Machine Learning Based Perception Architecture Design for Semi-autonomous Vehicles

Joydeep Dey and Sudeep Pasricha

1 Introduction

In 2021, it was reported that an estimated 31,730 people died in motor vehicle traffic crashed in the United States, representing an estimated increase of about 12 percent compared to 2020 [1]. By eliminating the possibility of human driving errors through automation, advanced driver assistance systems (ADAS) are becoming a critical component in modern vehicles, to help save lives, improve fuel efficiency, and enhance driving comfort. ADAS systems typically involve a 4-stage pipeline involving sequential execution of functions related to *perception*, decision, control, and actuation. An incorrect understanding of the environment by the perception system can make the entire system prone to erroneous decision making, which can result in accidents due to imprecise real-time control and actuation. This motivates the need for a reliable *perception architecture* that can mitigate errors at the source of the pipeline and improve safety in emerging semi-autonomous vehicles.

The standard SAE-J3016 effectively classifies the capabilities of a perception architecture supported by a vehicle according to their targeted level of autonomy. In general, an optimal vehicle perception architecture should consist of carefully defined location and orientation of each sensor selected from a heterogeneous suite of sensors (e.g., cameras, radars) to maximize environmental coverage in the combined field of view obtained from the sensors. In addition to ensuring accurate sensing via appropriate sensor placement, a high object detection rate and low false

J. Dey (✉)
Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA
e-mail: Joydeep.Dey@colostate.edu

S. Pasricha
Colorado State University, Fort Collins, CO, USA
e-mail: sudeep@colostate.edu

V. K. Kukkala, S. Pasricha (eds.), *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, https://doi.org/10.1007/978-3-031-28016-0_22

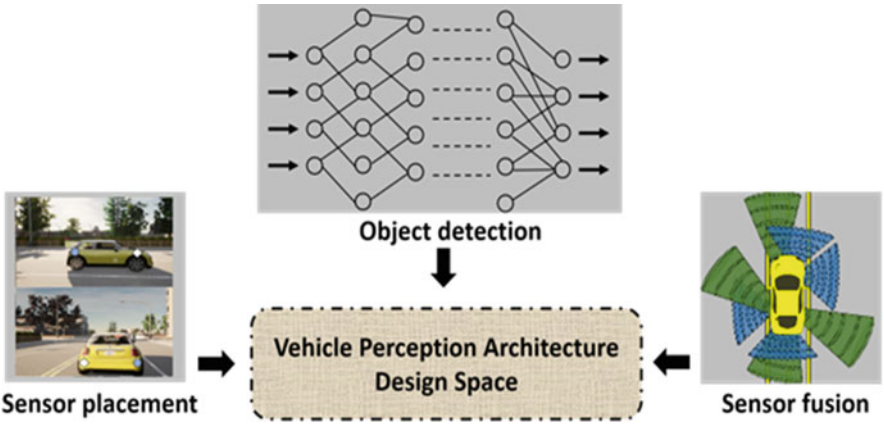


Fig. 1 Breakdown of perception architecture design space

positive detection rate needs to be maintained using efficient deep learning-based object detection and sensor fusion techniques.

State-of-the-art deep learning based object detection models are built with different network architectures, uncertainty modeling approaches, and test datasets over a wide range of evaluation metrics [2]. Object detectors that are capable of real time perception are resource-constrained by latency requirements, onboard memory capacity and computationally complexity. Optimizations performed to meet any one of these constraints often results in a trade-off with the performance of others [3]. As a result, comparison and selection from among the best set of deep learning based object detectors for perception applications remains a challenge.

In real-world driving scenarios, the position of obstacles and traffic are highly dynamic, so after detection of an object, tracking is necessary to predict its new position. Due to noise from various sources there is an inherent uncertainty associated with the measured position and velocity. This uncertainty is minimized by using sensor fusion algorithms [4]. An important challenge with sensor fusion algorithms is that the complexity of tracking objects increases as the objects get closer, due to a much lower margin for error (uncertainty) in the vicinity of the vehicle.

As summarized in Fig. 1, the design space of a vehicular perception architecture involves determining appropriate sensor selection and placement, object detection algorithms, and sensor fusion techniques. The possible configurations for each of these decisions is non-trivial and can easily lead to a combinatorial explosion of the design space, making exhaustive exploration impractical. Conversely, an optimization of each of these decisions individually before composing a final solution can lead to solutions that are sub-optimal and perform poorly in real environments. Perception architecture design depends heavily on the target features and use cases to be supported in the vehicle, making the already massive design space addressing the problem even larger and harder to traverse. Consequently,

today there are no generalized rules for the synthesis of perception architectures for vehicles.

In this chapter, we describe a framework called PASTA (*P*erception *A*rchitecture *S*earch *T*echnique for *A*DAS), first introduced in [37], to perform perception architecture synthesis for emerging semi-autonomous vehicles. Our experimental results indicate that the proposed framework is able to optimize perception performance across multiple ADAS metrics, for different vehicle types.

The main contributions in this chapter include:

- A global co-optimization framework capable of synthesizing robust vehicle-specific perception architecture solutions that include heterogeneous sensor placement, deep learning based object detector design, and sensor fusion algorithm selection;
- An exploration of various design space search algorithms tuned for the vehicle perception architecture search problem;
- A fast and efficient method for co-exploration of the deep learning object detector hyperparameters, through adaptive and iterative environment- and vehicle-specific transfer learning;
- A comparative analysis of the framework efficiency across different vehicle models (Audi TT, BMW Minicooper).

2 Related Work

State-of-the-art semi-autonomous vehicles require robust perception of their environment, for which the choice of sensor placement, object detection algorithms, and sensor fusion techniques are the most important decisions. These decisions are carefully curated to support ADAS features (e.g., blindspot warning, lane keep assist) that characterize the autonomy level to be supported by a vehicle under design.

Many prior works have explored vehicle perception system design with different combinations of sensor types to overcome limitations that plague individual sensor types. The work in [5] used a single camera-radar pair for perception of headway distance using a Continental radar mounted on the geometric center of the front bumper and a Nextbase 512G monocular camera behind the windscreen. Vehicle detection was performed on the collected camera frames, by sorting potential candidates in a fixed trapezoidal region of interest in the horizontal plane. In [5] a camera-radar fusion based perception architecture was proposed for target acquisition with the well-known SSD (Single Shot Detection) object detector on consecutive camera frames. This allowed their perception system to differentiate vehicles from pedestrians in real time. The detection accuracy was optimized with the use of a Kalman filter and Bayesian estimation, which reduced computational complexity compared to [5]. In [6] a single neural network was used for fusion of all camera and radar detections. The proposed neural fusion model (CRF-Net) used an optimized training strategy similar to the 'Dropout' technique, where all input

neurons for the camera data are simultaneously deactivated in random training steps, forcing the network to rely more on the radar data. The training focus towards radar overcame the bias introduced by starting with pre-trained weights from the feature extractor that was trained from the camera data. The work in [7] optimized merging camera detection with LiDAR processing. An efficient clustering technique inspired by the DBSCAN algorithm allowed for a better exploitation of features from the raw LiDAR point cloud. A fusion scheme was then used to sequentially merge the 2D detections made by a YOLOv3 object detector using cylindrical projection with the detections made from clustered LiDAR point cloud data. In [8], an approach to fuse LiDAR and stereo camera data was proposed, with a post-processing method for accurate depth estimation based on a patch-wise depth correction approach. In contrast to the cylindrical projection of 2D detections in [7], the work in [8] uses a projection of 3D LiDAR points into the camera image frame instead, which upsamples the projection image, creating a more dense depth map.

