Car Parking Occupancy Slot Detection

Name: Maryam Khaled Youssef

ID: 185629

Supervisor: Dr. Sherif Hussein

Faculty of Engineering, Computer Department,

The British University in Egypt, El Sherouk City, Cairo, Egypt.

Emails: maryam185629@bue.edu.eg, s.hussein@mtc.edu.eg

Abstract

In the latest years, the modern cities had a rapid growth in the number of vehicles. This leads to reduce the availability of parking slots as the searching process becomes a very hard task. This increases the traffic congestion, that leads to a waste of fuel during waiting time with a vehicle queuing and a traffic jam. According to the development in technologies and the spreading of the smart systems in several fields. Smart car parking occupancy detection system becoming a necessary, it mainly aims to decrease the traffic congestion that faces the parking areas. These systems are achieved using computer vision, machine learning and deep learning techniques, that is capable to determine if the parking slot is empty or occupied.

This research will create a comparative survey among parking systems that used a different technique. The comparison will discuss each system methodology, datasets, and results. Mostly, these systems are built using Deep Neural Networks (DNNs) architectures that fit each system. Smart parking system that are using a computer vision uses its input as a digital image or a video that is captured by a camera. Different systems will be analyzed and discussed in the literature survey and one algorithm will be chosen as the system methodology. In this report, the parking occupancy slot detection system will be implemented using two YOLO versions, YOLOv3, and the latest version (YOLOv8). The system methodology is divided into two main stages, the first stage is the slot detection, and the second stage is the slot occupancy detection. The system will output the slot detection using polylines and the vehicles will be detected using YOLO versions. After implementing both models there will be a comparison between them. Furthermore, the goal

at the end is to output the number of available slots, this process will be executed by counting the detected vehicles and then extract the total number of available slots as final output.

Problem Statement

The smart parking systems are spreading recently with various techniques and ways of implementation. Any system may face some obstacles that could affect on its efficiency. So, the smart parking system faced some problems that may affect the system goal to output the correct number of available parking slots. Many systems did not have a dataset with enough number of images which outputs inefficient accuracy. Also, some of the extracted datasets may be created due to a fixed time of the day or a specific weather, which makes the model not applicable with all conditions and affect negatively on the system efficiency.[1] Furthermore, choosing unsuitable architecture due to the dataset features or inconvenient number of layers in the system architecture leads to output bad accuracy. The existing systems aimed to enhance the accuracy of the prediction. During the implementation they tried to avoid using a small dataset or a poor architecture design to get a higher accuracy.[2]

Research Objectives

The research project has multiple objectives. First is to create a comparative study on different car parking occupancy detection techniques implemented in the previous studies, smart parking systems are mainly based on Machine Learning, Deep Learning and Computer Vision. Then choose a specific algorithm to implement in the methodology, this algorithm is YOLO. Implement the preprocessing steps such as camera positioning and dataset. Then Detect the slots using open-cv tools “polylines”. Furthermore, to detect the slot occupancy we will use two YOLO versions. YOLOv3 that is discussed in the literature review to execute the slot occupancy. The second is YOLOv8 (latest version of YOLO algorithm) to execute the slot occupancy. Collect the results and evaluate the implemented algorithm. Compare between the existed system results and the system will be implemented in this research.

Brief Background

Debaditya Acharya et.al [3] in 2017, had achieved a Parking Guidance and Information (PGI) system design. The system was implemented using a CNN model and the Binary SVM classifier. The system was divided into two main stages as: The first stage extracts the images features from the 21 layers of the CNN model from the public dataset (PKLot)[8]. PKLot consists of 12,415 images with three categories: cloudy, rainy and sunny. Then these extracted features used during the classification process to train and test the four binary SVM classifiers. The second stage is applying the process of cross validation using a public dataset and the technique of transfer learning is applied on a dataset named (Barry Street) created by the authors to evaluate the

classification accuracy. The model achieved on the PKLot dataset accuracy of 99.7% and the Barry Street dataset accuracy of 96.7%.

Wei Li et.al [1], in 2020 proposed a Park Assist System (PAS). The system uses the characteristic of the around view monitor (AVM). The system was implemented using the deep convolutional neural network (DCNN) with several methods that helped while detecting parking spaces. There are two modules applied by the VPS-Net localization procedure. The first module is named as "Parking Slot Detection". This module aimed to detect the parking slots in general of the whole parkin space, using YOLOv3 and ps2.0 dataset. The second module named as "Occupancy Classification". This module concerns to complete the aim of the PAS after detecting the parking slots, the system has to detect and classify the vacant parking slot and ignore occupied slots by using the PSV dataset that consists of 4249 images. The VPS-Net obtained a precision rate of 96.73 % and recall rate of 94.60%.

Methodology

YOLO is an algorithm that provides a generalizability for multi object detection classes. It contains a large number of neural networks that make the object detection process more efficient. The CNN layers are used to extract features and recognize patterns from the input, and the input is divided into grids to define the prediction result. YOLO recorded a high accuracy, with evaluation terms such as intersection over union (IoU) and mean average precision (mAP). YOLOv3 is the third version of YOLO, using a hybrid network that includes residual network and Darknet-19 network that updates the network to reach 53 convolutional layers.

