Estimation of Safety Distance Between Vehicles on Highways Using YOLOv4 from Aerial Images

Vitor Yeso Fidelis Freitas
Department of Computer Engineering and Automation
UFRN, Natal-RN, Brazil

Richardson Santiago Teles de Menezes
Department of Computer Engineering and Automation
UFRN, Natal-RN, Brazil

Francisco Vidal
School of Science and Technology
UFRN, Natal-RN, Brazil

Helton Maia
School of Science and Technology
UFRN, Natal-RN, Brazil
helton.maia@ufrn.br

***Abstract*—Traffic accidents are among the most worrying problems in modern life, often caused by human operational errors such as inattention, distraction, and misbehavior. Vehicle speed detection and safety distance measurement can help reduce these accidents. In this study, the computational development conducted was based on Convolutional Neural Networks (CNNs) and the You Only Look Once (YOLO) algorithm to detect vehicles from aerial images and calculate the safe distance and the vehicle's speed on Brazilian highways. The investigation was conducted to model the YOLO algorithm for detecting vehicles in different network architecture configurations. The best results were obtained with the YOLO Full-608, reaching a mean Average Precision (mAP) of 97.44%. Additional computer vision approaches have been developed to calculate the speed of the moving vehicle and the safe distance between them. Therefore, the developed system allows that, based on detecting the safe distance between moving vehicles on the highways, accidents are predicted and possibly avoided.**

I. Introduction

Even with the development of modern cars with self-driving capabilities [1], [2], traffic accidents remain a severe problem for humanity. In recent years, security cameras, sensors on highways, technology embedded in automobiles, and even ethical discussions have been collaborating to reduce accidents [3]. However, the World Health Organization (WHO) informs that about 1.35 million people annually have their lives disrupted due to traffic accidents [4].

The use of Unmanned Aerial Vehicles (UAVs) has been used in several areas, such as in agriculture for mapping and identifying problems [5], in civil construction [6], allowing quick and efficient access to carry out inspections and monitoring of works, delivery services and logistics are being developed [7], in addition to vehicle traffic control [8], [9].

The development of algorithms based on Deep Learning, especially those based on Convolutional Neural Networks (CNNs), have made it possible to achieve the state-of-the-art in detecting and tracking objects with a high degree of accuracy [9], [10]. In this way, a series of applications that combine UAVs and their high-definition cameras under highways have been widely developed [11].

Based on this scenario, this work aims to develop a system capable of tracking and estimating the safe distance between vehicles on highways from aerial images using the You Only Look Once (YOLOv4) algorithm. Six models based on YOLO [12] were trained and tested to perform vehicle detection, four of which were selected from the smaller version of YOLO, called Tiny YOLO, and two of them from the full version, called YOLO Full. The model called YOLO Full-608 achieved the best mAP. However, other configurations obtained an accuracy close to the best result with ten times less processing time, such as the 3-scale Tiny YOLO-416.

Finally, a tool was developed to calculate the safe distance between moving vehicles from aerial images. For the system user, it is possible to initially select a region of interest (ROI) in which the system will perform the analyses. In this way, it is possible to send security alerts about potential accidents in real-time.

This paper is structured as follows. In Section II, we describe the research method characterization. In Section III, we present the results. Finally, in Section IV, we present the conclusions.

II. Methods

The dataset for this work was created with aerial images from Brazilian highways captured by a DJI Spark drone. The computational development was entirely performed using the C/C++ programming languages and the YOLOv4 on Google Colab instances which uses a GPU NVIDIA Tesla T4, CPU Intel Xeon 2.20 GHz, 12GB of RAM, OS Ubuntu 18.04 LTS, and CUDA 11.2.

The workflow and all development steps are shown in the UML-based diagram in Figure 1.

