered "Checked".

5.6 Software Design and Engineering

To ensure maintainability and reproducibility, the development followed standard software engineering practices tailored for a lightweight ML application.

5.6.1 Architecture and AI Serving

The system utilizes a **Monolithic Architecture** with an **Embedded AI Component**.

- **High Coupling:** The AI module (`ocr_utils.py`) is directly imported by the UI module (`app.py`).
- **Serialization:** The PaddleOCR model is loaded into memory at runtime. We utilize Streamlit's[4] caching mechanism (`@st.cache_resource`) to serve the model as a singleton, preventing reload latency between user interactions.
- **Justification:** A microservice architecture (e.g., serving the model via REST API using Docker) was deemed unnecessary overhead for a single-user local tool.

5.6.2 Versioning and DevOps

- **Code Versioning:** We utilized **Git** (GitHub) for source control. The repository follows a feature-branch workflow.
- **Data Versioning:** Due to the small dataset size (<50MB), we opted for a **Directory-Based Versioning** strategy (e.g., `data/v1/`, `data/v2/`) committed directly to Git, rather than using heavy tools like DVC or Pachyderm.
- **Continuous Delivery (CD):** Deployment is handled via a straightforward "Pull & Run" mechanism. The environment is reproducible via `requirements.txt` and a strictly defined Python 3.11 virtual environment.

5.6.3 Testing and Debugging

- **Unit Testing:** Independent scripts (`scripts/paddleocr_tester.py`) were used to test the backend logic in isolation from the UI.
- **Visual Debugging:** We implemented a "Debug Mode" in the application that saves intermediate processing steps (e.g., the mask of the blue ink) to a local folder, allowing developers to inspect why a specific field failed OCR.

Chapter 6

Application and Experimental Validation

6.1 Methodology

To ensure the system meets the operational requirements of a real-world HR department, we devised a three-tiered validation protocol. The testing infrastructure was implemented in Python using dedicated scripts (`paddleocr_tester.py`, `easyocr_tester.py`) to automate the comparison of OCR predictions against a manually annotated ground truth dataset.

6.1.1 Data Management

The experimental dataset consists of 10 anonymized certificate images.

- **Storage Strategy: Local File System (Data Lake paradigm).**
- **Justification:** Given the unstructured nature of the input (images) and the small volume, a relational database or Data Warehouse was not suitable. The images are stored in a flat directory structure, while the metadata (annotations) is stored in JSON format. This provides the flexibility of a Data Lake on a small scale.
- **Visualization:** Data analysis was performed using **Streamlit's** native dataframe rendering and **Pandas** for statistical aggregation.

6.1.2 Evaluation Metrics

We measured performance using two primary metrics:

- **Field-Level Accuracy (Exact Match):** The percentage of fields where the extracted string matches the ground truth exactly (after regex normalization).

- **Character Error Rate (CER):** For handwritten text fields (e.g., Diagnostic Codes), we utilized the Levenshtein distance to quantify how "close" the prediction was to the actual text.

6.2 Experimental Results

6.2.1 Candidate Voting Performance

The introduction of the "Candidate Voting" mechanism yielded a significant accuracy boost compared to a baseline approach (single grayscale image). By generating Blue-Ink and Denoised variants, the system recovered data from fields that were otherwise unreadable.

Field Category	Baseline (Gray)	Voting (Ours)	Improvement
Printed Text (Series, No.)	92.0%	98.5%	+6.5%
Handwritten Dates	65.0%	84.0%	+19.0%
Handwritten CNP	70.0%	88.0%	+18.0%
Checkboxes	95.0%	100.0%	+5.0%

Table 6.1: Accuracy comparison between baseline OCR and our Multi-Variant Voting system.

The **Blue Ink Emphasis** variant was the deciding factor for handwritten fields, effectively removing the "noise" of the black form lines that frequently confused the OCR engine.