All of the prior works discussed above optimize vehicle perception performance for rigid combinations of sensors and object detectors, without any design space exploration. Only a few prior works have (partially) explored the design space of sensors and object detectors for vehicle perception. An approach for optimal positioning and calibration of a three LiDAR system was proposed in [9]. The approach used a neural network to learn and qualify the effectiveness of different LiDAR location and orientations. The work in [10] proposed a sensor selection and exploration approach based on factor graphs during multi-sensor fusion. The work in [11] heuristically explored a subset of backbone networks in the Faster R-CNN object detector for perception systems in vehicles. The work in [12] presented a framework that used a genetic algorithm to optimize sensor orientations and placements in vehicles.

The optimized perception techniques discussed in [5–12] provide highly accurate detections which enable design of efficient energy management strategies for ADAS. The work in [13] derives a prediction mechanism for optimal energy management for ADAS using a nonlinear autoregressive artificial neural network (NARX). Multiple sources are used as input to the neural network such as data from drive cycle information, current vehicle state, global positioning system, travel time data and detected obstacles. In addition, dynamic programming is used to derive an optimal energy management control strategy which shows significant fuel economy improvements compared to highly accurate predictive baseline models. The work in [14] proposes a predictive optimal energy management strategy that leverages sensor data aggregation and dynamic programming to achieve vehicle fuel economy improvement for ADAS compared to existing vehicle control strategies. The work discussed in [13, 14] leverage existing ADAS technology in modern vehicles to realize prediction based optimal energy management, which enables fuel economy improvements for ADAS with minor modifications.

Unlike prior works that fine-tune specific perception architectures, e.g., [5–8]*, or explore the sensing and object detector configurations separately, e.g* [9–12]*., this chapter proposes a holistic framework that jointly co-optimizes heterogeneous*

sensor placement, object detection algorithms, and sensor fusion techniques. To the best of our knowledge, this is the first effort that performs co-optimization across such a comprehensive decision space to optimize ADAS perception, with the ability to be tuned and deployed across multiple vehicle types.

3 Background

3.1 ADAS Level 2 Autonomy Features

In this chapter, our exploration of perception architectures on a vehicle, henceforth referred to as an *ego vehicle*, targets four ADAS features that have varying degrees of longitudinal (i.e., in the same lane as the ego vehicle) and lateral (i.e., in neighboring lanes to the ego vehicle lane) sensing requirements. The SAE-J3016 standard [15] defines adaptive cruise control (ACC) and lane keep assist (LKA) individually as level 1 features, as they only perform the dynamic driving task in either the latitudinal or longitudinal direction of the vehicle. Forward collision warning (FCW) and blindspot warning (BW) are defined in SAE-J3016 as level 0 active safety systems, as they only enhance the performance of the driver without performing any portion of the dynamic driving task. However, when all four features are combined, the system can be described as a level 2 autonomy system. Figure 2 shows an overview of the four features we focus on for level 2 autonomy, which are discussed next.

While modern ACC systems differ in their implementation and perception architectures, they take perform longitudinal control operations instead of the driver. The challenge in ACC is to maintain an accurate track of the lead vehicle (immediately ahead of the ego vehicle in the same lane) with a forward facing sensor and using longitudinal control to maintain the specified distance while maintaining driver comfort (e.g., avoiding sudden velocity changes). **LKA** (lane keep assist) systems determine whether the ego vehicle is drifting towards any lane boundaries and are an evolution of lane departure warning systems. LKA systems have been known to over-compensate, creating a "ping-pong" effect where the vehicle oscillates back and forth between the lane lines [16]. The main challenges in LKA are to reduce this ping-pong effect and the accurate detection of lane lines on obscured (e.g., snow covered) roads. **FCW** (forward collision warning) systems are used for real-time prediction of collisions with a lead vehicle. A critical requirement for FCW systems is that they avoid false positives and false negatives to improve driver comfort, safety and reduce rear end accidents [17]. Lastly, **BW** (blindspot warning) systems use lateral sensor data to determine whether there is a vehicle towards the rear on either side of the ego vehicle (Fig. 2) in a location the driver cannot see with their side mirrors. *A perception architecture designed to support Level 2 autonomy in a vehicle should support all four of these critical features.*

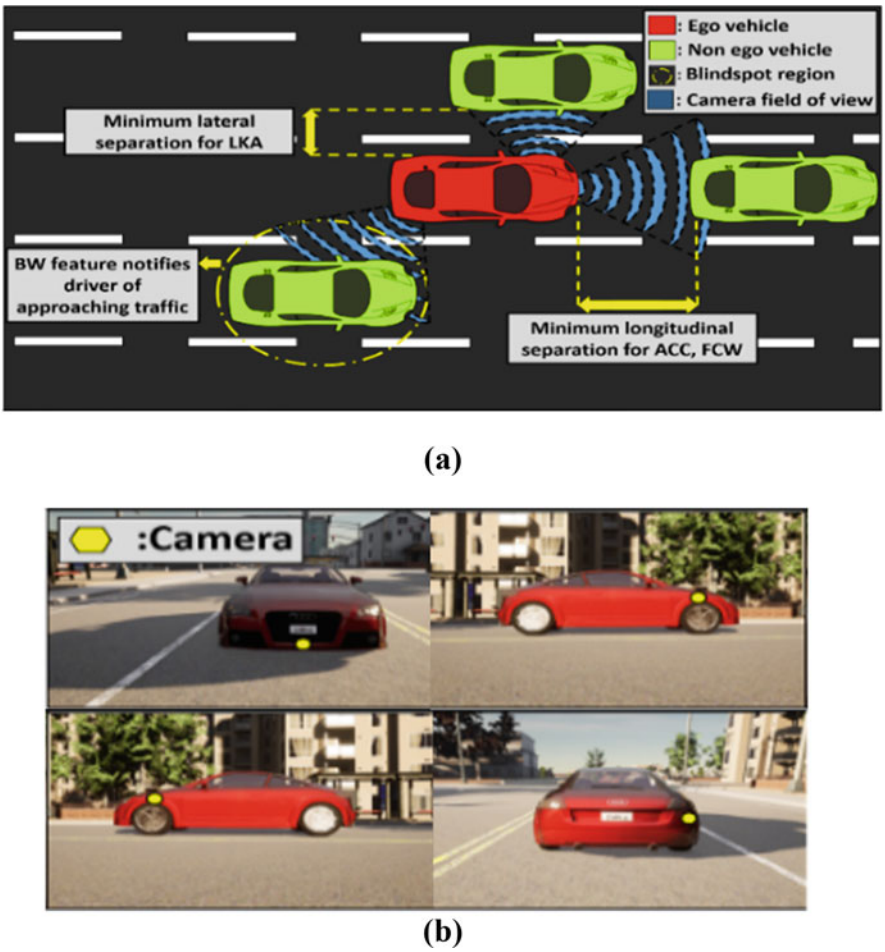


Fig. 2 Visualization of common scenarios in ACC, FCW, LKA, and BW

3.2 Sensor Placement and Orientation

In order to capture data most relevant to each feature, a strategic sensor placement strategy must be used on the ego vehicle such that the chosen position and orientation of selected sensors maximize coverage (of the vehicle environment). Figure 2 visualizes an example of field of view coverage (in blue) corresponding to three unique placements of camera sensors on the body of the ego vehicle (in yellow, lower images) to meet coverage goals. For the ACC and FCW features, the ego vehicle is responsible for slowing down to maintain a minimum separation between the ego and lead vehicle. The camera must be positioned somewhere on the front bumper to measure minimum longitudinal separation accurately while keeping

