Internet of Things 32 (2025) 101619

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

On-street parking space localization with deep learning using low-quality images from public cameras

José Ángel Morell *, Gabriel Luque, Enrique Alba

ITIS Software, University of Malaga, Spain

ARTICLE INFO

Keywords:
Computer vision
Cyber–physical systems
Deep learning
On-street parking space localization
Smart cities
Urban planning

ABSTRACT

The increasing demand for new services in cities leads to challenges that need an intelligent, holistic approach (smart cities). A crucial aspect of smart city planning is effectively managing public on-street parking spaces, influencing overall urban mobility and environmental sustainability. However, detecting these spaces is complex, often requiring costly cameras or sensors, and the variability in parking space sizes, depending on the vehicles parked, adds to the difficulty. Existing city cameras for traffic monitoring could be a solution, but their low image quality and frequent movements (taking images from different angles) make accurate detection challenging. We propose a novel method for locating on-street parking spaces using low-quality images from non-static public traffic cameras. This approach is dataset-independent, applicable to various cities, and employs deep-learning models pre-trained for tasks like vehicle detection, repurposing them for the novel task of identifying on-street public parking spaces. This method avoids specific retraining and intensive manual labeling. Tested in Malaga, Spain, the pipeline includes Extraction (sourcing images from Internet traffic cameras), Matching (recognizing common features between reference and new images for detecting camera movements), Preprocessing (comparing different denoising and image-enhancing techniques for improving model inference), Detection (using models like YOLOv8 and Detectron2 for vehicle detection), and Postprocessing (transforming perspectives to estimate real-world parking space coordinates and sizes). Experimental results demonstrate that our proposal achieves accurate parking space detection even in extreme light conditions and camera movements, providing a valuable new tool for parking management and urban planning.

1. Introduction

The enormous expansion of the global population and escalating urbanization are giving rise to larger cities facing intricate logistical challenges. Approximately 56% of the world's population, around 4.6 billion people, resides in urban areas, with projections anticipating this number to reach 6.7 billion by 2050 [1]. As cities expand, the need for efficient infrastructure and services becomes increasingly vital to maintain residents' quality of life and optimize urban resources. In this context, the concept of smart cities has emerged as a comprehensive solution to leverage cutting-edge technologies and improve various aspects of urban life, such as education, energy, mobility, and public services [2]. Among the essential factors to address in a smart city strategy is the management of parking spaces. The availability and accessibility of public parking areas significantly impact overall urban mobility and environmental sustainability.

* Corresponding author.
E-mail addresses: jamorell@uma.es (J.Á. Morell), gluque@uma.es (G. Luque), ealbat@uma.es (E. Alba).

https://doi.org/10.1016/j.iot.2025.101619
Received 30 December 2023; Received in revised form 11 March 2025; Accepted 19 April 2025
Available online 8 May 2025

Fig. 1. The proposed pipeline consists of five main stages: Extraction, Matching, Preprocessing, Detection, and Postprocessing.

The increasing demand for parking spaces in urban areas [3–5] requires efficient and accurate parking space location systems based on low-cost available resources. Sensors or cameras could be installed to achieve this goal, but it is prohibitively expensive. A more cost-effective approach involves using existing cameras in the city for another function, such as those already deployed for monitoring traffic and crime. Some studies propose using cameras for parking detection. However, they mainly focus on aerial images and rectangular, well-defined parking lots, where parking spaces are easily visible [6–11]—failing to address the challenges associated with detecting on-street parking spaces utilizing cameras already installed for traffic monitoring. These cameras capture low-quality images featuring camera movements, various angles, diverse weather conditions, and lighting variations. Overcoming these challenges, along with those related to detecting on-street parking of variable size (depending on the size of the parked vehicles), is crucial to repurpose them for parking detection, as they were not originally designed for this purpose.

In this work, we propose a novel dataset-independent method for accurately locating on-street parking spaces, which vary in size due to being filled with vehicles of different lengths. We utilize low-quality images from already installed moving public cameras for traffic monitoring. This method is easily deployable in various cities as it is independent of the dataset; it leverages pre-trained deep learning models for other tasks (e.g., image matching, vehicle detection, etc.), eliminating the need for manual labeling of large datasets, because retraining is unnecessary. Specifically, we present a complete pipeline from open data image extraction to leveraging pre-trained deep learning techniques and advanced image processing methods to address these challenges and accurately locate on-street parking spaces.

The proposed pipeline consists of five primary stages: Extraction, Matching, Preprocessing, Detection, and Postprocessing (see Fig. 1). In the Extraction stage, images are captured from openly available Internet traffic cameras. In the Matching stage, a pre-trained deep learning model matches correspondences between key points in the reference and new images, providing critical information for parking space localization even when camera movements and images are from different angles.

To compensate for the poor image quality from public cameras, the Preprocessing stage involves using denoising and image enhancement techniques. This work compares the suitability of various techniques, including bilateral filtering, median blur, fast non-local means denoising (colored), and super-resolution. These techniques enhance the effectiveness of the detection models.

The Detection stage utilizes information obtained from pre-trained deep-learning models designed for vehicle detection to accomplish the previously unknown task of localizing on-street parking spaces. These models allow us to achieve high precision in vehicle detection and differentiate them from other objects in the images without retraining. However, the challenges posed by low image quality, camera movements, and considerable distance from the parking spaces we want to locate require integrating these models with other techniques to determine the locations of on-street parking spaces accurately. We propose lowering the confidence threshold of the pre-trained deep learning models for vehicle detection, even when detecting the same vehicle multiple times. Subsequently, we sum the intersections of all detected vehicles and subtract the areas intersected by the vehicles from the parking line to identify the remaining portions of the parking line.

In the Postprocessing stage, the objects that do not intersect with the predefined parking area are discarded. Also, based on the known real-world coordinates of the selected key points of the reference image, a perspective transformation between the two images is performed, allowing the identification of the real-world coordinates of the detected parking spaces and accurately estimating their lengths, thereby facilitating the determination of the spacing between parked vehicles.

In summary, our main contribution is a novel dataset-independent method for accurately locating on-street parking spaces, which vary in size due to being filled with vehicles of different lengths. This method utilizes low-quality images under varying lighting conditions from pre-installed moving traffic cameras in the city. We propose a complete pipeline that includes image extraction, image preprocessing methods, and the integration of pre-trained deep-learning models for matching and detection. This pipeline is easily scalable and applicable in any city with publicly available cameras, as is common in most major cities. We validate our method using extracted open data images from Malaga (Spain) public traffic cameras. The experimental results confirm the efficacy of the proposed pipeline, positioning it as a promising solution for parking management and urban planning. Furthermore, it is worth noting that this approach leverages existing infrastructure without incurring any additional financial burden on the city and applies to other cities different from Malaga.

We have formulated the following research questions (RQs) to direct the investigation and evaluation of pertinent studies:

RQ1: Is it feasible to attain our objective using open data?
RQ2: Can we design a dataset-independent pipeline to ensure its applicability across various cities?

RQ3: Can we achieve our goal by leveraging existing urban resources while keeping additional infrastructure costs to a minimum?

The following sections present different aspects of our approach for on-street parking space localization. Section 2 provides an overview of the state-of-the-art in parking detection. Section 3 explains the extraction procedure. Section 4 introduces the concept of image matching and outlines our proposed method for addressing the challenges posed by camera movements. In Section 5, we discuss the techniques employed to overcome the limitations caused by the low-quality images captured by public cameras. Section 6 introduces the two state-of-the-art object detection models, YOLOv8 and Detectron2, utilized in our study. In Section 7, we describe the postprocessing step, wherein we select vehicles within the designated parking area and determine the real-world lengths of parking space lines. The experimental study and results are presented in Section 8. Finally, Section 9 discusses the adoption potential of our approach in smart cities, while Section 10 provides the main conclusions and outlines potential future work.

2. State-of-the-art

The use of public cameras installed in cities for parking detection offers a cost-effective and efficient strategy [12,13], particularly for computer vision-based approaches. These approaches have distinct advantages over individual sensors. A single camera can monitor a large parking area, reducing installation and maintenance costs. Furthermore, these cameras can perform additional tasks like detecting abnormal behavior and theft, making them versatile tools for enhancing public safety and urban space management. In summary, employing public cameras for parking detection eliminates the need for additional infrastructure costs and enhances public security, representing a smart investment for modern cities.

A literature review [12,14] has identified significant gaps in this field, needing further investigation. These gaps include the need for dataset-independent approaches and techniques for automatic parking situation detection, even when dealing with varying image angles and zoom levels. Additionally, effective methods are required that can consistently perform well under diverse lighting and weather conditions. Approaches to parking space detection are categorized into three main groups [12]: automatic detection of parking space positions (e.g., using bounding boxes), individual classification of parking spaces as occupied or vacant, and the detection and counting of vehicles in images.

Vehicle counting [12] is a regression problem that aims to predict the total number of vehicles in an image without the need to preprocess individual parking spaces. Most approaches employ deep learning methods and incorporate instance counts (e.g., bounding boxes) to make predictions. Manual labeling and training specific to a parking area are necessary for this method. On the other hand, classifying individual parking spaces as occupied or vacant [12] uses feature extraction and deep learning techniques. However, this task requires segmenting parking spaces before training and can be challenging to adapt to various image conditions. In some cases, transfer learning is utilized [15]. Nevertheless, the training process suffers from high computational costs and a shortage of labeled training data.

Many parking space detection systems are often not easily generalizable [12,14,16] due to their reliance on specific lighting conditions, weather, occlusions, and car sizes. Furthermore, the authors of [16] claim that no existing research has successfully achieved fully automated parking space detection, making it an open problem. In [17], a perspective transformation and then image segmentation are employed to automatically detect parking slots in a parking area. In [11], a CNN-LSTM model is proposed for the detection and prediction of parking spaces using fixed position cameras in structured parking lots. However, the image of the parking area is mostly aerial, with rectangular defined areas representing off-street parking. That does not reflect the real situation of on-street parking in the city. Other authors [12] explore interesting ideas, such as measuring a vehicle's duration in a location to determine if it's a parking zone. However, the results in this regard are inconclusive, requiring further research.

In summary, datasets and studies in this field often use rectangular off-street parking spaces [18] that do not accurately reflect real-life city scenarios. In a real urban environment, our goal is to analyze parking availability, which depends on both the user's vehicle size and the available space in the parking area. The monitoring of individual parking spaces in open areas remains as an open problem [12,13]. Our proposal aims to address these challenges.

In our study, we have curated images captured at different times and on different days, encompassing a wide spectrum of lighting scenarios and varying occupancy levels to ensure a diverse range of situations. This comprehensive approach ensures that we mitigate biases associated with analyzing images taken at the same location and time, where minimal changes might occur. We propose a method for detecting the parking area using pre-trained models (pre-trained for vehicle detection, not for parking detection) without retraining, even in the presence of camera angle variations and zoom levels. This method only requires identifying six points within a reference image and their Cartesian and geographic coordinates.

Furthermore, most works preprocess the images before model training. We analyze different preprocessing techniques before performing inference (not training) to enhance the predictions of the pre-trained models we employ. In this work, we explore a wide range of preprocessing techniques and comprehensively analyze them. Another type of optimization for selecting the most suitable preprocessing techniques would also be possible, as demonstrated in [19].

3. Extraction

In this section, we present the extraction process. Many cities offer images from public cameras that monitor city traffic as open data. The first stage of the pipeline involves extracting images from these cameras through web scraping with a carefully chosen 15-min interval. This interval is well-suited for our domain, striking a balance between gathering sufficient data for analysis, preventing system overload, and minimizing the risk of IP blocking, a potential consequence of excessive access requests.

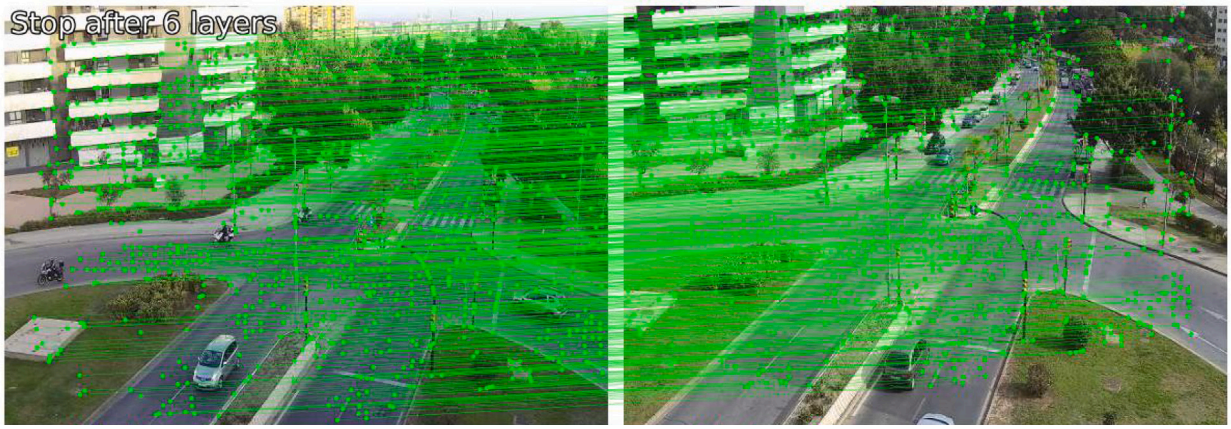

Fig. 2. Using LightGlue to match common features between the reference and new images.

Fig. 3. Matching key points. Parking DISTRITO_10_TV63-JUANFRANCESBOSCA.

4. Matching

In this section, we introduce the topic of image matching and present our proposed method for addressing the challenges posed by camera movements. These cameras often capture the same location, although not always from the same angle. Additionally, these cameras may sometimes move to observe other events.

This step is the only manual intervention required in the pipeline. Initially, a single image is selected as a reference, and six key points representing three lines that will be used to determine the parking location are identified by recording their pixel positions in the image alongside their corresponding Cartesian and real-world geographic coordinates. Subsequently, a pre-trained deep learning model matches common features between the reference image and new images. By tracking the movement of the nearest feature to each chosen reference point, its displacement is estimated in the compared image, thereby facilitating precise localization of the parking space despite camera movements. The effectiveness of our approach is demonstrated through visualizations illustrating the matched features in both the reference and new images (see Figs. 2, 3, and 4).

In future work, we plan to explore the selection of multiple reference images for different times of the day, weather conditions, or camera angles. We will then be able to choose the reference image with the most common features with the new image, which could further improve the final accuracy.

