Certified Control: A New Safety Architecture for Autonomous Vehicles

Jeff Chow* Valerie Richmond* Mike Wang* Uriel Guajardo*
Daniel Jackson* Nikos Arechiga† Geoffrey Litt* Soonho Kong† Sergio Campos‡
*MIT †Toyota Research Institute ‡Universidade Federal de Minas Gerais, Brazil

***Abstract*—Widespread adoption of autonomous cars will require greater confidence in their safety than is currently possible. Certified control is a new safety architecture whose goal is two-fold: to achieve a very high level of safety, and to provide a framework for justifiable confidence in that safety. The key idea is a runtime monitor that acts (along with sensor hardware and low-level control and actuators) as a small trusted base, ensuring the safety of the system as a whole. Unlike traditional runtime monitoring, however, a certified control monitor does not perform perception and analysis on its own. Instead, the main controller assembles unforgeable evidence (e.g., from sensors with cryptographically signed outputs) that the proposed action is safe into a certificate which is then checked independently by the monitor. This exploits the classic gap between the costs of finding and checking. The controller is assigned the task of finding the certificate, and can thus use the most sophisticated algorithms available (including learning-enabled software); the monitor is assigned only the task of checking, and can thus run quickly and be smaller and potentially verifiable. Certified control has been implemented in simulation and in a small racecar for two kinds of certificate, one for LiDAR proximity data and the other for visual lane following.**

I. Introduction

If autonomous cars are to become widespread, it will be necessary not only to ensure a high level of safety but also to justify our confidence that such a level has been achieved. This paper proposes a new architecture that is designed to make both safety and its justification possible. The key idea is a variant on the traditional concept of the runtime monitor. Like a conventional monitor, the monitor checks actions proposed by the main controller before passing them on to the actuators; if an action is found to be unsafe, it is blocked or replaced by a safer action.

Unlike a conventional monitor, however, the monitor does not make this decision based on its own evaluation of the environment. Instead, it checks a certificate generated by the main controller that contains, in addition to the proposed action, explicit and concrete evidence that the action is safe. When the monitor approves the certificate, it has essentially ratified a runtime safety case.

The generation of the certificate is the responsibility of the main controller, and is a byproduct of its normal analysis. The form of the certificate, and the argument that a successful check ensures safety, are developed at design time.

Certified control exploits four powerful computer science ideas in a novel combination and context: (1) the gap between the complexities of **finding vs. checking**; (2) the idea of a **small trusted base** (here, the monitor and sensors and actuators) allowing a guarantee of safety without dependence on unreliable components (in particular the perception system) outside the base; (3) the use of **authentication** (e.g., of sensor data) to guarantee safe transmission through an untrusted channel (namely the main perception/controller subsystems); and (4) the idea of **end-to-end safety cases** whose form is justified at design time, but which are applied on the fly.

The contributions of this paper include: (1) A review of existing approaches to ensuring safety in critical systems; (2) A characterization of the demands on a runtime monitor in terms of three essential properties; (3) An argument that, for autonomous vehicles, the classic runtime monitor or safety controller approach fails to satisfy all three properties, and is thus insufficient; (4) A new architecture, called certified control, that achieves the three properties; (5) An evaluation of the architecture in two applications, LiDAR-based obstacle detection and visual lane following, using simulation and a physical racecar; and (6) An analysis of the potential and limitations of the approach.

II. Background

The problem of safety for self-driving cars has two distinct aspects. First is the reality of numerous accidents, many fatal, either involving fully autonomous cars—such as the Uber that killed a pedestrian in Tempe, Arizona [1]—or cars with autonomous modes—such as the Tesla models, which have spawned a rash of social media postings in which owners have demonstrated the propensity of their own cars to repeat mistakes that had resulted in fatal accidents. The metric of "miles between disengagements," made public for many companies by the California DMV [2], has revealed the troublingly small distance that autonomous cars are apparently able to travel without human intervention. Even if the disengagement metric is crude and includes disengagements that are not safety-related [3], the evidence suggests that the technology still has far to go.

Second, and distinct from the actual level of safety achieved, is the question of confidence. Our society's willingness to adopt any new technology relies on our confidence that catastrophic failures are unlikely. But, even for the designs with the best records of safety to date, the number of miles traveled falls far short of the distance that would be required to provide statistical confidence of a failure rate that matches (or improves on) the failure rate of an unimpaired human driver. Even though Waymo, for example, claims to have covered 20

million miles—a truly impressive achievement—this still pales in comparison to the 275 million miles that would have to be driven for a 95% confidence that fully autonomous vehicles have a fatality rate lower than a human-driven car (one in 100 million miles) [4].

A. An alternative to testing

Statistical testing is the gold standard for quality control for many products (such as pharmaceuticals) because it is independent of the process of design and development. This independence is also its greatest weakness, because it denies the designer the opportunity to use the structure of the artifact to bolster the safety claim, and at the same time fails to focus testing on the weakest points of the design, thus reducing the potency of testing for establishing near-zero likelihood of catastrophic outcomes.

One alternative to statistical testing is to construct a "safety case:" an argument for safety based on the structure of the design [5]. The quality of the argument and the extent to which experts are convinced then becomes the measure of confidence. This approach lacks the scientific basis of statistical testing, but is widely accepted in all areas of engineering, especially when the goal is to prevent catastrophe rather than a wider range of routine failures. For example, confidence that a new skyscraper will not fall down relies not on testing (since each design is unique, and non-destructive tests reveal little) but on analytical arguments for stability and resilience in the presence of anticipated forces. In the UK, the use of safety cases is mandated by a government standard [6] for critical systems such as nuclear power plants.

In software too, there is growing interest in safety cases (or, more generally, assurance or dependability cases) [7]. For a cyber-physical system, the safety case is an argument that a machine, in the context of its environment, meets certain critical requirements. This argument is a chain of many links, including: the specification of the software that controls the machine, the physical properties of the environment (including peripheral devices such as sensors and actuators that mediate between the machine and the environment), and assumptions about the behavior of human users and operators. Each link in the chain needs its own justification, and together they must imply the requirements. Ideally, the justification takes the form of a mathematical proof: in the case of software, for example, a verification proof that the code meets the specification. But some links will not be amenable to mathematical reasoning: properties of the environment, and of physical peripherals, for example, must be formulated and justified by expert inspection.

