Architecting Dependable Learning-enabled Autonomous Systems: A Survey

Chih-Hong Cheng\textsuperscript{1}, Dhiraj Gulati\textsuperscript{1}, and Rongjie Yan\textsuperscript{2}

\textsuperscript{1} fortiss - Research Institute of the Free State of Bavaria, Germany
\textsuperscript{2} State Key Laboratory of Computer Science, China
\{cheng, gulati\}@fortiss.org, yrj@ios.ac.cn

Abstract. We provide a summary over architectural approaches that can be used to construct dependable learning-enabled autonomous systems, with a focus on automated driving. We consider three technology pillars for architecting dependable autonomy, namely diverse redundancy, information fusion, and runtime monitoring. For learning-enabled components, we additionally summarize recent architectural approaches to increase the dependability beyond standard convolutional neural networks. We conclude the study with a list of promising research directions addressing the challenges of existing approaches.

1 Introduction

Recent fatalities of autonomous vehicles on public roads such as the 2018 Uber crash \cite{1} have raised the necessity of creating safe autonomous systems. Apart from having a rigorous verification and validation (V\&V) plan, automotive safety standards such as ISO 26262 or ISO/PSA 21448 also emphasize the importance of designing dependable system architectures. Although researchers have developed various techniques that may act as building blocks for dependability, what remains missing, however, is a systematic categorization over existing research results while mapping these techniques to a reference architecture for dependable autonomy.

In this paper, based on an extensive review of prior works, we provide a summary over architectural approaches that can be used to construct dependable autonomous systems, with a focus on automated driving. We consider diverse redundancy, information fusion, and runtime monitoring as key architectural means for building dependable autonomous systems.

- **Diverse redundancy** refers to the use of different paradigms or algorithms to achieve the end-to-end workflow from sensing to actuation.
- **Information fusion** refers to the technique of merging homogeneous or heterogeneous information from diverse sources into one.
- **Runtime monitoring** refers to the technique of detecting obvious or intriguing abnormality of information being produced by components or subsystems.

Data-driven engineering (learning from data) is increasingly used to create perception, prediction, or even decision components whose precise specification
is hard to characterize in automated driving tasks. Therefore, as an addition to the above list, we also include

– architecture approaches for dependable machine learning (ML) that go beyond standard convolutional neural networks (CNN).

How the above mentioned four ingredients constitute to dependable architectures for autonomous driving can be understood using the example in Figure 1. It can be seen that a dependable ML is crucial to ensure the quality of the sub-systems learned from data, runtime monitoring examines erroneous behaviors of components created with diversity, and lastly, information fusion techniques may adjust the decision based on the result of monitors.

2 Architecting Dependable Autonomous Driving Systems

Figure 2 outlines the broad categorization of the four points mentioned in the previous section. The state-of-the-art architectures are further explored and divided based on this categorization.
2.1 Diverse redundancy

Intuitively speaking, realizing autonomous driving can be understood as implementing a transducer (function with states) that takes high-dimensional input (e.g., vision) and derives output in low-dimensional space (steering, acceleration and break). The overall process can be further refined into a sequence of high-level functionalities including perception, prediction, planning, and actuation. From the related work, there are three possible means to achieve diversity:

1. **Implementing the same functionality using different methods.** For each of the high-level functionalities, one can implement the same functionality using different methods.
   - For perception, one can either use stereo vision or lidar to perform 3D scene reconstruction, which internally employs different algorithms. Vision-based perception techniques involve many dimensions, such as distance estimation [81], painted line detection [54], and 3D dense semantic map [80]. Methods of perception with lidar have shifted from classification [9] to detection with convolutional networks, such as vehicle detection [41], zone detection [49], and semantic segmentation with conditional random field [75]. There is also a trend to fuse various sensory data for various detection purposes [7,78]. Additional to image perception, the work in [17] could spot acoustic events, such as horns for smart vehicles, with anomaly detection for the urban soundscape model.
   - For prediction, algorithm diversity includes probabilistic approaches, rule-based approaches, and ML-based approaches. For example, using these information and chassis information, probabilistic prediction in [11] is conducted for ego vehicle and surrounding vehicle separately, to facilitate motion planning. Using goal-directed planning, pedestrians can be predicted based on common behavior patterns learned by a fully convolutional network operating on maps of the environment [58]. By modeling the interaction of all its surrounding vehicles’ trajectories, without over-the-air communication between vehicles, the method proposed in [18] could predict the continuous lane-change trajectory of a target car.
work in [28] combines a Bayesian filtering technique for environment representation, and machine learning as long-term predictor, and applies future and past estimates to separate static and dynamic regions.

