Accountability Layers

Stress-testing Using Explainable AI for Safety-critical Systems

Leilani H. Gilpin
Assistant Professor
Dept. of Computer Science & Engineering, UC Santa Cruz

Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

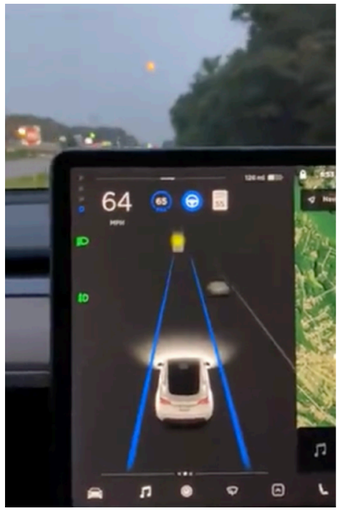
Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.

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

Autonomous Vehicles are Prone to Failure

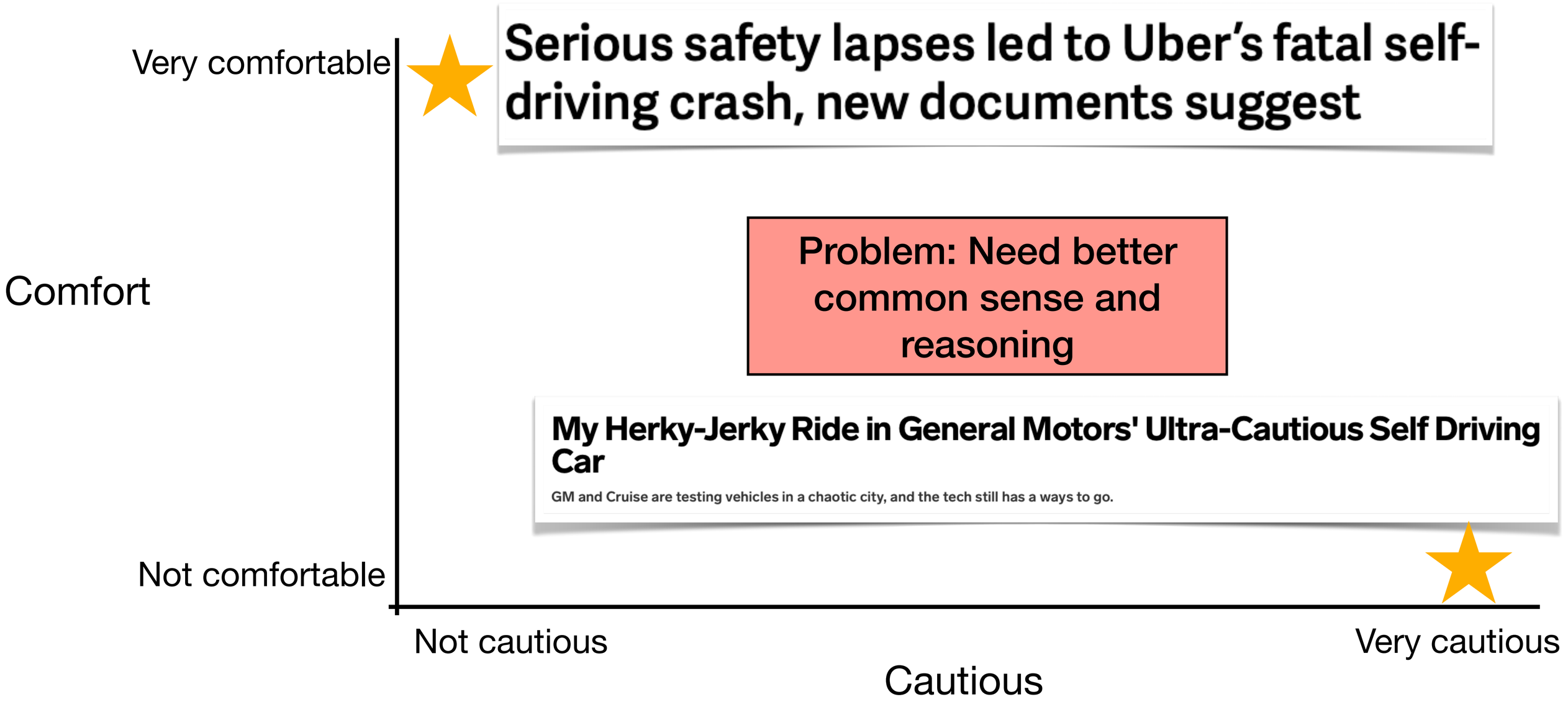
K. Eykholt et al. “Robust Physical-World Attacks on Deep Learning Visual Classification.”



