The Architectural Implications of Autonomous Driving: Constraints and Acceleration

Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach
Md E. Haque[1], Lingjia Tang, Jason Mars

University of Michigan, Ann Arbor
{shihclin,yunqi,hsuch,skachm,mdhaque,lingjia,profmars}@umich.edu

Abstract

Autonomous driving systems have attracted a significant amount of interest recently, and many industry leaders, such as Google, Uber, Tesla and Mobileye, have invested large amount of capital and engineering power on developing such systems. Building autonomous driving systems is particularly challenging due to stringent performance requirements in terms of both making the safe operational decisions and finishing processing at real-time. Despite the recent advancements in technology, such systems are still largely under experimentation and architecting end-to-end autonomous driving systems remains an open research question.

To investigate this question, we first present and formalize the design constraints for building an autonomous driving system in terms of performance, predictability, storage, thermal and power. We then build an end-to-end autonomous driving system using state-of-the-art award-winning algorithms to understand the design trade-offs for building such systems. In our real-system characterization, we identify three computational bottlenecks, which conventional multicore CPUs are incapable of processing under the identified design constraints. To meet these constraints, we accelerate these algorithms using three accelerator platforms including GPUs, FPGAs and ASICs, which can reduce the tail latency of the system by 169×, 10×, and 93× respectively. With accelerator-based designs, we are able to build an end-to-end autonomous driving system that meets all the design constraints, and explore the trade-offs among performance, power and the higher accuracy enabled by higher resolution cameras.

CCS Concepts • **Computer systems organization** → **Neural networks**; **Heterogeneous (hybrid) systems**;

ASPLOS'18, March 24–28, 2018, Williamsburg, VA, USA

ACM ISBN 978-1-4503-4911-6/18/03…$15.00
https://doi.org/https://doi.org/10.1145/3173162.3173191

Keywords Autonomous Driving Systems, Deep Neural Networks

1 Introduction

Google, Uber, Tesla, Mobileye and many automotive companies have recently invested significantly in the future application known as autonomous driving systems. The autonomous driving system allows the vehicle to drive by itself without requiring help from a human. The vehicle equipped with autonomous driving capability detects the environment, locates its position, and operates the vehicle to get to the specified destination safely without human input.

Demand of this application continues to grow leading to ever increasing investment from industry. Intel recently acquired Mobileye, a leader in computer vision-based autonomous driving technology, for $15.3 billion [66]. Reports show that by 2035, automobiles with autonomous driving features are expected to capture 25% of the automotive market, which translates to 18 million vehicles [21], and the size of the autonomous driving vehicle market is expected to leap to $77 billion by 2035 [21].

Despite the recent advancements in autonomous driving systems contributed by industry leaders like Google, Tesla and Mobileye, autonomous driving vehicles are still largely under experimentation and research. As such, architecting the right autonomous driving system still largely remains an open research question.

Architecting autonomous driving systems is particularly challenging for a number of reasons. These systems must make the "correct" operational decision at all times to avoid accidents, thereby advanced machine learning, computer vision and robotic processing algorithms, typically computationally intensive, are employed to deliver the required high precision. Despite the large amount of computation, it is critical for such mission critical system to be able to react to the traffic condition at real-time, which means the processing always needs to finish at strict deadlines. Furthermore, the system needs to perform the necessary computation under certain power budget to avoid negatively impacting the driving range and fuel efficiency of the vehicle by large amounts.

To address these challenges as a research community, there are several key questions that need to be answered:

[1]Haque is currently a Software Engineer at Google.

1. What are the design constraints for building autonomous driving systems?
2. What are the computational profile and bottlenecks of a state-of-the-art end-to-end autonomous driving system?
3. What architecture should we use when building such systems to meet all the design constraints?

To answer these questions, we first investigate the design constraints for autonomous driving systems. We identify six classes of constraints including performance, predictability, storage, thermal, power and others, and have arrived at some unique observations about autonomous driving systems. For instance, we discover it is critical for the system to be able to finish the end-to-end processing at a latency less than 100 ms and a frame rate higher than 10 frames per second, to react fast enough to the constantly changing traffic condition. To capture the stringent performance predictability requirements of such mission critical real-time system, tail latency (i.e., high quantiles of the latency distribution) should be used to quantify the performance of the system. We also find the power consumption of such system is heavily magnified (almost doubled) by the increased cooling load to remove heat generated by the computing system to keep the passenger cabin within a tolerable temperature, making it challenging to leverage power-hungry computational platforms without significantly degrading the vehicle driving range and fuel efficiency.

To understand the computational profile of such systems, we first need to address the challenge of there being a lack of publicly available end-to-end experimental frameworks that are representative of the state-of-the-art autonomous driving systems, which places a significant obstacle for our community to investigate these systems. Therefore, we build an end-to-end autonomous driving system based on most recently published system designs from academic research [37] and industry practitioners [72]. The algorithmic components we use represent the most up-to-date advancements in relevant fields, as they have won the corresponding machine learning challenges recently (e.g., YOLO [57] won the multi-object detection challenge [16]). In this architecture (detailed in Section 2.3), the video captured by cameras are streamed into an object detection engine to detect interested objects and a localization engine to locate the vehicle based on the nearby landmarks in parallel. The detected object coordinates are then fed into an object tracking engine that tracks the moving objects to predict their moving trajectories. The output of the object detection engine and the localization engine is then fused onto the same 3D coordinate space to plan out the operational decisions.

With this end-to-end experimental framework, we discover three computational bottlenecks, namely object detection, object tracking and localization, account for more than 94% of the computation. These computationally expensive bottlenecks prevent conventional multicore CPU systems from meeting the design constraints of such systems. Thus, we conduct an in-depth investigation to explore the viability of accelerating these algorithmic components using various accelerators. Lastly, we provide insights on how future architecture designs should evolve for this emerging and fast growing application. Specifically, we make the following contributions:

- **Identifying Design Constraints** – We identify and present the design constraints of autonomous driving systems in terms of performance, predictability, storage, thermal and power. Despite the stringent real-time performance constraints (i.e., a tail latency lower than 100 ms at a frame rate higher than 10 frames per second), the power consumption of such system is heavily magnified by the increased cooling load to remove the heat generated by the computing system, resulting in significant impact on the fuel efficiency and driving range of the vehicle (Section 2).
- **Experimental Infrastructure for Autonomous Driving** – To investigate the design of autonomous driving systems, we build an end-to-end system whose architecture aligns with latest industry products. The models and algorithmic components we employ in this system are representative of the state-of-the-art techniques, as they recently won the corresponding machine learning challenges and awards (Section 3).
- **Accelerating Autonomous Driving** – We identify three computational bottlenecks, including object detection, object tracking and localization, in the end-to-end system that must be accelerated to meet all the design constraints. To thoroughly investigate the implications and trade-offs across devices when accelerating these bottlenecks, we implement them on three different accelerator platforms including GPUs, FPGAs and ASICs and provide a quantitative analysis (Section 4).
- **In-Depth Exploration of Acceleration Landscape** – We explore the landscape of acceleration-based autonomous driving design choices across the three accelerator platforms and discuss the implication trade-offs for future architecture among performance, power and scalability (Section 5).

In summary, we find GPU-, FPGA-, ASIC-accelerated autonomous driving systems can significantly reduce the processing tail latency by factors of 169×, 10×, 93× respectively. This allows us to realize an end-to-end system that meets all the design constraints we identify in Section 2.4. We observe computational platforms with high performance predictability (e.g., GPUs, FPGAs, ASICs) are critical to meet the stringent real-time processing requirements of such mission critical application. However, power-hungry accelerator platforms like GPUs can significantly degrade the vehicle

Table 1. Summary of autonomous driving vehicles under experimentation in leading industry companies.

Manufacturer	Mobileye [47]	Tesla [15, 70]	Nvidia/Audi [67]	Waymo [22, 68, 77]
Automation	level 2	level 2	level 3	level 3
Platform	SoCs	SoCs + GPUs	SoCs + GPUs	SoCs + GPUs
Sensor	camera	camera, radar	lidar, camera, radar	lidar, camera, radar

driving range and fuel efficiency, since the added cooling load to remove the additional heat generated by the computing system almost doubles system power consumption.

2 Autonomous Driving

To investigate autonomous driving systems, we first present a formal taxonomy of such systems defined at different levels of automation ranging from no automation to full automation, as well as where the current industry stands based on this taxonomy. We then introduce the computational pipeline of the state-of-the-art highly automated autonomous driving systems. Based on this pipeline, we investigate and formalize the design constraints of architecting such systems.

2.1 Level of Automation

To facilitate the development of highly autonomous vehicles (HAVs), the National Highway Traffic Safety Authority released a guideline for autonomous driving systems [75] in which they referred to the six levels of automation defined by SAE International [59].

