OpenCV hardware acceleration with vAccel

Maria Goutha, Ilias Lagomatis*+, Anastassios Nanos*, Alexandros Patras+*

*NUBIS PC, +University of Thessaly

{mgouth,ilago,ananos}@nubis-pc.eu, patras@uth.gr

Eurosys 2024, Athens, Greece

April 22nd-25th 2024

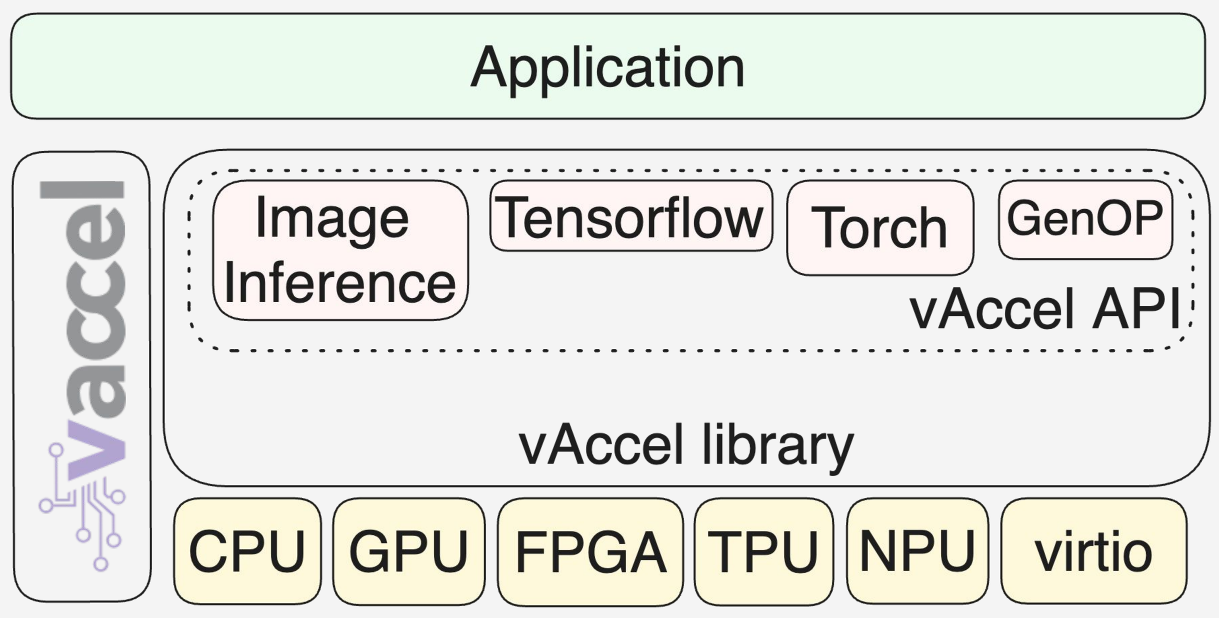
Motivation

Deployments in multi-tenant Cloud/Edge infra suffer from poor isolation ➜ sandbox user code in microVMs

- ➜ Limited support for accelerator drivers
  - ➜ Hardware partitioning – Porting of a device driver
  - ➜ Paravirtualization – Porting of a virtual device driver
  - ➜ Remote API – Porting of the framework
- ➜ Limited support for acceleration frameworks
  - ➜ Huge code base
  - ➜ Diverse dependencies
  - ➜ Bound to a language

