Towards Verified Artificial Intelligence

Sanjit A. Seshia

Professor

EECS Department, UC Berkeley

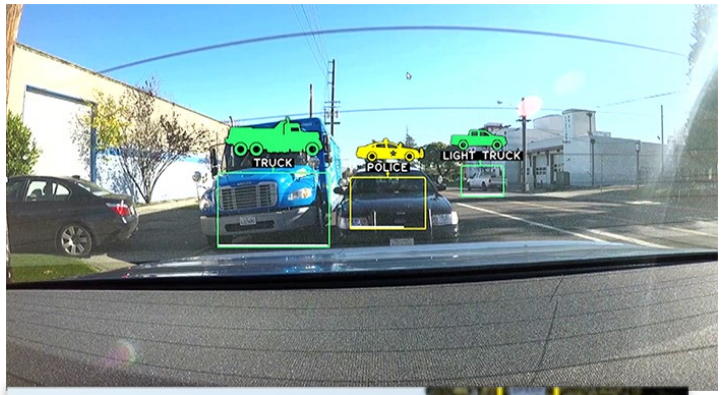
Joint work with:

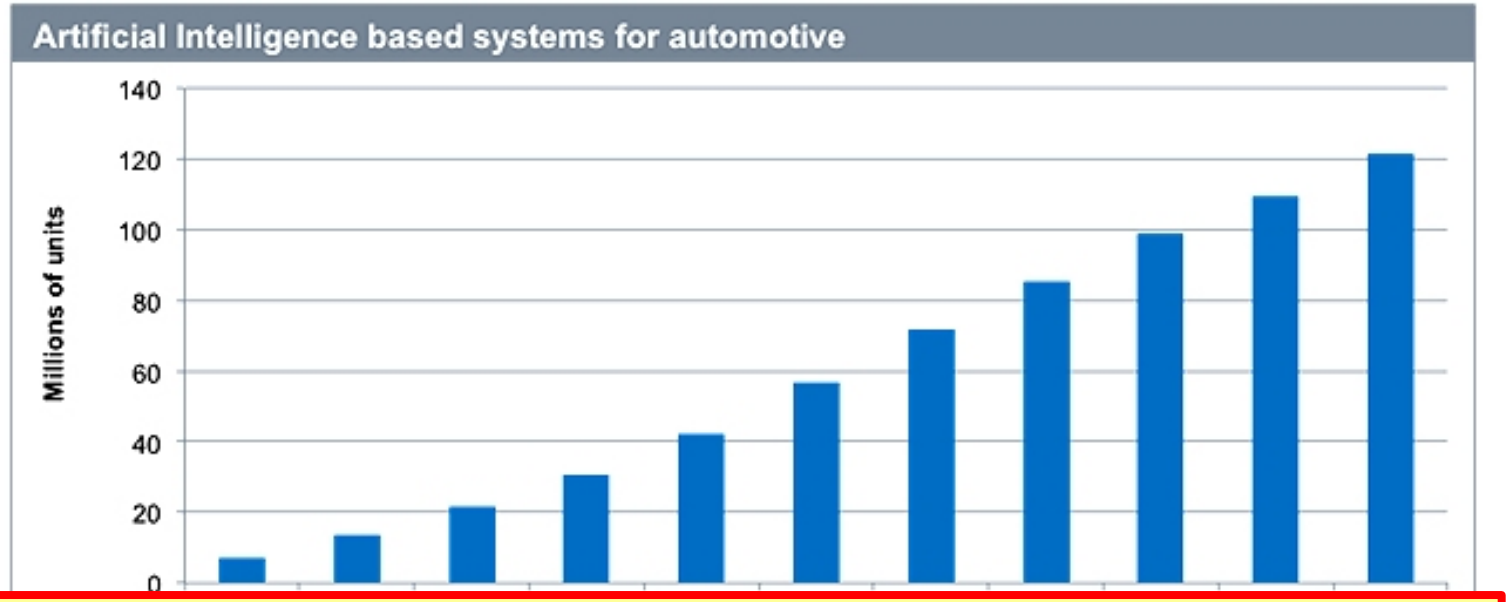
Ankush Desai, Tommaso Dreossi, Daniel Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, Xiangyu Yue, Dorsa Sadigh, Wenchao Li, Alexandre Donze, Alberto Sangiovanni-Vincentelli, S. Shankar Sastry, Natarajan Shankar, Ashish Tiwari

MLSE International Symposium

Tokyo, October 18, 2019

Growing Use of Machine Learning/Artificial Intelligence in Safety-Critical Cyber-Physical Systems





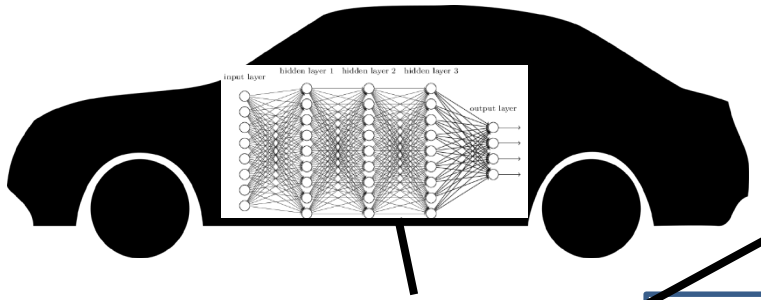
How do we make AI/ML-based Autonomous Systems Safe & Dependable?

Growing concerns about safety!

- Numerous papers showing that *Deep Neural Networks can be easily fooled*
- *Fatal accidents* involving potential failure of object detection/classification systems in self-driving cars

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry.
Towards Verified Artificial Intelligence. July 2016. https://arxiv.org/abs/1606.08514.



System **S**

Environment **E**

Specification **φ**

Does S || E satisfy φ?

YES [+ proof]

NO [+ counterexample]

Need to Search Very High-Dimensional Input and State Spaces

Design Correct-by-Construction?

Need Design Principles for Verified AI

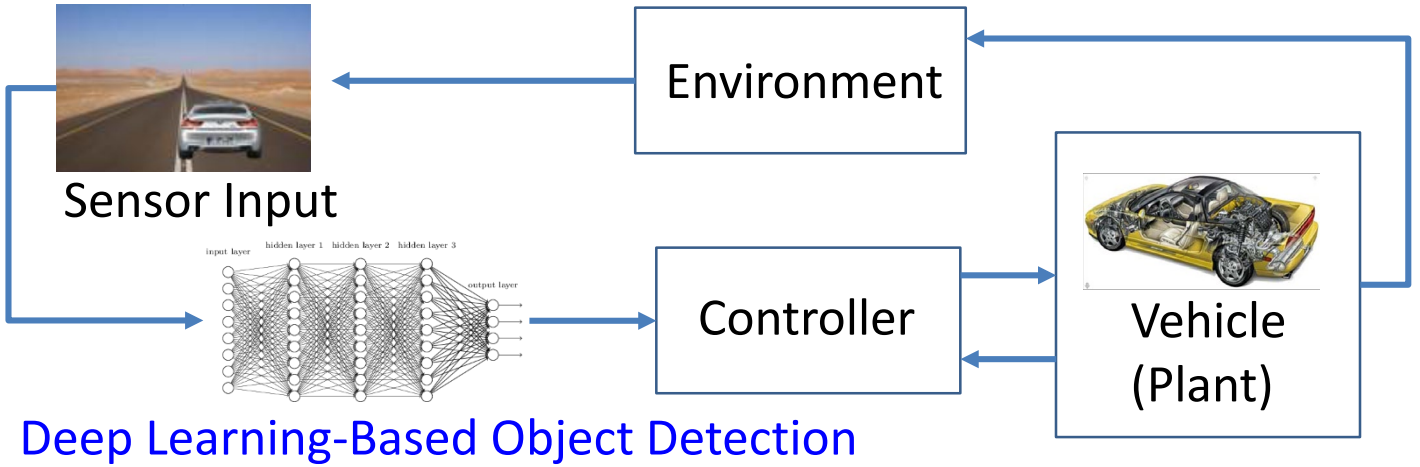
Challenges	Principles
1. Environment (incl. Human) Modeling →	
2. Formal Specification →	
3. Learning Systems Representation →	**Verified AI Project** **VerifAI software toolkit**
4. Scalable Training, Testing, Verification →	https://github.com/BerkeleyLearnVerify/VerifAI
5. Design for Correctness →	

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.

Outline

- Motivating Example
- Specification/Requirements
- Verification: Compositionality and Abstraction
- Environment Modeling – ODDs, Test Scenarios, etc.
- Synthesis: Data-Set Design, Run-Time Assurance, etc.
- The VerifAI toolkit & Conclusion

Example: Automatic Emergency Braking System (AEBS) using Deep Learning for Perception



- Goal: Brake when an obstacle is near, to maintain a minimum safety distance
- Models: Controller, Plant, Env modeled in a software-in-the-loop simulator
  - Matlab/Simulink, Udacity, Webots, CARLA, …
- Perception: Object detection/classification system based on deep neural networks
  - Inception-v3, AlexNet, … trained on ImageNet
  - squeezeDet, Yolo, … trained on KITTI

[Dreossi, Donze, Seshia, "Compositional Falsification of Cyber-Physical Systems with Machine Learning Components", NASA Formal Methods (NFM), May 2017.]