- **No Automation (Level 0) –** The human driver must complete all driving tasks even with warnings from vehicles.
- **Driver Assistance (Level 1) –** The automated system shares steering and acceleration/deceleration responsibility with the human driver *under limited driving conditions* (e.g., high speed cruising), and the driver handles the remaining driving tasks (e.g., lane change).
- **Partial Automation (Level 2) –** The automated system fully controls the steering and acceleration/deceleration of vehicles *under limited driving conditions*, and the human driver performs remaining driving tasks.
- **Conditional Automation (Level 3) –** The automated system handles all driving tasks *under limited driving conditions*, and expects that the human driver will respond to requests to intervene (i.e., resume driving).
- **High Automation (Level 4) –** The automated system handles all driving tasks *under limited driving conditions* even if the human driver does not respond to requests to intervene.
- **Full Automation (Level 5) –** The automated system takes full control of all driving tasks under *all driving conditions* that can be managed by a human driver.

In summary, level 1 and 2 of automation are still mostly driving assistance, where the human driver still handles a substantial portion of the driving tasks at all times under all conditions. Autonomous driving systems can take full driving responsibility at level 3-5 of automation under certain driving conditions, which are typically referred as HAVs. As they represent the future of autonomous driving systems, we focus on HAVs at level 3-5 for the rest of the paper.

2.2 Current Industry Status

To understand where current industry stands, we survey the industry leaders in the level of automation, the computing platform and sensors they leverage as presented in Table 1. As shown in the table, even leading industry companies like Tesla and Waymo can only achieve level 2 or 3 of automation, where the human driver is still heavily involved in the control of the vehicle. It demonstrates the challenges in building autonomous driving vehicles and motivates our research community to investigate this emerging application.

Looking at the computing platforms and sensors these industry leaders use, most of them leverage a combination of SoCs and GPUs to provide the large amount of computational capacity needed for autonomous driving systems. Another interesting observation is both Nvidia/Audi and Waymo, who are able to build experimentation autonomous driving vehicles at level 3 of automation, use Light Detection and Ranging (LIDAR) as part of the sensing devices, which is a remote sensing device used to examine surroundings of the vehicle at high precision by sending light beams. Although the high precision makes LIDAR a great fit as a sensing device for autonomous driving systems, the extreme high cost of LIDAR has been one of the primary reasons that prevent such systems from being commercially available on the market. Commercially available LIDAR devices are as expensive as $75,000 USD [76], which is much higher than the cost of the vehicle itself, even for some luxury cars. As a result, the industry has been trying to move away from LIDAR devices, and build vision-based autonomous driving systems instead, using only cameras and radars that are much cheaper for sensing the surroundings. For instance, companies like Mobileye [48, 50] and Tesla [69] have recently announced their plan for focusing on vision-based autonomous driving systems which are composed of mainly cameras and radars as sensing devices. Therefore, we focus on vision-based autonomous driving systems in this paper.

2.3 Autonomous Driving Pipeline

The task of autonomous driving is to operate the vehicle to reach a given destination, with the data captured on various real-time sensors such as video cameras, laser scanners, and milliwave radars. The autonomous driving system then performs the necessary processing to recognize the driving environments and makes operating decisions. Such system is often composed of three major components: scene recognition for localizing the vehicle at decimeter-level and tracking nearby objects, path planning for generating the future paths, and vehicle control for physically operating the vehicle to

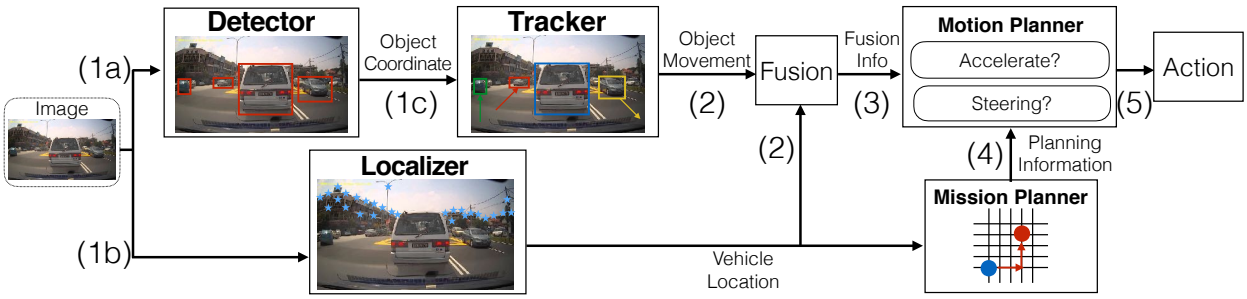


Figure 1. Overview of a state-of-the-art autonomous driving system, which is designed based on recent publications [37] and specifications released by industry companies [48, 72]. The video captured by cameras are streamed into both the object detection engine to detect objects (1a) and the localization engine to locate the vehicle (1b) in parallel. The detected objects then are passed to the object tracking engine to track moving objects (1c). The vehicle location and the tracked objects are projected into the same 3D coordinate space by the fusion engine (2), which will be consumed by the motion planner (3) to make operational decisions (5). The mission planner is only invoked when the vehicle deviates from the original routing plan generated by the navigation services like Google Maps (4).

follow the planned paths [37]. These algorithm components are the basis of most modern autonomous driving systems, which has been confirmed by the self-driving car Udacity built [72], and also aligns with how Mobileye designs their autonomous driving systems [48]. A detailed diagram of these components is presented in Figure 1.

The captured sensing data is first fed to an object detector (step 1a in Figure 1) and a localizer (step 1b in Figure 1) in parallel. The object detector detects the objects of interest around the vehicle, such as other vehicles, pedestrians, and traffic signals. The detected objects are then passed to an object tracker (step 1c in Figure 1) to associate the detected objects with their movements in the past, to predict the trajectories of moving objects. In parallel, the localizer determines the location of the vehicle at high precision. Subsequently, the object movement information from object tracker and the vehicle location information from localizer is combined and projected onto the same 3D coordinate space by a sensor fusion engine (step 2 in Figure 1).

The fused information about the vehicle location and moving objects is then passed to the motion planning engine to assign path trajectories (step 3 in Figure 1), such as lane change and setting the vehicle's velocity. The mission planning engine calculates the detailed operating motions to realize the planned paths and determine the routing path from source to destination (step 4 in Figure 1). The vehicle control engine simply follows the planned paths and trajectories by operating the vehicle (step 5 in Figure 1).

2.4 Design Constraints

Despite the extremely detailed regulations on conventional automobiles (e.g., crash test, fuel economy, vehicle inspection), regulatory authorities have only recently started forming these regulations regarding autonomous driving vehicles. In the Federal Automated Vehicle Policies published by the U.S. Department of Transportation [75], it was only mentioned that "significant emphasis should be placed on software development, verification and validation" without any specific details. Therefore, many of the design constraints we discuss in this section are derived from published materials by industry practitioners like Toyota [34], Udacity [72] and Mobileye [48].

2.4.1 Performance Constraints

To avoid car accidents, the autonomous driving system needs to be able to "understand" the real-time traffic condition and react to it fast enough. While autonomous vehicles have the potential to reduce traffic casualties, the actual performance requirement for autonomous driving system is still largely undefined. According to prior work in driver-assistance systems [51], the reaction time of an autonomous driving system is determined by two factors.

- **Frame rate:** The frame rate determines how fast the real-time sensor data can be fed into the process engine.
- **Processing latency:** The processing latency of recognizing scenes and making operational decisions determines how fast the system can react to the captured sensor data.

Human drivers take varying amount of time to respond based on the level of expectation and action chosen. For example, human drivers take 600 ms to react when they are expecting a possible interruption and 850 ms otherwise [33]. A typical driver takes 0.96 s to release accelerator, 2.2 s to reach maximum breaking, and 1.64 s to begin steering to avoid an accident [43]. The fastest possible action by a human driver takes 100–150 ms [54, 71]. To provide better safety, autonomous driving systems should be able to react faster than human drivers, which suggests the latency for processing traffic condition should be within 100 ms. This aligns with

the industry standards recently published by Mobileye [62] and the design specifications from Udacity [72].

In addition to processing latency, autonomous driving systems also need to frequently update their "understanding" to keep up with the continuously changing real-time traffic condition. In other words, the frame rate needs to be high, in case the real-time traffic condition changes drastically between two neighboring frames. To react quickly to the constantly changing traffic condition, the system should be able to react faster than human reaction time, which suggests a frequency of once every 100 ms. This also aligns with the frame rate of the collision prevention assistant systems built by Mobileye [49].

Performance Constraints: Autonomous driving system should be able to process current traffic conditions within a latency of 100 ms at a frequency of at least once every 100 ms.

2.4.2 Predictability Constraints

Autonomous driving is one of the mission critical applications that must be performed at real-time. What this means is the processing fails if not completed within a specific deadline, thereby the performance predictability is critical. Not being able to process in real-time can put the passengers in danger, and sometimes result in fatal accidents. Therefore, the performance of autonomous driving systems needs to be extremely predictable for them to be widely adopted. The predictability is defined as both the temporal aspects (i.e., meeting the specified timing deadline) and the functional aspects (i.e., making the correct operational decisions). From an architect's point of view, we focus on the predictability of the temporal aspects.

Specifically, the predictability of the processing latency is critical for the autonomous driving system to quickly react to the real-time traffic condition reliably. To capture the non-determinism of large-scale distributed systems, tail latency, defined as the high quantiles of the latency distribution (e.g., 95th-, 99th- percentile latency), is often used to evaluate the performance of such systems instead of mean latency. As we will show in Section 3.2, the localization algorithm has large performance variability, which is challenging for the autonomous driving system to react to the real-time traffic. As a result, tail latency, high quantiles like 99.99th-percentile or even worst case latency, should be used to evaluate the performance of such systems to reflect the stringent predictability requirements. We will also empirically demonstrate why tail latency should be used in Section 5.1.2.

