PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Image Analysis with OMERO

Marcelo Leomil Zoccoler

Post-doc

Bio-Image Analysis Technology Development – Physics of Life – TU Dresden

From Paper to Pixels - Navigation Through Your Research Data Symposium

@zoccolermarcelo

May 27th 2024

General Image Analysis Workflow

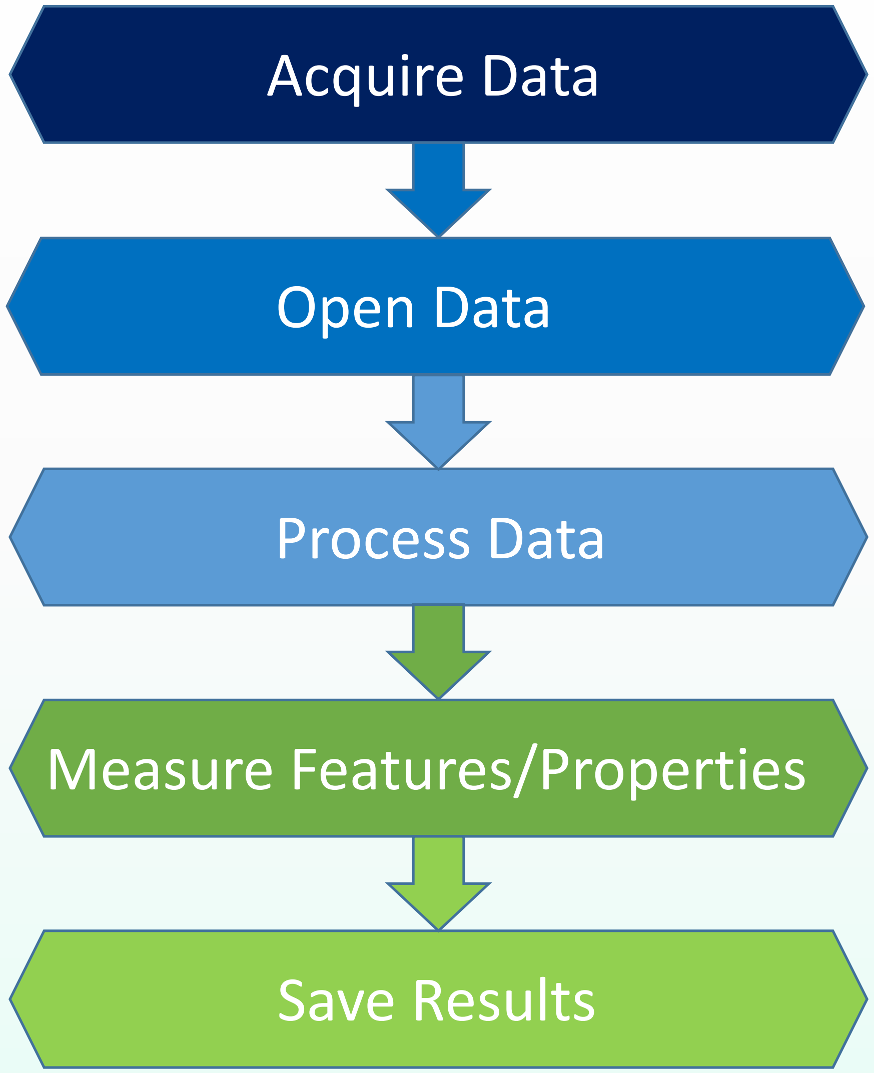
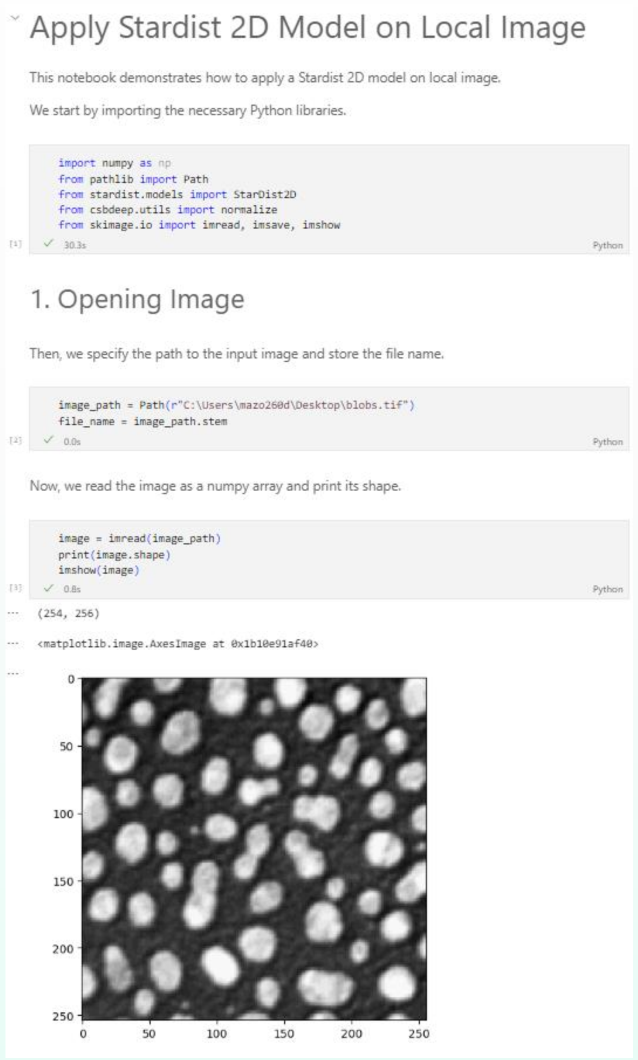


Image Analysis Workflow in Python

Example: Counting nuclei after segmentation with Stardist



Apply Stardist 2D Model on Local Image

This notebook demonstrates how to apply a Stardist 2D model on local image.

We start by importing the necessary Python libraries.

```
[1]
import numpy as np
from pathlib import Path
from stardist.models import StarDist2D
from csbdeep.utils import normalize
from skimage.io import imread, imsave, imshow
✓ 30.3s                                                Python
```

1. Opening Image

Then, we specify the path to the input image and store the file name.

