Parking Spot Detector and Classifier

Anshul Aggarwal
Department of Computer Science
University Of Galway
Galway, Ireland
a.aggarwal2@universityofgalway.ie

***Abstract*—The University of Galway aims to enhance its parking management system for permit holders by leveraging aerial drone footage to monitor parking space availability in real-time. Efficient parking management is critical for large campuses, where demand for parking spaces often exceeds supply during peak hours. Traditional manual monitoring is labour-intensive, prone to errors, and unable to provide real-time updates. The proposed system will identify and update the status of parking spaces whether occupied or vacant using classical image processing techniques. Parking spaces are classified as occupied (marked red) or vacant (marked green) based on overlap between detected cars and parking grids. It handles diverse lighting conditions, and complex scenarios using tailored pre-processing steps and parameter adjustments. This classical image processing-based parking detection system is an efficient, cost-effective, and scalable solution for managing parking.**

***Index Terms*—parking spaces, real-time, image processing, occupied, vacant, labour-intensive, cost-effective, scalable solution**

I. Introduction

Parking management is a significant challenge for large institutions, where a limited number of parking spaces must accommodate a high volume of vehicles. To address this issue, an intelligent parking management system that efficiently monitors and updates parking availability in real-time needs to be implemented.

The proposed solution makes use of classical image processing techniques to detect and classify parking spaces as either occupied or vacant. This system aims to provide a cost-effective and scalable alternative to traditional manual monitoring, which is labour-intensive, error-prone, and unable to deliver real-time updates. By automating the detection process and integrating it into a centralized system, it can significantly enhance user convenience, reduce traffic congestion, and optimize parking space utilization.

This report outlines the design and implementation of the parking detection system, detailing the preprocessing steps, feature detection methods, and classification algorithms used. The system is designed to handle diverse lighting conditions and complex scenarios, making it a robust solution for real-world applications. The following sections elaborate on the methods, results, and impact of this system.

II. Methodology

A. Image Preprocessing

The process is broken into two main components. (1) Detection of Parking Grids and Areas, (2) Detection of Cars in the Parking Areas.

Both components involved a series of preprocessing steps, morphological operations, and detection techniques, ensuring accurate and reliable results.

For both the components, the first step includes **Conversion to Grayscale,** The `convert_grey` function converts input images to grayscale using OpenCV's cv2.cvtColor function. Grayscale images simplify computational complexity by reducing image information to intensity values while retaining structural features necessary for edge detection. [1]

1) Detection of Parking Grids and Areas:

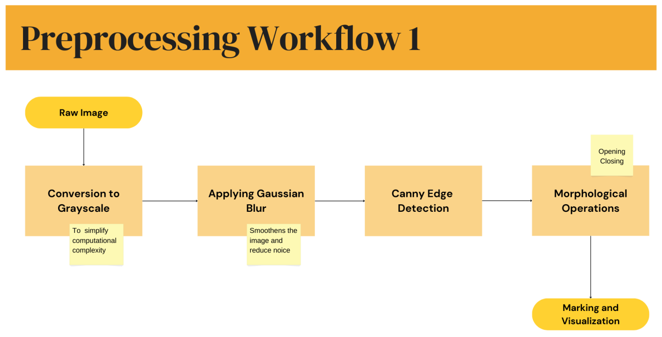

Fig. 1. Workflow for preprocessing for detecting parking grids.

- **Gaussian Blur:** Gaussian blur is applied using the `blur` function to smoothen the images. For specific challenging images, Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied to enhance contrast before blurring. Smoothing helps reduce noise, making edges and contours more distinguishable for subsequent processing steps. CLAHE is applied to images with poor contrast to improve the visibility of fine details.
- **Canny Edge Detection:** Edges are detected using the cv2.Canny function, followed by Otsu's thresholding for binarization in the `detect_canny_edges` function. Edges represent the boundaries of parking grids. Canny edge detection is robust for extracting such boundaries [2], while Otsu's thresholding ensures a clean binary representation.
- **Morphological Operations** The `morph_lines` function applies opening and closing operations using rectangular kernels.

- **Opening:** Removes small, irrelevant noise, ensuring that only significant parking grid lines remain. This is particularly helpful in clearing stray pixels or small objects that are not part of the parking grid. [1]
- **Closing:** Bridges small gaps or breaks in the detected grid lines, ensuring that the parking grid is continuous and complete. This is crucial for creating clearly defined parking boundaries. [1]

Morphological operations are crucial for refining grid lines and making them suitable for contour and line detection. Refer to Fig. 2. for output.

