

cle.\_

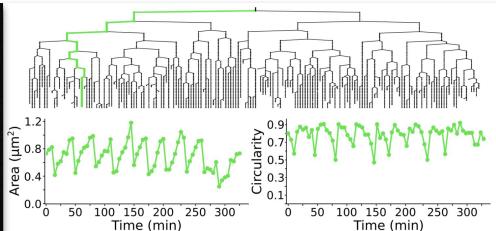
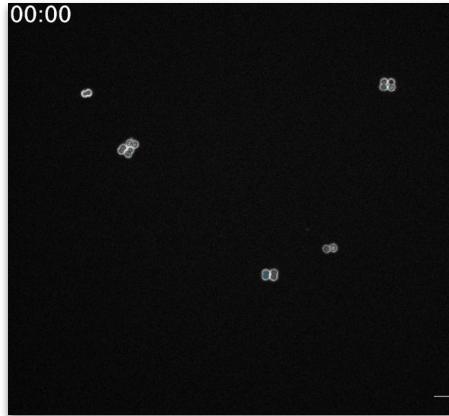


# Bio-Image Analysis accelerated by GPU using clesperanto

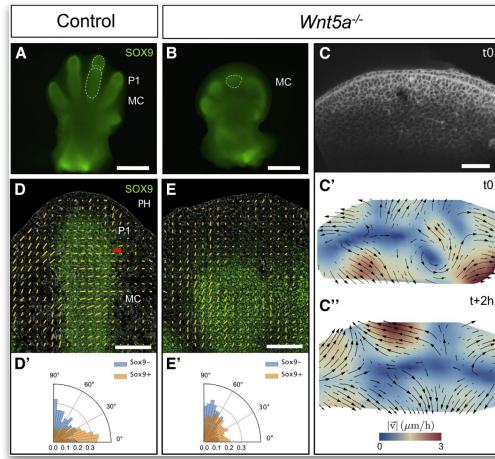
Stéphane Rigaud - Image Analysis Hub - Institut Pasteur, Paris

Reusing material from Robert Haase (TUD - PoL)

# Bio-Image Analysis



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



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

## Frameworks



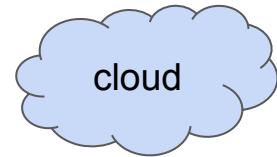
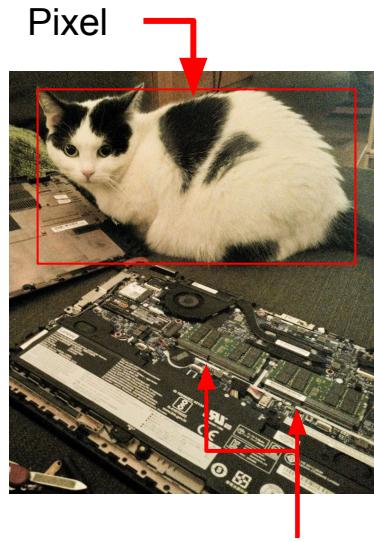
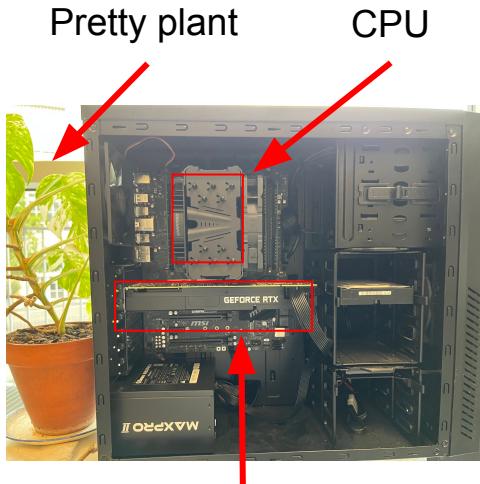
## Pipeline

Written in

Components

# Processing Units

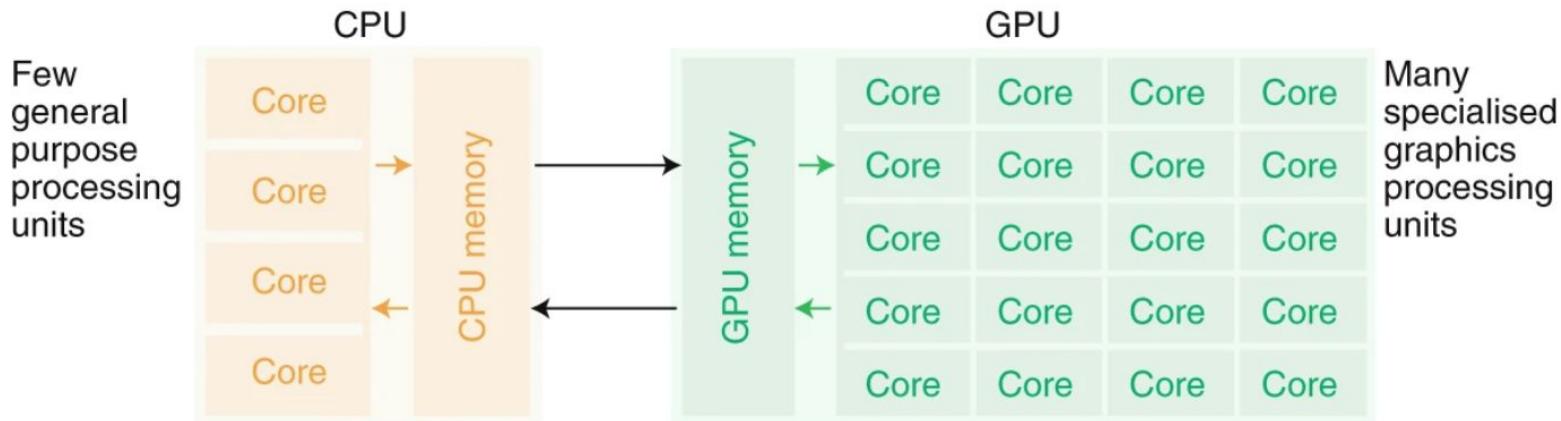
Diversity of units: DSP, TPU, FPGA, etc.



