Real-Time Computer Vision in Autonomous Vehicles: Real Fast Isn't Good Enough

University of North Carolina Chapel Hill
PI: Prof. James Anderson, co-PIs: Prof. Jan-Michael Frahm, Dr. Frank D. Smith, & Dr. Shige Wang
Students: Tanya Amert, Nathan Otterness, Thanh Vu, Sergey Voronov
Award ID: CPS 1837337

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL · GM · UNC REAL-TIME SYSTEMS GROUP · UNC COMPUTER VISION GROUP

Motivation

A significant gap exists between how computer-vision software and embedded safety-critical software define the term real time. In computer vision, "real time" often means "real fast" and refers to high average throughput with low latency. In contrast, safety-critical embedded systems consider "real time" to be a statement about predictable timing under continuous performance.

Put in a larger context, this disparity in definitions is merely a symptom of the fact that, historically, computer vision and embedded systems have evolved independently. With the advent of autonomous vehicles, where safety depends on both computer vision and predictable timing, a clear separation between vision and embedded software is no longer possible.

Problem

In the near future, autonomous vehicles will face the need for real-time certification, which will cause the "real-time" vs. "real-fast" disconnect to become more problematic as time passes.

Certifying applications that are merely "real-fast" will be almost impossible, but, as of now, there is no simple way to map traditional computer-vision applications into the frameworks needed to ensure predictable timing required by safety-critical systems.

Our project seeks to address this problem by developing a framework to bridge this gap between "real-fast" and "real-time" software.

Objectives

This project focuses on four principal objectives:

- Develop a real-time-aware computer-vision API.
- Develop real-time schedulability analysis targeting our new API.
- Develop real-time computer-vision algorithms that exploit our API's new features.
- Experimentally compare "real-fast" and "real-time" computer vision.

Activities

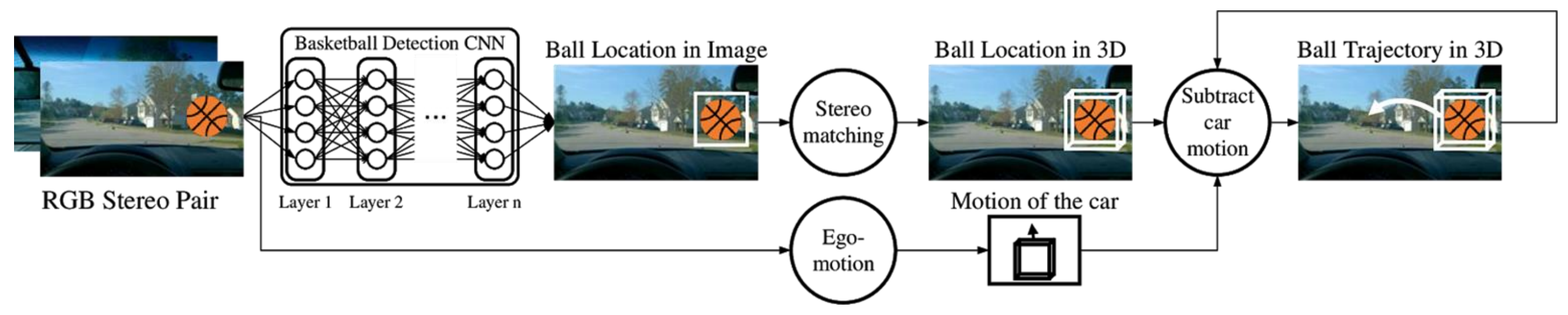
- Summer internship project in General Motors Research & Development, 2018
- Summer internship project in General Motors Research & Development, 2019
- Demonstrated autonomous driving simulation to middle school students, Spring 2019.
- Summer internship project in General Motors Research & Development, 2020.

