Cloud-based intelligent architecture and geometric post-processing for vehicle number plate recognition*

Volodymyr Avsiievych[1,*,†], Olga Pavlova[1,†], Andrii Kuzmin[1,†], Houda El Bouhissi[2,†] and Miroslav Kvassay[3,†]

[1] *Khmelnytskyi National University, Instytutska 11, 29000, Khmelnytskyi, Ukraine*
[2] *LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, 06000, Bejaia, Algeria*
[3] *Zilina University, Univerzitná 8215, 010 26 Žilina, Slovakia*

Abstract

This paper proposes a novel hybrid cyber-physical architecture for smart parking systems with automated license plate recognition on edge devices that resolves this trade-off. By offloading resource-intensive optical character recognition to the cloud and executing a lightweight mathematical post-processing layer locally, the system ensures high identification accuracy without requiring discrete GPUs. The proposed methodology introduces a formal geometric model utilizing a ray-casting point-in-polygon algorithm to associate disjointed text bounding boxes with vehicle instances. Furthermore, a linguistic normalization morphism with context-aware algebraic substitutions is applied to correct OCR optical illusions. Experimental evaluation on a custom dataset of 215 images demonstrates that the proposed pipeline achieves a classification accuracy of 86.9%, a precision of 95.9%, and a recall of 90.0%, outperforming the naive cloud implementation baseline. The results confirm the viability of the hybrid approach for secure access control in smart city infrastructure

Keywords

smart parking, automated license plate recognition, Cloud Vision API, edge computing, geometric post-processing, cyber-physical systems[1]

1. Introduction

The rapid development of Smart City infrastructure has significantly increased the demand for automated cyber-physical systems, particularly in the domain of smart parking and access control. Automatic license plate recognition serves as the core technology for these systems, enabling barrier automation without continuous human intervention.

Historically, achieving high-accuracy ALPR required deploying specialized, computationally expensive deep learning models directly on local servers equipped with discrete GPUs. However, scaling such solutions across multiple parking gates using low-power Edge devices (e.g., standard CPUs or single-board computers, such as Raspberry Pi 5) leads to bottlenecks. To overcome these hardware limitations, modern architectures often shift towards distributed computing by offloading the optical character recognition tasks to general-purpose APIs (such as Google Cloud Vision).

While cloud-based models eliminate the need for local processing power and offer robust multilingual text extraction, they introduce a critical new problem: spatial fragmentation. General-purpose APIs lack domain-specific geometric constraints for vehicles. Consequently, they frequently return disjointed text blocks, confuse visually similar letters and numbers (e.g., 'O' and '0'), or capture irrelevant background noise, such as dealer advertisements printed on license plate frames.

Therefore, the primary aim of this research is to increase the accuracy of the ALPR process by

* *IntelITSIS-2026: 7th International Workshop on Intelligent Information Technologies & Systems of Information Security, July 3, 2026, Khmelnytskyi, Ukraine*
[1,*] Corresponding author.
[†] These authors contributed equally.
avsiievychvr@gmail.com (V. Avsiievych); pavlovao@khmnu.edu.ua (O. Pavlova); andriy1731@gmail.com (A. Kuzmin); houda.elbouhissi@gmail.com (H. El Bouhissi); miroslav.kvassay@fri.uniza.sk (M. Kvassay)
0009-0004-2040-8756 (V. Avsiievych); 0000-0001-7019-0354 (O. Pavlova); 0009-0005-6489-225X (A. Kuzmin); 0000-0003-3239-8255 (H. El Bouhissi); 0000-0002-6450-5417 (M. Kvassay)

developing a hybrid, resource-efficient cyber-physical architecture that uses cloud-based OCR while mitigating its optical and spatial illusions through a local post-processing layer.

The main contributions of this paper include:

- A formal geometric model based on the ray-casting algorithm to match fragmented text sequences with specific vehicle bounding boxes.
- A context-aware linguistic normalization method using formal grammars and algebraic substitutions to resolve typical OCR misinterpretations.
- The implementation of local and global spatial fallback mechanisms to prevent data loss in cases of object detector failures.

The structure of the article is as follows. Section 2 reviews the related works. Section 3 details the proposed mathematical model and software methodology. Section 4 presents the experimental setup, comparative analysis, and error evaluation, followed by the conclusions in Section 5.

2. Related works

The evolution of vehicle monitoring systems reflects a clear transition from traditional hardware sensors to advanced computer vision frameworks [1]. Comprehensive literature reviews on ALPR emphasize the robustness of deep learning architectures [2], [3], [4]. Multiple studies have proven the high efficiency of convolutional neural networks, YOLO object detectors, and MobileNet variants for real-time vehicle localization tasks [5], [6], [7, 8]. For the OCR stage, hybrid models such as CNN-RNN [9] and CNN-LSTM [10], [11] are widely recognized as state-of-the-art for sequence recognition.

