Parking Lot Occupancy Detection using YOLOv5

Daniele Ninni
Department of Physics and Astronomy "Galileo Galilei", University of Padua, Italy
daniele.ninni@studenti.unipd.it

This report presents a computer vision algorithm for parking lot occupancy detection. It is based on the use of YOLOv5, a one-stage deep learning object detector. The proposed solution is tested on *CNRPark*, a dataset containing images of the parking lot of the CNR (National Research Council) in Pisa. The images are taken on different days and times, from different viewpoints and with different light and weather conditions. Some of them include shadows and occlusions which make the occupancy detection task even more challenging. The results of the evaluation show that the algorithm is effective as long as its parameters are properly tuned. If so, the proposed approach proves to be robust not only to the variety of *CNRPark* but also to shadows and occlusions.

***Index Terms*—Classification, Computer Vision, Convolutional Neural Networks, Deep Learning, Machine Learning, Object Detection, YOLO.**

I. Introduction

Object detection is a computer vision task that can be regarded as an advanced form of the image classification task. In particular, it refers to the detection of instances of visual objects belonging to a predefined set of classes within an image. In other words, the object detection task consists in assigning a class and a location to each object in an image. Typically, the location is assigned in the form of a rectangular bounding box.

This report presents a computer vision algorithm for parking lot occupancy detection, a task that can be reformulated as an object detection task as follows:

given the location of a parking space (i.e. the corresponding bounding box) within the image: ***if*** *an object identifiable as a vehicle is detected within the bounding box, then mark the parking space as occupied,* ***otherwise*** *mark it as free.*

The algorithm follows the trend of applying deep learning techniques to problems that require a high level of abstraction to be solved. Specifically, it is based on the use of YOLOv5, a one-stage deep learning object detector.

The proposed solution is tested on *CNRPark*, a dataset containing images of the parking lot of the CNR (National Research Council) in Pisa. The images are taken on different days and times, from different viewpoints and with different light and weather conditions. Some of them include shadows and occlusions which make the occupancy detection task even more challenging.

The report is structured as follows:

- **Section II** describes the *CNRPark* dataset;
- **Section III** discusses the proposed method;
- **Section IV** discusses the results obtained on *CNRPark*;
- **Section V** concludes the report.

II. Dataset

CNRPark [1] is a dataset containing images of the parking lot of the CNR (National Research Council) in Pisa. The evaluation of the proposed solution is carried out on a subset of *CNRPark*, called *CNR-EXT*, which is composed by images collected from November 2015 to February 2016 under various weather conditions by 9 cameras with different perspectives and angles of view. *CNR-EXT* captures different situations of light conditions, and it includes partial occlusion patterns due to obstacles (trees, lampposts, other cars) and partial or global shadowed cars.

The archive `CNR-EXT_FULL_IMAGE_1000x750.tar` (1.1 GB) contains the full frames of the cameras belonging to the *CNR-EXT* subset. Images have been downsampled from 2592x1944 to 1000x750 due to privacy issues. The archive has the following structure:

```
FULL_IMAGE_1000x750
└── <WEATHER>
    └── <CAPTURE_DATE>
        └── camera<CAM_ID>
            └── <CAPTURE_DATE>_<CAPTURE_TIME>.jpg
```

where:

- `<WEATHER>` can be `SUNNY`, `OVERCAST` or `RAINY`;
- `<CAPTURE_DATE>` is the zero-padded `YYYY-MM-DD` formatted capture date;
- `<CAM_ID>` is the number of the camera, ranging `1-9`;
- `<CAPTURE_TIME>` is the zero-padded 24-hour `HHMM` formatted capture time.

The archive contains also 9 CSV files (one for each camera) containing the bounding boxes of each parking space. Pixel coordinates of the bounding boxes refer to the 2592x1944 version of the images and need to be rescaled to match the 1000x750 version. Note that bounding boxes are non-rotated squares and often do not precisely or entirely cover the corresponding parking spaces. This constitutes a further complication to the occupancy detection task stated in the **Introduction**.

III. METHOD

This section introduces the YOLO object detector and then discusses the proposed algorithm to fulfill the occupancy detection task.

