A Real-Time Support for Safer Driving using Monocular Camera

Francesco Starna, Lorenzo Guercio

Abstract—In this paper, our goal is to help increasing the safety of the drivers thanks to a feedback system based exclusively on the use of a monocular camera mounted on the top of the vehicle. The proposed idea is a real-time application that warns the driver of imminent dangers on the road classified according to their risk level. This method takes into consideration various techniques related to computer vision, such as object detection, tracking and image processing. All these already existing techniques had the purpose of allowing us to do a detailed danger analysis in order to be able to carry out a potential hazard evaluating objects in the vehicle's range of motion. Finally, danger levels are used to warn the driver in a more or less invasive way, by means of a haptic feedback. The proposed method was tested using the KITTI dataset, obtaining positive results.

1 Introduction

In recent years, Artificial Intelligence for autonomous driving systems has become more and more important and is having a huge impact in our life. The most recent driving systems have been equipped with driver assistance functions such as Lane Keeping Assistant (LKA) [1] [2] [3], Adaptive Cruise Control (ACC) [4] and Brake Assist System (BAS) [5], in order to increase safety in driving. All of these assistant functions rely on Computer Vision algorithms, in particular on object detection and distance estimation. Those techniques may rely on different kinds of sensors, such as LiDAR scanner [6] [7], monocular [8] or stereo [9] cameras, and GPS [10]. The LiDAR scanner creates a 3D map of the surrounding environment and it is more accurate than 2D information, but it can be very expensive. GPS alone does not provide enough data for analysis. Camera images, instead, provide enough data in order to build a driver assistance function, and it is also a cheap sensor that can be installed in every system. In this paper we focus on developing a haptic feedback function that can be complementary to the EBA, to ensure safe driving and avoid potential risks before the latter is activated. The workflow is shown in Fig. 1. At each frame we first take the image coming from a monocular camera, placed above the vehicle roof. Then we detect and classify the objects in the scene using YOLOv4, that is a one-stage object detector. We manipulate the camera calibration matrix in order to recover longitudinal and lateral distances from the vehicle to the objects detected. After that, we apply an object tracking algorithm on the objects in order to track their motion. Then, for each object detected, we evaluate the potential danger based on different specific criteria. Finally, we warn the driver with a haptic feedback that depends on the danger, the higher the value the more intrusive the feedback, until we reach the maximum danger, where the EBA function will break the vehicle to avoid a hazard. We evaluate our method on the KITTI raw dataset, using mean average precision (mAP) for object detection, and root mean square error (RMSE) for distance estimation.

2 Related Works

2.1 Object Detectors

There are mainly two kinds of state-of-the-art detectors: two-stage detectors and one-stage detectors. The first ones (i) use a Region Proposal Network, which is a convolutional neural network, to identify potential regions of interest in a first stage, (ii) and then send to the pipeline the proposals for object classification and bounding box regression.

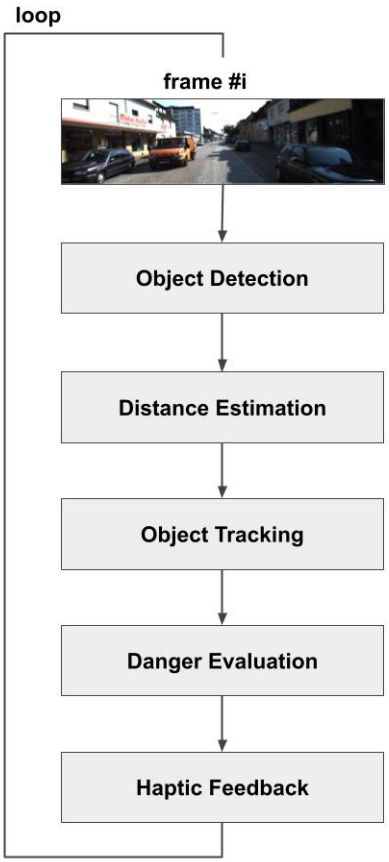


Fig. 1: Workflow of the application

These detectors are the most accurate but are typically slower. These include R-CNN [11], Faster R-CNN [12], and other variations. On the other hand, the one-stage detectors treat object detection as a simple regression problem, taking the images and learning bounding boxes coordinates and class probabilities [13]. These detectors reach generally a lower accuracy, but are really faster. Among the most known detectors there are the Single Shot Detector (SSD) [14] and You Only Look Once (YOLO) [15]. In order to develop a real-time system, we need a fast method that can run higher than 10 frames per second (FPS), because the human vision system can process approximately 10 images per second and perceive them individually, while higher FPS are perceived as motion. Therefore the choice of a one-stage detector was obvious, and in particular, the YOLOv4 [16] version was chosen because of its improved speed and performances.
For more accurate insights on the topic, Jiao Fellow etc. [17] have analyzed through a survey, describing a variety of object detection methods.