HPC

# Processing Units

Quality V.S. Quantity



# Processing Units

---

The forbidden slide

[Youtube mythbuster CPU vs GPU](#)

*80 ms by MythBuster (slow motion)*

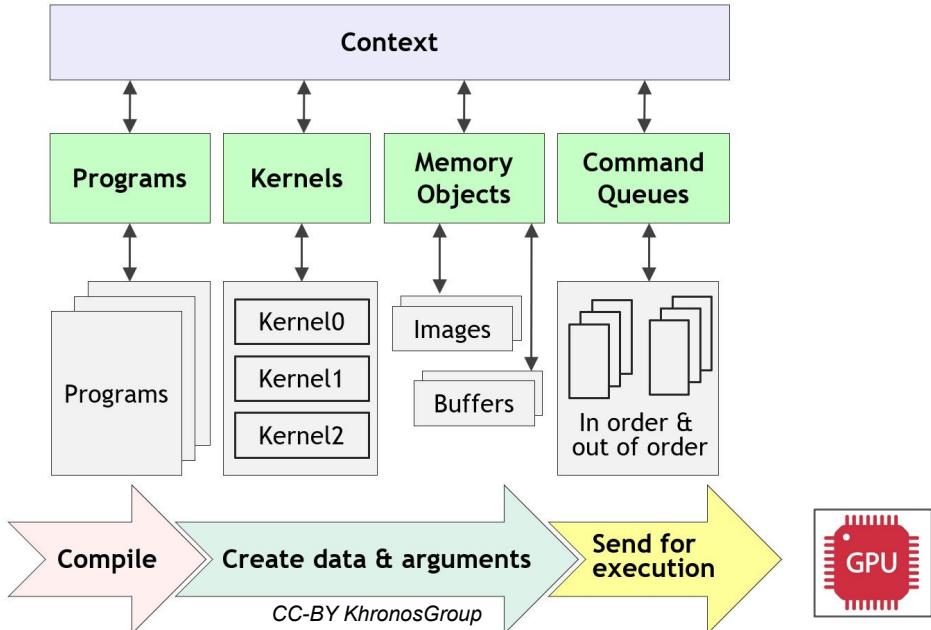
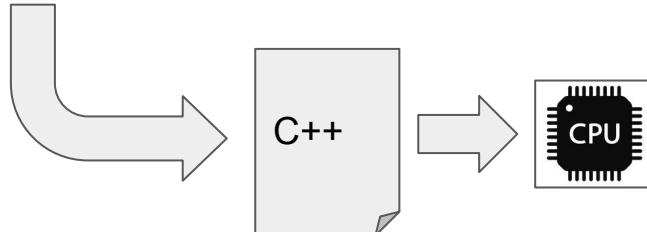
*4 years by Da Vinci*

# GPU programming

## 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 programming

Close to hell ... times 150

```
maximum_z_projection.cl ×
kernels > maximum_z_projection.cl
You, 18 months ago | 2 authors (You and others)
1 __constant sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE | CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;
2
3 __kernel void maximum_z_projection(
4     IMAGE_src_TYPE src,
5     IMAGE_dst_TYPE dst
6 )
7 {
8     const int x = get_global_id(0);
9     const int y = get_global_id(1);
10
11    IMAGE_src_PIXEL_TYPE max = 0;
12    for (int z = 0; z < GET_IMAGE_DEPTH(src); ++z)
13    {
14        const IMAGE_src_PIXEL_TYPE value = READ_IMAGE(src, sampler, POS_src_INSTANCE(x,y,z,0)).x;
15        if (value > max || z == 0) {
16            max = value;
17        }
18    }
19
20    WRITE_IMAGE(dst, POS_dst_INSTANCE(x,y,0,0), CONVERT_dst_PIXEL_TYPE(max));
21 }
```