Despite the high accuracy of these localized deep learning models, their deployment in real-world smart city infrastructure faces significant engineering challenges. Executing computationally heavy models (e.g., YOLOv8 or Faster R-CNN) requires powerful discrete GPUs, which are rarely available on low-power embedded edge platforms [12]. Recent benchmarking studies on edge AI energy consumption and inference times [13], [14] prove that running complex multi-stage ALPR models on standard CPUs (such as Raspberry Pi) leads to severe latency and thermal bottlenecks.

Consequently, researchers increasingly suggest the use of distributed computing paradigms by offloading heavy computational tasks—such as OCR and feature extraction—from Edge devices to powerful cloud-based APIs [15] for lowering hardware requirements. Comparative analyses have proven that general-purpose cloud services, such as Google Cloud Vision API, outperform traditional local OCR engines like Tesseract, especially for complex or degraded documents [16]. Moreover, the integration of advanced vision-language models and transformer-based architectures has shown potential in cloud-based optical recognition [17], [18], [19].

Nevertheless, deploying cloud vision APIs introduces a new fundamental challenge: these models are general-purpose and lack specific geometric constraints. As highlighted in recent document layout parsing research, cloud models often return fragmented text sequences, disjointed bounding boxes, and unstructured text blocks [20]. To address this, the scientific community has extensively studied post-OCR processing approaches [21]. Some methods rely only on lexical correction and domain-specific dictionary validation [22], [23]. Recent trends show that it's best to combine deep learning outputs with traditional image processing and mathematical heuristics [24]. Advanced studies have successfully employed geometric matching, graph convolutional networks based on bounding boxes [25], alignment algorithms using synthetic data mapping [26], and probabilistic heuristic rules [27] to reconstruct fragmented text regions.

Therefore, the purpose of this research is to increase the accuracy of the ALPR process by developing a hybrid, resource-efficient cyber-physical system that uses cloud-based OCR while mitigating its optical and spatial illusions through a local post-processing layer.

3. Proposed mathematical model and methodology

The proposed methodology follows a sequential data flow pipeline: video frame acquisition, cloud-based feature extraction, geometric spatial evaluation, and formal linguistic normalization.

To achieve high recognition accuracy on edge devices, the system implements a hybrid architecture that offloads heavy OCR tasks to the cloud while performing high-precision post-processing locally.

3.1. Formal geometric model of object localization

Let I be the input digital image defined as a matrix of dimension W x H. The cloud-based detection module identifies objects O and returns their coordinates as normalized polygons P_{norm} in the unit space $\mathbb{R}^2_{[0,1]}$. To facilitate local geometric analysis, these coordinates are mapped into the absolute pixel space $\mathbb{N}^2$ using a linear transformation morphism.

$$x_{abs} = x_{norm} \times W \tag{1}$$

$$y_{abs} = (1 - y_{norm}) \times H, \tag{2}$$

where

x_{norm} – normalized x coordinate of vertex;

y_{norm} – normalized y coordinate of vertex.

To mitigate localization errors caused by acute camera angles or partial obstructions, we define a confidence zone through a bounding box extension algorithm. Let $P = \{v_1, v_2, v_3, v_4\}$ be the coordinates of the vehicle bounding box. The vertices of the vehicle's polygon P are shifted outward by thresholds Δx and Δy calculated as a product of the image dimensions with heuristic coefficients α, β = 0.05.

$$\Delta x = \alpha \times W, \Delta y = \beta \times H. \tag{3}$$

The extended polygon P' is formed by shifting the vertices outward in the clockwise order:

$$v_1' = (\max(0, x_1 - \Delta x), \min(H, y_1 + \Delta y)) \tag{4}$$

$$v_2' = (\min(W, x_2 + \Delta x), \min(H, y_2 + \Delta y)) \tag{5}$$

$$v_3' = (\min(W, x_3 + \Delta x), \max(0, y_3 - \Delta y)) \tag{6}$$

$$v_4' = (\max(0, x_4 - \Delta x), \max(0, y_4 - \Delta y)). \tag{7}$$

3.2. Spatial point-in-polygon matching

A significant limitation of general-purpose cloud APIs is the fragmentation of recognized text blocks. To associate disjointed fragments with a specific vehicle, we utilize an analytical geometry approach based on the ray-casting algorithm.

For each recognized text block center $A(x, y)$, the system evaluates its inclusion within the expanded vehicle polygon P' by calculating the number of intersections k between a horizontal ray $R = \{(x', y) | x' \geq x\}$ and the polygon's edges E_i. This is defined by the following constraint:

$$\{y \in [y_1, y_2)\} \wedge ((x - x_1)(y_2 - y_1) - (x_2 - x_1)(y - y_1) \geq 0), \tag{8}$$

where

x_1, x_2, y_1, y_2 – coordinates of edge's E_i points.

The center point A is considered part of the vehicle if the intersection parity $k \equiv 1 (mod\, 2)$ is

satisfied. This geometric verification ensures that background text and advertisements are excluded from the ALPR pipeline.

3.3. Linguistic model and formal grammar

The raw OCR output undergoes a multi-stage linguistic transformation to align with the Ukrainian state standard for license plates.