The YOLOv8 version of the parking occupancy slot detection system is a robust version against occlusion and different whether and light conditions. It uses the same architecture as version 4, CPSDarknet-53, and the cross stage partial network (CPSNet) method. The aim of the system is to detect all the slots using preprocessing phase using open-cv and then detecting the vehicles that are occupying the slots using YOLO to output the total number of available slots. The YOLO models were trained on COCO dataset, which has many classes but only the road vehicles classes will be chosen.

System Framework

This section will review how the implementation is divided mainly into two phases as shown in Figure.1

The first phase is the "**Preprocessing**" for the system to check whether the base components is in appropriate way. This phase is divided into two main sub-tasks the first sub-task is to choose a good camera placement to ensure that the detection process will be done successfully in the future. The second sub-task is to choose the dataset that contains the five road vehicle classes that can occupy the slots in the parking area.

The second phase is the "**Available slot detection**" this is divided into two main sub-tasks. The first sub-task is the slot detection that will be achieved using two methods: detecting each slot separately and detecting the slots by row once a time. The second sub-task is defining the slot occupancy, and this will be executed using two YOLO versions YOLOv3 and YOLOv8. The YOLO versions will be used to achieve multi object detection for different vehicle classes.

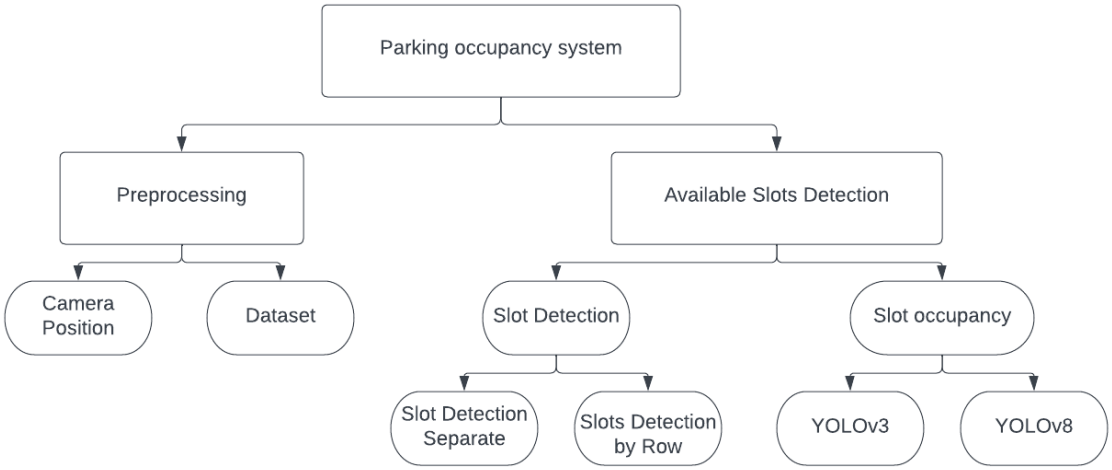

Figure 1: Parking Occupancy Slot Detection System Framework

This part will be viewing the good camera positions and how to avoid the bad positions for obtaining an efficient system. The important factor is to find a good camera placement by taking into consideration all the scenarios for occupying the parking lots. Also, if the parking area is outdoor the different whether conditions will affect the vehicles appearance by resulting occlusion and shadow. Always a high camera position that provides a wide range of vision is efficient, this position can reach to be perpendicular to the ground. The good camera positions differ from parking area to another, but the main point is to think about all the probabilities that may happen. This will help to reach the best choice that will give the system a very clear and wide vision while detecting the slots and vehicles. Figure.2 illustrates how the camera can be placed in good positions that provide a full vision without any distractions.

Figure 2: Good Camera Position Examples

Dataset

The COCO dataset is a public dataset created by Microsoft and other academic organizations. It consists of over 330,000 images and is divided into 80 different classes for different super directories such as person, vehicle, sport, etc. The dataset contains images that fit different functionalities such as object detection, segmentation, recognition, and captioning. It was used to create several pretrained models for object detection, and the parking occupancy slot detection system will define to detect only specific classes of vehicles if they occupy the parking slot.[4] Figure.3 illustrates the distribution of the classes in the dataset.