## User-friendly GPU extension for FIJI



new1.ijm

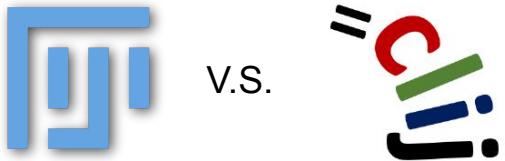
```
1 // To make this script run in Fiji, please activate
2 // the clij and clij2 update sites in your Fiji
3 // installation. Read more: https://clij.github.io
4
5 // Generator version: 2.5.1.5
6
7 // Init GPU
8 run("CLIJ2 Macro Extensions", "cl_device=0");
9
10 // Load image from disc
11 open("/var/folders/jd/bb9y9d46v8r3d25p_t90_00000gn/T/temp1693117583914.tif");
12 image_1 = getTitle();
13 Ext.CLIJ2_pushCurrentZStack(image_1);
14
15 // Copy
16 Ext.CLIJ2_copy(image_1, image_2);
17 Ext.CLIJ2_release(image_1);
18
19 Ext.CLIJ2_pull(image_2);
20
21 // Gaussian Blur
22 sigma_x = 2;
23 sigma_y = 2;
24 sigma_z = 2;
25 Ext.CLIJx(image_2)GaussianBlur(image_2, image_3, sigma_x, sigma_y, sigma_z);
26 Ext.CLIJ2_release(image_2);
27
28 Ext.CLIJ2_pull(image_3);
29
30 // Threshold Otsu
31 Ext.CLIJ2_thresholdOtsu(image_3, image_4);
32 Ext.CLIJ2_release(image_3);
33
34 Ext.CLIJ2_pull(image_4);
35
36 // Connected Components Labeling Box
37 Ext.CLIJ2_connectedComponentsLabelingBox(image_4, image_5);
38 Ext.CLIJ2_release(image_4);
39
40 Ext.CLIJ2_pull(image_5);
41 run("glasbey_on_dark");
42 Ext.CLIJ2_release(image_5);
```

Run Batch Kill REPL Show Errors Clear

Active language: None  
Active language: ImageJ Macro  
Autocompletion: Scilava supported triggered by Ctrl+Space & auto-display

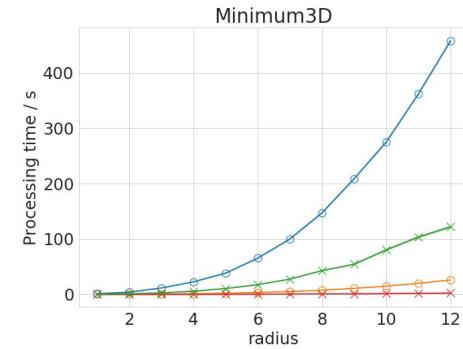
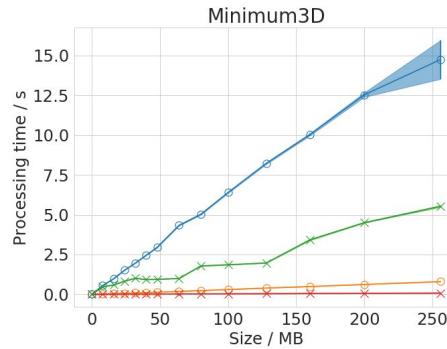
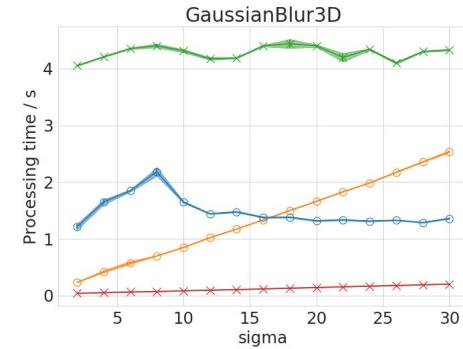
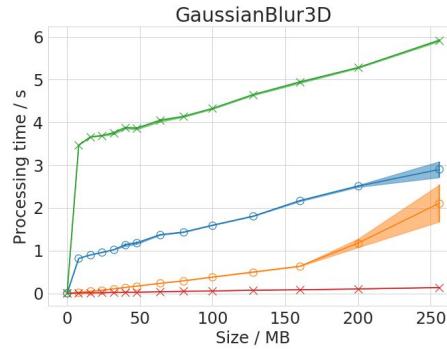


## Benchmarking



V.S.

