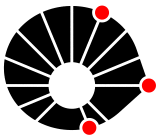
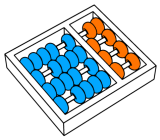




SmartParking
A smart solution using Deep Learning

J. V. Baggio *L. F. Gonzalez* *J. F. Borin*

Relatório Técnico - IC-PFG-20-10
Projeto Final de Graduação
2020 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

SmartParking - A smart solution using Deep Learning

João Victor Baggio* Luis Fernando Gonzalez† Juliana Freitag Borin‡

Abstract

The number of parking lots users at the University of Campinas main campus has been increasing every year. It is becoming a concern to the university to address the high demand for parking spaces on campus. To contribute with a solution to this problem, this project proposes the design and implementation of a smart parking system that can help users to easily find a free parking spot. The system uses cameras to capture parking lot images, so that, it can identify cars in those images and report the number of free parking spaces. A deep learning network is used to classify empty parking spaces.

1 Introduction

A study conducted in 2017 with nearly 6,000 drivers in 10 cities of the United States shows that Americans spend an average of 17 hours a year looking for parking spaces, which, in turn, results in wasted time, fuel and emissions [4]. Such scenario is not expected in a smart and sustainable city and could be avoided with smart parking solutions.

A smart parking application works as a decision support system when the driver is searching for parking. For instance, if the application shows a particular parking lot of choice to be full, the driver can search for nearby parking lots with available parking spaces or choose another destination. Smart parking solutions usually are based on one or more of the following features: i) to guide the driver to a parking lot with free parking spaces using display boards; ii) to reserve and authorise the driver to a parking lot; iii) to reserve and to guide the driver to a specific parking space using navigational information. Data acquired from monitored parking lots may also be used to predict people flow, detect anomalies, relate the use of cars with weather and day of the week, among many other uses.

There are some smart parking solutions on the market and they can be classified in four categories considering the type of device used for data acquisition: i) use of sensors in each parking space; ii) use of monitoring cameras; iii) use of sensors in each car; and iv) use of aerial images captured by drones.

In this work, a solution is presented using cameras to colect the parking data. Monitoring cameras are installed in a parking lot so that all spaces are covered, allowing that through the continuous capture of photographs the number of empty spaces can be computed using

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP
†Konker Labs
‡Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

a deep learning network. The main differential of this project compared to other works on smart parking is its implementation and validation in a real parking lot. All daily problems faced when installing a smart parking system using cameras were approached and solved. In this way, the main contributions of this work are:

- effective execution of the system in a functional prototype within Unicamp;
- the proposal of a strategy to avoid classifying objects in unwanted areas of a photo;
- the proposal of a strategy to handle overlapping classifications of different objects;
- analysis of the network's temporal performance on different hardware configurations.

In this work, a new neural network is not proposed, but a pre-trained network is explored by validating it in a specific dataset in order to show that such a network can be widely used for everyday scenarios beyond the academic environment.

2 Related Work

This section presents some of the main solutions used to implement smart parking solutions.

2.1 Sensors

The most usual approach is to use sensors in each parking space. Despite being a simple solution, the device and implantation costs are very high; moreover, it has a limited scalability and cannot be implemented in all parking lots. The main types of sensors used in these solutions are:

- **passive infrared sensor (IR)** - these sensors detect changes in energy; when a vehicle occupies a parking space, the sensor identify the change in energy detecting the vehicle presence. IR sensors are commonly used in security alarms and automatic lighting applications;
- **active infrared sensor** - these sensors emit infrared energy and detect any object or vehicle by the amount of reflected energy [6]. They are commonly used in obstacle detection systems (such as in robots);
- **ultrasonic sensor** - these sensors emit sound waves between 25 and 50 kHz and detect objects based on reflected energy. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception of the signal;
- **magnetometer** - these sensors detect the presence of a vehicle by detecting the change in the electromagnetic field. They need to be in close proximity to the vehicle, therefore, they are placed beneath the surface. There are wireless sensors with a battery life time of few years which can be used to detect real-time parking occupancy information. However, they are expensive to install and maintain on a large scale.

2.2 Cars

It is possible to build a smart parking system placing sensors on cars. One way to do this is using radio-frequency identification, such that each vehicle is given a radio frequency tag for identification. A transceiver and a antenna are installed at the entrance of a parking lot to identify the tag and allow the vehicle to occupy a parking spot [10]. The main problem with this kind of solution is that it requires a tag for every car that uses the parking lot, and it neither provides individual parking occupancy status nor facilitates the driver in finding a vacant parking space.

