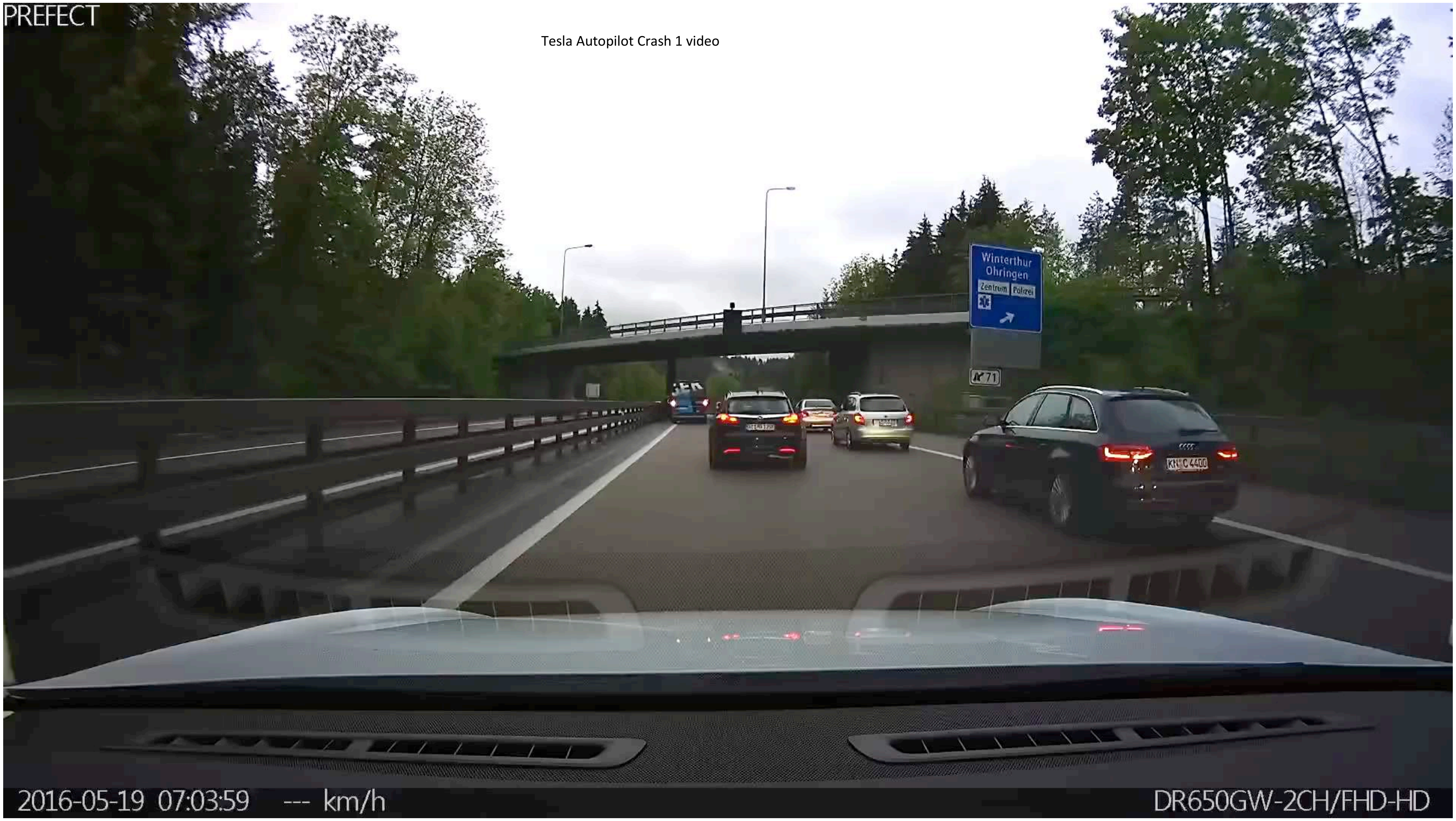
Challenge Problems in
Cyber Physical Systems and Industrial IoT

Prof. Rahul Mangharam
Director, Real-Time & Embedded Systems Lab
Dept. Electrical & Systems Engineering
Dept. Computer & Information Science
University of Pennsylvania
rahulm@seas.upenn.edu

What would you like for your Birthday?

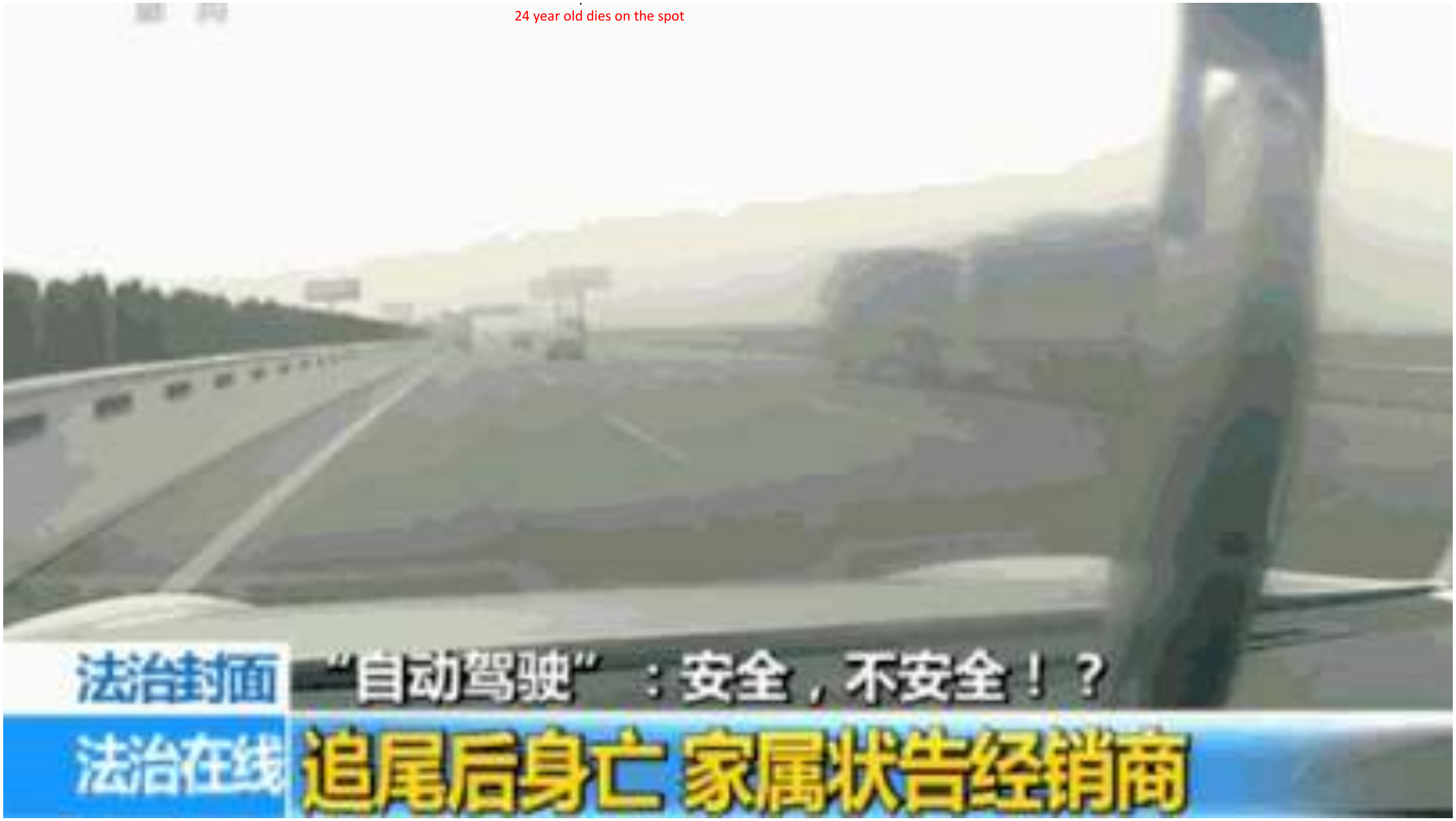
A Tesla with Autopilot?

Tesla Autopilot Crash 1 video



Tesla Autopilot Crash 2 video

24 year old dies on the spot



Mobility21

DoT National University Transportation Center [2017-2021]

Penn Engineering

Carnegie Mellon University

Technologies for Safe and Efficient Transportation

THE NATIONAL USDOT UNIVERSITY TRANSPORTATION CENTER FOR SAFETY

UNIVERSITY of PENNSYLVANIA

Carnegie Mellon University

A Driver's License Test for Autonomous Vehicles

Prof. Rahul Mangharam
Penn Director, Mobility21 DoT UTC
University of Pennsylvania
rahulm@seas.upenn.edu

What this talk is about?

1. Defining Safe Autonomous Systems
2. The Insurance Problem
3. The Guardian Angel Problem
4. Connected Autonomous Vehicles

Defining Safety: A Driver’s License Test for Autonomous Vehicles

Pennsylvania
visitPA.com
DRIVER'S LICENSE

No: 99 000 423 Dups: 01
DOB: 08/28/1960 Sex: M
Class: AI Eyes: IR
Endorse: Height: 6'00"
Com/Med Rstr: B
Issued: 04/26/2012
Expires: 04/26/2016

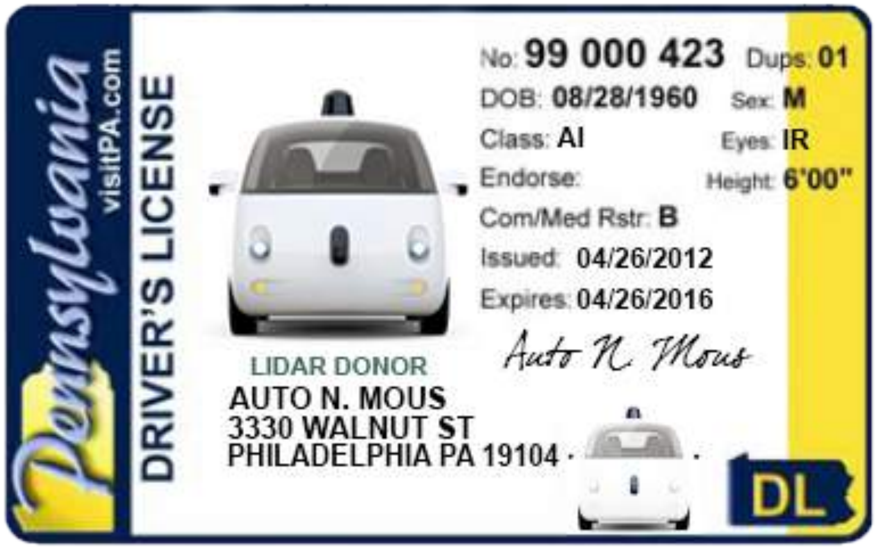
Auto N. Mous

LIDAR DONOR
AUTO N. MOUS
3330 WALNUT ST
PHILADELPHIA PA 19104

DL

- Under what *criteria* can we determine that an *autonomous vehicle is* ***safe***?
- How can we **verify** its actions ***beyond simple tests***?

Defining Safety: A Driver's License for Autonomous Vehicles



Pennsylvania
visitPA.com
DRIVER'S LICENSE

No: 99 000 423 Dups: 01
DOB: 08/28/1960 Sex: M
Class: AI Eyes: IR
Endorse: Height: 6'00"
Com/Med Rstr: B
Issued: 04/26/2012
Expires: 04/26/2016

Auto N. Mous

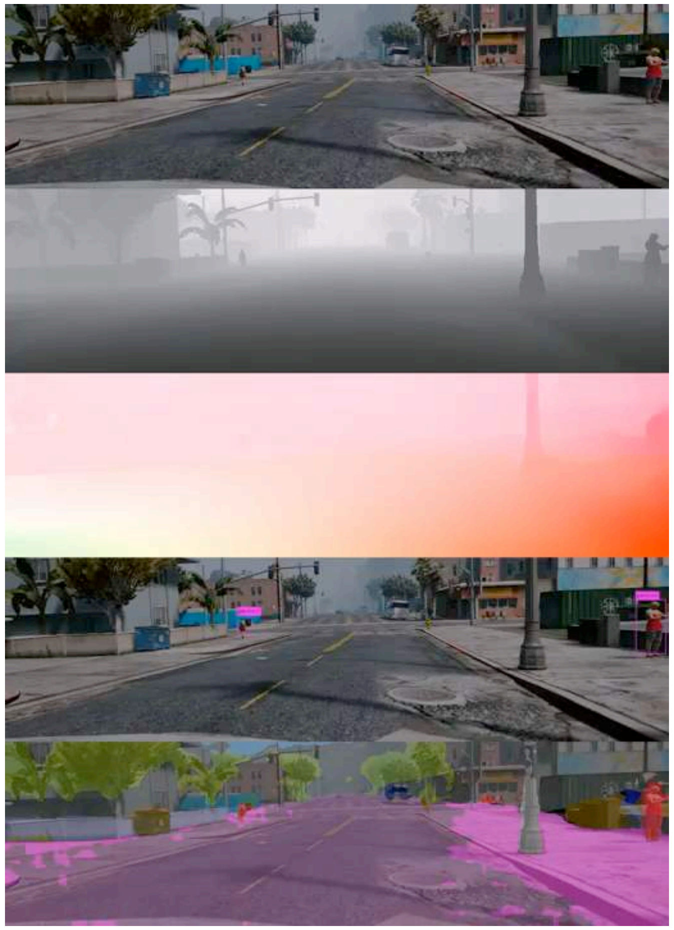
LIDAR DONOR
AUTO N. MOUS
3330 WALNUT ST
PHILADELPHIA PA 19104

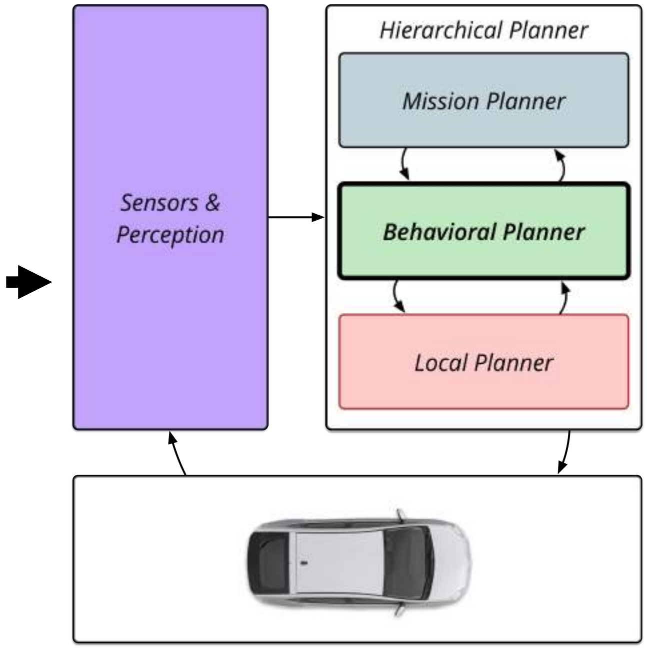
DL

- So what would the Autonomous Driver's license consist of?
  - *Automatically verified models of control and decision algorithms*
  - *For representative scenarios*
  - *With quantitative statistics regarding the state of the ego and environment*
  - *On a variety of roads*

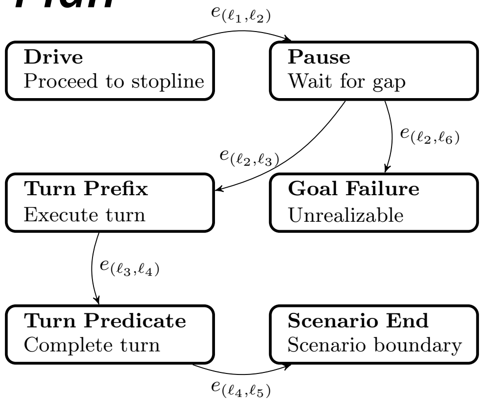
AV Perception, Planning, Control pipeline

Sense

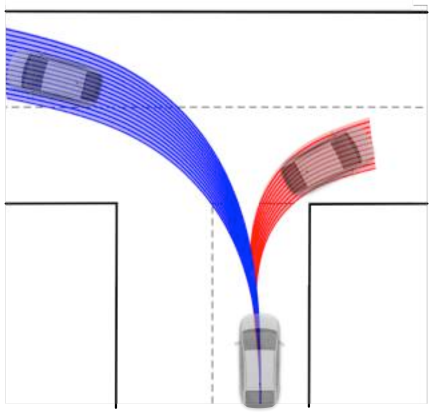




Plan



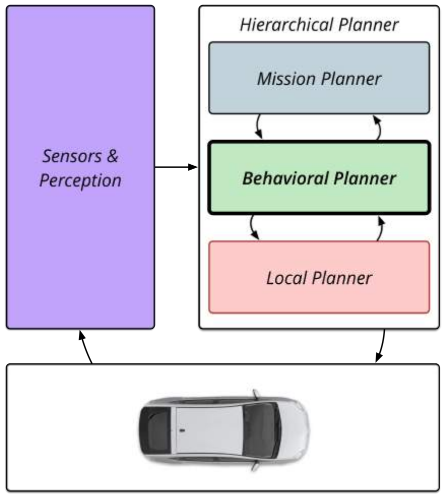
Act

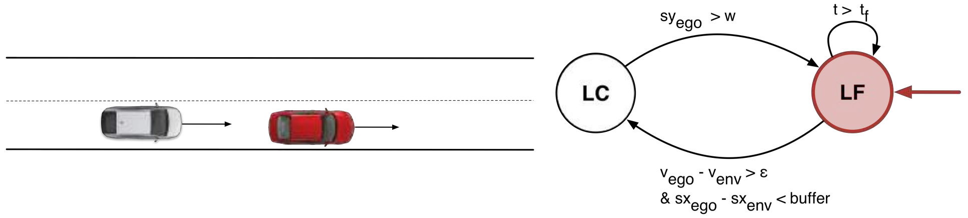


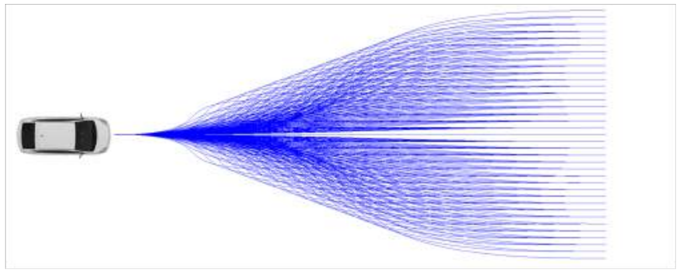
Everything combined:
a function that *generates* a sequence of ***steering*** and ***acceleration*** inputs...

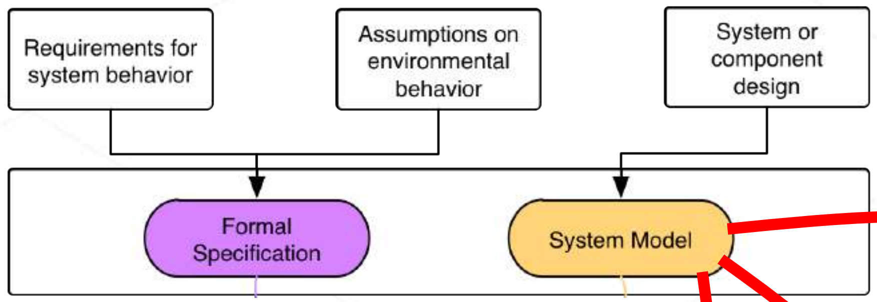
Defining Safety: A Driver's License for Autonomous Vehicles

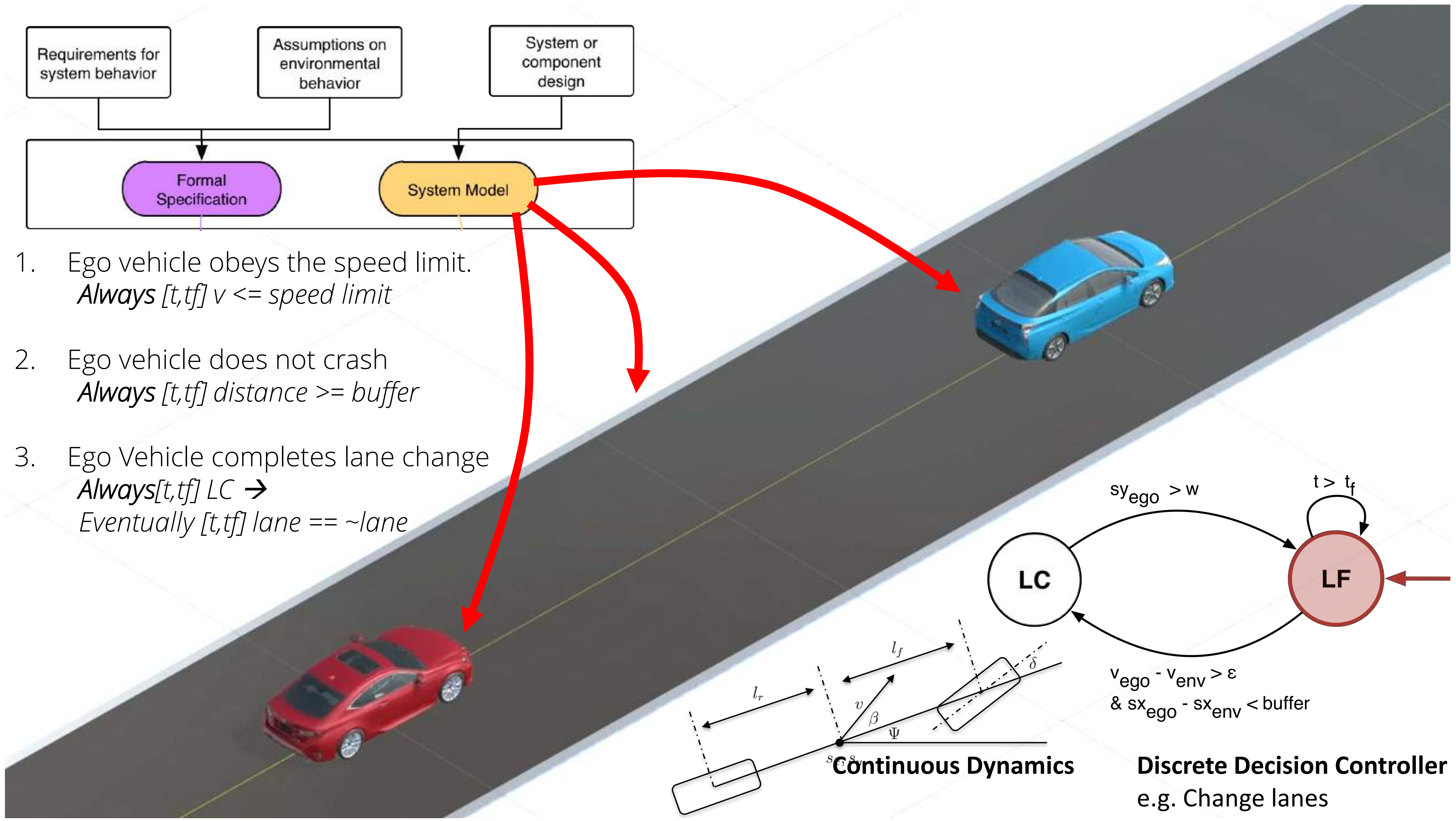
- Why its hard…
  - Automatically verified models of control and decision algorithms: ***non-linear vehicle dynamics and mode switching lead to intractable or undecidable decision problems.***




$sy_{ego} > w$
$t > t_f$
LC
LF
$v_{ego} - v_{env} > \varepsilon$ & $sx_{ego} - sx_{env} <$ buffer



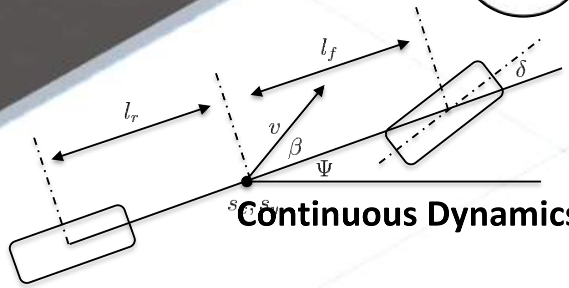

Requirements for system behavior
Assumptions on environmental behavior
System or component design
Formal Specification
System Model



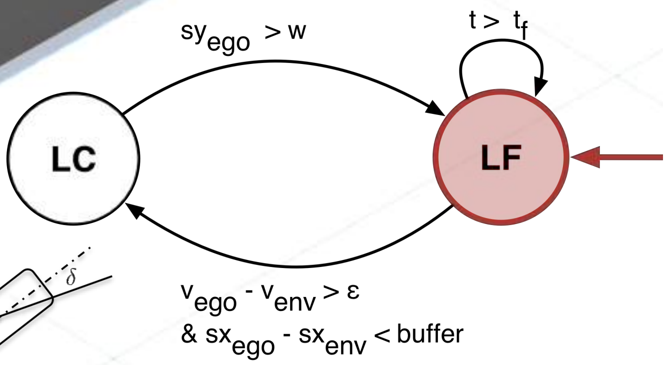
1. Ego vehicle obeys the speed limit.
   Always [t,tf] v <= speed limit

2. Ego vehicle does not crash
   Always [t,tf] distance >= buffer

3. Ego Vehicle completes lane change
   Always[t,tf] LC →
   Eventually [t,tf] lane == ~lane



Continuous Dynamics


$sy_{ego} > w$
$t > t_f$
LC
LF
$v_{ego} - v_{env} > \varepsilon$
$\& \; sx_{ego} - sx_{env} < \text{buffer}$