2.2 Distance Estimation

The state-of-the-art distance methods used in autonomous driving are mainly related to LiDAR technology. This technique is similar to radar: in both cases, the distance is calculated through the time of flight. These two techniques have in common the high cost required to install a good device on a vehicle. In our work we focused on using only images coming from a camera; therefore, using a much cheaper approach with respect to LiDAR and other expensive sensors. The methods using image features make use of computer vision techniques: one may exploit stereo cameras in order to triangulate objects and estimate distances. Many recent works use inverse perspective mapping (IPM) to flip the front view of the vehicle and compute the longitudinal distances using the corresponding homography. Kim [18] uses IPM and YOLO in order to estimate 2D bounding boxes and then compute distances. Qiao [19] exploits camera matrix, IPM and lane detection in order to recover x and y ratios and compute euclidean distances. Adamshuk [20] uses IPM and HSV colormap to distinguish the region of interest (ROI) and retrieve forward distances. However all these approaches rely on the IPM, which is built from image vanishing point and for this reason it strictly depends on the road conformation; in fact, the technique fails when approaching turns. Another possibility is to use machine learning techniques to directly estimate distances. In Disnet [21] they use YOLO for object detection and for training a neural network to estimate distances in a supervised approach, providing an annotated dataset with 2D bounding boxes and distances. Kumar [22] created a FisheyeDistanceNet in order to estimate an Euclidean depth map extracting features from fisheye images. Machine learning approaches are powerful but also time consuming, since they estimate distances based on supervised learned networks.

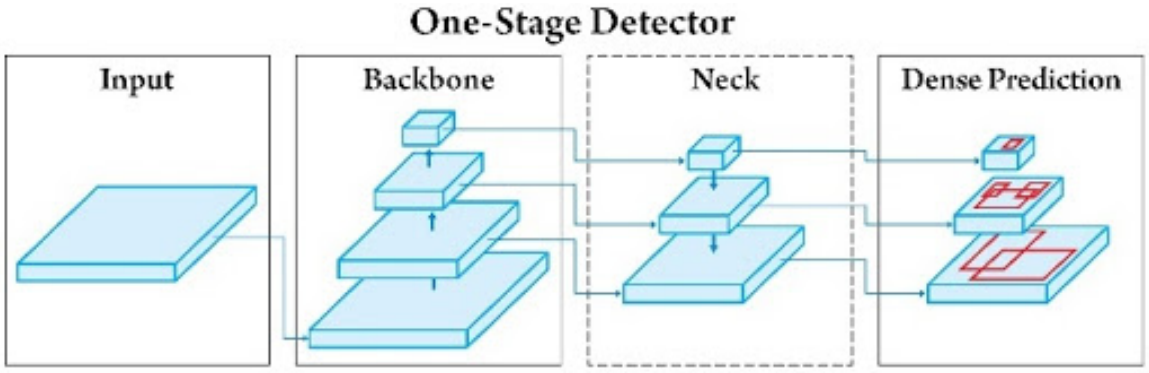


Fig. 2: One-Stage Object Detector

Our approach is fast and simple; we manipulate the camera matrix, and thanks to some assumptions, we are able to retrieve longitudinal and lateral distance only exploiting the 3D-2D correspondences given by the Pinhole camera model.

3 Pipeline

In this section we present our work. Each step of the pipeline is executed frame by frame in real time during the driving.

3.1 Object Detection

We want first to detect all the objects in the scene in order to evaluate a possible danger. We focused our attention on the one-stage detector YOLOv4, which is the fourth improved version of YOLO, according to inference time and average precision. A modern one-stage object detector is generally composed by three parts: (i) a backbone, which is pre-trained on ImageNet [23] and is a feature extractor, (ii) a neck which is a network that analyzes features from different stages, and (iii) a head, which is responsible for making bounding boxes and class predictions. An example can be seen in Fig. 2

YOLOv4 consists of:

- Backbone: CSPDarknet53
- Neck: SPP, PANet
- Head: YOLOv3

3.1.1 Backbone

The Cross Stage Partial Network (CSPNet) [24] was introduced to reduce the computation of heavy neural networks, which is fundamental to develop real-time applications in really small devices. YOLOv4 applied CSPNet to Darknet53, which is a convolutional neural network using residual connections introduced in YOLOv3 [25]. The network's structure can be seen in Fig. 3.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3: Network Structure

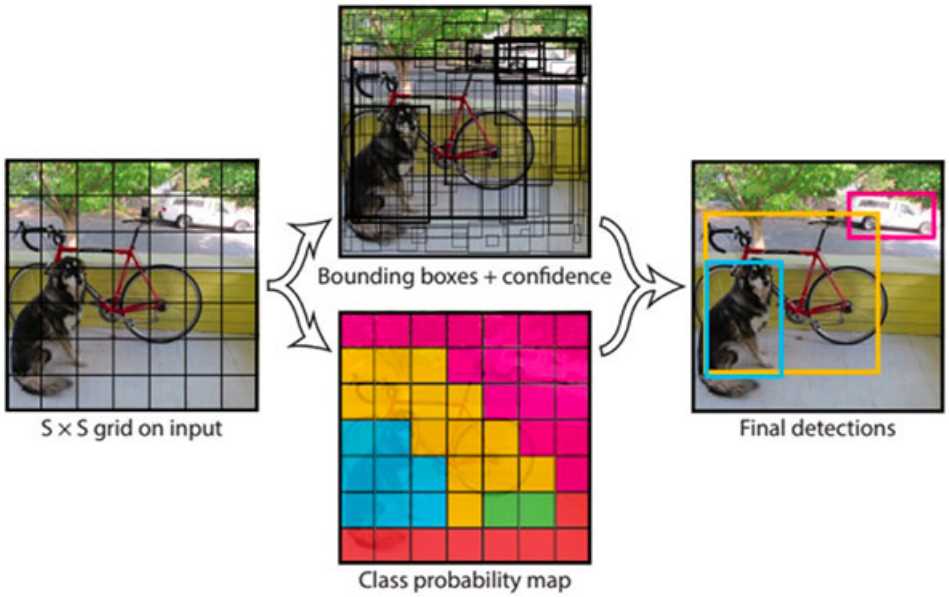


Fig. 4: YOLO Design

3.1.2 Neck

As additional blocks after the backbone YOLOv4 implements Spatial Pyramid Pooling (SPP) [26], which is a more robust method to image deformations (crop/warp) for both object detection and classification. To complete the section, the Path Aggregation Network (PANet) [27] is added to enhance the entire feature hierarchy, in order to let useful information in each feature level propagate directly to following proposal subnetworks.