– For planning and decision making, algorithm diversity includes sampling-based approaches, grid-based approaches, potential fields approach and others. There are several reviews for planning, such as [22,62]. As summarized in [62], planning with input space discretization, such as lattice planners [56] and road-aligned primitives [74] is simple and effective. Randomized planning, such as rapidly exploring random trees (RRT) [36], could explore large state space with a high computational cost. Initially applied for path following, constrained optimization [44] can also compute collision-free trajectories to avoid other traffic participants. The optimization scheme can also improve the quality of the solution [17]. Considering the dynamic interaction between autonomous vehicles and human drivers, Sadigh et al. provide an optimization-based planner in the dynamical system [59]. The recent review [62] has further summarized results on planning and decision-making for autonomous vehicles.

– For actuation, algorithm diversity ranges from Proportional-Integral-Differential (PID) control to complex model-predictive control. For example, an embedded fuzzy-logic-based control system controls both speed and steering of automated vehicles [51]. Considering the tire force saturations, a nonlinear feedback strategy is proposed to improve the transient performance and eliminate the steady-state errors in path-following control [71].

2. **Merging functionalities in the pipeline.** Apart from the classical pipeline and modular approach, one architectural diversity is to merge functionalities by utilizing machine learning. Figure 3 illustrates the possible architectures using this paradigm.

– One extreme is an end-to-end approach (behaviour reflex) [57,79,24,38], which directly trains a neural network as a single transducer (i.e., it merges four functionalities into one). To achieve the goal, it takes an image and other sensory inputs and directly outputs the steering angle and acceleration parameters.
While end-to-end approaches merges four functionalities into one, diversities can also be achieved by building a neural network that only performs the task of a subset of functionalities, i.e., the so called mid-to-mid approach.

- One realization is direct perception \[12,61\] which uses machine learning to translate the high dimensional sensor input to high-level features called affordances (e.g., distance to the preceding car, or detection of traffic lights), such that the features are used as input for the vehicle planning and control algorithms.

- Another realization can be found in the work of ChauffeurNet \[5\]. It starts by assuming that the bounding boxes of objects are made available on the bird’s view, and the system trains an RNN to perform motion prediction and path planning.

- Yet another possibility is to go from (visual) perception to planning. The work by Song et al. \[67\] trains a motion planning system, where internally it combines a convolution neural network and a long short-term memory (LSTM) to extract spatial and temporary features of the input images.

3. Moving intelligence outside the vehicle. Apart from making a vehicle intelligent, the third diversity is to offload the task of the vehicle by preparing a digital infrastructure that facilitates vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication \[26\]. For example, for a vehicle sensing a traffic light, state-of-the-art approaches rely on visual perception. Alternatively with V2I communication support, the vehicle can also confirm the detection of traffic light by receiving the signal from the infrastructure. With these two methods a redundancy in functionality is achieved with support from intelligence outside the vehicle. Various work in the literature provides the benefits of using such an approach.

- (V2V) For highway scenarios, when the autonomous vehicle receives information from other vehicles or sensory data from the infrastructure, due to there being more data sources over the nearby vehicles, the precision can be higher as demonstrated in cooperative localization \[39\], a concept that dated in early 90s. The CoPark method allows negotiation between various vehicles within its V2V communication range and decides on the search areas for parking \[2\]. Protocol MoZoa provides a moving-zone based architecture to facilitate information dissemination, where vehicles can collaborate with one another to form dynamic moving zones \[43\].

