PARKING LOT OCCUPANCY DETECTION

EMMANUEL KWAME ASIEDU
Index: 24259805
ME521
MEEE

January 2025

1 Introduction

The assignment given was to design a parking lot occupancy identification system to detect occupied and unoccupied parking spaces. To do this, some research was done into existing parking lot occupancy algorithms. From the vast research done, three sources were identified.

Several methods have been used to detect parking lots. Kommey, Addo, and Agbemenu designed a parking lot system that uses a reference image of the empty parking lot and compares it to a processed image of the parking lot with a vehicle in its place. This project works well only if a reference image of an empty parking space is provided, which poses an issue if the system is supposed to automatically pick up on the detection once installed. Computer vision engineer addresses this problem by actively selecting the parking lot space and determining which spaces are occupied and which are not. This manual process of parking lot detection is time consuming and needs high correction of parking lot size for various parking spaces. Ashqer and Bikdash presents a method of detection the capacity of a parking lot using MATLAB. Their work was effective because it was able to detect parking spaces with vehicles in them. Also, it was able to automatically rotate the frame of the image feed to allow for a better detection scheme. Therefore, with inspiration from [1][2] and [3], the parking lot occupancy detection algorithm has been designed to identify and classify occupied and unoccupied parking spaces.

2 Literature Review

2.1 A Smart Image Processing-Based System For Parking Space Vacancy Management

The authors propose a smart parking lot management system using image processing techniques. The system starts by storing an empty parking lot image, then obtains the coordinates of marked parking slots and their boundaries. The images are grayscaled, normalized, smoothed, and binarized to enhance object detection accuracy. Canny edge detection is performed using a Gaussian filter, Sobel-Feldman gradient operator, non-maximum suppression, and double thresholding. Edge tracking is done by hysteresis, and morphoogical dilation is performed using structuring elements. The system then marks parking spaces with bounding boxes based on the obtained coordinates.

2.2 Parking Lot Space Detection Based On Image Processing

The authors used MATLAB software to detect the capacity of a parking lot. They processed the image using various functions, including imrotate, hough transform, grayscale, binarization, area opening, dilation, closing, morphological remove, edge detection, imcomplement, regionprops, bounding box, and bwboundaries. The image was converted to grayscale, enhancing or suppressing specific colors, and the boundaries of foreground pixels were enlarged. The image was then combined, edge detected, and dilated to count parking spaces. The authors also used regionprops to measure rectangle boxes and bwboundaries to trace the exterior boundaries of the parking spaces.

3 Approach/Methodology

The method used was to first generate a reference image of an empty parking lot so that the lots could be used in the identification.

In designing the occupancy detection algorithm, I first acquired the image and converted it to an RGB color scheme. The image was converted to grayscale and two blurring approaches were used on the image in sequence. Usually, Gaussian blur is mostly used in image processing as the go-to blurring technique. However, I wanted to detect images where the edges were not excessively blurred. The median blur can effectively remove salt-and-pepper noise or small intensity changes while preserving the sharp edges. The bilateral blurring reduces noise while maintainig edge sharpness.

Using canny edge detection, the edges of the images are detected. The values, although general for all the images processed, allowed for the edges of all the images to be processed without tweaking the canny edge values.

In order to get the parking spaces detected, dilation was performed on the edges detected. After dilating the images, erosion is done to get a refined thickness and reduce noise. Small holes and small objects were removed to reduce the extra detail in the image. Then morphological opening was done to remove small noise or artifacts while preserving the structure of significant edges. After morphological closing was performed to close small gaps or holes within detected edges to create more coherent shapes. The image was dilated again to increase the edges. Then small holes and objects were removed again. Morphological closing was performed to fill the entire screen. This will be used in skeletonizing the closed image.

The skeletonization process is applied to the closing image using the skeletonize function. Skeletonization reduces the binary image to its essential structure, producing a thin representation of the shapes in the image while retaining their topology. Then boundary is plotted to observe the refined edges.

Then edge subtraction is done to get the parking spaces.

The subtracted edges are counted and their coordinates are saved. This information is then used in the parking lot occupancy algorithm which draws the subtracted edges over the original image and does an edge count in the image to determine which of the boxes contain a higher amount of edges and marks them with green or red.

4 Iteration In Approach

In designing the algorithm, the following were changed for different images:

- The neigbhourhood size for the median blur, where smaller values kept more details in the image compared to larger values.
- The diameter, color of each pixel and spatial distance for each pixel. This allowed control of how the colors in the neighbourhood affected each pixel and how far the pixels influenced each other.
- The canny edge detection was kept relatively the same across the images.
- The iterations for the dilations and errosions. This was done to close opened lines in the image after canny edge was performed.
- The minimum size for the remove small holes and remove small objects functions. This was to remove small objects and fill in holes in the image so that the parking spaces could be detected.
- The kernel sizes for the morphological opening and closing. This was done to further fill in gaps in the image.
- The minimum image edge thresholding value for the occupancy detection algorithm.