We define a normalization function f that eliminates noise characters (spaces, hyphens, and punctuation) to produce a clean character stream:

$$f(S)=S\{\text{‘ ’},\text{‘}-\text{‘},\text{‘.’}\}, \tag{9}$$

where

S – input string.

Also, we define a length limitation function λ to filter ad or car dealer text:

$$\lambda(S)=\begin{cases} S, & if\ |f(S)|\le 14 \\ \varnothing, & if\ |f(S)|>14 \end{cases}. \tag{10}$$

To validate the reconstructed string, we introduce a formal grammar P_{ANPR} representing the regular language of license plate masks:

$$P_{ANPR}=\Sigma^{*}\times L_{ukr}^{2}\times D^{4}\times L_{ukr}^{2}\times\Sigma^{*}, \tag{11}$$

where

Σ^{*} - any number (including zero) of characters before and after the plate number;

$L_{ukr}=\{\text{all letters}\}\cup\{0,1,8\}$ - letters and digits that look like letters;

$D=\{0,1,\ldots,9\}\cup\{O,I,B,J\}$ - digits and letters that look like digits.

String is considered a valid number if $\lambda(f(S))\in P_{ANPR}$.

The system resolves optical character illusions (e.g., confusing "O" with "0") using a system of algebraic substitutions ϕ and τ applied to specific segments of the plate (region, digits, and series zones):

$$\phi_{L\to D}:\{O\to 0,\ I\to 1,\ B\to 8,\ J\to 1\} \tag{12}$$

$$\phi_{D\to L}:\{1\to I,\ 0\to O,\ 8\to B\} \tag{13}$$

$$\tau_{Eng\to Ukr}:\{A\to A,\ B\to B,\ C\to C,\ \ldots,\ X\to X\}. \tag{14}$$

Let $M=M_1M_2M_3M_4M_5M_6M_7M_8$ be the valid recognized plate number string. The correction algorithm consists of applying these mappings piecewise to the corresponding segments of the string:

1. Region zone (1-2 positions) – $M'_{1,2}=\phi_{D\to L}(M_{1,2})$
2. Digits zone (3-6 positions) – $M'_{3,6}=\phi_{L\to D}(M_{3,6})$
3. Series zone (7-8 positions) - $M'_{7,8}=\phi_{D\to L}(M_{7,8})$

After that, the final transliteration is defined as:

$$ANPR_{final}=\tau_{Eng\to Ukr}(M'_{1,2})\oplus M'_{3,6}\oplus\tau_{Eng\to Ukr}(M'_{7,8}), \tag{15}$$

where

$\oplus$ - concatenation of strings.

This formal approach ensures that the final identification result satisfies the linguistic and

structural constraints of the target domain.

3.4. Software realization method

The described mathematical model is implemented via a distributed microservice architecture designed for hardware-independent deployment.

The cyber-physical system is partitioned into three autonomous modules to minimize local computational load. The interaction protocol between modules is illustrated in Figure 1.

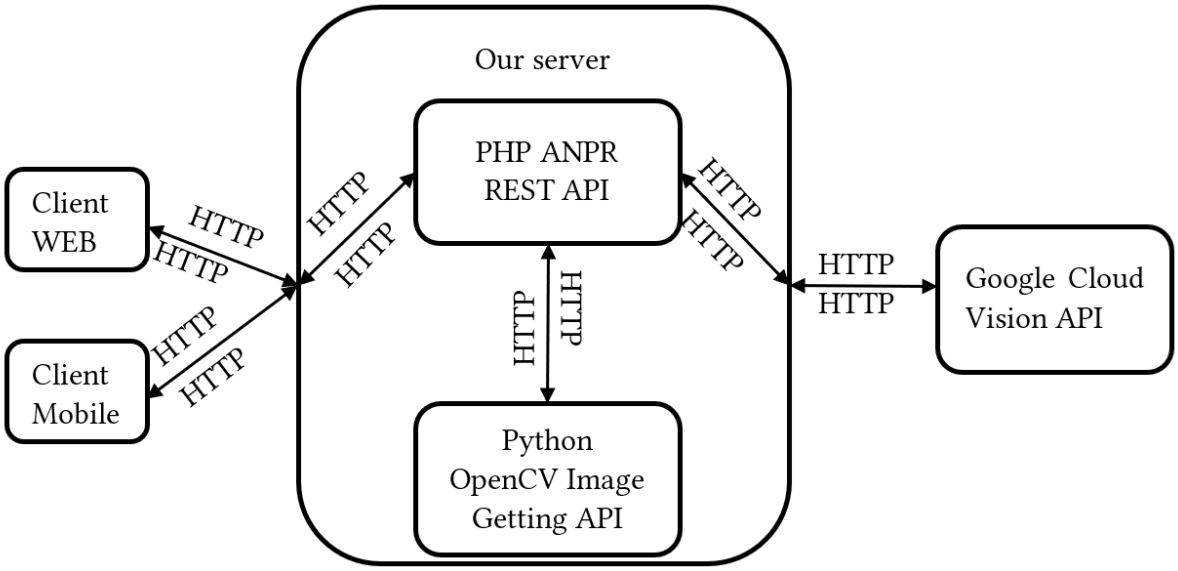

Figure 1: Software architecture of the proposed Smart Parking cyber-physical system.