the lead vehicle in the desired field of view. For LKA, there is a need to maintain a safe minimum lateral distance between non-ego vehicles in neighboring lanes. Here a front camera is needed to extract lane line information, while side cameras are required for tracking this minimum lateral separation. As BW requires information about a specific area near the rear of the vehicle, it is a challenge to find an optimal sensor placement that maximizes the view of the blind spot. If the sensor is too far forward or too far back, it will miss key portions of the blind spots. Beyond placement, the orientation of sensors can also significantly impact coverage for all features [17]. Thus sensor placement and orientation remains a challenging problem.

3.3 Object Detection for Vehicle Environment Perception

The two broad goals associated with deep learning based object detection are: determining spatial information (relative position of an object in the image) via *localization* followed by identifying which category that object instance belongs to via *classification* [18]. As an example, Fig. 3 shows object detection of multiple car instances (using the YOLOv3 deep learning based object detector [19]) by creating a bounding box around the 'car' object instances and predicting the object class as 'car'. The pipeline of traditional object detection models can be divided into informative region selection, feature extraction, and classification [20]. Depending on which subset of these steps are used to process an input image frame, object detectors are classified as single-stage or two-stage.

Modern single-stage detectors are typically composed of a feed-forward fully convolutional network that outputs object classification probabilities and box offsets (w.r.t. pre-defined anchor/bounding boxes) at each spatial position. The YOLO family of object detectors is a popular example of single-stage detectors [17]. SSD (single shot detection) is another example, based on the VGG-16 backbone [21]. An advantageous property of single-stage detectors is their very high detection throughput (e.g., ~40 frames per second with YOLO) that makes them suitable for real time scenarios. Two-stage detectors divide the detection process into separate region proposal and classification stages. The first stage involves identification of

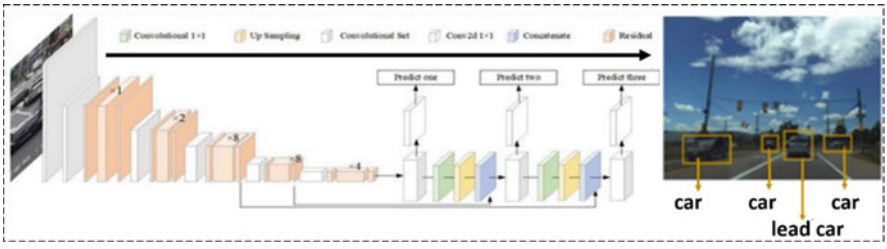


Fig. 3 Example of vehicle (object) detection with YOLOv3

several regions in an image that have a high probability to contain an object using a region proposal network (RPN). In the second stage, proposals of identified regions are fed into convolutional networks for classification. Region-based CNN (R-CNN) is an example of a two-stage detector [22]. R-CNN divides an input image into 2000 regions generated through a selective search algorithm, after which the selected regions are fed to a CNN for feature extraction followed by a Support Vector Machine (SVM) for classification. Fast R-CNN [23] and subsequently Faster R-CNN [24] improved the speed of training as well as detection accuracy compared to R-CNN by streamlining the stages.

Two-stage detectors have high localization and object recognition accuracy, whereas one-stage detectors achieve higher inference speed [25]. *In this chapter, we considered both types of object detectors to exploit the latency/accuracy tradeoffs during perception architecture synthesis.*

3.4 Sensor Fusion

Perception architectures that use multiple sensors in their sensing framework often must deal with errors due to imprecise measurements from one or more of the sensors. Conversely, errors can also arise when only a single sensor is used due to measurement uncertainties from insufficient spatial *(occlusion)* or temporal (*delayed sensor response time*) coverage of the environment. The Kalman filter is one of the most widely used sensor fusion state estimation algorithms that enables error-resilient tracking of targets [26]. The Kalman filter family is a set of recursive mathematical equations that provides an efficient computational solution of the least-squares method for estimation. The filters in this family have the ability to obtain optimal statistical estimations when the system state is described as a linear model and the error can be modeled as Gaussian noise. If the system state is represented as a nonlinear dynamic model as opposed to a linear model, a modified version of the Kalman filter known as the Extended Kalman Filter (EKF) can be used, which provides an optimal approach for implementing nonlinear recursive filters [27]. However, for real time ADAS operations the computation of the Jacobian (matrix describing the system state) in EKF can be computationally expensive and contribute to measurement latency. Further, any attempts to reduce the cost through techniques like linearization makes the performance unstable [28]. The unscented Kalman filter (UKF) is another alternative that has the desirable property of being more amenable to parallel implementation [29]. *In our design space exploration of perception architecture, we explore the family of Kalman filters as candidates for sensor fusion.*

4 PASTA Architecture

4.1 Overview

Figure 4 presents a high-level overview of our proposed *PASTA* framework. The heterogeneous sensors, object detection model library, sensor fusion algorithm library, and physical dimensions of the vehicle model are inputs to the framework. An algorithmic design space exploration is used to generate a perception architecture solution which is subsequently evaluated based on a cumulative score from performance metrics relevant to the ADAS autonomy level being targeted. As part of the framework, we evaluate the search efficacy of three design space search exploration algorithms: genetic algorithm (GA), differential evolution (DE), and the firefly algorithm (FA). The process of perception architecture generation and evaluation iterates until an algorithm-specific stopping criteria is met, at which point the best design points are output. The following subsections describe each component of our framework in detail.

4.2 Problem Formulation and Metrics

In our framework, for a given vehicle, a *design point* is defined as a perception architecture that is a combination of three components: a *sensor configuration* which involves the fixed deployment position and orientation of each sensor selected for the vehicle, an *object detector algorithm*, and a *sensor fusion algorithm*. The goal is to find an optimal design point for the given vehicle that minimizes the cumulative

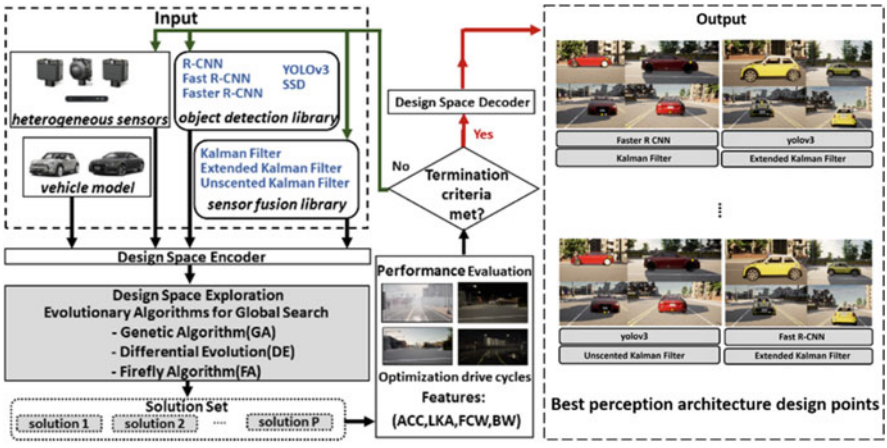


Fig. 4 An overview of the proposed PASTA framework

error across eight metrics that are characteristic of the ability to track and detect non-ego vehicles across road geometries and traffic scenarios.

