Lecture: Self-Driving Cars

- Topics: System analysis for two systems: with and without LIDAR

- Email me to communicate team/project and set up meeting times

The next two slides show the major companies pursuing various levels of autonomy and the definitions for level 2-5 autonomy. In terms of hardware, we've already discussed the Tesla FSD chip, Audi has a new board, and NVIDIA has a Xavier SoC.

Some companies (Waymo) seem to be investing in LIDAR sensors, while others (Tesla especially) have committed to vision-based LIDAR-free solutions. LIDAR sends lasers and estimates object locations based on the reflections, thus producing a point cloud. LIDAR is known to be very expensive ($75K), but one can expect steady cost reductions. Waymo announced a 10x reduction in cost in early 2019; a short-range LIDAR may cost around $5K.

Self-Driving Cars Intro

- Several players: Waymo, Tesla, Mobileye, Uber, NVIDIA Xavier SoC, Audi zFAS board
- Level 2 autonomy: car handles steering/accel in limited conditions, (alert) driver handles rest (hands-off)
- Level 3 autonomy: car handles everything in limited conditions, driver serves as back-up (eyes-off)
- Level 4 autonomy: driver helps only when entering unmapped areas or during severe weather (mind-off)
- Level 5 autonomy: no steering wheel

LIDAR vs. No-LIDAR

- LIDAR is very expensive ($75K); hence, Tesla and Mobileye are focusing on vision-based solutions
- Recent announcements claim that top-of-the-line LIDAR may cost $7,500, with short-range LIDAR being under $5K

  https://techcrunch.com/2019/03/06/waymo-to-start-selling-standalone-lidar-sensors/
- As we'll see, LIDAR has other latency problems

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

Self-Driving Car Pipeline

Three main bottlenecks in the pipeline. The detector and tracker use well-known DNNs. The localizer is the new and unintuitive step.

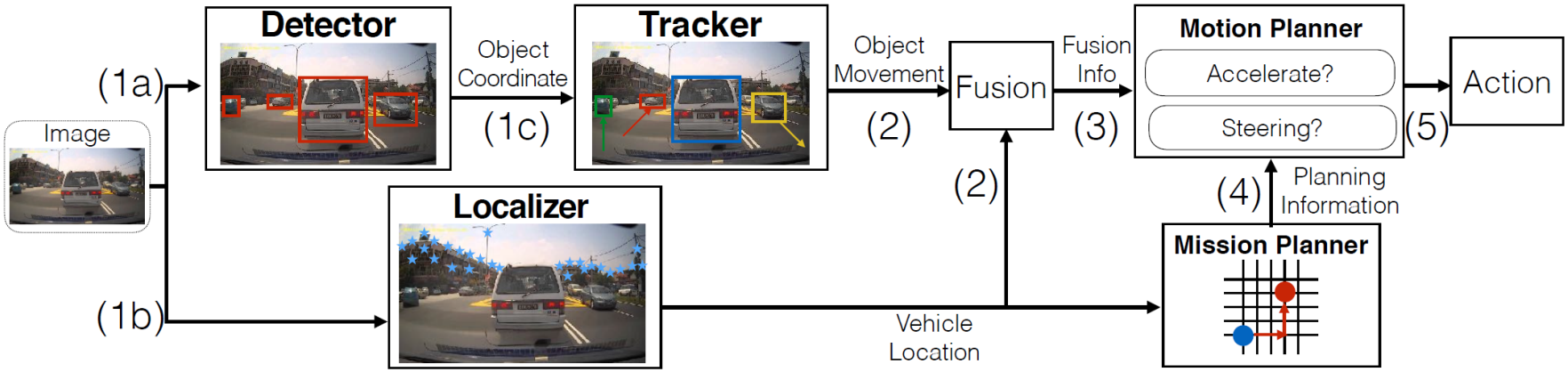


Figure 1. Overview of a state-of-the-art autonomous driving system, which is designed based on recent publications [37] and specifications released by industry companies [48, 72]. The video captured by cameras are streamed into both the object detection engine to detect objects (1a) and the localization 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).

- Need decimeter-level accuracy for localization; GPS fails; localizer gives them centimeter-level accuracy
- Need a locally stored map (41 TB for USA)

Self-Driving Car Pipeline

Actually identifying landmarks and using trig equations to pinpoint where you are at cm resolution. Impressive! But needs a local map.