- (V2I) As a recent example, the research project Providentia \[25\] builds a digital infrastructure on the German A9 highway to enable high speed V2I communication via 5G networks. For scenarios such as automated valet parking, the vehicle can retrieve information of free parking spaces by accessing the parking area infrastructure with occupancy detection camera \[45\].
2.2 Information fusion

As already discussed, to develop a dependable architecture, redundancy is required and hence multiple such pipelines are executed in parallel. Therefore, a dependable autonomous driving system should fuse not only the sensor data but also data originating in each pipeline with each other at the same stage.

As seen in the architectures investigated in the last section, multiple components can be assembled in various combinations to generate distinct pipelines. It is then the job of information fusion to choose the best possible outcome and deliver it to the next component in the pipeline. Such fusion is also known as decision fusion.

Table 1 (derived from [42] Chapter 5) presents an overview of the common methods used for the information to arrive at decisions. Hard decisions constitute the fusion methods when they result in a single optimal choice. The soft decisions are the result of methods where there may be more than one decision but each decision will have an uncertainty associated with it. Based on the requirements, different information fusion methods can be used in different parts of the automated driving pipeline.

Boolean information fusion method is the easiest to understand and implement. If one pipeline reports an object and other does not, a Boolean OR decision will pass its existence to the next level. On the other hand, a Boolean AND will not report it. A combination of such operators based on multiple data points generated from one step in a pipeline with another pipeline can result in a simple yet powerful decision.

Weighted sum score (also known as weighted sum model) [70] is another method which can also be used to arrive at hard decisions. If multiple sources result in information in same units, this method can be used to combine and arrive at one concrete answer. Each information can also be assigned some weight.

| Decision Type | Method                      | Description                                           |
|---------------|-----------------------------|-------------------------------------------------------|
| Hard decision | Boolean                     | Apply logical AND, OR to combine independent values   |
|               | Weighted Sum Score          | Weight values by inverse of their errors and sum to derive score function |
|               | M-of-N                      | Confirm information based on m-out-of-n sources that agree |
| Soft decision | Bayesian                    | Apply Bayes rule to combine independent conditional probabilities |
|               | Dempster-Shafer            | Apply Dempster’s rule of combination to combine information belief functions |
|               | Fuzzy variable             | Combine fuzzy variables using fuzzy logic (AND,OR) to derive combined membership function |

Table 1. Common information fusion combination alternatives to make decisions.
(based on some past knowledge like error, past observations etc.) and then can be combined to arrive at a decision.

*M-of-N* is a standard voting method where majority value is considered as the final decision. This methodology has been used in designing fault tolerant systems [52]. These methods can also be combined together with other to arrive at more powerful methods. For example, weights may be added before voting.

*Bayesian decision making* is one of the most common soft decision making processes. The commonly seen Kalman Filters [73] and Particle Filters [4] use Bayesian decision making to arrive at final decisions with covariances matrices. The matrices represent the uncertainty of the arrived final decision.

The *Dempster-Shafer theory*, also known as the theory of belief functions, is a generalization of the Bayesian theory of subjective probability [15]. Whereas the Bayesian theory requires probabilities for each information of interest, belief functions allow us to form degrees of belief for one information based on probabilities for a related information. Further, Dempster’s rule is used for combining such degrees of belief when they are based on independent items of evidences [65]. Where Bayesian decision making relies on *degree of agreement*, Dempster-Shafer method tries to measure *absence of conflict*. The output is the decision with a belief associated with it.

*Fuzzy logic system* is a nonlinear mapping of an input information vector into a scalar output. Fuzzy set theory and fuzzy logic establish the specifics of the nonlinear mapping [50]. Results from this non-linear mapping can be used along with Boolean operators to make decisions. Such rules which facilitate the fusion are called *fuzzy rules*. The result then is defuzzied to arrive at the final output. Since the initial error is assumed to be fuzzy, the output error also remains fuzzy [60].