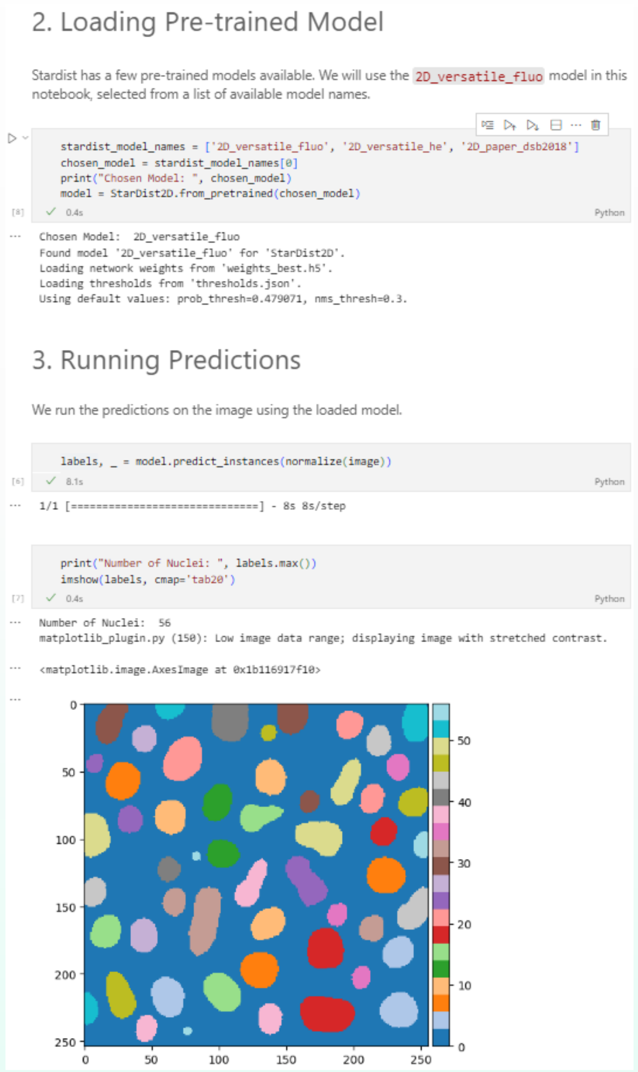
```
[2]
image_path = Path(r"C:\Users\mazo260d\Desktop\blobs.tif")
file_name = image_path.stem
✓ 0.0s                                                 Python
```

Now, we read the image as a numpy array and print its shape.

```
[3]
image = imread(image_path)
print(image.shape)
imshow(image)
✓ 0.8s                                                 Python
```

```
(254, 256)

<matplotlib.image.AxesImage at 0x1b10e91af40>
```



2. Loading Pre-trained Model

Stardist has a few pre-trained models available. We will use the `2D_versatile_fluo` model in this notebook, selected from a list of available model names.

```
[8]
stardist_model_names = ['2D_versatile_fluo', '2D_versatile_he', '2D_paper_dsb2018']
chosen_model = stardist_model_names[0]
print("Chosen Model: ", chosen_model)
model = StarDist2D.from_pretrained(chosen_model)
✓ 0.4s                                                 Python
```

```
Chosen Model:  2D_versatile_fluo
Found model '2D_versatile_fluo' for 'StarDist2D'.
Loading network weights from 'weights_best.h5'.
Loading thresholds from 'thresholds.json'.
Using default values: prob_thresh=0.479071, nms_thresh=0.3.
```

3. Running Predictions

We run the predictions on the image using the loaded model.

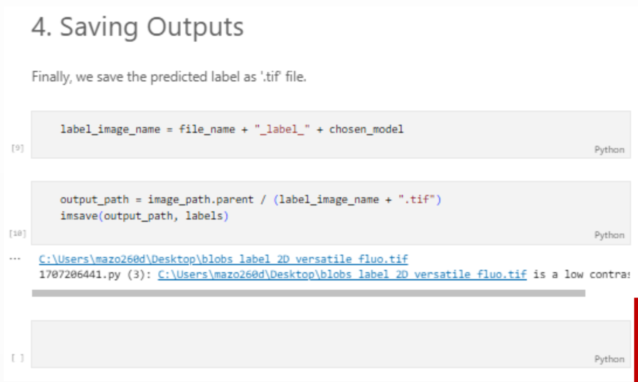
```
[6]
labels, _ = model.predict_instances(normalize(image))
✓ 8.1s                                                 Python
```

```
1/1 [==============================] - 8s 8s/step
```

```
[7]
print("Number of Nuclei: ", labels.max())
imshow(labels, cmap='tab20')
✓ 0.4s                                                 Python
```

```
Number of Nuclei:  56
matplotlib_plugin.py (150): Low image data range; displaying image with stretched contrast.

<matplotlib.image.AxesImage at 0x1b116917f10>
```



4. Saving Outputs

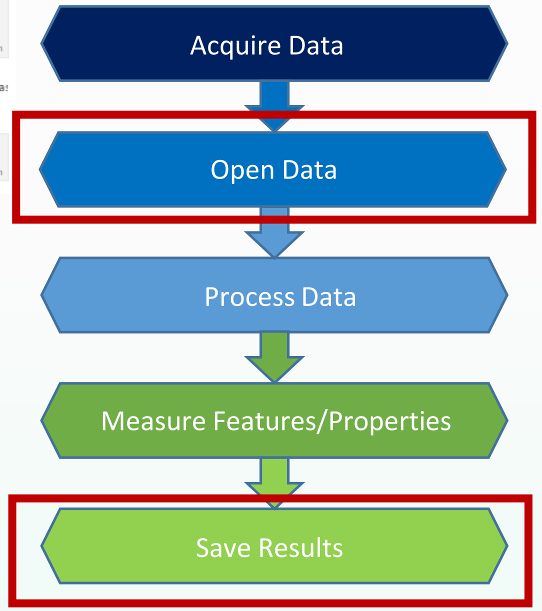
Finally, we save the predicted label as '.tif' file.

```
[9]
label_image_name = file_name + "_label_" + chosen_model
                                                       Python
```

```
[10]
output_path = image_path.parent / (label_image_name + ".tif")
imsave(output_path, labels)
                                                       Python
```

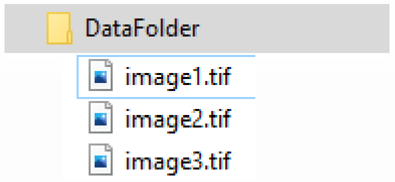
```
C:\Users\mazo260d\Desktop\blobs_label_2D_versatile_fluo.tif
1707206441.py (3): C:\Users\mazo260d\Desktop\blobs_label_2D_versatile_fluo.tif is a low contras
```

```
[ ]
                                                       Python
```