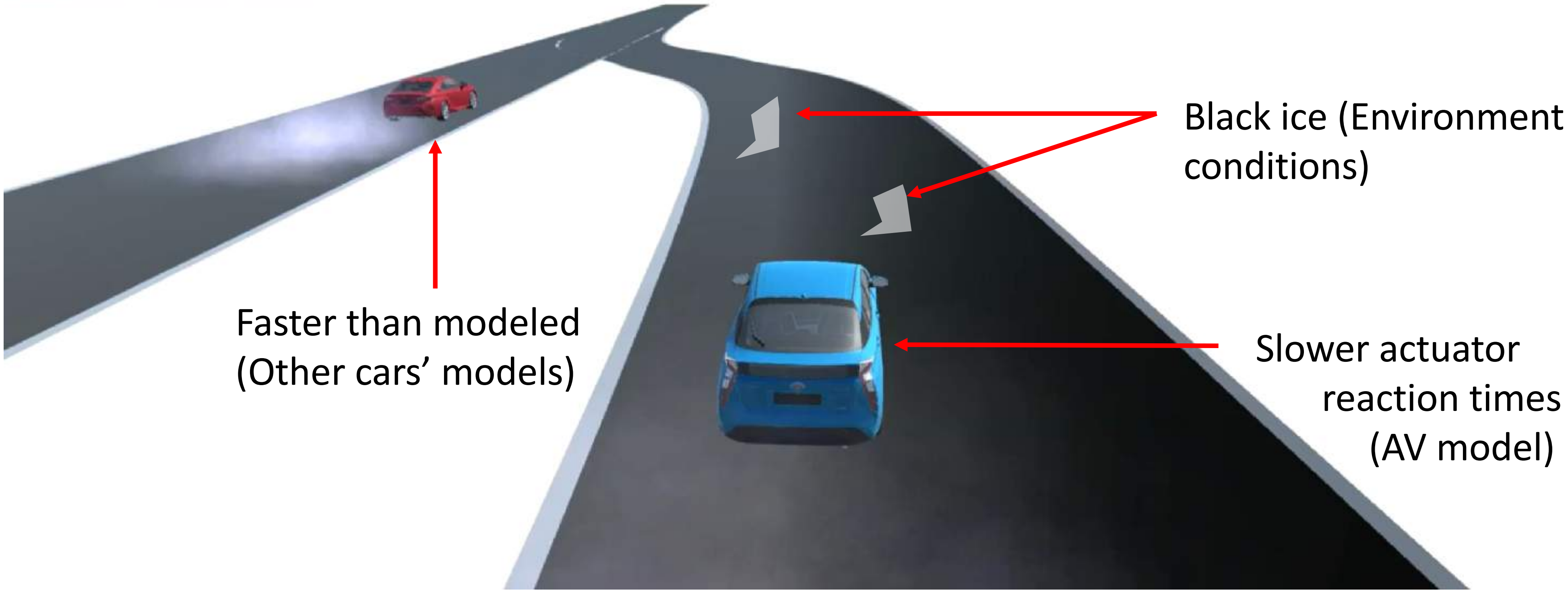
Discrete Decision Controller
e.g. Change lanes

Requirement:

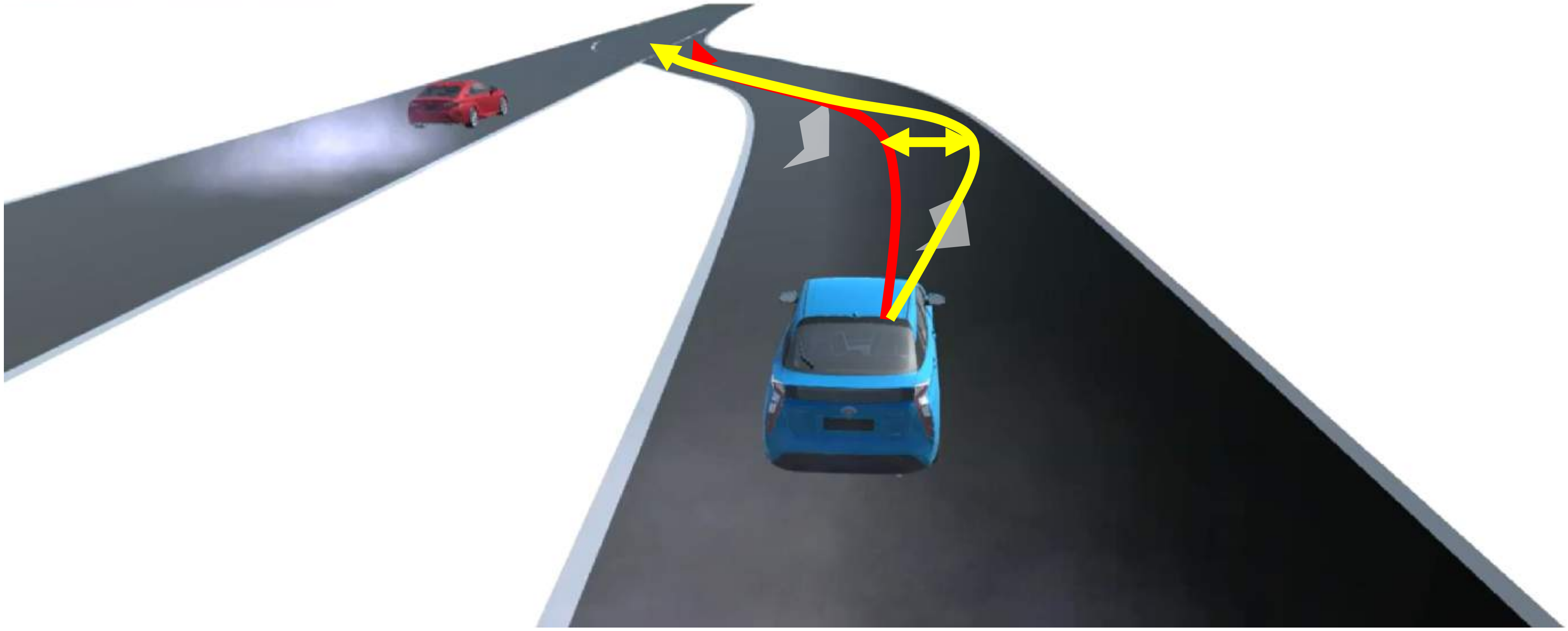
1. If car slips, recover within 3 seconds.
2. Stay within lane markers.
3. Maintain minimum distance to others.

Model is verified to satisfy this Requirement

A deployed system ALWAYS deviates from its model, and the environment ALWAYS deviates from what we expect.
Successive refinements of a model deviate from each other.


Black ice (Environment conditions)
Faster than modeled (Other cars' models)
Slower actuator reaction times (AV model)

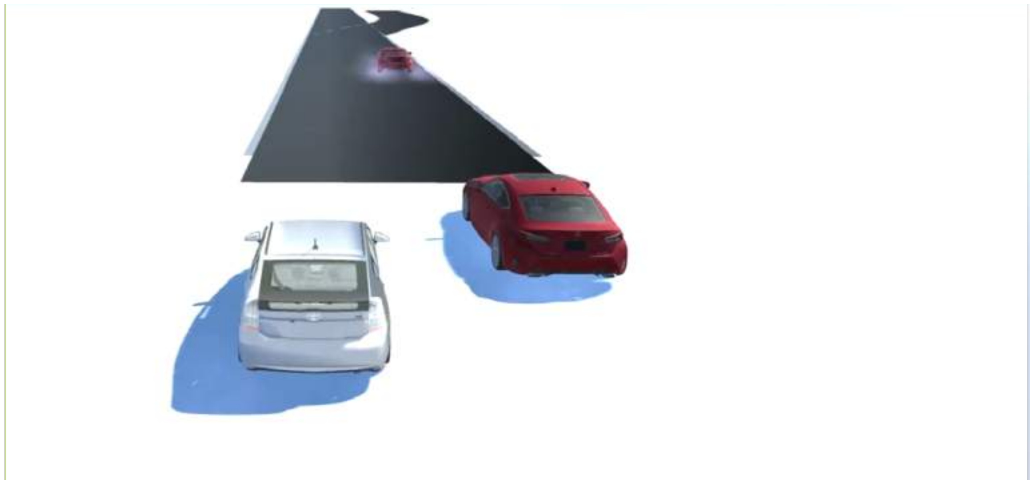
A deployed system ALWAYS deviates from its model, and the environment ALWAYS deviates from what we expect.
Successive refinements of a model deviate from each other.
Does the system have an “error margin” to tolerate unforeseen disturbances and errors?

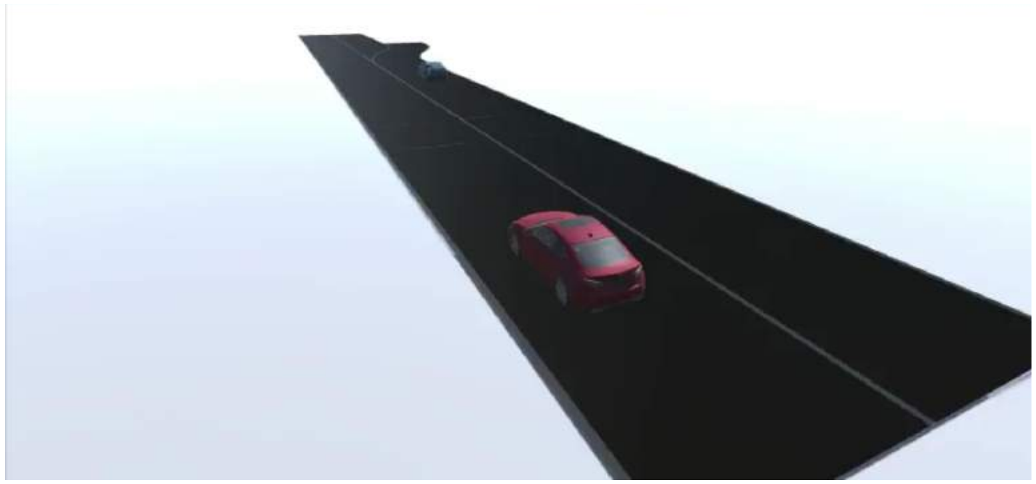


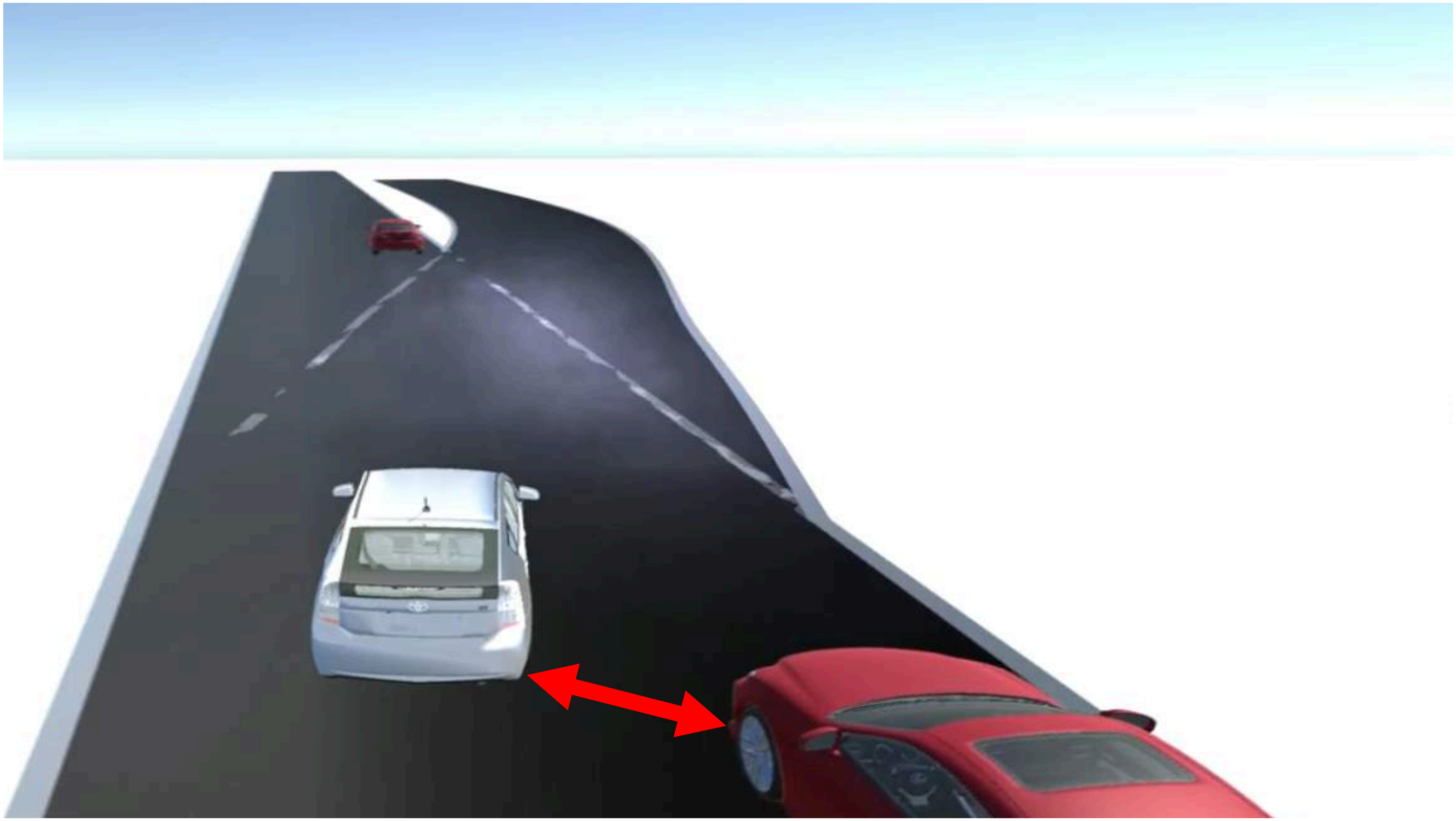
Safe

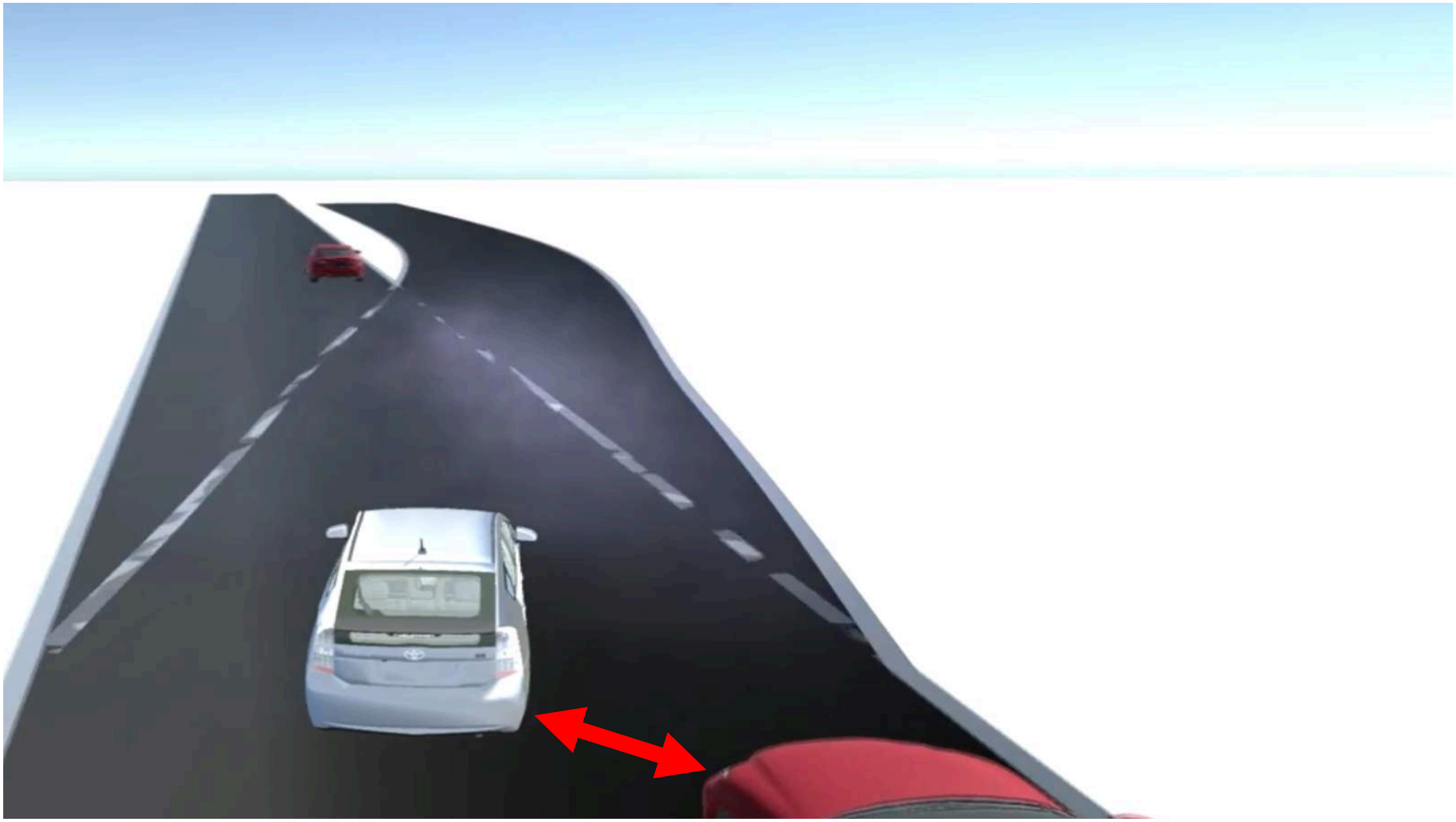
Unsafe

How can we automatically find diverse unsafe cases?
What about the marginal cases?







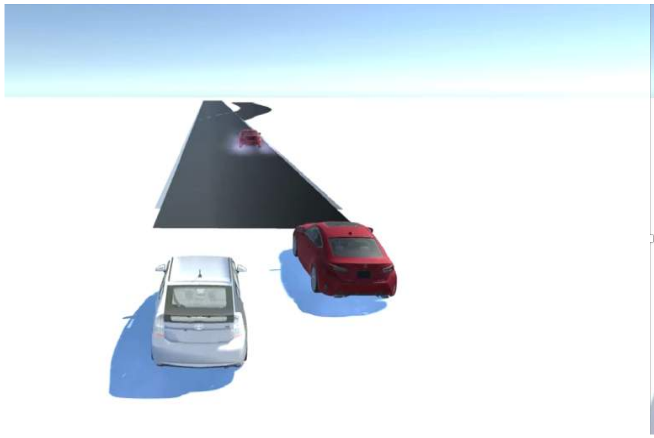
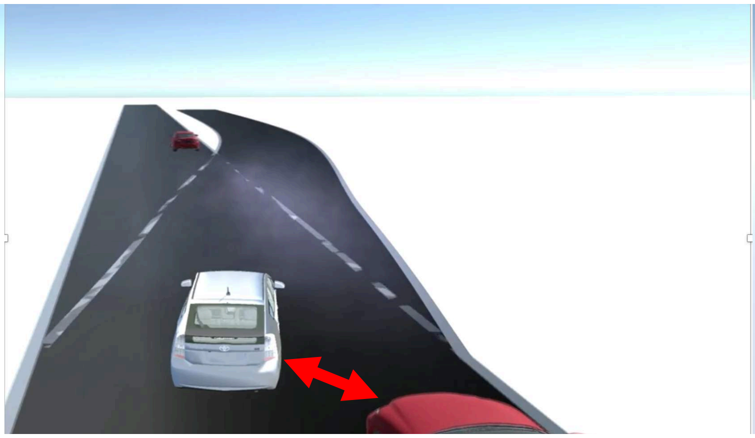
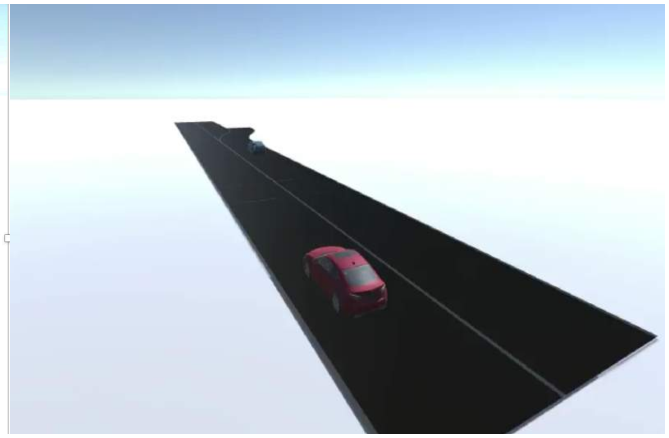


Safe

Unsafe

What about the marginal cases?

We should be interested in the *spectrum* of vehicle safety

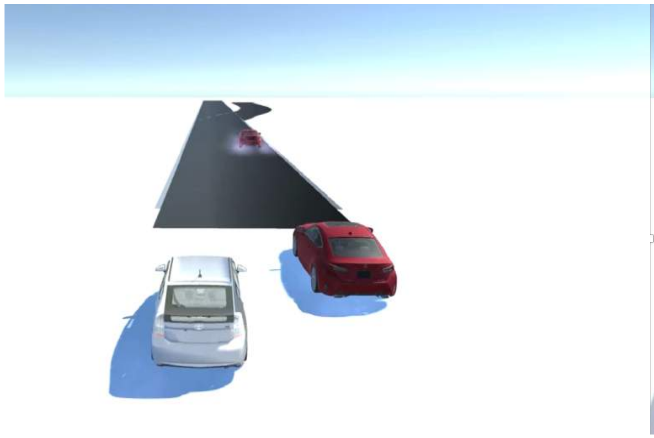




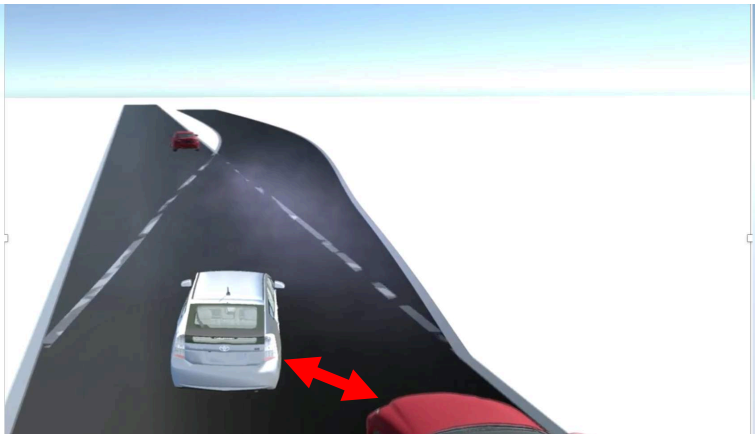
Safe

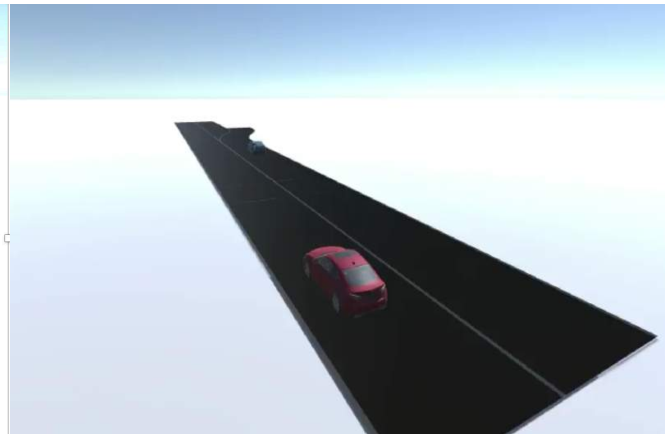
Unsafe

What about the marginal cases?

We should be interested in a *continuous measure* of vehicle safety







Defining Safety: A *continuous* measure of vehicle safety

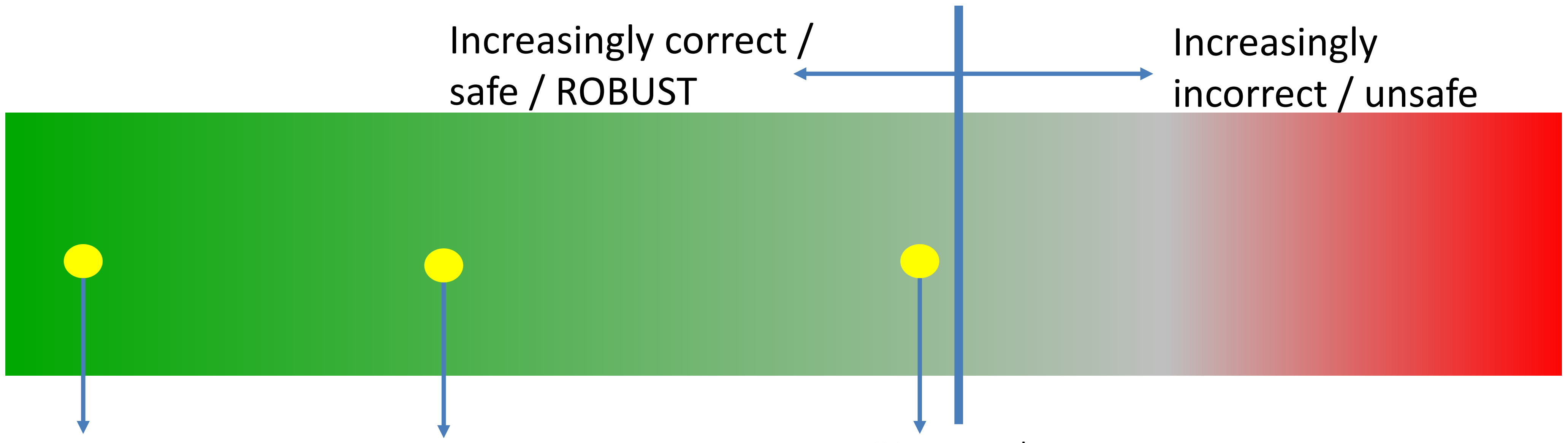
From:

*Does the Autonomous Vehicle **satisfy** the design requirements?*

To:

*How **robustly** (how well) does it satisfy the requirements?*

Safety (and more general correctness) as a continuous measure
Robustness



Increasingly correct / safe / ROBUST
Increasingly incorrect / unsafe

AV can tolerate large disturbances (e.g., wheel slippage, delayed actuation, aggressive drivers, etc)

AV can tolerate medium disturbances (e.g., wheel slippage, moderately delayed actuation)

AV **cannot** tolerate disturbances. *Small* deviations between model and system invalidate verification results

Robustness Guided Verification

Use ***robust simulations*** to guide and accelerate almost-exhaustive verification, so it can tackle these hard problems in autonomy

A primer on the robustness of Metric Temporal Logic formulas

and its use in Autonomous System falsification

Scenario Description Language: Operating Environment

The operating environment, or *road agent*, is defined by:

- *Static parameters* such as **geometry.**
- *External parameters* such as the **time of day** for the scenario.