The above methods by no means represent an exhaustive list of information fusion. In fact more than one of the above approaches can be combined to result in *Hybrid decisions*. For example, Bayesian approaches like Kalman Filter can be used to arrive at some information along with the required covariance matrix in multiple parallel pipelines. This matrix can be inverted and summed to derive appropriate weights for each of the pipeline process. For the next step a weighted sum score can be used to make the final decision. Here we converted the soft decision to the hard decision. Additionally, this implies that the weights of contribution of each pipeline may change over time.

### 2.3 Monitoring / Runtime verification

Monitors observe and report the abnormal outputs that deviate from the correct behavior by either observing the states or by analyzing finite history traces. As monitors are part of the concept of fault detection, isolation and recovery (FDIR) [32] which is seen in realizing safety critical systems such as avionics, here we restrict our focus to monitoring machine learning components and examining behavior of the autonomous vehicles. As detailed below, our categorization is based on whether a functional specification or an indirect correctness specification is used in the monitor.
Monitoring indirect correctness information For vision-based perception components, it is very hard to create correctness specification from first principles. Therefore, the labelled data is commonly considered the specification and neural networks are used for implementation. Due to the lack of specification, the goal of monitoring is to indirectly check if the data received in operation is not “distant” from training data in the input space (as this can imply excessive extrapolation from training data), or if there exists distribution shift, i.e., the data distribution in operation significantly deviates from the data distribution in the training (as this can imply that the network is not trained properly).

1. Monitoring computed values of a neural network. This line of research targets to detect if value computed by a neural network significantly deviates from the known values computed using the training data. Neural networks implementing zero-shot learning \[40\] can detect new classes (as abnormalities). The basic principle is to associate each class in training with a particular combination of predefined attributes\[^3\]. So the monitor records each class with the unique attribute pair. Whenever an input leads to a new set of attributes, the system considers it as a new type of object, and existing classification is not used. The work of Cheng et al. \[13\] uses a monitor to store the neuron on-off activation patterns of training data as a set using binary decision diagrams (BDD) \[10\]. In operation, the monitor rejects the decision made by the network if the computed activation pattern is not contained in the BDD.

2. Monitor the distribution of predicted classes. The goal is to understand if the distribution of the data has shifted. If so, then it might imply that the neural network originally trained under the specific distribution may not be able to perform (as there is a difference between training data set and operation data set) and the decision may be problematic. The common technique is based on a concept of change-point detection, where one takes \(k\) most recent points and compare it with \(2k\) most recent points, i.e., to examine it using two bags of data \[23\].

Monitoring with precise & correct specification Specification-based monitors usually adopt formal specifications extended from LTL such as MTL \[35\], STL \[72\] or other extensions \[16\]. These monitors are used to detect abnormal temporal behavior or assumption violation of individual modules \[64, 16\] or between the interaction of various modules \[72\]. The following categories are distinguished from most of the specification-based monitors.

1. Real-time monitors. They are implemented on physical devices that are connected to the interested system. They are constrained by the frequency of the target, and the availability of resources \[53, 64\]. Both qualitative \[33\] and quantitative \[34\] semantics for STL with real time operators and numerical inputs can be implemented in FPGA.

\[^3\] As an example, a stop sign in Germany can be associated with attributes such as “red”, “circle”, “octagon”.

2. **Failure prevention monitors.** Instead of purely reporting property or assumption violations, such monitors at runtime could enforce the system to behave correctly [77] or recover from abnormal behaviours [63]. Such monitors are automata-based, which are synthesized [77] or transformed [63] with respect to safety properties. Another type of monitors is to add interventions in the case of potential violations such that the system could reach functional states [48].

3. **Falsification methods.** Instead of passively observing abnormal behavior from runtime traces, the methods attempt to find counterexamples for property or assumption violations with the given model of the system (or the actual system). For systems containing machine-learning component, compositional methods are adopted to isolate possible falsifications and reduce the search space [19]. Subsequently, a temporal logic falsifier and a ML-based analyzer for detecting misclassifications cooperate to find falsifying executions of the considered model.

