Application of heterogeneous computing techniques for image-based hot spot detection

V. Costa, S. Esquembri, J. Nieto, A. de Gracia, A. Carpeño, M. Astrain, M. Ruiz

s.esquembri@upm.es

Instrumentation and Applied Acoustic Research Group,

Universidad Politécnica de Madrid (UPM), Madrid, Spain

POLITÉCNICA

GRUPO DE INVESTIGACIÓN EN INSTRUMENTACIÓN Y ACÚSTICA APLICADA

Index

- Context
- Objectives
- Proposed architecture
- Heterogenous computing (OpenCL)
- Connected Component Labeling
- Tests
- Results
- Conclusion
- Future work

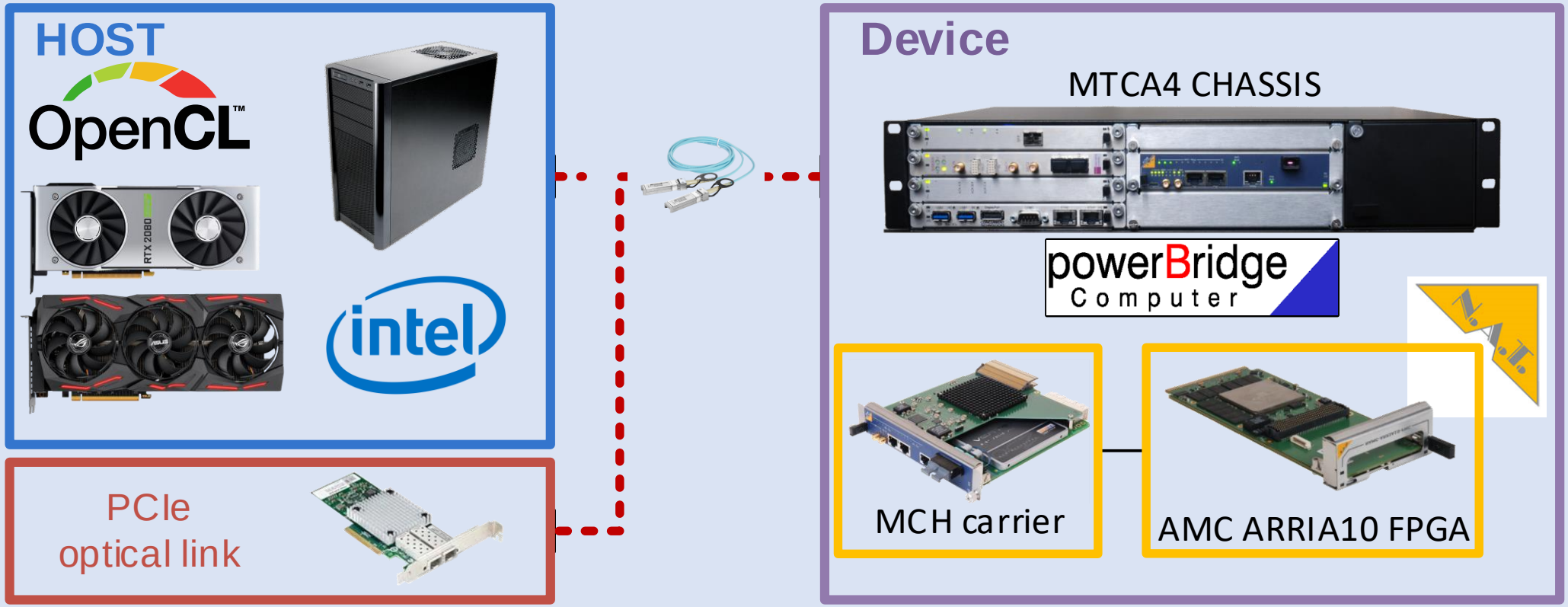
Context

- Computer vision is used in numerous experiments.
- Current frame grabber solutions are capable of high throughput. (1/10 GigE Vision, Cameralink)
- Hot spot detection algorithm is commonly used for machine protection.
- Computer vision algorithms can be developed in multiple platforms and devices.
- OpenCL offers a standard which make easier to develop heterogenous computing systems.

Objectives

- Create heterogenous computing architecture capable of processing parts of a hot spot detection algorithm.
- Modularize the CCL algorithm
- Propose execution flows using different platforms and devices with minimal code modifications.
- Analize performance using synthetic image testbench.
- Compare the results.

