# Software Engineering for Safe Self-Driving

Uber ATG

(Owen) Shang-Wen Cheng
2020.12.02
System Safety Software Engineer

# Outline

- Intro of ATG missions, vision, videos
- Self-driving basics on the Volvo XC-90 platform
- ATG’s Safety Case Framework and approach
- Discuss software engineering challenges
- Q&A

# Quick Bio: I've created software in many domains

- I'm continuing to find ways to incorporate practical lessons
  - … from software architecture, self-healing systems, and first-principles analysis
  - … into routine system and software engineering tasks
- Fun facts of this curious coder
  - First shell script: Lotus 1-2-3 application menu via DOS AUTOEXEC.BAT
  - First application: Wedding Cake POS system in dBase III Plus
  - First real-world job: Florida DHSMV Driver License Query System in Java + Oracle PL/SQL
- 8 yrs piling higher and deeper in Software Engineering at Carnegie Mellon U
  - Thesis - *Rainbow:  Cost-Effective Software Architecture-Based Self-Adaptation*
- 6 ½ yrs at NASA Jet Propulsion Laboratory as FSW Engineer plus Flight Ops
  - SMAP + StateChart Autocoder + SMAP Vitals
- 4 ½ yrs so far at Uber Advanced Technologies as System Software Engineer
  - Roles spanned log data (telemetry) offload, data platform, and system software safety

Uber ATG

# Why Self-Driving?

## Self-driving Matters for the world

Save lives.
Save time.
Save space.

→

## Self-driving matters for Uber

Providing safe, reliable, cost effective transportation is our priority.

→

## Uber matters to self-driving

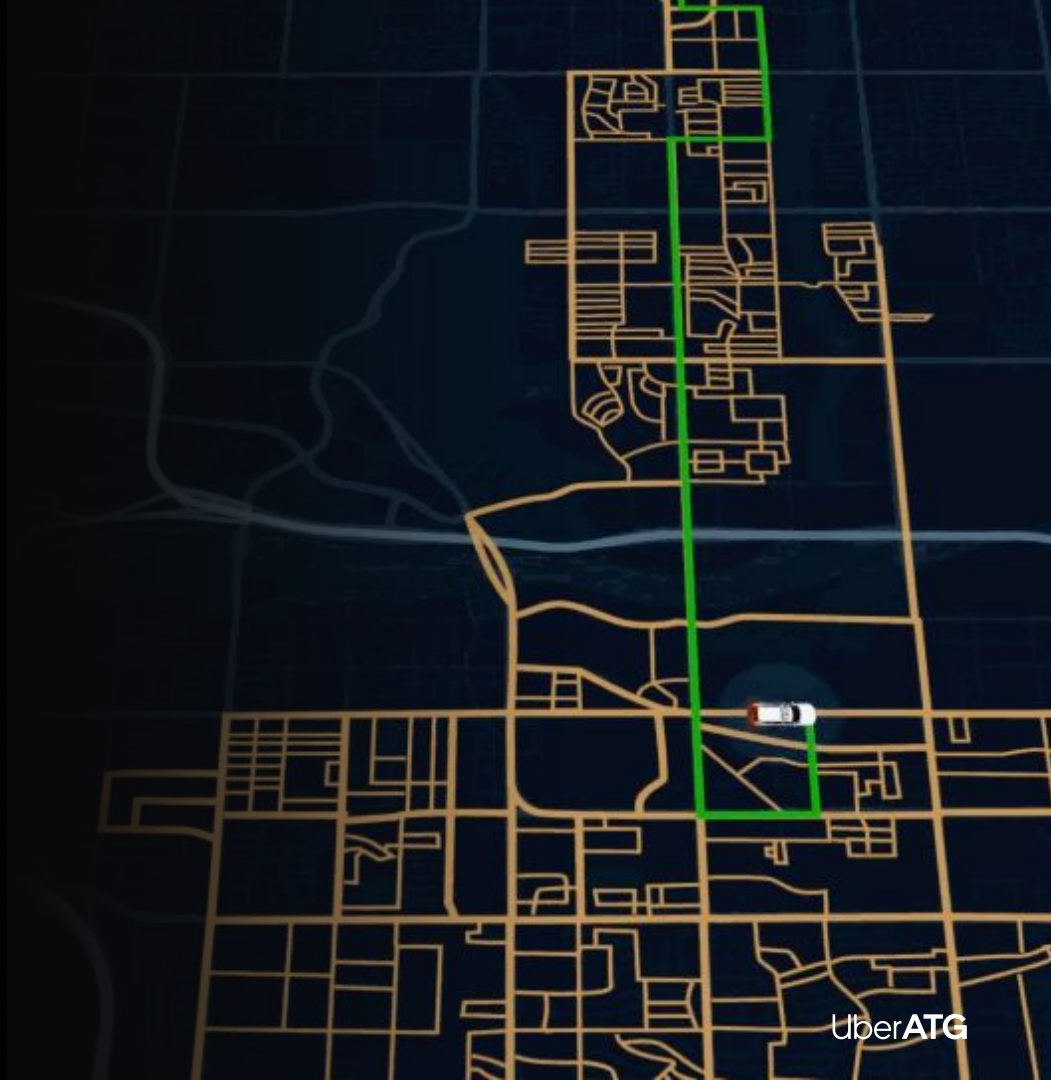
Our network allows us to scale self-driving globally.

# Our Goal

Self driving transportation for everyone and everything

## ATG Mission

Introduce self-driving technology to the Uber network in order to make transporting people and goods safer, more efficient, and more affordable around the world.

