

# Public Transport Access Detection

Dragoş-Mihail Comănac

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Context and motivation . . . . .	4
1.2	Objectives . . . . .	4
1.3	Paper structure . . . . .	4
<b>2</b>	<b>State of the art</b>	<b>5</b>
<b>3</b>	<b>Theoretical foundations</b>	<b>6</b>
<b>4</b>	<b>Design and implementation</b>	<b>7</b>
<b>5</b>	<b>Experimental results</b>	<b>8</b>
5.1	Dataset . . . . .	8
5.2	Performance evaluation . . . . .	10
5.3	Related work . . . . .	11
5.3.1	Radio frequency approach . . . . .	12
5.3.2	Computer Vision approach . . . . .	12
5.3.3	You only look once . . . . .	12
5.3.4	Single Shot MultiBox Detector . . . . .	14
<b>6</b>	<b>Conclusions and future work</b>	<b>16</b>
<b>7</b>	<b>Bibliography</b>	<b>17</b>

# Application and original contribution

## Classification

**AMS Mathematical Subject Classification - 68T45 Machine vision and scene understanding**

**ACM Computing Reviews Categories and Subject Descriptors - I.4 IMAGE PROCESSING AND COMPUTER VISION - I.4.6. Segmentation - Edge and feature detection**

## Application lifecycle

1. Application is opened
2. User chooses auditive or visualization mode using a voice command
3. In auditive mode, the live object detection automatically starts, and information is provided to the user in an auditive manner
4. In visualization mode the user can load an image and perform detection on it or can start object detection on the live feed from the camera
5. Application is closed by voice command or through a button

## Functionalities

- choose auditive mode or visualization mode using a voice command
- auditive mode: perform live object detection on cars, busses, and license plates and based on the results of detection, provide auditory clues that guide the user towards the detected car or bus. If the detected object is a bus, then provide information about the location of the doors and the line of the bus (if available). If the detected object is a car, then provide information about the location of the doors and the license plate (if available)

- visualization mode: display the results of object detection on an image or on the live feed from the camera
- close the app using a voice command

## Original contribution

The application aims to help the visually impaired persons by using computer vision and line detection, on a mobile device, to provide spatial information about public transport such as busses or ride sharing access. An object detector model will extract information about the vehicle and a line detector will analyze the part of the image containing the vehicle to extract information about the doors of the vehicle. This should eliminate the need for a dataset that has specific bounding box annotations for the doors.

# 1. Introduction

## 1.1 Context and motivation

## 1.2 Objectives

## 1.3 Paper structure

## 2. State of the art

- discussion on the current methods used in object detection and line detection

### 3. Theoretical foundations

- description of the basic concepts that are used

## 4. Design and implementation

- description of the proposed solution



## 5. Experimental results

### 5.1 Dataset

For training the object detection system we use the Open Images Dataset V4 [15], available at [13]. In total, the dataset contains 9.2 million images, including 14.6 million bounding boxes across 600 classes on 1.74 million images. We will use only a subset of classes: bus, car and license plate.

The tool used to download the Bus, Car and License plate classes and the corresponding bounding boxes is OIDv4 ToolKit [24].

	Number of images	Number of bounding boxes			
		Total	Bus	Car	License plate
Train	30939	80295	11927	60516	7852
Validation	4967	9985	92	9381	512
Test	14894	30660	353	28737	1570

Table 5.1: Statistics related to the subset of Open Images

Each image is resized so that it’s dimension is 416x416 and the bounding boxes are modified accordingly with the following formulas:

$$xmin = 416/width * xmin$$

$$ymin = 416/height * ymin$$

$$xmax = 416/width * xmin$$

$$ymax = 416/height * ymin$$

Where width and height correspond to the image and xmin, ymin, xmax, ymax correspond to the lower left corner and upper right corner of the bounding box. The resizing helps the object detection task, as explained in [21], because this way we can split the image in a grid of 13x13 cells of size 32x32 so that there is a cell in the center that can detect the larger objects that are centered in the middle, rather than have 4 cells in the middle that try to detect the same object.