Predictability Constraints: Due to the large performance variability of autonomous driving systems, tail latency (e.g., 99th-, 99.99th- percentile latency) should be used as the metric to evaluate the performance, in order to capture the stringent predictability requirement.

2.4.3 Storage Constraints

While GPS technology has been commonly adopted to identify the vehicle location for navigation systems, it does not provide the necessary level of precision (e.g., precision at decimeter-level is needed to keep the vehicle staying in certain lanes [40]) and the spacial accessibility [5] to localize the vehicle (step 1b in Figure 1) for autonomous driving tasks. Therefore, prior map-based localization has been widely used to provide localization capability at centimeter-level precision [44, 53, 64, 78, 79, 83], where the surrounding view is transformed into feature descriptions to map the feature points stored in the prior map to identify the location of the vehicle. However, it is infeasible to transmit the prior map from the cloud all the time, because the vehicle does not always have access to the Internet and the vehicle still needs to perform the necessary autonomous driving tasks even under limited accessibility. Therefore, the prior map needs to be stored on the autonomous driving vehicle.

However, prior maps of large environments (e.g., an entire country) consume significant amount of storage space. For example, a prior map of the entire United States takes 41 TB of storage space on an autonomous driving system [74].

Storage Constraints: Tens of TBs of storage space is needed to store the prior maps in large environments required by autonomous driving systems to localize the vehicle (e.g., 41 TB for an entire map of the U.S.).

2.4.4 Thermal Constraints

There are two aspects of thermal constraints in autonomous driving systems: 1) the temperature of the space to hold the computing system needs to be within the operating range the system can operate under; 2) the heat generated by the computing system should have relatively small impact on the thermal profile of the vehicle (e.g., not heat up the engine, which can potentially affect the reliability of the vehicle).

There are typically two temperature zones in modern autonomous driving vehicles: in the climate controlled passenger cabin or outside of [34]. Outside the passenger cabin, the operating temperature can get up to +105°C ambient [34], which is higher than most general-purpose computer chips could safely operate under (e.g., a typical Intel processor can only operate at temperature lower than 75°C [29]). Therefore, the autonomous driving system should be placed inside the climate controlled passenger cabin to avoid having to build climate control functionality for additional zones.

However, the passengers may no longer be able to tolerate the increased temperature when the computing system is placed inside the passenger cabin without additional cooling infrastructure. For instance, a computing system that consumes 1kW power (e.g., 1 CPU and 3 GPUs operating at full utilization) will raise the temperature by 10°C in a minute if no additional cooling is added in the passenger cabin [18]. In conclusion, additional air conditioning load needs to be

added to remove heat generated by the autonomous driving systems to keep the passenger cabin temperature tolerable.

Thermal Constraints: The computing system of autonomous driving vehicle needs to be put into the climate controlled passenger cabin to be able to operate safely, which means additional cooling capacity needs to be added to remove the additional heat generated by the computing system to maintain a tolerable temperature in the passenger cabin.

2.4.5 Power Constraints

In gasoline powered cars, the electrical system is typically on the order of 1-2 kW provided by the car's alternator [46, 55], which could be increased at the cost of reduction in fuel efficiency of the vehicle [60]. The exact reduction varies depending on the fuel efficiency of the car, but a rule of thumb for gas powered cars is that the miles per gallon (MPG) rating will be reduced by one for every additional 400 W of power consumption [17] (e.g., an additional 400 W power consumption translates to a 3.23% reduction in MPG for a 2017 Audi A4 sedan with 31 MPG originally [4]). Similarly, the additional power consumption will reduce the total driving range of electric vehicles (EVs) [69] due to the limited battery capacity.

The total power consumption of the autonomous driving system includes the consumption of the computing system, storage overhead (Section 2.4.3) and cooling overhead (Section 2.4.4). While the power consumption of the computing system heavily depends on the computing platform (e.g., CPUs, GPUs), a typical storage system consumes around 8 W to store every 3 TB data [61]. To remove the additional heat generated by the system, a typical automobile air conditioner consumes around 77% of the cooling load to dissipate the heat (i.e., a coefficient of performance of 1.3, representing the ratio of useful cooling provided to work required [35]). That is to say, a 100 W system imposes 77 W cooling overhead to remove the additional heat generated by the system.

In Figure 2, we analyze the reduction in driving range of a Chevy Bolt [10] as additional power needs are placed on an electric vehicle. The first three sets of bars on the left represent the power consumption and driving range reduction contributed by the computing engine only, and the ones on the right show the corresponding metrics for the entire system in aggregate (i.e., including storage and cooling overhead). Surprisingly, the computing engine only contributes about half of the power consumption and the driving range reduction, where the storage engine and especially the cooling overhead almost double the impact. For example, a computing engine equipped with 1 CPU and 3 GPUs operating at full utilization alone only reduces the driving range by 6%, while the entire system experiences almost doubled reduction (i.e., 11.5%).

Power Constraints: The power consumption of autonomous driving system consists of the consumption of the computing engines and the storage engine, and is heavily magnified by

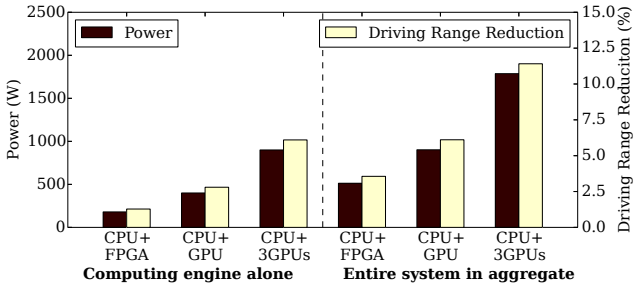


Figure 2. Driving distance per charge reduction contributed by the computing engine alone (on left) and the entire system in aggregate (on right) with respect to the additional power consumption generated by autonomous driving systems. The power consumption is heavily magnified by the storage engine and especially the cooling overhead resulting in driving range reductions as much as 11.5%.

the required cooling capacity to remove the additional heat. Having a power-hungry system can significantly degrade the vehicle fuel efficiency (e.g., as much as 11.5%).

2.4.6 Other Constraints

Additionally, there are also other constraints that we are not focusing on in this paper. Any equipment used in the car should be able to sustain the impulse and vibration of the vehicles. Sudden impulses can range somewhere between 50 g to 500 g (g is used to measure impulse and stands for gravitational force) and vibrations can be up to 15 g and 100–2000 Hz [34]. Hardware reliability is also a constraint in real-time systems, where airplanes typically employ triple-modular-redundancy to provide the safety guarantee [81]. However, autonomous driving vehicles experience much less environment variability (e.g., temperature, atmospheric pressure) than airplanes, which makes it less likely for rare events like radiation-induced soft errors to occur.

3 End-to-End System

Due to the lack of a publicly available experimental framework, we build an end-to-end workload to investigate the architecture implications of autonomous driving vehicles. In this section, we detail the state-of-the-art algorithmic components that constitute this end-to-end autonomous driving system and the design decisions in choosing these algorithmic components. We then characterize the end-to-end system to understand its computational profile and the potential bottlenecks for meeting all the design constraints.

3.1 Algorithmic Components

To build a representative end-to-end system for the state-of-the-art autonomous driving system, we select the best publicly available algorithms from the corresponding machine learning benchmark suites (e.g., VOT benchmark [39] for object tracking tasks).

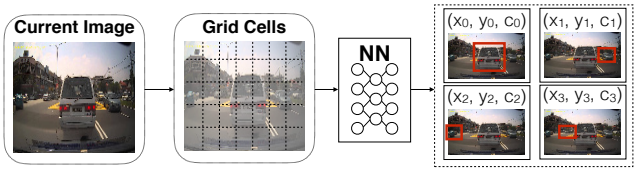


Figure 3. Overview of the object detection engine (DET). It partitions the input image into sub-regions and predicts the coordinates of detected objects and the confidence for each sub-region using Deep Neural Networks (DNNs).

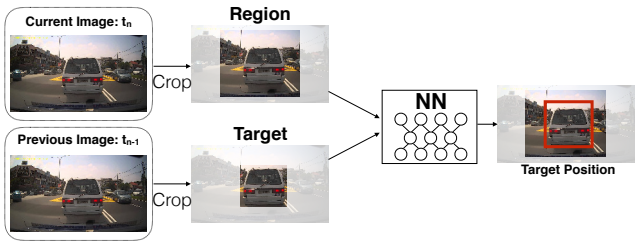


Figure 4. Overview of the object tracking engine (TRA). It crops the next image into a search region and the previous image into the tracking target only based on previous results. It then uses Deep Neural Networks (DNNs) to locate the target object in the search region.

