Technical University of Berlin

Fraunhofer Institute for Open
Communication Systems FOKUS

Faculty 4

Projekt Vertiefung / Vernetztes und
Automatisiertes Fahren
Summer term 2021

DCAITI Project Documentation

Advanced detection of static and dynamic free space for parking lot occupancy with OpenCV and YOLO

Team members:

Peng Zhang (415537)

Yihong Sheng (415172)

Di He (414832)

Chenrui Fu (350659)

Supervised by:

Marcus Witzke

August 8, 2021

Contents

1 Abstract

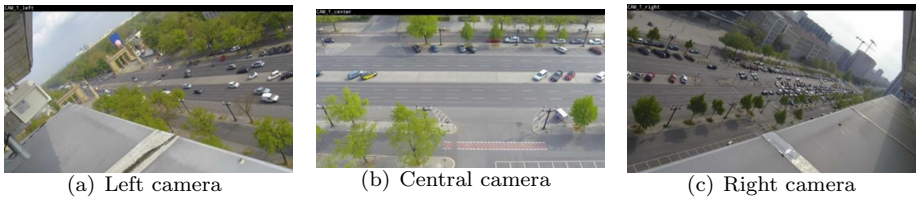
As the world's population continues to increase, the number of private vehicles has also increased, and the demand for parking lots has also increased. Although the parking problem is generally regarded as the main problem of the car power, this problem is indeed a global problem. With the improvement of global living standards and the acceleration of urbanization, especially in India and China, it is obvious that the number of private vehicles is increasing rapidly, and the demand for parking spaces is also increasing. In India, the number of private vehicles increased by nearly 400% between 2001 and 2015, and the number of private vehicles increased from 55 million to 210 million. As of 2017, there was a shortage of 50 million parking spaces in China.[16]

addition to the parking problem, the time it takes to find an available parking space is also a considerable problem. Imagine if you and your family have booked movie tickets to watch your favorite movies on the weekend. You and your family are looking forward to this day. When this day comes, you and your family are excited to drive to the movies. But the parking space in the movie theater is full. You spend a lot of time looking for a free parking space and miss the opening scene of the movie. What a disappointment this is.[10] This is undoubtedly a very serious problem in today's fast-paced life. For the rapid increase in the number of private vehicles and the demand for parking spaces, it is very important to detect free parking spaces quickly and accurately. And computer vision combined with neural network can accomplish this job well. Through neural network deep learning to accurately identify the target that needs to be detected. Combine the OpenCv library to build a computer vision system. So as to detect the target quickly and accurately.

This project uses the above technology to detect the free parking spaces in front of the TC building of the Technical University of Berlin, and will continuously update the parking space information.

2 Introduction

The task of this project is to use the three cameras in front of the TC building as visual sensors to detect the parking spaces in front of the building. We mainly use the central camera to accomplish these tasks. We divide the detected parking spaces into two types, one is a static parking garage with white lines, and the other is a dynamic parking garage that is not divided in the middle of the road.



(a) Left camera (b) Central camera (c) Right camera

Figure 1: Three cameras

(a) Type 1: Static parking space

(b) Type 2: Dynamic parking space

Figure 2: Two types of parking spaces

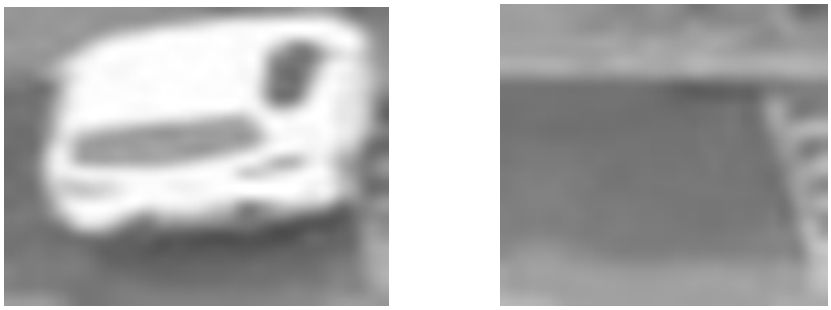
But through our review of the code of the earlier group and analysis of the methods they used, we found that the earlier group has completed these tasks very well. So we discussed with our project leader, and our project leader suggested that we integrate the code of the earlier group and optimize the methods they use. This is our latest task. Therefore, we once again analyze the methods used in the earlier group.

2.1 Earlier Group Method

We will briefly introduce the methods of the earlier groups to be merged and optimized.

2.1.1 JingGu Group

For static parking spaces, JingGu uses fully connected Neural Network. Training is carried out through keras, and the data set uses the MNIST data set. The training data is in 28x28 pixel image format. The original data is divided into two categories, one is occupied parking spaces, and the other is free parking spaces.



(a) Occupied parking spaces (b) Free parking space

Figure 3: Two categories of original data[13]

For dynamic parking spaces, the car part is turned into white by the K-means algorithm, and the background part is turned into black. Draw a line in the selected area, this line will produce an intersection with the car outline, and the black area between the two intersections is the free parking space.

Figure 4: Mask of free parking Space[13]

2.1.2 Lennart Group

Lennart's method of detecting dynamic parking spaces is similar to JingGu's. Draw two parallel straight lines in the selected area, the intersection of the straight line and the car can get a parallelogram, and the black parallelogram

area is the free parking space.

Figure 5: Detect free parking spaces

2.1.3 Magdalena Group

Magdalena's method is relatively simple, he divides the selected area into equal size.

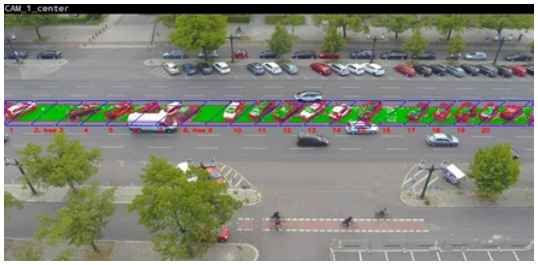


Figure 6: Mark dynamic parking spaces[19]

3 Basic of OpenCV

OpenCV,which is short for Open Source Computer Vision Library. It is an open source computer vision and machine learning software library. It has many optimization algorithms. These algorithms can detect and recognize the target, classify the target, find similar images from the database according to the target, and so on. It is the most widely used open source toolkit in the field of early stage computer vision and image processing. It is based on C/C++, supports Linux/Windows/MacOS, and also provides interfaces for languages such as Python, Matlab, and Java. Due to its rich interfaces, excellent performance and business-friendly license, OpenCv is very popular in academia and industry.[1]

OpenCV also has a modular structure, which means that it contains many shared libraries or static libraries. Its available modules include core function modules, image processing modules, video analysis modules, camera calibration and 3D reconstruction modules, 2D feature framework modules, object detection modules, advanced GUI modules and some other auxiliary modules.[2]

OpenCV can be used in many applications, such as Background subtraction ,which will be used in the project, to distinguish occupied parking areas and free parking areas.