Each image is associated with multiple bounding boxes and each cell is responsible with detecting multiple bounding boxes through the use of anchors as

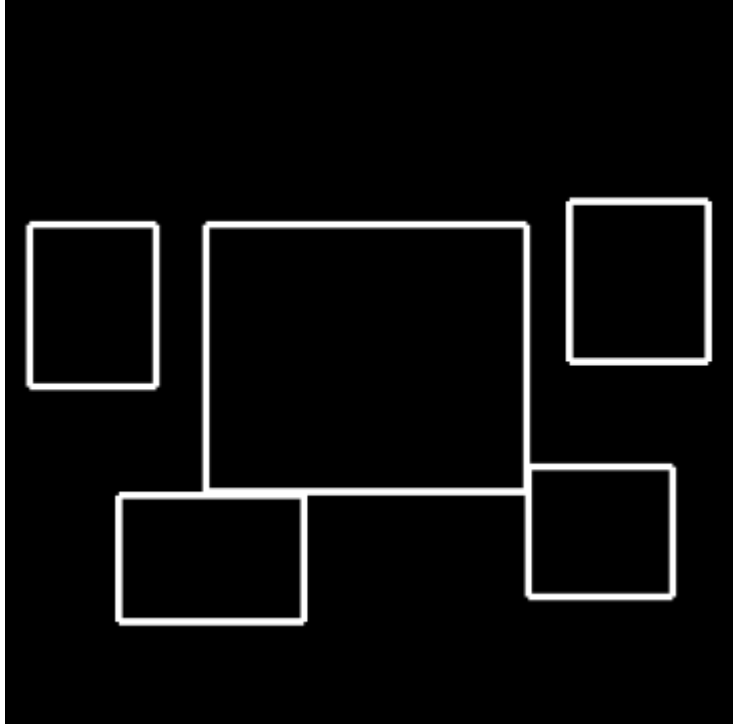
explained in [21]. We use 5 anchors per cell so that each cell can detect various shapes and sizes. The anchor boxes are chosen using K-Means over the whole dataset.

For computing the distance between the centroid and bounding box the following formula is used, as in [21]:

$$distance(centroid, box) = 1 - IOU(centroid, box)$$

Where IOU represents intersection over union and is calculated as the area of the intersection of the 2 boxes over the area of union of the 2 boxes.

Figure 5.1: Anchors for the bus, car and license plate classes



This shows that there are large boxes centered in the middle of the image and multiple smaller ones around.

Each image needs to be associated with a ground truth. We represent the ground truth as a  $C \times C \times B \times 8$  tensor. For each cell in the  $C \times C$  grid we associate  $B$  anchor boxes. Each anchor box has the following parameters:  $b_x$  and  $b_y$  represent the center of the box,  $b_w$  and  $b_h$  represent the width and height,  $c$  is the probability that the anchor predicts an object and  $c_1, c_2, c_3$  represent the class probabilities. In our case  $C$  is 13 and  $B$  is 5.

The following formulas are used to compute the center, width and height from the output of the model:

$$\begin{aligned}b_x &= \sigma(t_x) + c_x \\b_y &= \sigma(t_y) + c_y \\b_w &= p_w \cdot e^{t_w} \\b_h &= p_h \cdot e^{t_h}\end{aligned}$$

Where  $t_x, t_y, t_w, t_h$  represent the raw predictions to which the sigmoid function is applied,  $c_x, c_y$  represents the coordinates of the upper left corner of the cell that predicts the box and  $p_w, p_h$  represent the width and height of the anchor that predicts the box.

After the bounding boxes are interpreted, usually Non-maximum Suppression (NMS) is used to filter the boxes with the best score and overlap with the ground truth and then mean average precision is used to see how good are the predictions.

## 5.2 Performance evaluation

To measure an object detection system performance, usually frames per second (FPS) and mean average precision (mAP) are used. Precision measures the percentage of correct predictions for a given class.

$$Precision = \frac{TP}{TP + FP}$$

A prediction is considered true positive if the IOU is greater than a set threshold. Recall is another metric that measures the percentage of found positives (also known as true positive rate).

$$Recall = \frac{TP}{TP + FN}$$

Both precision and recall depend on the given threshold for true positives. This means that by plotting different values for precision and recall for different thresholds we get a curve. The area under the curve is called average precision. And the mean over the area under the precision-recall curve for all classes is the mean average precision.

