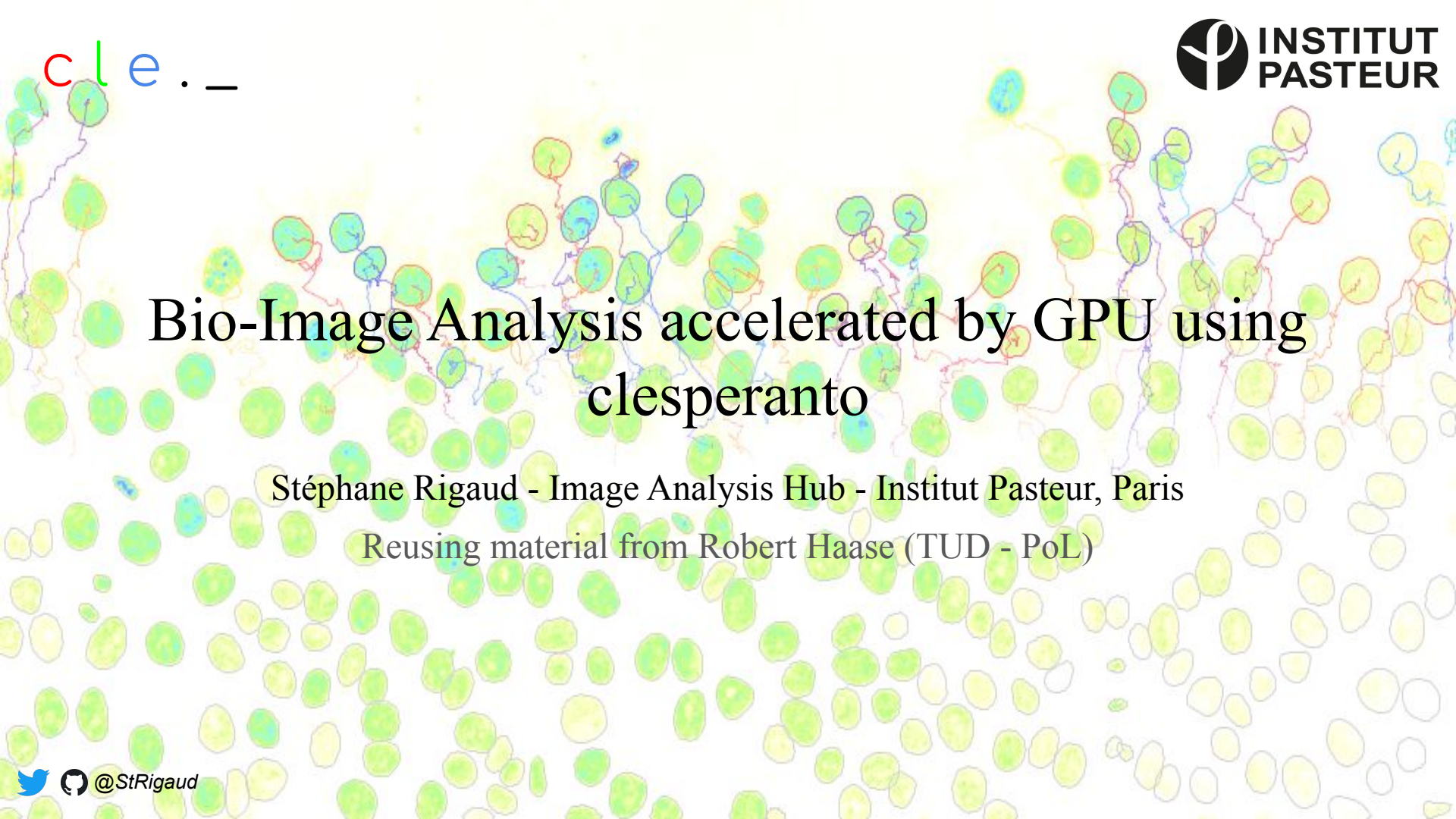


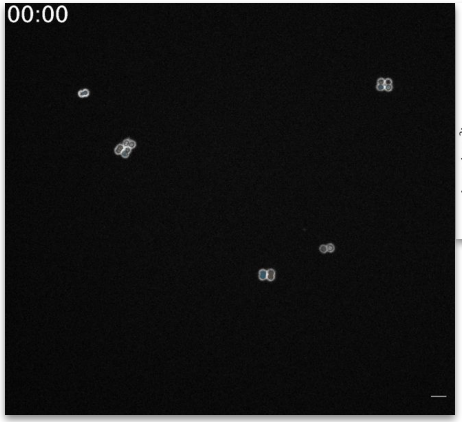
Bio-Image Analysis accelerated by GPU using clesperanto

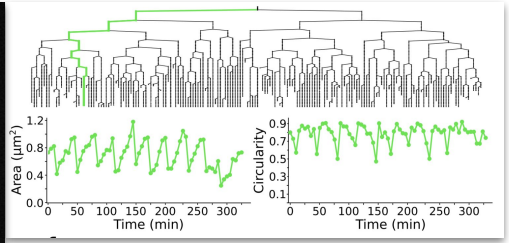
Stéphane Rigaud - Image Analysis Hub - Institut Pasteur, Paris

Reusing material from Robert Haase (TUD - PoL)

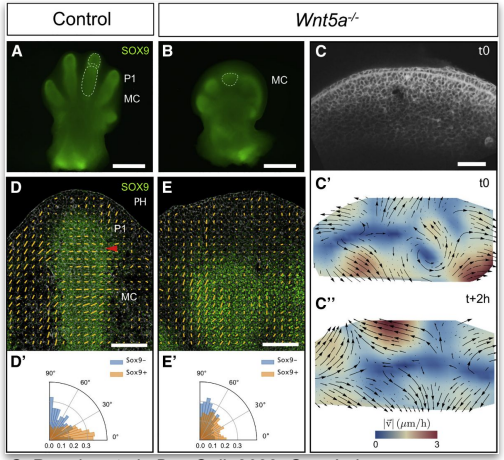
@StRigaud

Bio-Image Analysis





Ershov, Phan, Pylvänäinen, Rigaud, *et.al. nat methods, 2022*



C. Parada, *et.al., Dev Cell, 2022, Gros Lab*

Frameworks Libraries Languages

ImgLib2

OpenCV

Java

Written in

Components

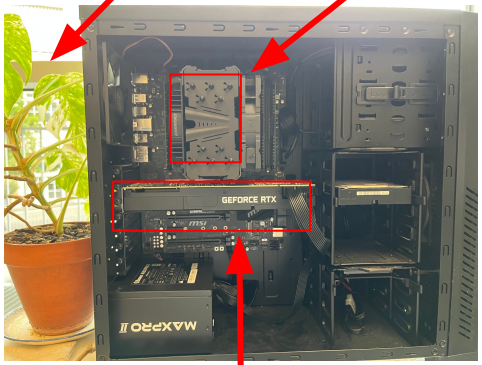
Pipeline

Processing Units

Diversity of units: DSP, TPU, FPGA, etc.