- ✖ Workstation CPU
- ✗ Workstation GPU
- Laptop GPU
- Laptop CPU



# 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);

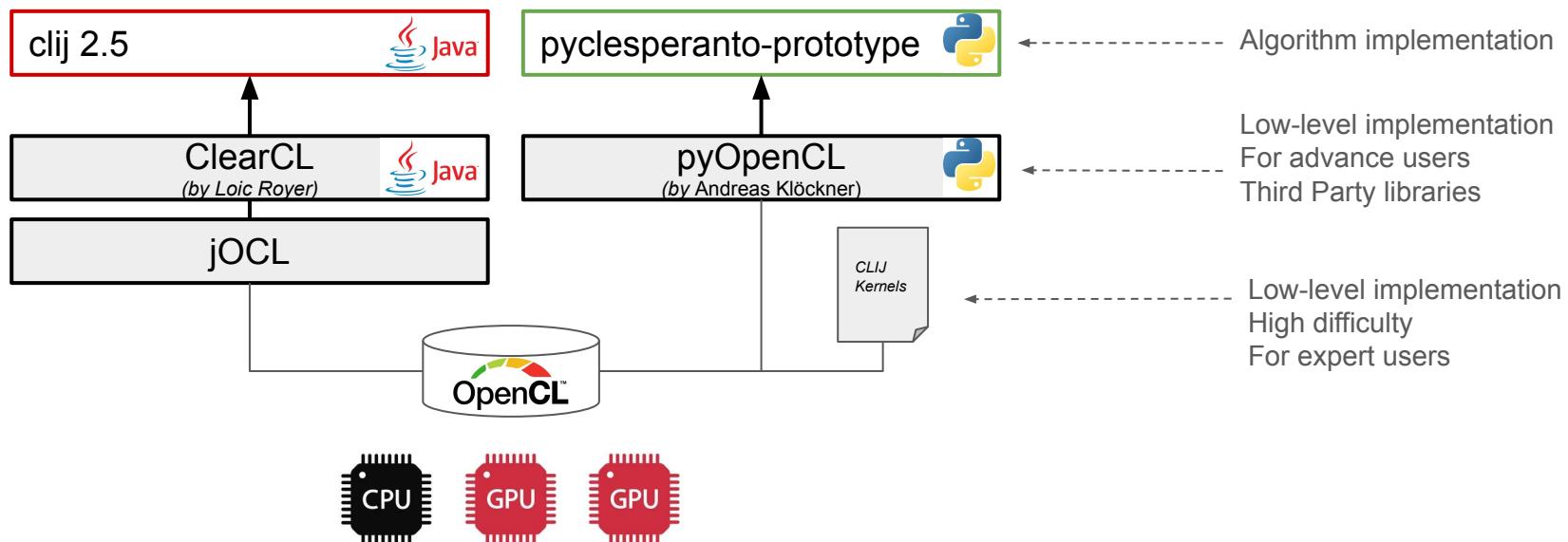
auto label = cle.create_like(binary);
cle.connect_components_labeling(binary, label);
```

With some  
limitations ...