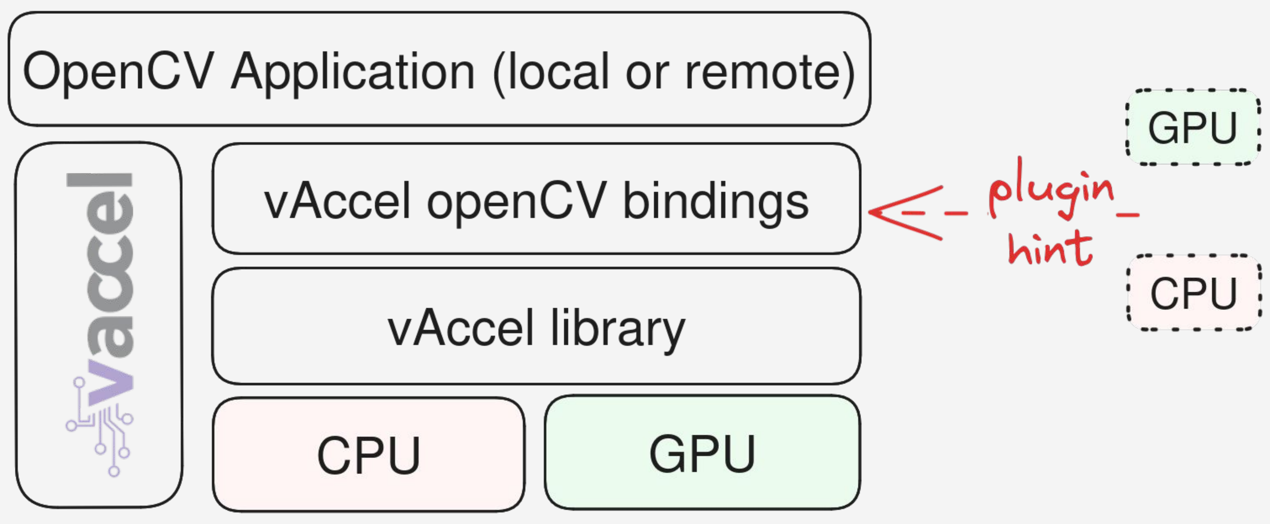
vAccel

- ➜ vAccel decouples the function call from its hardware-specific implementation
- ➜ Features:
  - ➜ **Hardware-agnostic** API
  - ➜ Acceleration in **function** granularity
  - ➜ **Portability** and **interoperability**