3.1.3 Head

YOLO divides the image into an S × S grid and for each grid cell predicts B bounding boxes, each one consisting of 5 predictions: *x*, *y*, *w*, *h* and *confidence*. Each grid cell also predicts C conditional class probabilities: *Prob (Class|Object)*. YOLOv3's main idea is totally based on the original work, even if it predicts boxes across three different scales, using a similar concept to Feature Pyramid Networks [28]. For this reason, the predictions for the third scale benefit from all the prior computation as well as fine grained features from early on in the network. The 2D bounding box predictions are sufficient to estimate the distances from the vehicle to all the objects in the scene, since we need to know only the lower side of the box, which represents the contact point of the object with the road. It can be seen from Fig. 4, how YOLO works.

3.2 Distance Estimation

Before going into details, we have to state two simplifying assumptions: (a1) the road on which the vehicle and all the objects in the scene lie, must be a planar surface, (a2) the camera installed on the vehicle must be stationary. These simplifications allowed us to develop an efficient, easy and fast computational method for distance estimation.

3.2.1 Camera Matrix

Multiple View Geometry in Computer Vision [29] describes how the pinhole camera model maps world points to image points. Using homogeneous coordinates we can write:

$$P = K[R \mid t] = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix}$$

where:

- K is the 3x3 camera calibration matrix containing the intrinsic parameters, describing the focal length, the optical center, and the skew coefficient.

- R and t are the extrinsic parameters, namely rotation and translation of a rigid transformation from 3D world coordinate system to the 3D camera's coordinate system.

In order to obtain the camera matrix, it is necessary to perform a camera calibration process that can be done in different ways, also using computer vision libraries such as OpenCV. Given the camera matrix P, we can define the mapping from an image point x to a world point X as:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (1)$$

3.3 2D-3D Correspondence

We start with defining the camera frame RF_c, which we will follow from now on for every transformation, as represented in Fig. 5. Given the camera matrix **P**, the height of the camera from the road to the vehicle roof **h**, and the longitudinal distance from the camera to the front bumper of the car **b**, we can now manipulate the camera in order to obtain a one-to-one correspondence that maps image points x to world points on the road X. We first translate the camera by multiplication with a transformation matrix **T**:

$$P_t = PT = P \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where $t_y = h$ and $t_x = camera\_to\_bumper$

As we can see in Fig. 6, we have translated the camera matrix on the road (height = 0) and towards the front side of the vehicle, through sequence of homogeneous transformations, so that the distances are computed directly from the central point of the bumper. At this point the assumptions come handy. Thanks to (a1) we simply eliminate from the camera matrix P_t the Y column, meaning that all the points projected to the real world have height equal

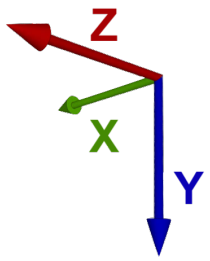


Fig. 5: Camera Frame. X axis points on the right side, Y axis points downwards the road, Z axis points in front of the car.

to zero. This is beneficial for computing the inverse projective mapping of (1), from world point to image point:

$$\begin{pmatrix} X \\ Z \\ 1 \end{pmatrix} = P^{-1}_{t,y=0} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2)$$

Thanks to (a2) we are able to define this mapping in every camera frame, ignoring all the disturbances due to road irregularities and vehicle movements. With this mapping it is easy to estimate longitudinal and lateral distances. Given a generic 2D bounding box (*x1*, *x2*, *x3*, *x4*), coming from the object detection step, we take the two contact points of the object with the road in image coordinates, and we apply the inverse mapping (2), which gives us the left end and right end sides in world coordinates. At this point we take the midpoint between the two (t_{long}, t_{lat}, 1), which corresponds to longitudinal and lateral planar distances from the vehicle bumper to that object.

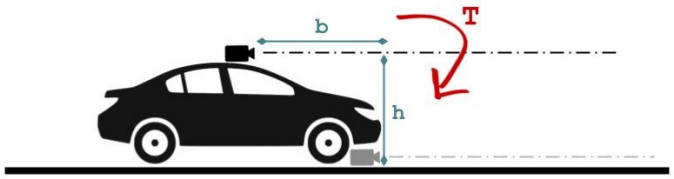


Fig. 6: Visualization of the T matrix transformation

3.4 Object Tracking

With the purpose of obtaining more specific information about the objects in the scene, we implemented an Euclidean object tracking system. It works by storing a dictionary of objects' longitudinal and lateral distances, with their IDs as keys. For each object detected in a generic frame, the tracker is updated, taking as input the vector (t_{long}, t_{lat}) of that object. The tracker compares the vector with the stored dictionary, using Euclidean difference, and if the new vector is sufficiently "close" to something, the dictionary entry of the corresponding point is updated with the new distances, otherwise it is marked as a new ID. The tracker works really well with objects detected within a certain lateral distance range, beyond which we are no more interested in tracking. Fig. 7 shows an example of tracking between a bunch of image frames. Once we have the IDs of the objects in the scene, we can compute a few more properties such as the relative longitudinal and lateral velocities (v_{long}, v_{lat}) of the objects with respect to the vehicle, which are important in the danger evaluation phase for making predictions of potential collisions.