B. The Cost of Verification

For software-intensive systems, the software itself can become a problematic link in the chain. Complex systems require complex software, and that inevitably leads to subtle bugs. Because the state space of a software system is so large, statistical testing can only cover a tiny portion of the space, and thus cannot provide confidence in its correctness. So for high confidence, verification seems to be the only option.

Unfortunately, verification is prohibitively expensive. Even for software produced under a very rigorous process that does not involve verification, the cost tends to be orders of magnitude higher than for conventional software development. NASA's flight software, for example, has cost over $1,000 per line of code, where conventional software might cost $10 to $50 per line [8]. Verifying a large codebase is a Herculean task. It may not be impossible, as demonstrated by the success of recent projects to verify an entire operating system kernel or file system stack. But it typically requires enormous manual effort. SEL4, a verified microkernel, for example, comprised about 10,000 lines of code, but required about 200,000 lines of hand-authored proof, whose production took about 25-30 person years of work [9].

C. Small trusted bases

One way to alleviate the cost of verification is to design the software system so that it has a small trusted base. The trusted base is the portion of the code on which the critical safety properties depend; any part of the system outside the trusted base can fail without compromising safety. This idea is exploited, for example, in secure transmission protocols that employ encryption (and is generalized in the "end-to-end principle" [10]). So long as the encryption and decryption algorithms that execute at the endpoints are correct, one can be sure that message contents are not corrupted or leaked; the network components that handle the actual transmission, in particular, need not be secure, because any component that lacks access to the appropriate cryptographic keys cannot expose the contents of messages or modify them without the alteration being detectable.

Of course, the claim that some subset of the components of a system form a trusted base—really that the other components fall outside the trusted base—must itself be justified in the safety case. It must be shown not only that the properties established by the trusted base are sufficient to ensure the desired end-to-end safety properties, but also that the trusted base is immune to external interference that might cause it to fail (a property often achieved by using separation mechanisms to isolate the trusted base).

D. Runtime monitors and safety controllers

One widely-used approach is to augment the system with a runtime monitor that checks (and enforces) a critical safety property. If isolated appropriately, and if the check is sufficient to ensure safety, the monitor serves as a trusted base.

For safety-critical systems, the runtime monitor might be an entire controller in its own right. This "safety controller" oversees the behavior of the main controller, and takes over when it fails. If the safety controller is simpler than the main controller, it serves as a small trusted base (along with whatever arbiter is used to ensure that it can veto the main controller's outputs). This scheme is used in the Boeing 777, which runs a complex controller that can deliver highly optimized behavior over a wide range of conditions, but at the same time runs a secondary controller based on the control

laws of the 747, ensuring that the aircraft flies within the envelope of the earlier (and simpler) design [11].

The Simplex architecture ([12], [13]) embodies this idea in a general form. Two control subsystems are run in parallel. The high assurance subsystem is meticulously developed with conservative technologies; the high performance subsystem may be more complex, and can use technologies that are hard to verify (such as neural nets). The designer identifies a safe region of states that are within the operating constraints of the system and which exclude unsafe outcomes (such as collisions). A smaller subset of these states, known as the *recovery* region is then defined as those states from which the high assurance subsystem can always recover control and remain within the safe region. The boundary of the recovery region is then used as the switching condition between the two subsystems.

E. The problem of perception in autonomous cars

The safety controller approach relies on the assumption that the controller itself is the most complex part of the system—that from the safety case point of view, the correctness of the controller is the weakest link in the argument chain. But in the context of autonomous cars, perception—the interpretation of sensor data—is more complicated and error-prone than control. In particular, determining the layout of the road and the presence of obstacles typically uses vision systems that employ large and unverified neural nets.

In standard safety controller architectures (such as Simplex [12], [13]), only the controller itself has a safety counterpart; even if sensors are replicated to exploit some hardware redundancy, the conversion of raw sensor data into controller inputs is performed externally to the safety controller, and thus belongs to the trusted base.

This means that the safety case must include a convincing argument that this conversion, performed by the perception subsystem, is performed correctly. This is a formidable task for at least two reasons. First, there is no clear specification against which to verify the implementation. Machine learning is used for perception precisely because no succinct, explicit articulation of the expected input/output relationship is readily available. Second, state of the art verification technology cannot handle the particular complications of deep neural networks—especially their their scale and their use of non-linear activation functions (such as ReLU [14]) which confound automated reasoning algorithms such as SMT and linear programming [15].

An alternative possibility is to not include perception functions in the trusted base. Instead, one could perhaps use a runtime monitor that incorporates both a safety controller and a simplified perception subsystem. Initially, this approach seemed attractive to us, but we came to the conclusion that it was not in fact viable. In the next section we explain why.

F. Desiderata for a runtime monitor: choose two

To see why a runtime monitor that employs simplified perception is not a solution to the safety problem for autonomous cars, we shall enumerate three critical properties that a monitor should obey, and argue that they are mutually inconsistent (at least for the conventional design).

The first property is that the monitor should be **verifiable**. That is, it should be small and simple enough to be amenable to formal verification (or perhaps to fully exhaustive testing). If not, the monitor brings no significant benefit in terms of confidence in the overall system safety (beyond the diversity of an additional implementation, which brings less confidence than is often assumed [13]).

The second property is that the monitor should be **honest**. It should intervene only when necessary, namely when proceeding with the action proposed by the main controller would be a safety risk. Applying emergency braking on a highway when there is no obstacle, for example, is clearly unacceptable. Even handing over control to a human driver is problematic, due to vigilance decrement [16].

The third property is that the monitor should be **sound**. This means that it should ensure the safety of the vehicle within an envelope that covers a wide range of typical conditions. It is not sufficient, for example, for the monitor to merely reduce the severity of a collision when it might have been able to avert the collision entirely.

Unfortunately, it seems that these three properties cannot be achieved simultaneously in a classic monitor design. The problem, in short, is that the combination of honesty and soundness requires a sophisticated perception system, leading to unverifiable complexity. It is easy to make a monitor that is sound but not honest simply by not allowing the vehicle to move; and conversely it is easy to make one that is honest but not sound by not preventing any collisions at all.