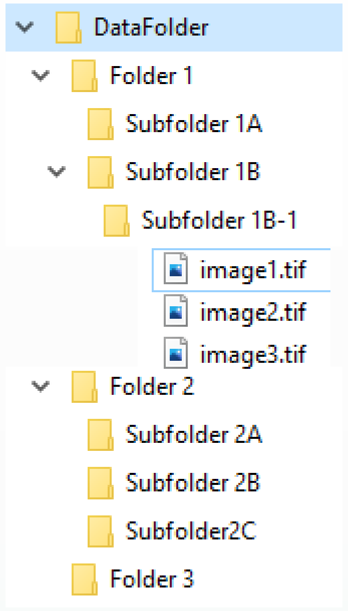


Neat! Let’s run it for another image!
Let me just plug my external HD…

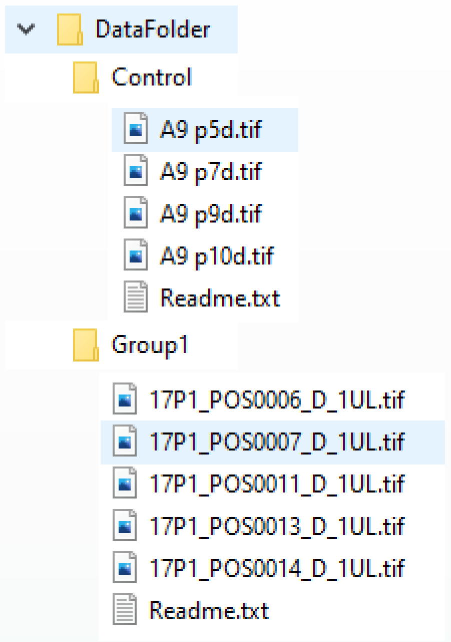
Data Structures



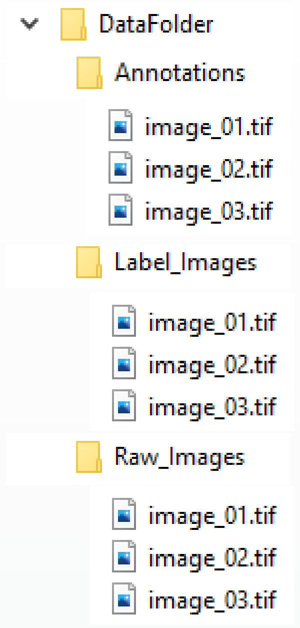
- DataFolder
  - image1.tif
  - image2.tif
  - image3.tif



- DataFolder
  - Folder 1
    - Subfolder 1A
    - Subfolder 1B
      - Subfolder 1B-1
        - image1.tif
        - image2.tif
        - image3.tif
  - Folder 2
    - Subfolder 2A
    - Subfolder 2B
    - Subfolder2C
  - Folder 3



- DataFolder
  - Control
    - A9 p5d.tif
    - A9 p7d.tif
    - A9 p9d.tif
    - A9 p10d.tif
    - Readme.txt
  - Group1
    - 17P1_POS0006_D_1UL.tif
    - 17P1_POS0007_D_1UL.tif
    - 17P1_POS0011_D_1UL.tif
    - 17P1_POS0013_D_1UL.tif
    - 17P1_POS0014_D_1UL.tif
    - Readme.txt



- DataFolder
  - Annotations
    - image_01.tif
    - image_02.tif
    - image_03.tif
  - Label_Images
    - image_01.tif
    - image_02.tif
    - image_03.tif
  - Raw_Images
    - image_01.tif
    - image_02.tif
    - image_03.tif

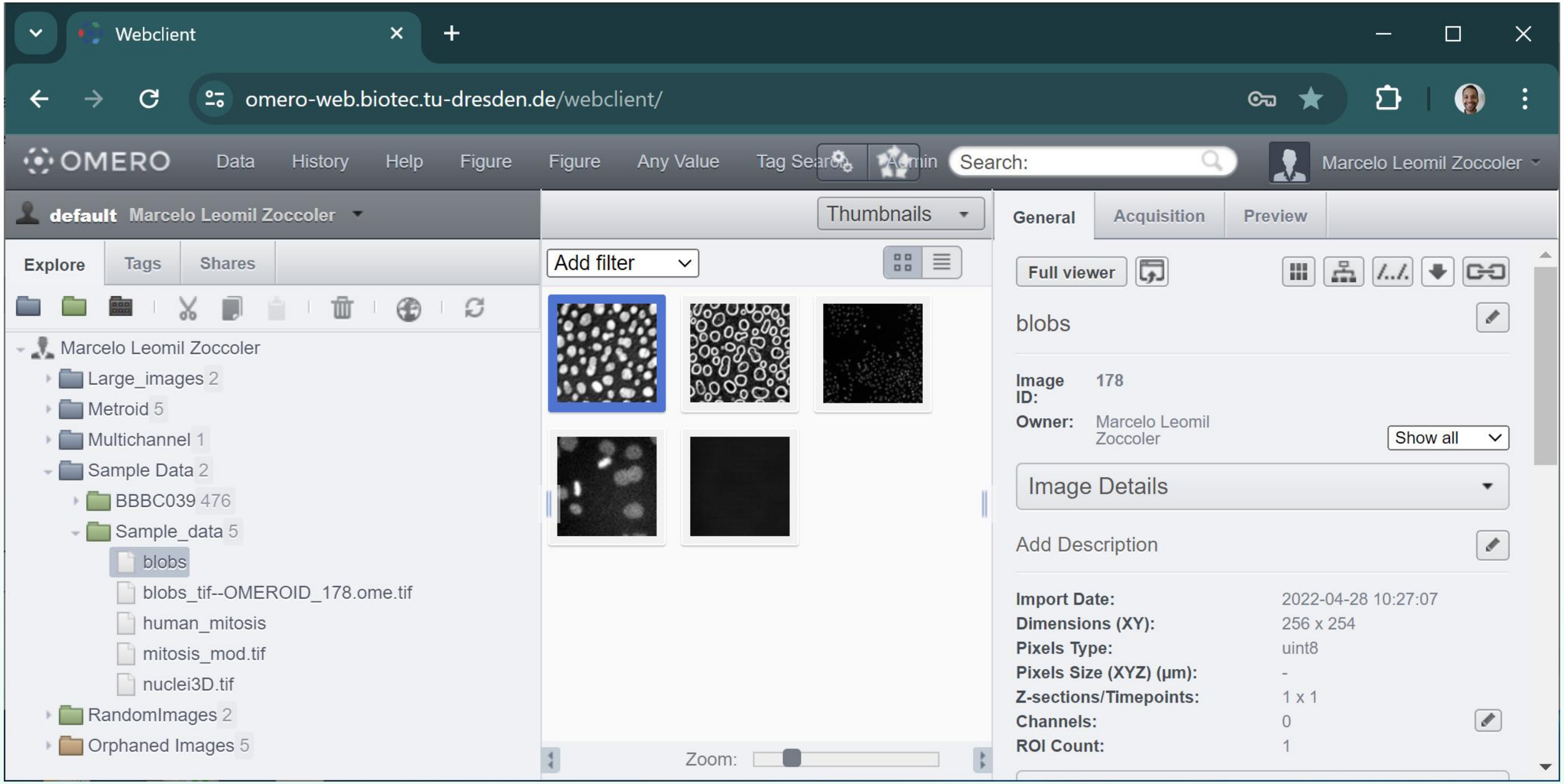
OMERO

Image by eikira from Pixabay (https://pixabay.com/photos/trunk-automobile-vehicle-luggage-1478832/)

