Safety Score: A Quantitative Approach to Guiding Safety-Aware Autonomous Vehicle Computing System Design

Hengyu Zhao, Yubo Zhang†, Pingfan Meng†, Hui Shi, Li Erran Li†, Tiancheng Lou†, Jishen Zhao
University of California, San Diego †Pony.ai
{*h6zhao, hshi, jzhao*}*@ucsd.edu*
†{*yubo, pmeng, erranlli, acrush*}*@pony.ai*

***Abstract*— High automated vehicles rely on the computing system in the car to understand the environment and make driving decisions. Therefore, computing system design is essential for ensuring the driving safety. However, to our knowledge, no clear guideline exists so far regarding how to guide the safety-aware autonomous vehicle (AV) computing system design. To understand the safety requirement of AV computing system, we performed a field study by operating industrial Level-4 AV fleets in multiple locations for three months. The field study indicates that traditional computing system performance metrics, such as tail latency, average latency, maximum latency, and timeout, cannot fully satisfy the safety requirement for AV computing system design. To address this issue, we propose the "safety score" as a primary metric for measuring the level of safety in AV computing system design.**

I. Introduction

The safety of autonomous vehicle (AV) has been given more attention, because AVs should be ensured the safe operation before released to the public. As a consequence, "A Vision for Safety" [6] is recently released by the U.S. Department of Transportation, which offers a path forward for the safe deployment of AVs and encourage ideas that deliver safer autonomous driving. Therefore, in current AV design, one of the most critical requirements is safety.

The AV computing system is one of the most critical components to ensure safety, because high automated vehicles depend on the computing system for driving decision making (§ III). The AV safety can be classified into two groups: functional safety and nominal safety (§ III-A). The functional safety is standardized by ISO 26262 [5]. The nominal safety has been studied by some previous works [15], [17], but these works only ensure that AVs can make safe and appropriate driving decisions without considering the latency consumed by computing systems. However, it is also essential for AV computing systems to make timely decisions. To the best of our knowledge, this paper is the first work that proposes a safety metric to guide safety-aware computing system design, in terms of timely decision making.

To explore nominal safety implications on the computing system design, we conducted a field study (§ IV) by running real industrial Level-4 AV fleets in various locations and traffic patterns. Based on the knowledge learned from extensive data collected during our field study, we identify that traditional computing system metrics, e.g., tail latency, average latency, maximum latency, and timeout, do not accurately reflect the safety requirement of AV computing system design (§ IV-B). Instead, we observe that the level of AV safety is a non-linear function of accumulative instantaneous computing system response time (§ IV-A) and various external factors – including the AV's velocity, physical properties (e.g., braking distance), and road condition.

To guide safety-aware AV computing system design, we propose "safety score", a safety metric that measures the level of AV safety based on computing system response time and the aforementioned external factors in a non-linear format (§ V). Our safety score is rigorously derived from a published industrial formal AV safety model, the Responsibility-Sensitive Safety (RSS) model [17]. The RSS model focuses on general rules for an AV to keep a safe distance with surrounding vehicles. Our safety score bridges such a conceptual model to AV computing system design with a quantitative approach. We hope the proposed safety score will be an essential contribution to the existing computer system metric portfolio. In summary, this paper makes the following contributions:

- We perform a field study with our AV fleets. We present observations and safety-aware computing system design challenges based on the extensive data collected from our field study.
- We propose the safety score, a metric that evaluates the level of safety in AV computing system design. By analyzing the safety score, we present a set of implications on safety-aware AV system design.
- We present the detailed safety score formulation steps. Our safety score is strictly derived from the RSS model.
- We compare our safety score with other latency metrics, and demonstrate that the regular latency metrics are lack of accuracy for evaluating AV safety.

II. Related Work

To our knowledge, this is the first paper to propose a safety metric to guide safety-aware AV computing system

and architecture design, based on a large-scale field study of real industry Level-4 AVs. In this section, we discuss related works on AV nominal safety policies and AV computing system design.