Fig. 7: Object Tracking Example

3.5 Danger Evaluation

Evaluating potential danger situations, in order to provide a danger haptic feedback (DHF) complementary to the EBA function, is the core work of our project.

3.5.1 Zone subdivisions and coefficients

We apply different criteria according to the following lateral distance subdivisions (distances are considered laterally in both directions):

- **danger zone**: from 0 to 2 m. These are all the objects detected right in front of the vehicle, considering a total span of 4 meters.
- **attention zone**: from 2 to 5 m. These are all the objects detected just close to the vehicle.
- **safe zone**: from 5 to 10 m. Further objects.

The zone limits have been chosen according to some considerations: 1) the danger zone corresponds to the maximum lane width (3.75 m) approximated to the next integer, 2) the attention zone comprises the next lane and an eventual sidewalk, 3) the safe zone extends till the predictions of the object recognition system give reliable results laterally (10 m). Distances are taken in modulus and computed starting from the midpoint of the front bumper of the vehicle, positive to the right. The subdivision can be seen in Fig. 8.

We assigned to each zone a support coefficient and to each object class a vulnerability coefficient. Such coefficients are reported in Tab. 1 and Tab. 2 respectively. The support coefficients

TABLE 1: Table of support coefficient for evaluating dangerousness in the different zones.

	Danger	Attention	Safe
Support	1.0	0.8	0.5

TABLE 2: Table of vulnerability coefficient for evaluating dangerousness for different classes.

	Car	Van	Truck	Tram	Misc	Pedestrian	Cyclist
Vulnerability	1.0	1.0	0.8	0.9	1.0	1.5	1.5

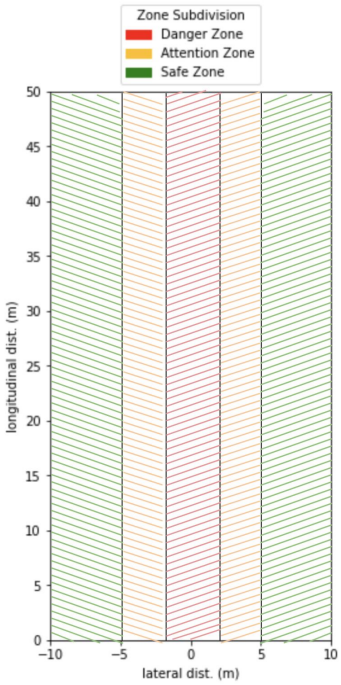


Fig. 8: Subdivisions of zones.

add a reduction contribution to dangerousness, according to the zone where the object detected is lying. The further the zone, the less dangerous the situation. The vulnerability coefficients take into consideration which class of object we are considering. Colliding a cyclist or a pedestrian, for example, is more dangerous than colliding a truck, because it may cause a serious damage to the integrity and security of people with a higher probability.

3.5.2 Evaluation

For each object detected, we evaluate the dangerousness of a potential accident in the following manner:

$$\boldsymbol{D} = remap(V_{cr} * \alpha_v * \beta_z)$$

where D is the evaluated dangerousnees, V_{cr} is the criterion value, α_v is the vulnerability class coefficient, β_z is the zone coefficient, and the $remap$ function remaps the value to the range 0, 10. Essentially the output value of the chosen criteria is then smoothed by the two coefficients and finally remapped to a valuable range. The criterion are: stopping (longitudinal) distance, Euclidean distance, intersection distance.