@zoccolermarcelo

OMERO Server

PoL Physics of Life TU Dresden



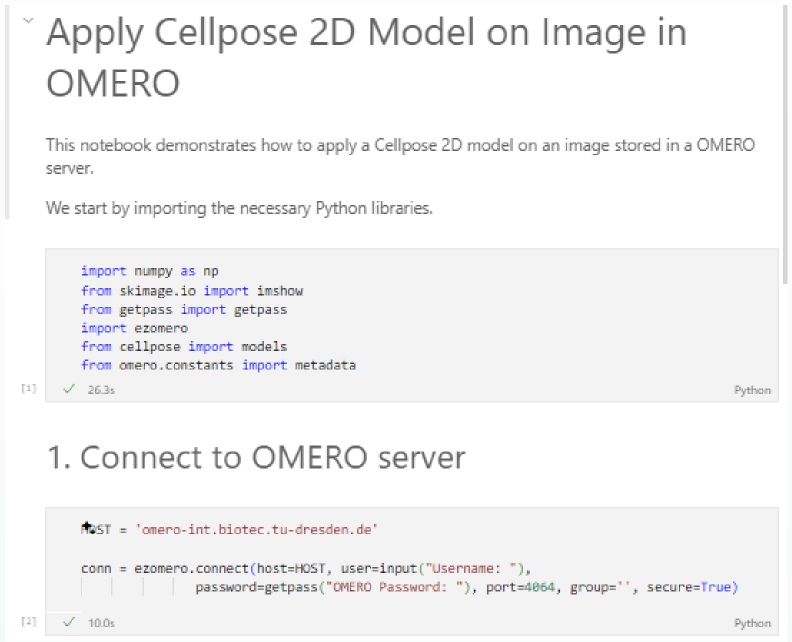
- Marcelo Leomil Zoccoler
  - Large_images 2
  - Metroid 5
  - Multichannel 1
  - Sample Data 2
    - BBBC039 476
    - Sample_data 5
      - blobs
      - blobs_tif--OMEROID_178.ome.tif
      - human_mitosis
      - mitosis_mod.tif
      - nuclei3D.tif
  - RandomImages 2
  - Orphaned Images 5

blobs

Image ID:	178
Owner:	Marcelo Leomil Zoccoler
Import Date:	2022-04-28 10:27:07
Dimensions (XY):	256 x 254
Pixels Type:	uint8
Pixels Size (XYZ) (µm):	-
Z-sections/Timepoints:	1 x 1
Channels:	0
ROI Count:	1

Image Analysis with OMERO

- Example: Counting nuclei after segmentation with Cellpose



Apply Cellpose 2D Model on Image in OMERO

This notebook demonstrates how to apply a Cellpose 2D model on an image stored in a OMERO server.

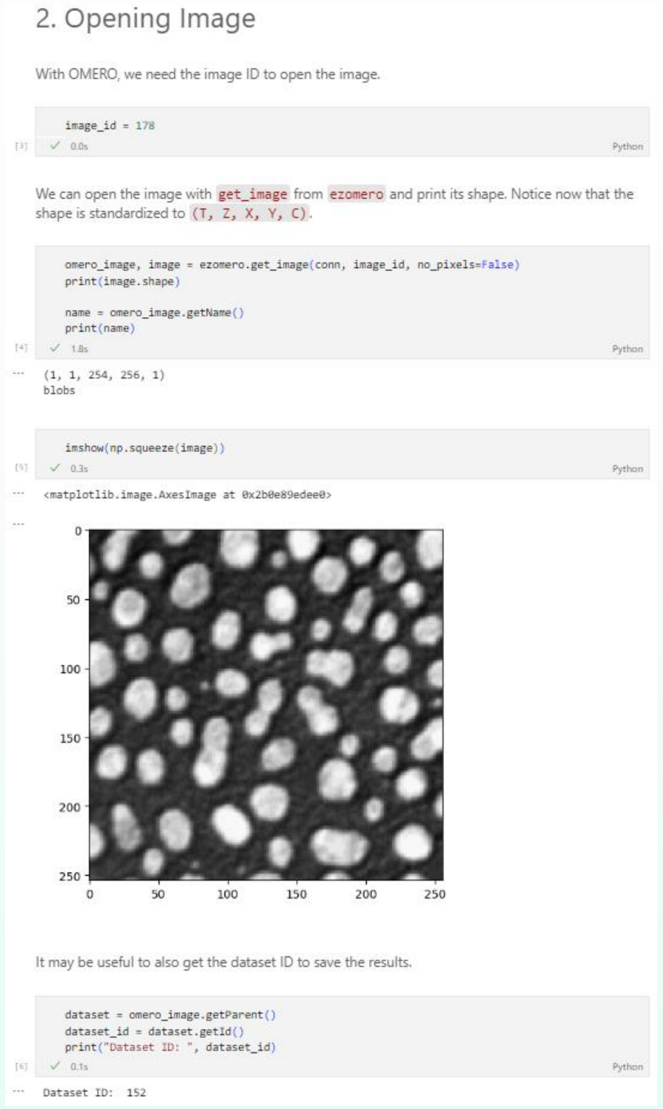
We start by importing the necessary Python libraries.

```python
import numpy as np
from skimage.io import imshow
from getpass import getpass
import ezomero
from cellpose import models
from omero.constants import metadata
```

1. Connect to OMERO server

```python
HOST = 'omero-int.biotec.tu-dresden.de'

conn = ezomero.connect(host=HOST, user=input("Username: "),
                       password=getpass("OMERO Password: "), port=4064, group='', secure=True)
```



2. Opening Image

With OMERO, we need the image ID to open the image.

```python
image_id = 178
```

We can open the image with `get_image` from `ezomero` and print its shape. Notice now that the shape is standardized to `(T, Z, X, Y, C)`.

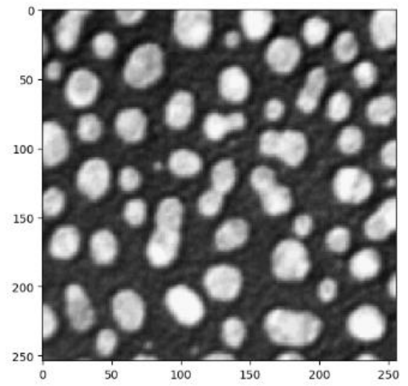
```python
omero_image, image = ezomero.get_image(conn, image_id, no_pixels=False)
print(image.shape)

name = omero_image.getName()
print(name)
```

```
(1, 1, 254, 256, 1)
blobs
```

```python
imshow(np.squeeze(image))
```

```
<matplotlib.image.AxesImage at 0x2b0e89edee0>
```



It may be useful to also get the dataset ID to save the results.

```python
dataset = omero_image.getParent()
dataset_id = dataset.getId()
print("Dataset ID: ", dataset_id)
```

```
Dataset ID:  152
```