AV nominal safety policies. Two recent studies, RSS [17] and SFF [16], investigated AV nominal safety policies. Both studies define the high-level concept of AV nominal safety. Unfortunately, neither of the studies investigates safety requirement on computing systems and architecture design. SFF [16] focuses on policies that ensure safe driving decisions, e.g., braking, acceleration, and steering, under given limitations. RSS [17] develops a set of general rules to ensure collision avoidance, by enforcing that the AV keeps a safe distance with surrounding vehicles. Our study focuses on the nominal safety requirements of AV computing systems and architecture design.

AV computing system design. Most previous studies on AV system and architecture designs focus on accelerator design and energy efficiency improvement of deep learning or computer vision software execution [13], [14]. Their performance and energy optimizations are guided by standard performance metrics. As our evaluation demonstrates, safety metric guided computing system design leads to different architecture and systems compared with traditional metrics guided designs. Moreover, our metric can also be used to guide deep learning and computer vision accelerator design in AV systems.

III. State-of-the-art AV System

High automated cars are under development and testing in both industry [1], [3], [4], [11] and academia [7], [9], [10]. To understand the current status of AV development, we examine the state-of-the-art AV design, including AV safety, and hardware and software organizations.

A. AV Safety

In automotive industry, safety requirements are classified into functional safety and nominal safety [17]. Functional safety refers to the integrity of operation of vehicle's electrical system, including hardware and software. For example, a hardware failure or a bug in software will both lead to a functional safety hazard. Functional safety is standardized by ISO 26262 [5], which defines various automotive safety integrity levels that offer (a) failure-in-time targets for hardware and (b) systematic processes for software development and testing that conform with appropriate systems engineering practices.

However, unlike in a conventional vehicle, the computing system plays an integral role in an AV. Even functionally safe, AVs can still crash due to untimely or unsafe driving decision. As Riccardo Mariani, Intel Fellow and chief functional safety technologist in the internet of things group, pointed out, current ISO standard of functional safety is inadequate to ensure the safety of AV [2]. This falls in the domain of nominal safety. Nominal safety refers to whether the AV makes timely and safe driving decisions, assuming that the hardware and software are operating error free (i.e., functionally safe).

This paper focuses on nominal safety of AV. While the software module algorithm design determines whether a driving decision (e.g., braking, acceleration, or steering) is safe, timely driving decision making heavily relies on the performance of computing system and architecture design. Two recent studies by Mobileye [17] and NVIDIA [16] defined formal nominal safety rules for AV. However, these studies focus on how to make planning and control decisions to keep a safe distance with surrounding vehicles. To our knowledge, no previous research investigates how to ensure timely driving decision making with safety-aware computing systems design. As such, this paper focuses on exploring safety-aware computing system design methodology that achieves timely driving decision making, by mapping formal safety requirements onto AV computing system design.

Note that it is impossible to guarantee that an AV will never be involved in an accident [17], because it is difficult to predict or control the activity of other vehicles [16]. Therefore, we intend to guarantee that the AV will be sufficiently careful to avoid becoming a cause of an accident, the same as recent AV safety studies [17].

B. AV Computing System Architecture

Low-end processors may be sufficient for lower level AVs, such as Level-2 and Level-3 AVs. However, current Level-4 AV computing systems always employ server-grade heterogeneous architectures, which include high-end CPUs, GPUs, storage devices (e.g., terabytes of SSDs), and domain specific accelerators. To meet the intensive computation requirements, our AV fleets adopt Intel Xeon CPUs and NVIDIA Titan V GPUs.

Implication: Current level-4 AV computing systems adopt server-grade processors. However, it is still difficult to complete all necessary computation tasks within a limited period, because of the considerable interdependently executing software modules (discussed in § III-C) and the limited hardware resources in the computing system. Therefore, it is challenging to effectively utilize the limited hardware resources to meet the given safety requirement.

