Safety-aware metrics for object detectors in autonomous driving

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—We argue that object detectors in the safety critical domain should prioritize detection of objects that are most likely to interfere with the actions of the autonomous actor. Especially, this applies to objects that can impact the actor's safety and reliability. In the context of autonomous driving, we propose new object detection metrics that reward the correct identification of objects that are most likely to interact with the subject vehicle (i.e., the actor), and that may affect its driving decision. To achieve this, we build a criticality model to reward the detection of the objects based on proximity, orientation, and relative velocity with respect to the subject vehicle. Then, we apply our model on the recent autonomous driving dataset nuScenes, and we compare eight different object detectors. Results show that, in several settings, object detectors that perform best according to the nuScenes ranking are not the preferable ones when the focus is shifted on safety and reliability.**

***Index Terms*—Object detection, autonomous driving, machine learning, metrics, safety, reliability**

I. Introduction

The goal of object detection is to perceive and locate instances of semantic objects of a certain class [1]. It is one of the most relevant branches of computer vision, and it has been widely applied in many different applications and domains. A multitude of solutions have been proposed for 2D and 3D object detection, based on data sensed through different sensors, like visual cameras and lidars [2], [3].

Object detection is fundamental in emerging safety-critical applications, and in particular it is a major pillar of present and future autonomous driving applications [4]. To study object detection in the autonomous driving domain, new datasets are continuously proposed, for example KITTI [5], VOC [6], CityScapes [7], and more recently Waymo [8], nuScenes [9], and Level5 Lyft [10]. The best-performing object detectors are evaluated on these datasets using widely acknowledged metrics [11], [12] and evaluation routines provided by the datasets' owners, allowing a fair comparison. In particular, the Average Precision metric (first presented in [13]) has emerged in recent years as a summary of the precision-recall curve, and it is currently deemed the most suitable metric to summarize and rank the performance of object detectors.

However, we argue that current metrics for object detection do not match the demands and peculiarities of a safety-critical system. The evaluations based on Average Precision typically focus on judging how well the object detectors detect objects, without any attention to the current behavior of these objects or to their possibility to interfere with the vehicle; in other words, their relevance for the driving task is not considered. To clarify, let us consider the typical pipeline for autonomous driving [14]: the subject vehicle is sensing the surroundings to perform object detection, and the detection output is used for path planning. Let us now consider other two vehicles in the sensed scenario, one directed straight towards the autonomous vehicle, in a colliding trajectory, and one headed away from the autonomous vehicle. Clearly, for the safety of the driving task, it is critical to detect the first one, while detection of the second vehicle is not relevant at all. Unfortunately, this is not captured by the metrics currently used in object detectors, which would consider both objects equally relevant.

In this paper we propose novel metrics for object detection in the safety-critical domain, with specific contextualization to the domain of autonomous driving. We propose evaluation metrics that reward object detectors based on their ability to correctly detect objects that may interfere with the subject vehicle, and whose presence requires proper response. To this end, we build an object criticality model that performs a rating of the objects, based on the distance from the vehicles, the possible colliding trajectory, and the expected time to collision (based on relative velocities). The model revises the usual metrics to include such rating, and to consider the concepts of safety and reliability.

We then exercise our approach on the autonomous driving dataset nuScenes. We apply eight 3D-object detectors: seven of them use lidars, while one relies on the visual camera. We compute our metrics for all the eight detectors, and we also show that under various settings the ranking we obtain differs from the one achieved using the nuScenes evaluation library, which relies on traditional metrics.

The rest of the paper is organized as follows. Section II presents basic notions and the related works. Section III shows the metrics we are introducing, including the model that is required for their computation. Section IV describes the application of the metrics to nuScenes. Section V illustrates the results, in which object detectors are ranked according to our metrics and to the traditional metrics, showing differences. Section VI concludes the paper.

II. Background and Related Works

A. Object detection with Deep Neural Networks (DNNs)

Object detection deals with detecting instances of visual objects of a certain class (such as humans, animals, or cars) in digital data [15], [1], for example in camera's RGB images and videos, or in lidar's pointclouds.

The spatial location and extent of an object are typically defined coarsely using a *bounding box*, i.e., an axis-aligned rectangle tightly bounding the object, although other less common approaches exist, for example pixelwise segmentation masks [16], [2]. For simplicity, in the rest of this paper we only consider bounding boxes. Object detectors compute bounding boxes with an assigned confidence score. Then, a *detection threshold* is applied as a configuration parameter: all bounding boxes with confidence score above the detection threshold are maintained (i.e., these are the predictions of the object detector), while those below the detection threshold are discarded by the object detector itself.

Most recent object detectors exploit Deep Neural Networks (DNNs), which notably outperform other machine learners in image-based applications [17] [18]. Especially when lidars' pointclouds are involved, such detectors are organized in a backbone that performs feature extraction from the input, an optional neck that further elaborates features, and an head that finally produces the predictions [19].

B. Identifying correct and incorrect detections

The predictions of an object detector are usually classified starting from the computation of: i) true positives (TPs), which are correct detections of ground truth bounding boxes; ii) false positives (FPs), which are incorrect detections of nonexistent objects or misplaced detections of existing objects; and iii) false negatives (FNs), which are undetected ground truth bounding boxes [12]. It is important to observe that true negatives (TNs) are not taken into account, because there are infinite bounding boxes that should *not* be detected within any given image [12].

To decide on TPs, FPs, and FNs, a comparison between the predicted bounding boxes and the ground truth bounding boxes is performed. Typically, this comparison is based on a measure of distance between bounding boxes, for example the distance between their center points [20]. Briefly, a detected object is a TP if the ground truth bounding box and the detected bounding box are closer than a *distance limit*. If there is no predicted bounding box that matches this criterion, then the object is not detected and it counts as an FN. Predicted bounding boxes that are farther than the distance limit from all ground truth bounding boxes are considered FPs.

C. Metrics for performance evaluation

The conventional approach to the evaluation of object detectors consists of a set of metrics that are derived from the TP, FP, and FN reviewed above. Such metrics form the basis for our safety-oriented metrics defined later, and they are reviewed below [11], [12], [21].

Precision (P) is the number of true positives (TP) over the number of true positives plus the number of false positives i.e., $P = TP/(TP + FP)$. Precision indicates how many of the selected items are relevant. If some non-relevant items are selected, this reduces precision. Precision is 1 if all the detected objects exist, and 0 in the opposite case.

Still, there could be some real objects that are not detected: this is not captured by precision, because it does not include FNs, and it is addressed by the recall. *Recall* (R) is the number of true positives (TP) over the number of true positives plus the number of false negatives (FN) i.e., $R = TP/(TP+FN)$. Recall indicates how many of the existing items are selected. If a detector has recall 1, it means it detected everything without any detection miss; in the opposite case, recall is 0.

An object detector with high recall but low precision outputs many predictions, but most of them are incorrect. An object detector with high precision but low recall is just the opposite: it returns very few predictions, but most of them are correct. This is captured very well by the *Precision-Recall Curve*, which shows the tradeoff between precision and recall for different detection thresholds.

Since recent years, the most frequently used summarizing metric is *Average Precision* (AP) [15]. Average precision summarizes the precision-recall curve as the weighted mean of precision scores achieved at different detection thresholds, with the increase in recall from the previous detection threshold used as the weight. More precisely [12], [21]:

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

where P_n and R_n are the precision and recall at the n-th detection threshold. In this paper, in agreement with [9], we calculate AP by applying Equation 1, but only for recall and precision above or equal to 0.1. Cases in which recall or precision is less than 0.1 are removed in order to minimize the impact of noise commonly seen in regions with low precision or low recall [9].

