Forward-Looking Statements

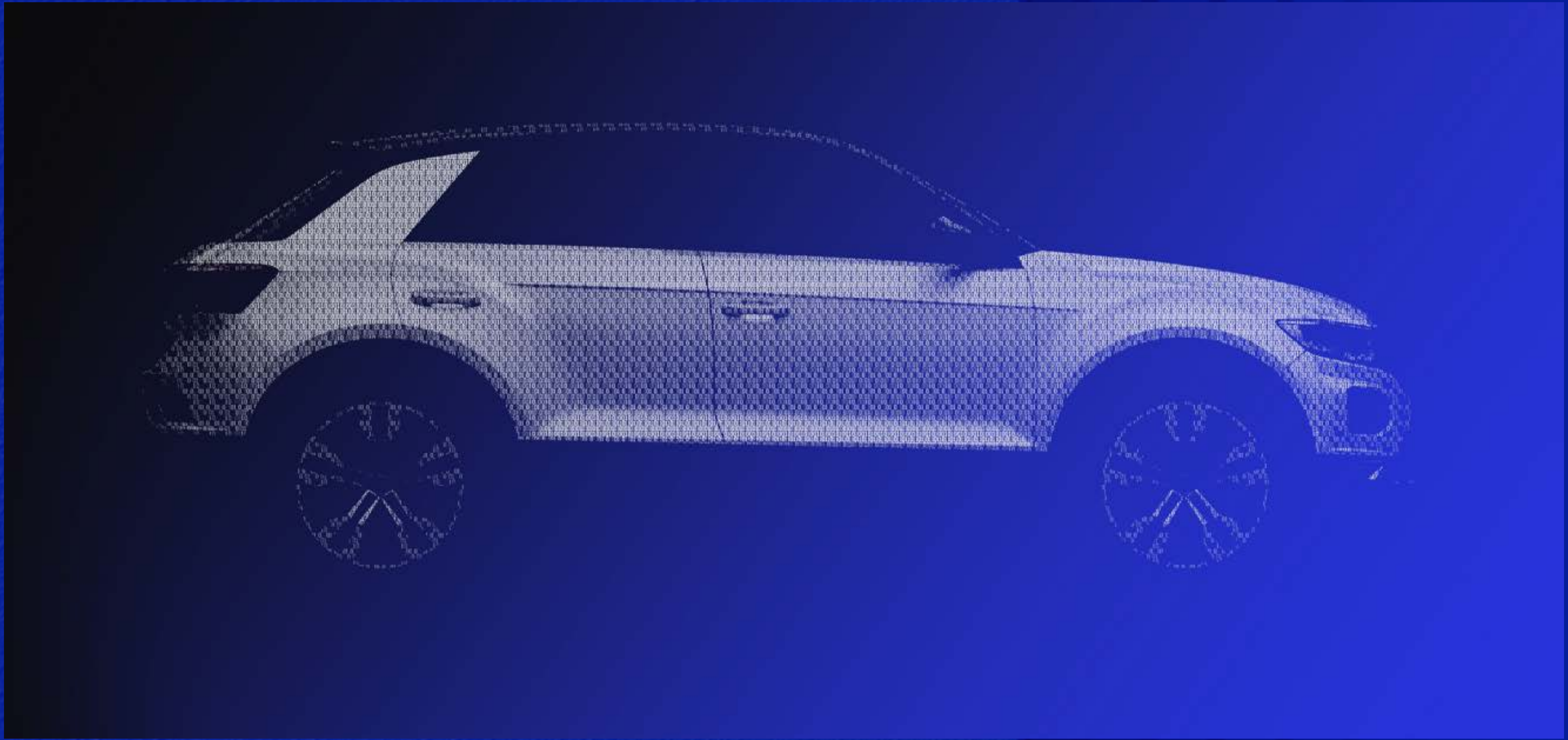
Mobileye's business outlook, guidance and other statements in this presentation that are not statements of historical fact, including statements about our beliefs and expectations, are forward-looking statements and should be evaluated as such. Forward-looking statements include information concerning possible or assumed future results of operations, including descriptions of our business plan and strategies, and in particular include statements about anticipated future orders. These statements often include words such as "anticipate," "expect," "suggests," "plan," "believe," "intend," "estimates," "targets," "projects," "should," "could," "would," "may," "will," "forecast," or the negative of these terms, and other similar expressions, although not all forward-looking statements contain these words. We base these forward-looking statements or projections on our current expectations, plans and assumptions that we have made in light of our experience in the industry, as well as our perceptions of historical trends, current conditions, expected future developments and other factors we believe are appropriate under the circumstances and at such time. You should understand that these statements are not guarantees of performance or results. The forward-looking statements and projections are subject to and involve risks, uncertainties and assumptions and you should not place undue reliance on these forward-looking statements or projections. Although we believe that these forward-looking statements and projections are based on reasonable assumptions at the time they are made, you should be aware that many factors could affect our actual financial results or results of operations and could cause actual results to differ materially from those expressed in the forward-looking statements and projections.

Other important factors that may materially affect such forward-looking statements and projections include the following: future business, social and environmental performance, goals and measures; our anticipated growth prospects and trends in markets and industries relevant to our business; business and investment plans; expectations about our ability to maintain or enhance our leadership position in the markets in which we participate; future consumer demand and behavior; current or future products and technology, and the expected availability, specifications and benefits of such products and technology; development of regulatory frameworks for current and future technology; projected cost and pricing trends; future production capacity and product supply; potential future benefits and competitive advantages associated with our technologies and architecture and the data we have accumulated; the future purchase, use and availability of products, components and services supplied by third parties, including third-party IP and manufacturing services; uncertain events or assumptions, including statements relating to our estimated vehicle production and market opportunity, potential production volumes associated with design wins and other characterizations of future events or circumstances; future responses to and effects of the COVID-19 pandemic; adverse conditions in Israel, including as a result of war and geopolitical conflict, which may affect our operations and may limit our ability to produce and sell our solutions; any disruption in our operations by the obligations of our personnel to perform military service as a result of current or future military actions involving Israel; availability, uses, sufficiency and cost of capital and capital resources, including expected returns to stockholders such as dividends, and the expected timing of future dividends; tax- and accounting-related expectations.

Detailed information regarding these and other factors that could affect Mobileye's business and results is included in Mobileye's SEC filings, including the company's Annual Report on Form 10-K for the year ended December 31, 2022, particularly in the section entitled "Item 1A. Risk Factors". Copies of these filings may be obtained by visiting our Investor Relations website at ir.mobileye.com or the SEC's website at www.sec.gov.

Driving-Experience-Platform: Architecture, Abstractions, APIs

mobileye™



Prof. Shai Shalev-Shwartz, CTO

Jan. 2024

Outline

What is a development platform, and why should you care?

Why previous platforms for self-driving have not been successful?

- The Sense-Plan-Act methodology
- The Differentiability-Scalability-Risk tradeoff
- The underestimation plague

Mobileye’s Driving-Experience-Platform (DXP)

- The Universal vs. Unique separation
- The When-What-How abstraction
- DXP solves the Expressivity-Scalability-Risk tradeoff

The main ingredients of the platform’s backbone

What is a Development Platform and Why Should You Care?

EXAMPLES:

Operating system
Linux, Windows, iOS, Android, etc.

Programming languages
C++, Python, Java, Swift, Cuda, etc.

Task specific developer packages
PyTorch, Spark, etc.

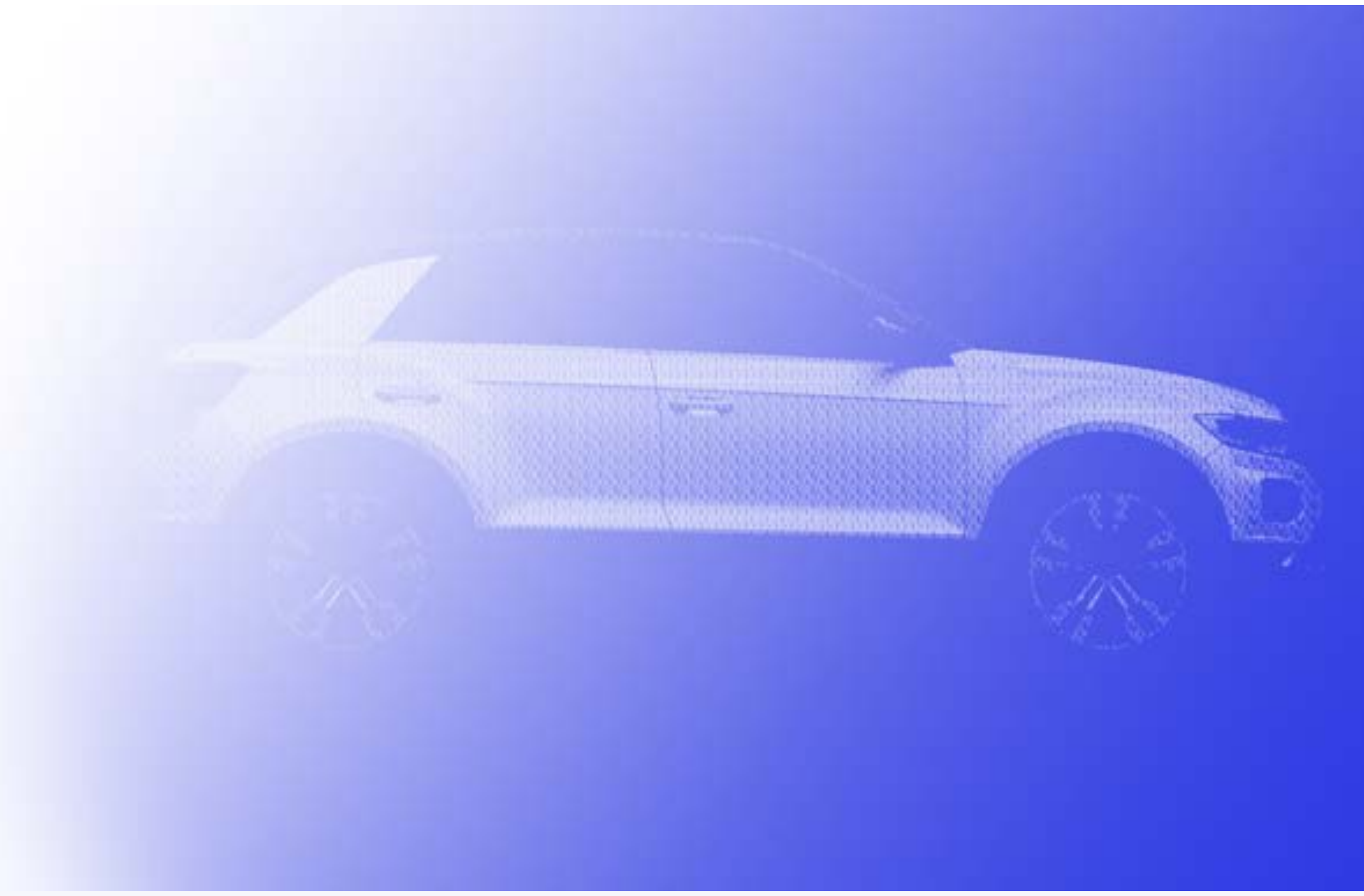
High-level interfaces
Chat-GPT, Wix, etc.

WHY USING A PLATFORM?

"Don't re-invent iOS when developing an iPhone app..."

For the user → save time and resources

For the supplier → enable scale



Outline

What is a development platform, and why should you care?

Why previous platforms for self-driving have not been successful?