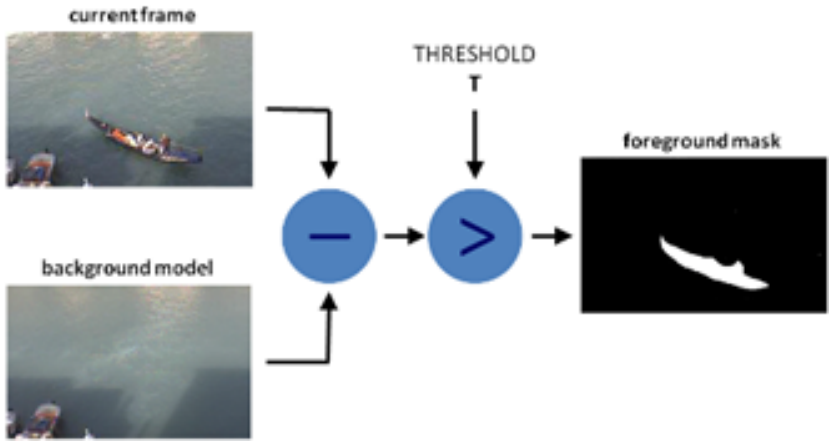


Figure 7: Background subtraction[3]

The background subtraction technique is widely used, and the foreground mask is generated by using a static camera. Perform a subtraction operation between the current frame and the background model, which includes the static part of the scene, and can also treat all content as a scene based on the detected scene.[3]

And in the modul Background subtraction of opencv, There are specific modules based on different algorithms, such as KNN based on K-nearest neighbours, and MOG2 based on Gaussian Mixture. We can choose the algorithm module with the best result according to the specific situation .

4 Geo-coordinate

4.1 Theory

The parking area in the video of this project can be regarded as three straight lines. Because two points can determine a straight line, if the geographic coordinates and pixel coordinates of two points on each line are known, the geographic coordinates of other points can be calculated from their pixel coordinates.

4.2 Implementation

4.2.1 Get Geo-coordinates from Google-map

we get geo-coordinates of 4 points of static parking area in the camera image by google-map, they are top-left parking lot , bottom-left parking lot, bottom-right parking lot and top-right parking lot. Because the image presented by the camera in the teaching building and the Google map are upside down, we need to rotate the Google map 180 degrees to obtain the geographic coordinates.

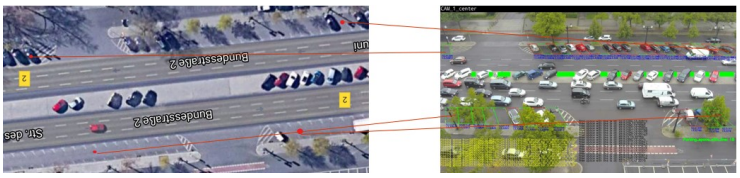


Figure 8: Map image and camera image

4.2.2 Get center Point of each Parking Lot

The coordinate of the center point of each parking lot is regarded as the its pixel coordinate. The existed database contains the pixel coordinates of the four vertices of each parking lot. The two diagonals of one parking lot can be calculated with four vertices, so the intersection of two diagonals is the center point of the parking lot.

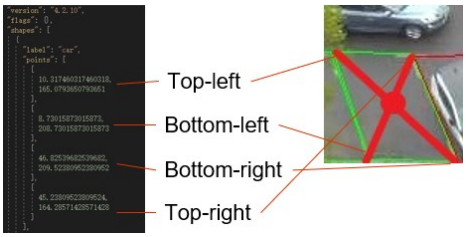


Figure 9: Pixel coordinate of four corner and the center point

4.2.3 Get Geo-coordinate of 2 Points in dynamic Parking Area

We create 2 specific points in the dynamic parking area, because there is no reference point in dynamic parking area. In the map image(figure10) ,the dynamic parking area is the symmetric axis of the static parking area on the both side, the x value of the specific point on the east side , which is longitude , is half of the sum of x3 and x4, and the y value, which is latitude, is half of the sum of y3 and y4. x-west is half of the sum of x1 and x2, y-west id half of the sum of y1 and y2.

$$x_{\text{east}} = \frac{x_3 + x_4}{2} \tag{4.2.1}$$

$$y_{\text{cast}} = \frac{y_3 + y_4}{2} \tag{4.2.2}$$

$$x_{\text{west}} = \frac{x_1 + x_2}{2} \tag{4.2.3}$$

$$y_{\text{west}} = \frac{y_1 + x_2}{2} \tag{4.2.4}$$

Figure 10: Specific points on map image

In camera image (figure11), y pixel coordinate of dynamic parking area is 310, and the pixel coordinates of four corners. So we can use the similar triangle method to calculate x pixel coordinate values of two specific points in the dynamic area.

$$y_{eastp} = 310 \tag{4.2.5}$$

$$x_{eastp} = \frac{310 - y_{3p}}{y_{4p} - y_{3p}} \cdot (x_{4p} - x_{3p}) + x_{3p} \tag{4.2.6}$$

$$y_{westp} = 310 \tag{4.2.7}$$

$$x_{westp} = \frac{310 - y_{2p}}{y_{1p} - y_{2p}} \cdot (x_{1p} - x_{2p}) + x_{2p} \tag{4.2.8}$$

Figure 11: Specific points on camera image

4.2.4 Calculation of Geo-coordinate of each Parking Space

The static parking spaces on the both sides and the dynamic parking space in the middle are three straight lines, the geo-coordinates and the pixel coordinates of two points on each straight line are known, So if the pixel coordinate of the parking space is known, the geo-coordinate of this parking space can be calculated.