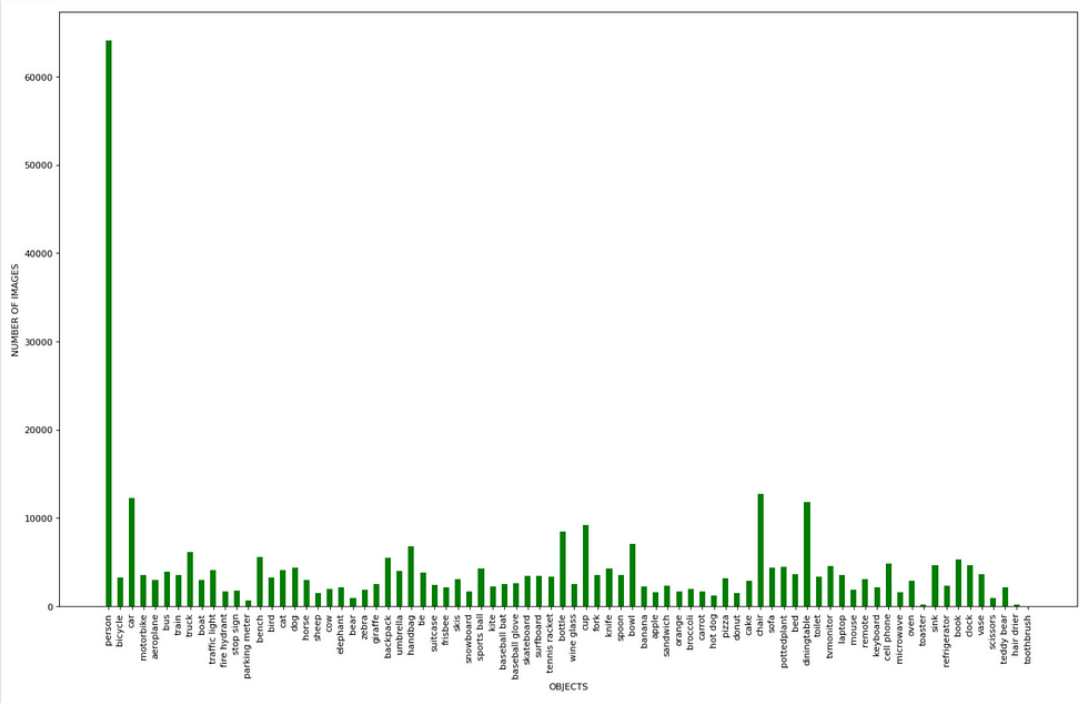

Figure 3: COCO dataset classes distribution

Slot Detection

The slot detection is implemented using two techniques:

Detection of each slot separately

Mainly this process is implemented by taking the four corners coordinates for each slot. After taking the coordinates we will assign a variable called “area” for each slot and assign the four x and y coordinates in this variable as an array. Then this area will be used to draw polylines between every two points. The polyline is a tool that is supported by open-cv library in python.

The reason of using the polylines especially is to fit all the slot shapes and angles. The slots can be slanted with acute or obtuse corner angles, and it can be with right corner angles, so we won’t choose square or rectangle to draw the slot area. The corner angle is defined between each two lines.

Detection of the area for each row of slots

This method is less time consuming than the previous one. The same steps will be implemented here but with taking only four coordinates for each parking row and also connect them together using polylines. For each row the coordinates are: top left corner, bottom left corner, top right corner, and bottom right corner. Each coordinate has x and y values represent its pixel location from the input. The detection of the vehicles will be done in the predefined area of the row.

In conclusion, for detecting the slots we need a tool that extract the pixel location from the image or the video stream. Examples for a tool to read the pixel coordinates from an image it is a "Microsoft Paint" and for the video "VLC Media Player". The two methods discussed to detect slots, or the area of the slots can output the same results, it just depends on the output visualization. As mentioned above that the detection for each slot separately assigned with the slot number can be more effective for the user.

Slot Occupancy

The slot occupancy detection will be implemented using YOLOv3 and YOLOv8 to detect all the vehicles that occupy the parking slots. The algorithm will represent the YOLO architectures, feature extraction, class prediction, anchor box, bounding box, and final prediction. The final prediction is chosen after eliminating inappropriate bounding boxes.

Whole System methodology:

After discussing above the full steps that will be taken to implement the parking occupancy system. This section will discuss how the parking occupancy slot detection system is implemented by setting all the system components together.

First the camera position was set up into a good position which gives the camera a full vision to all parking slots. The camera plays a roll of the human eye which will promote our neural networks to see the parking slots and then predict. The camera will be transferring a real-life video which will be the input for the system. Then as mentioned before the slots will be detected using two preprocessing methods. The first method is to define the four coordinates for each row of the parking area and the second method is to define the four coordinates for each slot separately. For better visualization and vehicles detection later the second method is better. It has a weakness point that it is more time consuming due to collecting the four coordinates for each slot at a time. On the other hand, this method is still less time and effort consuming than implementing a separate model to detect the slots areas.

Furthermore, when slots are defined using open-cv library "polylines" the pretrained models for YOLOv3 [5] and YOLOv8 [6] will be used to recognize the object from the parking row or parking slots. Both methods will give us a very close or maybe the same detection results. This is happened due to that the slot detection process is not based on prediction. Moreover, for counting the vehicles that occupy the slots a class list will be created to append all the detected vehicles in this list.

There is a file that named as coco.txt file, this file contains name tags for 80 classes. These classes belong to the COCO dataset that pretrained models built using it. When YOLO detect the object its corresponding name from the coco.txt file will be read. In our system there are only five classes of vehicles that were specified. After the YOLO detect the vehicles in the slots area and from the chosen classes, the class list will append these objects. The class list will count the detected vehicles as the count of the occupied slots.