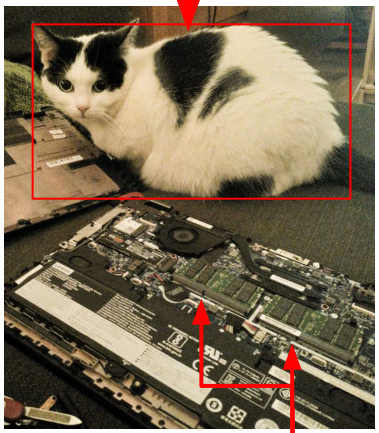
Pretty plant

CPU



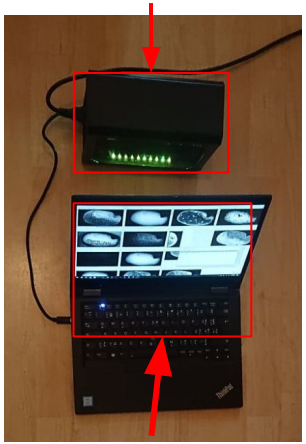
dedicated GPU

Pixel



integrated GPU

external GPU



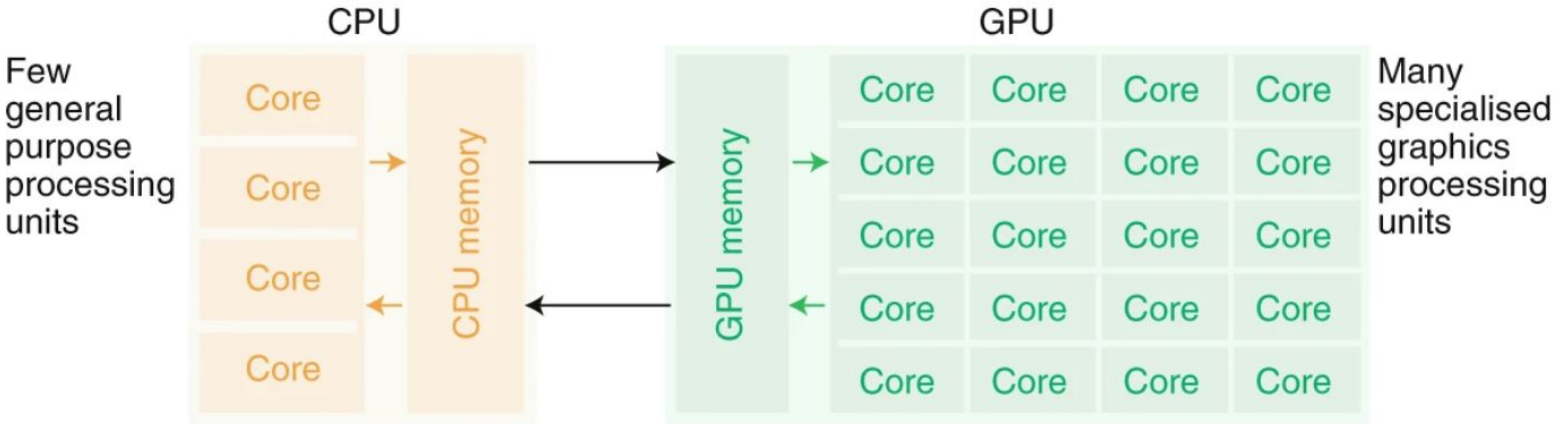
Showing off

cloud

HPC

Processing Units

Quality V.S. Quantity



Processing Units

The forbidden slide

Youtube mythbuster CPU vs GPU

80 ms by MythBuster (slow motion)