3.1.1 Object Detection

For object detection (DET), we use YOLO [57], a DNN-based detection algorithm, which outperforms all the other multiple object detection algorithms in both accuracy and speed on VOC benchmark suite [16]. Figure 3 presents an overview of the algorithm. It first partitions the image into sub-regions and predicts the coordinates of detected objects and the confidence for each region through fully convolutional networks. A threshold is then used to filter out the objects detected with lower probabilities. In the output. we focus on four categories that we care the most in autonomous driving, including vehicles, bicycles, traffic signs and pedestrians.

3.1.2 Object Tracking

For object tracking (TRA), we use GOTURN [26], a DNN-based single object tracking algorithm, which outperforms most state-of-the-art trackers in speed and accuracy in the VOT object tracking benchmark [39]. As shown in Figure 4, it crops the current image into a search region and the previous image into the tracking target only based on the previous result. Both the search region and the target are then fed as inputs to the neural networks, which generates the bounding box of the target to be tracked in the current image.

A pool of trackers is launched initially to wait for incoming tracking requests to avoid the initialization overhead. To record the number of objects tracked, we implement a tracked object table to store the objects that are being tracked currently. A threshold is set to determine whether an object is passing or leaving if it does not appear in ten consecutive images. We remove it from the tracked object table and set the tracker idle to wait for more incoming tracking requests.

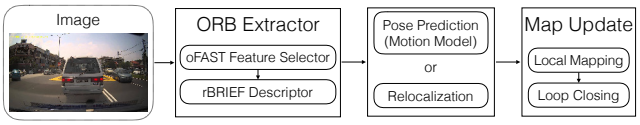


Figure 5. Overview of the localization engine (LOC). The algorithm takes images as input and extracts interesting features such as landmarks, and then generates a descriptor for each of the extracted features. The feature descriptors are consumed to locate the vehicle and predict vehicle poses.

3.1.3 Localization

For localization (LOC), we use ORB-SLAM [52], which has been ranked top of the available open source algorithms for localizing the vehicle on the KITTI datasets [20], as well as on the TUM benchmark suite [65]. Besides the high accuracy, the algorithm is also capable of localizing the vehicle regardless of viewpoints, which makes it a great fit for autonomous driving systems. Figure 5 presents the details of the algorithm. The incoming video stream is fed into the ORB extractor to detect feature points using oFAST algorithm. The rBRIEF algorithm is then invoked to generate descriptions of the extracted feature points.

ORB-SLAM then attempts to match the current descriptions with the prior map database (as mentioned in Section 2.4.3) to localize the vehicle position. It uses a constant motion model to identify the location if the matching succeeds, otherwise it relocalizes the position by using a wider search in the map around the location identified last time.

In addition, the algorithm needs to update the map in case the current surroundings are different from the prior map (e.g., the map is built under different weather conditions). Lastly, loop closing is executed periodically to detect the potential trajectory loop when the vehicle is moving in order to calibrate the vehicle position based on the map database.

3.1.4 Fusion

The fusion engine (FUSION) retrieves the coordinates of the objects being tracked by the trackers, and combines with the current vehicle location provided by the localization engine. The combined information is transformed into the same 3D coordinate space, and sent to the motion planning engine to make vehicle operation decisions.

3.1.5 Motion Planning

For motion planning (MOTPLAN), we use the algorithms in the Autoware, which is a recently published open-source framework [37]. It leverages a graph-search based approach to find the minimum-cost path in space lattices when the vehicle is in an large opening area like parking lot or rural

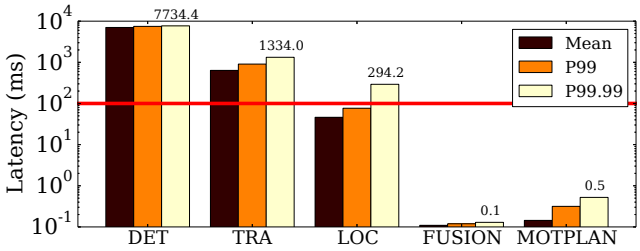


Figure 6. Latency of each algorithmic component on a multicore CPUs system in the end-to-end autonomous driving system. The latency contributed by each of object detection engine (DET), object tracking engine (TRA), and localization engine (LOC) has already exceeded the latency requirement of the end-to-end system. These three components dominate the end-to-end latency, and thereby are the computational bottlenecks that prevent us from meeting design constraints.

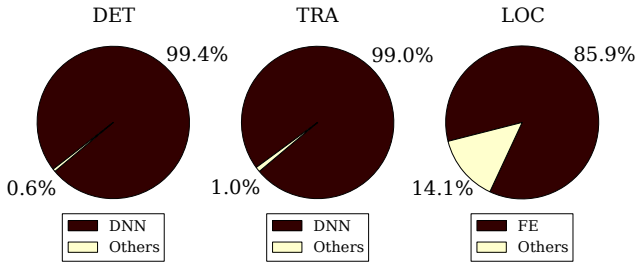


Figure 7. Cycle breakdown of the object detection (DET), object tracking (TRA) and localization (LOC) engines. The Deep Neural Networks (DNNs) portion in DET and TRA, and the Feature Extraction (FE) portion in LOC account for more than 94% of the execution in aggregation, which makes them ideal candidates for acceleration.

area [56]. When the vehicle is in structured areas (e.g., downtown area in cities), it uses conformal lattices with spatial and temporal information to adapt the motion plan to the environment [45, 73].

3.1.6 Mission Planning

For mission planning (MISPLAN), we also adopt the implementation from the state-of-the-art Autoware framework [37]. The policy is a rule-based approach which takes the traffic rules and the driving area condition to determine the routing path trajectory, which aligns with what industry leader Mobileye has been using [62]. The algorithm operates the vehicle to follow the route generated by navigation systems such as Google Maps. It is only executed once unless the vehicle deviates from planned routes.

3.2 System Characterization

To understand the computational profile and identify the potential bottlenecks in autonomous driving systems, we characterize our system using representative KITTI dataset [20] on an Intel server compiled with Intel Math Kernel Library (MKL), and the hardware specifications are listed in Table 2.

Figure 6 shows the measured mean, 99th- and 99.99th-percentile latency for each of the algorithmic components. Mission Planning (MISPLAN) is not included because it is only invoked once at the beginning to plan the route and will not be executed again unless the vehicle deviates from the routing path. As shown in the figure, the latency contributed by each of DET, TRA, and LOC individually has already exceeded the end-to-end system latency constraints at 100 ms, and these three components dominate the system latency. Therefore, we conclude DET, TRA and LOC as the computational bottlenecks of our autonomous driving system, and we will focus on them for the rest of the paper.

We then characterize DET, TRA and LOC in details to investigate where they spend most of their computing cycles. Figure 7 shows the cycle breakdown of each algorithmic component. From the figure, we can easily identify that Deep Neural Networks (DNNs) is the most computational intensive portion in DET and TRA, as they consume 99.4% and 99.0% of the execution time respectively. Unlike the other two DNN-based algorithms, Feature Extraction (FE) consumes more than 85% of the execution time in LOC. These large fractions indicate the DNN portion in DET and TRA, and the FE portion in LOC are good candidates for acceleration in order to meet the strict real-time processing performance constraints for autonomous driving systems.

4 Accelerating Autonomous Driving

As demonstrated in Section 3.2, conventional multicore CPU systems are not suitable to meet all the design constraints, particularly the real-time processing requirement. Therefore, we port the critical algorithmic components to alternative hardware acceleration platforms, and investigate the viability of accelerator-based designs. In this section, we detail our design and implementation on these accelerator platforms, and evaluate their performance in regards to the design constraints of autonomous driving systems.

4.1 Accelerator Platforms

We focus on three different state-of-the-art computing platforms including GPUs, FPGAs and ASICs, and use multicore CPUs as our baseline. The detailed specifications of the hardware are listed in Table 2. The multicore CPU platform we use is a server-grade dual-socket machine with 8 cores on each socket, which represents a more conventional computing system design. The GPU accelerator we study is a latest Nvidia Titan X GPU with Pascal microarchitecture with 12 GB on-board memory, which offers powerful computing capability with 3584 cores. For FPGA platform, we employ an Altera Stratix V development board equipped with 256 Digital Signal Processors (DSPs) and large reconfigurable fabric. Lastly, we explore ASIC designs using prior work built on

Table 2. Computing platform specifications. **: DSP (Digital Signal Processor)*

	CPU	GPU	FPGA	ASIC (CNN)	ASIC (FC)	ASIC (LOC)
Model	Intel Xeon E5-2630 v3	NVIDIA TitanX (Pascal)	Altera Stratix V	TSMC 65 nm	TSMC 45 nm	ARM 45 nm
Frequency	3.20 GHz	1.4 GHz	800 MHz	200 MHz	800 MHz	4 GHz
# Cores	16	3584	256*	N/A	N/A	N/A
Memory	128 GB	12 GB	2 GB	181.5 KB	N/A	N/A
Memory BW	59.0 GB/s	480.0 GB/s	6.4 GB/s	N/A	N/A	N/A

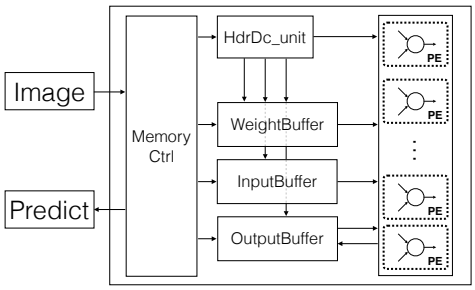


Figure 8. Diagram of our DNNs implementation on FPGAs. The limited on-chip memory on FPGAs is not sufficient to hold all the network architecture configurations, so the networks are executed layer by layer. For each layer, the memory controller initiates the data transfer and the layer definition is used by the header decoder unit (HdrDc_unit) to configure the layer. Processing Elements (PEs) consumes data in the WeightBuffer and InputBuffer to compute the output and store it to the OutputBuffer. To hide the data transfer latency, we implement double buffering for all buffers.