Over time, the mean average precision became the standard way of evaluating the performance of an object detection system in order to compare it with other solutions.

During training, the following sum-squared error loss function [20] is optimized to achieve better performance:

$$\begin{aligned}
& \lambda_{coord} \sum_{i=0}^{C^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{coord} \sum_{i=0}^{C^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \sum_{i=0}^{C^2} \sum_{j=0}^B 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
& + \lambda_{noobj} \sum_{i=0}^{C^2} \sum_{j=0}^B 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{C^2} \sum_{j=0}^B 1_{ij}^{obj} \sum_{c \in classes} (p_{ij}(c) - \hat{p}_{ij}(c))^2
\end{aligned}$$

Most grid cells do not contain any boxes. Therefore in order to balance the confidence scores  $\lambda_{coord}$  and  $\lambda_{noobj}$  are added in order to increase the loss from bounding box predictions and decrease the loss from confidence predictions. Also, the square root of the width and height is used to remedy the problem that error in small and large boxes is weighted the same.

$1_{ij}^{obj}$  is 1 for the  $j$ 'th bounding box in the  $i$ 'th cell.

### 5.3 Related work

Since the hardware became sufficiently advanced and accessible, the interest in object detection system increased, thus the need for a standardized dataset for comparing solutions appeared. Pascal VOC is one of the first datasets that proposes a variety of tasks such as object detection. It started as a challenge in 2007 [5] and it continued until 2012 [6]. Since then, mostly the combined datasets from 2007 and 2012 are used for training and validation and the test dataset from 2007 is used for testing and they are used as a common ground to compare performance across all kinds of computer vision tasks, such as object detection.

So far, the general idea in creating object detection datasets is to take an image and add bounding boxes around the objects of interest. But if we want to detect a specific feature of an object, for example if we want to detect the doors of a car, the current approach would be to take a dataset of images and add bounding boxes for the doors. This approach can take a long time and it requires further training of the model. Our approach should simplify this process, because we try to eliminate the need for additional bounding boxes and detect directly the required shapes. For example, if we need to detect bus doors we are looking for long vertical lines in the detected patch, that can be

extracted by a line detector that eliminates the need for extra training. It is also expected that our approach is still real time, even though it adds an extra task and the accuracy shouldn't be affected.

### 5.3.1 Radio frequency approach

The current approach in making busses more accessible is a solution based on Radio Frequency Identification (RFID) [4]. Basically, it uses wireless radio frequency transmissions to transfer data between two devices. The system is composed of four main parts: the device held by the VIP which is compatible with a braille keyboard, the bus station controller, a device on the bus and a database. The VIP searches the bus number and destination using his device. Then the bus controller sends signals to all busses in the radio frequency area, then passes back to the user the information about incoming busses, through a vocal message.

One disadvantage of this method is the complex infrastructure required. Basically, the bus, station and the user are dependent upon each other. This makes it hard to implement it on a large scale and it does not guide the user towards the bus. Also, it requires expensive equipment for the user such as a braille keyboard, which is around 1000 dollars, a RFID reader which is around 100 dollars, and a personal digital assistant, which is around 20 dollars. The hardware needed for stations and busses is cheaper by a lot (around 1 dollar per bus or station), but it wouldn't be feasible because of the integration costs.

### 5.3.2 Computer Vision approach

Another approach would be to make the user independent of any infrastructure. This should give the user the freedom to access any bus, in any city, regardless if there are any accessibility features available or not. This can be achieved by using a mobile version of a computer vision model that tells the user where the bus is located by using real time object detection. This way there is no need for extra equipment.

We will focus especially on the single shot detector class of object detection models. Because their detection pipeline consists only of one neural network, they achieve real time speed with decent accuracy compared to other slower solutions.