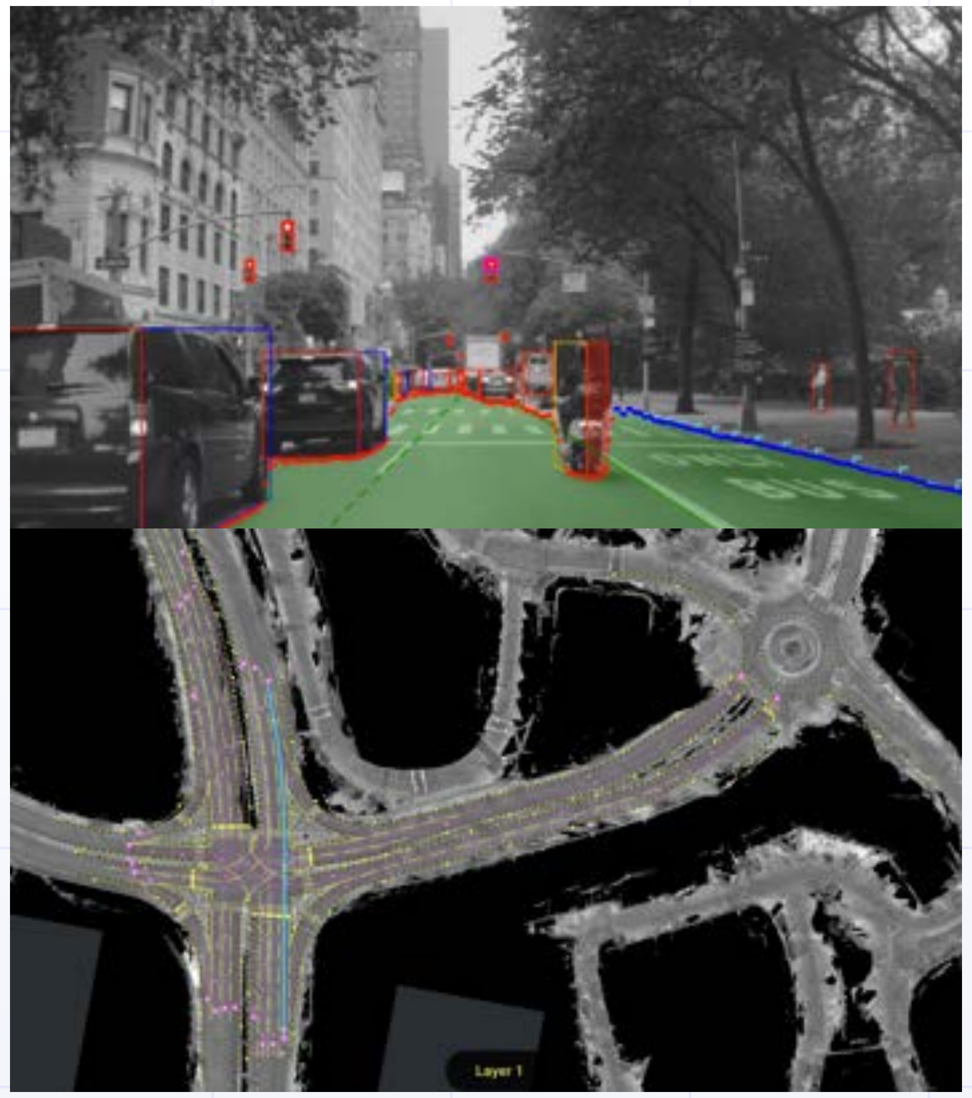
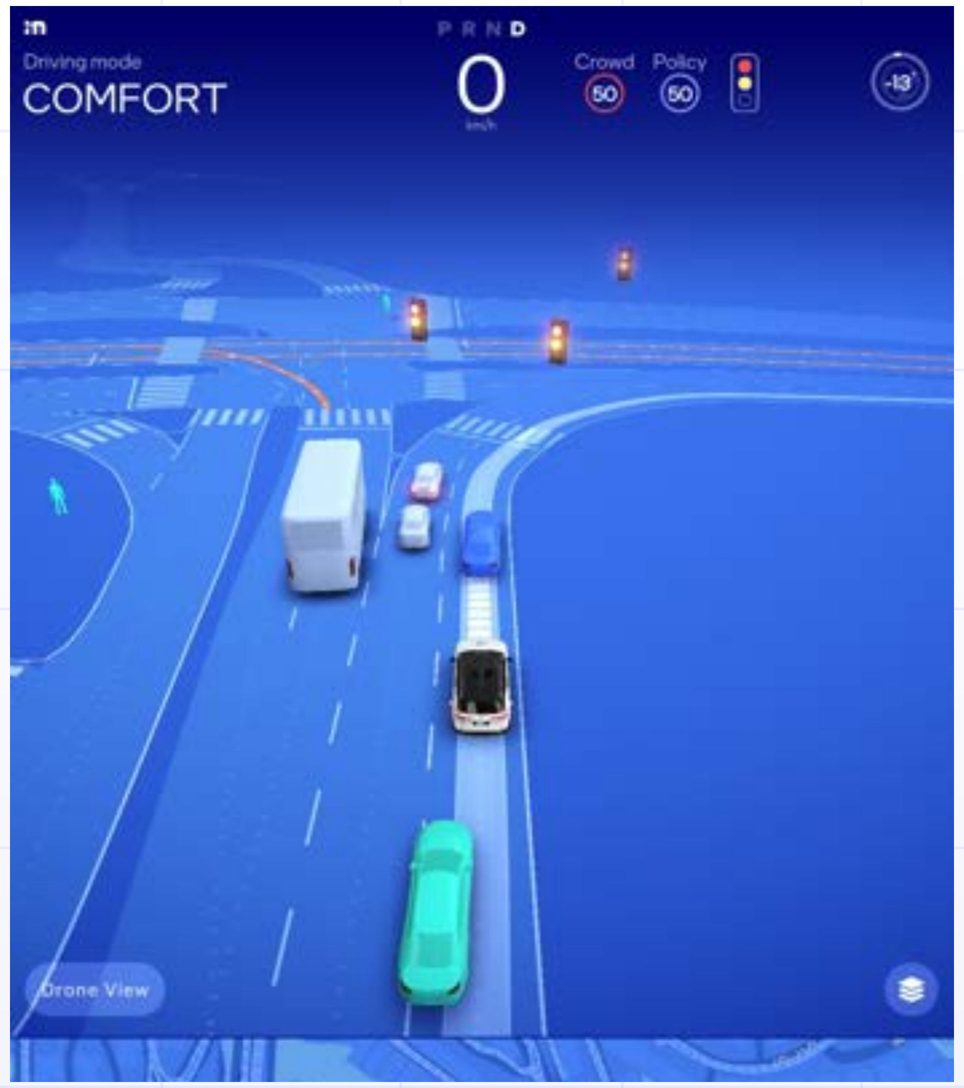
- The Sense-Plan-Act methodology
- The Differentiability-Scalability-Risk tradeoff
- The underestimation plague

Mobileye's Driving-Experience-Platform (DXP)

- The Universal vs. Unique separation
- The When-What-How abstraction
- DXP solves the Expressivity-Scalability-Risk tradeoff

The main ingredients of the platform's backbone

The Sense-Plan-Act Methodology

	Perception	Plan (Driving Policy)	Act (Control)
		Decision making “What would happen if” type of reasoning	Execute the plan
Sensing Mapping			

The Differentiability-Scalability-Risk Tradeoff

Differentiability

The user of the platform should be able to **differentiate** its product from other products

Scalability

The supplier's support resources must grow sub-linearly with the number of users

Risk

Using the platform should lead to a real product

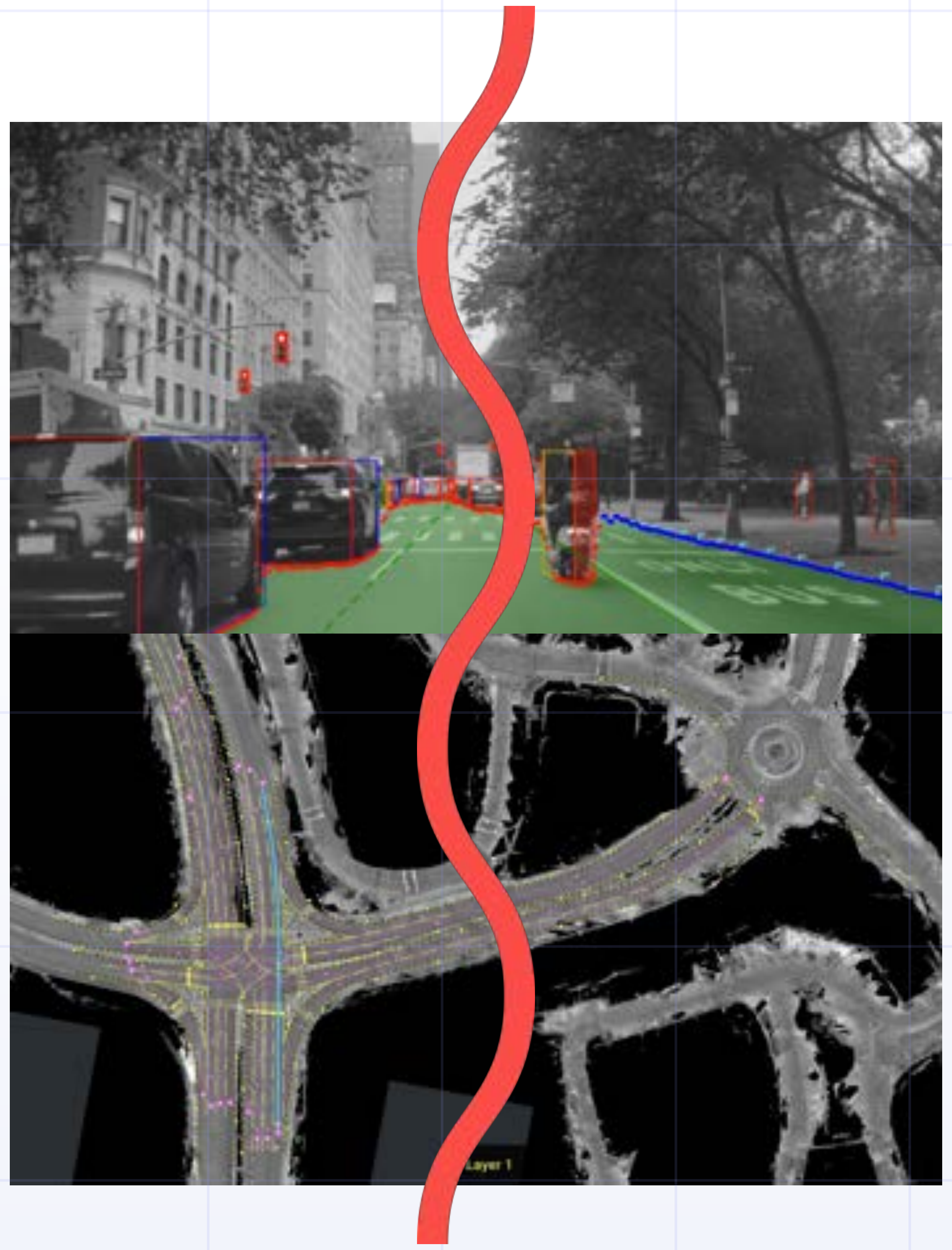
The Differentiability-Scalability-Risk Tradeoff

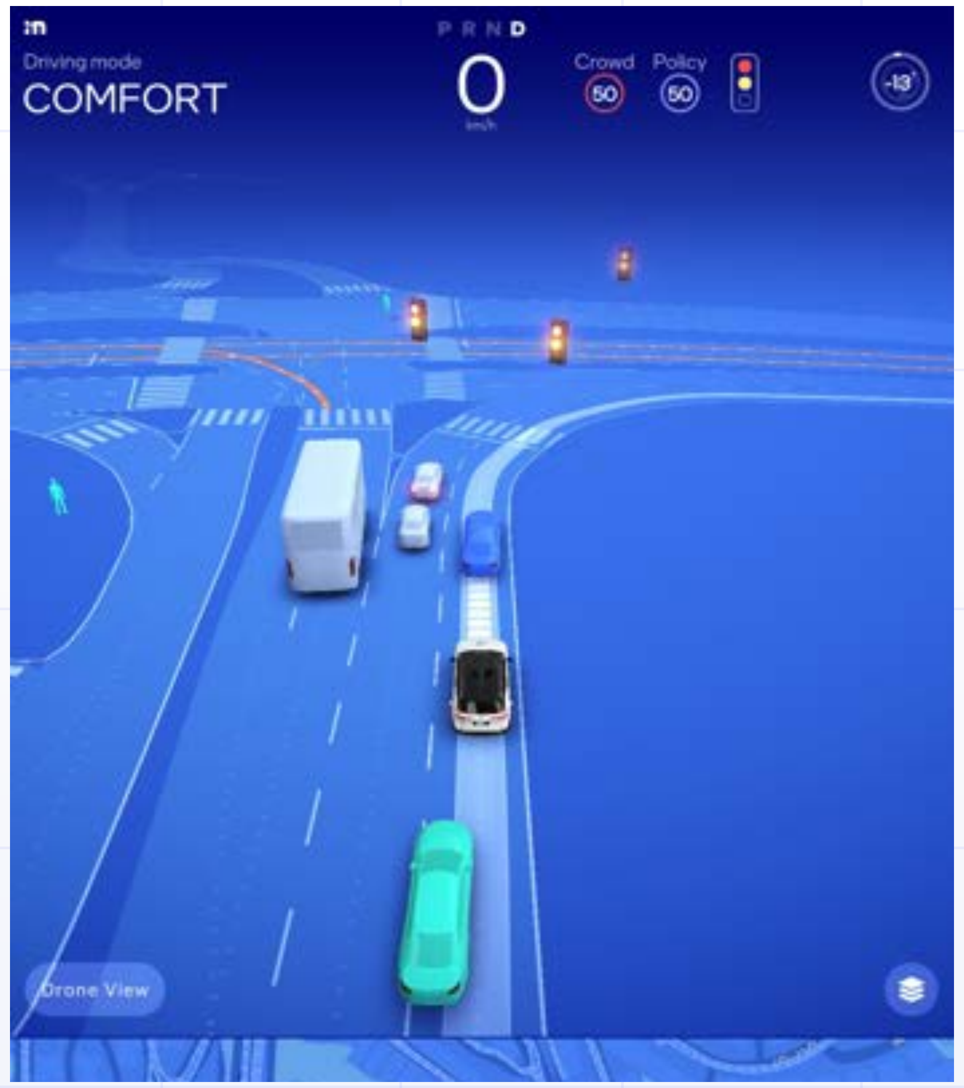
High risk

	Perception	Plan (Driving Policy)	Act (Control)
		Decision making	Execute the plan
		"What would happen if" type of reasoning	