# Clesperanto project

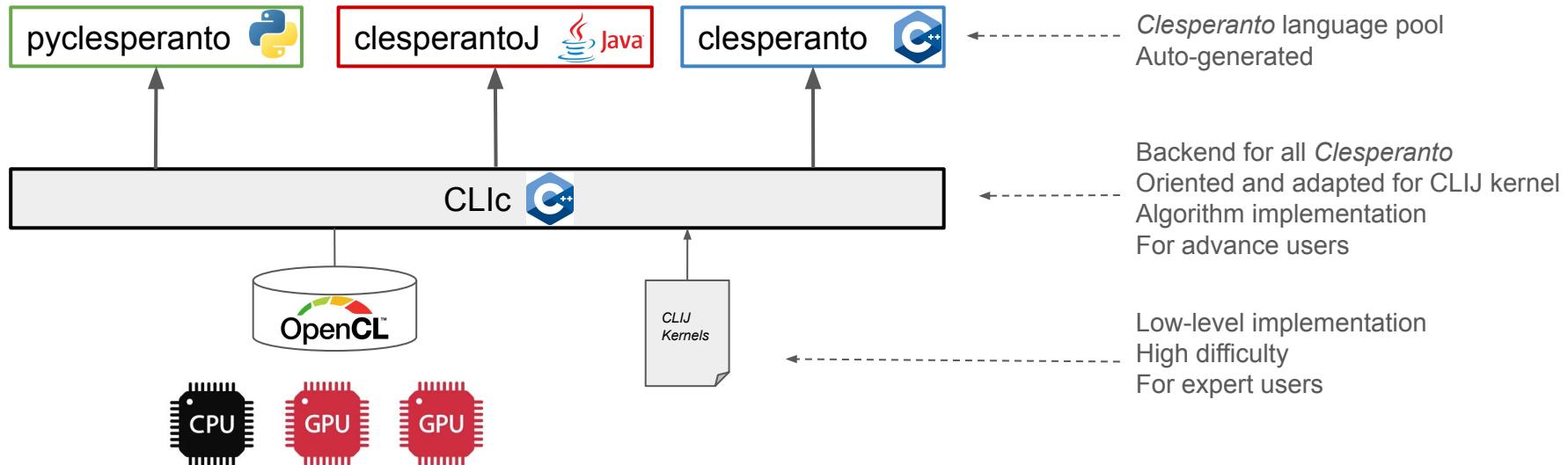
[clearcl](#) | [pyopencl](#) | [jocl](#)

## The Prototype



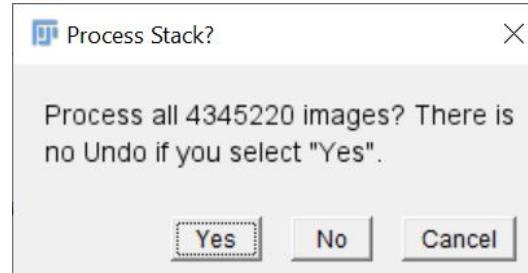
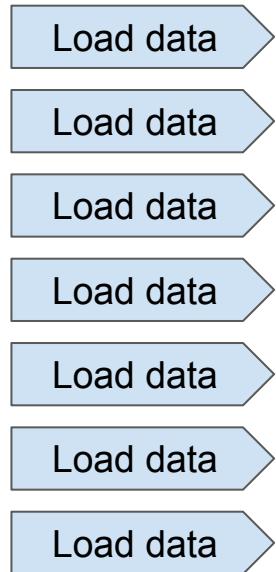
# Clesperanto project

## The Goal



# Memory management

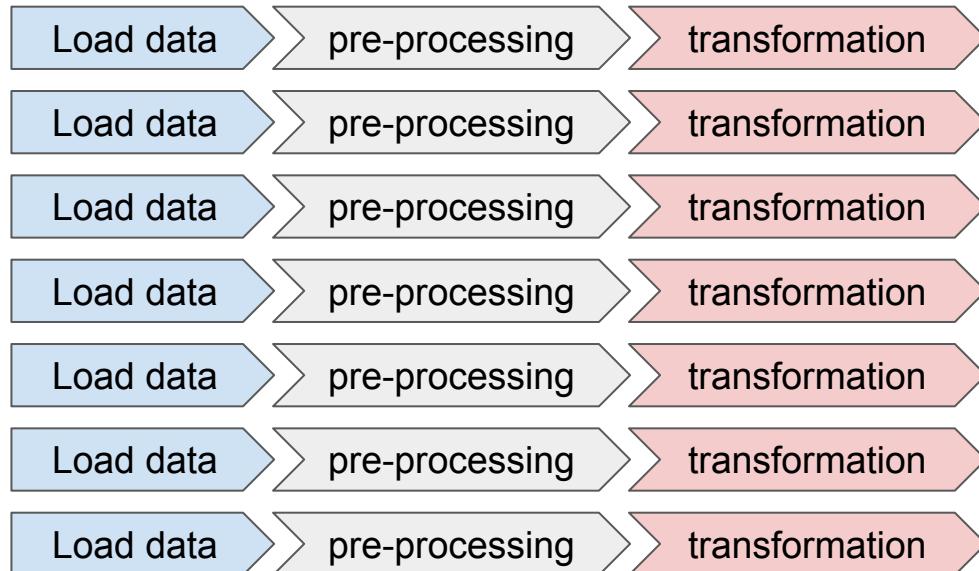
## Classical way of processing data



# Memory management

---