Developing a Framework for Real-Time Computer-Vision Workloads

Real-Time Computer-Vision Graph Scheduling

Real-time scheduling of computer-vision applications presents several challenges:

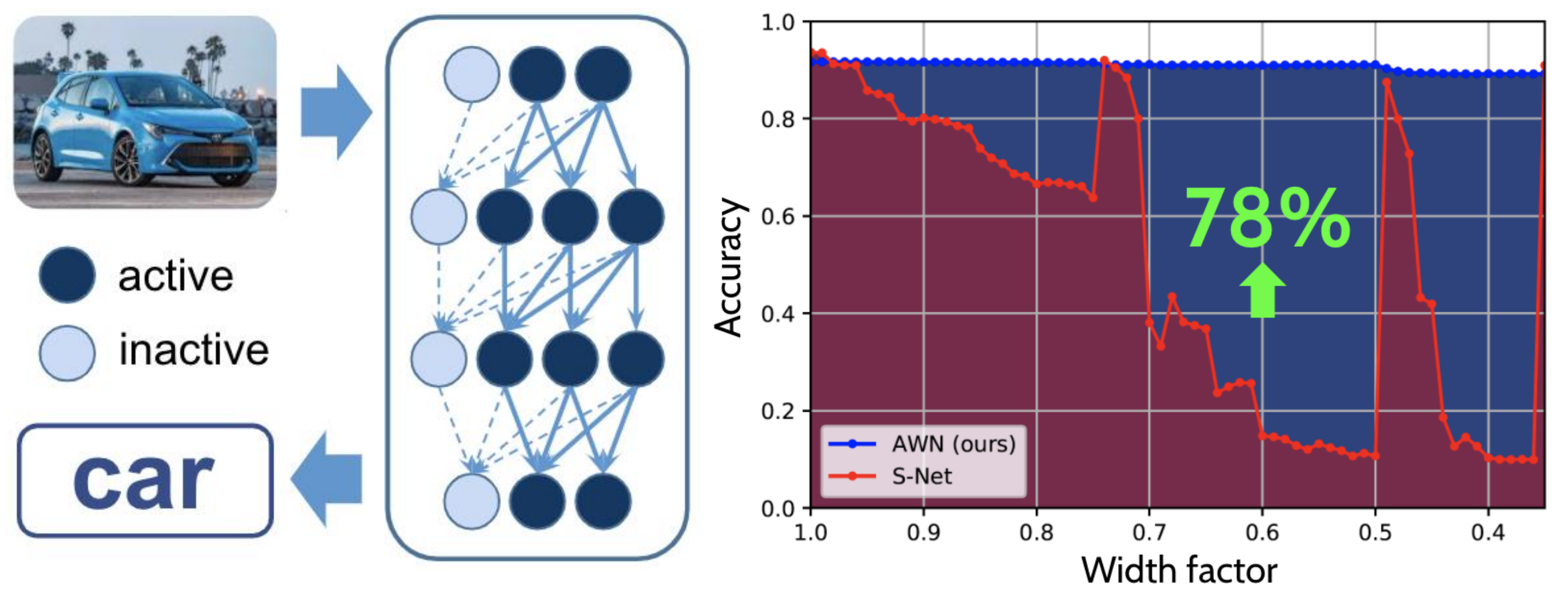
- Application graphs may include *cyclic dependencies*, such as using previous results in subsequent calculations.
- Computer vision often requires specialized computing hardware, *i.e.* GPUs, which have unknown timing and hardware-interference properties.
- Workloads may need to change dynamically, based on environmental conditions or fault-tolerance requirements.



Inference-Adjustable CNNs

Most CNNs operate as monolithic entities, posing a challenge for autonomous-driving applications with situationally varying budgets. We proposed an adjustable CNN architecture that allows fine-grained control over the speed-accuracy trade-off depending on the available resources during inference.

By leveraging statistical observations, we can provide flexible and maximally granular control along with state-of-the-art accuracy without additional post-training calibration.