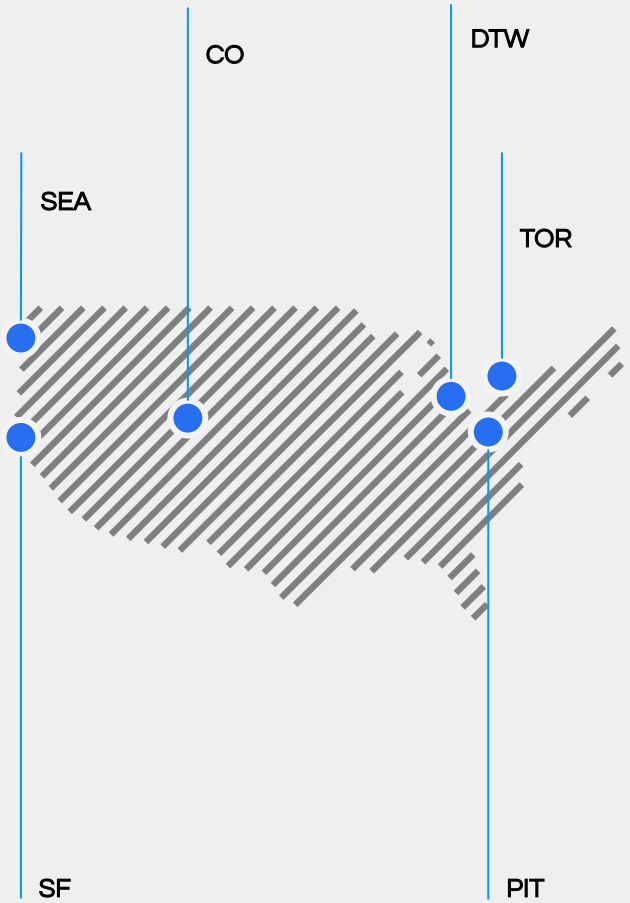


Vehicles At Scale

Self-Driving Systems

Fleet Operations

The Network




# >1000 Employees

We comprise:

- System Engineers
- Safety Engineers
- Mechanical and Hardware Engineers
- Software Engineers / DevOps / Infra
- Product and UX Engineers
- Program and Product Managers
- ...

UATG Careers URL:

https://www.uber.com/us/en/careers/teams/advanced-technologies-group/

**Side and rear facing cameras** work in collaboration to construct a continuous view of the vehicle's surroundings

**Top mounted lidar** provides a 360° 3-dimensional scan of the environment

**Forward facing camera array** focus both close and far field, watching for braking vehicles, crossing pedestrians, traffic lights, and signage

Roof mounted antenna provide **GPS positioning and wireless data capabilities**

**Automatic Emergency Braking system** operates independently as a safeguard for certain situations requiring activation of the vehicle braking system


**360° radar coverage** detects vehicles and other obstacles

**Custom designed compute and storage** allow for real-time processing of data while a fully integrated cooling solution keeps components running optimally

**Gateway Module** serves as a gateway to the base Vehicle Platform from the Uber Self Driving System, translating messages and commanding the vehicle's actuators (brakes, throttle, steering)

**Modified base Vehicle Platform** with Uber-specific mounting provisions, electrical harness, cooling interface, interior trim, and software control API

# Self-Driving Vehicle Basics

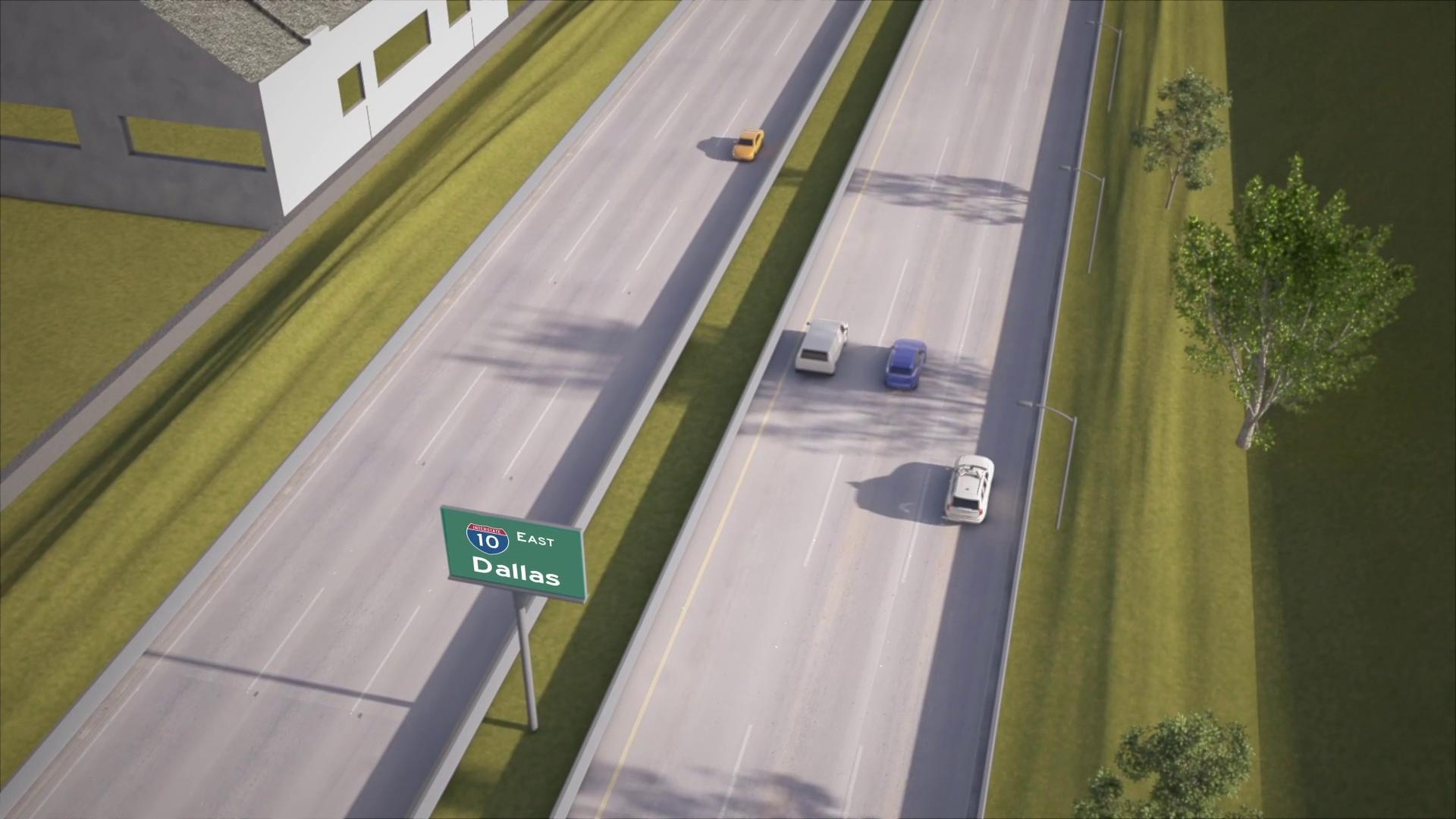
Camera
LiDAR
Radar
Ultrasonic
GPS
IMU
Wheel Encoder