This does not mean that monitors that fail to satisfy all three properties are not useful—only that such monitors are not sufficient to ensure safety. Automatic emergency braking (AEB) systems are deployed in many cars now, and use radar to determine when a car in front is so close that braking is essential. But because AEB might brake too late to prevent a collision, it is not generally sound. Responsibility-Sensitive Safety systems [17], on the other hand, use the car's full sensory perception systems to identify and locate other cars and pedestrians, and can therefore be both honest and sound, but due to the complexity of the perception are not verifiable.

III. Certified Control: A New Approach

Certified control (Fig. 1) is a new architecture that centers on a different kind of monitor. As with a conventional safety architecture, a monitor vets proposed actions emanating from the main controller. But the certified control monitor does not check actions against its *own* perception of the environment.

Instead, it relies on the main perception and control subsystems to provide a *certificate* that embodies evidence that the situation is safe for the action at hand. The certificate is designed to be unforgeable, so that even a malicious agent could not convince the monitor that an unsafe situation is safe. The evidence comprises sensor readings that have been selected to support a safety case in favor of the proposed action; because these sensor readings are only *selected* by the main perception and control subsystems (and are signed by the sensor units that produced them), they cannot be faked.

A certificate contains the following elements: (1) the proposed action (for example, driving ahead at the current speed); (2) some signed sensor data (for example, a set of LiDAR points or a camera image); (3) optionally, some interpretive data. This data indicates what inference should be drawn from the sensor readings. For example, if the sensor data comprises LiDAR points intended as evidence that the nearest obstacle is at least some distance away, the interpretive data might be that distance; if the sensor data is an image of the road ahead, the interpretive data might be the purported lane lines.

The evidence and interpretive data are evaluated by the certificate checker using a predefined runtime safety case. As an example, consider a certificate that proposes the action to continue to drive ahead using LiDAR data. In this case, the LiDAR points argue that there is no obstacle along the path; each LiDAR reading provides direct physical evidence of an uninterrupted line from the LiDAR unit to the point of reflection. Together, a collection of such readings, covering the cross section of the path ahead with appropriate density, indicates absence of an obstacle larger than a certain size.

Compare this with a classic monitor that interprets the LiDAR unit's output itself. The LiDAR point cloud is likely to include points that should be filtered out. In snow, for example, there will be reflections from snowflakes. Performing snow filtering would introduce complexity and likely render the monitor unverifiable. On the other hand, a simple monitor would not attempt to identify snow, and could set a low or a high bar on intervention—requiring, say, that 10% or 90% of points in the LiDAR point cloud show reflections within some critical distance. The low bar would result in a monitor that violates soundness, failing, for example, to prevent collision with a motorcycle whose cross section occludes less than the 10% of points. The high bar would result in a monitor that violates honesty, since it would likely cause an intervention due to snow even when the road ahead is empty of traffic.

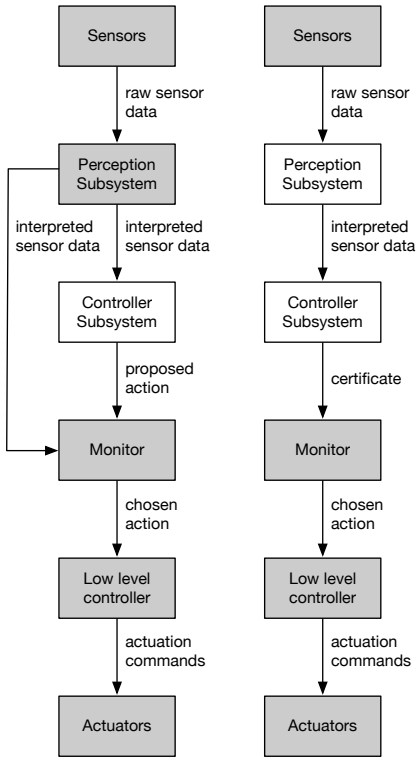
It should be noted that certified control does not remove the sensor and actuator units from the trusted base. What is removed is the main perception and controller subsystems, crucially including complex algorithms that process and interpret sensor readings.

A. Examples and Experimental Setup

In the following sections, three examples of certified control are described: (1) the example alluded to above, in which a certificate captures the result of LiDAR snow filtering; (2) a scheme for checking the result of a visual lane detection algorithm; (3) an augmented version of the lane detection example that combines both visual and LiDAR data to ensure that the purported lane lines lie on the ground plane.

The primary experimentation platform was a 1/10-scale remote-controlled car, outfitted with a Velodyne Puck VLP16 LiDAR scanner, a camera, and a CPU running Linux with ROS. Python2 implementations of a controller and interlock are loaded onto the car and executed in real time. We manipulated the car's environment to reflect various road conditions using colored tape to represent lane lines and other props, such as confetti to represent snow. The LiDAR certificate was tested only on the car; the visual detection certificate was tested on both the car and in virtual simulation using the *openpilot* system.

Fig. 1: A conventional runtime monitor (left) and certified control monitor (right). The trusted base is shown in gray.



The source code for our experimental implementations is publicly available; a URL will be provided after blind review.

IV. A LiDAR Certificate for Snow Filtering

The first example is a certificate that proposes continuing straight ahead, using LiDAR points as evidence that there is no impending obstacle. To ensure the safety of a controller's decision to move forward, the monitor checks that the points in the certificate satisfy two conditions: sufficient *spread* and sufficient *density*[1]. To achieve sufficient spread, certificate points must span the size of one lane in front of the car, both vertically and horizontally.[2]

A. Density Criteria

Normal notions of density could not be applied for the monitor's density check, due to the non-uniform nature of the LiDAR data. Our car's LiDAR unit provides only sixteen rows of rotating sensors, each of which produces thousands of points per scan, so the data is much more dense horizontally than vertically. The density check therefore differs in the two

[1]Of course the safety case also requires that the LiDAR points be beyond the stopping distance at the current speed; these concerns are straightforward and omitted here, but can be detailed in the final version of the paper.

[2]The size of the lane is hardcoded—the width based on US highway standards, and the height based on standard vehicle heights.

directions. The monitor considers the certificate sufficiently vertically dense if it includes points from each of the LiDAR scan rows which fall within the lane. The points are sufficiently horizontally dense if there are no horizontal gaps of a size greater than some pre-specified parameter.