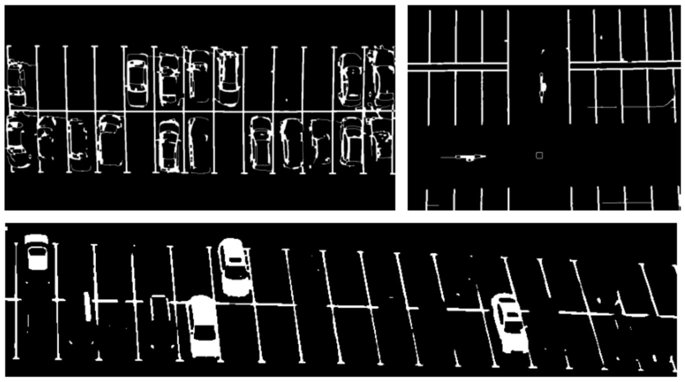
Fig. 2. Output after preprocessing for parking lines.

2) Detection of Cars in the Parking Areas:

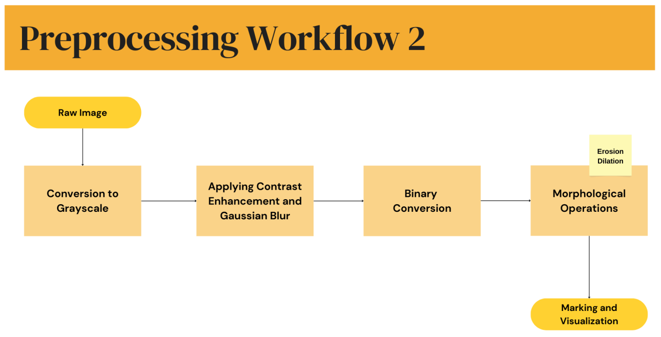

Fig. 3. Workflow for preprocessing for detecting cars.

- **Contrast Enhancement and Gaussian Blur:** Contrast enhancement (CLAHE) is combined with Gaussian blur in the `contrast_blur` function. Enhancing contrast ensures cars are visually distinct from the background, aiding binary conversion and subsequent detection.
- **Binary Conversion:** Otsu's thresholding is used in the `convert_binary` function to binarize the contrast-enhanced images. Binary images simplify object detection by representing cars as distinct regions of intensity.
- **Morphological Operations:** The `morph_cars` function applies erosion and dilation which plays a crucial role in refining the binary images for car detection.
  - **Erosion:** Removes small unwanted objects (noise) in the image by eroding away the boundaries of foreground objects. This is particularly useful as it eliminates stray pixels that might be falsely identified as part of an object. [1]
  - **Dilation:** Expands the boundaries of objects, filling small gaps or holes within an object, making the detected shapes more complete. It helps reconstruct the car shapes after erosion. [1]

Selective eroding and dilation are employed for challenging images to better isolate cars. These operations are tailored to emphasize the presence of cars while minimizing irrelevant artefacts. Refer to Fig. 4. for output.

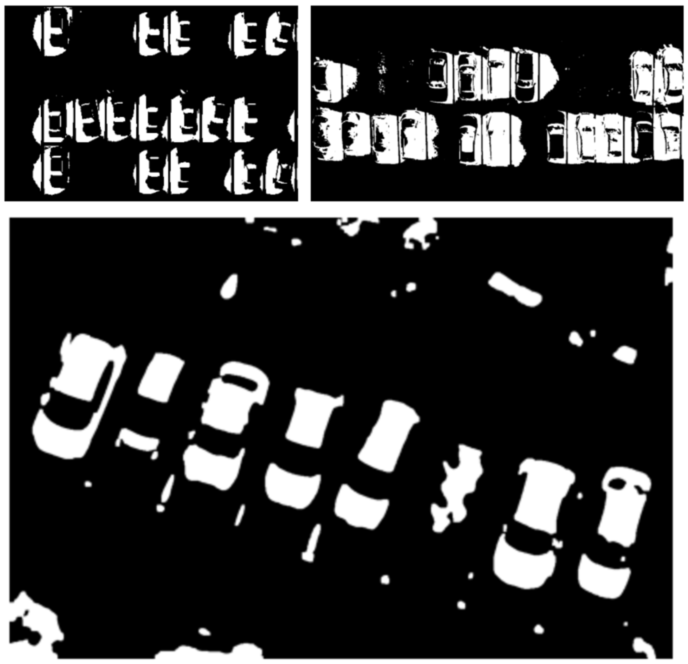
Fig. 4. Output after preprocessing for cars.

B. Marking and Visualization

This includes the final steps in the pipeline. They are responsible for identifying and visually marking parking spots and cars on the original images, enabling the system to classify spaces as either vacant or occupied. The proposed solution makes use of two functions. One of them focuses on **identifying parking grid lines and areas from the processed images and marking them as green** on the original image. The other is responsible for **detecting cars in the parking spaces and marking them within red rectangles**.