Specifically, we use LightGlue [20,21] as the pre-trained deep learning model for the matching process to automatically match common features between the reference image and new images (see Fig. 2). LightGlue is an innovative deep neural network designed for matching local features across images. It builds upon SuperGlue [22], the state-of-the-art sparse matching technique, and introduces simple yet effective improvements to enhance its efficiency, accuracy, and ease of training. One notable characteristic is LightGlue's adaptability to the complexity of image pairs, enabling faster inference for intuitively easy matches while maintaining robustness for challenging cases. LightGlue offers a Pareto-optimal [23] trade-off between efficiency and accuracy when compared to existing sparse and dense matchers. By tracking the movement of the nearest feature to each chosen reference point, we can

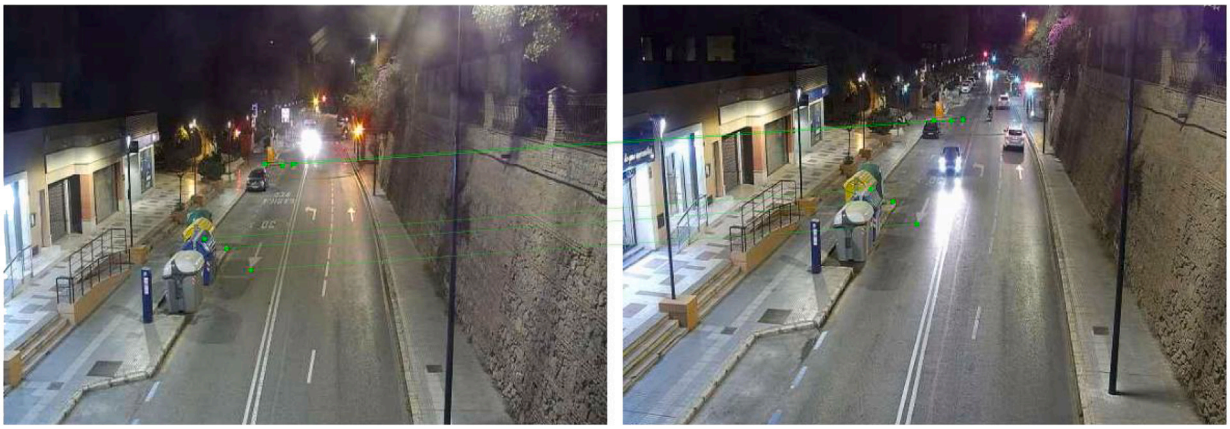

Fig. 4. Matching key points. Parking DISTRITO1TV38PRIES.

Fig. 5. Using a threshold M for common features, we determine whether the captured image corresponds to another location by comparing it to the reference image. This scenario occurs when the camera is pointed at a different location than usual to visualize a specific traffic event. Parking DISTRITO_10_TV63-JUANFRANCESBOSCA.

estimate its displacement in the compared image. This approach allows us to determine the location of the parking space we wish to measure its occupancy in the new image (see Figs. 3 and 4).

Furthermore, these cameras may occasionally point to entirely different locations from the usual ones to observe other events. To address this variability, we employ a threshold M to distinguish between images capturing distinct scenes. Specifically, images with a number of features below this threshold are considered to correspond to different scenes or locations. This thresholding step helps us filter out potential mismatches and improves the accuracy of our approach in scenarios where the cameras are observing different areas or viewpoints (see Fig. 5).

5. Preprocessing

In this section, we tackle the challenge posed by the poor quality of images from public cameras used in our study. To enhance the accuracy of our predictions, we explore various preprocessing techniques to improve the images before inputting them into the model predictor. Specifically, we compare the effects of the following methods: bilateral filtering, median blur, fast non-local means denoising (colored), normalization, histogram equalization and super-resolution.

5.1. Bilateral filtering

Bilateral filtering [24] is a non-linear edge-preserving smoothing technique that reduces noise while preserving the sharpness of edges in the image. It accomplishes this by applying a weighted average to the pixels based on both spatial distance and intensity difference. The formula is as follow (see Eq. (1)):

$$B(I,x) = \sum_{x_i \in \Omega} I(x_i) \cdot f_r(|I(x_i) - I(x)|) \cdot g_s(|x_i - x|) \tag{1}$$

- I: Represents a k-dimensional image.
- x: Represents the coordinates of the current pixel to be filtered.
- $B(I, x)$: Represents the output pixel value after bilateral filtering at coordinates x in the image I.
- Ω: Denotes a window centered in x, so $x_i \in \Omega$ is another pixel.
- $I(x_i)$: Denotes the intensity of the pixel at coordinates x_i in the input image I.
- $f_r(\|I(x_i) - I(x)\|)$: The range weight function f_r depends on the intensity difference between the pixel $I(x_i)$ and the central pixel $I(x)$. It controls the influence of neighboring pixels based on their intensity similarity. f_r typically uses a Gaussian function to compute the weight based on the intensity difference.
- $g_s(\|x_i - x\|)$: The spatial weight function g_s depends on the spatial distance between the neighboring pixel x_i and the central pixel x. It controls the influence of neighboring pixels based on their spatial proximity to the central pixel. g_s also typically uses a Gaussian function to compute the weight based on the spatial distance.

5.2. Median blur

Median blur is a simple yet effective noise reduction technique. It performs image smoothing using the median filter [25] with a specific window size of $k_{\text{size}} \times k_{\text{size}}$. If the image is multi-channel (e.g., RGB with three channels), each channel is processed independently.

The median filter operates by sliding a rectangular window of size $k_{\text{size}} \times k_{\text{size}}$ over the image. For each pixel in the image, the filter collects the pixel values within the window for each channel independently. The filter then sorts these pixel values in ascending order and replaces the central pixel's intensity with the median value from the sorted set.

The median filter is particularly useful for removing impulse noise, commonly known as salt-and-pepper noise, where some pixels have extremely high or low-intensity values compared to their neighbors. By considering the median value rather than the average of neighboring pixels, the median filter preserves edges and essential image features, making it suitable for applications where maintaining sharp boundaries is crucial.

5.3. Fast Non-Local means denoising (Colored)

Fast Non-Local Means Denoising (Colored) [26,27] is an advanced image denoising algorithm used to reduce Gaussian noise in colored (multi-channel) images. The method is an extension of the traditional Non-Local Means Denoising technique, which operates on grayscale images. It is highly effective in preserving image details while effectively reducing noise, making it a popular choice for various image processing applications.

This algorithm is based on the principle of comparing patches of pixels instead of individual pixels. It aims to estimate the noise-free pixel values by considering similarities between image patches. The technique exploits the redundancy in natural images, where similar patches tend to occur repeatedly in different regions.

The primary parameter for Fast Non-Local Means Denoising (Colored) is the *filter strength*, typically represented by h. The filter strength controls the amount of denoising applied to the image. A higher value of h results in stronger denoising, but it might also remove some fine details, while a lower value preserves more image details but may not be as effective in noise reduction.

This algorithm is particularly effective in reducing Gaussian noise, which is commonly encountered in various imaging scenarios. It reduces noise while preserving important image features, such as edges and textures, without causing significant distortion or blurring. Also, it is specifically designed to handle colored (multi-channel) images, making it suitable for a wide range of applications where color information is critical.

5.4. Image contrast enhancement using image normalization

Also known as "Contrast Stretching" [28], image normalization is a simple image enhancement based on a linear transformation technique that scales the maximum intensity of an image to 1 and the minimum intensity to 0, while the intermediate values are linearly mapped to the interval [0, 1] based on their original values. The normalized values are then multiplied by 255. In this case, we use image normalization on grayscale images.

5.5. Image contrast enhancement using CLAHE

Histogram equalization (HE) is a technique used to enhance the contrast of an image by redistributing the intensity levels across the entire range. The primary goal of histogram equalization is to stretch the histogram of an image to cover the entire available intensity range, thereby redistributing the intensity levels across the entire range and enhancing the overall contrast of the image.

Considering an image with pixels distributed over a specific range, for example, a bright image having pixels distributed in high values or a dark image having them in low values. The traditional Histogram Equalization (HE) technique redistributes the pixels to occupy the entire value range, stretching the histogram, which usually improves the overall contrast of the image. However, enhancing global contrast in an image often leads to undesired effects, such as noise amplification and the appearance of artifacts. Moreover, information can be lost due to excessive brightness since the histogram is not confined to a particular image region.

The Contrast Limited Adaptive Histogram Equalization (CLAHE) [29,30] method was proposed as an extension of the traditional Histogram Equalization (HE) method. It is an image processing technique that enhances local contrast in different image areas. It

Fig. 6. Image with no preprocessing.

employs adaptive histogram equalization. In this case, the image is divided into small "tiles" or blocks, for example, 8×8 pixels, and each block is independently equalized using a histogram, enhancing contrast locally and adaptively. Consequently, the histogram is confined to a small region except for cases with noise, in which case it is amplified. To prevent excessive noise amplification, contrast limitation is applied. If any tile exceeds the specified contrast limit, typically set at 40 in OpenCV, those pixels are clipped and evenly redistributed to other tiles before applying histogram equalization. After equalization, bilinear interpolation is applied to eliminate artifacts at the edges of the tiles.

The CLAHE method has been widely used in image processing and computer vision applications where local contrast is essential for improving image quality and extracting interesting features. For our implementation, we utilize the OpenCV default values for CLAHE, using 8×8 tile sizes and a contrast limit of 40 on grayscale images.

5.6. Super resolution

Super-resolution [31,32] is a process that aims to enhance the details of an image by upscaling it, i.e., reconstructing a high-resolution (HR) image from a low-resolution (LR) input. When increasing the dimensions of an image, additional pixels need to be generated, and basic image processing techniques often fail to provide satisfactory results as they do not consider contextual information during upscaling. Fortunately, deep learning techniques, particularly Generative Adversarial Networks (GANs) [33], have shown great promise in super-resolution tasks, producing significantly improved results.

Convolutional neural networks have proven successful in generating high-quality super-resolution images. However, many existing methods come with the drawback of requiring a large number of network parameters, leading to significant computational overhead during runtime for high-accuracy super-resolution results. One of the powerful models in the field of super-resolution is the *Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks* (LapSRN) [34].

LapSRN presents a novel approach to tackle these challenges by employing a deep Laplacian pyramid network. The network progressively reconstructs sub-band residuals of high-resolution images across multiple pyramid levels. Unlike other methods that rely on bicubic interpolation for pre-processing (which results in large feature maps), LapSRN directly extracts features from the low-resolution input space, significantly reducing computational load.

We apply the *LapSRN_x8* super-resolution model, which increases the size of the images by a factor of 8. Before performing this super-resolution technique, we perform a cropping process (see Figs. 6 and 7) around the parking area to reduce the image size.

6. Detection

Once we have successfully addressed the issues related to changes in camera positions and image quality, and once we have appropriate images, we proceed to vehicle detection. This section introduces two cutting-edge object detection models utilized in our study: YOLOv8 and Detectron2. These models are pretrained for detecting various types of vehicles with high accuracy, eliminating the need for retraining.

YOLOv8 [35], developed by Ultralytics, represents the latest iteration of the highly successful YOLO (You Only Look Once) series. YOLOv8 builds upon the advancements of its predecessors and incorporates innovative features and improvements, resulting in enhanced performance, efficiency, and ease of use. As a state-of-the-art (SOTA) model, YOLOv8 is renowned for its fast and accurate object detection capabilities. Its versatility and efficiency have made it a popular choice across diverse applications and domains. Specifically, we utilize the *yolov8x* model.

Detectron2 [36,37], developed by Facebook AI Research, is a state-of-the-art software system that implements cutting-edge object detection algorithms. Detectron2 offers a collection of base models, including R101-FPN and X101-FPN, commonly employed for

Fig. 7. Cropped and upscaled image using OpenCV super resolution.

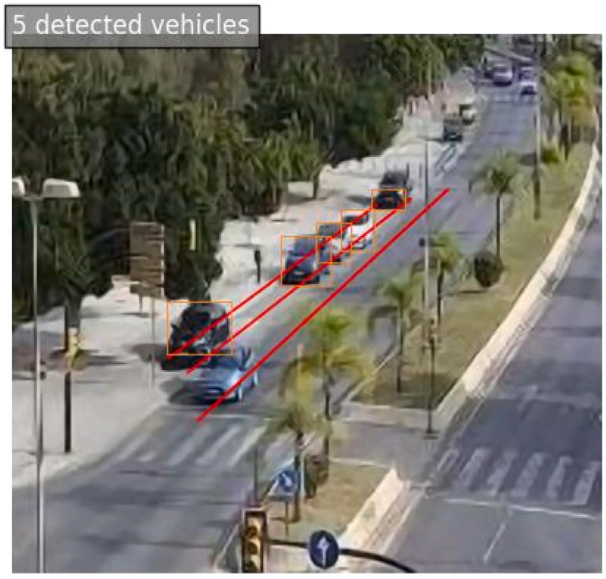

Fig. 8. Detecting the number of cars using YOLOv8x (confidence of 0.2) on a preprocessed image with OpenCV Super Resolution and Fast Non-Local Means Denoising (Colored) using an h parameter equal to 5. Parking DISTRITO_10_TV63-JUANFRANCESBOSCA.

Faster R-CNN. These base models serve as feature extractors from a large image set, such as ImageNet, which facilitates robust image representation for the detection process. In our study, we explore the Faster R-CNN approach using Detectron2 to expedite our development cycle and leverage the capabilities of a pre-existing, highly optimized model. We utilize the *X101-FPN* model, which has demonstrated exceptional performance in previous applications.

Each detection generated by the model includes the coordinates of the bounding box enclosing the detected object, the class label assigned to that object, and the confidence associated with that detection. The confidence level represents the estimated probability that a detected object belongs to a specific class.

We selected these pre-trained models because our goal is to achieve the best performance without the need for retraining and intensive data labeling to localize on-street parking spaces from the low-quality images captured by public cameras.

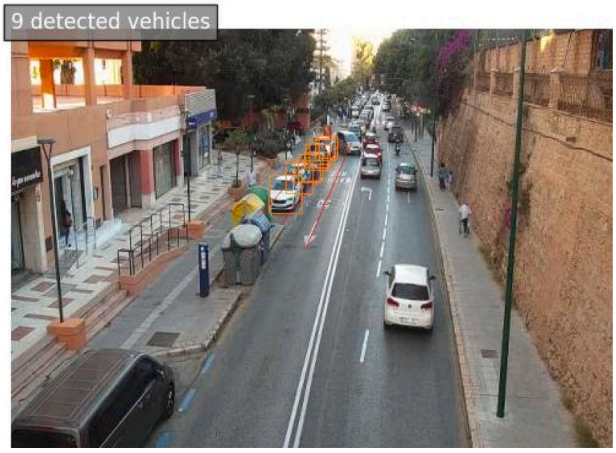

Fig. 9. An example of detecting multiple duplicate cars using YOLOv8x with a confidence threshold of 0.01. The intersection of these rectangles with the parking area boundary will be used to calculate the available parking spaces.

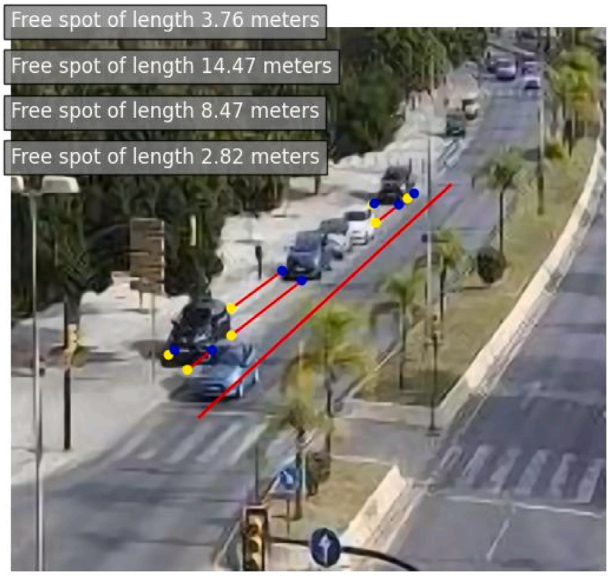

Fig. 10. Detecting the parking spaces using YOLOv8x (confidence of 0.2) on a cropped and preprocessed image with OpenCV Super Resolution and Fast Non-Local Means Denoising (Colored) using an h parameter equal to 5. Parking JUANFRANCESBOSCA.