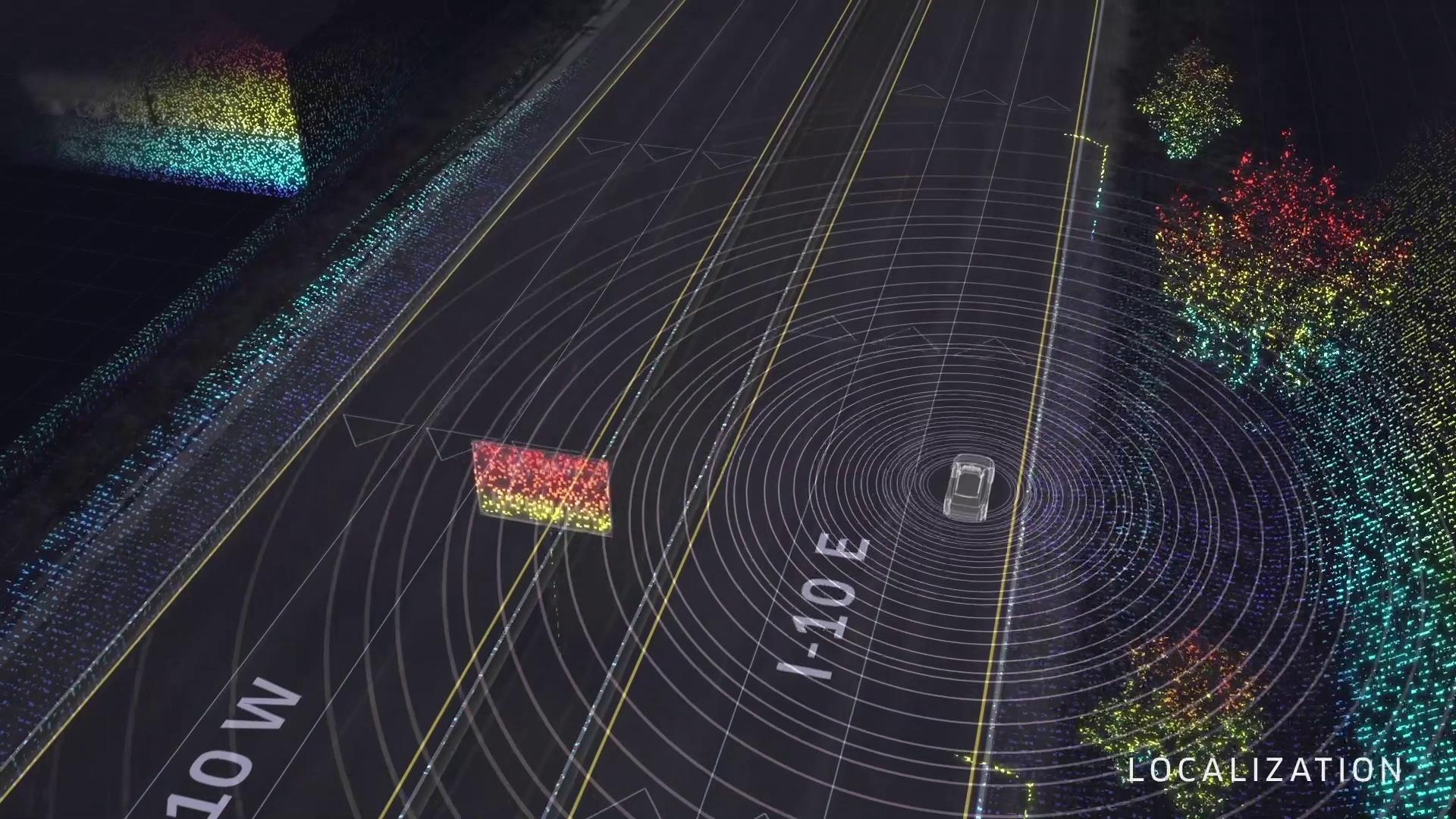
Perception
Prediction
Motion Planning
Control
Maps
Localization
Routing