### 2.4 Beyond standard CNN for dependable ML

This section summarizes additional tactics that can be viewed as extending existing standard CNN engineering approaches with the goal of increasing dependability. Some of the described techniques are recent in that no application in autonomous driving has been reported.

1. **Monte-Carlo Dropout Techniques [21].** Dropout is commonly used in training to prevent overfitting of neural networks. The basic principle of dropout is to randomly disable some neurons of a neural network in training time, in order to make sure that the decision of a neural network is not tightly bounded to certain neurons. Contrarily, the Monte-Carlo Drop Out technique is a way of using dropout in operation time. Let the original set of neurons be $N$. Whenever dropout is performed first on the set $A$ of neurons followed by set $B$ of neurons, then $N \setminus A$ and $N \setminus B$ are essentially two different networks. A direct consequence is that one can perform drop out many times, in order to perform voting (thus it can be viewed as a special form of ensemble), or to perform uncertainty estimation if out of 100 networks with neurons being dropped out, 10 of them have demonstrated deviating behaviours, then one may estimate that the network outputs the value with certainty $\frac{100-10}{100} = 90\%$.

2. **Interpretable CNN [82][83].** This division of research work focuses on modifying the training objective in design time, such that the loss function encourages (1) a neuron to be exclusively activated by a certain category $C$ and to keep silent on other categories, and (2) a neuron to only be activated by a single region of the feature map, instead of being repetitively triggered at different locations. Whenever a network is constructed using the above approach [82], as the activation of a neuron is specific to a location and to a specific type, it is then further possible to construct a decision tree by considering what forms a decision of classifying an object with category $C$ [83].
3. **Neural networks with provable defences** [37][68]. The neural network is first trained using standard approaches. Subsequently, one performs techniques similar to robust optimization techniques [6] to the training of the neural network, which guarantees that any point that is sufficiently close to the training data demonstrates the same behavior like the training data.

4. **Sequence models.** Unlike stateless CNN, recurrent neural networks such as LSTM [27] or GRU [14] can use their internal state (memory) to process a sequence of inputs. This allows ML models to incorporate temporal information to reduce noise and to make better predictions. Applications in autonomous driving include trajectory generation of ego vehicles [5][3] and the prediction of surrounding objects [55][84].

### 3 Concluding Remarks

In this paper, we provide a review on architectural means that can be used to construct dependable autonomous systems, with a focus on diverse redundancy, information fusion, runtime monitoring, and dependable ML techniques. The motivation of the study is to present researchers and practitioners a thorough understanding of the spectrum of existing approaches and to allow them to adapt the approaches in building dependable autonomous driving solutions. We conclude this report by highlighting some prominent research questions.

**Diverse redundancy** The first challenge for diverse redundancy refers to a need of a systematic and rigorous argument where in the operational design domain (ODD), implementations for diverse redundancy should not encounter a common failure scenario where all implementations fail. This requires a careful verification and validation plan which is beyond the scope of this paper, but many fruitful results are already targeting the V&V aspects of autonomous vehicles [30][31]. The second challenge refers to the use of ML for merging functionalities in the pipeline. For automotive domain, there are different vehicle configurations due to shared vehicle platform and due to faster cycles to produce new generations, and arguably the ML function may not be directly transferrable from one configuration to another.

**Information fusion** To arrive at correct decisions between parallel pipelines, first challenge is the fidelity of various models (like process and error) used in the fusion. Most solutions assume linear gaussian data and noises and any deviation in this assumption gives incorrect results. Second challenge is the rigid fusion architectures, where failure in one data source results in failure of the complete fusion process. One method to overcome this is to provide feedback from the monitors to the fusion process. This would result in an adaptive fusion process. This itself is a topic for ongoing research [29][46]. There has been some success in direction of adaptive kalman filters [20][69][66].