4.2.5 Add Information to the Video

After we get geo-coordinates of all parking space, we add this information to the frame of video. Blue texts show the geo-coordinate of each parking lot of static parking area. Black texts show the geo-coordinates of free parking lots of static parking area. We get the geo-coordinates of the free parking area by calculating the geo-coordinates of the boundary points of the car outline. The yellow texts show the geo-coordinate interval of free area and the number of parking on dynamic parking area.

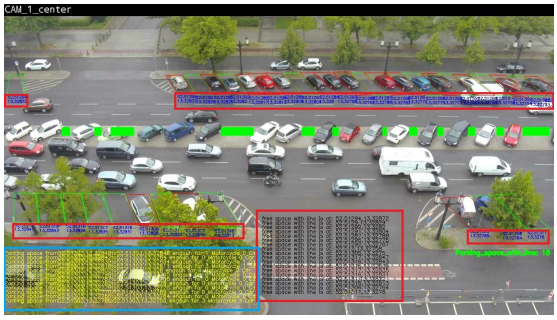


Figure 12: Geo-coordinate information on the frame

5 Automatic Adjustment

5.1 Target

Sometimes, the camera shakes slightly due to wind and other factors. At this time, the pixel coordinates of each parking space in the picture will also change, so the calculated geographic coordinates will also have errors. In order to avoid this situation, we want to write a program that can automatically adjust the geographic coordinates of the parking space when the camera shakes slightly.

5.2 Theory

A quadrilateral composed of four points can represent a plane, and the change of the four corner points can represent the change of the entire plane. So when the frame changes due to the shaking of the camera, we can calculate the changes of other points by measuring the changes of the four intersections of the quadrilateral.

5.3 Implementation

Step1, we manually get the four endpoints of two white lines on the road of the original image. The red area in fig is the operation area, the white points are clicked points. After we clicked these four points, we know the pixel coordinates of this four endpoints, e.g. (87,231),(130,232),(67,254),(113,256).

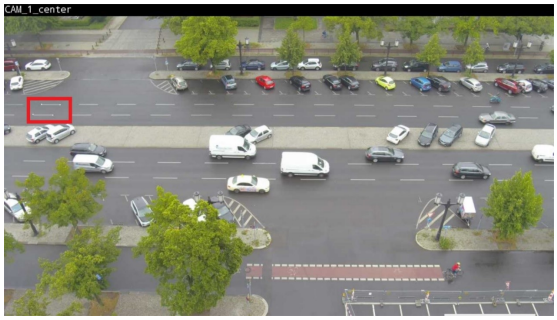


Figure 13: Operation area and clicked points

```
clicking:  87 231
clicking:  130 232
clicking:  67 254
clicking:  113 256
```

Figure 14: Pixel coordinate of clicked points

Step2, we use opencv to detect the endpoints of the operating area and use k-means to put these points into four clusters. Finally, we take the average of each cluster. The four average values are the four points pixel coordinate by detecting.

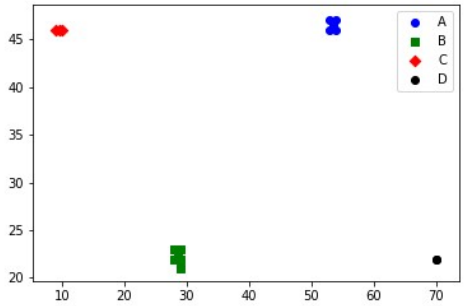


Figure 15: K-means of endpoints

Figure 16: Detected points on the frame

Step3, we compile constraint code to ensure that detected points are 4 endpoints. Because sometimes this area is covered by cars.

Step4, we calculate the new location information of each parking space. These four points can form a plane. The change of this plane can represent the change of the whole frame. So we can calculate the coordinates of each parking space after the camera moves by comparing the changes of these four points.

Step5, Compile constraint code to offset the detection error. There are some slight errors in opencv's detection and kmeans. So we set a confidence interval to overcome these errors.

6 Detect free Parking Lots on static Parking Area with DNN

6.1 Theory of DNN-MNIST

MNIST is a data set of handwritten digits. It contains about 250 pictures of personal handwritten digits (0 to 9). There are about 70,000 pictures in total, including 6,000 training pictures and 10,000 test pictures. Half of the training set and half of the test set are taken from the NIST training data set, and the other half of the training set and test set are the NIST test data set. The format of the pictures are basically processed, and they are all grayscale pictures with a size of 28*28.[4]

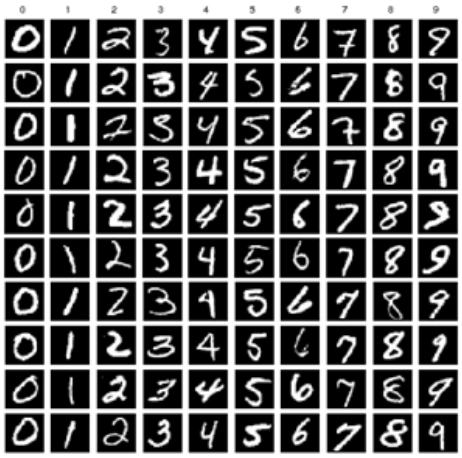


Figure 17: MNIST handwritten grayscale picture[5]

6.2 Optimization

Some parking spaces are partially obscured by trees, so it is not feasible to use general parking space size areas for DNN recognition. Some occupied parking spaces are detected as free status, because they are obscured by trees. We use if statement to find the specific parking spaces which are obscured by tree, and then reduce their detection area.

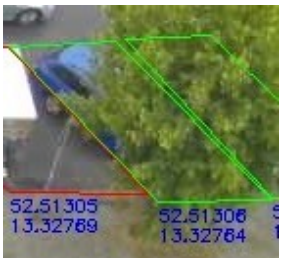


Figure 18: Parking status before optimization

Figure 19: Parking status after optimization

7 Detection of free Space on dynamic Parking Area by Segmentation

7.1 Theory of Segmentation

Image segmentation is to divide an image into corresponding areas or categories To different objects or parts of objects. Each pixel in the image is assigned to one of these categories. A good segmentation usually meet the following conditions: first, Adjacent pixels of different categories have different values, second, pixels of the same category have similar multi-value gray levels and form a connected area.

