Building the Computing System for Autonomous Micromobility Vehicles: Design Constraints and Architectural Optimizations

Bo Yu *et al.*, PerceptIn

Presented by Robin Dehler

Introduction

- First thorough study of a commercial computing system for micromobility

- Objectives:
  - Highlight design constraints unique to AD
  - Identify new architecture and system problems for autonomous machines

- Contributions:
  - introduce real autonomous vehicle workloads
  - present detailed performance characterization of the vehicles
  - highlight that the computing system shouldn’t be optimized alone
  - FPGA is an effective way of acceleration

Micromobility as a service

https://cdn.ibj.com/wp-content/uploads/2019/11/Perceptin_LESV_1200px-1024x791.jpg

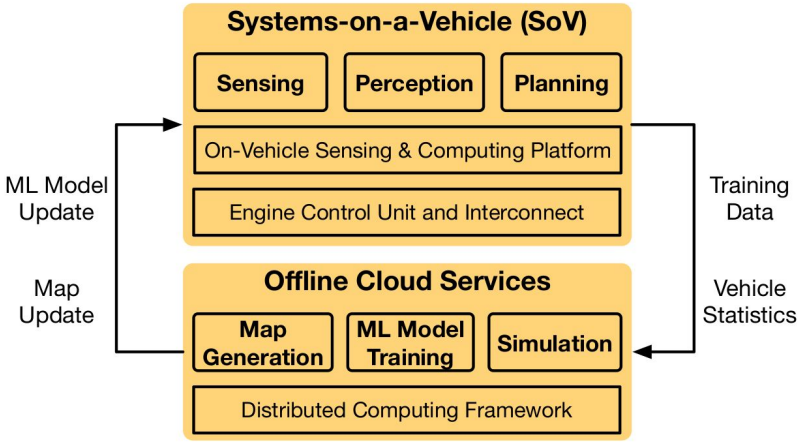
https://eenews.cdnartwhere.eu/sites/default/files/styles/inner_article/public/sites/default/files/images/2018-09-25-s20_perceptin_autonomous_vehicle_dragonfly_pod.jpg?itok=kmEQp9Bc

- Micromobility market will grow up to 500 billion $ by 2030
- Two autonomous vehicle designs
- Vehicles are in autonomy level 4
- On-vehicle online tasks and offline tasks hosted as cloud services

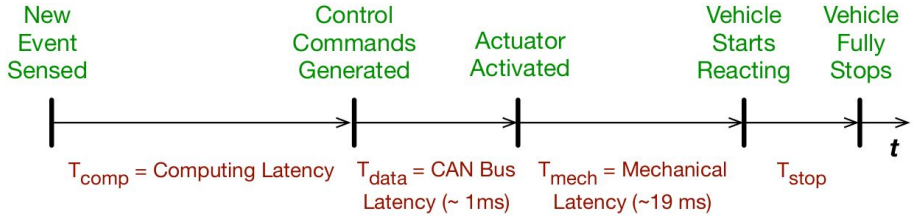
Autonomous Driving Infrastructure

- On-vehicle processing (SoV)
  - Sensing, perception, planning
  - Varieties of sensors

- (Offline) cloud services
  - Map generation, simulation and ML model training
  - Continuous data exchange
  - Limitation of communication bandwidth



Design Constraints



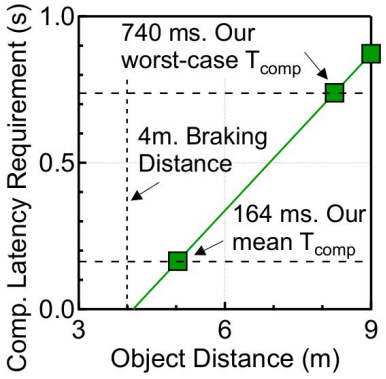
- Latency and throughput
  - For safety:

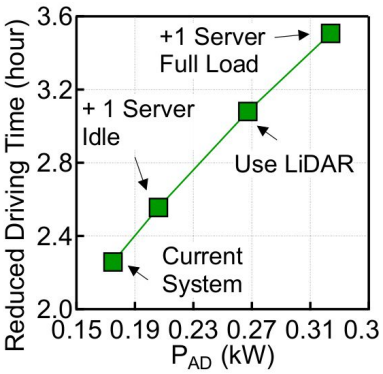
$$(T_{comp} + T_{data} + T_{mech}) \cdot v + \frac{1}{2} \cdot a \cdot T_{stop}^2 \leq D$$

  - Throughput requirement 10 Hz
- Energy and Thermal
  - No thermal constraints
  - Impact of energy consumption on driving time:

$$T_{reduced} = \frac{E}{P_V} - \frac{E}{P_V + P_{AD}}$$

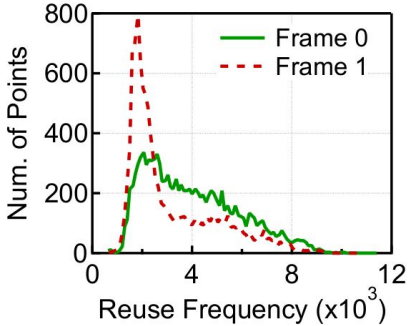
- Cost and Safety considerations
  - Cost of AV is complex function



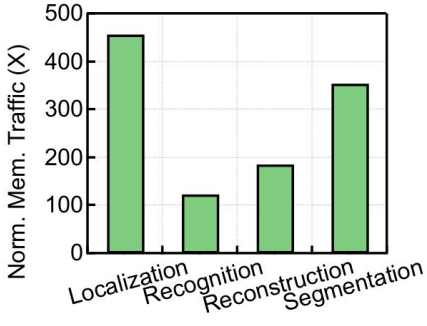


Case Study: LiDAR vs. Camera

- Vehicles without LiDAR
- Latency:
  - 100ms - 1s vs. 25 ms for vision algorithms
- Power:
  - One order of magnitude more power hungry
- Cost:
  - One order of magnitude more expensive than cameras
- Depth quality
  - LiDAR directly provides depth information with 2 cm precision

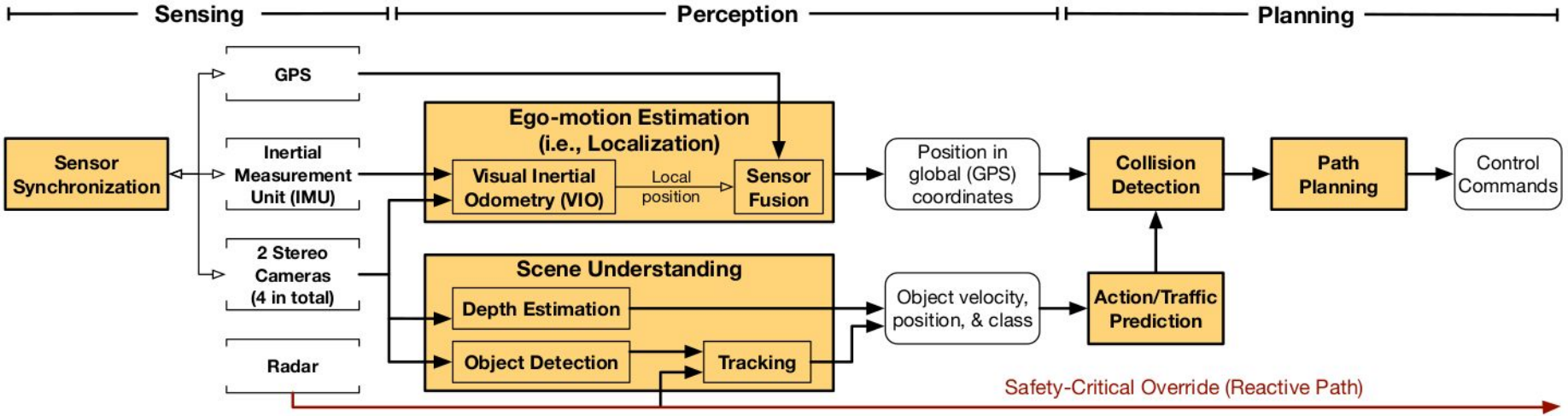


(a) Irregular data reuse.