For example, one choice of parameter ensures no obstacles of width 3.5 cm or greater at a distance of 3.7 m ahead, or of width 90 cm or greater at 96 m ahead (the 70 mph stopping distance). This parameter value is fairly conservative; it would be establish the absence of a car but not a motorcycle in the lane ahead. It yields an average certificate size of about 3 kilobytes (without compression), which the monitor checks in the order of tens of milliseconds. For greater safety, one could decrease the parameter to require more LiDAR points, increasing the certificate size and the time to check it proportionately.[3]

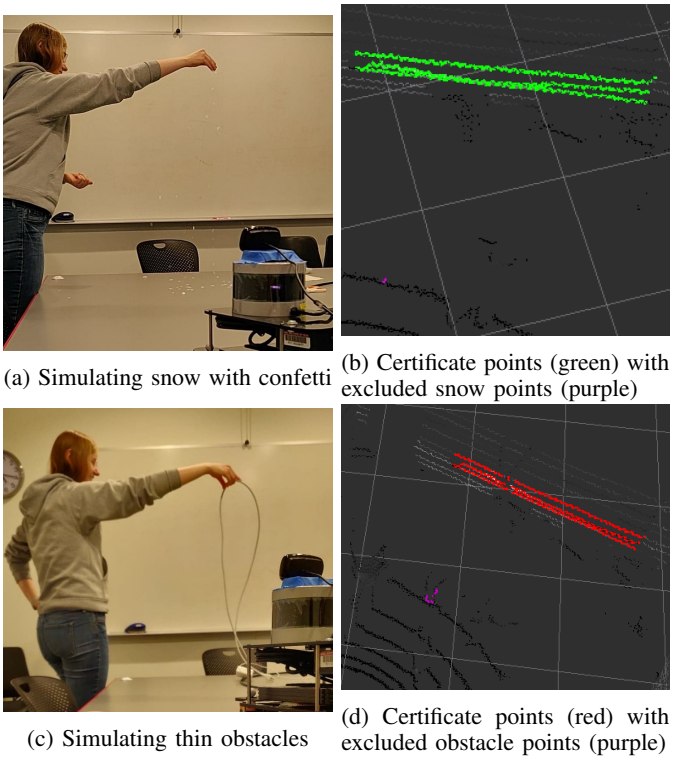
B. Snow Filtering Algorithm

Our controller uses radius outlier removal (ROR) filtering, an improvement upon statistical outlier removal (SOR), to identify LiDAR points reflecting off snow [18]. The controller preprocesses the entire LiDAR point cloud into a k-dimensional tree, which it then queries for nearest neighbor information. Points with few neighbors, relative to the average neighborhoods in the point cloud and based on tunable parameters, are labeled as snow. The controller then constructs the certificate by selecting points from the remaining subset (namely, those not identified as snow) to meet the spread and density criteria.

C. Experiment

We tested our monitor's performance in simulated snowy conditions by dropping confetti-like paper in front of the robot car (Fig. 2a) and confirming that the monitor accepts a certificate when there is sufficient space between the car and an obstacle ahead, despite the presence of simulated snow (Fig. 2b). When the controller's filtering parameters are set appropriately, it properly identifies points as snow and passes a certificate excluding them to the monitor. Assuming snow is distributed fairly evenly across the lane ahead (and there is not a total "white out") the remaining points are still sufficiently dense to establish absence of an obstacle in the lane ahead.

We also tested the the converse case. Tuning the filtering parameters, or adapting them to the perceived environment, can be difficult. One can imagine even a non-adversarial controller failing to set appropriate parameters, and therefore miscategorizing true obstacles as snow. To demonstrate that the monitor would detect such obstacles even when the controller fails to do so, we modified the controller so that it would erroneously filter out some sparse but significant obstacles, such as fallen tree branches (simulated, for our robot car, with plastic cables in Fig 2c). As expected, since the controller omits points on the obstacle from its certificate, the monitor's horizontal density check fails, the certificate is rejected, and the unsafe action is prevented (seen in Fig 2d, with the certificate points in red to show the monitor's rejection).



(a) Simulating snow with confetti

(b) Certificate points (green) with excluded snow points (purple)

(c) Simulating thin obstacles

(d) Certificate points (red) with excluded obstacle points (purple)

Fig. 2: Experiments to test LiDAR certificates against simulated snow (top) and thin obstacles (below) that may be mistakenly identified as snow by the main controller

D. Evaluation

These experiments show that our monitor is agnostic to the complexity of selecting points that meet the density criteria but do not represent reflections off snow particles. This is exactly the benefit certified control brings, since errors in the algorithm that selects the points are immaterial so long as the final point selection comprises adequate evidence. As demonstrated by the second experiment, errors related to overzealous filtering are caught by the monitor.

This behavior was achieved with a system that demonstrates the expected complexity gap between the controller and the monitor. Even with a controller performing only basic snow filtering, with no machine learning algorithms, the controller implementation requires a factor of almost 30 times more lines of code than the monitor (Fig. 3).

E. Limitations

The primary limitation of the LiDAR certificate is that it provides information only about the shape and size of obstacles[4], yet objects with the same shape and size may

[3]In a full implementation, the density of LiDAR points in the certificate would depend on the current velocity since greater resolution is required to prevent collision with the same size obstacle at a greater stopping distance.

[4]LiDAR sensors also provide information about reflectivity; however this information is not leveraged in our system and is not thought to be critical to obstacle detection.

Fig. 3: Comparison of code sizes for monitors vs. certificate generation within the controller. Only the naive vision case is included; the *openpilot* version uses a complex neural net.

Component	Lines of Code
LiDAR Monitor Total	**82**
Controller K-d tree (scipy.spatial)	739
Controller Numpy calls	1325
Controller self-written driver code	167
LiDAR Controller Total	**2231**
Vision Monitors library calls	90
Vision Monitors self-written code	235
Vision Monitors Total	**325**
Controller library calls	460
Controller self-written driver code	612
Vision Controller Total	**1072**

present varying degrees of danger. A paper bag, for example, could occlude the same region of points in a LiDAR certificate as a falling rock. Our monitor cannot distinguish between certificates with these same-sized holes.