5 Results and Observation

The code contained two identification functions. One that adjusts the boxes and another that just draws the boxes over the image. For some of the images, the best option was to draw over the image.

The occupancy detection algorithm worked but did not accurately detect all the spaces and in some images, detected non parking spaces. It also flagged a few empty spaces as occupied due to the processing done on those spaces.

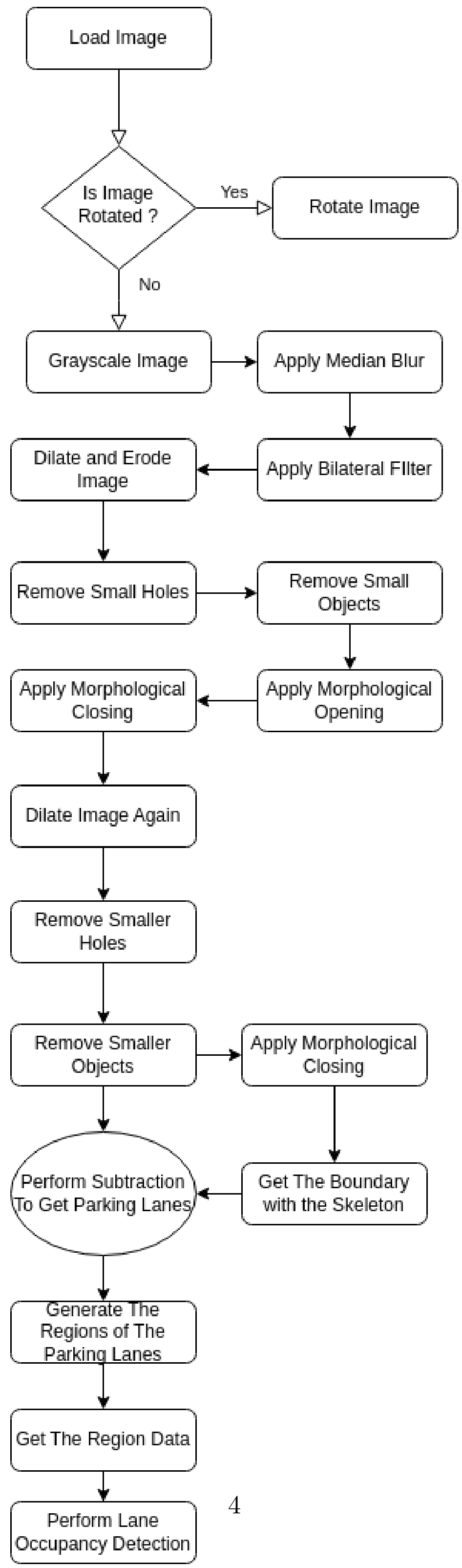

Figure 1: Iteration Process

Figure 2: Image 1

Improvements could be made to the algorithm to further identify spaces correctly and draw over them correctly.