Architecture



Architecture: Host

- CPU
  - Intel Core i7 Coffee Lake 3.60 GHz
- RAM
  - 16 GB DDR3
- Expansion slots
  - GPU
    - AMD RX 5600
    - NVIDIA RTX 2080 SUPER
  - Optical link PCIe 3.0 x16

Architecture: GPU

AMD Radeon RX 5700

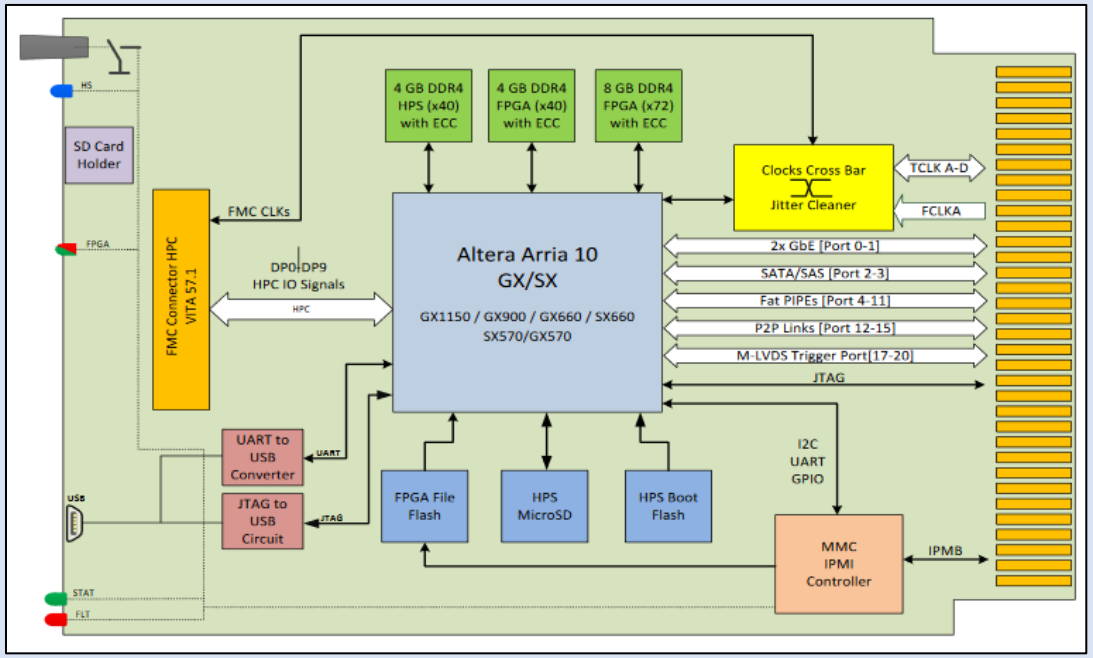
- Driver version: 20.20
- Memory: 8 GB
- Compute Units: 18
- Base Clock: 1610 MHz
- Boost Clock: 1750 MHz
- Memory Clock: 1750 MHz (14 Gbps)
- Geekbench 5 Result (compute): 51111

NVIDIA RTX 2080 Super

- Driver version: 450.57
- Memory: 8 GB
- Compute Units: 48
- Base Clock: 1650 MHz
- Boost Clock: 1815 MHz
- Memory Clock: 1937 MHz (15.5 Gbps)
- Geekbench 5 Result (compute): 119038

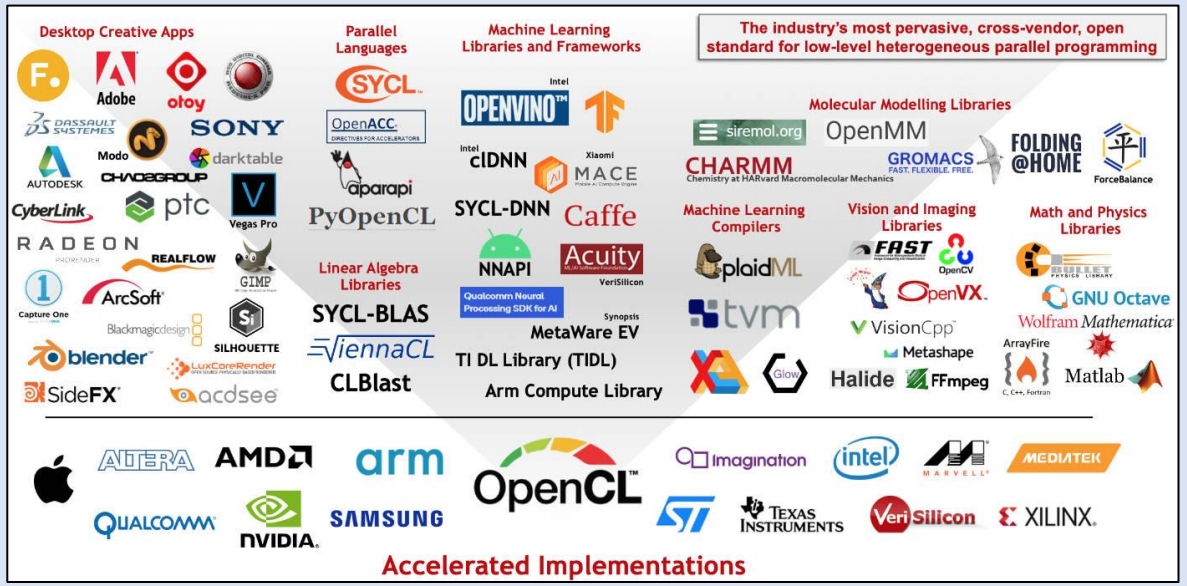
Architecture: FPGA

- NAMC-ARRIA10-FM
  - SoC ARM + FPGA Intel Altera Arria 10
  - 3 DDR4 memory interfaces
  - PCIe 3.0 (4 Gbps transfer)