Another limitation is the non-uniformity of the LiDAR data itself, mentioned before. The relative lack of vertical density of points means that the system (and indeed, any AV system based on the same LiDAR sensor) is not well-equipped to detect thin horizontal obstacles, like parking arms. Such obstacles, especially at a distance, may fall between horizontal scans, and therefore appear totally hidden to the LiDAR unit. The scan non-uniformity also makes it easier for thin vertical objects, like metal poles, to be overlooked, as they appear as narrow groups of relatively disjoint LiDAR points, one group on each LiDAR scan row.

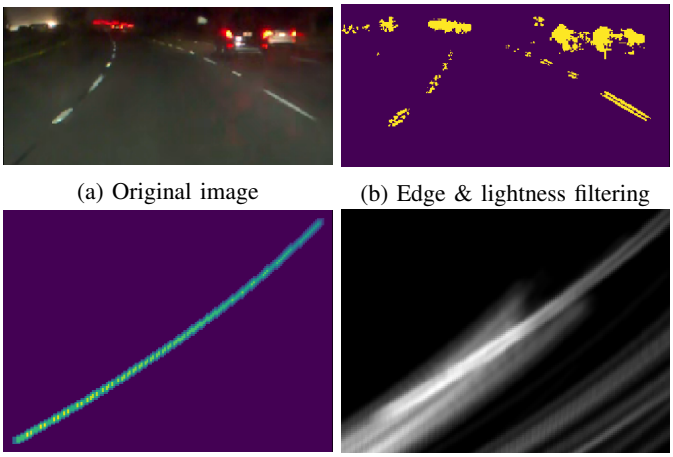
V. A CERTIFICATE FOR VISUAL LANE DETECTION

To explore the application of certified control in the domain of vision, we focused on the verification of lane line detection—that is, we designed a certificate that confirms that the perception system correctly identifies the location of lane lines at runtime.

The certificate contains the following elements:

- the image frame from which the lane lines were deduced;
- the position of the left and right lane boundaries as second-degree polynomials in the bird's-eye/top-down view ($L(i)$ and $R(i)$, respectively), both lines giving the distance from the left side of the top-down view as a function of distance from the front of the car;
- the bird's-eye transformation matrix T used to transform the lane lines;
- a series of color filtering thresholds used to process the image and highlight the presence of the lane lines.

The monitor verifies the certificate in two checks. The first check is geometric. In this test, the proposed lane lines are evaluated as a pair; if they conform to specific geometric bounds (such as parallelism) then the lane lines pass this test. The second check is a computer vision check. Computer vision techniques are used to determine whether the proposed lane lines correspond to lane line markings in the image. If they do, the lane lines pass. The proposed lane lines must pass both checks in order for the certificate to be accepted.



(a) Original image (b) Edge & lightness filtering

(c) Filter for left lane line (d) Correlating filter and image

Fig. 4: The steps of the conformance test.

A. Geometric Test

The geometric test ensures that the purported lane lines are parallel and spaced according to local regulations; in the US, for example, the width of a freeway lane is 12 feet [19]. To conduct the parallelism test, the monitor checks that the distance between points on $L(i)$ and the intersection of its normal with $R(i)$ remains relatively constant. To ensure correct spacing, this distance is required to be on average within a certain delta of the expected distance between lane lines.

The problem of checking the lane width and comparing lane boundaries for parallelism is greatly simplified by working with a bird's-eye perspective of the lane boundaries. The transformation and fitting of the lane-lines is done by the main controller, so the checks performed by the monitor remain simple.

B. Conformance Test

Proposed lane lines might happen to pass the geometric test (or be maliciously tailored to pass it) but not actually correspond to lane lines on the road itself. It is therefore essential to also check that the lane lines conform to a signed image of the road. To do this, the monitor applies simple and well-tested computer vision algorithms to determine if there are corresponding lane markings in the image. The match between the proposed lane lines and the image is checked as follows (and illustrated in Fig. 4) [5]:

[5]Our original monitor implementation used a fixed bird's-eye transformation matrix to convert the camera image to a top-down view (instead of converting the lane lines from the top-down to camera view). This allowed us to create an additional filter to specifically test for dotted lane lines. However, this implementation was less robust to curvy roads and varying camera angles. In addition, *openpilot* already computes a calibration matrix for converting lane lines from top-down view to camera view. We took advantage of this by integrating that matrix into the certificate, giving the monitor more flexibility while remaining simple.

Fig. 5: An example of a lane-detection failure detected by the monitor. The green lines represent the proposed lane lines.

Fig. 6: A frame that caused the conformance test to fail due to the faded right-hand lane line

1) The monitor computes the logical OR of the image filtered with an edge detection algorithm on lightness value (such as the Sobel operator) and the image filtered by lightness above a certain threshold. These thresholds are computed by the controller and passed to the monitor as part of the certificate.
2) Using the bird's-eye transformation matrix passed from the controller, the monitor transforms the lane lines $L(i)$ and $R(i)$ from the bird's-eye view to the image's view.
3) For the transformed left lane curve $L_T(i)$ (and correspondingly for the right), the monitor creates a filter based on the curve, slightly blurred to allow for some margin of error in the proposed lane lines. The bottom of the filter is weighted more heavily because deviations in the region closest to the car are more important. At points not on the curve, the filter is padded with negative values so that only thin lines that match the filter's shape will produce a high correlation output.
4) The monitor computes the cross-correlation C between the filter and the left side of the processed image (and correspondingly for the right), and finds the point of highest correlation in the image (namely $(i_{\max}, j_{\max})$ such that $C[i_{\max}, j_{\max}] = \max(C)$). It checks that the point of highest correlation lies on the transformed lane line $L(i)$, and that the maximum correlation is above a predefined threshold.

C. Experiments

The vision certificate checking scheme was implemented and evaluated against two different lane-detection software systems. The first was *openpilot* [20], in which we used real replay data from a car driven using the *openpilot* system. We used the *Comma2k19* dataset [21], which included driving segments with non-highway driving and adverse lighting conditions (such as a rainy night).

In several instances, our monitor caught lane detection failures (Fig. 5). In all failure cases, however, *openpilot* was able to successfully correct its lane detection within a few seconds. For some of these cases, it was not obvious from inspection of the image taken from the camera's perspective that the proposed lane lines were not geometrically correct. However, viewing from the bird's-eye perspective, it was easier to see that they could not correspond to plausible lane lines.