A. YOLO

YOLO [2] (You Only Look Once) is a state-of-the-art, real-time object detection algorithm. It models object detection as a regression problem that takes an input image and learns the coordinates of the bounding boxes and the probability of each class. Its workflow can be summarized as follows:

1) The image is divided into a $S \cdot S$ grid: each cell in the grid is responsible for detecting and localizing objects within itself;
2) For each cell, YOLO predicts N bounding boxes and confidences: the confidence represents the probability that the bounding box actually contains an object;
3) For each bounding box, YOLO also predicts the classification score for each class;
4) A total of $S \cdot S \cdot N$ boxes are predicted: however, most of them have a low confidence score, therefore a threshold to the confidence score itself can be applied to discard them.

YOLO is a one-stage detector, i.e., it predicts bounding boxes and class probabilities in a single forward pass through the CNN, without the need for the region proposal step. On the contrary, two-stage detectors based on the use of a region proposal network (e.g. R-CNN) first detect the possible regions of interest and then perform the detection on each of these regions separately. Hence, while RPN-based algorithms end up performing multiple iterations for the same image, resulting in a higher computation time, YOLO instead performs all its predictions in a single pass. This results in a significantly lower computation time and therefore allows YOLO to be run effectively even in real time.

YOLO is one of the best object detection algorithms as it represents an excellent balance between accuracy and speed. However, its downside is that each cell in the grid is constrained to detect only a single object, hence it struggles to detect and localize small objects that appear in groups.

B. YOLOv5

YOLOv5 [3] is a family of object detection architectures and models based on YOLO and pre-trained on the Microsoft COCO [4] dataset. It is an open-source project maintained by Ultralytics and represents the organization's research into the future of Computer Vision methods. The following is a brief description of each of the YOLOv5 models:

- **YOLOv5n**: the nano model, the smallest in the family and meant for IoT devices and mobile solutions;
- **YOLOv5s**: the small model with 7.2 million parameters and ideal for running inference on the CPU;
- **YOLOv5m**: the medium-sized model with 21.2 million parameters, it is perhaps the best suited model for most datasets as it provides a good balance between speed and accuracy;
- **YOLOv5l**: the large model with 46.5 million parameters, it is ideal for datasets where smaller objects need to be detected;
- **YOLOv5x**: the largest model with 86.7 million parameters, among the YOLOv5 models it is the one that achieves the highest mAP (Mean Average Precision) but it is also the slowest one.

Note that, at the time of development, YOLOv5n and YOLOv5s are the only ones supported by OpenCV DNN module. Therefore, it is convenient to choose YOLOv5s for the implementation of the occupancy detection algorithm.

C. Occupancy detection algorithm

This section discusses the proposed algorithm for determining whether each parking space is occupied or free. The algorithm is implemented entirely in C++ and is based on the use of the OpenCV 4.5.5 library.

Each detection made by YOLO consists of the following set of information:

- the coordinates of the detected bounding box;
- the confidence score `confidence`;
- the class scores.

First, all detections characterized by a low confidence score are discarded by applying the threshold `CONFIDENCE_THRESHOLD` to the confidence scores themselves. Formally, a detection characterized by the confidence score `confidence` is kept if and only if:

$$\texttt{confidence} \geq \texttt{CONFIDENCE\_THRESHOLD}$$

where $\texttt{CONFIDENCE\_THRESHOLD} \in [0, 1]$.

At this point, a possible approach could consist in identifying, for each detection, the class with the best score. This would allow to further filter the detections, for example keeping only those corresponding to a class belonging to the vehicle category. Such an approach would allow focusing only on objects identified as vehicles by YOLO.

The key problem is that the precision with which YOLO assigns the most probable class to each detection is necessarily limited. This implies the possibility that YOLO, while detecting the presence of an object in correspondence with a vehicle, assigns it a class that does not belong to the vehicle category. Therefore, filtering the detections by most probable class, all *false negative* detections would be discarded.

For this reason, the proposed approach assumes to keep also the detections identified as objects other than vehicles by YOLO. In fact, in this specific context, the goal is not to ensure that YOLO assigns the classes as precisely as possible, but rather that, in correspondence with a vehicle, it detects the presence of an object. In other words, it does not matter if YOLO detects the presence of a vehicle but does not assign it the correct class, the important thing is that it detects the presence of an object in a region of the image where it is highly probable that there is a vehicle. These regions coincide with the bounding boxes of each parking space, known a priori as they are included in *CNRPark*.