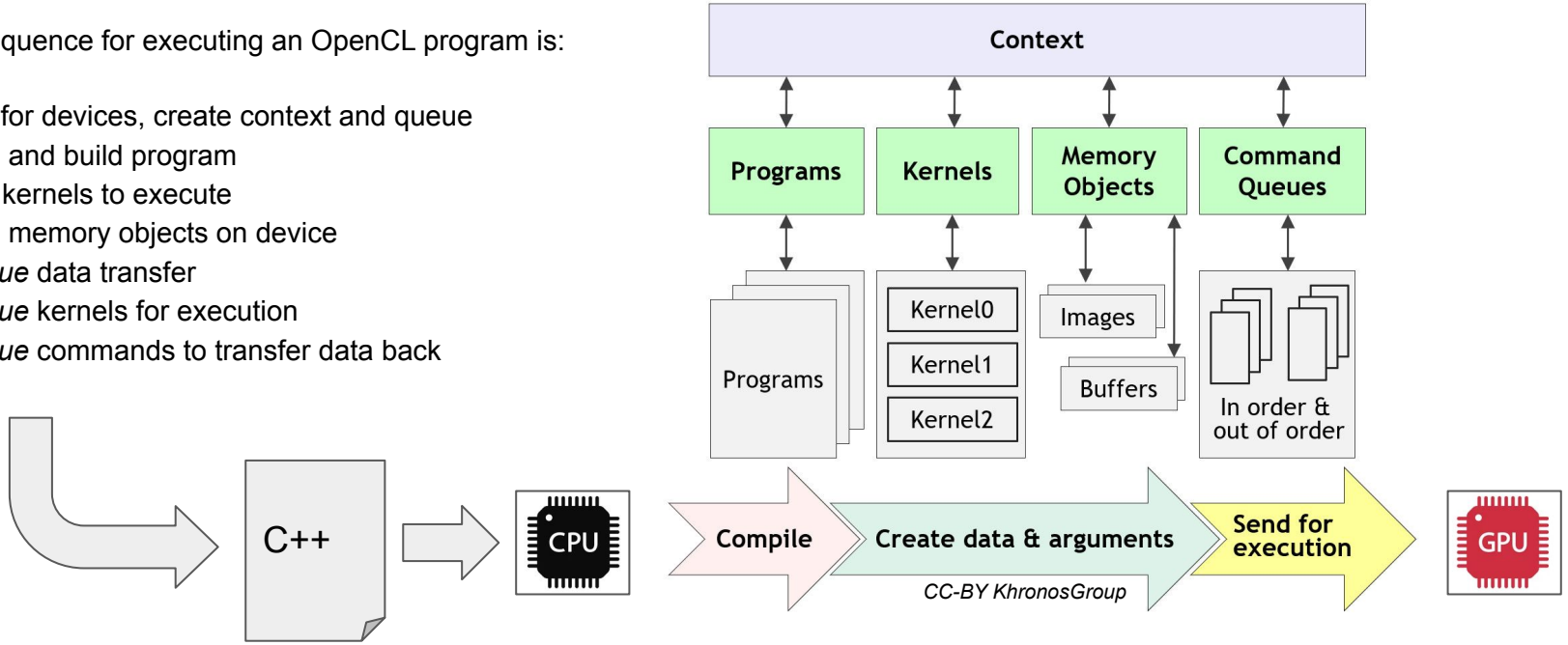
4 years by Da Vinci

GPU programing

Close to hell

A complete sequence for executing an OpenCL program is:

1. Query for devices, create context and queue
2. Create and build program
3. Select kernels to execute
4. Create memory objects on device
5. *Enqueue* data transfer
6. *Enqueue* kernels for execution
7. *Enqueue* commands to transfer data back



GPU programing

Close to hell … times 150

maximum_z_projection.cl

kernels > maximum_z_projection.cl

You, 18 months ago | 2 authors (You and others)

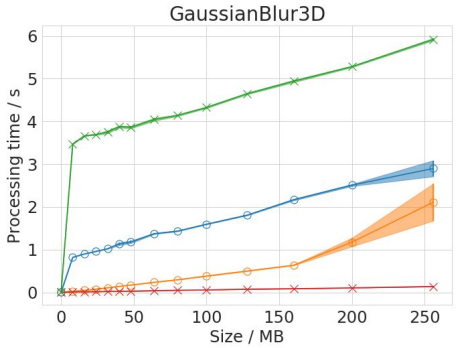
```
__constant sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE | CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

__kernel void maximum_z_projection(
    IMAGE_src_TYPE  src,
    IMAGE_dst_TYPE  dst
)
{
  const int x = get_global_id(0);
  const int y = get_global_id(1);

  IMAGE_src_PIXEL_TYPE max = 0;
  for (int z = 0; z < GET_IMAGE_DEPTH(src); ++z)
  {
    const IMAGE_src_PIXEL_TYPE value = READ_IMAGE(src, sampler, POS_src_INSTANCE(x,y,z,0)).x;
    if (value > max || z == 0) {
      max = value;
    }
  }

  WRITE_IMAGE(dst, POS_dst_INSTANCE(x,y,0,0), CONVERT_dst_PIXEL_TYPE(max));
}
```

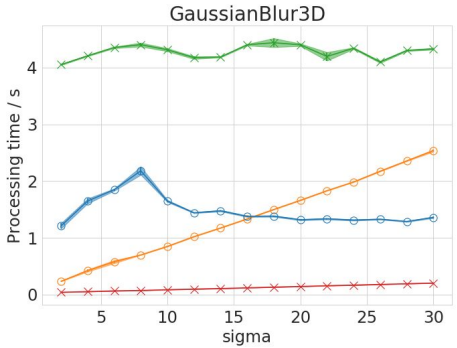
CLIJ (ancestor of clesperanto)

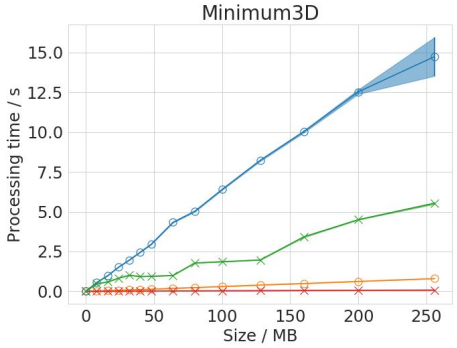
Haase et al. Nat Methods (2020)

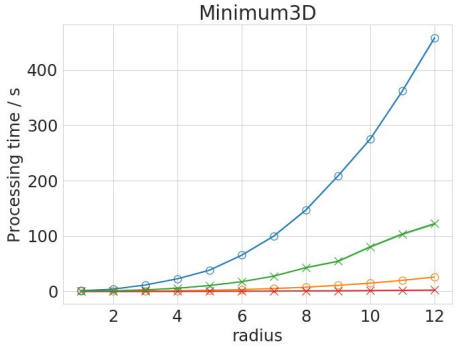
Benchmarking

V.S.