The existing image segmentation methods are mainly divided into the following categories: threshold-based segmentation methods, region-based segmentation methods, edge-based segmentation methods, and segmentation methods based on specific theories. From a mathematical point of view, image segmentation is the process of dividing a digital image into disjoint areas. The process of image segmentation is also a marking process, that is, the pixels belonging to the same area are assigned the same number.

7.1.1 The gray Threshold Segmentation

The gray threshold segmentation method is one of the most used parallel region techniques, and it is the most widely used category in image segmentation. The threshold segmentation method is the following transformation from the input image f to the output image g:

$$g(i,j) = \begin{cases} 1, & f(i,j) \geq T \\ 0, & f(i,j) < T \end{cases} \tag{7.1.1}$$

Among them, T is the threshold, $g(i,j) = 1$ for the image element of the object, and $g(i,j) = 0$ for the image element of the background.

The key to the threshold segmentation algorithm is to determine the threshold. If an appropriate threshold can be determined, the image can be accurately segmented. After the threshold value is determined, the threshold value and the gray value of the pixel points are compared one by one, and the pixel segmentation can be performed on each pixel in parallel, and the result of the segmentation is directly given to the image area.

The advantages of threshold segmentation are simple calculation, high computational efficiency and fast speed. It has been widely used in applications where computational efficiency is important .

Various threshold processing techniques have been developed, including global thresholds, adaptive thresholds, optimal thresholds, and so on.

Global threshold refers to the use of the same threshold for segmentation of the entire image, which is suitable for images with obvious contrast between the background and the foreground. It is determined based on the entire image: $T = T(f)$. However, this method only considers the gray value of the pixel itself, and generally does not consider the spatial characteristics, so it is very sensitive to noise. Commonly used global threshold selection methods include peak-valley method using image gray histogram, minimum error method, maximum between-class variance method, maximum entropy automatic threshold method, and other methods.

In many cases, the contrast between the object and the background is not the same everywhere in the image, and it is difficult to use a uniform threshold to separate the object from the background. At this time, different thresholds can be used for segmentation according to the local characteristics of the image. In actual processing, the image needs to be divided into several sub-regions to select thresholds according to specific problems, or to dynamically select the threshold at each point according to a certain neighborhood range to perform image segmentation. The threshold currently is an adaptive threshold.

The selection of the threshold value needs to be determined according to specific problems and is generally determined through experiments. For a given image, the best threshold can be determined by analyzing the histogram. For example, when the histogram is obviously bi-modal, the midpoint of the two peaks can be selected as the best threshold.

The following figure are the results of segmentation with global threshold and adaptive threshold.[6]

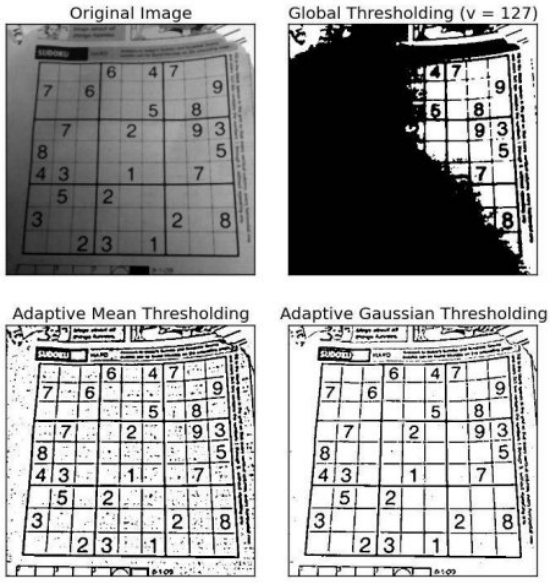


Figure 20: Segmentation with global threshold and adaptive threshold.

7.1.2 Region Segmentation

Region-growth and Split and merge segmentation are two typical serial region technologies. The subsequent steps of the segmentation process should be determined based on the results of the previous steps.

The basic idea of region growth is to group pixels with similar properties to form regions. Specifically, first, find a seed pixel for each region that needs to be segmented as the starting point for growth, and then select pixels that have the same or similar properties as the seed pixel in the neighborhood around the seed pixel (determined according to a certain predetermined growth or similarity criterion) Merge into the area where the seed pixel is located. Treat these new pixels as new seed pixels and continue the above process until no more

pixels meeting the conditions can be included. Such an image area is grown.

Regional growth needs to select a set of seed pixels that can correctly represent the desired region, determine the similarity criteria in the growth process, and formulate conditions or criteria to stop the growth. The similarity criterion can be characteristics such as gray level, color, texture, and gradient. The selected seed pixel can be a single pixel or a small area containing several pixels. Most area growth criteria use the local nature of the image. Growth criteria can be formulated according to different principles, and the use of different criteria will affect the process of regional growth. The advantage of the region growing method is that it is simple to calculate and has a better segmentation effect for more uniform connected targets. Its disadvantage is that the seed point needs to be determined artificially, which is sensitive to noise and may cause holes in the area. In addition, it is a serial algorithm. When the target is large, the segmentation speed is slow. Therefore, when designing the algorithm, it is necessary to improve the efficiency as much as possible.

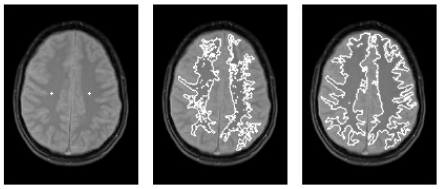


Figure 21: Regional growth segmentation of Brain MRI image.

Region growth starts from a certain or some pixel points, and finally obtains the entire region, and then achieves target extraction. Splitting and merging segmentation is almost the inverse process of region growth: starting from the entire image, continuously splitting to obtain each sub-region, and then combining the foreground regions to achieve target extraction. The hypothesis of splitting and merging segmentation is that for an image, the foreground area is composed of some interconnected pixels. Therefore, if an image is split to the pixel level, then it can be determined whether the pixel is a foreground pixel. After all pixels or sub-regions are judged, the foreground target can be obtained by combining the foreground regions or pixels.

Among these methods, the most used method is the quad-tree method.

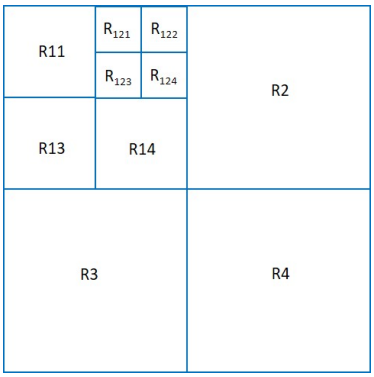


Figure 22: Quad-tree method.