C. Software Modules in AV Computing Systems

AV computing systems are composed by multiple functional modules, such as localization, perception and planning and control. According to our field study, perception is the most time consuming module. Moreover, as the perception heavily relies on LiDAR, and LiDAR perception consumes the most latency than other perception modules (e.g., cameras, radars), we present more details of LiDAR perception.

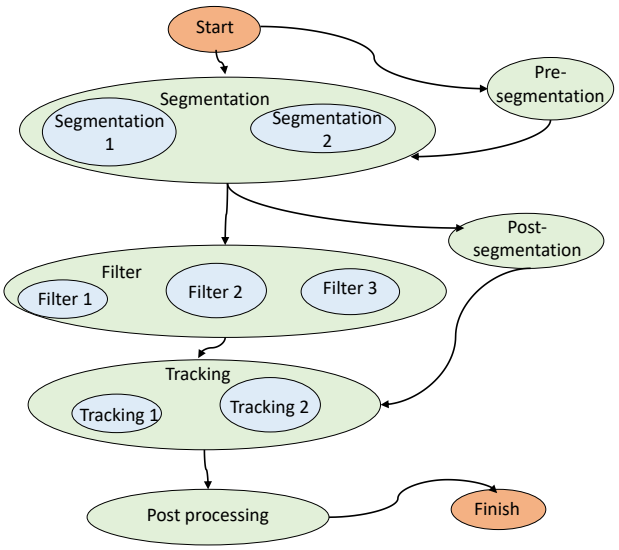


Fig. 1. Dependency graph of LiDAR perception.

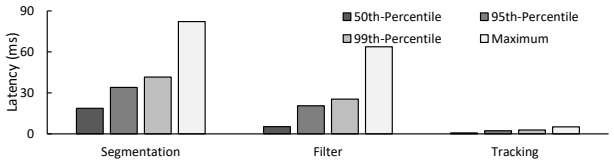


Fig. 2. Latency of LiDAR perception main modules.

Dependency graph. To process LiDAR data, the computing system executes a considerable number of inter-dependent software modules (i.e., software programs) at real-time. To understand the dependency relationship among these software modules, we build a dependency graph based on our AV system. Figure 1 shows an example of several typical modules in LiDAR perception. LiDAR perception consists of a set of main modules (e.g., segmentation, filter, tracking) and sub-modules (e.g., segmentation 1, filter 1, tracking 1). For example, *Segmentation* extracts useful semantic and instance information (e.g., surrounding vehicles and pedestrians) from the background. The *Filter* modules remove the noise in the point clouds. Some software modules, e.g., *Segmentation*, do not only impose long latency, but also have a strong dependency with other software modules.

Implication: Due to their long latency and high dependency, certain software modules contribute more to the perception latency and overall computing system response time than other modules. Therefore, these modules are typically more safety critical.

IV. Field Study and Observations

To explore realistic AV safety requirement and computing system hardware/software behaviors, we run a fleet of Level-4 AVs in various locations over three contiguous months. We collected two hundreds hours and two thousands miles of driving data. The data is collected at the smallest granularity (one sensor frame), that shows how the computing system interprets and responds to surrounding environments. For each frame, we record (1) the latency of each module, (2) total response time of the whole computing system, (3) the utilization of hardware resources such as CPUs, GPUs, and storage devices (4) velocity and acceleration of the AV.

A. Observations

Safety-critical software module analysis. We further analyze the execution latency of the main modules. Figure 2 shows an example with LiDAR perception modules by illustrating maximum, and tail latency of three representative software modules across thousands of executions during our field study. We have two critical observations. First, maximum, and tail latency yields different criticality rankings on the modules. For example, specific modules with high maximum latency (e.g., *Filter*) have low 50th-percentile tail latency. We observe that maximum latency is typically achieved in safety-challenging scenarios, such as heavy traffic and near-accident situations, e.g., Figure 3. Therefore, maximum latency better indicates safety-criticality of a software module than other formats of latency. Second, we observe that safety-critical modules also typically have strict inter-dependency with other modules in the dependency graph. Examples include *Segmentation* and *Filter*. Based on the two observations, safety-critical modules can be identified as those having long maximum latency and strict inter-dependency with other modules.