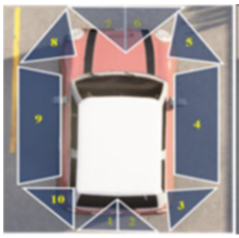
The eight selected metrics are related to our goal of supporting level 2 autonomy with the perception architecture. In the descriptions of the metrics below, the ground truth refers to the actual position of the non-ego vehicles (traffic in the environment of the ego vehicle). The metrics can be summarized as: (1) *longitudinal position error and* (2) *lateral position error:* deviation of the detected positional data from the ground truth of non-ego vehicle positions along the y and x axes, respectively; (3) *object occlusion rate:* the fraction of passing non-ego vehicles that go undetected in the vicinity of the ego vehicle; (4) *velocity uncertainty:* the fraction of times that the velocity of a non-ego vehicle is measured incorrectly; (5) *rate of late detection:* the fraction of the number of 'late' non-ego vehicle detections made over the total number of non-ego vehicles. Late detection is one that occurs after a non-ego vehicle crosses the minimum safe longitudinal or lateral distance, as defined by Intel RSS safety models for pre-crash scenarios GA is a popular evolutionary algorithm that can solve optimization problems by mimicking the process of natural selection [30].2. This metric directly factors in the trade-off between latency and accuracy for object detector and fusion algorithms; (6) *false positive lane detection rate:* the fraction of instances when a lane marker is detected but there exists no ground truth lane; (7) *false negative lane detection rate:* the fraction of instances when a ground truth lane exists but is not detected; and (8) *false positive object detection rate:* the fraction of total vehicle detections which were classified as non-ego vehicle detections but did not actually exist.

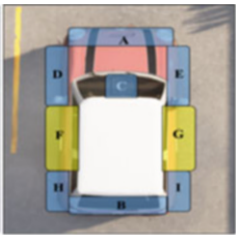
4.3 Design Space Encoder/Decoder

The design space encoder receives a set of random initial design points as input which are expressed as a vector. This encoded format is best suited for various kinds of rearrangement and splitting operations during design space exploration. The encoder adapts the initial selection of inputs for our design space such that a design point is defined by the location and orientation of each sensor's configuration (consisting of six parameters: x, y, z, roll, pitch, and yaw), together with the object detector and fusion algorithm. The design space decoder converts the solutions into the same format as the input so that the output perception architecture solution(s) found can be visualized with respect to the real-world co-ordinate system.

4.4 Design Space Exploration

The goal of a design space exploration algorithm in our framework is to generate perception architectures (design points) which are aware of feature to field of view (FOV) zone correlations around an ego vehicle. Figure 5a shows the 10 primary





Feature	Region	Zone
BW	B, H, I	1, 2, 3,10
LKA	E, I	3,4,5
	D, H	8, 9, 10
ACC , FCW	A, B,C	6, 7

(a) **(b)** **(c)**

Fig. 5 (**a**) Field of view (FOV) zones; (**b**) sensor placement regions; (**c**) feature, region, and zone relationship

FOV zones around the ego-vehicle. These zones of interest are defined as the most important perception areas in the environment for a particular ADAS feature. Figure 5b shows the regions on the vehicle on which sensors can be mounted (in blue). Regions F and G (in yellow) are exempt from sensor placement due to the mechanical instability of placing sensors on the door of a vehicle. The correlation between ADAS features, zones, and regions, is shown in Fig. 5c. For exploration of possible locations within a region, a fixed step size of 2 cm in two dimensions across the surface of the vehicle is considered, which generates a 2D grid of possible positions in each zone shown in Fig. 5b. The orientation exploration of each sensor involves rotation at a fixed step size of 1 degree between an upper and lower bounding limit for roll, pitch, and yaw respectively, at each of these possible positions within the 2D grid. The orientation exploration limits were chosen with caution with the caveat that some sensors, such as long range radars, have an elevated number of recorded false positives with extreme orientations.

To get a sense of the design space, consider four sensors (e.g., two cameras and two radars). Just the determination of the optimal placement and orientation of these sensors involves exploring ${}^{1.24e+26}C_4$ and ${}^{7.34e+25}C_4$ configurations for the Audi-TT and BMW-Minicooper vehicles, respectively. Coupled with the choice of different object detectors and sensor fusion algorithms, the resulting massive design space cannot be exhaustively traversed in a practical amount of time, necessitating the use of intelligent design space search algorithms that support hill climbing to escape local minima. In our framework, we explored three evolutionary algorithms: (1) *Genetic Algorithm (GA)*, (2) *Differential Evolution (DE)*, and the (3) *Firefly Algorithm (FA)*. As shown in Fig. 4, each algorithm generates a solution set of size *'P'* at every iteration until the termination criteria is met. The algorithms simultaneously co-optimize sensor configuration, object detection, and sensor fusion, and proceed to explore new regions of the design space when the termination (perception) criteria is not met. We briefly describe the three algorithms below.

4.4.1 Genetic Algorithm (GA)

GA is a popular evolutionary algorithm that can solve optimization problems by mimicking the process of natural selection [30]. Initially, the GA randomly selects a solution set of fixed size referred to as the population and then improves the quality of the candidate solutions in each iteration by modifying them using various GA operations. GA has the ability to optimize problems where the design space is discontinuous and also if the cost function is non differentiable. In our GA implementation, in the selection stage, the cost function values are computed for 50 design points at a time, and a roulette wheel selection method is used to select which set of chromosomes will be involved in the crossover step based on their cost function probability value (fraction of the cumulative cost function sum of all chromosomes considered in the selection). In the crossover stage, the crossover parameter is set to 0.5, allowing half of the 50 chromosomes to produce offspring. The mutation parameter is set to 0.2 which determines the new genes allowed for mutation in each iteration.

4.4.2 Differential Evolution (DE)

Differential Evolution (DE) [31] is another stochastic population-based evolutionary algorithm that takes a unique approach to mutation and recombination. An initial solution population of fixed size is selected randomly, and each solution undergoes mutation and then recombination operations. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector to achieve difference vector-based mutation. Next, crossover is performed, where the mutated vector's parameters are mixed with the parameters of another predetermined vector, the target vector, to yield a trial vector. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the next generation. To ensure that better solutions are selected only after generation of all trial vectors at every iteration, greedy selection is performed between the target vector and trial vector. Unlike GA where parents are selected based on fitness, every solution in DE takes turns to be one of the parents [30]. In our DE implementation, we set initial population size to 50 and use a crossover probability of 0.8 to select candidates participating in crossover.