Variations enter the search as *unknown parameters* **selected from a set**.

For example:

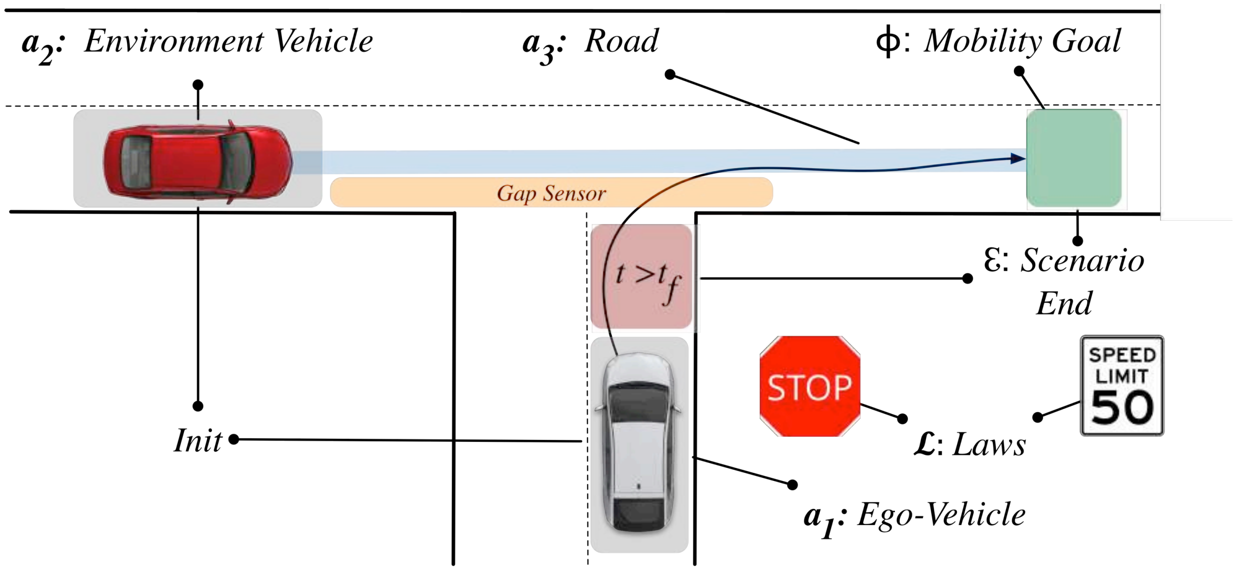
- Choose *road geometry* from a *discrete set of models*, ***{Location 1, Location 2}***.
- Choose *time* from a *continuous set* in R, ***[0,24]***.

Scenario Description Language: Other Traffic Participants

A traffic participant instance is defined by *static parameters* such as its **location** and **velocity** of other vehicles**.** Behaviors are influenced by *external parameters* such as the **goal** of traffic agent. **For example:**

- Choose ***location*** from a *continuous set* in R^2*: **x** in **[-25,-5]** and **y** in **[10, 12]***.
- Choose ***velocity*** from a *continuous set* in R*: **v** in **[0,30]***.
- Choose ***goal*** from a discrete set of actions: ***{Straight, Left Turn}***.

Scenario Description Language: Ego-Vehicle Initialization



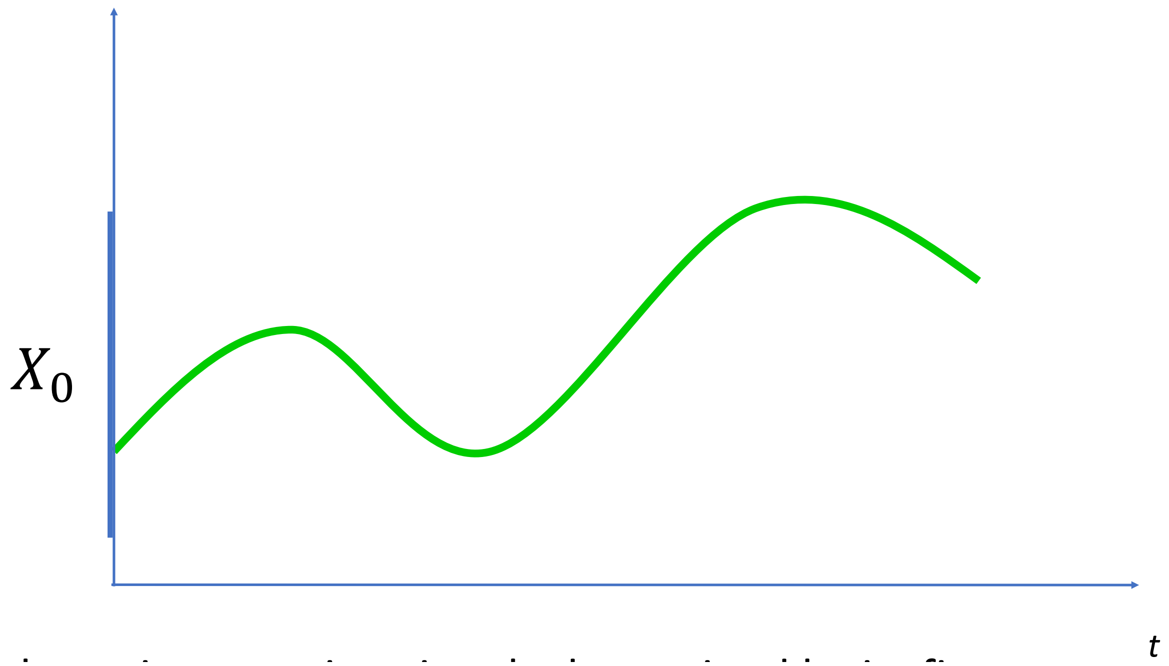
The ego-vehicle instance is defined by *static parameters* such as its **location** and **velocity** and **goal.** Additional parameters exist within its controllers. **For example:**

- Choose ***location*** from a *continuous set* in R^2*: **x** in **[5,10]** and **y** in **[-20, -5]***.
- Choose ***velocity*** from a *continuous set* in R*: **v** in **[0,30]***.
- Choose ***goal*** from a from a *continuous set* in R^2*: **x** in **[20,40]** and **y** in **[0, 5]***.

Ingredient 1: system simulation

A system has a set of initial conditions X_0.

From every initial state x(0) in X_0, it produces a state trajectory x(t).



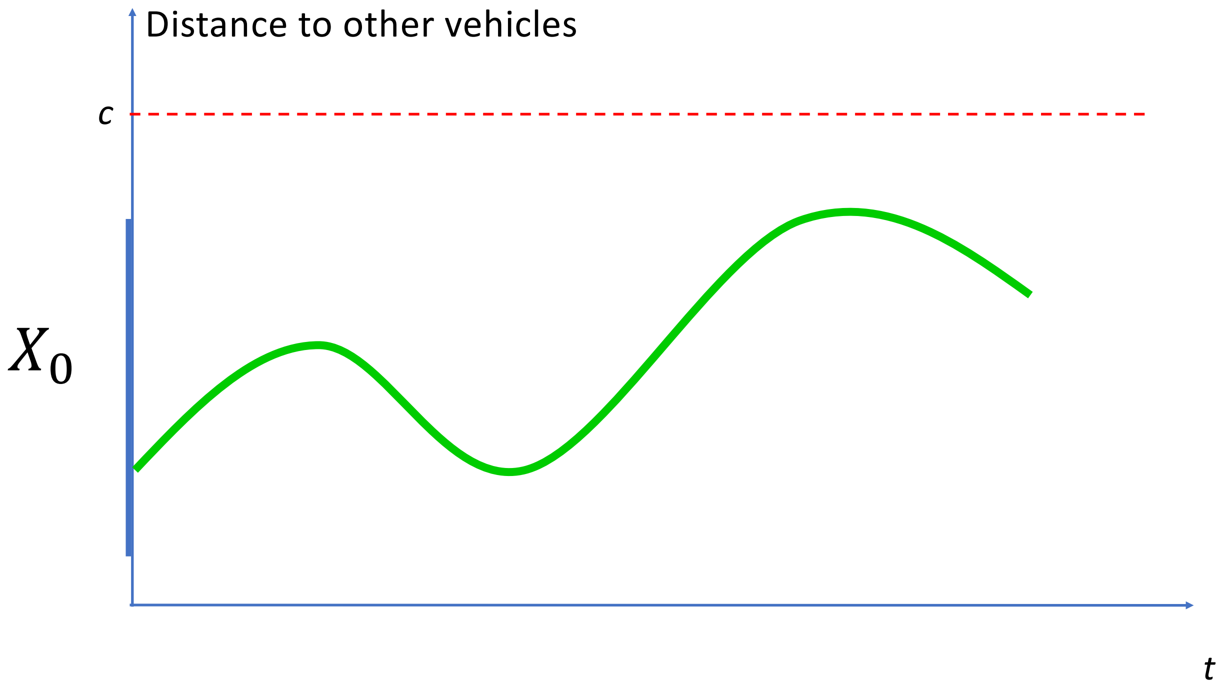
For a deterministic system, the trajectory x is uniquely determined by its first state x(0). That’s why robustness of a given formula is a function of the initial state.

Ingredient 2: a specification

The simplest specification: safety

Green trajectory, obtained by simulation, satisfies the **safety** spec:
Distance to other vehicles must Always be < c.

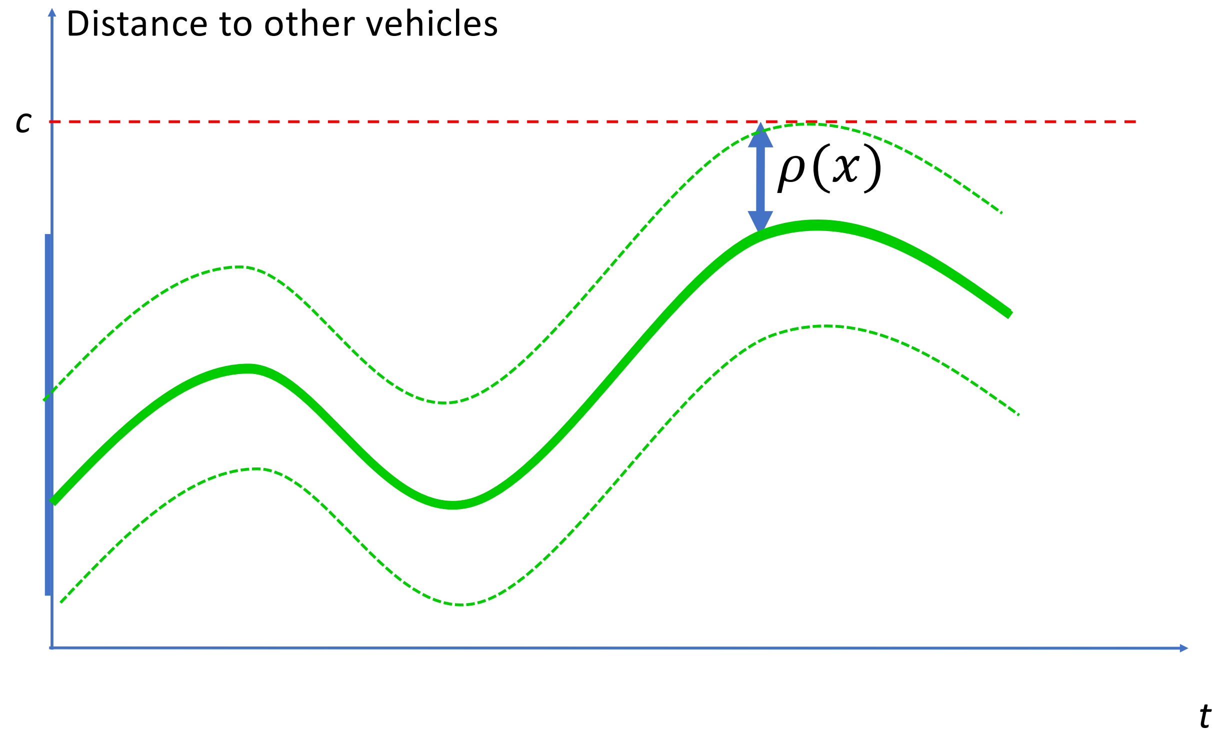
(could also be velocity < threshold, acceleration < threhsold, etc)



A Primer on Robust Simulation

The **robustness** of the trajectory x, in this special case, is defined to be the minimum distance between the trajectory and the red lines.

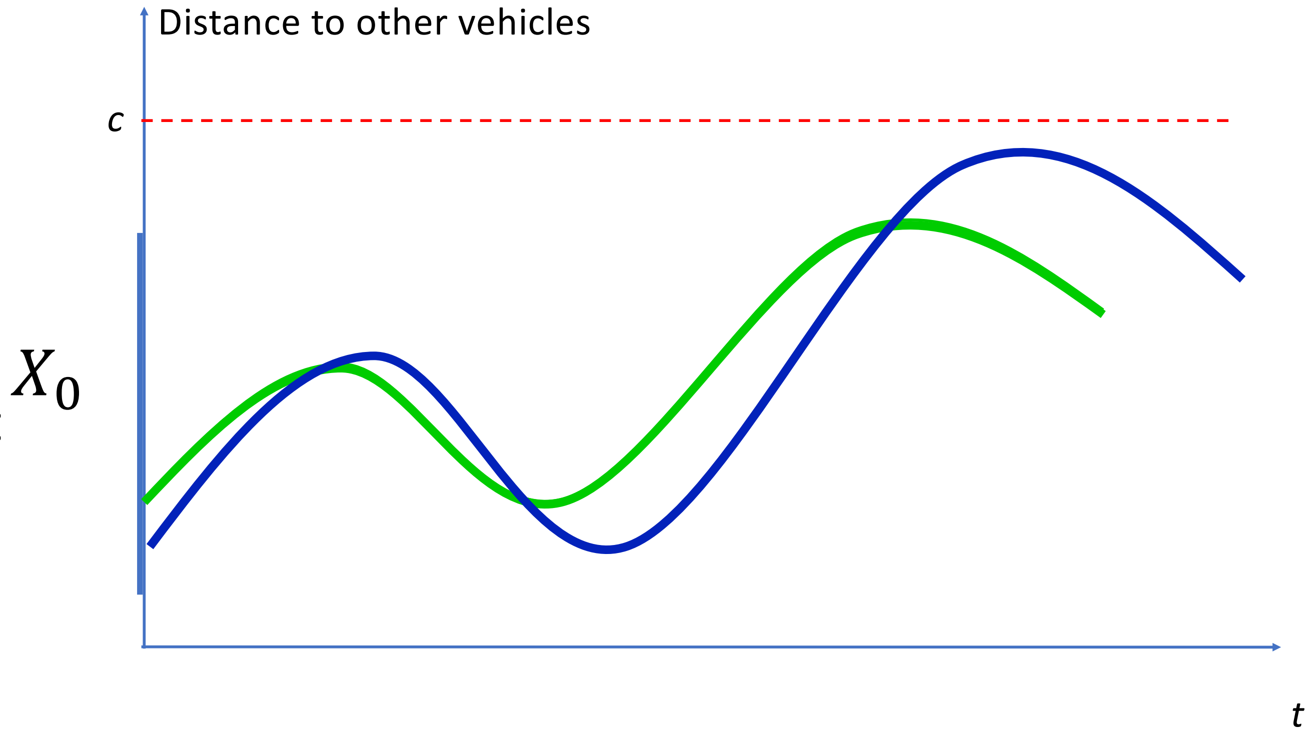
$$\rho(x) \coloneqq \min_{t} |x(t) - c|$$



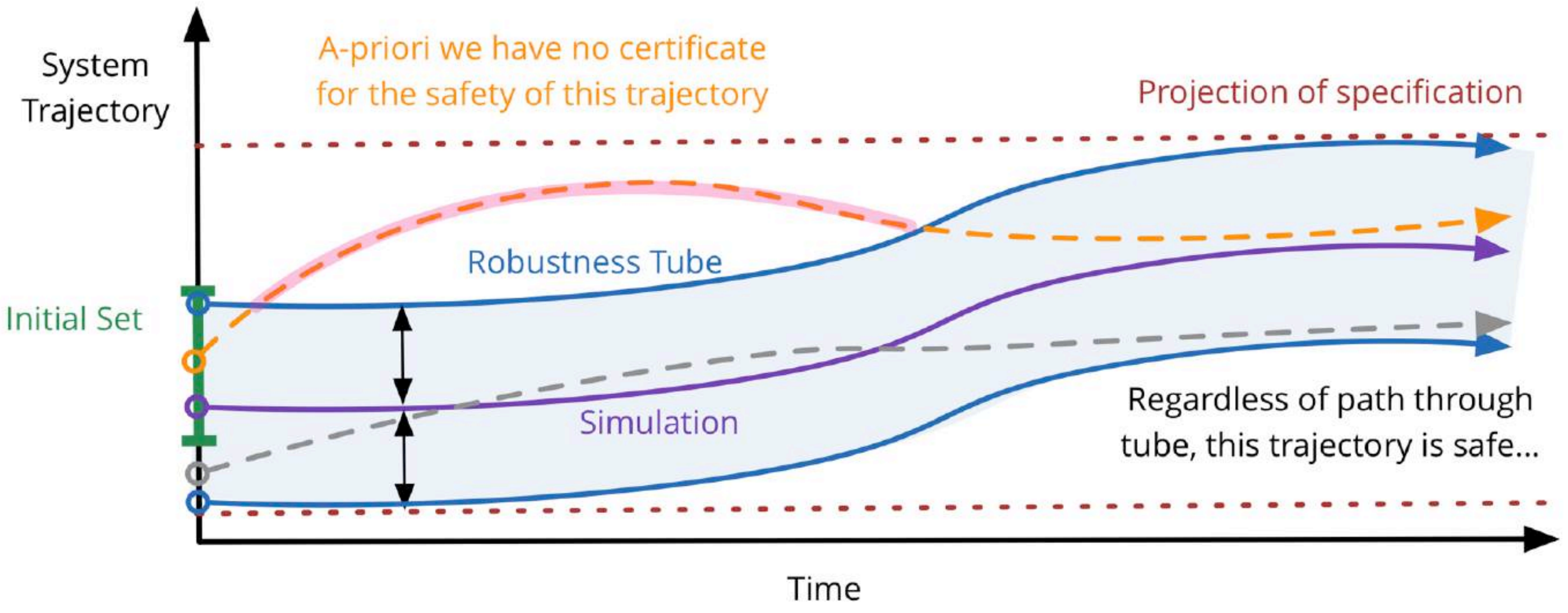
A Primer on Robust Simulation

The blue trajectory still satisfies the spec.

But in a sense, it is less robust than the green trajectory: it gets closer t violating the spec.

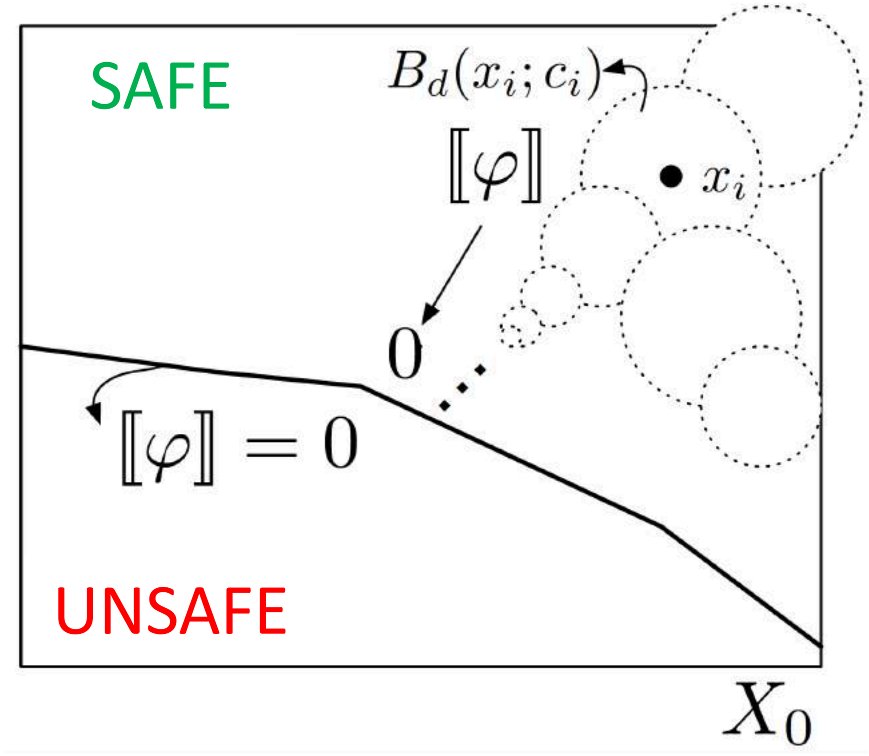


Properties: Robustness

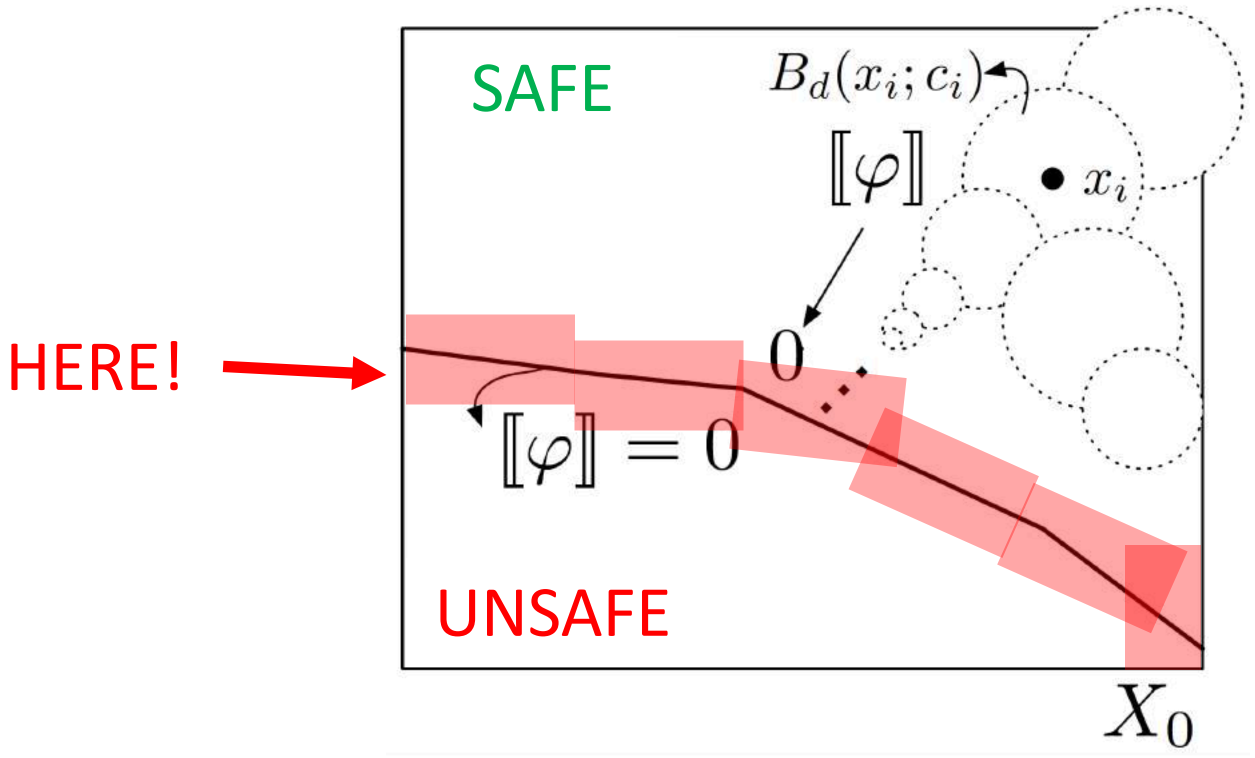

System Trajectory
A-priori we have no certificate for the safety of this trajectory
Projection of specification
Robustness Tube
Initial Set
Simulation
Regardless of path through tube, this trajectory is safe...
Time

However, any signal that ever leaves the robustness tube *may be* unsafe.