2.3 Drones

Drones are becoming a trend in today's IoT applications. Using drones with equipment capable of detecting cars can be very advantageous, since drones can carry numerous sensors and have great mobility being able to monitor large areas. An example of a study done with drones can be found in [2]. In this work, the authors proposed to paint marks on each parking space to allow the detection of their occupation. Low light, weather conditions and strategies to avoid collisions between drones are some of the problems that need to be addressed when developing a drone-based smart parking system.

During this project, we performed some tests using images captured by drones. The images were classified using the same methodology proposed in this work to classify the images captured by the monitoring cameras, however, the results were not encouraging as can be seen in Figure 1 showing photos captured from 3 different heights.

(a) High (b) Medium (c) Low

Figure 1: Drone Pictures Samples

Cars are objects that do not have vertical symmetry, that is, a frontal image of a car is very different from an aerial image of the same car. Neural networks are usually trained using front, side and rear views of cars, so classification of aerial photos was not possible. Further investigation on this topic is required.

2.4 Cameras

There are solutions using image processing [1] [3], placing cameras in each parking spot and obtaining other information such as license plate number and vehicle identification. However, collecting all of this data requires an enormous amount of bandwidth and this can be difficult to get.

Solutions using deep learning to classify parking spaces are rare and computationally costly [14]. Our approach uses a minimal implementation of the YOLOv3 network [7] [12]. Yolo applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. This approach makes Yolo classification extremely fast.

3 Methodology

The main idea of our solution is to use cameras to monitor several parking spaces. Photos are taken in a pre-defined interval and a machine learning algorithm is used to detect the empty spaces. This system can be implemented in two different ways:

- **using a cloud server to process the images:** here, the photos taken from the parking lot are sent to a server with great processing power hosted on the cloud. The advantage of this approach is the reduced processing time. However, there are disadvantages such as the high bandwidth requirement for sending images and bureaucracy involved in sending images from public places through the network to a server on the Internet;

- **using a local device to process** in this model, a device such as a Raspberry Pi3 is connected to the cameras and there is no photo upload on the network. This alternative guarantees privacy of the images and is cheaper than the previous solution. On the other hand, the photo classification process is slower. The big challenge is to use a model that has high performance detecting cars but that needs low computational power since the processing is done locally on the device connected to the camera without the aid of large processors and GPUs.

We tested both ways and classification time results are presented on Section 4. The result of this process is the number of available parking spots which can be informed to the users through a totem in the parking lot entrance or a mobile application. Figure 2 shows the architecture of our smart parking solution.

3.1 Scenario

We installed a rotating camera in a parking lot inside Unicamp. Two photo positions sufficed to cover the entire parking lot, 16 spaces, as shown in Figure 3. In parking lots with a good viewing angle, dozens of spaces can be monitored with a single camera. A previous study was made of which places have the best viewing angle in each of the positions. After that, we created a mask for each position, hiding the areas of the photo that we were not interested in classifying. Having done this pre-processing of the photos, these images are inserted into the neural network and we have a number of classified vehicles (cars, motorbikes, trucks, and buses). This number of vehicles is deducted from the total number of places to obtain the amount of free space.

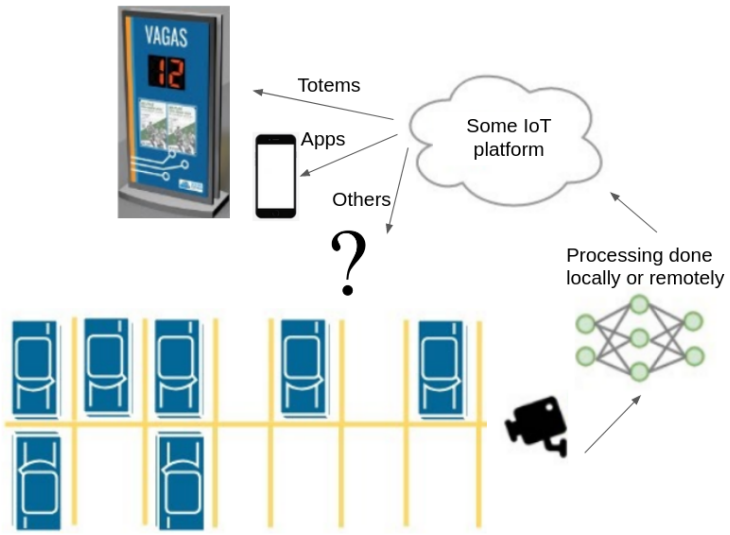


Figure 2: Smart Parking architechture

(a) Position 1 (b) Position 2

Figure 3: Position Samples

3.2 Dataset - collection and filtering

For several weeks, photos were captured for assembly and validation of our own dataset. There are two different photo angles in several parking scenarios (rainy, sunny, cloudy, day, night, crowded and empty).