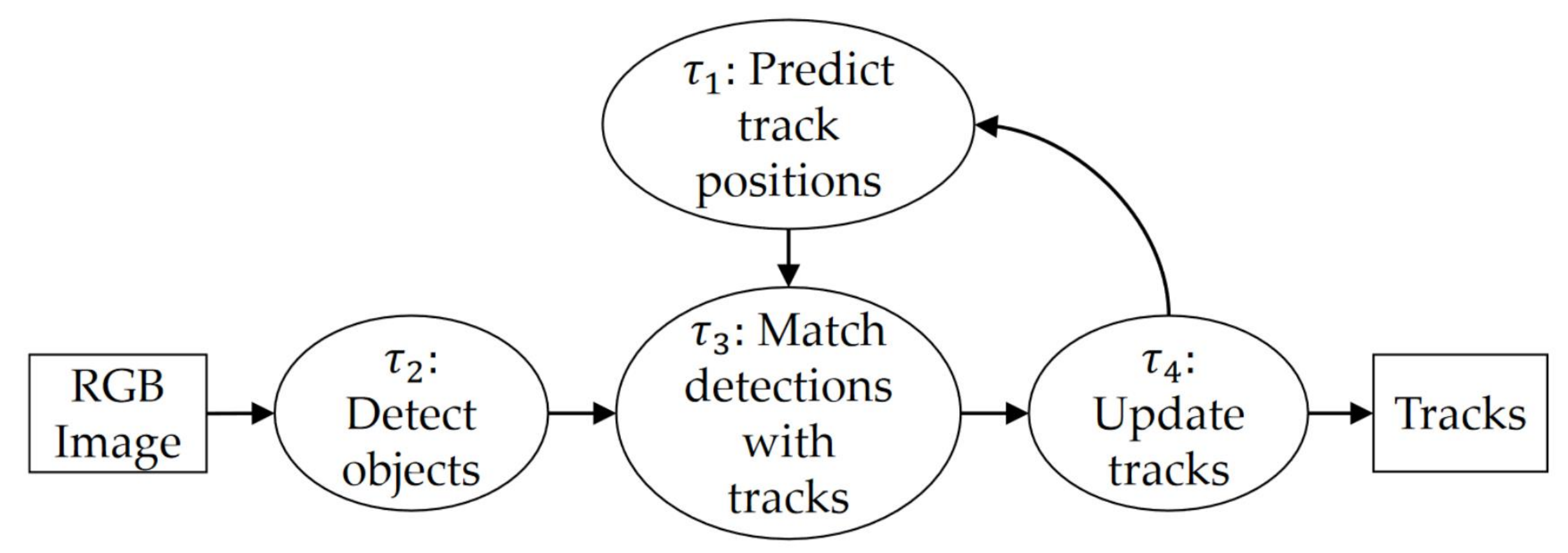


Selected Publications (out of 22 total)

- M. Yang, S. Wang, J. Bakita, T. Vu, F.D. Smith, J. Anderson, and J.-M. Frahm, *Re-thinking CNN Frameworks for Time-Sensitive Autonomous-Driving Applications: Addressing an Industrial Challenge*, in RTAS '19.
- T. Amert, S. Voronov, and J. Anderson, *OpenVX and Real-Time Certification: The Troublesome History*, in RTSS '19.
- T. Vu, M. Eder, T. Price, J.-M. Frahm, *Any-Width Networks*, in CVPR Workshops '20.
- N. Otterness, J. Anderson, *AMD GPUs as an Alternative to NVIDIA for Supporting Real-Time Workloads*, in ECRTS '20.
- T. Amert, M. Yang, S. Nandi, T. Vu, J. Anderson, and F.D. Smith, *The Price of Schedulability in Multi-Object Tracking: The History-vs.-Accuracy Trade-Off*, in ISORC '20.
- T. Amert and J. Anderson, *CUPiD^RT: Detecting Improper GPU Usage in Real-Time Applications*, in ISORC '21.
- N. Otterness, J. Anderson, *Exploring AMD GPU Scheduling Details by Experimenting With "Worst Practices"*, in RTNS '21.