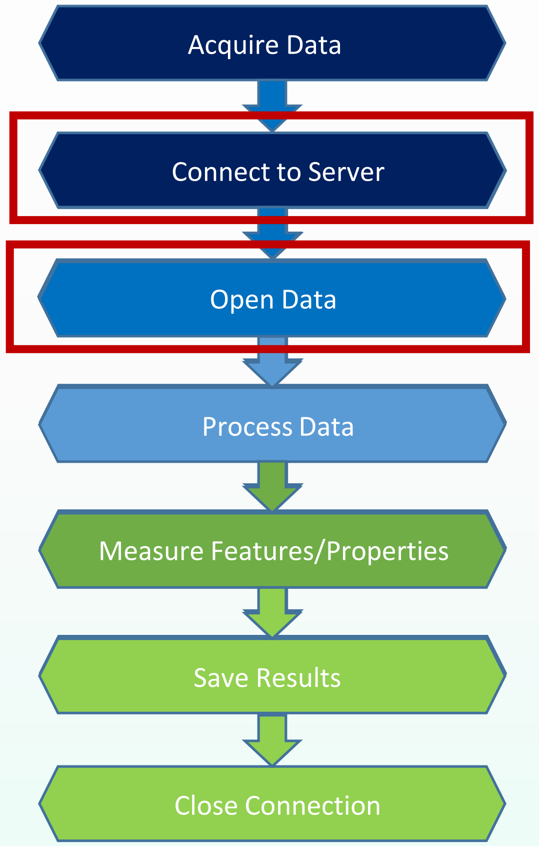
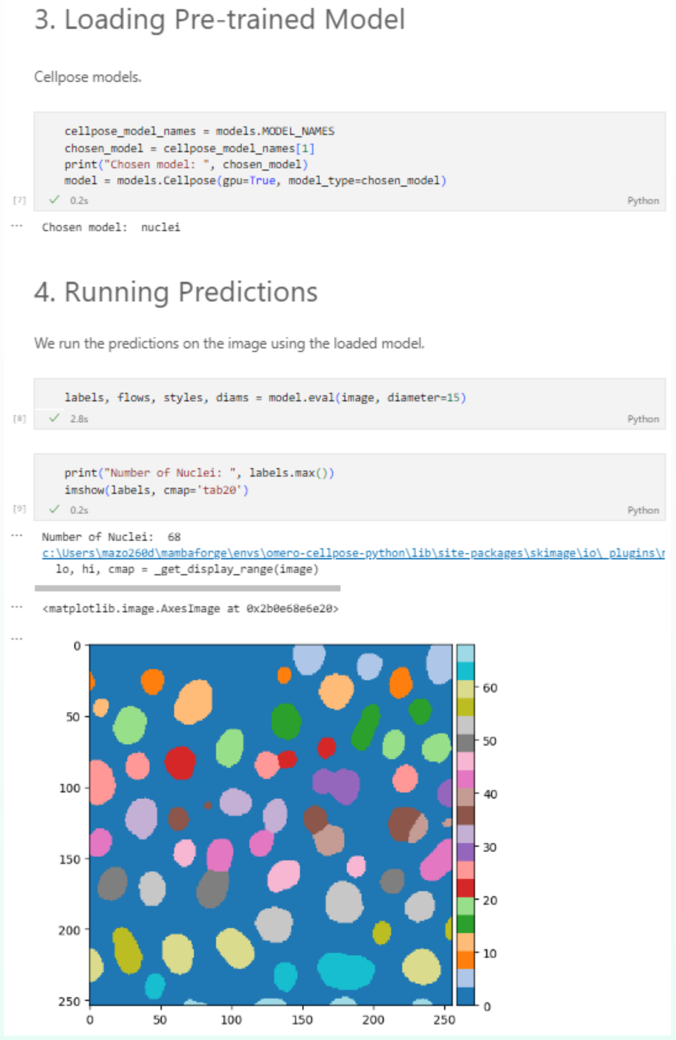


Image Analysis with OMERO

- Example: Counting nuclei after segmentation with Cellpose



3. Loading Pre-trained Model

Cellpose models.

```
cellpose_model_names = models.MODEL_NAMES
chosen_model = cellpose_model_names[1]
print("Chosen model: ", chosen_model)
model = models.Cellpose(gpu=True, model_type=chosen_model)
```

[7] ✓ 0.2s Python

```
Chosen model:  nuclei
```

4. Running Predictions

We run the predictions on the image using the loaded model.

```
labels, flows, styles, diams = model.eval(image, diameter=15)
```

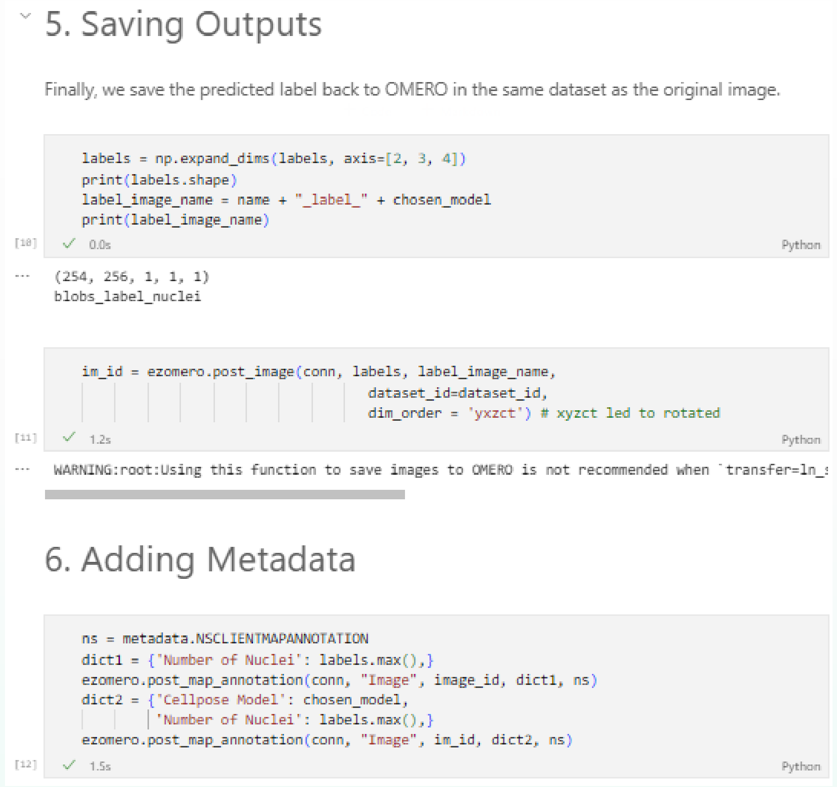
[8] ✓ 2.8s Python

```
print("Number of Nuclei: ", labels.max())
imshow(labels, cmap='tab20')
```

[9] ✓ 0.2s Python

```
Number of Nuclei:  68
c:\Users\mazo260d\mambaforge\envs\omero-cellpose-python\lib\site-packages\skimage\io\_plugins\r
  lo, hi, cmap = _get_display_range(image)
```

```
<matplotlib.image.AxesImage at 0x2b0e68e6e20>
```



5. Saving Outputs

Finally, we save the predicted label back to OMERO in the same dataset as the original image.

```
labels = np.expand_dims(labels, axis=[2, 3, 4])
print(labels.shape)
label_image_name = name + "_label_" + chosen_model
print(label_image_name)
```