TSMC 65nm and 45nm technology [9, 23], and also build our own implementation using ARM 45nm technology.

4.2 Porting Methodology

4.2.1 GPU Implementation

To port the three bottleneck algorithmic components (i.e., DET, TRA and LOC) to GPUs, we leverage highly optimized machine learning software libraries. In particular, we implement the YOLO [57] object detection algorithm using the cuDNN library provide by Nvidia [11]. We port GOTURN [26] object tracking algorithm to GPUs using Caffe [32], which again allows us to benefit from the highly optimized cuDNN library [11]. The ORB-SLAM [52] algorithm used for localization is ported to GPUs with OpenCV library [6].

4.2.2 FPGA Implementation

To port the three bottleneck algorithmic components to FPGAs, we focus on the most time-consuming algorithms, namely DNN and Feature Extraction (FE), which account for almost 95% of the total execution cycles. We build our own optimized implementations on an Altera Stratix V platform and detail our implementation of these two algorithms in the following sections.

DNNs on FPGAs Despite the high memory bandwidth and large external memory, the Altera Stratix V platform has limited on-chip memory which is insufficient to hold all the neural network architecture configurations (i.e., the network topology and weights of the neurons) and the intermediate results, because the advanced state-of-the-art DNNs we use have complex and deep network architectures.

An overview of our design is presented in Figure 8. Our implementation is capable of executing all the types of layers used in DET and TRA, including convolutional layers, pooling layers, ReLu layers and fully connected layers. The implementation is composed of four major components: memory controller, buffers, header decoder unit (HdrDc_unit) and the processing elements (PEs). The memory controller first initiates the data transfer between the host device and the FPGA accelerator, and the layer definition (i.e., layer type, weights) is fed to the header decoder unit (HdrDc_unit) to configure the layer. Each buffer stores the corresponding neural network weights, input values, and internal temporary variables until the execution of the layer has been completed. Each PE, primarily consists of multiply-accumulate (MAC) units instantiated by the digital processing processors (DSPs) on the fabric, then performs the necessary computation on the data stored in the WeightBuffer and InputBuffer and writes the output to the OutputBuffer. To hide the data transferring latency, we implement double buffering for all buffers to prefetch the needed data in advance while executing the current layer. Overall, we are able to achieve an utilization higher than 80% on the available adaptive logic modules (ALMs) and DSPs.

FE on FPGAs Feature extraction (FE) dominates the computation time in LOC, so we focus on porting FE when porting to FPGAs. There are two major components in the algorithm ORB we use: oFAST that extracts the feature points and rBRIEF that computes a descriptor for each feature point. An overview of our design is summarized in Figure 9.

For oFAST, we implement an image buffer (ImgBuffer) and a feature point buffer (FtrPntBuffer) using shift register, and the mask window is assigned to the corresponding register. As the input data streaming into the buffer, they are filtered by the mask window so the feature detector only receive the data it needs. Orient_unit is implemented with an `atan2` Lookup Table (LUT), to avoid the extensive use of multipliers and dividers of computing `atan2` naively.

For rBRIEF, we store the pattern information in the pattern LUT on-chip. Rotate_unit is implemented to rotate to the

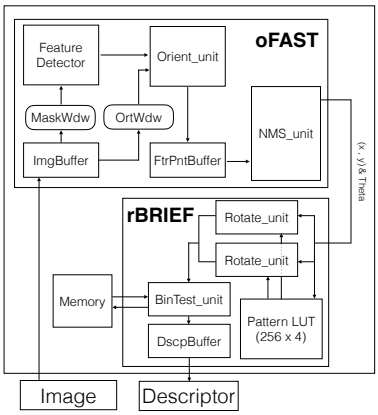


Figure 9. Diagram of our implementation of Feature Extraction (FE) on FPGAs. As the input images streaming into the image buffer (ImgBuffer), they are filtered by the mask window (MaskWdw). The feature detector extracts the features of interest and store them into the feature point buffer (FtrPntBuffer). It is then consumed by the rotate_unit to rotate the corresponding coordinates, and the generated feature descriptors are stored into the descriptor buffer (DscpBuffer). To optimize the performance of our design, we implement all the complex trigonometric functions with Lookup Tables (LUTs) to avoid the extensive use of multipliers and dividers, which reduces the latency by a factor of 1.5×.

corresponding coordinates. Similarly to how we implement `atan2` for oFAST, we implement `sin` and `cos` functions using LUTs to avoid the extensive use of multipliers and dividers. Due to the limited on-chip memory available, we execute one binary test at a time and store the result into the descriptor buffer (DscpBuffer) iteratively. As a result, 256 iterations are required to complete one feature point description. However, because of the simplicity of this design, we can achieve high clock rate and thereby low latency. By synthesizing on real systems, we demonstrate our FE implementation can execute at a frequency of 250MHz on the Stratix V development board. By implementing complex trigonometric functions with LUTs, we improve the performance of FE by a factor of 1.5×.

4.2.3 ASIC Implementation

We employ previously published ASIC implementations for DNNs [9, 23]. Due to the limited published ASIC implementation of feature extraction (FE), we implement our FE ASIC design and synthesize it with modern technology.

We implement FE ASIC following similar design as our FPGA implementation described in Figure 9 in Verilog and synthesize it using ARM Artisam IBM SOI 45 nm library. We verify the result and evaluate the design with post-synthesis simulation. Table 3 shows the details of of FE ASIC implementation, where we are able to achieve a clock frequency as high as 4 GHz. This is largely due to the simplicity of our design, as well as the optimized design with re-timing pipeline. Despite the more cycles needed for the entire pipeline, we achieve better performance comparing to more complex implementations we previously experimented with. Additionally, we are able to achieve a 4× reduction in latency by replacing complex trigonometric function computations with LUTs.

Table 3. Feature Extraction (FE) ASIC specifications.

	Specification
Technology	ARM Artisam IBM SOI 45 nm
Area	6539.9 um^2
Clock Rate	4 GHz (0.25 ns/cycle)
Power	21.97 mW

5 Evaluation

In this section, we conduct thorough evaluations of various acceleration platforms to explore the design landscape. We focus on the three computational bottlenecks identified in Section 3.2 as they constitute more than 94% of the end-to-end execution. In particular, we would like to answer the following questions in this section:

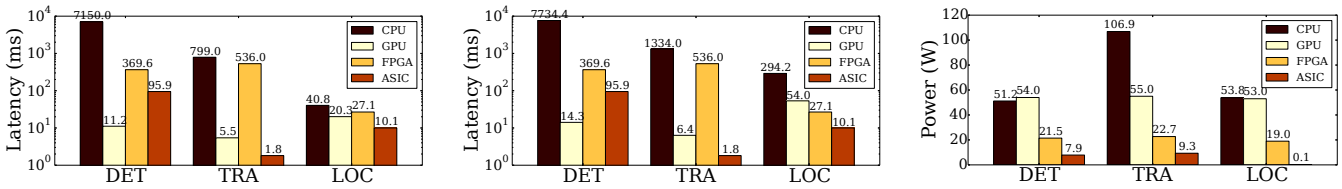
- How much speedup can these accelerators achieve for autonomous driving (Section 5.1)?
- Can accelerator-based autonomous driving systems meet the performance and predictability constraints (Section 5.2)?
- How does the power consumption of accelerator-based autonomous driving systems affect the vehicle (Section 5.3)?
- How scalable are such systems regarding the growing camera resolutions (Section 5.4)?

5.1 Acceleration Results

Figure 10 presents the acceleration results we are able to achieve across mean latency, 99.99th-percentile latency and the measured power consumption on the four computing platforms we investigate. We measure the performance and power on real systems for CPUs, GPUs, and FPGAs using Watts Up power meter [28]. For ASICs, we reference prior work [9, 23] for performance and power measurements for the two DNN-based algorithms object detection (DET) and object tracking (TRA), and extrapolate them based on the amount of processing units needed. For our own ASIC implementation of the feature extraction (FEs) algorithm, we use the power estimated in the post-synthesis result and simulate the performance with our post-synthesis module.

5.1.1 Mean Latency

As shown in Figure 10a, it is impractical to run either DET or TRA on the multicore CPU systems, as the latency of each individual component is already significantly higher than the end-to-end system latency constraints (i.e., 100 ms). This is because both of these components are using DNN-based algorithms, which demands large amount of computing capacity that conventional multicore CPUs does



(a) Mean Latency Across Platforms **(b)** 99.99th-Percentile Latency Across Platforms **(c)** Power Consumption Across Platforms

Figure 10. Acceleration results across various accelerator platforms. The latency of running DET or TRA alone on CPUs or FPGAs has already exceeded the end-to-end latency constraints, which suggests they are not viable candidates for running the complex DNN-based DET and TRA algorithms that demand large amount of computational resources.