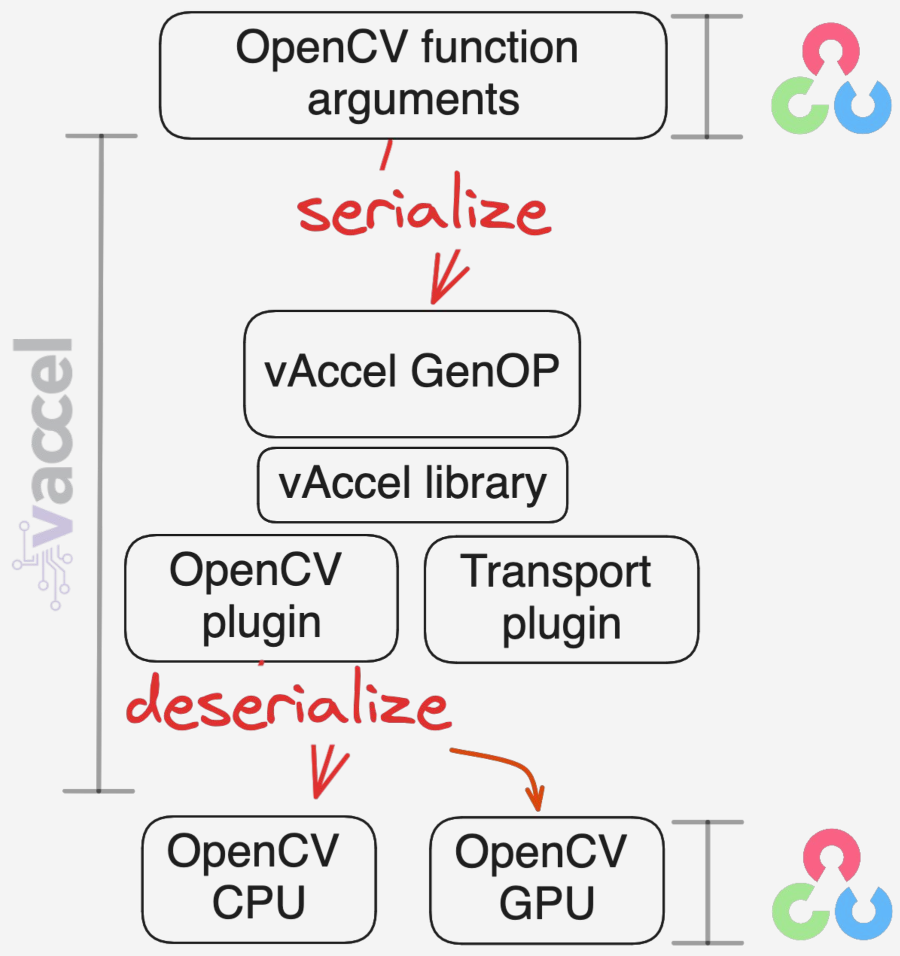
Plugin change at Runtime

- ➜ Introduce a mechanism for vAccel to change the backing plugin at runtime
  - ➜ Requirements Bitmap
  - ➜ Plugin features Bitmap



OpenCV Bindings

- ➜ Agnostic to the user
  - ➜ Overload original OpenCV function (e.g. `calcOpticalFlowPyrLK()`)
  - ➜ Parse arguments & **serialize** them
  - ➜ Issue the vAccel OpenCV operation (equivalent to vAccel `GenOP`)
  - ➜ Go through the relevant plugin (virtio / Transport or local)
  - ➜ In the plugin, **deserialize** the arguments
  - ➜ Call the respective OpenCV operation