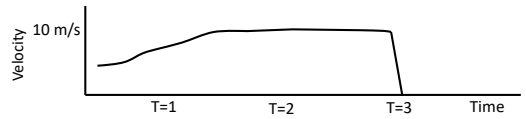
Latency accumulation effect. We observe a latency accumulation effect of certain software modules. Figure 4 illustrates an example of LiDAR perception modules: once LiDAR perception latency exceeds 100ms, the system response time increases at a much higher rate (in the region with a steeper slope in Figure 4) with the increase of LiDAR perception latency. The reason is two-fold. First, because LiDAR has a fixed sampling rate (with a 100ms interval in our AV), the perception, if not completed within the sampling interval, will further delay the processing of subsequent frames. Second, as software modules are inter-dependent, a delay of one software module will further delay its dependent software modules. The slopes and thresholds of the accumulation effect vary across different software modules, depending on individual characteristics and the inter-dependency relationship with other modules.

Impact of other external factors. Besides traffic, several other external factors also have a critical impact on the safety requirement of AV system design. These factors include the acceleration and velocity of the AV and other vehicles around it, AV's physical properties (e.g., braking distance), and road condition. Taking the scenario in Figure 3 as an example, if our AV had a higher velocity

(a) T=1, (Left: LiDAR point cloud graph, Right: camera graph)

(b) T=2, (Left: LiDAR point cloud graph, Right: camera graph)

(c) T=3, (Left: LiDAR point cloud graph, Right: camera graph)



(d) Velocity-time graph.

Fig. 3. An emergency hardbrake case that we encountered.

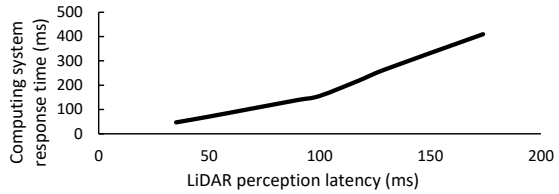


Fig. 4. The relationship between LiDAR perception latency and computing system response time.

or longer braking distance, the collision would also have happened.

B. Safety-aware Design Challenges

As discussed in § III-A, this paper focuses on nominal safety that ensures timely driving decision making. It seems that timely driving decision making can be achieved by minimizing the computing system response time. However, this is challenging. To ensure achievable minimum response time within finite computing system hardware resources, performance optimization needs to be guided by specific metrics. Based on our field study, traditional computing system performance metrics, such as average latency, maximum latency, tail latency, and timeouts, do not accurately reflect the safety requirement of AV systems due to the following reasons.

First, AV safety is determined by instantaneous response time (Figure 3), instead of statistical patterns adopted in traditional performance metrics. Second, it is challenging to predict the response times due to the complex inter-dependent relationships between software modules and the latency accumulation effect. Third, AV safety is also determined by various external factors as discussed in our field study observations. As a result, the level of AV safety is not a simple weighted sum of the elements, but a non-linear function of response time as we identified in our study (§ V). Finally, although timeout is widely used to measure real-time system performance [18], it is impractical to simply use timeout to evaluate the safety of AV system design: it is difficult to determine the required threshold to calculate the timeout of AV system response time, because each software module has a different threshold (e.g., different sampling interval with various sensors); External factors further lead to dynamically changing thresholds. Therefore, we need to rethink the fundamental metric used to guide safety-aware AV system design.

V. Safety Score

In this section, we propose the safety score, a metric that measures the level of AV safety based on computing system response time and external factors to guide safety-aware computing system design.