Sensing

Mapping



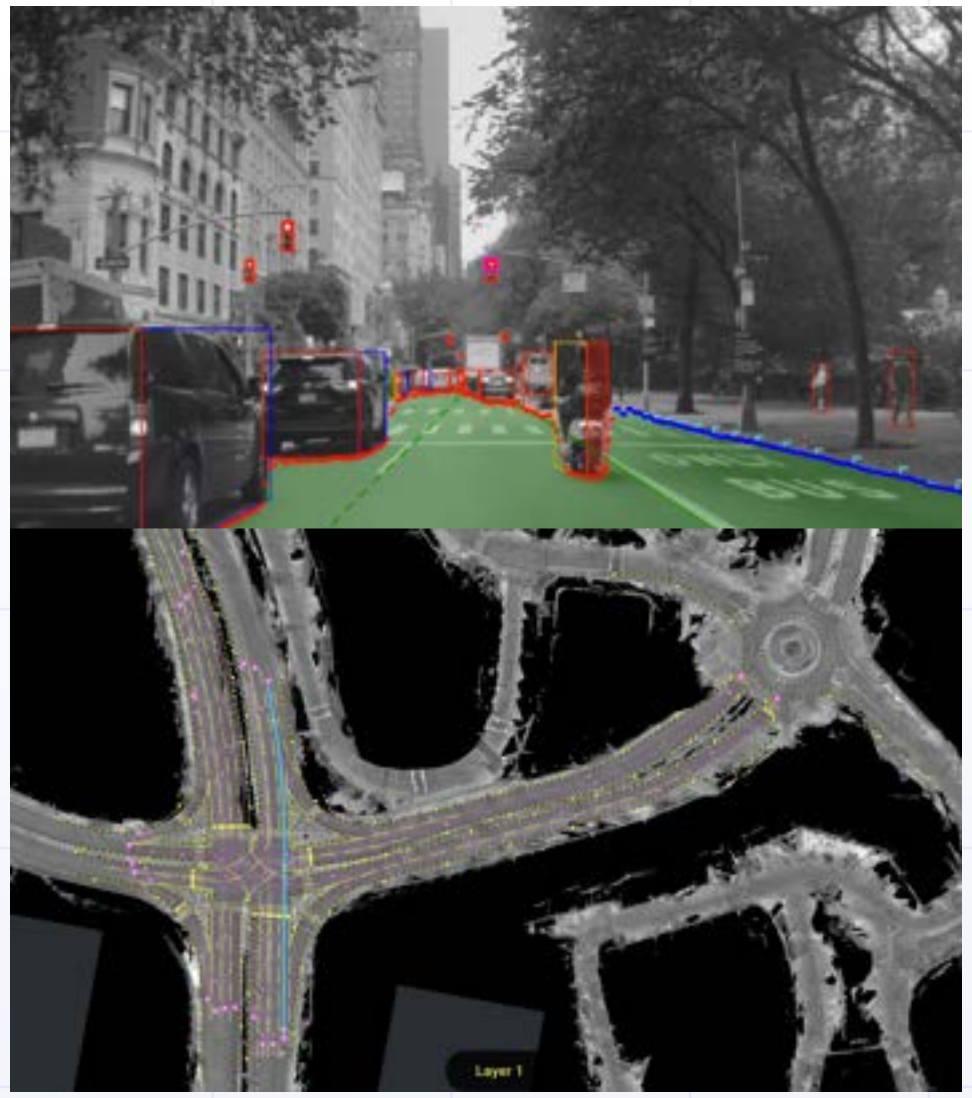
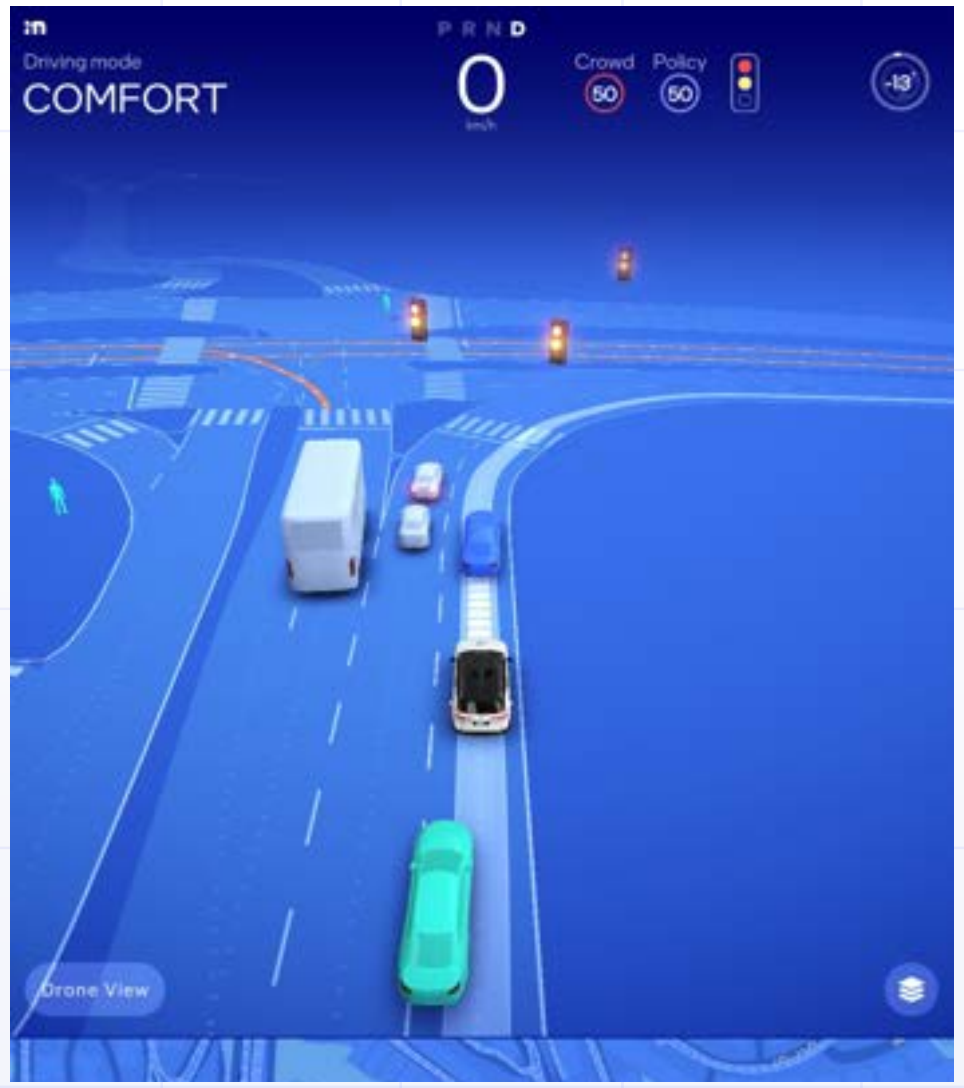


Platform

User

The Differentiability-Scalability-Risk Tradeoff

No differentiation or no scale

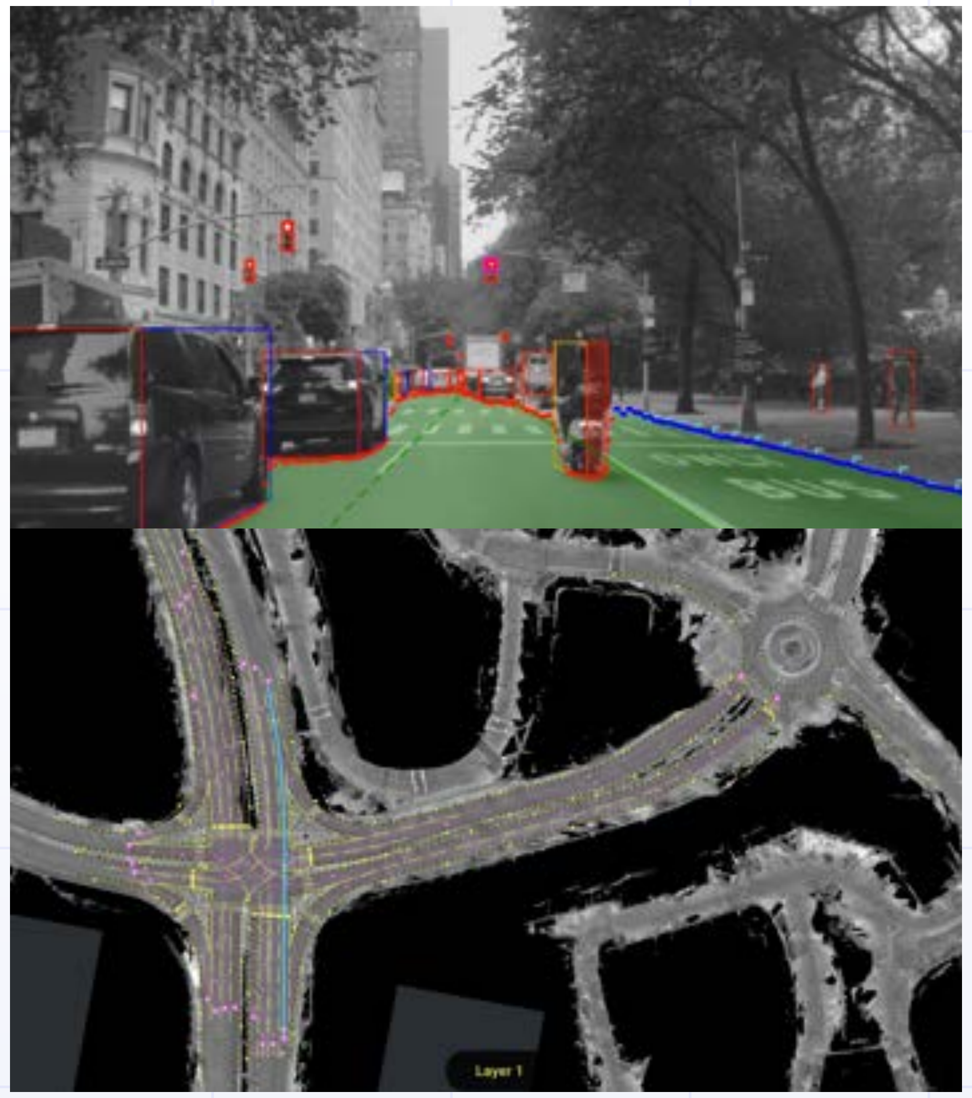
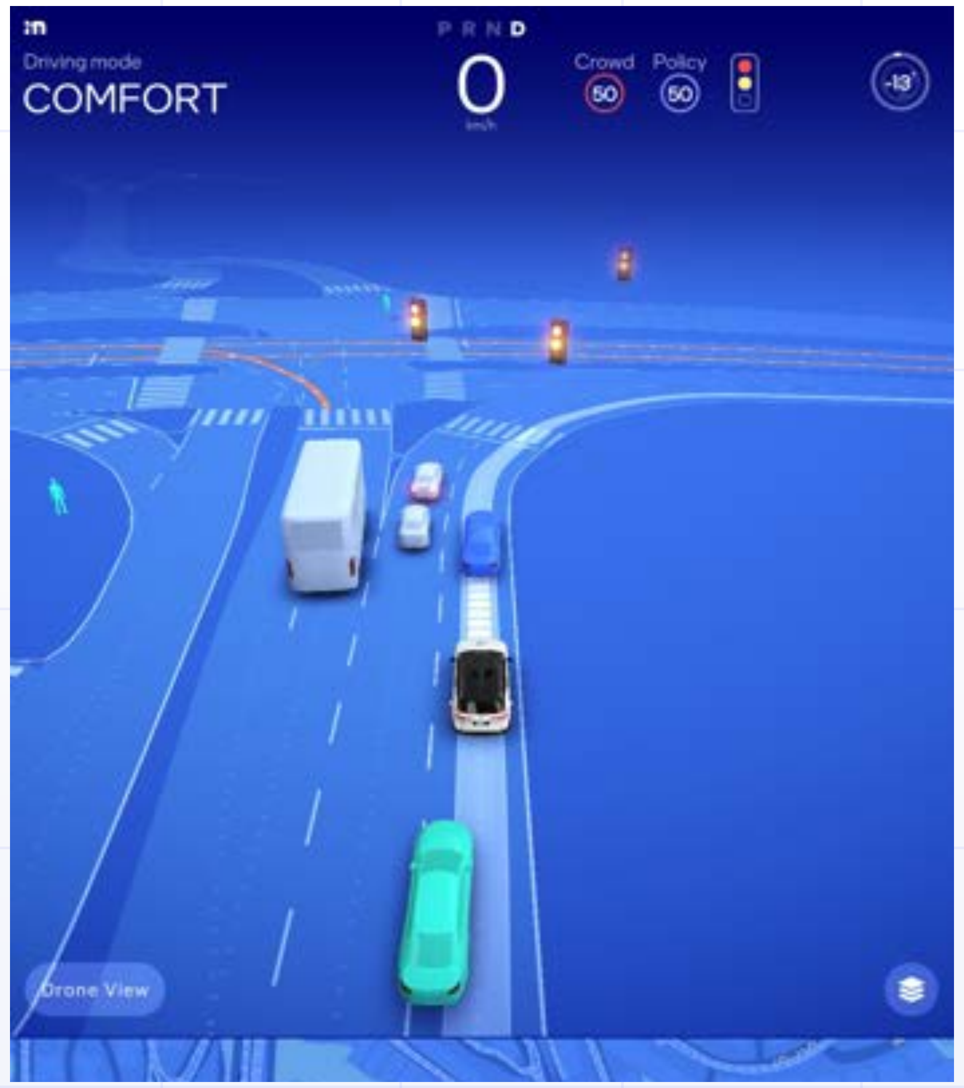
	Perception	Plan (Driving Policy)	Act (Control)
		Decision making "What would happen if" type of reasoning	Execute the plan
Sensing Mapping			