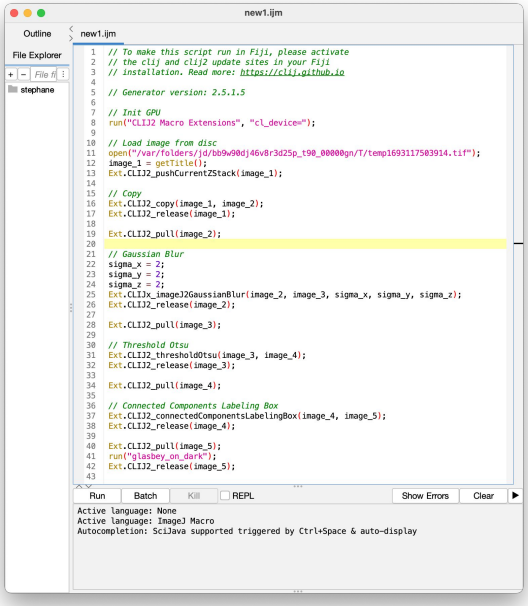


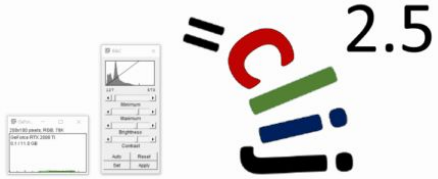


CLIJ

https://clij.github.io

User-friendly GPU extension for FIJI

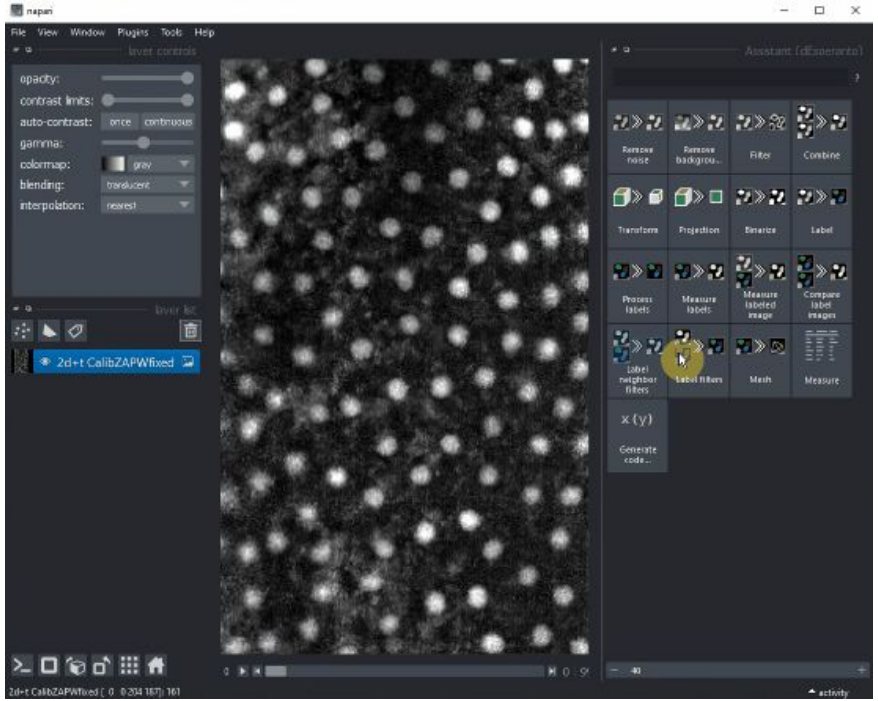




Napari-Assistant

github.com/clEsperanto/napari_pyclesperanto_assistant

User-friendly GPU extension for Napari



Clesperanto project

OpenCL - Esperanto

- ➔ Target multiple languages and frameworks
- ➔ Clear and standardized operation call

```
cle.operation_name( input, output, arguments )
```

- ➔ Maintain a common API usage between languages
- ➔ Reduce implementation redundancy and numerical inconsistency
- ➔ Improve code sharing and reproducibility
- ➔ Improve maintainability

Clesperanto project

OpenCL - Esperanto

Fiji jython Python C++

my_code.x

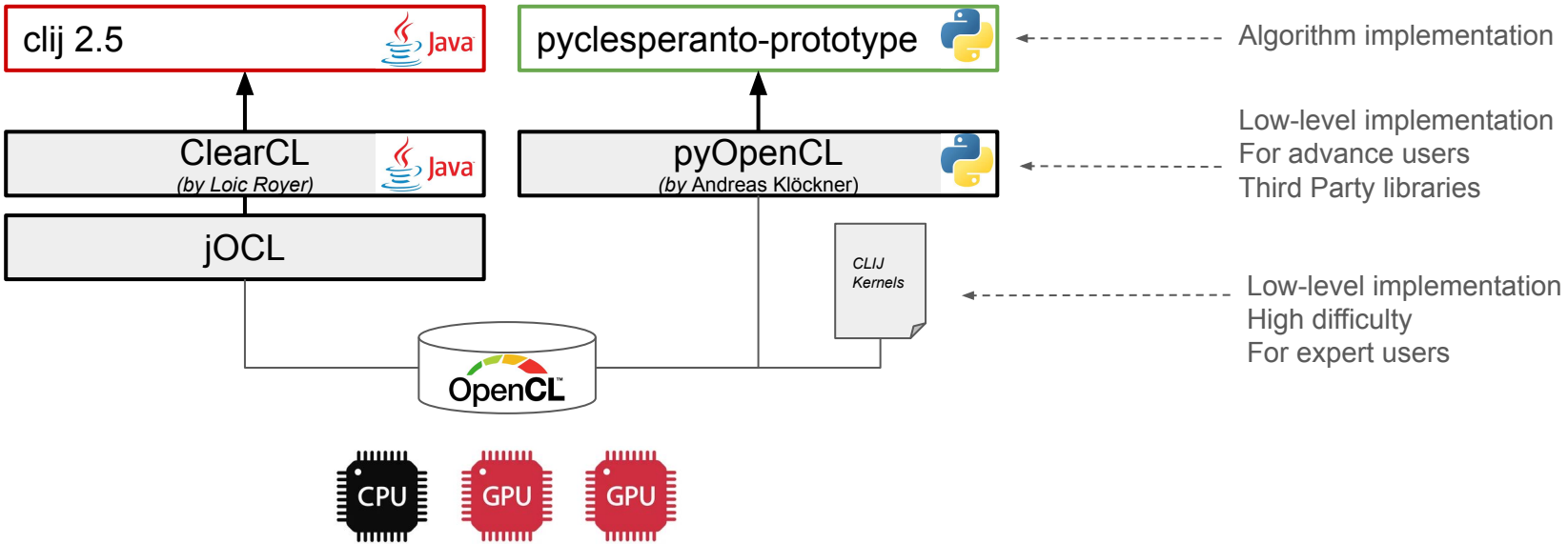
```
auto blurred = cle.create_like(blobs);
cle.gaussian_blur(blobs, blurred, 3);


auto binary = cle.create_like(blurred);
cle.threshold_otsu(blurred, binary);


auto label = cle.create_like(binary);
cle.connect_components_labeling(binary, label);
```