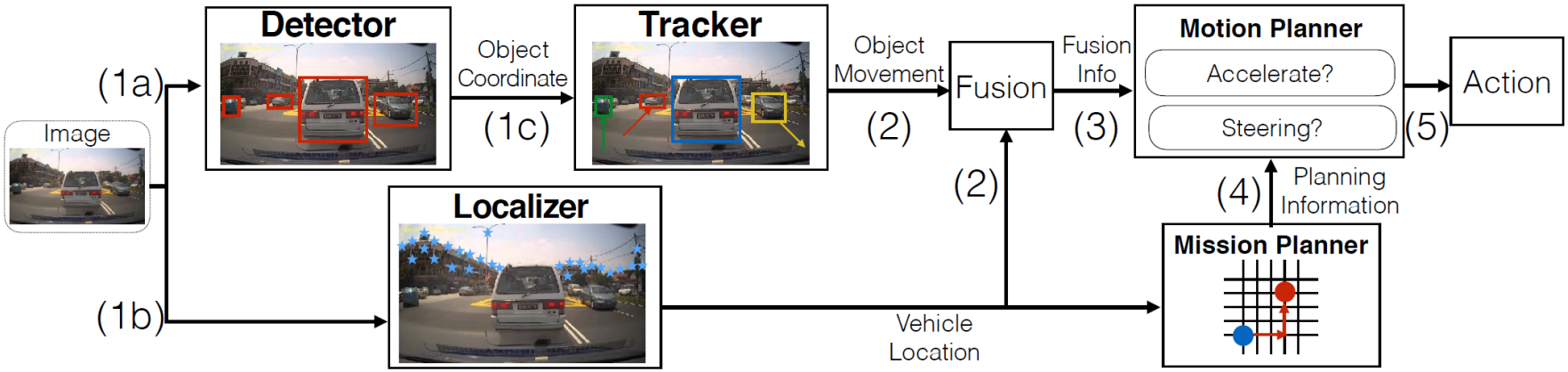


Figure 1. Overview of a state-of-the-art autonomous driving system, which is designed based on recent publications [37] and specifications released by industry companies [48, 72]. The video captured by cameras are streamed into both the object detection engine to detect objects (1a) and the localization 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).

- Need decimeter-level accuracy for localization; GPS fails; localizer gives them centimeter-level accuracy
- Need a locally stored map (41 TB for USA)

Metrics

The paper identifies important metrics. Not surprisingly, predictably low response times and power are the most important. The impact of these computers on driving range is clearly non-trivial.

- 99.99th percentile tail latency of 100ms would be faster than fastest human reaction times (target supported by industry, but will likely be a moving target); 10 frames/sec
- Power and cooling (need cooling because the computers will be inside the cabin); 77W cooling for every 100W dissipated; a 400W computer reduces MPG by 1; a CPU+3GPU system (~1000W) lowers driving range by 6% (and 11.5% once you add storage+cooling)
- Also need high storage and reliability (recall that Tesla FSD executes every computation twice)

Algorithms

Of these algorithms, the localization one is new. But they aren't very clear about what the feature extraction algorithm does (the blog post linked at the end has more details). The trig calculations are simplified with look-up tables.

- Object detection: YOLO; DNN based
- Object tracker: GOTURN; DNN based and a few book-keeping tables
- Localization: ORB-Slam; lots of trigonometric calculations
- Motion and Mission planning: MOTPLAN and MISPLAN from Autoware

Bottleneck Analysis

The CPU solution is clearly sub-optimal. This also shows that most of the time is in those three phases, with much of it spent in DNNs.