### 5.3.3 You only look once

You only look once (YOLO) [20] is an object detection system that achieves real time performance. Before it, the task of object detection was decomposed in multiple tasks, that together make a pipeline, which is hard to train. For example, R-CNN [7] and its faster variants first generate bounding boxes through selective search, then a convolutional network extracts features that are classified by a support vector machine. All of these steps slow down performance.

Instead of a long and complex detection pipeline, in YOLO object detection is treated as a regression problem. There is only one neural network that predicts the bounding boxes and class probabilities from a image. This way, the network can be optimized end to end, and the inference time is greatly reduced, thus achieving real time performance.

The image is split into an  $S \times S$  grid, each cell being responsible with the detection of one object. For each cell predicts  $B$  bounding boxes and for each box the position of the center, relative to the grid is predicted, the width and height are predicted relative to the image and a confidence score is predicted indicating if there is an object in that cell. Also,  $C$  class probabilities, conditioned on the cell containing an object, are predicted for each cell, regardless of the number of boxes  $B$ . Therefore, the network outputs a tensor of shape  $S \times S \times (B \times 5 + C)$ . In the second version of YOLO [21], the notion of anchors is introduced. This changes the output to be  $S \times S \times B \times (5 + C)$ , where  $B$  is the number of anchors. This allows the model to detect multiple objects in the same grid cell, with multiple shapes and sizes.

The network proposed by the authors is inspired by the GoogleLeNet model. It has 24 convolutional layers followed by 2 fully connected layers, although a faster version contains only 9 convolutional layers and fewer filters.

Leaky rectified linear activation function is on all layers, except the last one, where a linear activation function is used. Sum squared error is used for the loss function, but the authors note that it is not ideal because localization errors and classification errors are treated the same. Also, the cells that do not contain any images will have confidence scores close to zero, and can overcome the gradients from the cells that do contain objects, causing the training to diverge. To prevent this, two parameters that control the loss from bounding box coordinate predictions and the loss from confidence predictions are introduced.  $\lambda_{coord}$  increases the coordinate prediction loss and  $\lambda_{nobj}$  decreases confidence predictions loss.

For each cell, only one bounding box is chosen, the one with the highest input over union (IOU) with the ground truth. This causes the bounding box predictors to perform better on certain sizes, aspect ratios or classes, improving recall.

At test time, YOLO achieves great speeds because it only need a single pass through the network. Non-maximal suppression is used to choose between different bounding boxes.

Because each cell predicts only one object, multiple clustered small objects are harder to detect, thus the main source of error is localization error.

One important metric in measuring the performance of object detection systems is mean average precision (mAP). Precision measures the percentage of correct predictions for a given class. A prediction is considered true positive if the IOU is greater than a set threshold. Recall is another metric that measures the percentage of found positives. Both precision and recall depend on the given threshold for true positives. This means that by plotting different values for precision and recall for different thresholds we get a curve. The area under the curve is called average precision. And the mean over the area under the

precision-recall curve for all classes is the mean average precision.

In terms of performance on the Pascal VOC 2007 dataset, first version of YOLO achieves 63.4% mAP and 45 frames per second (FPS), and the faster version has 52.7% mAP and 155 FPS. For comparison Faster R-CNN achieves 73.2% mAP but 7 FPS, which is accurate but very slow, and 100Hz Deformable parts model achieves 100 FPS but it has 16 mAP%. Therefore YOLO strikes a good balance between speed and accuracy.

### 5.3.4 Single Shot MultiBox Detector

Single shot MultiBox Detector (SSD) [17] is another example of object detection system that achieves real time performance, encapsulating all operations in a single deep neural network. Similar to YOLO, this helps SSD to outperform previous approaches that use multiple stages in detection such as R-CNN.

The network is composed of two parts. The first one consists of what is called the base network, which is a truncated version of a image classifier, where the classification layers are removed, that is used to extract features. On top of the base network several structures specific to object detection are added.

The key features of SSD are the multi-scale feature maps, convolutional predictors and default boxes.

At the end of the base network there are added convolutional feature layers that progressively decrease in size the feature map from the base network. This way predictions are made for each newly added layer, therefore, the predictions are made at various scales.