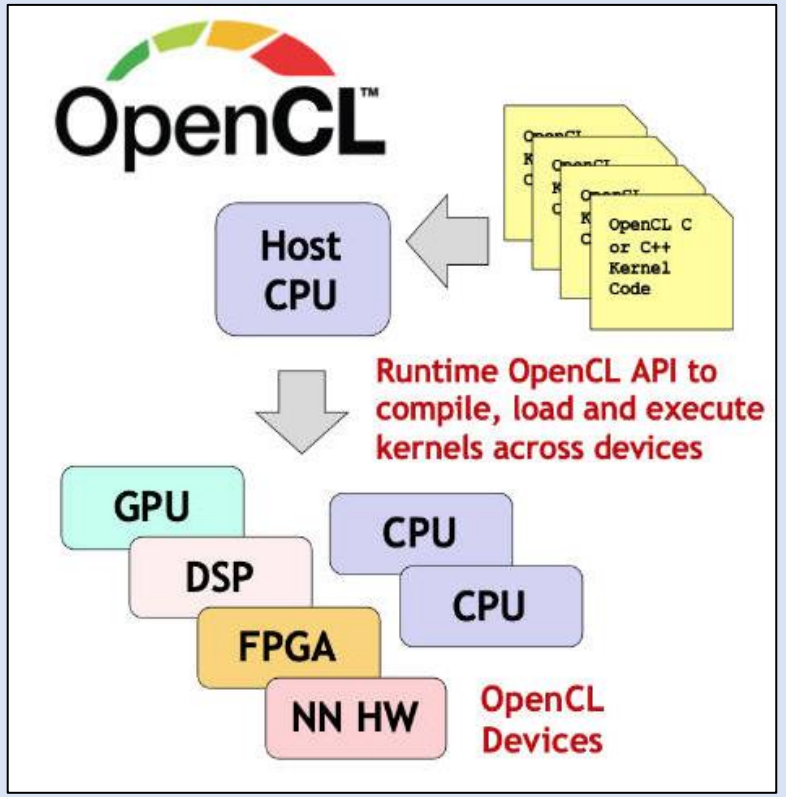
OpenCL

- Open, royalty-free standard maintained by Khronos Group.
- Can be used in multiple devices (CPUs, GPUs, FPGAs)
- Code once, run everywhere.*

