

File Formats in Microscopy



(for SW Developers)

Dr. Sebastian Rhode
Software Architect AI Solutions
Product Center for Software



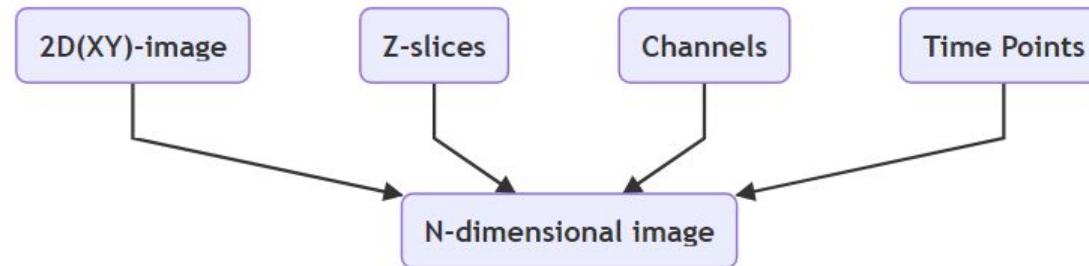
N-dimensional images



Apart from the X and Y dimensions, visible in the width and height of an image, image data can have additional dimensions.

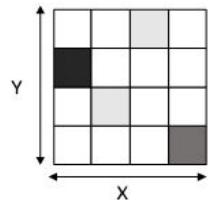
- Z dimension, providing depth to image data
- channels, showing data recorded with different detectors or detector settings
- time dimensions
- Well plate layout

Viewing multi-dimensional data can be challenging, because a computer monitor can only render a (multi-color) 2D representation.

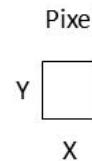


N-dimensional images

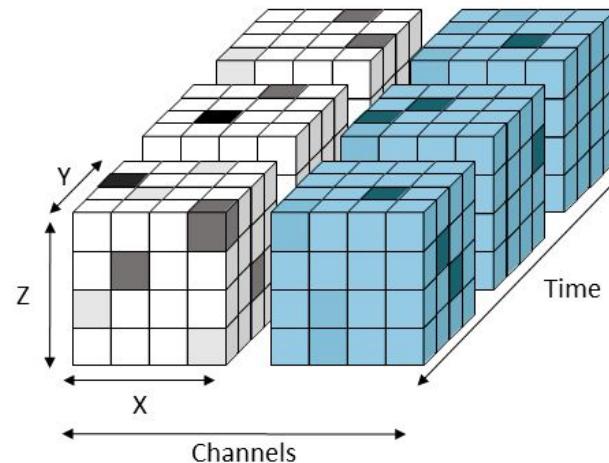
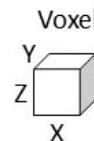
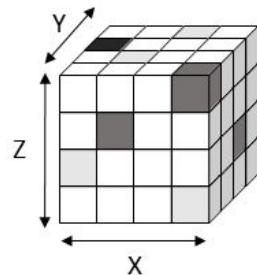
2D image