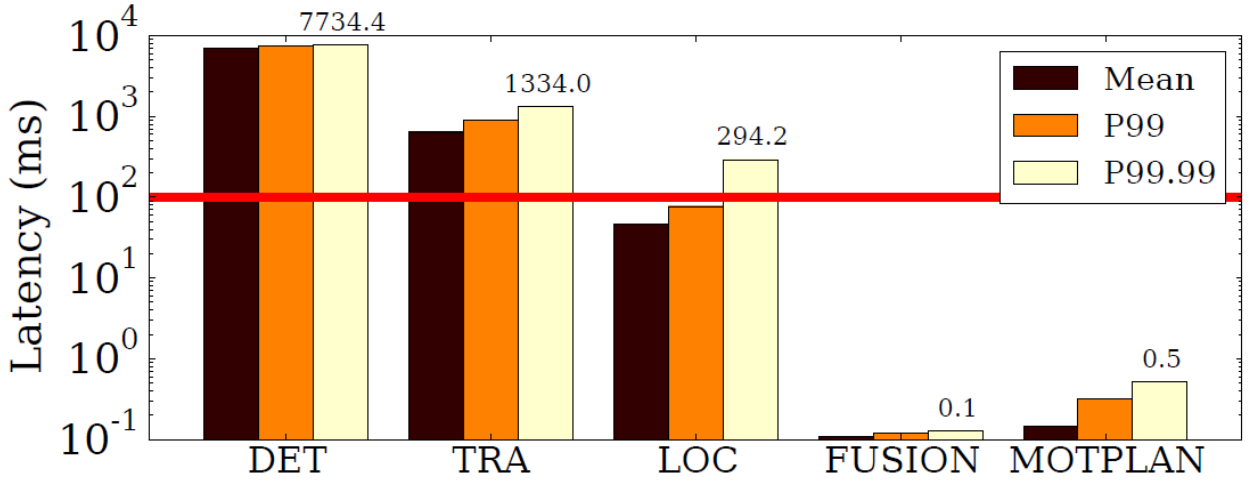
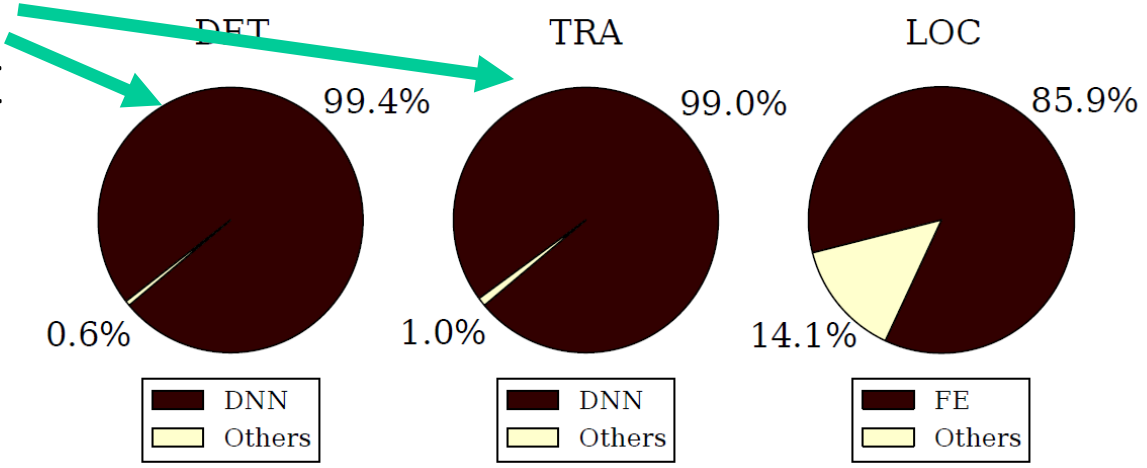


Figure 6. Latency of each algorithmic component on a multicore CPUs system in the end-to-end autonomous driving

DNNs implemented with Eyeriss and EIE



Localization

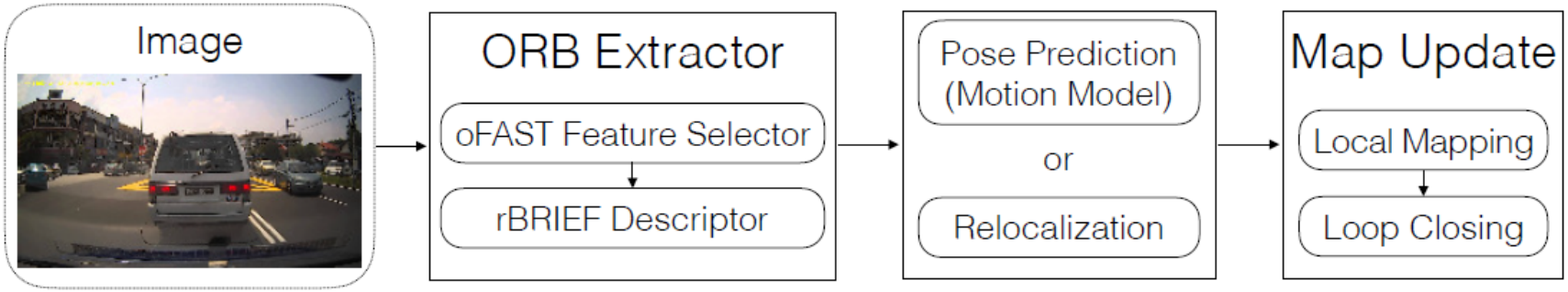
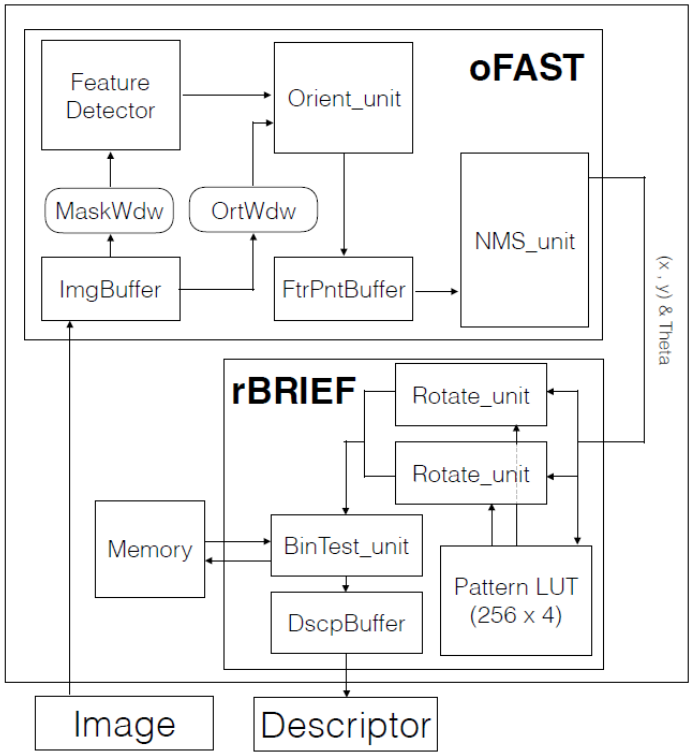