7. Postprocessing

In this section, we describe the postprocessing step of our approach for on-street parking space localization. After performing matching, detecting key points, applying the corresponding preprocessing, and detecting vehicles in the image, we proceed with a postprocessing step to select the vehicles located within the designated parking area. To achieve this, we draw three lines using the previously defined six key points. The first line serves to identify vehicles that intersect with the parking zone, essentially checking for overlaps between the rectangular bounding boxes representing the vehicle locations and the line. The third line is utilized to discard vehicles positioned on the road that, due to their size, might otherwise intersect with the parking line. Finally, the second line helps identify the remaining portions of the parking line by subtracting the areas intersected by the vehicles detected with the first line. We also apply two thresholds for the bounding boxes' width (W) and height (H). This step is necessary because the image quality is often poor, and at times, large artifacts are detected, which are not relevant vehicles for our intended detection. In summary, line 1 is used for vehicle detection, line 3 is employed to remove non-parking vehicles, and line 2 allows us to obtain the parking space lines (see Figs. 8, 9, 10, and 11).

After obtaining the lines, we need to determine their real-world lengths. Fortunately, we have the real-world coordinates of the key points we detected earlier. Therefore, we can find the perspective transformation matrix H between the source and destination

Fig. 11. Detecting the parking spaces using YOLOv8x (confidence of 0.01) on a cropped and preprocessed image with OpenCV Super Resolution and Fast Non-Local Means Denoising (Colored) using an h parameter equal to 15. Parking PRIES.

Fig. 12. Aerial image of parking DISTRITO 10 TV63 JUANFRANCESBOSCA. Length measured using Google Maps Tools: 57.52 m.

planes [38,39]. The perspective transformation equation is applied (see Eq. (2)) so that the back-projection error is minimized (see Eq. (3)).

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{2}$$

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \tag{3}$$

In the process of matching points between images, there may be some errors that could impact the accuracy of the results. To address this concern, the algorithm incorporates robust methods such as *RANSAC* (Random Sample Consensus) [40] or *LEAST MEDIAN* [41] to handle outliers and improve the estimation.

Subsequently, we can apply the calculated transformation get the coordinates of the start and end points of each line and therefore we estimate their lengths in meters. While the results may not be exact, they are sufficiently approximate for our intended purpose.

Fig. 13. Aerial image of parking DISTRITO 1 TV38-PRIES. Length measured using Google Maps Tools: 37.59 m.

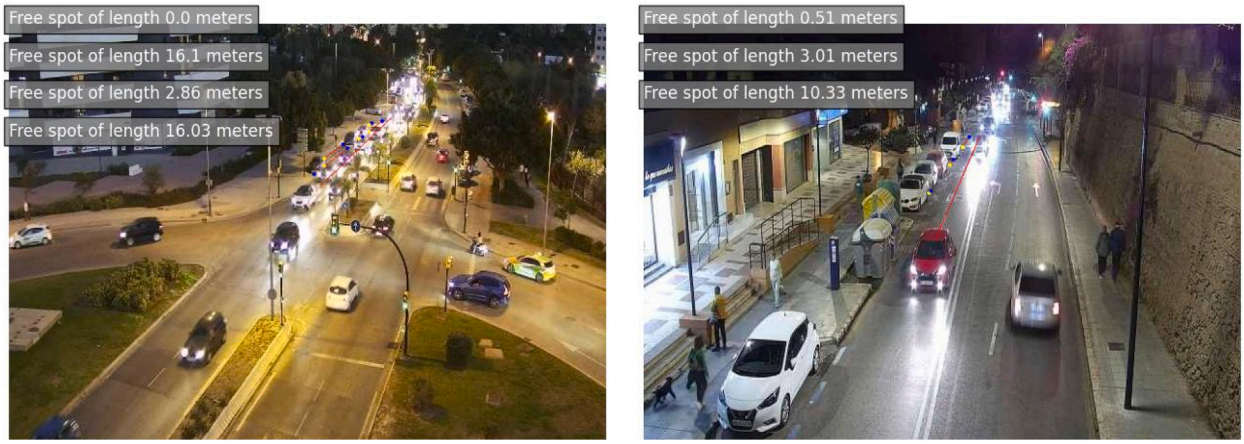

Fig. 14. Accurate measurements of parking area length are achieved even at night and in the presence of vehicle lights glare in both parking areas.

8. Experimentation

All experiments were conducted on a MSI GP66 Leopard laptop, equipped with an 11th Gen Intel(R) Core(TM) i7-11800H processor running at 2.30 GHz, 32 GB of RAM, and a GeForce RTX 3070 laptop GPU. We utilized two detection algorithms: *YOLOv8* with the model *yolov8x* and *Detectron2* with the model *X101-FPN*. The first one was executed using Python 3.10.11, nvidia-smi driver version 535.86.05, and CUDA version 12.2. However, the second one does not support CUDA version 12, so it was executed in a Docker container using Python 3.8.0, nvidia-smi driver version 535.86.05, and CUDA version 11.1.

For experimentation, we extracted open data images from public traffic cameras in Malaga. There are 89 traffic cameras located in 11 districts throughout the city of Malaga [42] (Spain), each serving a unique purpose. The primary objective of most of these traffic cameras is to monitor city traffic, and they are typically updated every minute. However, some of them, due to their strategic locations, can also be useful for additional purposes, such as, in this case, locating available parking spaces within the city. While not all of them have a direct view of on-street parking spaces, we collected data from all 89 cameras mentioned earlier, from November 12, 2022, to March 10, 2023, to support future analyses.

For evaluation purposes, we selected and labeled 82 images from two cameras (DISTRITO 10 TV63 and DISTRITO 1 TV38-PRIES) (see Figs. 12 and 13), each capturing on-street parking spaces within their field of view. These images encompassed various features, including parking filling percentages, camera distances, and lighting conditions (see Fig. 14), allowing us to test and validate our proposal thoroughly. Based on the results obtained in this initial work, we plan to expand this method to more parking areas in future research while introducing enhancements to the processes within our pipeline. In the labeling process, we take into account three aspects. Firstly, we consider the total number of vehicles present in the parking area. Secondly, we count the number of available parking spaces, regardless of their length; for example, if there are no vehicles, there would be one available space. Lastly, we provide an approximate measurement for each of the spaces we identify in each of these images based on visual inspection and leveraging the measurement tools in Google, which allows us to get an approximate idea of these measurements (see Figs. 12 and 13).

We aim to approach the problem in three ways:

1. We aim to analyze the accuracy of vehicle counting in parking areas, which is valuable for estimating the probability of available parking spaces.
2. We seek to analyze the accuracy in counting the number of available parking spaces where a car can fit, setting a minimum measurement of 4.3 m.

Table 1
The combination of preprocessing techniques analyzed.

ID	Preprocessing technique
1	No preprocessing
2	`medianBlur`
3	`bilateralFilter`
4	`fastNlMeansDenoisingColored5`
5	`fastNlMeansDenoisingColored10`
6	`fastNlMeansDenoisingColored15`
7	`clahe`
8	`normalization`
9	Cutting + Super resolution
10	Cutting + Super resolution + `medianBlur`
11	Cutting + Super resolution + `bilateralFilter`
12	Cutting + Super resolution + `fastNlMeansDenoisingColored5`
13	Cutting + Super resolution + `fastNlMeansDenoisingColored10`
14	Cutting + Super resolution + `fastNlMeansDenoisingColored15`
15	Cutting + Super resolution + `clahe`
16	Cutting + Super resolution + `normalization`
17	Cutting + `medianBlur` + Super resolution
18	Cutting + `bilateralFilter` + Super resolution
19	Cutting + `fastNlMeansDenoisingColored5` + Super resolution
20	Cutting + `fastNlMeansDenoisingColored10` + Super resolution
21	Cutting + `fastNlMeansDenoisingColored15` + Super resolution
22	Cutting + `clahe` + Super resolution
23	Cutting + `normalization` + Super resolution

3. We want to assess the precision in measuring the length of these spaces.

Counting the number of vehicles might seem like the most suitable solution. However, due to the distance between the parking area and the camera's position and the poor image quality, this is not always possible. We used different confidence levels to detect vehicles, but as we reduced this confidence interval to detect more challenging vehicles, some of the other vehicles were detected multiple times, resulting in an incorrect total car count. Nevertheless, by taking the intersection of the bounding boxes of all these vehicles, even if they are duplicated, we can calculate the difference with the parking line and obtain a fairly accurate estimation of the parking spaces. In this paper, we explore both possibilities for car counting and parking space estimation.

Finally, we compare the YOLOv8 and Detectron2 methods for these tasks, employing image enhancement and noise reduction techniques. Specifically, we analyze the combinations for both methods outlined in Table 1.

These 23 configurations are compared using 5 different confidence intervals (0.01, 0.05, 0.1, 0.15, 0.2) and each combination is tested with both algorithms and both parkings resulting in 460 different experiments.

The noise reduction techniques have different hyperparameters. Based on preliminary analysis, we have utilized the following settings:

- Bilateral Filter with the d parameter set to 21, and sigmaColor and sigmaSpace both set to 75.
- Fast Non-Local Means Denoising (Colored) with three different h parameter values: 5, 10, and 15 (*hColor* parameter equal to h).
- Median Blur using a kernel size of 3.

The remaining parameters are set to their default values.

To compare the techniques, we used the error metrics Mean Squared Error (MSE) and Mean Absolute Error (MAE) for all three types of comparisons (number of vehicles, number of spaces, and length of the detected spaces). We first analyze the performance of these combination techniques using MSE and MAE. Later, we analyze the accuracy levels and execution times of the techniques that got the best results. It is important to note that when referring to improvements, we specifically address MSE. At the same time, we provide both metrics for comprehensive reporting, with MAE primarily serving informational purposes, albeit leading to similar conclusions.

In Sections 8.1, 8.2, and 8.3, the three techniques that outperform the best none will be **bolded** in the text for each comparison. In the analysis in Section 8.4, the combined technique that successfully improves upon individual techniques will be **bolded** in the text.

8.1. Comparing techniques for vehicle count

In this section, we present and discuss the results obtained from different experiments on vehicle count using various methods and confidence levels (see Table 2). The best technique for each confidence level and model type is highlighted in gray. The best "none" for each confidence level and model type is highlighted in blue. The best "none" for all confidence levels and model types is highlighted in orange, and the best technique for all confidence levels and model types is highlighted in green.

Table 2
MSE and MAE results for vehicle counts in JUANFRANCESBOSCA and PRIES parking lots, along with counting of techniques beating 'none' for different confidence levels and against the best 'none'.