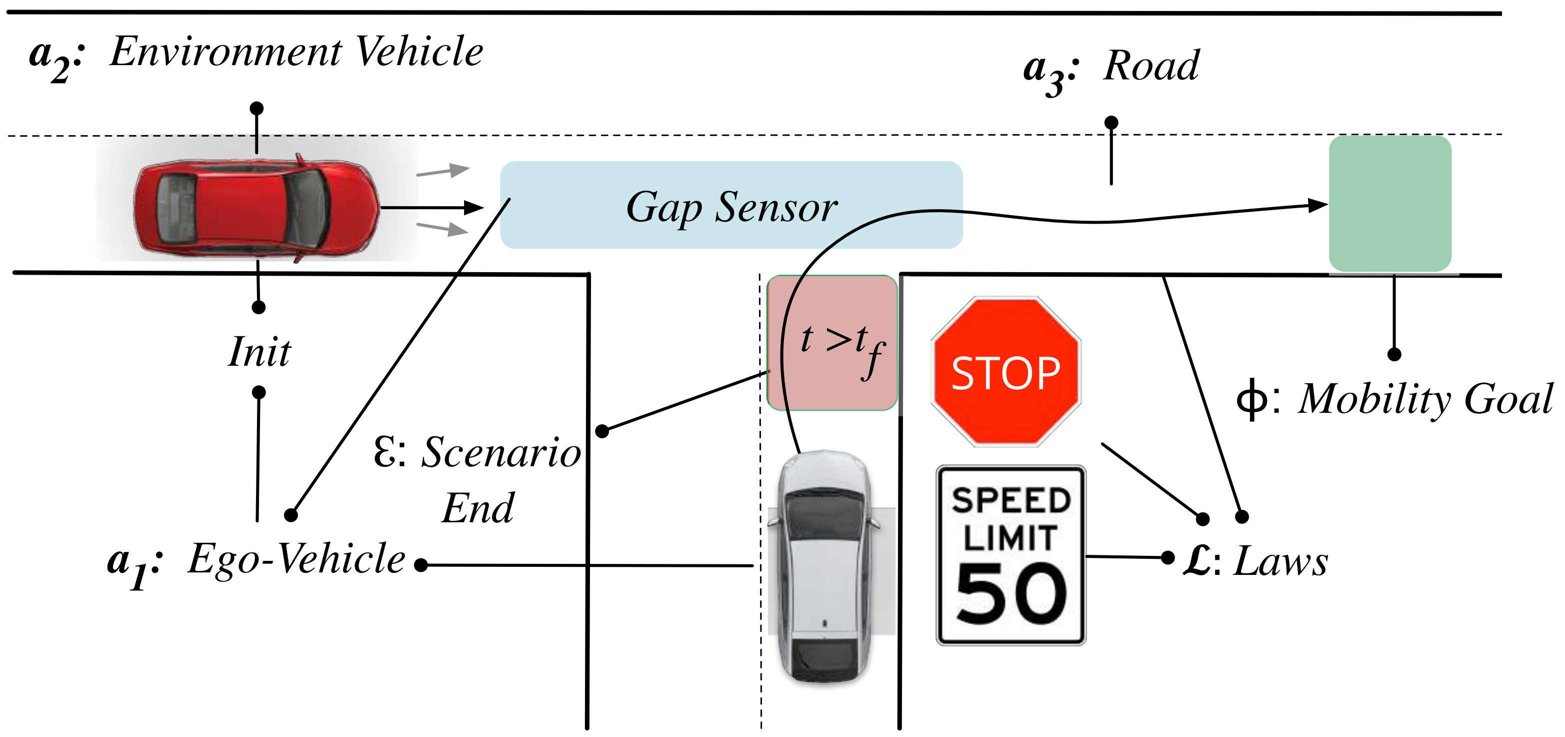
Where should we spend the verification effort?



Where should we spend the verification effort?



T-Junction Robustness Landscape

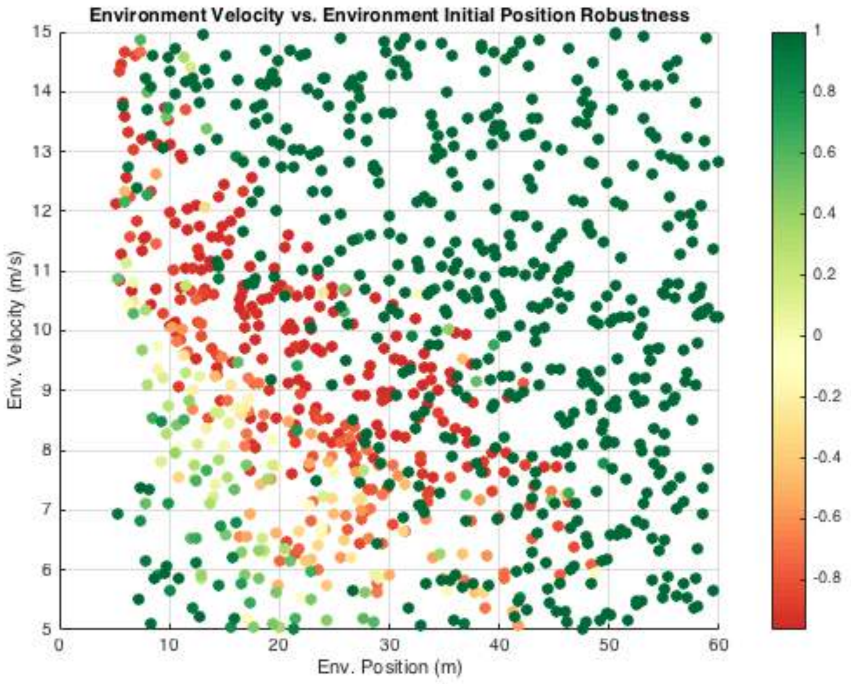


Practical Limitations: Testing

When can we draw high-confidence conclusions about *whole system behavior* from a finite number of tests?

Example:
Every point is a sample execution of system. Green = good, red = bad

Note how green and red mix, which requires a lot of samples in that area to draw high-confidence conclusions

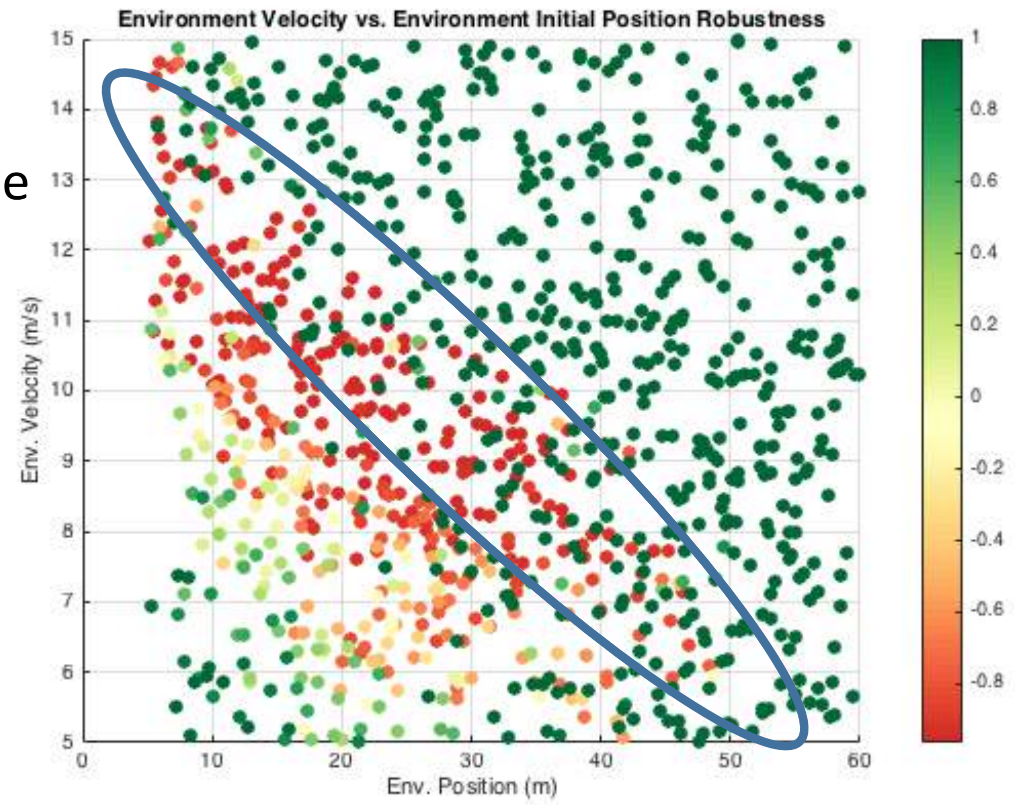


With testing you try to make a conclusion about the entire system from these samples.
What if the bad behavior is hiding between good behavior, and you never or rarely sample it?

Robustness-Guided Verification

Robustness-guided falsification leads us to the low-robustness ellipsoid.

Near-exhaustive verification decisively verifies this smaller behavior.



How to define robustness for more complex MTL specifications?

A more general mission requirement:

Prepare to exit highway through right lane in T seconds

[Refine] Sometime in the next T seconds, Position = right lane.

[Refine] Sometime in the next T seconds, (steering angle > 15 and acceleration > 0) until until Position = right lane.

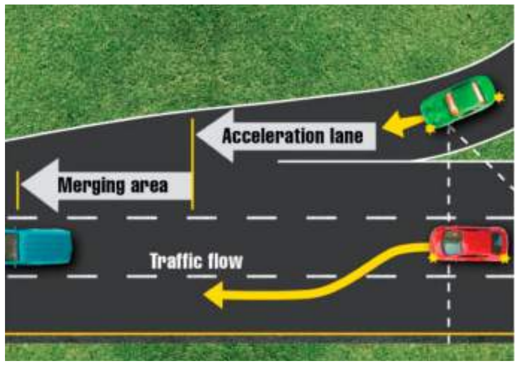
[Refine] Sometime in the next T seconds, (steering angle > 15 and acceleration > 0) until until Position = right lane, UNLESS right lane is occupied

[MTL]

$$\boldsymbol{RightLane.isFree} \rightarrow \boldsymbol{F}_{[0,T]}((\boldsymbol{angle} > \mathbf{15} \wedge \boldsymbol{acc} > \mathbf{0})\boldsymbol{U}(\boldsymbol{pos} = \boldsymbol{right}))$$

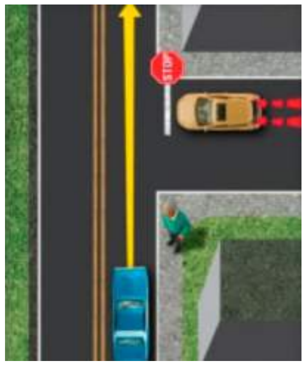
Modeling Framework: Agents operating within **scenarios**

Representative scenarios:

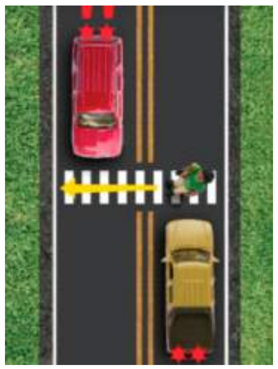


Lane merge

Roundabout

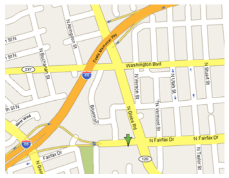


Stop signs


Pedestrians

Understand *common agents* for more intuitive modeling:

Target Vehicle


Road Network

Traffic Laws

Other Vehicles

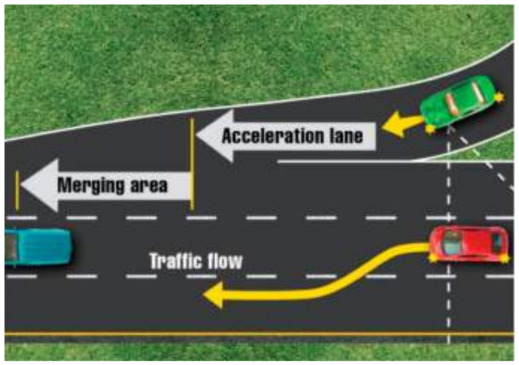
Pedestrians

Infrastructure

New Agents

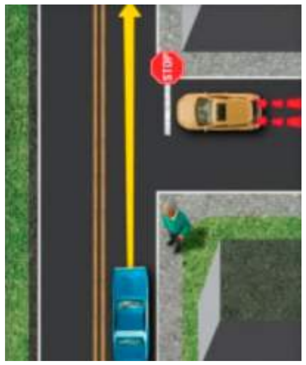
Modeling Framework: Problem Statement

For a *given scenario*, **vehicle model**, & requirements specified over a finite time…

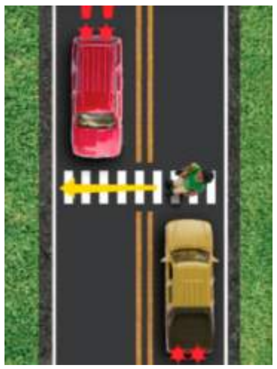


Lane merge

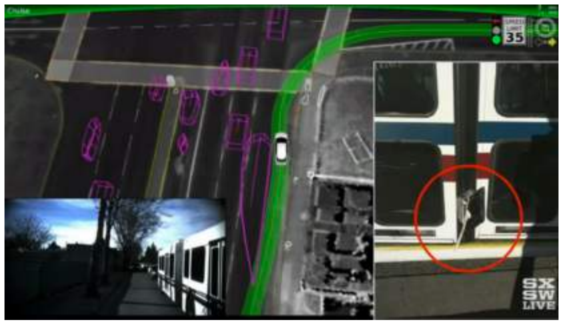
Roundabout



Stop signs


Pedestrians

Does **there exist** an **unsafe execution** of the controller?



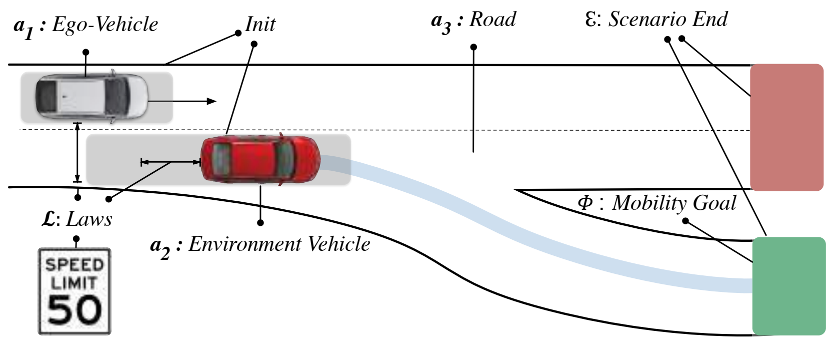
Robustness-Guided Verification: Tool development

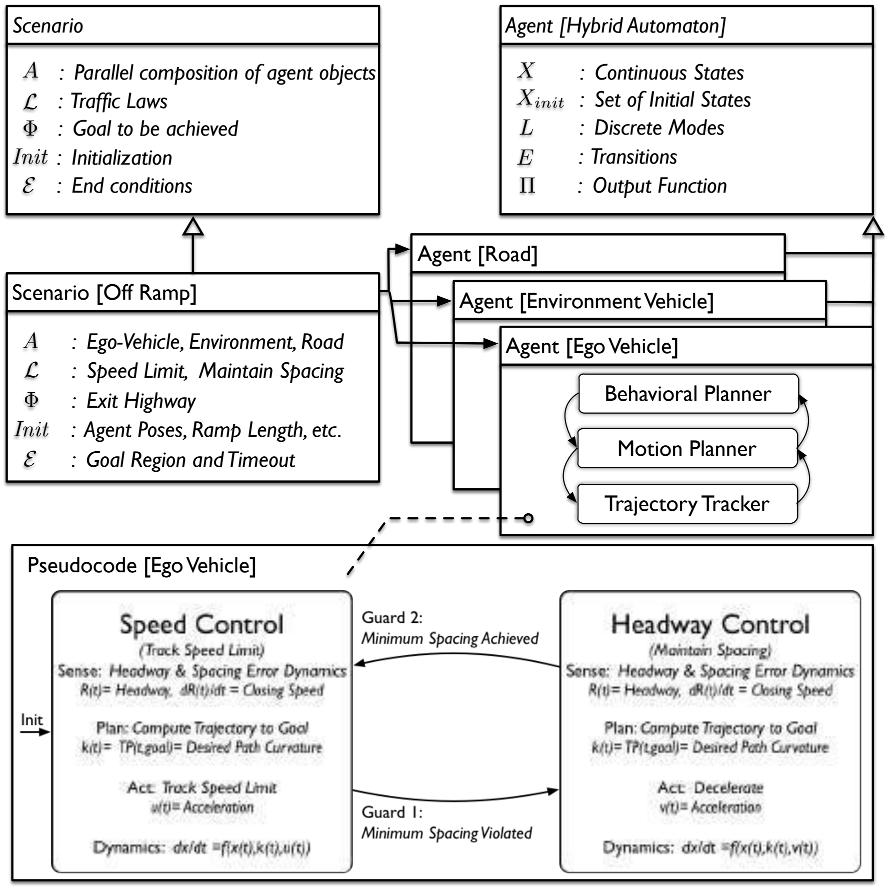
The tool-chain: One Scenario Entry Point

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
...
```

Scenario Definition Language





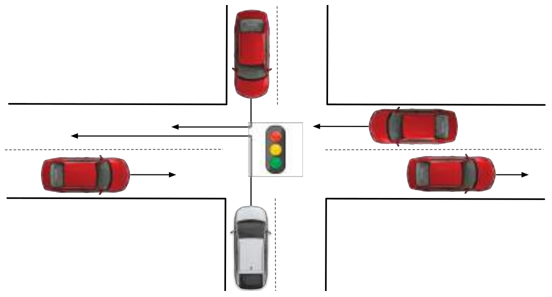
Scenario
A : Parallel composition of agent objects
$\mathcal{L}$: Traffic Laws
Φ : Goal to be achieved
$Init$: Initialization
$\mathcal{E}$: End conditions
Agent [Hybrid Automaton]
X : Continuous States
X_{init} : Set of Initial States
L : Discrete Modes
E : Transitions
Π : Output Function
Scenario [Off Ramp]
A : Ego-Vehicle, Environment, Road
$\mathcal{L}$: Speed Limit, Maintain Spacing
Φ : Exit Highway
$Init$: Agent Poses, Ramp Length, etc.
$\mathcal{E}$: Goal Region and Timeout
Agent [Road]
Agent [Environment Vehicle]
Agent [Ego Vehicle]
Behavioral Planner
Motion Planner
Trajectory Tracker
Pseudocode [Ego Vehicle]
Init
Speed Control
(Track Speed Limit)
Sense: Headway & Spacing Error Dynamics
R(t)= Headway, dR(t)/dt = Closing Speed
Plan: Compute Trajectory to Goal
k(t)= TP(t,goal)= Desired Path Curvature
Act: Track Speed Limit
u(t)= Acceleration
Dynamics: dx/dt =f(x(t),k(t),u(t))
Guard 2:
Minimum Spacing Achieved
Guard 1:
Minimum Spacing Violated
Headway Control
(Maintain Spacing)
Sense: Headway & Spacing Error Dynamics
R(t)= Headway, dR(t)/dt = Closing Speed
Plan: Compute Trajectory to Goal
k(t)= TP(t,goal)= Desired Path Curvature
Act: Decelerate
v(t)= Acceleration
Dynamics: dx/dt =f(x(t),k(t),v(t))

The tool-chain: Checking Engines

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
…
```

Scenario Definition Language



Falsification Engine (e.g., S-TaLiRo)

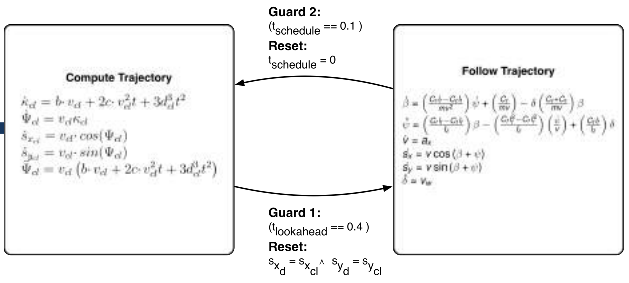
Almost-Exhaustive Verification Engine (e.g., dReach)

The tool-chain: Common formalism for simulation and verification

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
…
```

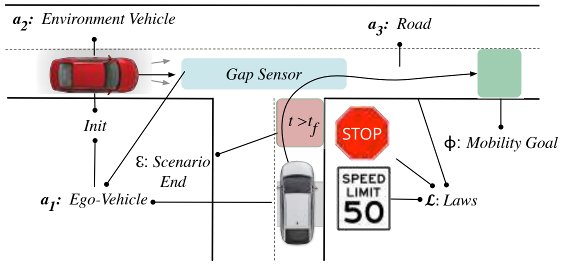
Scenario Definition Language

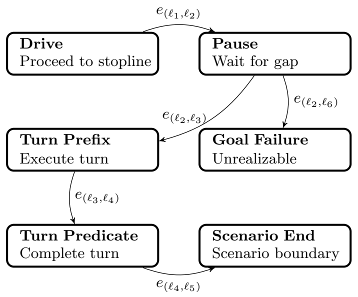

Compute Trajectory
$\dot{\kappa}_{cl} = b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2$
$\dot{\Psi}_{cl} = v_{cl}\kappa_{cl}$
$\dot{s}_{x_{cl}} = v_{cl} \cdot cos(\Psi_{cl})$
$\dot{s}_{y_{cl}} = v_{cl} \cdot sin(\Psi_{cl})$
$\dot{\Psi}_{cl} = v_{cl}\left(b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2\right)$
Follow Trajectory
$\dot{v} = a_x$
$\dot{s}_x = v\cos(\beta + \psi)$
$\dot{s}_y = v\sin(\beta + \psi)$
$\dot{\delta} = v_w$
Guard 2:
($t_{schedule}$ == 0.1)
Reset:
$t_{schedule}$ = 0
Guard 1:
($t_{lookahead}$ == 0.4)
Reset:
$s_{x_d} = s_{x_{cl}} \wedge s_{y_d} = s_{y_{cl}}$

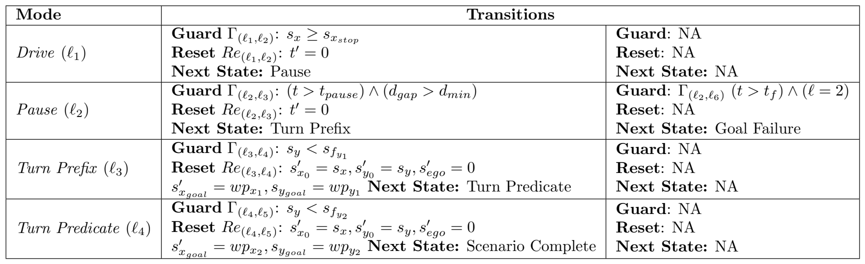
Common formalism
(Intermediate Representation)
for verification engines

Falsification Engine
(e.g., S-TaLiRo)

Almost-Exhaustive
Verification Engine
(e.g., dReach)







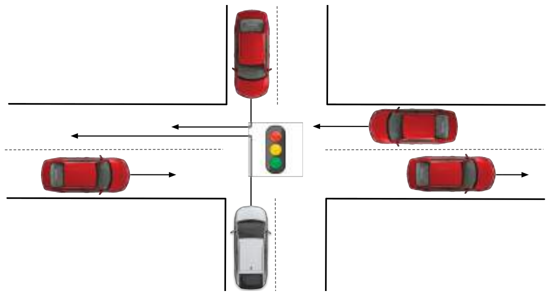
Mode	Transitions	
Drive (ℓ_1)	**Guard** $\Gamma_{(\ell_1,\ell_2)}$: $s_x \geq s_{x_{stop}}$ **Reset** $Re_{(\ell_1,\ell_2)}$: $t' = 0$ **Next State:** Pause	**Guard**: NA **Reset**: NA **Next State:** NA
Pause (ℓ_2)	**Guard** $\Gamma_{(\ell_2,\ell_3)}$: $(t > t_{pause}) \wedge (d_{gap} > d_{min})$ **Reset** $Re_{(\ell_2,\ell_3)}$: $t' = 0$ **Next State:** Turn Prefix	**Guard**: $\Gamma_{(\ell_2,\ell_6)}$ $(t > t_f) \wedge (\ell = 2)$ **Reset**: NA **Next State:** Goal Failure
Turn Prefix (ℓ_3)	**Guard** $\Gamma_{(\ell_3,\ell_4)}$: $s_y < s_{f_{y_1}}$ **Reset** $Re_{(\ell_3,\ell_4)}$: $s'_{x_0} = s_x, s'_{y_0} = s_y, s'_{ego} = 0$ $s'_{x_{goal}} = wp_{x_1}, s_{y_{goal}} = wp_{y_1}$ **Next State:** Turn Predicate	**Guard**: NA **Reset**: NA **Next State:** NA
Turn Predicate (ℓ_4)	**Guard** $\Gamma_{(\ell_4,\ell_5)}$: $s_y < s_{f_{y_2}}$ **Reset** $Re_{(\ell_4,\ell_5)}$: $s'_{x_0} = s_x, s'_{y_0} = s_y, s'_{ego} = 0$ $s'_{x_{goal}} = wp_{x_2}, s_{y_{goal}} = wp_{y_2}$ **Next State:** Scenario Complete	**Guard**: NA **Reset**: NA **Next State:** NA

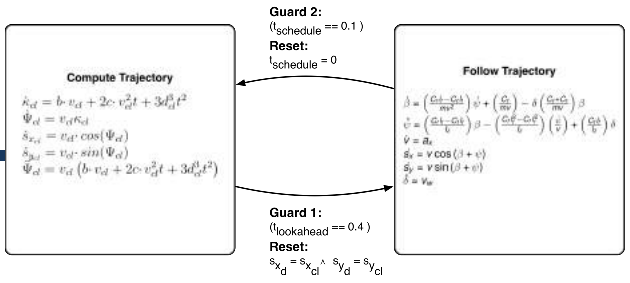
The tool-chain: Conversion from formalism to tool formats

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
...
```

Scenario Definition Language





Guard 2:
$(t_{schedule} == 0.1)$
Reset:
$t_{schedule} = 0$
Compute Trajectory
$\dot{\kappa}_{cl} = b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2$
$\dot{\Psi}_{cl} = v_{cl}\kappa_{cl}$
$\dot{s}_{x_{cl}} = v_{cl} \cdot cos(\Psi_{cl})$
$\dot{s}_{y_{cl}} = v_{cl} \cdot sin(\Psi_{cl})$
$\dot{\Psi}_{cl} = v_{cl}\left(b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2\right)$
Follow Trajectory
$\dot{\beta} = \left(\frac{C_r l_r - C_f l_f}{mv^2}\right)\dot{\psi} + \left(\frac{C_f}{mv}\right) - \delta\left(\frac{C_f + C_r}{mv}\right)\beta$
$\dot{\psi} = \left(\frac{C_r l_r - C_f l_f}{I_z}\right)\beta - \left(\frac{C_f l_f^2 - C_r l_r^2}{I_z}\right)\left(\frac{\dot{\psi}}{v}\right) + \left(\frac{C_f l_f}{I_z}\right)\delta$
$\dot{v} = a_x$
$\dot{s}_x = v\cos(\beta + \psi)$
$\dot{s}_y = v\sin(\beta + \psi)$
$\dot{\delta} = v_w$
Guard 1:
$(t_{lookahead} == 0.4)$
Reset:
$s_{x_d} = s_{x_{cl}} \wedge s_{y_d} = s_{y_{cl}}$