Predictive Inequity in Object Detection

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

Autonomous Vehicle Solutions are at Two Extremes



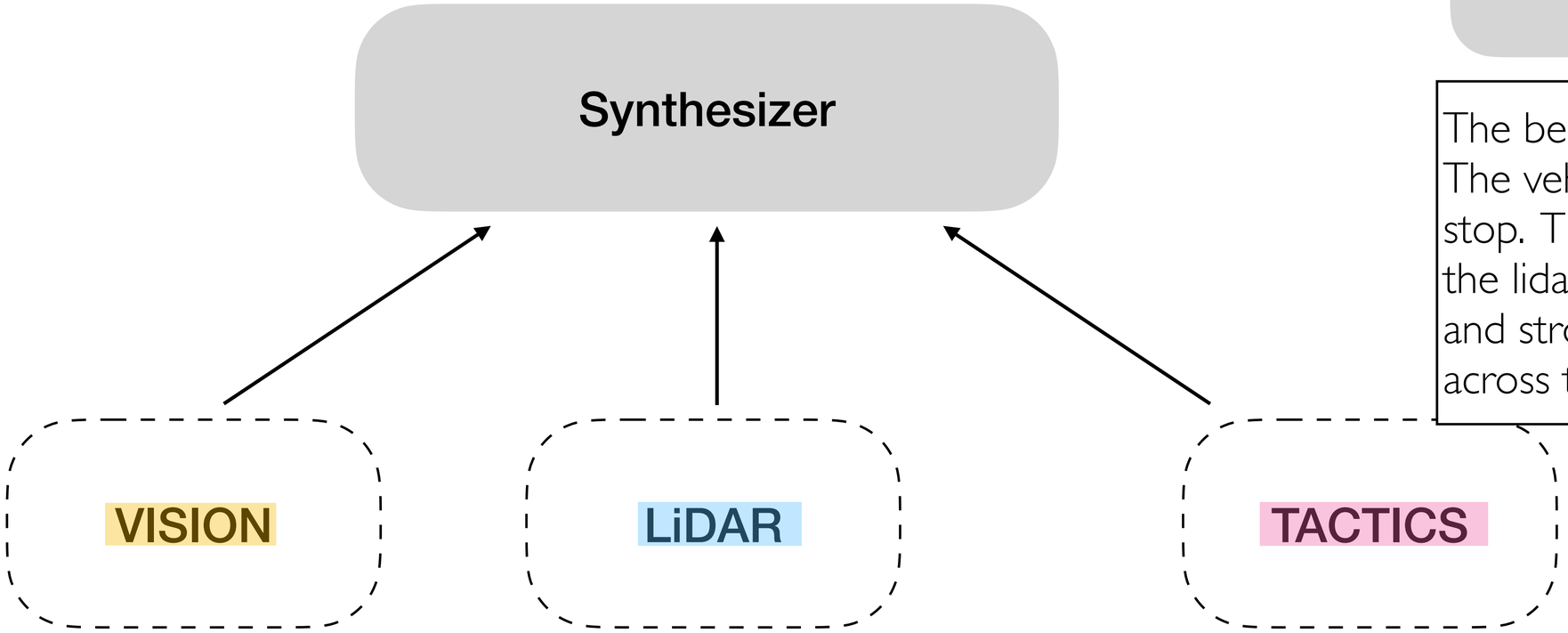
An Existing Problem

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

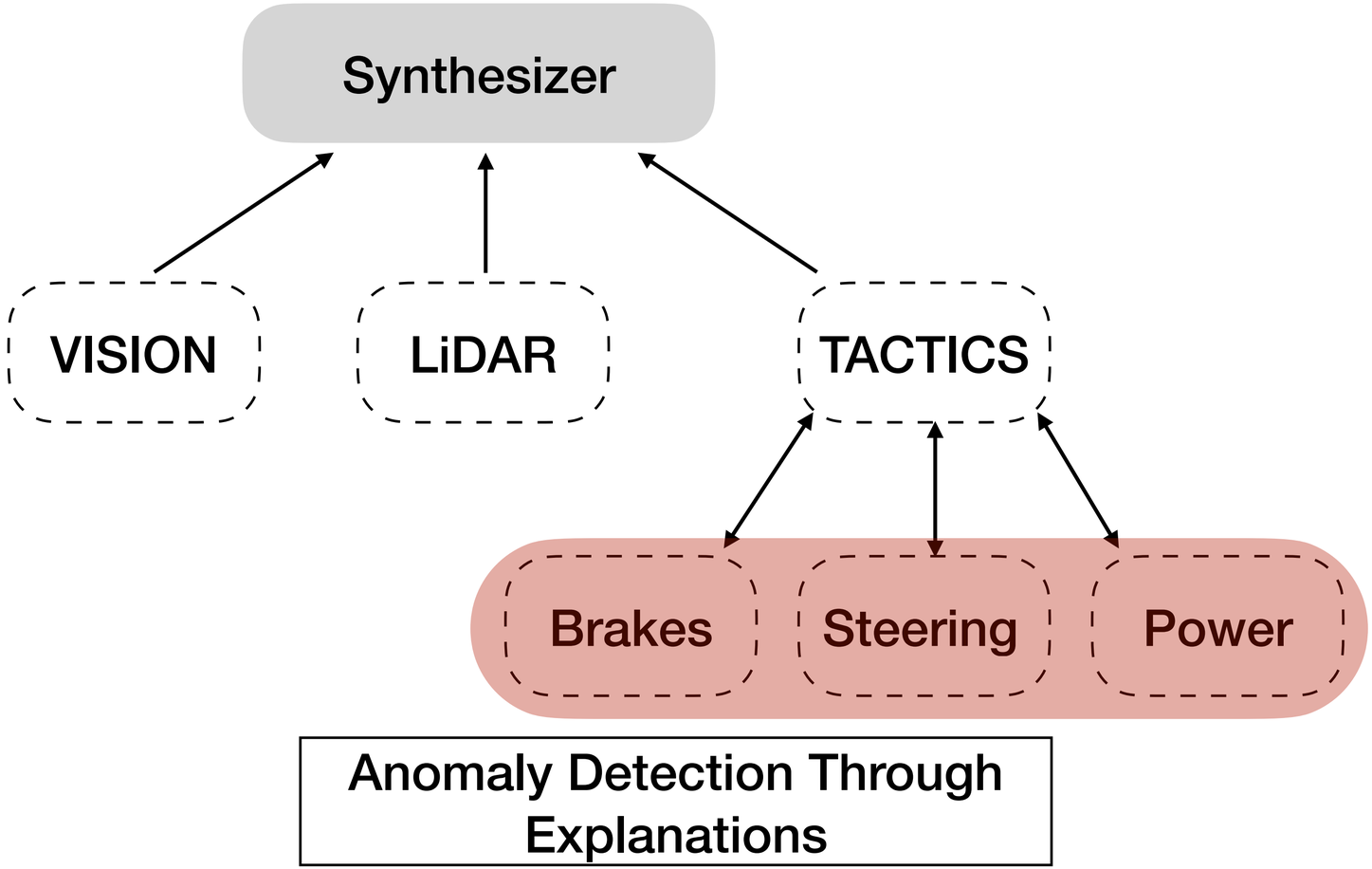
Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.