	YOLOv8 - JUANFRANCESBOSCA			Detectron2 - JUANFRANCESBOSCA			YOLOv8 - PRIES			Detectron2 - PRIES		
	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE
Confidence 0.01	**bilateralFilter_0.01**	**7.561**	2.000	**medianBlur_0.01**	**26.146**	3.951	**normalization_0.01**	**15.683**	**2.951**	**fastNlMeans15_0.01**	**36.073**	**5.049**
	normalization_0.01	9.951	**1.902**	**clahe_0.01**	27.610	**3.610**	medianBlur_0.01	20.976	3.463	normalization_0.01	41.244	5.634
	clahe_0.01	10.341	2.000	normalization_0.01	33.073	3.902	bilateralFilter_0.01	23.073	3.561	none_0.01	46.049	5.951
	none_0.01	10.610	2.317	fastNlMeans5_0.01	45.244	5.000	clahe_0.01	24.537	4.098	bilateralFilter_0.01	48.732	6.000
	fastNlMeans5_0.01	10.732	2.341	bilateralFilter_0.01	47.439	5.146	fastNlMeans5_0.01	25.634	4.073	medianBlur_0.01	54.756	6.171
	medianBlur_0.01	11.732	2.366	fastNlMeans10_0.01	49.195	5.146	none_0.01	26.024	3.780	fastNlMeans10_0.01	55.171	6.537
	fastNlMeans10_0.01	13.268	2.537	fastNlMeans15_0.01	50.000	5.220	fastNlMeans15_0.01	34.488	4.537	fastNlMeans5_0.01	59.463	6.829
	fastNlMeans15_0.01	15.512	2.732	none_0.01	52.317	5.098	fastNlMeans10_0.01	40.098	4.927	clahe_0.01	61.659	7.024
	upscaled_0.01	65.707	5.512	upscaled_0.01	91.268	7.659	upscaled_0.01	96.000	9.176	upscaled_0.01	84.024	7.829
Confidence 0.05	**fastNlMeans5_0.05**	**2.902**	**1.244**	**normalization_0.05**	**7.341**	**1.927**	**bilateralFilter_0.05**	**3.610**	**1.220**	**normalization_0.05**	**11.268**	**2.537**
	bilateralFilter_0.05	3.000	1.293	clahe_0.05	9.659	2.244	clahe_0.05	3.707	1.317	medianBlur_0.05	13.683	2.854
	clahe_0.05	3.268	1.268	medianBlur_0.05	10.878	2.585	none_0.05	3.732	1.293	upscaled_0.05	14.732	3.220
	none_0.05	3.366	1.366	none_0.05	18.829	3.024	**normalization_0.05**	3.805	**1.220**	fastNlMeans15_0.05	15.244	3.098
	medianBlur_0.05	3.415	1.317	bilateralFilter_0.05	20.171	3.244	fastNlMeans5_0.05	4.341	1.463	bilateralFilter_0.05	17.439	3.390
	normalization_0.05	3.463	1.268	fastNlMeans15_0.05	20.171	3.439	medianBlur_0.05	5.122	1.463	clahe_0.05	17.780	3.341
	fastNlMeans10_0.05	4.293	1.415	fastNlMeans5_0.05	20.317	3.195	fastNlMeans15_0.05	7.000	1.732	none_0.05	18.732	3.512
	fastNlMeans15_0.05	4.902	1.439	fastNlMeans10_0.05	24.268	3.390	fastNlMeans10_0.05	7.732	1.732	fastNlMeans10_0.05	21.366	3.659
	upscaled_0.05	5.634	1.683	upscaled_0.05	25.951	3.366	upscaled_0.05	10.488	2.488	fastNlMeans5_0.05	24.707	4.220
Confidence 0.1	**medianBlur_0.1**	**3.049**	1.293	**normalization_0.1**	**3.341**	**1.341**	**none_0.1**	**1.220**	**0.683**	**normalization_0.1**	**4.927**	1.707
	upscaled_0.1	3.195	1.293	clahe_0.1	3.707	1.366	clahe_0.1	1.341	0.756	**medianBlur_0.1**	5.683	**1.634**
	bilateralFilter_0.1	3.244	**1.195**	medianBlur_0.1	6.634	1.854	normalization_0.1	1.537	0.756	upscaled_0.1	5.878	1.976
	fastNlMeans15_0.1	3.390	1.244	fastNlMeans10_0.1	7.122	2.000	fastNlMeans5_0.1	1.683	0.854	none_0.1	6.805	2.073
	clahe_0.1	3.683	1.439	fastNlMeans15_0.1	8.073	2.122	bilateralFilter_0.1	1.878	0.902	bilateralFilter_0.1	7.439	2.122
	none_0.1	3.707	1.317	fastNlMeans5_0.1	8.244	2.098	medianBlur_0.1	2.415	1.098	fastNlMeans15_0.1	7.537	2.024
	fastNlMeans5_0.1	3.732	1.341	none_0.1	8.366	2.024	fastNlMeans15_0.1	3.561	0.976	clahe_0.1	8.415	2.122
	normalization_0.1	4.146	1.463	upscaled_0.1	10.171	2.073	upscaled_0.1	3.610	1.463	fastNlMeans10_0.1	9.927	2.220
	fastNlMeans10_0.1	4.317	1.341	bilateralFilter_0.1	10.220	2.366	fastNlMeans10_0.1	3.829	1.146	fastNlMeans5_0.1	10.732	2.634
Confidence 0.15	**upscaled_0.15**	**3.366**	**1.317**	**clahe_0.15**	**2.976**	**1.171**	**clahe_0.15**	**0.878**	**0.537**	**medianBlur_0.15**	**2.561**	**1.146**
	fastNlMeans15_0.15	4.000	**1.317**	normalization_0.15	3.171	1.317	bilateralFilter_0.15	1.000	0.610	none_0.15	3.122	1.220
	bilateralFilter_0.15	4.098	1.366	none_0.15	4.585	1.463	fastNlMeans5_0.15	1.195	0.707	upscaled_0.15	3.244	1.341
	medianBlur_0.15	4.341	1.512	fastNlMeans10_0.15	4.610	1.439	normalization_0.15	1.512	0.780	normalization_0.15	3.439	1.341
	fastNlMeans5_0.15	4.659	1.683	fastNlMeans5_0.15	5.268	1.659	medianBlur_0.15	1.585	0.854	clahe_0.15	4.780	1.561
	clahe_0.15	4.732	1.610	medianBlur_0.15	5.512	1.610	upscaled_0.15	1.780	1.000	bilateralFilter_0.15	4.976	1.707
	fastNlMeans10_0.15	4.951	1.634	fastNlMeans15_0.15	5.659	1.659	none_0.15	1.805	0.829	fastNlMeans10_0.15	5.341	1.488
	normalization_0.15	5.024	1.659	bilateralFilter_0.15	5.976	1.780	fastNlMeans10_0.15	2.146	0.927	fastNlMeans15_0.15	5.976	1.732
	none_0.15	5.171	1.610	upscaled_0.15	7.659	1.805	fastNlMeans15_0.15	2.561	0.902	fastNlMeans5_0.15	6.122	1.683
Confidence 0.2	**upscaled_0.2**	**4.268**	**1.488**	**clahe_0.2**	**2.951**	**1.146**	**bilateralFilter_0.2**	**0.878**	**0.585**	**medianBlur_0.2**	**2.024**	0.951
	bilateralFilter_0.2	4.805	1.537	normalization_0.2	3.195	1.293	medianBlur_0.2	1.341	0.756	normalization_0.2	2.098	1.024
	fastNlMeans15_0.2	5.024	1.561	none_0.2	3.512	1.268	clahe_0.2	1.341	0.707	none_0.2	2.146	**0.878**
	fastNlMeans5_0.2	5.268	1.805	fastNlMeans10_0.2	3.902	1.317	fastNlMeans5_0.2	1.488	0.805	clahe_0.2	2.268	1.098
	medianBlur_0.2	5.634	1.732	fastNlMeans5_0.2	4.439	1.561	normalization_0.2	1.561	0.829	upscaled_0.2	2.463	1.195
	fastNlMeans10_0.2	5.707	1.805	bilateralFilter_0.2	4.537	1.512	fastNlMeans10_0.2	1.805	0.878	bilateralFilter_0.2	2.780	1.122
	none_0.2	6.024	1.927	fastNlMeans15_0.2	4.634	1.463	fastNlMeans15_0.2	2.049	0.829	fastNlMeans10_0.2	3.439	1.146
	clahe_0.2	6.683	1.951	medianBlur_0.2	5.195	1.537	upscaled_0.2	2.073	1.049	fastNlMeans5_0.2	3.585	1.293
	normalization_0.2	6.707	1.976	upscaled_0.2	7.293	1.732	none_0.2	2.195	0.829	fastNlMeans15_0.2	3.829	1.244
	Experiment	Count		Experiment	Count	Total	Experiment	Count		Experiment	Count	Total
Per Confidence	**bilateralFilter**	**5**		bilateralFilter	1	6	**bilateralFilter**	**4**		bilateralFilter	1	5
	clahe	4		**clahe**	**5**	**9**	**clahe**	**4**		clahe	1	5
	fastNlMeans5	3		fastNlMeans5	2	5	fastNlMeans5	3		fastNlMeans5	0	3
	fastNlMeans10	2		fastNlMeans10	2	4	fastNlMeans10	1		fastNlMeans10	0	1
	fastNlMeans15	3		fastNlMeans15	2	5	fastNlMeans15	1		fastNlMeans15	2	3
	medianBlur	3		medianBlur	3	6	medianBlur	3		**medianBlur**	**4**	**7**
	normalization	2		**normalization**	**5**	7	normalization	3		**normalization**	**4**	**7**
	upscaled	3		upscaled	0	3	upscaled	2		upscaled	2	4
Outperforms Best none?	**bilateralFilter**	**Yes**		bilateralFilter	0	1	**bilateralFilter**	**Yes**		bilateralFilter	0	**1**
	clahe	**Yes**		**clahe**	**Yes**	**2**	**clahe**	**Yes**		clahe	0	**1**
	fastNlMeans5	**Yes**		fastNlMeans5	0	1	**fastNlMeans5**	**Yes**		fastNlMeans5	0	**1**
	fastNlMeans10	0		fastNlMeans10	0	0	fastNlMeans10	0		fastNlMeans10	0	0
	fastNlMeans15	0		fastNlMeans15	0	0	fastNlMeans15	0		fastNlMeans15	0	0
	medianBlur	**Yes**		medianBlur	0	1	medianBlur	0		**medianBlur**	**Yes**	**1**
	normalization	0		**normalization**	**Yes**	1	normalization	0		**normalization**	**Yes**	**1**
	upscaled	**Yes**		upscaled	0	1	upscaled	0		upscaled	0	0

In the row of the penultimate group of experiments (per confidence), we observe a count of how many times a preprocessing technique outperforms the case when there is no preprocessing (none). That involves summing each victory within each confidence interval analyzed.

In the row of the last group of experiments (outperform best none), we determine whether a preprocessing technique has succeeded in improving upon the best scenario with no preprocessing (none) across all confidence intervals for that specific parking area and model (1) or not (0). In both rows, there is a column (total) for each parking area, representing the sum of the counts

obtained across each model. That allows us to visualize which preprocessing technique has performed better in both models for that instance of the parking area.

Analyzing *confidence levels*, Detectron2 achieves the best results when using a confidence level of 0.2 in both parking areas. However, to achieve the best results with YOLOv8, it is necessary to lower the confidence level to 0.05 in the JUANFRANCESBOSCA parking area, which is the one with the farthest vehicles, indicating that it is more challenging to detect these vehicles. Nevertheless, YOLOv8 still outperforms Detectron2 in both parking areas despite having to lower the confidence level. As confidence levels increase, results tend to improve, as greater confidence in object detection often correlates with higher accuracy. However, a higher confidence level may lead to more precise detection but might overlook some instances of objects if they are less visible. Therefore, it is sometimes necessary to lower the confidence level to capture a broader range of objects.

Regarding which preprocessing techniques perform better, the results vary depending on the parking area and the model used, whether it's YOLOv8 or Detectron2. In the case of YOLOv8, the three best techniques that outperform the best none_0.05 (3.366 MSE) are **fastNlMeans5**_0.05 (2.902), **bilateralFilter**_0.05 (3.000), and **medianBlur**_0.1 (3.049) for the JUANFRANCESBOSCA parking area. However, it's interesting to note that there are many other techniques with different confidence intervals that manage to improve upon the best "none".

In the case of YOLOv8 in the PRIES parking, the three most effective techniques for improving the best none_0.1 (1.220) are **clahe**_0.15 (0.878), **bilateralFilter**_0.2 (0.878), and **bilateralFilter**_0.15 (1.000).

In the case of Detectron2 in the JUANFRANCESBOSCA parking, the three most effective techniques for improving the best none_0.2 (3.512) are **clahe**_0.2 (2.951), **clahe**_0.15 (2.976), **normalization**_0.15 (3.171).

As for Detectron2 in the PRIES parking, there are two techniques that improves the best none_0.2 (2.146) are **medianBlur**_0.2 (2.024) and **normalization**_0.2 (2.098). In summary, there are several techniques that prove effective in both parking areas and models.

The *technique that performs best* in terms of the count of instances with good results in the cases analyzed for vehicle counting is **clahe**, except in the case of Detectron2 in the PRIES parking lot.

Analyzing the preprocessing techniques and comparing them to the original, unprocessed images ("none"), we can see that they are effective in both YOLOv8 and Detectron2. YOLOv8 improves the Mean Squared Error (MSE) by 14% and 28% for the JUANFRANCESBOSCA and PRIES parking areas, respectively. Meanwhile, Detectron2 achieves a 16% and 6% improvement in MSE for the same areas. The results show that the preprocessing techniques are slightly more effective in the case of YOLOv8.

Finally, when *comparing the MSE of both YOLOv8 and Detectron2 models*, YOLOv8 achieved the best result when preprocessing techniques were not used and when they were applied. These findings suggest that the application of image processing techniques can enhance the performance of both YOLOv8 and Detectron2 models in vehicle counting tasks for parking scenarios. In this particular task, YOLOv8 achieves superior results in accurately detecting the number of vehicles in the parking area.

In summary, we performed a comprehensive exploration of diverse preprocessing techniques and confidence levels using YOLOv8 and Detectron2. While increased confidence levels generally led to improved results, this value needs to be reduced occasionally, being a trade-off between precise detection and overlooking less visible objects. Notably, YOLOv8 consistently outperformed Detectron2 with and without preprocessing techniques in both parking scenarios. The effectiveness of preprocessing techniques was evident across all cases, with clahe and normalization emerging as particularly successful techniques for this task on the models and parking areas analyzed. These findings highlight the potential of image processing techniques to enhance vehicle counting performance in parking scenarios for both YOLOv8 and Detectron2. However, accurately counting the number of vehicles in the parking area can be challenging due to poor image quality and the distance between the parking area and the camera position. These factors often necessitate lowering the confidence threshold for detecting more distant and difficult-to-see vehicles, which may result in multiple detections of the same vehicle. Increasing the confidence threshold improves the accuracy of individual car detection but may lead to missing some of the more challenging ones. Therefore, we propose a more effective approach of reducing the confidence threshold and calculating the intersection difference of all detected car rectangles with the parking lines. This method allows us to determine which sections of the parking area are unoccupied (see Figs. 10 and 11). In the following sections, we will provide a detailed analysis of this approach.

8.2. Comparing techniques for parking spaces count

In this section, we present and discuss the results obtained from different experiments on parking spaces count using various methods and confidence levels (see Table 3).

Analyzing *confidence levels*, the best results were obtained when using a confidence level of 0.01 in both parking areas, when using YOLOv8 and when using Detectron2. As mentioned earlier, lowering the confidence threshold results in multiple detections of the same vehicle. However, this is not a concern in this case, as we now calculate the intersection of all rectangles and subtract them from the parking line to determine the available parking area.

When it comes to evaluating *which preprocessing techniques yield better results*, the outcomes also vary depending on the parking area and the model used, whether it's YOLOv8 or Detectron2. In the case of YOLOv8 in the JUANFRANCESBOSCA parking, the three best-performing techniques that surpass the best none_0.05 (0.512 MSE) are **upscaled**_0.01 (0.293), **medianBlur**_0.01 (0.341), and **fastNlMeans15**_0.01 (0.415). Although, it's worth noting that there are several other techniques at different confidence intervals (0.01, 0.05, and 0.1) that also yield better results than the best none_0.05.

In the case of YOLOv8 for the PRIES parking area, the three techniques that outperform the best none_0.01 (0.415) are **upscaled**_0.01 (0.244), **fastNlMeans15**_0.01 (0.268), and **normalization**_0.05 (0.268).

Table 3
MSE and MAE results for parking spaces count in JUANFRANCESBOSCA and PRIES parking lots, along with counting of techniques beating 'none' for different confidence levels and against the best 'none'.

