

Monitoring of Perception Systems for Certifiable Autonomous Vehicles

Luca Carlone

Associate Professor

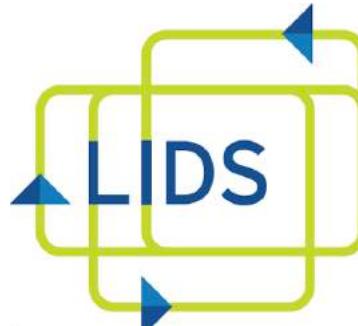
Massachusetts Institute of Technology



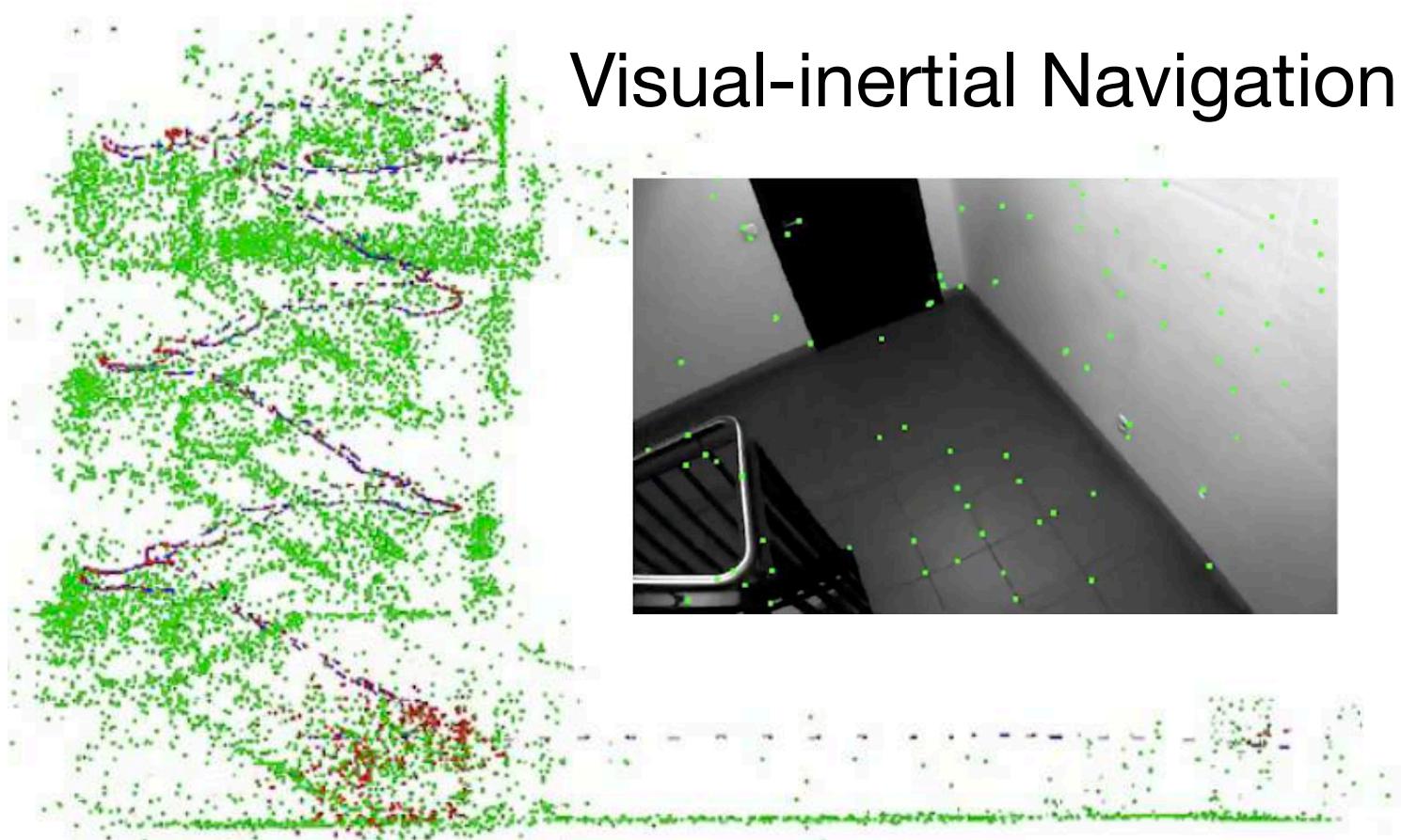
@lucacarlone1



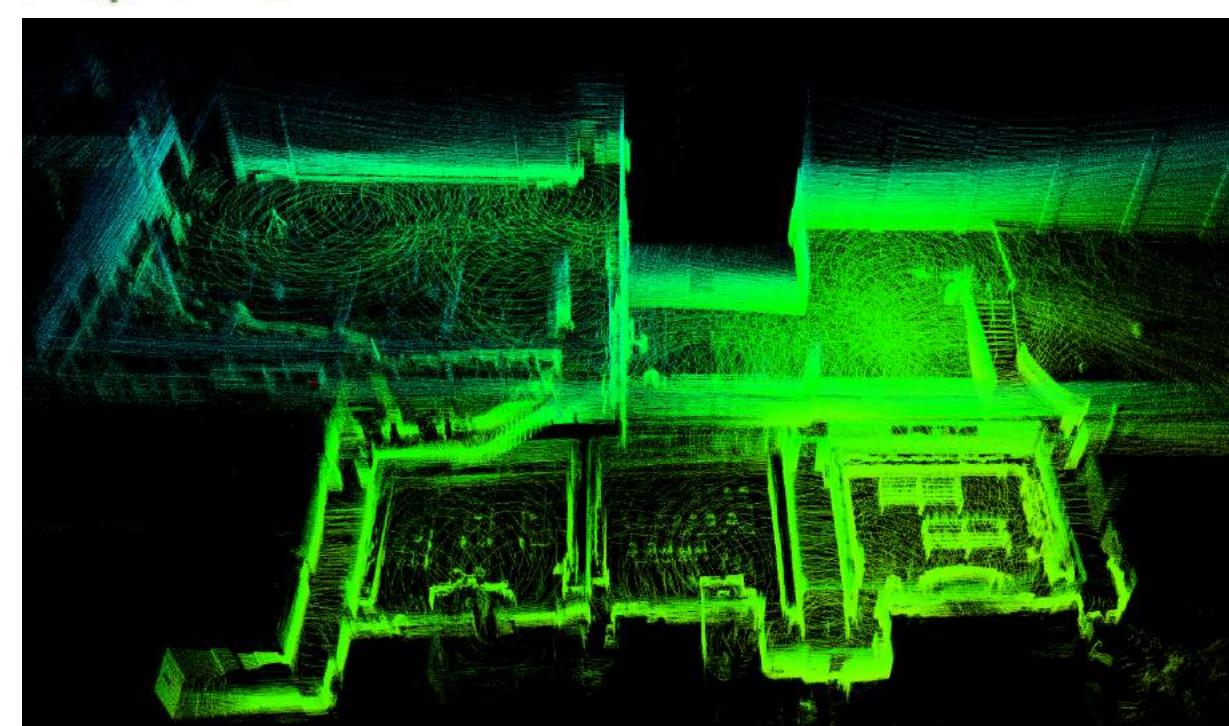
Workshop on Online Map Validation and Road Model Creation



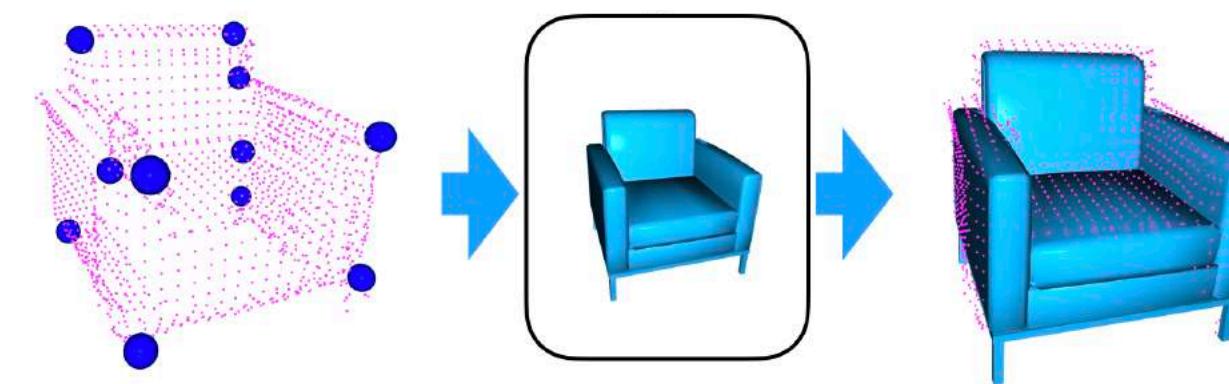
Robust and Certifiable Perception



Lidar-based SLAM

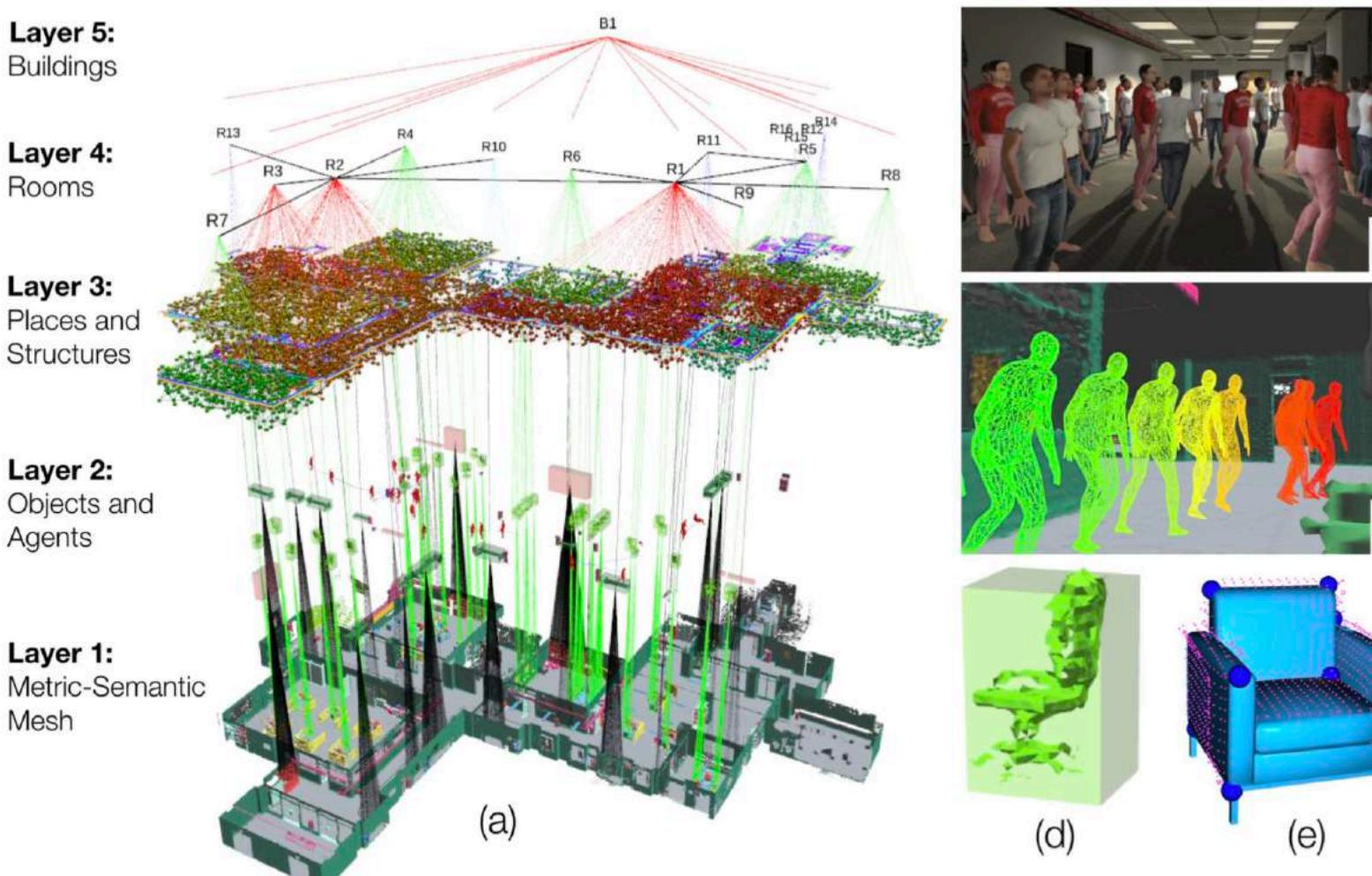
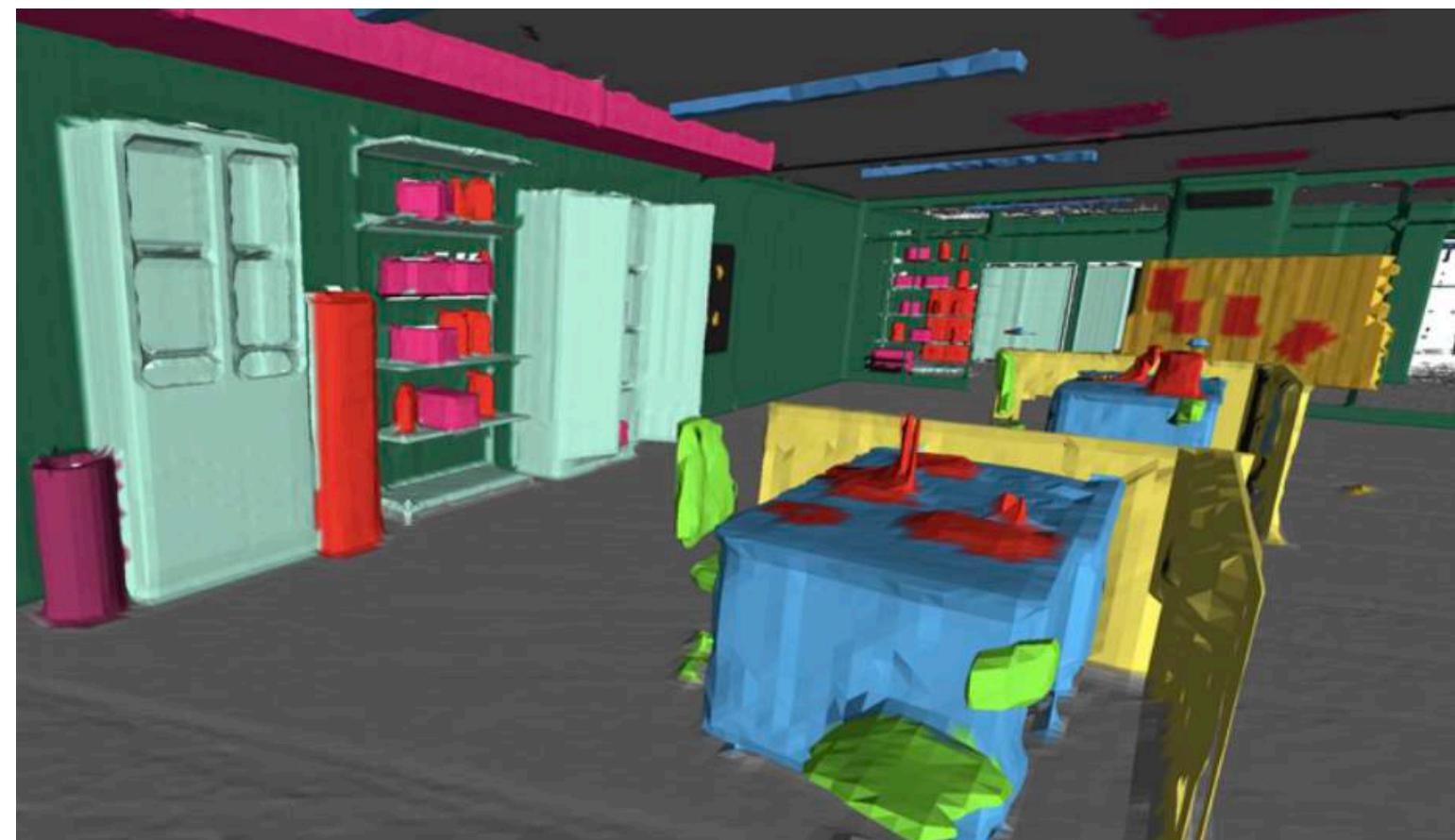


Certifiable Algorithms



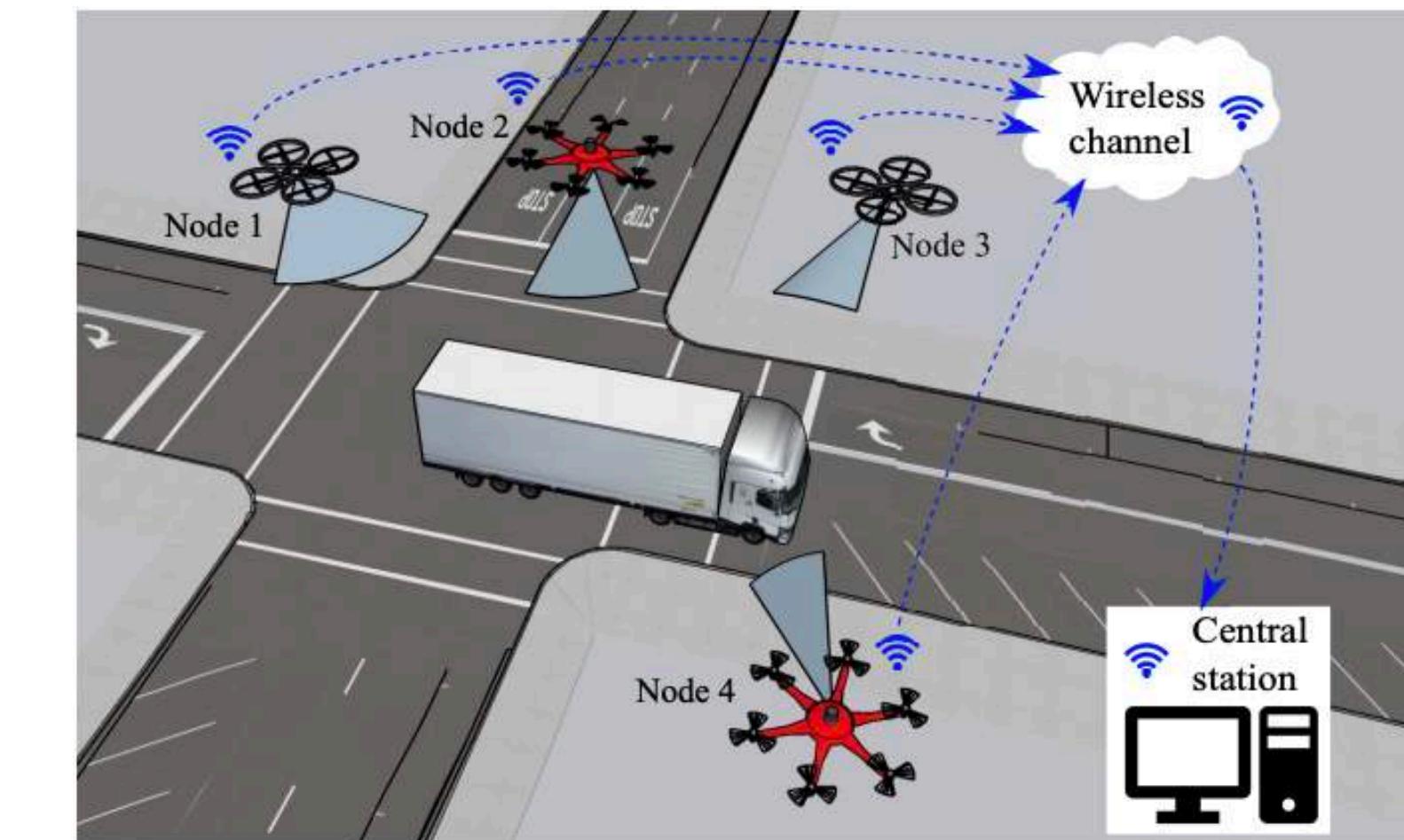
High-level Scene Understanding (Spatial AI)

Kimera: Metrics-semantic SLAM

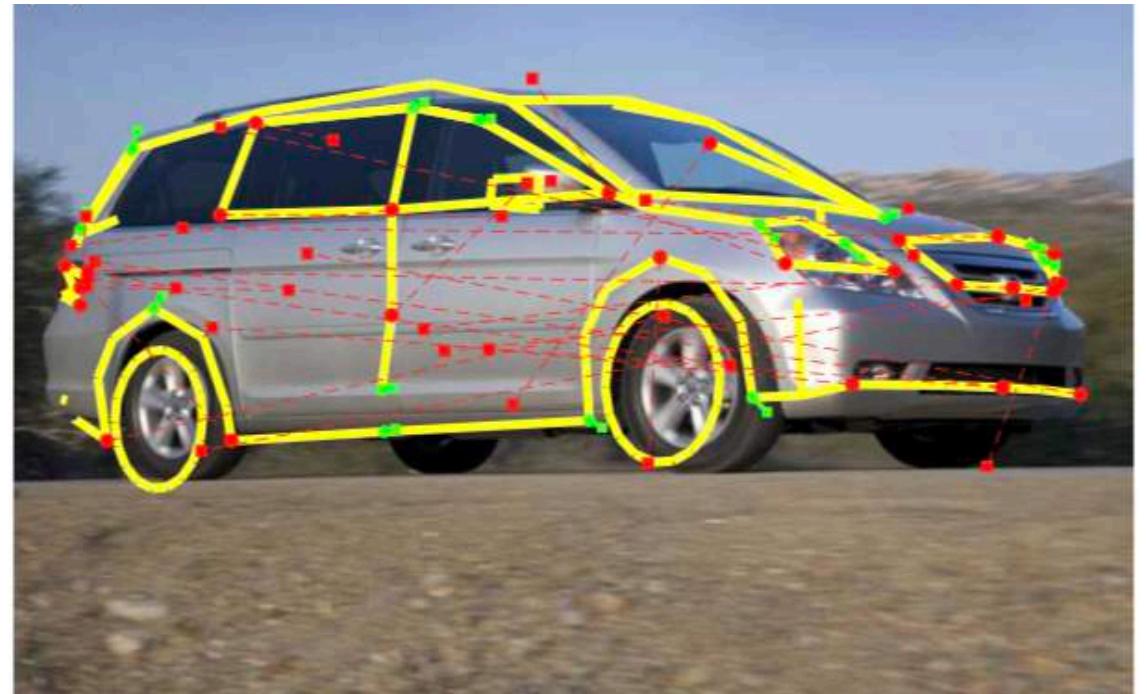


Co-design

- Computation-communication co-design
- Control and sensing co-design



Outline



Certifiable Algorithms for
Geometric Perception



Monitoring and Diagnosability
of Perception Systems



Outline



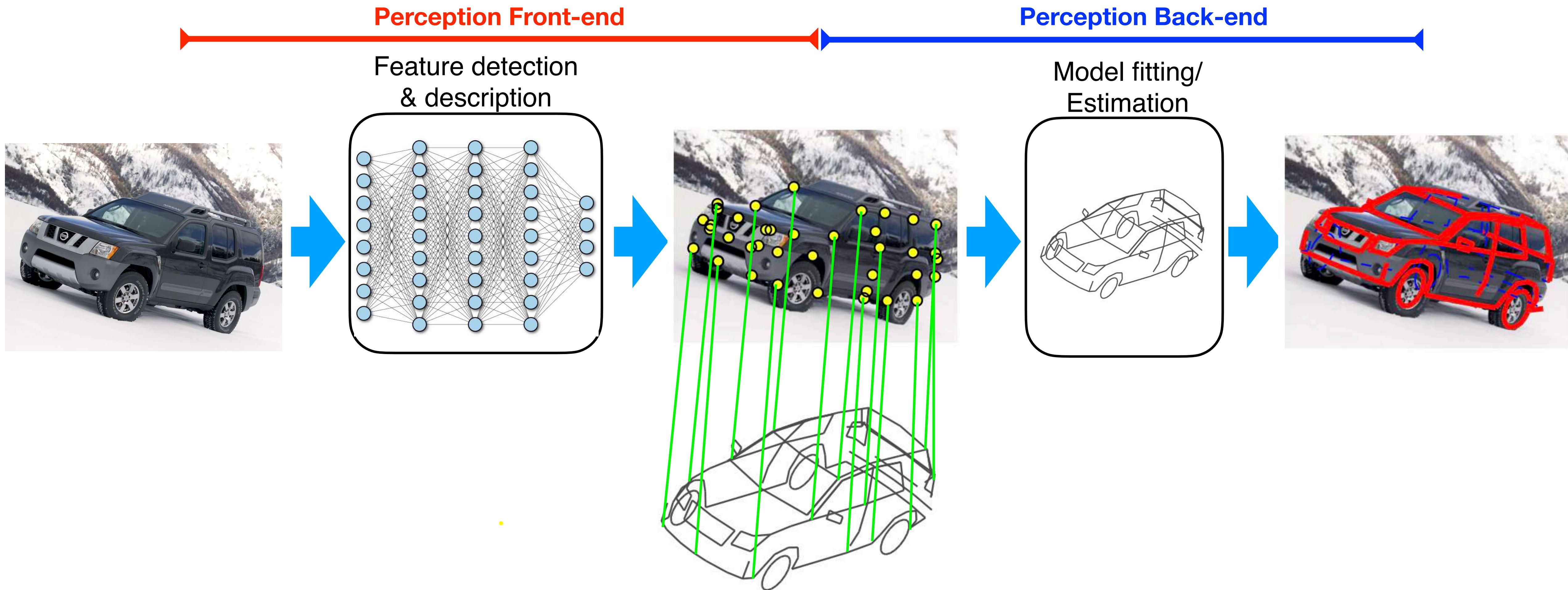
Certifiable Algorithms for
Geometric Perception



Monitoring and Diagnosability
of Perception Systems

Can we increase the robustness of
geometric perception algorithms and
potentially get performance guarantees?

Geometric perception example: image-based object localization

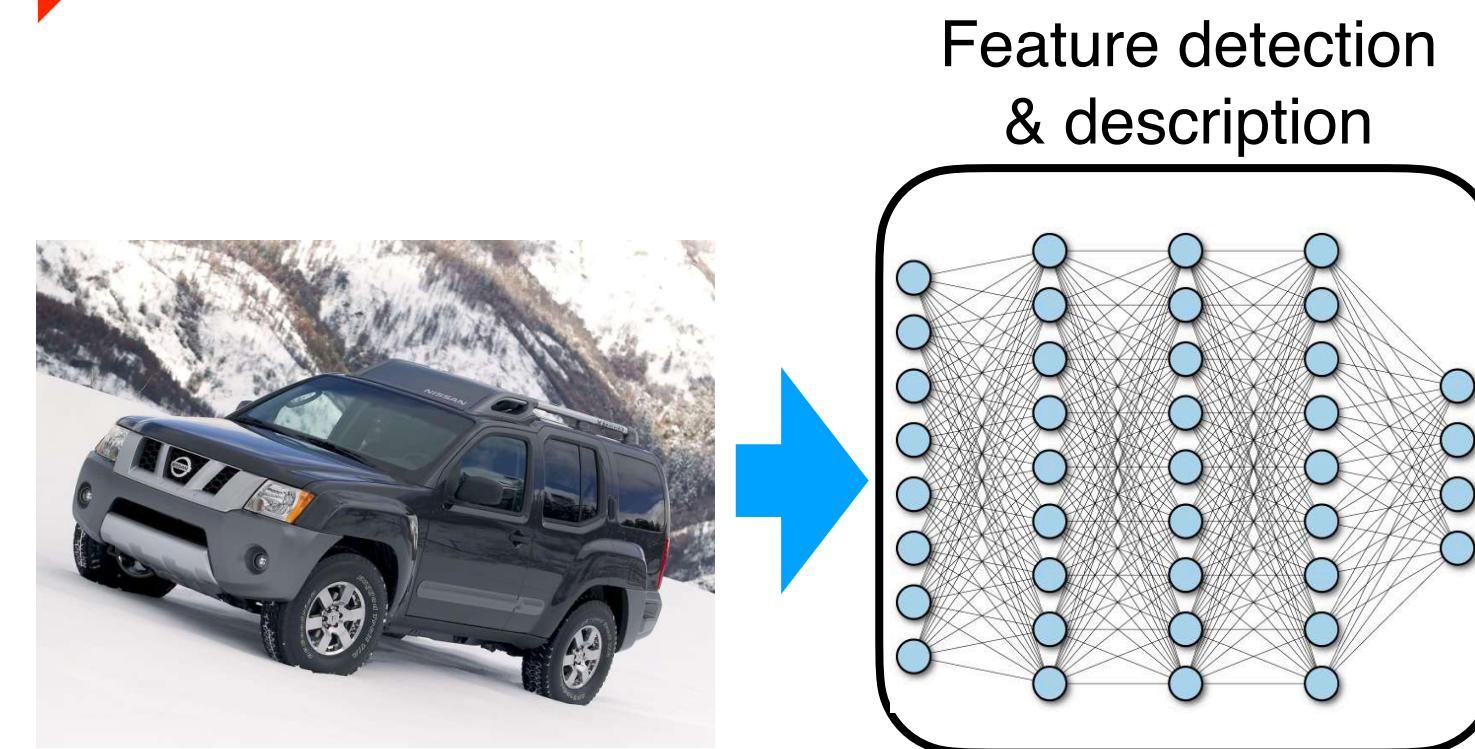


RGB images: [Gu&Kanade, CVPR'06][Lin, ECCV'14][Zhou, CVPR'15][Pavlakos, ICRA'17][Xinke, RSS'19][Yang, CVPR'20]
Point clouds: [PointNetLK, CVPR'19][DCP, ICCV'19][SmoothNet, CVPR'19][TEASER, RSS'19, TRO'20]

(Generality)

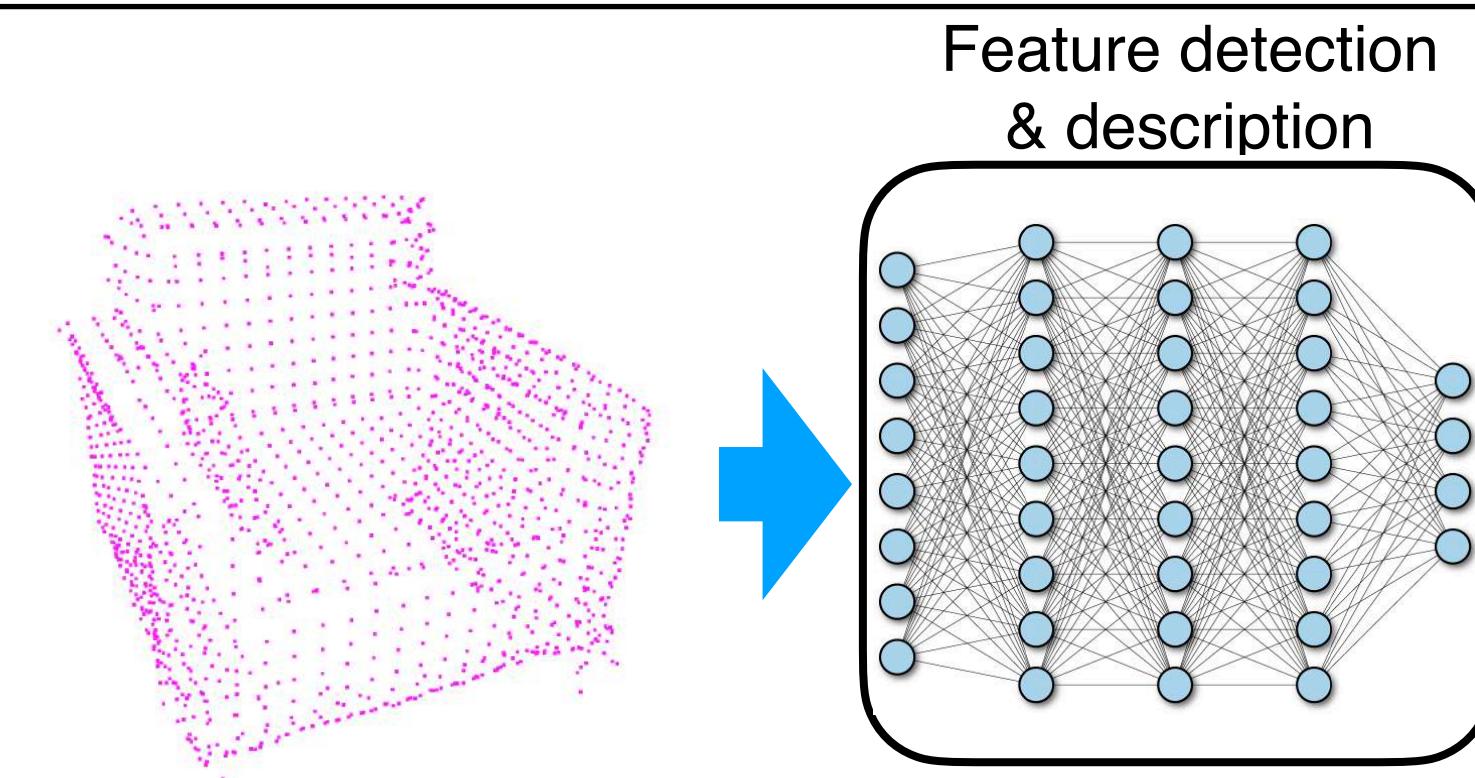
Perception Front-end

Object localization in images



Perception Back-end

Object localization in point clouds



SLAM
(visual-inertial navigation,
Structure from Motion)

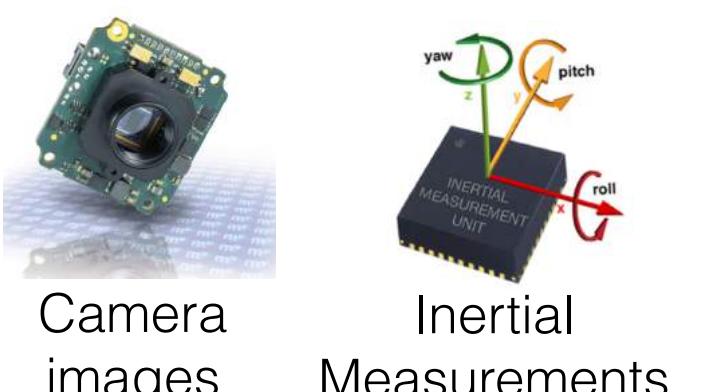
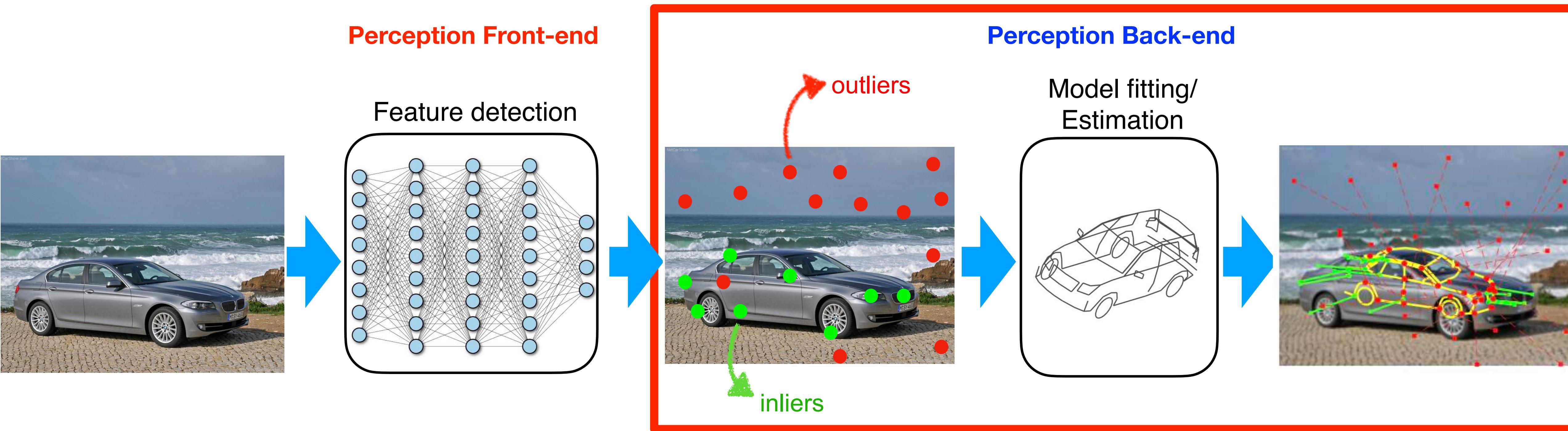


Image-based object localization: perception issues

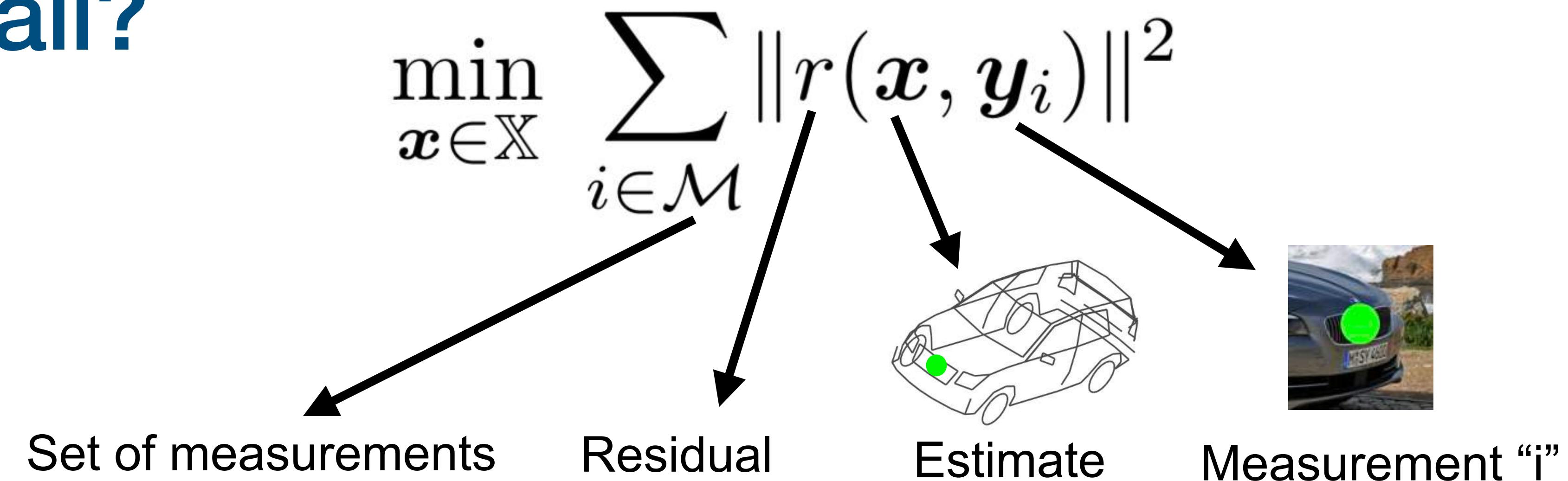


ISSUE 1: front-end (hand-crafted or deep-learned) can produce many misdetections (not uncommon to have >90% outliers)

ISSUE 2: back-end may fail if there are many outliers

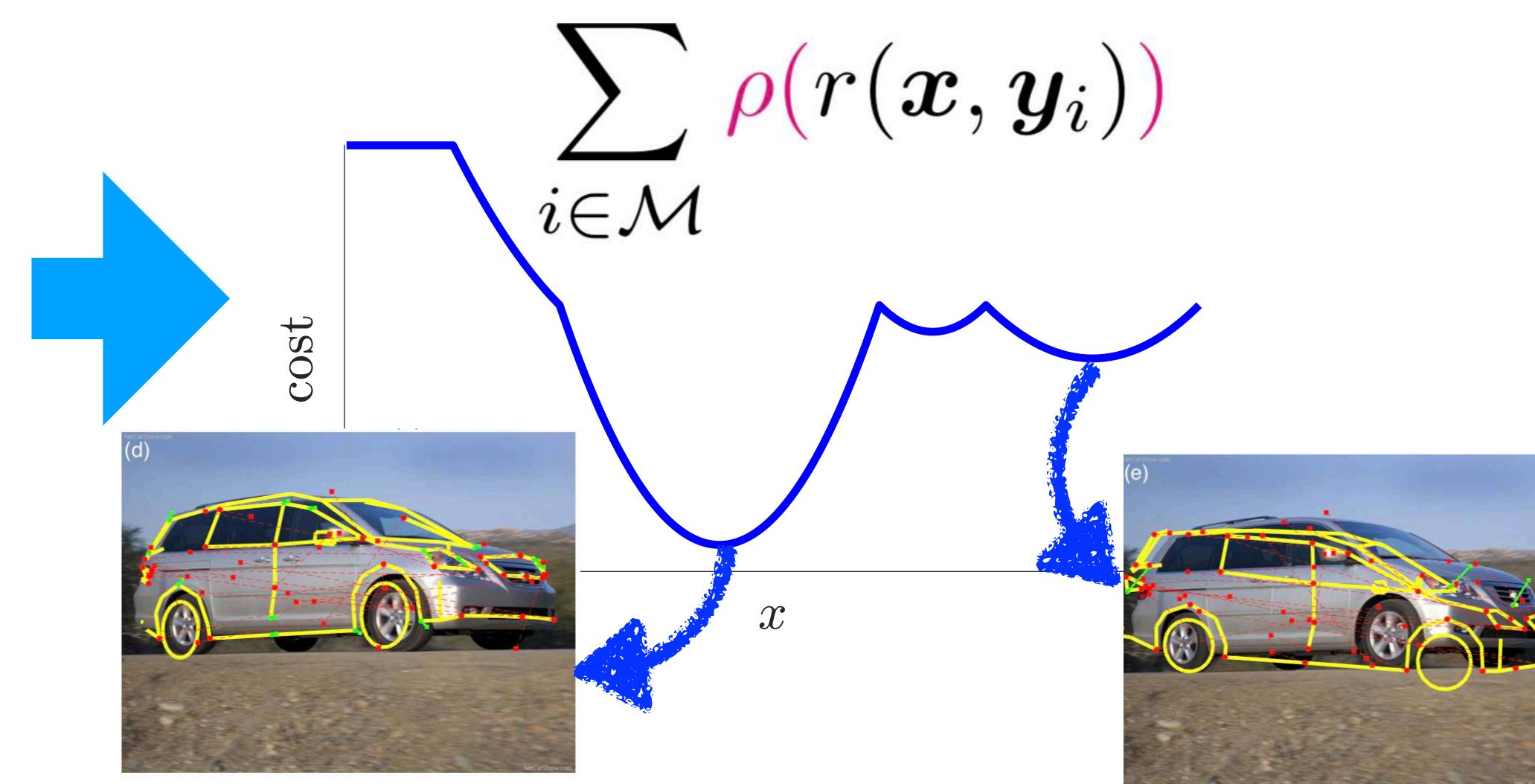
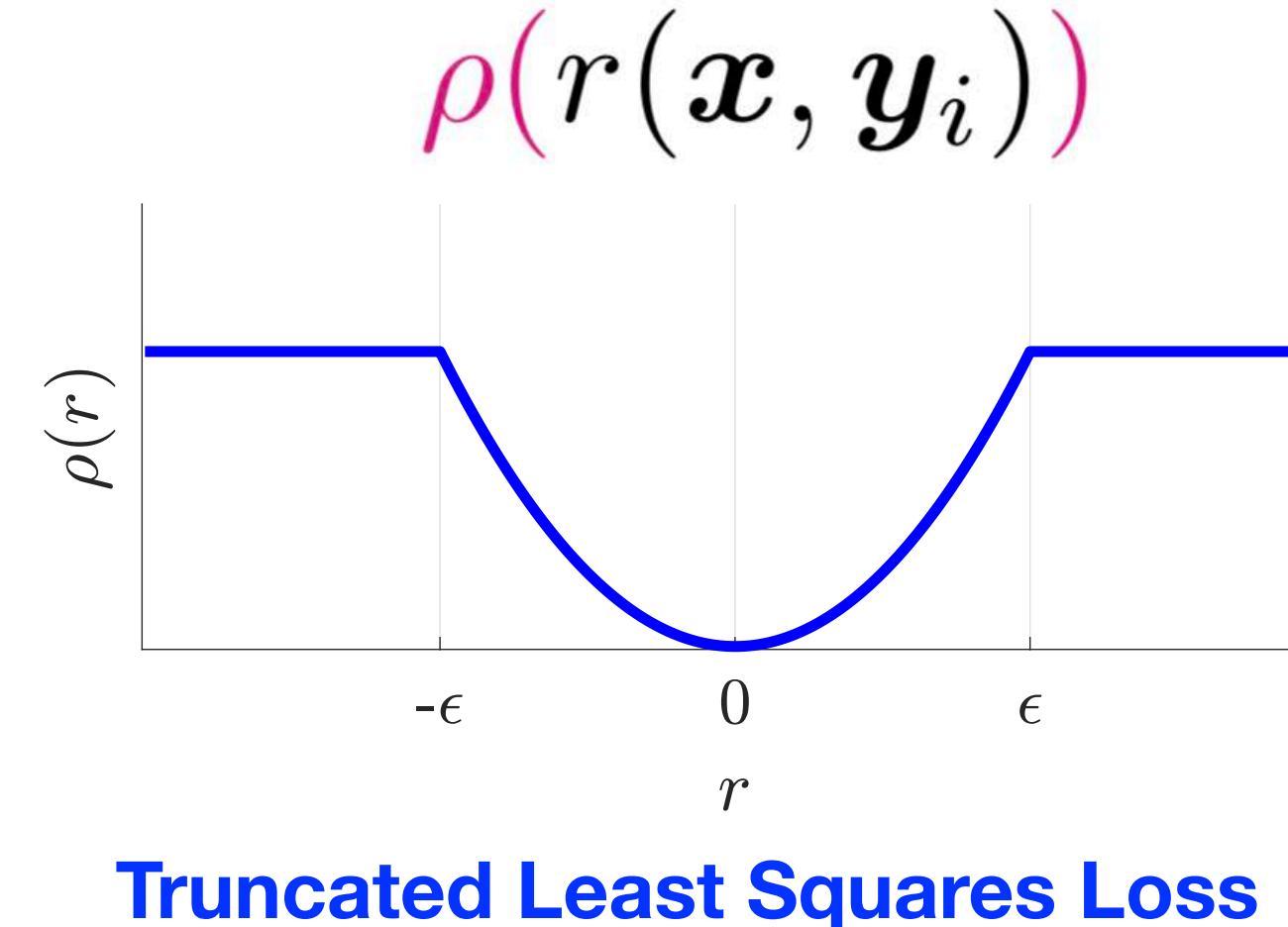
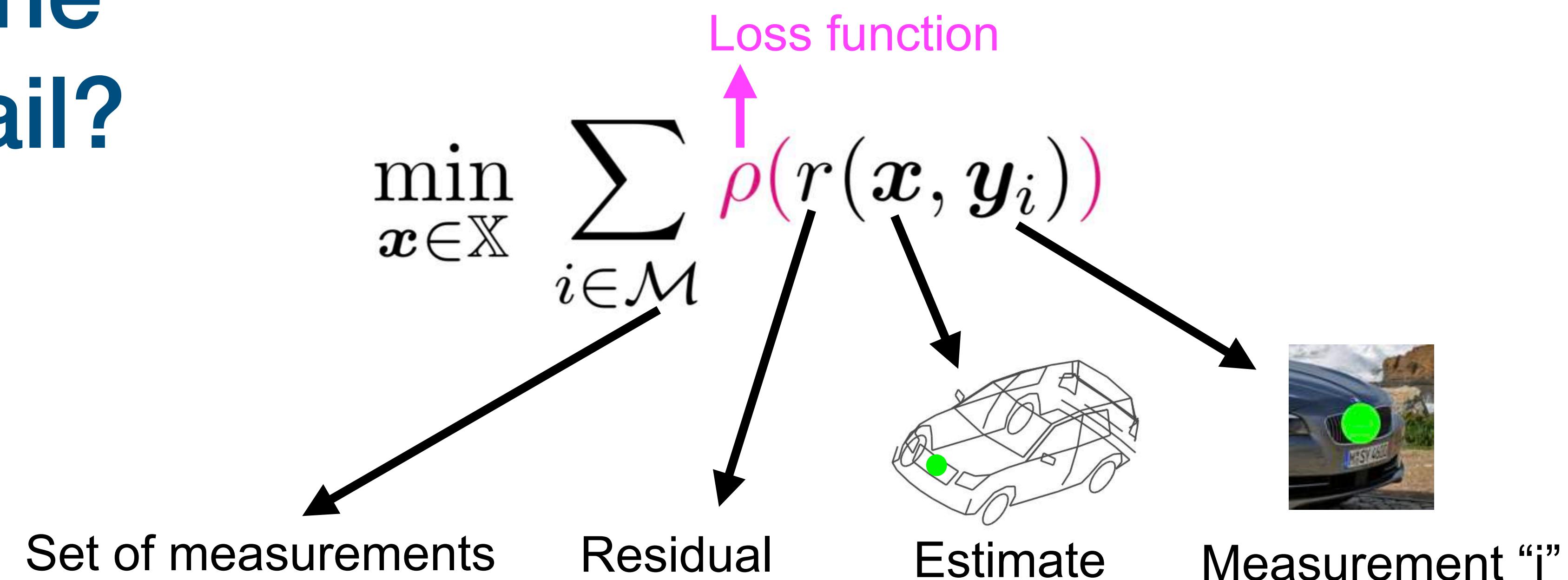
RGB images: [Gu&Kanade, CVPR'06][Lin, ECCV'14][Zhou, CVPR'15][Pavlakos, ICRA'17][Xinke, RSS'19][Yang, CVPR'20]
Point clouds: [PointNetLK, CVPR'19][DCP, ICCV'19][SmoothNet, CVPR'19][TEASER, RSS'19, TRO'20]

Why does the back-end fail?



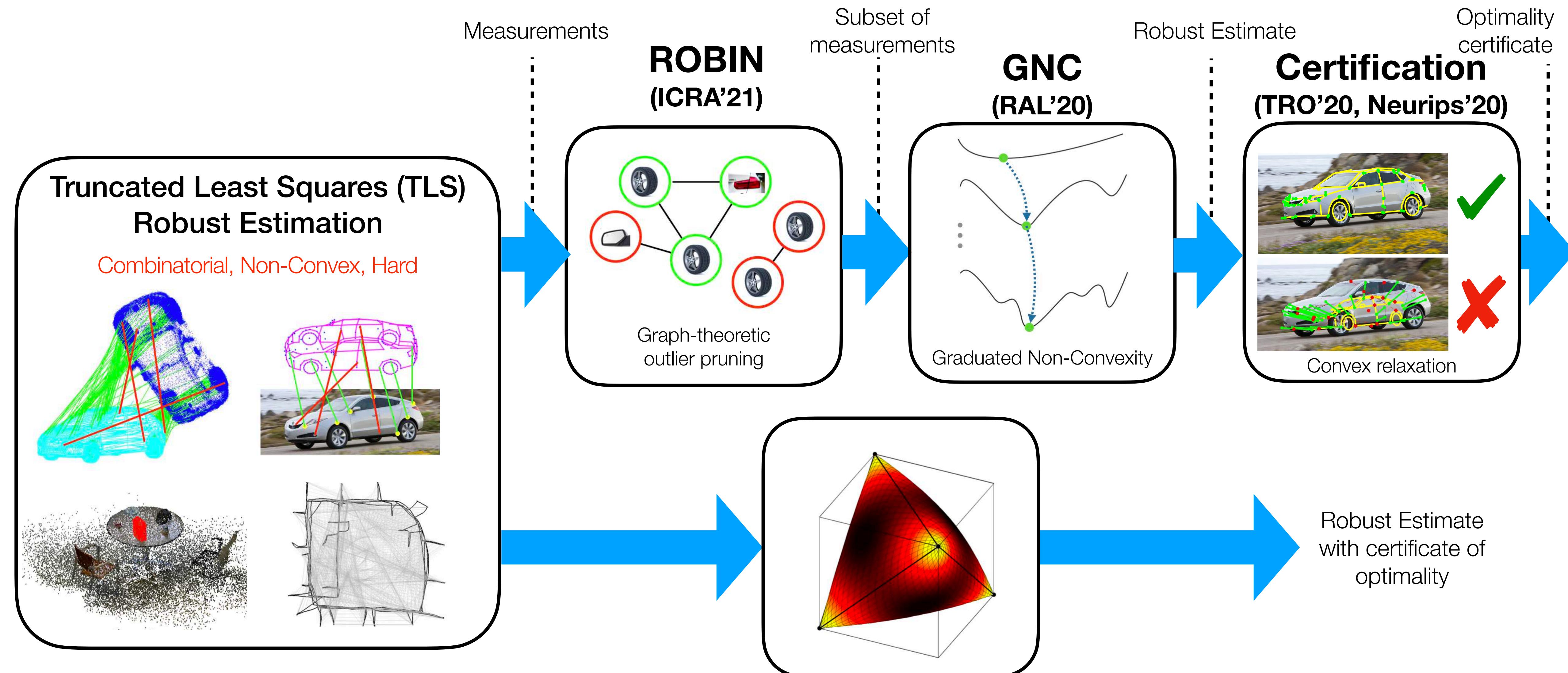
Why does the back-end fail?

ISSUE: for common choices of loss function ρ , the problem is non-convex:



- State of the art:**
- **Local solvers**: need initial guess, stuck in local minima
 - **RANSAC**: fails with many outliers, low-dimensional problems, non-deterministic
 - both fail without notice

Certifiable Perception Algorithms



$$\arg \min_{\substack{x \in \mathbb{X}, \\ \theta_i \in \{0,1\}, \forall i}} \sum_{i \in \mathcal{M}} \theta_i \|r_i(x, y_i)\|^2 + (1 - \theta_i) \bar{c}^2$$

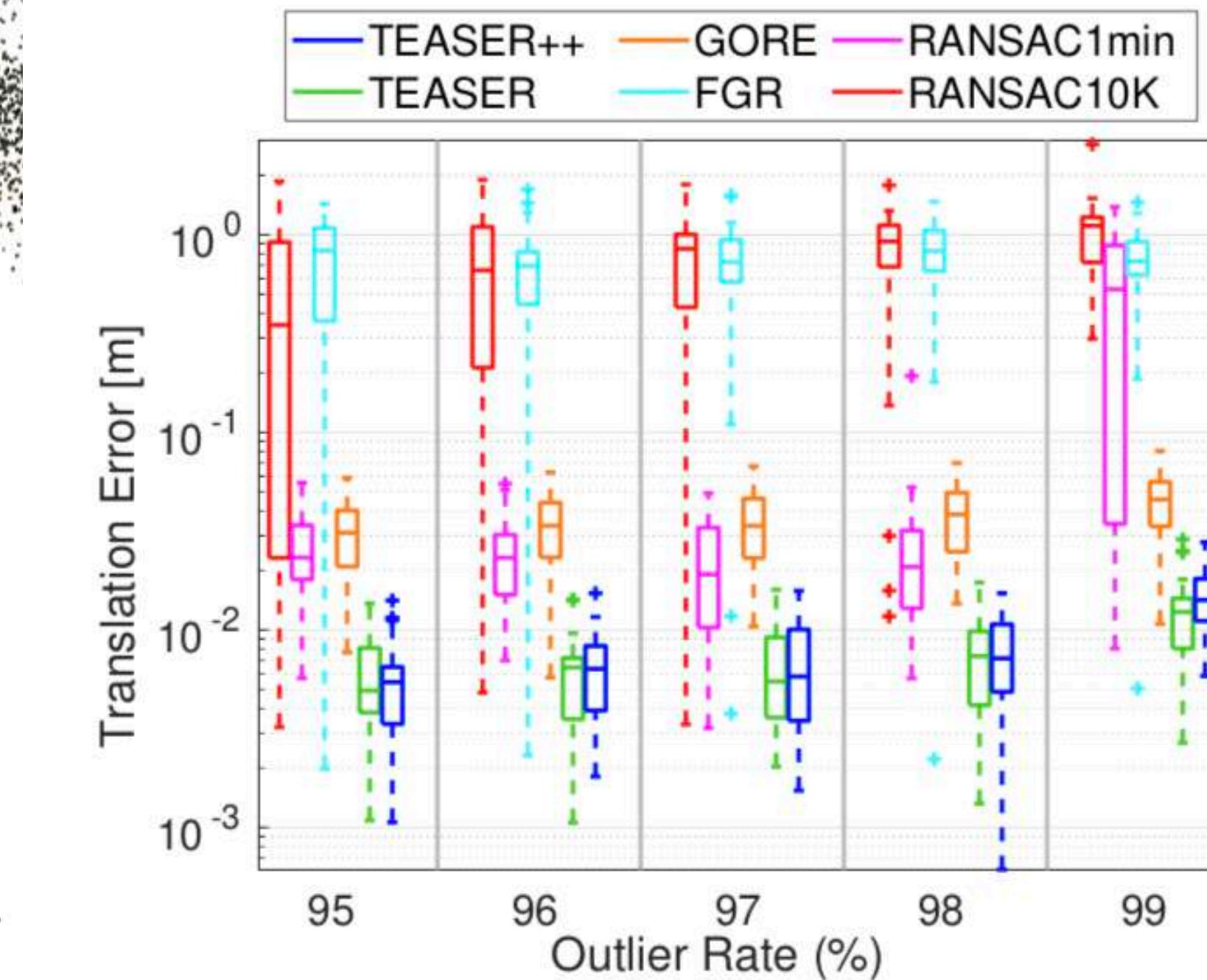
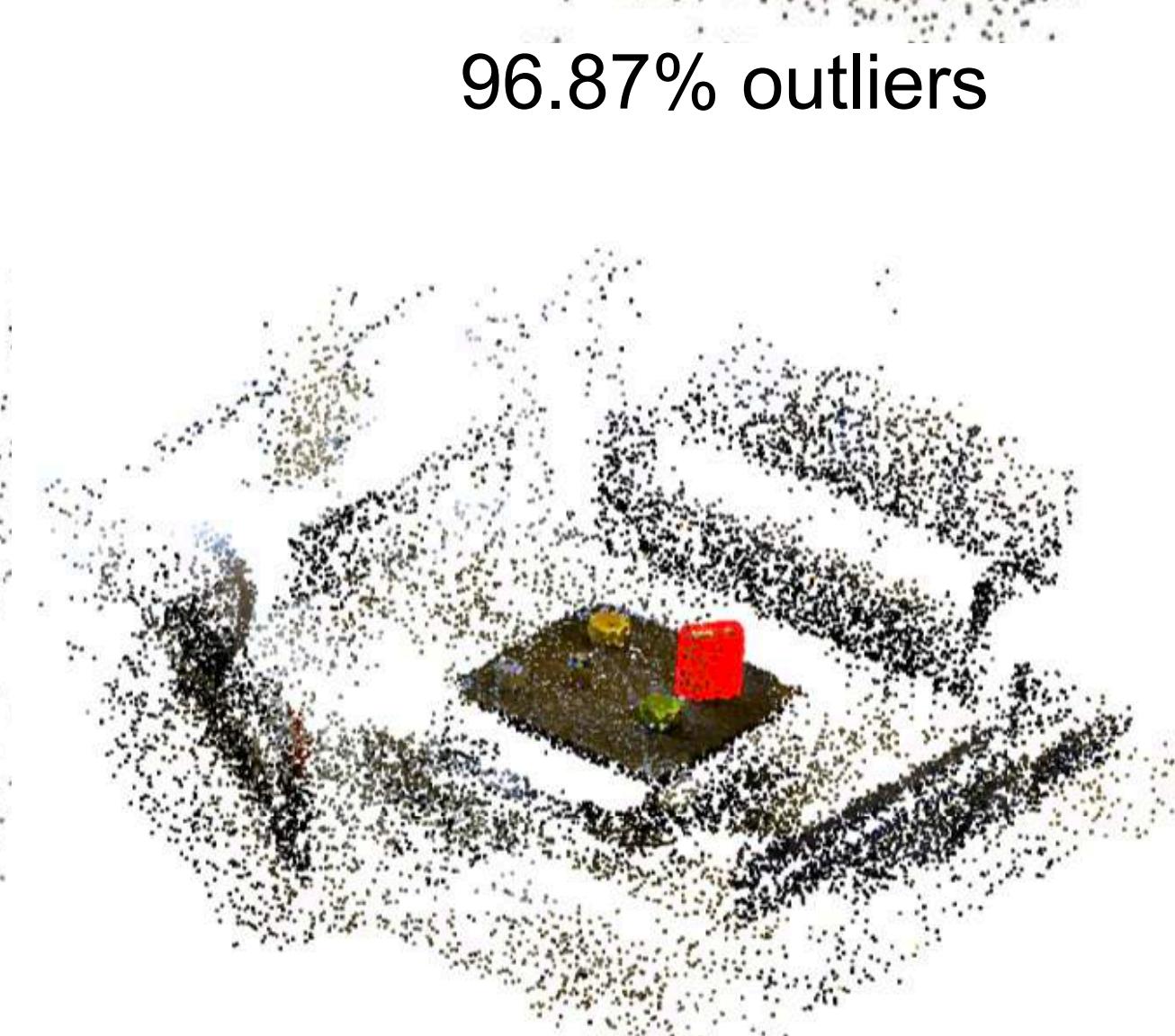
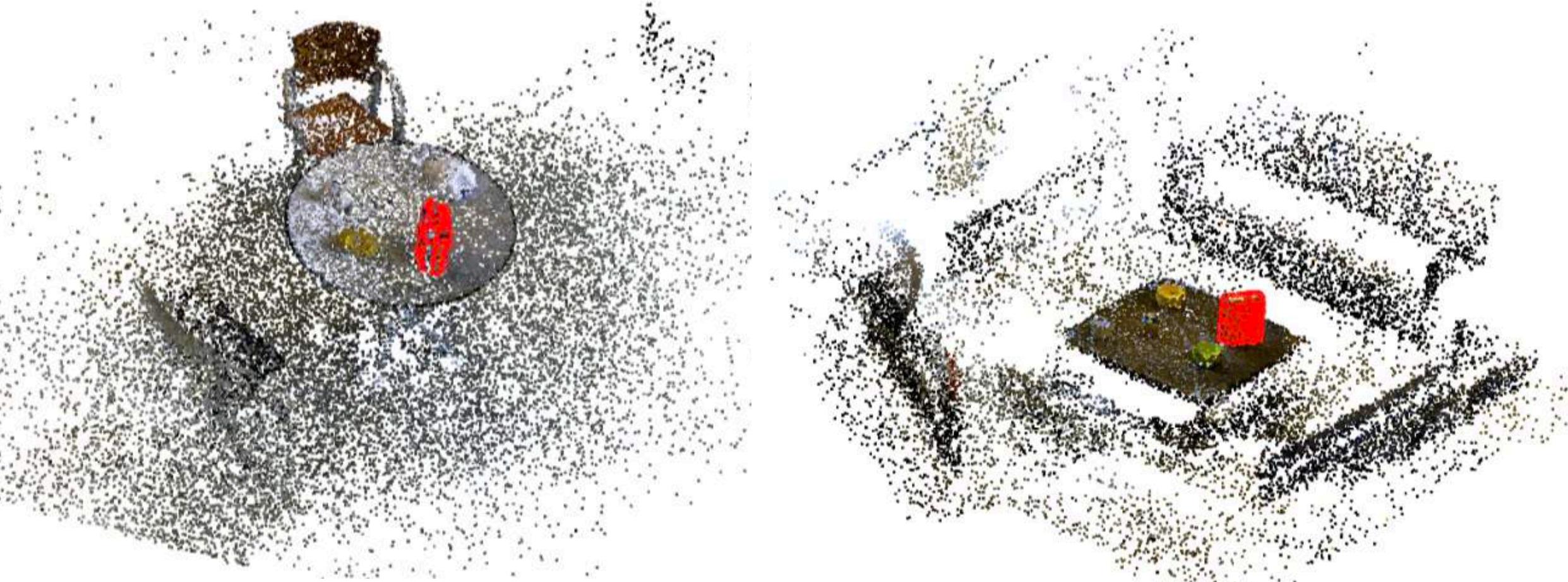
**Minimum-order Moment/SOS Relaxation
(Semidefinite Program)**

[ICRA'20-21, T-RO'20, Neurips'20]

Example 1: TEASER++ for object localization in point clouds

Registration with ROBIN + GNC

- RGB-D point cloud dataset
- ~500 FPFH features



TEASER and
TEASER++
(proposed) are
best approaches

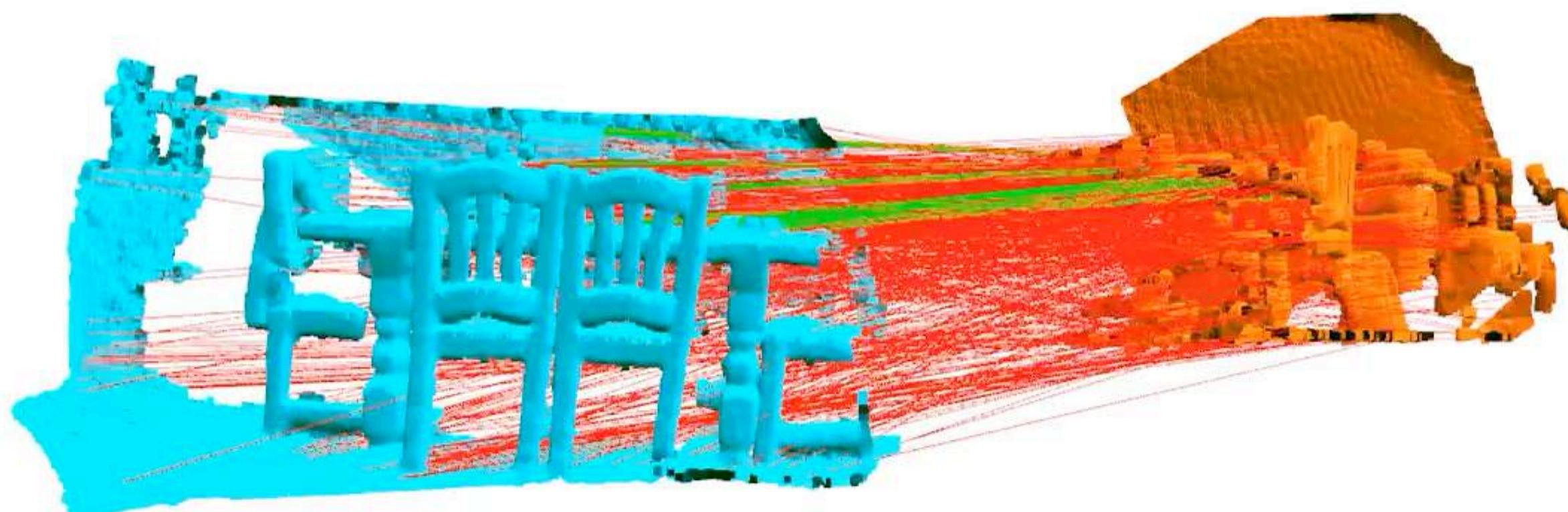


TEASER++
runs in 20ms

TEASER++ for point cloud registration

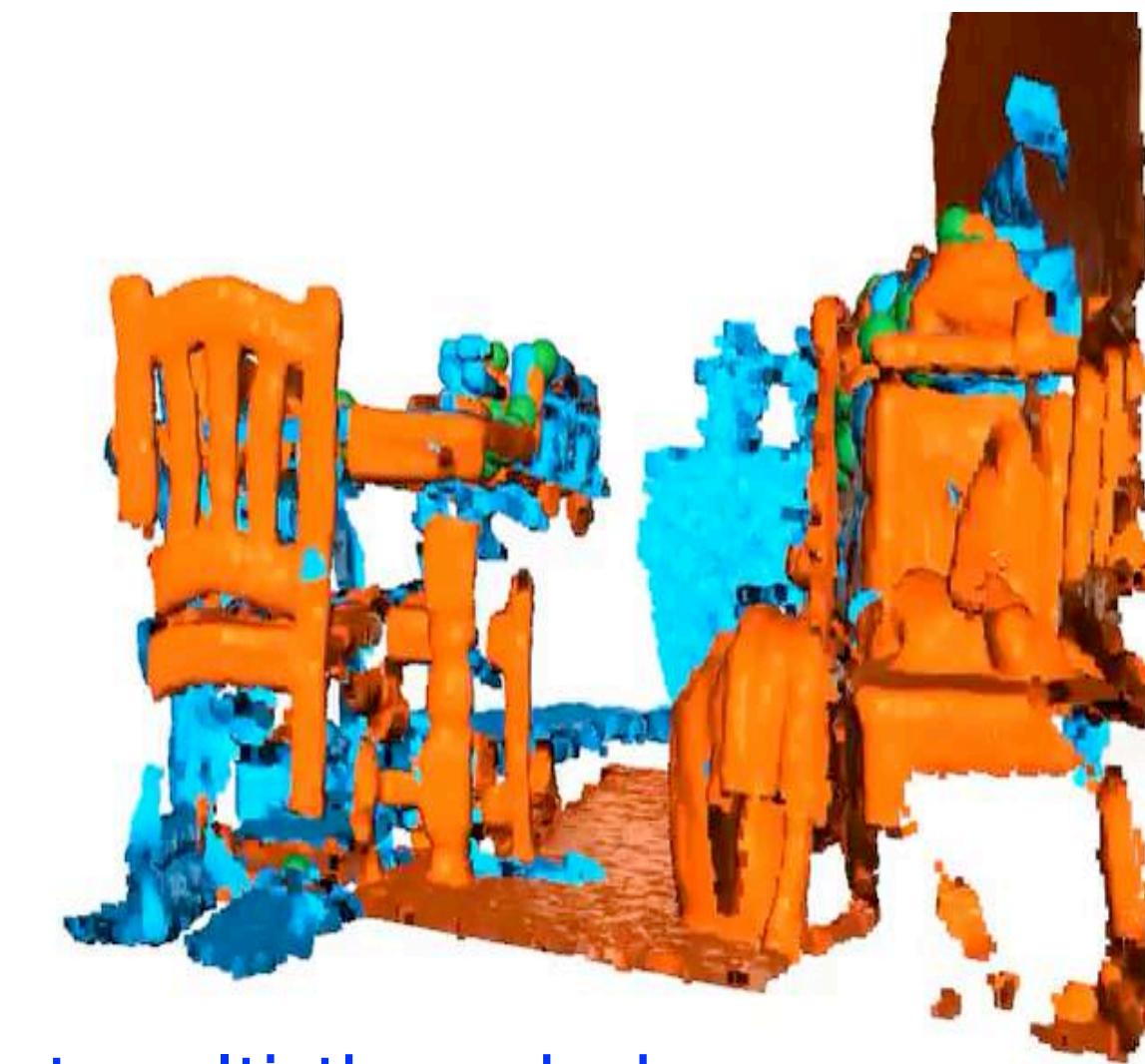
- 3DMatch dataset, ~1000 deep-learned features (using 3DSmoothNet)

Scenes	Kitchen (%)	Home 1 (%)	Home 2 (%)	Hotel 1 (%)	Hotel 2 (%)	Hotel 3 (%)	Study Room (%)	MIT Lab (%)
RANSAC-1K	90.9	91.0	73.1	88.1	80.8	87.0	79.1	81.8
RANSAC-10K	96.4	92.3	73.1	92.0	84.6	90.7	82.2	81.8
TEASER++	97.7	92.3	82.7	96.9	88.5	94.4	88.7	84.4
TEASER++ (CERT)	99.2	97.5	90.0	98.8	94.9	97.7	94.8	93.9



Green lines:
inlier correspondences

Red lines:
outlier correspondences



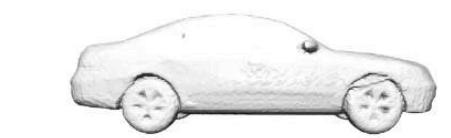
Fast multi-threaded open-source code:
<https://github.com/MIT-SPARK/TEASER-plusplus>

Extension: category-level object localization in point clouds

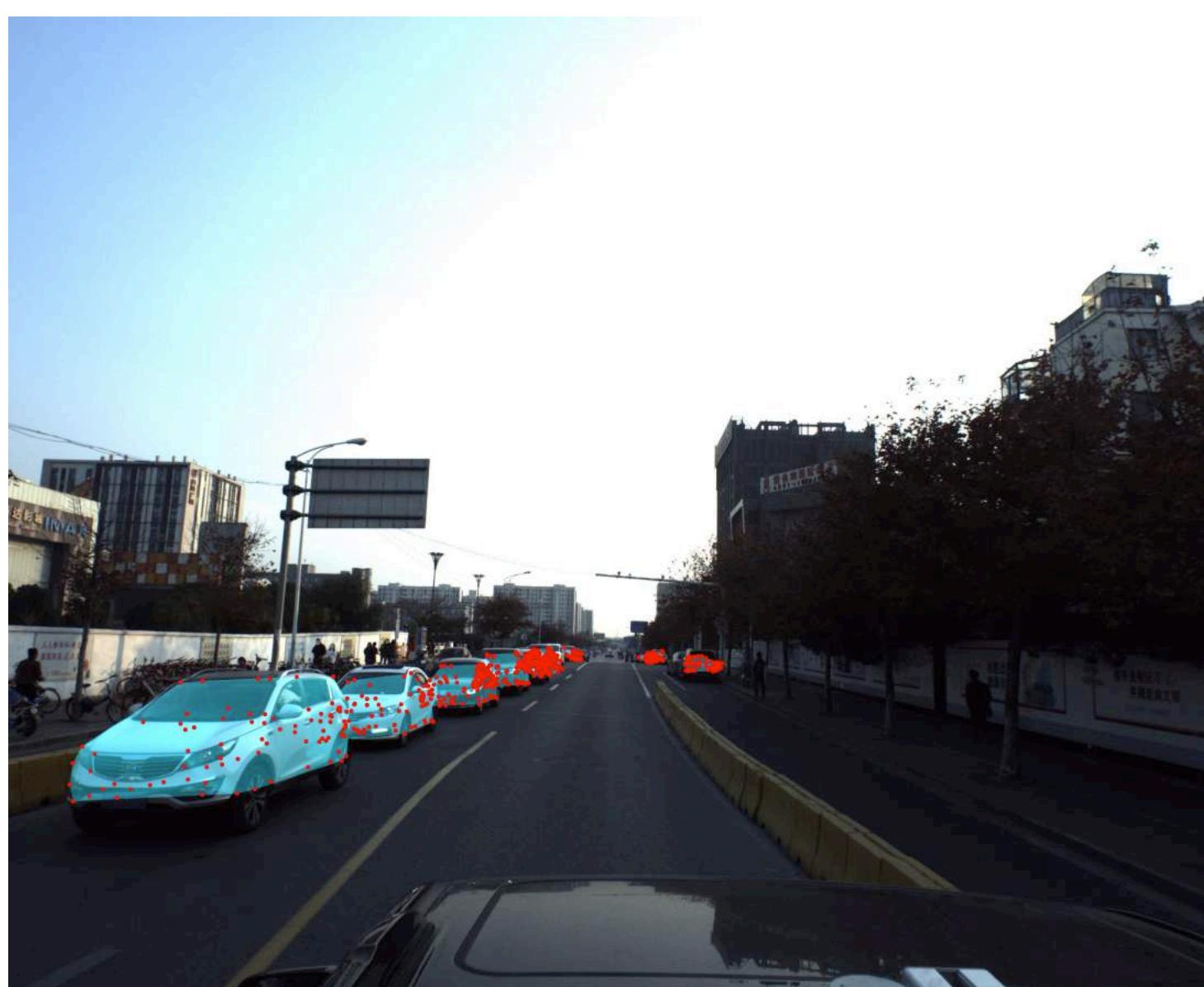
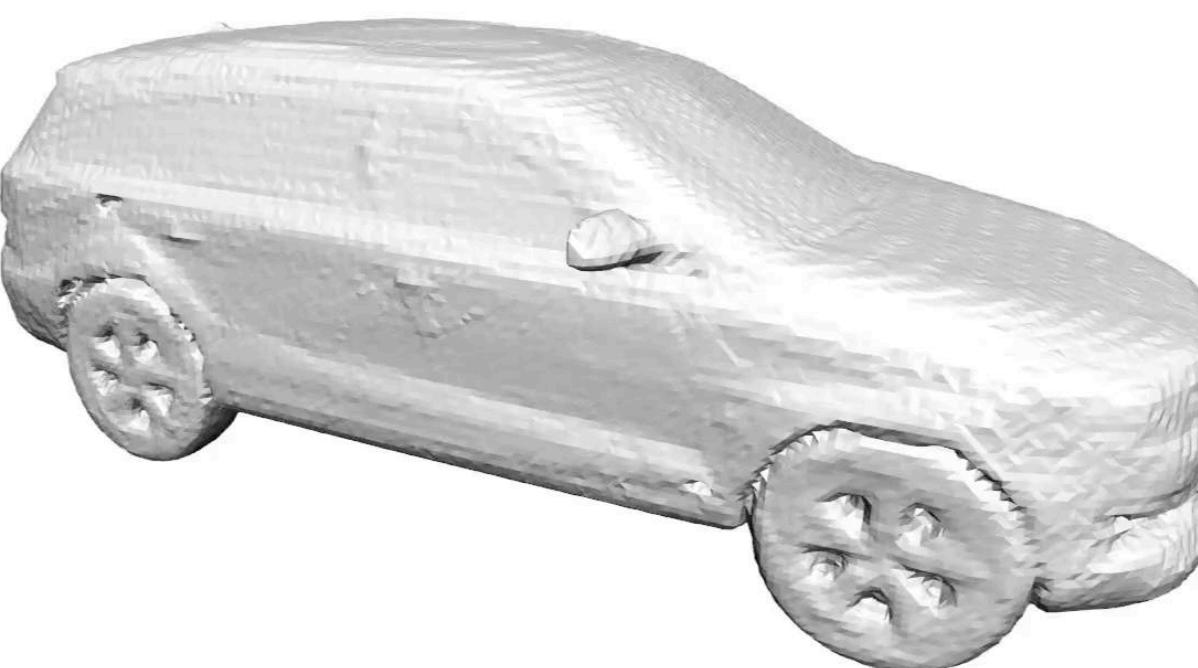
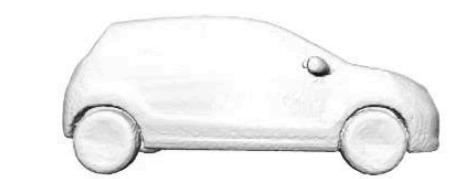
ApolloScape self-driving dataset: 200 images (validation split),
79 car models, 66 keypoints per car.



Hyundai Sonata



Suzuki Alto



Deep learning
baselines
(2017-2020)

Strict criterion: % of estimates with $<1.4m$
translation error and $<15^\circ$ for rotation error

	A3DP-Rel ↑			A3DP-Abs ↑		
	mean	c-l	c-s	mean	c-l	c-s
DeepMANTA [10]	16.0	23.8	19.8	20.1	30.7	23.8
3D-RCNN [34]	10.8	17.8	11.9	16.4	29.7	19.8
GSNet [30]	20.2	40.5	19.9	18.9	37.4	18.4
PACE#-ApolloDepths	25.9	35.7	33.7	22.4	34.7	31.6
PACE#-GTDepths	36.0	45.4	43.6	35.3	44.2	43.2
PACE#-GTKeypoints	64.5	88.1	86.0	64.3	88.1	86.1

GSNet keypoint detection	PACE# Max-clique	GNC
0.45 s	2ms	0.45 s

TABLE II: Timing breakdown for PACE#.

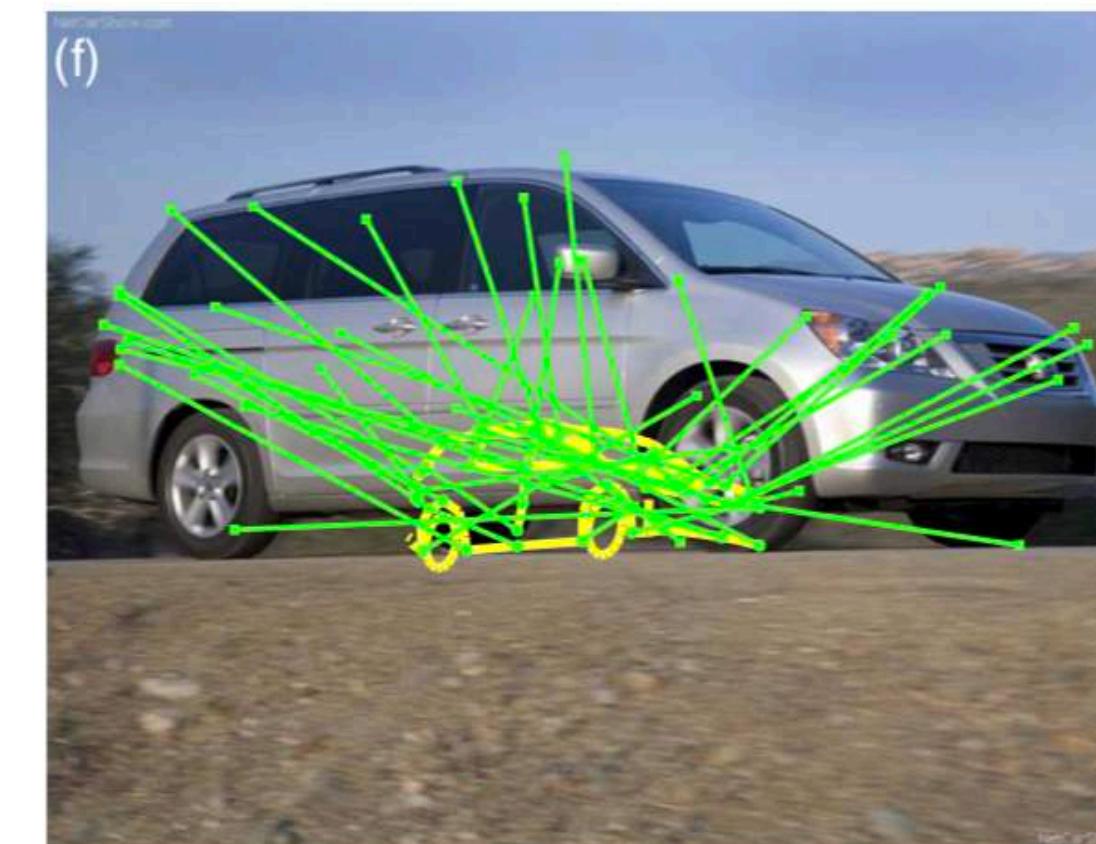
Non-optimized Python implementation

Example 2: Shape# for object localization in images

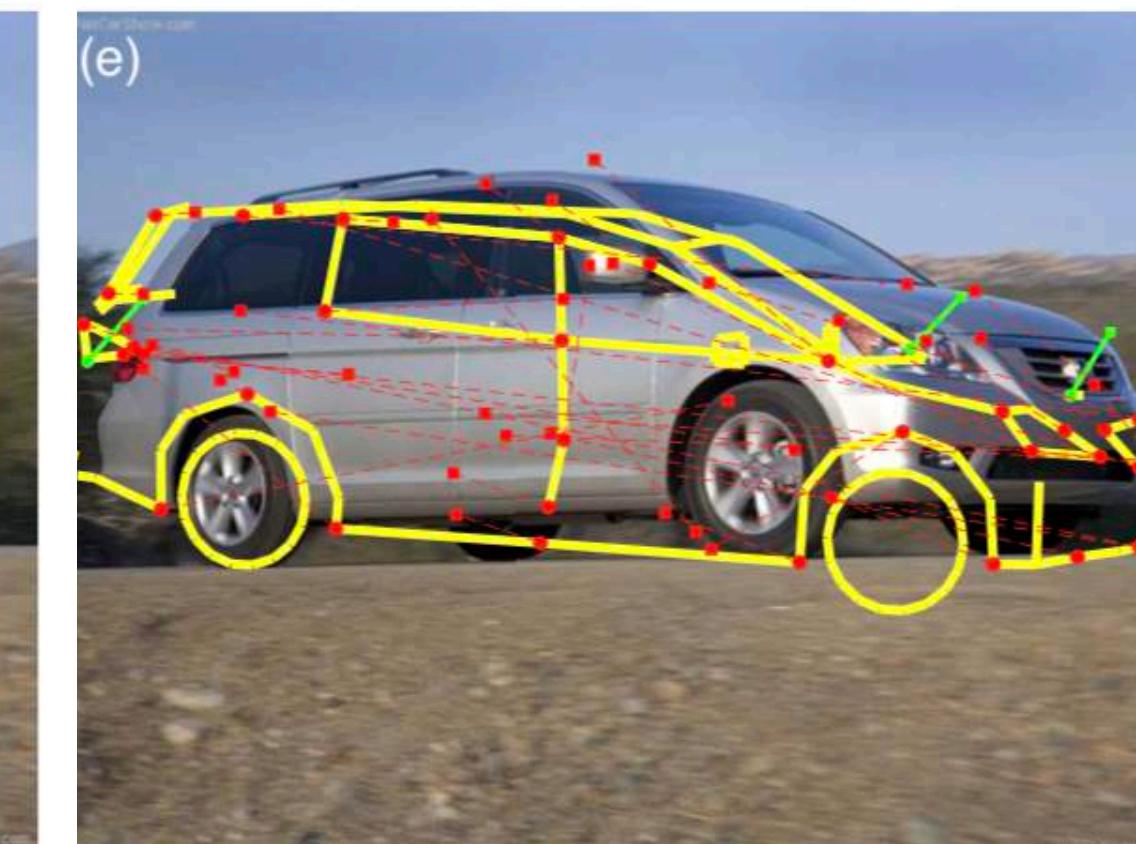
Shape alignment using GNC:

- FG3DCar dataset
 - 300 car images with corresponding CAD models

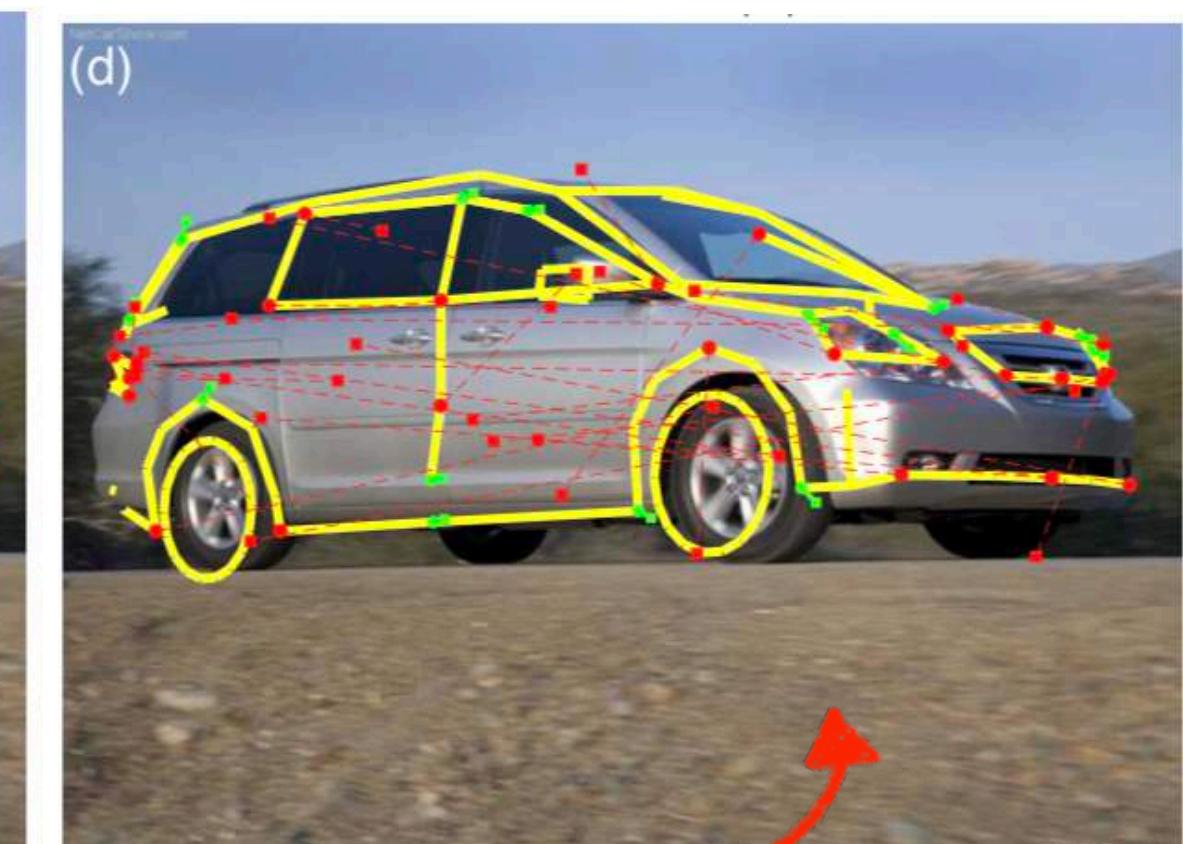
[Zhou, CVPR'15]



RANSAC



Shape#



70% outlier

Propose

Altern+Robust



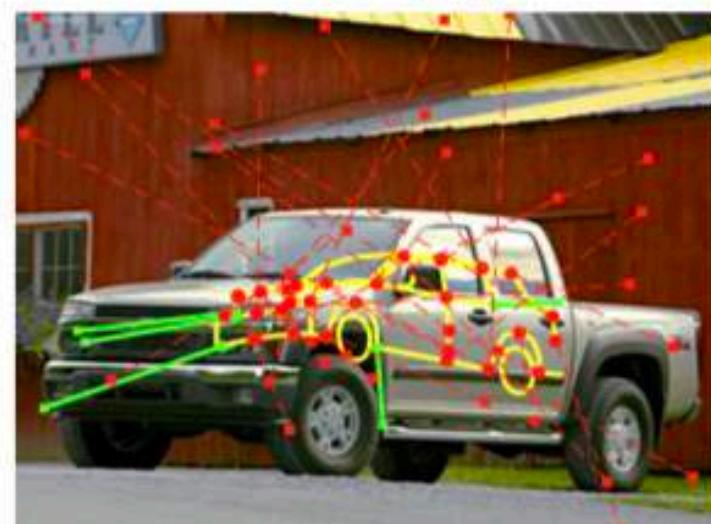
Convex+Robust



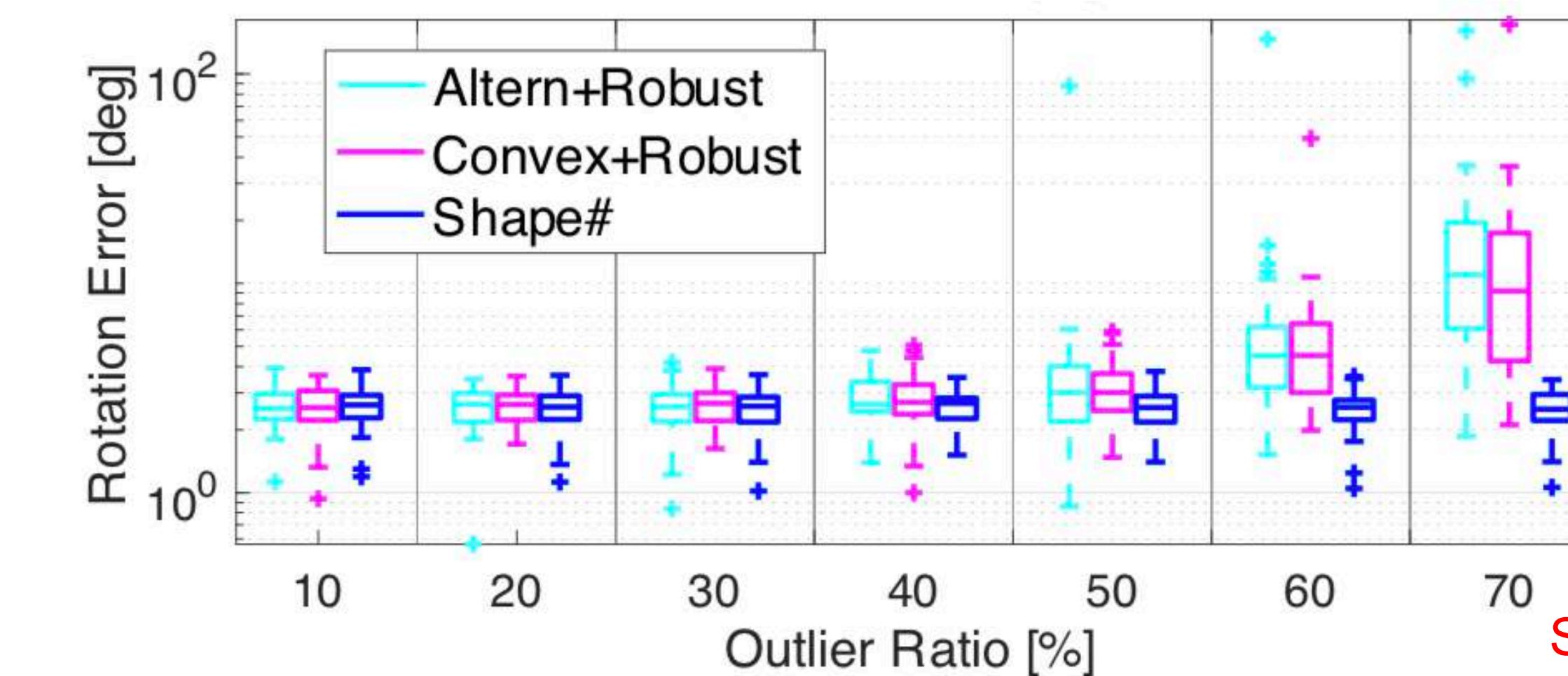
Shape



(a) Chevrolet Colorado LS 40% outliers

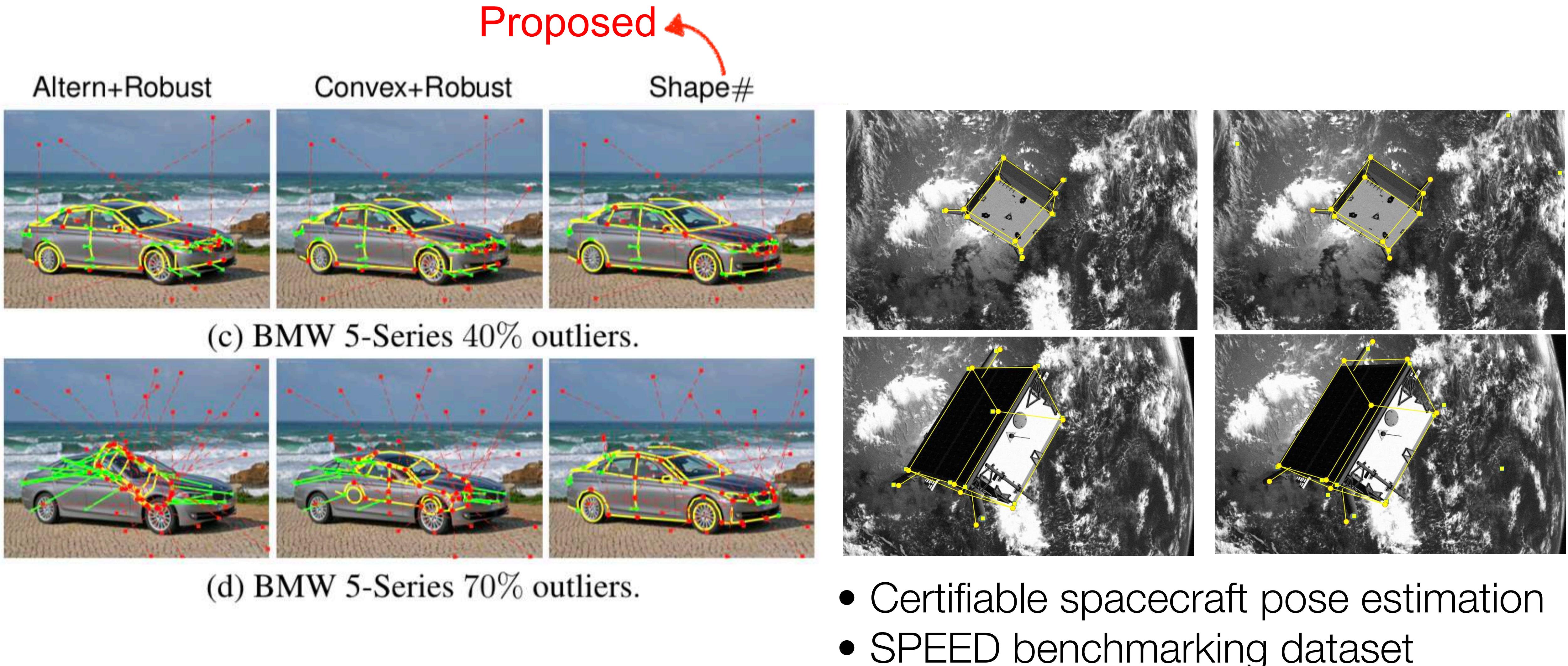


(b) Chevrolet Colorado LS 70% outlier



Shape#
(proposed) is
best approach

Shape#: experimental results

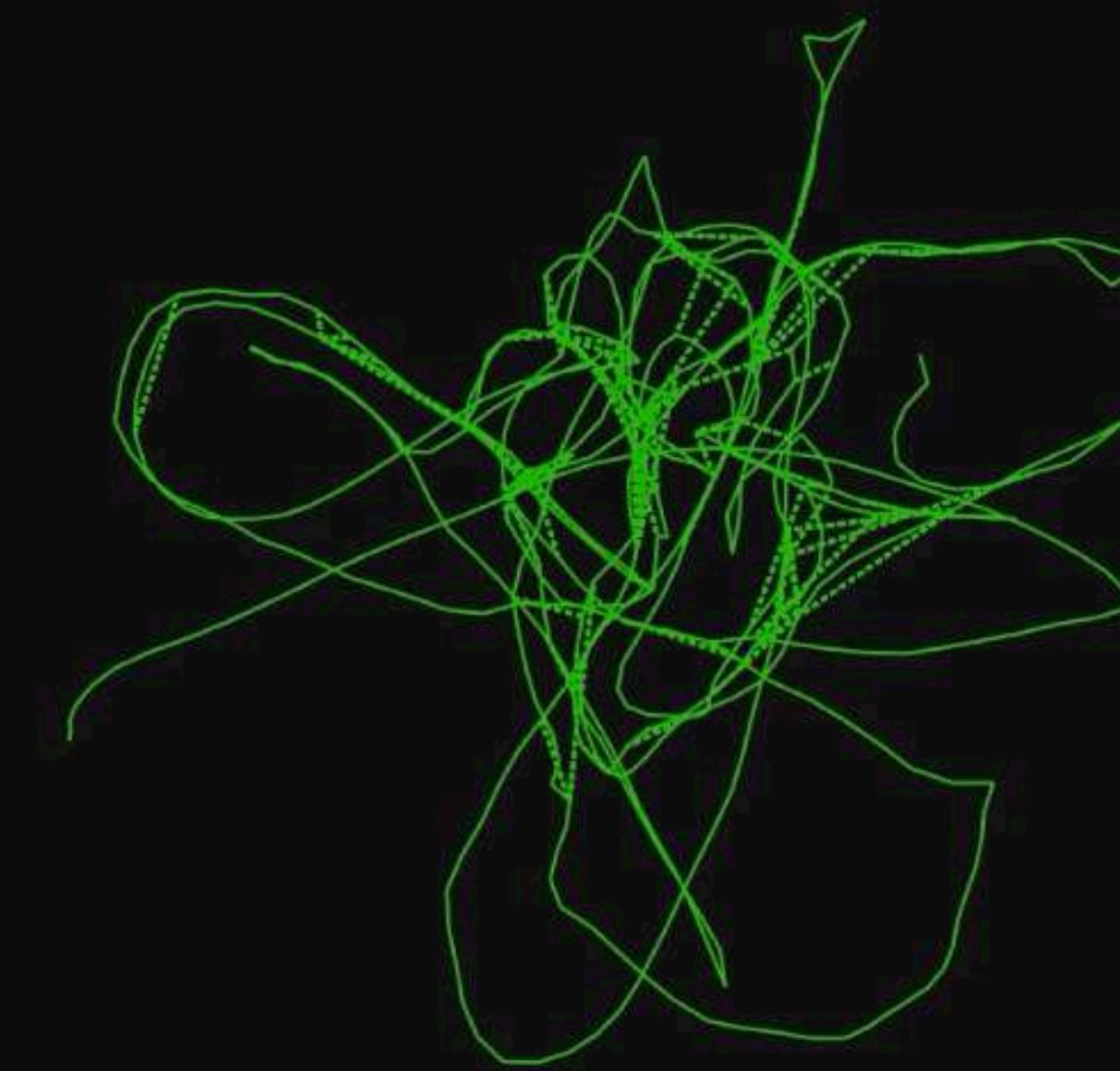


[Yang and Carlone. In Perfect Shape: Certifiably Optimal 3D Shape Reconstruction from 2D Landmarks. CVPR 2020]

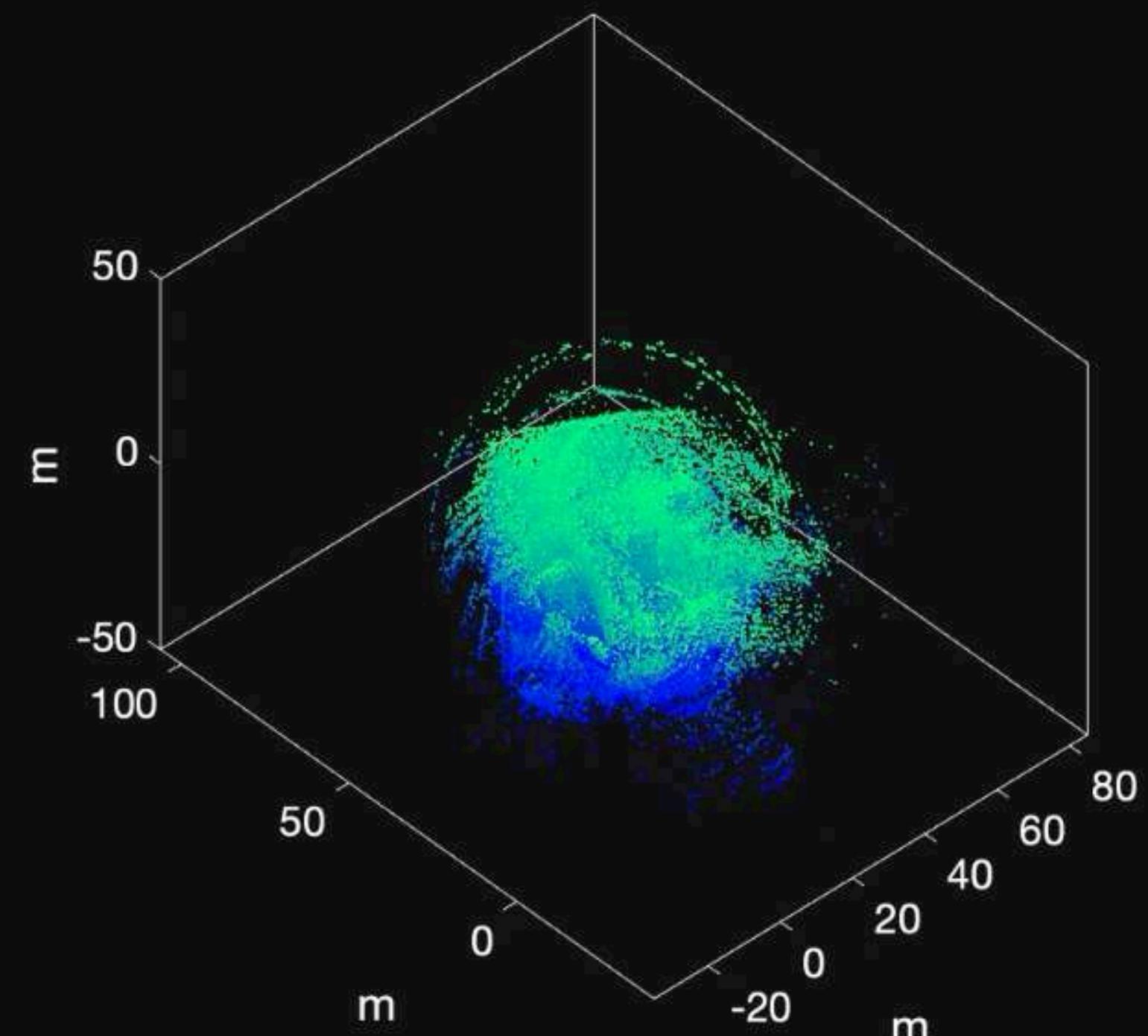
[Yang and Carlone, One Ring to Rule Them All: Certifiably Robust Geometric Perception with Outliers, NeurIPS'20.]

Example 3: Graduated Non-Convexity for SLAM

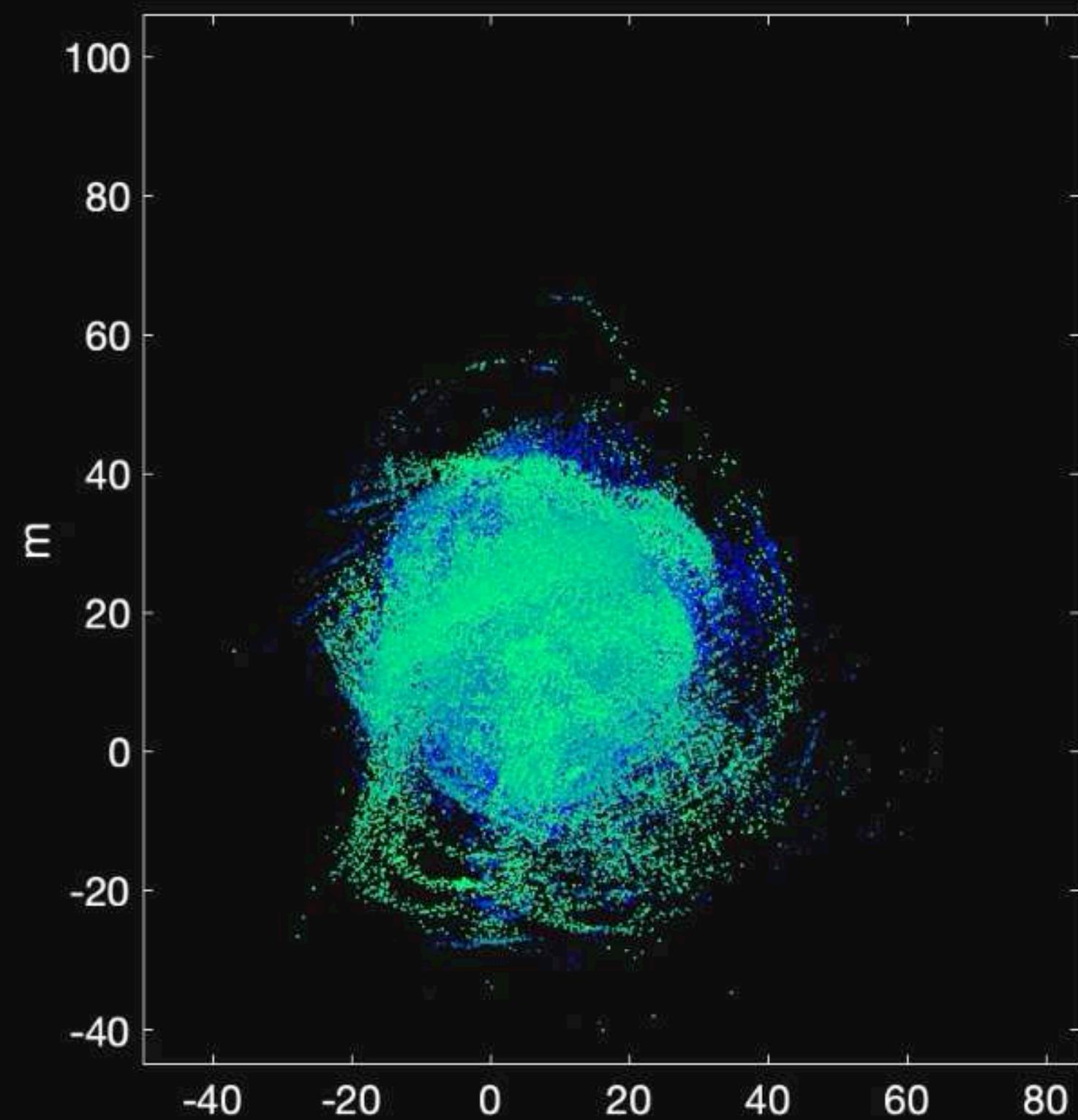
Robot Trajectory
(Outliers in red)



Reconstructed Map



Top View



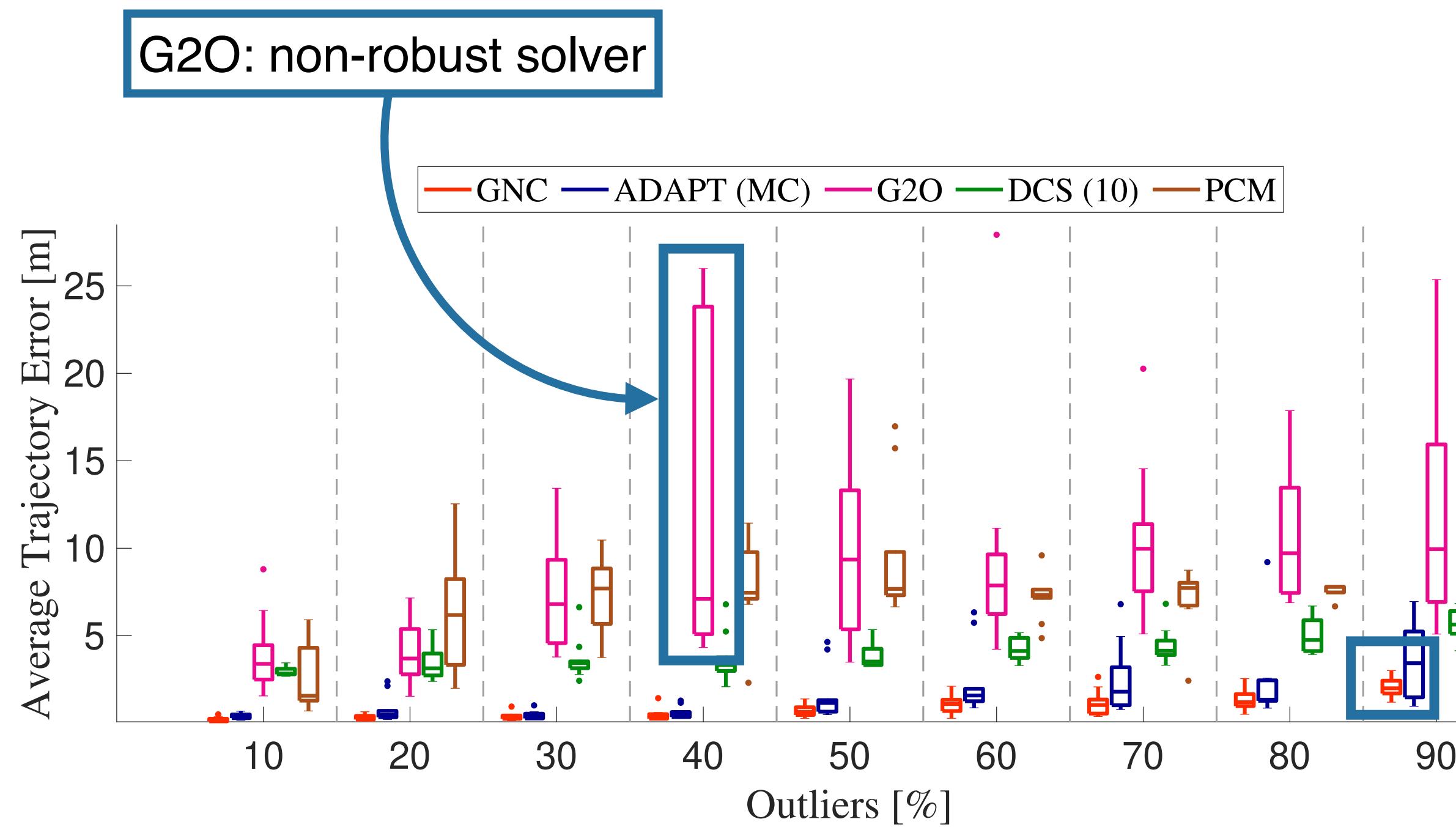
Inputs

Problem: estimate trajectory given motion estimates and loop closures.
Loop closures are contaminated with outliers

 SPARK
Lab

Graduated non-convexity for SLAM

Accuracy

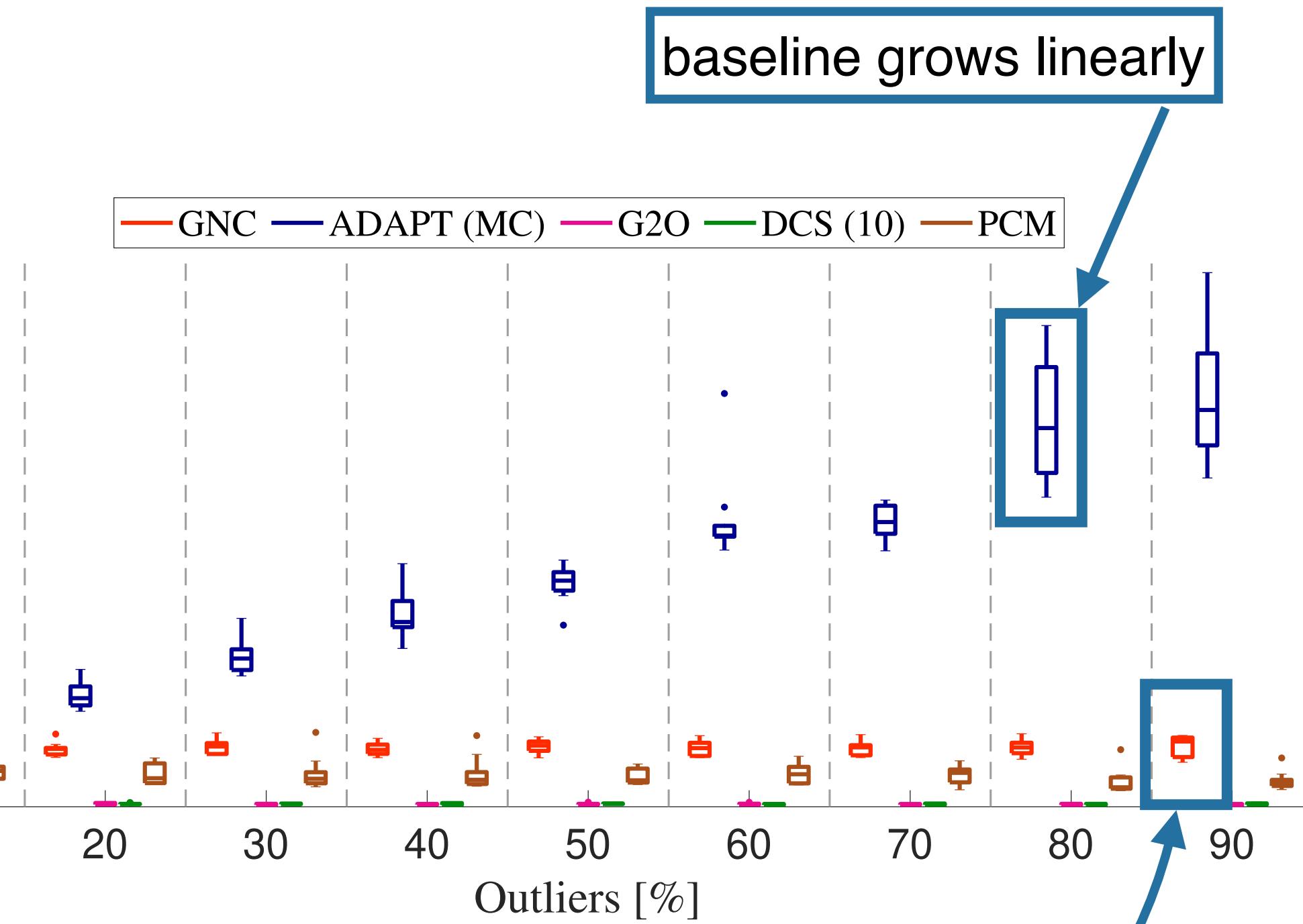


CSAIL SLAM dataset

GNC robust
up to 90% outliers

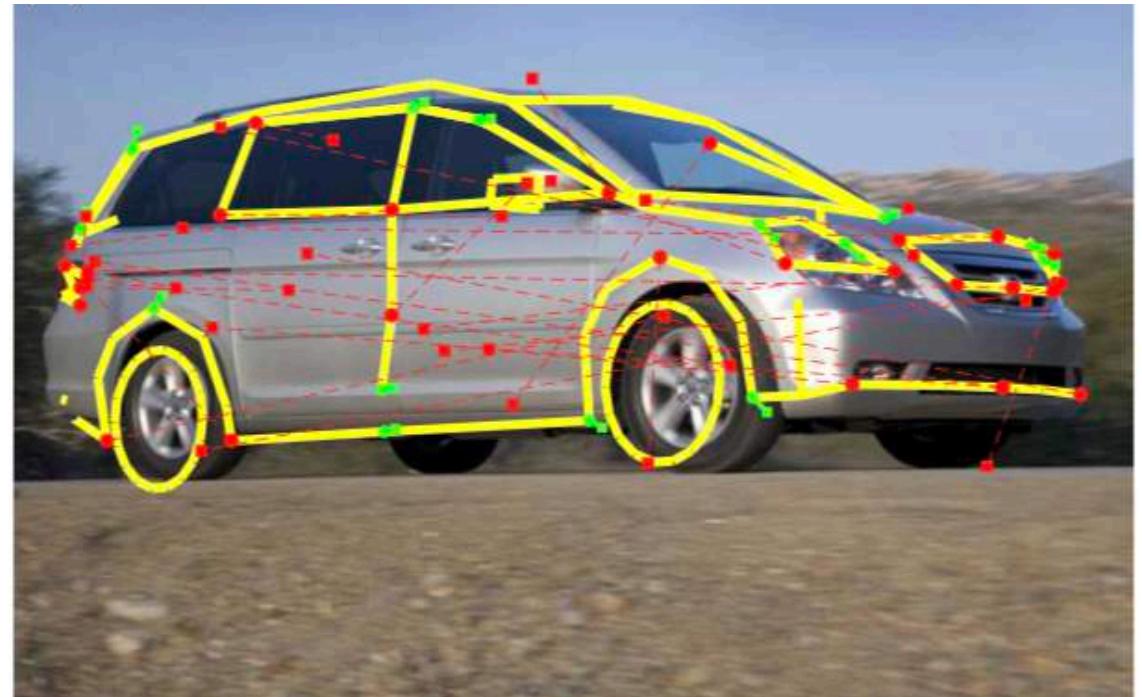
implementation available in Matlab and GTSAM

Runtime



GNC constant

Outline



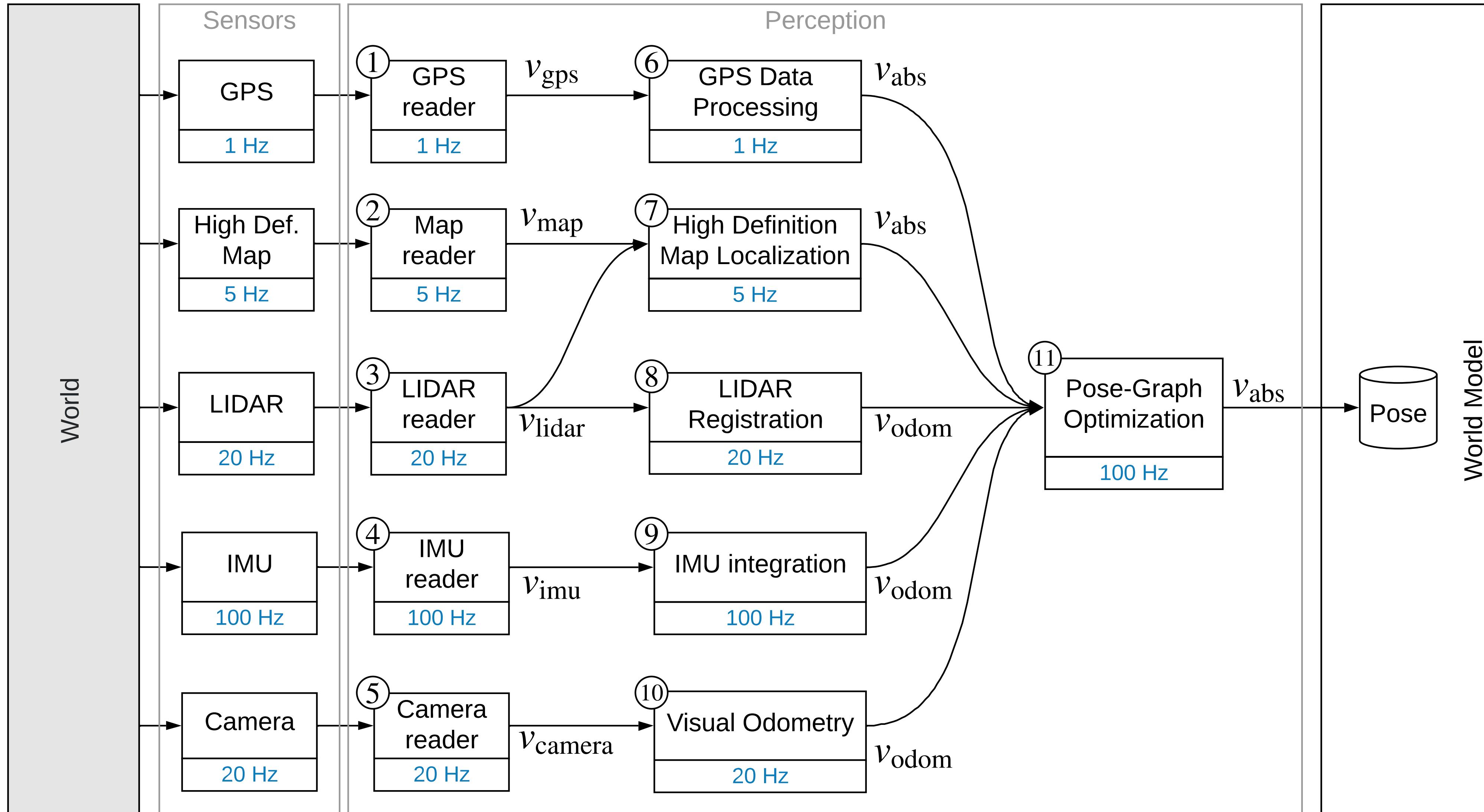
Certifiable Algorithms for
Geometric Perception



Monitoring and Diagnosability
of Perception Systems

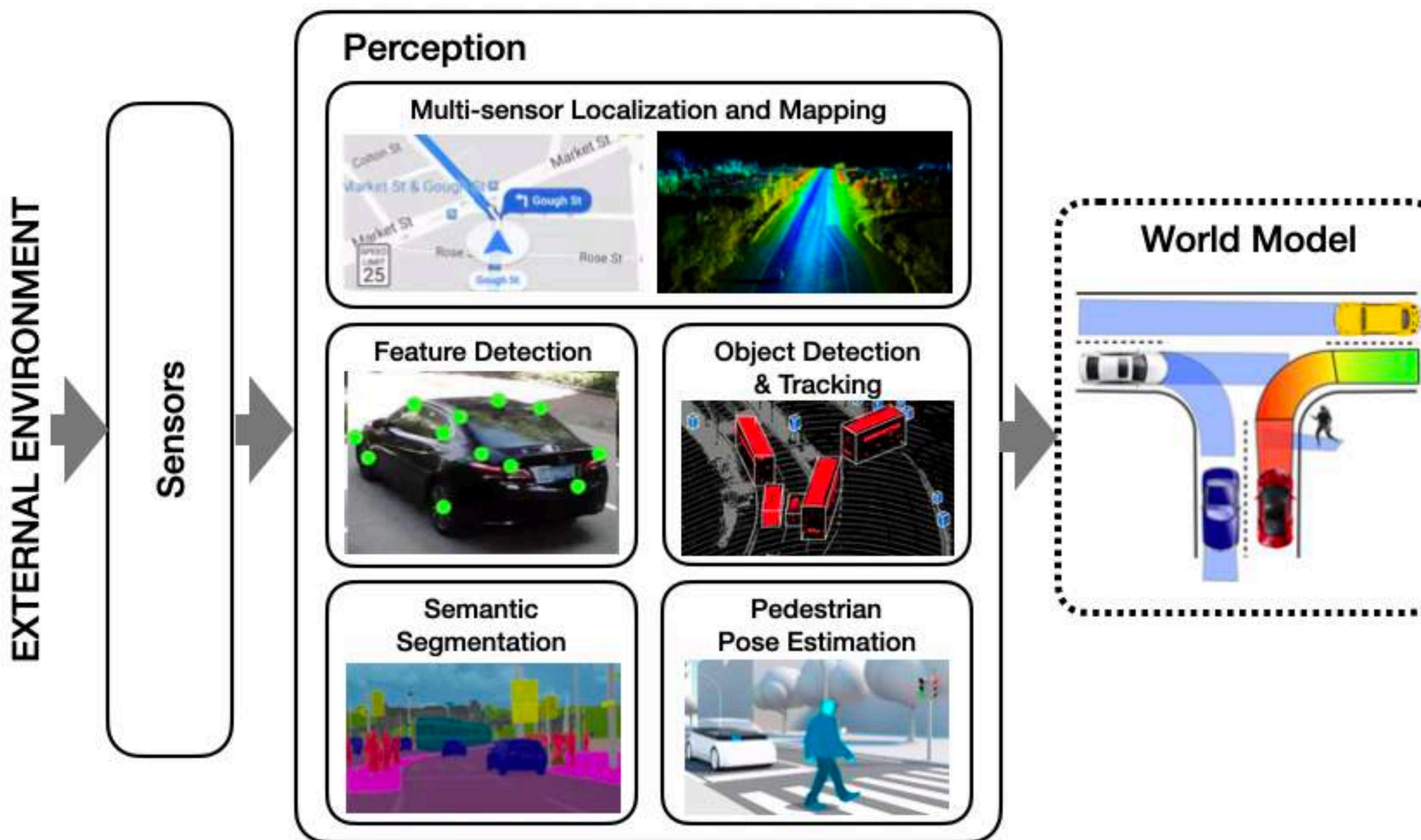
Monitoring of Perception Systems

Multi-sensor SLAM pipeline



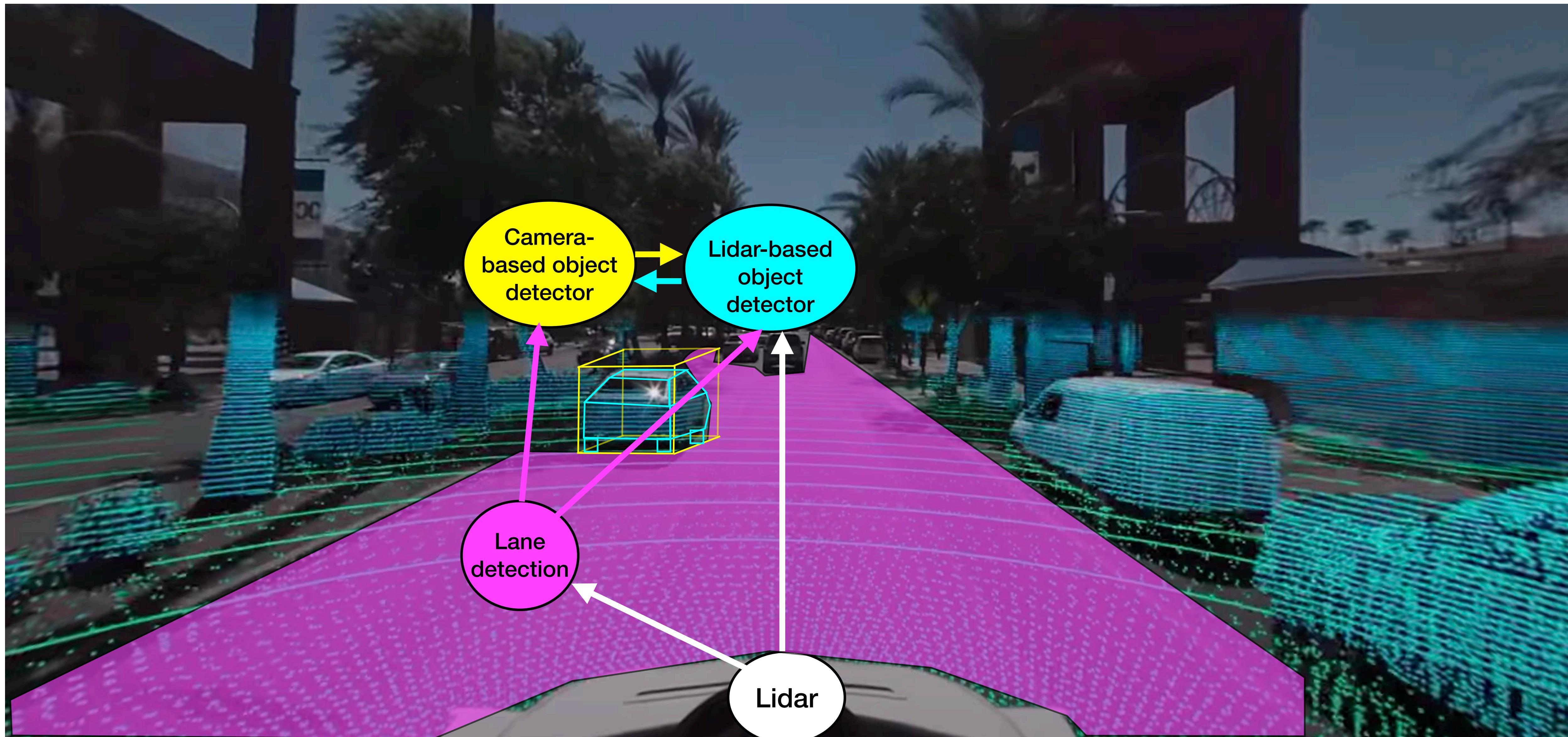
Monitoring of Perception Systems

- Other perception subsystems: pedestrian/vehicle detection, localization and tracking, lane detection, traffic lights and sign detection, ...

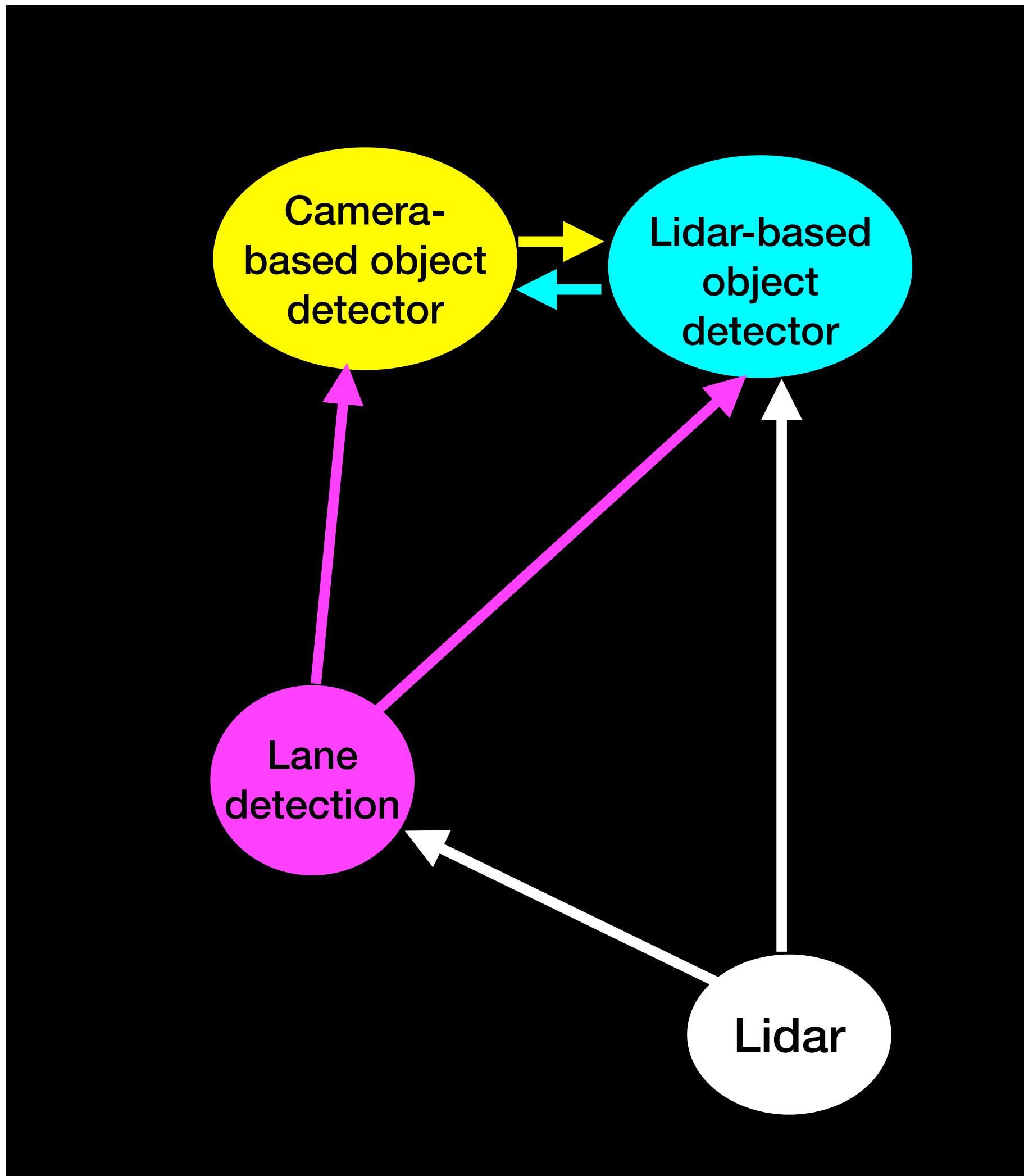


- **Q1:** can we check if our world model is correct?
- **Q2:** can we detect if some modules are producing incorrect results?
- **Q3:** what is the maximum number of faults that can be detected?

Diagnostic Graphs: Intuition



Diagnostic Graphs



- Each node represents a **module** in the perception system
 - can be: faulty/fault-free
- Each edge corresponds to a **test**
 - intuitively: tests check consistency between modules
 - test can be unreliable if computed by a faulty module
- Collection of all tests: **syndrome**

- Based on: PMC model (1967) for monitoring of multi-processor systems

Diagnosability

- A diagnostic graph is called **k-diagnosable** if given a syndrome, one can uniquely identify the faulty modules, assuming there are less than k faults.

Diagnosability is less than half
the number of modules

Diagnosability is less than the
minimum in-degree

Theorem 2 (Characterization of κ -diagnosability [50]). Let

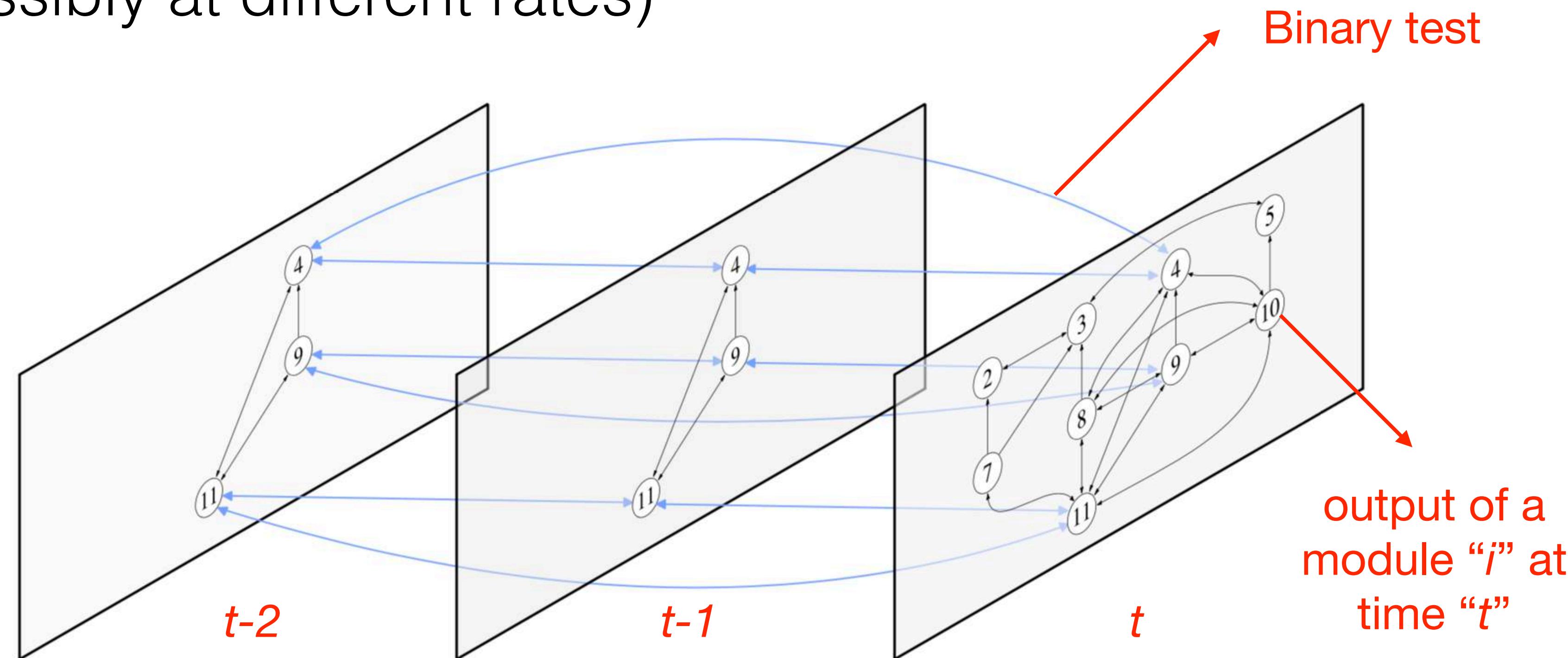
$D = (U, E)$ be a diagnostic graph with $|U|$ processors. Then
 D is κ -diagnosable if and only if

- (i) $\kappa \leq \frac{|U|-1}{2}$;
- (ii) $\kappa \leq \delta_{\text{in}}(D)$; and
- (iii) for each integer p with $0 \leq p < \kappa$, and each $X \subset U$ with $|X| = |U| - 2\kappa + p$ we have $|\Gamma(X)| > p$.

- Given a diagnostic graph, there exist fast algorithms to
 - compute diagnosability (max # of faults that can be uniquely identified)
 - identify the faulty modules
- Diagnosability: measure of robustness of a perception system

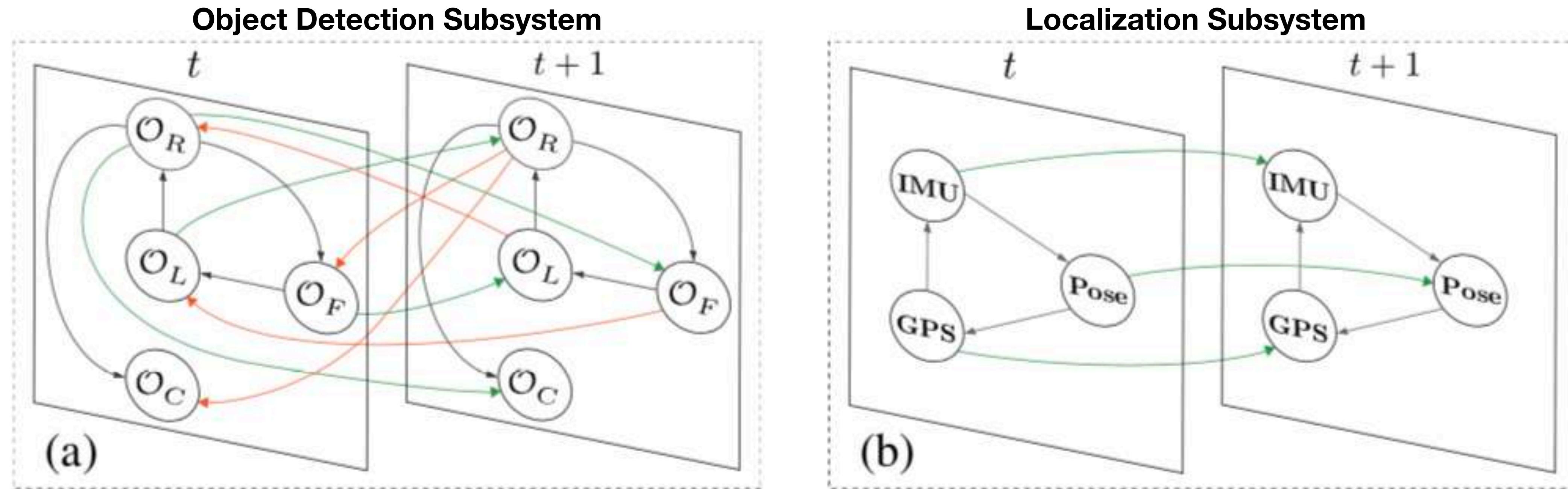
Temporal Diagnostic Graphs

- We extend diagnostic graphs to perception modules producing data over time (possibly at different rates)



- Theorem** [informal]: connecting diagnostic graphs over time, can only increase diagnosability.
- PerSyS:** PERception SYstem Supervisor

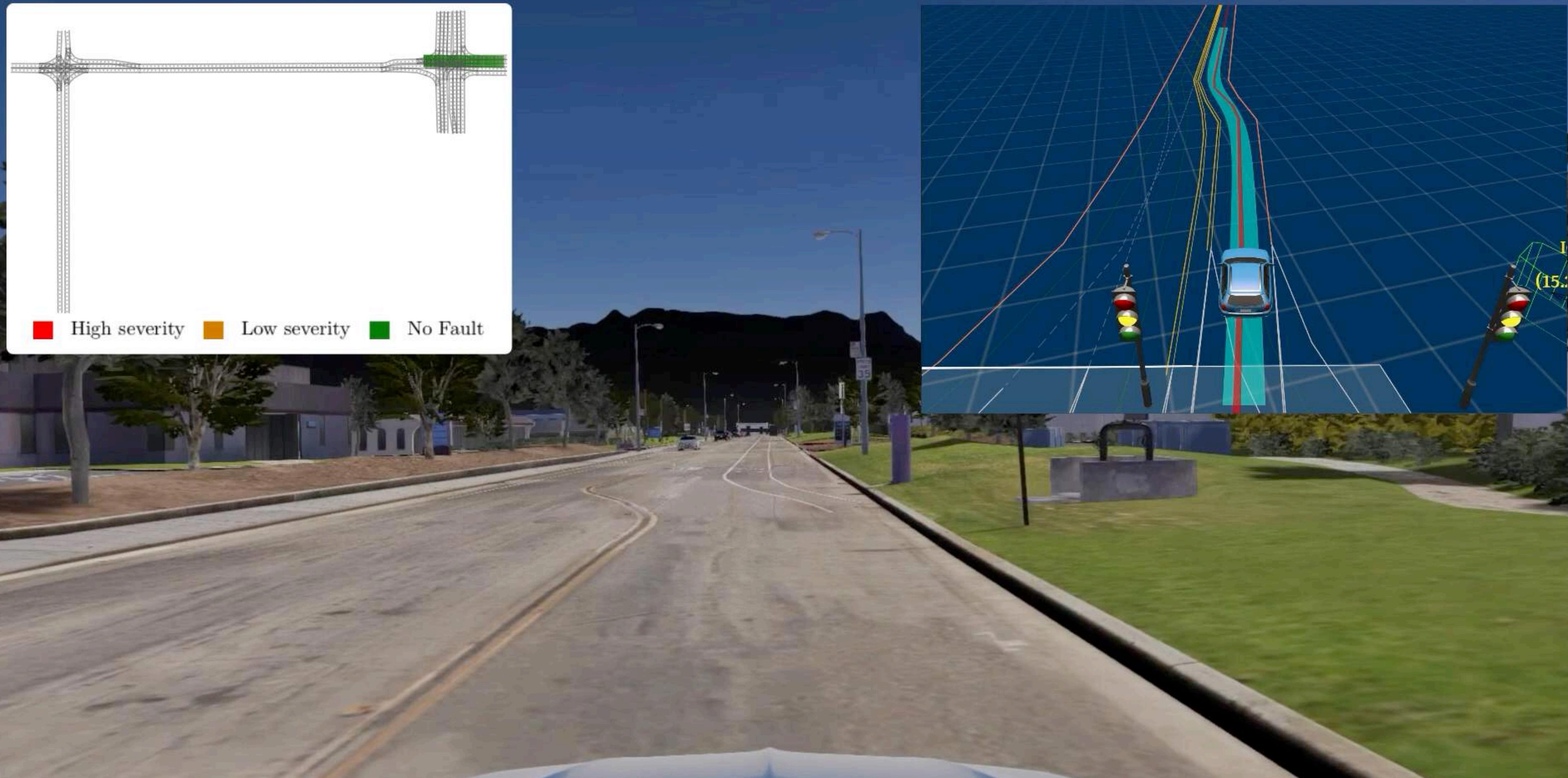
PerSyS: Evaluation



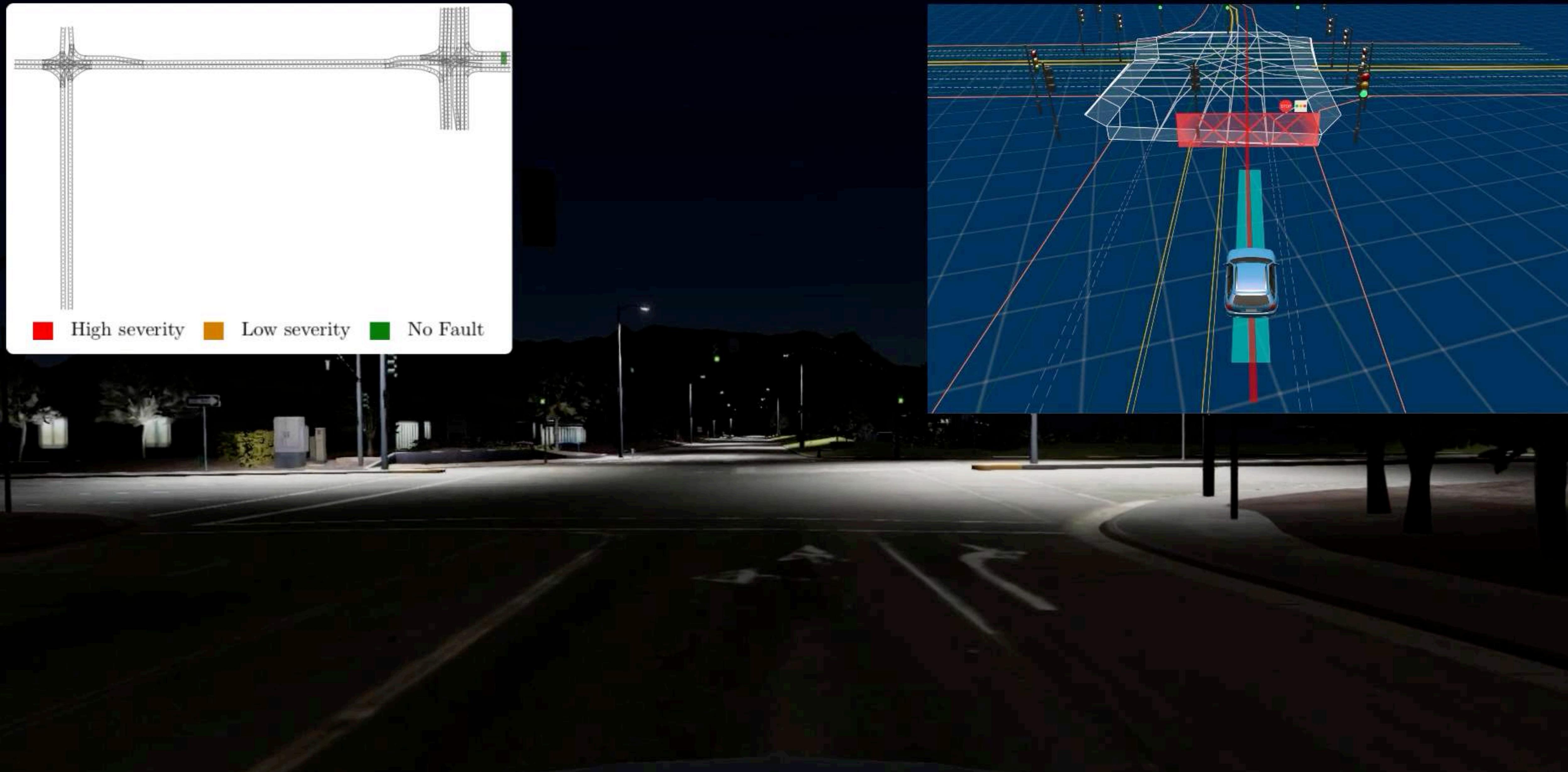
We test PerSyS using:

- **Perception system:** Baidu's Apollo Auto autonomous driving stack
- **Simulator:** LG's LGSVL Simulator (Lincoln MKZ car)
- **PerSyS** executes in real-time on a linux machine

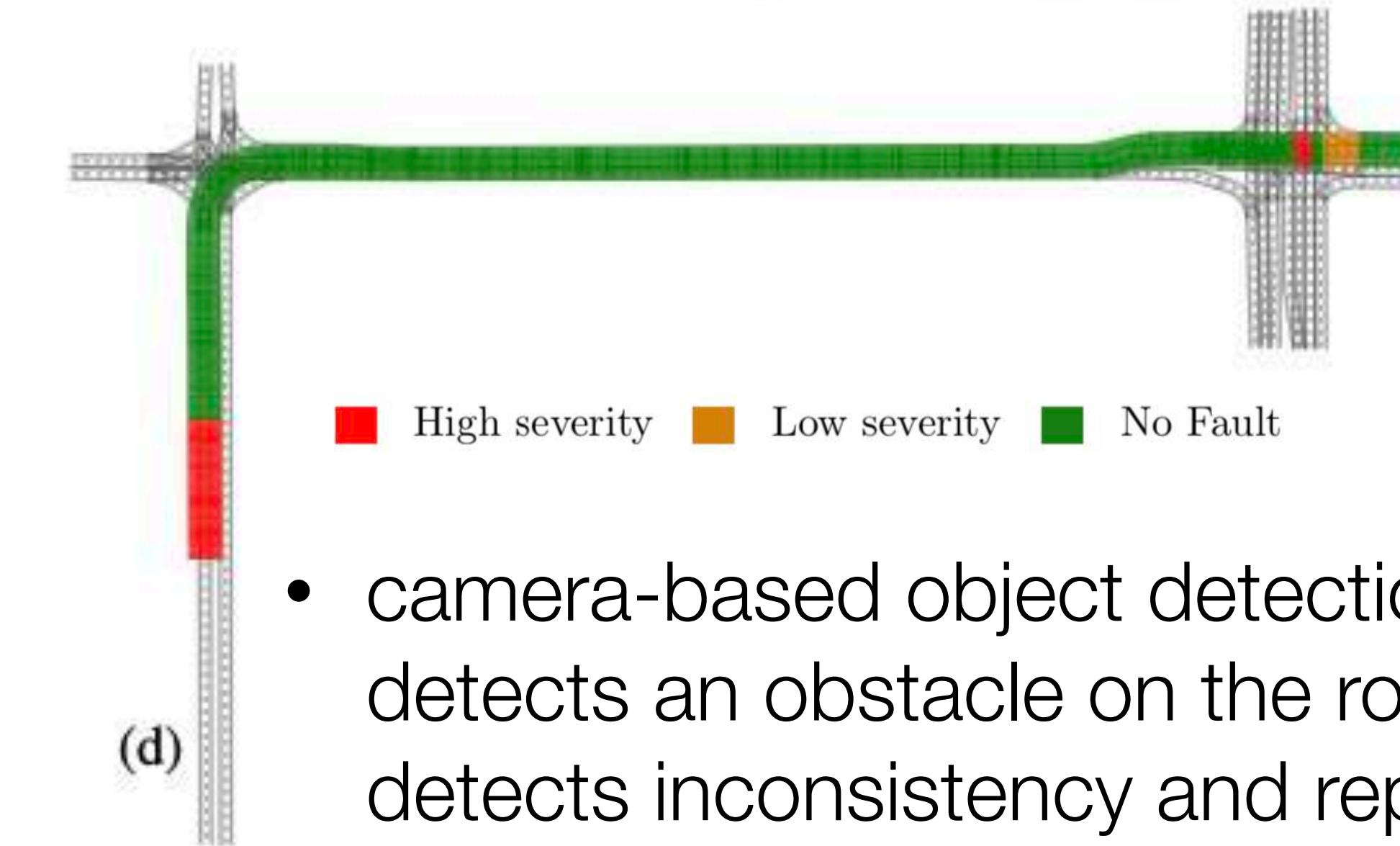
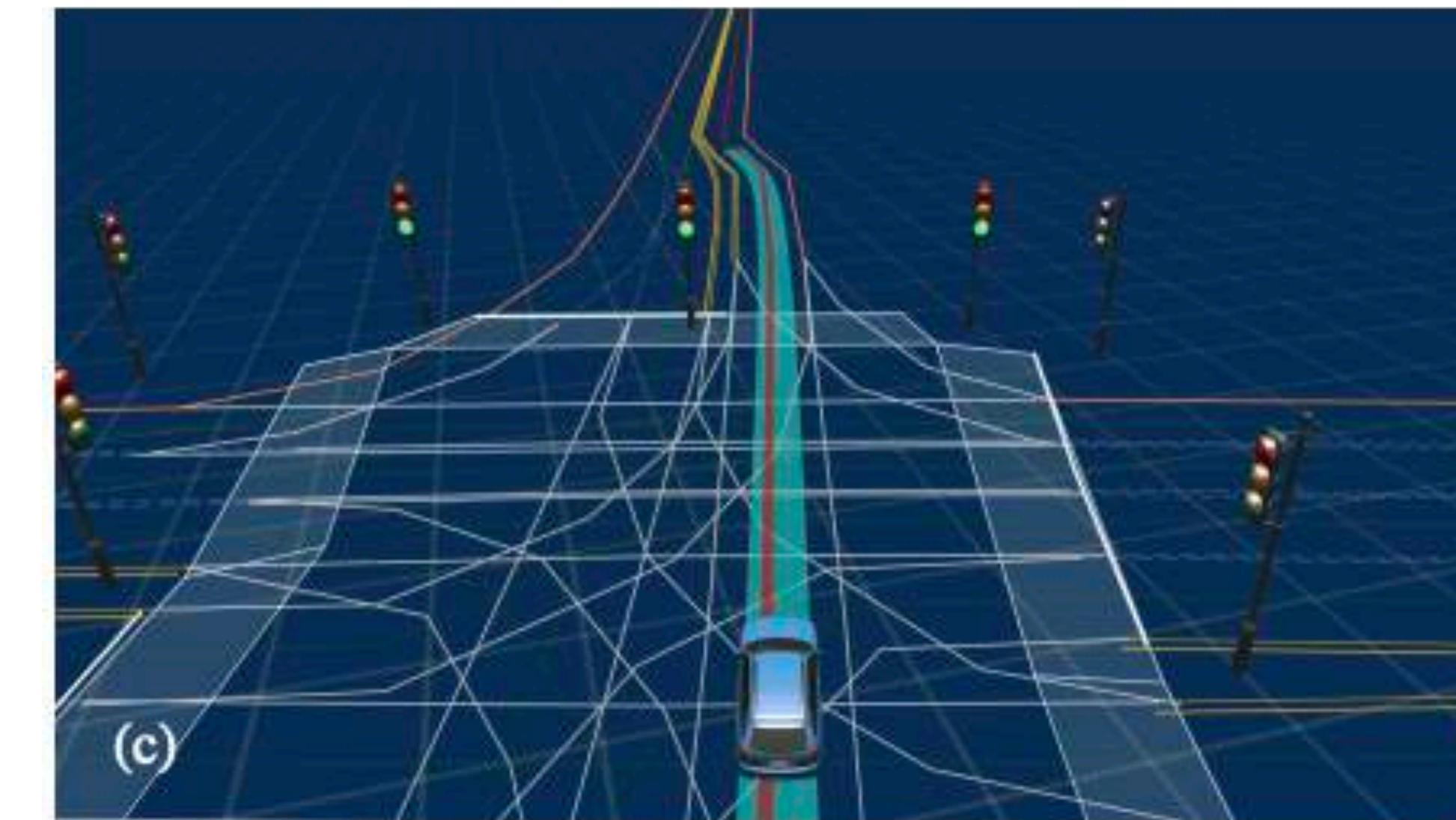
PerSyS: Evaluation - daytime



PerSyS: Evaluation - nighttime

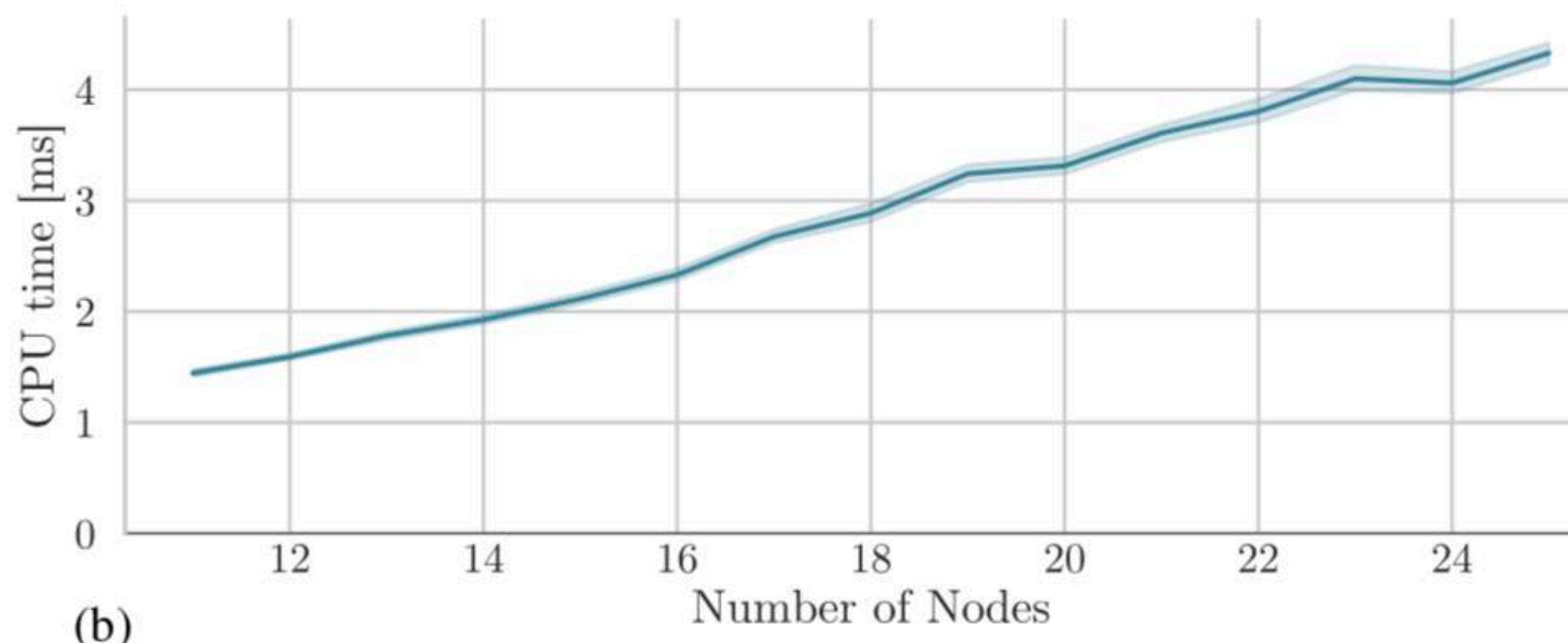
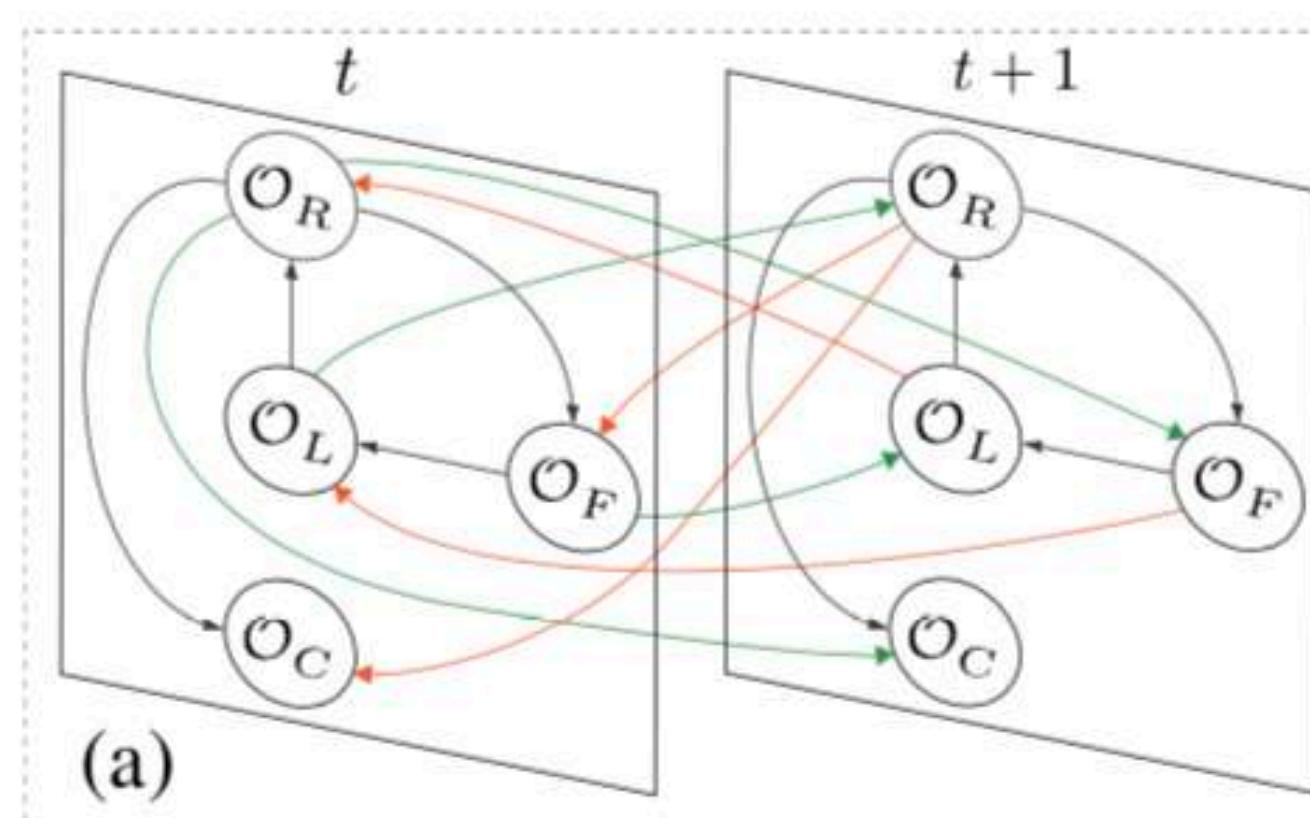


PerSyS: Evaluation - rain



PerSyS: Evaluation

Object Detection Subsystem



Scenario	False non critical (%)	False critical (%)	Correct critical (%)	Avg. time to impact from failure (s)
Simple	43	0	100	4.17
Night	4.44	0.58	99.42	3.65
Rain	49.55	8.47	91.53	4.04
Traffic	67.67	5.04	94.96	4.11

TABLE I
OBJECT DETECTION RESULTS AGGREGATED BY SCENARIO.

PerSyS able to identify critical faults with high probability

PerSyS may produce false alarms: issue mitigated by distinguishing critical from non-critical faults

Conclusion

- **Certifiable algorithms for geometric perception:** leverage tools from graph theory and optimization to enable accurate estimation (for SLAM and object detection) in the presence of many outliers
 - ➡ Graduated non-convexity, graph-theoretic outlier pruning, certification
- **Monitoring and diagnosability of perception systems:** temporal diagnostic graphs to detect and identify faults and “measure” how robust a system is
 - ➡ Initial results on diagnosability of object detection and localization

Thank you!

