SCIP: scalable cytometry image processing using Dask in a high performance computing environment

Software for distributed processing of bioimaging datasets
From images to biological insight?

Microscopy dataset of human blood cells

From a dataset of 250,000 images, each 12 channels

Generated in a matter of hours
Predict cell types from microscopy images of cells
Many steps are required to extract biological insight from raw microscopy data

Raw input data ➔ Transformation ➔ Pre-processing ➔ Quality control ➔ Insight

Segmentation
Masking
Filtering
Feature extraction

T-cell?
Neutrophil?
Monocyte?
Pre-processing software needs to scale with rapidly evolving imaging technology.

Local workstation execution with graphical interface.

Scale with **split-apply-combine** strategy or **vertically**.
Scalability has to be inherent to the pre-processing software

Focused on local workstation execution with GUI

Scale with split-apply-combine strategy or vertically

✔ Extensibility, interoperability, open-source

➕ Beyond split-apply-combine strategy

➕ Proper support for distributed computing
Scalable Cytometry Image Processing is a scalable, open-source preprocessing tool

- Executes all parts of preprocessing pipeline
- Embedded in the Python data science ecosystem
- Implemented on top of Dask, a framework for scalable computing with Python

https://github.com/ScalableCytometryImageProcessing/SCIP
SCIP's design allows more complex datasets to be pre-processed with more complex algorithms.
SCIP: scalable cytometry image processing

Scalability beyond split-apply-combine
Operations across large-scale datasets
Classifying human cells with SCIP output
SCIP: scalable cytometry image processing

**Scalability beyond split-apply-combine**
Operations across large-scale datasets
Classifying human cells with SCIP output
Modular pipeline steps make SCIP scalable and flexible

Steps are implemented with pure functions
- Output depends only on input and parameters
- Produce no side effects

Allows for steps to be
- interchangeable,
- chained together easily and
- executed independently.

Makes extensibility easier

API can be easily used in other programs
Out-of-core processing of large-scale datasets with lazy execution

Microscopy images can be very large, larger than memory. Spread reading from disk over entire execution. ⇒ Defer loading pixels to when they are needed.
Control over where steps can be executed is important for advanced pipelines.

- Image segmentation accelerated on the GPU
- Texture features computed on powerful CPUs
  - For example, Gray-level co-occurrence matrices
- Such steps have to be executed on specialized nodes
  - \( \Rightarrow \) Granular execution control
Dask is a framework for scaling up workflows with Python

- Enables all requirements to implement scalable bioimage pre-processing
- Scales from laptops to clusters
- Integrates seamlessly with other data science packages
- Easy to understand, but powerful
Dask DataFrame, Array and Bag are used throughout SCIP execution

DataFrame: features
Array: microscopy image planes
Bag: intermediate single-cell data

Provide map, fold, filter and aggregation functions

Make distribution logic transparent to the user

Task graphs are easily constructed using Dask collections

```python
images = Bag([im1.tiff, im2.tiff,...])
images = images.map(load_from_disk)
masked = images.map(mask)
features = images.map(extract)
```

df = features.compute()  →  Scheduler analyzes task graph and executes
Dask executes tasks using distributed workers orchestrated by scheduler

1. Set up cluster (local or distributed)
2. Connect client to cluster
3. Lazily define tasks in a task graph
4. Compute

Smart task scheduling uses computational resources as efficiently as possible

Fault tolerance makes SCIP more robust to hardware failure
Resource annotations allow steps to be computed on specialized hardware

Use heterogeneous resources as efficiently as possible

Scheduler sends tasks to appropriate workers

Other tasks continue on other nodes
SCIP: scalable cytometry image processing

Scalability beyond split-apply-combine

Operations across large-scale datasets

Classifying human cells with SCIP output
Image filtering prior to feature extraction requires reduction across dataset

Many cells are imaged, not all of interest
For example, dead cells

Solution: filtering prior to feature extraction
Discard cells with low signal

⇒ Requires reduction across dataset
SCIP: scalable cytometry image processing

Scalability beyond split-apply-combine
Operations across large-scale datasets
Classifying human cells with SCIP output
Overhead on runtime minimal from 100 000 images or more
Images per second approximately doubles when number of workers doubles
Processing a cytometry dataset of human immune system cells for classification

250,000 images of blood cells

12-channel image capturing different cell characteristics

Runtime: 101 min using 16 workers

<table>
<thead>
<tr>
<th>eccentricity</th>
<th>area</th>
<th>intensity</th>
<th>contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>200</td>
<td>30000</td>
<td>0.8</td>
</tr>
</tbody>
</table>

209 features / channel
SCIP features are used to predict cell type with machine learning

Using extreme gradient boosting

Balanced accuracy of 0.81 on test set

eccentricity | area | intensity | contrast
---|---|---|---
0.4 | 200 | 30000 | 0.8
Conclusion

- Tool for pre-processing large-scale bioimaging datasets
- Robust and inherently scalable
- Handles heterogeneous computational resources
- Enables implementation of dataset-wide computations
- Transform imaging data into machine learning-ready input
From images to biological insight?

Give SCIP a try!

https://github.com/ScalableCytometryImageProcessing/SCIP