Steering
Braking
Propulsion

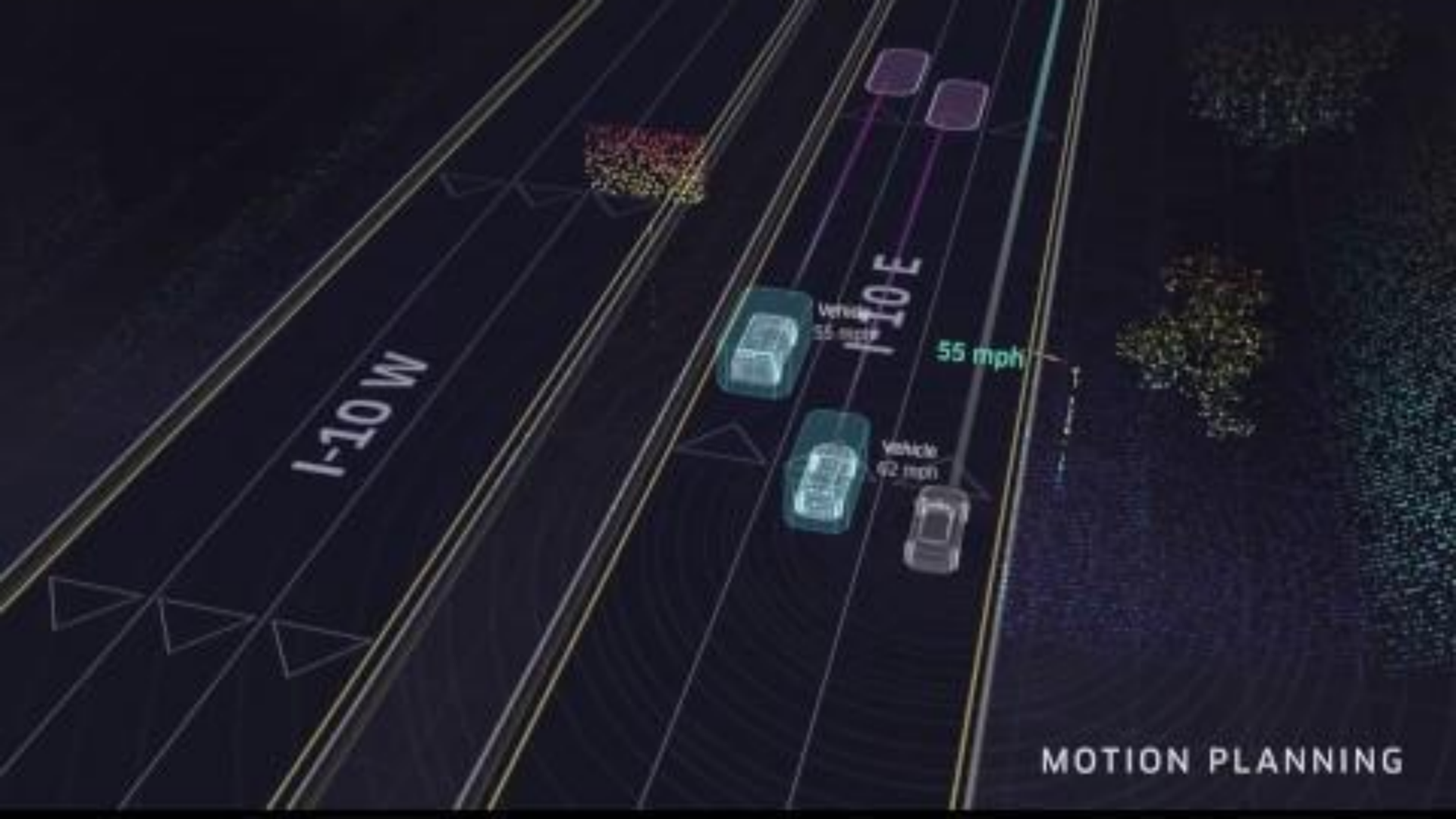

INTERSTATE
10
EAST
Dallas


MAP


I-10 E
LOCALIZATION


LIDAR
10
EAST
Dallas
PERCEPTION


I-10 W
I-10 E
Vehicle
56 mph
Vehicle
62 mph
PREDICTION


I-10 W
I-10 E
55 mph
MOTION PLANNING


I-10 W
I-10 E
Vehicle
56 mph
Vehicle
62 mph
CONTROL

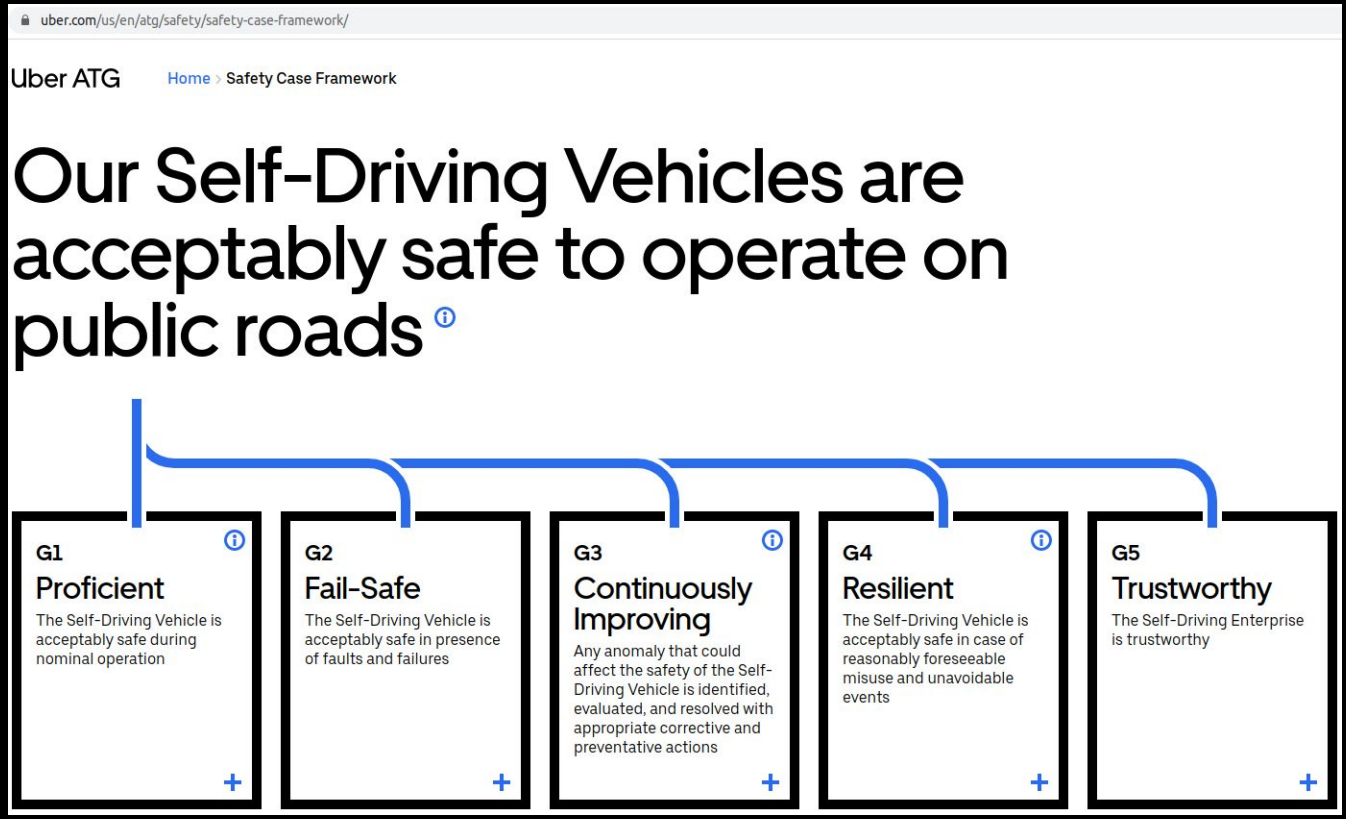

Controls
Motion Planning
Prediction
Perception
Sensors
HD Map and Localization

Critical Thinking Time:
What do you consider the biggest challenges to realizing this tech?

Source: https://uber.app.box.com/v/UberATGSafetyReport ; https://www.uber.com/us/en/atg/safety/safety-case-framework/

# UATG Safety Case Framework

- **Proficient**: Absent system faults, system is acceptably safe during nominal operations
- **Fail-Safe**: System is acceptably safe in the presence of faults and failures; mitigates harm
- **Continuously Improving**: Dev & ops processes identify, evaluate, resolve anomalies that can potentially affect safety of the self-driving vehicle (SDV); strong safety culture: employees all levels empowered & accountable for active participation
- **Resilient**: SDV is acceptably safe in case of reasonably foreseeable misuse or unavoidable events
- **Trustworthy**: Earn & keep trust of our riders, regulators, legislators, public safety officials, other road users, and advocacy organizations; provide them evidence of the safety measures of our self-driving enterprise