(b) High memory traffic.

Software pipelining



- On-vehicle processing system consists of sensing, perception and planning
- Control commands are transmitted through CAN bus to ECU

Perception and Planning Algorithms

- Depth estimation
  - ELAS for fast matching of high-resolution images
- Object detection
  - Increased accuracy of DNN models is worth the overhead
  - Object is then tracked
- Object tracking
  - Mostly done by Radar
- Localization
  - Uses images and IMU to estimate position
- Planning
  - MPC

Task	Algorithm(s)	Sensors(s)
Depth Estimation	ELAS [44]	Cameras
Object Detection	YOLO [48]/ Mask R-CNN [49]	Camera
Object Tracking	KCF [46]	Camera, Radar
Localization	VIO [41]	Cameras, IMU, GPS
Planning	MPC [47]	–

Safety

- Proactive path given by processing pipeline
  - Problems with latency and wrong vision algorithms results
- Last line of defense
  - Reactive path can override control commands
  - Uses Radar (& Sonar if available)
  - Latency of 30 ms -> approaches 4 m safety distance

Task-level parallelism

- Sensing, perception and planning are serialized
- Pipelining of the 3 modules improves throughput

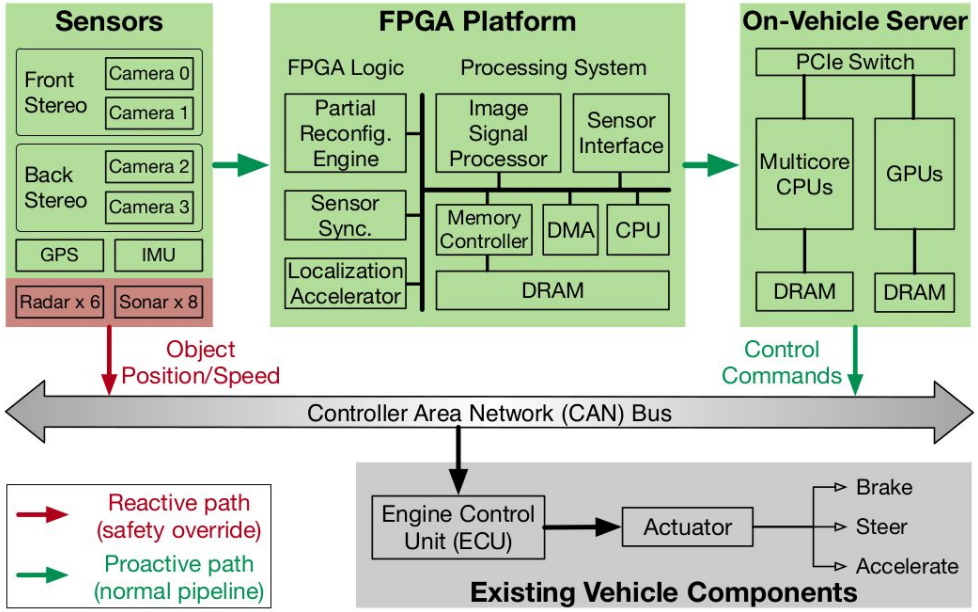
Systems-on-a-Vehicle (SoV)

- On-vehicle processing system
- Combination of
  - off-the-shelf mobile SoCs
  - off-the-shelf (ASIC-based) chips
- Mobile SoCs:
  - Qualcomm Snapdragon or Nvidia Tegra
  - Low compute capability, no data communication between different computing units, no sensor synchronization possible
- Automotive ASICs:
  - Specialized for AD
  - Provide high-performance at a much higher cost
  - Only focus on accelerating a subset of task in AD and insufficient sensor synchronization support