[10] ✓ 0.0s Python

```
(254, 256, 1, 1, 1)
blobs_label_nuclei
```

```
im_id = ezomero.post_image(conn, labels, label_image_name,
                           dataset_id=dataset_id,
                           dim_order = 'yxzct') # xyzct led to rotated
```

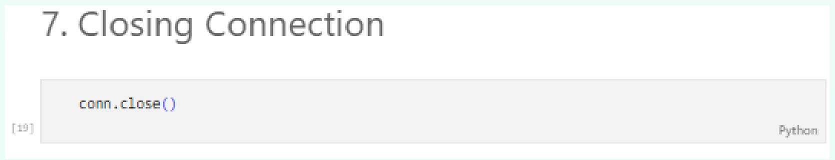
[11] ✓ 1.2s Python

```
WARNING:root:Using this function to save images to OMERO is not recommended when `transfer=ln_s
```

6. Adding Metadata

```
ns = metadata.NSCLIENTMAPANNOTATION
dict1 = {'Number of Nuclei': labels.max(),}
ezomero.post_map_annotation(conn, "Image", image_id, dict1, ns)
dict2 = {'Cellpose Model': chosen_model,
         'Number of Nuclei': labels.max(),}
ezomero.post_map_annotation(conn, "Image", im_id, dict2, ns)
```

[12] ✓ 1.5s Python



7. Closing Connection

```
conn.close()
```

[19] Python

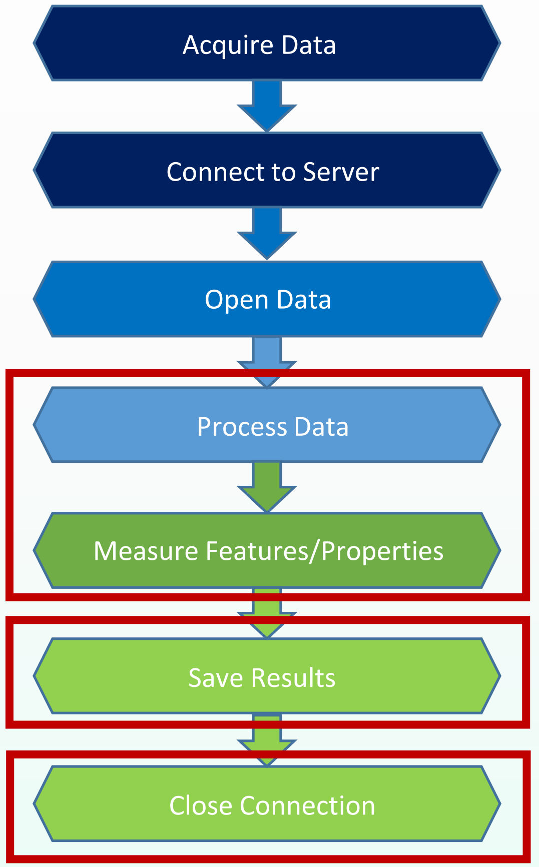
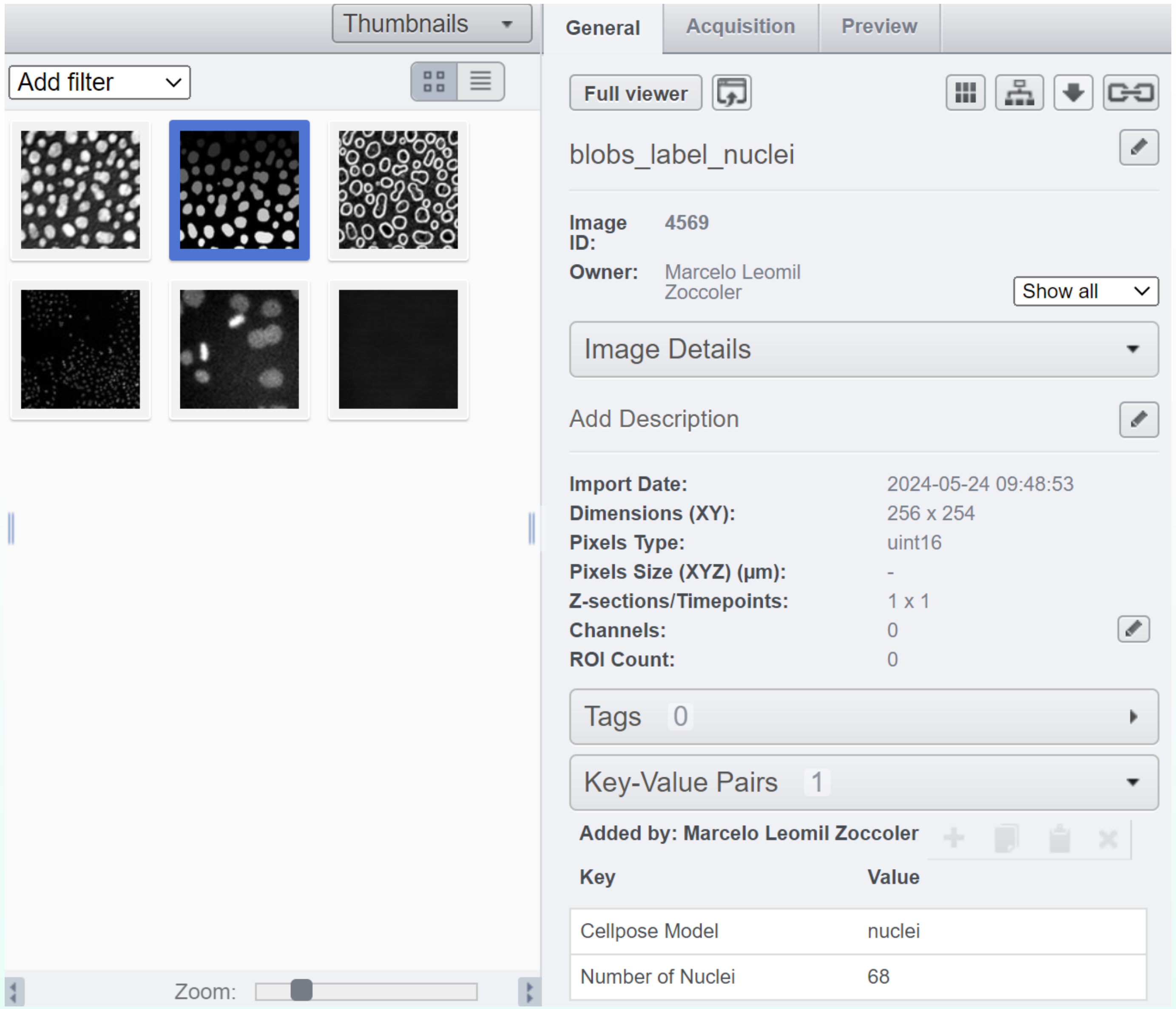


Image Analysis with OMERO



blobs_label_nuclei

Image ID:	4569
Owner:	Marcelo Leomil Zoccoler

Image Details

Add Description

Import Date:	2024-05-24 09:48:53
Dimensions (XY):	256 x 254
Pixels Type:	uint16
Pixels Size (XYZ) (µm):	-
Z-sections/Timepoints:	1 x 1
Channels:	0
ROI Count:	0