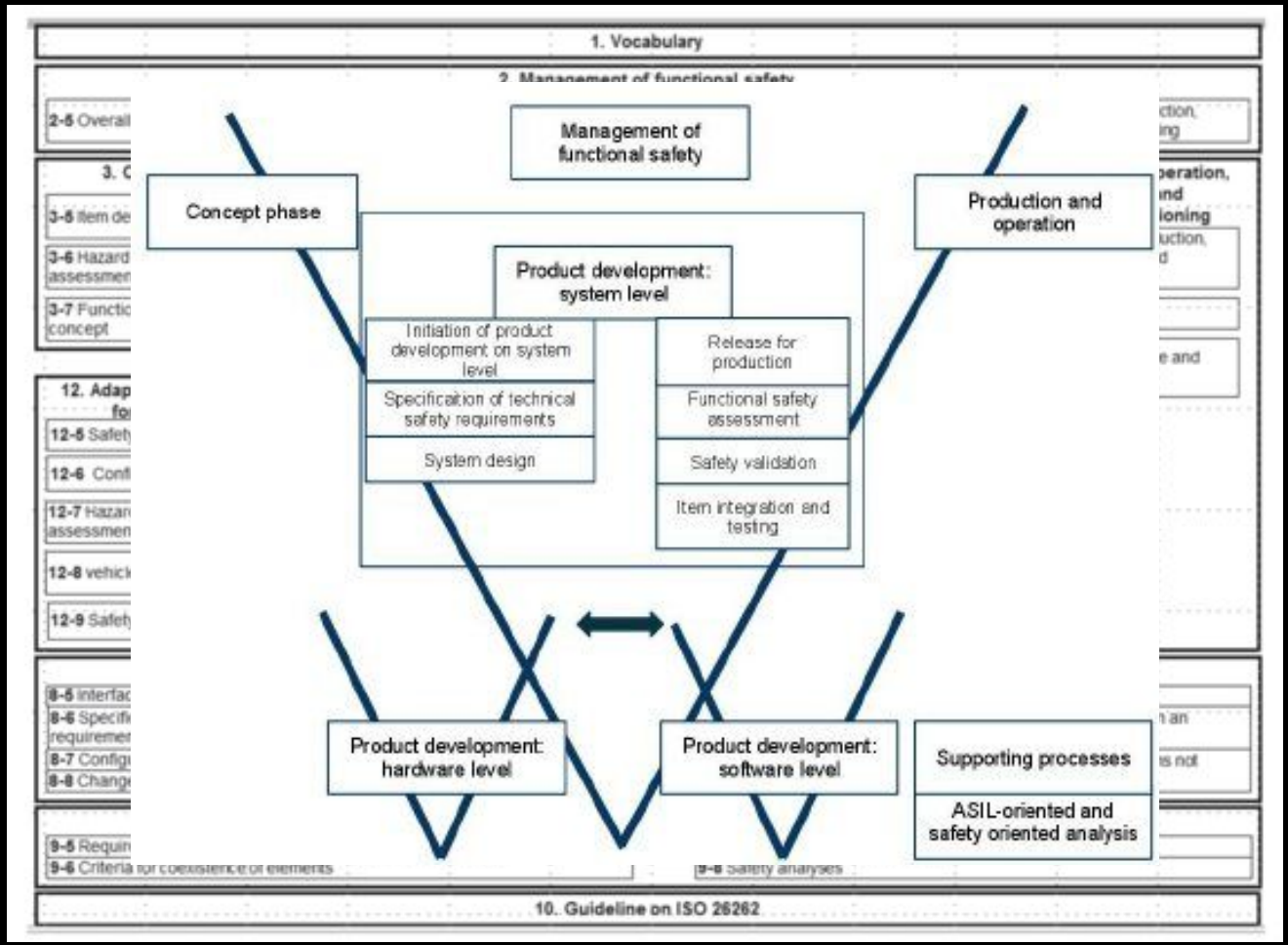


# Safety Case Framework Incorporates Safety Standards

- ISO 26262, aka *Functional Safety*: Acceptable level of safety for road vehicles
  - Requires absence of **unreasonable risk** caused by every hazard associated with system functionality and implementation
  - Particularly, malfunctions of the electrical and/or electronic (E/E) systems, including random HW failures, systematic failures
  - 26262 defines how to assess hazards, to define top-level safety requirements (safety goals), rigors of development, etc.
- ISO 21448, aka *Safety of the Intended Functionality* (SOTIF)
  - Some systems rely on sensing external/internal environment
  - Where hazardous behavior can be caused by intended functionality or performance limitation of system free from faults addressed in 26262
- UL 4600, latest standard on system safety
  - Complements the above (and others)
  - Specifically focuses on systems containing machine learning algorithms and data