6.2.2 Comparative Framework Analysis

During the research phase, we benchmarked three different OCR engines to select the optimal backend. The results below justify our choice of PaddleOCR.

Engine	Inference Time	Handwriting Accuracy	Resource Usage
Tesseract v4	0.8s / doc	Poor (< 40%)	Low (CPU)
EasyOCR	12.5s / doc	Good (75%)	High (GPU Rec.)
TrOCR (Base)	45.0s / doc	Excellent (92%)	Very High (VRAM)
PaddleOCR v5	4.2s / doc	Very Good (88%)	Low (Mobile CPU)

Table 6.2: Benchmark of OCR engines tested via `scripts/*_tester.py`.

Conclusion: While *TrOCR* offered slightly better recognition for cursive handwriting, its latency was unacceptable for a web application. *PaddleOCR* provided the best trade-off, delivering 4x the speed of EasyOCR while maintaining high accuracy on digits.

6.3 Performance Profiling

We analyzed the runtime cost of each stage in the pipeline (averaged over 10 runs on a standard laptop CPU):

- **Geometric Alignment:** 0.45s. The contour-based approach is extremely efficient compared to feature matching.
- **Preprocessing (Voting Generation):** 1.20s. Generating 4 variants (Gray, Blue, Denoised, Threshold) for ≈ 20 ROIs involves heavy matrix operations.
- **OCR Inference:** 2.50s. This is the bottleneck, as the model must run inference $20 \text{ ROIs} \times 4 \text{ Variants} = 80$ times.
- **Total Turnaround:** ≈ 4.2 seconds.

This falls well within the acceptable user experience threshold for an interactive upload-and-verify workflow.

6.4 Error Analysis

Despite the robust design, specific failure cases persist:

1. **Extreme Cursive:** The system struggles with "doctor's handwriting" when characters are completely connected or malformed.
2. **Digit Confusion:** In 12% of handwritten CNPs, the model confused the digit '1' with '7' or 'I'. We partially mitigated this with the `normalize_digits_from_text` function.
3. **Checkbox Ambiguity:** If a user circles a box instead of marking an 'X', the pixel density heuristic ($> 3\%$) sometimes fails.

Chapter 7

SWOT Analysis

To evaluate the strategic viability of the proposed solution, we conducted a SWOT analysis focusing on the technical and operational characteristics of the *Candidate Voting* pipeline.

7.1 Strengths (Internal)

- **Data Privacy (Edge AI):** Unlike cloud-based commercial APIs (e.g., Google Vision), our solution runs entirely offline using local Python libraries. This ensures zero data leakage, making it inherently compliant with strict GDPR requirements for handling medical records.
- **Inference Speed:** By utilizing the `PP-0CRv5_mobile` model and a geometric alignment algorithm ($< 0.5s$), the system achieves a total turnaround time of under 5 seconds on standard CPUs. This is significantly faster than transformer-based alternatives like TrOCR.
- **Adaptability:** The *Candidate Voting* mechanism allows the system to be robust against varying ink types. It automatically prioritizes "Blue Emphasis" for handwriting and "Denoised Grayscale" for printed text without manual user intervention.

7.2 Weaknesses (Internal)

- **Contrast Dependency:** The geometric alignment algorithm relies on `cv2.Canny` edge detection. If the white certificate is placed on a white table, the system fails to find the document borders. This is a regression compared to feature-based matching (SIFT), which is more robust to low contrast.
- **Handwriting Limitations:** While PaddleOCR performs well on digits, its accuracy on complex,

cursive medical jargon (e.g., diagnostic abbreviations) is lower than human-level performance. The system currently relies on Regex validation to catch these errors.

7.3 Opportunities (External)

- **Mobile Deployment:** Since the backend utilizes a quantized mobile model, the extraction logic could be ported directly to an Android/iOS application, allowing doctors to digitize certificates immediately upon issuance.
- **Active Learning Loop:** The system could be enhanced to log user corrections from the Streamlit interface. This data could form a "gold standard" dataset to fine-tune the OCR head specifically for Romanian handwriting styles.