At the end, we need to output the available slots, this is done by defining the total number of slots and subtract the detected vehicle in the class list from it. This will output the number of the empty slots that will be shown on the top-left corner of the screen as a fixed number if the input is an image and if the input is a video stream or a real-time the number of the slots will be updated automatically along the video frames when a new vehicle occupies a slot, or a vehicle exit the slot. The aim of the system is to provide the new vehicles that need to park a system that saves their time and effort. In conclusion, if the new vehicle knew before entering the parking area if there is available slots and how many are available this will also reduce the traffic jams and accidents may happen in the parking area.

Results

Results of the Slot Detection

This section will be representing the output sample of the slot detection using the two methods as discussed in the previous chapter. These two methods are: slots detected separately, and slots detected by row.

Output sample (slots detected separately) using polylines

To clarify how the detection is achieved this is a sample output while the parking area is empty. The parking area of the system consists of twelve parking slots. So, to detect the parking slots separately in this scenario we need to read 48 coordinates, 4 coordinates for each slot of the 12 slots. Connecting the four coordinates together was implemented using polylines provided by open-cv library in python. Figure.4 illustrates the parking area with the slots detected separately as twelve different areas with their captions. So, as no vehicle park the slots will be marked with green color and the count of the available slots will be shown at the top-left corner of the screen as 12. This method was implemented for both YOLOv3 and YOLOv8.

Figure 4 : YOLOv3 model slots detected separately using polylines output

Output sample (slots detected by row) using polylines

To clarify how the detection is achieved this is a sample output while the parking area is empty. As mentioned before the parking area has 12 parking lots. So, to detect the parking slots by row in this scenario it will easier than previous method we only need 4 coordinates for the whole row. Connection of the coordinates also is achieved using polylines. Figure.5 illustrates the parking area with the slots detected by row combining all the lots in one area. So, as no vehicle park the slots area the count of the available slots will be shown at the top-left corner of the screen as 12. This method was also implemented for both YOLOv3 and YOLOv8.

Figure 5 : YOLOv3 slots detected by row output

Results of YOLOv3 model occupancy detection

In this section, the results of YOLOv3 will be represented as two sections: output sample and accuracy results. The output sample will show how the model detect the vehicles in the slots using the two methods for the slot detection. The final output will be that the vehicles are detected, and the count of the available slots will be updated at the top-left corner of the screen. The second section “accuracy results” will illustrate the accuracy tables and graphs of the YOLOv3 model.

Output Sample:

Slots Detected Separately:

Figure.6 illustrates the first output sample in YOLOv3 for the slots detected separately. The scenario of the output that some vehicles occupy the slots. The new rectangle that appears and surround the vehicle is the bounding box and the red circle is the center of the bounding box after the vehicle is detected. The bounding box only appear if the vehicle detection is achieved in the slot area. The slot number, polylines and the center of the bounding box is converted to the red color after the detection is done. As the figure.6 clarifies that YOLOv3 is not robust enough among the occlusion happened in slot 9. Also, the count of the available slots is updated to 7.

Figure 6 : YOLOv3 slots detected separately, and vehicles occupy slots output

Slots detected by row:

Figure.7 illustrates the second output sample in YOLOv3 for the slots detected by row. The scenario of the output that some vehicles occupy the slots. The bounding box will surround the detected vehicle using a rectangle and circle that represent the bounding box center. As shown in the figure that the bounding box only appear if the vehicle detection is achieved in the row area that is predefined with the pixel coordinates. Note that the model detects different vehicles classes

such as car and truck. The model also counts three more classes: bus, motorcycle, and bicycle but these vehicles are not found in the slots. The count of the available slots is updated to 4.