Challenge: Formal Specification

Principle: Start at the System Level (i.e. Specify Semantic Behavior of the Overall System)

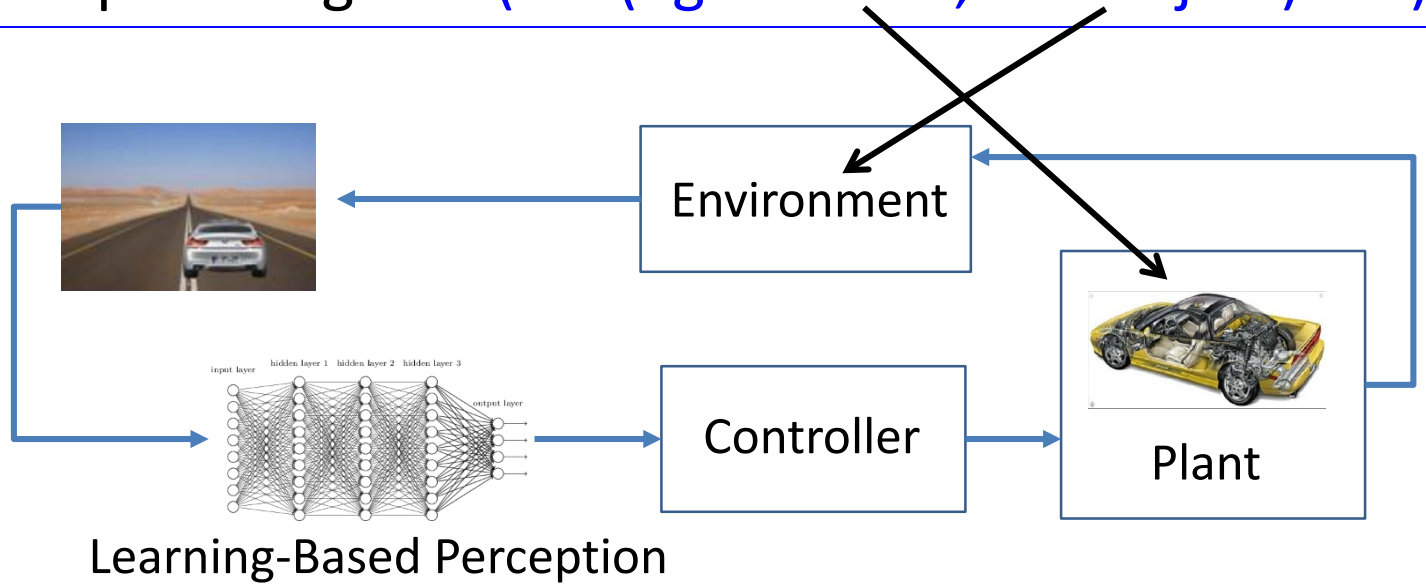
Our Approach: Start with a System-Level Specification

✗ "Verify the Deep Neural Network Object Detector"

✓ "Verify the System containing the Deep Neural Network"

Formally Specify the *End-to-End Behavior* of the System

Temporal Logic: **G** (*dist*(ego vehicle, env object) > Δ)



Property does not mention inputs/outputs of the neural network

Do We Need to Formally Specify ML Component Requirements?

Sometimes…

1. When Component Specifications are Meaningful
   - ML-based Controllers
   - Semantic Robustness*, Input-Output Relations, Monotonicity, Fairness, etc.

2. For Compositional Analysis
   - Derive component specifications from system-level specifications

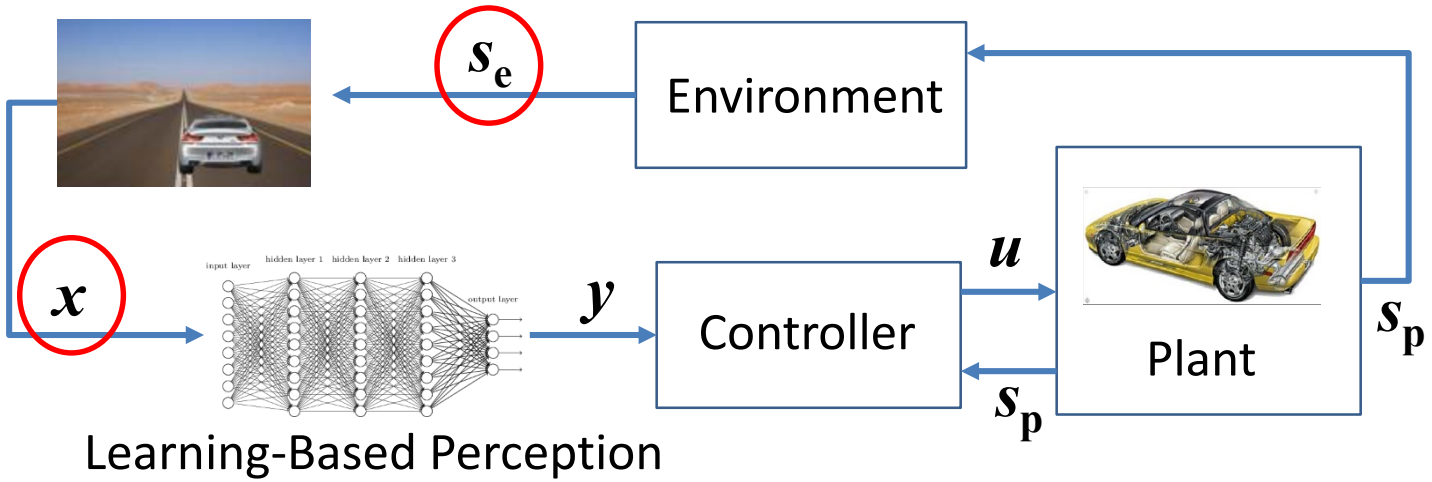
[S. A. Seshia, et al., "Formal Specification for Deep Neural Networks", ATVA 2018.]

Challenge: Scalability of Verification

Principle: Compositional Simulation-Based Verification (Falsification)

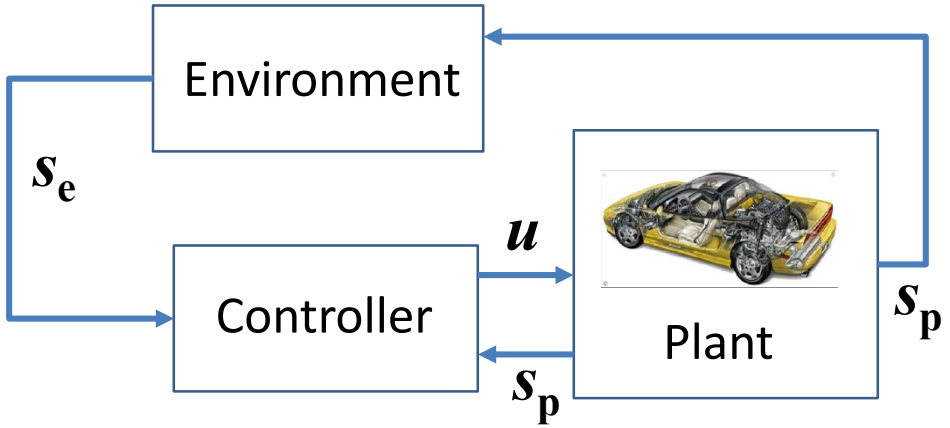
T. Dreossi, A. Donze, and S. A. Seshia. *Compositional Falsification of Cyber-Physical Systems with Machine Learning Components*, In NASA Formal Methods Symposium, May 2017. (Extended version: Journal of Automated Reasoning, 2019.)

Automotive system with ML-based perception (CPSML)



Challenge: $s_e << x$

Traditional closed-loop model (CPS)



Our Approach: Three Key Ideas

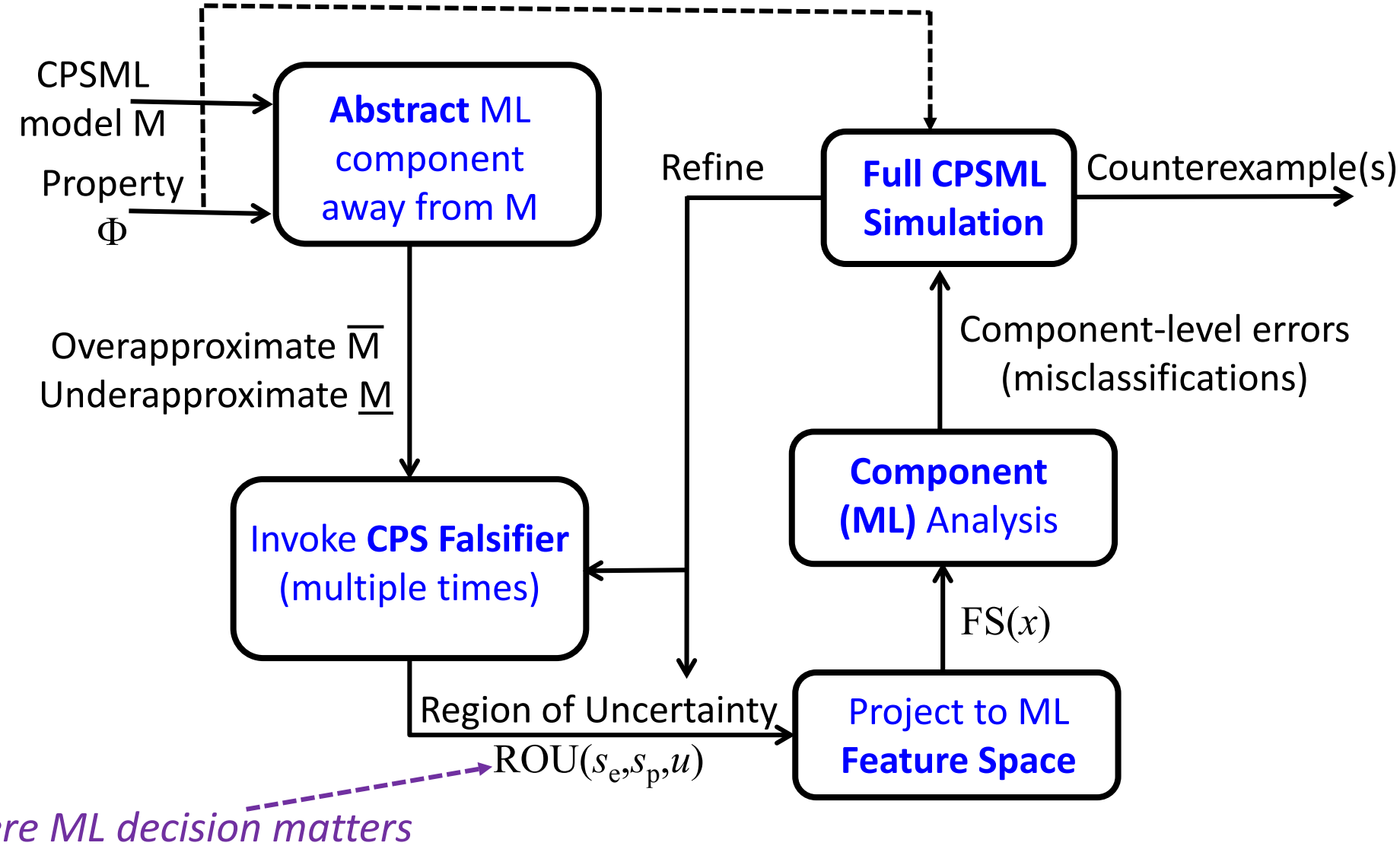
1. Reduce CPSML falsification problem to combination of CPS falsification and ML analysis by *abstraction*

