EdgeCompression: An Integrated Framework for Compressive Imaging Processing on CAVs

Sidi Lu¹, Xin Yuan², Weisong Shi¹

¹Wayne State University, ²Nokia Bell Labs
Introduction: take CAVs as an example

- Fast high-resolution video processing

- Conventional approaches
  - Reduce frame rates
  - Reduce the frame size

  losing informative high-speed info and/or small objects in frames

- Compressive Imaging (CI) camera
  - Complicated algorithms to retrieve the desired signal
  - High energy consumption

  Optical-domain compressed images (measurements)

- Goal: AI + CI + CAVs
  - Accelerating accurate video analysis
  - Decreasing energy consumption
Key Concepts

• **Vehicle-EdgeServer-Cloud** *closed-loop* framework (EdgeCompression)

• Experiments with four public datasets
  • Detection accuracy of measurements generated by the CI camera
    o Close to the accuracy on *reconstructed videos*
    o Comparable to the true value

• Framework and methodology can be applied to regular datasets of the traditional camera

The proposed approach is **generic**
Outline

• EdgeCompression Framework
• Experiment Setup
• Results and Main Observations
• Conclusion
Vehicle-EdgeServer-Cloud Framework

**Measurement**

- EdgeServer
  - YOLOv3: Object Detection
  - E2E-CNN: Reconstruction
  - Storage: Reconstruction Trigger
  - Measurements
    - CI Camera
    - Vehicle

**Vehicle Fleets**

- Cloud
  - Refined YOLOv3-Tiny
  - Measurements\((\Delta t)\)
  - Detection Results\((\Delta t)\)

**Local Processing**

- YOLOv3-Tiny
  - Object Detection
  - Detection Results
  - Notification
  - Feedback
  - Update YOLOv3-Tiny Model

- Object Detection
  - Y
  - N

**Reconstructed Video**

- Cloud
  - Measurement Detection Results
  - Verification
    - Y
    - N

- Cloud
  - Refined YOLOv3-Tiny
  - Updates YOLOv3-Tiny Model

- Cloud
  - Feedback
  - Notification

10/30/2020
Connected and Autonomous dRiving Laboratory
Different Roles in EdgeCompression Framework

1) Vehicles:
   • **Energy-efficient network**: make timely computation on measurements.

2) **EdgeServer** (more computational resources):
   • **Reconstruct** high-speed data with a triggered event
   • **Verify** the detection results of the Vehicle and send notifications

3) **Cloud**:
   • **Aggregates** all useful information
   • **Refine** the energy-efficient network on the Vehicle
   • **Big data analysis**: traffic control and path planning
Using video CI as an example

Video Compressive Imaging

1) Video are modulated at a higher speed than the capture rate of the camera

2) Modulated frames are then compressed into a single measurement

3) Multiple frames can be reconstructed from every single measurement.

• \( \otimes \): element-wise product

\( C_r \): compression ratio
1) **Save memory and bandwidth:**

Every $C_r$ frames collapsed to a single measurement

2) **Save the computation:**

The measurements captured by the CI cameras are already compressed

3) **Save the energy:**

- **Vehicle**: Detection directly on the measurements w/o reconstruction
- **EdgeServer**: only reconstruct video when the trigger is on
Dataset Selection

(a) AAU RainSnow Dataset

(b) BDD100K Dataset

(c) PDTV Dataset

(d) DynTex Dataset

- https://www.kaggle.com/aalborguniversity/aau-rainsnow
- https://bdd-data.berkeley.edu/
- http://www.tft.lth.se/english/research/video-analysis/cooperation/publicdataset/
Hardware Setup

We assume that a CAV is equipped with an Intel FRD, and the NVIDIA GPU Workstation is working as the EdgeServer.

<table>
<thead>
<tr>
<th></th>
<th>Intel FRD</th>
<th>NVIDIA GPU Workstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon E3-1275 v5</td>
<td>Intel Xeon E5-2690 v4</td>
</tr>
<tr>
<td>GPU</td>
<td>NONE</td>
<td>4 × 11 GB GeForce RTX 2080 Ti</td>
</tr>
<tr>
<td>Frequency</td>
<td>3.6 GHz</td>
<td>2.6 GHz</td>
</tr>
<tr>
<td>Core</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB</td>
<td>64 GB</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu 16.04.6 LTS</td>
<td>Ubuntu 16.04.6 LTS</td>
</tr>
</tbody>
</table>
Reconstruction Results

PSNR: signal-to-noise ratio between two images
Detection on the Measurements Directly
Detection Results

Grey-Scale Video (a)  |  YOLOv3-Tiny (trained on grey-scale video)

Measurement (b)  |  YOLOv3 (trained on grey-scale) (trained on measurements)

Reconstructed Video (c)  |  Upgrade model

Reconstructed Video (d)  |  Change training dataset of the model

Measurement (e)
Experiment results in terms of mAP

- **G:** grey-scale video
- **M:** measurements
- **R:** reconstructed video.
- mAP refers to the mean Average Precision.
Observation #1: TinyG-M vs. TinyM-M

- The mAP score of TinyM-M is significantly larger (almost double) than that of TinyG-M.

- Training the model specifically on the raw measurement is necessary for the accurate detection.
Observation #2: TinyM-M, TinyG-R, and TinyG-G

Vehicle detection results from the measurements (TinyM-M) achieve:

1) Comparable detection results to the reconstructed video (TinyG-R)

2) Close to the detection results from the ground truth video (TinyG-G) across all compression ratios.

Do not need to reconstruct the high-quality data in real-time, and we can still use CI cameras for real applications in CAVs.
Conclusion

• Vehicle-EdgeServer-Cloud *closed-loop* framework

• This is the first work that provides an alternative method to achieve fast object detection *on measurements*.

• Our code is hosted at [https://www.thecarlab.org/outcomes/software](https://www.thecarlab.org/outcomes/software)
lu.sidi@wayne.edu

https://www.thecarlab.org/