With some limitations …

Clesperanto project

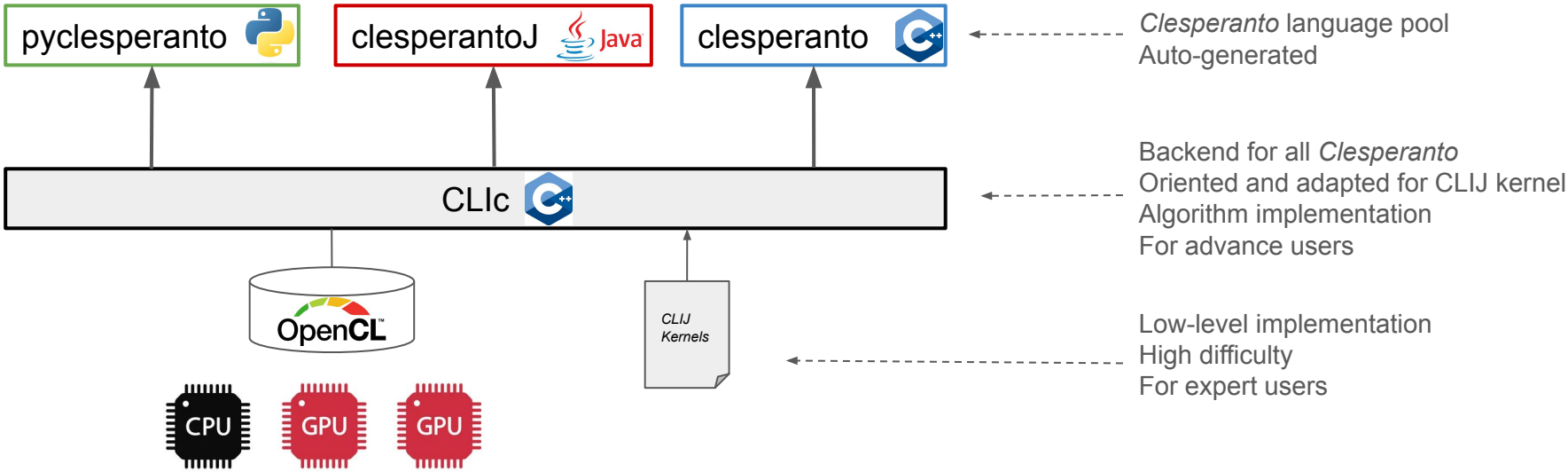
clearcl | pyopencl | jocl

The Prototype



Clesperanto project

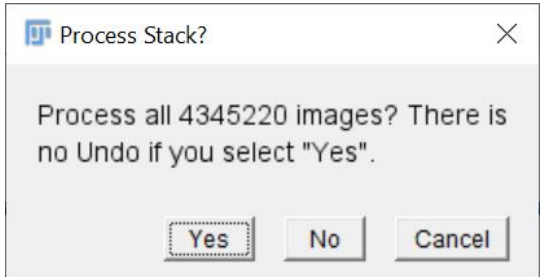
The Goal



Memory management

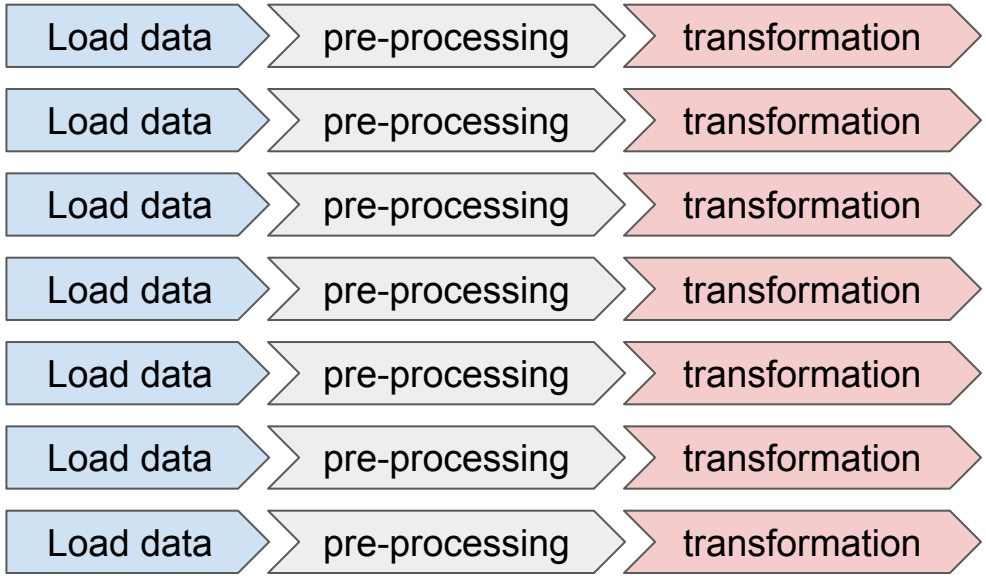
Classical way of processing data

- Load data
- Load data
- Load data
- Load data
- Load data
- Load data
- Load data