2. Simulation-based *temporal logic falsification* for CPS model
   - Scalable technology already used on production automotive systems (powertrain)

3. *Semantic feature space* analysis of ML component
   - *Derive constraints* on input feature space of neural network (pixels) from semantic constraints on environment model

[Dreossi et al., NFM'17, JAR'19; Dreossi et al., CAV'18]

Compositional Approach: Combine Temporal Logic CPS Falsifier with ML Analyzer

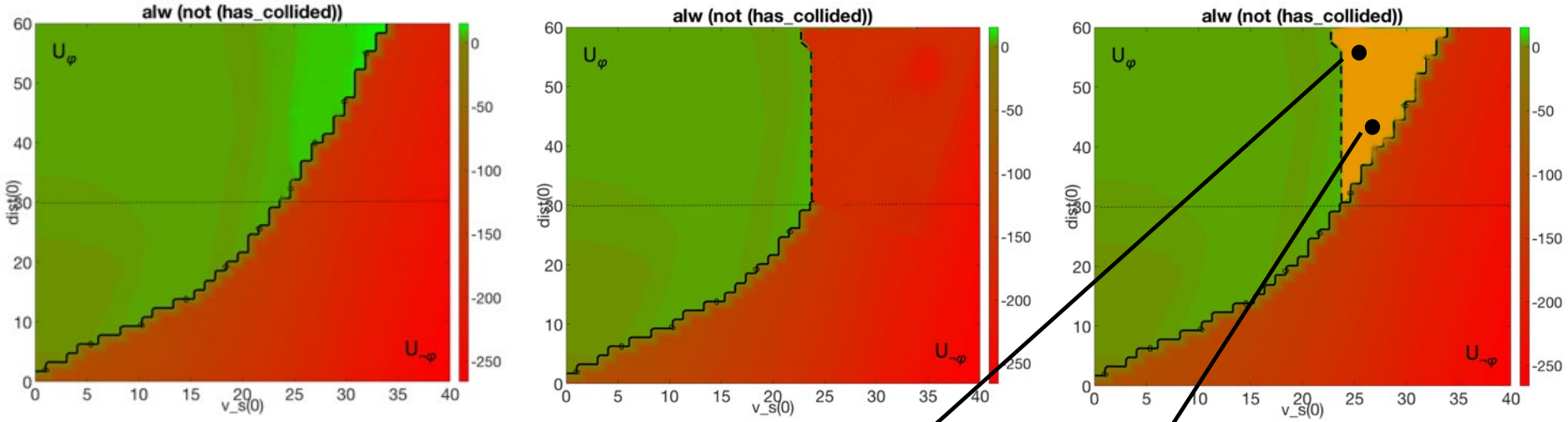


[Dreossi, Donze, Seshia, NFM 2017]

Identifying Component-Level Input Constraints (ROU) for Automatic Emergency Braking System

Green → environments where system-level safety property is satisfied

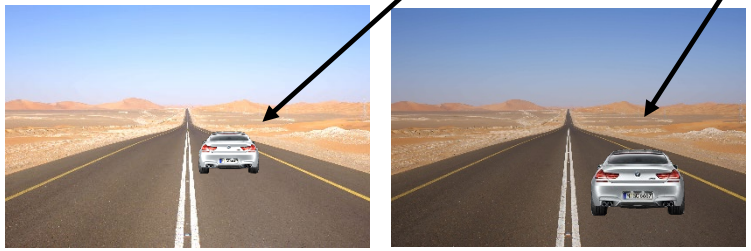
Underapproximate $\underline{M}$ Overapproximate $\overline{M}$



ML always correct

ML always wrong

Potentially unsafe region depending on ML component (yellow)



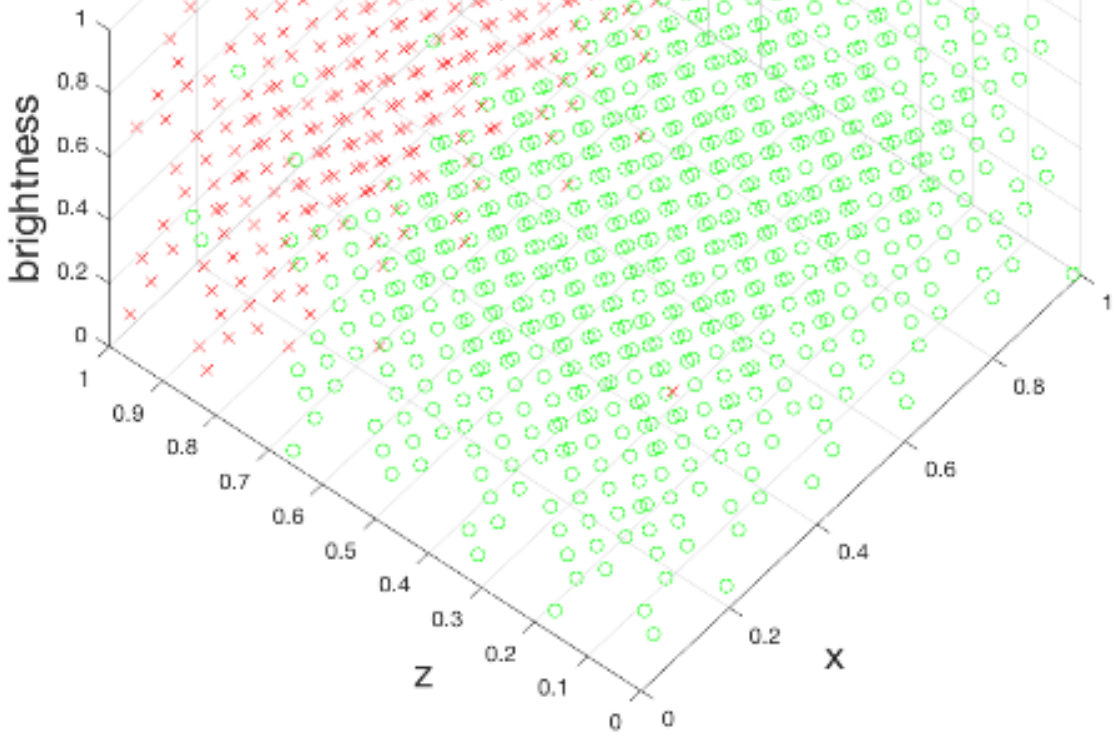
[Dreossi et al., NFM'17, JAR'19; Dreossi et al., CAV'18]