	YOLOv8 - JUANFRANCESBOSCA			Detectron2 - JUANFRANCESBOSCA			YOLOv8 - PRIES			Detectron2 - PRIES		
	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE
Confidence 0.01	**upscaled_0.01**	**0.293**	**0.293**	**bilateralFilter_0.01**	**0.317**	**0.317**	**upscaled_0.01**	**0.244**	**0.244**	**fastNlMeans15_0.01**	**0.220**	0.220
	medianBlur_0.01	0.341	0.341	none_0.01	0.415	0.415	fastNlMeans15_0.01	0.268	0.268	**bilateralFilter_0.01**	0.244	**0.195**
	fastNlMeans15_0.01	0.415	0.415	fastNlMeans10_0.01	0.415	0.366	bilateralFilter_0.01	0.317	0.317	upscaled_0.01	0.244	0.244
	clahe_0.01	0.463	0.415	fastNlMeans15_0.01	0.415	0.366	fastNlMeans5_0.01	0.317	0.317	none_0.01	0.268	0.268
	bilateralFilter_0.01	0.488	0.390	normalization_0.01	0.415	0.366	normalization_0.01	0.390	0.390	normalization_0.01	0.293	0.293
	none_0.01	0.512	0.415	medianBlur_0.01	0.463	0.415	none_0.01	0.415	0.366	medianBlur_0.01	0.341	0.341
	fastNlMeans5_0.01	0.512	0.415	fastNlMeans5_0.01	0.463	0.415	medianBlur_0.01	0.439	0.341	fastNlMeans5_0.01	0.366	0.317
	fastNlMeans10_0.01	0.512	0.415	clahe_0.01	0.463	0.366	clahe_0.01	0.463	0.463	fastNlMeans10_0.01	0.488	0.293
	normalization_0.01	0.512	0.415	upscaled_0.01	0.561	0.463	fastNlMeans10_0.01	0.488	0.439	clahe_0.01	0.488	0.390
Confidence 0.05	**clahe_0.05**	**0.415**	**0.366**	**none_0.05**	**0.415**	0.415	**normalization_0.05**	**0.268**	**0.268**	**bilateralFilter_0.05**	**0.293**	**0.244**
	fastNlMeans5_0.05	0.439	0.390	**bilateralFilter_0.05**	**0.415**	0.415	bilateralFilter_0.05	0.341	0.341	normalization_0.05	0.341	0.341
	fastNlMeans15_0.05	0.439	0.439	**normalization_0.05**	**0.415**	**0.317**	fastNlMeans15_0.05	0.341	0.293	upscaled_0.05	0.341	0.293
	upscaled_0.05	0.463	0.415	fastNlMeans10_0.05	0.463	0.415	clahe_0.05	0.341	0.341	none_0.05	0.366	0.366
	medianBlur_0.05	0.512	0.415	clahe_0.05	0.463	0.366	upscaled_0.05	0.341	0.341	fastNlMeans15_0.05	0.366	0.317
	normalization_0.05	0.512	0.415	medianBlur_0.05	0.512	0.463	medianBlur_0.05	0.488	0.390	medianBlur_0.05	0.390	0.341
	none_0.05	0.537	0.439	fastNlMeans15_0.05	0.610	0.463	fastNlMeans5_0.05	0.512	0.415	fastNlMeans5_0.05	0.439	0.390
	fastNlMeans10_0.05	0.561	0.512	fastNlMeans5_0.05	0.634	0.488	fastNlMeans10_0.05	0.512	0.463	clahe_0.05	0.463	0.366
	bilateralFilter_0.05	0.610	0.512	upscaled_0.05	0.659	0.561	none_0.05	0.610	0.463	fastNlMeans10_0.05	0.537	0.341
Confidence 0.1	**clahe_0.1**	**0.463**	**0.415**	**none_0.1**	**0.390**	**0.390**	**bilateralFilter_0.1**	**0.341**	**0.341**	**bilateralFilter_0.1**	**0.317**	**0.268**
	bilateralFilter_0.1	0.512	0.463	fastNlMeans10_0.1	0.415	0.415	**normalization_0.1**	0.390	**0.341**	upscaled_0.1	0.366	0.317
	fastNlMeans5_0.1	0.512	0.463	bilateralFilter_0.1	0.463	0.463	upscaled_0.1	0.390	0.390	none_0.1	0.415	0.366
	medianBlur_0.1	0.537	0.439	fastNlMeans15_0.1	0.512	0.463	clahe_0.1	0.415	0.366	medianBlur_0.1	0.439	0.341
	fastNlMeans15_0.1	0.537	0.488	normalization_0.1	0.512	0.415	fastNlMeans5_0.1	0.488	0.390	normalization_0.1	0.439	0.390
	upscaled_0.1	0.537	0.488	medianBlur_0.1	0.561	0.512	medianBlur_0.1	0.512	0.415	fastNlMeans5_0.1	0.512	0.415
	none_0.1	0.561	0.463	upscaled_0.1	0.585	0.488	fastNlMeans10_0.1	0.537	0.488	fastNlMeans10_0.1	0.561	0.366
	fastNlMeans10_0.1	0.561	0.512	clahe_0.1	0.610	0.512	fastNlMeans15_0.1	0.561	0.415	clahe_0.1	0.561	0.415
	normalization_0.1	0.561	0.512	fastNlMeans5_0.1	0.732	0.537	none_0.1	0.707	0.512	fastNlMeans15_0.1	0.634	0.439
Confidence 0.15	**none_0.15**	**0.512**	**0.463**	**fastNlMeans10_0.15**	**0.537**	**0.439**	**bilateralFilter_0.15**	**0.415**	**0.366**	**bilateralFilter_0.15**	**0.293**	**0.244**
	clahe_0.15	**0.512**	**0.463**	none_0.15	0.561	0.512	clahe_0.15	0.439	0.390	normalization_0.15	0.415	0.366
	medianBlur_0.15	0.537	0.488	fastNlMeans15_0.15	0.561	0.463	fastNlMeans15_0.15	0.512	0.415	none_0.15	0.439	0.390
	bilateralFilter_0.15	0.537	0.488	bilateralFilter_0.15	0.585	0.488	normalization_0.15	0.512	0.415	medianBlur_0.15	0.488	0.390
	upscaled_0.15	0.561	0.512	normalization_0.15	0.585	0.488	upscaled_0.15	0.512	0.415	upscaled_0.15	0.512	0.415
	fastNlMeans5_0.15	0.610	0.512	upscaled_0.15	0.610	0.512	fastNlMeans10_0.15	0.537	0.488	fastNlMeans10_0.15	0.561	0.366
	normalization_0.15	0.610	0.512	medianBlur_0.15	0.634	0.585	medianBlur_0.15	0.585	0.439	clahe_0.15	0.561	0.415
	fastNlMeans15_0.15	0.683	0.634	clahe_0.15	0.659	0.561	fastNlMeans5_0.15	0.610	0.463	fastNlMeans5_0.15	0.610	0.463
	fastNlMeans10_0.15	0.707	0.610	fastNlMeans5_0.15	0.780	0.585	none_0.15	0.659	0.512	fastNlMeans15_0.15	0.634	0.439
Confidence 0.2	**bilateralFilter_0.2**	**0.561**	**0.512**	**fastNlMeans10_0.2**	**0.463**	**0.415**	**clahe_0.2**	**0.463**	0.415	**bilateralFilter_0.2**	**0.390**	**0.293**
	medianBlur_0.2	0.585	0.537	none_0.2	0.537	0.488	**bilateralFilter_0.2**	0.488	**0.390**	medianBlur_0.2	0.488	0.390
	upscaled_0.2	0.634	0.537	medianBlur_0.2	0.634	0.585	**upscaled_0.2**	0.488	**0.390**	normalization_0.2	0.512	0.415
	fastNlMeans5_0.2	0.659	**0.512**	bilateralFilter_0.2	0.634	0.537	fastNlMeans15_0.2	0.512	0.415	upscaled_0.2	0.537	0.439
	clahe_0.2	0.683	0.585	normalization_0.2	0.634	0.537	normalization_0.2	0.561	0.463	fastNlMeans10_0.2	0.585	0.390
	fastNlMeans10_0.2	0.707	0.610	upscaled_0.2	0.634	0.537	none_0.2	0.610	0.463	fastNlMeans5_0.2	0.610	0.463
	fastNlMeans15_0.2	0.707	0.610	fastNlMeans15_0.2	0.683	0.488	medianBlur_0.2	0.610	0.463	fastNlMeans15_0.2	0.634	0.439
	normalization_0.2	0.805	0.610	clahe_0.2	0.756	0.610	fastNlMeans5_0.2	0.683	0.488	none_0.2	0.659	0.463
	none_0.2	0.829	0.634	fastNlMeans5_0.2	0.805	0.561	fastNlMeans10_0.2	0.707	0.561	clahe_0.2	0.829	0.537
	Experiment	Count		Experiment	Count	Total	Experiment	Count		Experiment	Count	Total
Per Confidence	bilateralFilter	3		bilateralFilter	1	**4**	**bilateralFilter**	**5**		**bilateralFilter**	**5**	**10**
	clahe	**4**		clahe	0	**4**	clahe	4		clahe	0	0
	fastNlMeans5	3		fastNlMeans5	0	3	fastNlMeans5	4		fastNlMeans5	1	5
	fastNlMeans10	1		**fastNlMeans10**	**2**	3	fastNlMeans10	3		fastNlMeans10	1	4
	fastNlMeans15	**4**		fastNlMeans15	0	**4**	**fastNlMeans15**	**5**		fastNlMeans15	2	7
	medianBlur	**4**		medianBlur	0	**4**	medianBlur	3		medianBlur	1	4
	normalization	2		normalization	0	2	**normalization**	**5**		normalization	3	8
	upscaled	**4**		upscaled	0	**4**	**upscaled**	**5**		upscaled	4	9
Outperforms Best none?	**bilateralFilter**	**Yes**		**bilateralFilter**	**Yes**	**2**	**bilateralFilter**	**Yes**		**bilateralFilter**	**Yes**	**2**
	clahe	**Yes**		clahe	0	1	**clahe**	**Yes**		clahe	0	1
	fastNlMeans5	**Yes**		fastNlMeans5	0	1	**fastNlMeans5**	**Yes**		fastNlMeans5	0	1
	fastNlMeans10	0		fastNlMeans10	0	0	fastNlMeans10	0		fastNlMeans10	0	0
	fastNlMeans15	**Yes**		fastNlMeans15	0	1	**fastNlMeans15**	**Yes**		**fastNlMeans15**	**Yes**	**2**
	medianBlur	**Yes**		medianBlur	0	1	medianBlur	0		medianBlur	0	0
	normalization	0		normalization	0	0	**normalization**	**Yes**		normalization	0	1
	upscaled	**Yes**		upscaled	0	1	**upscaled**	**Yes**		**upscaled**	**Yes**	**2**

For Detectron2 in the JUANFRANCESBOSCA parking area, only **bilateralFilter**_0.01 (0.317) improves upon the best none_0.01 (0.390).

In the case of Detectron2 in the PRIES parking area, the three top-performing techniques that outperform the best none_0.05 (0.268) are **fastNlMeans15**_0.01 (0.220), **bilateralFilter**_0.01 (0.244), and **upscaled**_0.01 (0.244).

The *technique that performs best* in terms of the count of instances with good results in the cases analyzed for counting the number of available parking spaces is **bilateralFilter**.

Analyzing the preprocessing techniques and comparing them to the original, unprocessed images ("none"), we can see that they are effective in both YOLOv8 and Detectron2 in all cases. YOLOv8 improves the MSE by 43% and 41% for the JUANFRANCESBOSCA and PRIES parking areas, respectively. Meanwhile, Detectron2 achieves a 19% and 18% improvement in MSE for the same areas. The preprocessing techniques are equally effective in both parking areas, and although they are also effective in both models, they have a greater impact in the case of YOLOv8 than in Detectron2.

When *comparing the MSE of both YOLOv8 and Detectron2* models, Detectron2 obtained much better results when no preprocessing techniques were applied (orange). In contrast, when using preprocessing techniques, YOLOv8 achieved better results in the JUANFRANCESBOSCA parking area, while Detectron2 performed better in the PRIES parking area, with both models yielding very similar results (green). In this specific task of counting parking spaces, YOLOv8 excels in one parking area, while Detectron2 excels in the other.

In summary, our parking space count analysis involved a detailed examination of various preprocessing techniques and confidence levels using YOLOv8 and Detectron2. Lowering the confidence threshold to 0.01 yielded the best results in all cases. Preprocessing techniques demonstrated effectiveness in the analyzed models and parking areas, showcasing significant MSE improvements. Bilateral filtering emerges as a preprocessing technique that works well in the models and datasets analyzed. Detectron2 performed exceptionally well without preprocessing, a noteworthy advantage for real-time detection scenarios due to the reduced processing time. However, with preprocessing techniques, YOLOv8 excelled in the JUANFRANCESBOSCA parking area, while Detectron2 outperformed in the PRIES parking area. This divergence emphasizes the model-specific effectiveness of YOLOv8 and Detectron2 in distinct parking scenarios.

8.3. Comparing techniques for measuring parking spaces

In the preceding section, we analyzed techniques for accurately measuring the total number of parking spaces, regardless of their length. In essence, we considered a parking space even when unoccupied, and if a car was positioned in the middle of a space, we counted it as two parking spaces—one on each side of the car. We only included spaces with measurements exceeding 4.3 m. However, if we know the length of the available spaces, we can provide a more accurate response to the user about the probability of available space when they arrive at the parking area. In this section, we will determine the approximate lengths of these parking spaces. This information will be valuable for users, helping them determine if their car can fit in a specific spot and assess the likelihood of finding available parking within the time it takes to reach the parking area (see Table 4). It should be noted that measuring the lengths of parking spaces in meters leads to higher MAE and MSE values due to the increased variability in space lengths compared to the previous studies, where the goal was to determine the total number of parking spaces or vehicles.

When analyzing *confidence levels*, the best results were achieved with a confidence level of 0.01, consistent with the previous section's findings when using YOLOv8 and Detectron2 in both parking areas.

When it comes to evaluating *which preprocessing techniques yield better results*, the outcomes also vary depending on the parking area and the model used, whether it's YOLOv8 or Detectron2. In the case of YOLOv8 in the JUANFRANCESBOSCA parking, the three best-performing techniques that surpass the best none_0.01 (59.843 MSE) are **upscaled**_0.01 (48.076), **fastNlMeans15**_0.01 (53.708), and **medianBlur**_0.01 (57.030).

In the case of YOLOv8 for the PRIES parking area, the three techniques that outperform the best none_0.01 (12.463) are **upscaled**_0.01 (6.261), **fastNlMeans15**_0.01 (7.468), and **fastNlMeans5**_0.01 (8.101).

For Detectron2 in the JUANFRANCESBOSCA parking area, no techniques improve upon the best none_0.01 (44.129). In the case of Detectron2 in the PRIES parking area, there are two techniques that outperform the best none_0.01 (8.838) which are **fastNlMeans10**_0.01 (7.324), and **bilateralFilter**_0.01 (7.730).

The *technique that performs best* in terms of the count of instances with good results in the cases analyzed for measuring the lengths of parking spaces is **bilateralFilter** in the PRIES parking. However, in the JUANFRANCESBOSCA parking, there is no clear winner. When using YOLOv8 in JUANFRANCESBOSCA, the best techniques are **fastNlMeans15** and **upscaled**. When using Detectron2 in JUANFRANCESBOSCA, the preprocessing techniques do not lead to performance improvements.

Analyzing the preprocessing techniques and comparing them to the original, unprocessed images ("none"), we can see that they are effective in both YOLOv8 and Detectron2, except in one particular case. YOLOv8 improves the Mean Squared Error (MSE) by 20% and 50% for the JUANFRANCESBOSCA and PRIES parking areas, respectively. Meanwhile, Detectron2 achieves a 0% and 17% improvement in MSE for the same areas. The results show that the preprocessing techniques are more effective in the case of YOLOv8.

When *comparing the MSE of both YOLOv8 and Detectron2* models, Detectron2 achieved significantly better results when no preprocessing techniques were applied. However, when using preprocessing techniques, Detectron2 performed better in the JUANFRANCESBOSCA parking area, while YOLOv8 outperformed in the PRIES parking area. Specifically, when measuring the lengths of parking spaces, both models yielded similar results when preprocessing was employed.

In summary, exploring techniques for measuring parking space lengths involved an in-depth analysis of confidence levels and preprocessing methods using YOLOv8 and Detectron2. Notably, a confidence level of 0.01 consistently produced the best results across both models and parking areas, aligning with the findings from the previous section. Preprocessing techniques were more beneficial in YOLOv8 than Detectron 2, which surprisingly worked well without preprocessing. Detectron 2 outperformed YOLOv8 when none of them used preprocessing techniques in all scenarios and also in the JUANFRANCESBOSCA parking without needing preprocessing when YOLOv8 used preprocessing techniques which is an advantage for Detectron 2 for real-time detection scenarios due to the reduced processing time. In the PRIES parking area, YOLOv8, using preprocessing techniques, got the best results. The fastNlMeans10 works well in most cases. The bilateral filter was helpful in the PRIES parking for both models, and upscaling was the best technique for YOLOv8 in both parking areas. These findings emphasize the model-specific effectiveness of preprocessing in measuring parking space lengths.

Table 4
MSE and MAE results for parking spaces length in JUANFRANCESBOSCA and PRIES parking lots, along with counting of techniques beating 'none' for different confidence levels and against the best 'none'.