Figure 5. Overview of the localization engine (LOC). The algorithm takes images as input and extracts interesting features such as landmarks, and then generates a descriptor for each of the extracted features. The feature descriptors are consumed to locate the vehicle and predict vehicle poses.

Feature Extraction

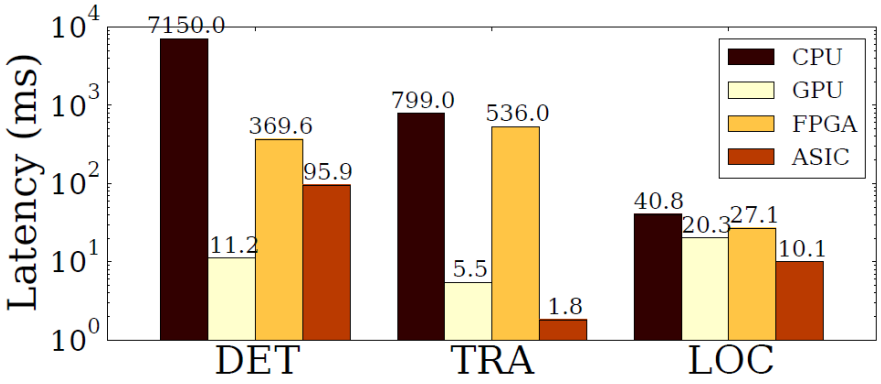


- Ops in feature detector not clear (a blog post that may be helpful is listed on the last slide)
- Rest are trig ops that are implemented with LUTs

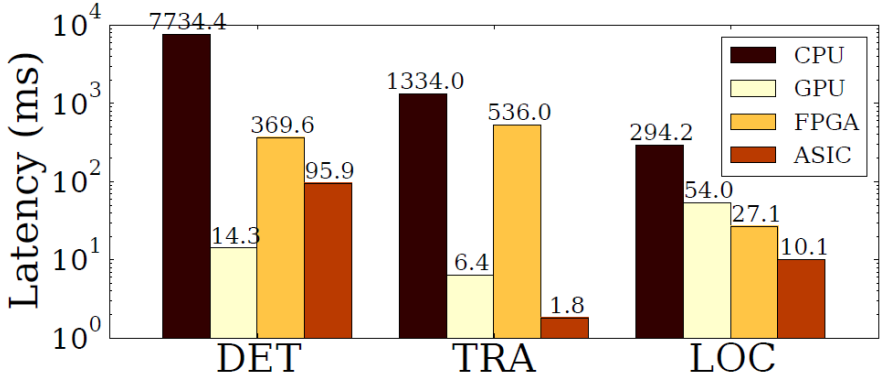
Figure 9. Diagram of our implementation of Feature Extraction (FE) on FPGAs. As the input images streaming into the image buffer (ImgBuffer), they are filtered by the mask window (MaskWdw). The feature detector extracts the features of interest and store them into the feature point buffer (FtrP-