Common formalism for verification engines

Falsification Engine (e.g., S-TaLiRo)

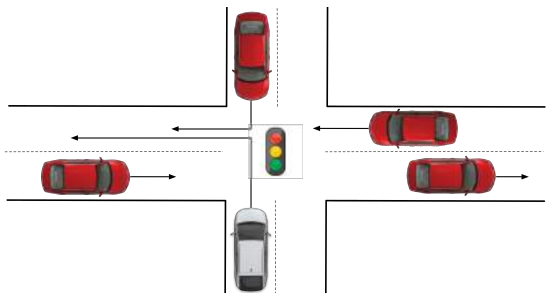
Almost-Exhaustive Verification Engine (e.g., dReach)

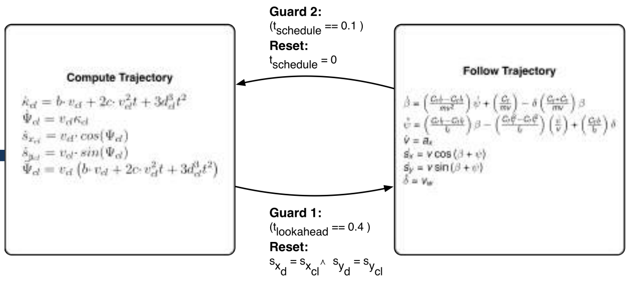
The tool-chain: Formal specification in Metric Temporal Logic

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road(‘highway’)
…
```

Scenario Definition Language




Guard 2:
$(t_{schedule} == 0.1)$
Reset:
$t_{schedule} = 0$
Compute Trajectory
$\dot{\kappa}_{cl} = b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2$
$\dot{\Psi}_{cl} = v_{cl} \kappa_{cl}$
$\dot{s}_{x_{cl}} = v_{cl} \cdot cos(\Psi_{cl})$
$\dot{s}_{y_{cl}} = v_{cl} \cdot sin(\Psi_{cl})$
$\dot{\Psi}_{cl} = v_{cl}(b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2)$
Follow Trajectory
$\dot{v} = a_x$
$\dot{s}_x = v \cos(\beta + \psi)$
$\dot{s}_y = v \sin(\beta + \psi)$
$\dot{\delta} = v_w$
Guard 1:
$(t_{lookahead} == 0.4)$
Reset:
$s_{x_d} = s_{x_{cl}} \wedge s_{y_d} = s_{y_{cl}}$

Common formalism for verification engines

Falsification Engine (e.g., S-TaLiRo)

Almost-Exhaustive Verification Engine (e.g., dReach)

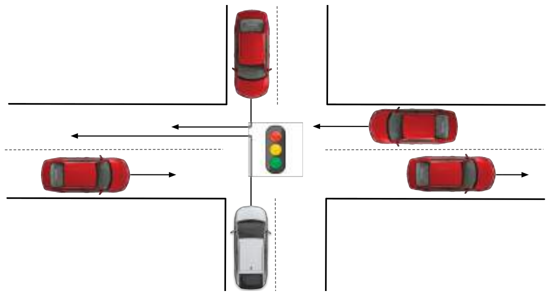
MTL Specification

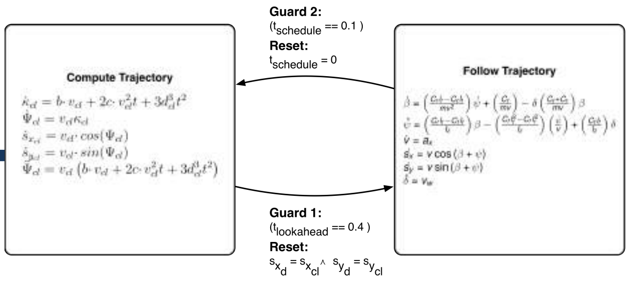
The tool-chain: Robustness-Guided Verification

SCENARIO ENTRY

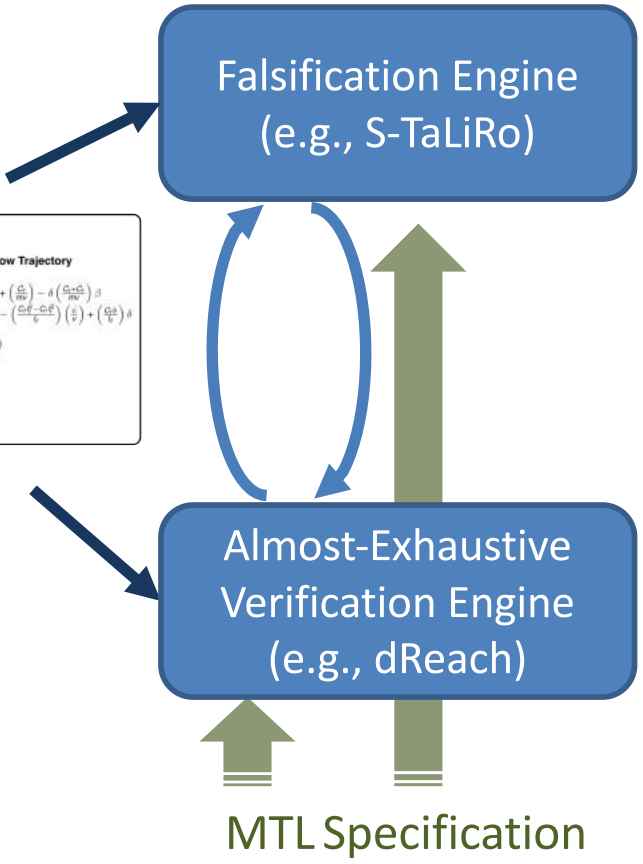
```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
...
```

Scenario Definition Language





Common formalism for verification engines


Falsification Engine (e.g., S-TaLiRo)
Almost-Exhaustive Verification Engine (e.g., dReach)
MTL Specification

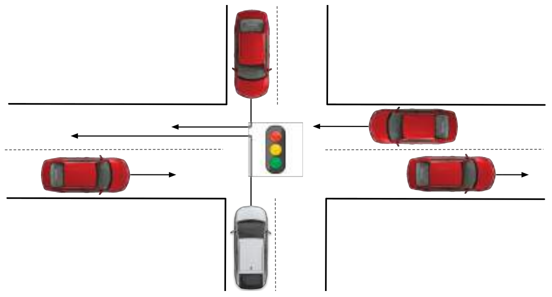
The tool-chain: Integration and testing of real code

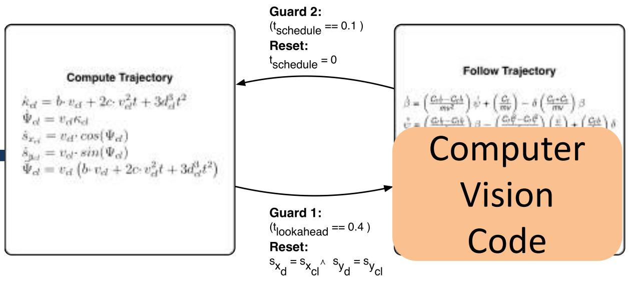
Falsification Engine
(e.g., S-TaLiRo)

SCENARIO ENTRY