A. Safety Score Description

Our safety score is rigorously derived from the published industrial formal AV safety model – Responsibility-Sensitive Safety (RSS) [17]. The RSS model is developed by Mobileye, an Intel subsidiary that develops advanced driver-assistance systems (ADAS) to provide warnings for collision prevention and mitigation. The RSS model defines safe distances between the AV and nearby objects, and its formal mathematical models identify emergency moments, where the AV needs to perform proper planning and control decisions when the safe distances are compromised. The RSS model has been endorsed by various AV projects, such as Valeo [12] and Baidu's Apollo Program [8].

Unfortunately, the RSS model does not provide many insights on safety-aware computing system design. To give a quantitative guideline for safety-aware computing system design, we map the RSS model to the safety score based on our field study observations. The safety score is defined as the following equation (check Section VI for precise mathematical formulation of safety score):

$$\text{Safety Score} = \begin{cases} \sigma[\alpha(\theta^2 - t^2) + \beta(\theta - t)], & if\ t < \theta \\ \eta[\alpha(\theta^2 - t^2) + \beta(\theta - t)], & elsewhere \end{cases} \quad (1)$$

The variables in the equation are described in Table I. Here, t is the "instantaneous" computing system response time, which we define as the response time to one frame of sensor data (or several frames depending on the perception granularity of an AV implementation). We calculate t by the following equation:

TABLE I

VARIABLES IN SAFETY SCORE.

Variable	Description
t	Instantaneous computing system response time, defined by Equation 2.
θ	Response time window.
α, β	Indicating the velocity and acceleration of an AV and surrounding vehicles, defined by Equation 4 in Section VI.
σ, η	Reward or penalty on the level of safety, when t is lower or higher than θ, respectively.

$$t = \sum_{i=1}^{N} w_i(t_i) \tag{2}$$

Here, t_i is the instantaneous latency of a safety-critical module on the critical path; N is the total number of such modules. Safety critical modules are determined based on the inter-dependency relationship in the dependency graph and the maximum latency via the extensive pre-product road tests (§ IV). $w_i()$ is an accumulation effect function, which reflects the contribution of t_i to system response time t. For each safe-critical module, we obtain a latency accumulation effect curve that can be described by a function $w_i()$. Figure 4 shows an example of such curves, where the x-axis is t_i and y-axis is the value of $w_i(t_i)$. But curve shapes vary across different modules.

The rest variables in Equation 1 reflect the impact of external factors. The response time window θ is the maximum time that the AV would take to avoid hitting another vehicle that is certain distance d away, if both vehicles keep moving at the same direction, velocity, and acceleration, i.e., neither of vehicles perform any response to a reducing distance. We allow AV developers to define the distance d in various manners, e.g., as (1) a minimum safe distance allowed by traffic rules, which is adopted by our experiments as an example to make our evaluation more concrete or (2) a user-defined confident safe distance. Figure 6 in Section VI provides a more detailed analysis of θ.

Variables α and β reflect the velocity and acceleration of the AV and the surrounding vehicle, calculated by Equation 4 in Section VI. Safety reward σ and penalty η are user-defined coefficients, which indicate the user's confidence of safety. We allow users to adjust the reward and penalty based on their own road test experiences, such as the velocity of the vehicles in the traffic that the AV is likely to run into, road condition, and AV braking distance. We discuss more detailed implications of the reward and penalty in § V-B. Section VI provides more detailed discussion of these variables.

No clear boundary is defined between safe and unsafe in various AV safety models [16], [17]. As such, the safety score does not intend to give a threshold of safe versus unsafe either. Instead, it implies the level of safety – higher safety score (could be either positive or negative) indicates safer AV system design.

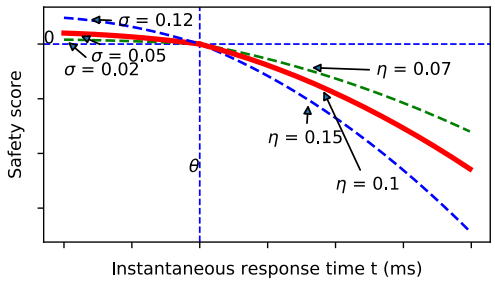


Fig. 5. The non-linear relationship between safety score and instantaneous response time.