The image getting API is developed in Python using the Flask microframework and the OpenCV computer vision library. This service manages the RTSP connections with network surveillance cameras. To eliminate transmission artifacts inherent to RTSP stream initialization, the module implements an algorithm based on pixel intensity analysis. For each i-th sequential frame, the arithmetic mean of pixel intensities ($Means_i$) is computed. The criterion for proper frame formation is defined by the difference between consecutive frames exceeding an empirical threshold of 15% ($\alpha = 0.15$):

$$\left|\text{Means}_i - \text{Means}_{i-1}\right| > \alpha \times \text{Means}_{i-1}. \quad (16)$$

Upon satisfying this condition, the loop terminates immediately, capturing the stabilized frame. If the stream fails to stabilize within a 10-frame window, a fallback is applied based on an upper intensity bound ($Means < 126$), filtering out saturated overexposures and noise. The validated frame is then encoded as a PNG with a compression coefficient of 0 (*cv2.IMWRITE_PNG_COMPRESSION = 0*) to ensure zero-loss pixel data retention for subsequent processing. Finally, the image is serialized via Base64 and dispatched within a JSON payload to the recognition core.

To manage the license plate recognition workflow and data streams between the edge buffer, cloud vision services, and the geometric post-processing core, a specialized controller was developed within the Laravel Backend, which also provides a REST API. This component implements formalized mathematical models and methods. The data processing pipeline consists of the following consecutive stages:

1. Cloud Vision API. Upon receiving a payload containing the serialized image, the controller initializes an ImageAnnotatorClient. A remote asynchronous request is sent to the Google Cloud Vision API, specifying text detection as the target feature. The cloud service returns a

raw hierarchical JSON response containing detected textual strings along with their respective bounding poly vertices.

2. Coordinate conversion. As was described in (1) and (2) the coordinates need to be converted. The controller implements the transformation routine normalizedVerticesToArray to get real pixel values of vertices.
3. Geometric filtering and vehicle-to-plate mapping. The raw OCR output contains not only vehicle plate text blocks (e.g., street signs, environmental noise, or vehicle decals). Firstly, the controller finds all car objects and extends their bounding boxes (4).Then, it finds text within bounding boxes by executing the spatial inclusion criteria (8). At the end, number plate candidates follow the grammar filter (9).
4. Response serialization. The validated and aggregated data, consisting of filtered license plate strings and the raw OCR text transcript, are mapped into a structured array. To prevent redundant processing of the same frame entities, processed text sentences are flagged. Finally, the controller serializes the data into a standardized JSON response and sends it to the client application with an HTTP 200 OK status.

To ensure accessibility and usability across different operational environments, the system provides two distinct client-side applications: a lightweight web-based interface for desktop access and a cross-platform mobile application. Both clients communicate with the backend via a RESTful API and are designed to render recognition data.

Figure 2 illustrates the data sequential workflow through these modules.

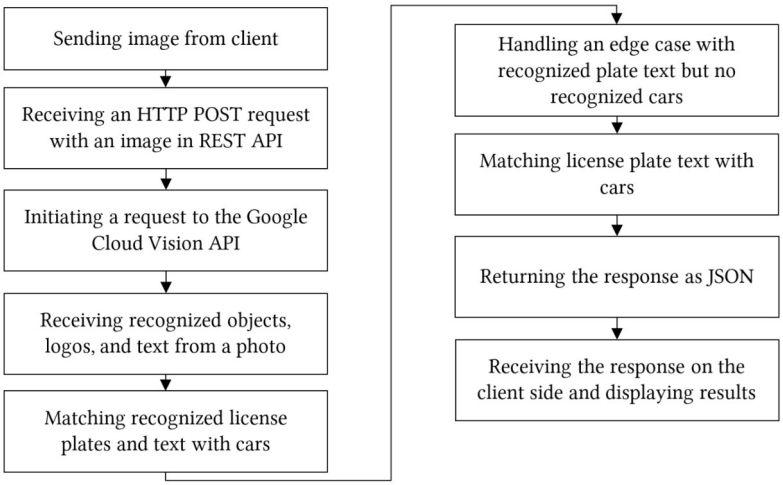

Figure 2: Algorithmic workflow of the data in the system.

4. Results and discussions

4.1. Algorithmic optimizations and fallback mechanisms

During initial testing of the Google Cloud Vision API, it was found that general-purpose object detection often misses the license plate object itself, even though the text itself is perfectly recognized. To solve this, two levels of algorithmic fallback mechanisms were implemented in the PHP ANPR REST API:

- Local fallback mechanism. If the system cannot form a valid license plate from the text inside the license plate bounding box, the algorithm automatically expands the search area to include all text within the larger car bounding box.

- Global fallback mechanism. In edge cases where the object detector entirely misses the vehicle but the OCR successfully reads the text, the algorithm scans all remaining unstructured text on the image.