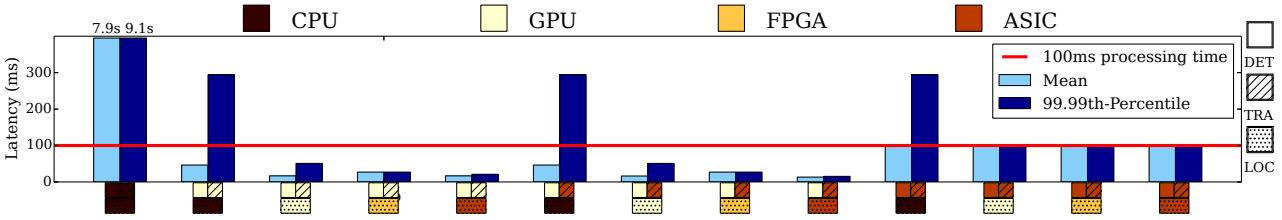


Figure 11. The mean and 99.99th-percentile latency of running different algorithmic components across different configurations denoted on x-axis. For each configuration, the color of each grid represents the computing platform each algorithm is running on (e.g., a red dotted grid represents running LOC on ASICs). There are several configurations that meet the performance constraints at tail latency of 100 ms, which means accelerator-based designs are viable for autonomous driving systems.

not offer. On the contrary, GPUs provide significantly lower mean latency across all three workloads benefiting from the massive parallel processing power provided by the large number of processors. Although FPGAs achieve significant latency reduction comparing to CPUs, their mean latency for DET (i.e., 369.6 ms) and TRA (i.e., 536.0 ms) are still too high to meet the latency constraints at 100 ms. This is largely due to the limited number of DSPs available on the fabric. To support these complex DNN-based algorithms, large amount of DSPs on FPGAs are needed to provide significant compute power, which can be achieved by much advanced FPGAs (e.g., Xilinx VC709 FPGA board [82]). As we expected, ASICs can achieve significant latency reduction, where the mean latency for executing TRA is as low as 1.8 ms. Note the reason why DET runs slower on ASICs than GPUs is because of the limited clock frequency at 200 MHz this particular design can operate at, which does not preclude similar designs with high clock frequencies to outperform GPUs.

Finding 1. *Multicore CPUs are not viable candidates for running object detection (DET) and object tracking (TRA), which are composed of complex DNN-based algorithms that demand large amount of computational resources. The limited number of DSPs becomes the main bottleneck preventing FPGAs from meeting the performance constraints.*

5.1.2 Tail Latency

Figure 10b presents the 99.99th-percentile latency across four platforms. As we can see from the figure, although multicore CPUs can execute the localization algorithm within the performance constraints on average (i.e., mean latency), they suffer from high tail latency across all three workloads. This empirically demonstrated our observation in Section 2.4.2 that due to its large performance variability, tail latency should be used to evaluate the performance of autonomous driving systems to meet the performance predictability constraints. The other computing platforms do not experience any significant increase from mean latency to the tail latency, which is highly preferable for such mission-critical real-time applications.

Finding 2. *Due to the large performance variability of localization algorithm, tail latency should be used to evaluate the performance of autonomous driving systems to meet the real-time constraints, whereas conventional metrics like mean latency can easily cause misleading conclusions.*

5.1.3 Power Consumption

We present the power consumption across 4 platforms for each algorithmic bottleneck in Figure 10c. Note that these measurements focus on the computing system only and do not count the power consumption of the storage engine or the impact of the thermal constraints. The measurements are taken for a single camera, where the end-to-end system consists of multiple cameras (e.g., eight for Tesla [70]) and each camera is paired with a replica of the computing engine to be able to process camera streams from different angles. We will discuss the end-to-end system power consumption in Section 5.3.

As shown in the figure, specialized hardware platforms like FPGAs and ASICs offer significantly higher energy efficiency comparing to general-purpose platforms such as

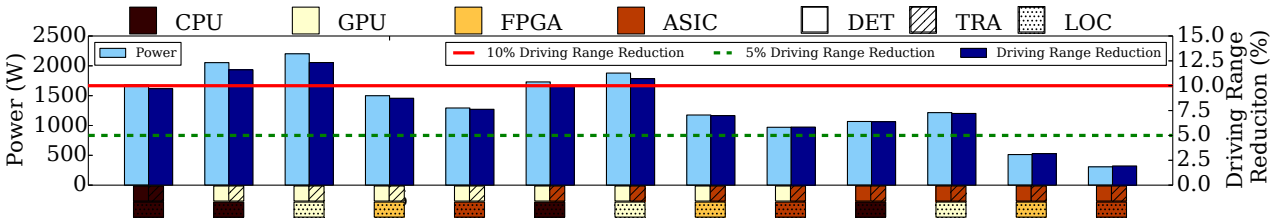


Figure 12. The power consumption and the corresponding driving range reduction of running different algorithmic components across different configurations. Configurations equipped with GPUs consume significant amount of power and reduce the driving range up to 12% while ASICs approaches can achieve efficiency which only reduce the driving range by 2%.

conventional multicore CPUs and GPUs. For instance, running DET on ASICs reduces the power consumption by almost a factor of 7 comparing to CPUs and GPUs. Although the measured power consumption of the computing engine for individual camera is relatively low, remember that the computing engines need to be replicated to handle multiple camera streams and the storage engine and thermal constraints will heavily magnify end-to-end system power as mentioned in Section 2.4.5. As we will demonstrate empirically in Section 5.3, the choice of accelerator platform has a significant impact on the vehicle driving range and fuel efficiency.

Finding 3. *Specialized hardware like FPGAs and ASICs offers significantly higher energy efficiency comparing to conventional general-purpose platforms like multiple CPUs and GPUs for autonomous driving tasks.*

5.2 End-to-End Performance

We then investigate the end-to-end system performance of these accelerator-based autonomous driving system designs. Figure 11 presents the mean latency and the 99.99th-percentile latency of the end-to-end system across different configurations. The x-axis denotes the configuration, where the color of each grid represents the computing platform each algorithmic component is running on (e.g., a red dotted box on the x-axis represents running LOC on ASIC). For example, the 2nd left-most set of bars in the figure represents the latency of running DET and TRA both on GPUs, and LOC on CPUs. Note the end-to-end latency is determined by the slowest path between LOC and DET + TRA, because they are executed in parallel.

We observe in the figure that certain configurations (e.g., LOC on CPUs, DET and TRA on GPUs) can meet the performance constraints at 100ms latency when mean latency is considered, but are no longer viable when considering the tail latency. This again confirms our observation that tail latency should be used when evaluating autonomous driving systems. In addition, none of the viable designs has multicore CPUs due to the inherent non-determinism and unpredictability. With acceleration, we are able to reduce the end-to-end tail latency from 9.1s (i.e., on multicore CPUs) to 16.1ms to meet the real-time processing constraints.

Finding 4. *Accelerator-based design is a viable approach to build autonomous driving systems, and accelerator platforms with high performance predictability (e.g., ASICs) are preferable to meet the real-time processing constraints.*

5.3 Power Analysis

In this section, we investigate the power consumption of the end-to-end accelerator-based autonomous driving systems and quantify the corresponding impact on the driving range and fuel efficiency of the vehicle. The results are evaluated based on a Chevy Bolt [10]. As mentioned in Section 2.4.5, the system power consumption includes the both computing engine and the storage engine, and will then be magnified by the required cooling capacity to remove the additional heat. We assume the system needs to store the map of the United States (i.e., 110 W power for 41 TB storage space), and is equipped with 8 cameras (i.e., the same as Tesla [70]) each connecting to a replica of the computing system.

Figure 12 presents the end-to-end power consumption of the same set of accelerator-based autonomous driving system configurations as Figure 11, where we use the same notation on x-axis to denote the system configurations. The light blue bars and left y-axis show the total power consumption of the end-to-end system, and the dark blue bars and right y-axis show its corresponding driving range reduction. While mentioned in Section 5.2 that powerful accelerators like GPUs can deliver the computation at low latency, it is shown in the figure that most of the configurations equipped with GPUs draw large amount of power (i.e., more than 1,000 W), which results in significant reductions in the driving range of the vehicle. In particular, performing all computing tasks on GPUs can reduce the vehicle driving range by as much as 12%, which dilutes the benefit of using GPUs. To minimize this negative impact, specialized hardware like FPGAs and ASICs are needed to reduce the driving range reduction to within 5%.