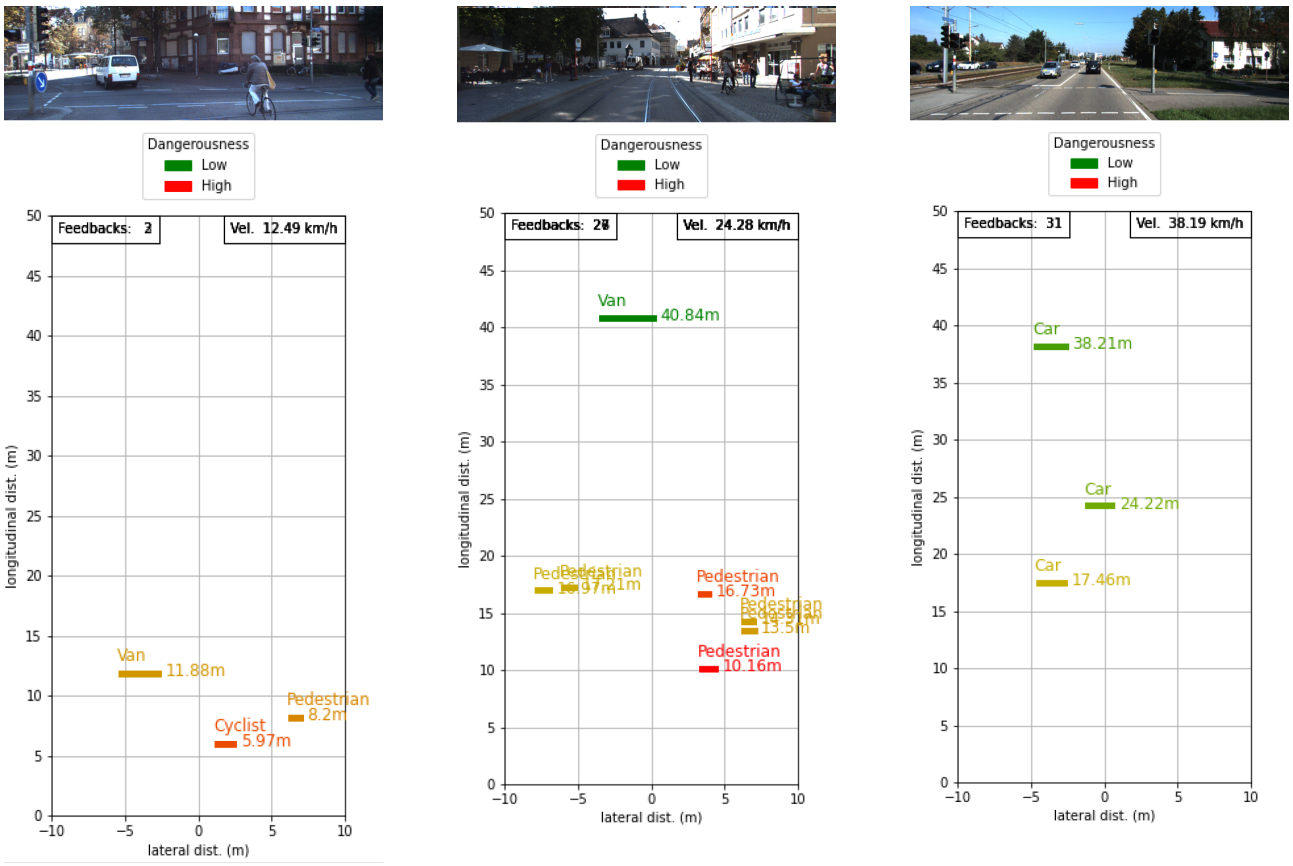
In this way we restrict the value of the multiplication from 0 to 10 applying different criteria according to the zone on which the object is laying. In every zone, we make use of the longitudinal velocity of the vehicle v_e, which can be retrieved by sensors, such as Active Sensor Bearing (ASB), GPS or Inertial Measurement Unit (IMU).

Danger zone. When considering an object detected in front of the vehicle, the danger comes from the possible collision due to insufficient *stopping distance* t_{stop}. In fact, in each frame, we take into consideration the closest (longitudinally) objects detected in the scene, taking their longitudinal distance. The smaller t_{stop} - t_{long}, the higher the probabilities to collide, if the object in front of the vehicle unexpectedly stops. We compute the stopping distance as the sum of the perception-reaction distance and the braking distance as:

$$v_e * t_{reaction} + v_e^2/(2 * \mu * g)$$

where $t_{reaction}$ is the reaction time, μ is the friction coefficient and g is the gravity of the earth. Reasonable values for $t_{reaction}$ and μ are respectively 1 second and 0.8, but these can vary according to the age of the vehicle driver and to what kind of vehicle he is driving [30].

Attention zone. In the nearby zone, the danger comes from the possibility of some objects



(a) frame#2-0005 (b) frame#70-0091 (c) frame#186-0015

Fig. 9: Examples of Danger Evaluation

to unexpectedly cross the danger zone and appear in front of the vehicle. In this case we increase or decrease the danger according to the following rules:

- If some object is going towards the danger zone and if the predicted lateral shift of that object intersects the predicted longitudinal shift of the vehicle (the directions of the object and the vehicle intersect), and it happens in less than 3 seconds, the danger increase inversely proportional to the intersection time. We call this criteria *intersection distance*.
- Otherwise, we consider only the *Euclidean distance*, from the vehicle to the objects laying on the attention zone.

Safe zone. Objects detected at a lateral distance higher than 5 meters are not dangerous at all. In order to preserve the previous criteria and provide continuity to danger evaluation, especially in the case of objects crossing from the safe to the attention zone, we use as a criterion the *Euclidean distance*, from the vehicle to the objects laying on the safe zone. Fig. 9 shows some examples of danger evaluation, in different scenarios.

3.6 Haptic Feedback

The value of dangerousness computed in the last step serves to give feedback to the driver. Different kinds of actuators for our feedback system are possible, e.g. vibration system installed on the steering wheel, smart band on the wrist, etc... Every system is capable of generating a vibration on a intensity scale from 1 to 3. The values of dangerousness may be relevant and worthy of attention when greater than 7. According to that value, we can map our haptic feedback system together with the EBA as shown in Tab. 3.

TABLE 3: Different level of haptic feedbacks depending on the dangerousness value

Dangerousness D	$7 \leq D < 8$	$8 \leq D < 9$	$9 \leq D < 10$	≥ 10
Haptic Feedback	1-level	2-level	3-level	Break (EBA)

TABLE 4: Characteristics of different scenarios

Scenario	**# Frames**	**# Labels**	**Labels per Frame**	**Difficulty**
0005	153	15	0.098	Easy
0015	297	36	0.121	Moderate
0091	340	68	0.200	Hard

TABLE 5: Performances on two different scenarios. The 0005 is calculated considering distances from a front cyclist. The 0015 is calculated considering distances from a front car.