D. Related works on object detection in critical systems

The adoption of artificial intelligence, including object detection tasks, in critical systems comes with a relevant set of renown challenges, because of the many distinguishing aspects of the problem, its complexity, and also the variety of applications [22], [23], [24]. For safety-critical systems, some incorrect predictions may lead to catastrophic losses, and therefore have the maximum relevance [25], while others may have minimal impact. However, when we consider the metrics to evaluate the performance of object detectors, safety or other system-level dependability attributes are generally not considered, and metrics from Section II-C are being used. This also applies to the wide domain of autonomous driving. Evidence is easily achieved considering the metrics used in object detection challenges for autonomous driving. For example, these challenges are continuously proposed by the owners of the most-known datasets, like KITTI [5], [26], CityScapes [7], [27], Waymo [8], [28], or nuScenes [9], [20],

and the selected metrics revolve around average precision and the concepts of Section II-C.

Up to now, very few or no approaches have attempted to define safety or reliability metrics for object detectors; the works that are closer to such goal are reviewed below. Noteworthy, all such works appeared in recent years, which underlines a recent understanding of the relevance of the subject.

The work in [29] introduces the distinction of a critical area, which is the area nearby the vehicle where failed detection of an object may lead to immediate safety risks. The work acknowledges that the design of a driving application is focused on both i) guaranteeing safety on such critical area, and ii) guaranteeing high detection accuracy on the non-critical area (in order to have a smooth driving). This observation leads the author to build different DNNs for the detection of objects in the two areas.

Some works address the problem of model uncertainty in autonomous driving, where the term uncertainty should be interpreted in the broad sense of how certain a model is about its predictions [30]. The work in [30] arguments that object detectors should also include prediction confidence, and it presents various methods to capture uncertainties in object detection for autonomous driving. Otherwise, object detectors can only tell the human drivers what they have seen, but not how certain they are about it. The work in [31] builds a model that includes information on uncertainty sources (e.g., sensor noise), the work in [32] includes uncertainty when computing the bounding box regression loss, and the work in [33] captures in the model both the noise inherent to the observations and the uncertainty that can be explained away given enough data. Last, despite not focusing on object detection, the work in [25] defines safety-oriented metrics by proposing that predictions with a confidence score close to the detection threshold should be treated differently and more suspiciously.

Still, the reviewed works weight all the detected objects the same, i.e., when assessing the object detector, the usual binary (i.e., yes/no) counting of TPs, FPs, and FNs is performed. Instead, in this work we claim that i) object detectors should be evaluated depending on the ability to detect those objects that are most likely to affect the driving task, i.e., impact on safety and reliability, and ii) this can be realized by weighting the objects based on their criticality, and by building specific metrics that consider such weights.

We conclude reporting that standardization initiatives to regulate the usage of machine learning in safety-critical systems are being proposed [34], [35], [36], [37], [38], which also describe means to assess applications that exploit machine learning. However, at present these initiatives do not include ad-hoc metrics like we are proposing in this paper.

III. Object Criticality Model

Our model is based on assigning a *criticality* value to each object that is in the scene, and then computing object detection metrics that consider this criticality. The description of the model is independent of the sensors used to capture the scene (e.g., cameras or lidars) and of the objects in the scene.

A. Requirements and assumptions

The application of the model requires i) a roving vehicle (named *ego* afterwards) that captures the scene with sensors as cameras and lidars, and ii) objects (other vehicles, pedestrians, etc.) that are within line-of-sight to ego and that are consequently captured by the sensors. This is the very typical situation of an autonomous vehicle which performs object detection.

We assume that the following ground truth values are available: i) 3D bounding boxes describing the size of the objects; ii) coordinates of ego and the objects; iii) velocity of ego and the objects. This means that, to apply our model, we need autonomous driving datasets equipped with such information. The most recent automotive datasets have very rich meta-data, typically including the above information; for example in Section IV and Section V we will use nuScenes [9], which satisfies our assumptions.

Further, we assume that the object detector outputs i) the computed 3D bounding boxes, ii) the estimated distance of detected objects from ego, and iii) an estimate of the velocity of objects. There are several 3D object detectors that include the estimates above in their output; noteworthy, these estimates are required in the object detection challenges of the nuScenes community [20].

For simplicity of the discussion, when computing coordinates of the objects and their relative distance from ego, we consider only the (x, y) coordinates i.e., we ignore the vertical position of the objects and of ego. For the nuScenes dataset we use in this paper, this is not an issue, because data was collected on flat lands. Extending the model to consider the relative elevation is definitely possible, only at the cost of slightly more complex geometric computations.

B. Structure of the model

We call ego the roving vehicle that mounts the sensors and collects data from the environment, and we call object B any other object. We do not restrict the type of objects, for example B can be a car, a pedestrian, a bike, etc. Note that for ego we only have ground truth values i.e., the object detector does not predict its own velocity or position.

The construction of our model is organized in 3 steps, which are repeated for all the objects B within the line of sight of ego, and for both the ground truth values and the predicted values of B.

The first step is the analysis of the collision scenario involving B and ego. In this step we calculate indicators that will be later used to define the criticality of B. In particular, we calculate i) the initial distance d between ego and B, ii) the closest distance r that ego and B would reach, and iii) the time Δt that ego and B require to reach such distance. These values are input to the following step, together with other information as the current position and velocity of ego

and B. The analysis of the collision scenario is explored in Section III-C.

The second step is the calculation of *criticality weights* that are assigned to each object B. These are $\kappa_d(B)$, $\kappa_r(B)$, and $\kappa_t(B)$: $\kappa_d(B)$ is based on the distance between ego and B; $\kappa_r(B)$ is based on the distance between ego and the potential collision point with B; and $\kappa_t(B)$ is based on the estimated time for ego and B to reach the collision point. The calculation of these values is explained in Section III-D.

These weights are assigned depending on the relevance of B for the driving task of ego, i.e., weights are higher if B may affect the behavior of ego. These weights are used either as rewards or penalties depending, respectively, on whether the object has been detected (true positive, TP) or missed (false negative, FN). Further, the weights also apply to false positives (FP), i.e., object falsely detected with nothing actually on the scene in the predicted position.

The third step exploits the assigned criticality to construct *safety and reliability metrics* that allows understanding the safety and reliability of the object detector. The computation of the criticality is described in Section III-E.

C. Analysis of the collision scenario

We define $ego = (ego_x, ego_y)$ the position of ego, and $B = (B_x, B_y)$ the position of the object B in the captured scene. Further, we define, in vector form, $v_{ego} = (v_{ego_x}, v_{ego_y})$ the velocity of ego, and $v_B = \left(v_{B_x}, v_{B_y}\right)$ the velocity of B.

We compute the relative velocity of B with respect to ego, as $v_{rel} = \left(v_{rel_x}, v_{rel_y}\right) = \left(v_{B_x} - v_{ego_x}, v_{B_y} - v_{ego_y}\right)$, that is, the vectorial difference of the velocity of ego and the velocity of B. This allows us simplifying the subsequent calculations: we can consider ego as stationary, while B is moving with the velocity resulting from the sum of the two velocity vectors v_{ego} and v_B.

Then, we identify the shortest distance from ego at which object B will pass if both continue moving with the same velocity (i.e., vectorial speed). This is the distance between ego and point $C = (C_x, C_y)$, with C being the point closest to ego on the trajectory of B. Point C can also be thought as the tangent point between the line representing the trajectory of B and a circle centered on ego (see Figure 1).