**Runtime monitoring** Apart from classical performance considerations, monitors for ML components may be improved by considering the following research
questions. (1) How to better formulate the “distance” concept for data in operation to data in the training set, such that it leads to the creation of resource efficient monitors with high error detection rate. (2) How to generate human interpretable monitors for ML-components. Examples for interpretable monitors in object detection include “impossibility for overly large objects to exist in the image”. For specification-based monitors, the foreseeable challenges are (3) how to prioritize specifications to be monitored based on a scientific characterization of occurrence, effect, and controllability, and (4) how to detect abnormality for unknown unknowns, preferably with ML-based techniques. One final challenge is to go from runtime monitoring to runtime enforcement [8,76].

References

1. NTSB’s Report on the 2018 Uber Crash. [https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf] 2018.
2. A. Aliedani, S. W. Loke, A. Desai, and P. Desai. Investigating vehicle-to-vehicle communication for cooperative car parking: The copark approach. In 2016 IEEE International Smart Cities Conference (ISC2), pages 1–8. IEEE, 2016.
3. F. Altché and A. de La Fortelle. An LSTM network for highway trajectory prediction. In IEEE 20th ITSC, pages 353–359. IEEE, 2017.
4. M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on signal processing, 50(2):174–188, 2002.
5. M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:1812.03079, 2018.
6. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. Robust optimization, volume 28. Princeton University Press, 2009.
7. J. S. Berrio, J. Ward, S. Worrall, W. Zhou, and E. Nebot. Fusing lidar and semantic image information in octree maps. In ACRA, 2017.
8. R. Bloem, B. Könighofer, R. Könighofer, and C. Wang. Shield synthesis. In TACAS, pages 533–548. Springer, 2015.
9. N. Brodu and D. Lague. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. ISPRS Journal of Photogrammetry and Remote Sensing, 68:121–134, 2012.
10. R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys (CSUR), 24(3):293–318, 1992.
11. H. Chae, Y. Jeong, S. Kim, H. Lee, J. Park, and K. Yi. Design and vehicle implementation of autonomous lane change algorithm based on probabilistic prediction. In 21st ITSC, pages 2845–2852. IEEE, 2018.
12. C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In ICCV. IEEE, 2015.
13. C. Cheng, G. Nührenberg, and H. Yasuoka. Runtime monitoring neuron activation patterns. In DATE, 2019.
14. K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.
15. A. P. Dempster. A generalization of bayesian inference. Journal of the Royal Statistical Society: Series B (Methodological), 30(2):205–232, 1968.
16. A. Dokhanchi, H. B. Amor, J. V. Deshmukh, and G. Fainekos. Evaluating perception systems for autonomous vehicles using quality temporal logic. In RV, 2018.
17. D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *IJRR*, 29(5):485–501, 2010.
18. C. Dong and J. M. Dolan. Continuous behavioral prediction in lane-change for autonomous driving cars in dynamic environments. In *21st ITSC*. IEEE, 2018.
19. T. Dreossi, A. Donzé, and S. A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. In *NFM*. Springer, 2017.
20. P. J. Escamilla-Ambrosio and N. Mort. Multi-sensor data fusion architecture based on adaptive kalman filters and fuzzy logic performance assessment. In *FUSION*, volume 2, pages 1542–1549 vol.2, July 2002.
21. Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059, 2016.
22. D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE TITS*, 17(4):1135–1145, 2016.
23. A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.
24. S. Hecker, D. Dai, and L. Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *ECCV*, pages 435–453, 2018.
25. G. Hinz, J. Eichinger, M. Büchel, and A. Knoll. Proactive video-based use of telecommunications technologies in innovative motorway scenarios. *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication*, page 41, 2017.
26. L. Hobert, A. Festag, I. Llatser, L. Alтомare, F. Visintainer, and A. Kovacs. Enhancements of v2x communication in support of cooperative autonomous driving. *IEEE Communications Magazine*, 53(12):64–70, Dec 2015.
27. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
28. S. Hoermann, M. Bach, and K. Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *ICRA*, pages 2056–2063. IEEE, 2018.
29. L. Hong. Adaptive data fusion. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, pages 767–772 vol.2, Oct 1991.