Scenario	**Distance**	**Long. RMSE**	**Lateral RMSE**
0005	<10m	0.908	0.306
	10m-30m	1.931	0.204
	30m-50m	None	None
	Total	1.492	0.262
0015	<10m	None	None
	10m-30m	2.425	0.130
	30m-50m	3.785	0.179
	Total	3.025	0.151

4 Experiments

In this section we report the experiments that we did and the results obtained, also mentioning the relative data used and the hardware on which we ran our project.

4.1 Dataset

We tested our work using KITTI [31], which provides an annotated dataset for 2D and 3D object detection, and also a raw dataset for testing purposes. We used the object data for training and validating the object detection model, while the raw data was used for testing the object detection, the distance estimation and the object tracking, since the IMU information is provided only for the least. We focused on the scenarios 0005, 0015, and 0091 of the raw data, which respectively represent urban, highway and limited traffic zone (ZTL) scenarios. Tab. 4 summarizes the characteristics of the different scenarios.

4.2 Performances

We developed and tested our project using Google Colab. At the moment the OS system was Ubuntu 18.04.5 LTS (Bionic Beaver) with Intel(R) Xeon(R) CPU @ 2.30GHz and Tesla T4 12 GB GPU. With this architecture our pipeline runs at 15 FPS.

4.2.1 *Object Detection*

We trained the YOLOv4 model on KITTI 2D object data, consisting of 7481 images randomly divided into 80% train, 10% validation and 10% test. The network has been initialized with pre-trained weights on ImageNet. Training has been done using stochastic gradient descent with warm restart (SGDR) [32] optimizer, with momentum 0.9, learning rate 10^{-3}, and decay $5*10^{-4}$, with batch size 64, for a total of 7500 iterations (about 60 epochs). We evaluated the object detection on a small amount of data (about 750 images), because unfortunately the test set is not annotated. We obtained a mean Average Precision (mAP) of 92.8%.

4.2.2 *Distance Estimation*

We evaluated the distances with the RMSE metric, distinguishing between short/long and longitudinal/lateral distances within 50 meters. Tab. 5 shows the results of the RMSE metric over two different KITTI videos, while in Fig. 10 the error plots can be seen . Fig. 11 shows some qualitative results of detection and distance estimation.

5 Conclusions

The proposed method has proved considerably effective in the conditions established at the be-