Each cell in each feature map has associated  $K$  default bounding boxes, whose positions are relative to their cell. Then for each box several kernels of size  $3 \times 3 \times P$ , where  $M \times N$  is the size of the feature map and  $P$  is the number of channels, are used to predict  $C$  class probabilities and 4 offsets to the respective box. This means that each box uses  $(C + 4) \cdot K$  filters, therefore the size of the predictions is  $(C + 4) \cdot K \cdot M \cdot N$ .

During training, the outputs need to be assigned to their corresponding ground truths. Then the loss function and back propagation are applied end-to-end. Furthermore, the set of default boxes and scales is chosen and hard negative mining and augmentation strategies are used.

For matching the outputs, each ground truth box is associated with the default box with the highest jaccard overlap. The novel approach here is that the default boxes are also matched with any ground truth box with jaccard overlap higher than a threshold. This allows predictions with high scores for multiple overlapping default boxes.

The loss is a weighted sum between confidence loss and localization loss.

Another crucial part is choosing scales and aspect ratios for the default boxes. Each feature map has a specific scale, distributed evenly between 2 values. The aspect ratios are chosen from a predefined set. The width and height are computed using the scale and the aspect ratio. The center is chosen based on the feature map cell size.

The matching steps produces more negatives than positives. This introduces an imbalance, and to fix this, the negatives are filtered by their confidence loss, so that a ratio of 3:1 is kept between the negatives and positives.

Also, data augmentation is used. During training each for each images either a patch is randomly sampled, a path is sampled so that the minimum jaccard overlap with the objects is higher than a threshold, or the original image is used. After this, the image is resized and flipped with a probability of 0.5 and some photo-metric distortions are applied.

Regarding performance on the Pascal VOC 2007 dataset, SSD achieves 74.3% mAP and 59 FPS, surpassing the first version of YOLO in terms of accuracy and speed balance.



## 6. Conclusions and future work

- summary of the solution - critical analysis of the solution - future improvements

## 7. Bibliography

- [1] Cuneys Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32:1633–1642, 10 2011.
- [2] Md. Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. The history began from AlexNet: A comprehensive survey on deep learning approaches. *CoRR*, abs/1803.01164, 2018.
- [3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.
- [4] Lamya El Alamy, Sara Lhaddad, Soukaina Maalal, Yasmine Taybi, and Yassine Salih-Alj. Bus identification system for visually impaired person. In *2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 13–17, 2012.
- [5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [7] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [8] Lilit Hakobyan, Jo Lumsden, Dympna O’Sullivan, and Hannah Bartlett. Mobile assistive technologies for the visually impaired. *Survey of Ophthalmology*, 58(6):513–528, 2013.

- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [11] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. *Computer Vision ECCV 2012*, pages 340–353, 2012.
- [12] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.
- [13] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Mallocci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from <https://storage.googleapis.com/openimages/web/index.html>*, 2017.
- [14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [15] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, and et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, Mar 2020.
- [16] Dongjin Lee, Hosub Yoon, Chankyu Park, Jaehong Kim, and Cheong Hee Park. Automatic number recognition for bus route information aid for the visually-impaired. In *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 280–284, 2013.
- [17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.
- [18] Wai-Ying Low, Mengqiu Cao, Jonas De Vos, and Robin Hickman. The journey experience of visually impaired people on public transport in london. *Transport Policy*, 97:137–148, 2020.
- [19] Geneva: World Health Organization. World report on vision. page 77, 2019.

- [20] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
- [21] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.
- [22] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.
- [23] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
- [24] Angelo Vittorio. Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset. [https://github.com/EscVM/OIDv4\\_ToolKit](https://github.com/EscVM/OIDv4_ToolKit), 2018.
- [25] Na Wang, Ruoyan Chen, and Kang Xu. A real-time object detection solution and its application in transportation. In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 486–491, 2021.
- [26] Junjie Yan, Zhen Lei, Longyin Wen, and Stan Z. Li. The fastest deformable part model for object detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2504, 2014.

# GitHub

The link to GitHub is the following: <https://github.com/ComanacDragos/PublicTransportDetector>.

And the link to the GitHub history is this <https://github.com/ComanacDragos/PublicTransportDetector/commits/main>.