The basic split and merge algorithm steps are as follows: Define criteria for homogeneity and divide the image into equal-sized regions,then calculate the homogeneity of each region and if the area is homogeneous, merge it with neighbors.Repeat the process until all areas pass the homogeneity test.[17]

The key to the split and merge method is the design of split and merge criteria. This method has a better segmentation effect on complex images, but this algorithm is more complex and computationally expensive, and the split may also destroy the boundary of the region.

The figure blow is the region segmentation with Spilt-merge and region growth.And each image shows the result after each steps of the segmentation: Original image (a), feature description image (b), image after the splitting process (c), image after the merging process (d), elimination of small regions (e), and the resulting segmentation classes obtained after the region growing procedure (f)

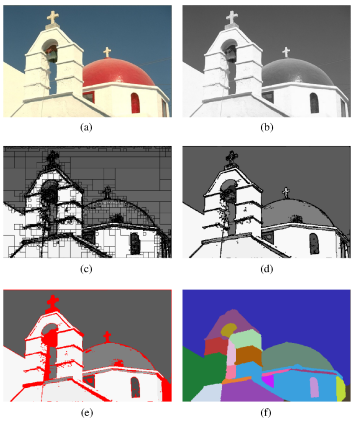


Figure 23: Region segmentation with Spilt-merge and region-growth.[12]

7.1.3 Edge Segmentation

An important way of image segmentation is through edge detection, that is, detecting the place where the gray level or structure has abrupt changes, indicating the end of one area and the beginning of another area. This discontinuity is called an edge. Different images have different gray levels, and there are generally obvious edges at the boundaries. This feature can be used to segment the image.

The gray values of the pixels at the edges of the image are discontinuous, and this discontinuity can be detected by taking the derivative. For a step-shaped edge, its position corresponds to the extreme point of the first derivative and the zero-crossing point (zero crossing point) of the second derivative. Therefore, differential operators are often used for edge detection. Commonly used first-order differential operators include Roberts operator, Prewitt operator and Sobel operator, and second-order differential operators include Laplace operator and Kirsh operator. In practice, various differential operators are often represented by small-area templates, and differential operations are realized by using templates and image convolution. These operators are sensitive to noise and are only suitable for images with low noise and less complex images.

Since edges and noise are gray discontinuous points, and both are high-frequency components in the frequency domain, it is difficult to overcome the influence of noise by directly using differential operations. Therefore, the image must be smooth-filtered before the edge is detected by the differential operator. LoG operator and Canny operator are second-order and first-order differential operators with smoothing function, and the edge detection effect is better. Among them, LoG operator uses Laplacian operator to get the second derivative of Gaussian function, Canny operator is the first derivative of Gaussian function, it has achieved a good balance between noise suppression and edge detection.[18]

Figure 24: Edge segmentation with canny operator.[7]

7.2 K-means Clustering

K-means clustering algorithm is an iterative solution clustering analysis algorithm,The step is to pre-divide the data into K groups, then randomly select K

objects as the initial cluster centers.Then calculate the distance between each object and each cluster center and assign each object to the cluster center closest to it. The cluster centers and the objects assigned to them represent a cluster. Each time a sample is allocated, the center of the cluster will be recalculated based on the existing objects in the cluster.This process will be repeated until a certain termination condition is met. The termination condition can be that no (or minimum number) objects are reassigned to different clusters, no (or minimum number) cluster centers change again, and the sum of squared errors is locally minimum.

There is a well-known explanation: Priest - Villagers model. it can well explain the principles of the K-means

There are K pastors $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ who go to the suburbs to preach, and choose k locations so that each villager $(x_1, x_2, ...x_n)$ can find the nearest preaching point to him: Given a set of observations $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$, where each observation is a d-dimensional real vector, k-means clustering must divide these n observations into k sets to minimize the sum of squares in the group,$\boldsymbol{\mu}_i$ is the mean of points in $\boldsymbol{S}_i$

$$\underset{S}{arg\,min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{7.2.1}$$

At first, the pastors randomly selected several preaching spots and announced the situation of these preaching spots to all the villagers in the suburbs: Select the initial k samples as the initial cluster centers: Select the initial k samples as the initial cluster centers: $m_1^{(1)}, ..., m_k^{(1)}$

So, every villager went to the sermon point closest to his home to attend the class: Assign each observation to the cluster so that the sum of squares within the group (WCSS) is minimized.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \forall j, 1 \leq j \leq k \right\} \tag{7.2.2}$$

After attending the class, everyone felt that the distance was too far, so each pastor counted the addresses of all the villagers in his class, moved to the center of all the addresses, and updated the position of his sermon on the poster: For each cluster obtained in the previous step, use the center of the observations in the cluster as the new mean point.

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_j \in S_i^{(t)}} x_j \tag{7.2.3}$$

Every time the pastor moves, it is impossible to get closer to everyone. Some people find that after Pastor S_1 moves, they might as well go to Pastor S_2 to

listen to a class closer, so each villager goes to the nearest sermon point...

In this way, the pastor renewed his position every week, and the villagers chose the preaching point according to their own situation, and finally stabilized: Repeat the above two steps until a certain termination condition is reached (number of iterations, minimum error change, etc.).

The k-means method has the following advantages: it is easy to understand, and the clustering effect is good. Although it is a local optimum, it is often sufficient; When dealing with large data sets, the algorithm can ensure better scalability; When the cluster is approximately Gaussian, the effect is very good; Algorithm complexity is low.

It also has many shortcomings: K value needs to be set manually, and different K values get different results; Sensitive to the initial cluster center, different selection methods will get different results; Sensitive to outliers; The sample can only be classified into one category, which is not suitable for multi-classification tasks; Not suitable for too discrete classification, unbalanced sample classification, and non-convex shape classification.[8]

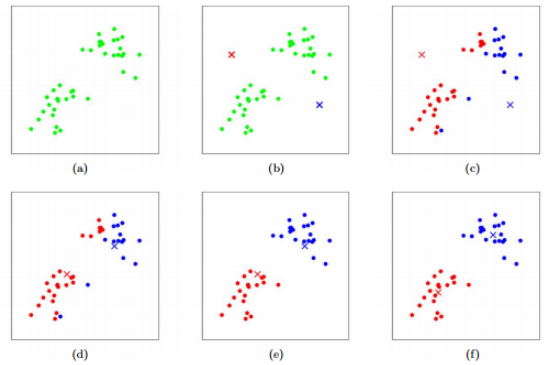


Figure 25: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses.[9]

7.3 Previous Work

Jing Gu intercepted the dynamic parking area in the middle of the picture and processed it with opencv, She uses k-means to divide the pictures into two categories, one is cars and the other is the ground. Then she detects the edge of the car and draws a line on the centerline of the intercepted area. The intersection of the center line and the edge of the car is the boundary point of each car.