B. Implications to Computing System Design

Based on our practical experience, safety score provides the following key implications on the safety-aware AV system design.

Nonlinearity. The safety score is a non-linear function of instantaneous response time t. Safety-aware computing system design approach highly relies on the shape of the curve. Figure 5 lists a few examples of safety score curves as a function of t. With a given set of external factors, the shape of the curve is determined by response time window θ, safety reward σ, and safety penalty η. Based on our road tests, the majority of safety emergency scenarios happen when $t > \theta$. When $t < \theta$, further optimizing the instantaneous response time does not significantly improve the level of safety. Therefore, in practice the shape of the safety score curve appears like the bold line (σ=0.05, η=0.1) shown in Figure 5, with a lower reward than penalty. As such, computing system design needs to prioritize reducing the instantaneous response time t, when $t > \theta$.

Instantaneous latency. As we discussed before, the safety score is computed by instantaneous response time per sensor frame. Therefore, it is infeasible to adopt traditional performance metrics calculated based on statistical performance behaviors, such as tail, maximum and minimum latency.

Safety impact across various software modules. Due to the complex inter-dependency relationship and latency accumulation effect of the software modules, the instantaneous latency of different software module has different impact on safety score. This is modeled by Equation 2, where each software module has a disparate form of the contribution to the instantaneous AV system response time.

VI. SAFETY SCORE FORMULATION

In this section, we show the safety score formulation steps. The safety score is derived from the Responsibility-Sensitive Safety (RSS) model [17], which formalizes safety as assuring five common-sense rules, and any AV driving policy should obey them. Specifically, the policy

will choose safe actions under various circumstances. The action is safe, if (1) given the current driving state and the action and (2) when the surrounding vehicles are acting adversely but reasonably, a collision can be avoided.

More formally, a given driving state $\{v, a_{max,+}, a_{min,-}\}$ of the AV and the driving state $\{v', a'_{max,+}, a'_{min,-}, a'_{max,-}\}$ of the surrounding vehicle that might have collision with the AV (referred to as "obstacle-of-attention"), together determine the minimum distance d_{min} to avoid collision by Equation 3:

$$\begin{aligned} d_{min} = vt + \frac{1}{2}a_{max,+}t^2 + \frac{(v+ta_{max,+})^2}{2a_{min,-}} \\ +m(v't' + \frac{1}{2}a'_{max,+}t'^2) \\ +n\frac{(v'+t'a'_{max,+})^2}{2a'_{*,-}} + d_\mu \end{aligned} \tag{3}$$

Here, v and v' are velocities of the AV and the obstacle-of-attention, respectively; $a_{min/max,+/-}$ is the minimum/maximum acceleration of speeding up(+) or slowing down(−). Let t and t' be the response times of the AV and the obstacle-of-attention. All variables are non-negative: velocities and accelerations only represent the absolute value.

Parameters m and n are used to represent whether the two vehicles are driving in the opposite or the same directions. The values are scenario-dependent. For example, $(m = 1, n = 1)$ indicates that the two vehicles, the AV and the obstacle-of-attention, are driving towards each other, either along the lane direction or perpendicular to the lane direction. The reasonable conditions are the obstacle-of-attention takes some response time t' before it applies a brake; the most adverse condition is to use the minimum slowing down acceleration $a'_{*,-} = a'_{min,-}$. When the two vehicles are driving towards the same direction, which is represented by $(m = 0, n = -1)$, the most adverse condition is that the obstacle-of-attention instantly brakes $(t' = 0)$ with highest possible acceleration $a'_{*,-} = a'_{max,-}$.