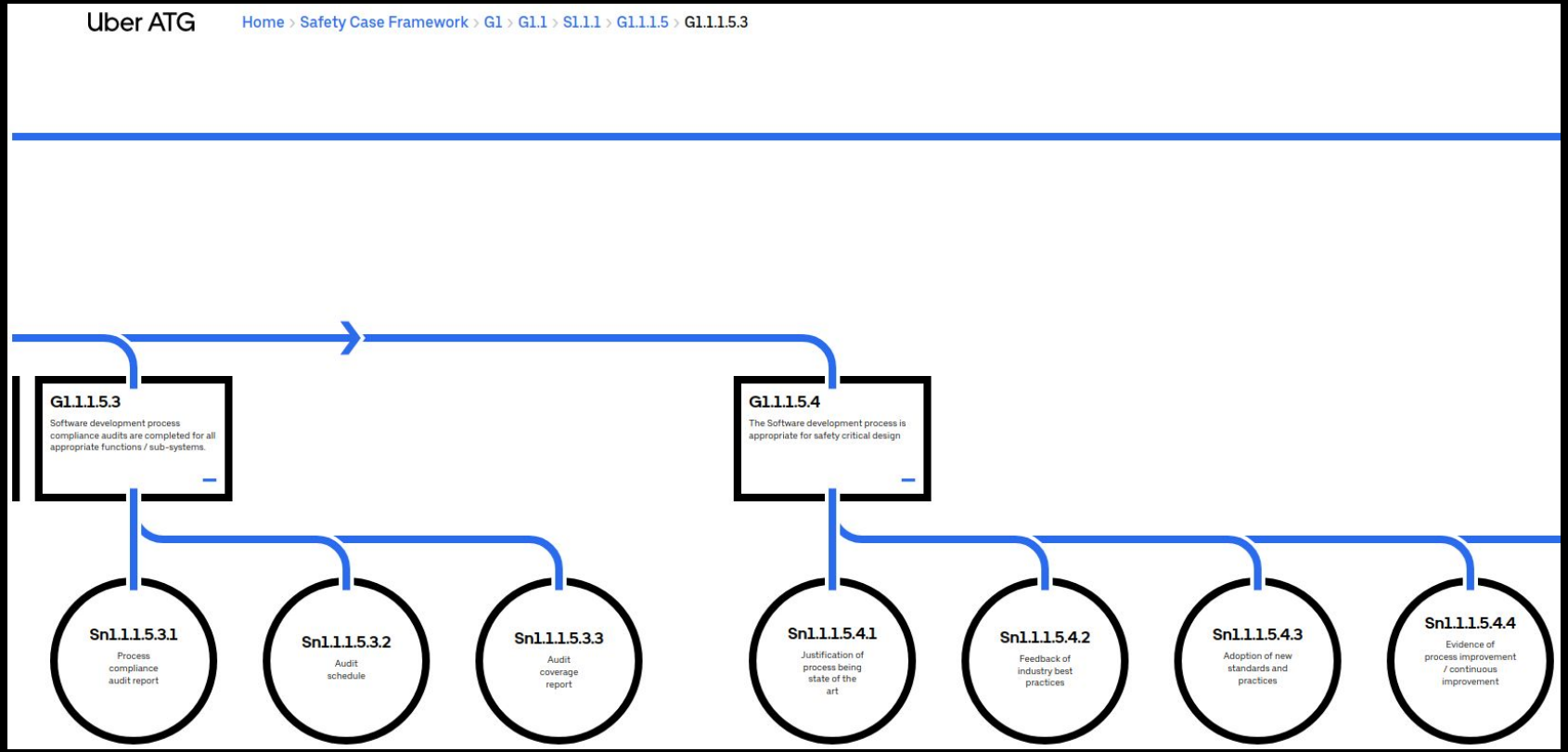
# System Engineering Processes

Sources: https://www.iso.org/obp/ui/#iso:std:iso:26262:-2:dis:ed-2:v1:en:fig:1 ; https://www.researchgate.net/figure/V-Model-from-ISO-26262_fig1_228746867



Source: https://www.uber.com/us/en/atg/safety/safety-case-framework/

# A Glimpse of the Software Engineering Claims



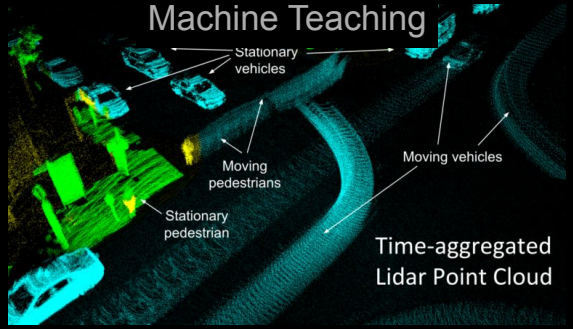
# Requirements Engineering for Self-Driving Software

- Traditional systems engineering demarcates problem and solution spaces
- Self-driving software engineering poses challenges to that approach
  - Problem space requires analysis and data-driven understanding
  - Solution options require testing to validate
  - Ideally, fast iteration for development velocity
  - Software engineers are domain experts
  - How to make this happen?
- Motivating jaywalking scenario:
  - How do human drivers handle jaywalking?
- Scenarios as proxy for requirements
  - How many types of scenarios do we handle?