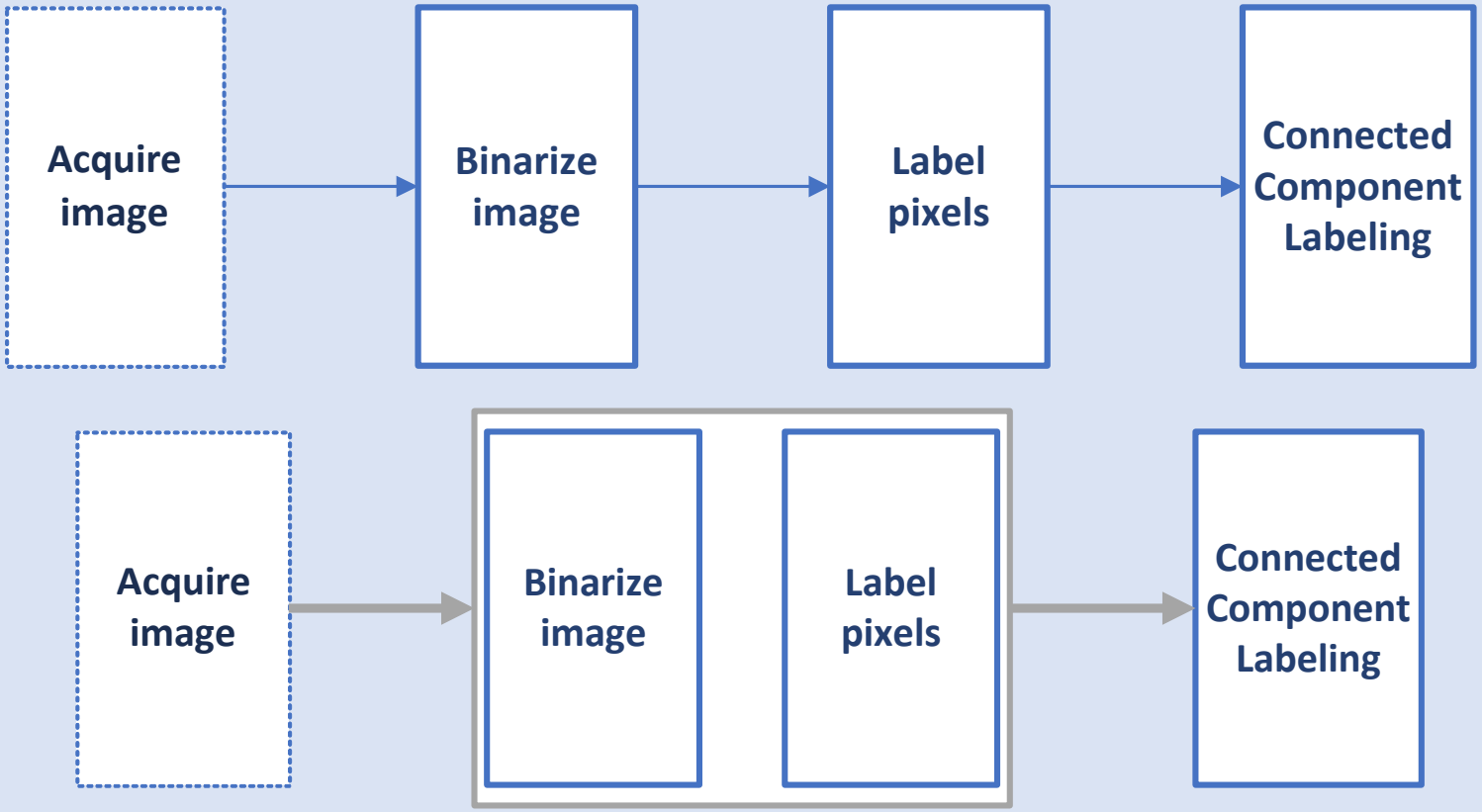


OpenCL

- The programs executed in the different devices are called kernels, which are written in a C code style.
- The functions to manage memory and launch kernels from the host are officially in C or, with a wrapper, C++. (Although there are unofficial implementations in Java and Python)