Reconciling Internal Disagreements

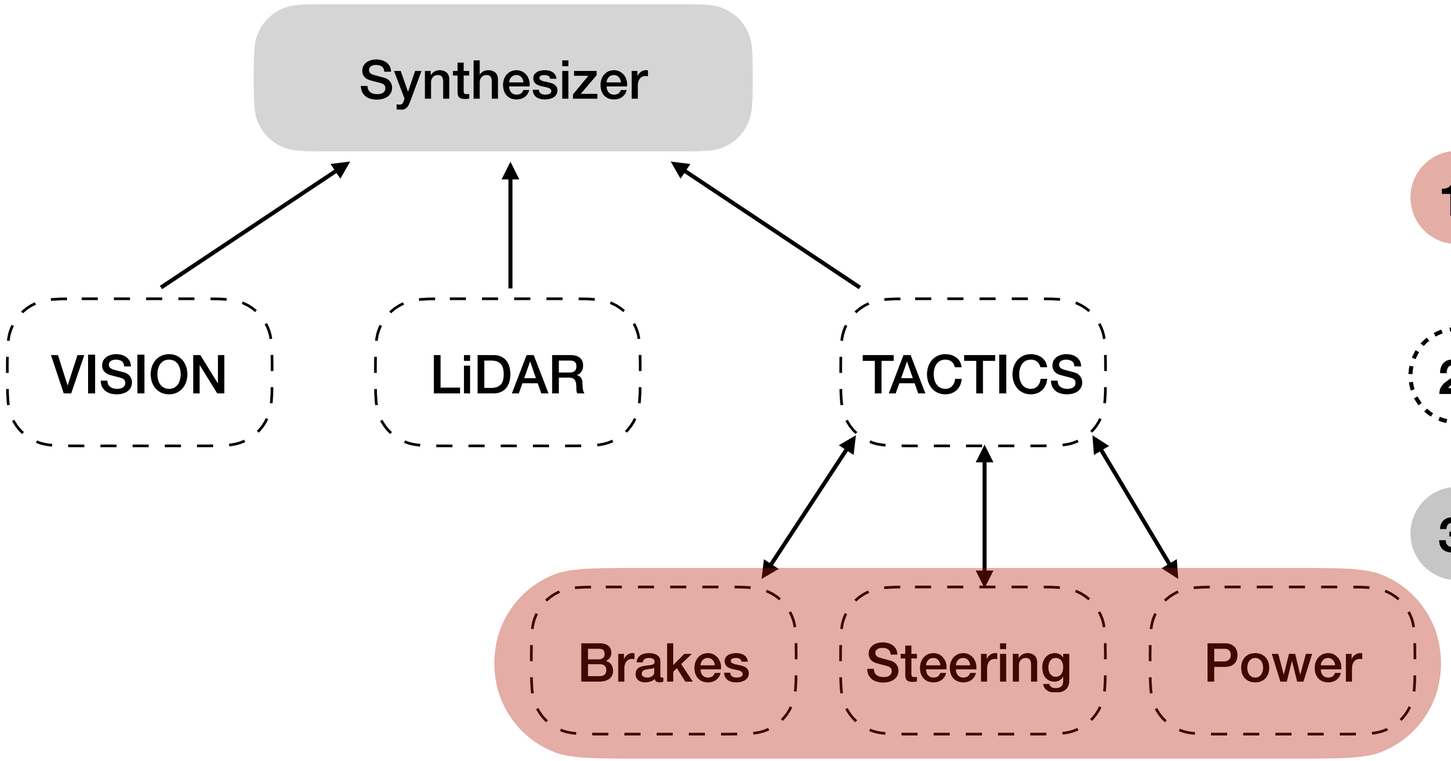
With an Organizational Architecture

- 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.

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

$$\Rightarrow \big(\textbf{passenger, hasProperty, safe}\big)$$

A passenger is safe if:

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

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

$$\Leftrightarrow \big((x, isA, threat), \textbf{state}, s\big))$$

3. Use a synthesizer to reconcile inconsistencies between monitors.

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

$$\Rightarrow \big(\textbf{passenger, hasProperty, safe}\big)$$

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')
```

!

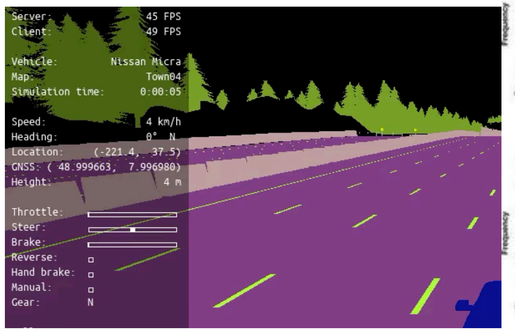
The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.

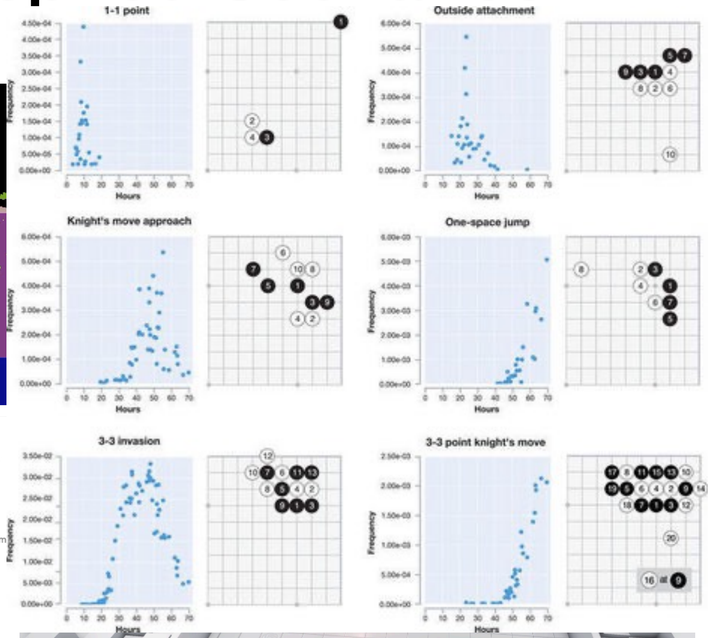
Uber Example in Simulation

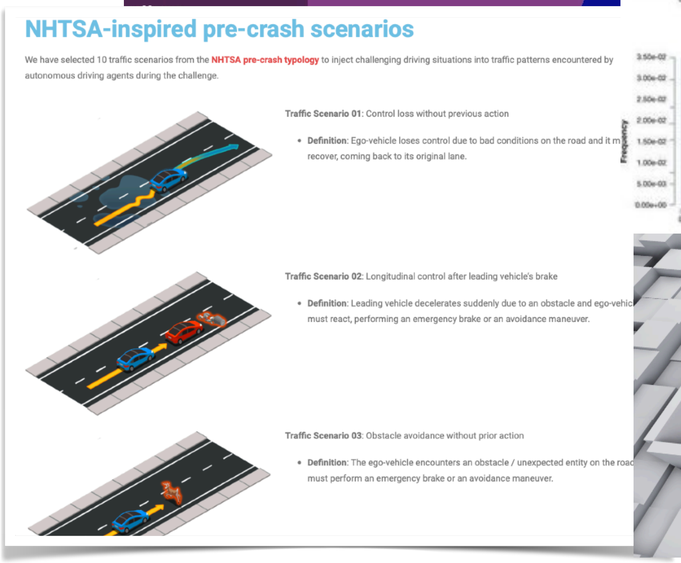