Parameters $t', a'_{*,+/-}, v', a_{*,+/-}, d_\mu$ depend on various environmental conditions and user's safety preference. In practice, under bad weather or road conditions, we may adopt lower v' and $a'_{max,+/-}$ than normal conditions, because vehicles tend to drive slower and apply cautious speed changes. d_μ is the minimum distance between the two vehicles, when they perform safe actions [17] and full stop. Users may specify larger d_μ to allow a larger safety margin or smaller d_μ to achieve higher driving flexibility. Higher $a_{*.+/-}$ enables the AV to react more agile, but may lead to the worse passenger experience. At any time, the AV speed v is known. The AV sensors detect the speed of the obstacle-of-attention. Collapsing all the parameters and grouping by the only variable t gives a quadratic form as Equation 4. Notice that α and β are non-negative.

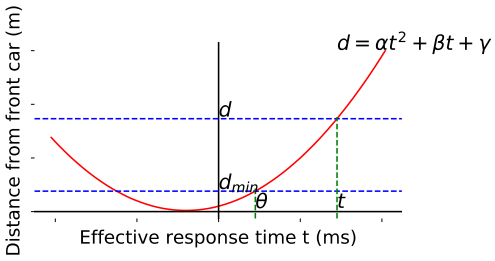


Fig. 6. Illustration of d-θ and t-d_{min}.

$$d_{min} = \alpha t^2 + \beta t + \gamma,$$
$$where\ \alpha = \frac{1}{2}a_{max,+} + \frac{a^2_{max,+}}{2a_{min,-}},\ \beta = v + \frac{va_{max,+}}{a_{min,-}},\ \gamma = \frac{v^2}{2a_{min,-}} \tag{4}$$

According to RSS [17], if the current distance d is larger than d_{min}, the safety criteria is met. Without loss of generosity, a safety score function can be defined as Equation 5. σ is the reward specified by users to achieve extra margin from minimal safety distance; η is the penalty of unit distance less than the minimum safety distance.

$$\text{Safety Score} = \begin{cases} \sigma(d - d_{min}), if d > d_{min} \\ \eta(d - d_{min}), elsewhere \end{cases} \tag{5}$$

We employ Equation 4 to bridge the safety score with computing system response time. First, the current distance d between the AV and the obstacle-of-attention will determine the response time window θ that will avoid the collision. The response time window θ is the time that the AV would take to hit the obstacle-of-attention, if both vehicles keep moving at the same velocity and acceleration – or more formally, if both vehicles conform the Duty of Care, i.e. an individual should exercise "reasonable care" while performing actions that might harm the safety of others [17]. For instance, the Duty of Care allows the front car to perform break, such that the distance to the rear car is reduced, but prohibits the front car to backup, when followed by another car. We determine θ by solving the following function:

$$d = \alpha\theta^2 + \beta\theta + \gamma \tag{6}$$

$$\text{Safety Score} = \begin{cases} \sigma[\alpha(\theta^2 - t^2) + \beta(\theta - t)], if \theta > t \\ \eta[\alpha(\theta^2 - t^2) + \beta(\theta - t)], elsewhere \end{cases} \tag{7}$$

Let t be the AV computing system response time and determine $d_{min} = \alpha t^2 + \beta t + \gamma$. Figure 6 depicts the relationship between Equation 4 with the current distance d and instantaneous computing system response time t. Then the safety score function can be written as Equation 7.

VII. The Need for Safety Score

Our AV computing system adopts GPUs and CPUs as computation devices; Each software module is processed

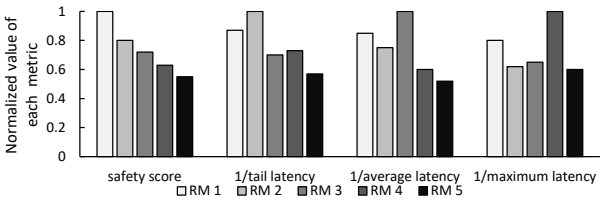


Fig. 7. Resource management (RM) guided by various metrics (y-axis is normalized to the maximum of each metric).

by either CPU or GPU. To demonstrate the use of our safety score, we optimize resource management (RM) to improve the safety of our AV system. A RM strategy assigns each module to execute on either CPU or GPU. The evaluation is performed by the our industrial AV simulator integrated with safety score.