This decision obviously leads to the need to introduce alternative criteria that allow to discard the detections not corresponding to vehicles in the parking spaces. Therefore, in addition to `CONFIDENCE_THRESHOLD`, the following thresholds are introduced:

- $\texttt{OVERLAP\_THRESHOLD} \in (0, 1]$
- $\texttt{DETECTION\_AREA\_THRESHOLD} \in [1, +\infty)$

On the one hand, `OVERLAP_THRESHOLD` is used to ensure that the detected bounding box:

- is located in correspondence with an actual parking space;
- does not correspond to an object much smaller than a vehicle.

On the other hand, `DETECTION_AREA_THRESHOLD` is used to ensure that the detected bounding box does not correspond to an object much larger than a vehicle.

Hence, the algorithm continues with a loop through the bounding boxes of the actual parking spaces. Then, for each parking space, a loop through YOLO detections is performed to figure out if there is at least one compatible with the parking space itself. Formally, consider the following notation:

- A_i^{TRUE} : area of the bounding box of the i-th parking space
- A_j^{YOLO} : area of the bounding box of the j-th YOLO detection

If both of the following conditions are met:

$$\left(A_j^{YOLO} \cap A_i^{TRUE}\right) \geq \texttt{OVERLAP\_THRESHOLD} \cdot A_i^{TRUE}$$

$$A_j^{YOLO} < \texttt{DETECTION\_AREA\_THRESHOLD} \cdot A_i^{TRUE}$$

then the i-th parking space is identified as occupied. Otherwise, it is identified as free.

It is evident that the effectiveness of the proposed algorithm depends on the values of all three thresholds. It is proposed to set the threshold values as follows:

- `CONFIDENCE_THRESHOLD`: set a rather low value in order to discard as few detections as possible;
- `OVERLAP_THRESHOLD`: it is reasonable to expect that A_j^{YOLO} and A_i^{TRUE} are significantly overlapped in the case of a correct detection, therefore set a value equal to at least 50%;
- `DETECTION_AREA_THRESHOLD`: it makes sense to expect that A_j^{YOLO} is not too much larger than A_i^{TRUE} in the case of a correct detection, therefore set a value approximately equal to 5.

The proposed implementation includes the use of OpenCV trackbars to simplify the exploration of threshold values and visualize their effects in real time.

IV. Results

This section shows the results obtained on 9 sample images chosen at random from the *CNR-EXT* dataset, one for each of the 9 cameras.

Fig. 1. OVERCAST, 2015-11-16, camera1, 09:10.

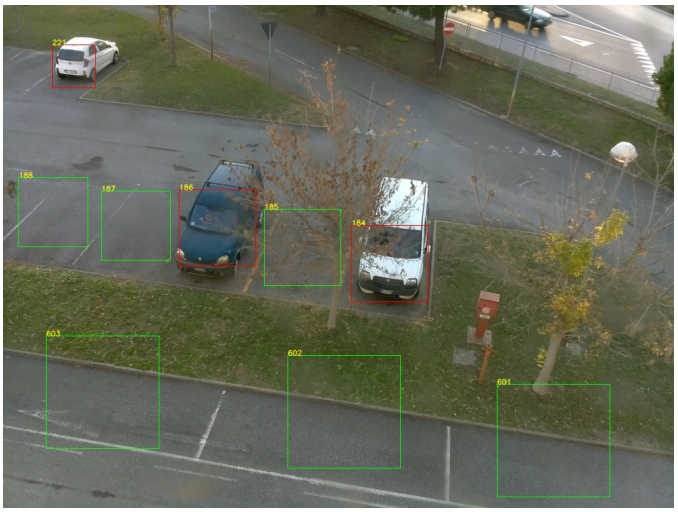

Fig. 2. OVERCAST, 2015-11-29, camera2, 16:14.

Fig. 3. OVERCAST, 2015-12-19, camera3, 12:48.

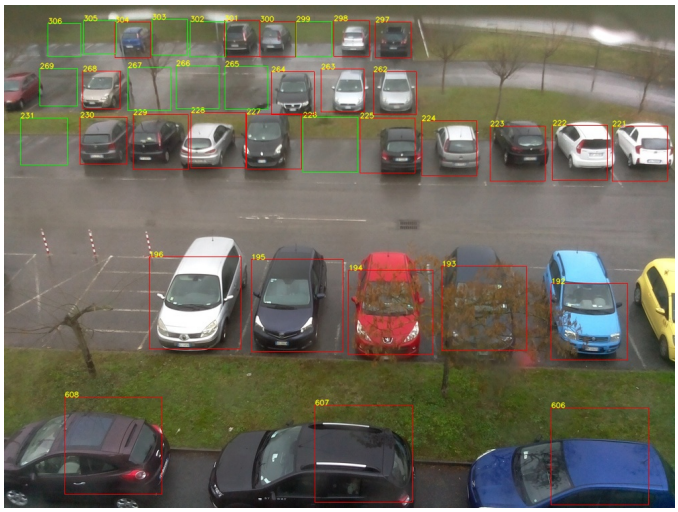

Fig. 4. RAINY, 2015-12-22, camera4, 09:51.