7.4 Optimization

Jing Gu used K-means to divide the dynamic parking area into two segments, she cut the complete dynamic parking area for segmentation(fig26). But it is not a top view, the cars on the street below will cause errors.

Figure 26: Segmentation with complete middle area

To avoid it, we raised the bottom border of the cropped picture which is used for segmentation up(fig27). This will reduce the interference between segmentation and edge detection of cars on the road.

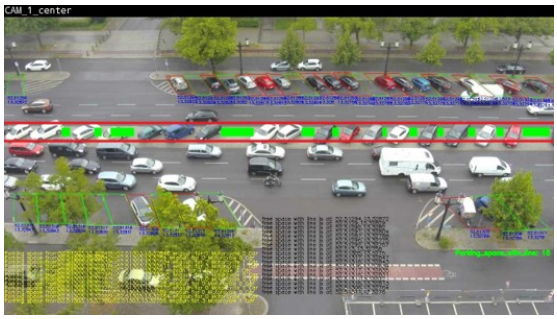


Figure 27: Segmentation with raised bottom border area

8 Basic of YOLO

You only look once (YOLO) is an approach to object detection[15]. In the past object detection used repurposing of classfiers to do the detection. But YOLO takes the detection as a regression problem with separating the bounding boxed and class probabilities spatially[14]. YOLO is a neural nework, which predicts the bounding boxes and class probabilities. These prediction is basic on the

evaluation of the full image. YOLO is a unified and real-time detector.

8.1 Structure of YOLO

YOLO is inspired By GoogleNet. In the YOLOv1 it has 24 convolutional layers, after that it is 2 fully connected layers. YOLO uses a "flatten" funktion, that is a 1*1 reduction layer follwed by 3*3 convolutional layers. The full layer frame is shown in the Figuer 28. The final output is a 7*7*30 tensor.

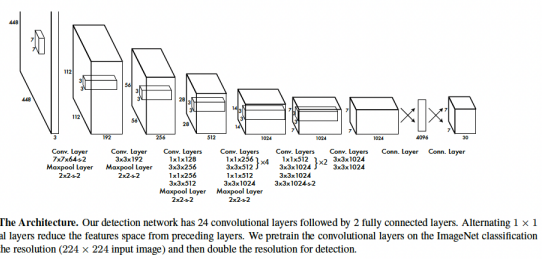


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 28: The Structure of YOLOv1[14]

8.2 The key Point of YOLO

The detection Process can be divided into 5 steps:

1. Dividing the input image into S*S grids. If an center of object falls into one grid, then this grid will detect this object. 29

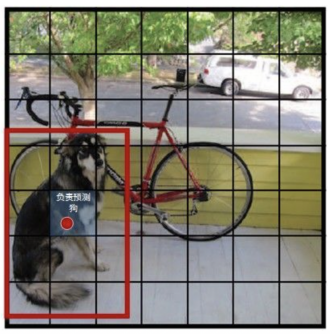


Figure 29: Detect grid of YOLOv1[11]

2. Each grid will predict B bounding boxes and C class possibilities Pr(class—object). C is the total numbers of net class. It is decided in the training. In YOLO every

grid estimate only one class.

3. every bounding box has a confidence scores.

$$confidence = Pr(Object) * IOU_{truth}^{pred} \tag{8.2.1}$$

If the center of object fall into the grid then Pr(object)=1; or Pr(Object)=0. The IOU is intersection over union between bounding box and ground truth.

In every bounding box there are five prediction parameters: (x, y, w, h, confidence). x and y is coordinate of the left upper point of this box. w and h is the width and height of the bounding box.

4. The output of the gird is S*S*(5*B+C)

5. Use NMS(non maximum suppression) to filter some not exact box also make the outlook better.

8.3 The Normalization of Bounding Box

The normalization of the coordinate parameter is very important to YOLO. Like the detail in Image30 the input picture will be divided into S*S grids. The image width and height will be singed in widthimage and heightimage. the width and height of bounding box will be singed in widthbox and heightbox. The coordinate of the center of the object is (Xc, Yc)

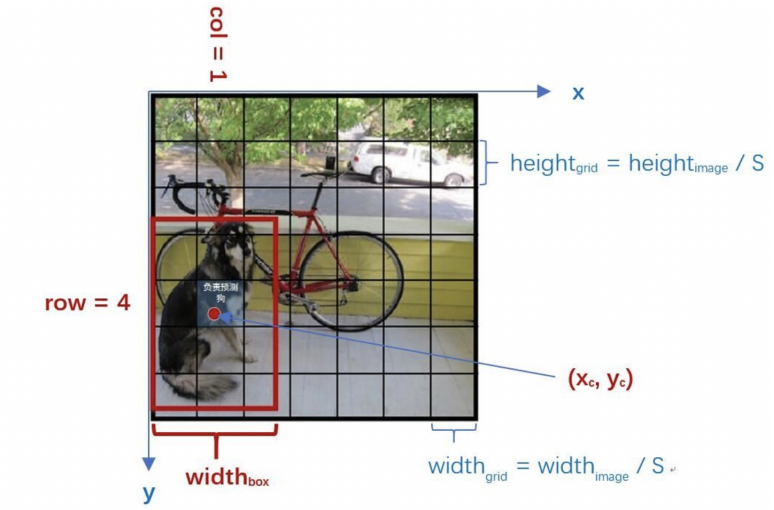


Figure 30: Normalization of YOLOv1[11]

The normalization of bounding box is to make the output between 0 and 1:

$$w = \frac{width_{box}}{width_{image}} \tag{8.3.1}$$

$$h = \frac{height_{box}}{height_{image}} \tag{8.3.2}$$

The use offset to normalize the center coordinate of the bounding box:

$$x = x_c \cdot \frac{S}{width_{image}} - col = \frac{x_c}{width_{grid}} - col \tag{8.3.3}$$

$$x = y_c \cdot \frac{S}{height_{image}} - row = \frac{y_c}{height_{grid}} - row \tag{8.3.4}$$

Row and col is the center of bounding box to all the grids.
with the over four equations we can get the coordinate of the bounding box after normalization(x, y, w, h). Then add the "confidence", which has already mentioned before, we can get a bounding box, which will used in the real regression net.