To compare with RM guided by other metrics, we also use our simulator to determine the RM strategies selected by traditional computing system metrics, including 95th and 99th percentile tail latency (both yield similar results), average latency, and maximum latency. We perform an exhaustive search of RMs for software modules. Due to space limitation, Figure 7 shows an example of five RMs. But the following observations hold with our exhaustive search. For fair comparison, we present the results obtained under the same driving scenario.

The results (the higher the better) show that safety-score-guided resource management leads to entirely different results than other metrics. For example, the safety score indicates that we should choose *RM-1*, whereas *RM-2*, *RM-3*, and *RM-4* leads to the best tail latency, average latency, and maximum latency, respectively. Therefore, guiding the computing system design by these traditional latency metrics can be misleading; it is critical to adopt safety score.

VIII. Conclusion

In summary, this paper proposes a safety score to guide the safety-aware AV computing system design. First, we performed a field study to show the safety issues in current AVs. Furthermore, we introduced the safety score and presented its derivation steps. Finally, we evaluated selected RMs by using multiple metrics, and concluded that regular latency metrics are misleading for safety-aware AV computing system design. We believe timely decision making is essential for ensuring AV safety, and hope our safety score will inspire substantial follow-up work.

IX. Acknowledgement

We thank the anonymous reviewers for their valuable feedback. This paper is supported in part by NSF grants 1829524, 1829525, 1817077 and SRC/DARPA Center for Research on Intelligent Storage and Processing-in-memory.

References

[1] "Audi and NVIDIA automotive partners," https://www.nvidia.com/en-us/self-driving-cars/partners/audi/.
[2] "AV safety ventures beyond ISO 26262," https://www.eetimes.com/document.asp?doc_id=1334397#.
[3] "Model S," https://www.tesla.com/models.
[4] "Waymo," https://waymo.com/.
[5] "ISO, Road vehicles: Functional safety," 2011.
[6] "NHTSA, Automated driving systems: A vision for safety," 2017, https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/ 13069a-ads2.0_090617_v9a_tag.pdf.
[7] "Self-driving carts to make their debut on UC San Diego roads in January," 2017. [Online]. Available: http://jacobsschool.ucsd.edu/news/news_releases/release.sfe?id=2350
[8] "Baidu to integrate Mobileye's responsibility sensitive safety model into Apollo program," 2018. [Online]. Available: https://newsroom.intel.com/news/baidu-integrate-mobileyes-responsibility-sensitive-safety-model-apollo\-program/
[9] "Illinois to develop testing program for self-driving vehicles," 2018. [Online]. Available: https://www.chicagotribune.com/news/ct-biz-illinois-self-driving-vehicles-20181025-story.html
[10] "Texas A&M launches autonomous shuttle project in downtown Bryan," 2018. [Online]. Available: https://engineering.tamu.edu/news/2018/10/texas-am-launches-autonomous-shuttle-project-in-downtown-bryan.html
[11] "Uber advanced technologies group," 2019. [Online]. Available: https://www.uber.com/info/atg/
[12] "Valeo signs an agreement with Mobileye to develop a new autonomous vehicle safety standard," 2019. [Online]. Available: https://www.valeo.com/en/valeo-signs-an-agreement-with-mobileye-to-develop-a-new-\autonomous-vehicle-safety-standard/
[13] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5119–5132, 2015.
[14] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2018, pp. 751–766.
[15] D. Nistér, H.-L. Lee, J. Ng, and Y. Wang, "An introduction to the safety force field," in *NVIDIA White Paper*, 2019.
[16] D. Nistér, H.-L. Lee, J. Ng, and Y. Wang, "The safety force field," in *NVIDIA White Paper*, 2019.
[17] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.
[18] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 11, no. 4, p. 85, 2012.