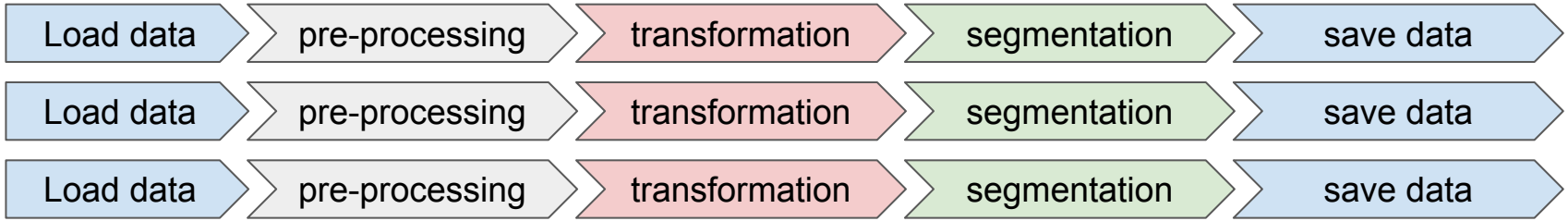
Memory management

Classical way of processing data … is suboptimal



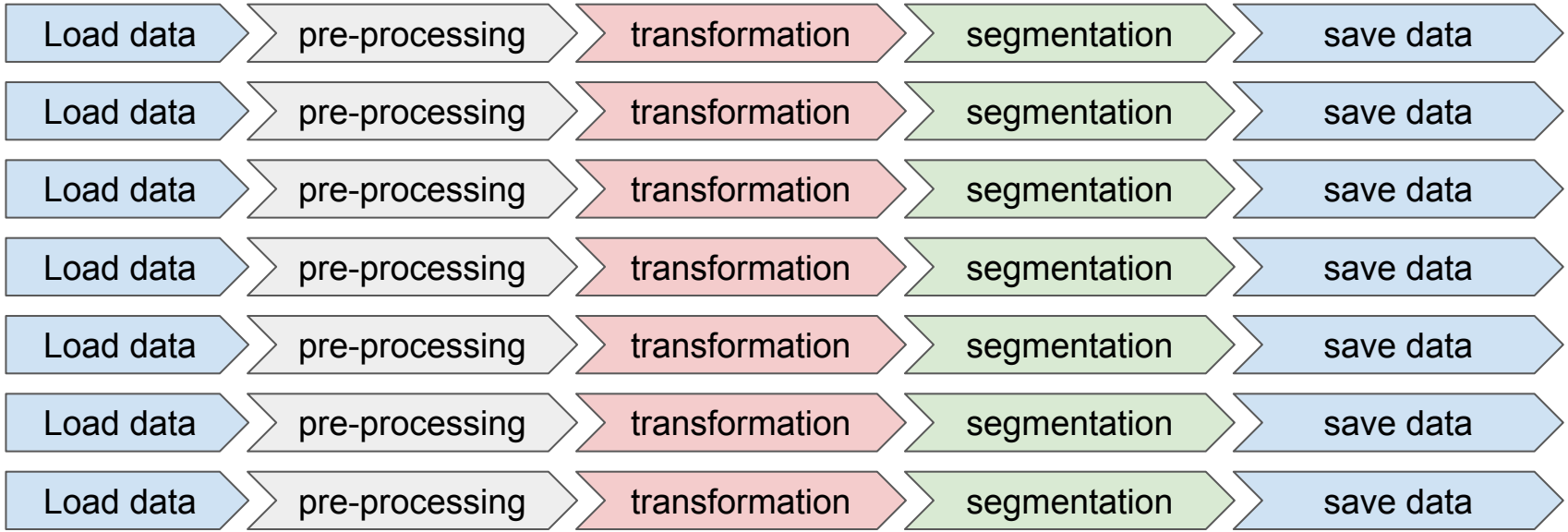
Memory management

Process per data units …



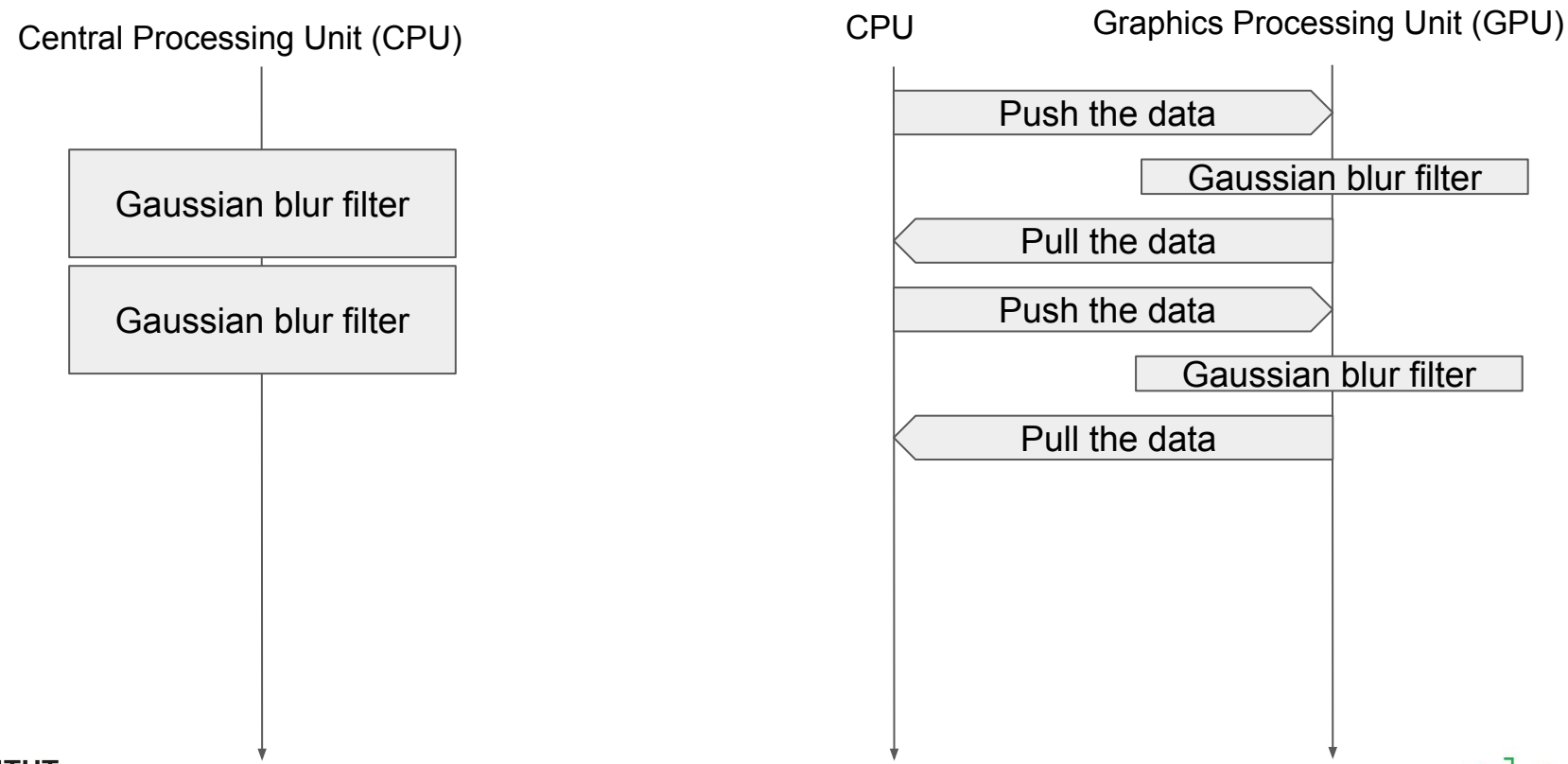
Memory management

Process per data units … and distribute it between systems

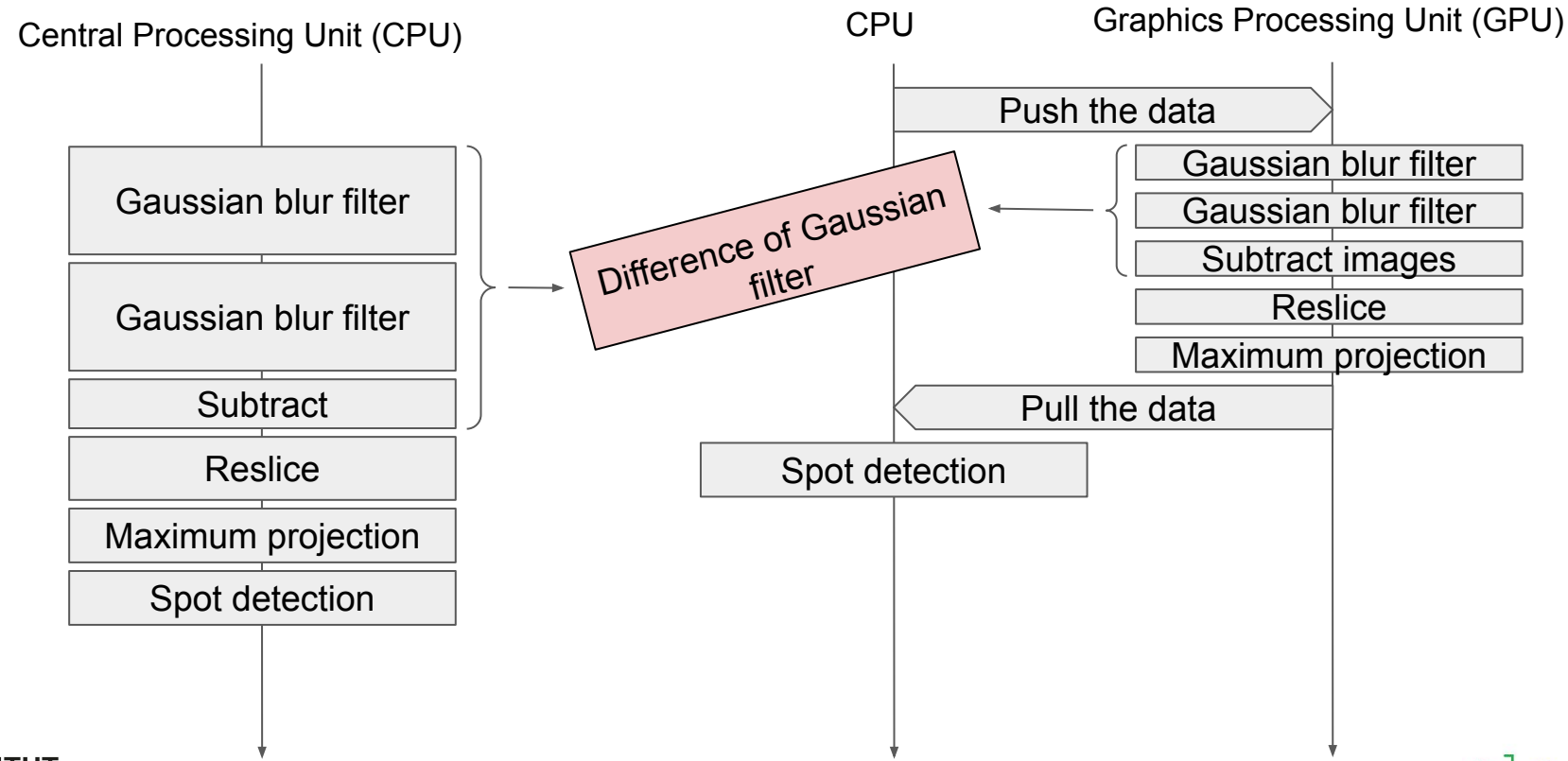


Would need to rely on a task distribution library dask

Workflow construction

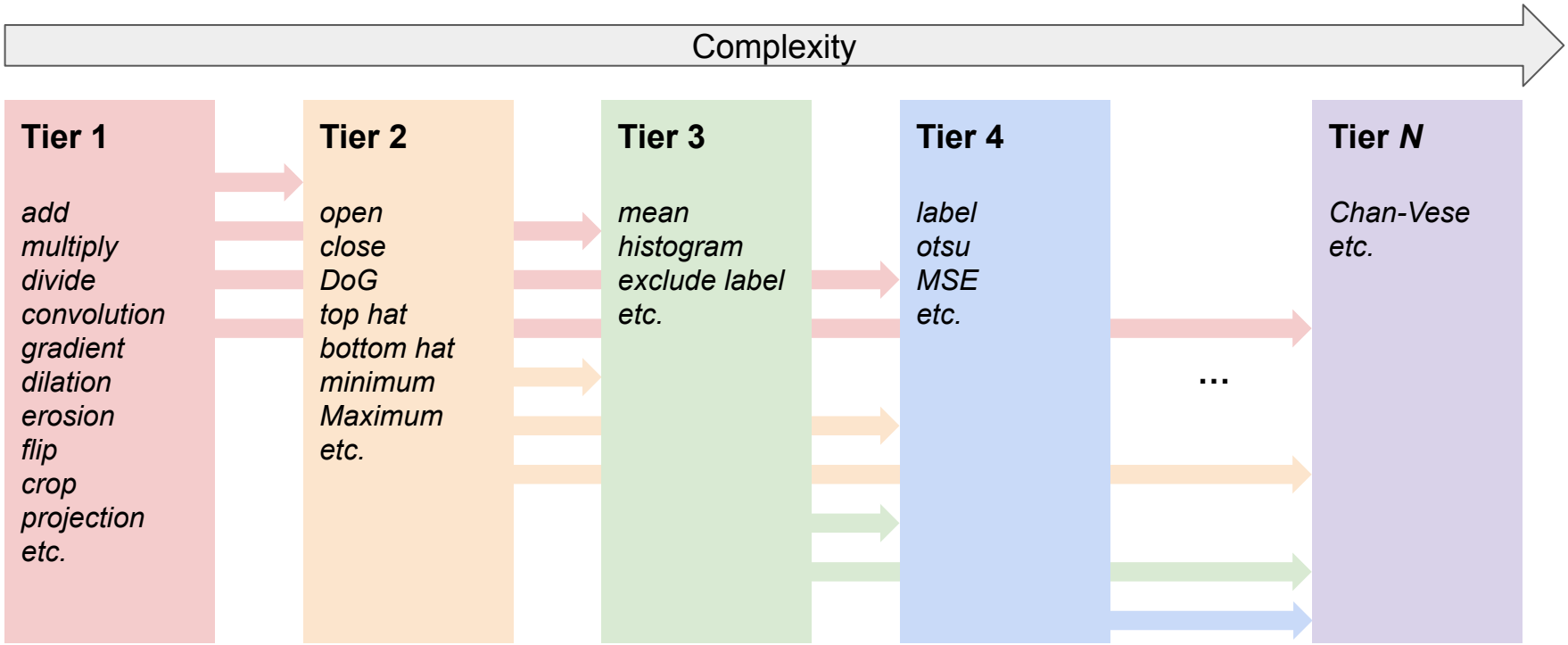


Workflow construction



Extensibility

Towards advance algorithm



Extensibility

Open-source community

pyclesperanto-prototype

```python
from .._tier0 import execute
from .._tier0 import create_2d_yx
from .._tier0 import plugin_function
from .._tier0 import Image

@plugin_function(output_creator=create_2d_yx, categories=["projection", "in assistant"])
def maximum_z_projection(source: Image, destination_max: Image = None) -> Image:

    parameters = {
        "ds
        "sr
    }

    execute(
        __fi
        "../
        "max
        dest
        para
    )

    return d
```

```python
from pyclesperanto_prototype._tier0 import Image
from pyclesperanto_prototype._tier1 import gaussian_blur
from pyclesperanto_prototype._tier0 import create_like
from pyclesperanto_prototype._tier0 import plugin_function
from pyclesperanto_prototype._tier2 import subtract_images

