Doing More with Less: Cost-Effective Infrastructure for Automotive Vision Capabilities

PI: Prof. James Anderson[1], co-PIs: Prof. Sanjoy Baruah[1], Prof. Alexander Berg[1] & Dr. Shige Wang[2]
Students: Andrew Hassevoort[1], Eunbyung Park[1], Ming Yang[1]
[1]The University of North Carolina at Chapel Hill & [2]General Motors

Motivation

➢ Many safety-critical cyber-physical systems rely on advanced sensing capabilities to react to changing environmental conditions. However, cost-effective deployments of such capabilities have remained elusive. Such deployments will require software infrastructure that enables multiple sensor-processing streams to be multiplexed onto a common hardware platform at reasonable cost, as well as tools and methods for validating that required processing rates can be maintained.

Problem

➢ Currently, advanced driver assistance system (ADAS) capabilities have only been implemented in prototype vehicles using hardware, software, and engineering infrastructure that is very expensive. Prototype hardware commonly includes multiple high-end CPU and GPU chips and expensive LIDAR sensors.

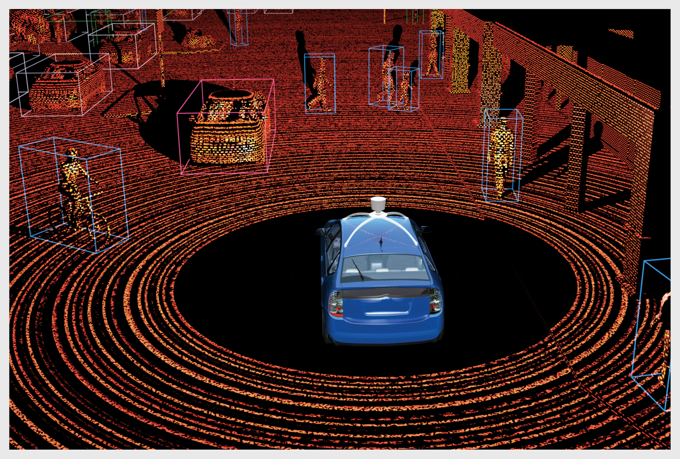


Image sources: http://www.popsci.com/cars/article/2013-09/google-self-driving-car, http://www.routescene.com/products/product/velodyne-lidar/

➢ Focusing directly on judicious resource allocation, this project seeks to enable more economically viable implementations. Such implementations can reduce system cost by utilizing cameras in combination with low-cost embedded multicore CPU+GPU platforms.

Image sources: http://www.carstuff.com.tw/car-news/item/5890-mobileye-560.html, http://www.anandtech.com/show/7905/nvidia-announces-jetson-tk1-dev-board-adds-erista-to-tegra-roadmap

Objectives

➢ This project focuses on three principal objectives:
- New implementation methods for multiplexing disparate image-processing streams on embedded multicore platforms augmented with GPUs.
- New analysis methods for certifying required stream-processing rates.
- New computer-vision methods for constructing image-processing pipelines.

Activities

➢ Automotive Cyber-Physical Systems graduate-level course at UNC Chapel Hill. (http://www.cs.unc.edu/~anderson/teach/comp790a/)

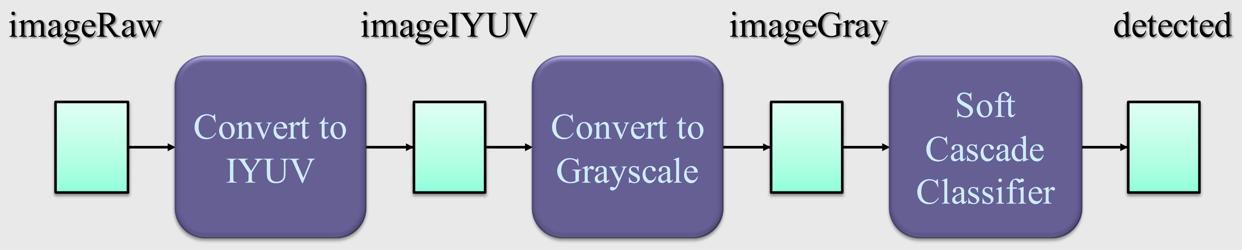
➢G. Elliott, K. Yang, and J. Anderson, "Supporting Real-Time Computer Vision Workloads using OpenVX on Multicore+GPU Platforms", *Proceedings of the 36th IEEE Real-Time Systems Symposium*, December 2015, to appear.