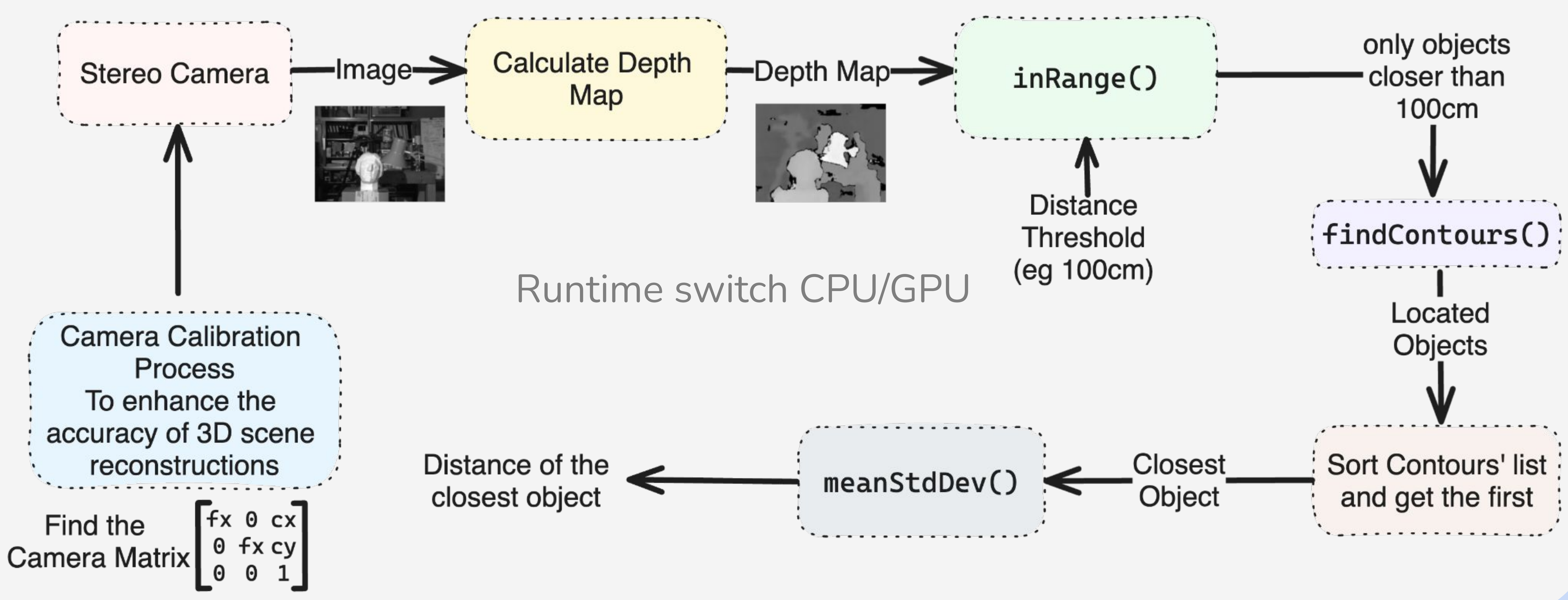


Obstacle Avoidance

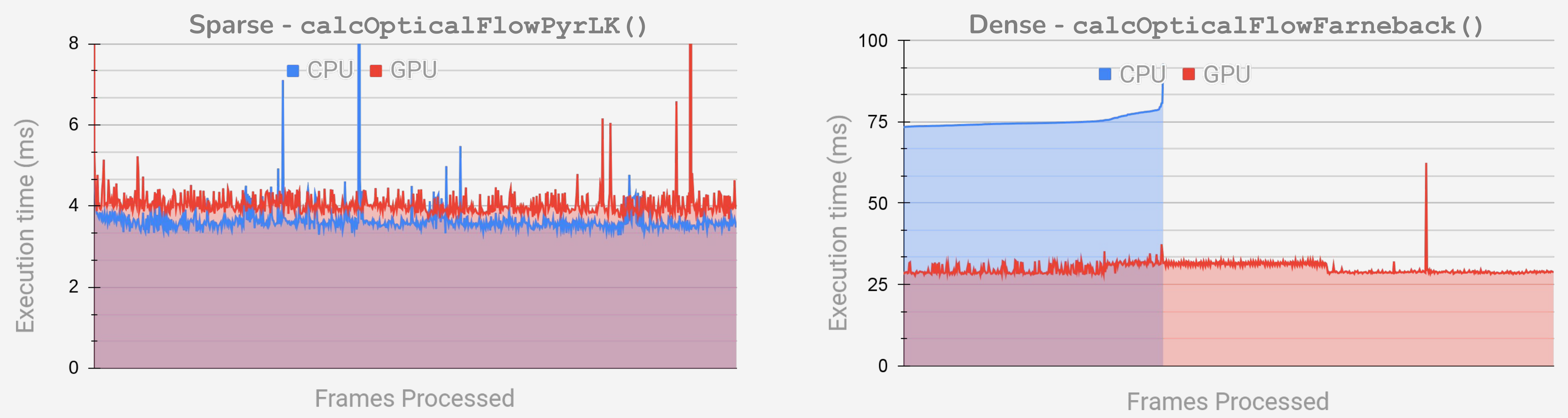
- ➜ Stereo Calibration and Rectification
- ➜ Disparity Map and Depth Map
- ➜ Obstacle Avoidance

Criteria:

- ➜ response latency
- ➜ energy consumption
- ➜ execution time



Initial Evaluation - Optical Flow (sparse / dense)