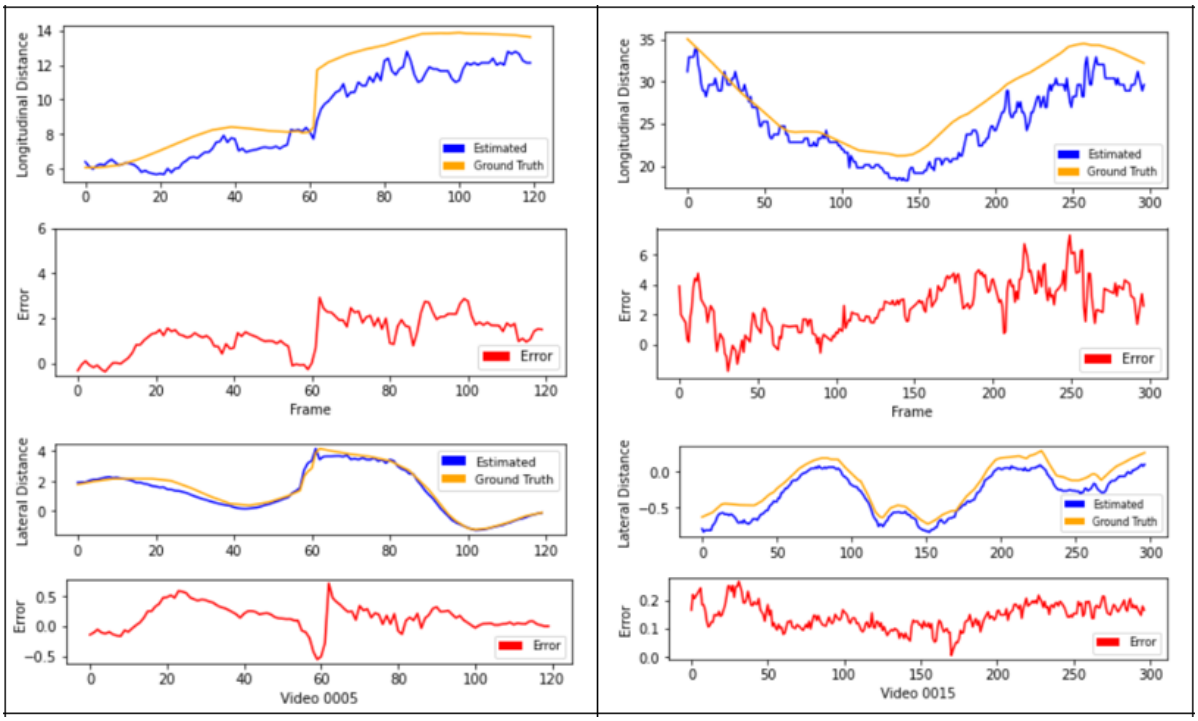
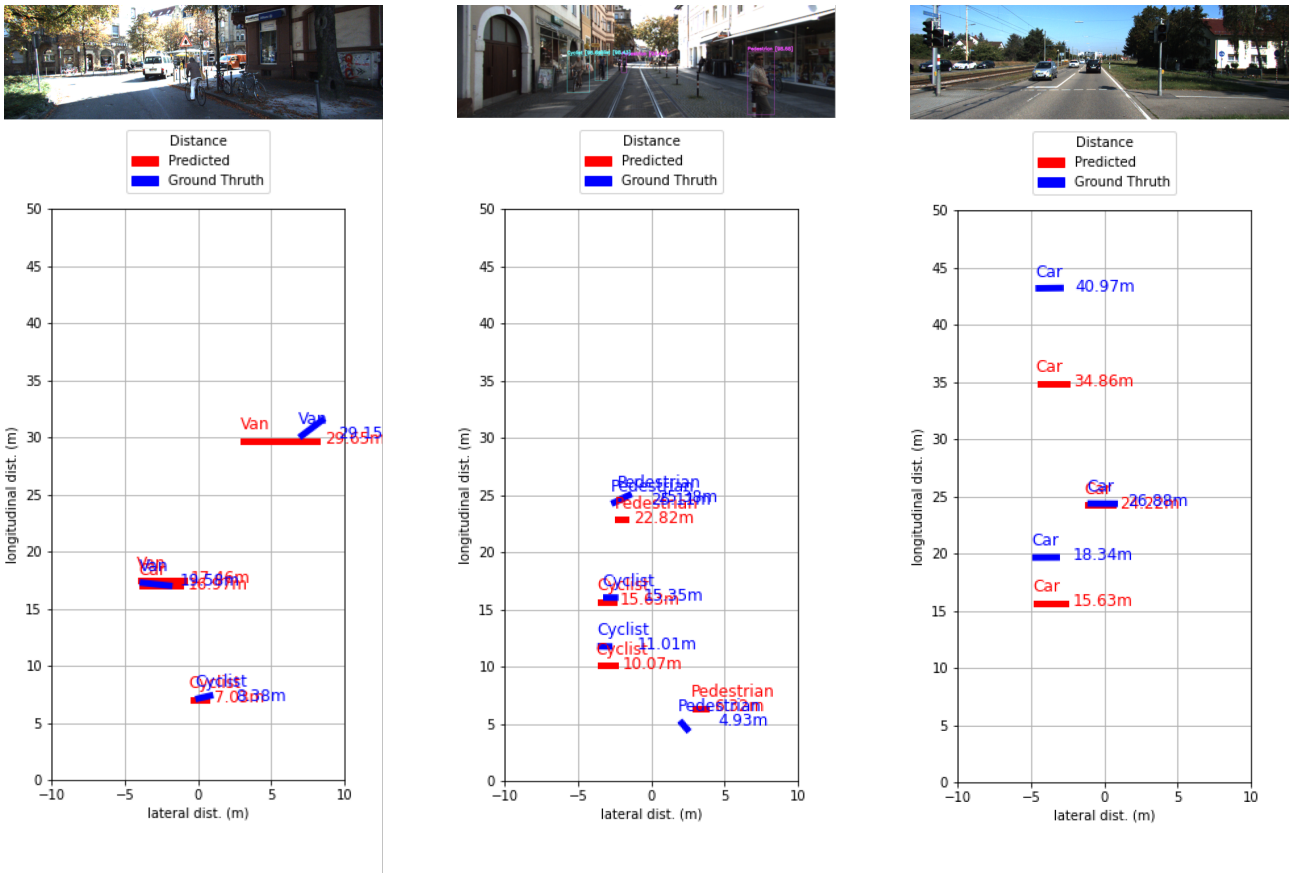


Fig. 10: Distance performances on 2 different scenarios



(a) frame#41-0005 (b) frame#187-0091 (c) frame#301-0015

Fig. 11: Detection and Distance Estimation Plots

ginning of the paper and the budget necessary for its implementation is very low. However, it is not free from weaknesses and improvement points.

5.1 Weaknesses

- Total reliance on object detection: if done wrong, there is no other type of check that we can perform.
- Object detection detects up to more or less 50 meters: it would be better if the detection reached about 100 meters (in order to be more effective in highways for example)
- The proposed method works in a worse way in case of adverse visibility conditions or at night, due to a worse performance obtained with the object detection.
- Due to the initial assumptions, we have a considerable margin of error when the machine ”goes up and down” due to the bumps.

5.2 Improvements Points

- The evaluation of the danger must be improved because it gives too many feedbacks to the vehicle driver.
- The object detection could be improved especially when vehicles approach ours; considering the orientation of the objects could help significantly.
- Develop a smoothing algorithm to avoid oscillating distances and to compensate for the car jolts.
- Improve the danger evaluation through other parameters, such as driver's age and experience, or using a learning algorithm that learns how the human behaves when driving and produces a negative or positive parameter.