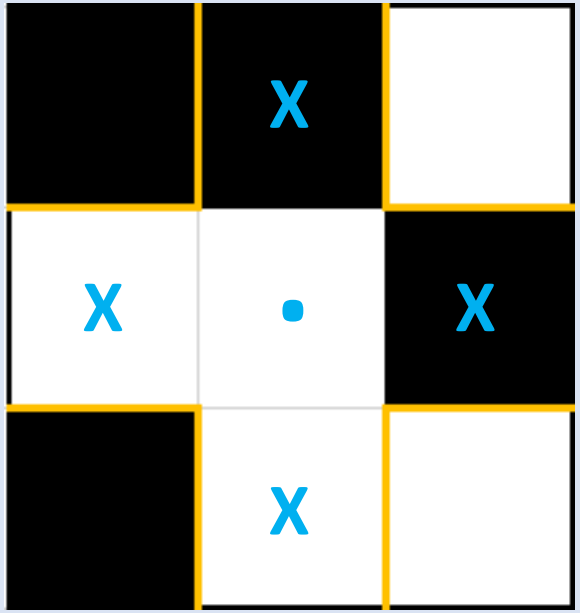


Algorithm: CCL

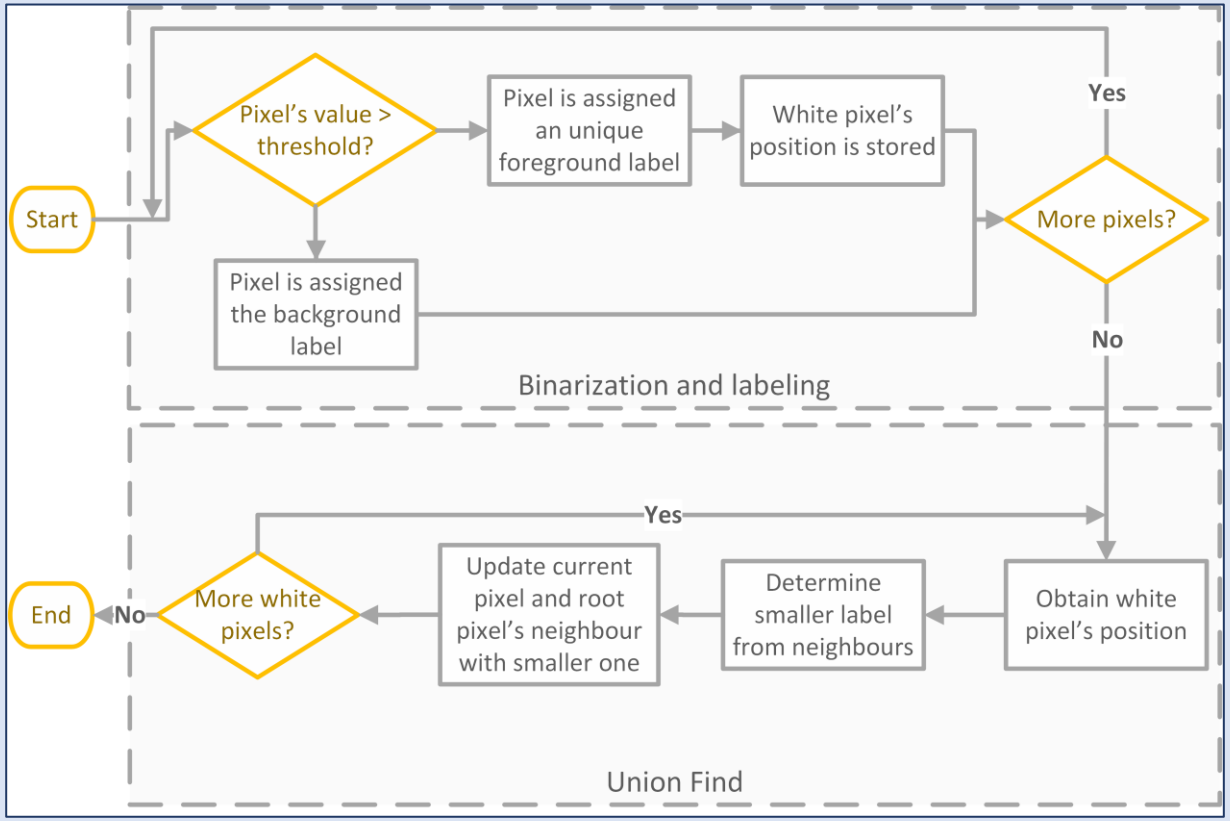


Algorithm: Union Find

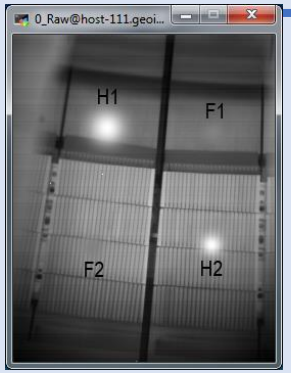
- One of the possible implementations for Connected Components Labeling.
- Check neighbour pixels and assign same label if related
- In this case we used 4-connectivity