L. H. Gilpin, V. Penubarthi and L. Kagal, "Explaining Multimodal Errors in Autonomous Vehicles," *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, 2021, pp. 1-10, doi: 10.1109/DSAA53316.2021.9564178.

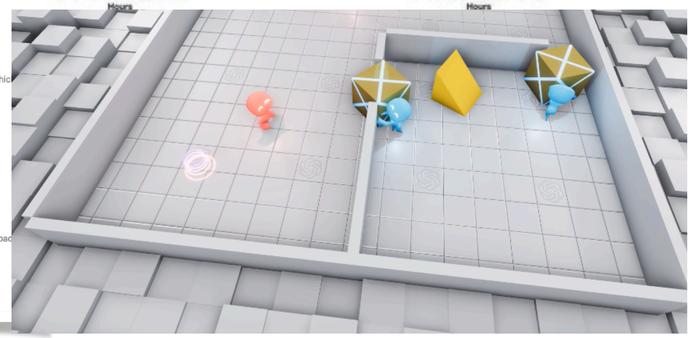
Evaluation of Error Detection is Difficult

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.

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.

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

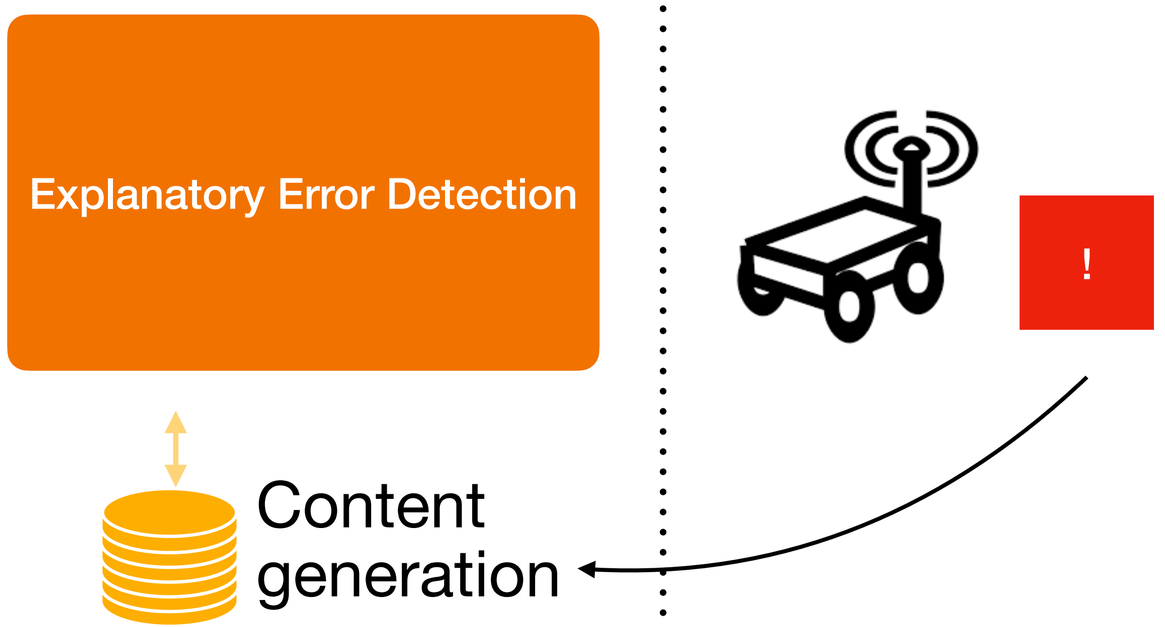
Anticipatory Thinking Layer for Error Detection

“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.

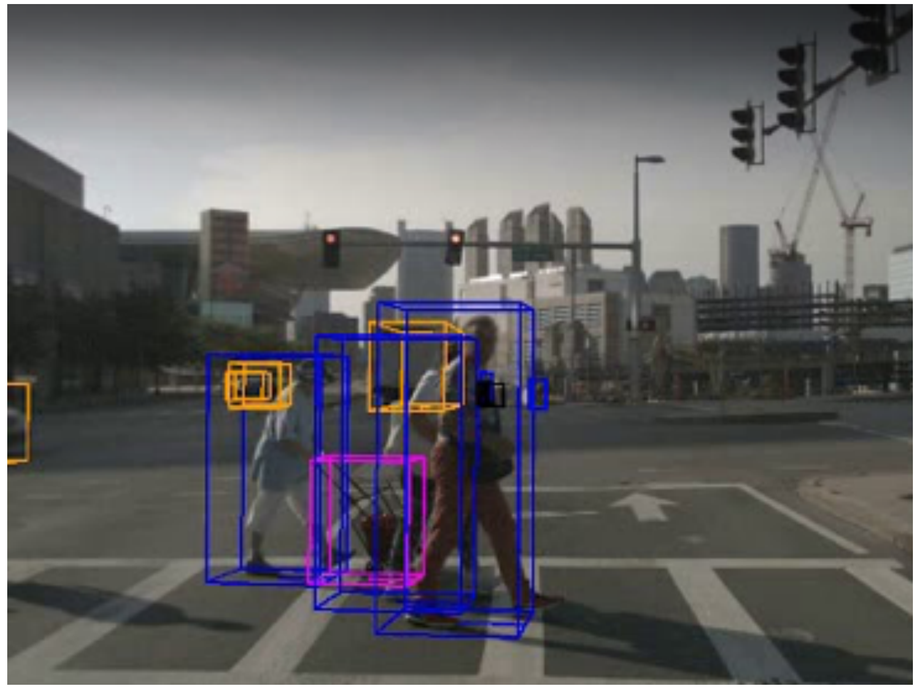
Testing Framework in Two Parts

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.