1) Outlining Parking Area: To detect and mark the parking grid lines, **Hough Line Transformation** is used. It detects straight lines in the parking grid. Detected lines are drawn onto the original image using "cv2.line". This step helps in outlining parking boundaries clearly.

Next, parking regions are identified by finding continuous boundaries in the processed image. The **contours** are filtered by size to focus on significant areas, eliminating irrelevant small regions. For valid contours, rectangular bounding boxes are drawn (green for vacant spaces) onto the original image.

This function identifies potential parking areas (grids) based on grid lines and enclosed regions. It provides a visual representation of the parking layout, making it easier to distinguish between spaces and surrounding areas.

*2) **Outlining Cars**:* Their contours are detected to detect and mark the cars. Contours are filtered by size to focus on regions that match the expected dimensions of a car (w > lower and h > lower and w < upper and h < upper). For valid contours, red bounding boxes are drawn around cars to indicate occupancy.

This function overlays the detected car regions onto the original image. Highlighting cars in red, provides a visual cue for occupied spaces, distinguishing them from the vacant spots identified earlier. Refer to Fig. 5. for output.

III. Results

A. Easy Set:

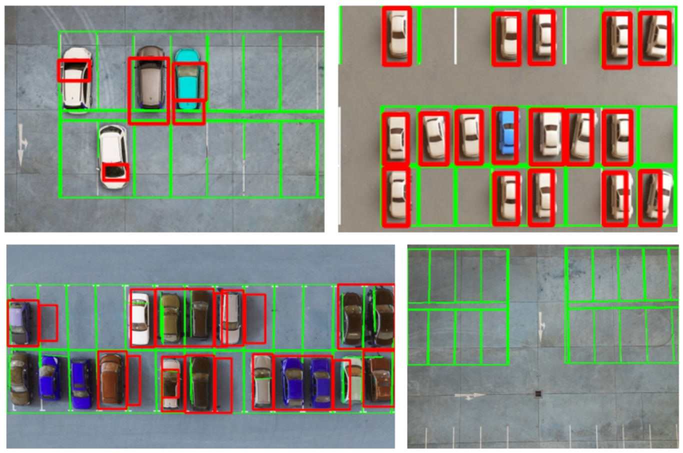
Fig. 5. Final Easy set Outputs after marking.

B. Medium and Hard Set:

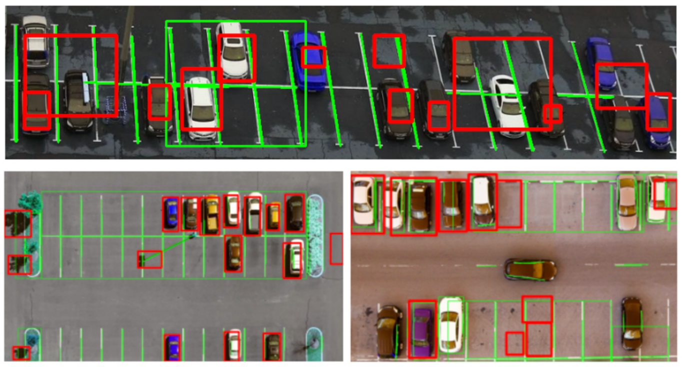
Fig. 6. Final Medium set Outputs after marking.

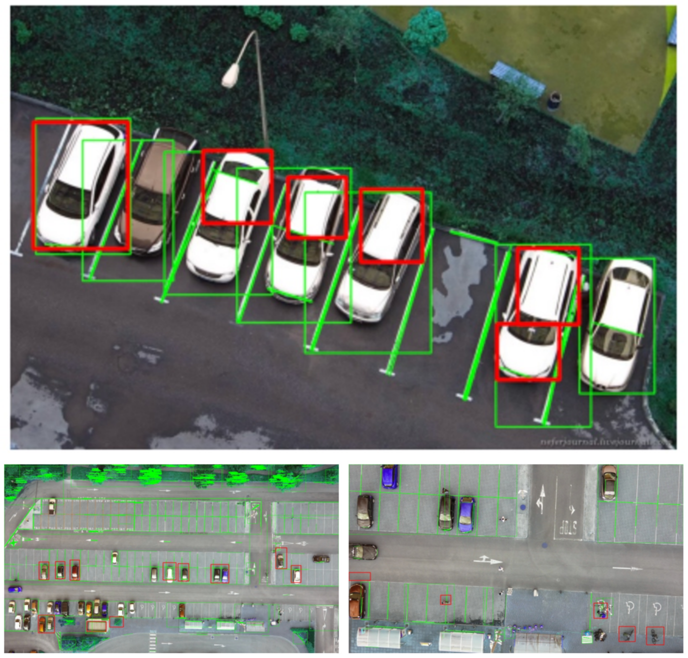
Fig. 7. Final Hard set Outputs after marking.

