Pedestrians, Signs, Crosswalks Detection System

Alessandro Castellucci, Matteo Corradini, Alessandro Lugari

{228058,204329,243111}@studenti.unimore.it

University of Modena and Reggio Emilia

1 Introduction

Nowadays car driving safety is a hot topic. In an increasingly smart and interconnected world, where autonomous driving can be surely considered the future (and in some contexts already the present), people's safety remains a fundamental and challenging topic. For these reasons, the goal of this project is to detect pedestrians, signs, and crosswalks to help autonomous cars and car drivers in avoiding collisions and secure the pedestrians outside and inside the vehicle.

In order to fulfill project constraints and requirements, we choose to implement the retrieval system for signs detection, a deep learning approach using a convolutional neural network for detecting pedestrians, and a geometry-based approach for the crosswalks detection. Section 2 summarizes how pedestrian detection was implemented. Section 3 describes how the signs are detected with retrieval techniques. Section 4 deals with crosswalks detection using a geometry-based approach. Finally, Section 5 contains our conclusions and final thoughts about this experience.

The implementation of the project can be found in the following GitHub repository.

2 Pedestrians Detection

In this section, we present briefly the dataset used for the pedestrian detection task, followed by a description of the model used for achieving this goal.

2.1 Datasets

The datasets that we have used were COCO and PennFudan, respectively for training/testing purposes (the first) and for checking the generalization properties (the second one).
During the fine-tuning process, we picked randomly 200 images for training and 50 each for validation and testing, using COCO images.
Only after the process of training, we have tested the model over a small subset of PennFudan's images.

2.1.1 COCO

The version of the dataset is COCO 2017, which presents various classes, 91 in total, but we trained the model over only two classes, person and background, by filtering the images properly, obtaining from this dataset a total of 2693 images containing person boxes.

2.1.2 PennFudan

This is a small dataset (about 100 images), specialized in pedestrian images, that we used only for testing purposes, in particular, we used those images to check if the model, fine-tuned on COCO, was performing well with images of a completely different dataset.

2.2 Model

The detection of pedestrians is a task that requires both classification and regression. We managed through the usage of a Faster R CNN model.

The backbones used were *Resnet50* and *MobilenetV3*, both provided by PyTorch. These two backbones have pros and cons, for example, Resnet50 is more accurate but slower, instead, MobilenetV3 is faster (also the training was faster) but the performances were slightly lower.
In the end, we found out that the model with these two backbones showed similar performances.
During training we froze the last 5 layers of the model to fine-tune it with images of COCO, then the output features are passed to the last component, the head. This component is the one that manages both the classifier and box regressor. It is used an MLP for both tasks (classification and regression). The configuration used the default one, with no hidden layers. Later we tried to change the configuration of the head by substituting the default networks with more complex ones, but since we didn't see any major improvement, we decided to stick to the basic head configuration, which is also a little bit faster during inference (less calculation needed).
For which concern the optimizators, we tested SGD and Adam, both with the same learning rate values. Adam converged quickly with respect to SGD, but SGD could reach a slightly better value. In general, Adam is better for prototyping because with fewer epochs we can reach good performances.
The learning rate values chosen for training are 0.005 and 0.001, our choice is driven by the fact that values outside that range could not lead the model to converge.

2.2.1 Performance

Finally, some considerations on performances.
First, Python COCO APIs were used for evaluating metrics. We chose Average Precision in two cases. The first when IoU is greater than 0.5 and then the averages of APs from 0.5 to 0.95. For recall, instead, it was provided only the Average Recall with IoU going from 0.5 to 0.95.
For this kind of application, recall is more important than precision, since we want to have fewer possible false negatives (in this case a false negative is when the model predicts no person within a certain box when there is a person). Another important fact that needs to be taken into consideration when evaluating the various models, is the confidence threshold. We chose to set it to 0.8 to be very sure about each bounding box, with a lower value performances can improve slightly but we thought wasn't worth it.
Precision and recall are evaluated also on box area (small-medium-large), then performed average, so what happens is that the overall value is lower due to small boxes that are harder to detect and so worsen the performances. This can be seen in looking at the two performance tables. The values show how the model, trained on COCO, performs better on PennFudan.
The reason is that pedestrians in PennFudan images are usually not too far (this is equal to saying that the boxes are larger) and so the model can detect them more easily.
Instead, on COCO, we found more general images of 'person', in different contexts, that could be near a road or inside a house. This leads to more difficult detections due to the nature of images (e.g. a person may be partially hidden behind another object or its bounding box is very small) and so the overall performances are lower, especially when detecting small bounding boxes.

3 Signs Retrieval

The module aims to detect road signs in the picture looking for a possible pedestrian crosswalk sign. In order to accomplish the project constraints and to exploit different computer vision approaches, we choose to apply an image retrieval neural network to identify the road signs eventually present on the street. We choose to implement both a Siamese and Triplet neural network for the completeness of the project's results.

Furthermore, we choose to train those neural networks on different datasets in order to achieve better results and study their behavior on different inputs and classes. We also trained a custom neural network to classify the road sign.

The retrieval pipeline follows these steps:

1. For all the datasets, it selects all the road signs in the image based on their bounding boxes and it creates new datasets composed by these images.
2. It creates the embeddings given the specific weights for each datasets.

Model	Optimizer	Learning Rate	AP @0.5	AP @[0.5:0.95]	AR @[0.5:0.95]
mobilenetv3	SGD	0.005	0.56	0.27	0.35
mobilenetv3	SGD	0.001	0.61	0.30	0.41
mobilenetv3	Adam	0.005	0.59	0.25	0.34
mobilenetv3	Adam	0.001	0.60	0.29	0.40
resnet50	SGD	0.005	0.64	0.31	0.42
resnet50	SGD	0.001	0.76	0.43	0.54
resnet50	Adam	0.005	0.73	0.41	0.54
resnet50	Adam	0.001	0.78	0.46	0.55

Table 1: Performances on COCO after 10 epochs of training

Model	Optimizer	Learning Rate	AP @0.5	AP @[0.5:0.95]	AR @[0.5:0.95]
mobilenetv3	SGD	0.005	0.96	0.71	0.79
mobilenetv3	SGD	0.001	0.95	0.70	0.77
mobilenetv3	Adam	0.005	0.95	0.70	0.78
mobilenetv3	Adam	0.001	0.97	0.72	0.78
resnet50	SGD	0.005	0.92	0.63	0.74
resnet50	SGD	0.001	0.93	0.68	0.78
resnet50	Adam	0.005	0.95	0.67	0.76
resnet50	Adam	0.001	0.96	0.72	0.80

Table 2: Performances on PennFudan after 10 epochs of training

3. It then returns the best rank given a query image representing a road sign.

3.1 Dataset

As said in section the previous paragraph, in order to have larger and various cases of study and improve the performance of the neural networks, we use more than one dataset for training and evaluation. We start evaluating the most famous and popular datasets and then move to some others less known.

3.1.1 German Traffic Sign Recognition Benchmark (GTSRB)

The first road sign dataset very popular that has been used is called German Traffic Sign Recognition Benchmark (GTSRB)[1]. This dataset is composed of 43 classes, 12630 images for the evaluation set and 39209 for the training set.

Even if very popular and well known in the road sign classification field, unfortunately, this dataset does not contain the informative pedestrian crosswalk sign.

3.1.2 Mapillary Traffic Sign Dataset

The second dataset used in our project is called Mapillary Traffic Sign Dataset[2]. This dataset is well known and popular for the road sign classification problem, due to the big number of images and classes. The dataset contains nearly 52000 fully annotated and high-resolution images, with more than 300 road signs from all over the world.

Due to the constraints of the available capacity, we choose a subset of 5000 images and 42 classes to train the neural networks. In particular, between all the various classes representing the pedestrian crosswalk, we choose the class *information–pedestrians-crossing–g1* to classify the informative road sign.

3.1.3 Road Sign Detection Dataset

The third dataset used is simply called Road Sign Detection Dataset[3]. This dataset presents very different characteristics compared with the others. The dataset is quite small, and it contains only 4 classes and 877 images with the relatives bounding box.

We choose to train and evaluate the performance of this dataset in order to compare its results with one of the bigger sizes. Due to the small number of images, it was possible to train the neural networks for numerous epochs, differently from what we did for other datasets.

3.2 Neural network

In order to experiment with different strategies of image retrieval, we choose to implement custom neural networks called *RoadSignNet*.

The core of the neural networks is made by sequential blocks of the following operations:

1. Convolution with zero padding and kernel size 3;
2. ReLu activation function;
3. Batch normalization to speed up the training[4];
4. Convolution with zero padding and kernel size 3;
5. ReLu activation function;
6. Batch Normalization;
7. Max pooling to down size the image.

Finally, a fully connected layer ends the neural networks. This custom neural network has been also used for classification tasks.

We tried to use the pre-trained neural network ResNet 18 given by PyTorch as the backbone of the Siamese and Triplet networks.[5]

3.2.1 Retrieval

In order to satisfy the retrieval tasks, we created 2 different neural networks: *SiameseNet* and *TripletNet*. As it is possible to guess, the networks implement each the siamese[6] and the triplet[7] network.

3.2.2 Siamese

In the Siamese network, during the training is given the index of the current element. Then is randomly choosen whenever the other element should belong to the same class or a different one. Then, a random index is selected from the correct class. For the testing phase, an equal number of positive and negative couples are created during the initialization of the dataset.

The loss used to train the network is the *Contrastive loss*[8]:

$$loss(d, Y) = \frac{1}{2} * Y * d^2 + (- Y) * \frac{1}{2} * max(0, m - d)^2 \quad (1)$$

Where:

- d is the distance between the outputs and the encoder
- Y is the label (1 if the elements belong to the same class, 0 otherwise)
- m is the margin parameter

3.2.3 Triplet

In the Triplet network, during the training is given the index of the current element (the anchor). Then are randomly chosen two different elements: one of the same class (the positive) and another one of a different class (the negative). For the testing phase, random triplets are created following the same rule.

The loss used to train the network is the *TripletMarginWithDistanceLoss*[9], with the *PairwiseDistance* function:

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} \quad (2)$$

3.3 Detection

We use an edge detection-based approach in order to find the squared road sign in the pictures. The steps are the following:

1. Convert the picture in grayscale;
2. Apply a threshold to convert the picture into black and white binary color;

3. Apply an opening morphology (erosion then dilation) and a median blur filter to delete the noise;
4. Apply an algorithm contour detection to find the shapes[10];
5. Apply the Douglas-Peucker algorithm to approximate the shapes;
6. Take only the square shapes located in the upper part of the image.

3.4 Retrieval Performance

The training has been done with the following properties:

- SDG as optimizer, with a learning rate of $1x10^{-3}$ and a momentum factor of 0.9;
- A variable number of epochs (from 5 to 30);
- A variable number of the convolutional blocks (from 3 to 5) described in Paragraph 3.2;
- 3 fully connected layers.

The performances of the networks are evaluated using the MAP score (Mean Average Precision), with different scopes. We computed the average precision using the pedestrian crosswalk road sign pictures in the dataset. Then, we compared them with the other images in the dataset, considering as relevant the images belonging to the same class.

The results are shown in the Table 4.

3.5 Classification Performance

The training has been done with the following properties:

- SDG as optimizer, with the configuration of the Section 3.4;
- *CrossEntropyLoss* as loss function;
- A variable number of epochs;
- 3 layers of the convolutional blocks described in Paragraph 3.2, and 3 fully connected layers.

The results are shown in the Table 4.

4 Crosswalks Detection

This section describes how crosswalks detection is performed. In particular, we deliberately adopted non-DL-based techniques, and to satisfy the project requirements there were used geometric-based approaches. Briefly, the approach is the following:

1. Images pre-processing: white balancing, grayscale conversion, bilateral filtering, Canny edge detection, bird's eye view computation, erosion, and thinning.
2. Hough transform for finding horizontal and vertical lines.
3. Intersection points to search.
4. Evaluation of all possible combinations of four points to find quadrangles that can be proper stripes belonging to a crosswalk.
5. If a crosswalk is detected, an approximate distance between car-crosswalk is computed.

To accomplish the task OpenCV [11] and NumPy [12] libraries were used. All the hyper-parameters, thresholds, and constraints are been empirically found.

4.1 Dataset

The dataset used is the DR(eye)VE dataset [13], a dataset composed by 5 minutes video sequences recorded in Modena (Italy), primarily thought for attention-based tasks. For the detection of crosswalks, we use a subset of the video sequences. In particular have been chosen four video sequences (05, 06, 26, and 35) from which we extract 600 1920x1080 frames per video. From the 600 frames per sub-dataset, we selected 480 frames per sub-dataset, discarding very similar frames, e.g. in situations where the car is stopped at traffic lights. Afterward, the four sub-datasets were annotated by hand, indicating in which frame there's a crosswalk and in which there's not with a boolean value (0 or 1). The frames come from a car roof-mounted camera. The 05 sub-dataset shows a countryside cloudy morning situation, the 06 a downtown sunny morning, the 22 a countryside rainy morning, and the 35 a downtown cloudy morning.

Dataset	Network	Epochs	Layers Conv	Pretrained	Scope	MAP
German	Siamese	10	3	No	10	0.28
German	Siamese	10	5	No	10	0.65
German	Triplet	10	3	No	10	0.27
German	Triplet	10	5	No	10	0.15
Unknown	Siamese	10	3	No	10	0.08
Unknown	Siamese	20	3	No	10	0.30
Unknown	Siamese	30	3	No	10	0.27
Unknown	Triplet	10	3	No	10	0.91
Mapillary	Siamese	10	3	No	10	0.49
Mapillary	Siamese	10	5	No	10	0.10
Mapillary	Triplet	5	3	No	10	0.25
Mapillary	Triplet	5	5	No	10	0.31
Mapillary	Triplet	10	3	No	10	0.72
Mapillary	Siamese	10	3	Yes	10	0.12
Mapillary	Triplet	7	3	Yes	10	0.15

Table 3: Retrieval performance

Dataset	Epochs	Precision	Recall	Accuracy	f-score
German	30	0.50	0.73	99.68%	0.59
Unknown	30	0.95	0.90	98.30%	0.92
Mapillary	10	0.87	0.90	99.15%	0.89
Mapillary	20	0.97	0.79	98.90%	0.87
Mapillary	30	0.94	0.88	99.27%	0.90

Table 4: Classification performance

4.2 Images Pre-processing

In the beginning, images are pre-processed for ease subsequently the detection task. Firstly the color image is white-balanced to adjust colors, important when dealing with crosswalks, which are basically white in Italy. The white balancing is performed with the Gray-World assumption, i.e. the average intensity value of an image is assumed gray. After it's computed the grayscale conversion.

At this point, with the intention of reducing noise but at the same time preserving the sharpness of edges, a bilateral filter [14] is used. A bilateral filter takes into account two Gaussian filters, one as a function of space for blurring pixels, and one as a function of pixel difference for ensuring that only those pixels with similar intensities to the central pixel are considered for blurring. For these reasons we wanted to ensure blurred stripes but with preserved edges. As mentioned above the hyper-parameters were empirically chosen, and so it has been also for the next step, which is the edge detection, performed with Canny [15].

Once the edges were found, a bird's eye view is computed with a proper homography matrix for each sub-dataset. The matrix differs in each sub-dataset because of slightly different views, surely because videos were recorded on different days, and sometimes with a different car. Also for this, and also because we hadn't further information about intrinsic camera parameters, the mounting height, and the camera mounting angles, the transformations for the bird's eye views have been estimated by selecting by hand four source and four destination pixel coordinates for each sub-dataset. For these reasons, the bird's eye views aren't perfect and suffer a bit from distortions. This will affect also the distance computations performed at the end of the detection tasks. Surely can be in general improved. After this process, an iteration of the erosion morphological operator and the execution of the Zhang-Suen thinning

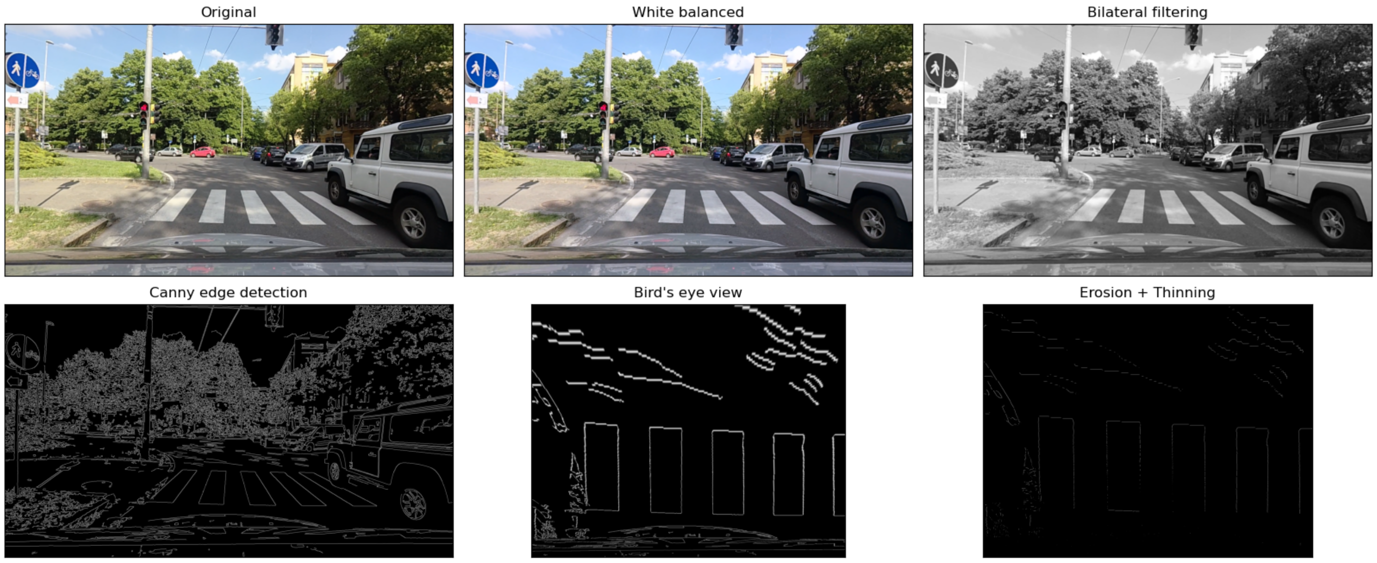

Figure 1: Pre-processing pipeline for crosswalks detection.

algorithm [16] are performed; these two steps are performed to try to ensure thin shapes as input for the detection stage.

An example of the pre-processing pipeline is shown in Fig. 1.

4.3 Detection

After the pre-processing pipeline, starts the detection stage. The first step consists of executing the Hough transform [17] for finding lines in the image. At this point, there's a process to get rid of useless lines. Firstly are discarded non-horizontal and non-vertical lines. The Hough transform returns many lines for the same line in the image because lines in the edges images are rarely really straight, but they are a composition of straight segments. To overcome this drawback, a "clusterization" process is performed to try to reduce the number of lines. Afterward intersection points between horizontal and vertical lines are found. Fig. 2 shows these steps.