Platform

User

The Differentiability-Scalability-Risk Tradeoff

Good?

	Perception	Plan (Driving Policy)	Act (Control)
		Decision making “What would happen if” type of reasoning	Execute the plan
Sensing Mapping			

Platform

User

The Underestimation Plague

Back in 2016, headlines of "self-driving is around the corner"

Since then, most projects started optimistically and ended-up poorly

Self-driving is hard!

Self-driving main challenge is the combination of:

The complexity of advanced AI systems

Extremely high precision

Deep Learning to The Rescue?

Challenges of Mass Market Autonomy

Safety

- With millions of cars on the road, even a great, super-human system will cause several accidents every week
- There is no way to guarantee absolute safety. So, what are the KPIs for a safe system?

Usefulness

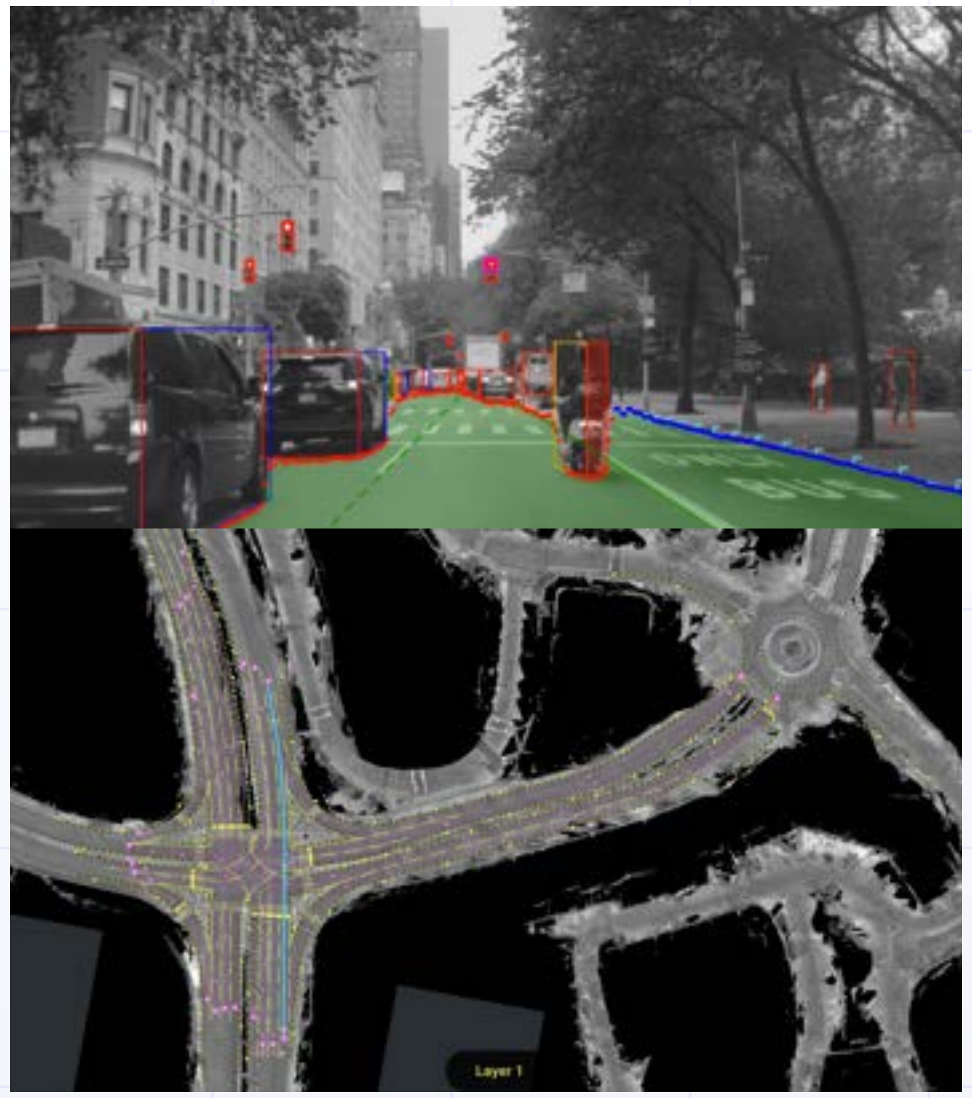
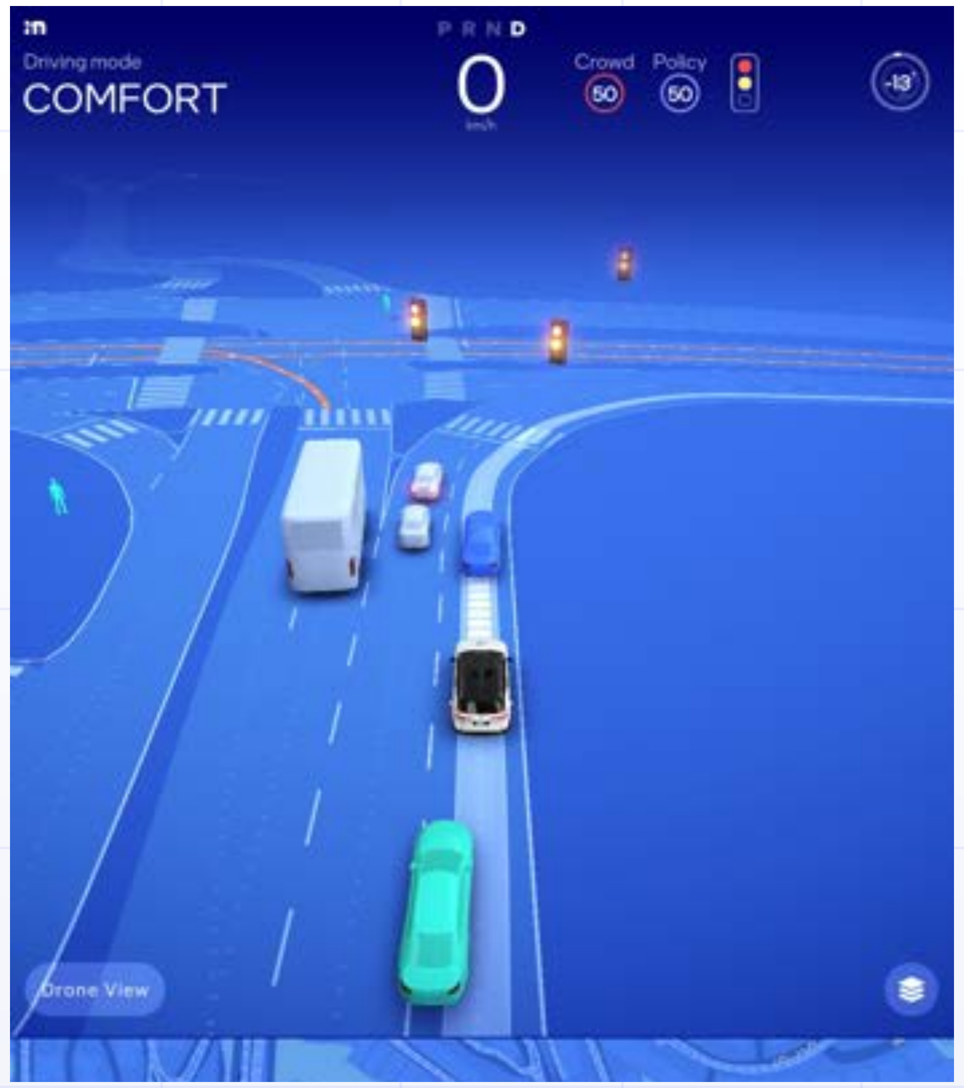
- Availability, Scalability, and Affordability

Modern Deep Learning Systems (GNNs, Transformers, BevFormers, etc.)

- Still make unintuitive errors
- Bad at edge cases
- Struggle with planning
- Reaching accuracy of 99.999999% with a statistical approach is unprecedented …

The Differentiability-Scalability-Risk Tradeoff

Good?

	Perception	Plan (Driving Policy)	Act (Control)
		Decision making "What would happen if" type of reasoning	Execute the plan
Sensing Mapping			

Platform

User

Boundary at Perception?

HIGH RISK

- Must deal with predictions, intentions, uncertainties, risks of decision-making errors, efficiency of planning
- Driving policy is also hard!

NOT SCALABLE

- Perception is never perfect, so driving policy must be intimately integrated with perception
- If perception is changed (even improved), driving policy must be adapted and re-validated

Outline

What is a development platform, and why should you care?

Why previous platforms for self-driving have not been successful?