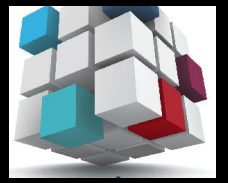
Critical Thinking Time:
How would you engineer the software for a self-driving car?

Sources: https://eng.uber.com/avs-autonomous-vehicle-visualization/ ; https://eng.uber.com/machine-learning/ (Fig. 10) ; https://www.uber.com/us/en/atg/ ; https://www.geekwire.com/2019/uber-acquires-seattle-startup-mighty-ai-fuel-push-self-driving-cars/

# Ecosystem of Self-driving Software Development

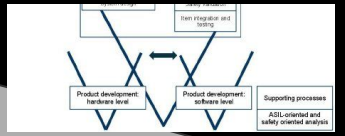














Safety Management System

Source: https://www.shopulstandards.com/ProductDetail.aspx?productid=UL4600

# What Makes Self-Driving Software Hard?

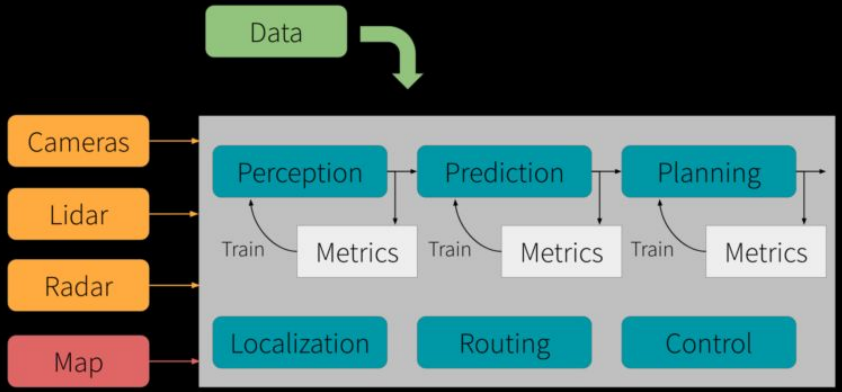
UL 4600 Section 8 provides specific guidance on Autonomy Functions and Support:

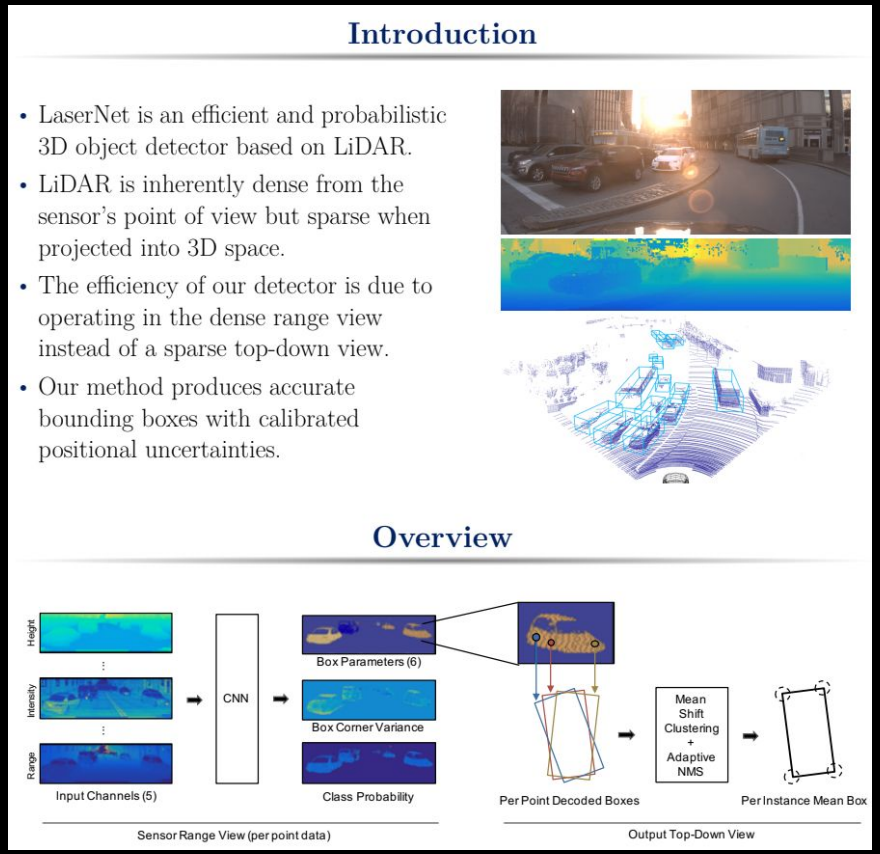
- 8.1 Hazards related to autonomy have been identified and mitigated; concept of operations described.
- 8.2 The Operational Design Domain (ODD) shall be defined in an acceptably complete manner, violations handled safely, changes detected and tracked to resolution.
- 8.3 The sensors shall provide acceptably correct, complete, and current data (within the defined ODD).
  - Calibration, data filtering, data processing, data identification techniques result in acceptable performance
  - Sensor fusion and redundancy management techniques shall be used as necessary
  - Risks resulting from potential sensor performance degradation shall be mitigated
- 8.4 Perception shall map sensor inputs to the perception ontology with acceptable performance.
- 8.5 The machine learning architecture, training, and V&V approach shall provide acceptable performance.
  - 8.5.3 Machine learning training and V&V shall use acceptable data.
  - 8.5.4 Machine learning-based functionality shall be acceptably robust to data variation.
  - 8.5.5 Post-deployment changes to machine learning behavior shall not compromise safety.
- 8.6 Planning capabilities are acceptable, documented, have acceptable V&V, and risks from failures mitigated.
- 8.7 Prediction functionality shall have acceptable performance.
- 8.8 Trajectory and system control shall have acceptable performance.
- 8.9 Actuator faults shall be detected and mitigated.
- 8.10 Timing performance of autonomy functions shall be acceptable.