Result on AEBS Example

This misclassification not of concern

Misclassifications

Inception-v3 *Neural Network* *(pre-trained on ImageNet using TensorFlow)*

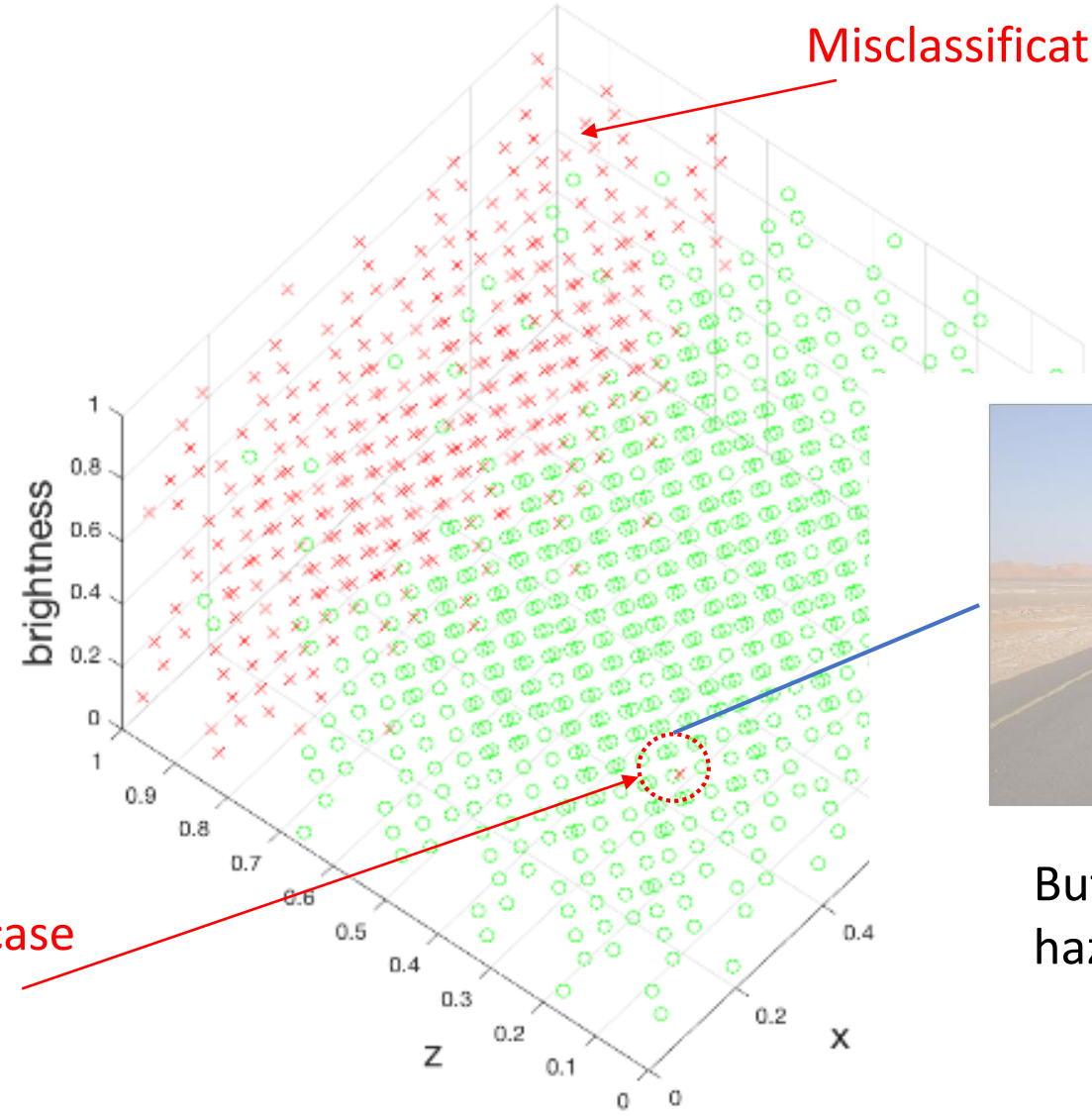


Sample image

[Dreossi, Donze, Seshia, NFM 2017; J. Automated Reasoning 2019]

Result on AEBS Example

Inception-v3 *Neural Network* *(pre-trained on ImageNet using TensorFlow)*



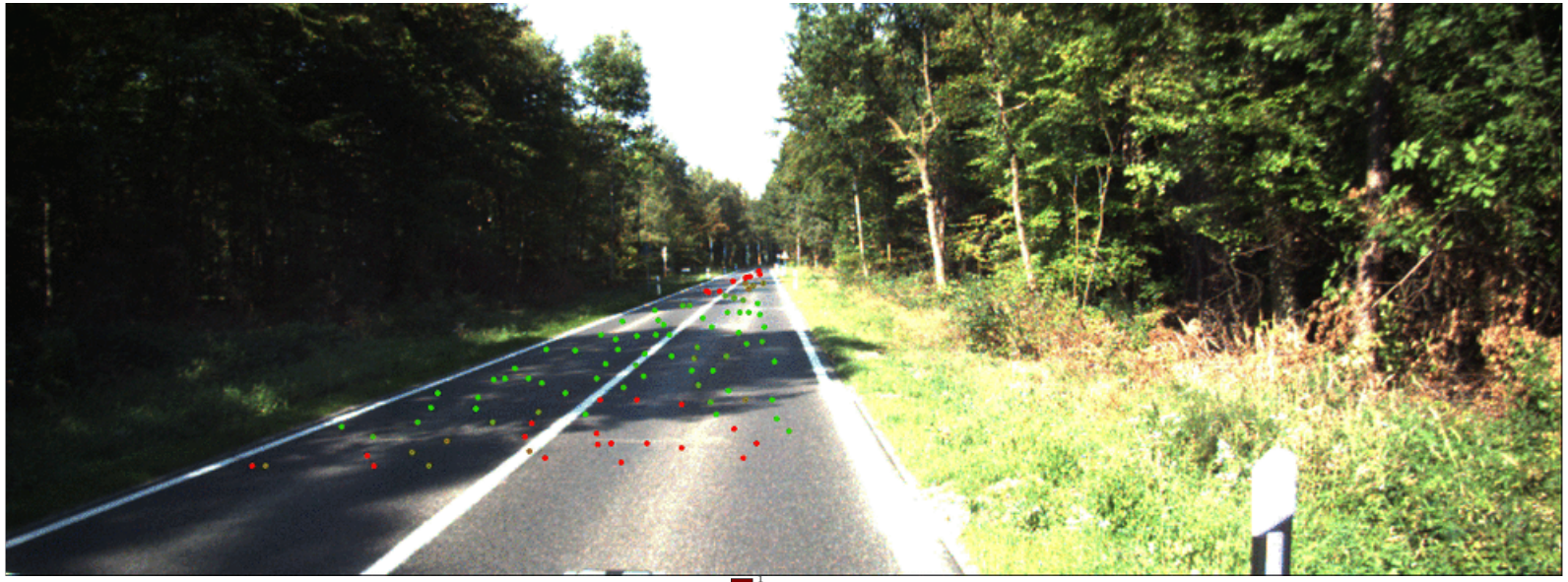
Corner case Image

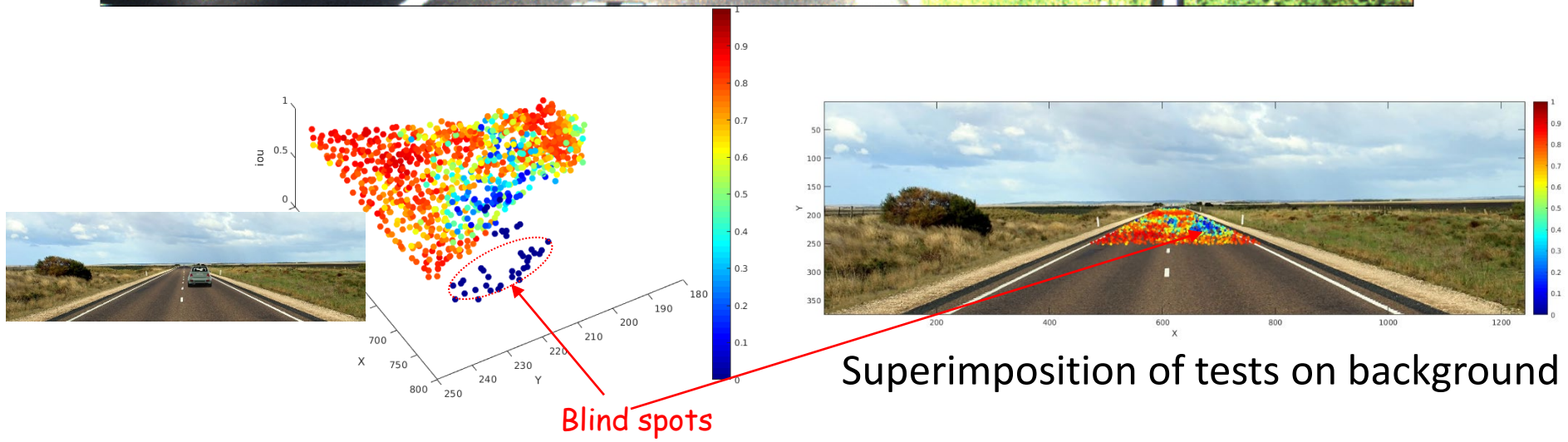
But this one is a real hazard!

[Dreossi, Donze, Seshia, NFM 2017; J. Automated Reasoning 2019]

Extending to Image Streams

[Dreossi, Ghosh, et al., ICML 2017 workshop]





Superimposition of tests on background

Results on squeezeDet NN and KITTI dataset for autonomous driving

Challenge: How to (Re)Design Learning Components

Principle: Use

Oracle-Guided Inductive Synthesis (OGIS)

Standard Machine Learning

LEARNER

labeled/unlabeled data

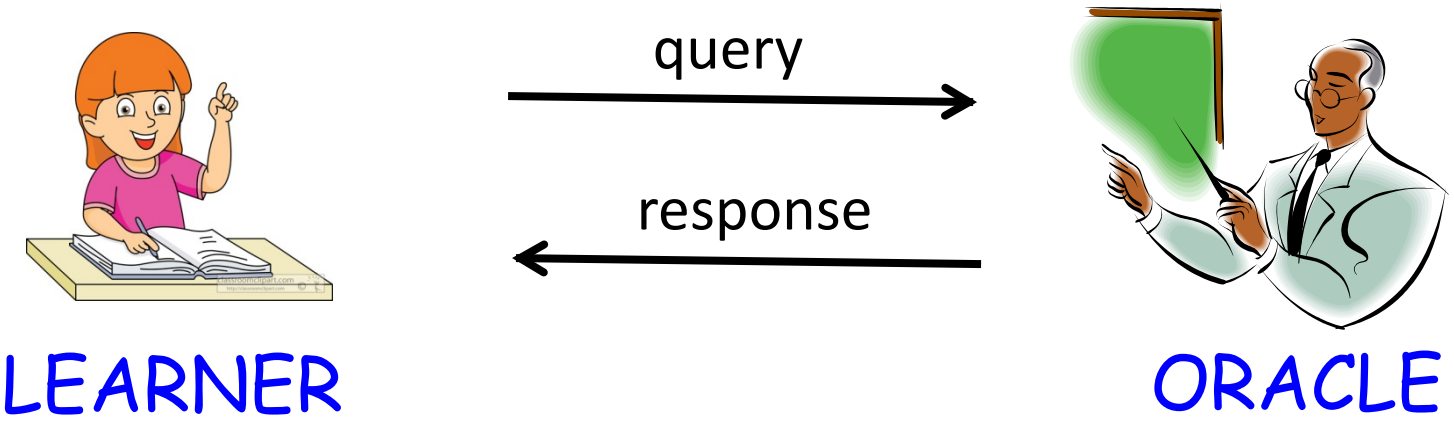
SOURCE OF DATA

Learned Model only as good as the Data!

Correct-by-Construction Design with Formal (Oracle-Guided) Inductive Synthesis/Learning

Key Idea: **Oracle-Guided Learning**

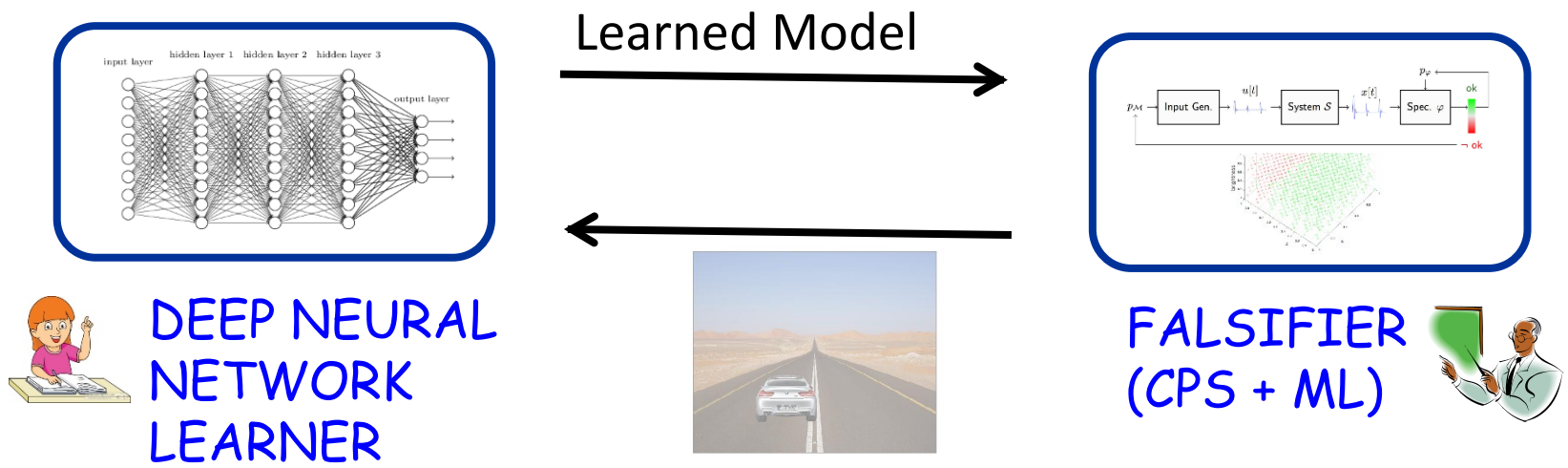
Combine Learner with Oracle (e.g., Verifier) that answers Learner's Queries



[Jha & Seshia, "A Theory of Formal Synthesis via Inductive Learning", 2015, Acta Informatica 2017.]

Counterexample-Guided Training of Deep Neural Networks

- Instance of Oracle-Guided Inductive Synthesis
- Oracle is Verifier (CPSML Falsifier) used to find counterexample inputs to DNN
- Substantially increase accuracy with relatively few additional examples



"Counterexample-Guided Data Augmentation", T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI 2018.

Challenge: Environment Modeling

Principles: Data-Driven, Probabilistic, Introspective, Modeling

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.

SCENIC: Scenario Description Language

- *Scenic* is a probabilistic programming language defining *distributions over scenes*
- *Use cases:* data generation, test generation, verification, debugging, design exploration, etc.

- Example scenario: a badly-parked car

```
from gta import Car, curb, roadDirection

ego = Car

spot = OrientedPoint on visible curb
badAngle = Uniform(1.0, -1.0) * (10, 20) deg
Car left of (spot offset by -0.5 @ 0),
    facing badAngle relative to roadDirection
```

Images created with GTA-V

[D. Fremont et al., "Scenic: A Language for Scenario Specification and Scene Generation", EECS TR March 2018, in PLDI 2019.]

SCENIC: Scenario Description Language

Scenic makes it possible to specify broad scenarios with complex structure, then generate many concrete instances from them automatically:

Platoons

Bumper-to-Bumper Traffic

Use Case: Debugging a Known Failure

Use Case: Debugging a Known Failure

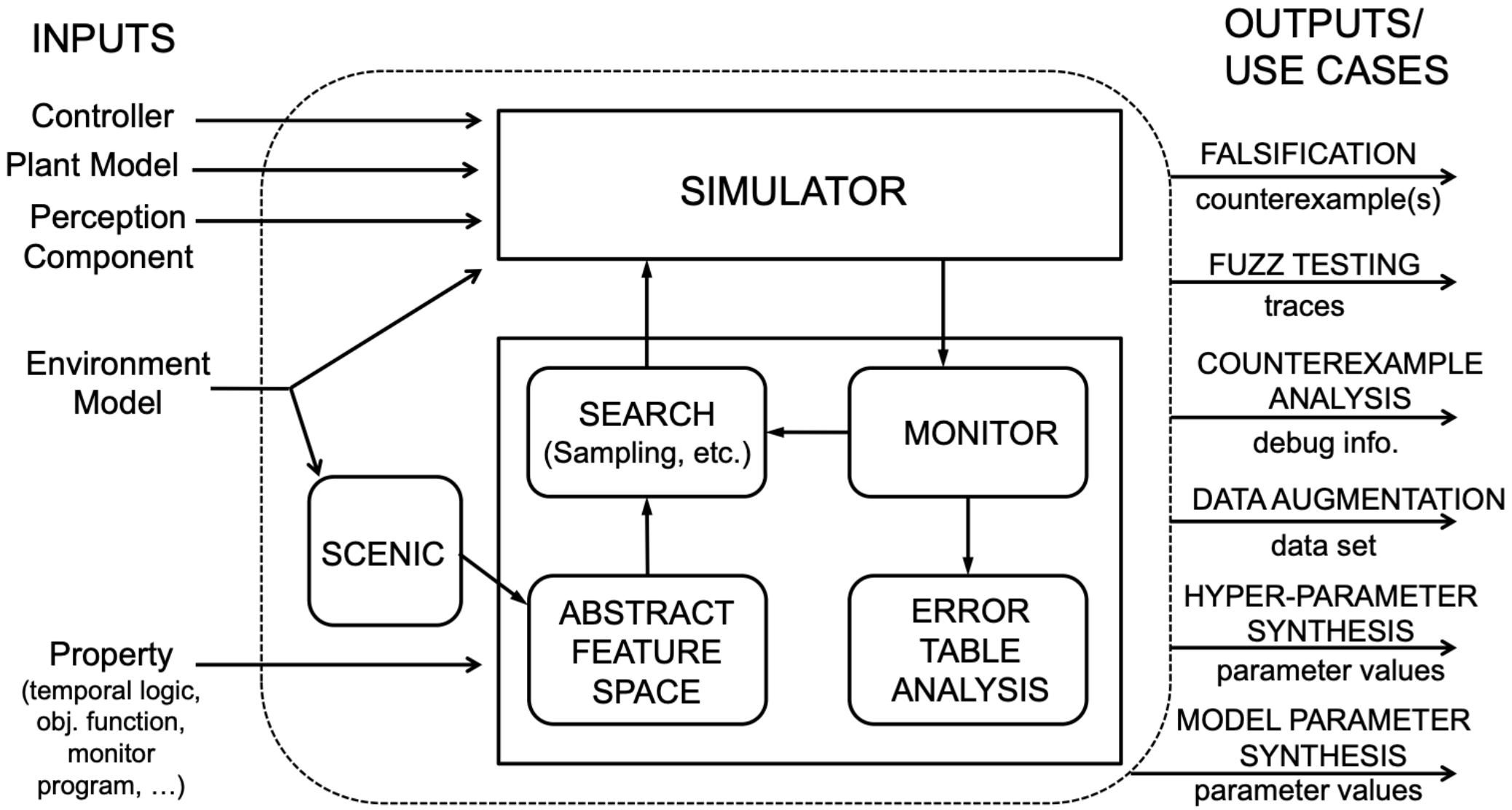
Scenic makes it easy to vary a scenario along different dimensions:

Add noise

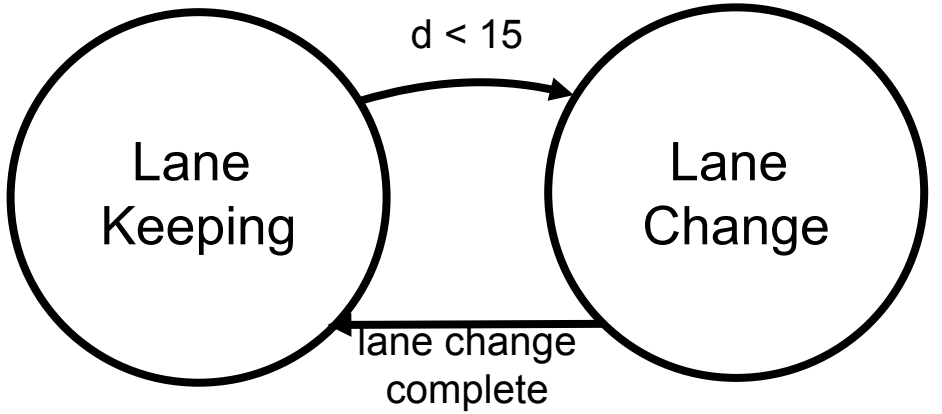
Change car model

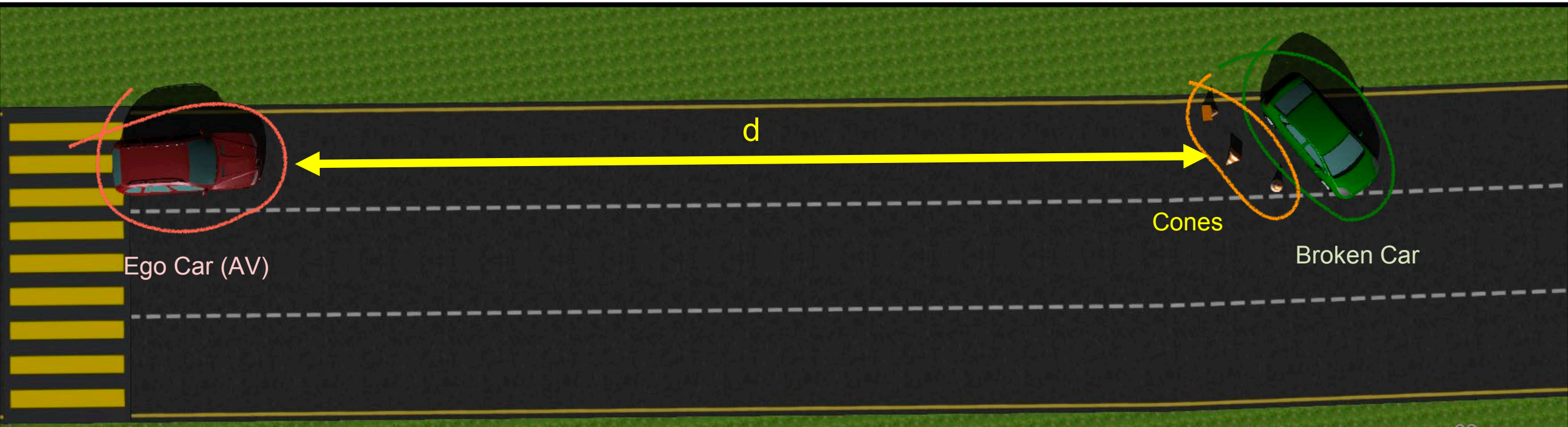
Change global position

VERIFAI: A Toolkit for the Design and Analysis of AI-Based Systems https://github.com/BerkeleyLearnVerify/VerifAI



Case Study for Temporal Logic Falsification with VerifAI: Navigation around an accident scenario



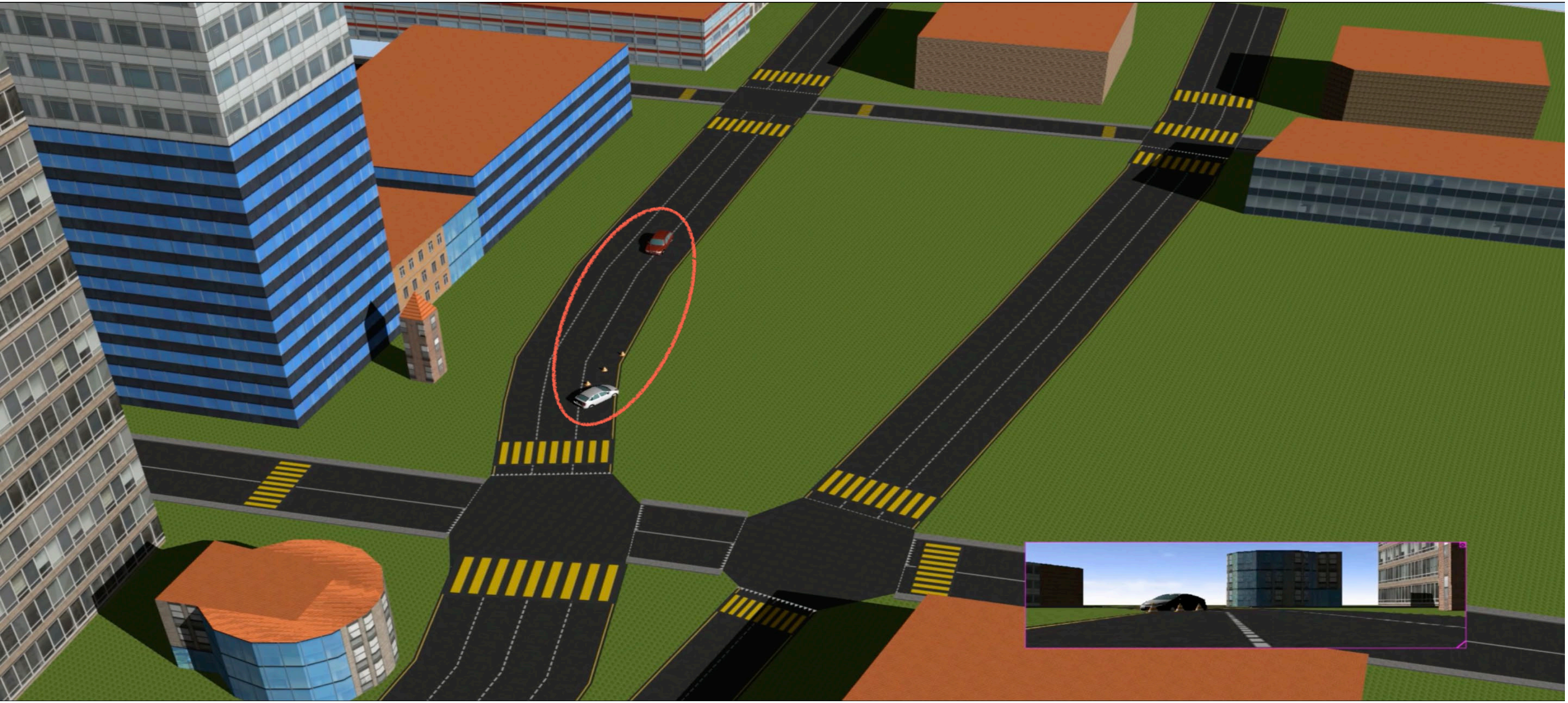


Modeling Case Study in the SCENIC Language

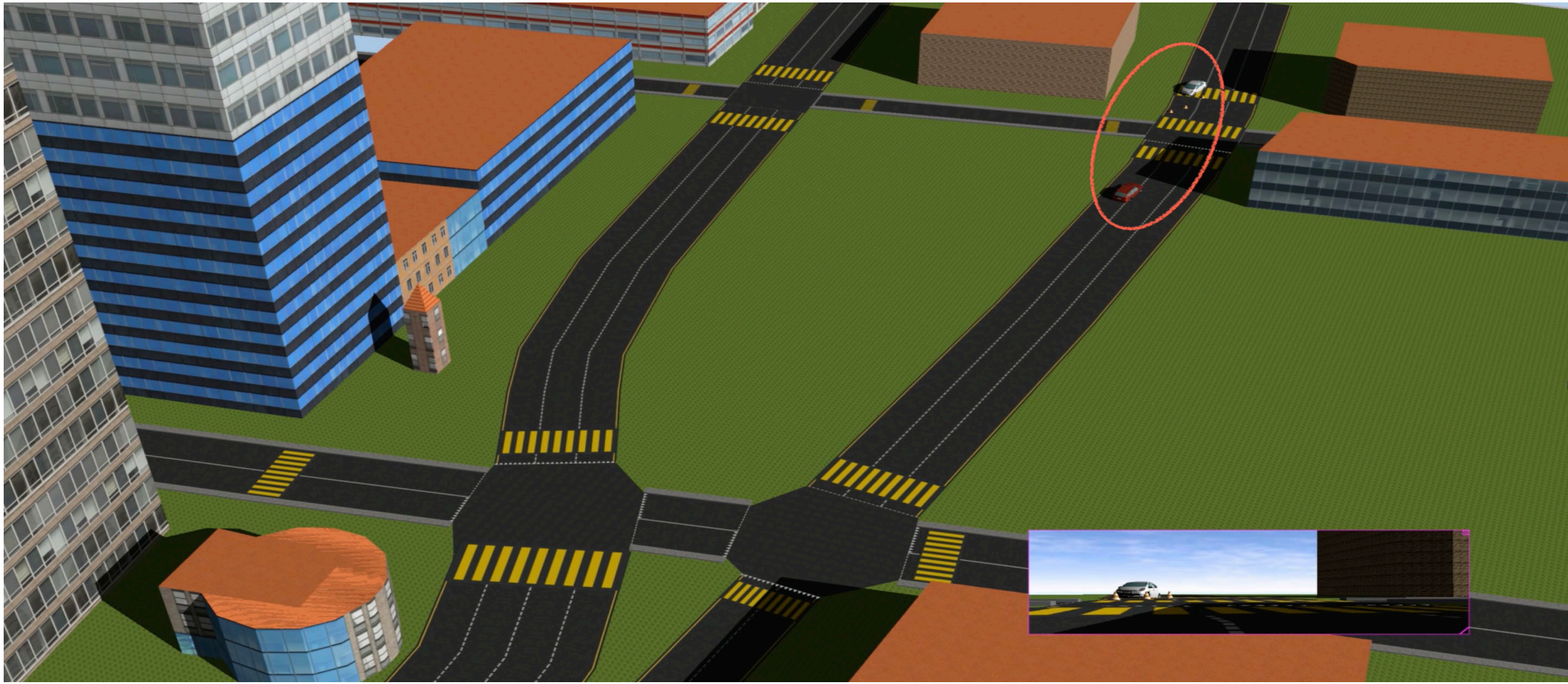
```
# Pick location for blockage randomly along curb
blockageSite = OrientedPoint on curb

# Place traffic cones
spot1 = OrientedPoint left of blockageSite by (0.3, 1)
cone1 = TrafficCone at spot1,
                     facing (0, 360) deg

...

# Place disabled car ahead of cones
SmallCar ahead of spot2 by (-1, 0.5) @ (4, 10),
          facing (0, 360) deg
```

Fremont et al., *Scenic: A Language for Scenario Specification and Scene Generation*, PLDI 2019.