The certificate used for *openpilot* required about 45KB of storage, dominated by the image. On average, checking of the certificate took about 0.18 seconds on a Dell XPS 9570 Intel Core i7-8750H CPU with 16GB RAM. This would not be adequate performance in a production system, but it is within an order of magnitude of what would be required. With proper optimization of the checking algorithm, and some additional compression of the image, performance seems unlikely to be a problem.

The second experiment was conducted with the physical racecar. We implemented a naive lane-detection scheme, and used colored tape on the floor to simulate lane lines. By placing additional tape segments in inappropriate positions, we were able to get the lane detector to report bad lane lines; in all cases, the monitor correctly rejected them.

D. Limitations

This certificate is inherently less trustworthy than the LiDAR certificate. This seems unavoidable, since unlike the physical obstacles detected by the LiDAR scheme, following lane lines is a social convention, with the lane lines acting as signs whose interpretation is not a matter of straightforward physical properties.

There are two failure cases that we encountered during testing. The first involves obstructed lane lines. Our design assumes that the lane boundaries are always at least partially visible. Since the dataset includes segments with non-highway driving, there were cases in which multiple cars were lined up at a stoplight and obscuring the lane lines. On the highway, the lane lines could be similarly obstructed during traffic or when a car in front is changing lanes. The second failure case involves poorly painted lane lines. In one case, a lane line was so faded it caused the correlation output of the conformance test to not reach the desired threshold and therefore fail the check (Fig. 6). This situation could be mitigated if color filter thresholds were dynamically computed and passed by the controller to the monitor. However, since we could not find such thresholds in the *openpilot* replay data, we passed constant values to the monitor. This failure case also reflects the fact that compelling evidence of the presence of a lane will require well-drawn lane lines; a successful deployment of autonomous cars might simply require higher standards of road markings. It is essential to realize that certified control does

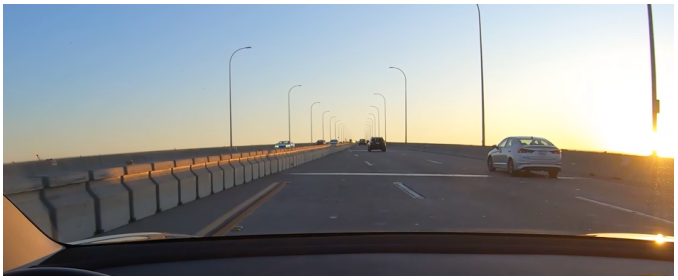


Fig. 7: An image taken from dashcam footage in which a car using Tesla's autopilot almost ran into a concrete barrier. The malfunction was presumably caused by the perception system mistaking the reflection along the barrier for a lane line.

not create or even exacerbate this problem but merely exposes it. If the designer of an autonomous vehicle were willing to rely on inferring lanes from poorly drawn lines, the certificate requirements could be reduced accordingly, so that the level of confidence granted by the certificate reflect the less risk-averse choice of the designer.

A third failure case is of particular importance because it has been observed in erroneous behavior that has caused some cars (notably those produced by Tesla which rely heavily on vision, since they have no LiDAR units) to swerve towards concrete barriers. In this case, the problem is that bright lines, that seem to be lane lines, appear in the camera's image [22]. These lines are actually bands of direct or reflected sunlight. In most cases, we believe that our tests would catch such anomalies; we took one particular image from an online video (Fig. 7) illustrating this problem and confirmed that the inferred lane lines would indeed fail the geometric test and be correctly rejected.

It is possible that these spurious lane lines would have passed the geometry and conformance tests. Apparently a more basic property is being violated: the detected lane line is on the barrier and not on the ground. This motivates a new type of certificate, which we now turn to.

VI. A Certificate Combining Vision and LiDAR

Image data lacks the inherent physical properties of LiDAR data: a single pixel, unlike a single LiDAR point, says nothing about the car's environment, absent other context. As a result, one can only perform limited checks on the lane lines using vision alone. Indeed in the Tesla example, the yellow ray of sun looked, in the image, like a plausible lane line. However, the line was not spatially on the ground plane.

This motivated a certificate that integrates LiDAR and vision data for verifying lane line detection. In addition to providing the lane line polynomials and the camera image (as above), the certificate also provides a set of LiDAR points that lie on the purported lane lines. The monitor checks that these LiDAR points indeed correspond to the lane lines and that they reside on the ground plane. The controller, as usual, is given the more complex task, in this case selecting the LiDAR points.

A. Experiment

In the controller, we take points from the image's lane lines and transform them into the LiDAR space, yielding the corresponding LiDAR readings. Doing so, given knowledge of the camera and LiDAR specifications and their mounting locations on the vehicle, is a matter of basic trigonometric transformations.

We additionally task the controller with identifying a ground plane. Our controller implementation runs a random sample consensus (RANSAC) algorithm [23], augmented with some constraints on necessary ground plane features (e.g. slope, height relative to the car), to determine the plane.

Given the LiDAR lane line points and the ground plane, the monitor checks that the lane points do lie sufficiently close to the ground plane. To simulate the Tesla barrier malfunction, we lined up two rows of tape to look like lane lines. The tape for the left line was placed on the ground while the tape for the right lane was elevated on a platform; from the camera's perspective, the pair of lane lines looked geometrically plausible. When we ran our combined vision-LiDAR implementation on this scenario, the monitor correctly rejected the certificate because the points from the right lane line were not on the ground plane. This check would also detect false lane lines which lie above or below the ground plane for any other reason.

B. Evaluation

The integration of vision with LiDAR presents a few unique challenges. First, the ground plane detector we implemented is not robust to steeply sloped roads or to very uneven road surfaces. Second, since the controller does not include in its certificate any proof of validity of the ground plane it detected, the ground plane algorithm must be regarded as within the trusted base. The obvious remedy—namely including the ground plane detection in the monitor—is not straightforward for two reasons: the algorithm is too complex to be easily verifiable, and the monitor only has access to the points in the certificate, not the entire LiDAR point cloud. Nevertheless, we hope to solve this problem with our usual approach: by tasking the controller with producing a certificate which proves to the monitor the validity of its suggested ground plane, e.g. by inclusion of certain low-lying points.