➢K. Yang, G. Elliott, and J. Anderson, "Analysis for Supporting Real-time Computer Vision Workloads using OpenVX on Multicore+GPU Platforms", *Proceedings of the 23rd International Conference on Real-Time Networks and Systems*, November 2015, to appear.

Supporting Real-Time Computer Vision Workloads

OpenVX

- Computer vision algorithms are commonly expressed using dataflow graphs.
- A standard computer vision API – OpenVX – has been created that allows for specification of such graphs.



- Node dependencies (i.e., edges) are derived from how data objects are bound to the inputs and outputs of the nodes.
- Each node is a basic operation in a computer vision algorithm.
- A given basic operation has a set of well-defined inputs and outputs, and may be performed on either CPU or GPU (depending on implementation).

OpenVX vs. Real-Time

- Our team developed a new OpenVX implementation that extends a current OpenVX implementation by NVIDIA.

Existing OpenVX implementation	Our extension ([1] describes details)
No notion of repeating (periodic or sporadic) task	The source node of each graph is invoked sporadically
Does not define a threading model	Each node is assigned a dedicated thread
Requires a graph to execute end-to-end before it may be re-executed	Graph execution can be pipelined

GPU accesses are managed by GPUSync [2].

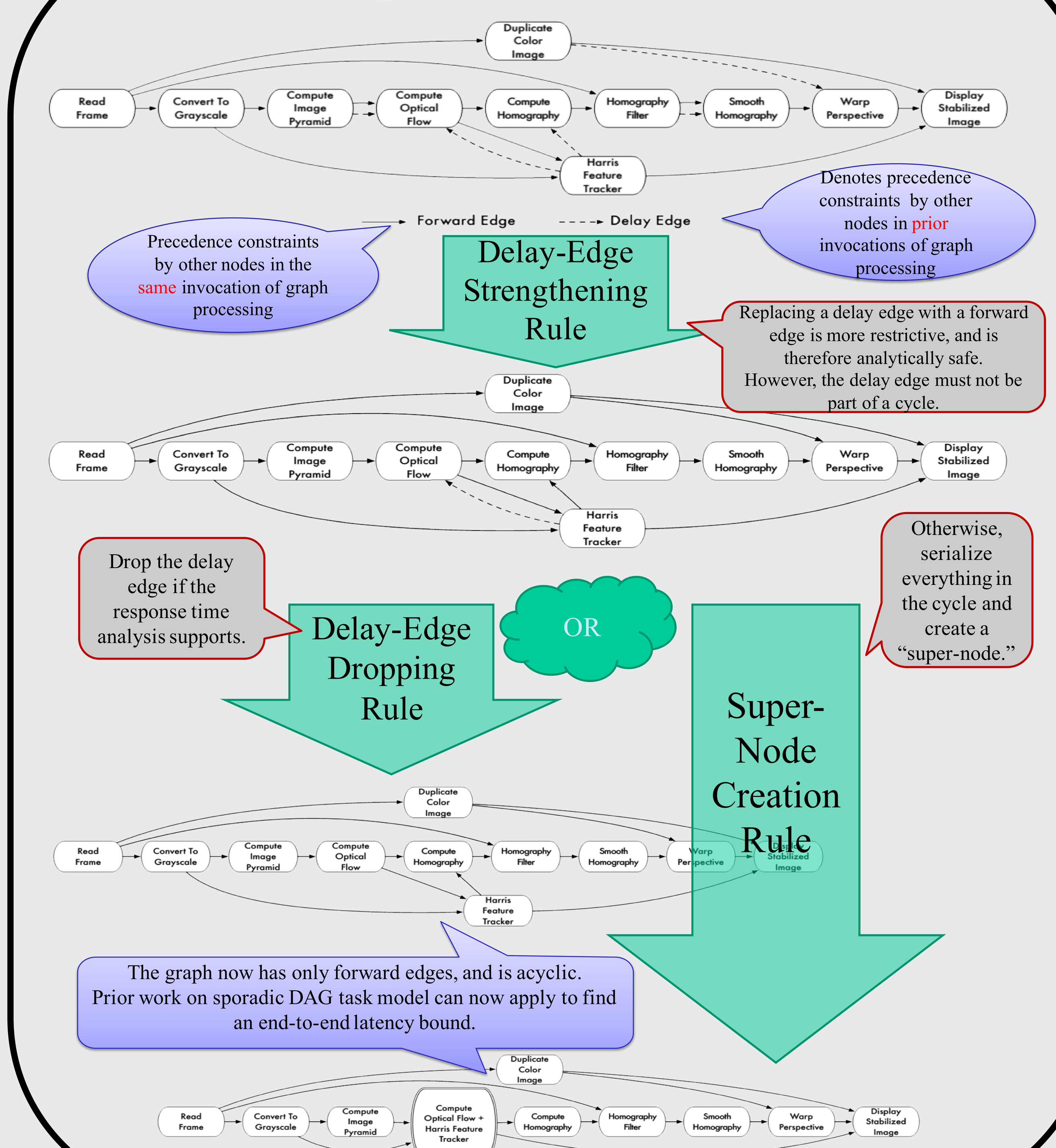
[1] G. Elliott, K. Yang, and J. Anderson, "*Supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms*," RTSS 2015.
[2] G. Elliott, "*Scheduling of GPUs, with applications in advanced automotive systems*," Ph.D. dissertation, The University of North Carolina at Chapel Hill, 2015.