- The Sense-Plan-Act methodology
- The Differentiability-Scalability-Risk tradeoff
- The underestimation plague

Mobileye's Driving-Experience-Platform (DXP)

- The Universal vs. Unique separation
- The When-What-How abstraction
- DXP solves the Expressivity-Scalability-Risk tradeoff

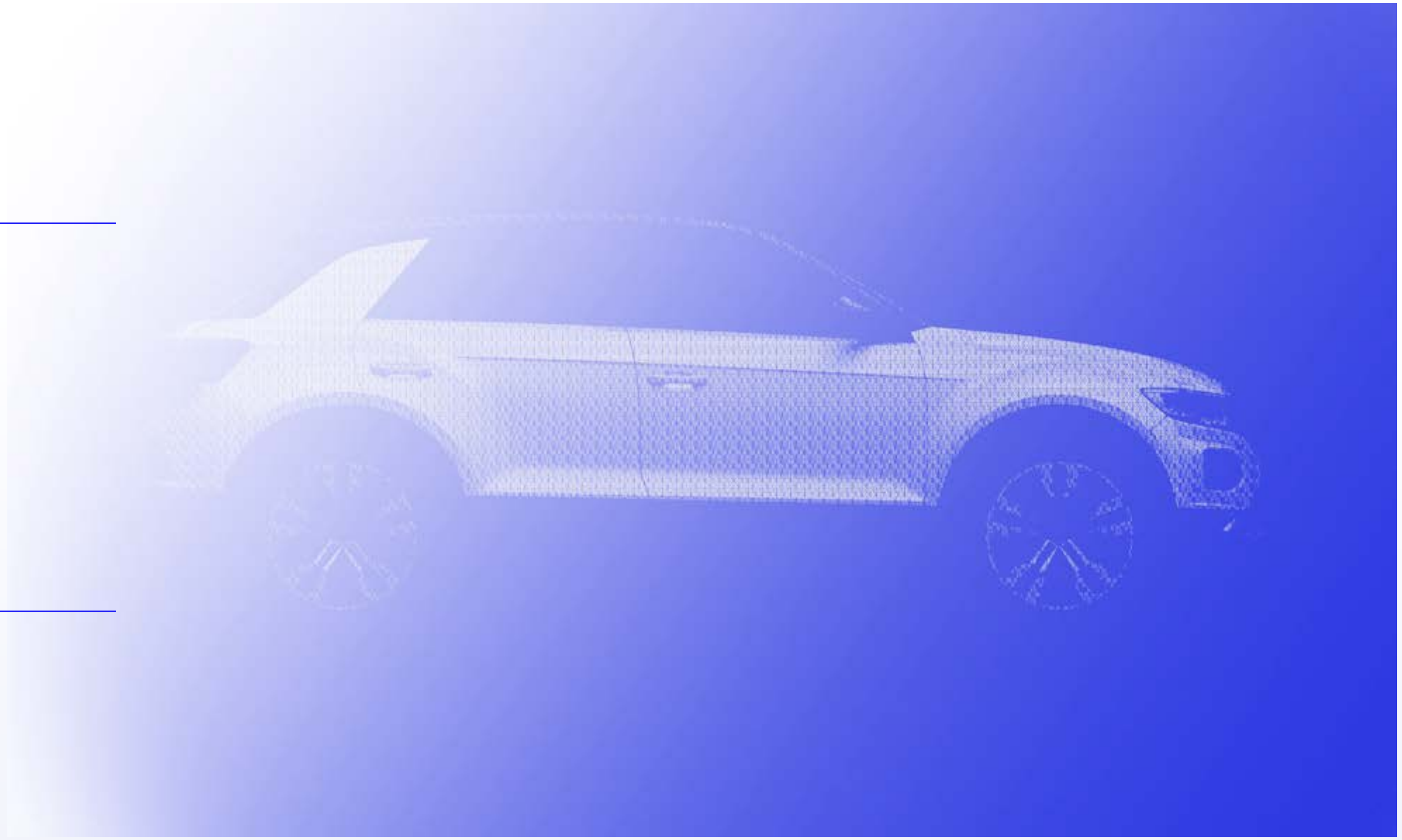
The main ingredients of the platform's backbone

How to Design a Good Self-Driving Platform?

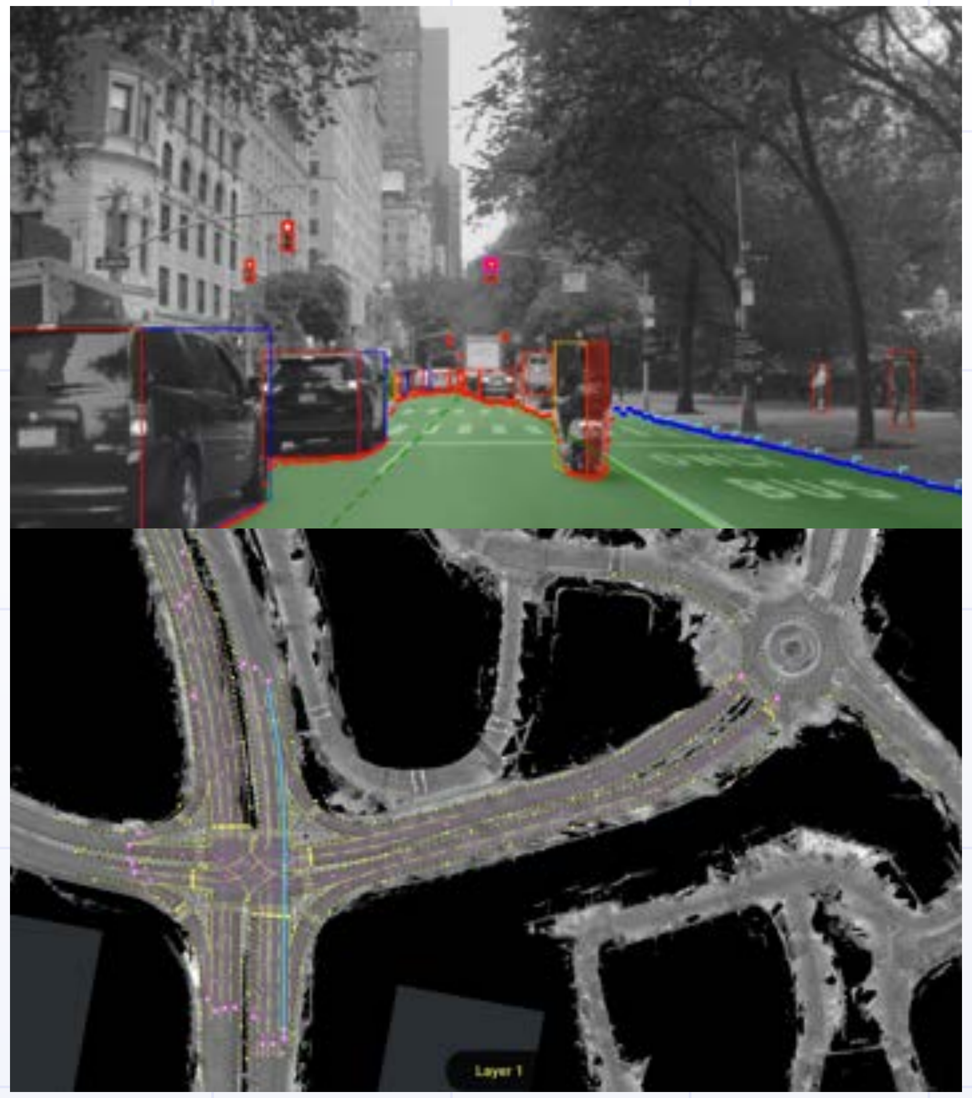
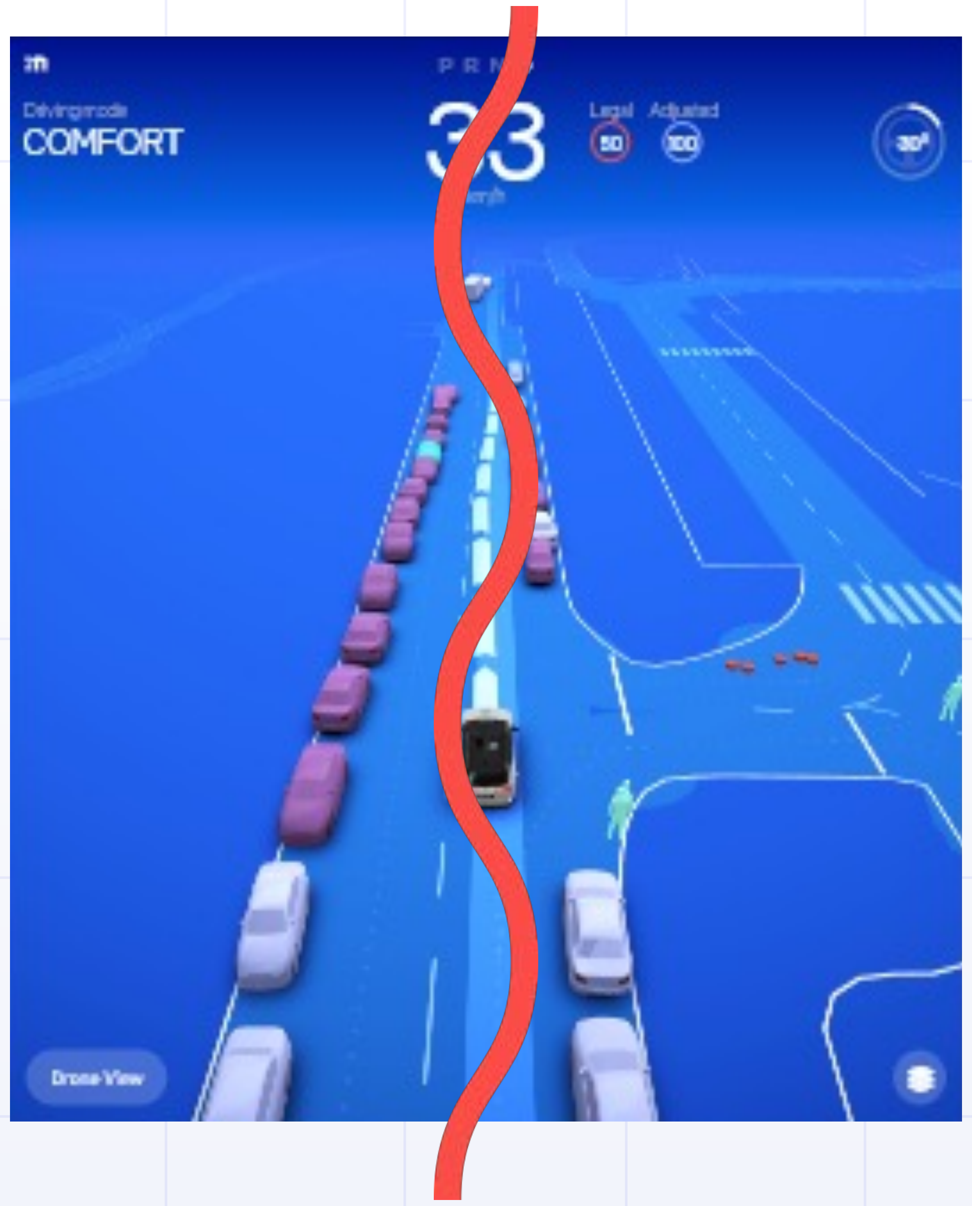
How to enable **Differentiation** while minimizing risk and enabling scalability?

Design methodology: **hide universal content**, because it is shared among all platform users, and focus on **unique content**

Main art: find the right granularity of abstractions



The Universal vs. Unique Separation

	Perception	Plan (Driving Policy)	Act (Control)
		Decision making "What would happen if" type of reasoning	Execute the plan
Sensing Mapping			

The Universal vs. Unique Separation

Universal

Facts

Kinematic states of other road users, hazards, traffic lights and stop lines, lanes and their semantic, routing, intersections and priority, traffic rules, etc.

Uncertainties

Lack of visibility, occlusions, error bars, etc.

Semi-facts (predicting the future)

Intentions (parking/stuck, cut-in, cut-out, reverse into a parking spot, U-turn, etc.)

Optimization

Efficient data structures (e.g. "find all lanes at distance d from a query point")

Optimization engines (e.g. "given desired offset per each road user, and lateral limiters, optimize a trajectory")

Unique

Discrete driving decisions

Lane changes

Overtake or stay behind

Yield or take way

Negotiation

Continuous longitudinal planning

Acceleration and braking profiles

Acceleration and jerk limiters

Margins (keeping distance, headway, etc.)

Lateral planning

Lateral acceleration and velocity

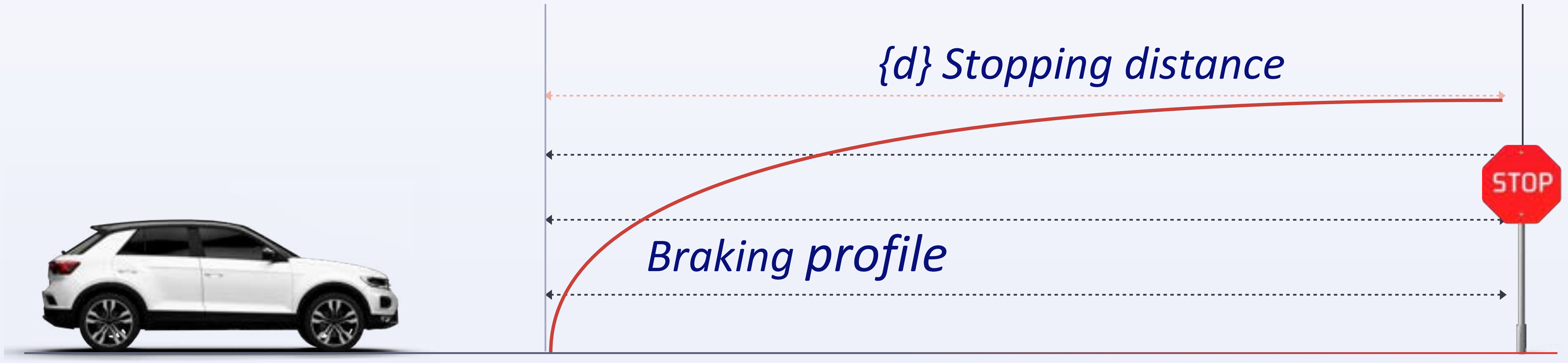
Offset parameters per road user

Control

HMI

The Driving-Experience-Platform (DXP) Language

Universal		Unique
When →	What →	How
“Approaching a stop sign”	“Brake to stop”	“Braking profile”
“Approaching a roundabout”	“Yield or Takeway”	“Yield logic”



Universal: The “When” And “What” Abstractions

When		What
Approaching a stop sign	→	Brake to stop
Red light	→	Brake to stop
Blocked junction	→	Brake to stop
...		
Approaching a curve	→	Comfortly reach speed
Approaching speed bump	→	Comfortly reach speed
...		
Approaching roundabout	→	Yield/takeway decision
Changing lanes	→	Yield/takeway decision
Cut-in vehicle	→	Yield/takeway decision
...		...