Tags 0

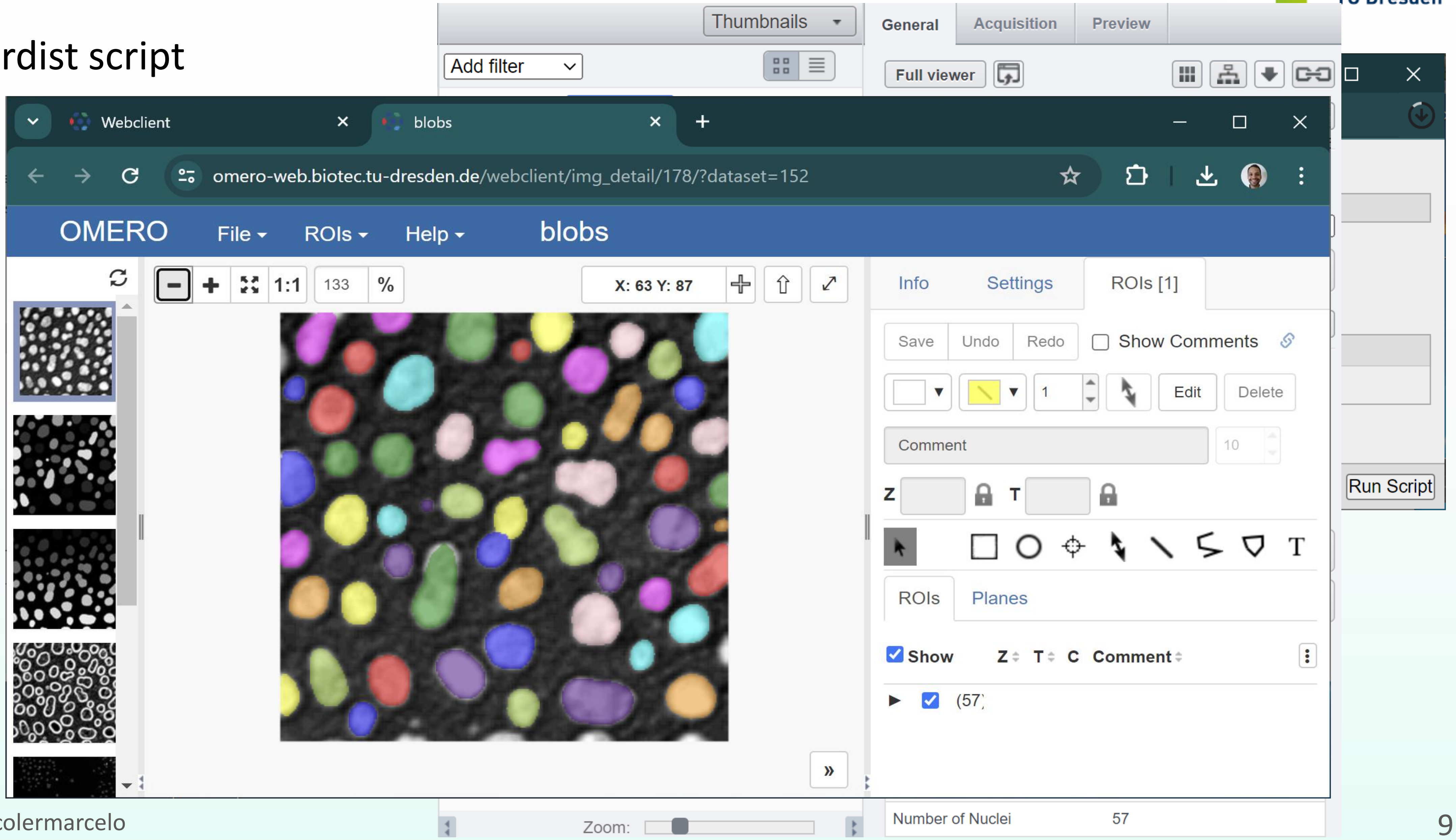
Key-Value Pairs 1

Added by: Marcelo Leomil Zoccoler

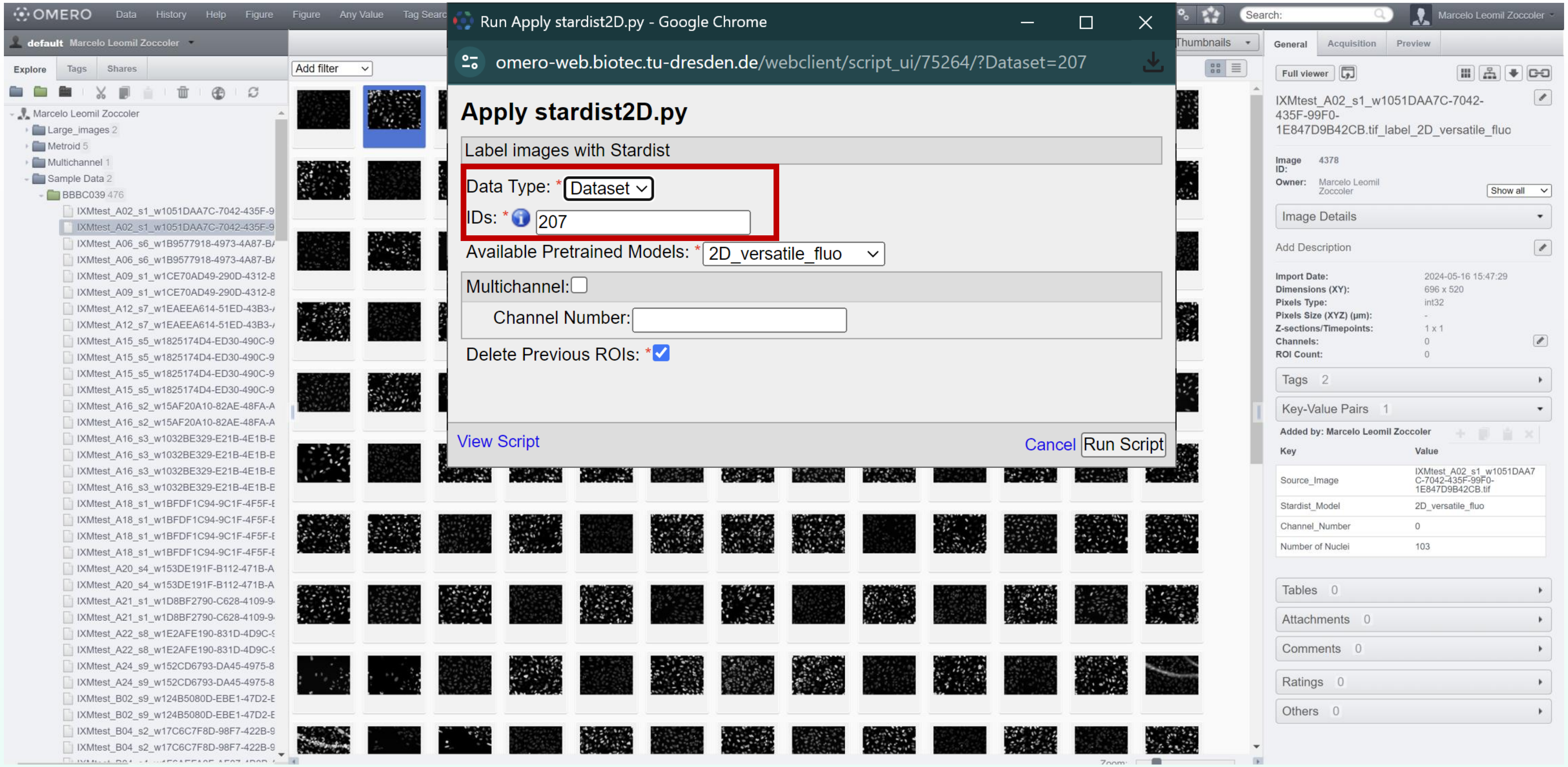
Key	Value
Cellpose Model	nuclei
Number of Nuclei	68