4.4.3 Firefly Algorithm (FA)

FA is a swarm-based metaheuristic [32] that has shown superior performance compared to GA for certain problems [33]. In FA, a solution is referred to as a firefly. The algorithm mimics how fireflies interact using flashing lights (bioluminescence). The algorithm assumes that the attractiveness of a firefly is directly proportional to its brightness which depends on the fitness function value. Further, a given firefly can be attracted by any other firefly in the design space irrespective of the gender of

both. Initially, a random solution set is generated and the fitness (brightness) of each candidate solution is measured. In the design space, a firefly is attracted to another with higher brightness (more fit solution), with brightness decreasing exponentially over distance. FA is significantly different from DE and GA, as both exploration of new solutions and exploitation of existing solutions to find better solutions is achieved using a single position update step.

4.5 Performance Evaluation

Each iteration of the design space exploration involves performance evaluation of the generated solution set where each design point undergoes multiple drive cycles. A drive cycle here refers to a virtual simulation involving an ego-vehicle (with a perception architecture under evaluation) following a fixed set of waypoint co-ordinates, while performing object detection and sensor fusion on the environment and other non-ego vehicles. A total of 20 different drive cycles were considered, with 5 drive cycles customized for each ADAS feature. As an example, drive cycles for ACC and FCW involve an ego vehicle following different lead vehicles at different distances, velocities, weather conditions, and traffic profiles. The fitness of the perception architectures generated by the framework are computed using the cumulative metric scores (Sect. 4.2) across the drive cycles.

5 Experiments

5.1 Experimental Setup

To evaluate the efficacy of the PASTA framework we performed experiments in the open-source simulator CARLA (Car Learning to Act) implemented as a layer on Unreal Engine 4 (UE4) [34]. The UE4 engine provides state-of-the-art physics rendering for highly realistic driving scenarios. We leveraged this tool to design a variety of drive cycles that are roughly 5 min long and contain scenarios that commonly arise in real driving environments, including adverse weather conditions (rain, fog) and a few overtly aggressive/conservative driving styles observed with vehicles. To ensure generalizability, we consider a separate set of test drive cycles to evaluate solution quality, which are different from the optimization drive cycles used iteratively by the framework to generate optimized perception architecture solutions.

We target generating perception architectures to meet level 2 autonomy goals for two vehicle models: Audi-TT and BMW-Minicooper (Fig. 6). A maximum of 4 mid-range radars and 4 RGB cameras are considered in the design space, where

BMW Minicooper

Length	Width	Height	Wheelbase
3.835	1.727	1.414	2.495

(a)

Audi TT

Length	Width	Height	Wheelbase
4.177	1.832	1.353	2.505

(b)

Fig. 6 BMW Minicooper (top) and Audi TT (bottom)

each sensor can be placed in any zone (Fig. 5a, b). Using a greater number of these sensors led to negligible improvements for the level 2 autonomy goal. The RGB cameras possess 90° field of view, 200 fps shutter speed, and image resolution of 800 × 600 pixels. The mid-range radars selected generate a maximum of 1500 measurements per second with a horizontal and vertical field of view of 30° and a maximum detection distance of 100 m. We considered 5 different object detectors (YOLOv3, SSD, R-CNN, Fast R-CNN, and Faster R-CNN) and 3 sensor fusion algorithms (Kalman filter, Extended Kalman filter, and Unscented Kalman filter). For the design space exploration algorithms, the cost function was a weighted sum across the eight metrics discussed in Sect. 4.2, with the weight factor for each metric chosen on the basis of their total feature-wise cardinality across all zones shown in Fig. 5c. During design space exploration, if the change in average cost function value was <5% over 250 iterations, the search was terminated. All algorithmic exploration was performed on an AMD Ryzen 7 3800X 8-Core CPU desktop with an NVIDIA GeForce RTX 2080 Ti GPU.

Table 1 Object detector latency and accuracy comparison

Object detector	Latency GPU (ms)	Latency CPU (ms)	mAP(%)
R-CNN	48956.18	66090.83	73.86
Fast R-CNN	1834.71	2365.86	76.81
Faster R-CNN	176.99	286.72	79.63
SSD	53.25	70.32	70.58
YOLOv3	24.03	32.92	71.86

5.2 Experimental Results

In the first experiment, we explored the inference latency and accuracy in terms of mean average precision (mAP) for the five different object detectors considered in this chapter. Table 1 summarizes the inference latency on a CPU and GPU, as well as the accuracy in mAP for the object detectors on images from our analyzed drive cycles, with all detectors trained on the MS-COCO dataset. It can be observed that the two-stage detectors (R-CNN, Fast R-CNN, and Faster R-CNN have a higher accuracy than the single stage detectors (SSD, YOLOv3). However, the inference time for the two-stage detector is significantly higher than for the single stage detectors. For real-time object detection in vehicles, it is crucial to be able to detect objects with low latency, typically less than 100 ms [35]. As a result, single stage detectors are preferable, with YOLOv3 achieving slightly better accuracy and lower inference time than SSD. However, in some scenarios, delayed detection can still be better than not detecting or wrongly detecting an object (e.g., slightly late blindspot warning is still better than receiving no warning) in which case the slower but more accurate two-stage detectors may still be preferable. Our PASTA framework is aware of this inherent trade-off and factors in the detection accuracy and rate of late detection in performance evaluation metrics (Sect. 4.2) to explore both single-stage and two-stage detectors. Also, detectors with a higher mAP value sometimes did not detect objects that other detectors with a lower mAP were able to; thus, we consider all five detectors in our exploration.

Next, we explored the importance of global co-optimization for our problem. We select the genetic algorithm (GA) variant of our framework to explore the entire design space (GA-PASTA) and compared it against five other frameworks. Frameworks GA-PO and GA-OP use the GA but perform a local (sequential) search for sensor design. In GA-PO, sensor position is explored before orientation, while in GA-OP the orientation for fixed sensor locations (based on industry best practices) is explored before adjusting sensor positions. For both frameworks, the object detector used was fixed to YOLOv3 due to its sub-100 ms inference latency and reasonable accuracy, while the extended Kalman filter (EKF) was used for sensor fusion due to its ability to efficiently track targets following linear or non-linear trajectories. The framework GA-VESPA is from prior work [12] and uses GA for exploration across sensor positions and orientations simultaneously, with the YOLOv3 object detector and EKF fusion algorithm. Frameworks GA-POD and GA-POF use GA for a more

comprehensive exploration of the design space. GA-POD simultaneously explores the sensor positioning, orientation, and object detectors, with a fixed EKF fusion algorithm. GA-POF simultaneously explores the sensor positioning, orientation, and sensor fusion algorithm, with a fixed YOLOv3 fusion algorithm.

Figure 7a depicts the average cost of solution populations (lower is better) for the BMW-Minicooper across the different frameworks plotted against the number of iterations, with each exploration lasting between 80–100 h. It can be observed that GA-PO performs better than GA-OP, which confirms the intuitive importance of exploring sensor positioning before adjusting sensor orientations. GA-VESPA outperforms both GA-PO and GA-OP, highlighting the benefit of co-exploration of sensor position and orientation over a local sequential search approach used in GA-PO and GA-OP. GA-POD and GA-POF in turn outperform these frameworks, indicating that decisions related to object detection and sensor fusion can have a notable impact on perception quality. GA-POD terminates with its solution set having a lower average cost than GA-POF, which indicates that co-exploration of object detection and sensor placement/orientation is slightly more effective than co-exploration of sensor fusion and sensor placement/orientation. Our proposed GA-PASTA framework achieves the lowest average cost solution, highlighting the tremendous benefit that can be achieved from co-exploring sensor position/orientation, object detection, and sensor fusion algorithms. Figure 7b summarizes the objective function cost of the best solution found by each framework, which aligns with the population-level observations from Fig. 7a.