5D image



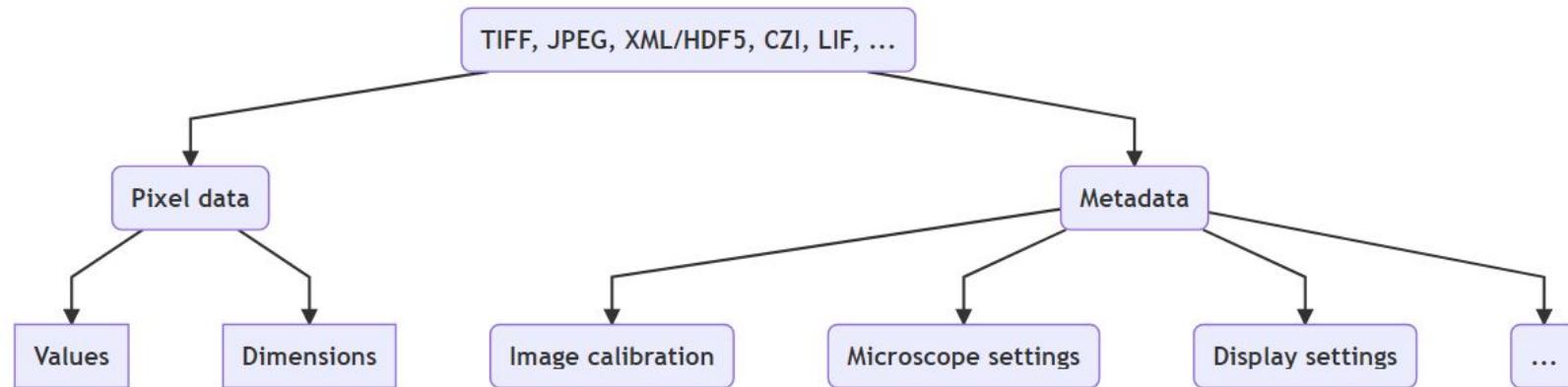
3D image



Schematic representation of 2D, 3D, and 5D image data. 2D images are made up of tiny squares called pixels, whereas 3D images are made up of cubes called voxels.

In general there are numerous ways how to save image data

- Virtually every microscope vendor has their own file format. It is thus very important to understand how to open those files and inspect their content.
- Some software will open only specific image file formats and thus it is sometimes necessary to re-save the data.
- During such image file format conversions information can be lost; it is important to be aware of this and avoid such information loss as much as possible.



File Formats – General remarks



- Image files consist of **pixel values** and **metadata**
- Some file formats are suitable for **data to analyze**, others for **data only to display**
- Metadata essential for analysis can be lost by saving in a **non-scientific file format**
- **Compression** can be either **lossless** or **lossy** – with different results
- Original pixel values can be lost through **lossy compression**, **conversion to RGB**, or by **removing dimensions**

File Formats – Working with image formats



- Always keep your original image files, in their original format
- During analysis:
 - Use software designed for scientific data (i.e. not a general photo editor)
 - If you need to save images, use the default file format for the software (e.g. TIFF for ImageJ)
 - **Check everything!** Make sure you can reopen the image after saving, and any pixel values, pixel size information and dimensions are retained exactly.
- Use general-purpose file formats (e.g. PNG, JPEG) for figures, presentations or websites – but not later analysis
- Use TIFF cautiously, because it can be used both as a scientific format suitable for analysis and a general-purpose format suitable only for display
 - OME-TIFF is often a better alternative to TIFF, since it standardizes how the metadata is stored

File Formats – Common formats in Microscopy Image Analysis



Format	Extensions	Main Use	Compression	Comment
TIFF	.tif, .tiff	Analysis, display (print)	None, lossless, lossy	Very general image format
OME-TIFF	.ome.tif, .ome.tiff	Analysis, Display (print)	None, lossless, lossy	TIFF, with standardized metadata for microscopy
OME-NGFF	ome.zarr	Analysis, Display	None, lossless, lossy	Emerging format, great for big datasets
PNG	.png	Display (web, print)	Lossless	Small(ish) file sizes without compression artefacts
JPEG	.jpg, .jpeg	Display (web)	Lossy (usually)	Small file sizes, but visible artefacts