IV. Discussion

A. *Challenges*

The development of the parking detection system faced several challenges, reflecting the complexities of real-world image processing tasks. These include:

*1) **Varying Lighting Conditions**:* Differences in lighting, such as shadows, overexposure, or low light, affected image clarity and required additional preprocessing, such as histogram equalization, to enhance contrast and improve detection accuracy.

*2) **Parameter Tuning**:* The classical image processing methods heavily relied on fine-tuned parameters, such as kernel sizes for morphological operations, thresholds for edge detection, and Hough Line Transform settings. Finding the optimal values was time-consuming and required iterative testing.

*3) **Handling Overlaps and Noise**:* Overlapping cars, faded parking lines, or occlusions posed challenges in accurately detecting grid lines and distinguishing between occupied and empty spaces. Noise in images further complicated the detection process.

*4) **Limited Robustness Compared to Deep Learning Models**:* While classical image processing methods are efficient and cost-effective, they lack the adaptability of deep learning-based approaches in handling more dynamic or ambiguous scenarios, such as partially visible vehicles or unconventional parking spaces.

B. *Analysis of Results*

The system was tested on three datasets of varying complexity: easy, medium, and hard. The following metrics were used to evaluate the performance:

- Ground Truth: The number of parking spaces in the dataset images as per manual annotation.
- True Positives: Parking spaces accurately identified as vacant or occupied.
- False Positives: Areas incorrectly marked.
- Missed Detections: Actual parking spaces not detected by the system.

Dataset Complexity	***Ground Truth***	***True Positives***	***False Positives***	***Missed***
Easy	80	73	3	4
Medium	96	65	15	16
Hard	227	124	53	50

Accuracy:

$$\frac{TruePositive}{TruePositive + FalsePositive + Missed} \tag{1}$$

So, the accuracy scores for each dataset complexity are:

Easy - **91.25%**
Medium - **67.8%**
Hard - **54.7%**

1) Code Optimization: The system was designed with efficiency in mind, optimizing image processing steps and ensuring compatibility with embedded systems.

- Testing revealed that down-sampling images by a factor of 2 reduced processing time by approximately 30
- Steps such as Gaussian blur and morphological operations were batched to minimize redundant computations.

2) Big O Analysis: The computational complexity of the pipeline is primarily determined by the following steps:

- Morphological Operations: $O(n^2)$, where n is the kernel size. [2]
- Edge Detection (Canny): $O(m * n)$, where m and n are image dimensions.[2]
- Hough Line Transform: $O(k * \rho * \theta)$, where k is the number of edge pixels, and ρ and θ are resolution parameters. [2]
- Overall complexity: $O(m * n)$, since image dimensions dominate.

Aspect	**Classical Image-Processing**	**Deep Learning**
Accuracy	Moderate, sensitive to tuning	High, robust to varying scenarios
Computational Cost	Low	High, requires GPU for training
Development Time	Short	Long, requires training and annotation
Adaptability	Limited	Easily adaptable to new datasets

3) Comparison with Deep Learning Approaches: The classical approach provides a lightweight, cost-effective solution, especially suitable for embedded systems and low-resource environments. However, it is less robust than deep learning models in handling complex scenarios and varied conditions.[3]

V. Conclusion

The developed pipeline demonstrates a methodical and scalable approach to detecting parking spots and cars using a combination of image-processing techniques. The modularity of functions allows easy adaptation to other image datasets.

The results highlight the effectiveness of combining edge detection, morphological operations, and contrast enhancement to handle complex scenarios and noise. The solution is cost-effective, scalable, and provides a practical approach to enhancing parking management systems. Its ability to process diverse image formats and handle real-time requirements ensures adaptability to real-world applications, such as parking space management.

While the system achieves its goals using classical methods, limitations such as dependence on parameter tuning and preprocessing for specific scenarios highlight potential areas for improvement. Future iterations can integrate **machine learning models** to enhance detection accuracy, and **process video streams** for continuous monitoring.

Overall, this system lays the groundwork for a more efficient and automated parking management system, significantly improving user convenience and operational efficiency.

References

[1] M. Roushdy, "Comparative Study of Edge Detection Algorithms Applying on the Grayscale Noisy Image Using Morphological Filter", Volume 6, Issue 4, December, 2006

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., "Introduction to Algorithms (3rd Edition)", 2009.

[3] M. Cheshfar, M. H. Maghami, P. Amiri, H. G. Garakani and L. Lavagno, "Comparative Survey of Embedded System Implementations of Convolutional Neural Networks in Autonomous Cars Applications," vol. 12, pp. 182410-182437, 2024, doi: 10.1109/ACCESS.2024.3510677.