	YOLOv8 - JUANFRANCESBOSCA			Detectron2 - JUANFRANCESBOSCA			YOLOv8 - PRIES			Detectron2 - PRIES		
	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE	Experiment	MSE	MAE
Confidence 0.01	**upscaled_0.01**	**48.076**	**3.722**	**none_0.01**	**44.129**	4.118	**upscaled_0.01**	**6.261**	**1.193**	**fastNlMeans10_0.01**	**7.324**	1.684
	fastNlMeans15_0.01	53.708	4.135	fastNlMeans15_0.01	45.638	**3.781**	fastNlMeans15_0.01	7.468	1.577	bilateralFilter_0.01	7.730	**1.683**
	medianBlur_0.01	57.030	4.021	bilateralFilter_0.01	49.216	3.796	fastNlMeans5_0.01	8.101	1.688	none_0.01	8.838	1.780
	none_0.01	59.843	4.193	fastNlMeans5_0.01	55.757	4.366	medianBlur_0.01	10.384	1.896	fastNlMeans5_0.01	9.561	1.877
	fastNlMeans5_0.01	59.995	4.474	normalization_0.01	59.321	4.109	normalization_0.01	10.476	2.085	medianBlur_0.01	10.683	1.975
	fastNlMeans10_0.01	60.269	4.382	clahe_0.01	60.092	4.258	fastNlMeans10_0.01	10.908	2.035	clahe_0.01	10.942	2.197
	bilateralFilter_0.01	65.908	3.972	fastNlMeans10_0.01	67.168	4.408	bilateralFilter_0.01	11.797	2.026	normalization_0.01	11.187	1.988
	normalization_0.01	69.579	4.671	upscaled_0.01	88.490	5.482	none_0.01	12.463	2.260	fastNlMeans15_0.01	12.408	1.829
	clahe_0.01	75.698	4.793	medianBlur_0.01	125.622	6.130	clahe_0.01	14.440	2.544	upscaled_0.01	24.849	2.515
Confidence 0.05	**upscaled_0.05**	**60.554**	**4.481**	**none_0.05**	**55.011**	4.420	**fastNlMeans15_0.05**	**9.757**	**1.804**	**none_0.05**	**10.297**	1.941
	fastNlMeans15_0.05	64.829	4.845	normalization_0.05	62.652	**4.277**	fastNlMeans5_0.05	9.935	1.924	bilateralFilter_0.05	10.618	**1.917**
	fastNlMeans5_0.05	69.255	5.065	fastNlMeans10_0.05	69.050	4.446	normalization_0.05	10.651	1.852	clahe_0.05	11.215	1.938
	none_0.05	72.307	5.212	clahe_0.05	84.468	5.025	upscaled_0.05	11.055	1.825	fastNlMeans10_0.05	11.411	1.991
	bilateralFilter_0.05	77.551	4.733	fastNlMeans5_0.05	85.514	4.909	bilateralFilter_0.05	12.618	2.015	medianBlur_0.05	11.882	2.121
	fastNlMeans10_0.05	85.232	5.461	fastNlMeans15_0.05	86.068	5.208	medianBlur_0.05	12.811	2.171	fastNlMeans5_0.05	11.977	2.086
	medianBlur_0.05	92.186	5.764	bilateralFilter_0.05	88.074	4.671	none_0.05	16.700	2.553	fastNlMeans15_0.05	12.965	1.977
	normalization_0.05	97.784	5.600	upscaled_0.05	107.716	6.172	clahe_0.05	21.052	2.553	normalization_0.05	14.447	2.237
	clahe_0.05	103.637	5.739	medianBlur_0.05	142.521	6.561	fastNlMeans10_0.05	22.229	2.635	upscaled_0.05	26.963	2.995
Confidence 0.1	**fastNlMeans15_0.1**	**96.550**	5.923	**none_0.1**	**63.151**	**4.800**	**fastNlMeans5_0.1**	**12.604**	2.103	**bilateralFilter_0.1**	**12.068**	**2.141**
	medianBlur_0.1	104.977	6.599	fastNlMeans10_0.1	79.739	5.025	bilateralFilter_0.1	13.090	**2.099**	clahe_0.1	12.844	2.206
	upscaled_0.1	106.557	5.756	normalization_0.1	95.155	5.861	normalization_0.1	14.472	2.211	fastNlMeans10_0.1	14.913	2.295
	clahe_0.1	112.847	6.584	fastNlMeans5_0.1	101.250	5.553	medianBlur_0.1	17.468	2.459	fastNlMeans5_0.1	15.525	2.496
	none_0.1	114.655	6.730	clahe_0.1	103.935	5.452	none_0.1	19.470	2.774	fastNlMeans15_0.1	15.604	2.362
	bilateralFilter_0.1	118.027	**5.571**	fastNlMeans15_0.1	105.565	6.284	upscaled_0.1	20.250	2.341	normalization_0.1	16.190	2.479
	fastNlMeans5_0.1	120.429	7.074	bilateralFilter_0.1	112.892	5.927	clahe_0.1	21.785	2.622	none_0.1	19.293	2.692
	normalization_0.1	122.623	6.959	upscaled_0.1	135.658	6.943	fastNlMeans15_0.1	22.739	2.681	medianBlur_0.1	22.368	2.807
	fastNlMeans10_0.1	130.452	7.348	medianBlur_0.1	157.565	7.319	fastNlMeans10_0.1	24.710	2.796	upscaled_0.1	28.511	3.111
Confidence 0.15	**upscaled_0.15**	**131.855**	**6.839**	**fastNlMeans10_0.15**	**90.302**	5.689	**bilateralFilter_0.15**	**13.921**	**2.176**	**fastNlMeans10_0.15**	**14.912**	**2.293**
	bilateralFilter_0.15	147.563	6.918	none_0.15	93.897	**5.554**	fastNlMeans5_0.15	15.734	2.456	clahe_0.15	15.566	2.403
	fastNlMeans15_0.15	153.388	8.468	normalization_0.15	100.572	6.303	normalization_0.15	16.000	2.409	fastNlMeans15_0.15	15.968	2.429
	clahe_0.15	161.005	8.292	fastNlMeans5_0.15	109.615	6.007	medianBlur_0.15	22.465	2.829	fastNlMeans5_0.15	18.405	2.787
	medianBlur_0.15	173.739	8.367	fastNlMeans15_0.15	115.010	6.544	fastNlMeans15_0.15	24.312	2.821	bilateralFilter_0.15	22.950	2.763
	normalization_0.15	182.821	8.932	clahe_0.15	118.890	6.283	upscaled_0.15	25.217	2.823	medianBlur_0.15	23.835	2.995
	fastNlMeans10_0.15	192.598	9.393	bilateralFilter_0.15	122.204	6.277	clahe_0.15	25.582	2.908	normalization_0.15	26.657	3.092
	fastNlMeans5_0.15	193.884	9.197	upscaled_0.15	146.839	7.289	fastNlMeans10_0.15	25.970	2.867	none_0.15	30.322	3.535
	none_0.15	203.049	9.039	medianBlur_0.15	174.742	7.676	none_0.15	30.722	3.203	upscaled_0.15	30.707	3.422
Confidence 0.2	**bilateralFilter_0.2**	**151.759**	**7.358**	**none_0.2**	**97.795**	**5.755**	**bilateralFilter_0.2**	**15.218**	**2.305**	**fastNlMeans10_0.2**	**15.588**	**2.390**
	upscaled_0.2	162.629	7.943	fastNlMeans10_0.2	100.143	6.317	fastNlMeans5_0.2	17.905	2.667	fastNlMeans15_0.2	18.228	2.543
	fastNlMeans15_0.2	198.394	9.648	fastNlMeans5_0.2	119.828	6.539	normalization_0.2	18.835	2.696	clahe_0.2	18.641	2.796
	medianBlur_0.2	199.725	9.193	bilateralFilter_0.2	122.282	6.439	medianBlur_0.2	24.917	2.960	bilateralFilter_0.2	24.111	2.911
	fastNlMeans10_0.2	213.956	10.606	normalization_0.2	124.070	6.688	fastNlMeans15_0.2	26.135	2.931	medianBlur_0.2	27.070	3.297
	fastNlMeans5_0.2	218.330	9.828	clahe_0.2	139.364	6.985	fastNlMeans10_0.2	26.838	3.073	fastNlMeans5_0.2	27.186	3.303
	clahe_0.2	242.249	10.083	fastNlMeans15_0.2	141.344	7.499	clahe_0.2	27.175	3.022	normalization_0.2	28.766	3.308
	none_0.2	257.525	10.790	upscaled_0.2	171.213	7.831	upscaled_0.2	35.068	3.221	none_0.2	31.779	3.690
	normalization_0.2	259.805	10.798	medianBlur_0.2	182.812	7.822	none_0.2	46.147	3.611	upscaled_0.2	32.955	3.672
	Experiment	Count		Experiment	Count	Total	Experiment	Count		Experiment	Count	Total
Per Confidence	bilateralFilter	2		bilateralFilter	0	2	**bilateralFilter**	**5**		**bilateralFilter**	**4**	**9**
	clahe	3		clahe	0	3	clahe	2		clahe	3	5
	fastNlMeans5	3		fastNlMeans5	0	3	**fastNlMeans5**	**5**		fastNlMeans5	3	8
	fastNlMeans10	2		**fastNlMeans10**	**1**	3	fastNlMeans10	3		fastNlMeans10	4	7
	fastNlMeans15	**5**		fastNlMeans15	0	**5**	fastNlMeans15	4		fastNlMeans15	3	7
	medianBlur	4		medianBlur	0	4	**medianBlur**	**5**		medianBlur	2	7
	normalization	1		normalization	0	1	**normalization**	**5**		normalization	3	8
	upscaled	**5**		upscaled	0	**5**	upscaled	4		upscaled	0	4
Outperforms Best none?	bilateralFilter	0		bilateralFilter	0	0	**bilateralFilter**	**Yes**		**bilateralFilter**	**Yes**	**2**
	clahe	0		clahe	0	0	clahe	0		clahe	0	0
	fastNlMeans5	0		fastNlMeans5	0	0	**fastNlMeans5**	**Yes**		fastNlMeans5	0	1
	fastNlMeans10	0		fastNlMeans10	0	0	**fastNlMeans10**	**Yes**		**fastNlMeans10**	**Yes**	**2**
	fastNlMeans15	**Yes**		fastNlMeans15	0	**1**	**fastNlMeans15**	**Yes**		fastNlMeans15	0	1
	medianBlur	**Yes**		medianBlur	0	**1**	**medianBlur**	**Yes**		medianBlur	0	1
	normalization	0		normalization	0	0	**normalization**	**Yes**		normalization	0	1
	upscaled	**Yes**		upscaled	0	**1**	**upscaled**	**Yes**		upscaled	0	1

8.4. Combining super resolution with other preprocessing techniques

In this section we want to analyze whether the super-resolution technique (upscaled) can benefit from combining it with the other preprocessing techniques. We examine the impact of applying one of the presented preprocessing techniques, both before and

Table 5
MSE, MAE, ACC0, ACC1, ACC2, and Time results for vehicle counts in JUANFRANCESBOSCA and PRIES parking lots, which include super-resolution (upscaled) in combination with other preprocessing techniques.

JUANFRANCESBOSCA											
							Time				
Experiment	MSE	MAE	ACC0	ACC1	ACC2	Total (s)	Matching %	Cutting %	Filter 1 %	Filter 2 %	Detecting %
YO_upscaled_bilateralFilter_0.1	**2.293**	**0.976**	0.415	**0.732**	**0.951**	5.955	29%	0%	56%	1%	14%
YO_upscaled_bilateralFilter_0.15	2.659	1.049	**0.439**	0.683	0.902	6.057	29%	0%	55%	1%	15%
YO_clahe_upscaled_0.1	2.683	1.122	0.415	0.634	0.878	6.686	29%	0%	0%	56%	15%
YO_fastNlMeans5_0.05	2.902	1.244	0.317	0.659	0.829	3.268	59%	0%	10%	–	31%
D2_clahe_0.2	**2.951**	**1.146**	**0.390**	0.683	0.854	2.126	91%	0%	0%	–	9%
D2_clahe_0.15	2.976	1.171	0.341	0.707	0.878	2.128	91%	0%	0%	–	9%
D2_normalization_0.15	3.171	1.317	0.244	0.659	0.878	2.112	92%	0%	0%	–	8%
YO_none_0.05	3.366	1.366	0.268	0.634	0.829	**2.705**	72%	0%	–	–	28%
D2_none_0.2	3.512	1.268	0.317	0.707	0.854	**2.106**	92%	0%	–	–	8%
PRIES											
Experiment	MSE	MAE	ACC0	ACC1	ACC2	Total (s)	Matching %	Cutting %	Filter 1 %	Filter 2 %	Detecting %
YO_clahe_0.15	**0.878**	**0.537**	**0.610**	**0.878**	0.976	**2.595**	67%	0%	0%	–	33%
YO_bilateralFilter_0.2	**0.878**	0.585	0.561	0.854	**1.000**	2.834	61%	0%	1%	–	37%
YO_bilateralFilter_0.15	1.000	0.610	0.561	0.854	0.976	2.816	62%	0%	1%	–	37%
YO_none_0.1	1.220	0.683	0.512	**0.878**	0.927	2.675	65%	0%	–	–	35%
D2_medianBlur_0.2	**2.024**	0.951	0.390	0.805	0.902	1.909	91%	0%	0%	–	9%
D2_normalization_0.2	2.098	1.024	0.341	0.780	0.878	1.908	91%	0%	0%	–	9%
D2_normalization_upscaled_0.2	2.098	0.976	0.415	0.732	0.927	6.018	29%	0%	0%	68%	3%
D2_none_0.2	2.146	**0.878**	**0.439**	**0.854**	0.902	**1.904**	91%	0%	–	–	9%

Table 6
MSE, MAE, ACC0, ACC1, ACC2, ACCFREE, and Time results for parking spaces counts in JUANFRANCESBOSCA and PRIES parking lots, which include super-resolution (upscaled) in combination with other preprocessing techniques.

JUANFRANCESBOSCA												
Experiment	MSE	MAE	ACC0	ACC1	ACC2	ACCFREE	Total (s)	Matching %	Cutting %	Filter 1 %	Filter 2 %	Detecting %
YO_upscaled_0.01	**0.293**	**0.293**	**0.707**	**1.000**	**1.000**	**1.000**	6.254	31%	0%	59%	–	9%
YO_medianBlur_upscaled_0.01	**0.293**	**0.293**	**0.707**	**1.000**	**1.000**	**1.000**	6.581	33%	0%	0%	51%	16%
YO_upscaled_fastNlMeans5_0.01	0.317	0.317	0.683	**1.000**	**1.000**	**1.000**	6.485	24%	0%	45%	17%	14%
D2_bilateralFilter_0.01	**0.317**	**0.317**	0.683	**1.000**	**1.000**	**1.000**	2.145	91%	0%	2%	–	8%
D2_normalization_upscaled_0.01	0.366	0.366	0.634	**1.000**	**1.000**	**1.000**	5.841	32%	0%	0%	65%	3%
D2_none_0.1	0.390	0.390	0.610	**1.000**	**1.000**	**1.000**	**2.102**	92%	0%	–	–	8%
YO_none_0.01	0.512	0.415	0.634	0.951	**1.000**	**1.000**	**2.539**	77%	0%	–	–	23%
PRIES												
Experiment	MSE	MAE	ACC0	ACC1	ACC2	ACCFREE	TIME (s)					
D2_bilateralFilter_upscaled_0.01	**0.171**	**0.171**	**0.829**	**1.000**	**1.000**	**0.951**	6.087	30%	0%	0%	67%	3%
YO_upscaled_fastNlMeans15_0.01	**0.195**	**0.195**	**0.805**	**1.000**	**1.000**	**0.951**	7.173	20%	0%	48%	18%	13%
D2_medianBlur_upscaled_0.01	0.220	0.220	0.780	**1.000**	**1.000**	0.878	6.038	28%	0%	0%	70%	2%
D2_fastNlMeans15_0.01	0.220	0.220	0.780	**1.000**	**1.000**	0.927	2.224	88%	0%	4%	–	8%
YO_medianBlur_upscaled_0.01	0.220	0.220	0.780	**1.000**	**1.000**	0.902	6.583	25%	0%	0%	62%	13%
YO_upscaled_0.01	0.244	0.244	0.756	**1.000**	**1.000**	0.878	6.995	25%	0%	59%	–	16%
D2_none_0.01	0.268	0.268	0.732	**1.000**	**1.000**	0.902	**1.903**	91%	0%	–	–	9%
YO_none_0.01	0.415	0.366	0.659	0.976	**1.000**	0.829	**2.646**	66%	0%	–	–	34%

after super-resolution, to assess its potential for improving the results. While various other combinations of preprocessing techniques are possible, we will defer this exploration to future research.