Finding 5. *While power-hungry accelerators like GPUs can deliver the computation at low latency, the driving range of the vehicle can be significantly reduced by as much as 12%, largely due to the magnifying effect of the thermal constraints in such systems. Specialized hardware like FPGAs and ASICs are needed to restrain the impact under 5%.*

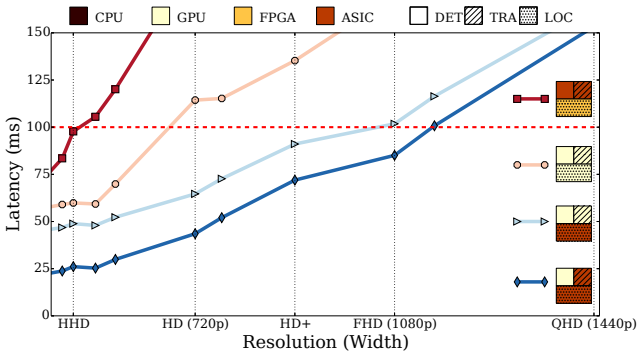


Figure 13. Performance scalability regarding camera resolutions of various configurations. Although some configurations can meet the design constraints at Full HD (FHD) resolution, none of them can sustain higher resolutions like Quad HD (QHD). This suggests computational capacity still remains the bottleneck that prevents us from achieving the higher accuracy enabled by higher resolution cameras.

5.4 Scalability Analysis

Besides the processing latency, the performance predictability of autonomous driving systems is also determined by the functional aspects – the accuracy of making the correct operational decisions. As demonstrated by prior work [3], increasing camera resolution can significantly boost the accuracy of the autonomous driving systems by sometimes as much as 10%. For example, doubling the input resolution can improve the accuracy of VGG16, a DNN-based state-of-the-art object detection algorithm, from 80.3% to 87.4%. Therefore, we investigate the system scalability of our accelerator-based autonomous driving systems in supporting future higher resolution cameras. We modify the resolution of the benchmarks to study this question, and present the end-to-end latency as a function of the input camera resolution in Figure 13 of various accelerator-based configurations. As we can see from the figure, although some of the ASIC- and GPU-accelerated systems can still meet the real-time performance constraints at Full HD resolution (1080p), none of these configurations can sustain at Quad HD (QHD).

Finding 6. *Computational capability still remains the bottleneck that prevents us from benefiting from the higher system accuracy enabled by higher resolution cameras.*

6 Related Work

Prior work has surveyed the common algorithmic components for autonomous driving systems [37]. However, the algorithms presented by Kato et al. are quite outdated, which do not represent the state-of-the-art autonomous driving systems. To address this, we design and develop an end-to-end autonomous driving system with recently developed algorithms that offer significantly higher accuracy. Geiger et al. design and develop a benchmark suite of computer vision applications for studying and evaluating autonomous driving systems [20]. Amer et al. present a review of the state-of-the-art algorithmic components used for path tracking in autonomous driving systems [2].

There is also a large body of work accelerating machine learning-based applications using various accelerator platforms [1, 7–9, 12–14, 19, 23–25, 30, 31, 36, 38, 41, 42, 58, 63, 80]. Specifically, GPUs have been shown to offer orders of magnitude performance improvement over multicore CPUs [24, 25, 27, 58]. This is because many machine learning algorithms spend a large fraction of their execution time performing matrix multiplication, which can be parallelized on the large number of threads offered by GPUs. The commonality exists in these machine learning algorithms, especially DNNs, allows researchers to design ASICs to accelerate them, which offer even higher performance benefits and energy efficiency [1, 7–9, 13, 14, 23, 36, 41]. FPGAs have been discussed as another alternative, which also provide high performance and energy efficiency with the additional capability of reconfiguring the fabric programmatically [19, 42, 63]. In addition to computation, prior work also explored novel memory architectures to bring memory closer to processors [1, 12, 38].

7 Conclusion

This paper presents and formalizes the design constraints in performance, predictability, storage, thermal and power when building autonomous driving systems. To investigate the design of such systems, we build a representative end-to-end autonomous driving system using state-of-the-art machine learning algorithmic components. Using this system, we then identify three computational bottlenecks, namely localization, object detection and object tracking. To design a system that meets all the design constraints, we explore three different accelerator platforms to accelerate these computational bottlenecks. We show that GPU-, FPGA-, and ASIC-accelerated systems can reduce the tail latency of these algorithms by 169×, 10×, and 93× respectively. Based on these accelerated system designs, we further explore the tradeoffs among performance, power and future scalability of autonomous driving systems. We find that while power-hungry accelerators like GPUs can predictably deliver the computation at low latency, their high power consumption, further magnified by the cooling load to meet the thermal constraints, can significantly degrade the driving range and fuel efficiency of the vehicle. We also demonstrate that computational capability remains the bottleneck that prevents us from benefiting from the higher system accuracy enabled by higher resolution cameras.

8 Acknowledgement

We thank our anonymous reviewers for their feedback and suggestions. This work was sponsored by Ford, Michigan Institute for Data Science (MIDAS) and National Science Foundation under grants NSF CAREER SHF-1553485.

References

[1] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos. 2016. Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 1–13. https://doi.org/10.1109/ISCA.2016.11

[2] Noor Hafizah Amer, Hairi Zamzuri, Khisbullah Hudha, and Zulkiffli Abdul Kadir. 2016. Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges. *Journal of Intelligent & Robotic Systems* (2016), 1–30.

[3] Khalid Ashraf, Bichen Wu, Forrest N Iandola, Mattthew W Moskewicz, and Kurt Keutzer. 2016. Shallow networks for high-accuracy road object-detection. *arXiv preprint arXiv:1606.01561* (2016).

[4] Audi USA. 2017. 2017 Audi A4 ultra offers highest EPA-estimated fuel economy in competitive segment. (2017).

[5] Christian Berger and Bernhard Rumpe. 2014. Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system. *arXiv preprint arXiv:1409.0413* (2014).

[6] G. Bradski. 2000. *Dr. Dobb's Journal of Software Tools* (2000).

[7] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 269–284. https://doi.org/10.1145/2541940.2541967

[8] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. DaDianNao: A Machine-Learning Supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-47)*. IEEE Computer Society, Washington, DC, USA, 609–622. https://doi.org/10.1109/MICRO.2014.58

[9] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 367–379.

[10] Chervolet. 2017. Chevrolet Bolt EV. http://www.chevrolet.com/bolt-ev-electric-vehicle. (2017).

[11] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014).

[12] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. 2016. PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 27–39. https://doi.org/10.1109/ISCA.2016.13

[13] Zidong Du, Daniel D. Ben-Dayan Rubin, Yunji Chen, Liqiang He, Tianshi Chen, Lei Zhang, Chengyong Wu, and Olivier Temam. 2015. Neuromorphic Accelerators: A Comparison Between Neuroscience and Machine-learning Approaches. In *Proceedings of the 48th International Symposium on Microarchitecture (MICRO-48)*. ACM, New York, NY, USA, 494–507. https://doi.org/10.1145/2830772.2830789

[14] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. 2015. ShiDianNao: Shifting Vision Processing Closer to the Sensor. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture (ISCA '15)*. ACM, New York, NY, USA, 92–104. https://doi.org/10.1145/2749469.2750389

[15] Electrek. 2017. Elon Musk clarifies Tesla's plan for level 5 fully autonomous driving: 2 years away from sleeping in the car. (2017).

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2012. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. (2012).

[17] R Farrington and J Rugh. 2000. Impact of vehicle air-conditioning on fuel economy, tailpipe emissions, and electric vehicle range. In *Earth technologies forum*. 1–6.

[18] Mohammad Ali Fayazbakhsh and Majid Bahrami. 2013. *Comprehensive modeling of vehicle air conditioning loads using heat balance method*. Technical Report. SAE Technical Paper.

[19] Mingyu Gao, Christina Delimitrou, Dimin Niu, Krishna T. Malladi, Hongzhong Zheng, Bob Brennan, and Christos Kozyrakis. 2016. DRAF: A Low-power DRAM-based Reconfigurable Acceleration Fabric. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 506–518. https://doi.org/10.1109/ISCA.2016.51

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Global Management Consulting. 2016. Autonomous Vehicle Adoption Study. (2016).

[22] Google. 2016. Are We There Yet? Silicon in Self-Driving Cars. (2016).

[23] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016. EIE: efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 243–254.

[24] Johann Hauswald, Yiping Kang, Michael A. Laurenzano, Quan Chen, Cheng Li, Trevor Mudge, Ronald G. Dreslinski, Jason Mars, and Lingjia Tang. 2015. DjiNN and Tonic: DNN As a Service and Its Implications for Future Warehouse Scale Computers. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture (ISCA '15)*. ACM, New York, NY, USA, 27–40. https://doi.org/10.1145/2749469.2749472

[25] Johann Hauswald, Michael A. Laurenzano, Yunqi Zhang, Cheng Li, Austin Rovinski, Arjun Khurana, Ronald G. Dreslinski, Trevor Mudge, Vinicius Petrucci, Lingjia Tang, and Jason Mars. 2015. Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '15)*. ACM, New York, NY, USA, 223–238. https://doi.org/10.1145/2694344.2694347

[26] David Held, Sebastian Thrun, and Silvio Savarese. 2016. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*. Springer, 749–765.

[27] Parker Hill, Animesh Jain, Mason Hill, Babak Zamirai, Chang-Hong Hsu, Michael A Laurenzano, Scott Mahlke, Lingjia Tang, and Jason Mars. 2017. DeftNN: addressing bottlenecks for DNN execution on GPUs via synapse vector elimination and near-compute data fission. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 786–799.