Overall, even though the racecar used only a naive lane-finding algorithm with no machine learning, the algorithms still used three times as many lines of code as the monitor's three vision checks combined (Fig. 3). Production controllers are of course much more complex—*openpilot*'s deep learning container for lane-finding on github has around 26 convolutional and 7 fully connected layers.

VII. Related Work

The literature on safety assurance of vehicle dynamics splits roughly into two camps, both of which focus on assurance of the planning/control system, and assume perception is assured by some other mechanism.

One focuses on numerical analysis of reachable states. For example, reachable set computations can justify conflict resolution algorithms that safely allocate disjoint road areas to traffic participants [24]. The other focuses on deductive proofs of safety. The work of [25] models cars with double

integrator dynamics, and uses the theorem prover KeYmaera to prove safety of a highway scenario, including lane changing, with arbitrarily many lanes and arbitrarily many vehicles. The work of [26], [27] demonstrates how these safety constraints can be used for verification and synthesis of control policies, including control policies with switching. In [28], the authors develop safety contracts that include intersections, and provide KeYmaera proofs to demonstrate safety.

Responsibility-Sensitive Safety (RSS) [17] is a framework that assigns responsibility for safety maneuvers, and ensures that if every traffic participant meets its responsibilities, no accident will occur. The monitoring of these conditions is conducted as the last phase of planning, and is not separated out as a trusted base. The RSS safety criteria have been explored more formally to boost confidence in their validity [29].

All of these works focus on control, and assume that the perception system is reliable. In contrast, certified control takes the perception subsystem out of the trusted base. Nevertheless, these approaches are synergistic with ours. In our design, the low-level controller is still within the trusted base and would benefit from verification. RSS provides more sophisticated runtime criteria than those we have considered that could be incorporated into certificates (e.g., for avoiding the risk of collisions with traffic crossing at an intersection).

Several approaches aim, like ours, to establish safety using some kind of monitor. The Simplex Architecture [13], [30], described above, uses two controllers: a verified safety controller and a performance controller. When safety-critical situations are detected, the system switches to the verified controller, but otherwise operates under the performance controller. [31] extends Simplex to neural network-based controllers. As we noted, this approach does not address flaws in perception. In contrast, reasonableness monitors [32], [33] defend against flawed perception by translating the output of a perception system into relational properties drawn from an ontology that can be checked against *reasonableness constraints*. This ensures that the perception system does not make nonsensical inferences, such as mailboxes crossing the street, but aims for a less complete safety case than certified control. Similarly, the work of [34] seeks to explain perception results by analyzing regions of an image that influence the perception result. These techniques may be useful to enable the perception system to present pixel regions to the safety monitor as evidence of a correct prediction.

Other techniques seek to use the safety specifications to automatically stress test the implementation [35], [36], [37]. A different approach checks runtime scenarios dynamically against previously executed test suites, generating warnings when the car strays beyond the envelope implicitly defined by those tests [38]. Certified control is similar in that the certificate criteria represent the operational envelope considered by the designers, and outside that envelope, it will likely not be possible to generate a valid certificate, leading to a safety intervention.

Assurance of perception systems needs to grapple with two key challenges. The first is the difficulty of determining appropriate specifications for perception systems, and the second is with developing scalable reasoning techniques to ensure that the implementation satisfies its specifications. Reasonableness monitors provide a partial answer to the first. The second is an active area of research: [39], [40], and [41], for example, develop efficient techniques to prove that a deep neural network satisfies a logical specification. While these technologies are promising, their applicability to industrial-scale applications has not yet been demonstrated. Perception is a particularly thorny problem, since it is not clear what properties of a perception system one would want to formally verify.

VIII. Prospects and Challenges

To evaluate the potential of certified control, let's consider each of its key aspects in turn. Some aspects are essential to the scheme; others reflect limitations of the research in its current state. There are three essential aspects:

- *Justifiable design*. Our most fundamental assumption is that the design of the main controller can be justified by a logical safety case. Prior to the recent advances in machine learning, this would have been uncontroversial: after all, it seems reasonable to insist that any designer should be able to explain why their system works. In contrast, end-to-end machine learning [42] connects sensors to actuators through a single network. While exciting, the dangers of this approach are clear, since behavior may be unpredictable in scenarios not covered by training data (or in the presence of adversarial attacks [43]).
- *Checking end-to-end safety*. The checking of the certificate is intended to establish an end-to-end safety case; for example, a subset of LiDAR points is taken as evidence of absence of an obstacle allowing the car to continue ahead. This contrasts with a sanity check approach that relies on a more ad hoc collection of partial (and even internal) checks. The problem with our approach is that it is more burdensome to implement; its advantage is that it gives greater confidence, since it is possible for a failing controller to pass a collection of sanity checks. Of course, a wise designer would include sanity checks (for example as runtime assertions) in any case.
- *Dynamic checking.* We assume that it is possible to check safety at runtime by executing a predicate on a certificate comprising sensor data and its interpretation. This may not be possible if the main controller uses algorithms whose correctness on an execution cannot be effectively checked, because the gap between finding and checking is small. For example, when an algorithm that decides whether two graphs are isomorphic returns "no" it seems that checking the answer would require enumerating all candidate mappings between the graphs. But even in such cases Blum has shown that probabilistic checks can provide arbitrarily high degrees of confidence [44].

Less essential aspects that suggest further work are:

- *Time independence*. In the current design, each certificate is evaluated independently. In practice, the controller would present a certificate in every controller cycle, and the checking of the certificates would accumulate evidence across frames. This would allow lane lines still

to be inferred even though they are periodically obscured by other vehicles.

- *Determinism*. A related limitation is that certificates are currently interpreted deterministically. In practice, it would obviously be desirable to introduce a probabilistic component. For example, a LiDAR certificate might allow areas of insufficient density so long as they do not occur systematically in subsequent frames.
- *Single sensor*. Currently, certificates contain readings from the sensors used by the main controller. An alternative scheme would allow obtain sensor readings from an additional sensor of its own. This need not compromise the essential idea that the monitor does not include the interpretation algorithms. For example, a LiDAR-based certificate, instead of including actual signed LiDAR points, might include LiDAR coordinates that tell the monitor where to look to obtain the relevant points.