If a text string matches the strict regular expression of a Ukrainian license plate, it is saved as an unidentified vehicle. Furthermore, data normalization was applied before the regular expression check. By programmatically removing spaces, dots, and hyphens from the raw OCR string, the system ensures that the validation logic works with a clean stream of characters, significantly reducing false negatives.

4.2. Dataset

To quantitatively evaluate the proposed architecture, an experimental study was conducted using a custom dataset of vehicle images. It consisted of the Roboflow car plates Ukraine v6 dataset [28, 29] and manually taken photos of various cars in different locations. The dataset includes 215 photos with 245 license plate states. Although the Roboflow dataset contained associations, they weren't describing text on license plates, but only locations. Therefore, we designed a tool for manual photo labeling with metric evaluation. It allows us to extend the dataset with manually taken photos.

Figure 3 illustrates the tool interface.

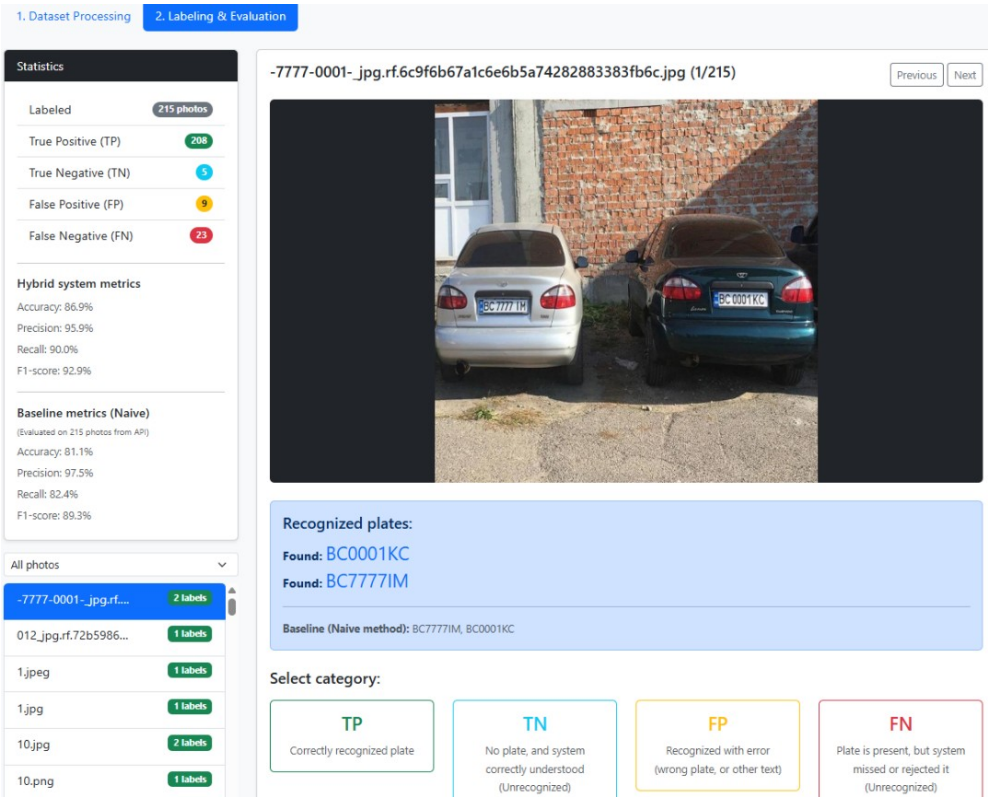

Figure 3: The interface of the evaluation tool.

This tool is used to confirm recognized plates within 4 basic categories of classification, filtering results for quick error analysis, and comparing the baseline method with the hybrid.

4.3. Experimental setup and evaluation metrics

The performance was evaluated considering the specific business logic of a smart parking barrier, where the system must strictly prevent unauthorized entry. The evaluation metrics were defined as follows:

- True Positive (TP) – the system correctly recognized the full license plate, allowing the authorized car to enter.
- True Negative (TN) – the image contained no car or plate, and the system correctly ignored it.
- False Positive (FP) – the system recognized a wrong plate or treated random text as a valid plate, creating a security risk.
- False Negative (FN) – the system failed to recognize a visible plate, preventing an authorized client from entering.

The system's efficiency was calculated using standard classification metrics (Precision, Recall, Accuracy, and F1-score). The quantitative results of the experiment are presented in Table 1.

Table 1
Evaluation metrics of the proposed hybrid ALPR system

Metric	Value (%)
Accuracy	86.9%
Precision	95.9%
Recall	90.0%
F1-score	92.9%

As seen in Table 1, the proposed geometric post-processing layer achieved a remarkably high precision of 96.4%. In the context of access control systems, this high precision is a critical success factor, as it indicates a low False Positive rate, ensuring that unauthorized vehicles are not granted access. The F1-score of 90.7% demonstrates a strong overall balance of the mathematical model. The slightly lower recall (85.7%) reflects the nature of the heuristic filtering, which intentionally discards ambiguous data to prioritize security.