8.4 Training of YOLO

When training the detector, loss of prediction is always a important factor. YOLO take these three aspects into consideration.
First is the loss of prediction of coordinate of bounding box. In the past detection experience, we can find, when we use different size bounding box to prediction, the same loss to the smaller box can not be accepted compared to the same loss of a bigger box. So YOLO uses the square root to make the loss acceptable.

Second is the prediction loss of the confidence. Due to the most grids contain none objects, so YOLO will give a weight to make the prediction better $\lambda_{noobj} = 0.5$. And YOLO also give a weight $\lambda_{coord} = 5$ to the coordinate to make a balance between the loss of coordinate and confidence.
Third is the loss of class prediction.

$$\begin{aligned} Loss = &\lambda_{coord} \cdot Loss_{coord} + (LossofConfidence_{obj} + \\ &\lambda_{noobj} \cdot LossofConfidence_{noobj}) + Loss_{class} \end{aligned} \tag{8.4.1}$$

8.5 Error Analysis

When YOLO compare to R-CNN, there are some differences:

YOLO focus on the accuracy of location, so it will give more account for location error. Fast R-CNN take more care of the Backgrounds, so it can detect the

background quicker.

The detail of Error Analysis is in figure31

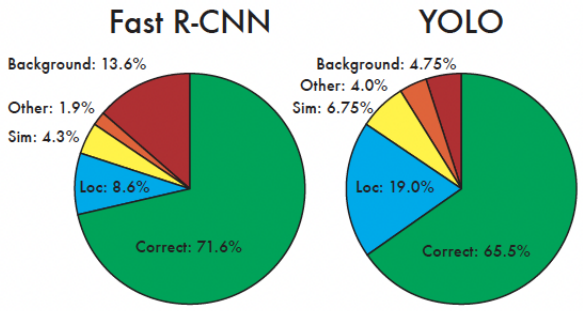


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Figure 31: Comparison of YOLOv1 and Fast R-CNN[14]

9 Detection of free Space on dynamic Parking Area by YOLO

From the first plan we used open-cv to detect the cars in the middle dynamic parking area. Then we want to try YOLO combined with our current work to make the accuracy of detection better.
With the consideration of stability and calculation speed, we choose YOLOv3 to be the detector.
We find, that the project from Monika can be a reference.

9.1 Abstract the useful Part

When we first get the project code from Monika, the code can not be runned. After we fixing the bug from projects and delete the useless codes the programm finally can be used. The result is showed in figure32

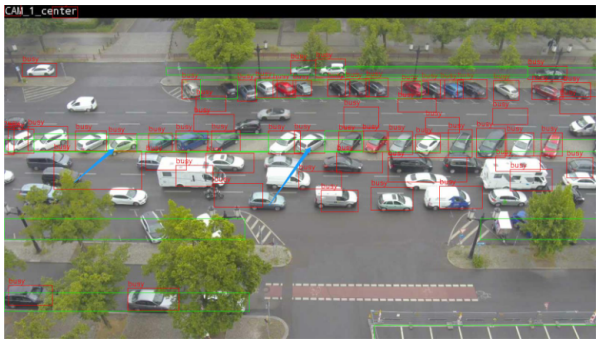


Figure 32: Project result from Monika

It is obviously, that the accuracy of detect is very poor. And the output result is hard to get the useful information. User can not get the number position of parking car in the middle parking area clear. The detect error on the roads would also be a factor, which easily makes the judgement from users to wrong. After analysis of the error detection we abstract the useful part to our work from this program. It is the combination between YOLOv3 and drawing bounding box. They consider the part of video reading very exhaustive, so we also take some codes as a reference.

9.2 Combination with Current Work

When we try to combine the two codes together, the first difficulty is how to scratch the middle dynamic parking space as the only input. YOLO will detect the full input source. We can not select point from the video as the input area, we can not also just take the middle detect result to connect with our work, because in the same time the program can only take one detector to calculate. So we get an solution:

We let the input information outside the middle parking space as blank.
Then we can get a pure middle parking space input source. After that we use YOLO to detect the input information. It works fine(figure33).

Figure 33: Middle Parking Space Input

Finally we can add the code in our program without worries.
We calculate twice with different sources. One is the full input video source with our current work(the one with geo-coordinates and output texts), it will detected by dnn. The other is this middle parking area detected information, this will detected by YOLOv3.

9.3 Calculate the Rest Parking Space with more Accuracy

In our current work the rest parking space can not be calculate exactly, because the open-cv is with the help of color difference to detect the car and segment them. But there would be some cars with the similar or same color with the parking slots. That would be a big difficulty for detector to distinguish them, then the error will appear. However when we with the help of YOLO to determine the position of parking cars, we can calculate the rest parking space better.

As we mentioned in the previous subsection "The Normalization of Bounding Box" the bounding box has five key parameters. The first four parameters is the information of the bounding box, the coordinate of the upper left point of the box and the width and height of the box. We also draw the box on the video based on these information.