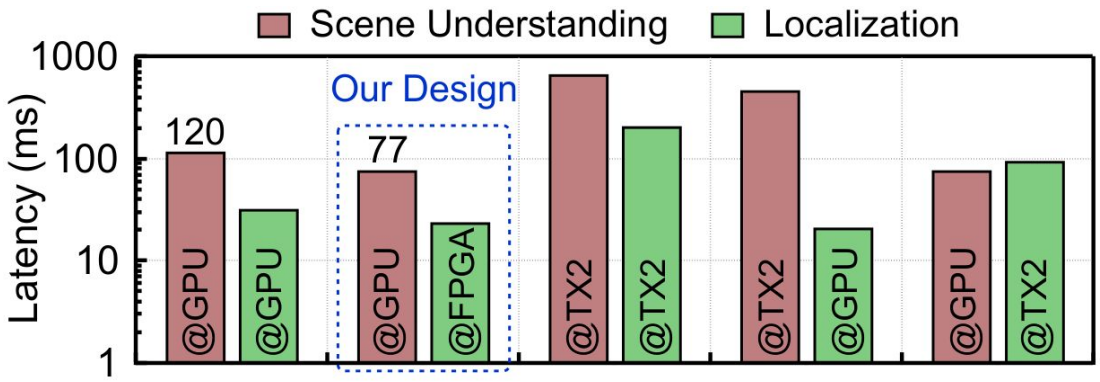
Systems-on-a-Vehicle (SoV)

- Current SoV design:
  - on-vehicle server machine
  - FPGA platform
  - software-hardware collaborative technique

- Xilinx Zynq UltraScale+ FPGA
- On-vehicle PC with Intel Coffee Lake CPU and Nvidia GTX 1060 GPU



Algorithm-Hardware Mapping



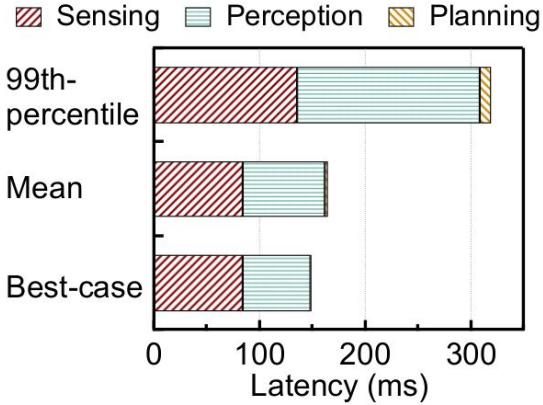
- Sensing is mapped to FPGA
- Planning task is assigned to CPU
- Localization is done on FPGA, other perception tasks on GPU (e.g. scene understanding)
  - Latency dictates user-experience and safety

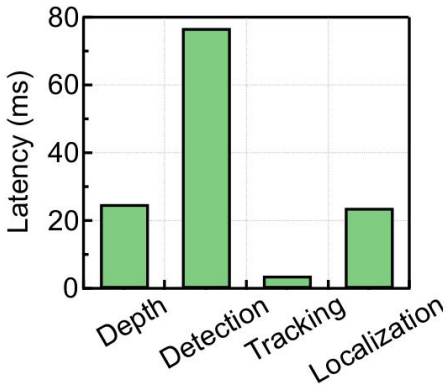
Partial reconfiguration

- RPR exploits FPGA's reconfigurability
- Time-sharing
- Provide hardware support on the FPGA
- Design that decouples receiving bitstream data
- Transmitter als DMA engine, receiver transfers data from the FIFO to the FPGA