OMERO Scripts

- Stardist script



Batch Processing with OMERO



Marcelo Leomil Zoccoler
- Large_images 2
- Metroid 5
- Multichannel 1
- Sample Data 2
  - BBBC039 476
    - IXMtest_A02_s1_w1051DAA7C-7042-435F-9
    - IXMtest_A02_s1_w1051DAA7C-7042-435F-9
    - IXMtest_A06_s6_w1B9577918-4973-4A87-BA
    - IXMtest_A06_s6_w1B9577918-4973-4A87-BA
    - IXMtest_A09_s1_w1CE70AD49-290D-4312-8
    - IXMtest_A09_s1_w1CE70AD49-290D-4312-8
    - IXMtest_A12_s7_w1EAEEA614-51ED-43B3-A
    - IXMtest_A12_s7_w1EAEEA614-51ED-43B3-A
    - IXMtest_A15_s5_w1825174D4-ED30-490C-9
    - IXMtest_A15_s5_w1825174D4-ED30-490C-9
    - IXMtest_A15_s5_w1825174D4-ED30-490C-9
    - IXMtest_A15_s5_w1825174D4-ED30-490C-9
    - IXMtest_A16_s2_w15AF20A10-82AE-48FA-A
    - IXMtest_A16_s2_w15AF20A10-82AE-48FA-A
    - IXMtest_A16_s3_w1032BE329-E21B-4E1B-E
    - IXMtest_A16_s3_w1032BE329-E21B-4E1B-E
    - IXMtest_A16_s3_w1032BE329-E21B-4E1B-E
    - IXMtest_A16_s3_w1032BE329-E21B-4E1B-E
    - IXMtest_A18_s1_w1BFDF1C94-9C1F-4F5F-E
    - IXMtest_A18_s1_w1BFDF1C94-9C1F-4F5F-E
    - IXMtest_A18_s1_w1BFDF1C94-9C1F-4F5F-E
    - IXMtest_A18_s1_w1BFDF1C94-9C1F-4F5F-E
    - IXMtest_A20_s4_w153DE191F-B112-471B-A
    - IXMtest_A20_s4_w153DE191F-B112-471B-A
    - IXMtest_A21_s1_w1D8BF2790-C628-4109-9
    - IXMtest_A21_s1_w1D8BF2790-C628-4109-9
    - IXMtest_A22_s8_w1E2AFE190-831D-4D9C-9
    - IXMtest_A22_s8_w1E2AFE190-831D-4D9C-9
    - IXMtest_A24_s9_w152CD6793-DA45-4975-8
    - IXMtest_A24_s9_w152CD6793-DA45-4975-8
    - IXMtest_B02_s9_w124B5080D-EBE1-47D2-E
    - IXMtest_B02_s9_w124B5080D-EBE1-47D2-E
    - IXMtest_B04_s2_w17C6C7F8D-98F7-422B-9
    - IXMtest_B04_s2_w17C6C7F8D-98F7-422B-9

Run Apply stardist2D.py - Google Chrome

omero-web.biotec.tu-dresden.de/webclient/script_ui/75264/?Dataset=207

Apply stardist2D.py

Label images with Stardist

Data Type: * Dataset

IDs: * 207

Available Pretrained Models: * 2D_versatile_fluo

Multichannel: ☐

Channel Number:

Delete Previous ROIs: * ✓

View Script   Cancel   Run Script

IXMtest_A02_s1_w1051DAA7C-7042-435F-99F0-1E847D9B42CB.tif_label_2D_versatile_fluo

Image ID:	4378
Owner:	Marcelo Leomil Zoccoler

Show all

Image Details

Add Description

Import Date:	2024-05-16 15:47:29
Dimensions (XY):	696 x 520
Pixels Type:	int32
Pixels Size (XYZ) (µm):	-
Z-sections/Timepoints:	1 x 1
Channels:	0
ROI Count:	0

Tags 2

Key-Value Pairs 1

Added by: Marcelo Leomil Zoccoler

Key	Value
Source_Image	IXMtest_A02_s1_w1051DAA7C-7042-435F-99F0-1E847D9B42CB.tif
Stardist_Model	2D_versatile_fluo
Channel_Number	0
Number of Nuclei	103

Tables 0

Attachments 0

Comments 0

Ratings 0

Others 0

@zoccolermarcelo

Considerations

- Fixed 2-level hierarchy forces researchers to structure their data from start
- Extra levels of complexity could get achieved by adding tags
- Data, metadata and results from analysis can be all in the same place and linked (FAIR)
- OMERO scripts are very handy for light simple algorithms
- Complex algorithms that require lots of computational resources, like deep-learning, would need special servers and would be best loaded and processed in HPCs or powerful workstations
- Tiled processing for large images is not straight-forward (bandwidth limitations for writing whole large images)
- ezomero library eases a lot programming with OMERO compared to default Python bindings, but it does not have some functionalities yet (like write tiles)

Useful Links for Python Developers

- OMERO guide Python: https://github.com/ome/omero-guide-python
- OMERO Python Language Bindings: https://docs.openmicroscopy.org/omero/5.4.5/developers/Python.html
- ezomero: https://thejacksonlaboratory.github.io/ezomero/

Acknowledgements

- Cornelia Wetzker

BiAPoL team

- Marcelo Zoccoler
- Johannes Soltwedel
- Maleeha Hassan
- Stefan Hahmann

Former lab members:

- Robert Haase
- Allyson Ryan
- Till Korten
- Mara Lampert
- Svetlana Iarovenko
- Ryan George Savill
- Laura Zigutyte
- Somashekhar Kulkarni

CMCB TECHNOLOGY PLATFORM
Center for Molecular and Cellular Bioengineering TU Dresden

Networks

BioDIP Biopolis Dresden Imaging Platform

NFDI4 BIOIMAGE

image.sc

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Funding

DFG public

Chan Zuckerberg Initiative