@plugin_function(categories=["filter", "background removal"])
def difference_of_gaussian(
    source: Image,
    destination: Image = None,
    sigma1_x: float = 2,
    sigma1_y: float = 2,
    sigma1_z: float = 2,
    sigma2_x: float = 2,
    sigma2_y: float = 2,
    sigma2_z: float = 2,
) -> Image:
    temp1 = create_like(destination)
    temp2 = create_like(destination)

    gaussian_blur(source, temp1, sigma1_x, sigma1_y, sigma1_z)
    gaussian_blur(source, temp2, sigma2_x, sigma2_y, sigma2_z)

    return subtract_images(temp1, temp2, destination)
```

pyclesperanto

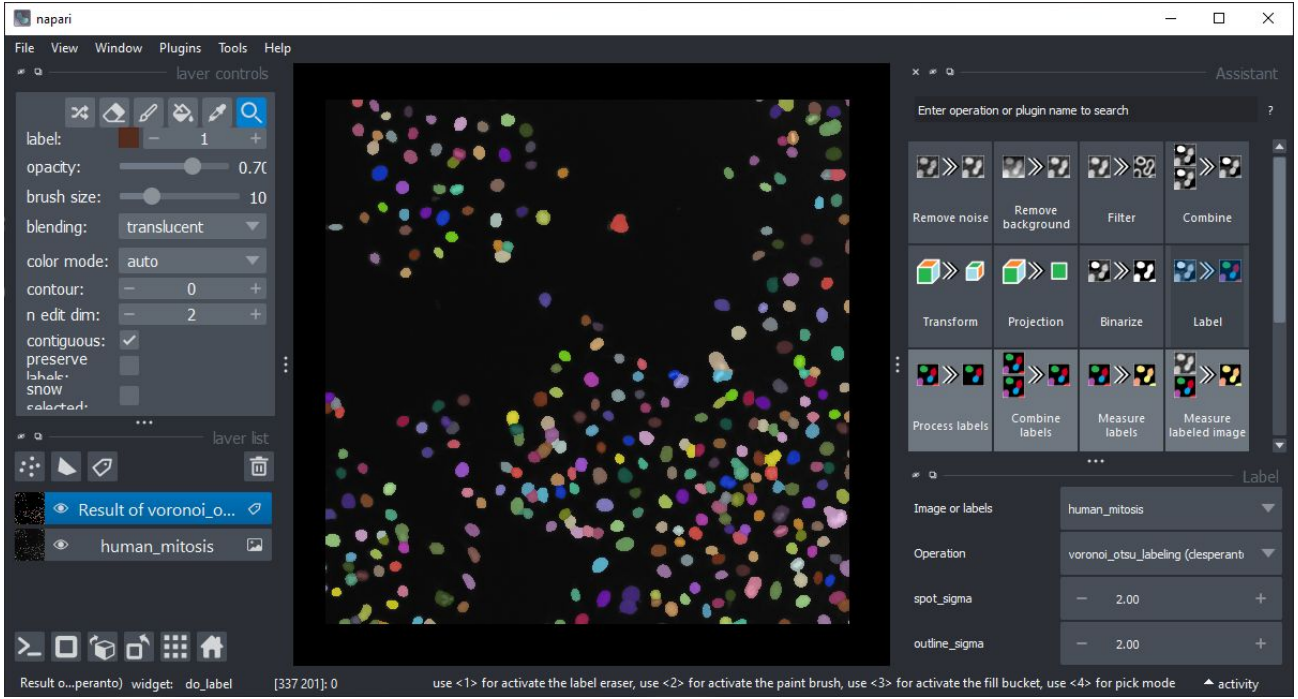
```cpp
auto
maximum_z_projection_func(const Device::Pointer & device, const Array::Pointer & src, Array::Pointer dst)
  -> Array::Pointer
{
  tier0::create_xy(src, dst);
  const KernelInfo     kernel = { "maximum_z_projection", kernel::maximum_z_projection };
  const ParameterList params = { { "src", src }, { "dst", dst } };
  const RangeArray     range = { dst->width(), dst->height(), dst->depth() };
  execute(device, kernel, params, range);
  return dst;
}
```

```cpp
auto
difference_of_gaussian_func(const Device::Pointer & device,
                            const Array::Pointer &  src,
                            Array::Pointer          dst,
                            float                   sigma1_x,
                            float                   sigma1_y,