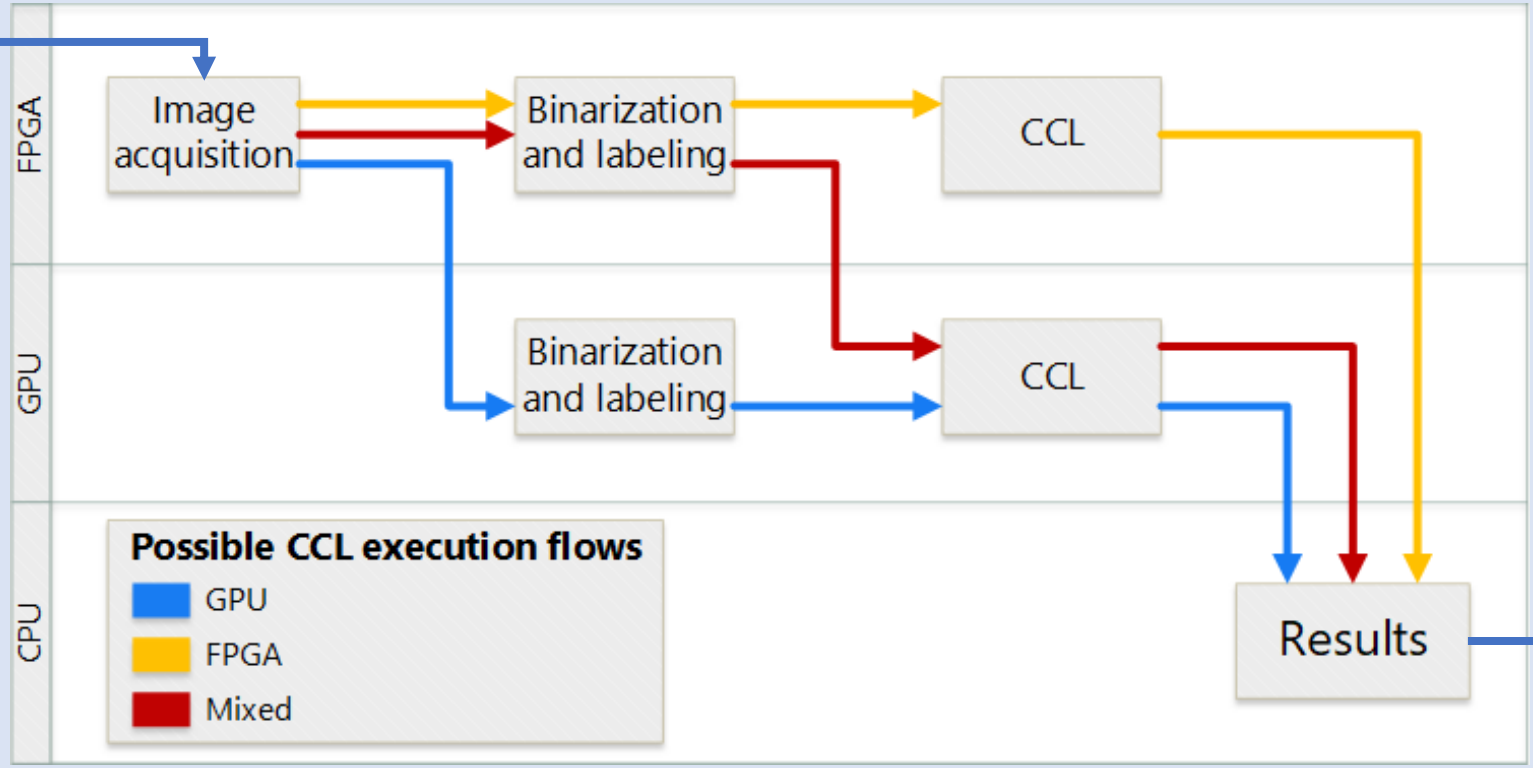


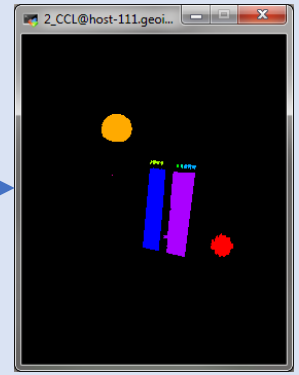
Algorithm


Start
Pixel's value > threshold?
Pixel is assigned an unique foreground label
White pixel's position is stored
More pixels?
Yes
No
Pixel is assigned the background label
Binarization and labeling
Yes
Obtain white pixel's position
Determine smaller label from neighbours
Update current pixel and root pixel's neighbour with smaller one
More white pixels?
No
End
Union Find