4.4. Error analysis

Despite the high accuracy achieved by the geometric post-processing layer, a detailed error analysis was conducted to identify the remaining limitations of the Google Cloud Vision API. The primary causes of False Negatives, which affected the recall metric, were:

- Vehicles with personalized or vanity plates (e.g., random words) were intentionally discarded by the system's regular expression, which strictly expects the standard format.
- Partial overlap by dirt or physical frames made it impossible for the OCR to read the full string. In some cases, characters located at the very edge of the plate (e.g., the letter 'I') blended with the physical border, causing the string length to shrink and fail the validation mask.

The primary causes of False Positives were directly related to the optical illusions of the cloud OCR engine:

- The cloud model occasionally confused visually similar characters, recognizing 'E' as 'F' or 'B' as 'U'.
- Lighting reflections, shadows, or license plate frames were sometimes misinterpreted by the OCR as extra letters at the beginning or end of the plate.
- Pixel degradation for vehicles located far from the camera led to digit misinterpretation.

A specific source of False Positives was the small text printed on the plastic frames of license plates (e.g., car dealership names or phone numbers). Because the global fallback mechanism aggressively scans all available text to prevent data loss, it occasionally reconstructed a sequence of letters and numbers from these dealer ads that accidentally matched the ALPR validation mask.

However, the application of mapping dictionaries (e.g., translating identical English letters to Ukrainian and fixing common number-letter swaps) significantly reduced these OCR illusions.

To further mitigate this, future versions of the system can introduce stricter heuristic cropping based purely on the OBJECT_LOCALIZATION coordinates, dynamically ignoring text blocks that fall on the extreme outer edges of the vehicle's bounding box.

4.5. Comparative analysis

To demonstrate the scientific value of the proposed post-processing architecture, a comparative analysis was conducted against a naive baseline implementation.

The baseline model represents a standard integration of the Google Cloud Vision API: it extracts the recognized text, applies basic string normalization (removing spaces and hyphens), and strictly filters the string using the Ukrainian ALPR regular expression mask. The proposed hybrid system utilizes the full pipeline: ray-casting geometric matching, the context-aware transliteration dictionary, and the local/global spatial fallback mechanisms.

Table 2
Comparison of naive cloud implementation vs. proposed hybrid system

Metric	Naïve impl. value (%)	Hybrid impl. value (%)
Accuracy	81.1%	86.9%
Precision	97.5%	95.9%
Recall	82.4%	90.0%
F1-score	89.3%	92.9%

As shown in Table 2, the results illustrate a precision-recall trade-off. The naive baseline acts as a strict filter, achieving a high precision (97.5%) but suffering from a significantly lower recall (82.4%). This happens because the baseline strictly rejects plates with minor OCR illusions (e.g., confusing the letter 'O' with the number '0', or 'I' with '1').

By introducing the dictionary mapping and spatial fallback mechanisms, the proposed system recovered these fragmented or misread plates, boosting the recall to 90.0% and improving the overall F1-score to 92.7%. The slight decrease in precision in the proposed system is a controlled side-effect of the global fallback mechanism, which occasionally misinterprets background text as a valid license plate. Overall, this confirms that the proposed heuristic layer provides a much more balanced and practical solution for access control systems, maximizing the number of correctly admitted vehicles.

4.6. System implementation and user interfaces

To evaluate the practical applicability of the proposed cyber-physical system in real-world smart parking environments, two distinct client-side applications were implemented. These interfaces act as lightweight presentation layers that render the serialized JSON data generated by the core geometric post-processing backend.

The primary monitoring interface for parking security personnel is implemented as a web-based dashboard using the Laravel Blade templating engine and Bootstrap framework. Utilizing a server-

side rendering architecture ensures minimal client-side computational overhead, making the dashboard accessible even on low-end hardware. The web interface is designed to dynamically parse the JSON response from the REST API and insert this data into the DOM. The dashboard categorizes the output into three distinct visual blocks:

- Raw OCR payload displays the fragmented, unformatted text strings exactly as returned by the Google Cloud Vision API.
- Verified number plate string displays the final, sanitized license plate sequence that successfully passed both the spatial inclusion criteria and the linguistic formal grammar validation.
- Auxiliary objects, such as detected vehicle brand logos.

Additionally, the dashboard provides dynamically generated links to the public 'CarPlates' service. This dashboard is illustrated in Figure 4.

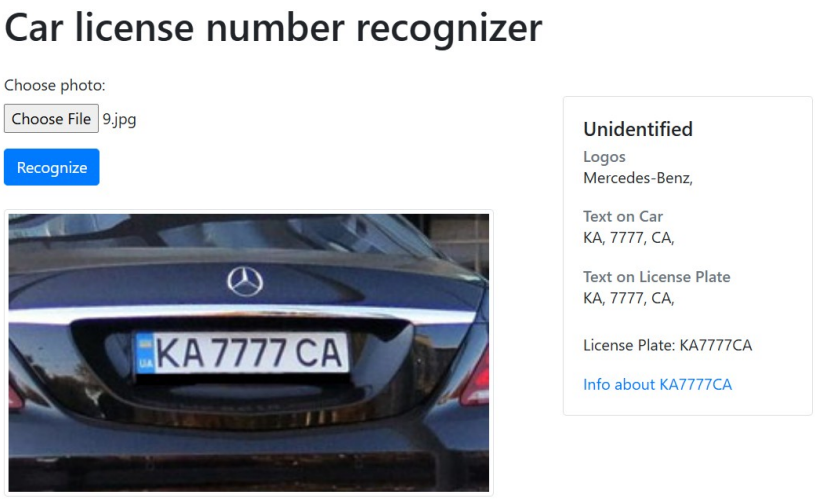

Figure 4: The interface of the web dashboard.