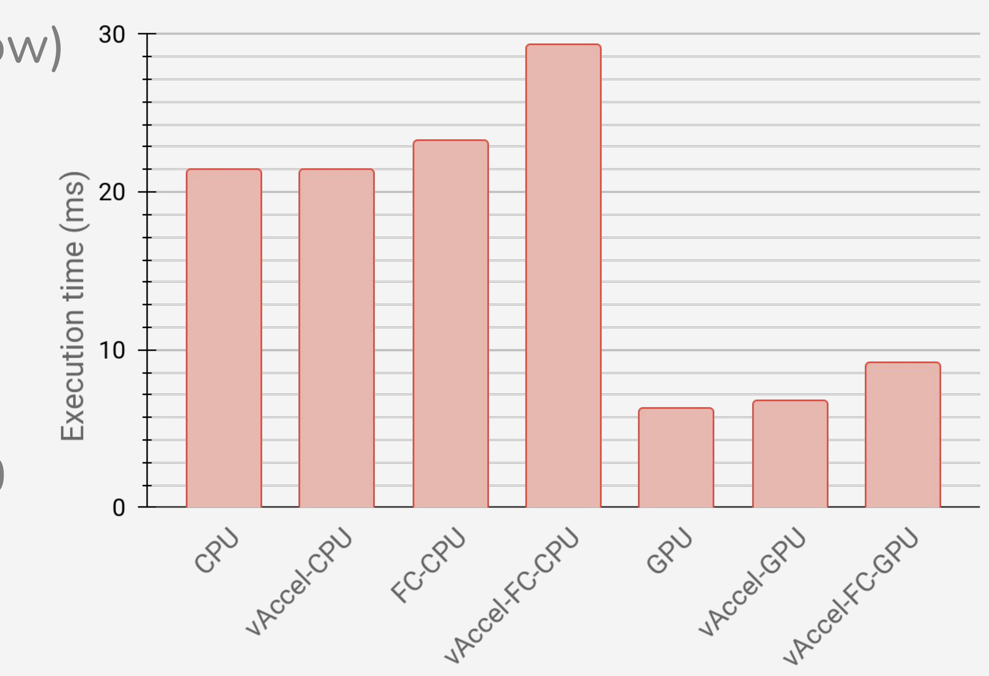
- ➜ 913-Frame video (OpenCV example)
- ➜ Hardware: Jetson AGX Orin
- ➜ generic & sandboxed containers
  - ○ runc
  - ○ AWS firecracker (kata-containers)
- ➜ Sparse behaves marginally better on the CPU
- ➜ Dense cannot cope with the frame rate on the CPU, leading to frame drops.
- ➜ The plugin change is done at runtime

Initial Evaluation - native vs vAccel

Naive example (OptFlow)

Benchmark Spec:

- ➜ Jetson AGX Orin
- ➜ local/VM execution
- ➜ AWS Firecracker
- ➜ CPU/GPU
- ➜ two frames for 1000 iterations (avg)



Challenges & Plan

- ➜ Address argument serialization / deserialization
  - ○ push logic to the transport layer (?)
- ➜ Address copy overheads (serde & transport)
- ➜ Simplify build process (CUDA/GPU support)
- ➜ Provide end-to-end function execution for the Obstacle Avoidance example
- ➜ Finetune CPU/GPU execution of OpenCV functions to optimize:
  - ➜ power
  - ➜ performance
  - ➜ execution time

[1] vAccel: https://docs.vaccel.org
[2] vAccel OpenCV bindings: https://github.com/nubificus/opencv-vaccel
[3] Batuhan Hangün, Önder Eyecioğlu, Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations, 2019, https://doi.org/10.48550/arXiv.1906.08819
[4] Jung Hyeonseok, Kyoseung Koo, and Hoeseok Yang. "Measurement-Based Power Optimization Technique for OpenCV on Heterogeneous Multicore Processor" 2019 Symmetry 11, no. 12: 1488. https://doi.org/10.3390/sym11121488
[5] Yuan, J., Jiang, T., He, X. et al. Dynamic obstacle detection method based on U–V disparity and residual optical flow for autonomous driving. Sci Rep 13, 7630 (2023). https://doi.org/10.1038/s41598-023-34777-6
[6] Alexandros Patras, Foivos Pournaropoulos, Nikolaos Bellas, Christos D Antonopoulos, Spyros Lalis, Maria Goutha, and Anastassios Nanos. 2024. A Minimal Testbed for Experimenting with Flexible Resource and Application Management in Heterogeneous Edge-Cloud Systems. In Proceedings of the 2023 International Conference on Embedded Wireless Systems and Nnetworks (EWSN '23). Association for Computing Machinery, New York, NY, USA, 327–332.

Acknoweldgements

The research leading to these results has received funding from the European Commision through Horizon Europe under grant agreement no 101092912 (MLSysOps).