After obtaining all the photos, manual filtering was done in order to have a problem with the automatic diaphragm of the camera causing overexposure or underexposure in the photos, as shown in Figure 4.

In all, 1000 images were selected, in each image there are 8 possible parking places, in a total of 8000 spaces. We did not train using this dataset, the pre-trained weights of the

(a) Overexposure

(b) Underexposure

Figure 4: Samples of filtered images

network were used, so 100% of it was used in the validation.

3.3 Method

3.3.1 About Pytorch

PyTorch is an open source library for Python programs that facilitates building deep learning projects such as computer vision and natural language processing. It was created by Facebook's AI research group.

Like NumPy, PyTorch is for tensor operations but adds support for GPU and other hardware acceleration and efficient tools for AI researchers to explore different domains.

3.3.2 YOLO

YOLO stands for You Only Look Once. It is an object detector that uses features learned by a deep convolutional neural network to detect an object. It is one of the fastest object detection algorithms currently available. Although it is no longer the most accurate object detection algorithm, it is a very good choice when you need real-time detection, without loss of too much accuracy.

YOLO uses regression; every input image to the model is split into grids and each cell in a grid predicts some bounding boxes and gives a confidence value which indicates how sure the model is that the box contains an object. After this step, the model knows where the object is in that image, but it does not know what that object is. For knowing the class, each cell predicts a class probability using the pre-trained weights. Finally, the box and class predictions are combined to identify the object. Further information can be found at [7] and [11].

3.3.3 Intersection over Union between different classes

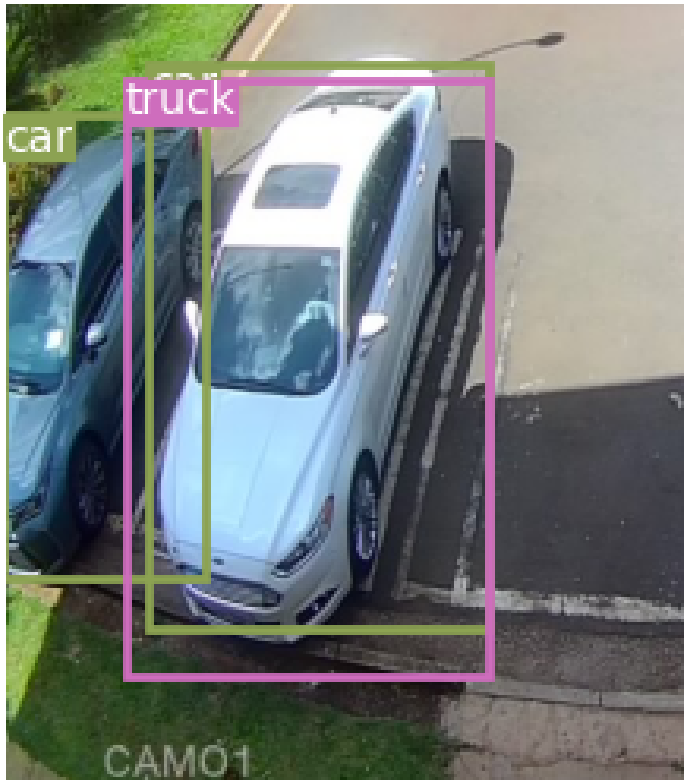


Figure 5: Classification without IoU between classes

Intersection over Union (IoU) is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. Most of the frameworks that create bouding boxes use this metric like YOLO, CNN Based Object detectors, among others. It can be computed as follows:
$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{|I|}{|U|}$

where A and B are the prediction and ground truth bounding boxes. This metric is use to classify the prediction boxes. The higher the score, the better the prediction. It is typically used between predictions of the same type of object to find the best prediction

box.

To the best of our knowledge there is no discussion of Intersection over Union between different classes on the literature. As shown in Figure 5, the network can classify an object as two different classes, such as a car and a truck. To solve this problem, an IoU evaluation metric between classes was added to the network. Instead of comparing with a truth bounding box, the comparison is made with different classes predictions that overlaps. If IoU is more than 0.75, we exclude the prediction that has the lowest level of certainty in the network.