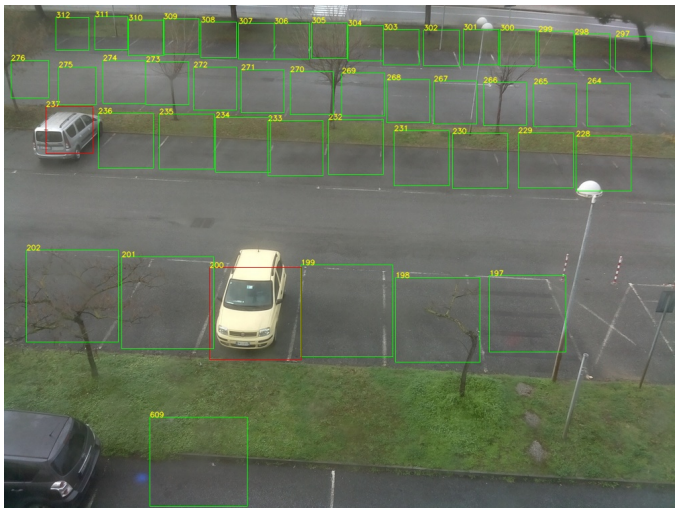

Fig. 5. RAINY, 2016-01-09, camera5, 09:27.

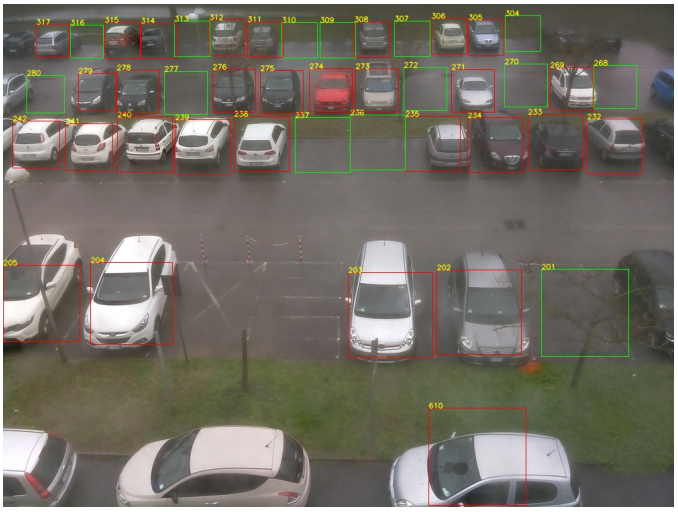

Fig. 6. RAINY, 2016-02-12, camera6, 16:54.

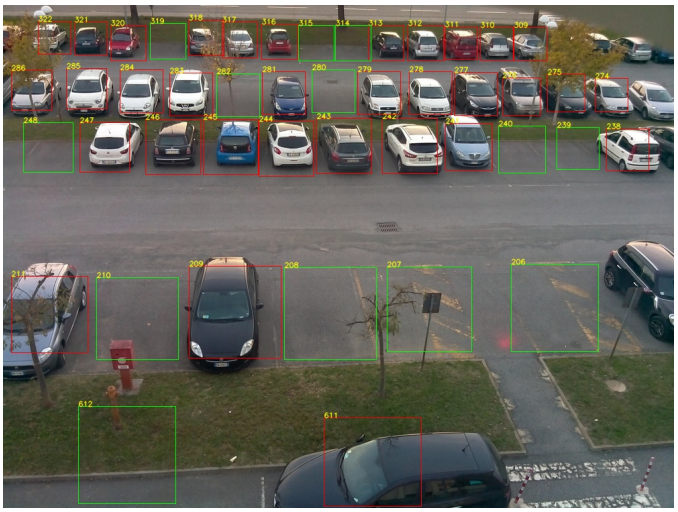

Fig. 7. SUNNY, 2015-11-12, camera7, 16:47.