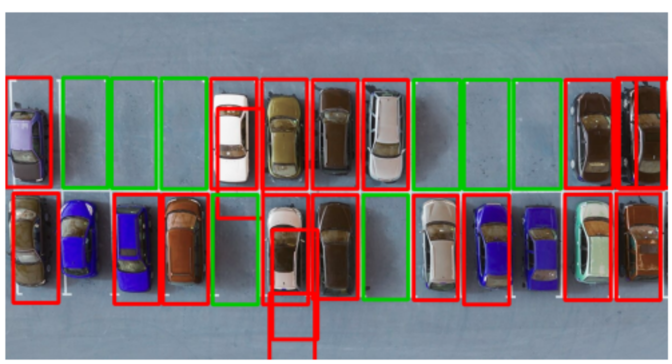

Figure 3: Image 2

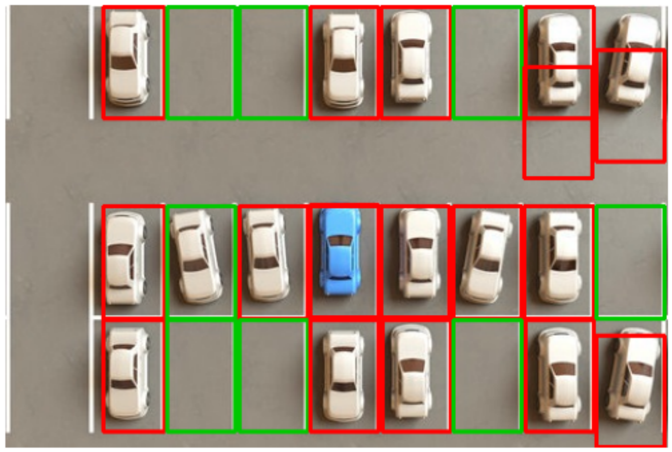

Figure 4: Image 3

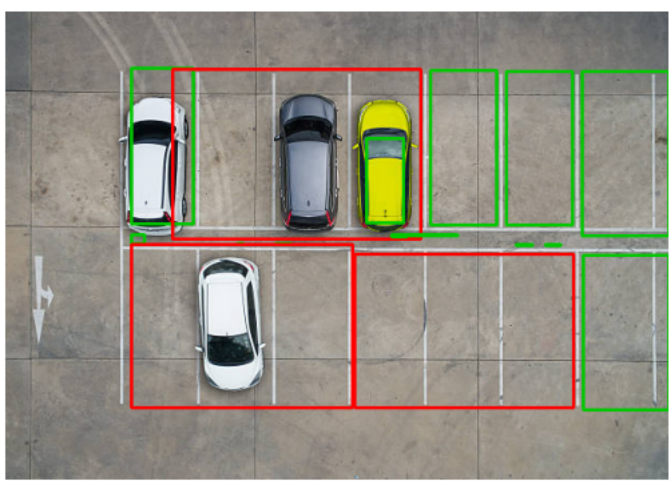

Figure 5: Image 4

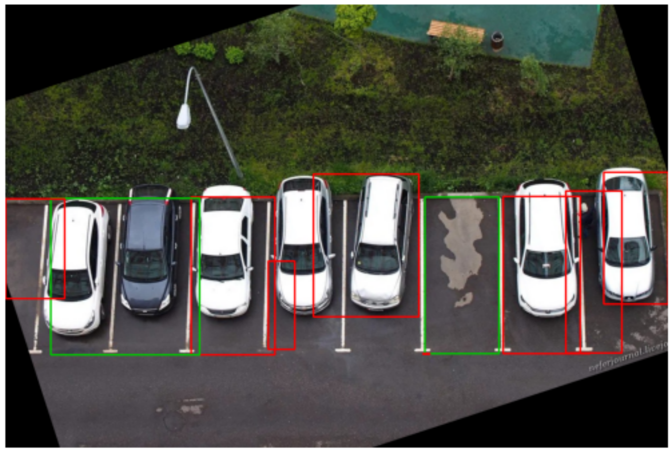

Figure 6: Image 5

Figure 7: Image 6

Figure 8: Image 7

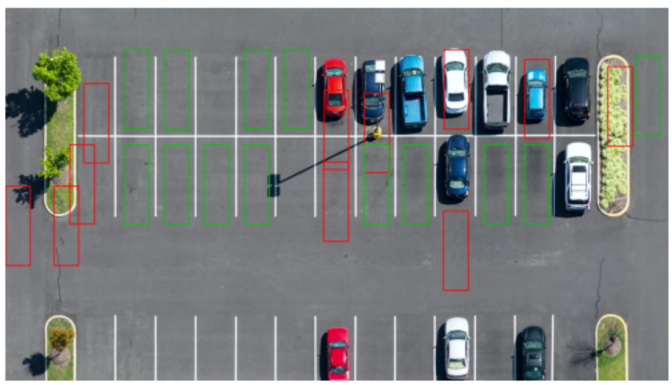

Figure 9: Image 8

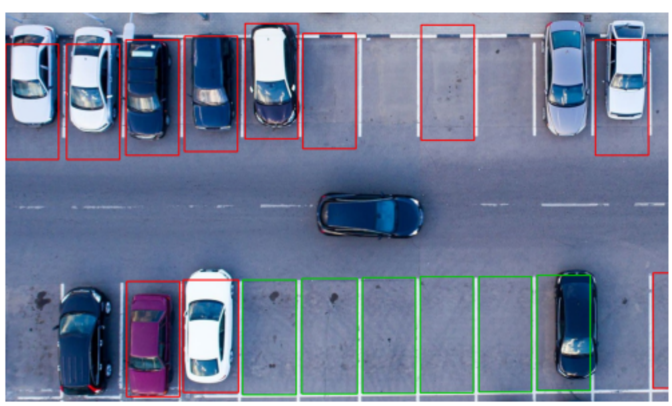

Figure 10: Image 10

6 References

References

Ashqer, M. I., & Bikdash, M. (2019). Parking lot space detection based on image processing. *2019 SoutheastCon*, 1–6.

Computer vision engineer. (2023). *Parking spot detection and counter — computer vision engineer.* `https://youtu.be/F-884J2mnOY`. (Accessed: 2025-01-06)

DigitalSreeni. (2020a). *22 - denoising microscope images in python.* `https://youtu.be/2PrzKWkqOtU`. (Accessed: 2025-01-06)

DigitalSreeni. (2020b). *26 - denoising and edge detection using opencv.* `https://youtu.be/-Qnb8Wv2p1c`. (Accessed: 2025-01-06)

DigitalSreeni. (2020c). *27 - clahe and thresholding using opencv in python.* `https://youtu.be/WQK`$_{o}OWW5Zo.(Accessed: 2025-01-06)$

Kommey, B., Addo, E. O., & Agbemenu, A. S. (2018). A smart image processing-based system for parking space vacancy management. *International Journal of Computer Applications*, *182*(5), 1–6.

1. (Kommey et al., 2018)
2. (DigitalSreeni, 2020c)
3. (Kommey et al., 2018)
4. (DigitalSreeni, 2020a)
5. (Computer vision engineer, 2023)
6. (DigitalSreeni, 2020b)