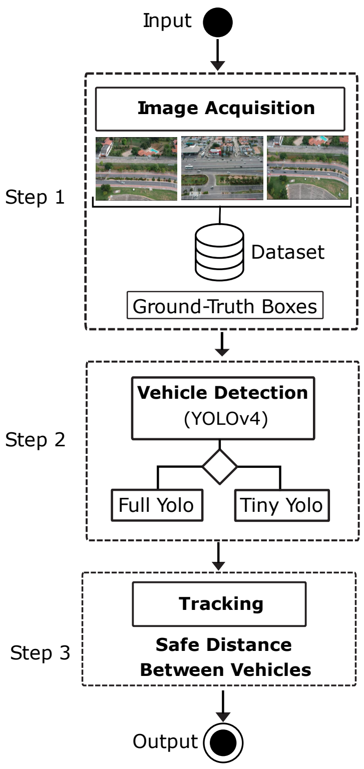

Fig. 1. UML activity diagram of the developed system for vehicle tracking and safe distance detection.

At the first step, videos were recorded using the drone, capturing aerial traffic on the highways. After that, were performed frame extraction and a manual annotation of the ground-truth boxes enclosing the objects of interest. In Addition, the appropriate dataset with vehicles and their respective classes was developed.

In the second stage, different network architectures of YOLOv4 Full and Tiny were trained and validated. Thus, the detection and classification of vehicles were carried out in order to evaluate the models.

During the third stage, the results obtained from the detections will feed the tracking system, allowing the estimation of speed and safe distance between vehicles.

A. Image Acquisition and Dataset

The drone captured the videos at FHD 1920x1080 resolution, $30fps$, on different streets and altitudes. A sample of 896 frames spaced in time was selected, avoiding similar images. Figure 2 illustrates a couple of sample frames from the assembled dataset.

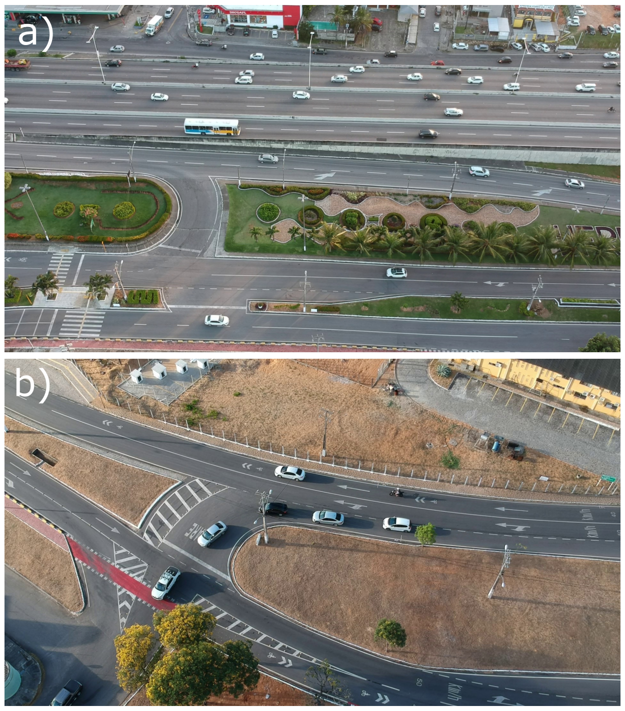

Fig. 2. Images captured by the DJI Spark drone. (a) Highway BR-101 at an altitude of 100m. (b) Streets near the campus of the Federal University of Rio Grande do Norte - UFRN at an altitude of 80m.

The sample frames were randomly divided into training, validation, and test sets, divided as 60% (636 images), 20% (180 images), and 20% (180 images), respectively. Each frame has multiple objects of different classes, with more occurrence of cars than buses and motorcycles. Table I shows the distribution of classes over the dataset.

TABLE I
DISTRIBUTION OF OBJECTS IN THE COMPOSITE DATASET

Dataset	**Training**	**Validation**	**Test**
Images	536	180	180
Instances	10126	3265	3827
Car	8982	2888	3376
Bus	303	109	112
Motorcycle	841	268	339

B. Image Processing and Annotation

Supervised learning algorithms need labeled data to learn patterns. For the object detection problem, that labels are annotations for each image, containing the position of each object in the image and your respective class.