We describe the computation of $C = (C_x, C_y)$ using basic Euclidean geometry. The line defining the direction of the relative movement of B is obtained from the general equation of a line, i.e., $y - y_0 = m(x - x_0)$. We are looking for the line passing from point (B_x, B_y) and whose angular coefficient (i.e., orientation with respect to the x axis) is given by the ratio between the y and x components of the velocity vector of B (refer again to Figure 1). That is:

$$y - B_y = m(x - B_x), \quad \text{with} \quad m = \frac{v_{rel_y}}{v_{rel_x}}. \tag{2}$$

The shortest distance between a line l and a point p is given by the line perpendicular to l passing from p. By definition, a line perpendicular to one having coefficient m has coefficient

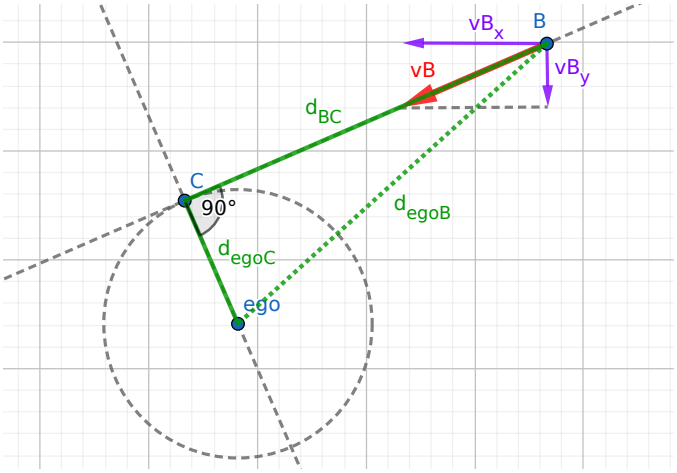


Fig. 1: Geometrical representation of the main elements of our model.

$m' = -1/m$. To find the line that passes from the position of ego is thus $y - ego_y = m'(x - ego_x)$ with $m' = -\frac{1}{m}$, that is:

$$y = -\frac{v_{rel_x}}{v_{rel_y}}(x - ego_x) + ego_y. \tag{3}$$

Point (C_x, C_y) is then the point at which the lines intersect. We omit computations for brevity, but the point is obtained by solving Equation 4:

$$\begin{cases} C_y = \frac{v_{rel_y}}{v_{rel_x}}(C_x - B_x) + B_y \\ C_y = -\frac{v_{rel_x}}{v_{rel_y}}(C_x - ego_x) + ego_y \end{cases} \tag{4}$$

Applying the Euclidean distance, we can easily compute (Figure 1): i) the distance d_{egoB} between ego and B, ii) the distance d_{egoC} between ego and C, and iii) the distance d_{BC} between B and C.

Assuming that both ego and B continue moving with the same speed, the time Δt that B needs to reach the collision point C is then computed as the distance divided by the scalar speed of B, i.e., $\Delta t = d_{BC}/|v|$, where $|v| = \sqrt{v_x^2 + v_y^2}$. Note that ego is considered to be stationary, while B moves with a relative velocity obtained as the difference of the velocities of the two objects.

The model above does not consider some corner cases, for which the result would be undefined. We treat these cases by skipping the model calculation and setting the criticality values to fixed fallback values. Corner cases are discussed in Section III-D.

D. Computation of criticality weights

In the definition of our criticality weights, we are inspired by reliability analysis [39], in which metrics like *reliability* (or *safety*) are defined in the interval $[0, 1]$. Note, however, that we do not propose probability values.

Each object B identified by the object detector (being either a TP, a FP, or a FN) is assigned a *criticality weight* $\kappa(B)$. This weight is obtained combining the three criticality values

$\kappa_d(B)$, $\kappa_r(B)$, and $\kappa_t(B)$, as explained below. Note that for a given object B, its criticality $\kappa(B)$ may be different if calculated with its predicted properties (e.g., position and velocity) or the ground truth ones. Furthermore, for some objects we may have ground truth values only (FNs) or predicted values only (FPs). When needed, we indicate with $\kappa'(B)$ the criticality value computed with predicted properties of object B, as opposed to $\kappa(B)$ that is calculated based on the ground truth.

1) Distance Criticality: $\kappa_d(B)$ is a criticality score based on the distance d_{egoB} between ego and the object. This score does not depend on velocity, but only on the position of objects in the scene. We want the score to be maximum when the distance to the object is zero, and then decrease to zero when reaching a maximum distance $D_{max} > 0$.

We compute the weight $\kappa_d(B)$ as a second-degree equation (downward parabola) passing from points $(0, 1)$ and $(D_{max}, 0)$:

$$\kappa_d(B) = -\frac{1}{D_{max}^2} d_{egoB}^2 + 1. \tag{5}$$

This means that the maximum value is 1.0 when $d_{egoB} = 0$ and it decreases until 0.0 when $d_{ego} = D_{max}$. The parabola shape allows the criticality to decrease non-linearly with respect to distance: the decrease is slow for values near to zero (i.e., close to the vehicle), and it gets faster when approaching D_{max} (i.e., far from the vehicle).

We also need to enforce that $\kappa_d(B)$ is always in the interval $[0, 1]$, and therefore the final equation is:

$$\kappa_d(B) = max\left(0, -\frac{1}{D_{max}^2} d_{egoB}^2 + 1\right). \tag{6}$$

2) Collision Distance Criticality: $\kappa_r(B)$ is a criticality score based on the distance between ego and the potential collision point C. It is an indicator of how close the object is likely to pass to the subject (i.e., the autonomous vehicle).

The weight $\kappa_r(B)$ is calculated following the same rationale of $\kappa_d(B)$, but with a different parameter $R_{max} > 0$. If the potential collision point is farther than R_{max}, the corresponding criticality is zero. The corresponding formula is then:

$$\kappa_r(B) = max\left(0, -\frac{1}{R_{max}^2} d_{egoC}^2 + 1\right). \tag{7}$$

3) Collision Time Criticality: The third score, $\kappa_t(B)$, is based on the time Δt for B to reach the potential collision point. All the other things unchanged, this score depends on the (relative) velocity of the object B with respect to ego.

Similarly, we set a parameter T_{max}, beyond which we consider that the collision time is not critical. The score is again calculated as a second-degree equation passing from $(0, 1)$ and $(T_{max}, 0)$, so that the maximum value is 1.0 when $\Delta t = 0$ and it decreases until 0.0 when $\Delta t = T_{max}$. Again, the result should be in the interval $[0, 1]$, thus:

$$\kappa_t(B) = max\left(0, -\frac{1}{T_{max}^2} \Delta t^2 + 1\right). \tag{8}$$

4) Criticality of Object: The final criticality $\kappa(B)$ is obtained by combination of the three criticality values $\kappa_d(B)$, $\kappa_r(B)$, $\kappa_t(B)$. The resulting metric is defined following three requirements: i) it should range in the interval $[0, 1]$; ii) it should be 0 if all values are zero; iii) it should be 1 if at least one of the values is 1; and iv) it should increase if any of the three values increases.

Inspired by classic reliability analysis [39], we observe that the failure probability of a three-components series system has exactly the same properties. Its formula is given by:

$$\begin{aligned} R_{series} &= R_A \cdot R_B \cdot R_C \\ 1 - F_{series} &= (1 - F_A) \cdot (1 - F_B) \cdot (1 - F_C), \end{aligned} \tag{9}$$

where $R_a = 1 - F_a$ is the reliability of component a. Our final criticality weight is then computed as:

$$\begin{aligned} 1 - \kappa &= (1 - \kappa_d) \cdot (1 - \kappa_r) \cdot (1 - \kappa_t) \\ \kappa &= 1 - (1 - \kappa_d) \cdot (1 - \kappa_r) \cdot (1 - \kappa_t), \end{aligned} \tag{10}$$

where $\kappa(B)$, $\kappa_d(B)$, $\kappa_r(B)$, and $\kappa_t(B)$ have been abbreviated for simplicity as κ, κ_d, κ_r, and κ_t. The final criticality $\kappa(B)$ is therefore a measure of: how much the object is close, how much it is likely to pass close in the near future, and how much time is available to react.