Results

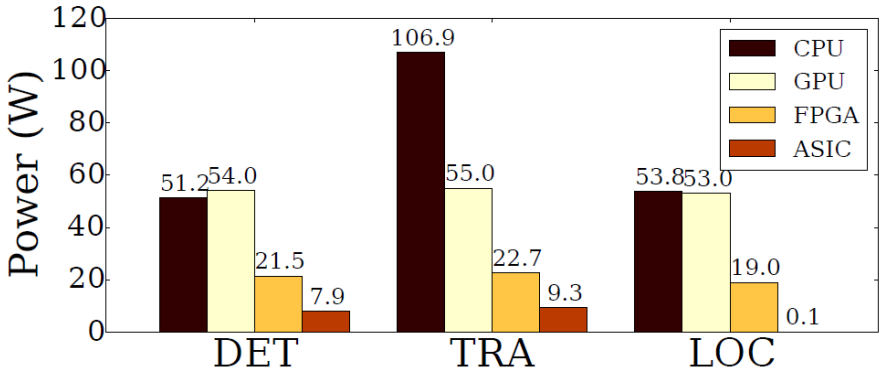
They do much better with ASICs. The GPU does great too, but consumes too much power. The FPGA is fine if an ASIC is not possible. The overall latency is close to the 100ms target.