Lack of Data and Challenges for AVs

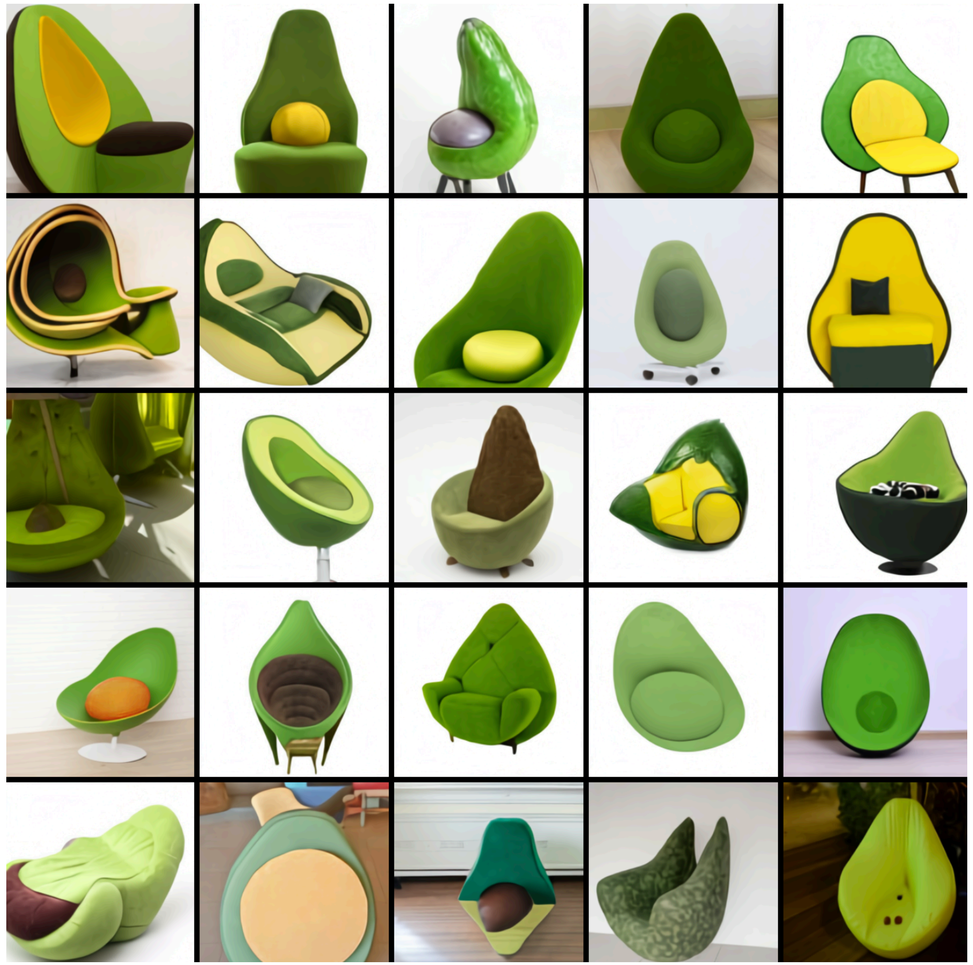
- Existing Challenges
  - Targeted as optimizing a mission or trajectory and not safety.
  - Data is hand-curated.
- Failure data is not available
  - Unethical to get it (cannot just drive into bad situations).
  - Want the data to be realistic (usually difficult in simulation).



Data from NuScenes

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.

Generate “dangerous driving.”

“Realistic”

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.

Generate “dangerous driving.”

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.

Generate “dangerous driving.”