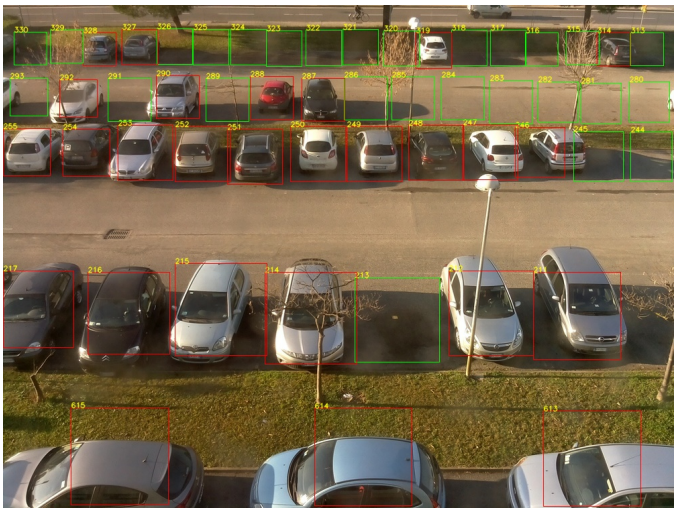

Fig. 8. SUNNY, 2015-12-17, camera8, 09:41.

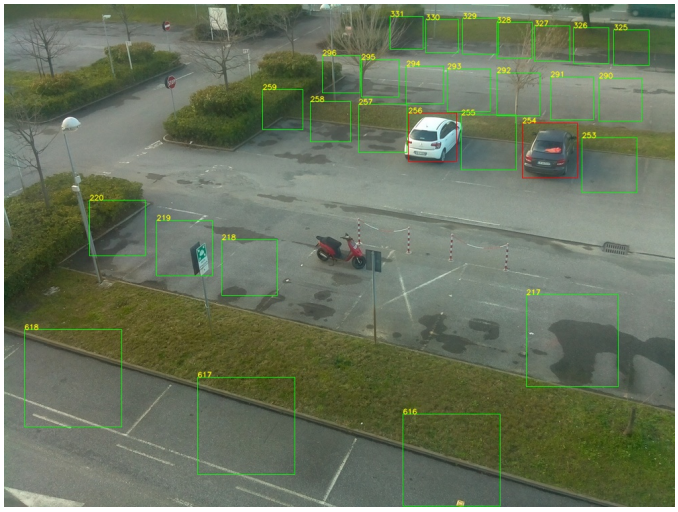

Fig. 9. SUNNY, 2016-01-16, camera9, 09:40.

The values of the thresholds leading to the above results are shown in Table I.

TABLE I
THRESHOLD VALUES.

Threshold / Fig.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
CONFIDENCE (%)	1	1	1	1	3	3	1	10	15
OVERLAP (%)	85	50	50	50	50	50	50	50	50
DETECTION_AREA	5	5	5	5	5	5	3	3	5

The results of the evaluation show that the algorithm is effective as long as its parameters are properly tuned. If so, the proposed approach proves to be robust not only to the variety of *CNRPark* but also to shadows and occlusions. It turns out to be quite robust even to non-standard parking behaviors, such as cars occupying more than one parking space. In that case, the algorithm tends to mark both parking spaces as occupied.

V. CONCLUSIONS

This report proposes and evaluates a computer vision algorithm for parking lot occupancy detection. It is based on the use of YOLOv5, a one-stage deep learning object detector. The proposed solution is tested on *CNRPark*, a dataset containing images of the parking lot of the CNR (National Research Council) in Pisa. The results of the evaluation show that the algorithm is effective as long as its parameters are properly tuned. If so, the proposed approach proves to be robust not only to the variety of *CNRPark* but also to shadows and occlusions. It turns out to be quite robust even to non-standard parking behaviors, such as cars occupying more than one parking space. In that case, the algorithm tends to mark both parking spaces as occupied.

REFERENCES

[1] CNRPark+EXT: A Dataset for Visual Occupancy Detection of Parking Lots. [Online]. Available: http://cnrpark.it

[2] YOLO: Real-Time Object Detection. [Online]. Available: https://pjreddie.com/darknet/yolo

[3] YOLOv5. [Online]. Available: https://github.com/ultralytics/yolov5

[4] Microsoft COCO: Common Objects in Context. [Online]. Available: https://cocodataset.org/

[5] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327–334, 2017.

[6] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *Computers and Communication (ISCC), 2016 IEEE Symposium on*. IEEE, 2016, pp. 1212–1217.