The comparative analysis for the BMW-Minicooper was repeated three times with different initializations for all six frameworks, and the results for the other two runs show a consistent trend with the one shown in Fig. 7. Note also that the relative trend across frameworks observed for the Audi-TT is similar to that observed for the BMW-Minicooper, and thus the results for the Audi TT are omitted for brevity.

In the next experiment, we explored the efficacy of different design space exploration algorithms (GA, DE, and FA; see Sect. 4.4) to determine which algorithm can provide optimal perception architecture solutions across varying vehicle models. Figure 8 shows the results for the three variants of the PASTA framework, for the Audi-TT and BMW-Minicooper vehicles. The best solution was selected across three runs of each algorithmic variant (variations for the best solution across runs are highlighted with confidence intervals, with bars indicating the median). It can be seen that for both considered vehicle models the FA algorithm outperforms the DE and GA algorithms. For Audi-TT, the best solution found by FA improves upon the best solution found with DE and GA by 18.34% and 14.84%, respectively. For the BMW-Minicooper the best solution found by FA outperforms the best solution found by DE and GA by 3.16% and 13.08%, respectively. Figure 9a depicts the specific sensor placement locations for each vehicle type, with a visualization of sensor coverage for the best solutions found by each algorithm shown in Fig. 9b.

Finally, in our quest to further improve perception architecture synthesis in PASTA, we focused on a more nuanced exploration of the object detector design space. We selected the FA search algorithm due to its superior performance over

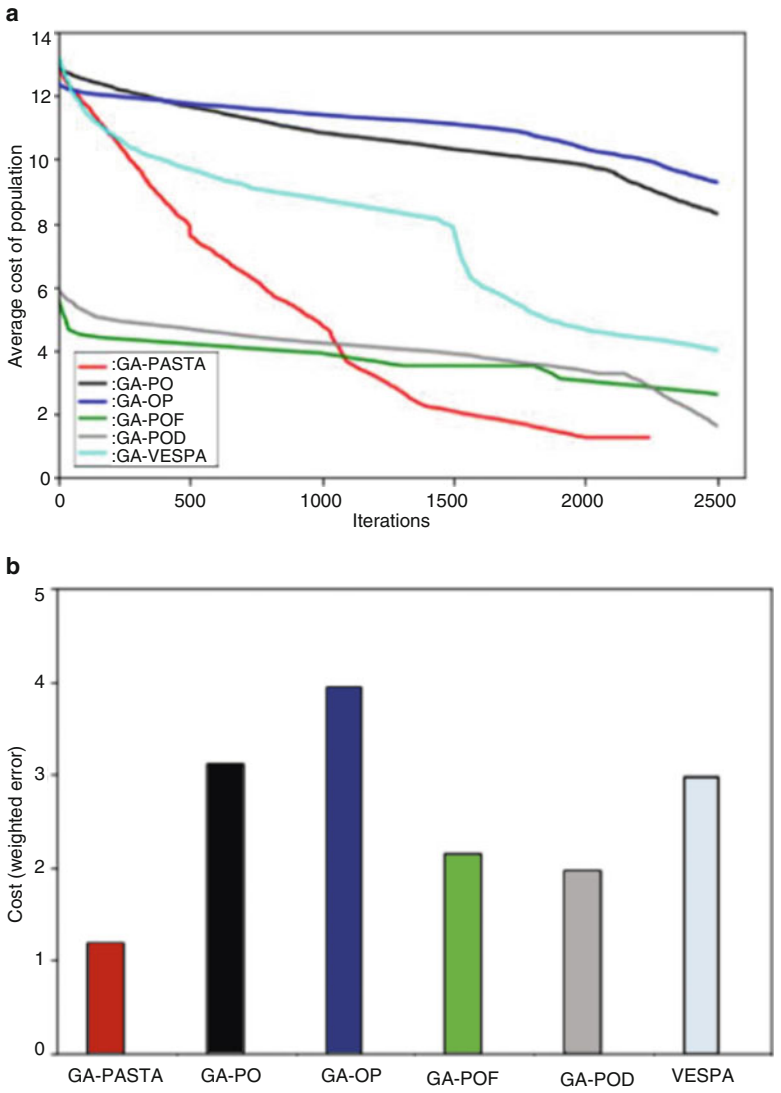


Fig. 7 (**a**) Comparison of perception architecture exploration frameworks; (**b**) Cost of best solution from each framework

GA and DE, and modified FA-PASTA to integrate a neural architecture search (NAS) for the YOLOv3 object detector, with the aim of further improving YOLOv3 accuracy across drive cycles while maintaining its low detection latency. Our NAS for YOLOv3 involved transfer learning to retrain network layers with a dataset consisting of 6000 images obtained from the KITTI dataset, using the open source tool CADET [36]. The NAS hyperparameters that were explored involved

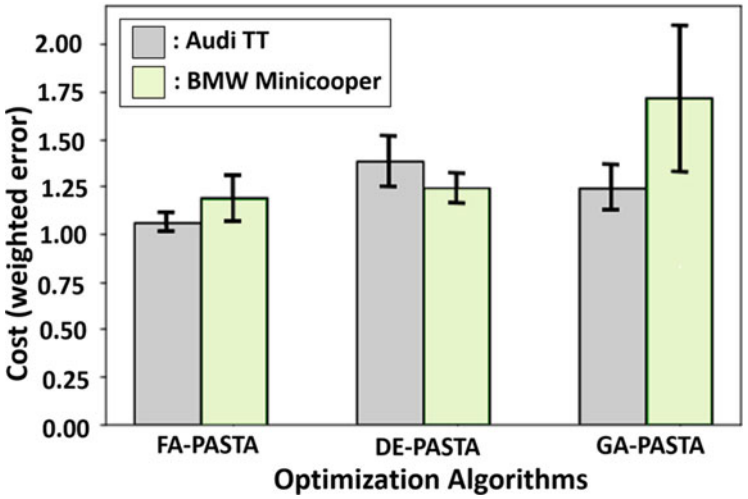


Fig. 8 Comparison of three variants of PASTA framework with genetic algorithm (GA), differential evolution (DE), and Firefly algorithm (FA)