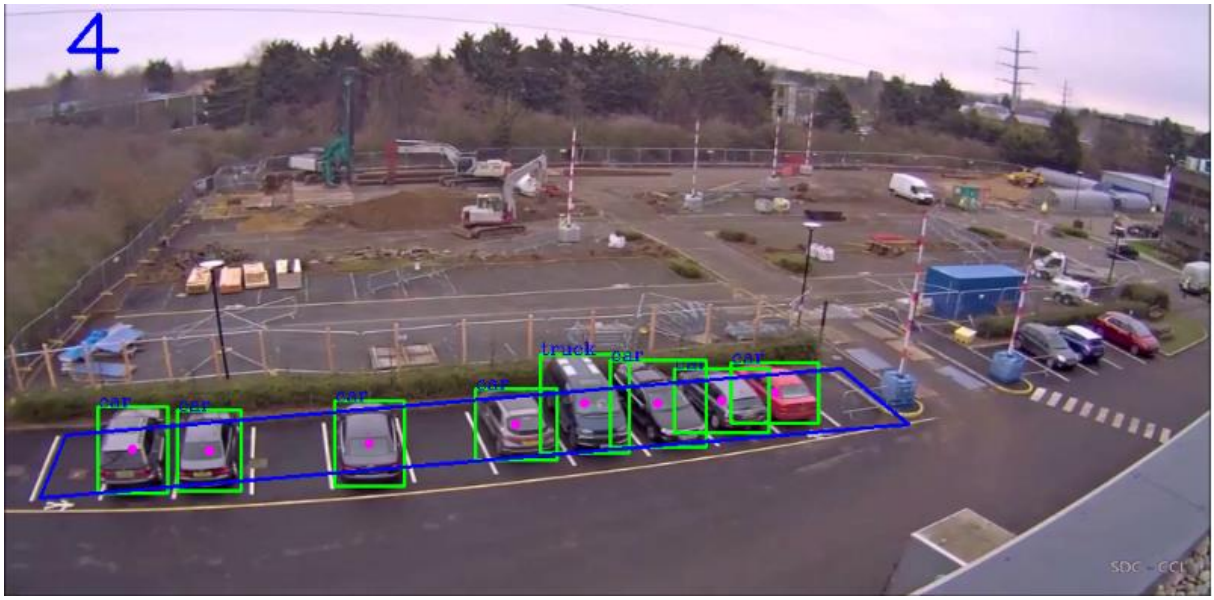

Figure 7 : YOLOv3 slots detected by row and vehicles occupy slots output sample

YOLOv3 model accuracy

Table.1 illustrates the YOLOv3 results compared with previous versions. The version that was used is YOLOv3-416. These numbers refer to the size of the images used for the model. As the table clarifies that when the resolution (number of pixels) increases the accuracy enhanced. The high resolution helps while extracting the features from the input for better detection results. COCO dataset as mentioned before is the dataset used for training and testing the model. The mAP for the desired model is 55.3 % at IoU threshold value of 0.5. The overall mAP is set for the whole model performance at all IoU threshold values is recorded as 31.0 %.

Table 1: YOLOv3 mAP Results [7]

Model	Train	Test	mAP at 0.5	Overall mAP
YOLOv3 - 320	COCO trainval	test-dev	51.5 %	28.2 %
YOLOv3 - 416	**COCO trainval**	**test-dev**	**55.3 %**	**31.0 %**
YOLOv3 - 608	COCO trainval	test-dev	57.9 %	33.0 %

Table.2 refers to the YOLOv3 results of true postive and ground truth ratio results on COCO dataset. The output results is for the five vehicle classes only.

Table 2 : COCO dataset True Positive and Ground Truth Ratio for vehicles classes [7]

Class	Number of TP	Number of GT	Ratio
Car	1113	1532	72.7 %
Truck	229	335	68.4 %
Bus	193	217	88.9 %
Motorcycle	201	261	77.0 %
Bicycle	138	216	63.9 %

The mAP-50 score measures the average precision of the algorithm over a range of detection thresholds, where the threshold is set at 50% Intersection over Union (IoU) between the predicted bounding boxes and the ground truth bounding boxes. Figure.8 illustrates the graph of YOLOv3-416 mAP at 0.5 that records 55.3 % on the COCO dataset.

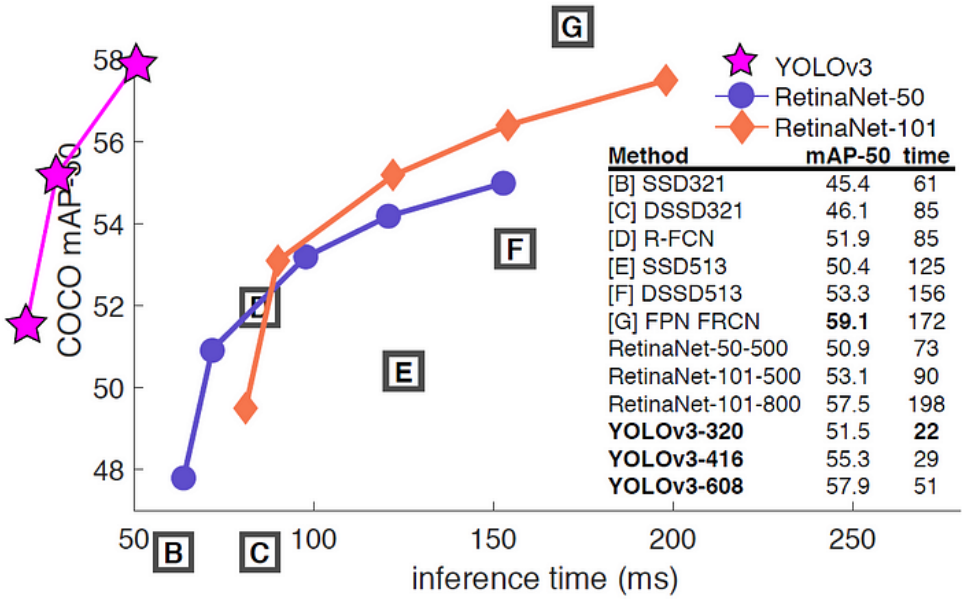

Figure 8 : YOLOv3 on COCO dataset at mean average precision (mAP) = 0.5 [5]

The overall mAP that is different from the previous one. This is calculated by taking the average of AP (Average Precision) scores across all classes in a dataset over recall value from 0 to 1. Figure.9 illustrates YOLOv3-416 records overall mAP by 31.0 % on COCO dataset.