Sources: https://arxiv.org/abs/1903.08701 ; https://uber.app.box.com/v/UberATGSafetyReport ; https://eng.uber.com/machine-learning (Fig. 9)

# Perception and Prediction Software

- Perception (ref. LaserNet)
  - Sensors ⇒ Detect & track objects by types
  - Fallback: tracks objects by ballistic motion
- Prediction
  - Estimates object future motion & likelihood
  - Presents intents of actor to be in SDV path
- Training Pipeline (below)



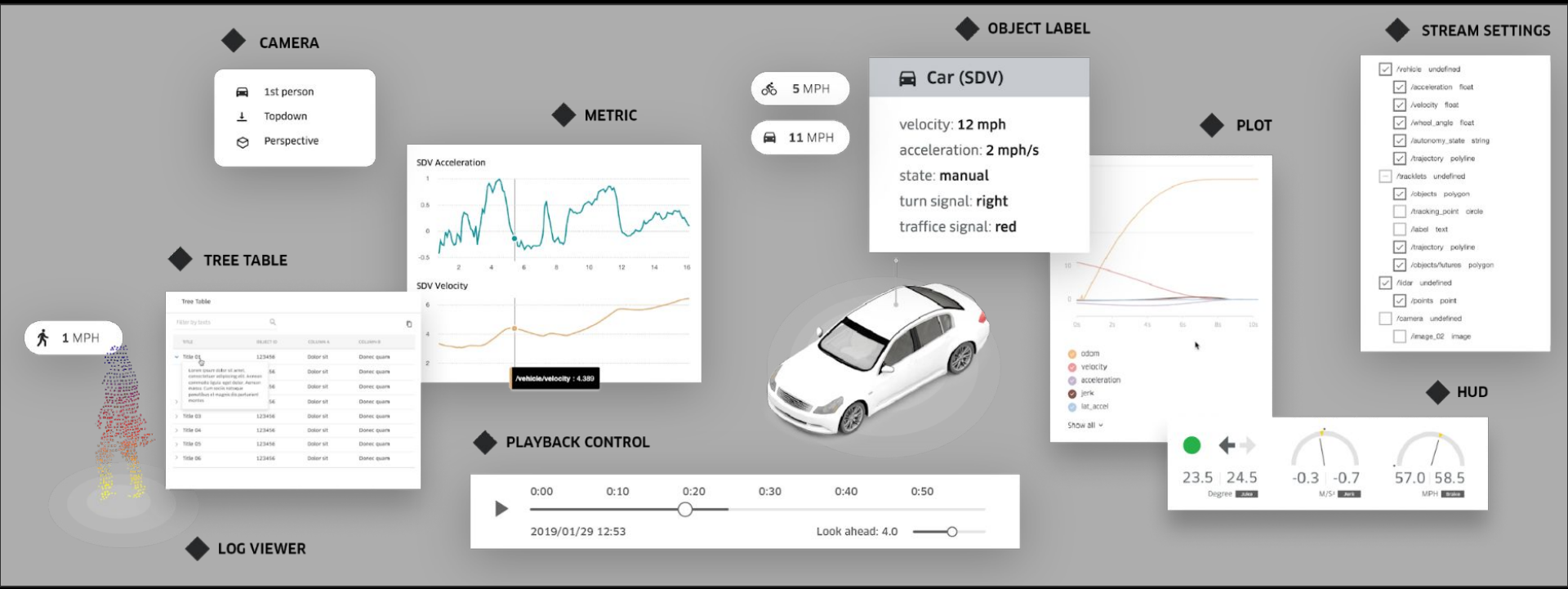



# What Makes Testing Hard?

- Ginormous V&V combinatorics
  - Take camera sensor, for example, and consider the following ingredients:
    - Just one camera, 4096x2168 pixels, 24-bit RGB, 24 fps
    - Single scene detected for one second
  - Assume we want to test proper perception, prediction, and motion plan for all possible inputs
  - What is the state space?
    - 1 sec x 24 frame/sec x 2^24 /bit x (4096 x 2168) bit/frame = 3,575,611,813,527,552
- Simulation is essential for testing
  - Must weigh fidelity, from sensors to vehicle dynamics
    - Sensor simulation super complex: physics of photons picked up by sensors
    - Vehicle dynamics also super complex: the physics of control on road surface
  - Complicated by other environmental factors, e.g., lighting, visibility, humidity, etc.

Sources: https://eng.uber.com/avs-autonomous-vehicle-visualization/

# System Visualization Key Enabler for Development



# What are your key takeaways for engineering AI-enabled system software?

# My Experiences for Software Engineering Students

- Engineering in general, software engineering in particular:
  - Requirements engineering: thoroughly understand the problem space before tackling solution
    - In today's tech world, we tend to go seeking nails for the hammer we ourselves possess
    - Problem space analysis takes a diverse team of experts: consider jaywalking problem
  - Abstractions: It's a skill not just for software engineers
    - But, very useful to tackle the complex problems of the world today
  - Range: Applying ideas from different domains to your current area
- Suggestions: expand not just your depth, but breadth! ⇒ T/X/Γ-shaped skills
  - My own:
    - Read 100 books a year to expand horizons (one Senator's family practice inspired me)
      - Blinkist is a great resource! My invite link: http://blinki.st/d1a17e4dce19

# Q&A