Bounding boxes are used to separate objects in the images, as seen in Figure 3. Each box has a certain width and height given by Δx_i and Δy_i respectively, as well as its respective centroid given by $C_i = (x_{c_i}, y_{c_i})$. In addition, that data should be properly normalized to improve training and make the task easier to learn [13]. All the dataset annotations were performed using YoloMark and LabelMe [14] software.

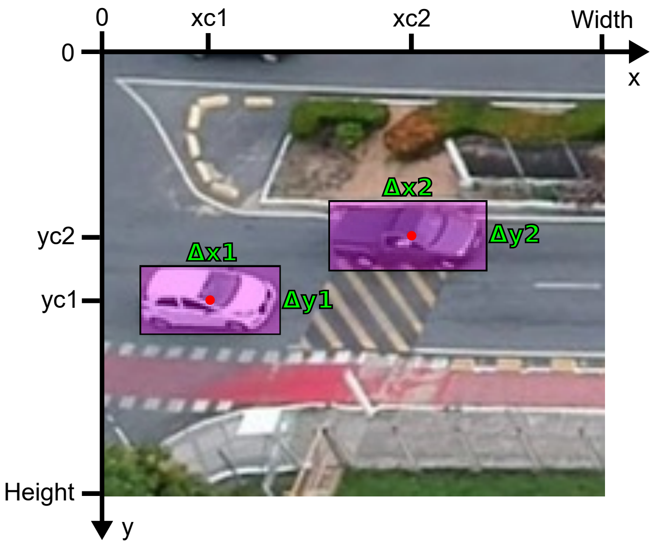

Fig. 3. Graphical image annotation approach used for dataset construction.

C. Vehicle detection with YOLOv4

The state-of-the-art algorithms for object detection are deep learning-based and can learn the feature extraction step. This ability comes from the extensive use of convolutional neural networks (ConvNet or CNN). In this context of CNNs, the You only look once [12] is one of the aforementioned state-of-the-art algorithms which targets real-time applications. Unlike some of its competitors, it is not a traditional classifier re-purposed as an object detector [15].

YOLO works by dividing the input image into a grid of $S \times S$ cells, where each of these cells is responsible for five bounding boxes predictions that describe the rectangle around the object. It also outputs a confidence score, which measures the certainty that an object was enclosed. Therefore, the score does not relate to the kind of object present in the box, only with the box's shape.

For each predicted bounding box, a class is also predicted to work just like a regular classifier, resulting in a probability distribution over all the possible classes. The confidence score for the bounding box and the class prediction combines into one final score that specifies the probability for each box includes a specific type of object. Given these design choices, most boxes will have low confidence scores, so only the boxes whose final score is beyond a threshold are kept.

As described in [12], the YOLOv4 network consists of the CSPDarknet53 backbone, SPP or PAN neck and YOLOv3 [16] head, and many other features, like Mosaic Data Augmentation, DropBlock Regularization, etc. An example of vehicle detection is shown in Figure 4.

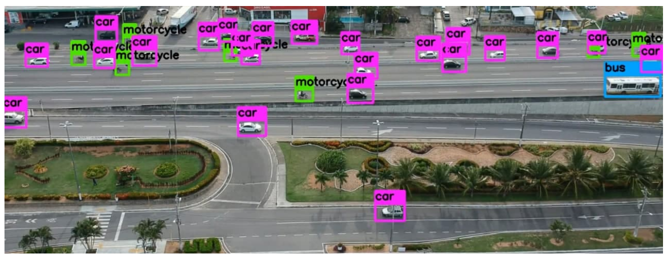

Fig. 4. Network outputs example for traffic on highway BR-101.