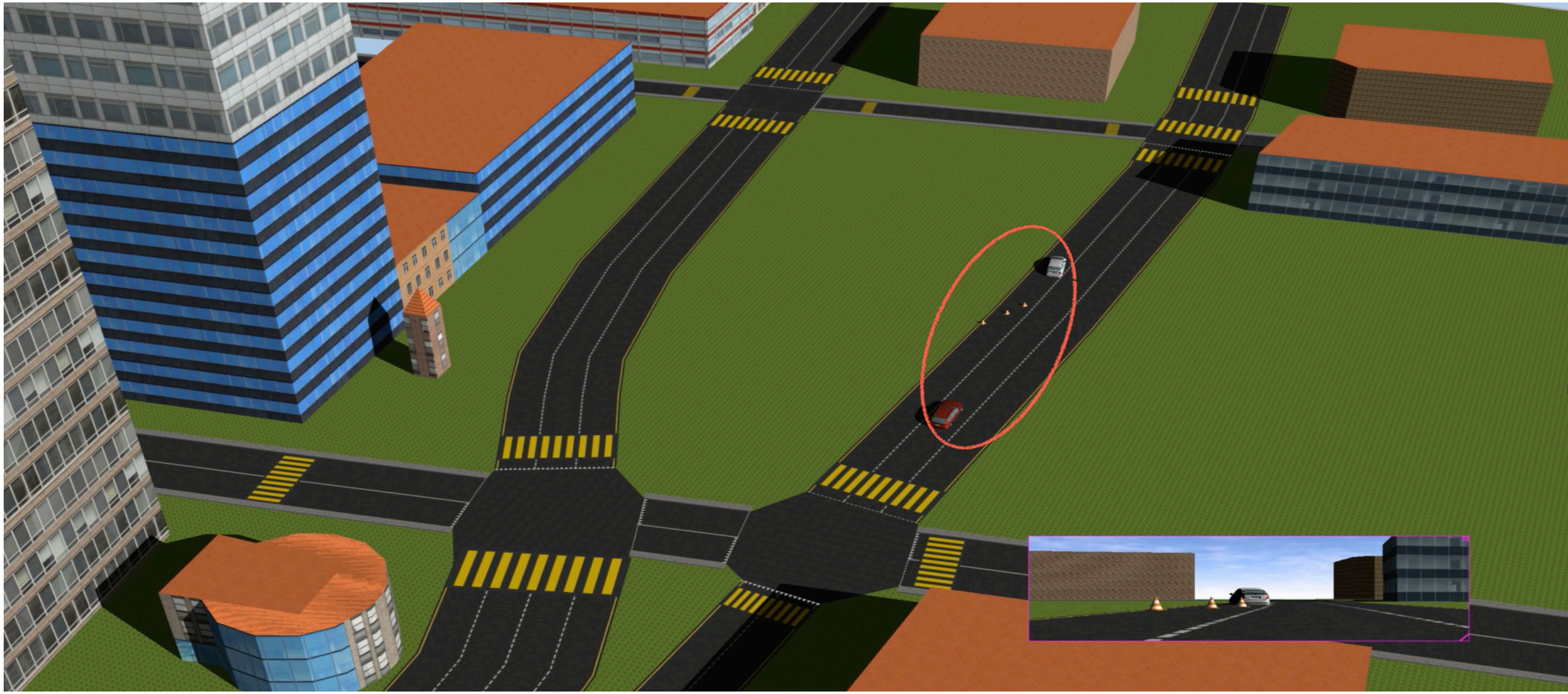
Using Scenic to Generate Initial Scenes



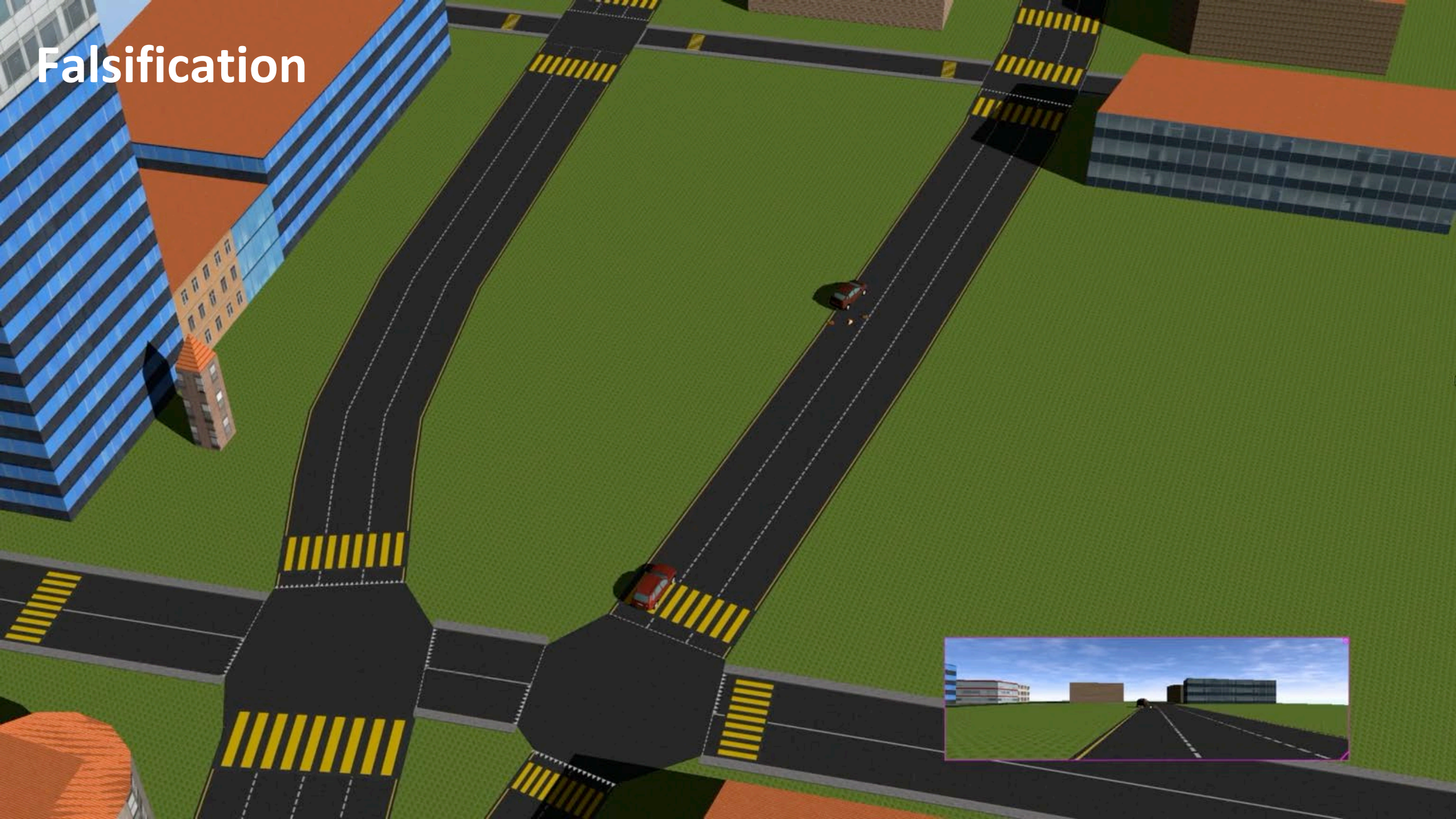
Using Scenic to Generate Initial Scenes



Using Scenic to Generate Initial Scenes



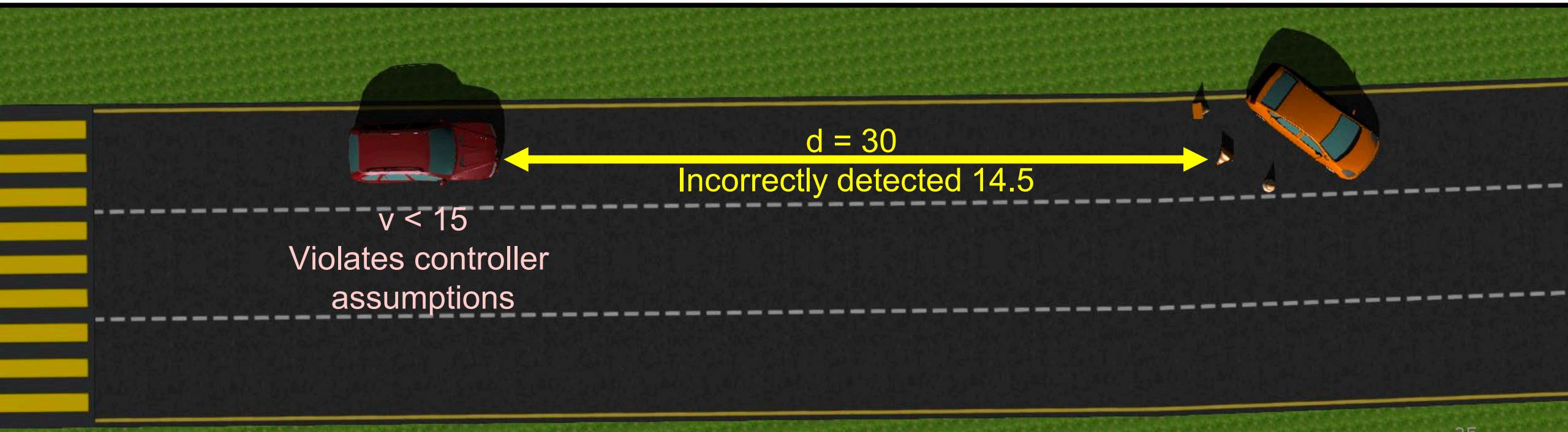
Falsification



Analyzing the failure

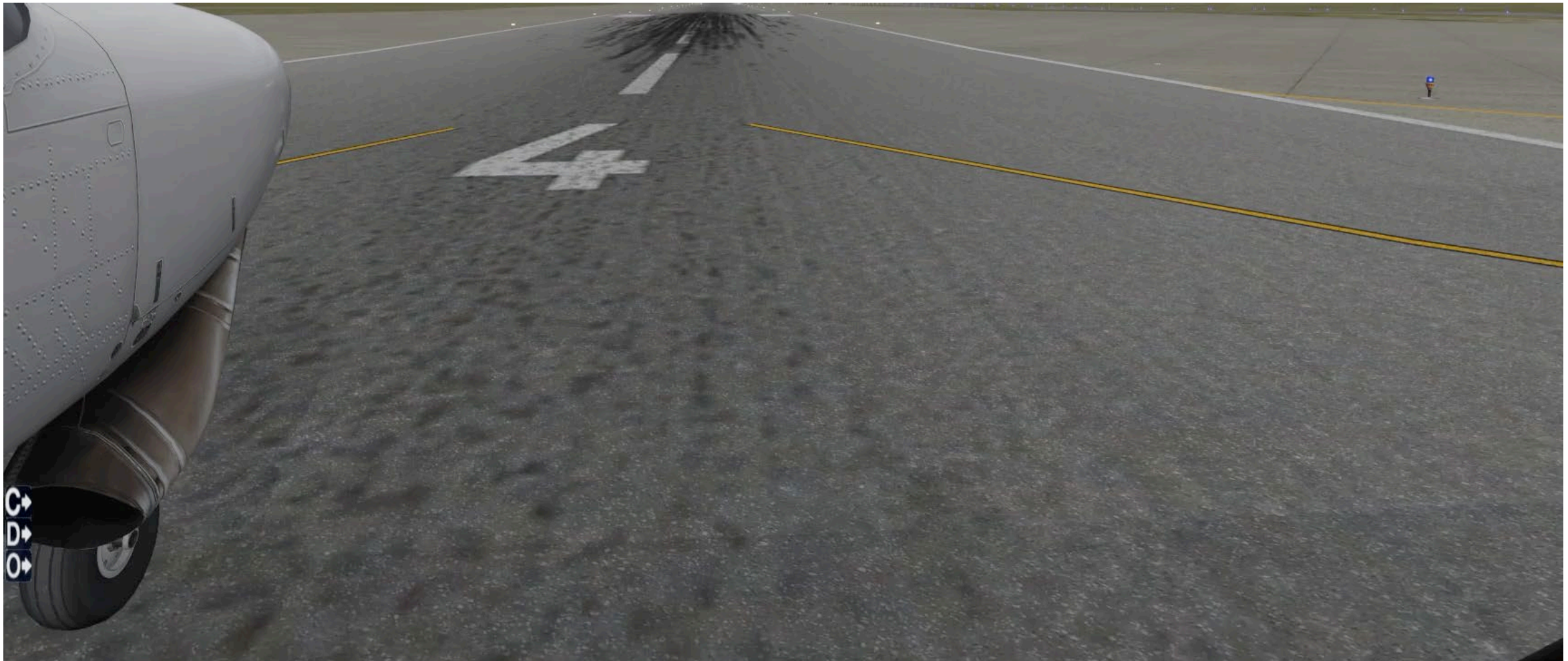
Fix the controller:
Update model assumptions
and re-design controller

Retrain the perception module:
Collect the counter-example images and
retrain the network [IJCAI'18]

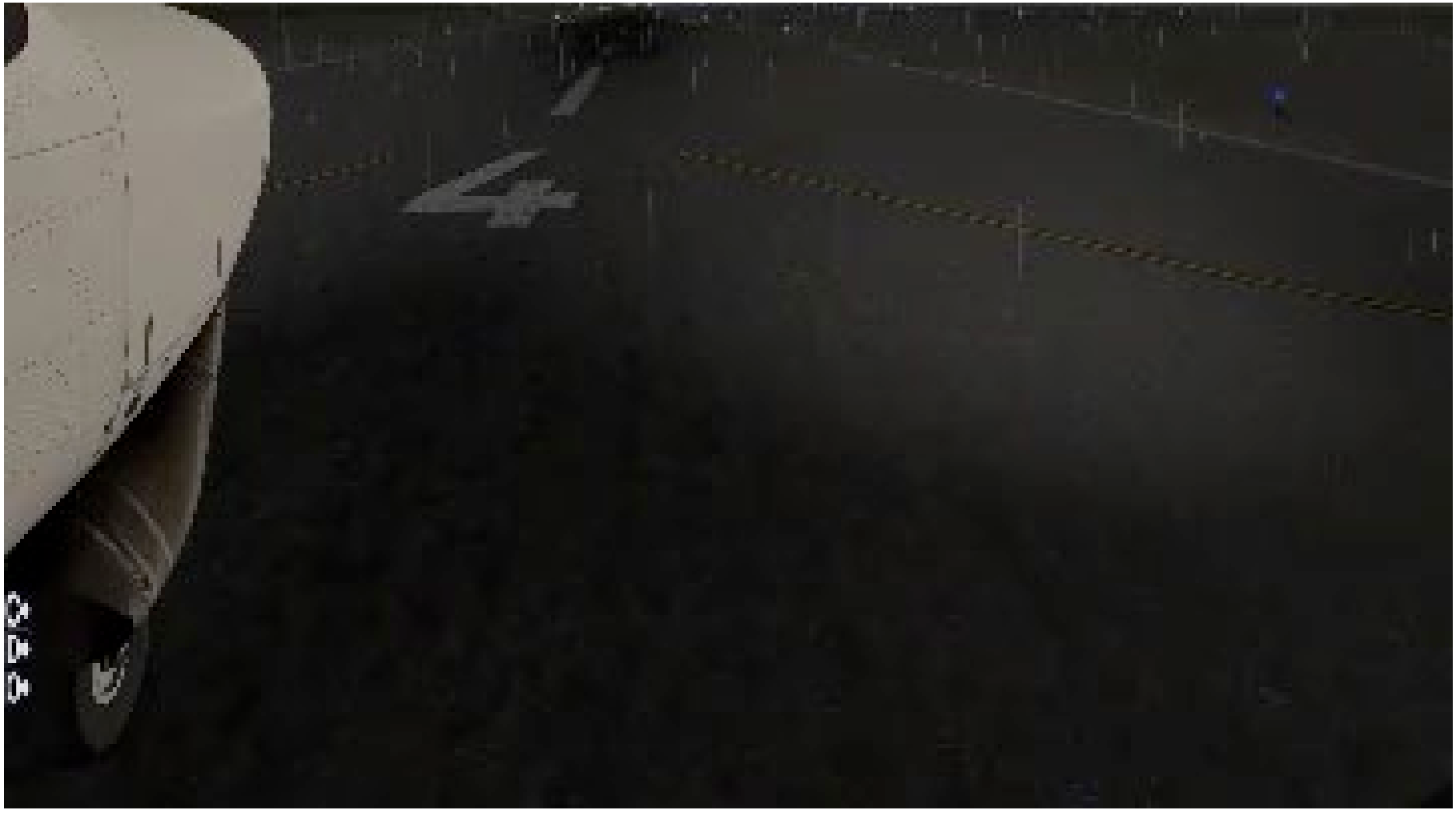


Different Case Study: Automated Taxiing

- NN-based research prototype from Boeing
- Specification: track centerline within 1.5 m



Falsifying Run Found with VerifAI/Scenic



Links and Ongoing Work

- Open-Source Releases: the Verified AI toolkit [CAV 2019]
  - VerifAI: https://github.com/BerkeleyLearnVerify/VerifAI
  - Scenic: https://github.com/BerkeleyLearnVerify/Scenic
  - Operate with any simulator
- Other results/ongoing projects:
  - Bridging Simulation and Real World
    - Domain adaptation to produce “real” data from simulated data
    - Quantifying distance between simulated and real behaviors [HSCC 2019]
  - More Complex Sensors: Video + LiDAR + RADAR + …
  - Counterexample-Guided Retraining/Data Set Design [IJCAI 2018]

Conclusion: Towards Verified AI/ML based CPS

Challenges		Core Principles
1. Environment (incl. Human) Modeling	→	Data-Driven, Introspective, Probabilistic Modeling
2. Specification	→	Start with System-Level Specification, then Component Spec (robustness, …)
3. Learning Systems Complexity	→	Abstraction, Semantic Representation, and Explanations
4. Efficient Training, Testing, Verification	→	Compositional Analysis and Semantics-directed Search/Training
5. Design for Correctness	→	Oracle-Guided Inductive Synthesis; Run-Time Assurance

Exciting Times Ahead!!! Thank you!

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.