Unique: The “How” Abstraction

What	Platform families of How	User-specific How
Brake to stop	Impl_type_0(parameters)	Instance_0
	Impl_type_1(parameters)	Instance_1
	...	...

Working with DXP

01

User constructs packages of “how instances” out of the platform’s “how families”

Platform provides offline and online tools for creating these packages (simulator, online injection, recording)

Platform provides reference design to all required packages, so the user can focus only on packages in which he wants to differentiate

02

User creates code that selects packages based on application parameters such as locality, road types, regulation, driving modes, weather conditions, etc.

Platform provides reference design

DXP Solves The Differentiability-Scalability-Risk Tradeoff

Differentiability

The user of the platform controls the unique content, hence can differentiate.

Scalability

The right abstraction separates universal from unique in a way that prevents the need of intimate integration

Reference design and supplementary development tools allow sub-linear growth

Risk

Platform is based on a real working product

User gets a reference design, hence has a working solution from day one and can focus efforts on differentiation

Example Code

```cpp
// At init time:
// Define the brake to stop specific implementation chosen by OEM
std::vector<std::function<float(float, float, float)>> BrakeToStopFns;
// And initialize specific implementation well tuned by OEM
for (const auto& p : _ego.config().BrakeToStop.BrakeToStopAccFormulas) {
    BrakeToStopFns.emplace_back(std::bind(policy::brake_to_stop_acc_formula,
        std::placeholders::_1, std::placeholders::_2, std::placeholders::_3, _ego.DT(), p));
}
for (const auto& p : _ego.config().BrakeToStop.BrakeToStopAccJerkFormulas) {
    BrakeToStopFns.emplace_back(std::bind(policy::brake_to_stop_jerk_formula,
        std::placeholders::_1, std::placeholders::_2, std::placeholders::_3, _ego.DT(), p));
}
for (const auto& p : _ego.config().BrakeToStop.BrakeToStopJerkOptimizations) {
    BrakeToStopFns.emplace_back(std::bind(PolicyAlgoUtils::acc_for_brake_on_red,
        std::placeholders::_1, std::placeholders::_2, std::placeholders::_3, _ego.DT(), p));
}
```

Example Code

```
// Definition of scenarios
struct BrakeToStop {
    unsigned short tfl_red = 0;
    unsigned short tfl_right_on_red = 0;
    unsigned short tfl_yield_at_blinking_red = 0;
    unsigned short tfl_yield_at_green = 0;
    unsigned short tfl_dont_block = 0;
    unsigned short stop_sign = 0;
    unsigned short end_of_path_road_edge = 0;
    unsigned short distance_till_must_perform_lc = 0;
    unsigned short bottleneck_with_oncoming = 0;
    unsigned short stop_line_2nd = 0;
};
```

Example Code

```cpp
// At every iteration:
void fill_brake_to_stop(BrakeToStop& scenariosForBrakeToStop, ...)
{
    // example condition
    if ((country_code == "Germany" && road_type == "Highway")
        || weather_condition == "Rain"
        || driving_mode == "comfort")
    {
        scenariosForBrakeToStop.distance_till_must_perform_lc = ...
    }

    // additional code
}
```

Combination A

Combination B

Outline

What is a development platform, and why should you care?

Why previous platforms for self-driving have not been successful?

- The Sense-Plan-Act methodology
- The Differentiability-Scalability-Risk tradeoff
- The underestimation plague

Mobileye's Driving-Experience-Platform (DXP)

- The Universal vs. Unique separation
- The When-What-How abstraction
- DXP solves the Expressivity-Scalability-Risk tradeoff

The main ingredients of the platform's backbone

How to Build a Capable Driving System?

Separate driving-policy (“plan”) from perception (“sense”)

Perception:

Modular design as opposed to moonshots

Redundancy!

Driving Policy:

The Responsibility-Sensitive-Safety (RSS) model

Intentions vs. Predictions

Example: Redundant Object Detection Systems

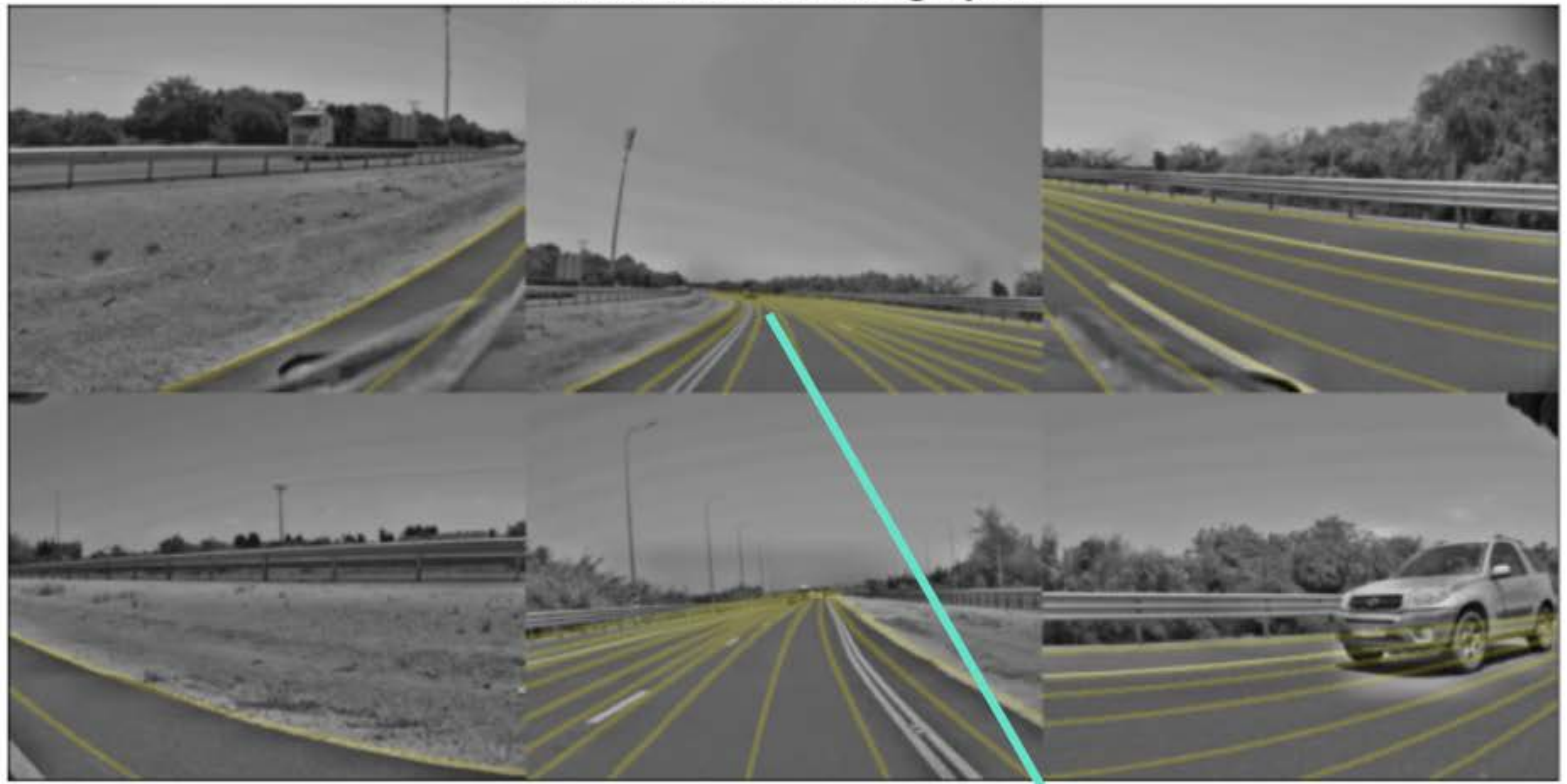
4 “axes” of redundancy

- Camera vs. Radar vs. Lidar
- Decomposable vs. End-to-end approaches
- Appearance vs. Geometric approaches
- Learning vs. Model-based approaches