These coordinate information would be used again.

$$
\begin{aligned}
coordinategetfromYOLO : (&xfromleftupperpoint,\\
&yfromleftupperpoint,\\
&length,\\
&height)\\
Thencoordiantionofbox : &[[x,y],[x+height,y],\\
&[x+height,y+length],x,y+length]\\
Thedistancebetweentwocars :&\\
&width = y_1 - (y_0 + length_0)\\
&height = x_1 + length_1 - x_0
\end{aligned}
\tag{9.3.1}
$$

After the calculation we can get the final detected free parking space in the middle dynamic parking area(shown in fugure34 with green)

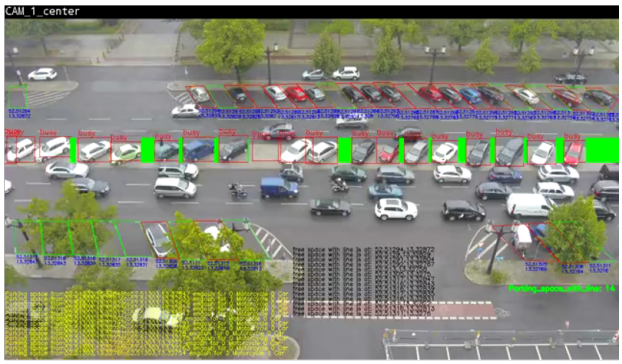


Figure 34: Free Parking Space

9.4 Training of Detection

In this project we do the detection with the help of COCO Dataset. One side is, that the COCO Dataset has already contained 164k images. It can do a great help to the object detection. The other side is, that the computers of all teammates didn't have a great GPU, so that we can not train the model well and also would waste too much time on the training. After we add COCO Dataset into detection, it has a very high accuracy with 99 % correct. We thought it is enough for the normal using of cars' detection and statistics of free parking spaces.

9.5 Summary of Detection with YOLO

Detection with the help of YOLOv3 has a better accuracy than direct detected by segmentation. It works fine with our project and gets a excellent result. It also shows very clean and clear. The speed of detection is enough for the real time detection. In conclusion, we thought it is a better way to detect the parking cars and get the free parking slots information.

10 Output of LOG File

Our final output is a log file, which contains all the information. This log file is divided into three parts. Part1 shows the geo-coordinates of the free parking space and its number. Part2 shows the geo-coordinates of the occupied parking space and its number. Part3 shows the geo-coordinate interval of free area in the middle area and the count of the parking.

num1parking lot is free at52.51294,13.32872
num3parking lot is free at52.5129,13.32829
num5parking lot is free at52.51292,13.3282

⋮

num2parking lot is occupied at52.51294,13.32833
num4parking lot is occupied at52.51291,13.32825
num6parking lot is occupied at52.51289,13.32817

⋮

Parking space from52.51303,13.32848to52.51303,13.32846 enough for 0 Motorcycle 0 Car
Parking space from52.51303,13.32842to52.51303,13.3284 enough for 0 Motorcycle 0 Car
Parking space from52.51303,13.32838to52.51303,13.32829 enough for 5 Motorcycle 2 Car

⋮

11 Conclusion

11.1 Challenge of Project

When we get the project material code at the beginning, there is the following problems:

1. The detection accuracy of the dynamic parking space is insufficient, and there is no way to give the intermediate parking position in real time.

2. In the static parking space part, because of the trees, some vision of parking spaces is blocked (static parking space in the lower right corner), there is no way to use the previous codes and algorithm to detect parking status.

3. no geographic coordinates for each parking space.

4. did not consider the disturbance of the camera.

11.2 Production of Project

To complete the project objectives, we have taken the following methods:

1. Re-dividing the intermediate detection area, reducing the interference detection of intermediate parking spaces due to passed cars and use segmentation as detect method. In this way, we significantly improve the detection accuracy of the intermediate dynamic parking space, which can be obtained for a relatively satisfactory real-time result.

2. In the middle dynamic parking detection area, we also use YOLO detection methods. By determining the division of the test area, we can detect each vehicle in the dynamic parking space area in real time. Further, the parking status of the middle dynamic parking space can be obtained. With Yolo, we can get more accurate test results than Segmentation, but YOLO requires a sufficiently large training set to detect.

3. In solving parking space covered by trees, we narrowed the detection range of the parking space, excluding the interference of the tree on the test results, and finally got a precise test result.

4. we combine each static parking space and dynamic parking test results with geographic coordinates, so that the result of output can be simple to determine the specific location of each parking space. In this way, the user can easily find the specific parking position.

5. We use the K-Means method to determine geographic coordinates for the four vertices of two adjacent lane lines and eliminate the disturbances that the camera may occur by correcting the lane cable correction.

In summary, we can finally through two different methods: Segmentation and Yolo, get accurate real-time parking space information and output all detection results every twenty seconds to update storage. We have completed the requirements of the project and get some suggestions for further optimization.

11.3 Next to do

In some special cases, the parking space will have darker stains or obstructions that may be mistakenly detected as vehicles. This problem can be solved through deep learning retraining in future projects.

Because of the server problem, we did not construct the REST-API in this semester, however, we output all the test results and update the storage every twenty second. The test results we outputted can be used directly to perform REST-API construction in the subsequent project, so that users can get all parking spaces status directly in real time.

References

[1] https://opencv.org/about/.

[2] https://docs.opencv.org/4.5.2/d1/dfb/intro.html.

[3] https://docs.opencv.org/master/d1/dc5/tutorial_background_subtraction.html.

[4] https://ichi.pro/de/trainieren-sie-den-mnist-datensatz-nach-dem-lenet-modell-107204824

[5] https://jason-chen-1992.weebly.com/home/hello-deep-learning-dnn-mnist.

[6] https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html.

[7] https://en.wikipedia.org/wiki/Edge_detection.

[8] https://zhuanlan.zhihu.com/p/78798251.

[9] https://stanford.edu/~cpiech/cs221/handouts/kmeans.html.

[10] How to solve the parking space problem? https://blog.quickride.in/how-to-solve-the-parking-space-problem/.

[11] Tang Bai. You look only once learning. https://zhuanlan.zhihu.com/p/31427164.

[12] Raul E. Sanchez-Yanez Fernando E. Correa-Tome. Integral split-and-merge methodology for real-time image segmentation.), 013007 (2015).

[13] Jing Gu. Detection of dynamic free space for parking lot occupancy with opencv. August, 2020.

[14] Santosh Divvala et al. Joseph Redmon. You only look once: Unified, real-time object detection. *Allen Institute for AIy, Facebook AI Research*, May, 2016.

[15] Joseph Redmon. You look only once. https://pjreddie.com/darknet/yolo/.

[16] Patrick Sisson. Cities have a parking problem. more parking is not the solution. https://citymonitor.ai/transport/cities-have-a-parking-problem-more-parking-is-not-the-solution.

[17] Scott E. Umbaugh. Digital image processing and analysis with matlab and cviptools, third edition (3rd ed.), 2017.

[18] Xiaoting Rui Weibo Wei. Research on image edge detection method.), 2002.24(6):91-94.

[19] Magdalena Yordanova. Detection of dynamic free space for parking lot occupancy with opencv.