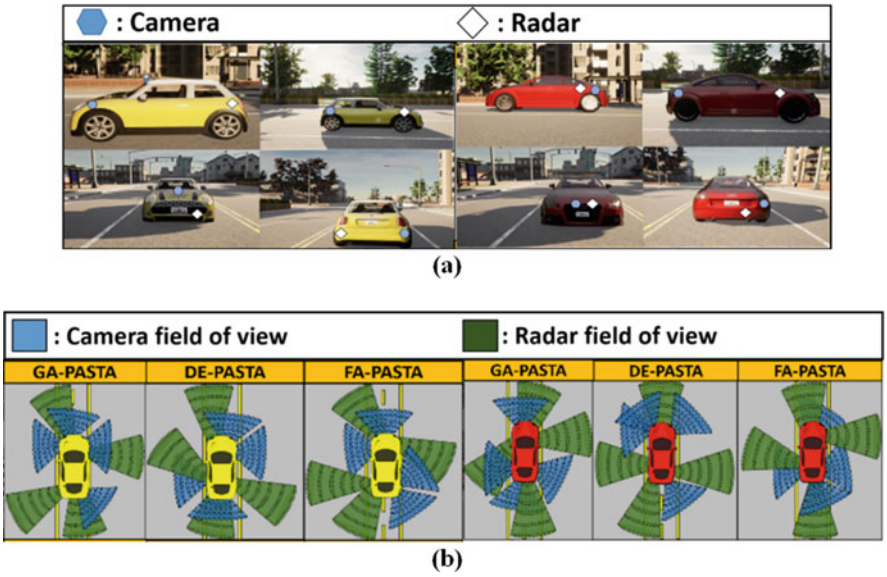


Fig. 9 (**a**) Sensor placement for best solution found with FA algorithm (top yellow vehicle: BMW-Minicooper, bottom red vehicle: Audi-TT) (top); (**b**) Sensor coverage for best solutions found by GA, DE, and FA search algorithms (bottom)

the number of layers to unfreeze and retrain (from a total of 53 layers in the Darknet-53 backbone used in YOLOv3; Fig. 10a), along with the optimizer learning rate, momentum, and decay. The updated variant of our framework, FA-NAS-

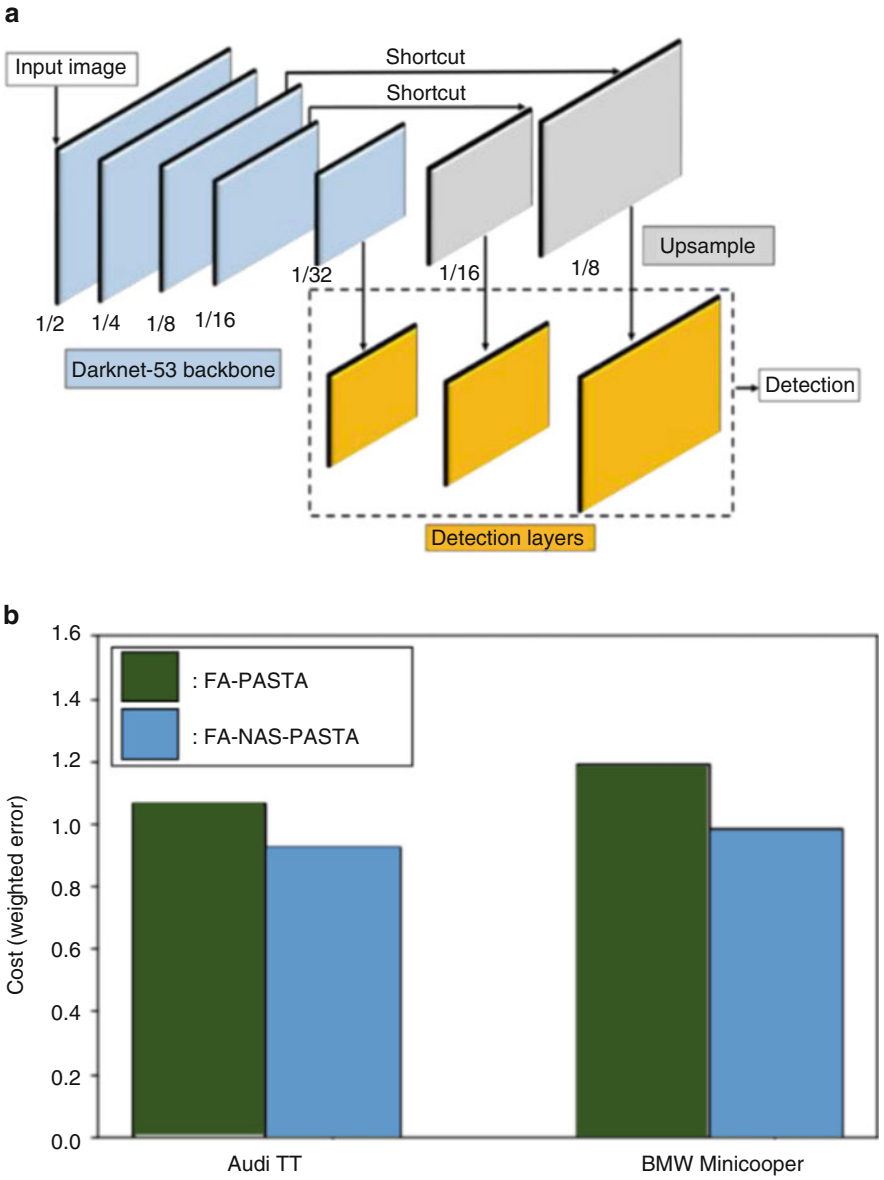


Fig. 10 (**a**) YOLOv3 object detector architecture with Darknet-53 backbone network that was fine-tuned using neural architecture search (NAS); (**b**) results of integrating object detector NAS with PASTA

PASTA, considered these YOLOv3 hyperparameters along with the sensor positions and orientations, and sensor fusion algorithms, during iterative evolution of the population of candidate solutions in the FA algorithm.

Figure 10b shows the results of this analysis for the two vehicles considered. FA-PASTA is the best performing variant of our framework (from Fig. 8), while FA-NAS-PASTA is the modified variant that integrates NAS for YOLOv3. It can be observed that fine tuning the YOLOv3 object detector during search space exploration in FA-NAS-PASTA leads to notable improvements in the best perception architecture solution, with up to 14.43% and 21.13% improvement in performance for the Audi-TT and BMW-Minicooper, compared to PASTA-FA.

6 Conclusions

In this chapter, we propose an automated framework called *PASTA* that is capable of generating perception architecture designs for modern semi-autonomous vehicles. *PASTA* has the ability to simultaneously co-optimize locations and orientations for sensors, optimize object detectors, and select sensor fusion algorithms for a given target vehicle. Our experimental analysis showed how PASTA can synthesize optimized perception architecture solutions for the Audi TT and BMW Minicooper vehicles, while outperforming multiple semi-global exploration techniques. Integrating neural architecture search for the object detector in PASTA shows further promising improvements in solution quality. Our future work will explore how to integrate PASTA with machine learning based techniques for anomaly detection [38–43] and robust vehicle network scheduling [42–44] in semi-autonomous vehicles.