To avoid overwhelming the reader, in Tables 5–7, we only display the combinations that resulted in an improvement over the no-preprocessing baseline, excluding all others. Specifically, in these tables, we present the results of the best "none" (no preprocessing), the best preprocessing technique from the previous section (for comparison purposes), and the top 3 final preprocessing techniques for both YOLOv8 and Detectron2 models. The best "none" for both models is highlighted in orange, the other "none" is in blue, the best technique from the previous section is in green, and the techniques that utilized double preprocessing (super-resolution plus another preprocessing technique), which we are currently analyzing, are highlighted in yellow.

When it comes to *detecting the number of vehicles*, specifically for the JUANFRANCESBOSCA parking area (see Table 5), it is observed that, when using YOLOv8, combining super-resolution either with bilateral filtering or clahe yields better results than the best achieved with a single preprocessing method, YO_fastNlMeans5_0.05 (2.902). In this case, the lowest MSE is achieved using **YO_upscaled_bilateralFilter**_0.1 (2.293) for YOLOv8. For Detectron2, the best MSE result remains as D2_clahe_0.2 (2.951). However, in the PRIES parking area (see Table 5), no combination leads to an improvement in the best MSE obtained for vehicle detection, which is still YO_clahe_0.15 (0.878).

Table 7
MSE, MAE, ACC0, ACC1, ACC2, and Time results for length of parking spaces in JUANFRANCESBOSCA and PRIES parking lots, which include super-resolution (upscaled) in combination with other preprocessing techniques.

JUANFRANCESBOSCA											
Experiment	MSE	MAE	ACC0	ACC1	ACC2	Total (s)	Matching %	Cutting %	Filter 1 %	Filter 2 %	Detecting %
D2_none_0.01	**44.129**	4.118	0.379	0.713	0.874	**2.111**	92%	0%	–	–	8%
YO_upscaled_fastNlMeans5_0.01	**44.251**	**3.618**	0.482	**0.807**	0.867	6.485	24%	0%	45%	17%	14%
D2_fastNlMeans15_0.01	45.638	**3.781**	0.400	**0.788**	**0.894**	2.459	79%	0%	14%	–	7%
YO_normalization_upscaled_0.01	45.966	3.999	0.460	0.724	**0.885**	6.863	27%	0%	0%	55%	18%
YO_bilateralFilter_upscaled_0.01	46.476	3.751	0.500	0.750	0.881	6.299	32%	0%	0%	52%	15%
YO_upscaled_0.01	48.076	3.722	0.524	0.756	0.866	6.254	31%	0%	59%	–	9%
D2_bilateralFilter_0.01	49.216	3.796	0.412	0.776	**0.894**	2.145	91%	0%	2%	–	8%
YO_none_0.01	59.843	4.193	0.453	0.767	0.872	**2.539**	77%	0%	–	–	23%
PRIES											
Experiment	MSE	MAE	ACC0	ACC1	ACC2	TIME (s)					
YO_upscaled_0.01	**6.261**	**1.193**	0.726	0.903	0.984	6.995	25%	0%	59%	–	16%
YO_upscaled_fastNlMeans10_0.01	6.451	1.322	**0.790**	0.871	0.984	6.641	21%	0%	50%	18%	11%
YO_upscaled_bilateralFilter_0.01	6.646	1.371	0.714	**0.921**	0.984	6.547	25%	0%	60%	1%	14%
D2_fastNlMeans10_0.01	**7.324**	1.684	0.538	0.908	**1.000**	2.229	88%	0%	4%	–	7%
D2_bilateralFilter_0.01	7.730	1.683	0.619	0.873	**1.000**	1.943	89%	0%	2%	–	9%
D2_upscaled_bilateralFilter_0.01	8.171	1.674	0.581	0.855	**1.000**	5.772	28%	0%	68%	1%	3%
D2_none_0.01	8.838	1.780	0.541	0.902	0.984	**1.903**	91%	0%	–	–	9%
YO_none_0.01	12.463	2.260	0.547	0.797	0.984	**2.646**	66%	0%	–	–	34%

For the task of *detecting parking spaces*, particularly in the JUANFRANCESBOSCA parking area (see Table 6), we can observe that the combination of super-resolution techniques with image preprocessing does not improve the best MSE obtained by YO_upscale_0.01 (0.293). However, there is an improvement in the case of the PRIES parking area (see Table 6), where **D2_bilateralFilter_upscaled**_0.01 achieves the best MSE result (0.171), surpassing the previous best result of D2_fastNlMeans15_0.01 (0.220). YO_upscaled_fastNlMeans15_0.01 (0.195) is also close to that result. In this case, both YOLOv8 and Detectron2 were able to benefit from combining preprocessing.

For *measuring the length of parking spaces* in the JUANFRANCESBOSCA parking area (see Table 7), the combination of the mentioned techniques does not improve upon the best MSE result obtained by D2_none_0.2 (44.129). The same holds for the PRIES parking area (see Table 7), where the best MSE result is achieved by YO_upscaled_0.01 (6.261), and the combination of techniques does not surpass it.

In summary, the combination of the super-resolution technique (upscaled) with other preprocessing methods yields diverse outcomes across models and parking areas. In the JUANFRANCESBOSCA parking area, combining super-resolution with bilateral filtering results in an improved vehicle detection MSE using YOLOv8, whereas no enhancement is observed in the PRIES parking area.

In PRIES, three combinations show improvement over the previously achieved best MSE results, using YOLOv8 or Detectron2 when counting parking spaces. Notably, the same combination technique that produced the best result in vehicle detection, albeit in reverse order, attains the optimal MSE for parking space detection in PRIES when using Detectron2. Conversely, no combination demonstrated improvement in JUANFRANCESBOSCA.

When measuring the length of parking spaces, the combined techniques do not improve the best MSE results obtained by individual preprocessing methods for both parking areas.

The consistently successful technique combination in most parking areas and models is using the upscaled with the bilateral filtering technique combined in both ways. Further exploration of the parametrization of this combination can be studied to achieve even better results. Additionally, exploring additional combinations of preprocessing techniques before inference on these tasks could enhance these results. In conclusion, these findings suggest that the effectiveness of combining super-resolution with other preprocessing techniques is task-specific and dependent on the characteristics of the parking area. However, the promising results indicate the potential for further improvement in outcomes by combining preprocessing techniques, which needs more exploration in future research.

8.5. Measuring accuracy and time

While understanding the models' errors is insightful for analyzing and comparing their performance, as demonstrated in the previous sections, our ultimate goal is to evaluate their accuracy and execution time to determine their real-world applicability. This section delves into this aspect, focusing on preprocessing techniques that demonstrated improvement over no preprocessing (none). As in the previous section, we exclude those that did not to avoid overwhelming the reader (see Tables 5, 6, and 7).

We use three accuracy metrics, ACC0, ACC1, and ACC2, representing accuracy with margin tolerances of 0, 1, and 2 vehicles or parking spaces, respectively. However, the previously mentioned margin tolerances differ when measuring parking space length (see Table 7). In that case, for ACC0, the tolerance is set at 1 meter. For ACC1, it is defined as the minimum length of a car we selected (4.3 m) plus 1 meter, calculated as $4.3+1=5.3$. For ACC2, the tolerance is $4.3\cdot 2+1=9.6$. Using these margin tolerances, we consider

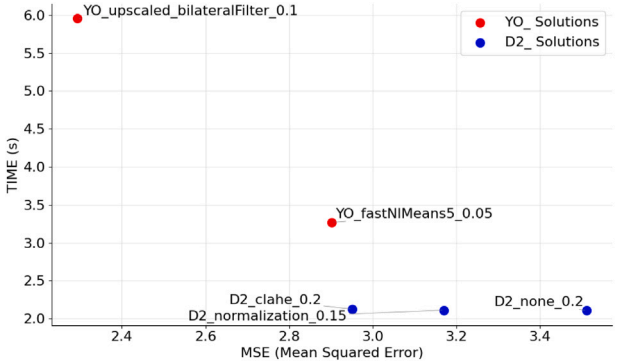

Fig. 15. Pareto Front. Vehicles Count. JUANFRANCESBOSCA.

a prediction accurate (per image) if it falls within the specified error margins. In other words, achieving 100% accuracy is possible if all the detected elements are within the defined margins in all the images. Additionally, we introduce the metric ACCFREE in Table 6, indicating whether at least one parking space is available.

Regarding the *accuracy in detecting the number of parked vehicles in the parking area* (see Table 5), there is suboptimal accuracy when no margin of error is used to detect the total number of vehicles. However, we achieve accuracy percentages of 73.2% and 87.8% when allowing a margin of error of one vehicle, and 95.1% and 100% with a margin of error of two vehicles for the JUANFRANCESBOSCA and PRIES parking areas, respectively. These results suggest that while detecting the number of parked vehicles may be useful, it may not be the most accurate technique due to the variability in vehicle sizes.

The *accuracy percentage in predicting available parking spaces* is 70.7% and 82.9%, without any margin of error for the JUANFRANCESBOSCA and PRIES parking areas, respectively (see Table 6). If we allow a margin of error of 1 space, the accuracy percentage becomes 100%. Moreover, we can predict with almost 100% accuracy whether there will be available parking spaces in the JUANFRANCESBOSCA and PRIES parking areas, respectively (ACCFREE). These results are promising and allow accurate predictions of both parking space availability and the number of parking spaces, considering each parking space's length (in meters). Moreover, knowing the length of parking spaces enables us to provide personalized information to the user regarding parking availability for their specific vehicle.

Additionally, when *measuring the length of available parking spaces*, we can do so with a margin of error of 5.3 m and an accuracy percentage of 80.7% and 92.1% for the JUANFRANCESBOSCA and PRIES parking areas, respectively. If we increase the margin of error to 9.6 m, the accuracy percentage increases to 89.4% and 100% (see Table 7). These results are promising for estimating the number of vehicles that can fit in those spaces depending on their sizes, providing a better assessment of parking availability when the user arrives at the parking area.

In summary, the accuracy results show that counting available parking spaces and measuring their lengths are more accurate than counting the vehicles in the parking area. Also, these techniques can be adapted to specific user vehicle types and sizes when looking for available parking.

Concerning *execution times* (see Tables 5, 6, and 7), Detectron2 demonstrates slightly faster processing times than YOLOv8 for the selected models, specifically *X101-FPN* and *yolov8x*, respectively. Analyzing the time required for different processes (see Table 5), the super-resolution filter is the slowest, and the matching process takes longer than the inference itself.

In multi-objective optimization, the Pareto front [23] consists of a set of non-dominated solutions considered optimal, as any enhancement in one objective comes at the cost of at least one other. These Pareto fronts provide decision-makers with valuable insights, enabling them to select the most suitable model for their specific problem. Whether prioritizing optimal accuracy or constrained by time considerations, decision-makers can make informed choices based on the trade-offs between accuracy and speed.

Detectron2 consistently achieves non-dominated solutions on the Pareto front (see Figs. 15, 16, 17, 18, 19, and 20) with the shortest computation time in all experiments. In contrast, YOLOv8 typically achieves lower MSE (in four of the six cases) by effectively leveraging preprocessing techniques, albeit at the cost of a longer computation time. The figures illustrate the pseudo-optimal Pareto fronts composed of non-dominated solutions obtained during our research, effectively addressing both primary objectives: enhancing accuracy and reducing processing time.

9. Adoption potential in smart cities

Several cities have adopted AI-powered parking solutions, highlighting the growing role of computer vision, IoT, and deep learning in urban mobility. For example, Barcelona [43] uses embedded IoT sensors in parking spots to monitor occupancy and guide drivers to available spaces. San Francisco deployed *SFpark* [44], which dynamically adjusts parking rates using real-time sensor data, optimizing availability rather than detecting individual spaces. Amsterdam [45] applies machine learning to historical data to predict and optimize enforcement patrols, reducing illegal parking. Los Angeles [46] deploys AI-powered cameras mounted

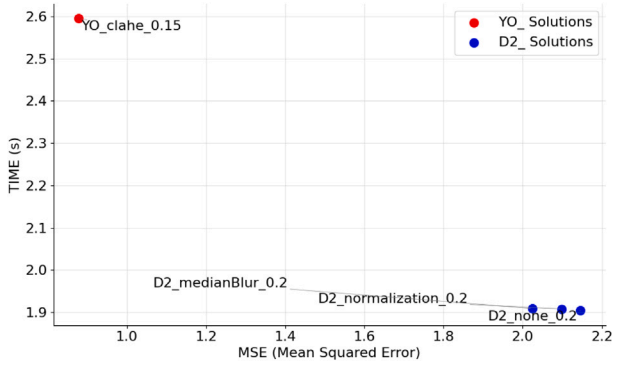

Fig. 16. Pareto Front. Vehicles Count. PRIES.

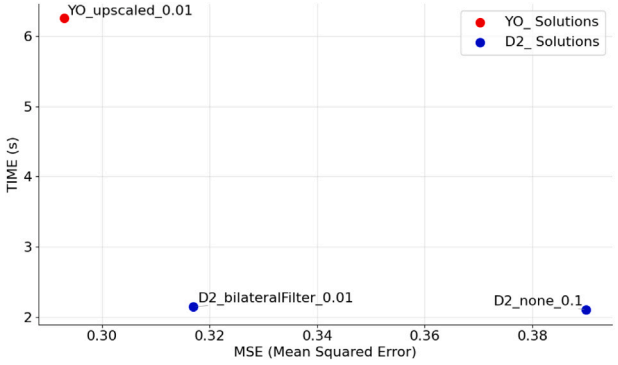

Fig. 17. Pareto Front. Parking Spaces Count. JUANFRANCESBOSCA.

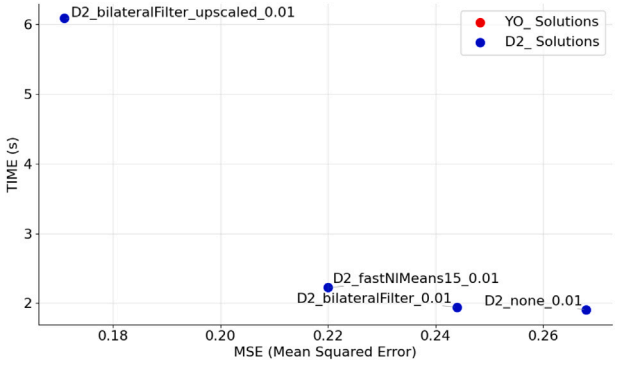

Fig. 18. Pareto Front. Parking Spaces Count. PRIES.

on buses to detect illegal parking in bus lanes, facilitating enforcement rather than real-time parking monitoring. Paris [47] leverages deep learning models applied to traffic camera feeds to detect illegal parking violations and optimize enforcement.

Despite these advances, none of these solutions directly detect and monitor available parking spaces in real time using low-resolution, publicly available traffic cameras. Existing implementations primarily focus on pricing optimization (San Francisco), enforcement patrol optimization (Amsterdam), sensor-based occupancy detection (Barcelona), or illegal parking identification (Paris and Los Angeles). However, these methods typically require sensor installations or high-resolution camera systems, increasing costs and limiting scalability. In contrast, our approach provides a cost-effective, scalable, and infrastructure-independent alternative that can be deployed across multiple cities without requiring additional hardware investments.

In addition to solving an unmet challenge, our method provides several advantages over existing AI-driven parking solutions:

1. *No dedicated infrastructure required:* Unlike sensor-based or high-resolution camera systems, our method repurposes publicly available traffic cameras, reducing costs and eliminating maintenance overhead.

