Maneuver and Data Optimization for High Confidence Testing of Future Automotive Cyber-Physical Systems

Ilya Kolmanovsky (PI), Ella Atkins (Presenter, Co-PI), Barzan Mozafari (Co-PI), Mark Oliver (AVL)

University of Michigan
(Contact: ematkins@umich.edu)

Motivation and Objectives

- The current lack of toolchain for high confidence testing, validation and verification of advanced, connected and automated/autonomous vehicles can impede and even entirely prevent the introduction of such vehicles into mass production.

- Project Objectives:

(i) Game theory-based simulation environment to inform in-traffic relevant trajectories.

(ii) Model-free trajectory optimization techniques for actively falsifying time domain specifications.

(iii) CPS Smart Black Box with sampling-based vehicle data acquisition and management strategies to uncover faults in both existing and future vehicle fleets.

Game Theory Based Traffic Simulator for V&V of Automated Driving Algorithms

To facilitate the testing of future vehicles, we develop a traffic simulator that can be used to evaluate the performance of various automated driving algorithms in different traffic scenarios.

The simulator is focusing on the modeling of driver-driver & driver-automation interactions by exploiting the hierarchical reasoning game theory.

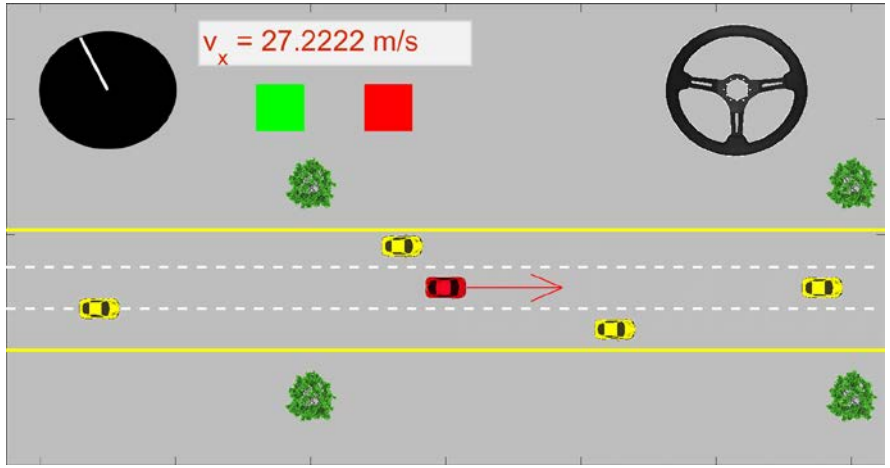


Fig.1: The testing of a car (in red) controlled by an automated driving algorithm based on decision tree approach in the traffic simulator.

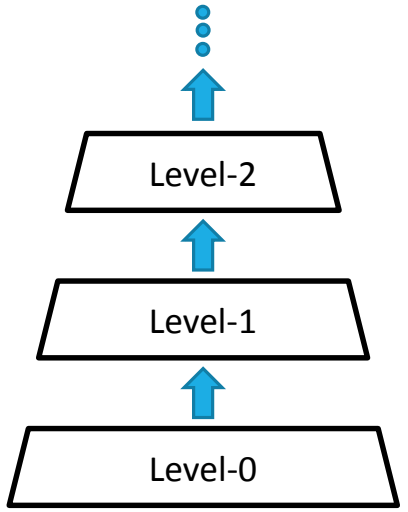


Fig.2: The reasoning depth hierarchy

- A level-k driver predicts the decisions of all other drivers by assuming them be level-(k-1), and makes decisions as the best response to their decisions based on the prediction.
- Different driver type (level-0,1,2,…) represents different driving habit and proficiency of a driver.

Game Theory Based Traffic Simulator for V&V of Automated Driving Algorithms

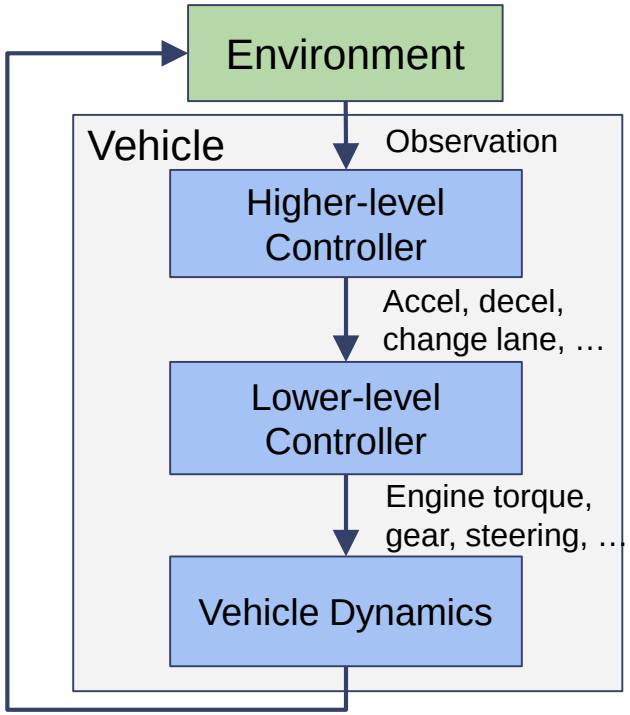


Fig.3: The control structure of a car

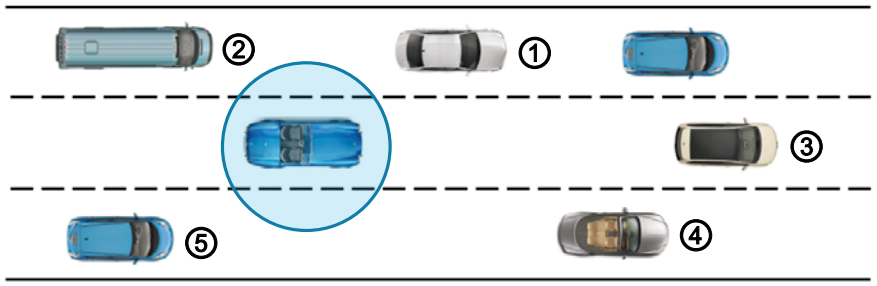


Fig.4: The observation space of a car

- The underlying dynamics of the traffic is assumed to be Markov, while each driver is assumed to only be able to observe the information from his/her vicinity, i.e., the system states are only partially observable to each agent.

- Reinforcement learning is used to solve for the control policies of level-k (0,1,2,…) sequentially.
- By assigning policies of different levels to the drivers, we can model traffic consisting of different human drivers.

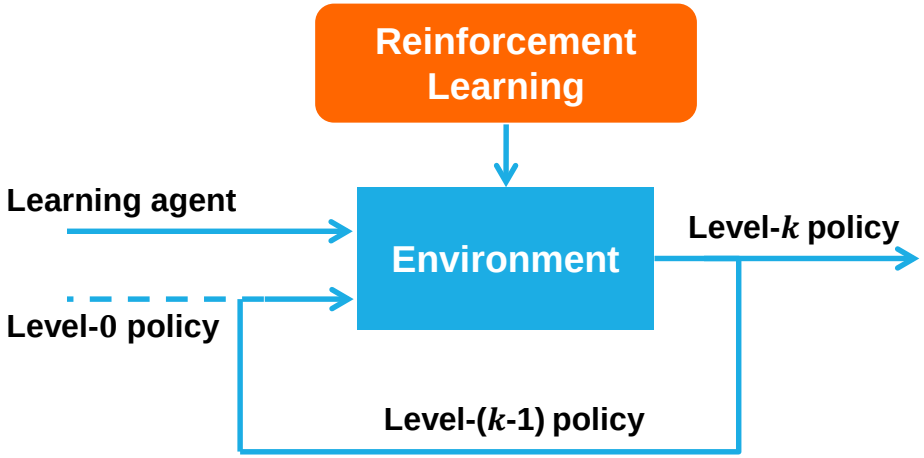


Fig.5: Algorithm to solve for the level-k policies

Presentation Focus:

The Smart Black Box

The Aircraft Black Box - History

The Black Box was proposed by the Civil Aeronautics Board (CAB) in the late 1940's and was adopted by the late 1950's.

- Original flight data: Engine power (RPM), angle of attack, pitch, roll, yaw, longitudinal trim, elevator position, aileron position, rudder position, air temperature, wing flap position.
- Secondary data: Thrust reverser, fuel flow, power lever, cabin pressure, master fire warning, autopilot state, hydraulic pressure, CG, speed brake, engine vibration, gross weight, smoke detection, yaw damper, electrical power, engine fire warning
- Used for maintenance and accident investigations; anomalies could also be documented
- Flight data recorder (FDR) "tapes" originally saved about 60 hours of data as a circular buffer
- Cockpit voice recorders (CVRs) also adopted with a similar circular buffer format

Primary Sources:

- B. Allen and J. Leak. "The potential role of flight recorders in aircraft accident investigation", Aviation Safety Meeting, Meeting Paper Archive, AIAA, 1966 http://dx.doi.org/10.2514/6.1966-810
- R. D. Morris. "Parameter selection for in-flight recording", Journal of Aircraft, Vol. 1, No. 5 (1964), pp. 300-303. http://dx.doi.org/10.2514/3.43597

Data Caching in the 21st Century

Broad suite of data now stored in cars, ships, & planes.

- Comprehensive low-bandwidth data stored to support diagnostics, e.g., CAN bus data on a car.
- Experimental vehicles equipped with high-bandwidth video, LIDAR, radar also captured in raw form over short durations (!).
- Data typically stored onboard then downloaded after a drive concludes.
- Low-bandwidth maintenance and geo-location messages transmitted to a company server (e.g., ACARS for aircraft, fleet vehicle GPS for commercial trucks).

Remaining Challenges.

- High-bandwidth raw data cannot be stored onboard long-term – just too much data (1 Tbyte per second!)
- Storing only “output” (e.g., objects) cannot reveal errors in data processing (e.g., false positives, missed detections)
- Exploit the V2C2V link to capitalize on “infinite storage capacity cloud,” though data uplink/storage still has cost

Smart Black Box – Data Acquisition and Pre-processing

Raw Lidar frames

Raw camera frames

Data reader, pre-processor

Raw LIDAR data → Object detection → Object data

Raw image data → Object detection → Object data

CAN Bus data

Mobileye data

Object data:

data describes the position and size of objects

500 ms

1000 ms

CAN Messages:

8 bytes/message

	0	1	2	3	4	5	6	7	
2 ms									e.g. ID0, engine & braking
4 ms									e.g. ID1, electronics
6 ms									e.g. ID2, navigation & control
8 ms									e.g. ID0, engine & braking
10 ms									e.g. ID1, electronics
12 ms									e.g. ID2, navigation & control

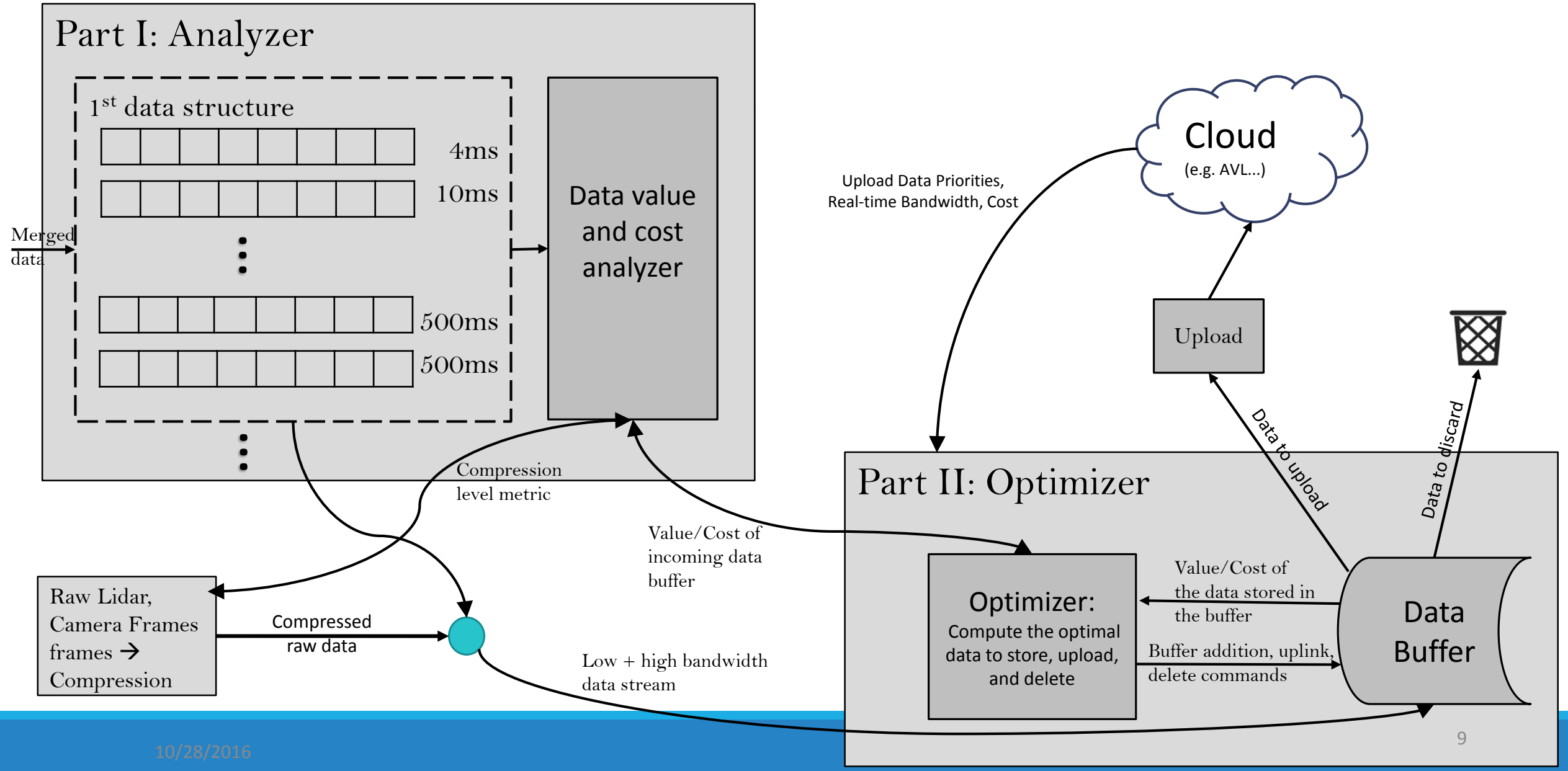
Merge into same time stamp

Merge into same time stamp

Smart Black Box Part I: Analyzer

Smart Black Box: Raw Data Compression (Mozafari et al)

Smart Black Box – Data analyzer and optimizer

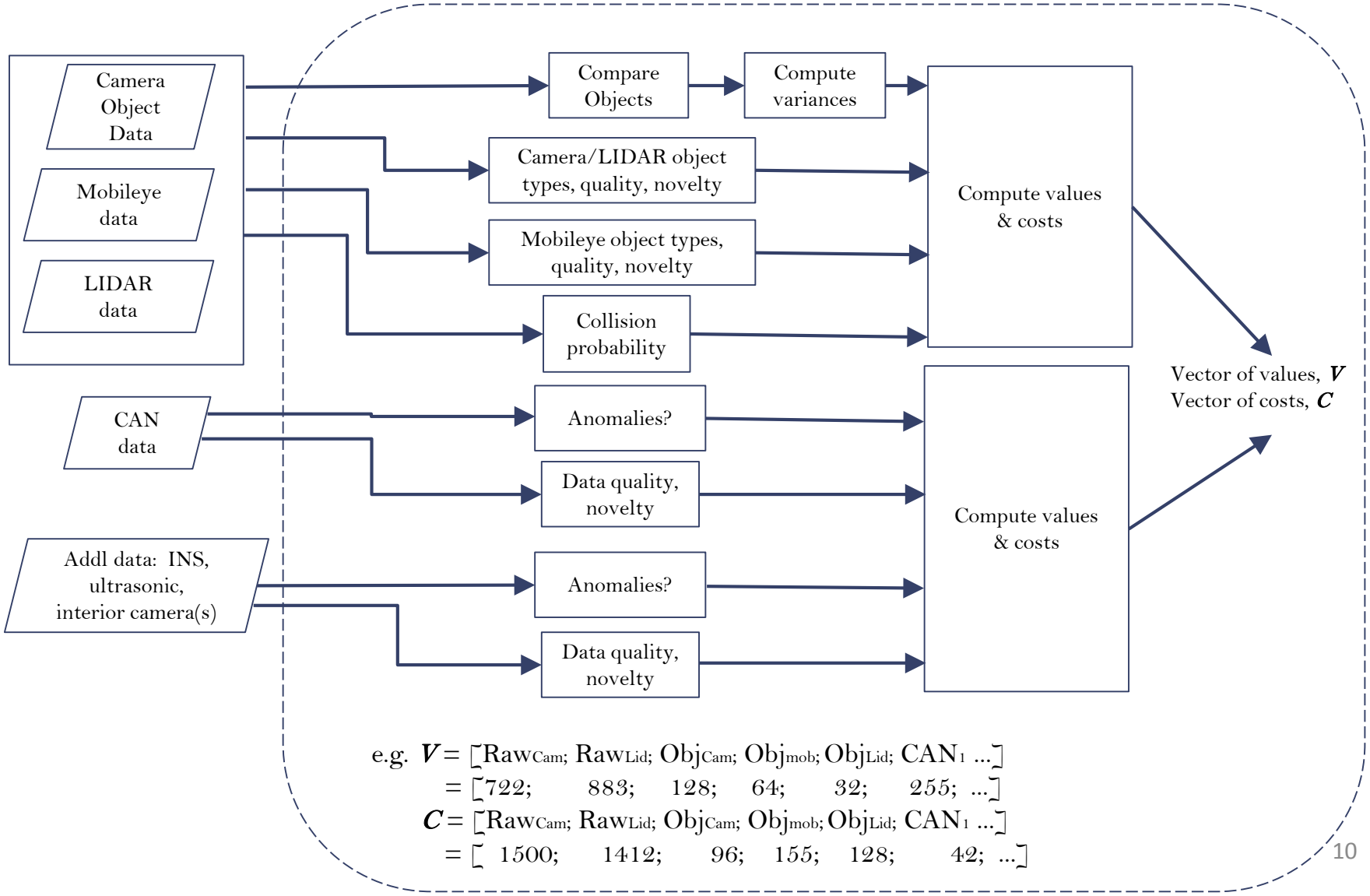


Smart Black Box – Data value/cost analyzer

* Factors considered
  - Data agreement
  - Upper and lower bound
  - Probability
  - Boolean flag status
  - Variance of same kind of data from different sensors
  - …

* Simulations → generate sample (CAN, object) datasets

* Experiments / large data set access → critical for realistic analysis



Smart Black Box Part II – Data Storage Optimizer

- Suppose value vector V and cost vector C have been defined.
- Optimizer (Part II) objective: merge data to maximize overall value and minimize cost:
  - Per byte of data stored onboard each vehicle
  - Per byte of data uplinked to the cloud
- Asynchronous data management:
  - *Stage 1:* Cache data streams from each sensor in a [circular] buffer
  - *Stage 2:* Migrate data of sufficient value to non-volatile memory onboard or immediately uplink; overwrite remaining data
  - *Stage 3:* Uplink data cached onboard and clear from memory
  - *Garbage collection:* Delete lowest-value "aging" data from onboard memory to free space as needed

Smart Black Box Optimizer – Low-bandwidth mode (driving)

Mode description

- With little to no connection to the cloud, incoming data must be stored locally. Data must be discarded as needed to prevent memory/disk overflow.

Formulation

- $d_{out}(t_k)$ = NULL; no data can be uploaded
- State: $\overrightarrow{X(t_k)} = \{V_{local}(t_k), C_{local}(t_k), M(t_k)\}$
- Action: $\overrightarrow{U(t_k)} = \{r_{in}(t_k), d_{trash}(t_k)\}$

Optimization

- $\overrightarrow{U(t_k)} = \underset{\overrightarrow{U(t_k)}}{\arg\max}(F_{local}(t_k)) = \underset{\overrightarrow{U(t_k)}}{\operatorname{argmax}}\big(V_{local}(t_k) - C_{local}(t_k)\big)$

Smart Black Box Optimizer – High-bandwidth mode (driving)

Mode description

- When the car is driving with strong low-cost internet connectivity, the optimizer can store valuable data in onboard memory and also upload the most valuable data to the cloud.

Formulation

- State: $\overrightarrow{X(t_k)} = \{V_{local}(t_k),\ C_{local}(t_k),\ V_{cloud}(t_k),\ C_{cloud}(t_k),\ M(t_k)\}$
- Action: $\overrightarrow{U(t_k)} = \{r_{in}(t_k), d_{out}(t_k), d_{trash}(t_k)\}$

Optimization

- $\overrightarrow{U(t_k)} = \underset{\overrightarrow{U(t_k)}}{\arg\max}(F_{cloud}(t_k) + \lambda * F_{local}(t_k)) = \underset{\overrightarrow{U(t_k)}}{\operatorname{argmax}}\left(V_{cloud}(t_k) - C_{cloud}(t_k) + \lambda * \left(V_{local}(t_k) - C_{local}(t_k)\right)\right)$
- where $\lambda \in R, \lambda > 0$ is a user-defined weight to establish relative priority for local vs. cloud-based data storage.

Smart Black Box Optimizer – "Garaged" mode

Mode description

- When the car is not running and a high-bandwidth low-cost internet connection is available (e.g. in the owner's garage), the smart black box can transmit all valuable data to the cloud over time.

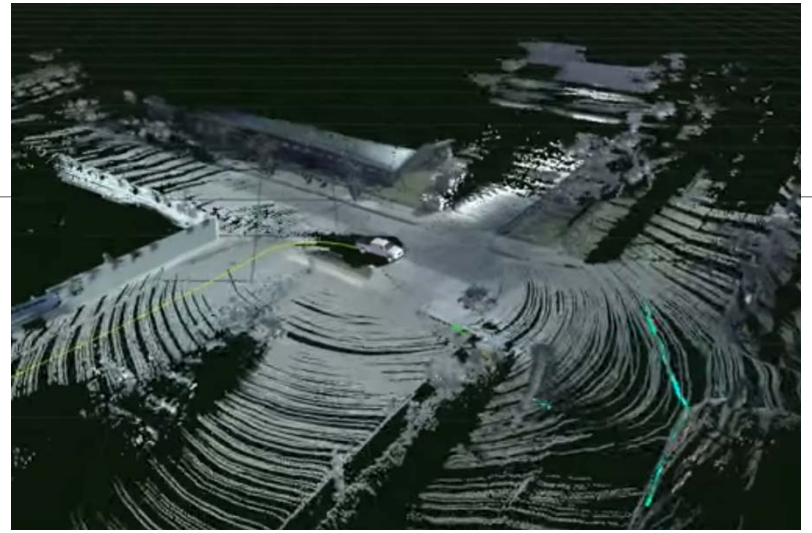
Formulation

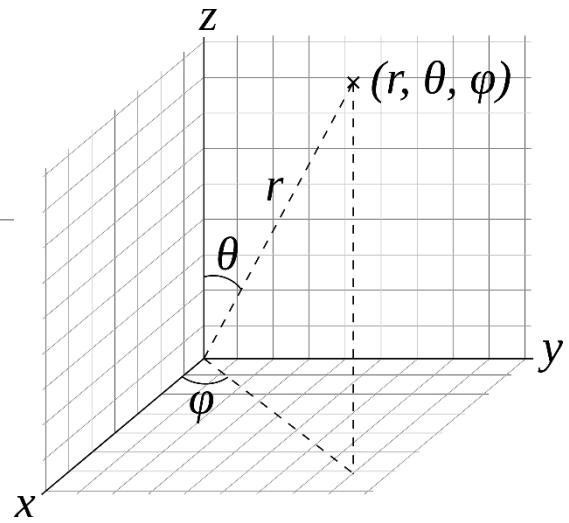
- $r_{in}(t_k)$ zero: no new data
- State: $\overrightarrow{X(t_k)} = \{V_{cloud}(t_k),\ C_{cloud}(t_k)\}$
- Action: $\overrightarrow{U(t_k)} = \{d_{out}(t_k), d_{trash}(t_k)\}$

Optimization

- $\overrightarrow{U(t_k)} = \underset{\overrightarrow{U(t_k)}}{\arg\max}(F_{cloud}(t_k)) = \underset{\overrightarrow{U(t_k)}}{\operatorname{argmax}}\big(V_{cloud}(t_k) - C_{cloud}(t_k)\big)$

Lidar Data





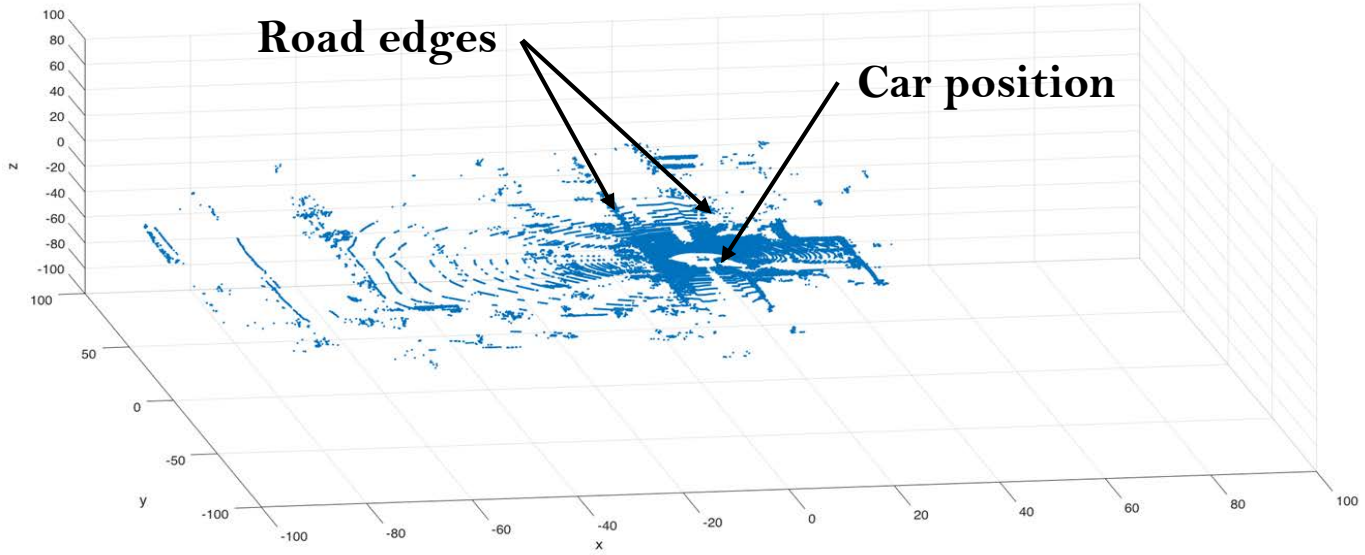
Ford Campus Vision & Lidar Data:

- 360° horizontal field of view
- 26.8° vertical field of view
- 0.08° horizontal angular resolution
- 0.4° vertical angular resolution
- 10 Hz

Measures the distance to objects

at certain angles:

$r = \text{dist}\,(\theta, \varphi)$



Why save Lidar data?

Potentially useful for:

Accident Diagnosis:

- Uptake collects sensor data up to for 24 months

Anomaly Detection:

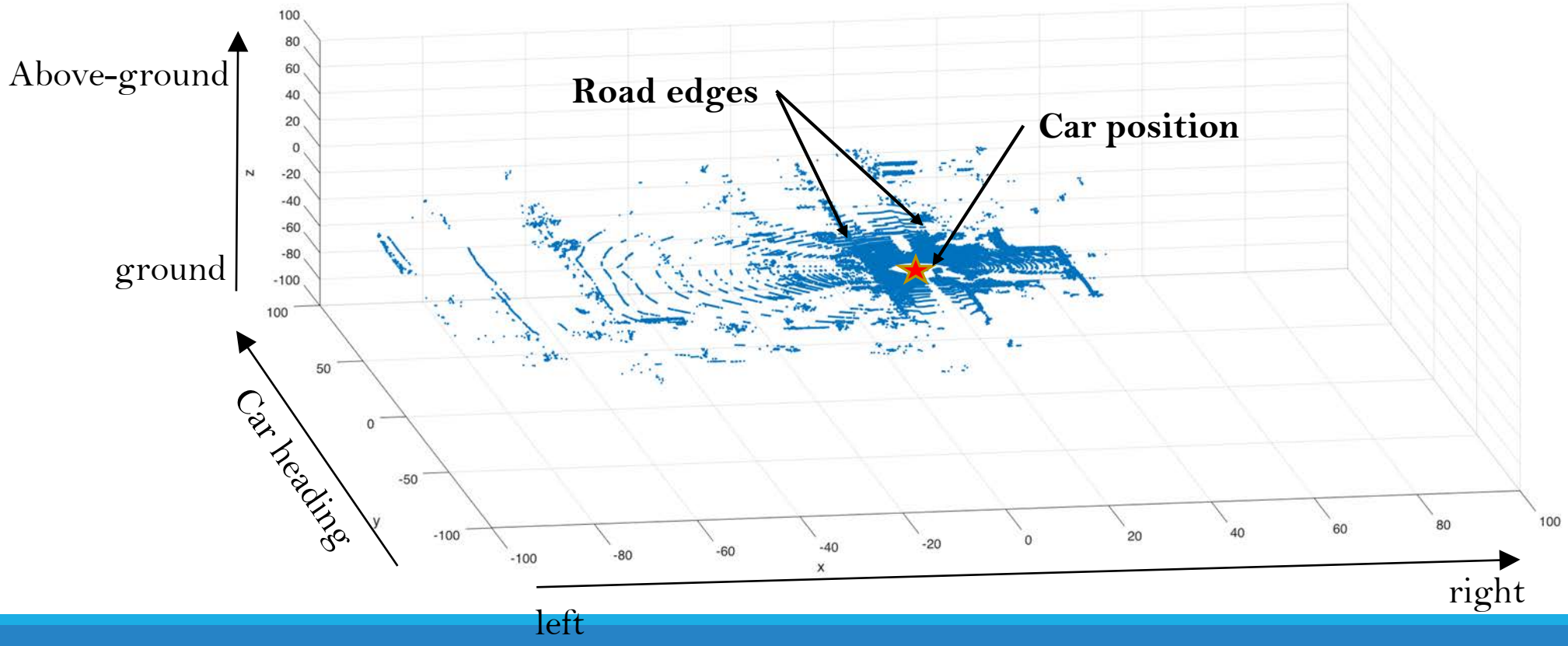
- Boeing downloads 1TB of data for every flight

Driving Pattern Analysis:

- Nationwide provides member discounts based on their driving patterns

Lidar based obstacle extraction - review

- Ford Campus Vision and Lidar Data
  - http://robots.engin.umich.edau/SoftwareData/Ford
- Lidar Reference Frame



Size of Lidar Data

80,000 of (x, y, z) coordinates, every 0.1 second

Say, a floating point takes 4 bytes, then

- 34.56 GB / hour
- 3.1 TB / month *(with 3 hours drive every day)*
- USD 100 / month for *every* vehicle *(newegg HDD price)*

Hard to compress due to few repetitions

- So, we perform **Data Reduction**

Possible Ideas for Data Reduction

1. Random Sampling
   - Simple, Fast
   - No guarantees on errors *(different from aggregation)*

2. Image Processing
   - Well-developed theories/algorithms
   - We lose distance information completely

3. Choosing a subset *S* (of size *K*) that *minimize errors*

$$\min_S \sum_\theta \sum_\varphi (\mathrm{dist}(\theta, \varphi) - R_S(\theta, \varphi))^2$$

where $R_S(\theta, \varphi)$ is *an estimate* based on S.

   - Small errors on average compared to random sampling
   - No idea on how bad / good

Our Proposal: Absolute Guarantee

Absolute Guarantee on Error:

$$\text{argmin}_S \sum_{\theta} \sum_{\varphi} \left(\frac{R_S^l(\theta,\varphi) + R_S^u(\theta,\varphi)}{2} - \text{dist}\,(\theta,\varphi) \right)^2$$

such that

$$\boldsymbol{R_S^l(\theta,\varphi) \leq\ dist(\theta,\varphi) \leq R_S^u(\theta,\varphi)} \qquad \text{and} \qquad |S| = K$$

$\text{dist}\,(\theta,\varphi)$: True distance at angle (θ,φ)

$R_S^l(\theta,\varphi)$: Lower bound (estimated from S)

$R_S^u(\theta,\varphi)$: upper bound (estimated from S)

Benefits:

Not only a point estimate, but an error bound

Absolute bounds on calculating: (1) Distance to a front car, (2) Relative speed with respect to a front car

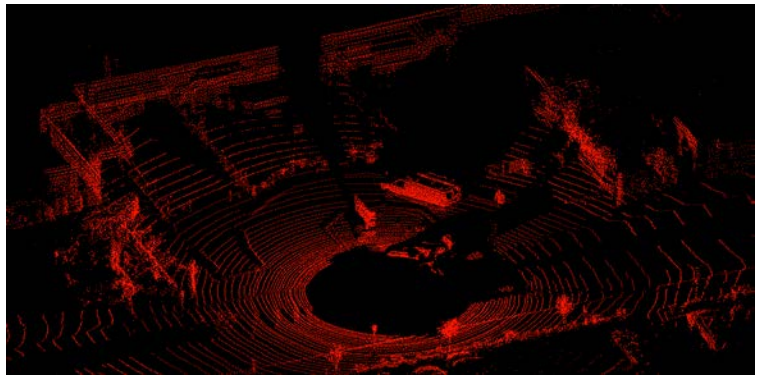
Estimating Miss Distances

The subset *S* doesn't include all data points.

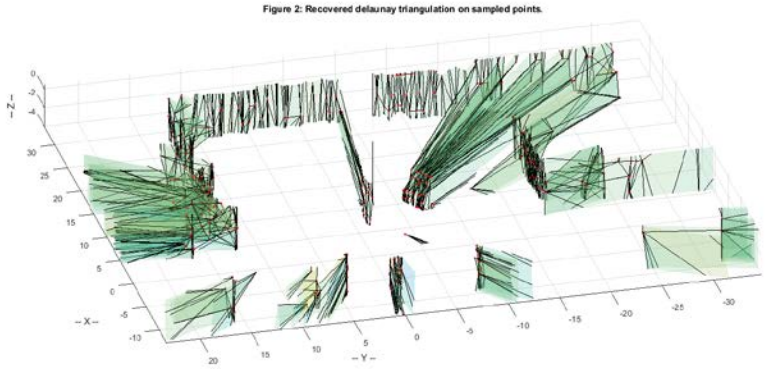
But, if S includes other points near (θ, φ),

we can get pretty good estimates.

Reconstruction Model: **Interpolation (in 3d)!**



Visualization of original data points



Our reconstruction by interpolating from sampled data points

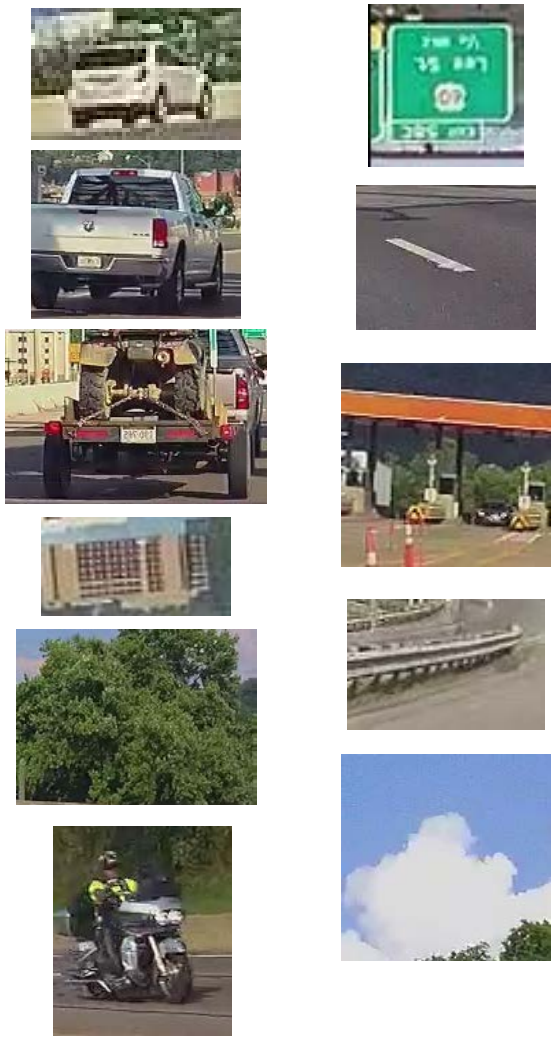
Ongoing Camera based object detection

- Training iterations and parameters
  - Less iterations, tune the parameters
- Performance
  - On existing test videos, the BVLC_caffenet trained on nasa_256 works much better than CIFAR10_caffenet trained on nasa_32.
  - It's hard to classify the aircraft types (bixler, sr2 or y6?)
  - The validation accuracy of CIFAR10_caffenet is around 75%, but the accuracy is much lower when testing
- Overfitting
  - Some of the training cases could be overfitted but we need more diverse test videos to prove that. For example,100,000 iterations is more likely to be overfitted, but 5,000 iterations also can overfit data.

Video object detection with Caffe Deep CNN – Car dataset

11 categories:

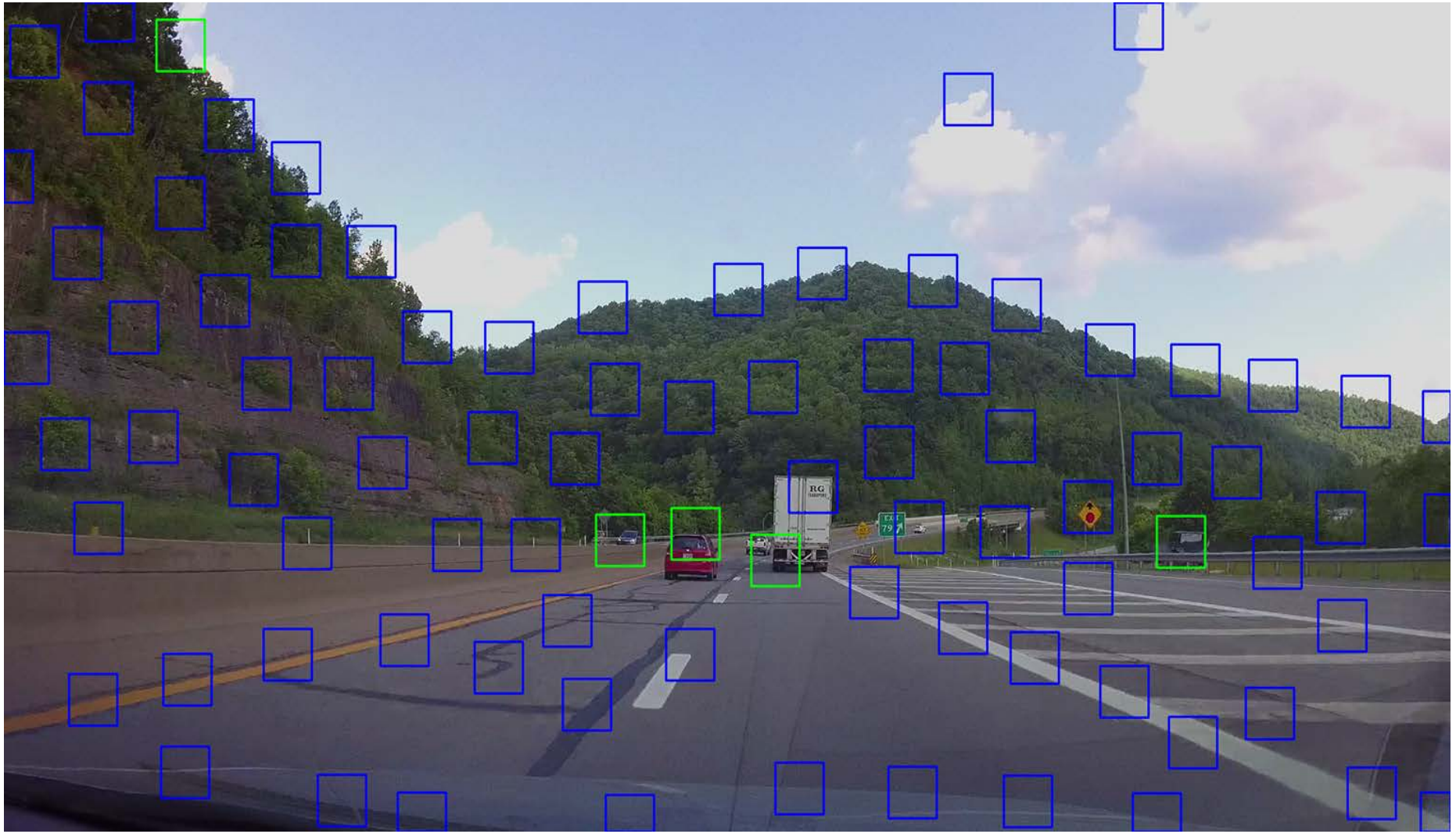
- Car(C)
- Truck(T)
- Wagon(W)
- Building(B)
- Tree(R)
- Traffic sign(N)
- Lane(L)
- Toll(O)
- Edge(E)
- Sky and cloud(S)
- Motorcycle(M)



Transformation

More diversity,
more information

Camera based object detection – Car detection results



Probabilities:
0.720(C),
0.509(T),
0.809(C),
0.524(C),
0.619(C)