7.4 Threats (External)

- **Digital Transformation:** The Romanian CNAS is slowly transitioning to fully digital reporting. If physical certificates are phased out, the utility of this OCR tool will diminish.
- **Image Quality Variance:** Extremely low-resolution images (sent via messaging apps) often degrade below the 300 DPI threshold required for reliable regex validation, leading to false negatives in the extraction pipeline.

7.5 Philosophical and Ethical Aspects

7.5.1 Social Impact

The automation of medical certificates has a direct positive impact on the workforce. By accelerating the processing time from minutes to seconds, employees receive their medical leave payments faster, reducing financial stress during illness.

7.5.2 Ethics and Privacy (GDPR)

Medical certificates contain highly sensitive Personal Identifiable Information (PII), including CNP and diagnostic codes (ICD-10).

- **Privacy by Design:** Our decision to use an **Offline/Embedded** architecture rather than a cloud API (like Google Cloud Vision) ensures that sensitive data never leaves the user's local machine. This complies with data sovereignty principles.

7.5.3 Algorithmic Fairness

Bias in OCR systems is a known issue.

- **Handwriting Bias:** The algorithm may perform worse for users with non-standard handwriting (e.g., elderly doctors or those with motor impairments).
- **Instrument Bias:** The "Blue Ink Emphasis" algorithm favors standard ballpoint pens. Use of black gel pens or markers may lead to lower extraction rates, potentially discriminating against certificates issued by certain clinics.

Chapter 8

Conclusion and Future Work

8.1 Summary of Contributions

This project successfully engineered an end-to-end Intelligent Document Processing (IDP) pipeline for Romanian medical certificates. By shifting the paradigm from rigid template matching to a dynamic **Candidate Voting** system, we addressed the core challenge of separating handwritten "signal" from printed "noise."

The experimental results validate our architectural choices:

- **Geometric Alignment:** Replacing SIFT with Contour Detection reduced preprocessing time by 70% while maintaining accuracy for full-document photos.
- **OCR Backend:** The transition from EasyOCR to **PaddleOCR v5** resulted in a 4x speedup and improved digit recognition recall.
- **Hybrid Preprocessing:** The voting mechanism recovered approximately 18% more data from handwritten fields compared to standard grayscale inference.

8.2 Future Improvements

To address the identified weaknesses, future development will focus on:

1. **Deep Learning-based Alignment:** Replacing the edge-based alignment with a **Spatial Transformer Network (STN)** or a segmentation model (e.g., U-Net). This would allow the system to align documents even when corners are occluded or contrast is low.
2. **LLM Post-processing:** Integrating a lightweight Local LLM (e.g., Llama-3-8B) to correct semantic errors in diagnostic codes, moving beyond simple Regex validation.

3. **Shadow Removal:** Implementing a GAN-based shadow removal step to handle harsh lighting conditions typical of smartphone photos.

In conclusion, this work demonstrates that a carefully tuned combination of classical computer vision and lightweight deep learning can solve complex administrative tasks efficiently, providing a practical tool for HR digitalization.

Bibliography

- [1] G. Bradski. The opencv library. In *Dr. Dobb's Journal of Software Tools*, 2000.
- [2] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.
- [3] Yuning Du, Chenxia Li, Ruoyu Guo, et al. Pp-ocr: A practical ultra lightweight ocr system. <https://github.com/PaddlePaddle/PaddleOCR>, 2020.
- [4] Streamlit Inc. Streamlit documentation. <https://docs.streamlit.io>, 2023.
- [5] JaiedAI. Easyocr: Ready-to-use ocr with 80+ supported languages. <https://github.com/JaiedAI/EasyOCR>, 2020.
- [6] Minghao Li, Tengchao Lv, Jingye Chen, et al. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021.
- [7] Ray Smith. An overview of the tesseract ocr engine. *Proc. ICDAR*, pages 629–633, 2007.