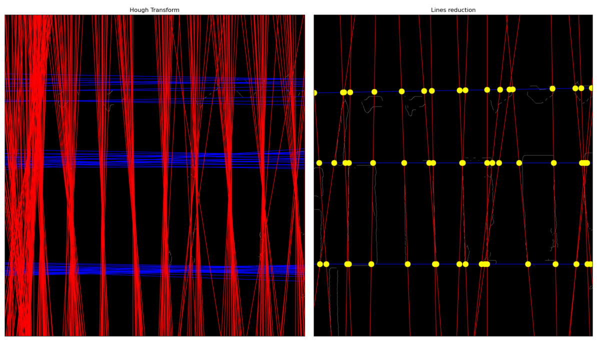

Figure 2: Hough transform, lines reduction and intersection points.

After intersection points have been found, we must find possible stripes belonging to a crosswalk. We check all possible 4-combinations without repetition of the intersection points to see if each quartet of points can be a suitable quadrangle representing a stripe. To filter out some combinations, if a quartet has at least three collinear points, the quartet is discarded.

To assess that a quadrangle is a stripe, some constraints must be achieved. First of all the height must be greater than the width, in addition, have been found empirically that the aspect ratio ($width/height$) and the area should be in a certain range of feasible values. But what makes the crosswalks recognizable is the pattern of white stripes interspersed with darker zones. For this reason, in each quadrangle the mean intensity value from the grayscale image is checked. If the mean value inside the quadrangle is quite lighter than the mean intensity value of the outside quadrangle frame, that quadrangle is considered a stripe. If at least two stripes are found, the system will assess that a crosswalk is detected and a bounding box is computed, firstly in the bird's eye view image and secondly in the original image with the inverse homography matrix.

In addition to detection, if a crosswalk is detected, also an approximate distance from the car to the crosswalk is performed. The quality of the distance measure depends on the goodness of the bird's eye view computation and the goodness of the detection, considering also the absence of information about camera parameters. The approximate distance is computed in the bird's eye view image in the following way:

$$d_{cm} = d_{pixels} * \frac{width_{cm}^{stripe}}{width_{pixels}^{stripe}}.$$

The reference measure to ensure the computation of the distance is the width of the stripes: in Italy is a constant, 50cm [18].

4.4 Results

Now some considerations about results. They are strongly dependent on different aspects: the goodness of the bird's eye view computation, and the ability to intercept the right lines for the Hough transform, but what is really important for the success of the detection task are the crosswalks conditions in the original images. Often the quality of crosswalks is very bad, in the sense that stripes aren't perfect, for example partially or completely erased. The quality is also affected by the weather conditions and sometimes by the presence of obstacles in the image.

Having made these necessary premises and considering that perfection was not the aim of the project, the measures used for evaluating the performances are Precision, Recall (also known as True Positive Rate), True Negative Rate, Accuracy, F-Measure, and G-Mean. The results are summarized in Table 5. Accuracy is not a useful measure in this context, because the subdatasets are strongly unbalanced and so the Negatives are much more than the Positive examples. What really describe the results are Precision, Recall (or True Positive Rate), and True Negative Rate, which together lead to more informative measures like the F-Measure (the harmonic mean between Precision and Recall) and the G-Mean (the geometric mean between True Positive Rate and True Negative Rate), very useful for unbalanced datasets:

$$TPR = \frac{TP}{TP + FN}$$
$$TNR = \frac{TN}{FP + TN}$$
$$G\text{-}Mean = \sqrt{TPR * TNR}$$

In Fig. 3 are shown one True Positive, one False Positive and one False Negative representative examples. Further details are explained in Section 5.

5 Conclusions

At the end of the project, we are able to express some consideration regarding the results of the entire computer vision pipeline.

Concerning the pedestrian module, we are satisfied with the results, but we can definitely improve the model, especially for detecting pedestrians far away from the camera. Another important aspect is related to model inference speed. A YOLO model could be implemented for video processing, due to its own architecture which is faster than what we used so far.