References

1. NHTSA (National Highway Traffic Safety Administration), National Center for Statistics and Analysis, “Data Estimates Indicate Traffic Fatalities Continued to Rise at Record Pace in First Nine Months of 2021” (2022)
2. Gupta, A., Anpalagan, A., Guan, L., Khwaja, A.S.: Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array. **10**, 100057 (2021)
3. Feng, D., Harakeh, A., Waslander, S.L., Dietmayer, K.: A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving, vol. 23, pp. 9961–9980. IEEE Transactions on Intelligent Transportation Systems (2021)
4. Kukkala, V., Tunnell, J., Pasricha, S.: Advanced driver assistance systems: A path toward autonomous vehicles. IEEE Cons. Electron. **7**(5) (2018)
5. Zhexiang, Y., Jie, B., Sihan, C., Libo, H., Xin, B.: Camera-Radar Data Fusion for Target Detection via Kalman Filter and Bayesian Estimation. SAE Technical Paper (2018)
6. Nobis, F., Geisslinger, M., Weber, M., Betz, J., Lienkamp, M.: A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. IEEE Sensor Data Fusion: Trends, Solutions, Applications (SDF) (2020)
7. Verucchi, M., Bartoli, L., Bagni, F., Gatti, F., Burgio, P., Bertogna, M.: Real-time Clustering and LiDAR-Camera Fusion on Embedded Platforms for Self-Driving Cars. IEEE International Conference on Robotic Computing (IRC) (2020)

8. Meng, L., Yang, L., Tang, G., Ren, S., Yang, W.: An Optimization of Deep Sensor Fusion Based on Generalized Intersection Over Union. International Conference on Algorithms and Architectures for Parallel Processing, Springer (2020)
9. Meadows, W., Hudson, C., Goodin, C., Dabbiru, L., Powell, B., Doude, M., Carruth, D., Islam, M., Ball, J.E., Tang, B.: Multi-LIDAR Placement, Calibration, Co-registration, and Processing on a Subaru Forester for off-Road Autonomous Vehicles Operations. Autonomous Systems: Sensors, Processing and Security for Vehicles and Infrastructure (2019)
10. Chen, H., Ling, P., Danping, Z., Kun, L., Yexuan, L., Yu, C.: An Optimal Selection of Sensors in Multi-Sensor Fusion Navigation with Factor Graph. Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS) (2018)
11. Ji-Qing, L., Sheng Fang, H., Shao, F., Zhong, Y., Hua, X.: Multi-scale Traffic Vehicle Detection Based on Faster R–CNN with NAS Optimization and Feature Enrichment. Defence Technology (2021)
12. Dey, J., Taylor, W., Pasricha, S.: VESPA: A Framework for Optimizing Heterogeneous Sensor Placement and Orientation for Autonomous vehicles. IEEE Consumer Electronics Magazine (2020)
13. Asher, Z., Tunnell, J., Baker, D.A., Fitzgerald, R.J., Banaei-Kashani, F., Pasricha, S., Bradley, T.H.: Enabling Prediction for Optimal Fuel Economy Vehicle Control. SAE International (2018)
14. Tunnell, J., Asher, Z., Pasricha, S., Bradley, T.H.: Towards Improving Vehicle Fuel Economy with ADAS. SAE International (2018)
15. SAE International Standard J3016, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles" (2018)
16. C. Kirchner, "Lane keeping assist explained," Motor review, [Online]. Available: https://motorreview.com/lane-keeping-assist-explained (2014)
17. Li, H., Zhao, G., Qin, L., Aizeke, H., Zhao, X., Yang, Y.: A survey of safety warnings under connected vehicle environments. IEEE Trans. Intell. Transp. Syst. **22** (2020)
18. Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X.: Object Detection with Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems (2019)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Realtime Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
20. Han, J., Zhang, D., Cheng, G., Liu, N., Xu, D.: Advanced deep-learning techniques for salient and category-specific object detection: a survey. IEEE Signal Process. Mag. **35**, 84–100 (2018)
21. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: "SSD: Single Shot Multibox Detector", European Conference on Computer Vision. Springer (2016)
22. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014)
23. Girshick, R.: Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision (2015)
24. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing systems (NIPS) (2015)
25. Fayyad, J., Jaradat, M.A., Gruyer, D., Najjaran, H.: Deep learning sensor fusion for autonomous vehicle perception and localization: A review. Sensors. **20** (2020)
26. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Transactions of the American Society of Mechanical Engineers (ASME) –Journal of Basic Engineering. (1960)
27. Simon, J., Uhlmann, J.: New Extension of the Kalman Filter to Nonlinear Systems. Signal Processing, Sensor Fusion, and Target Recognition International Society for Optics and Photonics (1997)
28. Yeong, D., Hernandez, G., Barry, J., Walsh, J.: Sensor and sensor fusion technology in autonomous vehicles: A review. Sensors. **21** (2021)

29. Wan, A., Merwe, R.: The Unscented Kalman Filter for Nonlinear Estimation. Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (2000)
30. Reeves, C.: Genetic Algorithms: Handbook of Metaheuristics. International Series in Operations Research & Management Science (2003)
31. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
32. Yang, X.S.: Firefly algorithms for multimodal optimization. Stochastic Algorithms: Foundations and Applications (2009)
33. Zhou, G.D., Yi, T.H., Zhang, H., Li, H.N.: A Comparative Study of Genetic and Firefly Algorithms for Sensor Placement in Structural Health Monitoring. Shock and Vibration (2015)
34. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An Open Urban Driving Simulator. 1st Annual Conference on Robot Learning Conference on robot learning (2017)
35. Brekke, Å., Vatsendvik, F., Lindseth, F.: Multimodal 3d Object Detection from Simulated Pretraining. Symposium of the Norwegian Artificial Intelligence Society, Springer (2019)
36. Lin, S., Zhang, Y., Hsu, C., Skach, M., Haque, M., Tang, L., Mars, J.: The Architectural Implications of Autonomous Driving: Constraints and Acceleration. Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (2018)
37. Dey, J., Pasricha, S.: Co-Optimizing Sensing and Deep Machine Learning in Automotive Cyber-Physical Systems. IEEE Euromicro Conference on Digital Systems Design (2022)
38. Thiruloga, S.V., Kukkala, V.K., Pasricha, S.: TENET: Temporal CNN with Attention for Anomaly Detection in Automotive Cyber-Physical Systems. IEEE/ACM Asia & South Pacific Design Automation Conference (ASPDAC) (2022)
39. Kukkala, V.K., Thiruloga, S.V., Pasricha, S.: LATTE: LSTM Self-Attention Based Anomaly Detection in Embedded Automotive Platforms. IEEE/ACM CODES+ISSS (ESWEEK) (2021)
40. Kukkala, V.K., Thiruloga, S.V., Pasricha, S.: INDRA: Intrusion Detection Using Recurrent Autoencoders in Automotive Embedded Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (TCAD). **39**(11) (2020)
41. Kukkala, V.K., Thiruloga, S.V., Pasricha, S.: "Roadmap for Cybersecurity in Autonomous Vehicles", to Appear, vol. 11, pp. 13–23. IEEE Consumer Electronics (2022)
42. Kukkala, V., Pasricha, S., Bradley, T.H.: SEDAN: Security-aware design of time-critical automotive networks. IEEE Transactions on Vehicular Technology (TVT). **69**(8) (2020)
43. Kukkala, V., Pasricha, S., Bradley, T.H.: JAMS-SG: A framework for jitter-aware message scheduling for time-triggered automotive networks. ACM Transactions on Design Automation of Electronic Systems (TODAES). **24**(6) (2019)
44. Kukkala, V.K., Pasricha, S., Bradley, T.: JAMS: Jitter-aware message scheduling for FlexRay automotive networks. IEEE/ACM International Symposium on Networks-on-Chip (NOCS) (2017)