Camera, Learning, Decomposable, Appearance-Based

Camera, Learning, End-to-End, Appearance-Based

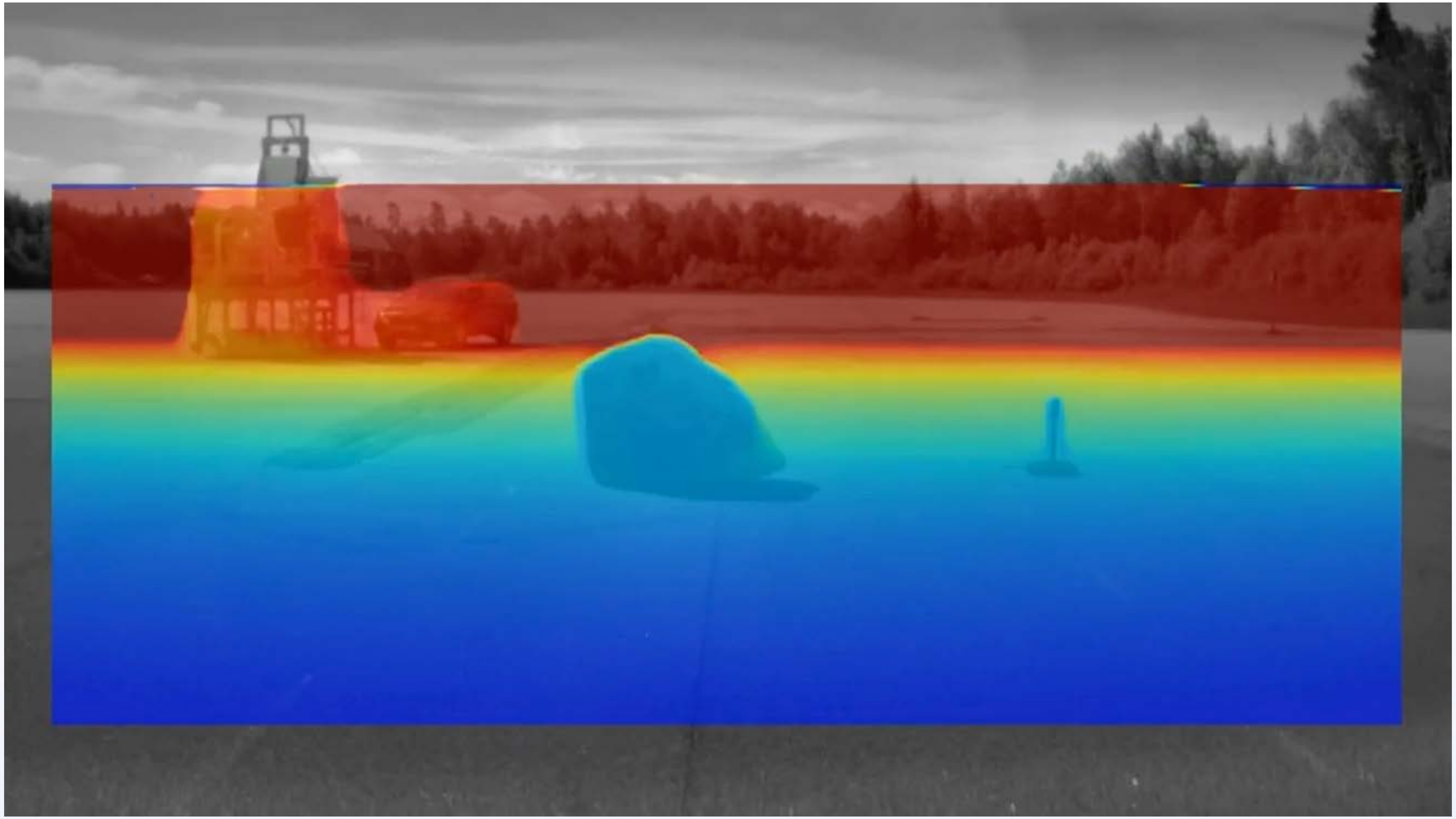
surround frames w/ ego path



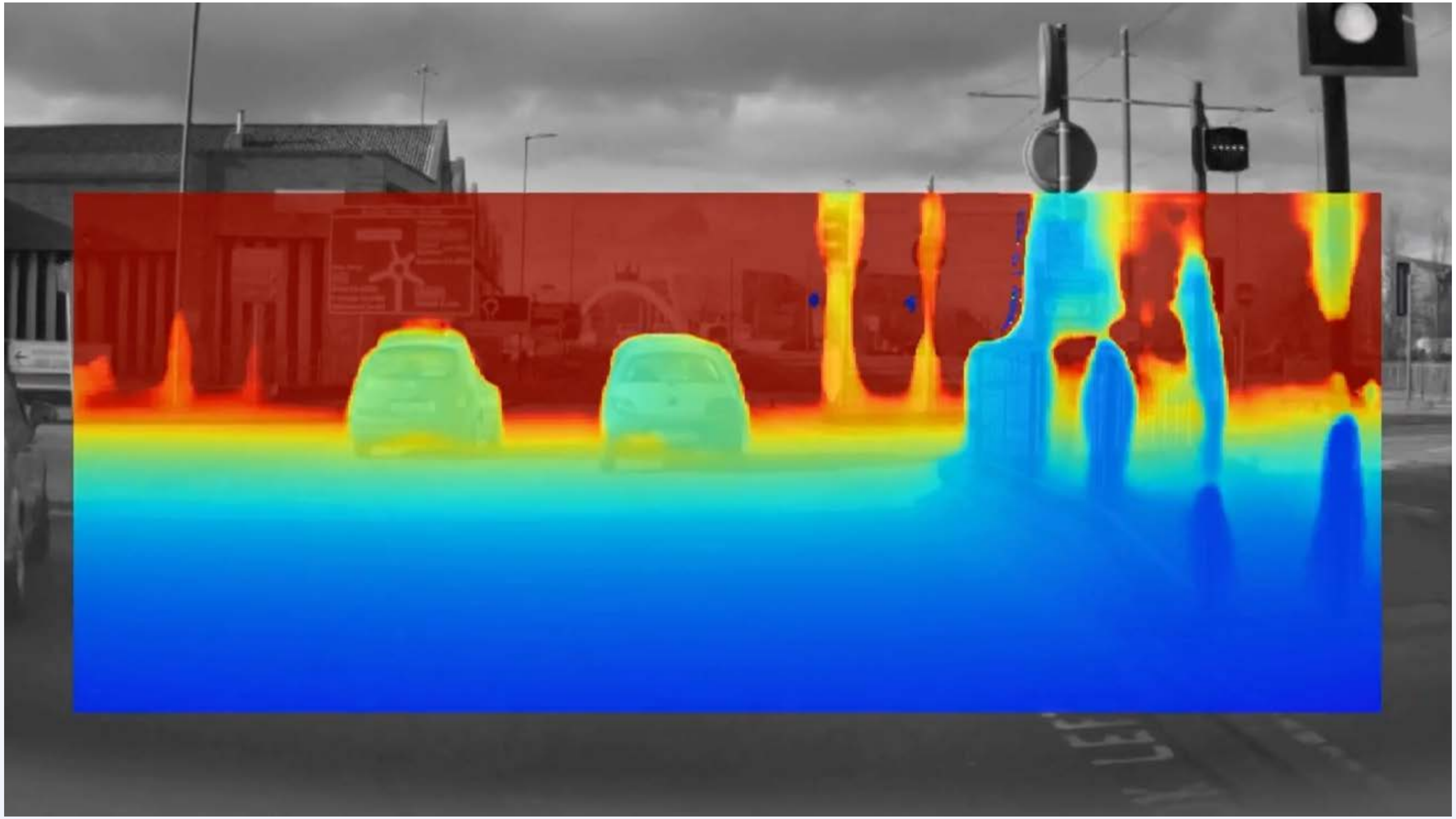
first 17 tokens: (out of 180)

idx: 0, 1, 2, 3, 4, 5, 6, 7, 8,

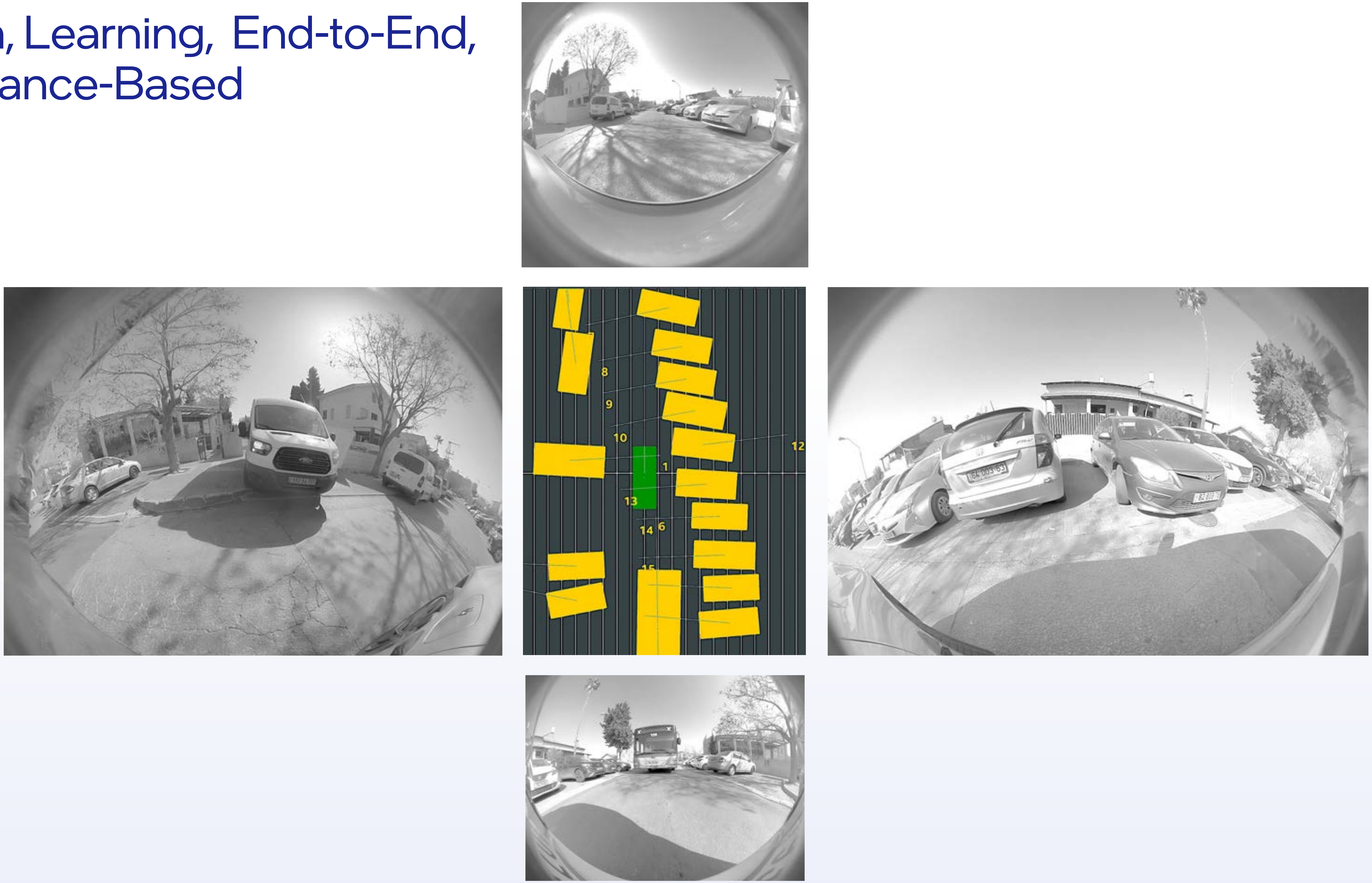
Camera, Model-Based, Decomposable, Geometry-Based



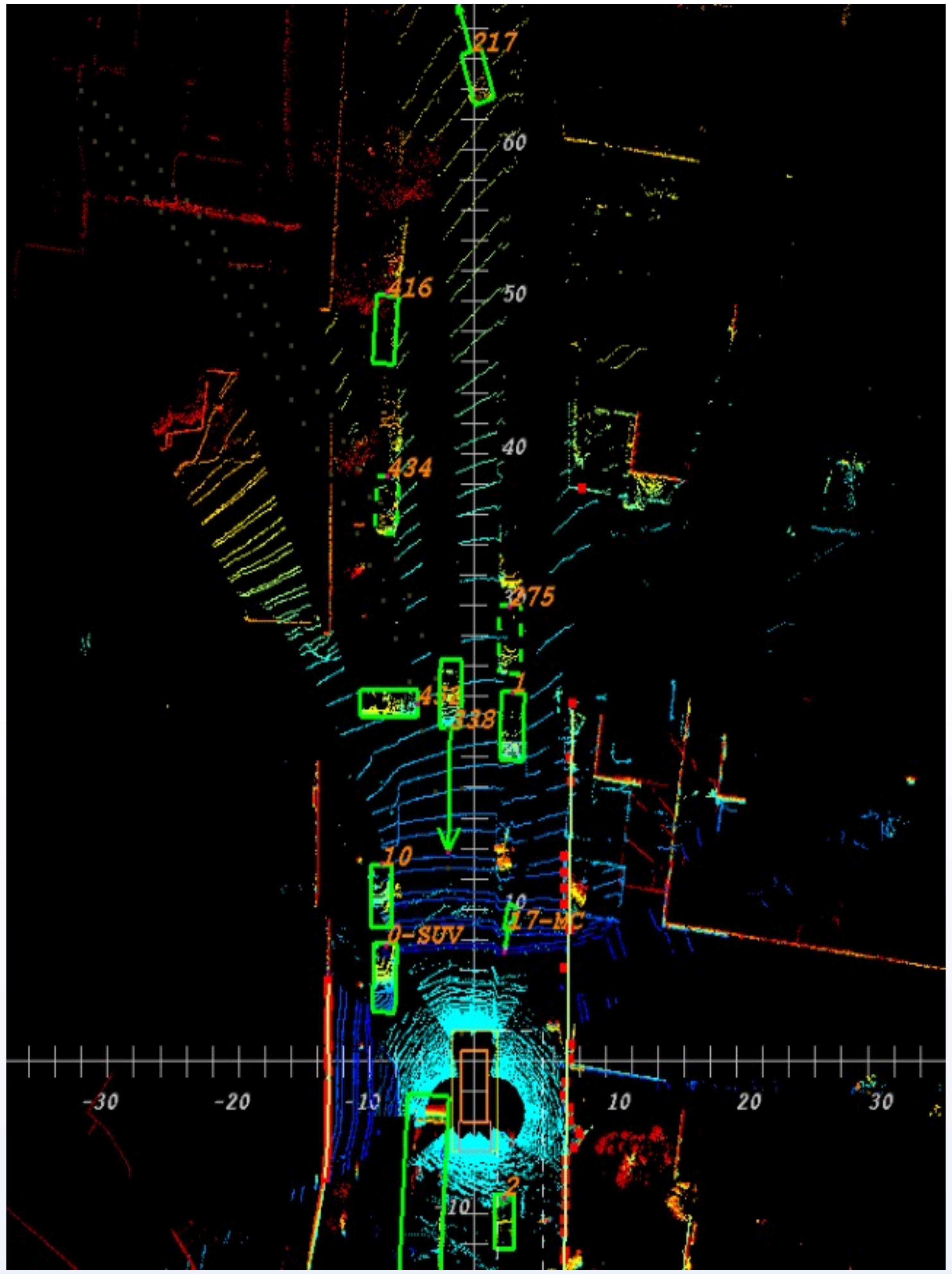
Camera, Learning-Based, Decomposable, Geometry-Based