```
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
…
```

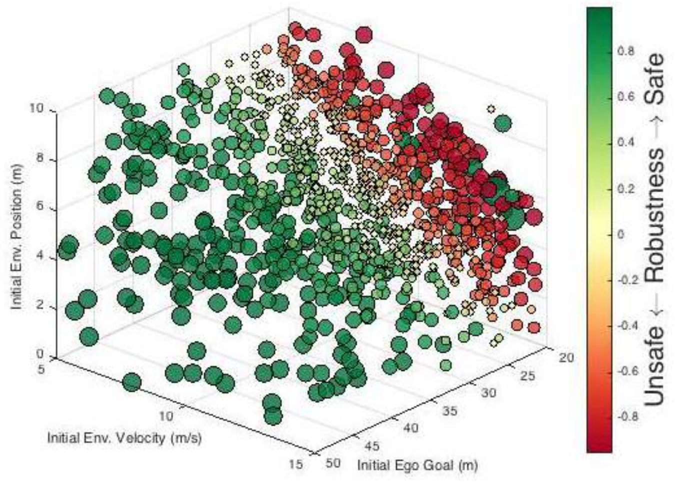
Scenario Definition Language





Computer Vision Code

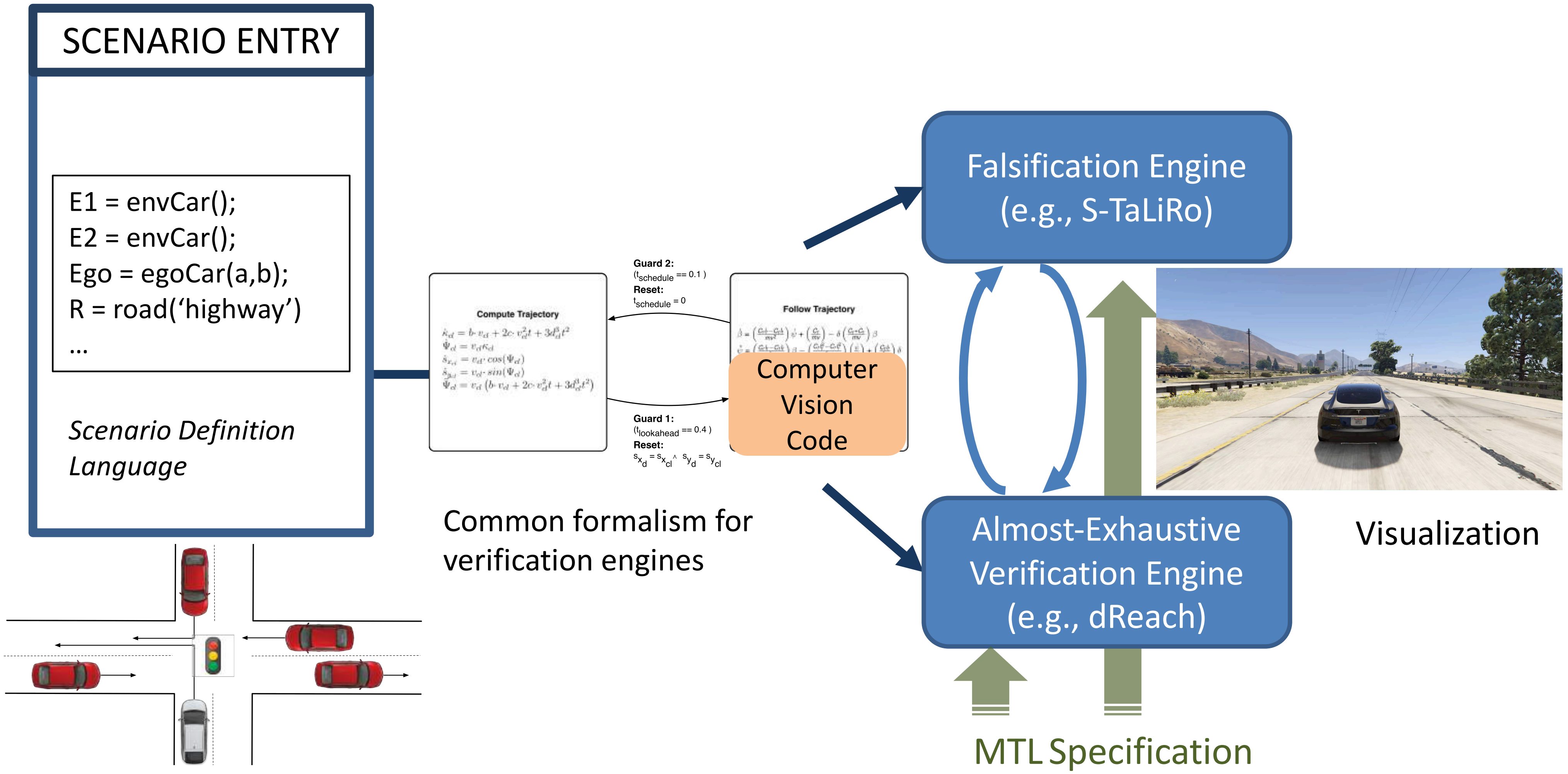
Common formalism for verification engines



Almost-Exhaustive Verification Engine
(e.g., dReach)

MTL Specification

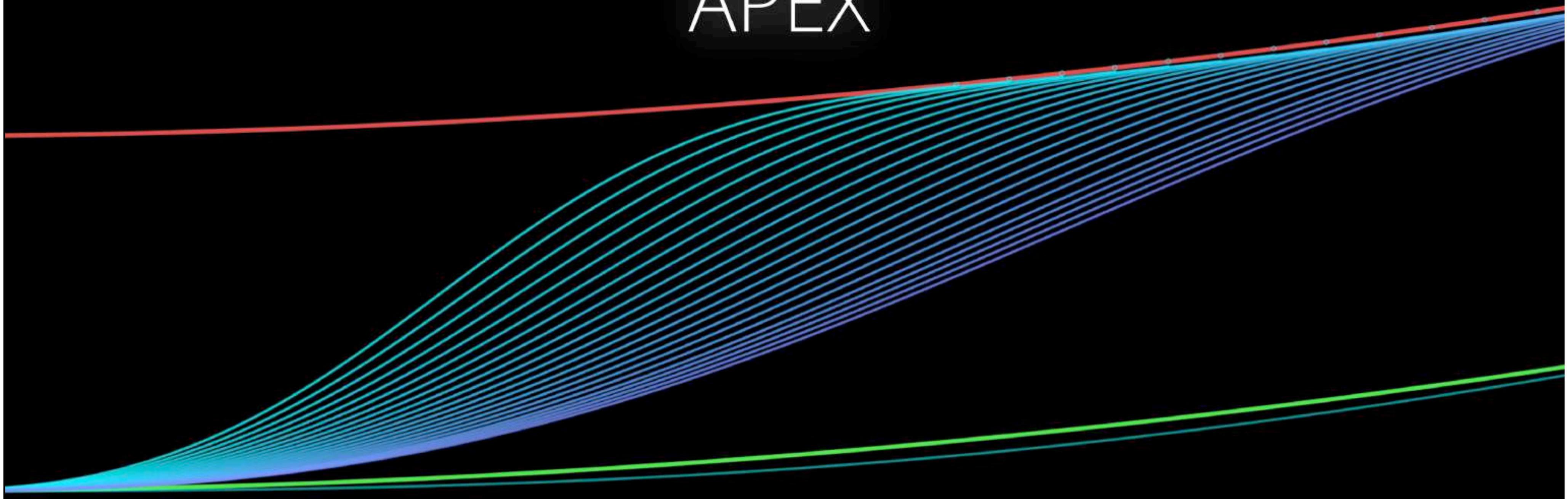
The tool-chain: Visualization of accidents and violations



SCENARIO ENTRY
E1 = envCar();
E2 = envCar();
Ego = egoCar(a,b);
R = road('highway')
…
Scenario Definition Language
Compute Trajectory
$\dot{\kappa}_{cl} = b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2$
$\dot{\Psi}_{cl} = v_{cl}\kappa_{cl}$
$\dot{s}_{x_{cl}} = v_{cl} \cdot cos(\Psi_{cl})$
$\dot{s}_{y_{cl}} = v_{cl} \cdot sin(\Psi_{cl})$
$\dot{\Psi}_{cl} = v_{cl}(b \cdot v_{cl} + 2c \cdot v_{cl}^2 t + 3d_{cl}^3 t^2)$
Guard 2:
$(t_{schedule} == 0.1)$
Reset:
$t_{schedule} = 0$
Follow Trajectory
Computer Vision Code
Guard 1:
$(t_{lookahead} == 0.4)$
Reset:
$s_{x_d} = s_{x_{cl}} \wedge s_{y_d} = s_{y_{cl}}$
Common formalism for verification engines
Falsification Engine (e.g., S-TaLiRo)
Almost-Exhaustive Verification Engine (e.g., dReach)
Visualization
MTL Specification

Autonomous vehicle **P**lan verification and **EX**ecution

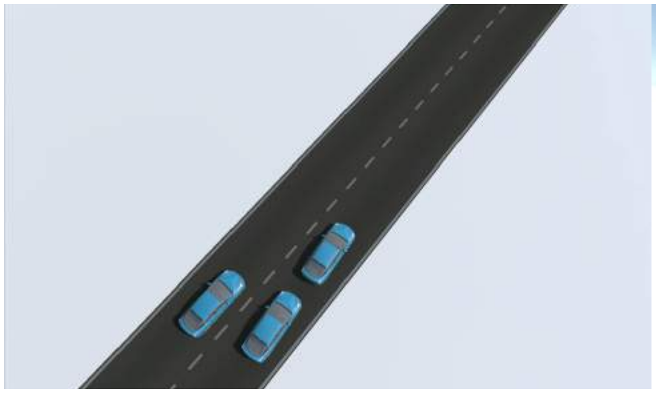
APEX

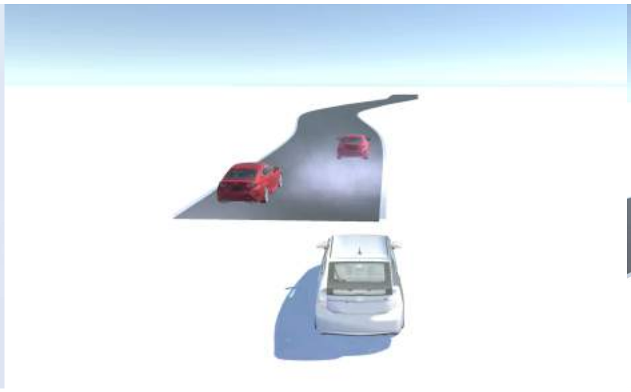


PRECISE

PENN RESEARCH IN EMBEDDED COMPUTING
AND INTEGRATED SYSTEMS ENGINEERING

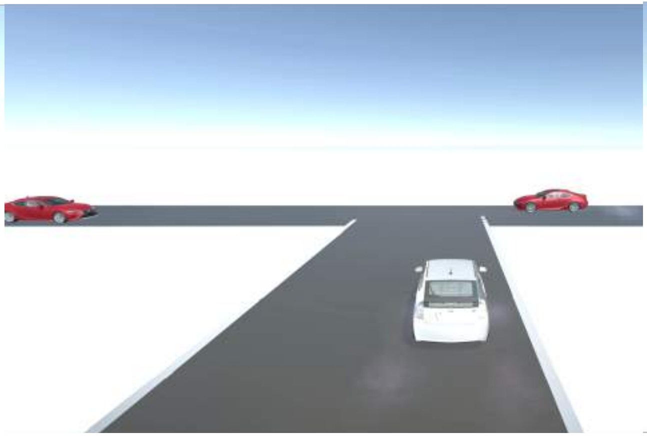
APEX Toolbox: Basic Scenario Library


Multi-lane Merges

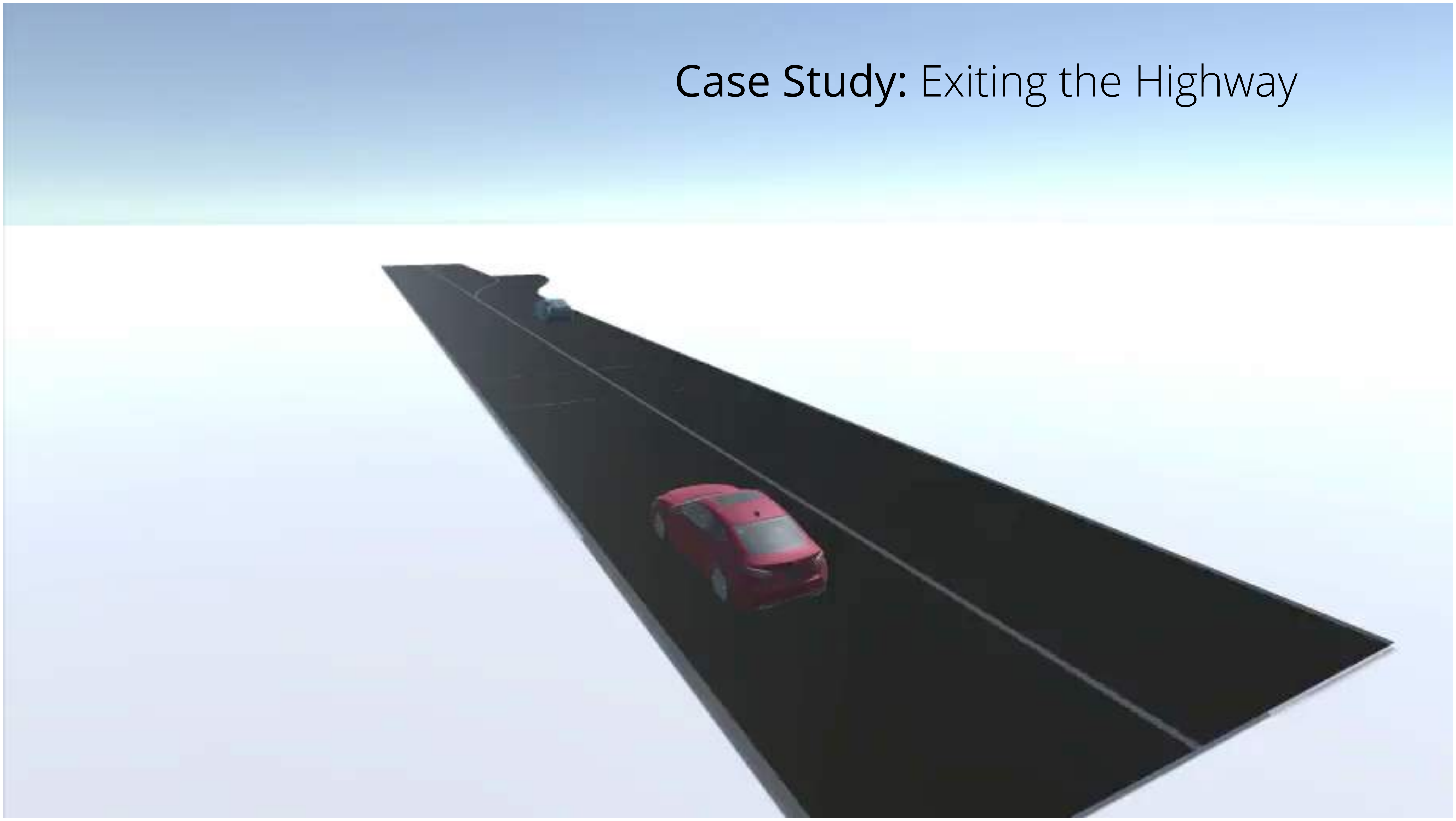

Curved Roads

Highway On-ramps

Highway Exit


T-Junctions

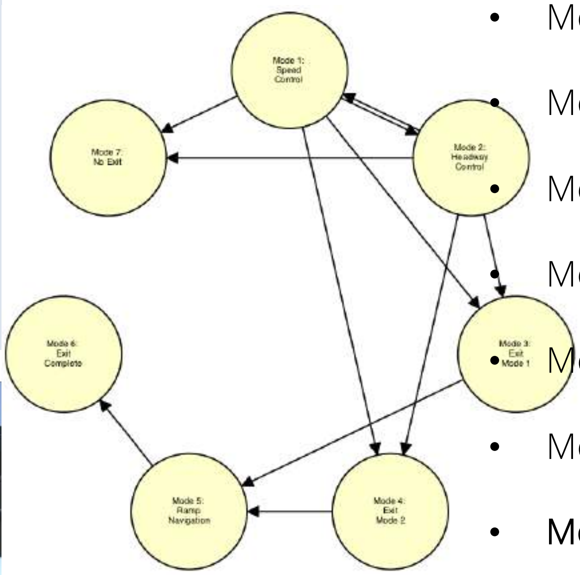
Roundabouts



Case Study: Exiting the Highway

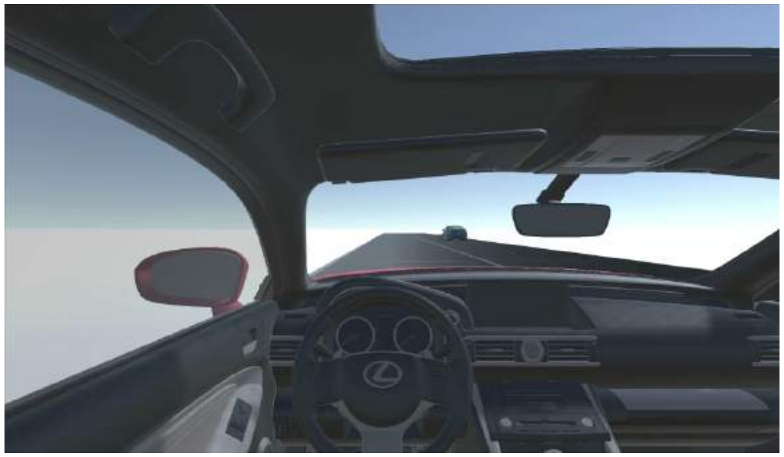
Case Study : Exiting the Highway

An unsafe execution…



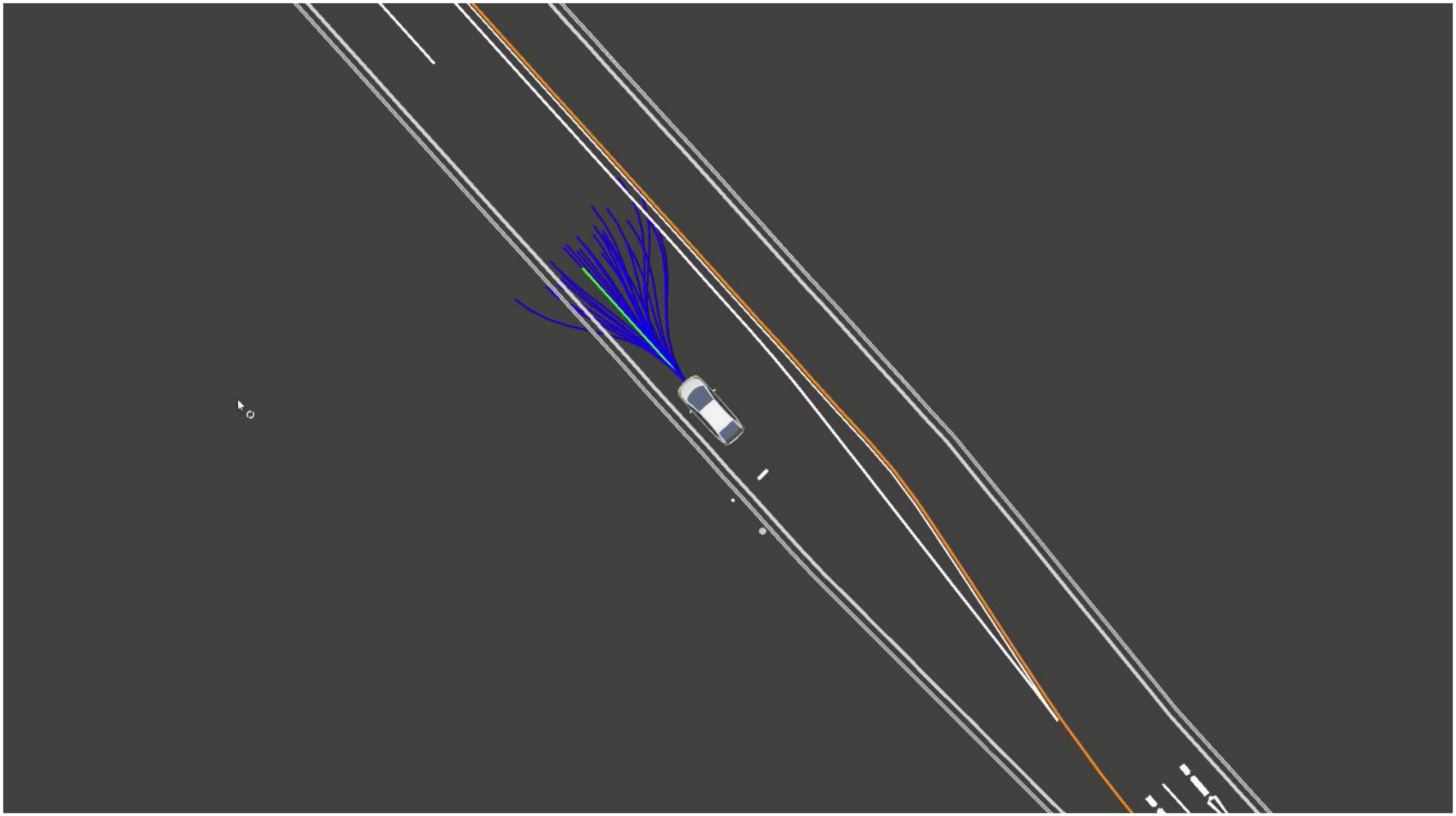
- Mode 1 to Mode 2
  - Large gap…
- Mode 1 to Mode 3
  - Proximity to exit point, ordering 1.
- Mode 1 to Mode 4
  - Proximity to exit point, ordering 2.
- Mode 1 to Mode 7
  - Pass point of no return.
- Mode 2 to Mode 1
  - Exceeding safe speed
- Mode 2 to Mode 3
  - Proximity to exit point, ordering 1.
- Mode 2 to Mode 4
  - Proximity to exit point, ordering 2.
- Mode 2 to Mode 7
  - Pass point of no return.
- Mode 3 to Mode 5
  - At exit point, replan
- Mode 4 to Mode 5
  - At exit point, replan