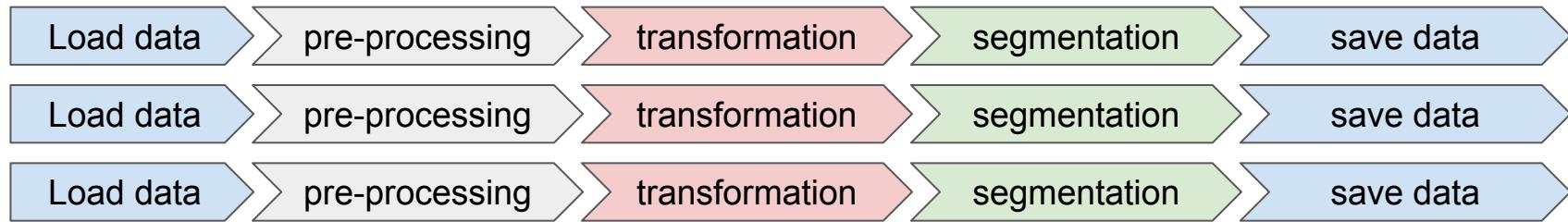
Classical way of processing data ... is suboptimal



# Memory management

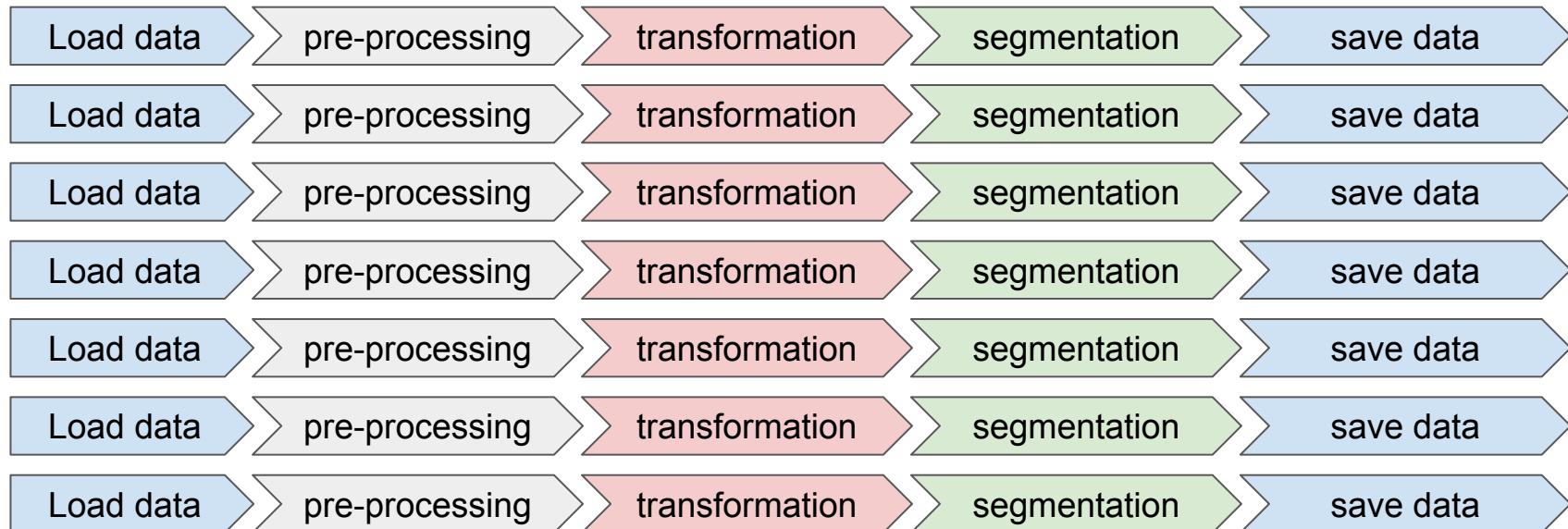
---

Process per data units ...



# Memory management

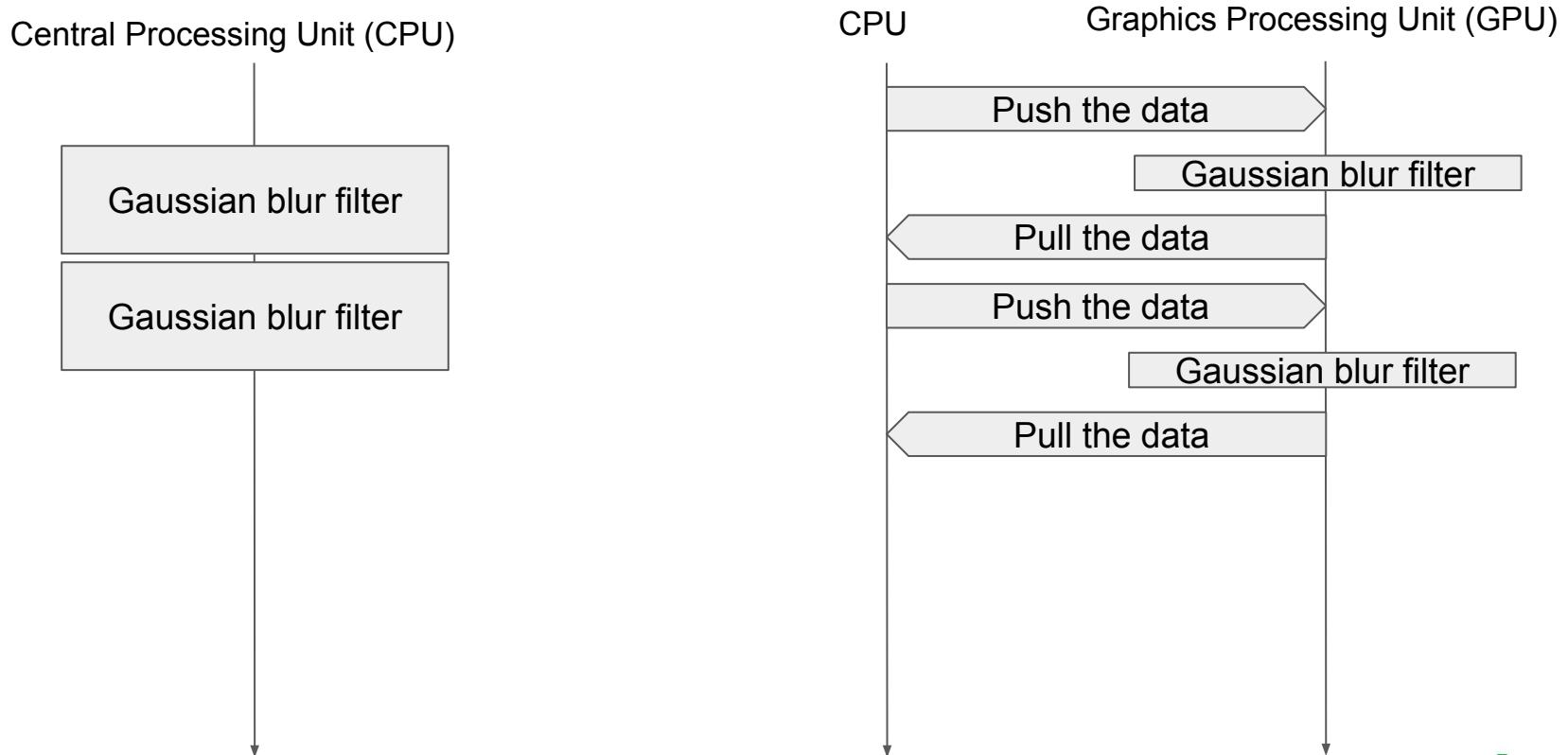