Regarding the crosswalk sign retrieval, we can say that the best performances are obtained from the Mapillary dataset using the Triplet network. This network performs better also on the Unknown dataset. On the other hand, the best approach to retrieve the best match between the detected road sign is still a classification CNN.

With regard to the crosswalks detection task, results are strictly dependent on the quality of stripes and on the quality of the homography matrices, which are computed by selecting source and destination points manually. For this reason, the task can't be really generalizable to other datasets until the bird's eye view computation is properly automated. Naturally this influences also the distance computation, that should be calculated knowing the measure in pixels of focal lenght

Sub-dataset	Precision	Recall|TPR	TNR	G-Mean	Accuracy	F-Measure
05	1.00	0.67	1.00	0.82	1.00	0.80
06	0.40	0.53	0.77	0.64	0.71	0.46
26	0.57	0.50	0.97	0.70	0.94	0.53
35	0.88	0.41	0.99	0.64	0.90	0.56
Average	0.71	0.53	0.93	0.70	0.89	0.59

Table 5: Crosswalks classification results.

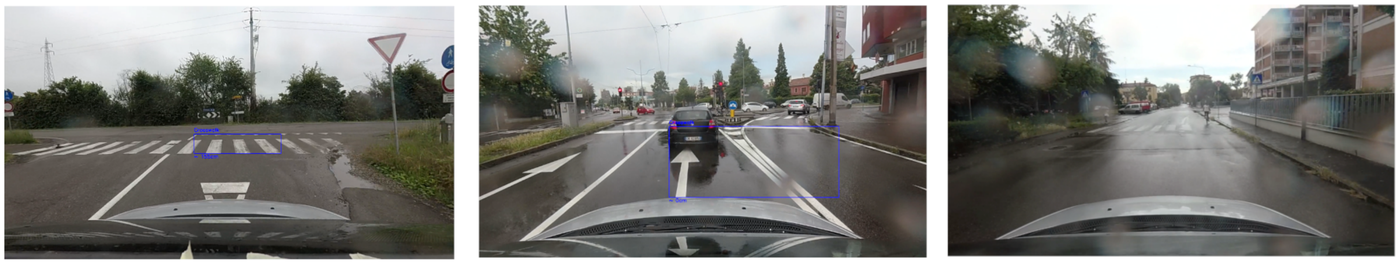

Figure 3: From left to right: TP, FP, FN.

and the camera mounting height. In addition, the task is accomplished voluntarily avoiding more modern approaches (i.e. Neural Networks) so that all project requirements could be satisfied. Best results are obtained in sub-datasets 05 and 26. The last row of Table 5 shows the average measures along the four sub-datasets. What mostly emerges is that the system suffers from False Negatives.

Finally, we can suggest that the union of these three components, with some regard also to real-time programming precautions, can accomplish different use cases. For example, whenever there's at least a pedestrian next to a crosswalk, an autonomous car can take into consideration decelerating or stopping. Otherwise, it can simply decelerate whenever a road sign and a crosswalk are present in the view.

In general, these situations may also fall within the fields of saliency, attention-based tasks, and tracking analyzes (for pedestrians). Further improvements can be taken in this direction.

References

[1] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012.

[2] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, and Y. Kuang, "Traffic sign detection and classification around the world," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[3] "Road signs dataset."

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] S. Chopra, R. Hadsell, and Y. Lecun, "Learning a similarity metric discriminatively, with application to face verification," vol. 1, pp. 539– 546 vol. 1, 07 2005.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[8] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, 2006.

[9] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *BMVC*, 2016.

[10] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[11] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[12] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[13] A. Palazzi, D. Abati, S. Calderara, F. Solera, and R. Cucchiara, "Predicting the driver's focus of attention: the dr(eye)ve project," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[14] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 839–846, 1998.

[15] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, p. 679–698, jun 1986.

[16] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns.," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[17] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, p. 11–15, jan 1972.

[18] "Regolamento art. 40 - codice della strada italiano."