Analysis

- Main Idea: transform OpenVX graphs to directed acyclic graphs (DAGs), then apply prior work on the sporadic DAG model.
- Challenges:
  - Delay edges are not the same as edges in DAGs.
  - Delay edges may cause cycles that are not allowed in DAGs.
- The following graphs use a video stabilization application as an example to illustrate the transformation techniques.
- Detailed analysis can be found in [3], including an analysis of the size of buffers required to support pipelined processing.

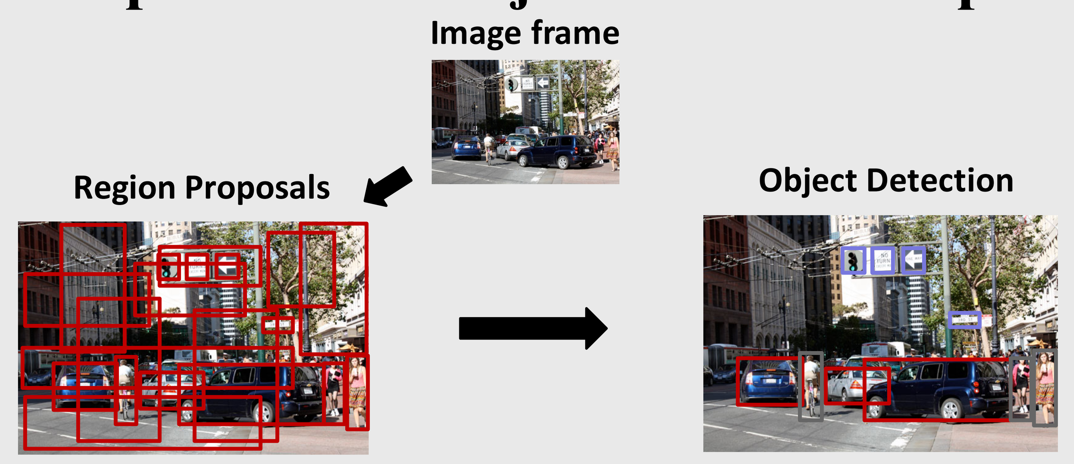
[3] K. Yang, G. Elliott, and J. Anderson, "Analysis for *supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms*," RTNS 2015.

Graph Transformation



Case Study of an Object Detection Pipeline for ADAS Applications

Deep Network Object Detection Pipeline



- Region based object detection based on deep convolutional neural network.
  - State-of-the-art performance.
  - Less computational complexity compared to sliding-window approaches.

Accuracy vs. Time

- Exploring factors in designing deep networks for tradeoffs between accuracy and computation time.
  - Depth, width, number of filters, filter size, etc.
  - Number of region hypotheses evaluated vs. accuracy.

Deep Network Object Detection Pipeline