File Formats – Pixel values and Compression

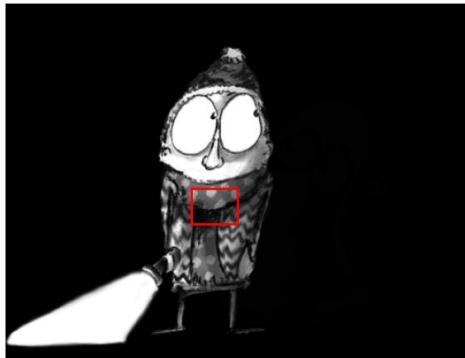


- **No compression.** The bytes representing the pixels are stored directly in the file. This means that the pixel values are preserved exactly, but the file can be quite large.
 - Examples: *TIFF (Uncompressed), ICS/IDS, CZI (uncompressed)*
- **Lossless compression.** The bytes representing the pixels are stored using a compression algorithm that (usually) results in less storage space being required, while making it possible to reconstruct the original values exactly by decompression. Compared to uncompressed data, the file size using lossless compression is generally smaller but reading or writing the file takes longer.
 - Examples: *TIFF (LZW compressed), PNG, BMP, JPEG2000 (lossless), ZSTD*
- **Lossy compression.** The bytes representing the pixels are stored using a compression algorithm that does *not* have to be able to reconstruct the original values exactly. This can result in dramatically smaller file sizes, but at a loss of information in the image – and often visual artefacts.
 - Examples: *TIFF (JPEG compressed), JPEG, JPEG-XR, GIF, JPEG2000 (lossy)*

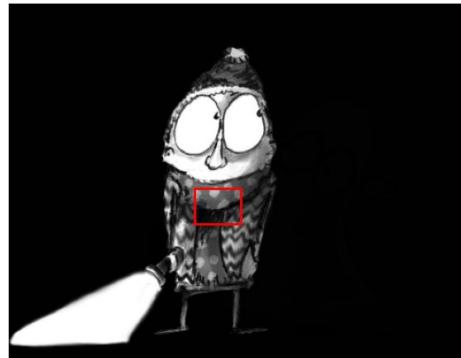
File Formats – Lossy compression is BAD for image Analysis



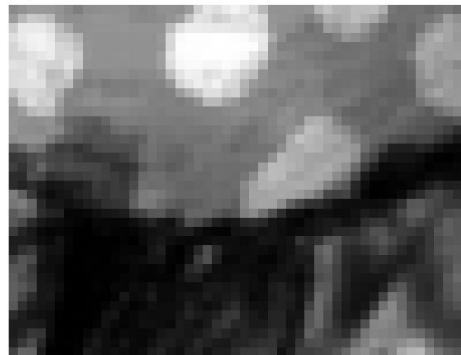
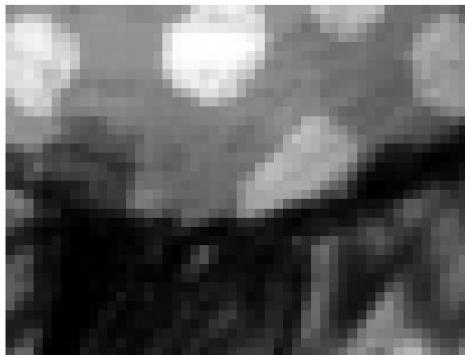
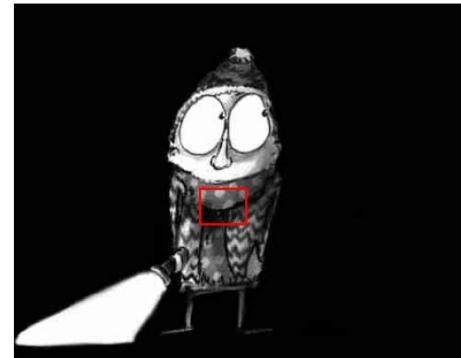
(A) TIFF uncompressed (189.7 KB)



(B) PNG compressed (25.9 KB)