For end-users, a cross-platform mobile application was developed using the React Native framework, enabling a unified JavaScript codebase to compile into native components for both Android and iOS ecosystems.

It features a capture mechanism that can use the device's native camera hardware or media gallery. To handle the latency of the cloud API and the local post-processing execution, the application implements asynchronous state updates. During the HTTP request lifecycle, the UI remains responsive, displaying real-time loading indicators. Upon receiving the HTTP 200 OK response containing the data payload, the application renders the authorized license plate string. Figure 5 illustrates the mobile client interface.

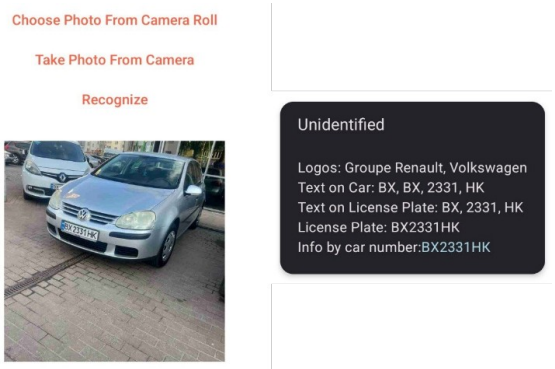

Figure 5: Mobile client interface developed with React Native (the image is divided in 2 columns).

5. Conclusions

This research presented a hybrid cyber-physical architecture for automated vehicle identification in smart parking environments. By combining general-purpose Cloud Vision APIs with a local heuristic post-processing layer based on ray-casting geometry and formal linguistic grammars, the system effectively overcomes the limitations of resource-constrained edge hardware.

The experimental evaluation on a custom dataset of 215 images confirmed the high efficiency of the proposed methodology. The system achieved a classification accuracy of 86.9%, a precision of 95.9%, and a recall of 90.0%, outperforming naive cloud-based implementations. The comparative analysis demonstrated that the developed local and global fallback mechanisms successfully mitigate spatial fragmentation and optical illusions in general-purpose OCR models, recovering a portion of vehicle plates.

Future work will focus on optimizing the heuristic cropping algorithms to minimize False Positives caused by background noise, such as text on license plate frames. Additionally, the integration of lightweight temporal stabilization for video streams will be explored to further enhance the system's robustness.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] J. Kim, I. Jeong, J. Jung, J. Cho, Resource-Efficient Design and Implementation of Real-Time Parking Monitoring System with Edge Device, Sensors 25.7 (2025) 2181. doi:10.3390/s25072181.

[2] J. Shashirangana, H. Padmasiri, D. Meedeniya, C. Perera, Automated License Plate Recognition: A Survey on Methods and Techniques, IEEE Access (2020) 1. doi:10.1109/access.2020.3047929.

[3] D. Tran-Anh, K. L. Tran, H.-N. Vu, License Plate Recognition Based On Multi-Angle View Model, 2023. URL: https://arxiv.org/abs/2309.12972.

[4] R. A. Tavares, Comparison of Image Preprocessing Techniques for Vehicle License Plate Recognition Using OCR: Performance and Accuracy Evaluation, 2024. URL: https://arxiv.org/abs/2410.13622.

[5] S. Swati, S. Dinesh Kawa, S. Kamble, D. Desai, P. Himanshu Karelia, P. Engineer, Efficient Deep Learning based License Plate Recognition for Smart Cities, ELCVIA Electron. Lett. Comput. Vis. Image Anal. 23.2 (2024) 50–64. doi:10.5565/rev/elcvia.1917.

[6] B. G V, Object Detection using OpenCV and Deep Learning, Int. J. Res. Appl. Sci. Eng. Technol. 9.VI (2021) 3920–3923. doi:10.22214/ijraset.2021.35880.

[7] N. AlDahoul, M. J. T. Tan, R. R. Tera, H. A. Karim, C. H. Lim, M. K. Mishra, Y. Zaki, Advancing vehicle plate recognition: multitasking visual language models with VehiclePaliGemma, 2024. URL: https://arxiv.org/abs/2309.12972.

[8] V. N. Ribeiro, N. S. Hirata, Efficient Video-Based ALPR System Using YOLO and Visual Rhythm, 2025. URL: https://arxiv.org/abs/2501.02270.

[9] S. Lee, G. Park, A Comparative Study of OCR Architectures for Korean License Plate Recognition: CNN–RNN-Based Models and MobileNetV3–Transformer-Based Models, Sensors 26.4 (2026) 1208. doi:10.3390/s26041208.