                            float                   sigma1_z,
                            float                   sigma2_x,
                            float                   sigma2_y,
                            float                   sigma2_z) -> Array::Pointer
{
  tier0::create_like(src, dst, dType::FLOAT);
  auto gauss1 = tier1::gaussian_blur_func(device, src, nullptr, sigma1_x, sigma1_y, sigma1_z);
  auto gauss2 = tier1::gaussian_blur_func(device, src, nullptr, sigma2_x, sigma2_y, sigma2_z);
  return tier1::add_images_weighted_func(device, gauss1, gauss2, dst, 1, -1);
}
```

Interoperability - napari assistant

www.napari-hub.org/plugins/napari-assistant

User-friendly bio-image analysis



Acknowledgments

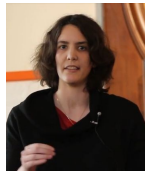

Jean-Yves Tinevez


Marvin Albert


Minh-Son Phan


Cherif Latrech


Laura Xenard


Gaëlle Letort

https://fiji.sc

https://napari.dev


Robert Haase


Till Korten


Johannes Soltwebel


Brian Northon


Pradeep Rajasekhar


Mathew McCormick

Rest of the World

https://image.sc

https://eubias.org/NEUBIAS


Martin Jones


Jon Smith


Grahams Ross

Funding:

Exercises

In Github >

Installation_instruction/3_gpu_napari_assistant

Install either the FIJI plugin or the Napari:

- **FIJI > known territory, safe world, with flowers and rainbow 🌈 🦄 🧸**
- **Napari > adventure, new world, the future, super cool with a dash of danger 😎 🚀 🤘**