REFERENCES

[1] *Lane Departure Warning System*. https://en.wikipedia.org/wiki/Lane_departure_warning_system

[2] S. Ishida and J. E. Gayko, *Development, evaluation and introduction of a lane keeping assistance system*. IEEE Intelligent Vehicles Symposium, 2004, 2004, pp. 943-944 url-https://doi.org/10.1109/IVS.2004.1336512

[3] A. Mammeri, G. Lu and A. Boukerche, *Design of lane keeping assist system for autonomous vehicles*. 7th International Conference on New Technologies, Mobility and Security (NTMS), 2015, pp. 1-5 https://doi.org/10.1109/NTMS.2015.7266483

[4] Greg Marsden, Mike McDonald, Mark Brackstone, *Towards an understanding of adaptive cruise control*. Transportation Research Part C: Emerging Technologies, 2001, https://www.sciencedirect.com/science/article/pii/S0968090X0000022X

[5] Hirose T., Taniguchi T., Hatano T., Takahashi K., et al., *A Study on the Effect of Brake Assist Systems (BAS)*. SAE Int. J. Passeng. Cars - Mech. Syst. 1(1):729-735, 2009, https://doi.org/10.4271/2008-01-0824

[6] Li, You and Ibanez-Guzman, Javier *Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems*. IEEE Signal Processing Magazine 10.1109/MSP.2020.2973615

[7] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger *Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving*. https://arxiv.org/abs/1812.07179

[8] Chen Xiaozhi, Kundu Kaustav, Zhang Ziyum, Ma Huimin, Fidler Sanja, Urtasun Raquel *Monocular 3D Object Detection for Autonomous Driving*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 https://www.cs.toronto.edu/~urtasun/publications/chen_etal_cvpr16.pdf

[9] Peiliang Li, Xiaozhi Chen, Shaojie Shen *Stereo R-CNN Based 3D Object Detection for Autonomous Driving*. https://arxiv.org/abs/1902.09738

[10] Kumar, S., Moore, K.B *The Evolution of Global Positioning System (GPS) Technology*. Journal of Science Education and Technology 11, 59–80 (2002) https://doi.org/10.1023/A:1013999415003

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*. CoRR, abs/1311.2524, 2013 http://arxiv.org/abs/1311.2524

[12] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. CoRR, abs/1506.01497, 2015 http://arxiv.org/abs/1506.01497

[13] Petru Soviany, Radu Tudor Ionescu, *Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction*. CoRR, abs/1803.08707, 2018 http://arxiv.org/abs/1803.08707

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, Alexander C. Berg, *SSD: Single Shot MultiBox Detector*. CoRR, abs/1512.02325, 2015 http://arxiv.org/abs/1512.02325

[15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*. CoRR, abs/1506.02640, 2015 http://arxiv.org/abs/1506.02640

[16] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*. CoRR, abs/2004.10934, 2020 https://arxiv.org/abs/2004.10934

[17] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, Rong Qu, *A Survey of Deep Learning-based Object Detection*. CoRR, abs/1907.09408, 2019 http://arxiv.org/abs/1907.09408

[18] Youngseok Kim, Dongsuk Kum, *Deep Learning based Vehi-*

cle Position and Orientation Estimation via Inverse Perspective Mapping Image. CoRR, 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 317-323 https://doi.org/10.1109/IVS.2019.8814050

[19] D. Qiao, F. Zulkernine, *Vision-based Vehicle Detection and Distance Estimation*. 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 2836-2842 https://doi.org/10.1109/SSCI47803.2020.9308364

[20] R. Adamshuk et al., *On the applicability of inverse perspective mapping for the forward distance estimation based on the HSV colormap*. 2017 IEEE International Conference on Industrial Technology (ICIT), 2017, pp. 1036-1041 https://doi.org/10.1109/ICIT.2017.7915504

[21] M. Haseeb, *DisNet: A novel method for distance estimation from monocular camera*. Institute of Automation, University of Bremen, 2018

[22] V. R. Kumar et al., *FisheyeDistanceNet: Self-Supervised Scale-Aware Distance Estimation using Monocular Fisheye Camera for Autonomous Driving*. 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 574-581 https://doi.org/10.1109/ICRA40945.2020.9197319

[23] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, *ImageNet: A large-scale hierarchical image database*. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255 https://doi.org/10.1109/CVPR.2009.5206848

[24] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. CoRR, abs/1911.11929, 2019 http://arxiv.org/abs/1911.11929

[25] Joseph Redmon, Ali Farhadi, *YOLOv3: An Incremental Improvement*. CoRR, abs/1804.02767, 2018 http://arxiv.org/abs/1804.02767

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. CoRR, abs/1406.4729, 2014 http://arxiv.org/abs/1406.4729

[27] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, *Path Aggregation Network for Instance Segmentation*. CoRR, abs/1803.01534, 2018 http://arxiv.org/abs/1803.01534

[28] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, Serge J. Belongie, *Feature Pyramid Networks for Object Detection*. CoRR, abs/1612.03144, 2016 http://arxiv.org/abs/1612.03144

[29] Hartley, R., Zisserman, A., *Multiple View Geometry in Computer Vision (2nd ed.)*. Cambridge: Cambridge University Press., 2004 https://doi.org/10.1017/CBO9780511811685

[30] *Braking Distance*. https://en.wikipedia.org/wiki/Braking_distance

[31] Andreas Geiger, Philip Lenz, Christoph Stiller, Raquel Urtasun, *Vision meets Robotics: The KITTI Dataset*. International Journal of Robotics Research (IJRR), 2013

[32] Ilya Loshchilov, Frank Hutter, *SGDR: Stochastic Gradient Descent with Restarts*. CoRR, abs/1608.03983, 2016 http://arxiv.org/abs/1608.03983