S. Xu, L. Mi and L.H. Gilpin. “A Framework for Generating Dangerous Scenes for Testing Robustness.” Under Review. 2022.

Approach: Content Generation

Anticipatory Thinking Layer for Error Detection

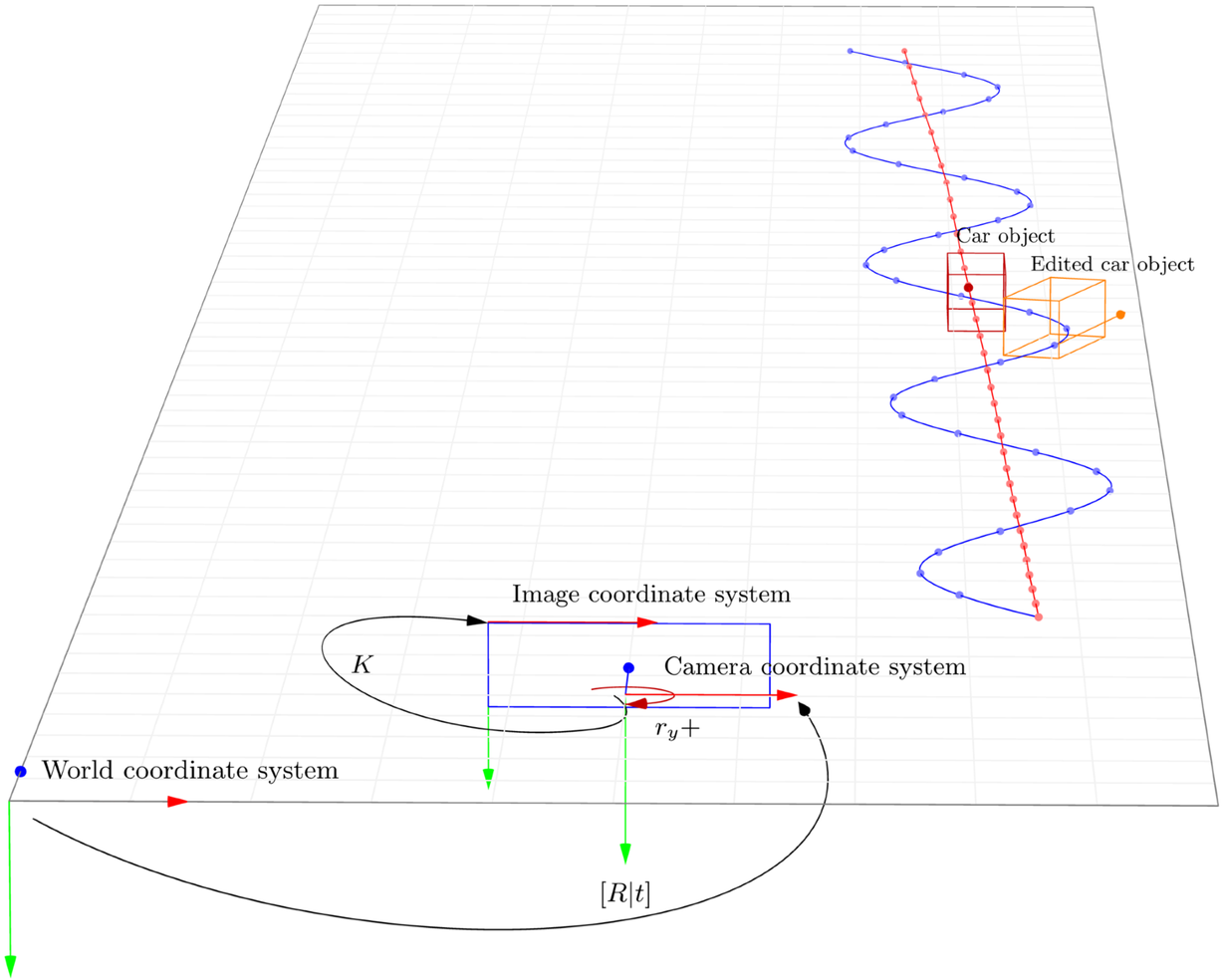
+

Generate images with shadows before tunnels.

Generate images with fallen signs.

Generate images with trucks carrying traffic lights.

Generate “dangerous driving.”

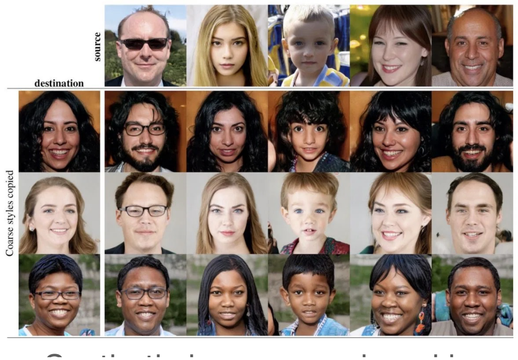


S. Xu, L. Mi and L.H. Gilpin. “A Framework for Generating Dangerous Scenes for Testing Robustness.” Under Review. 2022.

Approach: Content Generation

Anticipatory Thinking Layer for Error Detection

+



!

Shadows