(a) Mean Latency Across Platforms



(b) 99.99th-Percentile Latency Across Platforms

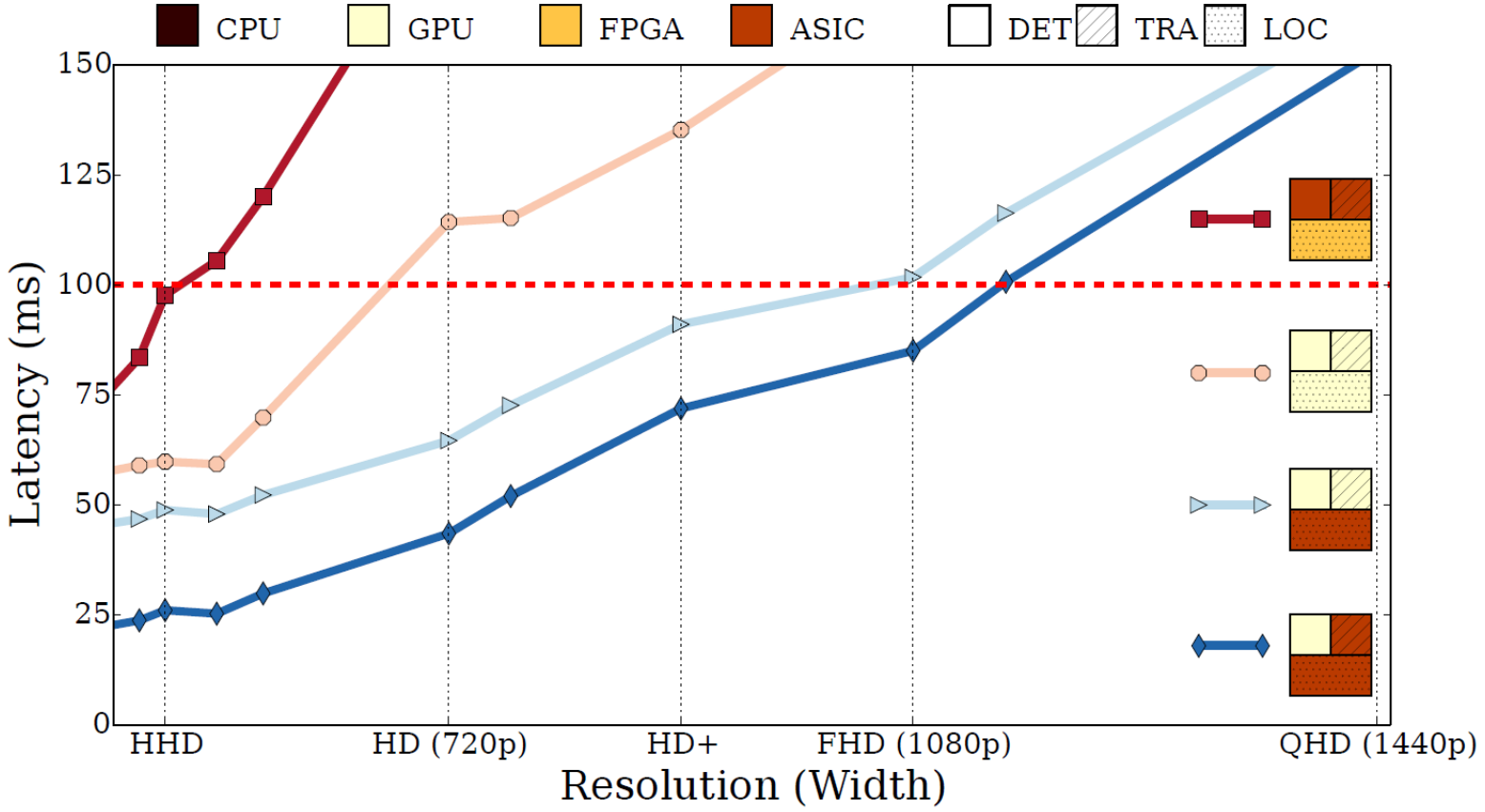


(c) Power Consumption Across Platforms

- Power for a single camera (Tesla has 8)
- For the whole system, GPU lowers driving range by 12% and ASICs by 2%

Scalability Results

The accuracy improvement with higher resolution is compelling. But they show that latency is too high. Can probably do it with multiple ASICs. The system will also scale up along other axes: more cameras, higher resolution, more redundancy, larger networks.



- VGG accuracy improves from 80% to 87% when resolution is doubled
- Not clear how the bottlenecks change (compute, memory, reuse patterns)

Summary

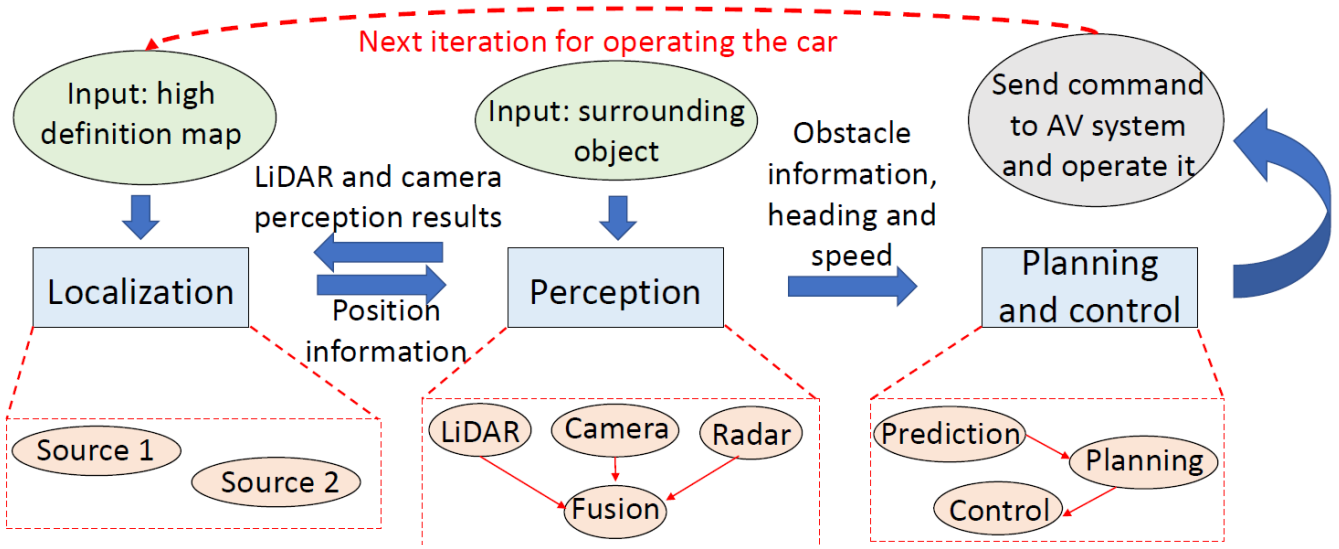
- Detection, Localization, Tracking, Planning are major steps
- The first three take up 94% of compute
- Contributions: a pipeline with publicly available frameworks, bottleneck analysis, acceleration with GPU/FPGA/ASIC
- GPUs and ASIC offer two orders of magnitude speedup, but GPUs consume too much power
- More work remains – lower latency, lower power, higher resolution, feature extraction, other bottlenecks