5) Corner Cases: The following corner cases are considered:

- When *ego* and B are moving at the exact same velocity (in both dimensions), the resulting relative velocity is zero, and Δt cannot be computed. We solve this case by setting κ_r and κ_t to zero.
- The case in which only one component of the relative velocity is zero *does not* need to be treated differently. The resolution of Equation 3 yields a form in which the denominator is the sum of squares of the two components of the velocity. The denominator is thus zero only when both components of the velocity are zero, which is already treated in the previous case.
- In the calculation of Δt we need to verify if the object B is actually moving towards point C, and not on the same line but in the opposite direction. In case B is moving in the opposite direction, κ_r and κ_t are again set to zero.
- In rare cases, where the collision point is particularly far away or the speed is particularly low, the calculation of Δt may generate an overflow or a not-a-number (NaN) value: in this case κ_t is set to 0.1. The rationale is to set it to a low value, but still greater than zero.
- The dataset may contain invalid values, or the detector may not be able to provide estimates. In particular, when we are not able to obtain the velocity of the object, we set κ_r and κ_t to 1 (their maximum value).

E. Proposed summarizing metrics

For each object B, we compute $\kappa_d(B)$, $\kappa_r(B)$, $\kappa_t(B)$, and ultimately $\kappa(B)$, for both the ground truth values and the predicted values. We base on these values to propose metrics that can rank object detectors according to principles of safety and reliability of the driving task. We want to upgrade

the traditional recall and precision metrics, such that they also measure the safety and the reliability offered by object detectors.

1) Reliability: We start from *reliability*, i.e., continuity of correct service [40], first explaining the rationale of our approach, and then discussing the metric. For a reliable driving task, a good object detector should *not* predict false positives that correspond to dangerous situations, because they could lead to an interruption of the driving task, for example to an unnecessary brake. Our approach is based on the consideration that the detection of false objects could disrupt the continuity of service, and thus system reliability. In other words, visualizing objects that are actually not present could lead to useless safety braking or alternative response strategies. Instead, to guarantee continuity of the driving mission, some risks of collision may be unavoidable. This clearly conflicts with safety, but it is widely accepted that safety and reliability have different goals [40].

For this reason, the metric we choose for *object detection reliability* is based on a revised definition of *precision*. The idea is that false positives are penalizing the continuity of the driving process, with a greater impact the closer they are, or are likely to be, to the ego vehicle.

We revise the precision metric such that TPs and FPs are weighed according to the criticality $\kappa(B)$ of the associated object B. In other words, considering the formula of precision, $P = \frac{TP}{TP+FP}$, in case an object B is detected, we do not add 1 to the count of TPs, but we add its criticality $\kappa(B)$, and analogously for FPs.

To compute the weight for a single object, we could use the κ value computed either using the ground truth or the predicted values. We use ground truth values at the numerator, and predicted values at the denominator. The idea is that the detector might detect a greater criticality (denominator) than what is actually present (numerator), thus reducing reliability. Also, clearly we do not have ground truth values for FP, because they are non-existent objects.

Denoting with $\kappa(B)$ the criticality of object B, calculated with the methods discussed above, we can calculate the *reliability-weighted precision* as:

$$P_{\mathcal{R}} = \frac{\sum_{B \in TP^*} \kappa(B)}{\sum_{B \in TP^*} \kappa'(B) + \sum_{B \in FP^*} \kappa'(B)}, \qquad (11)$$

where TP^* is the set of true positive objects (detected and existing), and FP^* is the set of false positive objects (detected, but not existing). Note that the $P_{\mathcal{R}}$ may in principle raise above 1, in case the detected criticality is consistently lower than the ground truth. To be consistent with the classic definitions of precision, we limit the maximum value of $P_{\mathcal{R}}$ to 1.

2) Safety: In the case of *safety*, it is requested that the object detector predicts the dangerous objects. We define the *safety-weighted recall*, $R_{\mathcal{S}}$, computed starting from the usual recall $R = TP/(TP + FN)$. We use the ground truth values at the denominator, and the detected values at the numerator. In fact, the metric should reflect how much of the existing criticality (denominator) has been detected by the object detector (numerator). Also, we clearly do not have predicted values for FN, which are objects that have been missed.

We can calculate the *safety-weighted recall* as:

$$R_{\mathcal{S}} = \frac{\sum_{B \in TP^*} \kappa'(B)}{\sum_{B \in TP^*} \kappa(B) + \sum_{B \in FN^*} \kappa(B)} \qquad (12)$$

where TP^* is the set of true positive objects (detected and existing), and FN^* is the set of false negative objects (not detected, but existing). Also for $R_{\mathcal{S}}$ we limit its maximum value to 1.

3) Safety-Reliability Tradeoff and Average Precision: The proposed criticality values depend on three parameters, namely D_{max}, R_{max}, and T_{max}. We can compute $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for different values of these parameters, to understand their evolution when different subset of objects are considered. In analogy to the precision-recall curve (see Section II-C), this allows computing several $P_{\mathcal{R}}$-$R_{\mathcal{S}}$ curves, one for each combination of values $(D_{max}, R_{max}, T_{max})$; consequently, the AP (Average Precision) can be computed from each of the $P_{\mathcal{R}}$-$R_{\mathcal{S}}$ curves. In the following, we call AP_{crit} the AP computed based on our definitions of $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$.

Depending on the driving scenario and the specific application of the object detector, different values of D_{max}, R_{max}, and T_{max} may be favored. For example, an object detector which is very good on $P_{\mathcal{R}}$ could be safely used in a highway under low traffic conditions; however, if it is not good on $R_{\mathcal{S}}$ it should not be used in an urban scenario, where cars may approach from different directions at essentially any angle.

Last, we remark that our metrics model should not replace other metrics, but should be complementary to them to provide a complete understanding of the object detection performance. Further, it is easy to note that our metrics can be reduced to the traditional ones, by simply setting all the weights (κ values) to 1.

IV. Case Study on the nuScenes Dataset

To exercise our model, we choose the nuScenes dataset for the following reasons: i) it is very recent and extensive, forged with the latest technology for autonomous driving sensors; ii) very recent object detectors are available; iii) it includes all the information we need to apply the model in Section III.

A. The nuScenes Dataset

NuScenes [9], [20] is a recent large-scale dataset for autonomous driving that reports scenes collected from a vehicle equipped with 6 cameras, 5 radars and 1 lidar, all with full 360 degrees field of view. In addition, GPS coordinates and movement dynamics from an Inertial Measurement Unit (IMU) sensor are reported. The dataset comprises 1000 scenes, each being 20 seconds long and fully annotated with 3D bounding boxes. *Keyframes* (image, lidar, radar) are sampled at 2Hz (every 0.5 seconds); in each keyframe there are on average 7 pedestrians and 20 vehicles. Five intermediate frames are collected between keyframes. Driving routes capture a diverse set of locations (urban, residential, nature, and industrial),

times (day and night), and weather conditions (sun, rain, and clouds).

B. 3D object detection in nuScenes

Following common practices in datasets of this kind [5], [41], nuScenes defines an object detection task and proposes related metrics to officially rank object detectors on its website [20]. The detection task in nuScenes consists in predicting the objects at time t, using sensors data collected between $(t-0.5, t]$ seconds. The scene collected at time t is called *keyframe*, and for the purpose of detection five intermediate frames collected in $(t-0.5, t]$ are used. Detectable objects are all objects within 50 meters from ego and with line of sight. For each object, ground truth 3D bounding boxes, attributes (e.g., orientation), and velocities are provided. Detectable objects are organized in 10 classes: barrier, traffic cone, bicycle, motorcycle, pedestrian, car, bus, construction vehicle, trailer, and truck: in this paper we consider only the car class for brevity. A detection is successful if the distance between the centers of the predicted and ground-truth bounding boxes is less than a distance limit l; four different values of l are considered, which are $\{0.5, 1, 2, 4\}$ meters.