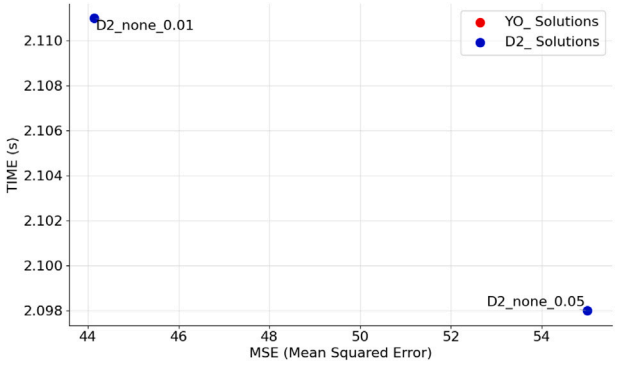

Fig. 19. Pareto Front. Length of Parking Spaces. JUANFRANCESBOSCA.

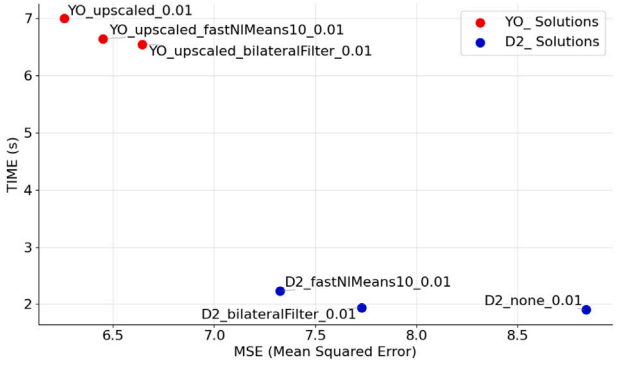

Fig. 20. Pareto Front. Length of Parking Spaces. PRIES.

2. *Adaptability to real-world conditions:* While most solutions rely on fixed-position, high-resolution cameras, our approach is designed to process low-quality, moving traffic camera feeds, handling variations in angles, lighting, and resolution.
3. *Cross-city scalability:* Instead of requiring city-specific datasets and frequent retraining, our method leverages pre-trained deep learning models, enabling broader deployment with minimal adaptation.

To facilitate real-world adoption, we propose the following strategies:

1. *Pilot Programs:* Conduct small-scale pilot studies in selected urban areas to validate the effectiveness and adaptability of the method.
2. *Public-Private Partnerships:* Collaborate with technology firms specializing in smart city solutions to enhance deployment and scalability.
3. *Standardization and Policy Advocacy:* Work with policymakers to establish ethical AI guidelines for parking management, ensuring transparency and compliance with data protection regulations.

By addressing these adoption factors, our approach has the potential to transform urban parking management, contributing to the development of more efficient, sustainable, and intelligent cities.

10. Conclusions and future work

In this work, we addressed the challenge of on-street parking space localization in smart cities using publicly available cameras primarily used for traffic monitoring. We proposed a novel dataset-independent method for accurately locating on-street parking spaces of variable size using low-quality images captured by public moving traffic cameras.

The experimental results demonstrated our pipeline's effectiveness in accurately detecting parking spaces. We also gained valuable insights into developing a cost-effective system for identifying on-street parking spaces in urban environments by utilizing openly available Internet data resources (RQ1). We have demonstrated the feasibility of this approach, guaranteeing its scalability and adaptability to diverse urban contexts since this proposal is dataset-independent, avoiding costly manual labeling processes and model training (RQ2). Also, this approach leverages existing urban resources without extra infrastructure costs, which facilitates global deployment independent of the city (RQ3). Several key conclusions can be drawn from our study:

The application of image preprocessing techniques, such as bilateral filtering, fast non-local means denoising, and super-resolution, significantly improved the accuracy of parking space detection. These techniques enhanced the image clarity, allowing the object detection models to perform more effectively, especially in scenarios with low-quality images and distant parking areas. However, these techniques were more effective in the YOLOv8 model than in the Detectron2 model.

Our experiments showed that counting parking spaces was more accurate than counting the total number of vehicles in the parking area. Counting the number of vehicles may not be the most suitable approach due to the challenges posed by the variable size of the vehicles, low image quality, camera movements, and distance from the parking spaces. However, determining the availability of parking spaces using image processing techniques and confidence levels provided more accurate results.

Combining super-resolution techniques with other image preprocessing techniques improved vehicle detection in some cases but did not consistently outperform individual preprocessing techniques. However, it showed promise in improving parking space detection, especially in scenarios where parking areas were distant from the camera. Other combinations of preprocessing methods could be explored in future work, as well as their integration with other optimization methods to achieve the most optimal preprocessing for our specific use case.

The pipeline achieved a high level of accuracy in determining parking space availability, with the accuracy percentage exceeded 95%. Estimating the length of available parking spaces improved the accuracy of parking resource utilization predictions, making it valuable for users seeking available parking spaces.

Except for counting vehicles, which we have seen is not the most appropriate technique to obtain the most accurate measurements, in the other metrics of counting the number of parking spaces and measuring their length, both models exhibit very similar performances, with YOLOv8 slightly outperforming Detectron2 in some experiments but Detectron2 achieving the non-dominated solutions with lower times in all experiments which is interesting for processing in real time. However, the YOLOv8 model benefits more from image preprocessing techniques than Detectron2.

Based on our analysis of the chosen models, namely *X101-FPN* and *yolov8x*, Detectron2 demonstrates slightly faster processing times compared to YOLOv8. Upon closer inspection of the various processes, it is apparent that the super-resolution filter is the slowest, and the matching process takes longer than the actual inference.

Overall, our proposed pipeline provides a valuable tool for urban planners and parking managers to optimize parking resource utilization, enhance urban mobility, and reduce traffic congestion. Combining pre-trained deep learning techniques, image processing, and confidence level adjustments enables accurate parking space detection without requiring intensive data labeling or model retraining. Furthermore, some preprocessing techniques perform better on certain types of images than others. Therefore, these techniques could be adjusted according to the time of day and weather conditions (e.g., nighttime images, excessive glare from vehicle lights, rain, etc.). Additionally, some techniques excel in detecting distant vehicles, while others perform better with close-range ones. Hence, we consider the possibility of conducting parallel detections using different techniques and combining their results to achieve a more accurate parking line detection.

Future work could focus on further improving the performance of the pipeline in scenarios with challenging lighting conditions, adverse weather (e.g., rain, snow), or occlusions. Additionally, we plan to explore tailored preprocessing techniques to enhance image quality in low-light scenarios, ensuring robustness across varying environmental conditions. Another promising direction is the real-time implementation of the pipeline for continuous monitoring of parking space availability in smart cities. Furthermore, integrating the pipeline with real-time data from parking meters and navigation systems could enhance the overall parking management system, providing real-time information to drivers and optimizing urban parking utilization.

CRediT authorship contribution statement

José Ángel Morell: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Gabriel Luque:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition. **Enrique Alba:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is funded by PID 2020-116727RB-I00 (HUmove) funded by MCIN/AEI/10.13039/501100011033; and TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme. Funding for open access charge: Universidad de Málaga / CBUA. The authors thank the Supercomputing and Bioinnovation Center (SCBI) for their provision of computational resources and technical support. The views expressed are purely those of the writer and may not in any circumstances be regarded as stating an official position of the European Commission.

Data availability

Data will be made available on request.

References

[1] Urbanization — ourworldindata.org, 2023, https://ourworldindata.org/urbanization. (Accessed 02 August 2023).
[2] A. Camero, E. Alba, Smart city and information technology: A review, Cities 93 (2019) 84–94.
[3] J.Á. Morell, Z.A. Dahi, F. Chicano, G. Luque, E. Alba, Time series forecasting for parking occupancy: Case study of malaga and birmingham cities, in: International Conference on Optimization and Learning, Springer, 2023, pp. 368–379.
[4] X. Xiao, Z. Peng, Y. Lin, Z. Jin, W. Shao, R. Chen, N. Cheng, G. Mao, Parking prediction in smart cities: A survey, IEEE Trans. Intell. Transp. Syst. (2023).
[5] W. Shao, Y. Zhang, B. Guo, K. Qin, J. Chan, F.D. Salim, Parking availability prediction with long short term memory model, in: Green, Pervasive, and Cloud Computing: 13th International Conference, GPC 2018, Hangzhou, China, May 11-13, 2018, Revised Selected Papers 13, Springer, 2019, pp. 124–137.
[6] X. Ding, R. Yang, Vehicle and parking space detection based on improved yolo network model, in: Journal of Physics: Conference Series, vol. 1325, IOP Publishing, 2019, 012084, 1.
[7] D. Acharya, W. Yan, K. Khoshelham, Real-time image-based parking occupancy detection using deep learning., Research@ Locate 4 (2018) 33–40.
[8] C. Huang, S. Yang, Y. Luo, Y. Wang, Z. Liu, Visual detection and image processing of parking space based on deep learning, Sensors 22 (17) (2022) 6672.
[9] L.-C. Chen, R.-K. Sheu, W.-Y. Peng, J.-H. Wu, C.-H. Tseng, Video-based parking occupancy detection for smart control system, Appl. Sci. 10 (3) (2020) 1079.
[10] S.N.R. Mettupally, V. Menon, A smart eco-system for parking detection using deep learning and big data analytics, in: 2019 SoutheastCon, IEEE, 2019, pp. 1–4.
[11] Z. Xu, X. Tang, C. Ma, R. Zhang, Research on parking space detection and prediction model based on CNN-LSTM, IEEE Access (2024).
[12] P.R.L. de Almeida, J.H. Alves, R.S. Parpinelli, J.P. Barddal, A systematic review on computer vision-based parking lot management applied on public datasets, Expert Syst. Appl. 198 (2022) 116731.
[13] V. Paidi, H. Fleyeh, J. Håkansson, R.G. Nyberg, Smart parking sensors, technologies and applications for open parking lots: A review, IET Intell. Transp. Syst. 12 (8) (2018) 735–741.
[14] C. Ma, X. Huang, J. Li, A review of research on urban parking prediction, J. Traffic Transp. Eng. (Engl. Edition) (2024).
[15] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? Adv. Neural Inf. Process. Syst. 27 (2014).
[16] P. Meduri, A. Estebanez, A brief review of convolutional neural networks based solutions for smart parking systems, in: 2018 International Conference on Computational Science and Computational Intelligence, CSCI, IEEE, 2018, pp. 454–457.
[17] R. Bohush, P. Yarashevich, S. Ablameyko, T. Kalganova, Extraction of image parking spaces in intelligent video surveillance systems, Mach. Graph. Vis. 27 (2018).
[18] R.M. Nieto, A. Garcia-Martin, A.G. Hauptmann, J.M. Martinez, Automatic vacant parking places management system using multicamera vehicle detection, IEEE Trans. Intell. Transp. Syst. 20 (3) (2018) 1069–1080.
[19] J. Correia, D. Lopes, L. Vieira, N. Rodriguez-Fernandez, A. Carballal, J. Romero, P. Machado, Experiments in evolutionary image enhancement with ELAINE, Genet. Program. Evolvable Mach. 23 (4) (2022) 557–579.
[20] GitHub - cvg/LightGlue: LightGlue: Local feature matching at light speed (ICCV 2023) — github.com, 2023, https://github.com/cvg/LightGlue. (Accessed 06 August 2023).
[21] P. Lindenberger, P.-E. Sarlin, M. Pollefeys, LightGlue: Local feature matching at light speed, 2023, arXiv preprint arXiv:2306.13643.
[22] P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: Learning feature matching with graph neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4938–4947.
[23] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Ev. Comp. 6 (2) (2002) 182–197.
[24] F. Banterle, M. Corsini, P. Cignoni, R. Scopigno, A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain, in: Computer Graphics Forum, vol. 31, Wiley Online Library, 2012, pp. 19–32, 1.
[25] B. Justusson, Median filtering: Statistical properties, in: Two-Dimensional Digital Signal Prcessing II: Transforms and Median Filters, Springer, 2006, pp. 161–196.
[26] A. Buades, B. Coll, J.-M. Morel, A non-local algorithm for image denoising, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'05, vol. 2, Ieee, 2005, pp. 60–65.
[27] A. Dauwe, B. Goossens, H.Q. Luong, W. Philips, A fast non-local image denoising algorithm, in: Image Processing: Algorithms and Systems VI, vol. 6812, SPIE, 2008, pp. 324–331.
[28] L.F. Rodrigues, M.C. Naldi, J.F. Mari, Comparing convolutional neural networks and preprocessing techniques for HEp-2 cell classification in immunofluorescence images, Comput. Biol. Med. 116 (2020) 103542.
[29] G.-h. Yun, S.-j. Oh, S.-c. Shin, Image preprocessing method in radiographic inspection for automatic detection of ship welding defects, Appl. Sci. 12 (1) (2021) 123.
[30] G. Yadav, S. Maheshwari, A. Agarwal, Contrast limited adaptive histogram equalization based enhancement for real time video system, in: 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI, IEEE, 2014, pp. 2392–2397.
[31] Super resolution in opencv — learnopencv.com, 2023, https://learnopencv.com/super-resolution-in-opencv. (Accessed 06 August 2023).
[32] GitHub - vashiegaran/opencv-super-resolution — github.com, 2023, https://github.com/vashiegaran/Opencv-Super-Resolution. (Accessed 06 August 2023).
[33] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.
[34] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Fast and accurate image super-resolution with deep laplacian pyramid networks, IEEE Trans. Pattern Anal. Mach. Intell. 41 (11) (2018) 2599–2613.
[35] G. Jocher, A. Chaurasia, J. Qiu, YOLO by ultralytics, 2023, URL: https://github.com/ultralytics/ultralytics.
[36] GitHub - facebookresearch/detectron2: Detectron2 is a platform for object detection, segmentation and other visual recognition tasks. — github.com, 2023, https://github.com/facebookresearch/detectron2. (Accessed 06 August 2023).
[37] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2, 2019, https://github.com/facebookresearch/detectron2.
[38] Opencv: Camera calibration and 3D reconstruction — docs.opencv.org, 2023, https://docs.opencv.org/3.4/d9/d0c/group_calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780. (Accessed 06 August 2023).
[39] E. Dubrofsky, Homography Estimation, (Diplomová Práce), vol. 5, Univerzita BritskÉ Kolumbie, Vancouver, 2009.
[40] R. Raguram, J.-M. Frahm, M. Pollefeys, Exploiting uncertainty in random sample consensus, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 2074–2081.
[41] P.J. Rousseeuw, Least median of squares regression, J. Amer. Statist. Assoc. 79 (388) (1984) 871–880.

[42] Cámaras — movilidad.malaga.eu, 2023, https://movilidad.malaga.eu/es/servicios/camaras-de-trafico. (Accessed 06 August 2023).
[43] Harvard Data-Smart City, How smart city Barcelona brought the internet of things to life, 2021, URL https://datasmart.hks.harvard.edu/news/article/how-smart-city-barcelona-brought-the-internet-of-things-to-life-789[datasmart.hks.harvard.edu/news/article/how-smart-city-barcelona-brought-the-internet-of-things-to-life-789]. (Accessed 08 March 2025).
[44] G. Pierce, D. Shoup, Sfpark: Pricing parking by demand, in: Parking and the City, Routledge, 2018, pp. 344–353.
[45] Egis Group, Amsterdam smart parking, 2022, URL https://www.egis-group.com/projects/amsterdam-smart-parking. (Accessed 08 March 2025).
[46] Los Angeles Metro, Bus lane enforcement program, 2023, URL https://www.metro.net/projects/ble-program/. (Accessed 08 March 2025).
[47] Artelys, Artificial Intelligence for Parking Control and Data in Paris, 2023, URL https://www.artelys.com/news/artificial-intelligence-parking-control-data-paris/. (Accessed 08 March 2025).