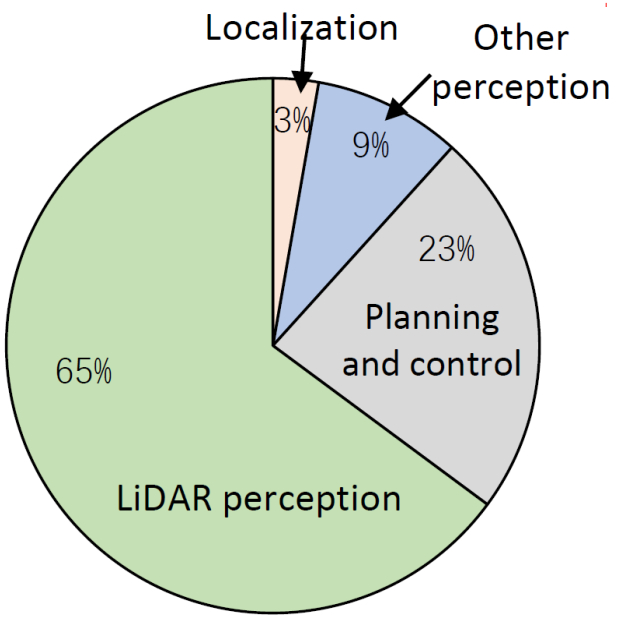
LIDAR-based Approach [Zhao et al.]

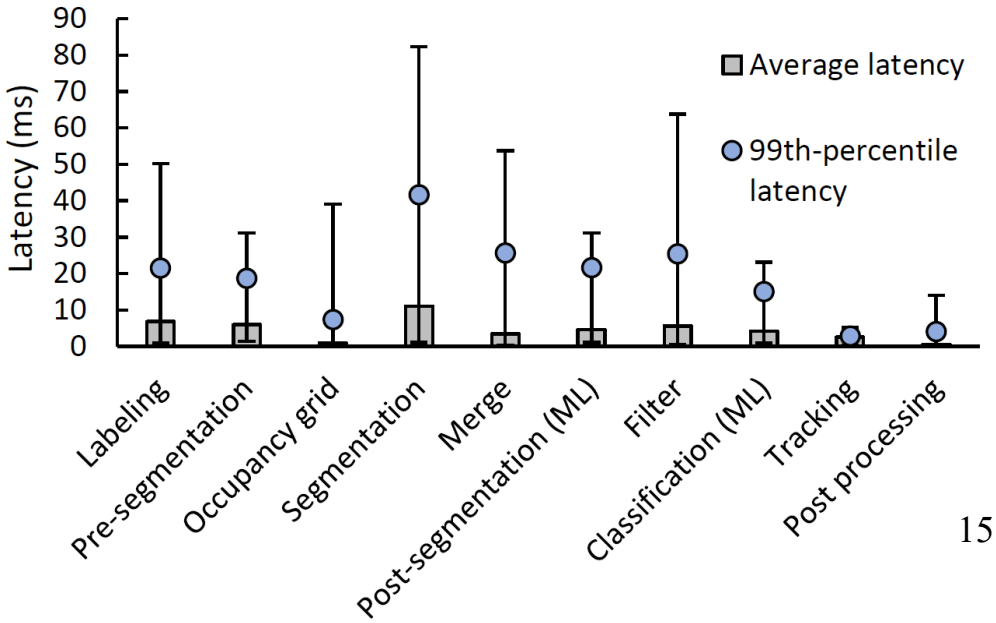
- Joint work with Pony.AI; characterization based on several hours of driving on their test fleet over 3 months
- They rely on LIDAR and their software pipeline has relatively little DNN

The second paper considers a LIDAR-based pipeline. It does have a limited amount of DNN computation. But as shown in the next slide, 65% of the computation is LIDAR based perception. This last computation is itself spread across several functions, of which Segmentation is the largest (the computations that identify/separate major objects from the background). This makes it harder to create an accelerator that improves all the functions. It turns out that LIDAR perception has varying latency based on the nature of the point cloud. Scenes with more nearby objects take longer to analyze; this is especially problematic since those are the scenes that need speedy resolution.