D. *Safe distance between moving vehicles*

Using vehicle positioning information over time returned by the system, it is possible to calculate the speed of each object. After 30 frames (1 per second second), the Euclidean distance in pixels of the objects between the position of the initial and final centroids is calculated. Thus, the speed of the moving vehicle in pixels per second is obtained.

The traffic of vehicles close to each other can lead to collisions, and this is a common type of problem all over the world [17]. Therefore, drivers must keep a safe distance from the car in front. There are laws in each country to determine the safe distance between vehicles, which may vary according to speed, weather, driving conditions, type of vehicle, etc.

In this context, a tool is proposed that considers both the speed and the distance between vehicles, allowing to determine a safe distance between them. An example can be seen in Figure 5.

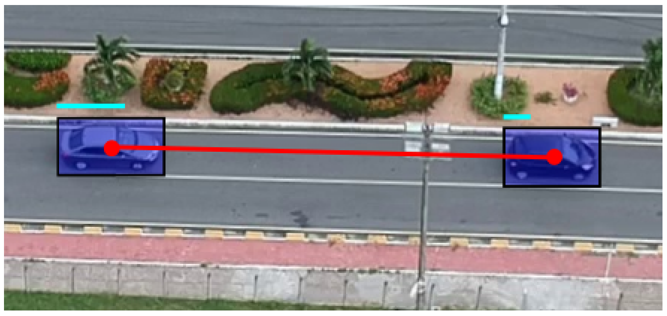

Fig. 5. Sample output of the safe distance tool. The bars on top of vehicles' bounding boxes are proportional to their speed, high speeds combined with small vehicle separation represent an unsafe driving situation.

III. Results

A. *Detection and Tracking*

Three types of models, Full, Tiny, and Tiny 3 Scales, each with two input resolutions (416×416 and 608×608), were trained and tested. Full YOLOv4 network performs 3 scales of detections and requires a size of 244.2MB. On the other hand, Tiny YOLOv4, which uses a smaller backbone than Full YOLOv4, performs 2 scales of detections and requires 22.4MB. The third model, called Tiny YOLOv4 3 Scales, use the Tiny YOLOv4 backbone, performs 3 scales of detections, and requires a size of 23.4MB.

In the training stage, we use the settings recommended by YOLOv4, they are: batch size of 64, the learning rate of 0.001, the momentum of 0.949, and the decay of 0.0005 for the optimizer algorithm to train all networks. The networks was trained fine-tuning an Imagenet [18] pre-trained weights, available on https://github.com/AlexeyAB/darknet.

The following metrics were used to evaluate these models: Intersection Over Union (IOU) and Average Average Precision (mAP). The IOU is defined by the intersection area of predicted and ground-truth boxes, divided by the union area of them, as shown in Equation 1.

$$IOU = \frac{B_P \cap B_T}{B_P \cup B_T} \tag{1}$$

where B_P and B_T are the predicted bounding box, and the truth bounding box, respectively. In this way were delimited true and false detections using an IOU threshold.

The Average Precision (AP) gets the area under the Precision-Recall curve, defined by Equation 2.

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{2}$$

where P_n and R_n are Precision and Recall at a certain threshold. The AP is the area under the interpolated recall-precision curve, calculated using the Intersection Over Union (IOU), with a threshold of 0.5, delimiting true and false detections. Thus, the mAP was defined as a mean AP for all classes.

The training curves of networks can be seen in details in Figures 6-9.

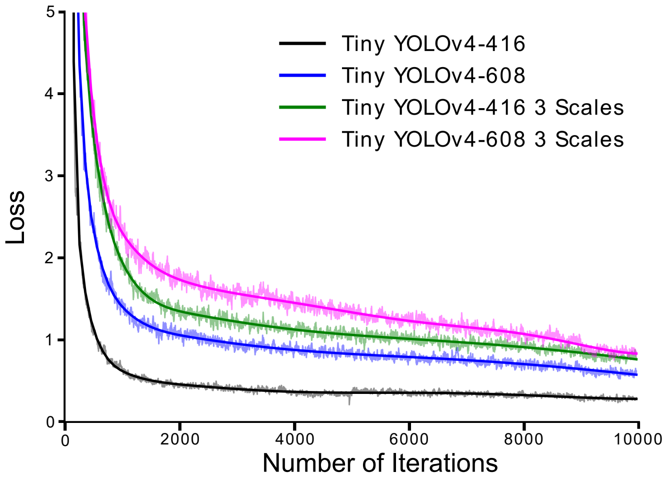

Fig. 6. Minimization of the loss function during the training process of the tested Tiny YOLOv4 configurations.

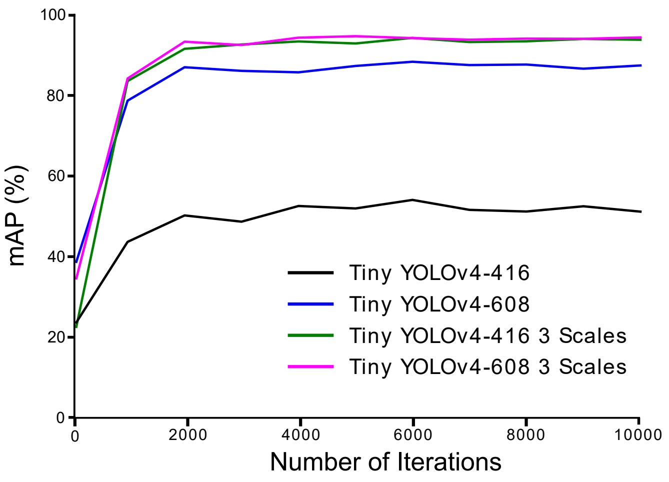

Fig. 7. Optimization of mAP during the training process of the tested Tiny YOLOv4 configurations.

Figure 8 shows the loss function over the iterations through the training dataset for the YOLOv4 Full models. Were observed that despite YOLOv4 Full 416 gets the lower loss, YOLOv4 Full 608 achieves the best validation mAP, how Figure 9 shows.

In the other models, were saw more stable loss curves because they have considerably fewer weights to train , as depicted in Figure 6. Tiny models get lower loss than Tiny 3 scales too, but Tiny 3 Scales presents best validation mAP, how were expected, as shown in figure 7.

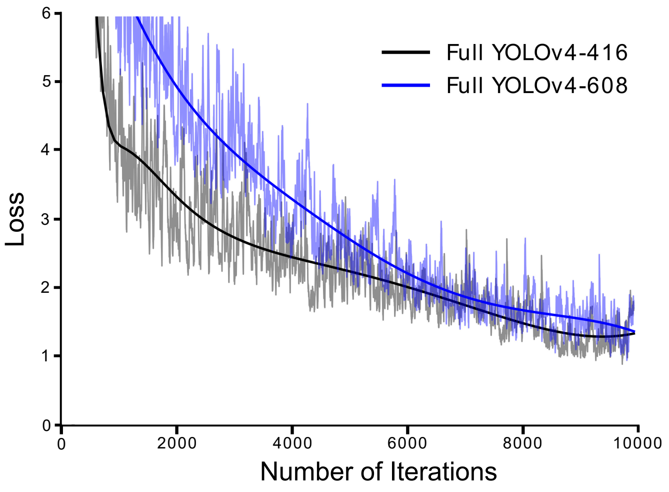

Fig. 8. Minimization of the loss function during the training process of the tested Full YOLOv4 configurations.

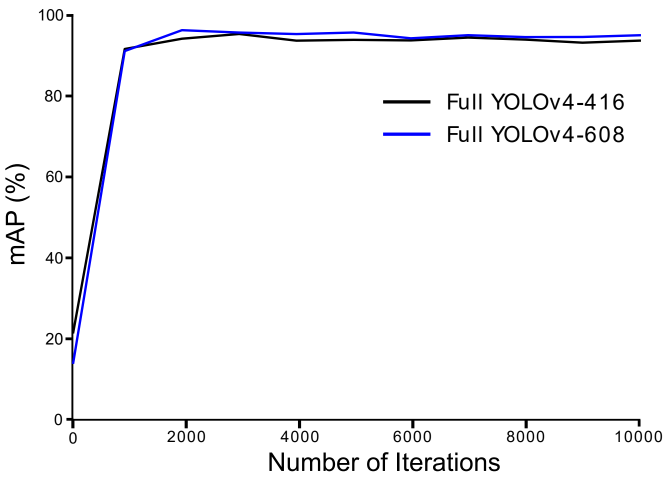

Fig. 9. Optimization of mAP during the training process of the tested Full YOLOv4 configurations.

The models were expected to generalize well for objects of different sizes and in regions with higher object density. Thus, 608×608 models filed better accurate results than 416×416. This point is fundamental for the knowledge of the dataset used, as it has some small objects (motorcycle class) that can appear in a large amount in a compact area. Table II shows the difference in training and inference time for the networks.

TABLE II
TRAINING AND INFERENCE TIMES FOR THE TESTED YOLOV4 CONFIGURATIONS.

Network	Training Time (hrs)	Inference Time (ms)
Tiny 416	3	4.96
Tiny 608	6	8.25
Tiny 416*	4	5.48
Tiny 608*	6	8.47
Full 416	20	32.76
Full 608	26	62.75

Were observed that Tiny 3 Scales models achieve similar time scores as Tiny models, but outperform in mAP, as shown in the table III. 608×608 models get the best IOU, Full 608 gets the best mAP, and Tiny 3 Scales models archive considerably better mAP than the Tiny models.

B. Safe Distance Tool

The tool developed begins with a region of interest (ROI) in the image, defined by the user. Thus, the detection of the safe distance between vehicles and velocity will be inside the ROI, as shown in Figure 10. The ROI is the blue box in Figure 10

TABLE III
ACCURACY COMPARISONS OF THE TESTED YOLOV4 CONFIGURATIONS.

Network	mAP(%)	Car(%)	Bus(%)	Mot.(%)	IOU(%)
Tiny 416	52.48	50.54	99.02	7.90	52.48
Tiny 608	89.68	92.60	**99.57**	76.88	**73.04**
Tiny 416*	96.54	95.42	98.89	95.31	68.47
Tiny 608*	96.83	96.61	99.17	94.73	70.00
Full 416	97.28	96.81	99.43	95.61	70.80
Full 608	**97.44**	**98.33**	99.11	**95.61**	71.35

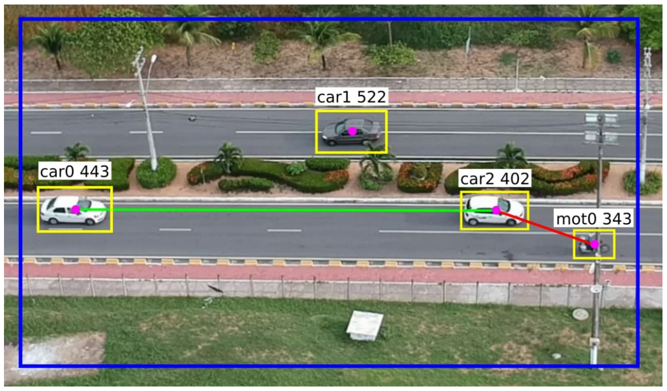

Fig. 10. Sample image of the Safe Distance Tool for two-way traffic.

The detected vehicle class, index, and distance to the closest vehicle are shown above each bounding box, and by the side, the vehicle velocity is displayed, as seen in Figure 10. Vehicles on different lanes can stay close and pose no danger. To avoid that, the vehicles are considered to be in the same lane and share very similar values for the y-coordinate of the center of mass, so this problem is solved by adding a limit for the y-coordinate.