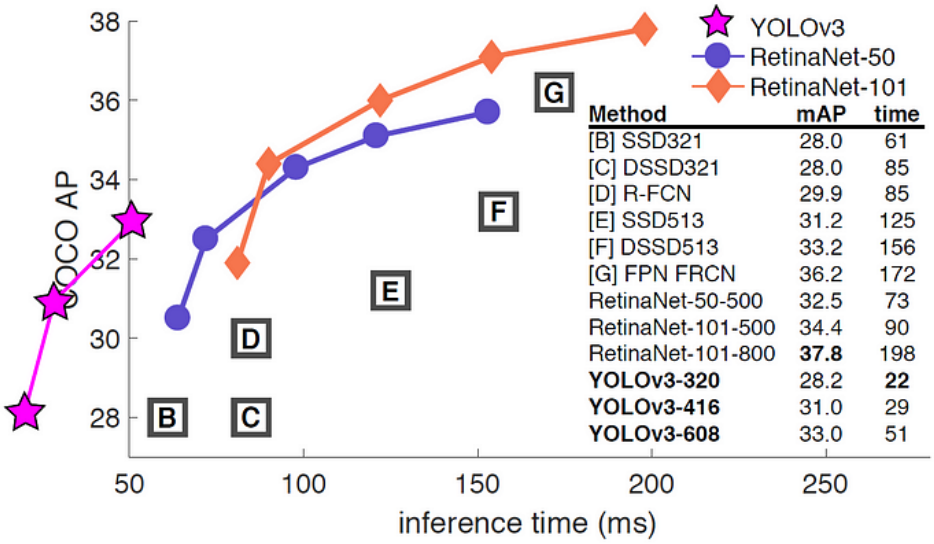

Figure 9 : YOLOv3 on COCO dataset overall mean average precision (mAP) [5]

Results of YOLOv8 model occupancy detection

In this section, the results of YOLOv8 will be represented as two sections: output sample and accuracy results. The output sample will show how the model detect the vehicles in the slots using the two methods for the slot detection. The final output will be that the vehicles are detected, and the count of the available slots will be updated at the top-left corner of the screen. The second section "accuracy results" will illustrate the accuracy tables and graphs of the YOLOv8 model.

Output Sample:

Slots detected separately:

Figure.10 illustrates the first output sample in YOLOv8 for the slots detected separately. The scenario of the output that some vehicles occupy the slots. The new rectangle that appears and surround the vehicle is the bounding box and the red circle is the center of the bounding box after the vehicle is detected. The bounding box only appear if the vehicle detection is achieved in the slot area. As the Figure.10 clarifies that YOLOv8 is robust against the occlusion happened in slot 9. Also, the count of the available slots is updated to 6.

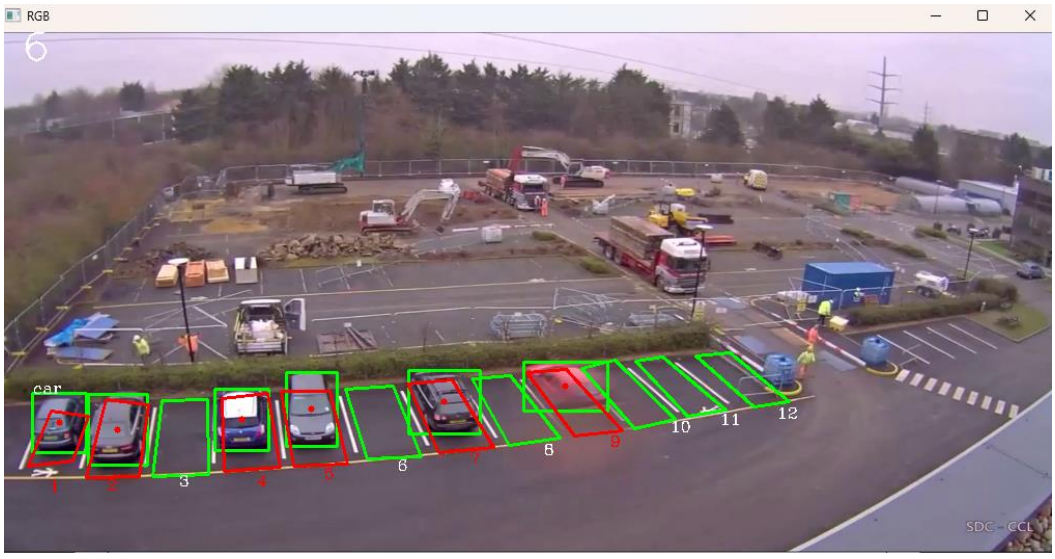

Figure 10 : YOLOv8 slots detected separately, and vehicles occupy slots output

Slots detected by row:

The last scenario in YOLOv8, we have some vehicles that occupy the slots. The bounding box will surround the detected vehicle using a rectangle and circle that represent the center of the bounding box. Figure.11 shows that the bounding box only appear if the vehicle detection is achieved in the row area that is predefined with the pixel coordinates. Furthermore, the vehicles are detected efficiently in different light conditions. The count of the available slots is updated to 8 in the top-left corner.