Executions flows





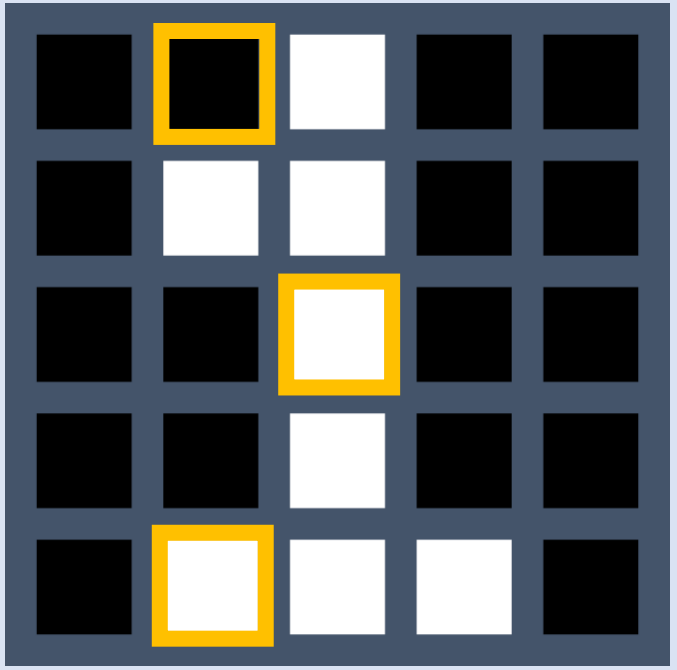


POLITÉCNICA

GRUPO DE INVESTIGACIÓN EN INSTRUMENTACIÓN Y ACÚSTICA APLICADA

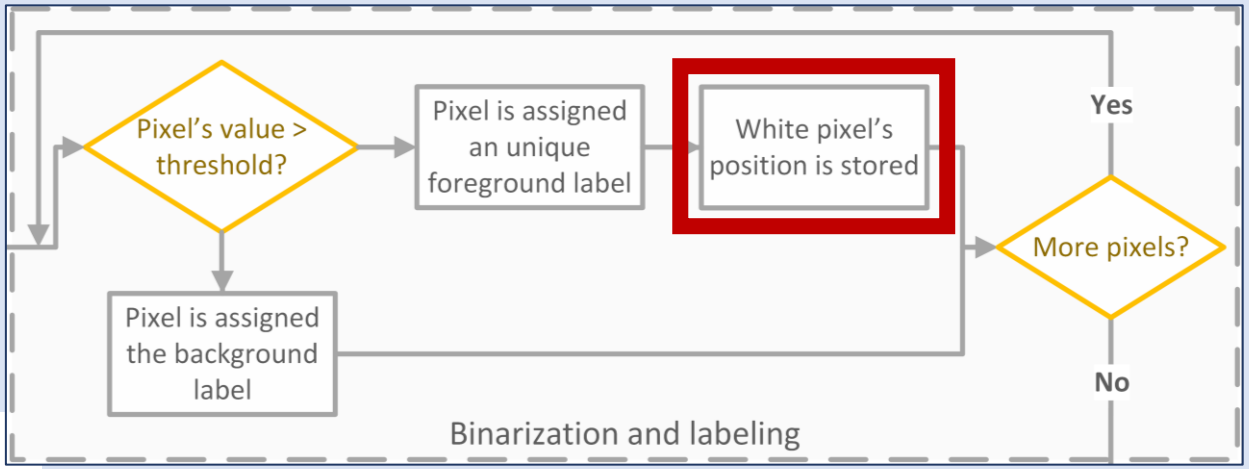
Algorithm optimizations: GPU

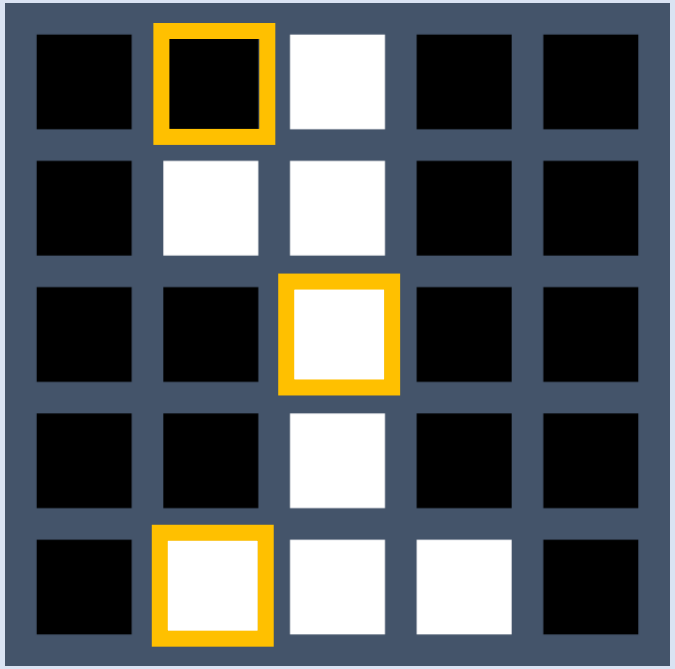
- The pixels' process order can not be determined
- Multiple pixels are processed at the same time



Algorithm optimizations: GPU

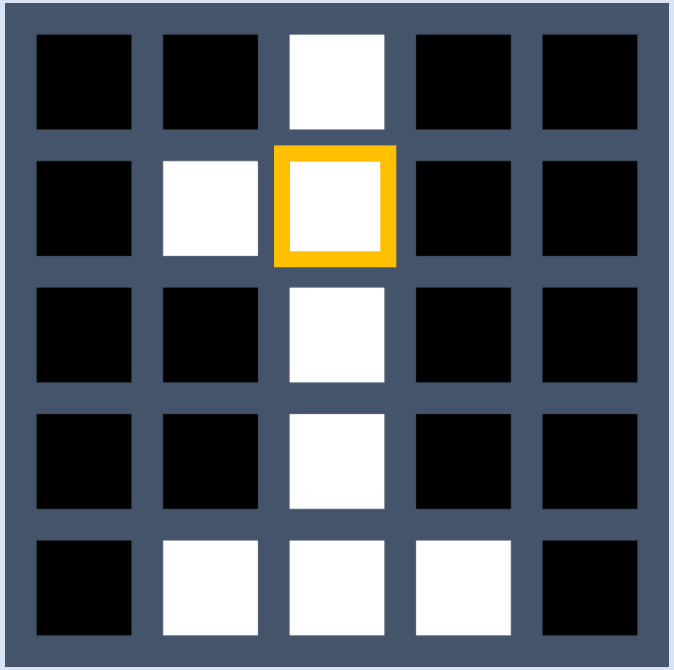
- Check the 4 cardinal pixels
- Use atomics to avoid concurrency problems when storing white pixel positions





Algorithm optimizations: FPGA

- Pixels are processed sequentially
  - Only the west and north pixels of the current pixel need to be checked
  - No concurrency problem = No use of atomics
- Access to global memory is slow
  - Limit percentage of white pixels in a image to copy the data to local memory
  - Limit image size to copy it to local memory
- Pipeline processing