The main metrics nuScenes proposes to summarize results are precision, recall, precision-recall curve, and the Average Precision. These metrics are computed for each individual class and for each distance limit l.

C. Object detection algorithms

We select eight 3D object detectors from the "model zoo" of mmdetection3d [42], an open-source object detection toolbox based on PyTorch for 3D detection which also provides downloadable weights of the trained models. We review the object detectors below. Noteworthy, mmdetection3d models are listed in the nuScenes rankings of object detectors [20], where models are evaluated using the nuScenes evaluation libraries at [43].

We present the object detectors together with an acronym to easily distinguish them in the rest of the paper. For each detector we distinguish the backbone, the neck, and the head.

FCOS [19] uses only the visual cameras. It is a monocular 3D object detector adapted from the 2D detector at [44]. The backbone is a pretrained ResNet101 [45] with deformable convolutions [46]. The neck is the Feature Pyramid Network (FPN, [47]), which generates a pyramid of feature maps, and it is an effective recognition system for detecting objects at different scales [47]. The head that produces final predictions (deciding on object class, location, etc.) is performed using an approach similar to RetinaNet [48], which applies shared heads to operate detection of multiple targets.

The other seven object detectors rely on the lidar's point-cloud and they are based on the Pointpillars [49] network. *Pointpillars* is well-known both for its speed and its accuracy. It exploits an encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D oriented bounding boxes for objects. The Pointpillars network consists of three main stages: i) a feature encoder network that converts a point cloud to a structured representation, namely a sparse pseudoimage; ii) a 2D convolutional backbone to process the pseudo-image into high-level representation, extracting the features map upon which the rest of the network is used; and iii) a detection head that detects and regresses 3D bounding boxes. We consider seven alternatives based on Pointpillars; essentially, they use the pillar-based method from [49] to convert the point cloud in a sparse pseudoimage, and differentiate from [49] by applying different backbones, and optionally changing the necks and heads.

FPN. The backbone used in this case is the Feature pyramid networks FPN [47] already described.

REG400. The backbone is the REGNETX-400MF model from [50]. This model is achieved from a methodology to design network design spaces, where a design space is a parametrized set of possible model architectures. With this approach, the resulting network model is elaborated automatically, starting from an initial, unconstrained design space which is progressively simplified. The RegNet400 under consideration uses a training regime of 400 MF (Million Flops, where flops mean multiply-adds [50]).

REG400SEC. With respect to *REG400*, it includes the neck SECOND [51]: the SECOND network allows reduced angle loss regression to improve the orientation estimation performance.

REG1.6. It uses the backbone REGNETX-1.6GF model from [50]. Its rationale is the same as REG400, but it considers a training regime of 1.6GF (Giga Flops) instead of 400 MF.

SEC. It uses the FPN [47] backbone, and the SECOND [51] neck, both already described.

SSN. In addition to the FPN backbone and the SECOND neck, it includes the shape-aware grouping heads from SSN [52], which explore the shape information from point clouds. This head consists of multiple branches for objects with similar shape and scale (objects with similar shape and scale share the same head e.g., buses and trucks).

SSNREG. With respect to the SSN case above, it uses the backbone REGNETX-400MF model from [50], previously discussed.

We ran all the models on the nuScenes validation set identified in [42], which consists of 150 frame sequences of 20 seconds each, and achieved the exact same results of their authors reported at [42]. This confirms that our setup of mmdetection3d is successful. The Average Precision for the detection of cars, which will be our reference in the rest of the paper, is reported in Table I. We can easily observe that, despite we use state of the art object detectors, the detection capability is still very far from perfect, especially when $l = 0.5$. However, it is well-known that object detection in complex scenarios is still prone to several misdetections.

D. Implementation of the criticality model in nuScenes

The implementation of our criticality model exploits the library nuScenes-dev [43], written in Python and available with open-source license. This library is the development kit of nuScenes, and it includes the source code to evaluate

TABLE I: Average Precision (AP) for the detection of cars. Values are ordered according to the $l = 0.5$ column.

Detector	$l = 0.5$	$l = 1$	$l = 2$	$l = 4$
FCOS	0.118	0.372	0.655	0.804
SEC	0.677	0.796	0.833	0.852
FPN	0.682	0.814	0.857	0.872
REG400	0.690	0.827	0.870	0.884
SSN	0.696	0.818	0.860	0.876
REG400SEC	0.713	0.825	0.863	0.875
SSNREG	0.717	0.835	0.872	0.886
REG1.6	0.722	0.837	0.874	0.889

object detectors. For example, the ranking of object detectors available at the nuScenes website [20] is computed using the code of this library (but on a different test set, whose ground truth information is not released to the public).

We extended the nuScenes-dev library, so that the metrics from our model are computed in addition to the usual metrics from nuScenes. The resulting library is available at [53]. Its usage is straightforward: it is sufficient to have a working installation of nuScenes-dev, and replace with the files in [53] the corresponding files of the nuScenes-dev installation. Then, the set of results will appear enriched with our metrics. Therefore, any object detector whose output is compatible with nuScenes-dev can be also evaluated using our library. The release at [53] includes a usage example, which allows repeating our experiments from the execution of the object detectors from mmdetection3d to the computation of results.

V. Experiments and Results

We execute the eight object detectors on the validation set previously described. In addition to AP, P, R, we compute AP_{crit}, $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for different values of D_{max}, R_{max}, T_{max}. More specifically, we consider several configurations $(D_{max}, R_{max}, T_{max})$, with $D_{max} \in \{5, 10, \ldots, 50\}$ meters, $R_{max} \in \{5, 10, \ldots, 50\}$ meters, and $T_{max} \in \{2, 4, \ldots 30\}$ seconds. Since distance is measured starting from the center of ego, a distance of 5 meters only includes vehicles that are very close to ego; also, we remind that 50 meters is the maximum distance from ego that is considered in the nuScenes object detection challenge, where objects farther than 50 meters from ego are ignored. Overall, this leads to 1500 configurations $(D_{max}, R_{max}, T_{max})$, repeated for the 8 object detectors. The entire computation was performed on a Dell Tower 7810, and lasted approximately one week.

A. Research goals

We present results by formulating and discussing the following three research goals.

G1: Investigate the ranking of the 8 object detectors when AP_{crit} is used. We will show that, depending on the triple $(D_{max}, R_{max}, T_{max})$, the ranking changes with respect to the one obtained with plain AP. This means that, when selecting the most suited object detector according to our considerations on safety and reliability, the choice may be different than just considering the AP ranking in Table I.

G2: Investigate AP_{crit} for different values of D_{max}, R_{max}, and T_{max}. This allows understanding possible trends from our set of configurations. We will observe that low values of R_{max} and T_{max} lead to the highest AP_{crit}, while the contribution of D_{max} is less relevant. Also, we will observe that the image-based object detector FCOS, which has an AP value much lower than the lidar-based object detectors, exhibits an AP_{crit} that is competitive for certain $(D_{max}, R_{max}, T_{max})$ configurations.

G3: Investigate the value of $P_{\mathcal{R}}$ at given $R_{\mathcal{S}}$ levels. Intuitively, this matches the question: "given a safety target on the detection, what is the possibility of driving the car with good mission reliability, i.e., without being forced to interrupt the driving continuously because of false positives?". First, we will observe that the plain AP metrics from Table I are inadequate to depict such situation. Then, we will show that the introduction of $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ opens to new, different conclusions, under specified restrictions defined by the triples $(D_{max}, R_{max}, T_{max})$. We will show that, under specific configurations, object detectors are much more suited for detecting relevant objects that may interfere with the ego vehicle than initially expected.

B. G1: Ranking of object detectors

We calculate the rankings of detectors based on AP_{crit} for all the 1500 configurations $(D_{max}, R_{max}, T_{max})$. Many of them produced a different ranking with respect to the one based on AP (i.e., the one in Table I).

Table II reports the number of configurations in which the ranking according to AP_{crit} differs from the one with AP. Table II also reports the number of positions that changed, and the maximum changes in position for a single object detector. For example, with $l = 0.5$ we have that 567 out of 1500 rankings do not match the AP ranking. For each of these 567 rankings, the difference with respect to Table I is 2 or 4 positions, and the positions that change are from the first one to the seventh one. Last, the maximum change is of 1 position: this means that, with $l = 0.5$, we have at most the inversion of object detectors that are adjacent in Table I. For higher values of l, the number of rankings inverted, the number of position changed, and the number of changes raise significantly. In general, the whole set of object detectors may change position with respect to Table I, with the except of the detector in the 8^{th} position which is always FCOS.

C. G2: on AP_{crit} and $[D_{max}, R_{max}, T_{max}]$ configurations

We argue that higher values of AP_{crit} are achieved with low values of R_{max} and T_{max}. Intuitively, low R_{max} and T_{max} reduce the number of vehicles to be considered in our

TABLE II: Details on the rankings that differ from Table I.

	$l = 0.5$	$l = 1$	$l = 2$	$l = 4$
Configurations with changes	567	1256	1064	1425
Number of changes per configuration	2 or 4	2–6	2–6	2–6
Positions with changes	1–7	1–6	1–6	1–7
Max position changes per detector	1	2	3	3

TABLE III: Maximum AP_{crit}, for all configurations $[D_{max}, R_{max}, T_{max}]$, ordered by the $l = 0.5$ column.

	$l = 0.5$	$l = 1.0$	$l = 2.0$	$l = 4.0$
FCOS	<u>0.335</u>	<u>0.661</u>	0.851	0.911
SEC	0.844	0.918	0.933	0.942
FPN	0.847	0.924	0.943	**0.948**
REG400	0.852	0.930	**0.947**	0.952
REG400SEC	0.860	0.927	0.940	0.949
SSN	0.862	0.927	0.945	0.950
SSNREG	0.871	0.932	0.944	0.953
REG1.6	0.875	0.937	0.950	0.957

analysis: only those that are really relevant for the detection are included. To confirm this, we analyze the maximum AP_{crit} for the eight object detectors and the different l values, with the help of Table III.

First, We observe how FCOS reduces its distance to the other detectors with respect to Table I, especially with $l = 2.0$ and $l = 4.0$. This is a relevant result, because FCOS, which exploits the camera instead of the lidar, was considered by far less performing than the other detectors in Table I, while it can be considered almost on par with the other detectors in this case.

Next, we observe that the maximum AP_{crit} values of Table III are obtained with the triple $(D_{max}, R_{max}, T_{max}) = (25, 5, 2)$, with just some exceptions: the two occurrences in bold, which are obtained with $(30, 5, 2)$, and the two underlined occurrences, which are obtained with $(20, 5, 2)$. This does not mean that the configuration $(25, 5, 2)$ is the "best" one: in fact, this configuration sets strong constraints on the objects that are interesting for detection. We believe the entire set of configurations needs to be studied, and the choice of the best suited configuration should be based on considerations on the AP_{crit} measured at $(D_{max}, R_{max}, T_{max})$, the target system, and its expected usage.

To explore trends of AP_{crit}, we select a representative example: we pick the object detector ranked first in Table III, which is REG1.6 with $l = 4.0$. In Figure 2 we show the AP_{crit} values of REG1.6 when $D_{max} = 25$, and for different R_{max}, T_{max}. In the lower part of the z axis, the corresponding $AP = 0.889$ is plotted for reference; it is not affected by D_{max}, R_{max} and T_{max}, so it is represented as a horizontal surface. The figure shows that the AP_{crit} is higher than AP under the considered configurations. In fact, setting $D_{max} = 25$ reduces the impact of objects farther than 25 meters, which possibly are a significant contribution to misdetections.

In Figure 3 instead we show the AP_{crit} when $R_{max} = 20$; the figure clearly shows how the highest AP_{crit} values are achieved when D_{max} is set in the range $[20, 30]$. This is possibly due to the fact that setting D_{max} very low excludes a lot of "easy" (i.e., close) objects from the relevant ones, thus deteriorating AP_{crit}. Conversely, when D_{max} becomes much greater than R_{max}, a lot of distant but not relevant objects are included, which are unlikely to reach a collision point closer than R_{max}. For the other object detectors the trends are similar.

We now confirm that AP_{crit} is in general higher than AP. In rare cases, AP is slightly above AP_{crit}; in our experiments it is at worst $\max(AP - AP_{crit}) = 0.015$ greater than AP, using SSN, $l = 2$, and $(D_{max}, R_{max}, T_{max}) = (5, 50, 30)$. This is expected, because the AP_{crit} gives less weight to many vehicles that are harder to detect, e.g., those at a farther distance from ego. In fact: i) $D_{max} = 5$ rewards only the detection of vehicles that are exceedingly close, and ignore the others, and ii) $R_{max} = 50$, $T_{max} = 30$ exclude the peculiarities of this model, because they are large values that introduce small or no AP_{crit} penalizations on most of the vehicle with respect to the AP case.

To provide a summarizing view, in Figure 4 we show a 4D plot of SSNREG with $l = 1$: configurations $(D_{max}, R_{max}, T_{max})$ producing higher values of AP_{crit} are

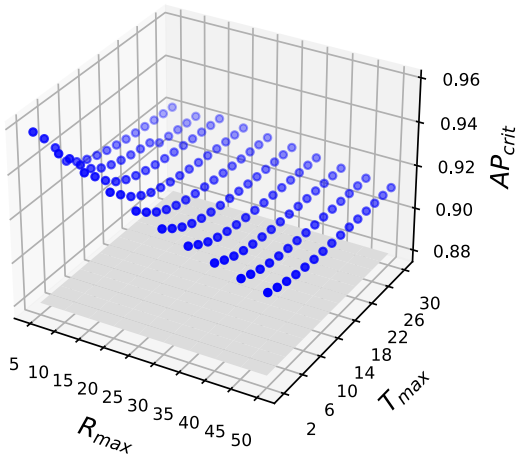


Fig. 2: AP_{crit} measured on REG1.6 with $l = 4.0$ and $D_{max} = 25$, for the different R_{max} and T_{max}.

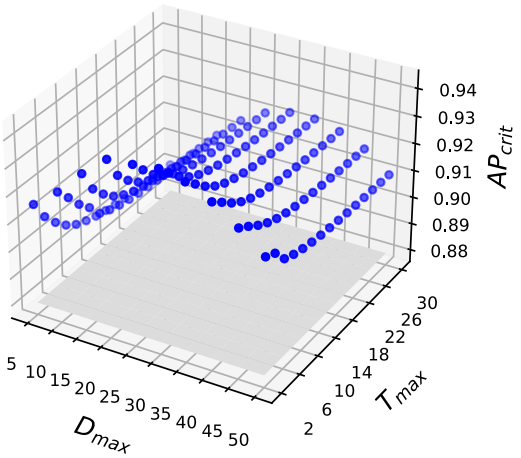


Fig. 3: AP_{crit} measured on REG1.6 with $l = 4.0$ and $R_{max} = 20$, for the different D_{max} and T_{max}.

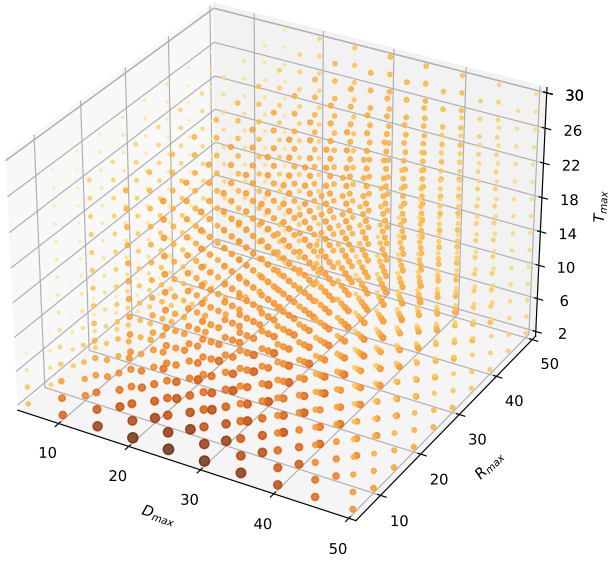
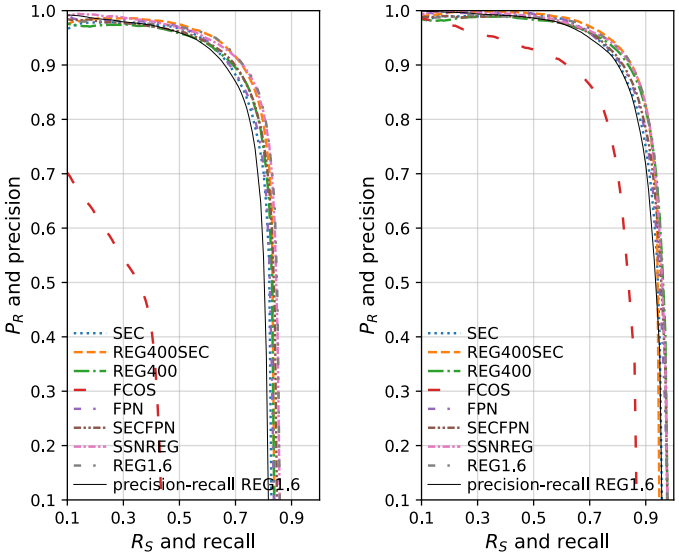


Fig. 4: AP_{crit} for SSNREG with $l = 1$ and different configurations of $(D_{max}, R_{max}, T_{max})$. The darker and larger the dots, the higher the AP_{crit}. *(best viewed in color)*



(a) $l = 0.5$ (b) $l = 2$

Fig. 5: Relations between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for all object detectors with $l \in \{0.5, 2\}$ and configuration $(20, 15, 8)$. Precision-recall curve is drawn for REG1.6.

represented with larger and darker dots. Figure 4 confirms the previous consideration on the evolution of AP_{crit} values: the highest AP_{crit} values are obtained with low values of R_{max} and T_{max}, and with D_{max} in the interval $[20, 30]$. A similar trend is observed for the other object detectors and l values.

D. G3: on the value of $P_{\mathcal{R}}$ at given $R_{\mathcal{S}}$ levels.

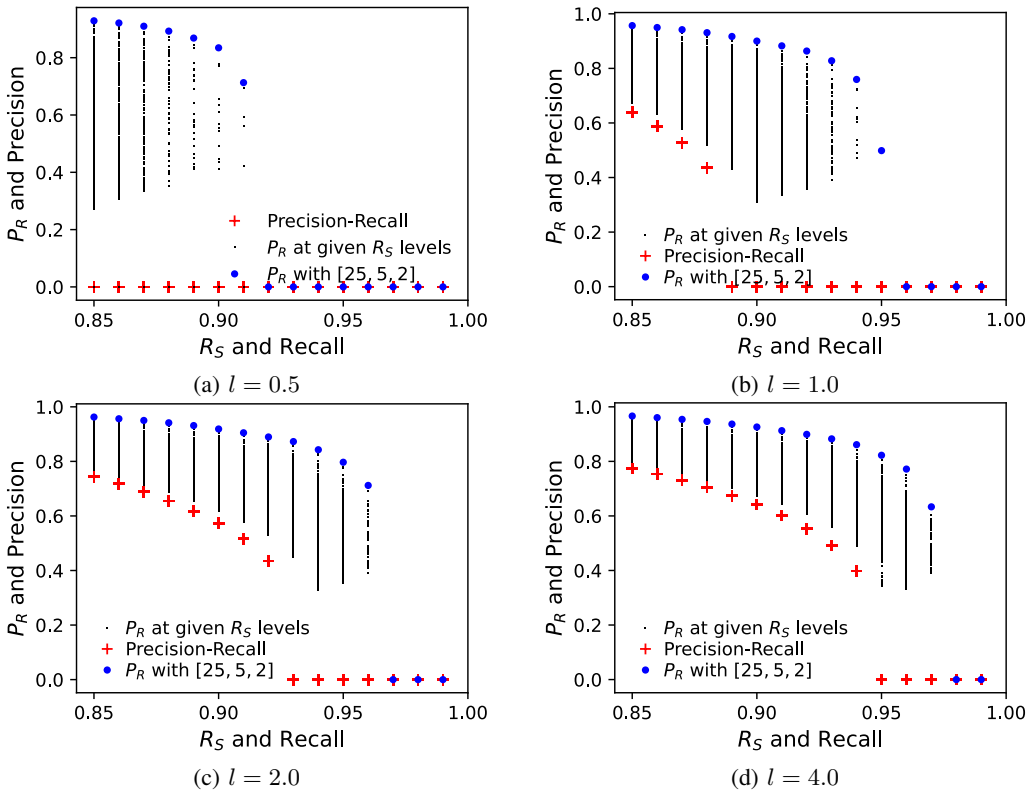
To discuss the relations between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$, we use the same approach of the traditional precision-recall curve. In Figure 5, we plot the eight object detectors for $(D_{max}, R_{max}, T_{max}) = (20, 15, 8)$ with $l = 0.5$ (left) and $l = 2$ (right). We select such triple as a representative configuration because, intuitively, it is one of the settings that may have practical use: it includes objects that are in the close surroundings of ego while it is roving around the city. We also plot the precision-recall curve for REG1.6, which is the best object detector when traditional P and R are considered. The trends are broadly similar for other triples $(D_{max}, R_{max}, T_{max})$ and l values, and match the expected behavior of a precision-recall curve. In agreement with [9], cases in which recall or precision is less than 0.1 are removed; this avoids showing the noise commonly seen in low precision and low recall regions.

We notice that the higher the l, the higher the value of $P_{\mathcal{R}}$ given a target $R_{\mathcal{S}}$. This can be observed comparing the difference between Figure 5a and Figure 5b, and it is clearly visible for FCOS. This is expected, because with a higher l value, the number of bounding boxes that are considered correct detections is increased.

We carefully investigate the relations between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for high values of $R_{\mathcal{S}}$ (safety-weighted recall), which are of particular interest in the reference domain of this work. This way we can study the $P_{\mathcal{R}}$ that we achieve when safety is enforced thanks to a high $R_{\mathcal{S}}$. This allows answering the question defined for G3 in Section V-A, on the ability of safely driving the car while maintaining acceptable mission reliability. For example, the safest condition is $R_S = 1$, but $P_{\mathcal{R}}$ is typically 0 in such cases. Still, a very high $R_{\mathcal{S}}$ is necessary to enforce safety of the detection. We elaborate this with the aid of Figure 6, where we use SSNREG with $l \in \{0.5, 1, 2, 4\}$. We compute $R_{\mathcal{S}}$ and R at steps of 0.01, starting from 0.85. Red crosses represent precision-recall pairs (P, R) from the traditional precision-recall curve. The black dots represent pairs $(P_{\mathcal{R}}, R_{\mathcal{S}})$; these are computed for each configuration $(D_{max}, R_{max}, T_{max})$, thus yielding 1500 black dots for each $R_{\mathcal{S}}$ value. The large blue dots on top of each plot are the $(P_{\mathcal{R}}, R_{\mathcal{S}})$ values achieved using SSNREG with the configuration leading to the highest AP_{crit}, which is $(25, 5, 2)$.