The reason we do this is because we want to add up all the predictions of different classes, like cars, motorcycles, trucks, etc, and, if there are two predictions for the same object, the total vehicle count would be wrong.

3.3.4 Avoiding unwanted classifications

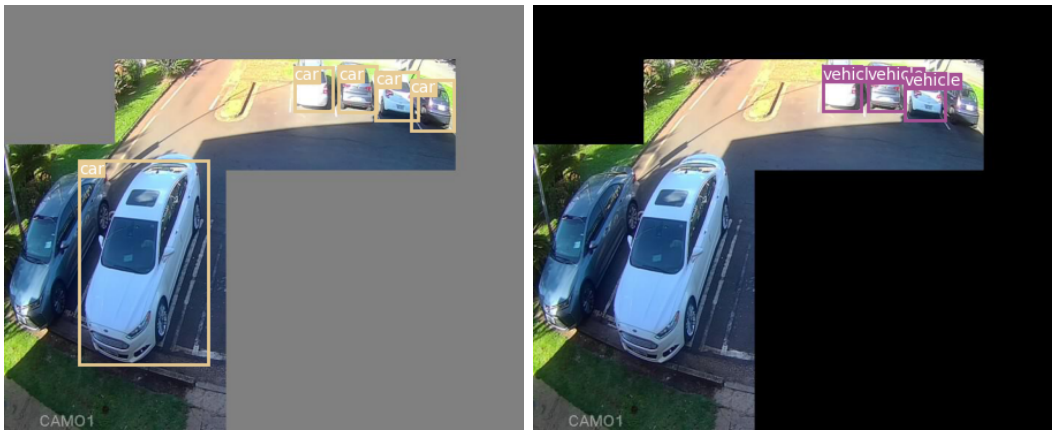
(a) Original picture (b) Picture with a mask

Figure 6: Mask sample

In several parking lots there are places where cars are not allowed to park or there are spaces reserved for the disabled and the elderly. In addition, the camera can be at an angle that also obtains images of cars on the street outside the parking lot. All of these areas should be avoided from classifying.

To avoid classifying cars in unwanted places, we use a mask in the photos as illustrated in Figure 6. The use of these masks, however, influences the classification of the images as shown in Figure 7.

Figure 7 shows that the same network, with the same weights, classifies the photos differently. The reason for this is due to the normalization [8] made in the image by Yolo. Masks with very high contrast in relation to the image will generate errors in the classifications.



(a) Picture with a gray mask (b) Picture with a black mask

Figure 7: Mask limitation sample

4 Results

4.1 Main Results

1000 images were selected to classify 8000 parking spaces. A special care was taken in the formation of this validation dataset so that there were examples of different luminosities and climatic conditions. After the classification we have as a result the confusion matrix in Figure 8.

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be misleading. Let's now define the most basic terms:

- true positives (TP): we predicted an object, and this prediction is true;
- true negatives (TN): we predicted nothing, and there is nothing to predict;
- false positives (FP): we predicted an object, but actually there is nothing to predict;
- false negatives (FN): we predicted nothing, but actually there is a object to predict.

Analyzing the matrix it is possible to infer the following results:
$accuracy = \frac{TP+TN}{TP+FP+TN+FN} = 91.88\%$