Software Pipeline

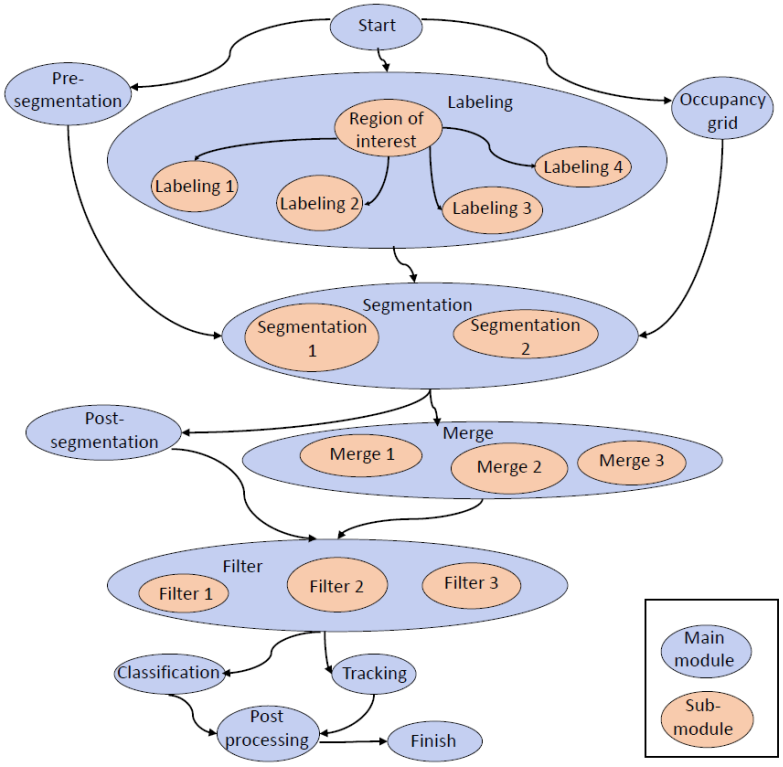






LIDAR Pipeline

- Several steps, with segmentation being a major overhead
- Segmentation extracts useful semantic info (relevant objects) from the background
- The latency is worse when there are more nearby objects, since there are more reflected points to deal with
- This leads to high variance, especially when collisions are more likely



LIDAR Perception Latency

The authors show that if LIDAR perception latency exceeds the LIDAR sampling rate (100ms), the impact on other stages of the software pipeline is even higher.

- Max time is a relevant metric because it disproportionately impacts safety
- They observe that long LIDAR perception latency also disproportionately impacts overall latency

(The LIDAR sampling rate is 100ms)

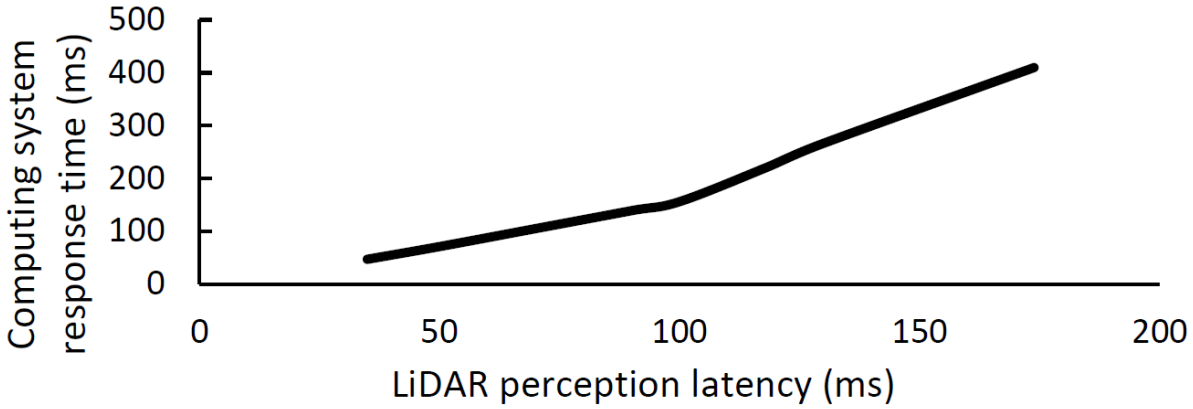


Figure 6: The relationship between LiDAR perception latency and computing system response time.

Safety Score

To improve performance, they first predict if the point cloud will take longer to analyze, if yes, then they re-allocate CPU/GPU resources across the many software functions. The impact is evaluated with a safety score that shows how the response latency ultimately impacts safety. The equation factors the sampling rate, vehicle acceleration and velocity.

- They compute a safety score that factors in the response time along with velocity/acceleration
- They design a predictor to estimate response time based on the count/proximity of objects in the scene and how hardware resources (CPU/GPU) are allocated to each part of the software pipeline

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

Variable	Description
t	Instantaneous computing system response time, defined by Equation 2.
θ	Response time window.
α, β	Indicating the velocity and acceleration of an AV and surrounding vehicles, defined by Equation 8 in Appendix A.
σ, η	Reward or penalty on the level of safety, when t is lower or higher than θ, respectively.

References

- “The Architectural Implications of Autonomous Driving: Constraints and Acceleration”, S.-C. Lin et al., ASPLOS 2018
- “Towards Safety Aware Computing System Design in Autonomous Vehicles”, Zhao et al., Arxiv, 2019
- “Driving into the Memory Wall”, Jung et al., MemSys 2018
- ORB-Slam details:

  https://medium.com/software-incubator/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf