Explaining Errors in Autonomous Vehicles

A Diagnosis Tool and Testing Framework for Robust Decision Making

Leilani H. Gilpin, PhD
Dept. of CSE, UC Santa Cruz
JHU CaSE Graduate Seminar
November 11, 2021

Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

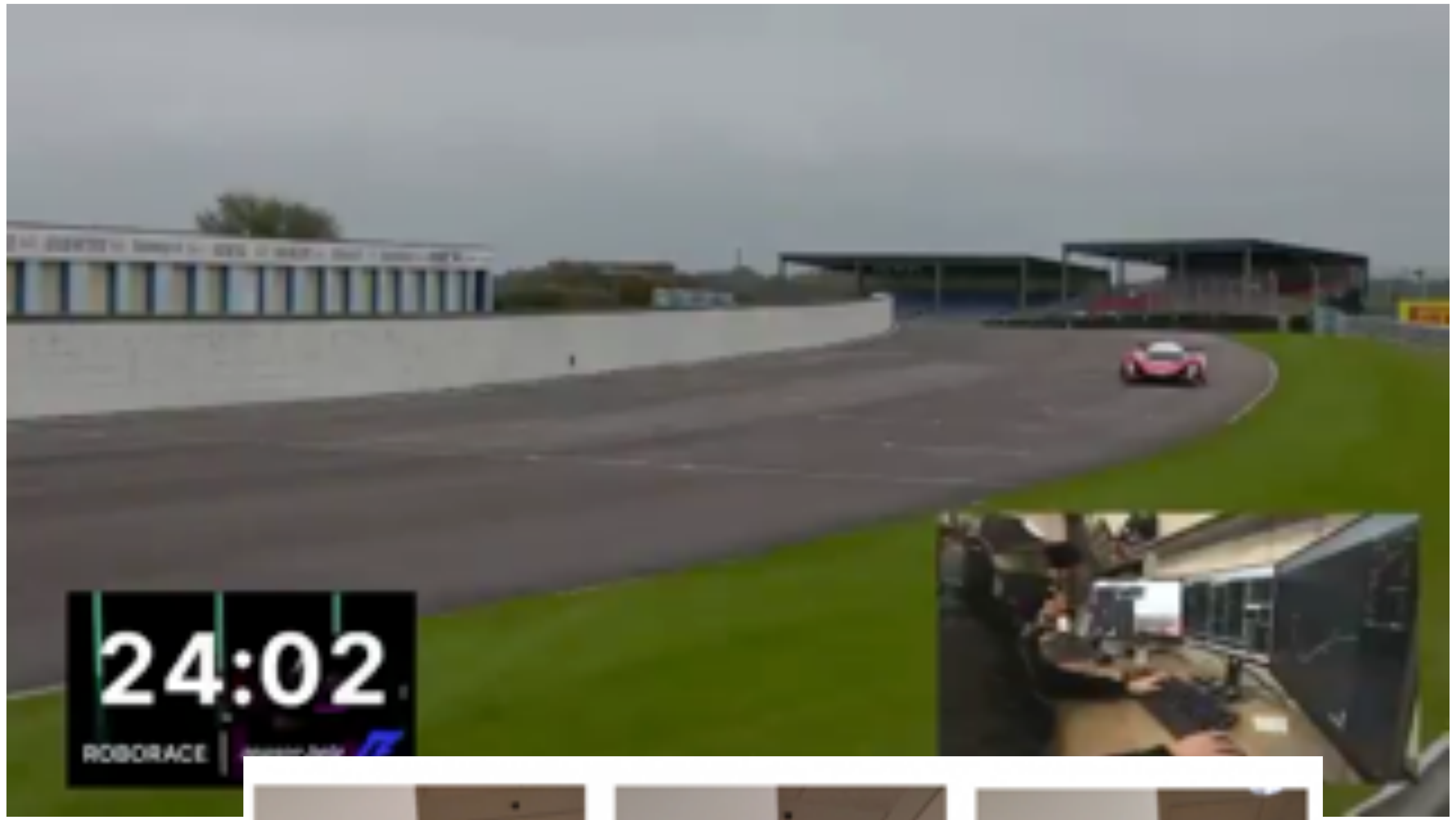
Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.

Question: How to develop self-explaining architectures that can help anticipate failures instead of after-the-fact?

Autonomous Vehicles are Prone to Failure



K. Eykholt et al. “Robust Physical-World Attacks on Deep Learning Visual Classification.”

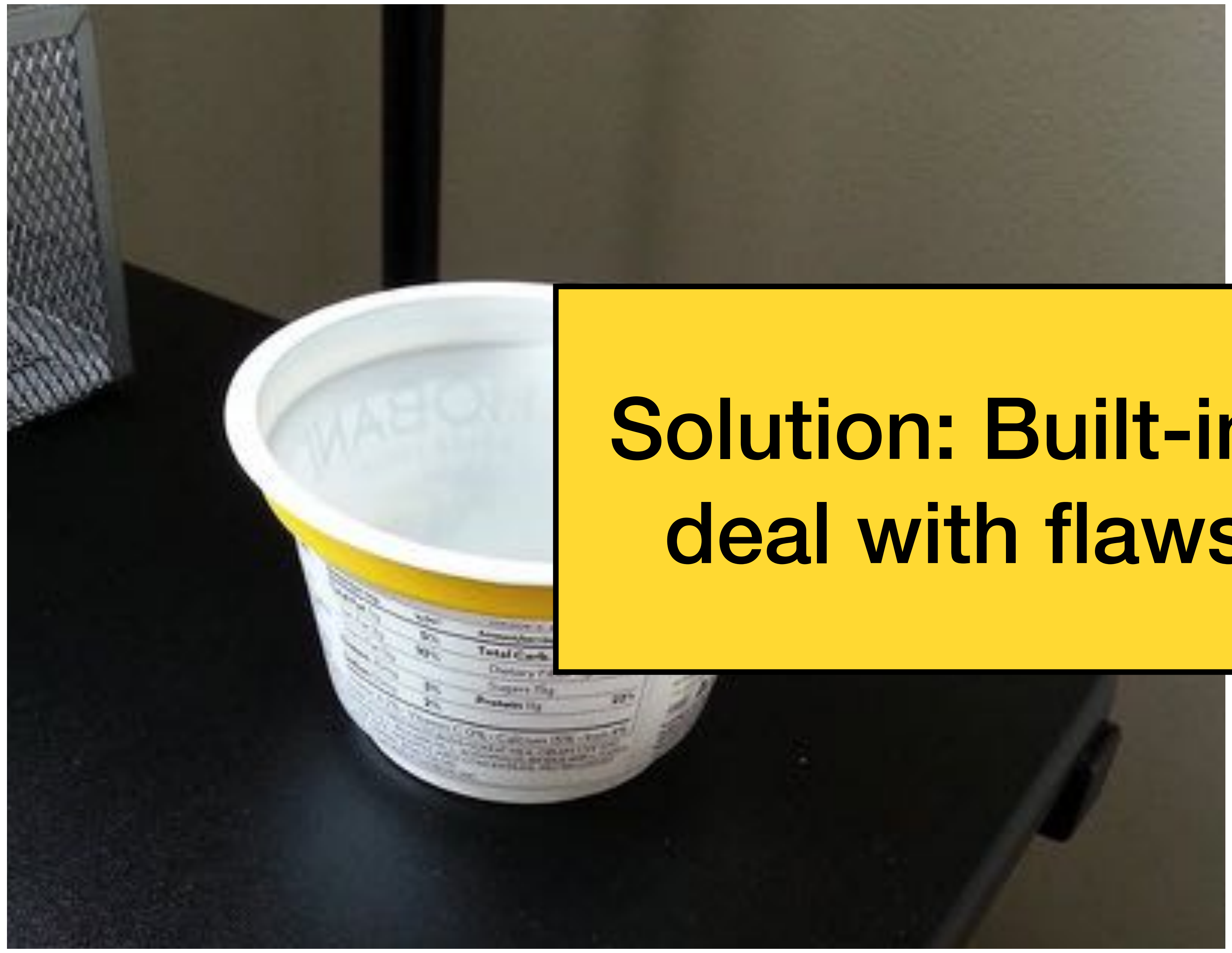
Predictive Inequity in Object Detection

Benjamin Wilson [1] Judy Hoffman [1] Jamie Morgenstern [1]

Autonomous Vehicle Solutions are at Two Extremes

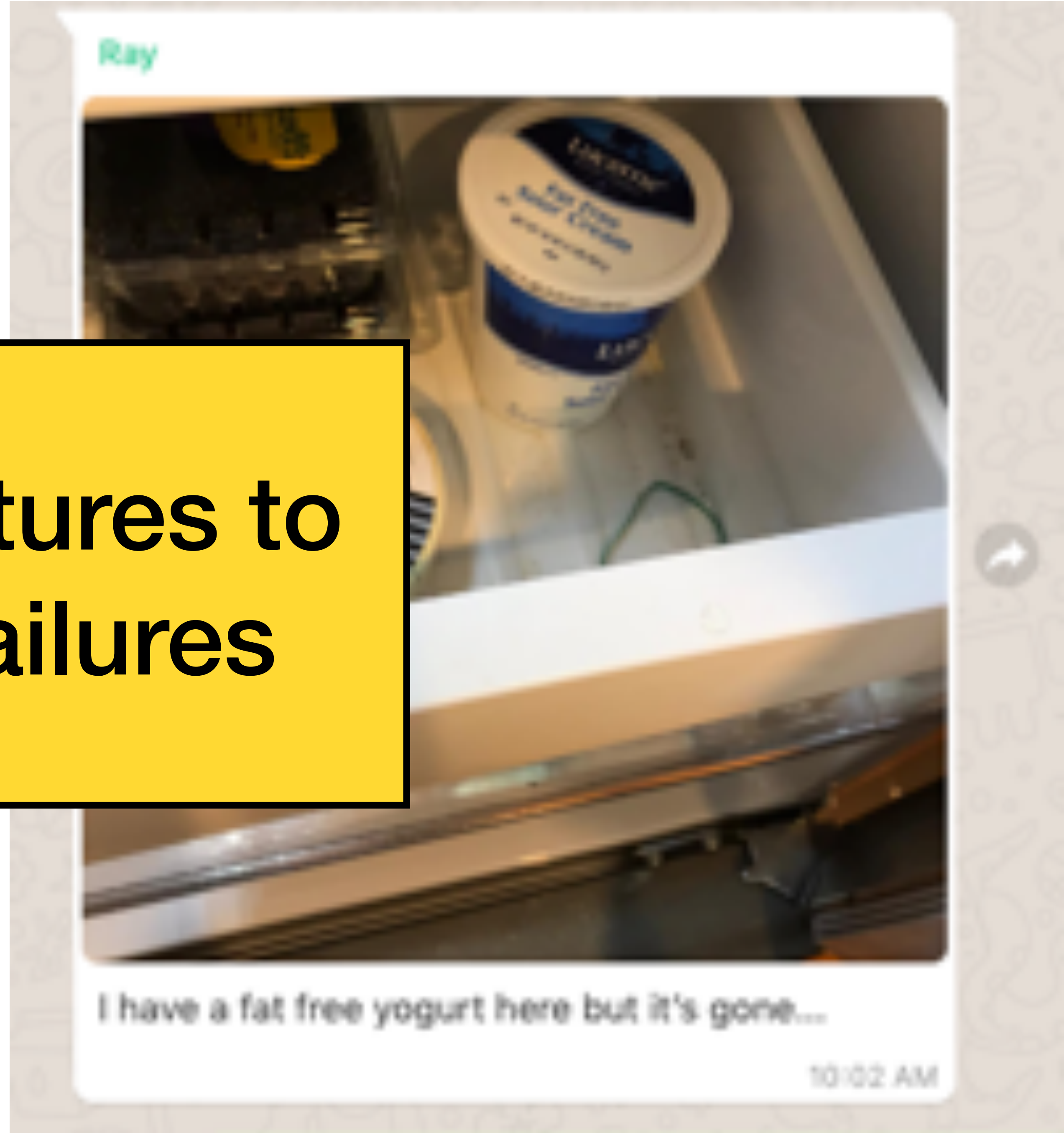
Complex Systems Include People

Misalignment of Expectations



Lack of communication

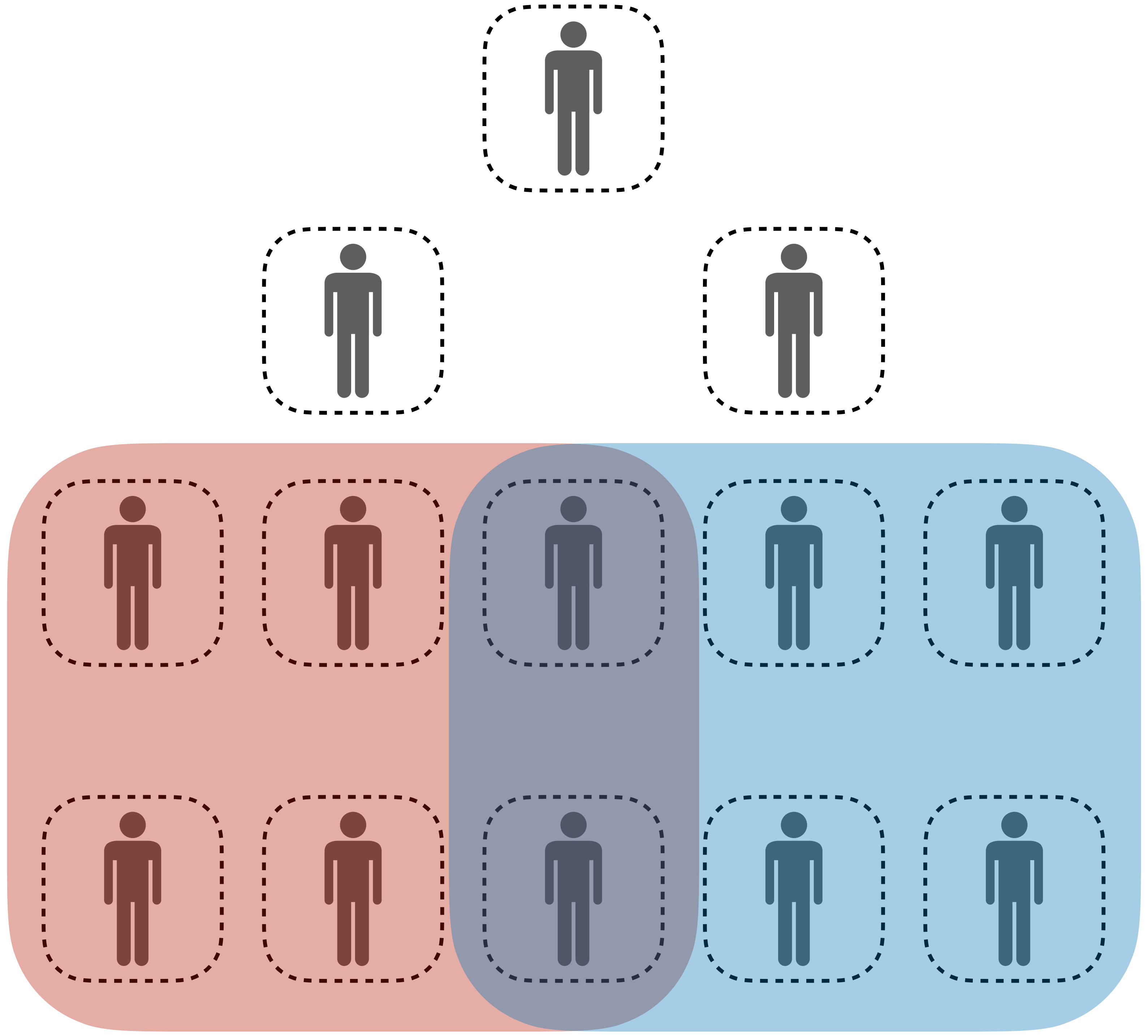
Solution: Built-in structures to deal with flaws and failures



Expectation

Architecture Inspired by Human Organizations

Communication and Sanity Checks



Local Sanity Checks

Synthesizer to reconcile inconsistencies between parts.

1. Hierarchy of overlapping committees.
2. Continuous interaction and communication.
3. When failure occurs, a story can be made, combining the members' observations.

An Architecture to Mitigate Common Problems

Synthesizer to reconcile inconsistencies between parts.

Reconcile conflicting reasons.

Local Sanity Checks


future tense

The Trollable Self-Driving Car

Justify new examples.

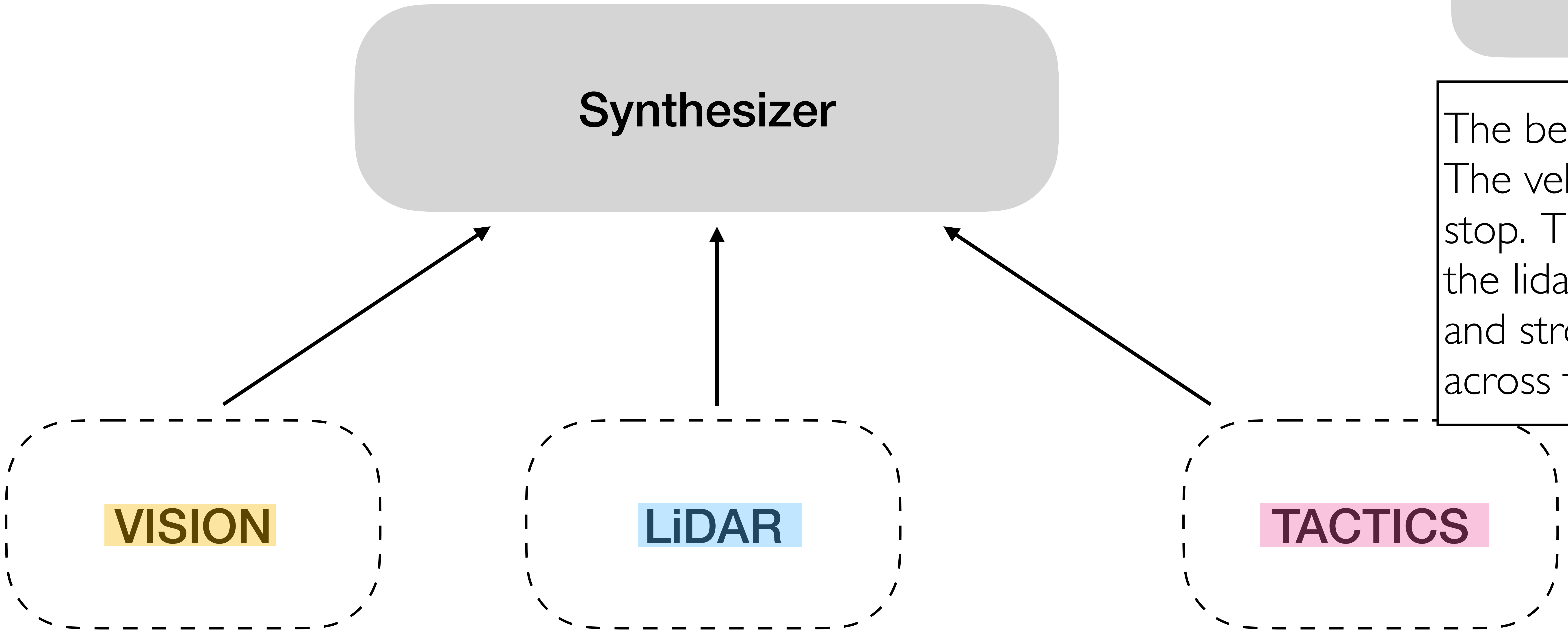
An Existing Problem

The Uber Accident

Solution: Internal Communication

Anomaly Detection through Explanations

Synthesizer to reconcile inconsistencies between monitor outputs.



The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.

Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.

Limited Internal Reasoning

A Google self-driving car caused a crash for the first time

A bad assumption led to a minor fender-bender

Serious safety lapses led to Uber's fatal self-driving crash, new documents suggest

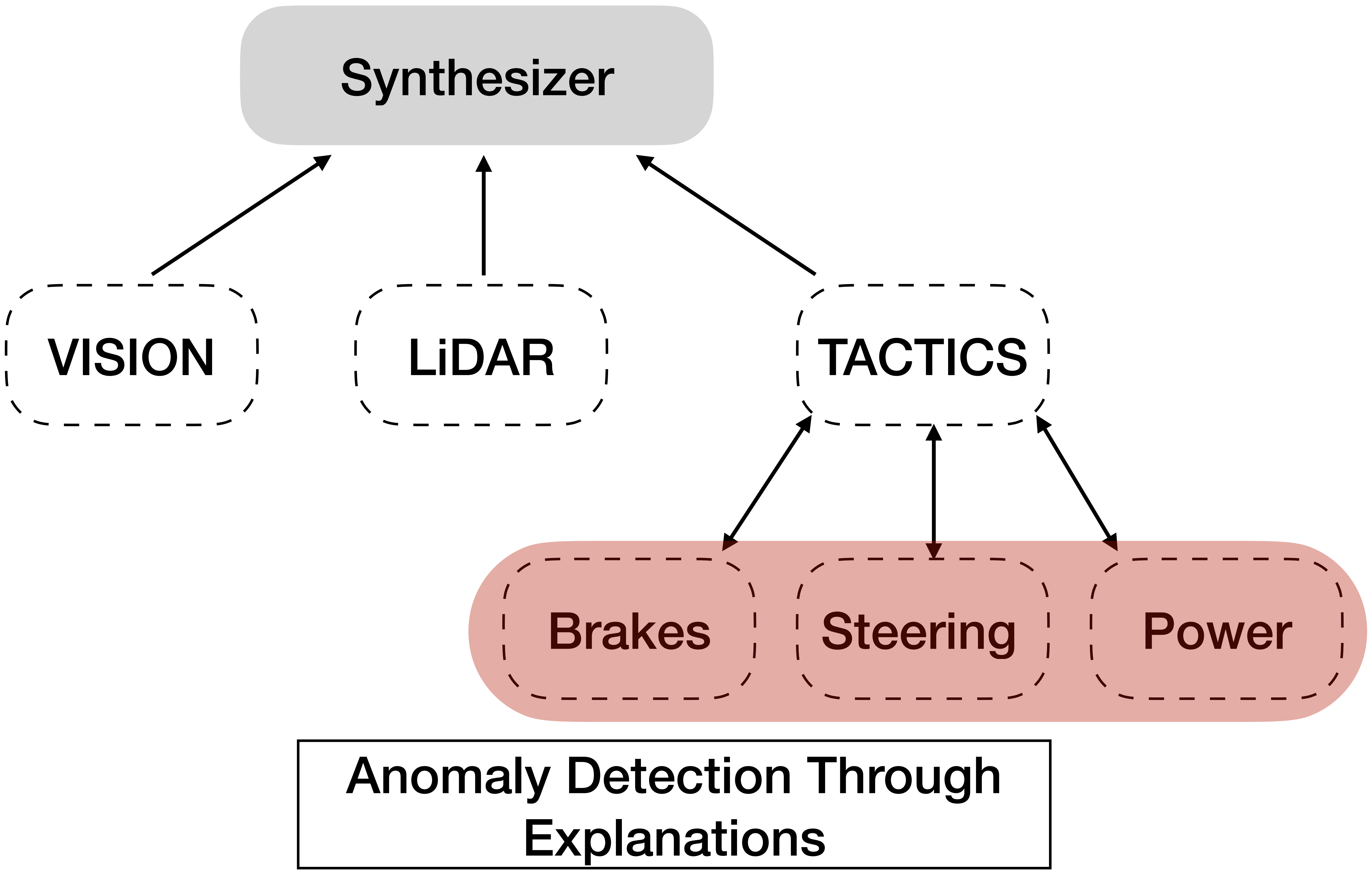
My Herky-Jerky Ride in General Motors' Ultra-Cautious Self Driving Car

GM and Cruise are testing vehicles in a chaotic city, and the tech still has a ways to go.

Reconciling Internal Disagreements

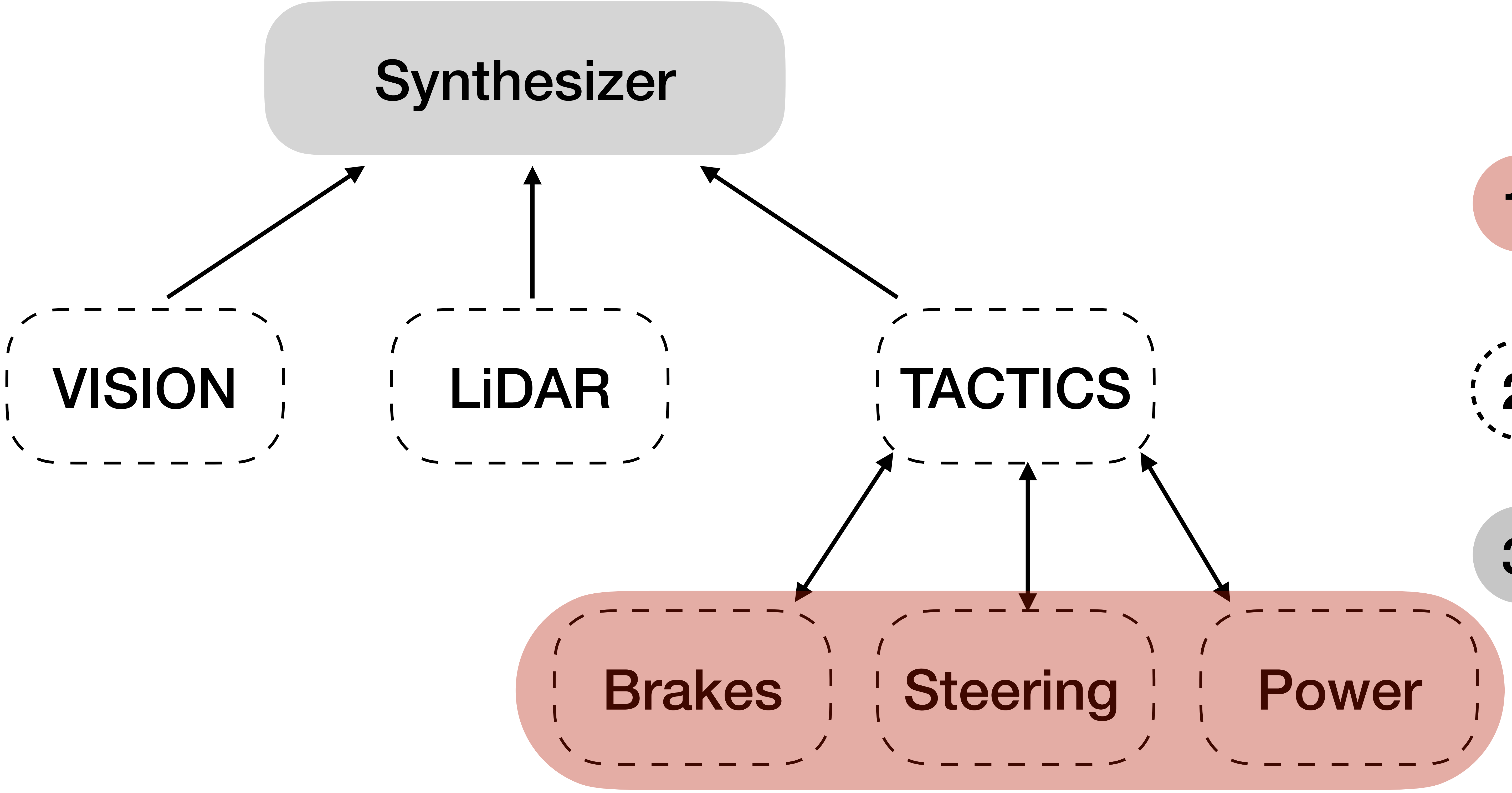
With an Organizational Architecture

- Monitored subsystems combine into a system architecture.
- Explanation synthesizer to deal with *inconsistencies.*
    - Argument tree.
    - Queried for support or counterfactuals.



Anomaly Detection through Explanations

Reasoning in Three Steps



1. Generate Symbolic Qualitative Descriptions for each committee.
2. Input qualitative descriptions into local "reasonableness" monitors.
3. Use a synthesizer to reconcile inconsistencies between monitors.

3. Use a synthesizer to reconcile inconsistencies between monitors.

Synthesizer + Priority Hierarchy → Abstract Goals

- Explanation synthesizer to deal with *inconsistencies.*
  - Argument tree.
  - Queried for support or counterfactuals.

1. Passenger Safety
2. Passenger Perceived Safety
3. Passenger Comfort
4. Efficiency (e.g. Route efficiency)

→ A passenger is safe if:

- The vehicle proceeds at the same speed and direction.
- The vehicle avoids threatening objects.

3. Use a synthesizer to reconcile inconsistencies between monitors.

$$
\begin{aligned}
&(\forall s, t \in STATE, v \in VELOCITY \\
&\quad \big((self, moving, v), \textbf{state}, s\big) \wedge \\
&\quad (t, \textbf{isSuccesorState}, s) \wedge \\
&\quad \big((self, moving, v), \textbf{state}, t\big) \wedge \\
&\quad (\nexists x \in OBJECTS \textbf{ s.t.} \\
&\quad \big((x, isA, threat), \textbf{state}, s\big) \vee \\
&\quad \big((x, isA, threat), \textbf{state}, t\big))) \\
&\qquad \Rightarrow \big(\textbf{passenger, hasProperty, safe}\big)
\end{aligned}
$$

A passenger is safe if:

- The vehicle proceeds at the same speed and direction.
- The vehicle avoids threatening objects.

$$
\begin{aligned}
&(\forall s \in STATE, x \in OBJECT, v \in VELOCITY \\
&\quad \big((x, moving, v), \textbf{state}, s\big) \wedge \\
&\quad \big((x, locatedNear, self), \textbf{state}, s\big) \wedge \\
&\quad \big((x, isA, large\_object), \textbf{state}, s\big) \\
&\qquad \Leftrightarrow \big((x, isA, threat), \textbf{state}, s\big))
\end{aligned}
$$

3. Use a synthesizer to reconcile inconsistencies between monitors.

$$
\begin{aligned}
&(\forall s, t \in STATE, v \in VELOCITY \\
&\quad \big((self, moving, v), \textbf{state}, s\big) \wedge \\
&\quad (t, \textbf{isSuccesorState}, s) \wedge \\
&\quad \big((self, moving, v), \textbf{state}, t\big) \wedge \\
&\quad (\nexists x \in OBJECTS \textbf{ s.t.} \\
&\quad \big((x, isA, threat), \textbf{state}, s\big) \vee \\
&\quad \big((x, isA, threat), \textbf{state}, t\big))) \\
&\qquad \Rightarrow \big(\textbf{passenger, hasProperty, safe}\big)
\end{aligned}
$$

Abstract Goal Tree

```
'passenger is safe',
AND(
  ‘safe transitions’,
  NOT(‘threatening objects’)
```

3. Use a synthesizer to reconcile inconsistencies between monitors.

Abstract Goal Tree

```
'passenger is safe',
AND(
  'safe transitions',
  NOT('threatening objects')
```

List of Rules → Backwards Chain → AND/OR TREE

```
IF ( AND('moving (?v) at state (?y)',
             '(?z) succeeds (?y)',
             'moving (?v) at state (?z)'),
     THEN('safe driving at (?v) during (?y) and (?z)'))

IF (OR('obj is not moving',
       'obj is not located near',
       'obj is not a large object')),
    THEN('obj not a threat at (?x)'))

IF (AND('obj not a threat at (?y)',
        'obj not a threat at (?z)',
        '(?z) succeeds (?z)',
    THEN('obj is not a threat between (?y) and (?z)'))
```

```
passenger is safe at V between s and t
  AND (AND (moving V at state s
             t succeeds s
             moving V at state t )
        AND (
             OR ( obj is not moving at s
                  obj is not locatedNear at s
                  obj is not a large object at s )
             OR ( obj is not moving at t
                  obj is not locatedNear at t
                  obj is not a large object at t ) ) )
```

3. Use a synthesizer to reconcile inconsistencies between monitors.

```
(monitor, judgement, unreasonable)
(input, isType, labels)
(all_labels, inconsistent, negRel)
(isA, hasProperty, negRel)
…
(all_labels, notProperty, nearMiss)
(all_labels, locatedAt, consistent)
(monitor, recommend, discount)
```

```
(monitor, judgement, reasonable)
(input, isType, sensor)
…
(input_data[4], hasSize, large)
(input_data[4], IsA, large_object)
(input_data[4], moving, True)
(input_data[4], hasProperty, avoid)
…
(monitor, recommend, avoid)
```

!

```
(monitor, judgement, reasonable)
(input, isType, history)
(input_data, moving, True)
(input_data, direction, forward)
(input_data, speed, fast)
(input_data, consistent, True)
(monitor, recommend, proceed)
```

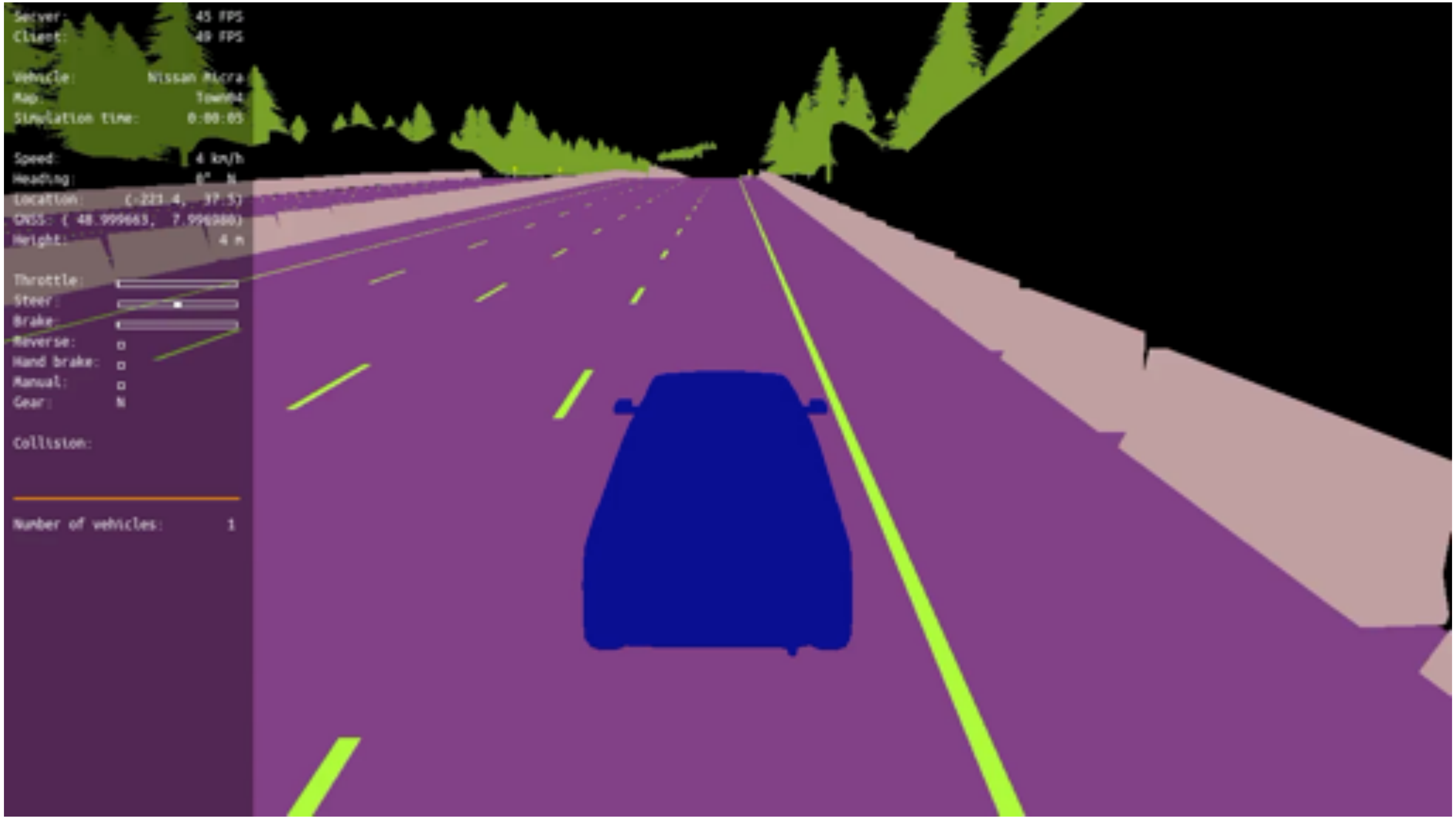
Abstract Goal Tree

```
'passenger is safe',
AND(
  'safe transitions',
  NOT('threatening objects')
```

!

The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.

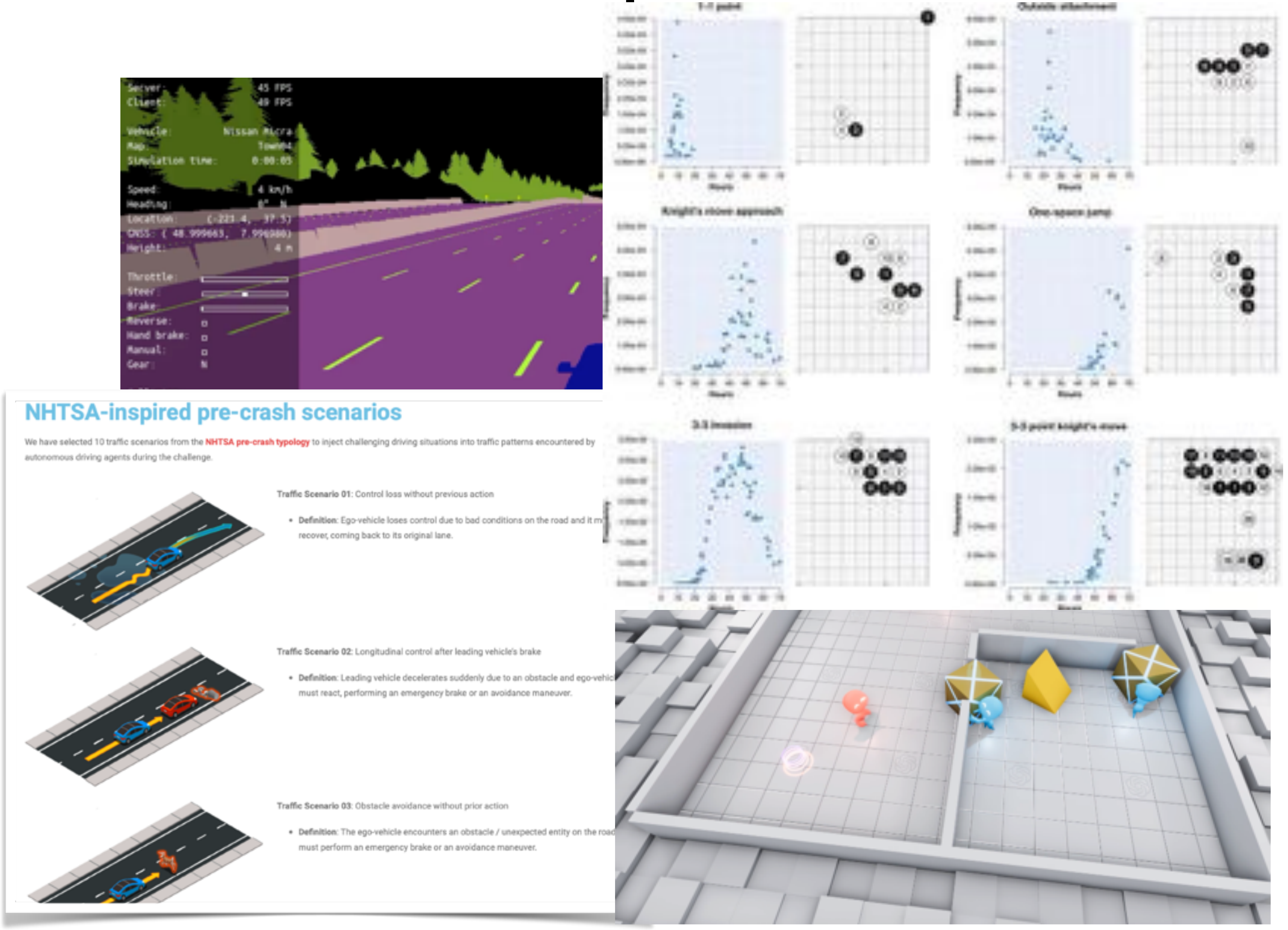
Uber Example in Simulation



L. H. Gilpin, V. Penubarthi and L. Kagal, "Explaining Multimodal Errors in Autonomous Vehicles," *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, 2021, pp. 1-10, doi: 10.1109/DSAA53316.2021.9564178.

Evaluation of Error Detection is Difficult

Real-world Inspired Scenarios



Reconcile Inconsistencies

- Detection: Generate logs from scenarios to detect failures.
- Insert errors: Scrambling *multiple* labels on existing datasets.
- Real errors: Examining errors on the validation dataset of NuScenes leaderboard.

Priority	Correctness	False Positives	False Negatives
No synthesizer	85.6%	7.1%	7.3%
Single subsystem	88.9%	7.9%	3.2%
Safety	93.5%	4.8%	1.7%

Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

<u>Adversarial Examples as a Testing Framework for Autonomous System Robustness.</u>

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.

Vision: Real World Adversarial Examples

“Realistic” Adversarial examples

L. H. Gilpin, A. Amos-Binks, "Close Syntax but Far Semantics: A Risk Management Problem for Autonomous Vehicles.” *To Appear in Abstracts of the AAAI Fall Symposium on Cognitive Systems for Anticipatory Thinking.*

Vision: Real World Adversarial Examples

Anticipatory Thinking Layer for Error Detection

"Realistic" Adversarial examples

The traffic lights are on top of the truck. The lights are not illuminated. The lights are moving at the same rate as the truck, therefore this is not a "regular" traffic light for slowing down and stopping at.

Testing Framework in Two Parts

The traffic lights are on top of the truck. The lights are not illuminated. The lights are moving at the same rate as the truck, therefore this is not a "regular" traffic light for slowing down and stopping at.

Explanatory Error Detection

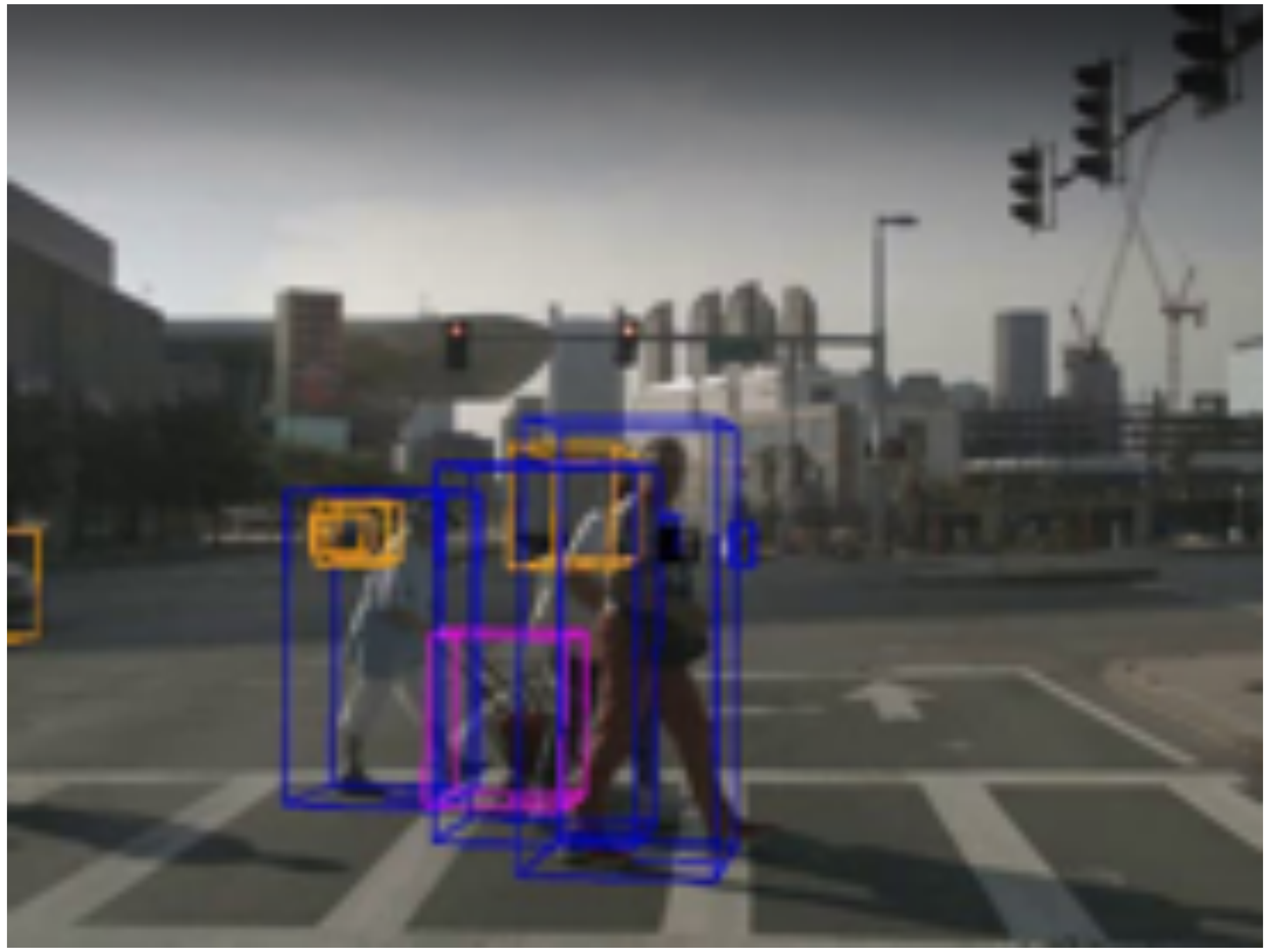
Content generation

!

Deploy

Lack of Data and Challenges for AVs

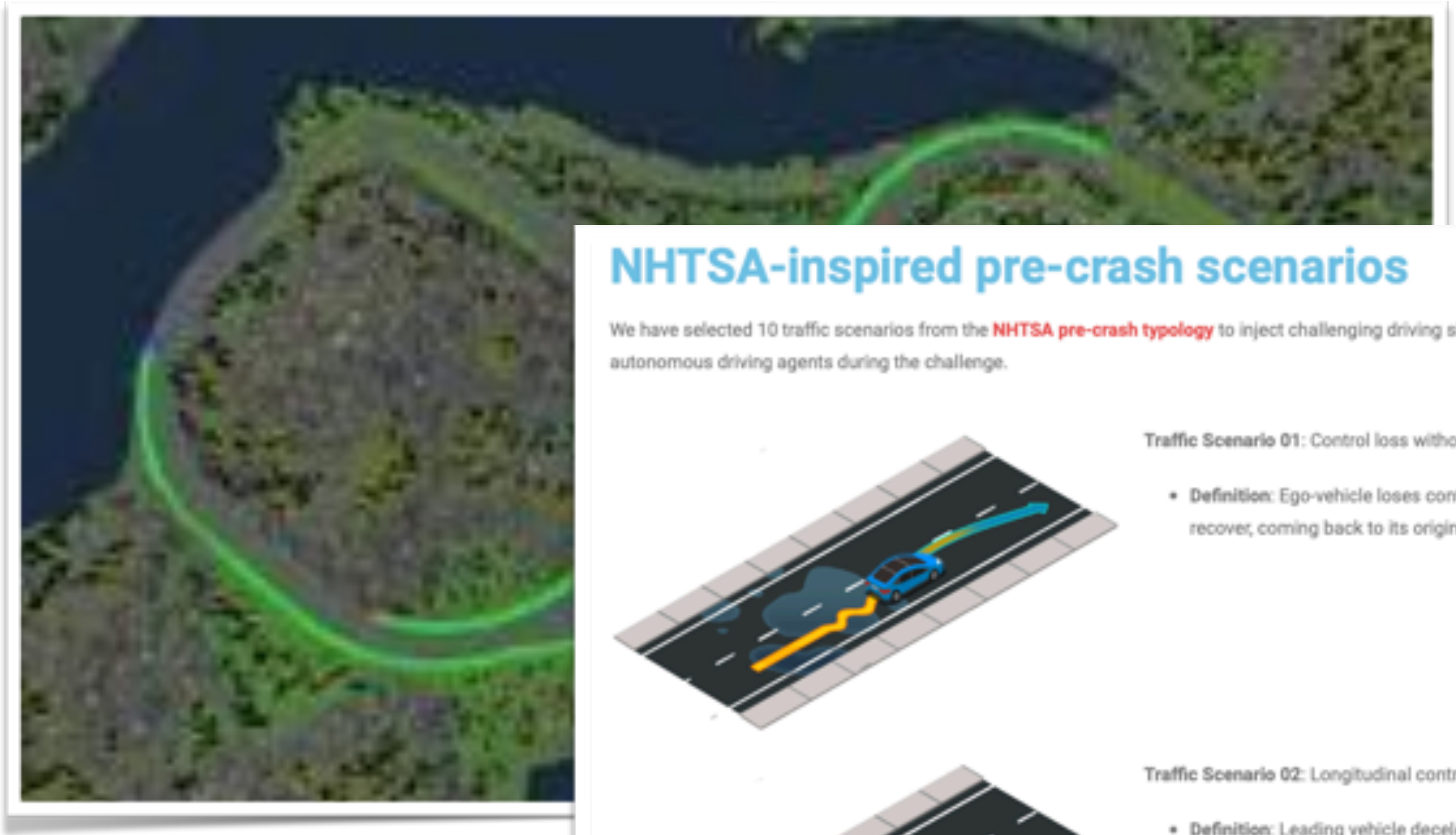
- Existing Challenges
    - Targeted as optimizing a mission or trajectory and not safety.
    - Data is hand-curated.
- Failure data is not available
    - Unethical to get it (cannot just drive into bad situations).
    - Want the data to be realistic (usually difficult in simulation).

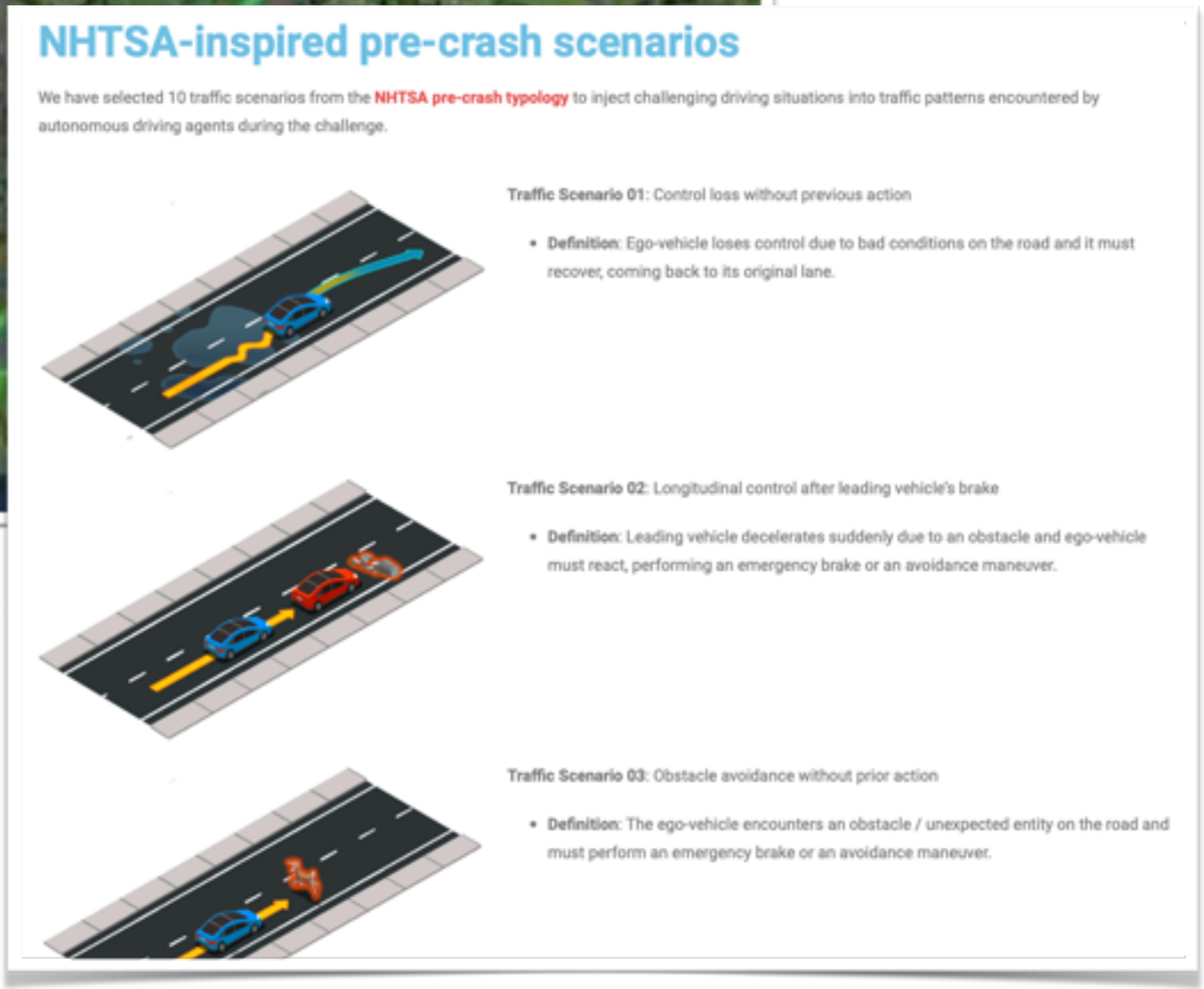


Data from NuScenes

Existing Challenges and Benchmarks

Not Focused on Out of Domain Errors





NHTSA-inspired pre-crash scenarios

We have selected 10 traffic scenarios from the NHTSA pre-crash typology to inject challenging driving situations into traffic patterns encountered by autonomous driving agents during the challenge.

Traffic Scenario 01: Control loss without previous action

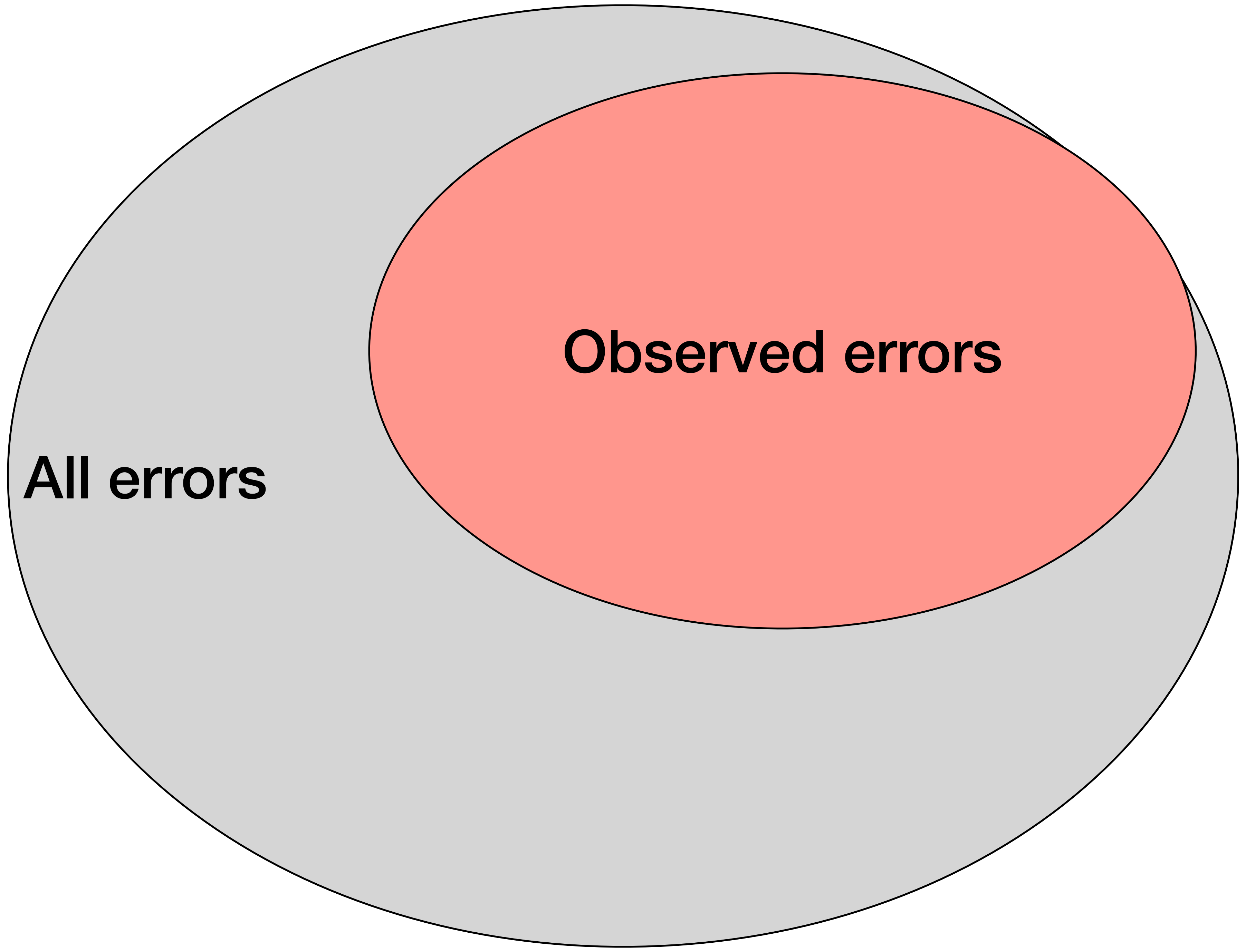
- **Definition:** Ego-vehicle loses control due to bad conditions on the road and it must recover, coming back to its original lane.

Traffic Scenario 02: Longitudinal control after leading vehicle's brake

- **Definition:** Leading vehicle decelerates suddenly due to an obstacle and ego-vehicle must react, performing an emergency brake or an avoidance maneuver.

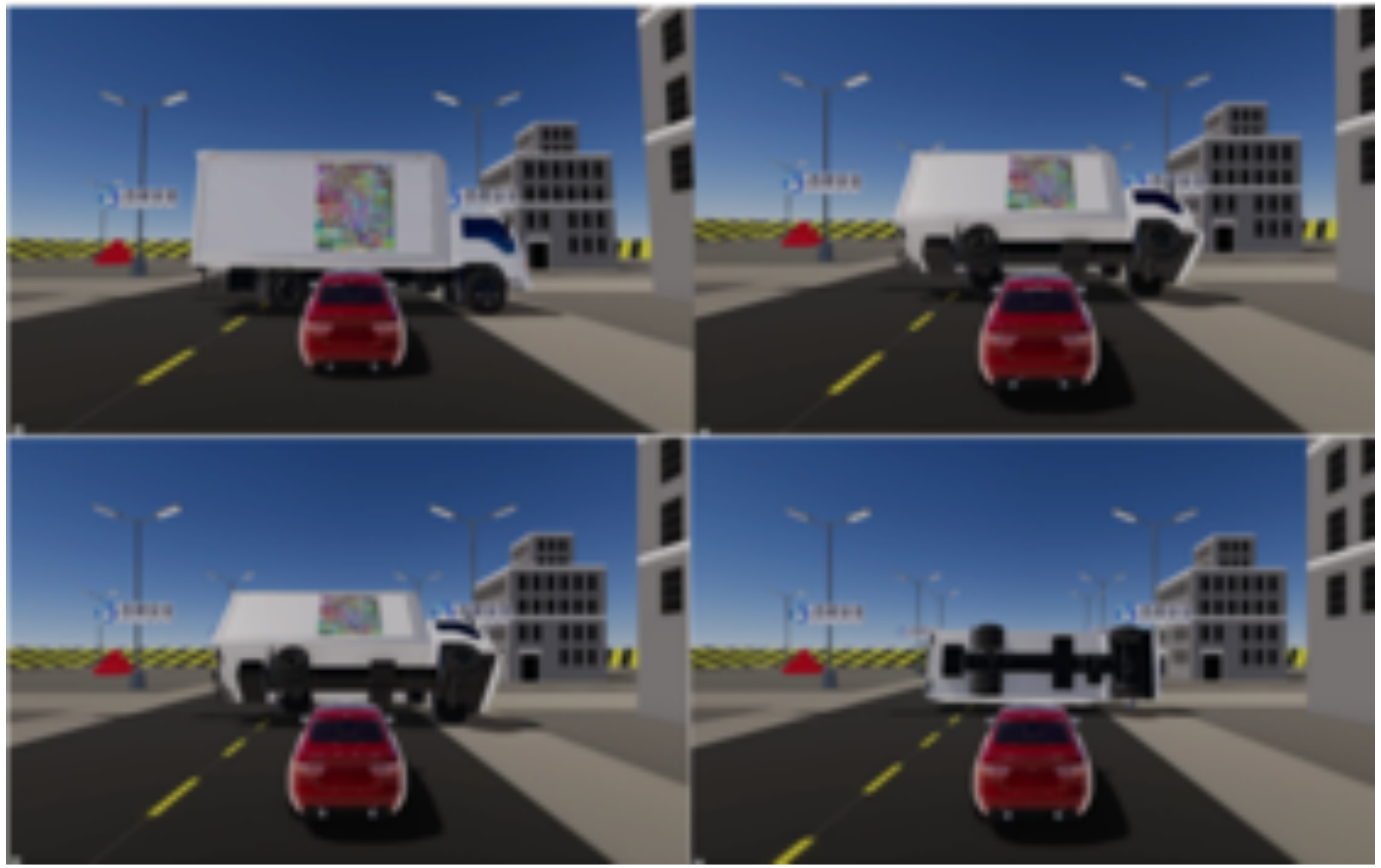
Traffic Scenario 03: Obstacle avoidance without prior action

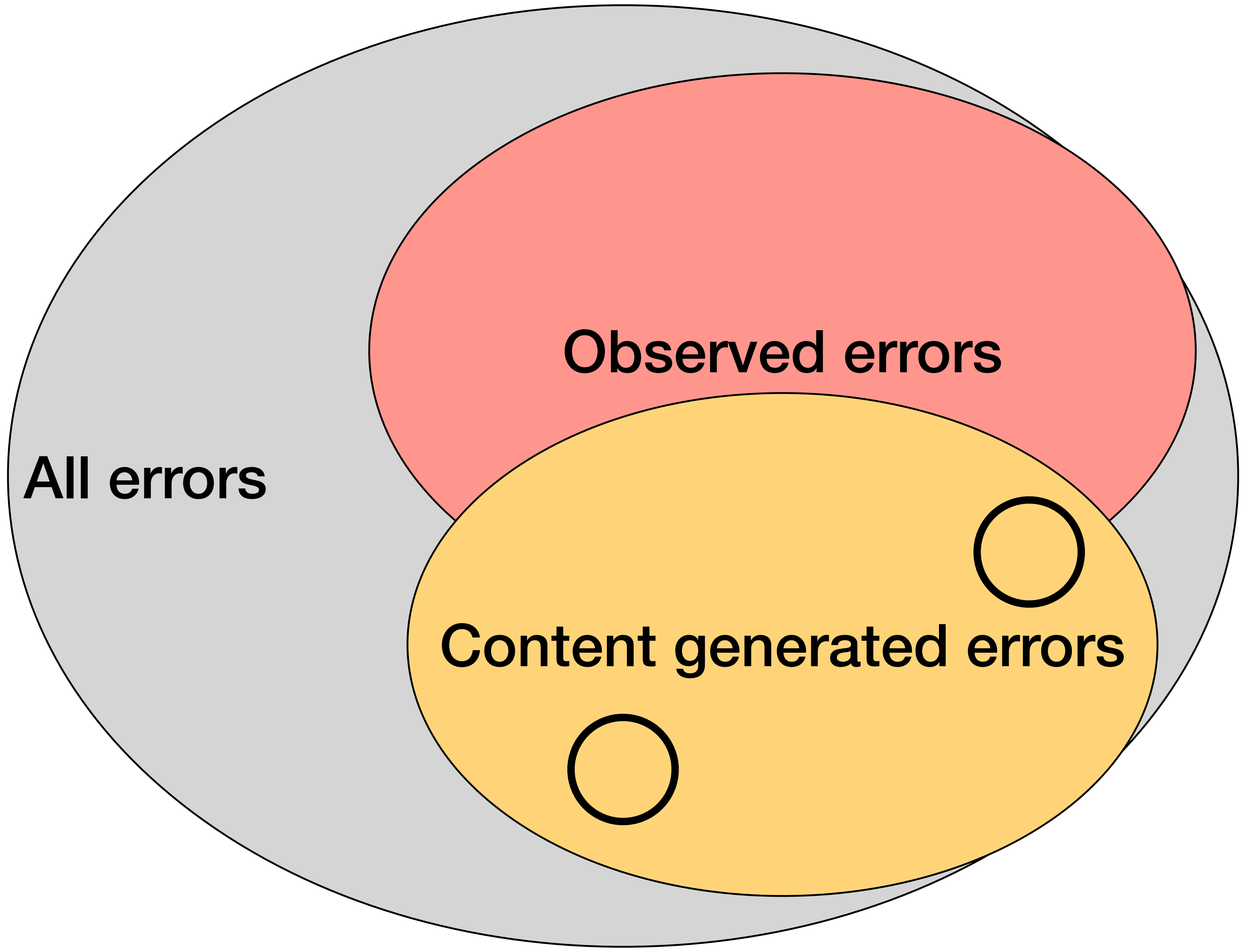
- **Definition:** The ego-vehicle encounters an obstacle / unexpected entity on the road and must perform an emergency brake or an avoidance maneuver.



Other Challenges Not Anticipatory

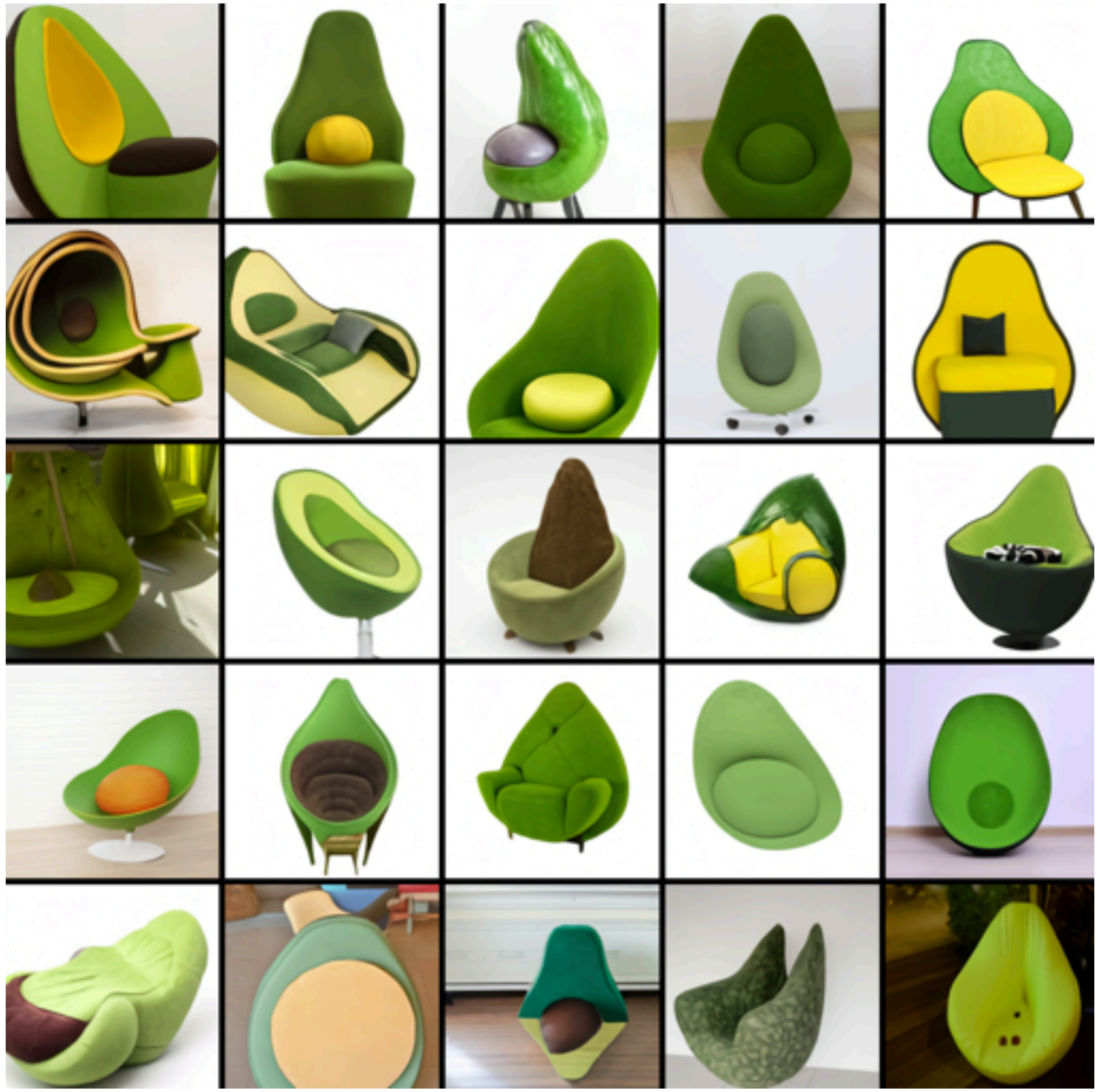
Not Focused on Error Detection





Approach: Content Generation

Anticipatory Thinking Layer for Error Detection



DALL-E Generates “A chair in the shape of an avocado”

Synthetic images produced by StyleGAN, a GAN created by Nvidia researchers.

Approach: Content Generation

Anticipatory Thinking Layer for Error Detection

+

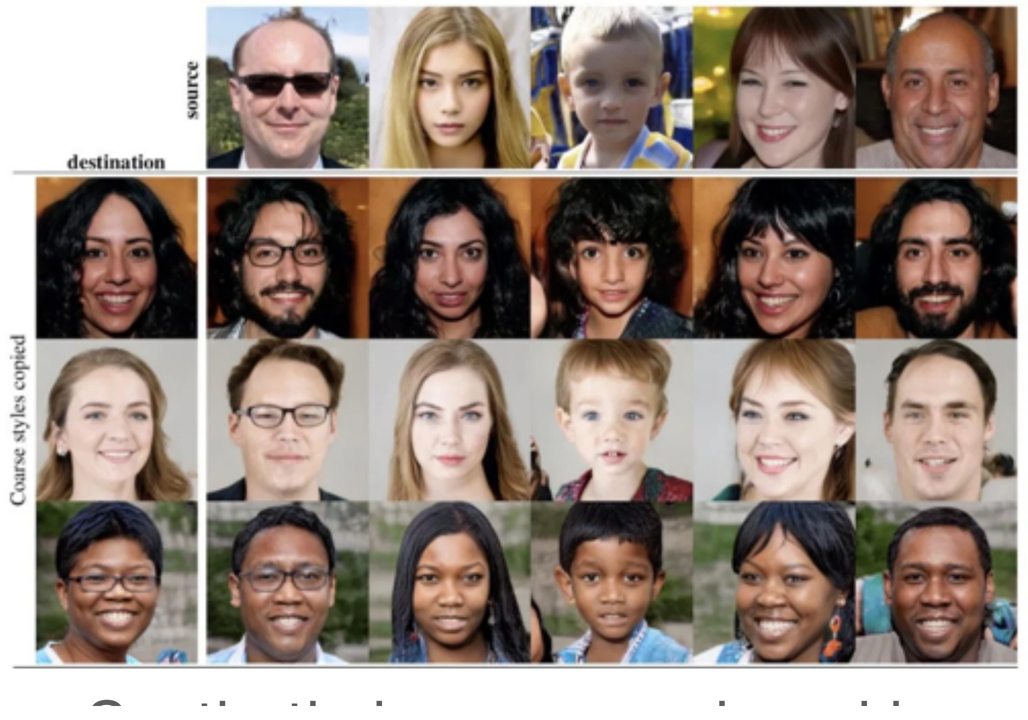
Generate images with shadows before tunnels.

Generate images with fallen signs.

Generate images with trucks carrying traffic lights.

Approach: Content Generation

Anticipatory Thinking Layer for Error Detection



!

Generate images with shadows before tunnels.

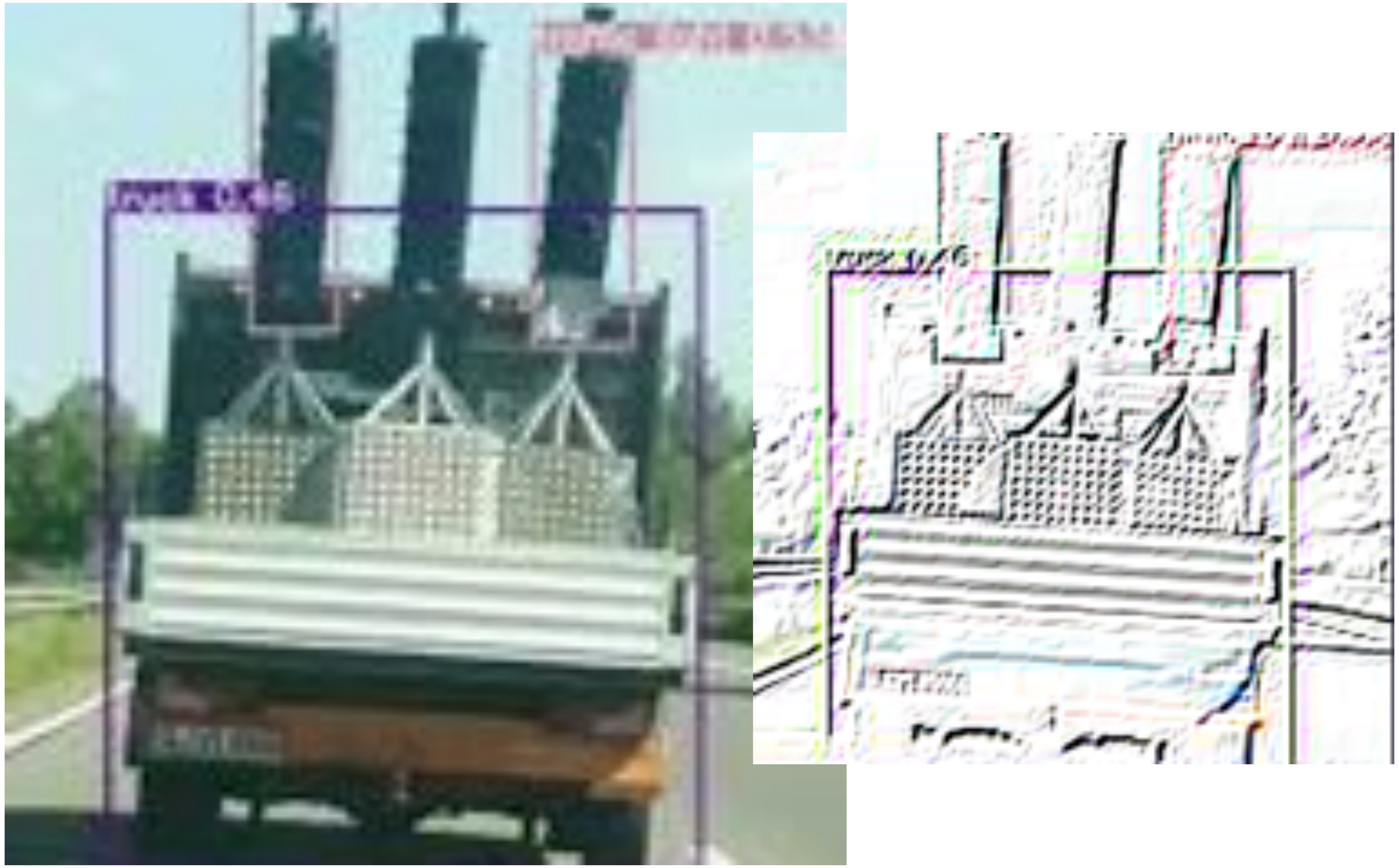
Shadows

Generate images with fallen signs.

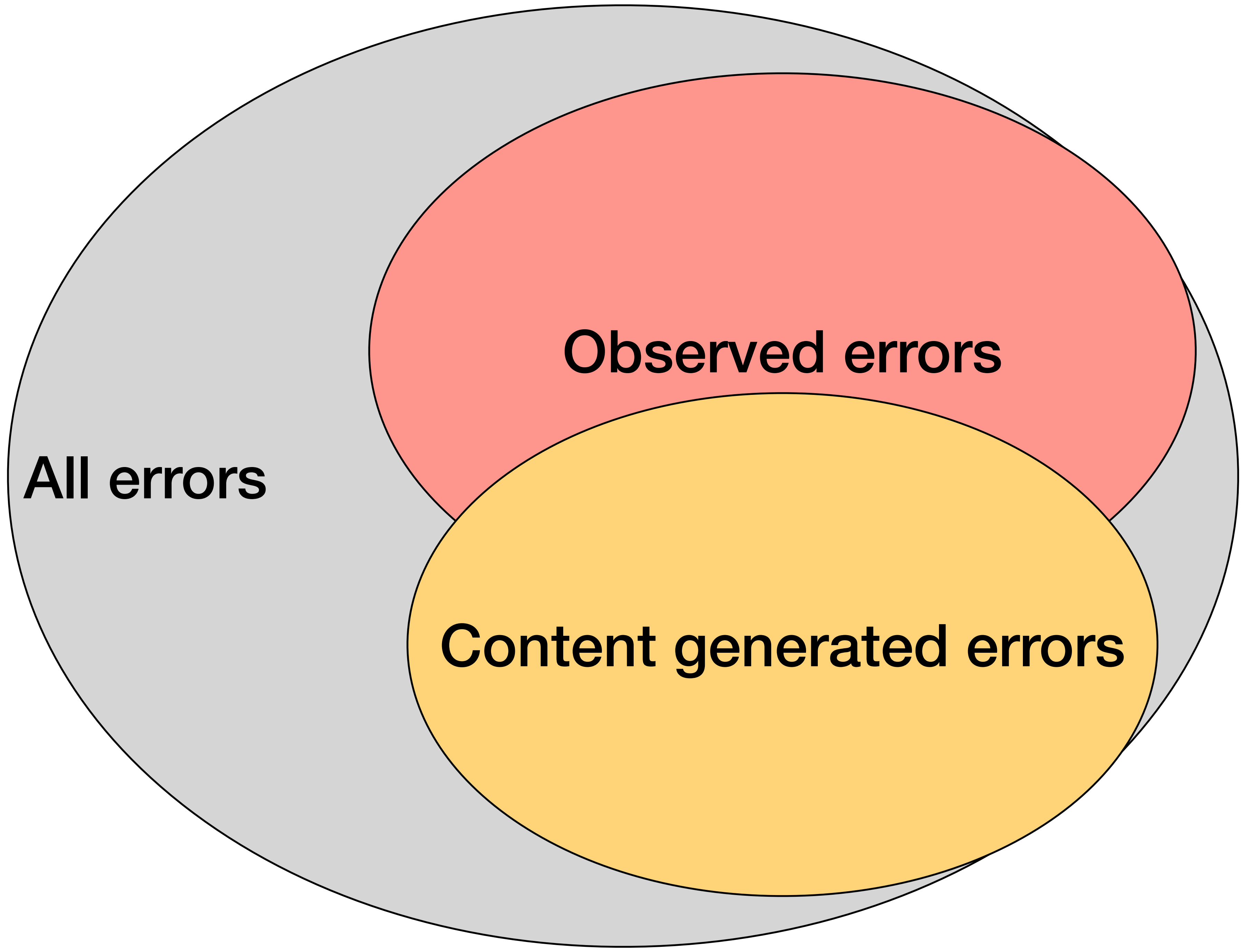
Fallen signs

Generate images with trucks carrying traffic lights.

Need for Context



“Realistic” Adversarial

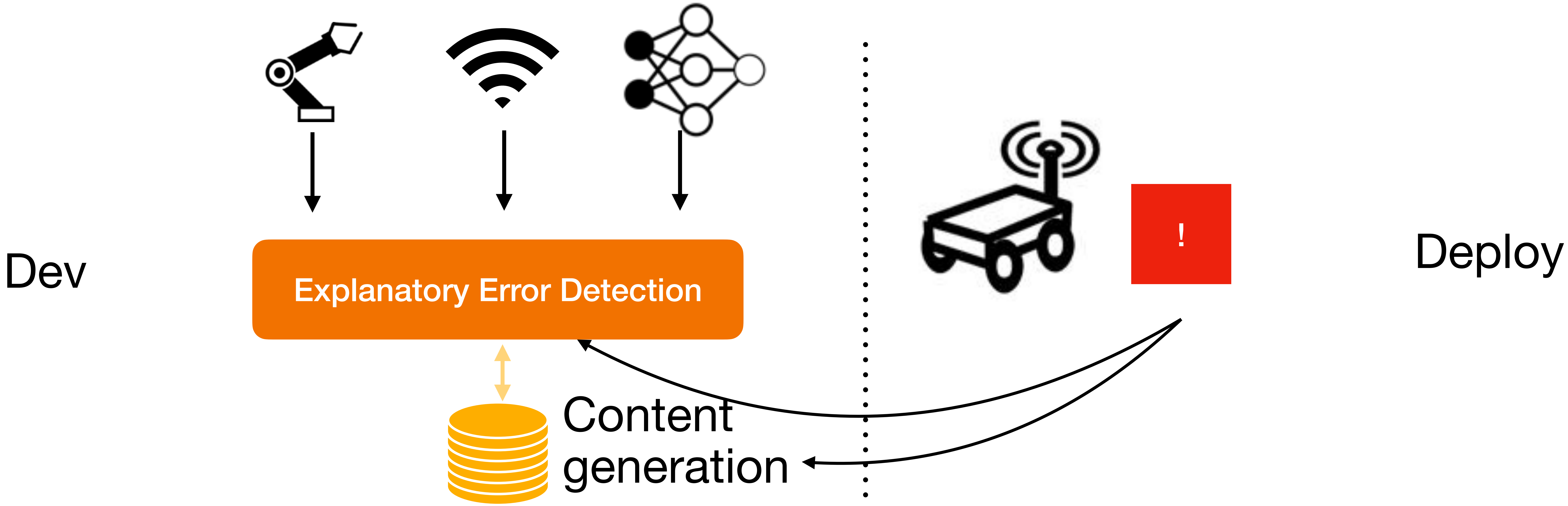


Approach: How it Works

Use Adversarial Images in Dev Testing

- Solution: Use a cognitive architecture that helps to anticipate and understand these failure cases.

- Assess autonomous vehicles for their risk management capabilities **before** being deployed and provide incident level risk management explanations in human readable form.




Dev
Deploy
Isolated error detection
Error detection
Error detection
Error detection
!
All errors
Observed errors
Integrated error detection
Explanatory Error Detection
Content generation
!
All errors
Observed errors
Content generated errors

Impact

Anticipatory Thinking Layer for Error Detection

- Goal - Develop methods that a priori can explain an autonomous vehicle’s ability to manage the risks stemming from errors in perceiving their environment.

- One possible solution is to explain why the autonomous behavior is safe (or risky, trustworthy, etc.) or not.

- Impact - Consumer confidence and safety features, appropriate legal and regulatory oversight.

Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

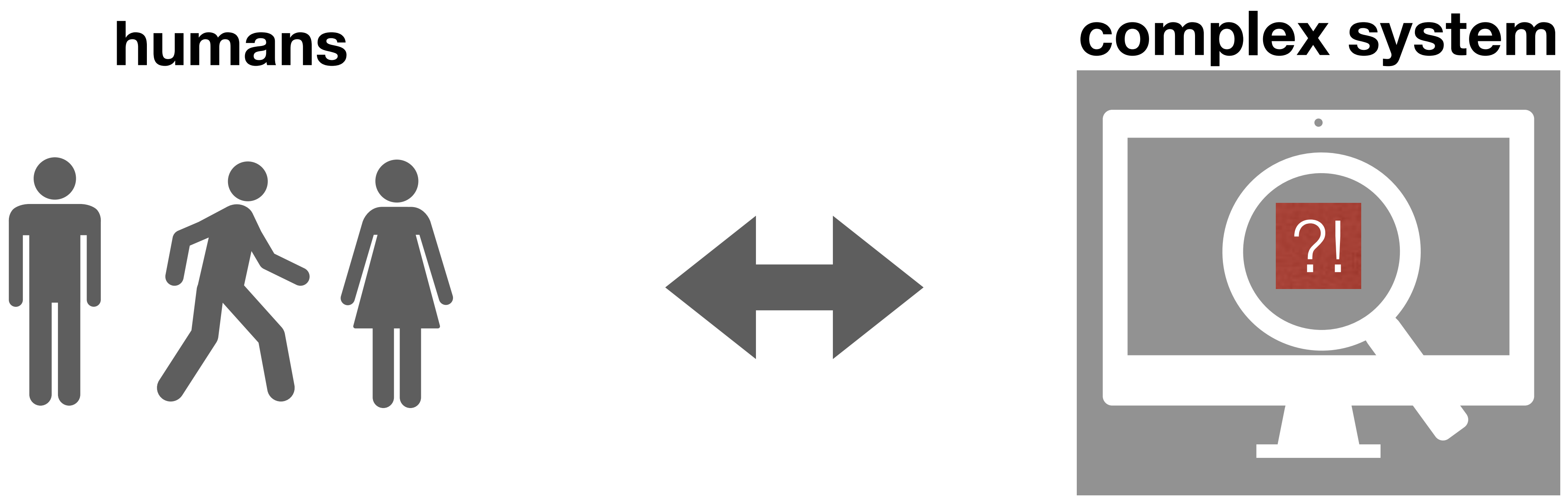
Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

<u>Future work: Explainable Tasks for Robust and Secure Hybrid Systems.</u>

Hybrid Systems with Humans and Machines

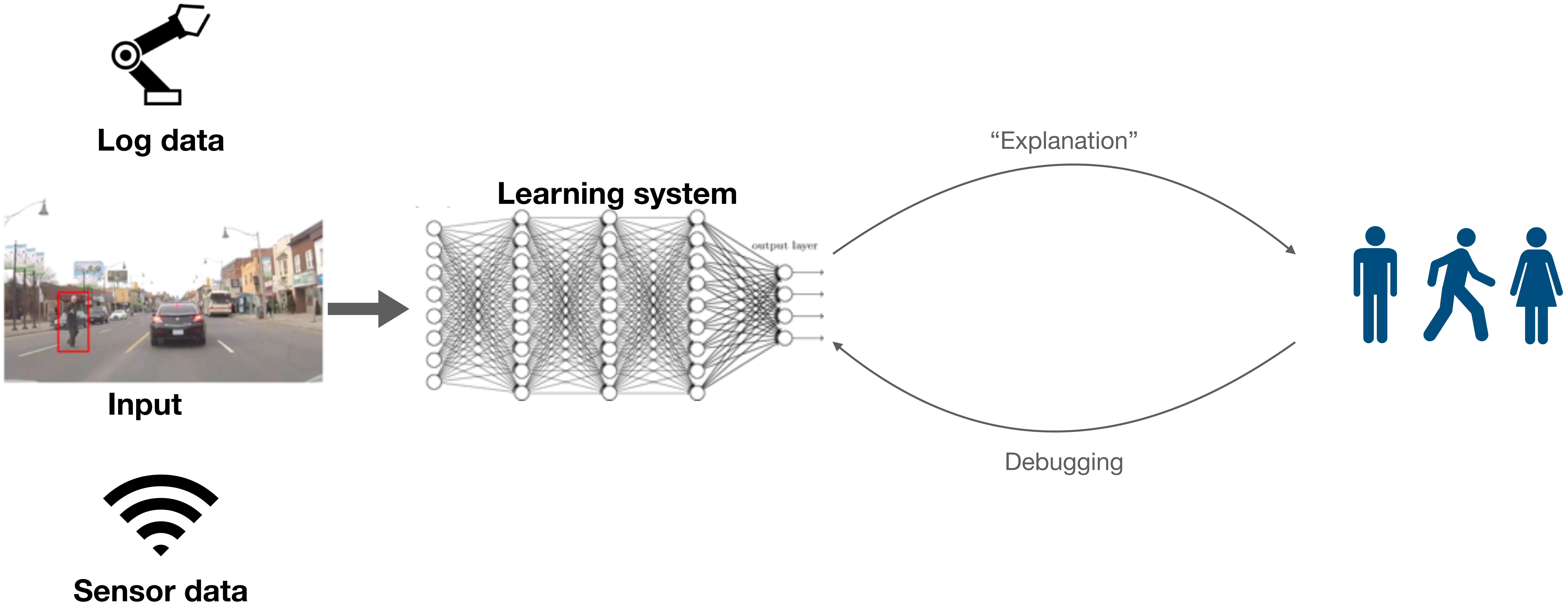
Working Together on Shared Tasks

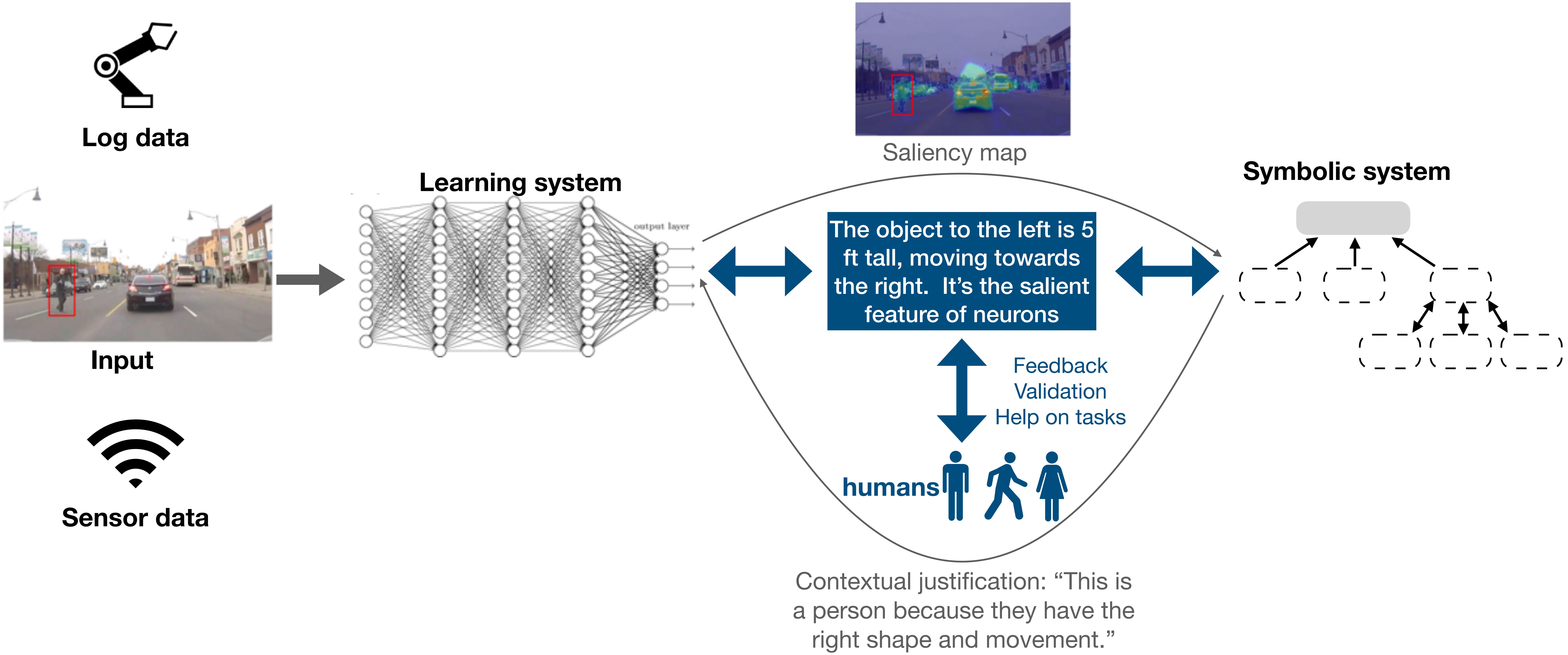


Explanations are a debugging language.

- Debugging: humans can improve complex systems.
- Education: complex systems can “improve” or teach humans.

Ex post facto explanations



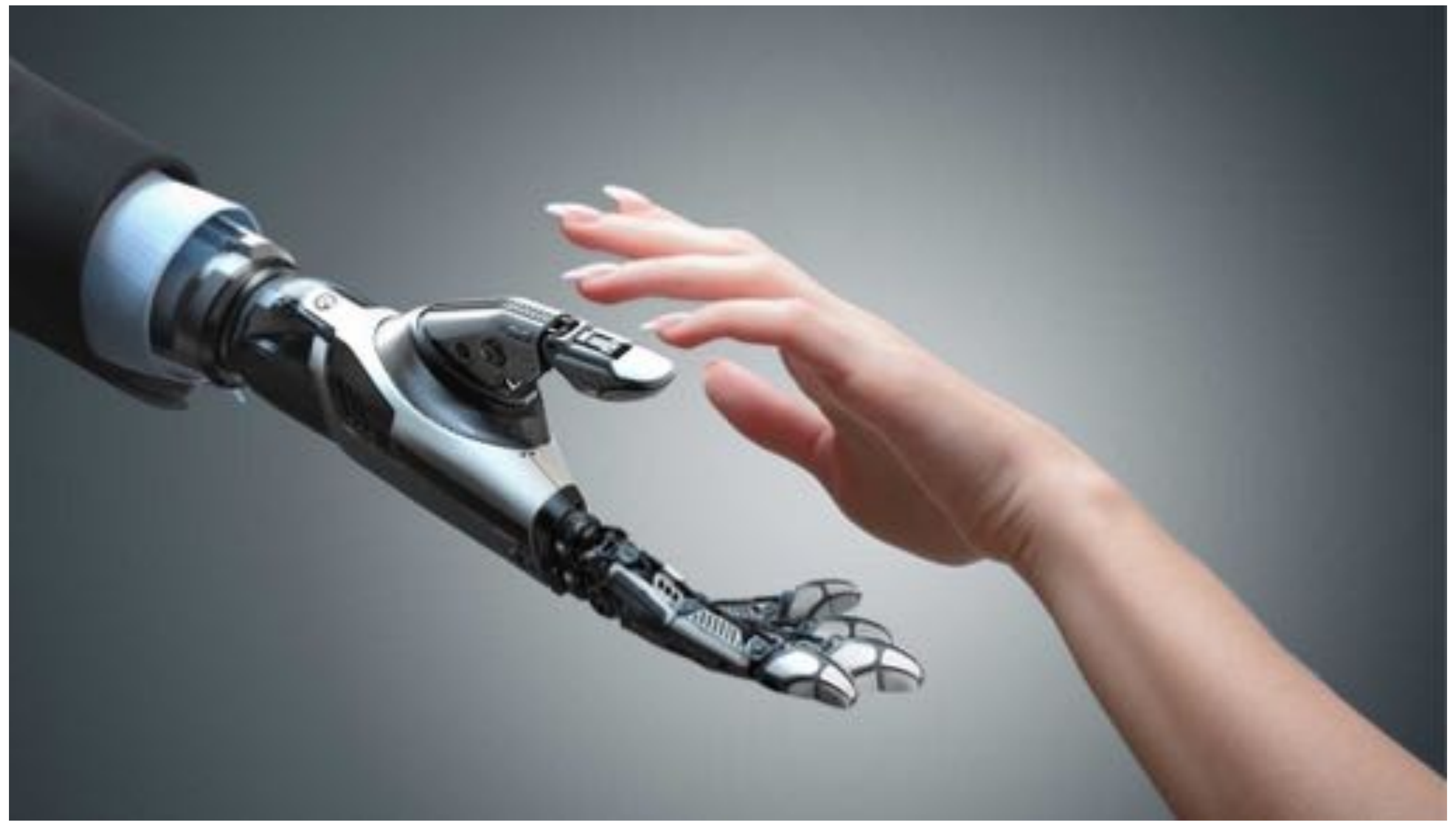


Dev testing Game adversaries Security

Impact

Confidence and Integrity of Systems

Society



Systems that articulately communicate with humans on shared tasks.

Liability

Systems that can testify, answer questions, and provide insights.

Robustness

Dynamic detection of failure and intrusion with precise mitigation.

Contributions

The problem: Autonomous Vehicles are Prone to Failure.

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Explainable Tasks for Robust and Secure Hybrid Systems.