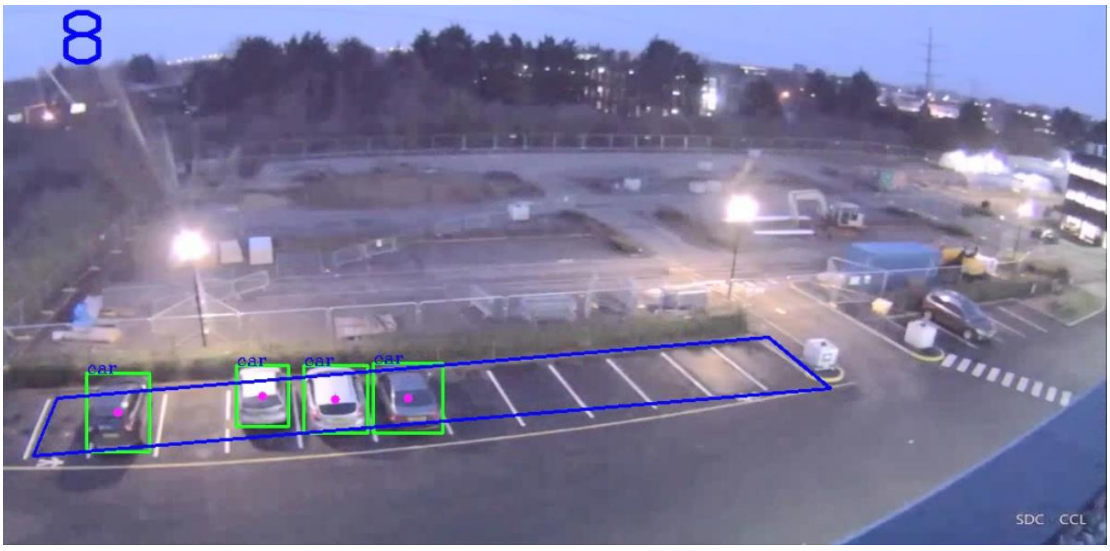

Figure 11 : YOLOv8 slots detected by row and vehicles occupy slots output sample

Figure.12 clarifies how the video stream updates the number of available slots and also the YOLOv8 detects all the objects of the 80 classes found in COCO dataset, but the other objects won't be counted in the class list. Only the five classes were chosen are added to the class list.

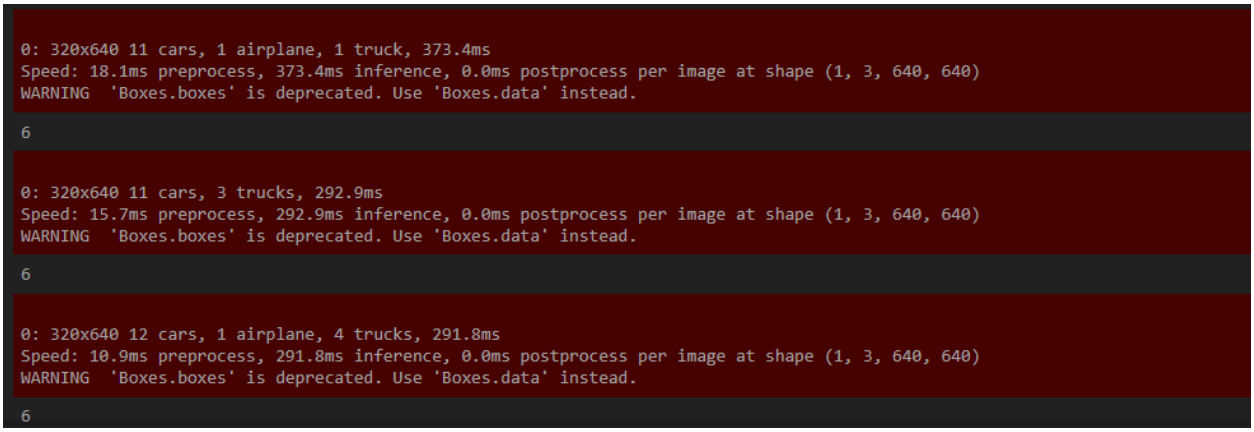

Figure 12 : YOLOv8 output from video stream

YOLOv8 model accuracy

Table.3 clarifies the mean average precision for different YOLOv8 versions. The desired model for the project is "YOLOv8x" which records the highest mAP among other versions. The different versions depend on the size of the architecture and the speed, so the "nano" version is the smallest one, but it has the fastest speed. On the other hand, the "extended" version is the largest one and the slowest. But this won't affect our system because we need to be updated with the output every 3 to 5 minutes and the speed comparison is not important.