$recall = \frac{TP}{TP+FN} = 99.67\%$

$precision = \frac{TP}{TP+FP} = 95.07\%$

$fscore = 2 * \frac{precision*recall}{precision+recall} = 97.32\%$

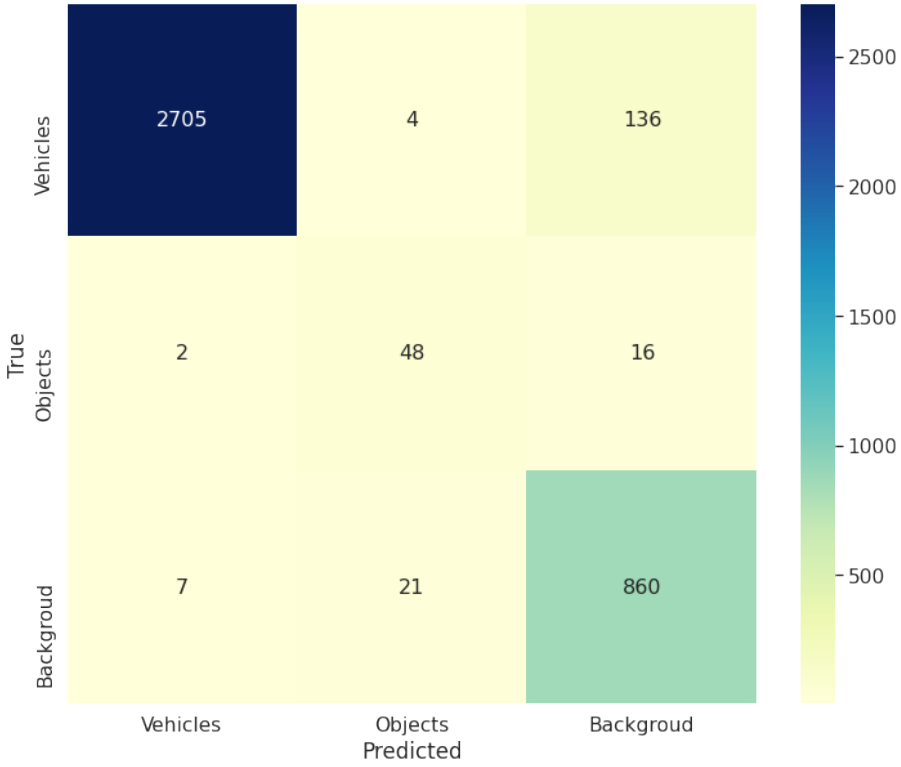


Figure 8: Confusion matrix

4.2 Time Analysis

Here we can compare the classification time using different hardware. The times were obtained with the average time of classification of the images. The remote server was tested using a GTX Titan Black [9].

Time Classification per Picture		
Raspberry PI 3	Raspberry PI 4	Remote Server
24.5s	9.12s	>1s

Table 1: Time Analysis.

Although the classification time on a remote server is very small, the transfer time of the image on the network was not considered because it depends on the quality of the network being used. When a parking lot uses multiple cameras to monitor parking spaces, this time must be multiplied by the number of cameras to obtain the total time.

4.3 Stability Analysis

It is important for the neural network to have stability in its results, to test this, we took 100 sequential photos, that is, all photos are extremely similar and we evaluated whether the network generates the same classification for all photos, whether it is correct or not. Each image set was obtained at three different times of the day (8:00, 14:00 and 20:00) in order to relate the stability of the network to the image's brightness. In total 300 photos were used for this classification.

The percentages presente in Table 2 refer to the accuracy of the classifications, that is, the median of the classified images was said to be true and all others to be false.

Stability during day		
Morning	Afternoon	Evening
98.57%	98.12%	94.31%

Table 2: Stability Analysis.

4.4 Cost

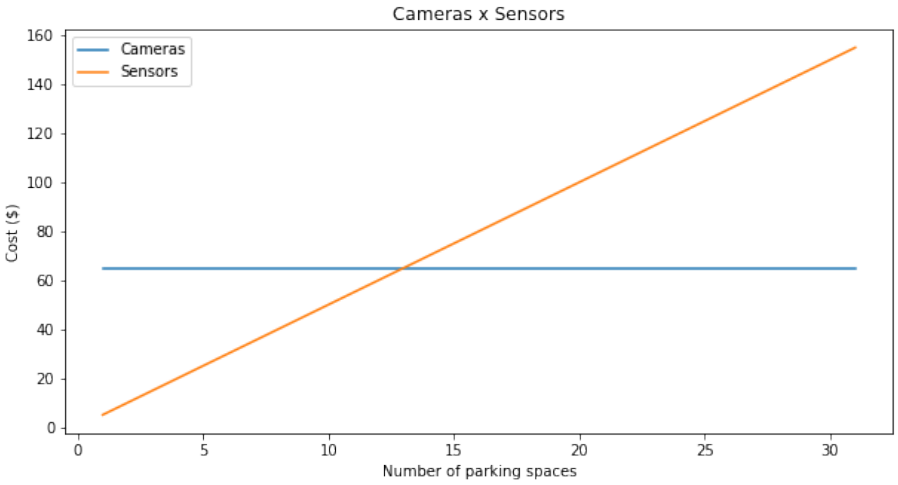


Figure 9: Cost Analysis

Figure 9 shows the estimated costs of the solutions using sensors and cameras in parking lots of different sizes.

We estimated our solution using cameras at a total cost of $ 65 ($ 35 for the raspberry pi4 + $ 30 for the surveillance camera). This estimate may vary slightly depending on the type of camera used. As for the solution using sensors, which is currently the most used commercially, we calculate an average of $ 5 per parking space considering that each space requires a sensor and some information transmission module.