Real-Time Task Model for Graphs with Cycles

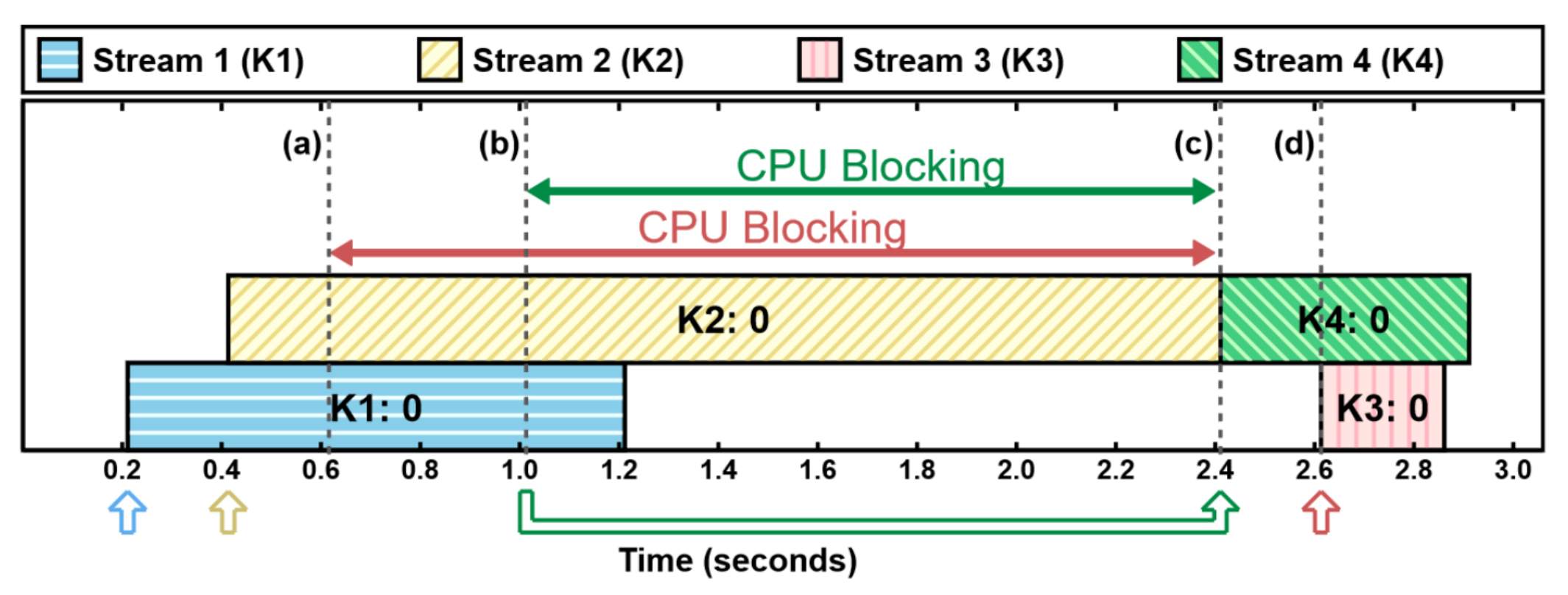
Computer-vision algorithms to track that movement of pedestrians and other vehicles are usually expressed as real-time processing graphs. These graphs contain cycles due to back edges that provide history information. Using older back history enables parallelism in cycle execution at the expense of possibly affecting tracking accuracy.


RGB Image
τ_1: Predict track positions
τ_2: Detect objects
τ_3: Match detections with tracks
τ_4: Update tracks
Tracks

- We explored the trade-off between response times, intra-task parallelism, and accuracy for Multi-Object Tracking.
- We found that allowing non-immediate back history had only a marginal impact on accuracy.

CUPiD^RT: Detecting Improper GPU Usage

Computer-vision algorithms typically rely on GPUs to accelerate computations. However, improper GPU usage can lead to unexpected delays on the GPU *and on the host CPU*.



We developed a software library to detect the improper use of GPUs for safety-critical computer-vision applications.

- We found that all ten GPU-using OpenCV sample applications we considered had issues flagged by our tool.
- Fixing detected issues resulted in shorter and more predictable execution times and GPU kernel launch times.

Investigating Open-Source GPU-Compute Software

GPUs, or similar accelerators, are fundamental to timely operation of sophisticated neural-network operation in safety-critical systems. NVIDIA's CUDA framework for GPU programming remains massively popular in this space, despite being closed-source: unamenable to auditing or modification.

GPUs manufactured by AMD instead use *ROCm*, an open-source competitor to CUDA. Unfortunately, many behavioral and timing characteristics differ between CUDA and ROCm.

- We used a combination of experiments and sparse public documentation to publish a unified picture of how AMD GPUs and ROCm schedule compute workloads.
- We measured the efficacy of ROCm's built-in support for *hardware partitioning*, including effective partitioning strategies.