- Mode 5 to Mode 6
  - Exit Complete

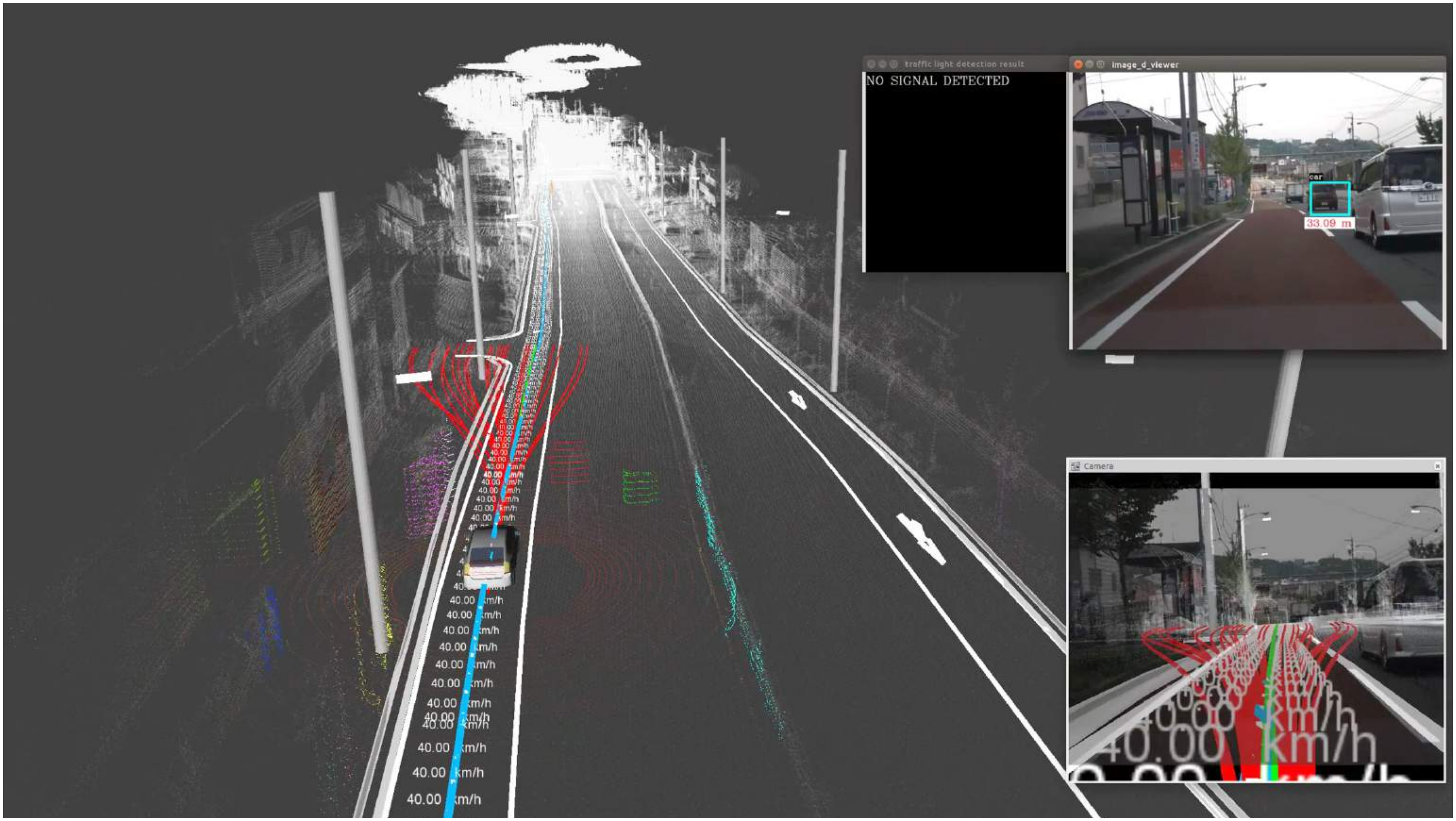
Case Study : Counterexample



Exceeded allowable speed on curve. Forgot to change desired velocity on the exit ramp...

	Robustness	Falsified	Time
Run 1	2.923954853228472	0	1.015345936
Run 2	5.08758785008145	0	0.64356243
Run 3	1.58322571985417	0	0.739456261
Run 4	-1.33481474494335	1	0.647890734
Run 5	1.19092922455614	0	0.653613874
Run 6	0.7644210734606593	0	0.424821741
Run 7	3.50257488220876	0	0.417468565
Run 8	1.67075771080459	0	0.422870814
Run 9	-0.06934283643283 12	1	0.246647509
Run 10	0.840635324412428	0	0.416844002
Run 11	0.0583178152910584	0	0.412734793
Run 12	0.408473731737928	0	0.414736315
Run 13	0.0880809121895942	0	0.420027416
Run 14	0.323334605645278	0	0.399421832
Run 15	1.86618290153492	0	0.718619361
Run 16	0.357522415132801	0	0.637938451
Run 17	-0.424533871152353	1	0.677554789
Run 18	-1.52676106159999	1	0.652093781
Run 19	0.276072018492533	0	0.641657179
...	...	...	...

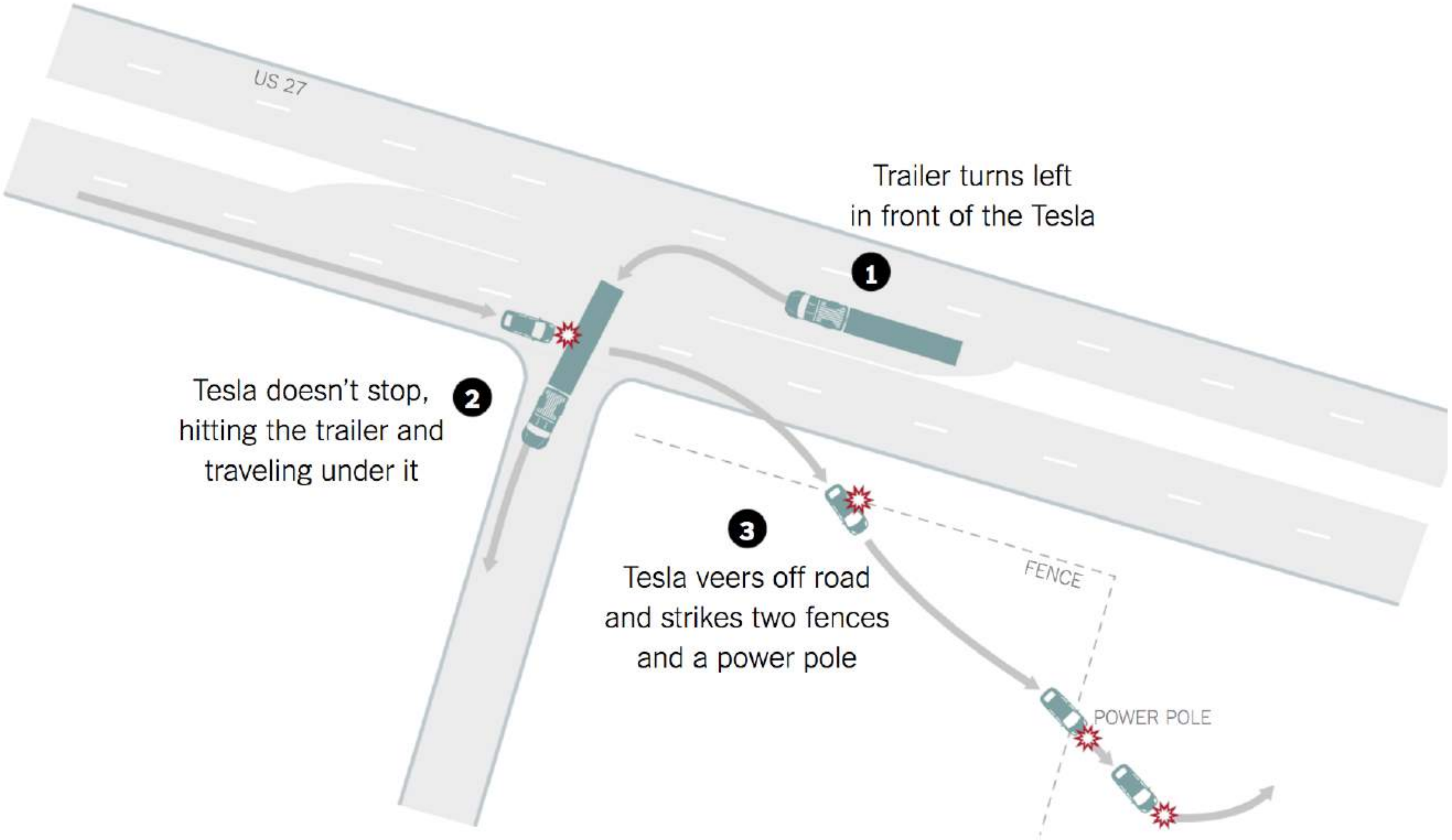




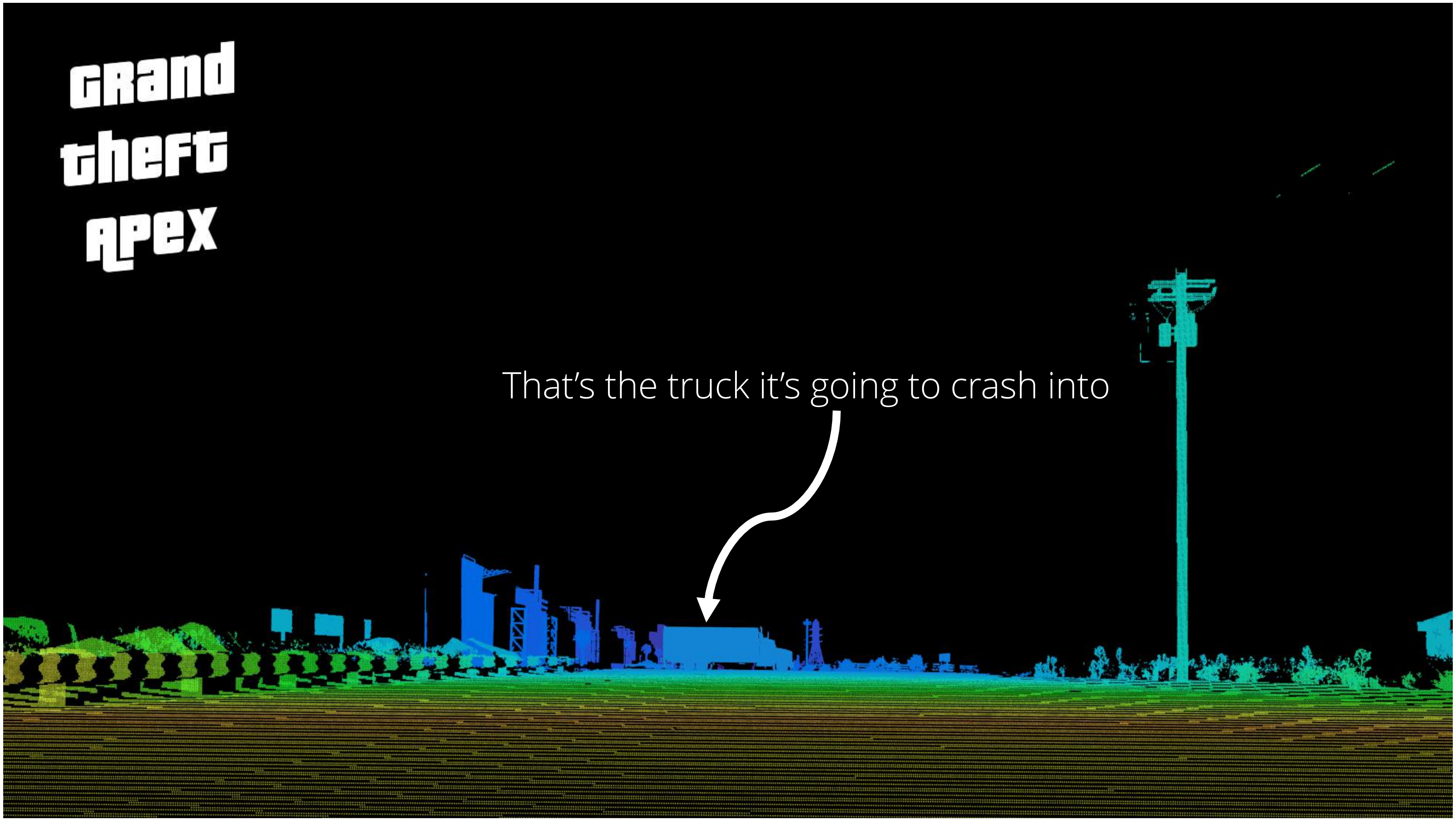
NO SIGNAL DETECTED

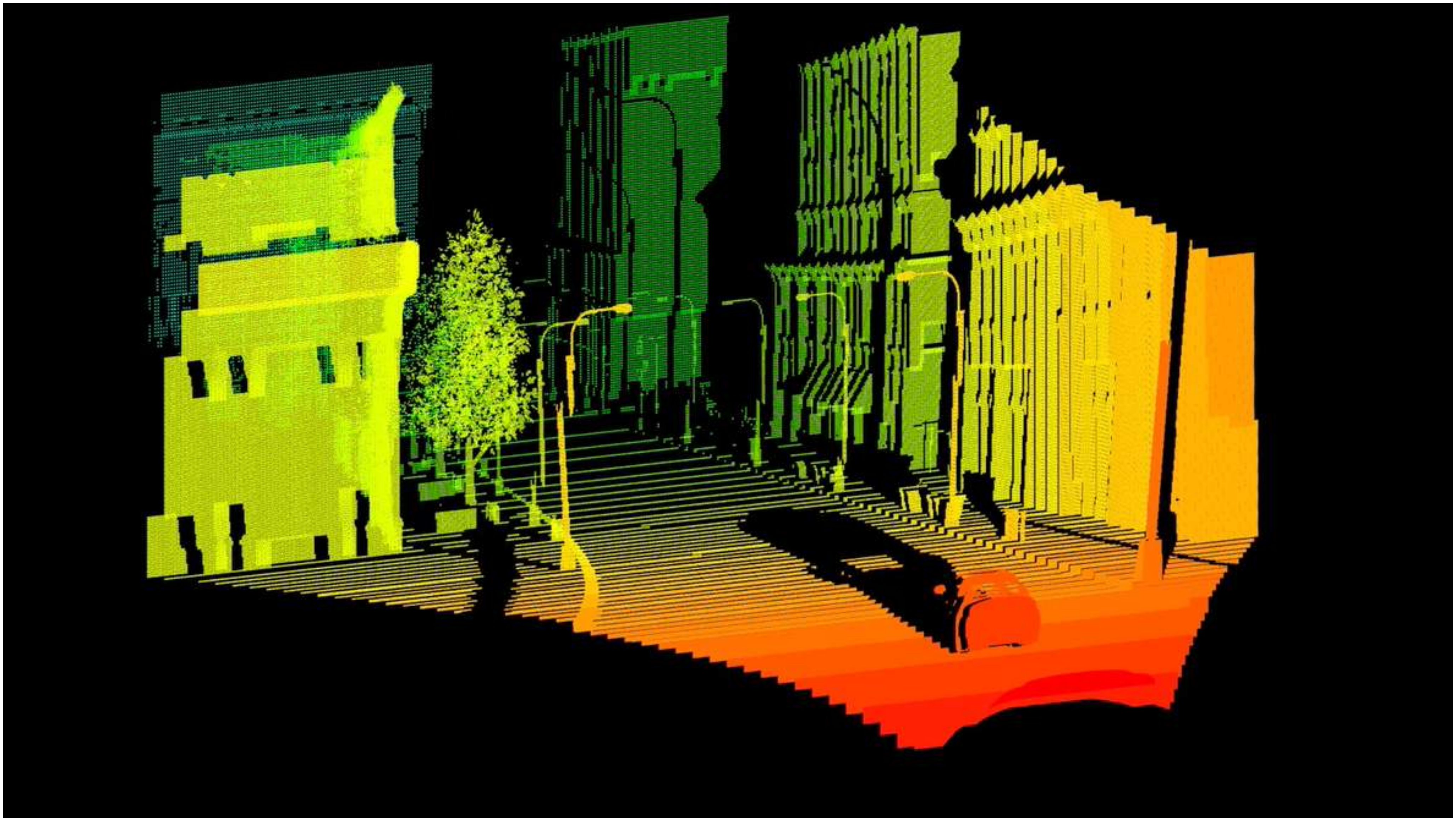
Post Accident Analysis: Adversarial Search

Meanwhile: Things are getting serious...

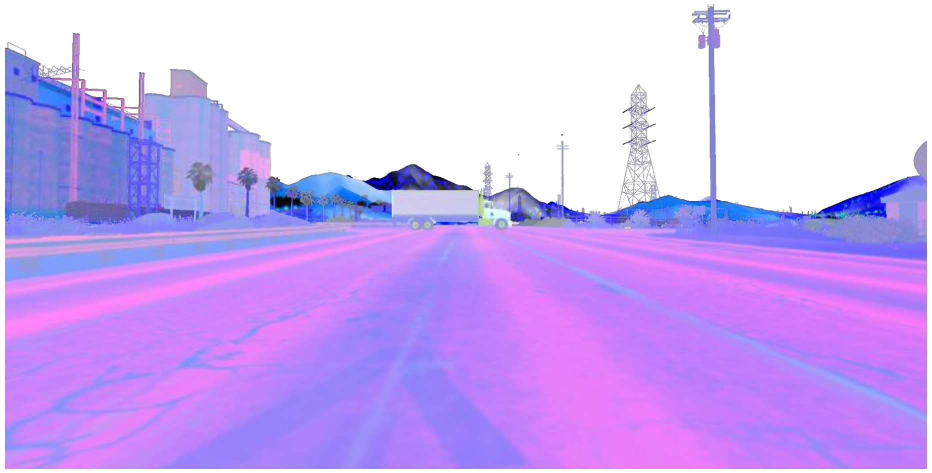
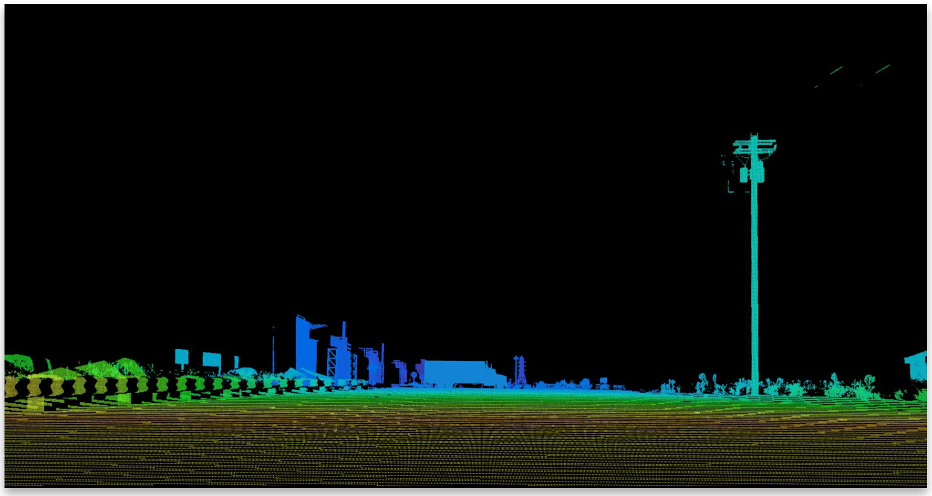

US 27
Trailer turns left in front of the Tesla
1
Tesla doesn't stop, hitting the trailer and traveling under it
2
3
Tesla veers off road and strikes two fences and a power pole
FENCE
POWER POLE

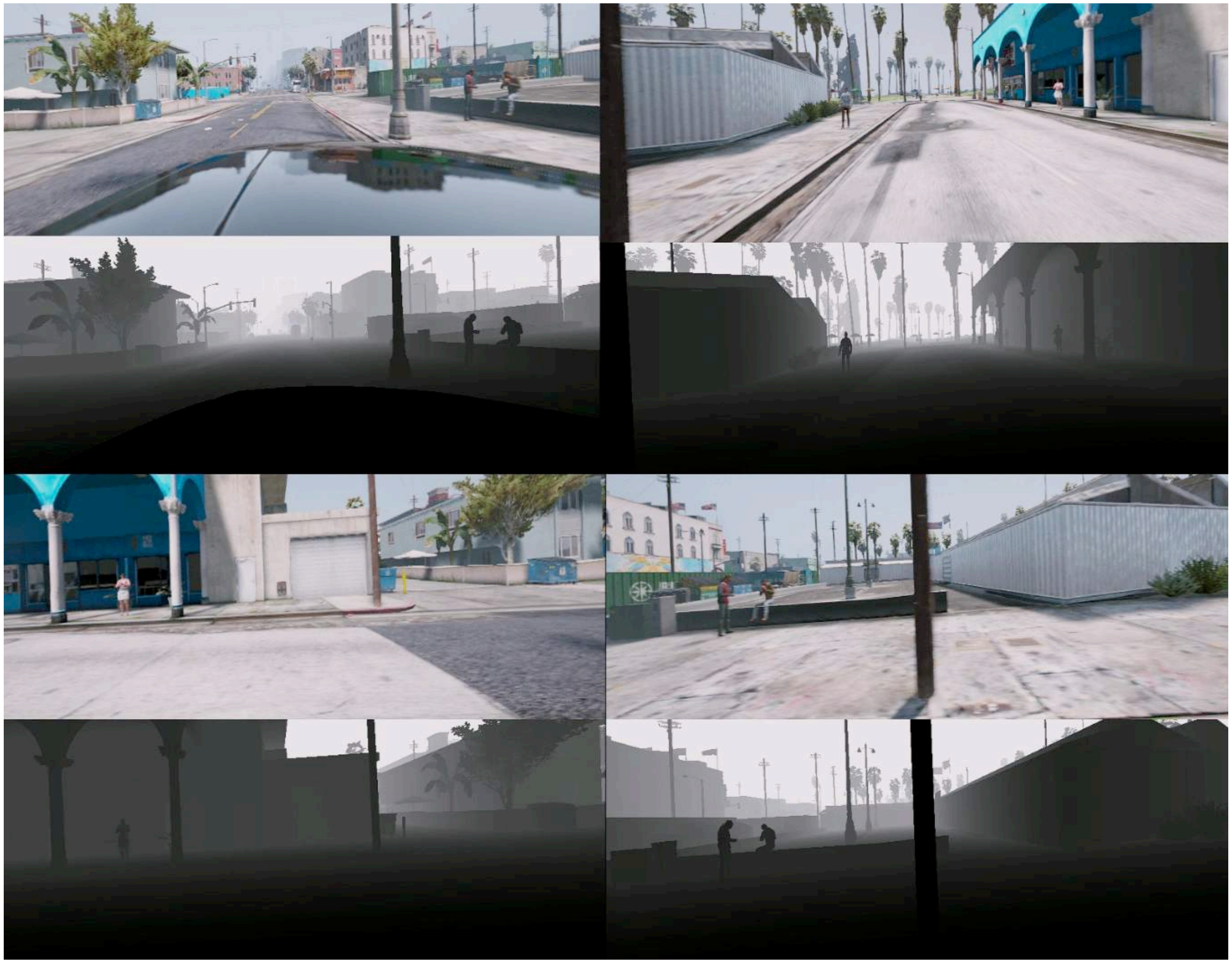
Embed self-reflective capacity in AV agents operating in a photorealistic open world, enabling robustness guided data synthesis, and unguided scenario generation leveraging multiple agent experiences simultaneously


Grand
theft
Apex
That’s the truck it’s going to crash into



Simulation: Artificial Sensors

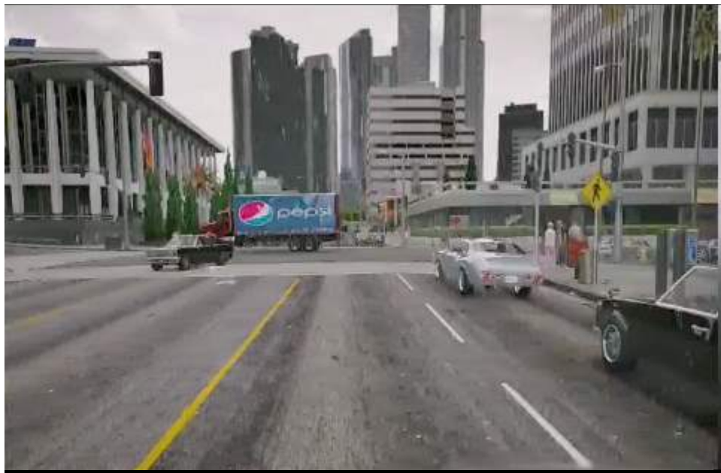


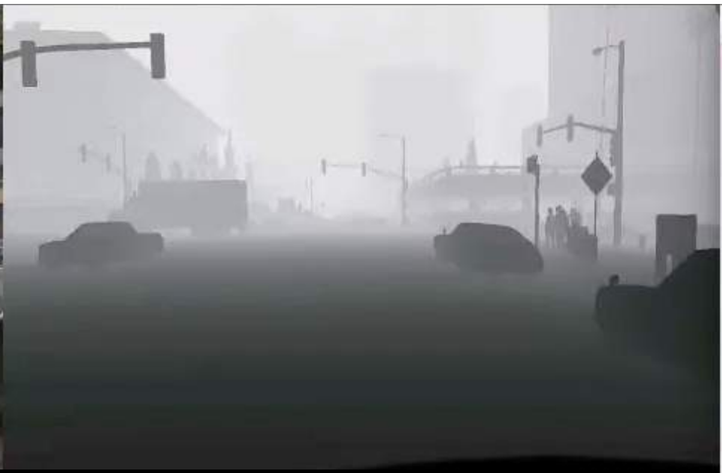
Multi-rendering from single game instance…

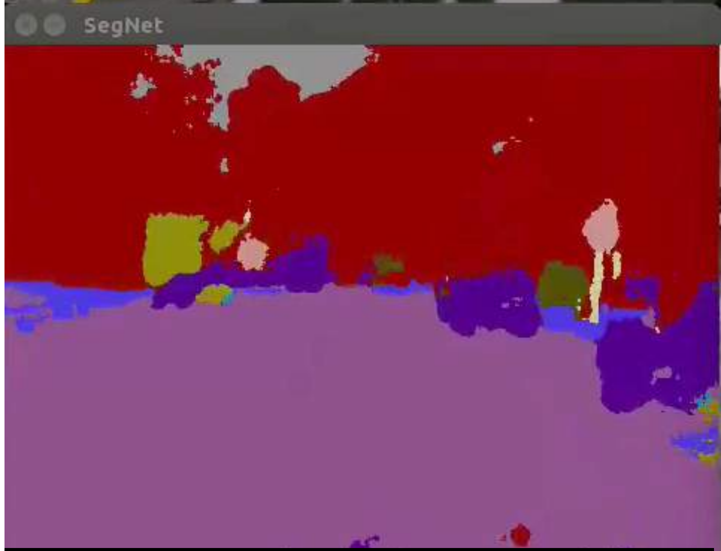
Can ***mimic modern camera based*** *SDC systems ie AP2…*

Top: *RGB*
Bottom: *Depth*

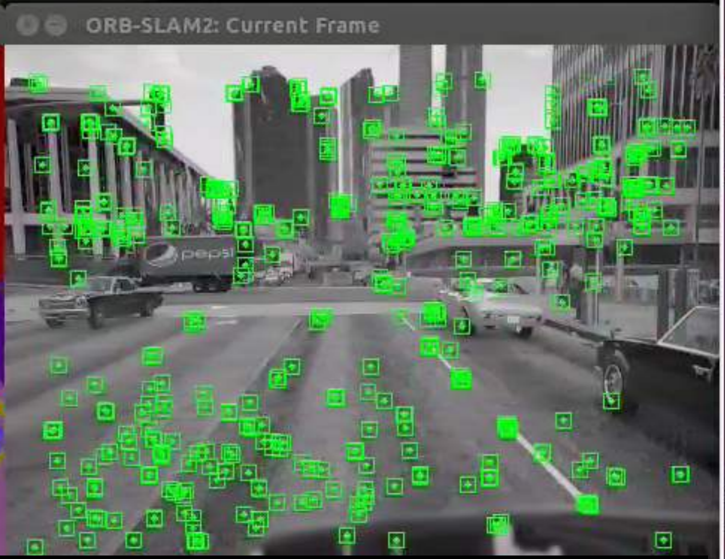
Clockwise:
Front
Back
Left
Right


RGB – Simulate Camera

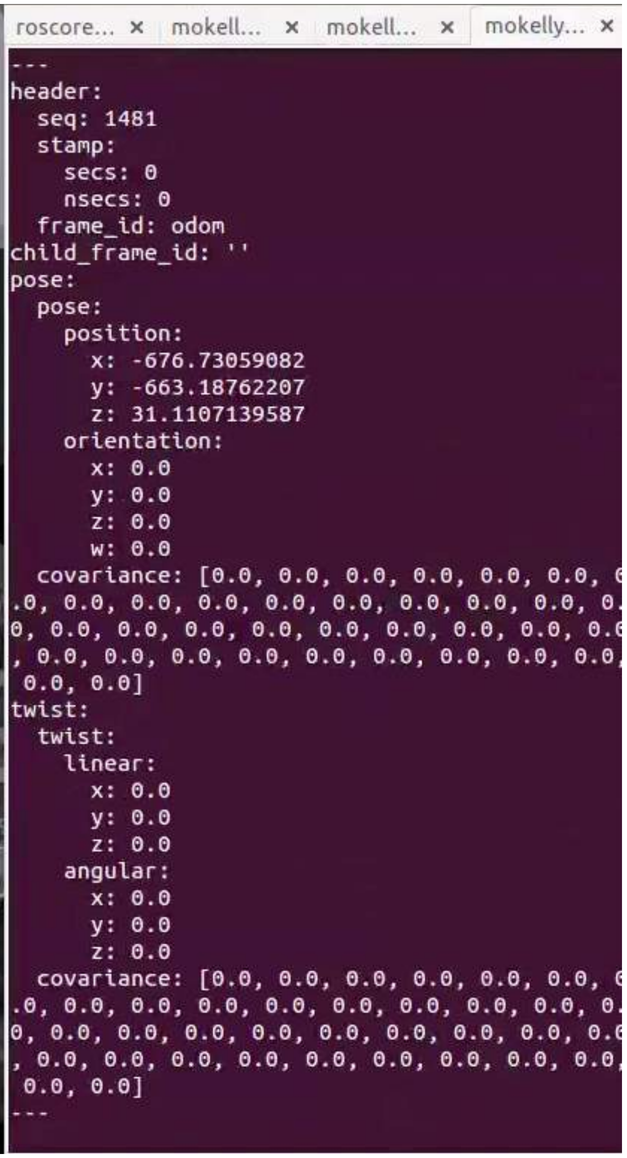

Depth – gives us ground truth



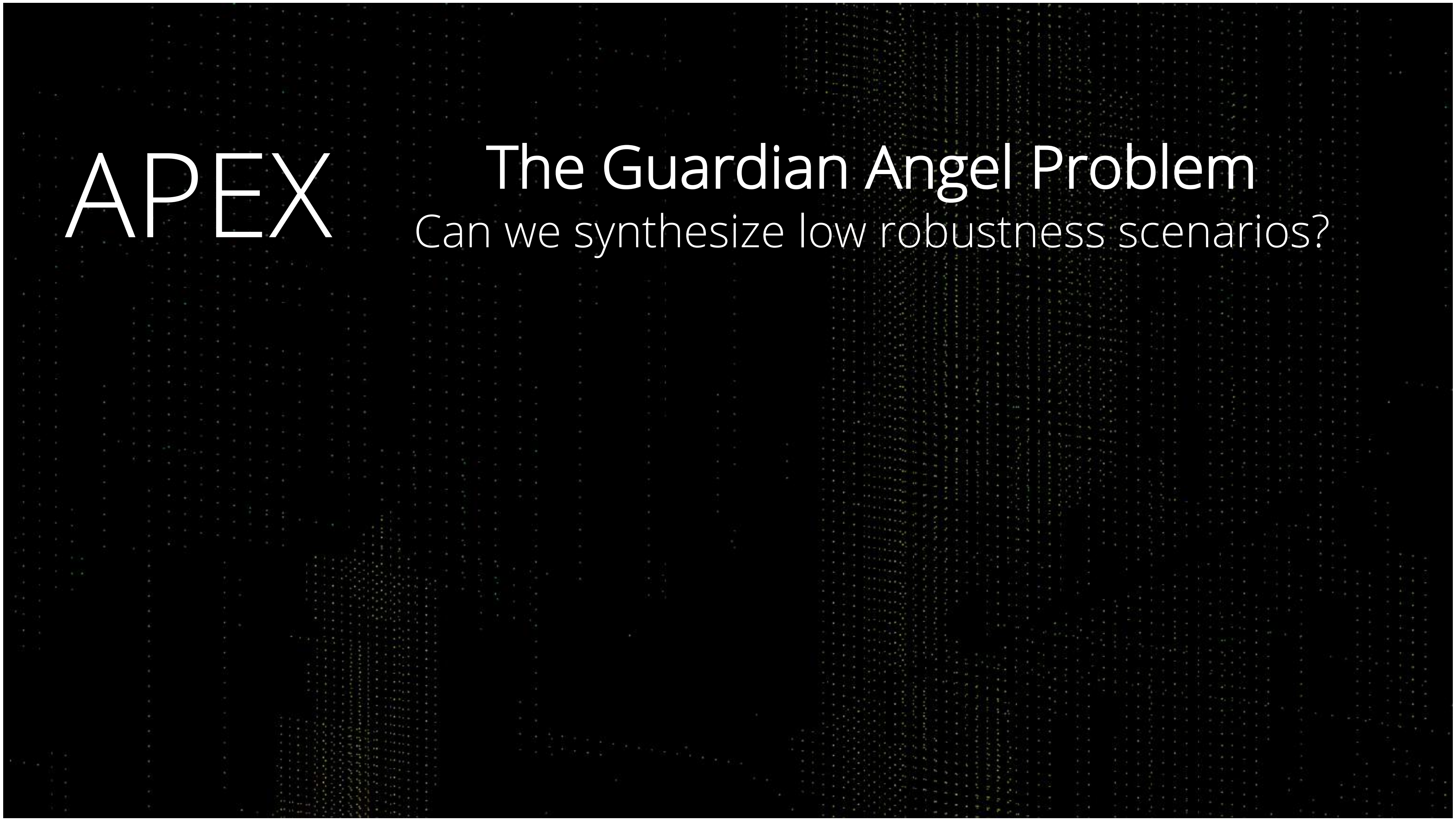
Segmentation - use mpc planners



SLAM