Process per data units ... and distribute it between systems



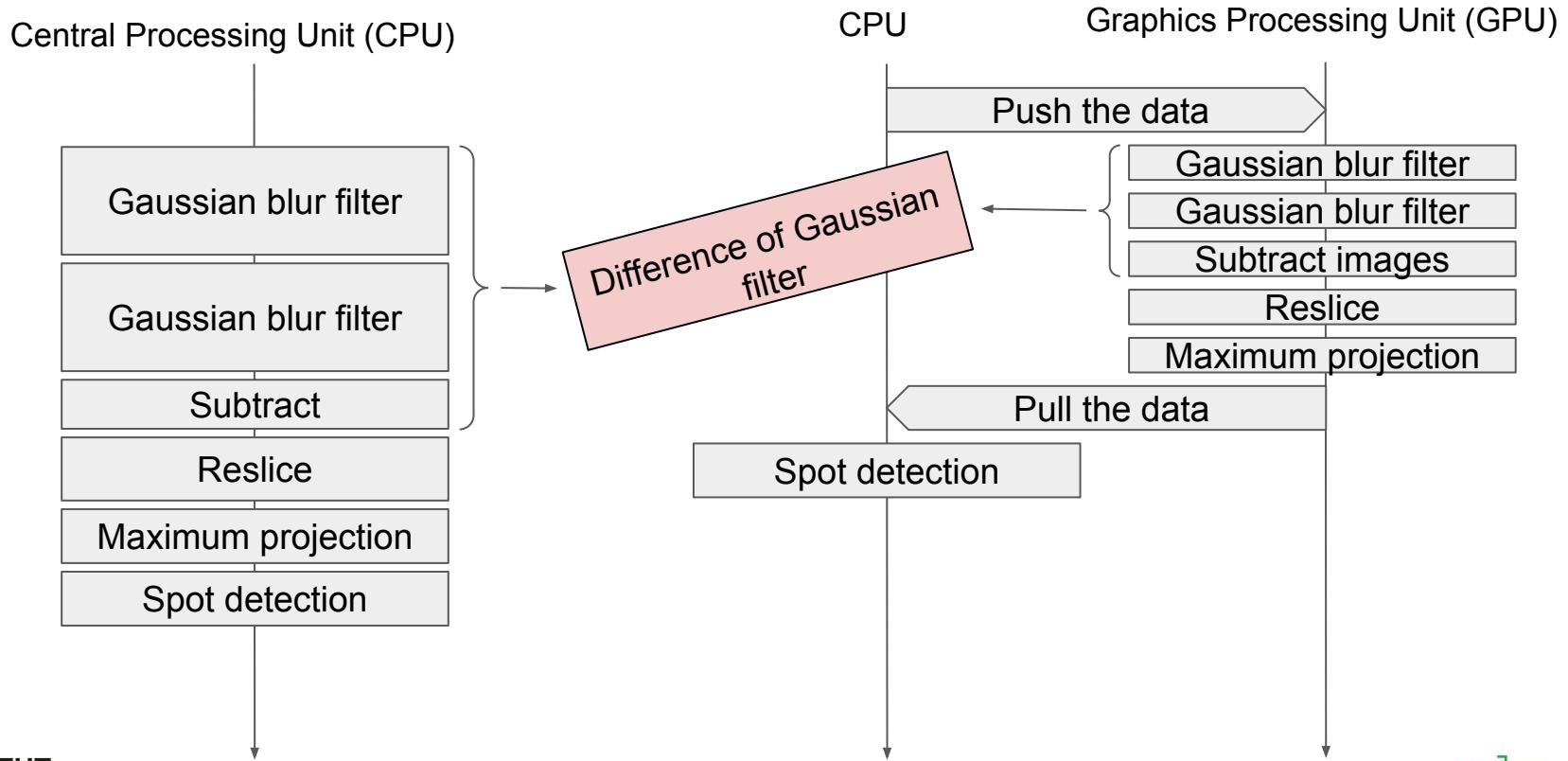
Would need to rely on a task distribution library



# Workflow construction

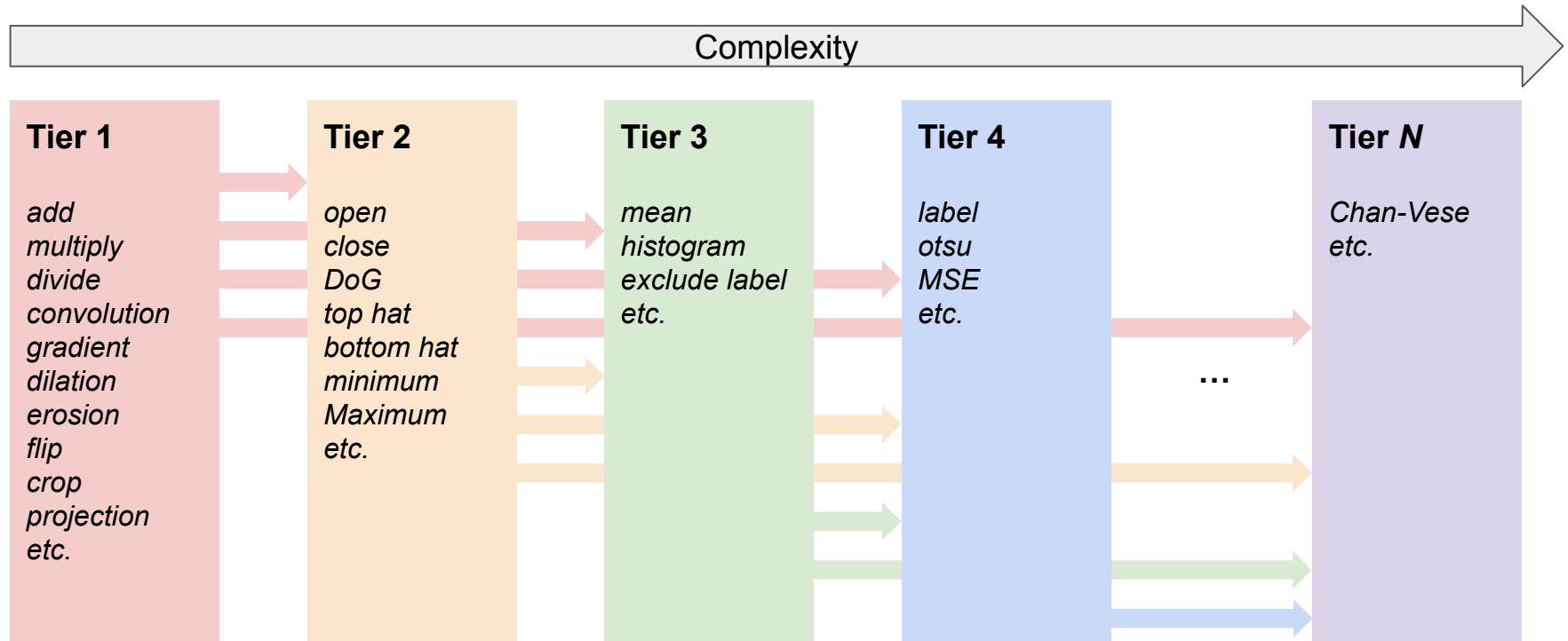


# Workflow construction



# Extensibility

Towards advance algorithm



# Extensibility

## Open-source community