For traditional metrics, we can observe that increasing the recall R, the precision P quickly drops to 0, and it is always zero in the case of Figure 6a where $l = 0.5$. This means that SSNREG can offer a high recall i.e., high ability of detecting all the objects, only at the cost of many false positives: this is clearly of little or no use in practice. Instead, if we restrict the scope of the object detector thanks to our model, we reach different conclusions. For example, consider again the case $l = 0.5$ (Figure 6a). Even with $R_{\mathcal{S}} \geq 0.9$, there are some configurations in which $P_{\mathcal{R}} > 0.8$, which is clearly a much more comforting result, showing confidence in the detection at least to some extent. In other words, our conclusion on SSNREG can be very different from those we achieve using P and R, when we reduce the bounding boxes to be examined (both the predicted and ground truth ones) to match the criteria of $R_{\mathcal{S}}$ and $P_{\mathcal{R}}$.

Clearly, this analysis is not meant to prove that the evaluated object detectors are safe and reliable. Rather, it shows how our



(a) $l = 0.5$ (b) $l = 1.0$

(c) $l = 2.0$ (d) $l = 4.0$

Fig. 6: $P_{\mathcal{R}}$, $R_{\mathcal{S}}$, P and R for SSNREG when $R_S \geq 0.85$ and $R \geq 0.85$, for $l \in \{0.5, 1, 2, 4\}$.

model allow establishing sound parameters that can be used to build, assess and tune object detectors for their application in safety critical domains, in a way much more useful than the widely used precision and recall. We remark that object detection in complex scenarios is still an open research topic that makes improvements every year [54], with new detectors that are proposed continuously; however, defining new object detectors is beyond the scope of this paper.

VI. Conclusions and Future Works

We argued that metrics for object detection do not match the demands and peculiarities of a safety-critical system. Within the autonomous driving domain, we show that the state-of-the-art evaluation of object detectors do not consider the possible role of the objects in a specific scene, and in particular with respect to the driving task of the vehicle performing the detection. In other words, the metrics typically used to rate object detectors describe how good an object detector is at detecting *all* the objects on the scene, while instead, for the purpose of an autonomous driving system, we are interested in detecting all the objects that *will likely interfere* with the driving task of the vehicle.

To cover this gap, in this paper we propose novel metrics for object detection that take into account the concepts of safety and reliability. We build and exercise an object criticality model that performs a rating of the objects, based on the distance from the subject vehicle, the possible colliding trajectory, and the expected time to collision (based on relative velocity). We reward object detectors based on their ability to correctly detect objects that have the possibility to interfere with the vehicle, and whose presence requires proper response from the autonomous vehicle.

The model is exercised on eight object detectors. Amongst the main results, we show that our judgement on the performance of object detectors may be very different when we consider the detection of i) everything on the scene (as it is usually done), or ii) only the relevant items. Depending on which of the two cases is of interest, we may end up choosing different object detectors. Further, we also show that object detectors that have bad performance under case i) can instead turn competitive in case ii) under specific configurations.

Future works will focus on considering alternatives to the metric model in Section III. Although the model in this paper is fully applicable, it is also obvious that more complex models could be elaborated, for example to represent the likelihood that cars will steer or change speed. With the data available on nuScenes, this is doable because trajectories of all vehicles are traced, and they can be introduced in the model only at the cost of a more complicated formulation to compute the intersection points. Further, we ignored possible vertical offsets of objects, given the characteristics of the dataset available; in our future works, we plan to also address these in the model so that it can be used for datasets with vehicles on road slopes.

Acknowledgment

Removed for double blind review.

REFERENCES

[1] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[2] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[3] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[4] C. Premebida, G. Melotti, and A. Asvadi, "Rgb-d object classification for autonomous driving perception," in *RGB-D Image Analysis and Processing*. Springer, 2019, pp. 377–395.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[6] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," *arXiv preprint arXiv:2104.10133*, 2021.

[9] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[10] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," *arXiv preprint arXiv:2006.14480*, 2020.

[11] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[12] R. Padilla, S. L. Netto, and E. A. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020, pp. 237–242.

[13] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. mcgraw-hill, 1983.

[14] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[15] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[17] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Science and Information Conference*. Springer, 2019, pp. 128–144.

[18] M. Miškuf and I. Zolotová, "Comparison between multi-class classifiers and deep learning with focus on industry 4.0," in *2016 Cybernetics & Informatics (K&I)*. IEEE, 2016, pp. 1–5.

[19] T. Wang, X. Zhu, J. Pang, and D. Lin, "FCOS3D: Fully convolutional one-stage monocular 3d object detection," *arXiv preprint arXiv:2104.10956*, 2021.

[20] "nuscenes web-site, https://www.nuscenes.org/."

[21] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of outlier detection: Measures, datasets, and an empirical study continued," *Lernen, Wissen, Daten, Analysen 2016*, 2016.

[22] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[23] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big data*, vol. 5, no. 3, pp. 246–255, 2017.

[24] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[25] M. Gharib, T. Zoppi, and A. Bondavalli, "Understanding the properness of incorporating machine learning algorithms in safety-critical systems," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 232–234.

[26] "Kitti vision benchmark suite, http://www.cvlibs.net/datasets/kitti/."

[27] "Cityscapes dataset, https://www.cityscapes-dataset.com/."

[28] "Waymo open dataset, https://www.waymo.com/open."

[29] C.-H. Cheng, "Safety-aware hardening of 3d object detection neural network systems," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2020, pp. 213–227.

[30] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3266–3273.

[31] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020.

[32] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *arXiv preprint arXiv:1703.04977*, 2017.

[34] P. Koopman, U. Ferrell, F. Fratrik, and M. Wagner, "A safety standard approach for fully autonomous vehicles," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2019, pp. 326–332.

[35] ISO, "Iso/pas 21448-road vehicles-safety of the intended functionality," *International Organization for Standardization*, 2019.

[36] I. V. T. Society, "IEEE P2846 - IEEE Draft Standard for Assumptions for Models in Safety-Related Automated Vehicle Behavior," 2020.

[37] S. Global, "UL4600 -Standard for Evaluation of Autonomous Products, Edition 1," 2020.

[38] S. International, "J3237 (Work In Progress)- Operational Safety Metrics for Verification and Validation (VV) of Automated Driving Systems (ADS) ," 2021.

[39] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice Hall, 1982.

[40] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[41] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[42] M. Contributors, "Mmdetection3d: Open-mmlab next-generation platform for general 3d object detection," 2020.

[43] "nuscenes-dev libraries, https://github.com/nutonomy/nuscenes-devkit."

[44] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. Int. Conf. Computer Vision (ICCV)*, 2019.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[46] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[47] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[48] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[49] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[50] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[51] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[52] X. Zhu, Y. Ma, T. Wang, Y. Xu, J. Shi, and D. Lin, "Ssn: Shape signature networks for multi-class object detection from point clouds," *arXiv preprint arXiv:2004.02774*, 2020.

[53] "Modified nuscenes-dev library used in this paper, https://anonymous.4open.science/r/DSN2021-metrics_model-1CDB/."

[54] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.