30. X. Huang, D. Kroening, M. Kwiatkowska, W. Ruan, Y. Sun, E. Thamo, M. Wu, and X. Yi. Safety and trustworthiness of deep neural networks: A survey. *arXiv preprint arXiv:1812.08342*, 2018.
31. C. Hutchison, M. Zizyte, P. E. Lanigan, D. Guttendorf, M. Wagner, C. L. Goues, and P. Koopman. Robustness testing of autonomy software. In *ICSE: Software Engineering in Practice*, pages 276–285. ACM, 2018.
32. I. Hwang, S. Kim, Y. Kim, and C. E. Seah. A survey of fault detection, isolation, and reconfiguration methods. *IEEE TCST*, 18(3):636–653, 2010.
33. S. Jakšić, E. Bartocci, R. Grosu, R. Kloibhofer, T. Nguyen, and D. Ničković. From signal temporal logic to fpga monitors. In *MEMOCODE*. IEEE, 2015.
34. S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković. Quantitative monitoring of stl with edit distance. In *RV*, pages 201–218. Springer, 2016.
35. A. Kane, O. Chowdhury, A. Datta, and P. Koopman. A case study on runtime monitoring of an autonomous research vehicle (arv) system. In *RV*. Springer, 2015.
36. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
37. J. Z. Kolter and E. Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.
38. J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez. Evolving large-scale neural networks for vision-based torcs. 2013.
39. R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *ICRA*, pages 1250–1257. IEEE, 1994.
40. C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453–465, 2014.
41. B. Li, T. Zhang, and T. Xia. Vehicle detection from 3D lidar using fully convolutional network. *arXiv preprint arXiv:1608.07910*, 2016.
42. M. Liggins II, D. Hall, and J. Llinas. *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC press, 2017.
43. D. Lin, J. Kang, A. C. Squicciarini, Y. Wu, S. Gurung, and O. K. Tonguz. Mozo: A moving zone based routing protocol using pure v2v communication in vnet. *IEEE Transactions on Mobile Computing*, 16:1357–1370, 2017.
44. A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5), 2015.
45. C. Lörper, C. Brunken, G. Thomaidis, S. Lapoehn, P. P. Fouopi, H. Mosebach, and F. Köster. Automated valet parking as part of an integrated travel assistance. In *16th IEEE ITSC*, pages 2341–2348. IEEE, 2013.
46. G. Loy, L. Fletcher, N. Apostoloff, and A. Zelinsky. An adaptive fusion architecture for target tracking. In *IEEE ICAFGR*, pages 261–266, May 2002.
47. L. Marchegiani and I. Posner. Leveraging the urban soundscape: Auditory perception for smart vehicles. In *ICRA*, pages 6547–6554. IEEE, 2017.
48. L. Masson, J. Guiochet, H. Waeselynck, K. Cabrera, S. Cassel, and M. Törngren. Tuning permissiveness of active safety monitors for autonomous systems. In *NFM*, pages 333–348. Springer, 2018.
49. D. Maturana and S. Scherer. 3D convolutional neural networks for landing zone detection from lidar. In *ICRA*, pages 3471–3478. IEEE, 2015.
50. J. M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377, March 1995.
51. J. E. Naranjo, M. A. Sotelo, C. Gonzalez, R. Garcia, and T. De Pedro. Using fuzzy logic in automated vehicle control. *IEEE intelligent systems*, 22(1):36–45, 2007.
52. V. P. Nelson. Fault-tolerant computing: fundamental concepts. *Computer*, 23(7):19–25, July 1990.
53. T. Nguyen, E. Bartocci, D. Ničković, R. Grosu, S. Jalsic, and K. Selyunin. The harmonia project: hardware monitoring for automotive systems-of-systems. In *ISoLA*. Springer, 2016.
54. M. Olivares-Mendez, J. Sanchez-Lopez, F. Jimenez, P. Campoy, S. Sajadi-Alamdari, and H. Voos. Vision-based steering control, speed assistance and localization for inner-city vehicles. *Sensors*, 16(3):362, 2016.
55. S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *IV*, pages 1672–1678. IEEE, 2018.
56. M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3), 2009.
57. D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, pages 305–313, 1989.
58. Y. Qian, M. Yang, C. Wang, and B. Wang. Pedestrian feature generation in fish-eye images via adversary. In *ICRA*, pages 2007–2012. IEEE, 2018.
59. D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *RSS*, volume 2, 2016.
60. V. N. S. Samarasooriya and P. K. Varshney. A fuzzy modeling approach to decision fusion under uncertainty. In *MFI*, pages 788–795, Dec 1996.
61. A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. *arXiv preprint arXiv:1806.06498*, 2018.
62. W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.
63. K. Selyunin, S. Jaksic, T. Nguyen, C. Reidl, U. Hafner, E. Bartocci, D. Nickovic, and R. Grosu. Runtime monitoring with recovery of the sent communication protocol. In *CAV*, pages 336–355. Springer, 2017.