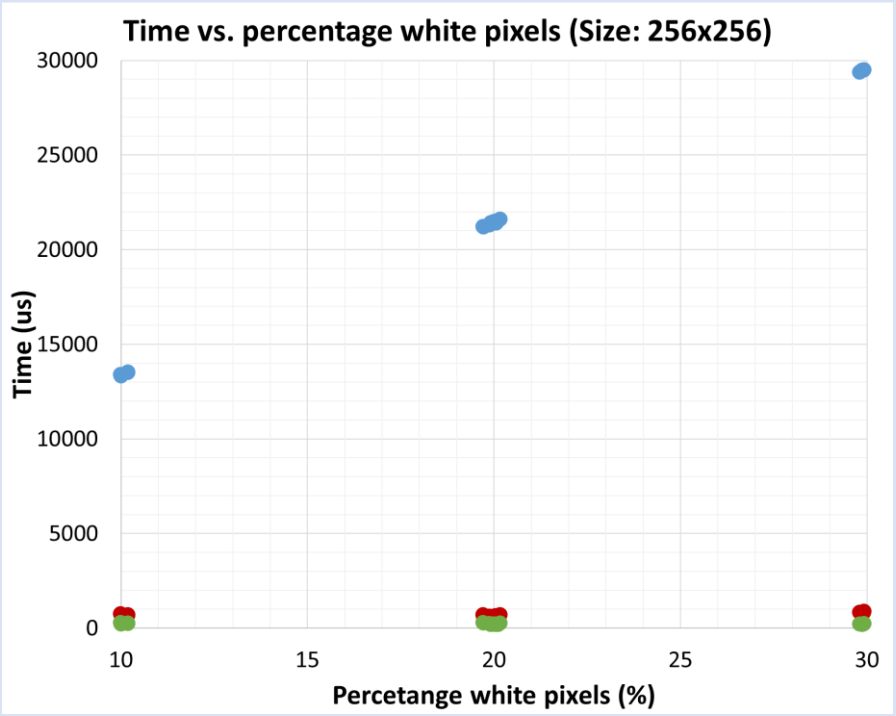


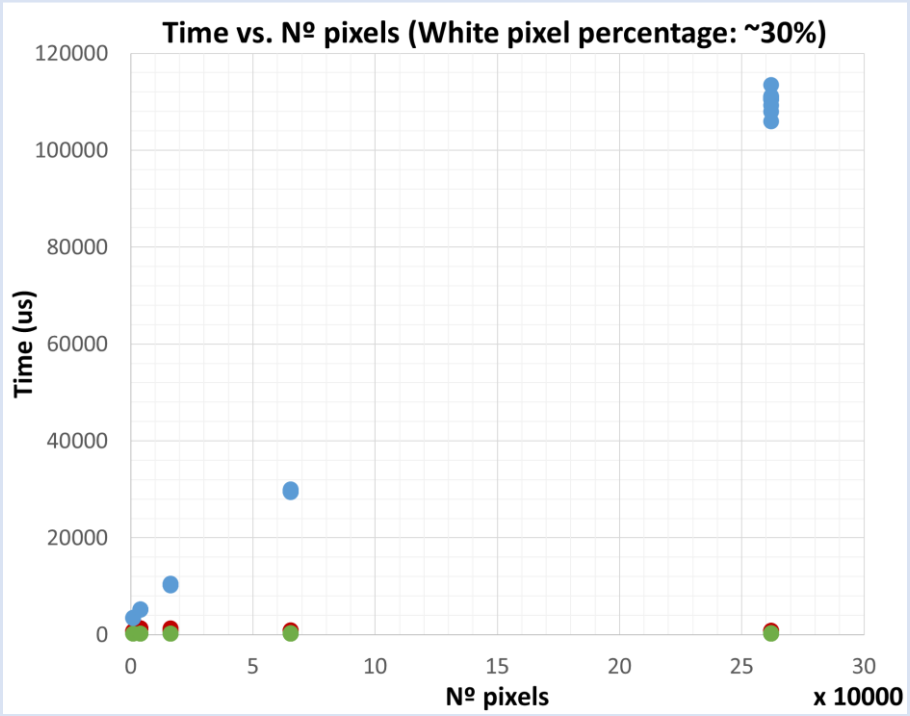
Tests

- Synthetic images composed of black and white pixels placed randomly.
- White pixel density range from 10% to 30%.
- Image sizes from 32x32 to 512x512 (8-bit pixels. (Due to memory limitations on the FPGA)
- Repeated 1000 times in each device

Results

FPGA
AMD RX 5600
NVIDIA RTX 3080 SUPER





*FPGA running at 146.07 MHz

POLITÉCNICA

GRUPO DE INVESTIGACIÓN EN INSTRUMENTACIÓN Y ACÚSTICA APLICADA

Conclusions

- Not GPU available in data acquisition platforms (PXIe/MTCA) -> Acquisition done by FPGA
- Image data transfer implies time
- FPGA can binarize and label while pixels arrive (pipelining)
- Using an FPGA results in a lower power consumption.
- OpenCL makes easier testing different execution flows due to only using one programing language.
- GPU better suited for processing the whole image at the same time
- FPGA could improve performance by parallelization and pipelining
- Scalability of FPGA performance with bigger FPGAs may improve results while conserving determinism.

Future work

- Improve the FPGA algorithm
- Use a newer FPGA
  - Higher clock speed (Currently 146.07 Mhz)
  - Faster memory access
- Test in a full heterogenous system