```
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": from pyclesperanto_prototype._tier0 import Image,
        "src": from pyclesperanto_prototype._tier1 import gaussian_blur
    } from pyclesperanto_prototype._tier0 import create_like
    from pyclesperanto_prototype._tier0 import plugin_function
    execute from pyclesperanto_prototype._tier2 import subtract_images
    f1
    "...@plugin_function(categories=[\"filter\", \"background removal\"])"
    "max"
    "def difference_of_gaussian(
    dest source: Image,
    destination: Image = None,
    parameters: dict,
    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)
```



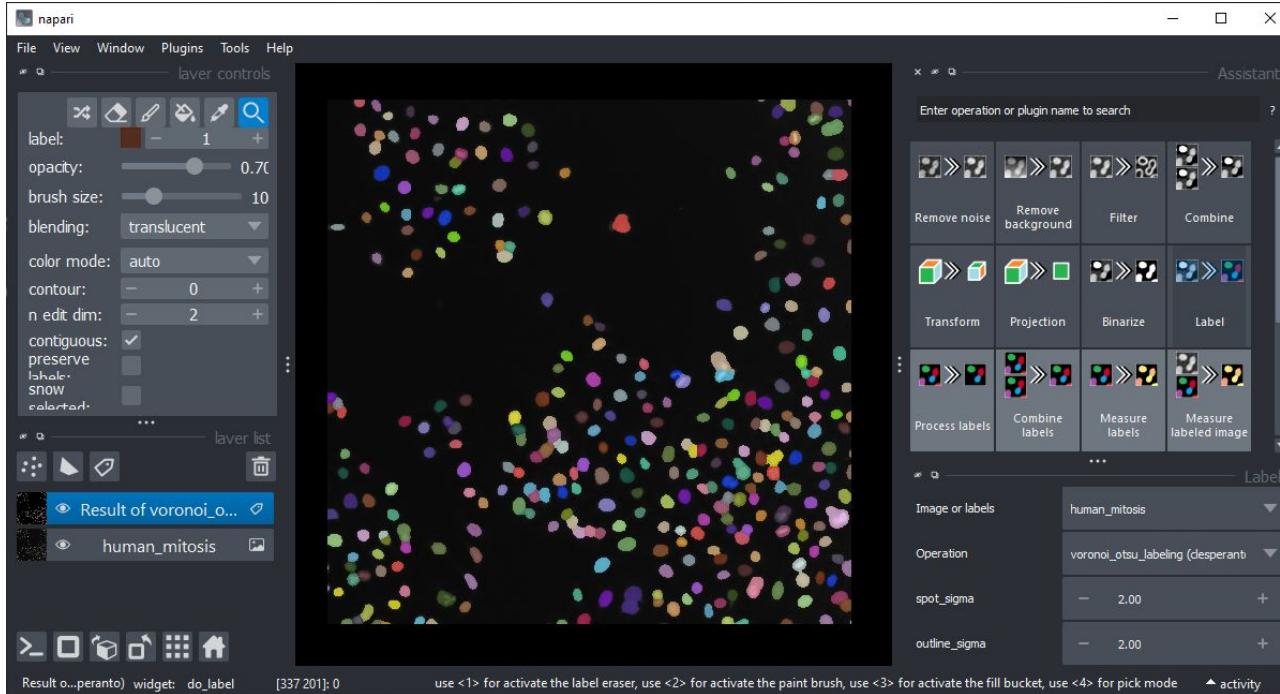
```
auto
maximum_z_projection_func(const Device::Pointer & device, const Array::Pointer & src, Array::Pointer dst
| -> Array::Pointer
{
    tier0::create_xy(src, dst);
    const KernelInfo kernel = { "maximum_z_projection", kernel::maximum_z_projection };
    const ParameterList params = { { "src", src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

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](https://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 Soltweibel



Brian Norton



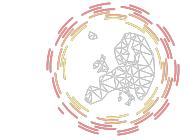
Pradeep Rajasekhar



Mathew McCormick



<https://image.sc>



[https://eubias.org/  
NEUBIAS](https://eubias.org/NEUBIAS)



Martin Jones



Jon Smith



Grahams Ross

## Rest of the World

### Funding:



FRANCE  
BIOIMAGING

CZ  
CHAN  
ZUCKERBERG  
INITIATIVE

# Exercises

---

<https://github.com/BiAPoL/PoL-Biolimage-Analysis-TS-GPU-Accelerated-Image-Analysis>

**Create an environment to work in:**

```
mamba create -n myenv python=3.9  
mamba activate myenv
```

**Install the package library**

```
mamba install -c conda-forge pyclesperanto-prototype
```

---

**(Alternative) Try the in-development version**

```
pip install pyclesperanto
```

*Danger (but fun) zone!*