[10] A. N. Kumar, S. Aparna, Intelligent Optical Character Recognition Through CNN-LSTM Fusion With Dictionary Validation and Error Correction, J. Theor. Appl. Inf. Technol. 104.3 (2026). doi:10.5281/zenodo.18666690.

[11] A. Firasanti, T. E. Ramadhani, M. A. Bakri, E. A. Zaki Hamidi, License Plate Detection Using OCR Method with Raspberry Pi, in: 2021 15th International Conference on Telecommunication Systems, Services, and Applications (TSSA), IEEE, 2021. doi:10.1109/tssa52866.2021.9768252.
[12] K. Kim, J. Lee, I. Jeong, J. Jung, J. Cho, Real-Time Parking Space Management System Based on a Low-Power Embedded Platform, Sensors 25.22 (2025) 7009. doi:10.3390/s25227009.
[13] D. K. Alqahtani, M. A. Cheema, M. A. Rodriguez, A. N. Toosi, A Comprehensive Evaluation of Deep Learning Object Detection Models on Heterogeneous Edge Devices, 2026. URL: https://arxiv.org/abs/2409.16808.
[14] H. Padmasiri, J. Shashirangana, D. Meedeniya, O. Rana, C. Perera, Automated License Plate Recognition for Resource-Constrained Environments, Sensors 22.4 (2022) 1434. doi:10.3390/s22041434.
[15] P. Savvidis, G. A. Papakostas, Edge Computing for Computer Vision in IoT: Feasibility and Directions, EAI Endorsed Trans. Things 11 (2025). doi:10.4108/eetiot.9404.
[16] K. Thammarak, P. Kongkla, Y. Sirisathitkul, S. Intakosum, Comparative analysis of Tesseract and Google Cloud Vision for Thai vehicle registration certificate, Int. J. Electr. Comput. Eng. (IJECE) 12.2 (2022) 1849. doi:10.11591/ijece.v12i2.pp1849-1858.
[17] K. Sivakoti, Neural Sentinel: Unified Vision Language Model (VLM) for License Plate Recognition with Human-in-the-Loop Continual Learning, 2026. URL: https://arxiv.org/abs/2602.07051.
[18] H. Gong, H. Liu, LP-LLM: End-to-End Real-World Degraded License Plate Text Recognition via Large Multimodal Models, 2026. URL: https://arxiv.org/abs/2601.09116.
[19] Y. Sun, D. Zhou, C. Lin, C. He, W. Ouyang, H.-S. Zhong, LOCR: Location-Guided Transformer for Optical Character Recognition, 2024. URL: https://arxiv.org/abs/2403.02127.
[20] Y. Li, G. Yang, H. Liu, B. Wang, C. Zhang, dots. ocr: Multilingual document layout parsing in a single vision-language model, 2025. URL: https://arxiv.org/abs/2512.02498.
[21] T. T. H. Nguyen, A. Jatowt, M. Coustaty, A. Doucet, Survey of Post-OCR Processing Approaches, ACM Comput. Surv. 54.6 (2021) 1–37. doi:10.1145/3453476.
[22] K. Thammarak, Y. Sirisathitkul, P. Kongkla, S. Intakosum, Automated Data Digitization System for Vehicle Registration Certificates Using Google Cloud Vision API, Civ. Eng. J. 8.7 (2022) 1447–1458. doi:10.28991/cej-2022-08-07-09.
[23] A. N. Kumar, S. Aparna, Intelligent Optical Character Recognition Through CNN-LSTM Fusion with Dictionary Validation and Error Correction, J. Theor. Appl. Inf. Technol. 104.3 (2026). doi:10.5281/zenodo.18666690.
[24] T. N. Nguyen, J. C. Burie, T. L. Le, A.-V. Schweyer, Text line segmentation approach combining deep learning model and traditional image processing techniques - application to transliteration of Cham manuscripts, Multimed. Tools Appl. (2025). doi:10.1007/s11042-025-20602-x.
[25] R. Wang, Y. Fujii, A. C. Popat, Post-OCR Paragraph Recognition by Graph Convolutional Networks, in: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), IEEE, 2022. doi:10.1109/wacv51458.2022.00259.
[26] J. P. Naiman, M. G. Cosillo, P. K. G. Williams, A. Goodman, Large Synthetic Data from the arXiv for OCR Post Correction of Historic Scientific Articles, 2023. URL: https://arxiv.org/abs/2309.11549.
[27] B. G V, Object Detection using OpenCV and Deep Learning, Int. J. Res. Appl. Sci. Eng. Technol. 9.VI (2021) 3920–3923. doi:10.22214/ijraset.2021.35880.
[28] Car plates Ukraine Computer Vision Dataset. URL: https://universe.roboflow.com/detect-special-cars/car-plates-ukraine.
[29] V. Talapchuk , E. Zaitseva . Mobile-oriented cyber-physical system for food allergen detection based on machine learning and image analysis. *Computer Systems and Information Technologies* 1 (2025) 29–35. https://doi.org/10.31891/csit-2025-1-3