Camera, Learning, End-to-End, Appearance-Based

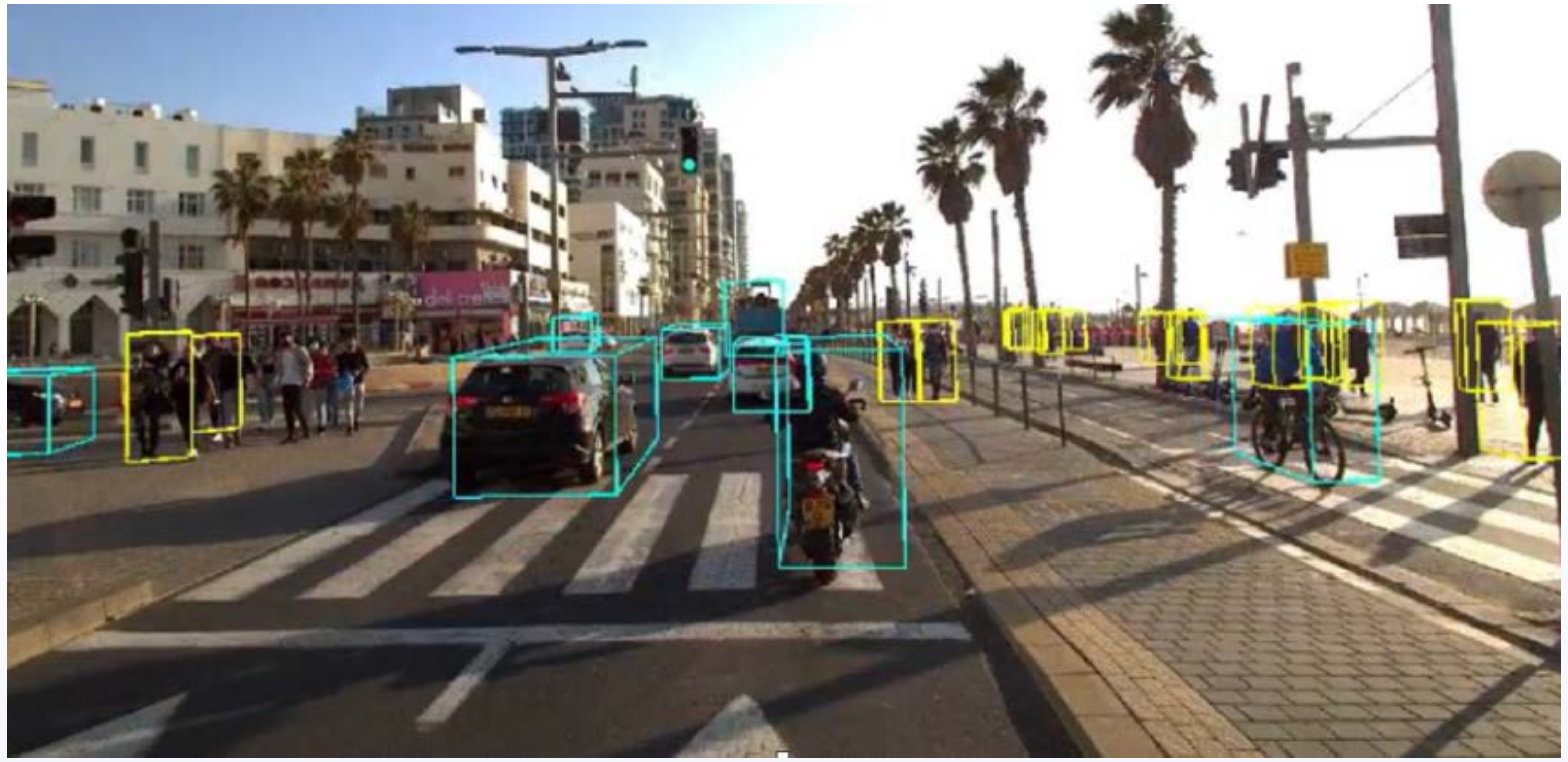


Lidar, Model, Decomposable, Geometry



Lidar, Learning, End-to-End, Geometry

Imaging Radar, Learning, End-to-End, Geometry



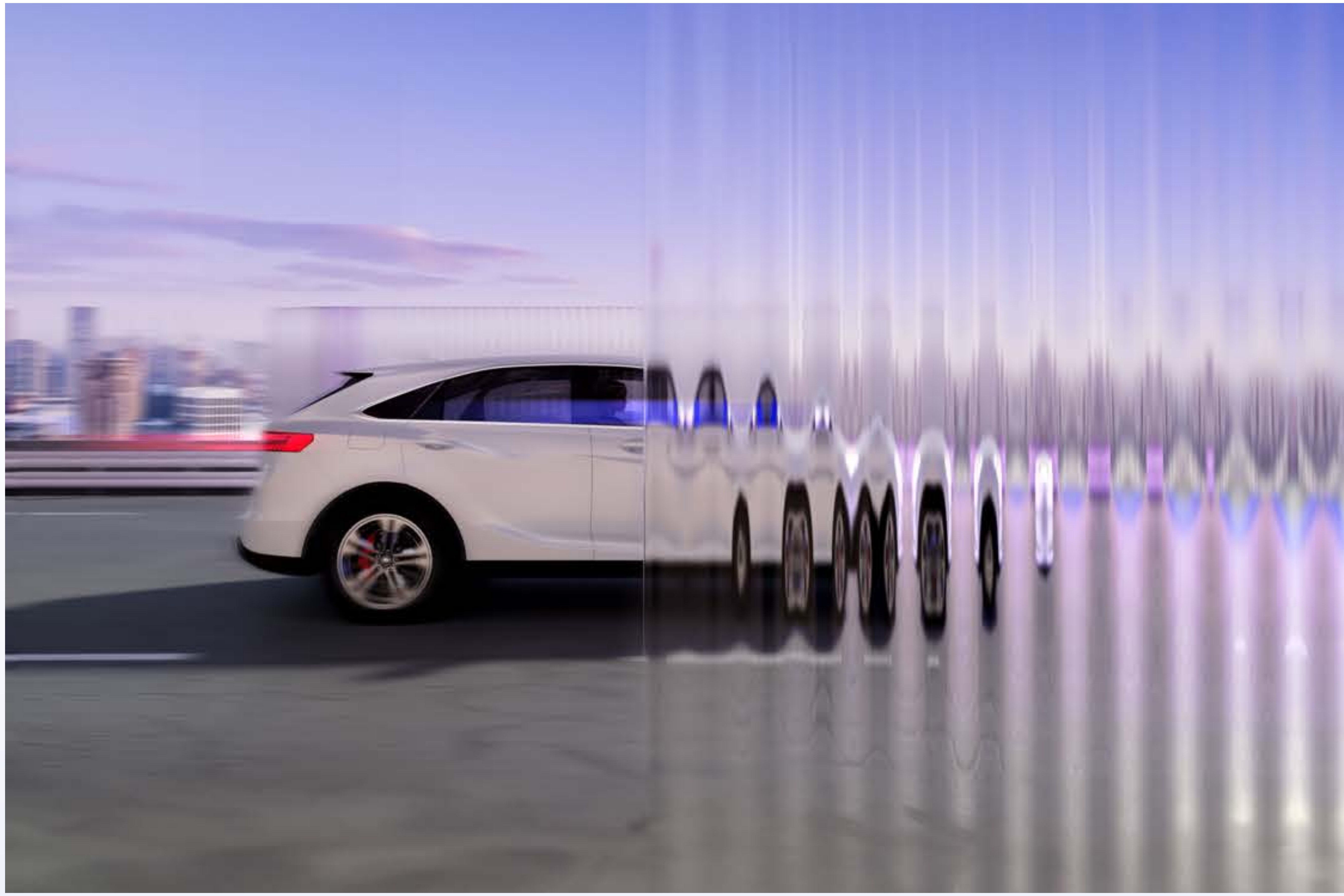
Why Driving Policy is Difficult?

Unlike the sensing part, there **is no "ground truth"**

Actions that are performed now may have **long term effect on the future**

Close loop: Actions of the ego vehicle affect other road users (e.g., when "pushing" in a lane change)

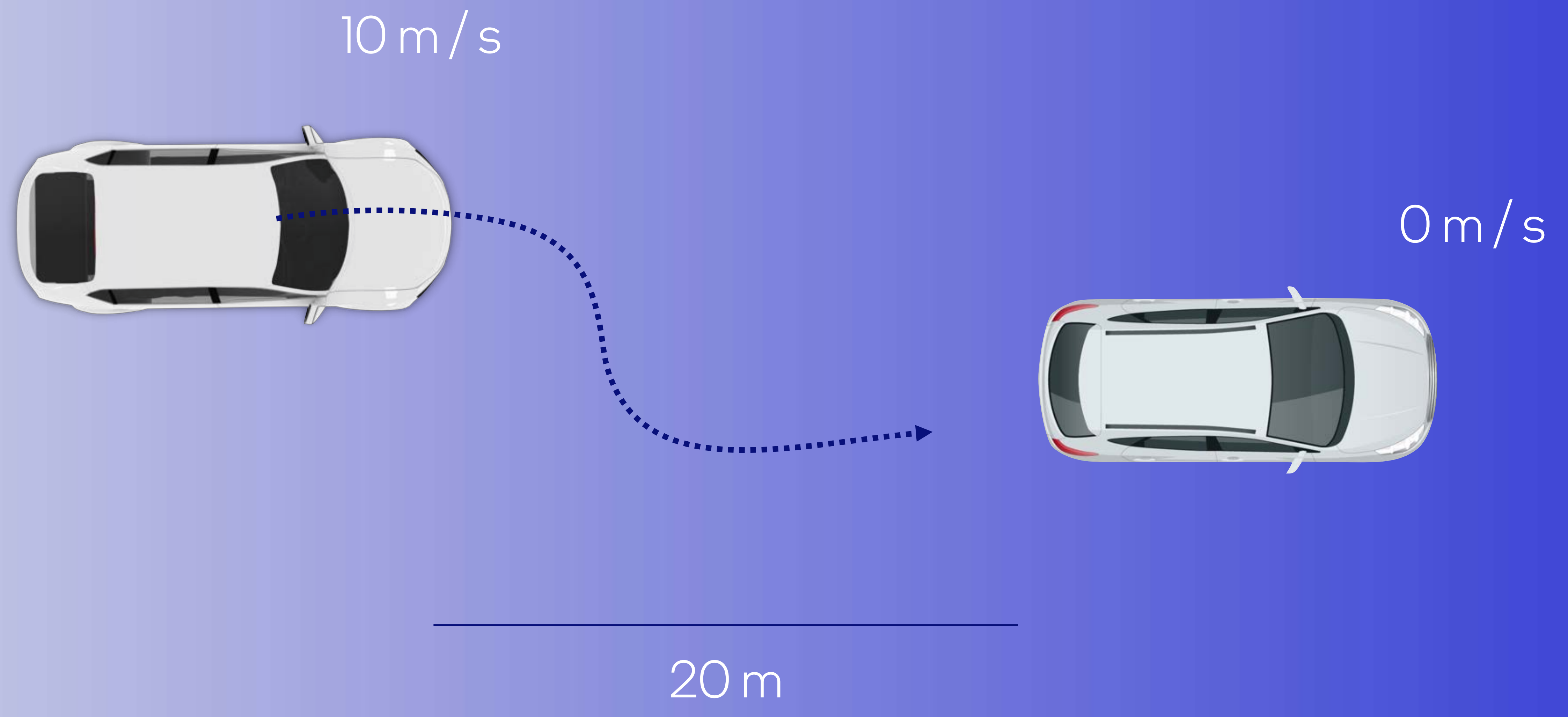
Must handle **uncertainties** about the future (what others might do)



Driving Policy | The Computational Challenge

Actions that are performed now may have long term effects on the future

Must plan for a sufficiently long time, because a bad plan might look perfectly fine at the near future



Mobileye's Solution: RSS + Analytical Calculations + Intentions

- Assume the worst-case under a well-defined set of reasonable assumptions
- Couple all the future into the present using analytical calculations
- Unlike Dynamic Programming methods, which requires predictions, for our method predictions are unnecessary, because we analytically couple all possible reasonable futures into the present

- Construct "intentions" of other agents (e.g. car is yielding or take right-of-way)
- Those "intentions" control parameters of the "reasonable assumptions"
- Using "intentions" rather than "predictions" yields a "human-like" behavior

- Using modern AI (deep learning and other methods) to construct intentions

Comparison To Other Approaches

	ME's approach	Monte-Carlo Tree Search (MCTS)	Dynamic Programming (DP) on MDP or LQR	End-to-End Learning
Transparency	Yes	Yes	Yes	No
Controllability	Yes	**Yes**	**Yes**	No
Performance	Guaranteed (math proof)	**Depends on #rollouts and agent model**	Requires predictions	Black-box, only statistical guarantees
Efficiency	Yes	**Requires many rollouts**	**Curse of dimensionality**	Yes

Summary

Mobileye's Driving-Experience-Platform (DXP)

- The Universal vs. Unique separation
- The When-What-How abstraction
- DXP solves the Expressivity-Scalability-Risk tradeoff

The main ingredients of the platform's backbone

- Redundancy is key for perception
- Driving Policy using RSS + analytical calculation + intentions

Thank you.

mobileye™