[28] Jason M Hirst, Jonathan R Miller, Brent A Kaplan, and Derek D Reed. 2013. Watts Up? PRO AC Power Meter for automated energy recording: A product review. *Behavior Analysis in Practice* 6, 1 (2013), 82.

[29] Intel. 2017. Intel Core i7-4790K Processor. (2017).

[30] Animesh Jain, Parker Hill, Shih-Chieh Lin, Muneeb Khan, Md E Haque, Michael A Laurenzano, Scott Mahlke, Lingjia Tang, and Jason Mars. 2016. Concise loads and stores: The case for an asymmetric compute-memory architecture for approximation. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 1–13.

[31] Y. Ji, Y. Zhang, S. Li, P. Chi, C. Jiang, P. Qu, Y. Xie, and W. Chen. 2016. NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13. https://doi.org/10.1109/MICRO.2016.7783724

[32] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

[33] Gunnar Johansson and Kåre Rumar. 1971. Drivers' brake reaction times. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 13, 1 (1971), 23–27.

[34] R Wayne Johnson, John L Evans, Peter Jacobsen, James R Thompson, and Mark Christopher. 2004. The changing automotive environment: high-temperature electronics. *IEEE Transactions on Electronics Packaging Manufacturing* 27, 3 (2004), 164–176.

[35] Khalid A Joudi, Abdul Sattar K Mohammed, and Mohammed K Aljanabi. 2003. Experimental and computer performance study of an automotive air conditioning system with alternative refrigerants. *Energy conversion and Management* 44, 18 (2003), 2959–2976.

[36] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos. 2016. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–12. https://doi.org/10.1109/MICRO.2016.7783722

[37] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. 2015. An open approach to autonomous vehicles. *IEEE Micro* 35, 6 (2015), 60–68.

[38] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay. 2016. Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 380–392. https://doi.org/10.1109/ISCA.2016.41

[39] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojíř, Gustavo Fernández, Alan Lukežič, Aleksandar Dimitriev, Alfredo Petrosino, Amir Saffari, Bo Li, Bohyung Han, CherKeng Heng, Christophe Garcia, Dominik Pangeršič, Gustav Häger, Fahad Shahbaz Khan, Franci Oven, Horst Possegger, Horst Bischof, Hyeonseob Nam, Jianke Zhu, JiJia Li, Jin Young Choi, Jin-Woo Choi, João F. Henriques, Joost van de Weijer, Jorge Batista, Karel Lebeda, Kristoffer Öfjäll, Kwang Moo Yi, Lei Qin, Longyin Wen, Mario Edoardo Maresca, Martin Danelljan, Michael Felsberg, Ming-Ming Cheng, Philip Torr, Qingming Huang, Richard Bowden, Sam Hare, Samantha YueYing Lim, Seunghoon Hong, Shengcai Liao, Simon Hadfield, Stan Z. Li, Stefan Duffner, Stuart Golodetz, Thomas Mauthner, Vibhav Vineet, Weiyao Lin, Yang Li, Yuankai Qi, Zhen Lei, and Zhi Heng Niu. 2015. *The Visual Object Tracking VOT2014 Challenge Results*. Springer International Publishing, Cham, 191–217. https://doi.org/10.1007/978-3-319-16181-5_14

[40] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. 2007. Map-Based Precision Vehicle Localization in Urban Environments.. In *Robotics : Science and Systems (RSS), 2007*.

[41] Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. 2015. PuDianNao: A Polyvalent Machine Learning Accelerator. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '15)*. ACM, New York, NY, USA, 369–381. https://doi.org/10.1145/2694344.2694358

[42] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J. K. Kim, and H. Esmaeilzadeh. 2016. TABLA: A unified template-based framework for accelerating statistical machine learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 14–26. https://doi.org/10.1109/HPCA.2016.7446050

[43] Daniel V McGehee, Elizabeth N Mazzae, and GH Scott Baldwin. 2000. Driver reaction time in crash avoidance research: Validation of a driving simulator study on a test track. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 44. SAGE Publications, 3–320.

[44] Colin McManus, Winston Churchill, Ashley Napier, Ben Davis, and Paul Newman. 2013. Distraction suppression for vision-based pose estimation at city scales. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 3762–3769.

[45] Matthew McNaughton, Chris Urmson, John M Dolan, and Jin-Woo Lee. 2011. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 4889–4895.

[46] JM Miller, A Emadi, AV Rajarathnam, and M Ehsani. 1999. Current status and future trends in more electric car power systems. In *Vehicular Technology Conference, 1999 IEEE 49th*, Vol. 2. IEEE, 1380–1384.

[47] Mobileye. 2017. Autonomous Driving. https://www.mobileye.com/. (2017).

[48] Mobileye. 2017. Enabling Autonomous. http://www.mobileye.com/future-of-mobility/mobileye-enabling-autonomous/. (2017).

[49] Mobileye. 2017. Mobileye C2-270 Essentials. (2017).

[50] Mobileye. 2017. Mobileye CES 2017 Press Conference. (2017).

[51] Mihir Mody. 2016. ADAS Front Camera: Demystifying Resolution and Frame-Rate. EETimes. (2016).

[52] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163.

[53] Ashley Napier and Paul Newman. 2012. Generation and exploitation of local orthographic imagery for road vehicle localisation. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 590–596.

[54] Allen Newell and Stuart K Card. 1985. The prospects for psychological science in human-computer interaction. *Human-computer interaction* 1, 3 (1985), 209–242.

[55] David J Perreault and Vahe Caliskan. 2004. Automotive power generation and control. *IEEE Transactions on Power Electronics* 19, 3 (2004), 618–630.

[56] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. 2009. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics* 26, 3 (2009), 308–333.

[57] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242* (2016).

[58] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. 2016. vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13. https://doi.org/10.1109/MICRO.2016.7783721

[59] SAE International. 2014. AUTOMATED DRIVING, Levels of driving automation are defined in new SAE International standard J3016. http://www.sae.org/misc/pdfs/automated_driving.pdf. (2014).

[60] Erwin M Schau, Marzia Traverso, and Matthias Finkbeiner. 2012. Life cycle approach to sustainability assessment: a case study of remanufactured alternators. *Journal of Remanufacturing* 2, 1 (2012), 1–14.

[61] Seagate Technology LLC. 2017. Seagate Desktop HDD Specification. http://www.seagate.com/consumer/upgrade/desktop-hdd/#specs. (2017).

[62] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems* (2016).

[63] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh. 2016. From high-level deep neural models to FPGAs. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–12. https://doi.org/10.1109/MICRO.2016.7783720

[64] Alexander D Stewart and Paul Newman. 2012. Laps-localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2625–2632.

[65] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 573–580.

[66] TechCrunch. 2017. Intel buys Mobileye in $15.3B deal, moves its automotive unit to Israel. (2017).

[67] TechCrunch. 2017. Nvidia is powering the world's first Level 3 self-driving production car. (2017).

[68] TechCrunch. 2017. Waymo reveals completely homegrown sensor suite for Pacifica autonomous test car. (2017).

[69] Tesla. 2017. Full Self-Driving Hardware on All Cars. https://www.tesla.com/autopilot. (2017).

[70] Tesla Inc. 2017. Tesla Autopilot: Full Self-Driving Hardware on All Cars. https://www.tesla.com/autopilot. (2017).

[71] Simon Thorpe, Denis Fize, Catherine Marlot, et al. 1996. Speed of processing in the human visual system. *nature* 381, 6582 (1996), 520–522.

[72] Udacity. 2017. An Open Source Self-Driving Car. https://www.udacity.com/self-driving-car. (2017).

[73] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. 2008. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics* 25, 8 (2008), 425–466.

[74] U.S. Department of Transportation – Federal Highway Administration. 2015. Highway Statistics 2015. https://www.fhwa.dot.gov/policyinformation/statistics.cfm. (2015).

[75] U.S. Department of Transportation – National Highway Traffic Safety Administration. 2017. Federal Automated Vehicles Policy: Accelerating the Next Revolution in Roadway Safety. https://www.transportation.gov/AV. (2017).

[76] Velodyne. 2017. Velodyne HDL-64E LiDAR . http://velodynelidar.com/hdl-64e.html. (2017).

[77] Waymo. 2017. Introducing Waymo's suite of custom-built, self-driving hardware. (2017).

[78] Ryan W Wolcott and Ryan M Eustice. 2014. Visual localization within LIDAR maps for automated urban driving. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 176–183.

[79] Ryan W Wolcott and Ryan M Eustice. 2015. Fast LIDAR localization using multiresolution Gaussian mixture maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2814–2821.

[80] R. Yazdani, A. Segura, J. M. Arnau, and A. Gonzalez. 2016. An ultra low-power hardware accelerator for automatic speech recognition. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–12. https://doi.org/10.1109/MICRO.2016.7783750

[81] Ying C Yeh. 1996. Triple-triple redundant 777 primary flight computer. In *Aerospace Applications Conference, 1996. Proceedings., 1996 IEEE*, Vol. 1. IEEE, 293–307.

[82] Chen Zhang, Zhenman Fang, Peipei Zhou, Peichen Pan, and Jason Cong. 2016. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*. IEEE, 1–8.

[83] Ji Zhang and Sanjiv Singh. 2015. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2174–2181.