```
---
header:
  seq: 1481
  stamp:
    secs: 0
    nsecs: 0
  frame_id: odom
child_frame_id: ''
pose:
  pose:
    position:
      x: -676.73059082
      y: -663.18762207
      z: 31.1107139587
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 0.0
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0]
twist:
  twist:
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0]
---
```



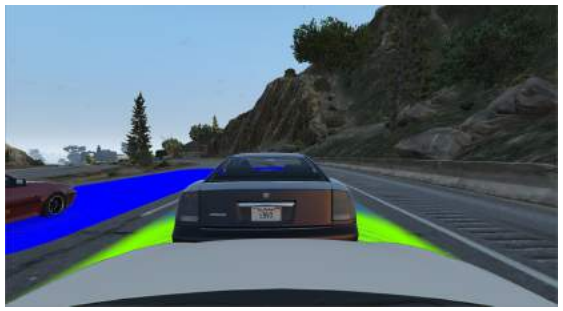
APEX

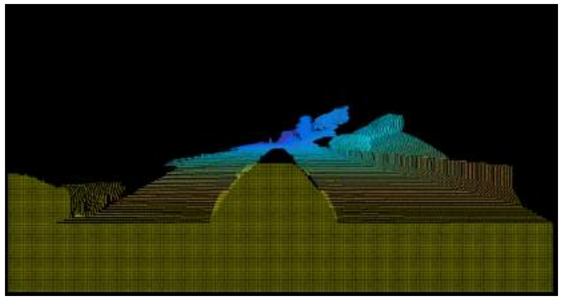
The Guardian Angel Problem

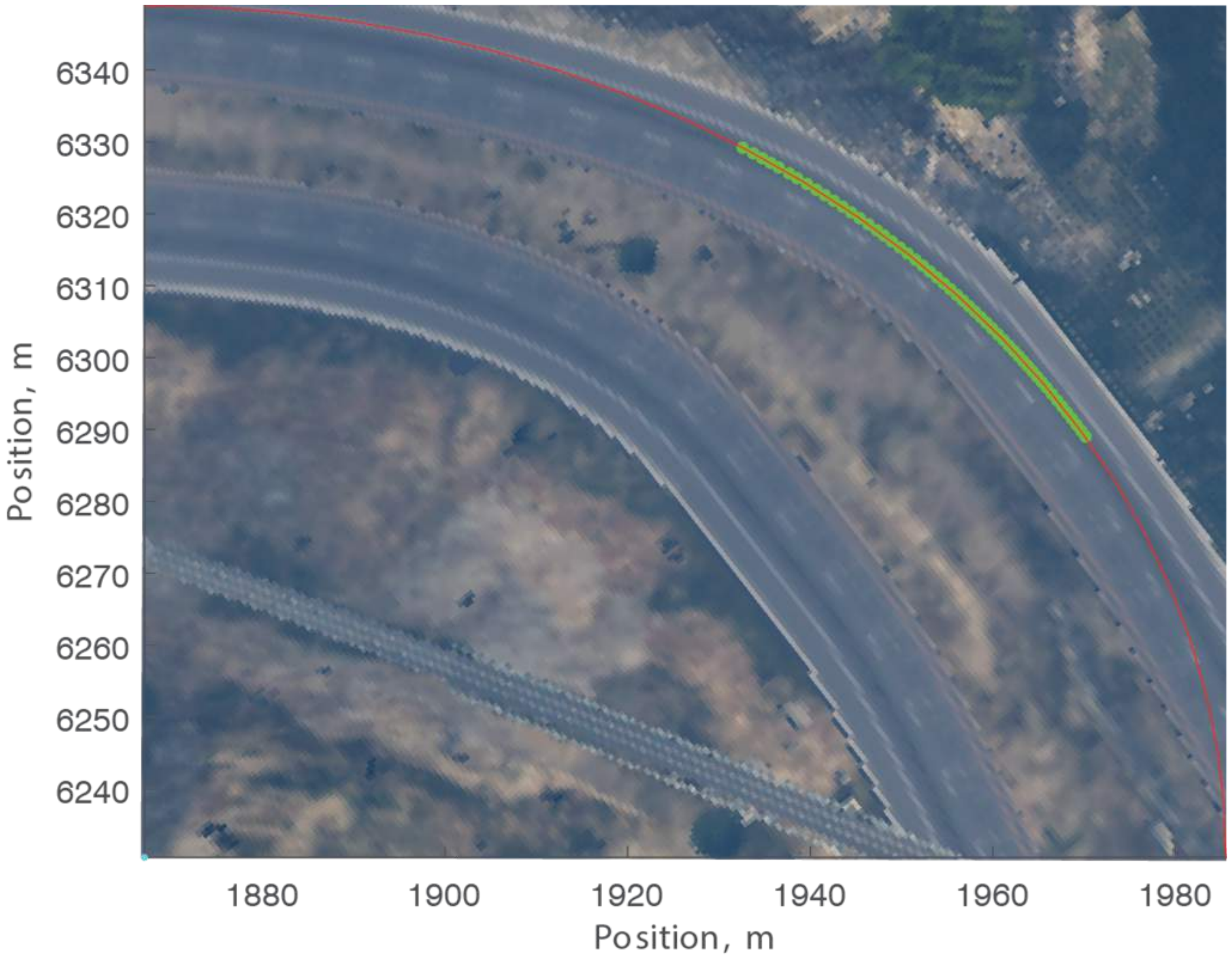
Can we synthesize low robustness scenarios?

Scenario Search

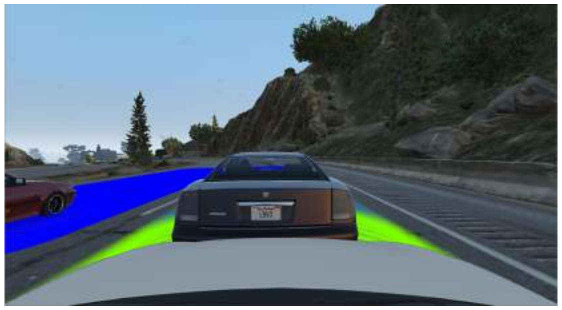
Control Interfaces: APEX Robust Testing and Verification

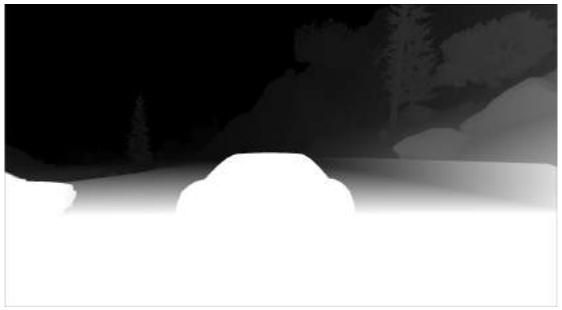


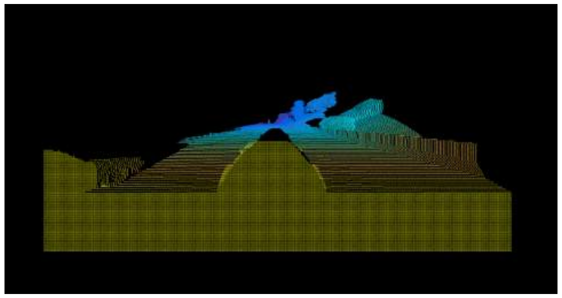


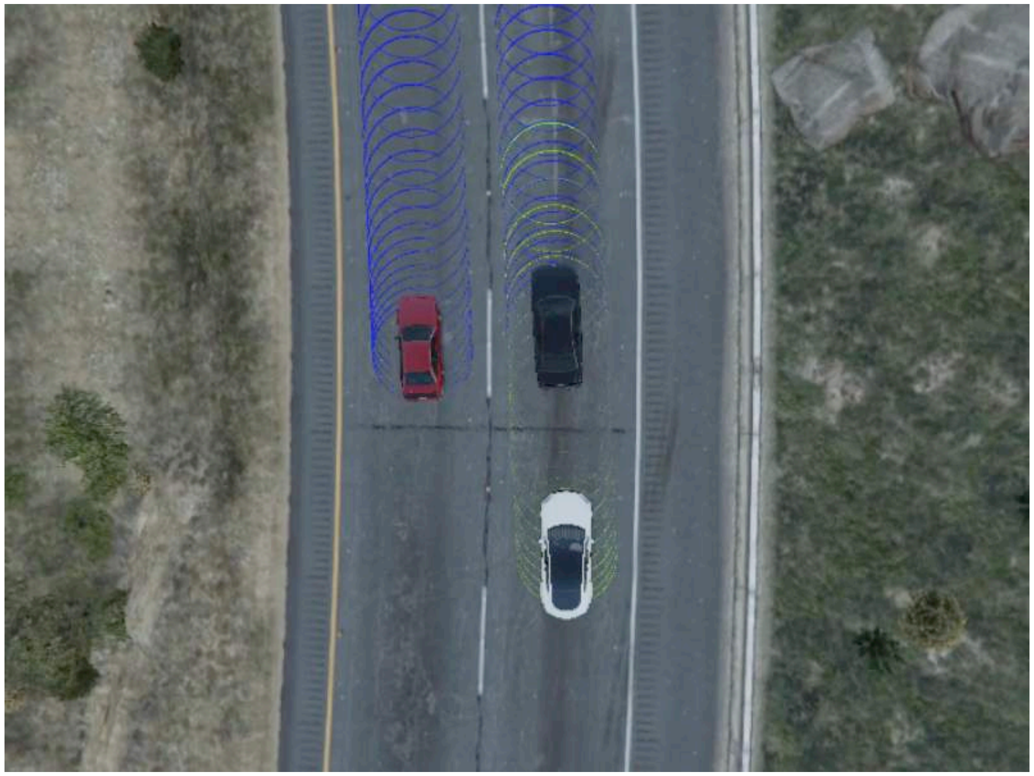


Control Interfaces: APEX Robust Testing...



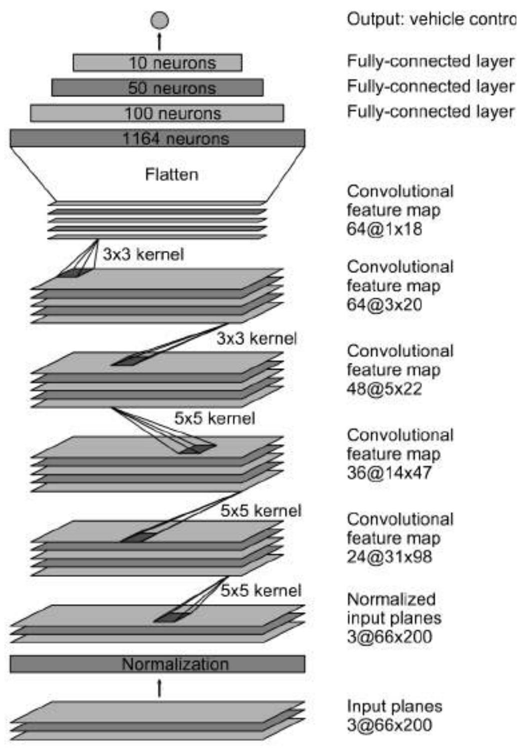


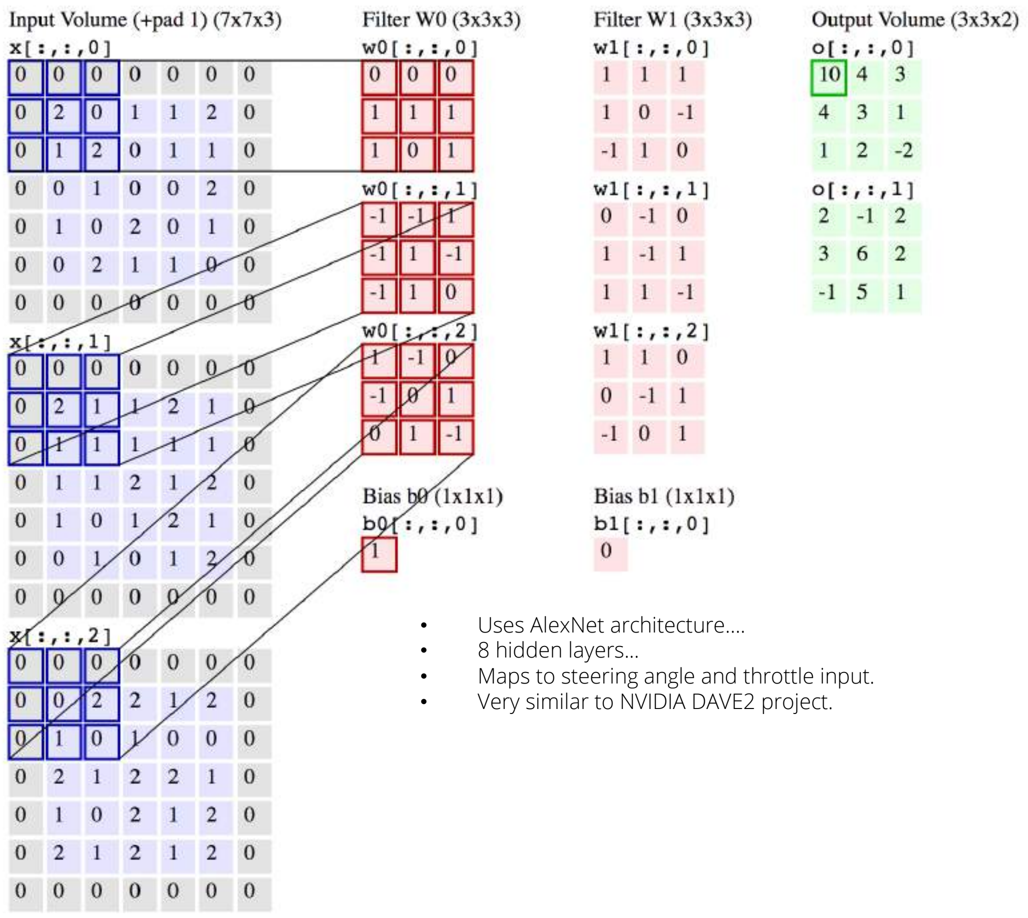




End-to-End Learning

Control Interfaces: End-to-end CNN

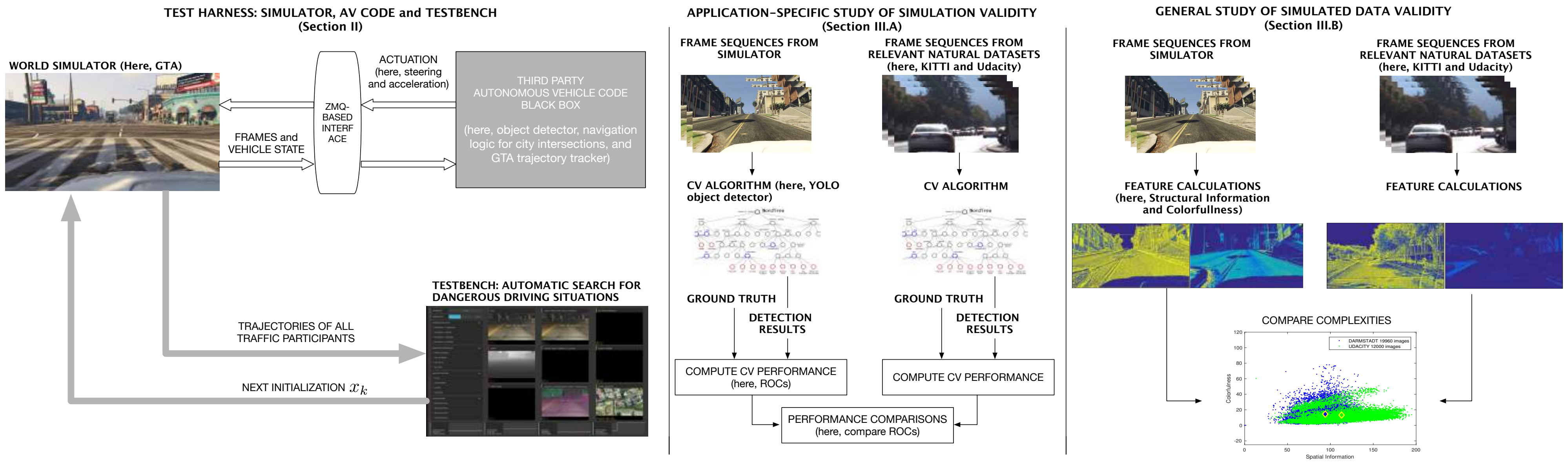




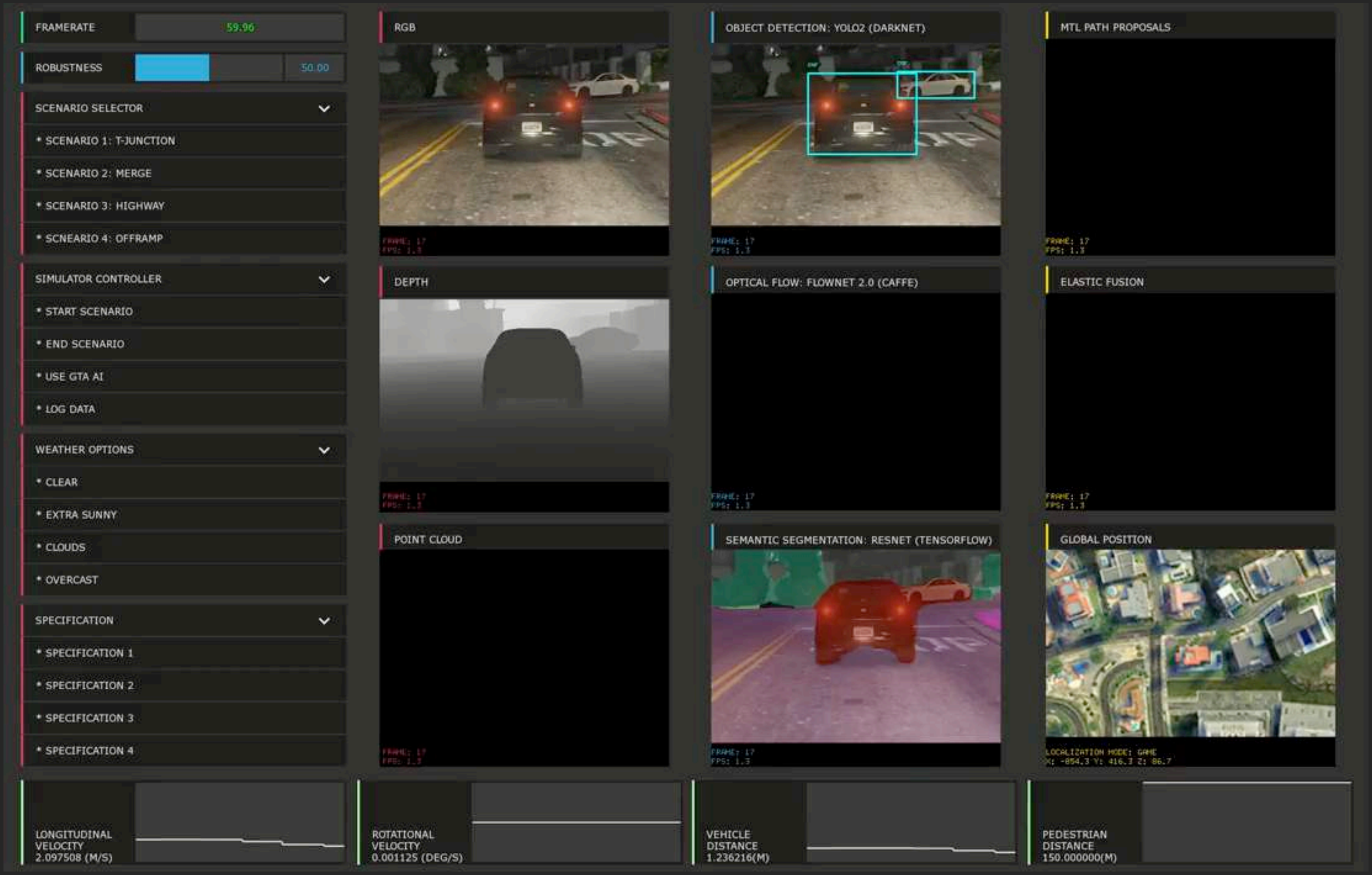
- Uses AlexNet architecture….
- 8 hidden layers…
- Maps to steering angle and throttle input.
- Very similar to NVIDIA DAVE2 project.

Grand
theft
Apex

AVCAD Toolchain - Testing in Synthetic Worlds


TEST HARNESS: SIMULATOR, AV CODE and TESTBENCH (Section II)
WORLD SIMULATOR (Here, GTA)
ZMQ-BASED INTERFACE
ACTUATION (here, steering and acceleration)
FRAMES and VEHICLE STATE
THIRD PARTY AUTONOMOUS VEHICLE CODE BLACK BOX
(here, object detector, navigation logic for city intersections, and GTA trajectory tracker)
TESTBENCH: AUTOMATIC SEARCH FOR DANGEROUS DRIVING SITUATIONS
TRAJECTORIES OF ALL TRAFFIC PARTICIPANTS
NEXT INITIALIZATION x_k
APPLICATION-SPECIFIC STUDY OF SIMULATION VALIDITY (Section III.A)
FRAME SEQUENCES FROM SIMULATOR
FRAME SEQUENCES FROM RELEVANT NATURAL DATASETS (here, KITTI and Udacity)
CV ALGORITHM (here, YOLO object detector)
CV ALGORITHM
GROUND TRUTH
DETECTION RESULTS
COMPUTE CV PERFORMANCE (here, ROCs)
COMPUTE CV PERFORMANCE
PERFORMANCE COMPARISONS (here, compare ROCs)
GENERAL STUDY OF SIMULATED DATA VALIDITY (Section III.B)
FRAME SEQUENCES FROM SIMULATOR
FRAME SEQUENCES FROM RELEVANT NATURAL DATASETS (here, KITTI and Udacity)
FEATURE CALCULATIONS (here, Structural Information and Colorfullness)
FEATURE CALCULATIONS
COMPARE COMPLEXITIES
DARMSTADT 19960 images
UDACITY 12000 images
Colorfulness
Spatial Information

AVCAD: **Robust Testing** Interface



FRAMERATE 59.96

ROBUSTNESS 50.00

FRAME: 17
FPS: 1.3

LOCALIZATION MODE: GAME
X: -854.3 Y: 416.3 Z: 86.7

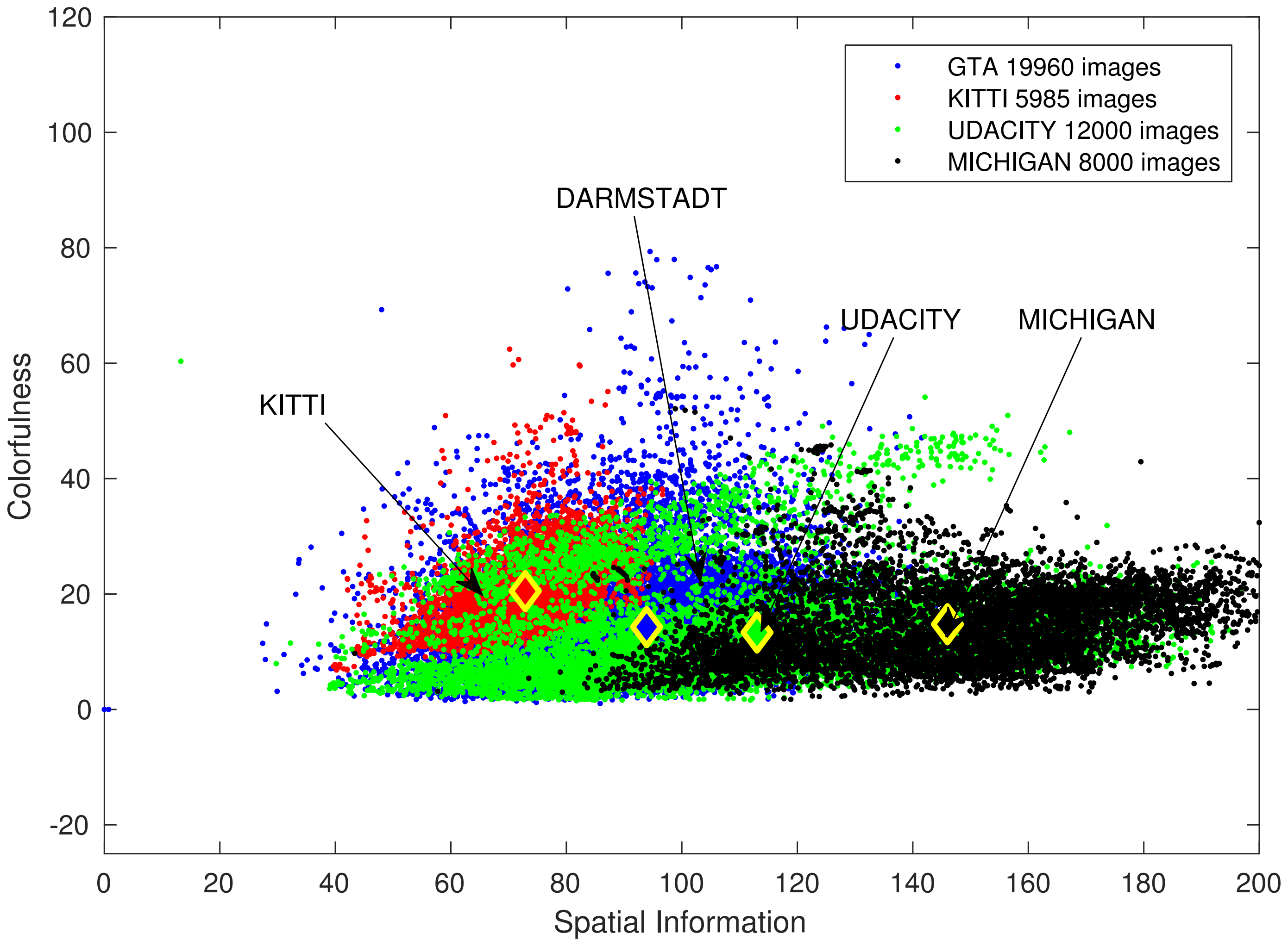
LONGITUDINAL VELOCITY 2.097508 (M/S)

ROTATIONAL VELOCITY 0.001125 (DEG/S)

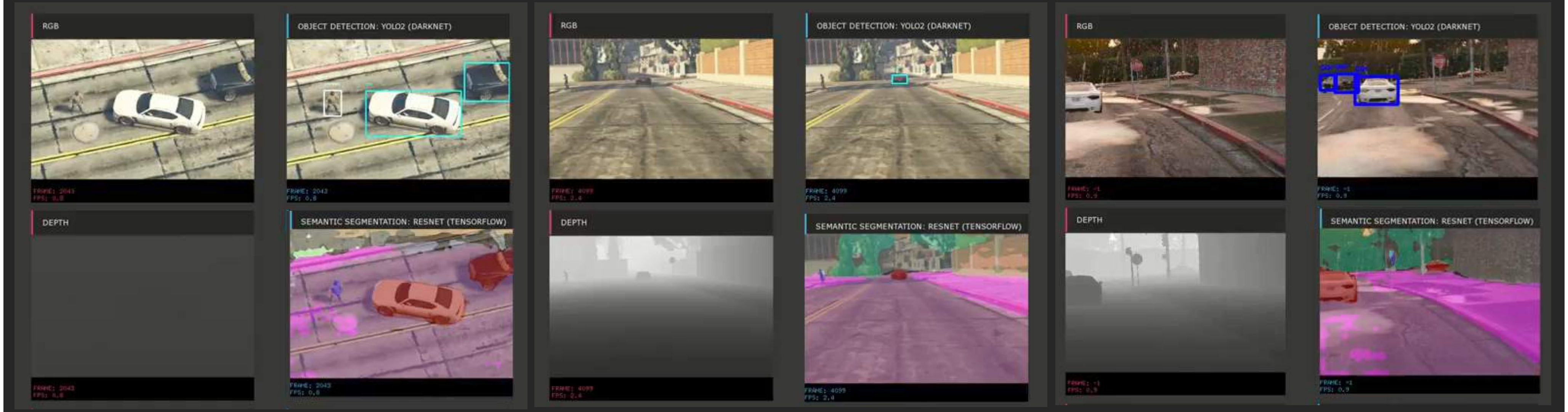
VEHICLE DISTANCE 1.236216(M)

PEDESTRIAN DISTANCE 150.000000(M)

How to Compare Synthetic and Real Images



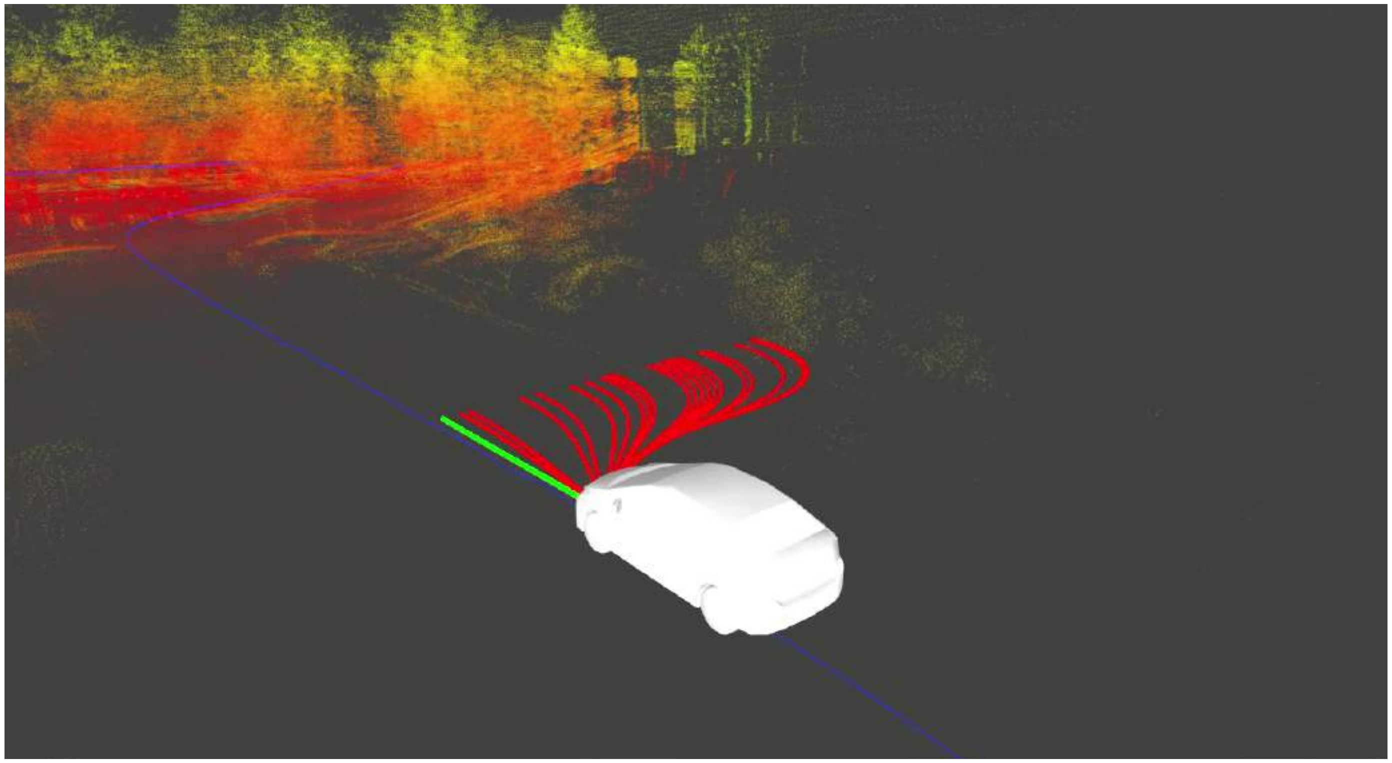
Next Steps: Deep Learning & Vision Based Perception


RGB
OBJECT DETECTION: YOLO2 (DARKNET)
DEPTH
SEMANTIC SEGMENTATION: RESNET (TENSORFLOW)
RGB
OBJECT DETECTION: YOLO2 (DARKNET)
DEPTH
SEMANTIC SEGMENTATION: RESNET (TENSORFLOW)
RGB
OBJECT DETECTION: YOLO2 (DARKNET)
DEPTH
SEMANTIC SEGMENTATION: RESNET (TENSORFLOW)

- Sometimes perception works perfectly, but the controller doesn't know how to handle the scenario, when does this happen, how often?
- Is the system still performant if a key sensor is unable to observe a traffic sign (i.e. it leaves the field of view)?
- How will weather affect the safety of the overall system?
- Modeling, robust testing, and verification give us the tools to address these questions in a meaningful way without building a fleet of vehicles.

Results: Implementation of Trajectory Generator

AutoWare



Integration with ROS and Autoware Open Source Vehicle OS

Use linear optimization to learn weights for a network of radial basis functions. Quickly compute a variety of trajectories in the configuration space of the robot in order to create local plans...

Pennovation Center

A dedicated physical lab for experimental and field-tested ideas

A 23-acre brand new urban campus for Innovation

F1/10 Autonomous Racing

Development of Community Platforms for Safe and Affordable Autonomy

Autonomous Racing

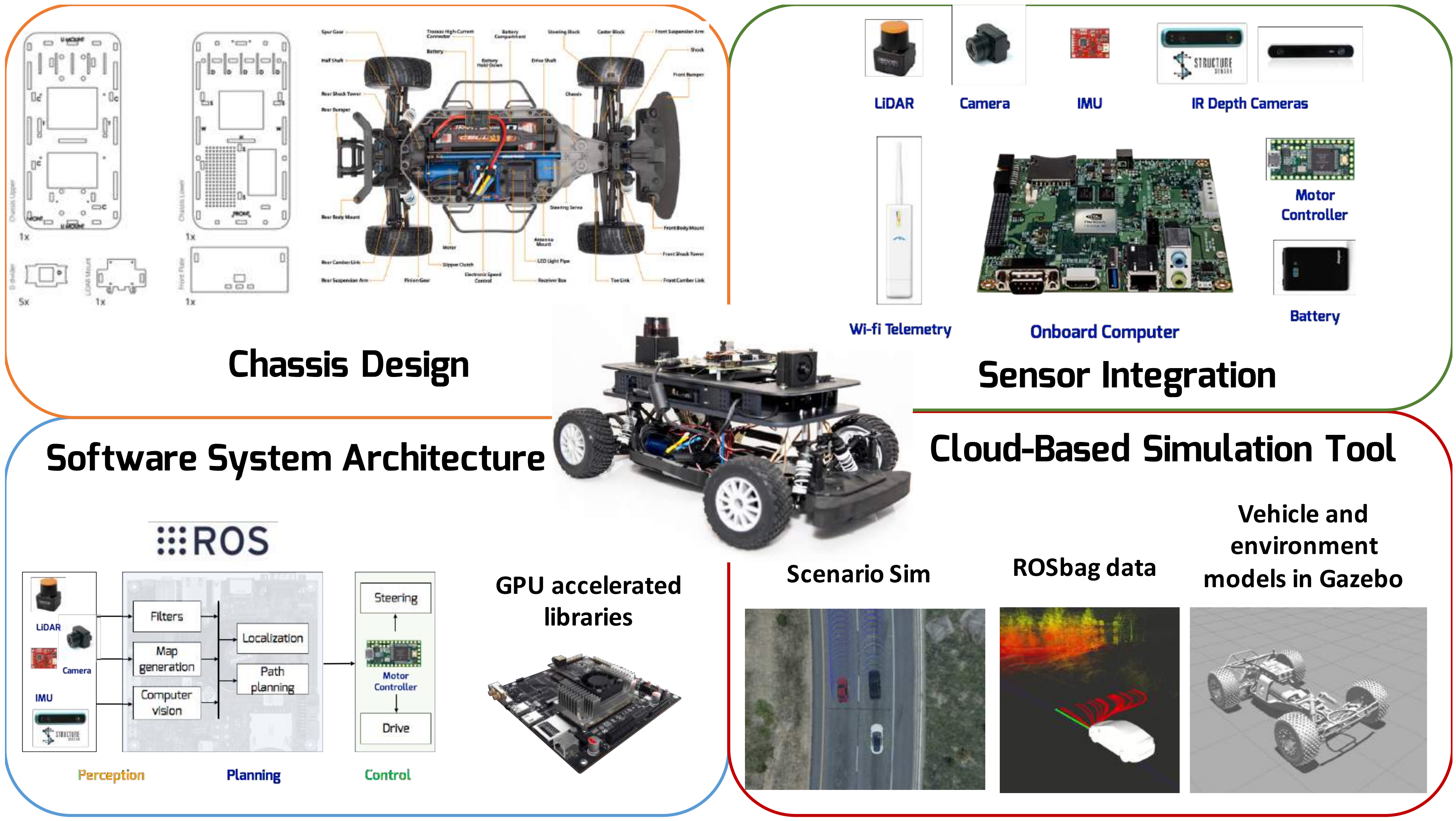
1/10 the scale. 10X the fun!

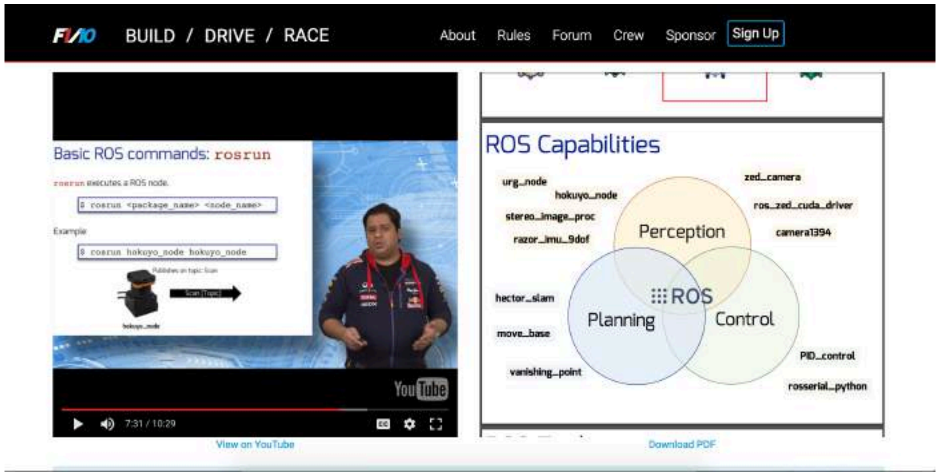
Rahul Mangharam
University of Pennsylvania

Madhur Behl
University of Virginia

Sertac Karaman
MIT

Venkat Krovi
Clemson University


Chassis Design
Sensor Integration
LiDAR
Camera
IMU
IR Depth Cameras
Wi-fi Telemetry
Onboard Computer
Motor Controller
Battery
Software System Architecture
ROS
Filters
Map generation
Computer vision
Localization
Path planning
Steering
Motor Controller
Drive
Perception
Planning
Control
GPU accelerated libraries
Cloud-Based Simulation Tool
Scenario Sim
ROSbag data
Vehicle and environment models in Gazebo



f1tenth.org: Video Tutorials, lectures, and code walkthroughs

MIT Beaver Works Summer Institute – 24 schools, 46 students

Highlights from the 2016 F1/10 Racing Competition

Courses and hackathons

“Essentially all models are wrong, but some are useful”

- George E.P. Box