REFERENCES

[1] D. Wakabayashi, "Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam," *The New York Times*, Mar. 2018.

[2] "Testing of Autonomous Vehicles," https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing.

[3] O. Cameron, "The Driverless Readiness Score," https://olivercameron.substack.com/p/the-driverless-readiness-score, library Catalog: olivercameron.substack.com.

[4] N. Kalra and S. M. Paddock, "Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?" RAND Corporation, Tech. Rep. RAND RR-1478-RC, 2016.

[5] C. B. Weinstock, J. B. Goodenough, and J. J. Hudak, "Dependability Cases," CMU Software Engineering Institute, Tech. Rep. CMU/SEI-2004-TN-016, 2004.

[6] "Safety Management Requirements for Defence Systems: Part 2: Guidance on Establishing a Means of Complying with Part I"," UK Ministry of Defense, Tech. Rep., 2007.

[7] D. Jackson, M. Thomas, and L. I. Millett, Eds., *Software for Dependable Systems: Sufficient Evidence?* National Research Council, 2007.

[8] "They Write the Right Stuff," https://www.fastcompany.com/28121/they-write-right-stuff.

[9] G. Klein and et. al., "seL4: Formal verification of an OS kernel," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles - SOSP '09*. Big Sky, Montana, USA: ACM Press, 2009, p. 207.

[10] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[11] Y. C. Yeh, "Dependability of the 777 Primary Flight Control System," in *Dependable Computing for Critical Applications*, 1998.

[12] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar, "The simplex reference model: Limiting fault- propagation due to unreliable components in cyber-physical system architectures," in *IEEE International Real-Time Systems Symposium*, 2007.

[13] Lui Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, Jul. 2001.

[14] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *International Conference on Machine Learning*, 2010, p. 8.

[15] L. Pulina and A. Tacchella, "Challenging SMT solvers to verify neural networks," *AI Communications*, vol. 25, no. 2, pp. 117–135, 2012.

[16] E. T. Greenlee, P. DeLucia, and D. C. Newton, "Driver Vigilance in Automated Vehicles: Hazard Detection Failures Are a Matter of Time." *Human Factors*, 2018.

[17] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," *arXiv:1708.06374 [cs, stat]*, Oct. 2018.

[18] N. Charron, S. Phillips, and S. L. Waslander, "De-noising of Lidar Point Clouds Corrupted by Snowfall," in *Computer and Robotic Vision*, 2018.

[19] W. J. Stein and T. R. Neuman, "Mitigation strategies for design exceptions," United States. Federal Highway Administration. Office of Safety, Tech. Rep., 2007.

[20] "Comma AI OpenPilot Software," https://github.com/commaai/openpilot, Apr. 2020.

[21] "Comma AI Driving Dataset," https://github.com/commaai/comma2k19, Apr. 2020.

[22] "Tesla Autopilot Drives Straight Towards Concrete Barrier on Highway."

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[24] S. Manzinger and M. Althoff, "Tactical Decision Making for Cooperative Vehicles Using Reachable Sets," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI: IEEE, Nov. 2018, pp. 444–451.

[25] S. M. Loos, A. Platzer, and L. Nistor, "Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified," in *FM 2011: Formal Methods*, M. Butler and W. Schulte, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 6664, pp. 42–56, series Title: Lecture Notes in Computer Science.

[26] N. Arechiga, S. M. Loos, A. Platzer, and B. H. Krogh, "Using theorem provers to guarantee closed-loop system properties," in *2012 American Control Conference (ACC)*. Montreal, QC: IEEE, Jun. 2012, pp. 3573–3580.

[27] N. Arechiga and B. H. Krogh, "Using verified control envelopes for safe controller design," in *American Control Conference*, 2014.

[28] S. M. Loos and A. Platzer, "Safe intersections: At the crossing of hybrid systems and verification," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Washington, DC, USA: IEEE, Oct. 2011, pp. 1181–1186.

[29] P. Koopman, B. Osyk, and J. Weast, "Autonomous Vehicles Meet the Physical World: RSS, Variability, Uncertainty, and Proving Safety," in *Computer Safety, Reliability, and Security*, A. Romanovsky, E. Troubitsyna, and F. Bitsch, Eds. Cham: Springer International Publishing, 2019, vol. 11698, pp. 245–253, series Title: Lecture Notes in Computer Science.

[30] D. Phan and et. al., "A Component-Based Simplex Architecture for High-Assurance Cyber-Physical Systems," *2017 17th International Conference on Application of Concurrency to System Design (ACSD)*, pp. 49–58, Jun. 2017.

[31] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural Simplex Architecture," *arXiv:1908.00528 [cs, eess]*, Mar. 2020.

[32] L. H. Gilpin and J. C. Macbeth, "Monitoring Scene Understanders with Conceptual Primitive Decomposition and Commonsense Knowledge," *Advances in Cognitive Systems*, p. 20, 2018.

[33] L. H. Gilpin, "Reasonableness Monitors," *AAAI*, 2018.

[34] J. Kim and J. Canny, "Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017, pp. 2961–2969.

[35] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium*, 2018.

[36] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh, "Automatic Testing and Falsification with Dynamically Constrained Reinforcement Learning," *arXiv:1910.13645 [cs, eess]*, Feb. 2020.

[37] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validation," in *IEEE Intelligent Transportation Systems Conference*, 2019.

[38] M. Mauritz, F. Howar, and A. Rausch, "Assuring the Safety of Advanced Driver Assistance Systems Through a Combination of Simulation and Runtime Monitoring," in *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2016, vol. 9953, pp. 672–687, series Title: Lecture Notes in Computer Science.

[39] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," *arXiv:1702.01135 [cs]*, May 2017.

[40] R. Ehlers, "Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks," *arXiv:1705.01320 [cs]*, Aug. 2017.

[41] G. Katz and et. al., "The Marabou Framework for Verification and Analysis of Deep Neural Networks," in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, vol. 11561, pp. 443–452, series Title: Lecture Notes in Computer Science.

[42] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang,

J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv:1604.07316 [cs]*, Apr. 2016.

[43] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Machine Learning," *Asia Conference on Computer Science and Security*, Mar. 2017.

[44] M. Blum and S. Kannan, "Designing Programs that Check Their Work," *Journal of the ACM*, 1995.