Despite having a higher initial cost, the solution using cameras is less expensive in larger parking lots, since a single camera can monitor a great number of parking spaces.

Another aspect that we can think of is in relation to the maintenance cost. Solutions using sensors have a high maintenance cost due to the high number of modules needed while our solution uses few modules that would eventually need maintenance.

4.5 Scalability

To install the project in a new parking lot, an initial calibration of the camera is necessary, collecting data such as how many spaces that camera is covering and which is the area of legal spaces, that is, places where cars can legally park, preventing the system from accusing available spaces but in improper places.

A previous study of the parking lot as a whole is also necessary to plan the necessary number of cameras and which spaces would be covered by more than one camera to generate cohesive data with each other.

A single-camera can handle several parking spots as long as its angle of view allows it.

4.6 Interface with users

Throughout this project, a prototype was developed inside a parking lot at the Unicamp Computing Institute. A totem with a display was built, as seen in the Figure 10, at the entrance to the parking lot informing how many available spaces are still left.

This totem proved to be valid as a form of user interface in addition to being a way of increasing the visibility of the SmartCampus [13] project.

A mobile application is also being developed as an alternative user interface.

5 Conclusion

In this project we proposed and implemented a smart parking system. The proposed solution overcame disadvantages of solutions using sensors currently available on the market. We dealt not only with the technical part but practical problems such as classifying cars in inappropriate areas were considered. This solution has the potential to grow easily and at low cost.

We also discussed whether it is possible for a famous pre-trained network, in this case YOLO V3, to be applied to a solution in a real scenario without being retrained. We made changes in this network to deal with problems ignored in classification competitions, but very important in a real application like the intersection of classifications of different objects.

In the end, this research overcame the theoretical limitations and implemented a practical solution in a parking lot and currently helps the community of the University of Campinas to save time and find parking spaces.

Figure 10: Display Prototype

Acknowledgment

The work and research presented in this paper was financed by KonkerLabs [5].

References

[1] Sayanti Banerjee, Pallavi Choudekar, and Mohan Muju. "Real time car parking system using image processing". In: *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology* 2 (Apr. 2011). DOI: 10.1109/ICECTECH.2011.5941663.

[2] Matteo D'Aloia et al. "A Marker-Based Image Processing Method for Detecting Available Parking Slots from UAVs". In: *New Trends in Image Analysis and Processing –*

ICIAP 2015 Workshops. Ed. by Vittorio Murino et al. Cham: Springer International Publishing, 2015, pp. 275–281. ISBN: 978-3-319-23222-5.

[3] Md Omar Hasan, Md Islam PhD, and Yazed Alsaawy. "Smart Parking Model based on Internet of Things (IoT) and TensorFlow". In: June 2019, pp. 1–5. DOI: 10.1109/ICSCC.2019.8843651.

[4] INRIX. *Parking Pain research*. 2017. URL: https://inrix.com/press-releases/parking-pain-us/ (visited on 04/07/2020).

[5] Konker. *Konker IoT Solutions*. URL: http://www.konkerlabs.com/.

[6] T. Lin, H. Rivano, and F. Le Mouël. "A Survey of Smart Parking Solutions". In: *IEEE Transactions on Intelligent Transportation Systems* 18.12 (2017), pp. 3229–3253.

[7] Erik Linder-Norén. *Minimal PyTorch implementation of YOLOv3*. 2018. URL: https://github.com/eriklindernoren/PyTorch-YOLOv3 (visited on 04/07/2020).

[8] "Normalization image processing". In: *Wikipedia*. Wikimedia, 2020. URL: https://en.wikipedia.org/wiki/Normalization_(image_processing.

[9] Nvidia. *GTX Titan web page*. URL: https://www.nvidia.com.br/gtx-700-graphics-cards/gtx-titan-black/.

[10] M. S. Rahman, Y. Park, and Ki-Doo Kim. "Relative location estimation of vehicles in parking management system". In: *2009 11th International Conference on Advanced Communication Technology*. Vol. 01. 2009, pp. 729–732.

[11] Joseph Chet Redmon. *Yolo web page*. URL: https://pjreddie.com/darknet/yolo/.

[12] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[13] SmartCampus. *SmartCampus Unicamp web page*. URL: https://smartcampus.prefeitura.unicamp.br/.

[14] S. Valipour et al. "Parking-stall vacancy indicator system, based on deep convolutional neural networks". In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 2016, pp. 655–660.