Fallen signs

Generate images with shadows before tunnels.

Generate images with fallen signs.

Generate images with trucks carrying traffic lights.

Generate “dangerous driving.”

Need for Context and Explanation

“Realistic” Adversarial

en a driveway — UsedFor → en a truck
Weight: 2.83

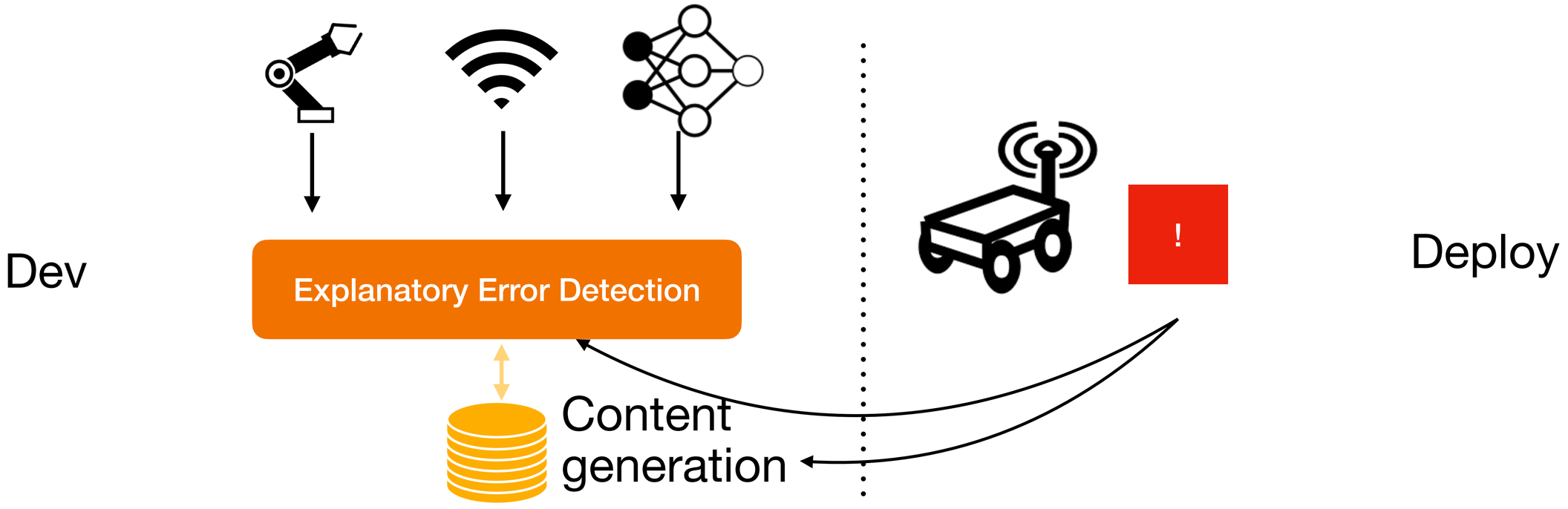
en A truck — UsedFor → en hauling things
Weight: 1.0

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.



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.

.

Contributions

The problem: Autonomous Vehicles are Prone to Failure.

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

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.