Table 3 : YOLOv8 different versions results, the desired model is YOLOv8x [8]

Model	Train	Test	Overall mAP
YOLOv8n (nano)	COCO trainval	test-dev	37.3 %
YOLOv8s (small)	COCO trainval	test-dev	44.9 %
YOLOv8m (medium)	COCO trainval	test-dev	50.2 %
YOLOv8l (large)	COCO trainval	test-dev	52.9 %
YOLOv8x (extended)	**COCO trainval**	**test-dev**	**53.9 %**

Figure.13 clarifies the YOLOv8 model using different versions to express the latency of the versions and the mean average precision for each version.

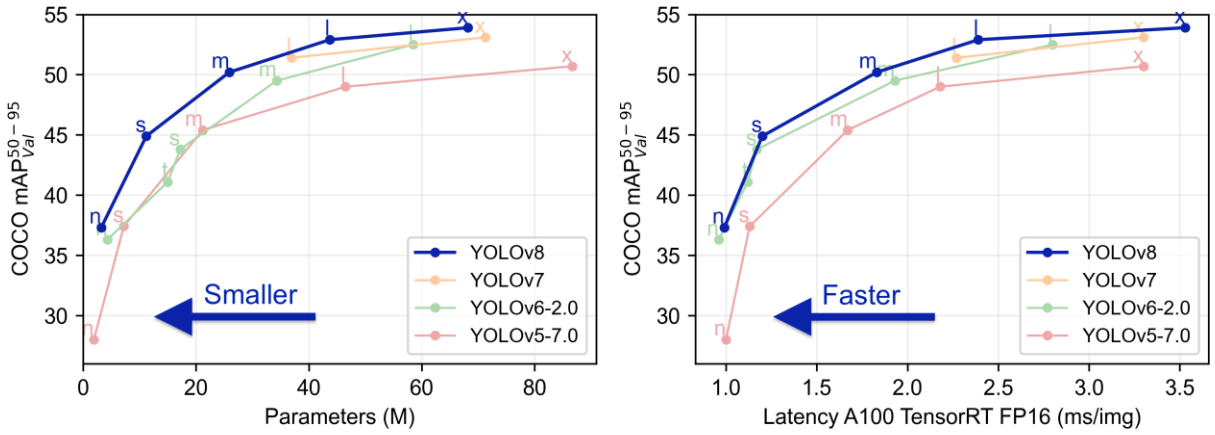

Figure 13 : YOLOv8 on COCO dataset overall mean average precision (mAP) [8]

In conclusion, the YOLOv3 and YOLOv8 has achieved efficient results regarding the mAP results. The YOLOv8 has reached a better result than YOLOv3 by 20.0 % mAP. YOLOv8 is more robust to the occlusion and faster in speed. YOLOv3 also achieved a good ratio result for TP/GT in the desired classes (car, truck, bus, motorcycle, and bicycle). The ratio average among the five classes is 74.18 %.

Conclusion

This section summarizes the main achievements that were achieved during the research project and what will be achieved as a future work. There was a motivation for the bad management of parking areas that led to traffic jams, waste of fuel, and time consumption. There were technical problems that faced the systems that tried to manage the parking areas, such as choosing or creating a poor dataset and choosing inappropriate models to implement the system. The YOLO algorithm was chosen among the other models because it was achieving good accuracy results and the YOLO version that was implemented was published in 2018 and there is a new version that enhance the model accuracy. The preprocessing phase was done by placing the camera in a good position and choosing an appropriate dataset.

The models that were implemented were the YOLOv3 and YOLOv8x pretrained models on COCO dataset. The YOLOv3 achieved overall mAP 31.0 % and the YOLOv8x achieved overall mAP 53.9 % both on COCO dataset. The comparison among these two models was competitive regarding their results.

Future Work

The future work of the parking occupancy slot detection system is to train the YOLOv8x model on a custom dataset. The dataset will be consisting of the five vehicles classes that were used in this project. Implementing the model on a custom dataset will be a comparison that will test if the public dataset is more efficient than the custom dataset or not. The goal for this

comparison is to study that it necessarily to create a new dataset for any system or there are some public datasets that will give efficient results while testing the model in real time.

References

[1] Wei Li et. al, “Vacant parking slot detection in the around view image based on deep ...,” 10-Apr-2020.

[2] S. Rahman et. al, “Convolutional neural network customization for parking ... – IEEE xplore,” 2020/.

[3] D. Acharya, W. Yan, and K. Khoshelham, “Real-time image-based parking occupancy detection using Deep Learning,” Real-time image-based parking occupancy detection using deep learning, 2018.

[4] T.-Y. Lin et al., “Microsoft Coco: Common Objects in Context,” arXiv.org, https://arxiv.org/abs/1405.0312.

[5] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv.org, https://arxiv.org/abs/1804.02767.

[6] D. Reis, J. Kupec, J. Hong, and A. Daoudi, “Real-time flying object detection with yolov8,” arXiv.org, https://arxiv.org/abs/2305.09972.

[7] D. H. Kim, Evaluation of coco validation 2017 dataset with Yolov3 - JMEST, http://www.jmest.org/wp-content/uploads/JMESTN42352998.pdf.

[8] S. Rath, “Yolov8 ultralytics: State-of-the-art yolo models,” LearnOpenCV, https://learnopencv.com/ultralytics-yolov8/.