64. K. Selyunin, T. Nguyen, E. Bartocci, and R. Grosu. Applying runtime monitoring for automotive electronic development. In *ICRV*, pages 462–469. Springer, 2016.

65. G. Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

66. H. E. Soken and C. Hajiyev. Adaptive unscented kalman filter with multiple fading factors for pico satellite attitude estimation. In *RAST*, pages 541–546, June 2009.

67. S. Song, X. Hu, J. Yu, L. Bai, and L. Chen. Learning a deep motion planning model for autonomous driving. In *IV*, pages 1137–1142. IEEE, 2018.

68. J. Steinhardt, P. W. W. Koh, and P. S. Liang. Certified defenses for data poisoning attacks. In *NIPS*, 2017.

69. A. D. Tafti and N. Sadati. Novel adaptive kalman filtering and fuzzy track fusion approach for real time applications. In *IEEE IEA*, pages 120–125, June 2008.

70. E. Triantaphyllou. *Multi-Criteria Decision Making Methods*, pages 5–21. Springer US, Boston, MA, 2000.

71. R. Wang, C. Hu, F. Yan, and M. Chadli. Composite nonlinear feedback control for path following of four-wheel independently actuated autonomous ground vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):2063–2074, 2016.

72. K. Watanabe, E. Kang, C. Lin, and S. Shiraishi. Runtime monitoring for safety of intelligent vehicles. In *DAC*, pages 31:1–31:6, 2018.

73. G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.

74. M. Werling, S. Kammel, J. Ziegler, and L. Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *IJRR*, 31(3), 2012.

75. B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3D lidar point cloud. In *ICRA*, pages 1887–1893. IEEE, 2018.

76. M. Wu, H. Zeng, C. Wang, and H. Yu. Safety guard: Runtime enforcement for safety-critical cyber-physical systems. In *DAC*, pages 1–6. IEEE, 2017.

77. M. Wu, H. Zeng, C. Wang, and H. Yu. Safety guard: Runtime enforcement for safety-critical cyber-physical systems: Invited. In *DAC*, pages 84:1–84:6, 2017.

78. F. Wulff, B. Schäufele, O. Sawade, D. Becker, B. Henke, and I. Radusch. Early fusion of camera and lidar for robust road detection based on u-net fcn. In *IEEE IV*, pages 1426–1431. IEEE, 2018.

79. H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint*, 2017.

80. Y. Yang, F. Qiu, H. Li, L. Zhang, M.-L. Wang, and M.-Y. Fu. Large-scale 3D semantic mapping using stereo vision. *IJAC*, 15(2):194–206, 2018.

81. J. Yebes, L. Bergasa, and M. García-Garrido. Visual object recognition with 3D-aware features in kitti urban scenes. *Sensors*, 15(4):9228–9250, 2015.

82. Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *CVPR*, pages 8827–8836. IEEE, 2018.

83. Q. Zhang, Y. Yang, Y. N. Wu, and S.-C. Zhu. Interpreting CNNs via decision trees. *arXiv preprint [arXiv:1802.00127]* 2018.

84. A. Zyner, S. Worrall, and E. Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. *arXiv preprint [arXiv:1807.09995]* 2018.