The distance calculated with the tool proposed here is in pixels. Therefore, to be translated into real-world measurements, a series of calibration steps must be performed [19]. With proper software calibration, a safe distance is then defined, so whenever a vehicle in the video is at a shorter distance, the bounding box and line for the nearest vehicle will turn red, alerting the user to the infraction, and stay green for a safe distance, as is shown in Figure 10.

IV. CONCLUSION

The results presented show promising results, with high accuracy mainly using the YOLOv4 algorithm in the Full YOLOv4-608 and 3-scale Tiny YOLOv4-416 architectures. This allows proper tracking of vehicles from cameras with the superior vision on highways or even using drones.

The computational development presented in this work showed how it is possible to calculate the speed of vehicles in real-time. In this way, the risk of collision can be predicted in advance by calculating the safe distance between vehicles.

Finally, using techniques that relate computer vision and machine learning in their state of the art for object detection allowed the development of a system that, even preliminary, brings to discussion the development of technologies to reduce the high accident rates of traffic on the highways.

ACKNOWLEDGMENT

This work was supported by the School of Sciences and Technology at the Federal University of Rio Grande do Norte (ECT-UFRN).

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[2] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, “Self-driving cars,” *Computer*, vol. 50, no. 12, pp. 18–23, 2017.

[3] N. J. Goodall, “Can you program ethics into a self-driving car?” *IEEE Spectrum*, vol. 53, no. 6, pp. 28–58, 2016.

[4] W. H. Organization, “Global status report on road safety,” 2008.

[5] J. Kim, S. Kim, C. Ju, and H. I. Son, “Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications,” *IEEE Access*, vol. 7, pp. 105 100–105 115, 2019.

[6] W. W. Greenwood, J. P. Lynch, and D. Zekkos, “Applications of uavs in civil infrastructure,” *Journal of infrastructure systems*, vol. 25, no. 2, p. 04019002, 2019.

[7] K. T. San, S. J. Mun, Y. H. Choe, and Y. S. Chang, “Uav delivery monitoring system,” in *MATEC Web of Conferences*, vol. 151. EDP Sciences, 2018, p. 04011.

[8] A. Koubaa and B. Qureshi, “Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet,” *IEEE Access:10.1109/ACCESS.2018.2811762*, 2018.

[9] M. E. da Silva Bastos, V. Y. F. Freitas, R. S. T. de Menezes, and H. Maia, “Vehicle speed detection and safety distance estimation using aerial images of brazilian highways,” in *Anais do XLVII Seminário Integrado de Software e Hardware*. SBC, 2020, pp. 258–268.

[10] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.

[11] D. Fernández Llorca, A. Hernández Martínez, and I. García Daza, "Vision-based vehicle speed estimation: A survey," *IET Intelligent Transport Systems*, 2021.

[12] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[13] M. Shanker, M. Y. Hu, and M. S. Hung, "Effect of data standardization on neural network training," *Omega*, vol. 24, no. 4, pp. 385–397, 1996.

[14] K. Wada, "labelme: Image Polygonal Annotation with Python," https://github.com/wkentaro/labelme, 2016.

[15] R. S. T. de Menezes, R. M. Magalhaes, and H. Maia, "Object recognition using convolutional neural networks," in *Recent Trends in Artificial Neural Networks-from Training to Prediction*. IntechOpen, 2019.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[17] S. E. Lee, E. Llaneras, S. Klauer, and J. Sudweeks, "Analyses of rear-end crashes and near-crashes in the 100-car naturalistic driving study to support rear-signaling countermeasure development," *DOT HS*, vol. 810, p. 846, 2007.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[19] M.-C. Lu, C.-C. Hsu, Y. Y. Lu *et al.*, "Improvements and application of the image-based distance measuring system," in *Proc. WSEAS Int. Conf.(CISST)*, 2007, pp. 17–19.