Performance Characterizations

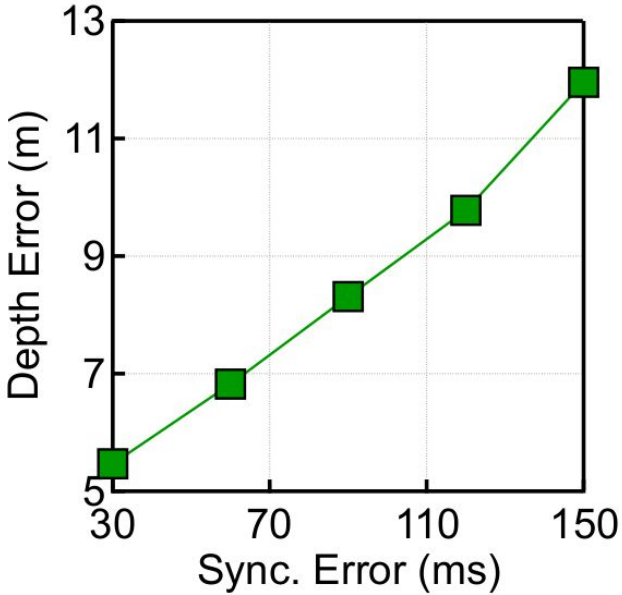
- Mean latency close to best-case latency, but long tail
- Vehicles stay in proactive paths for over 90 %
- Latency distribution
  - Sensing latency bottlenecked by camera sensing pipeline
  - Perception latency also big contribution
  - Detection and tracking are serialized
- Sensing, perception and planning modules operate at 10Hz - 30Hz





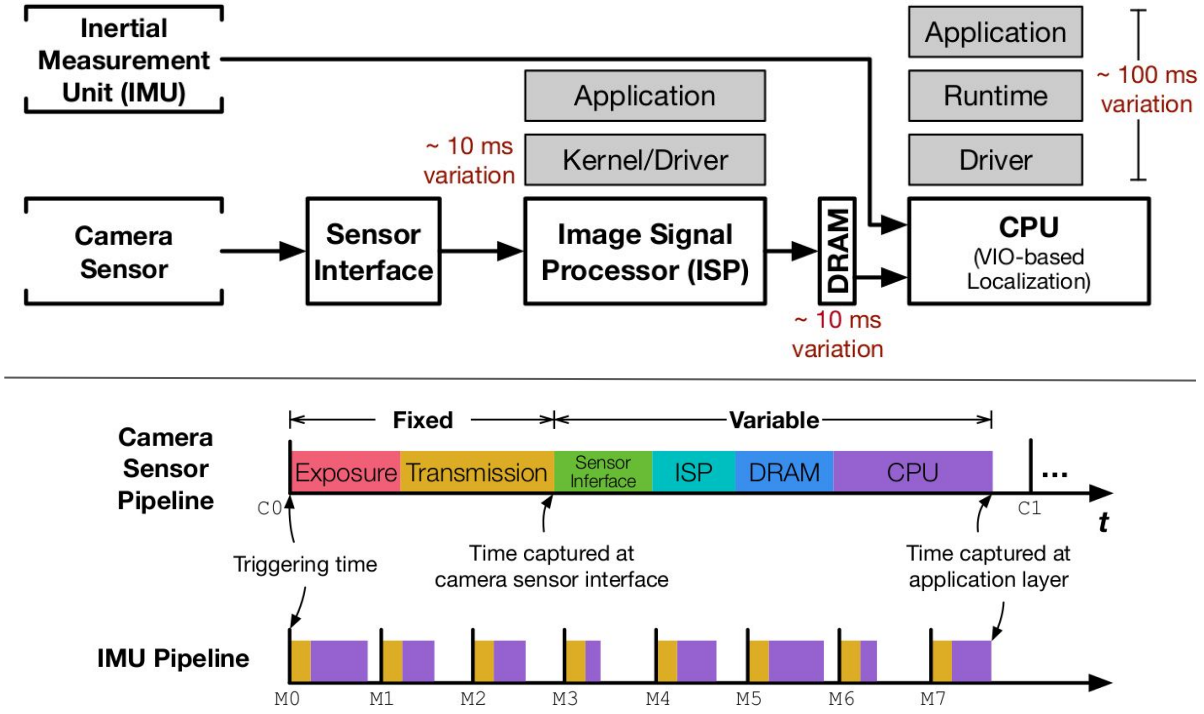
Sensor synchronization

- Sensing has been overlooked
  - Most research work focus on perception and planning
- Sensing is crucial for perception
  - e.g. in sensor fusion
- Simultaneous triggering and precise timestamps
- Sensor synchronization at application layer
- Inaccurate software-only synchronization



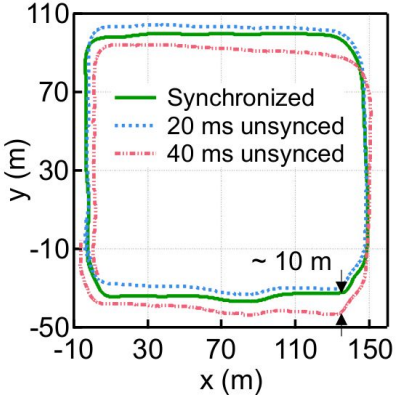
Sensing-Computing Co-Design

- Localization
  - Sync sensor samples from both the camera and the IMU
  - Processing pipeline introduces variable latency
- Camera-IMU sync
  - IMU with short processing time
  - Camera with long processing time

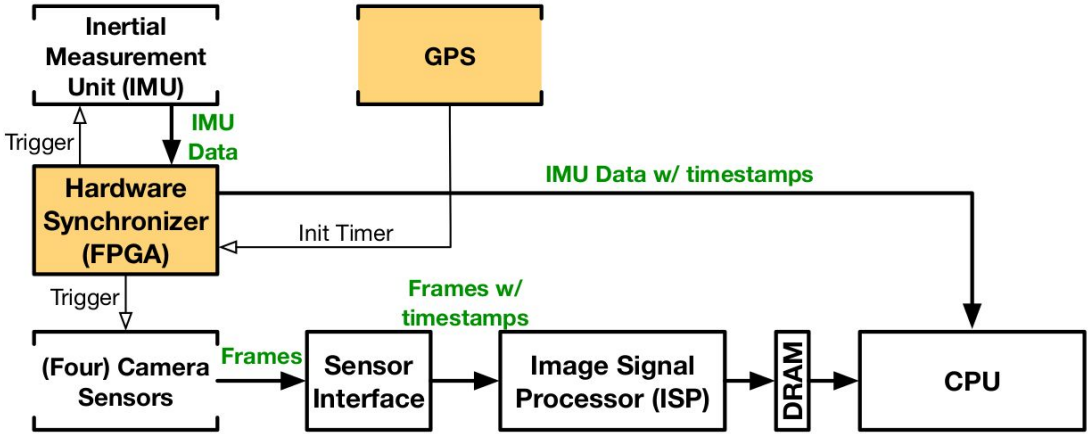


Software-hardware collaborative sensor synchronization

- based on two principles: near-sensor synchronization
- Hardware synchronizer triggers camera sensors and IMU
- Sends camera frames directly to SOC's sensor interface without timestamps from hardware synchronizer



- Design
  - GPS provides satellite atomic time as universal time source
  - Camera 30 FPS IMU 240 FPS
  - Camera trigger signal is down-sampled 8 times from IMU signal



Performance and Cost

- Localization results indistinguishable from ground-truth
- Lightweight design
  - Consumes 5 mWs
  - Less than 1 ms delay
- Can easily be extended
- First sensor synchronization technology in public domain that can flexibly adapt to different sensors

Augmenting Computing with sensors

- Perception system driven by vision algorithms
- Additional sensors could reduce computing cost
- Localization
  - VIO as localization algorithm has cumulative errors
  - GPS-VIO hybrid approach: Combine GNSS signal with planning module, fusing by EKF
- Tracking
  - Tracking algorithms replaced by Radar sensors
  - Radar robust to weather conditions
  - Spatial synchronization needed

Concluding remarks

- AD is very complex with many different tasks across computing and sensor domains
- Systematic understanding of end-to-end system matters the most
- SoV of AV includes many components
- Presentation of generic latency, throughput, power/energy and cost models of AVs and demonstration of concrete examples
- Discussion of opportunities and feasibility of time-sharing the FPGA resources
- Paper presents key aspect of AV’s cost