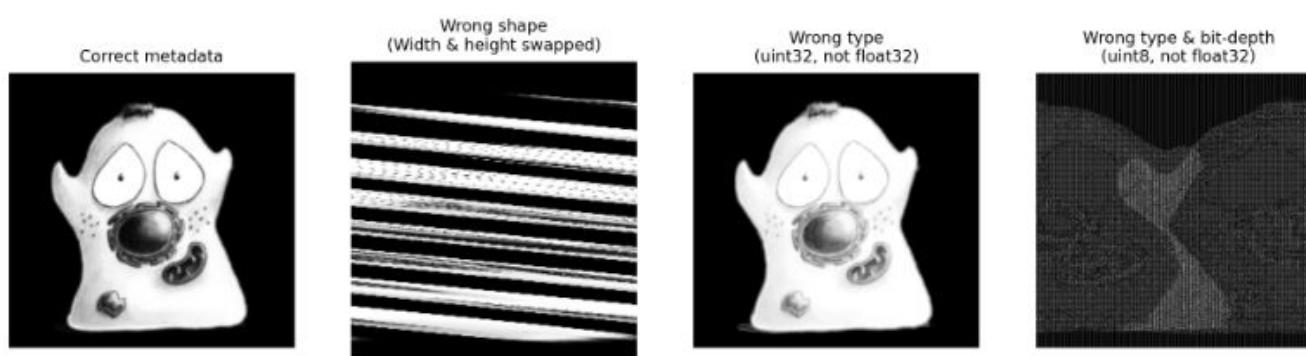
(C) JPEG compressed (5.5 KB)



File Formats – Core Metadata



- Pixel values are represented as a [stream of bits: ones and zeros](#).
- A few core pieces of metadata *must* be stored alongside these bits to interpret them as an image, such as the dimensions, bit-depth and type.
- If this information is missing – or wrong – then the image usually either cannot be read, or looks strange in some way



File Formats – Additional Metadata



- The most important piece of ‘non-core’ metadata is the pixel size. Unlike core metadata, it is *not* necessary for a file format to preserve the pixel size for an image to be opened and appear ‘correct’.
- This is a problem because if the pixel size is incorrect or missing, it remains possible to make length, area or volume measurements within an image – but these measurements are likely to be wrong.
- **In microscopy, the pixel size is typically represented as a value in $\mu\text{m}/\text{pixel}$. Saving in some file formats (e.g. JPEG and PNG) tends to lose pixel size information.**
- **Others (e.g. TIFF) *might* preserve the pixel size, *might* lose it, or *might* convert it to something else entirely.**

File Formats – Image Acquisition & Image Analysis



The general rule is: **Use the default file format for your microscope and acquisition software**

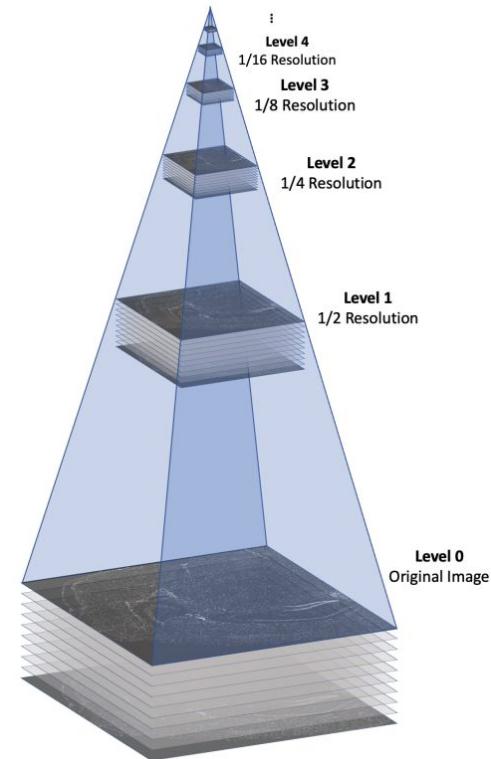
- The rationale is that the manufacturer's format ought to preserve all the information and metadata.
- Often, the acquisition software supports saving the data in some other way – but that can be risky. There's a strong chance that at least some metadata could be lost.
- Sometimes, the alternative export might do even worse things – such as convert the data to RGB.
- If in doubt export images as OME-TIFF for further analysis, but not as TIFF, PNG, JPEG or RGB images

Usually there is no need to export CZI in a different format because most Image Analysis & processing tools can read them directly!

File Formats – Pyramidal Images

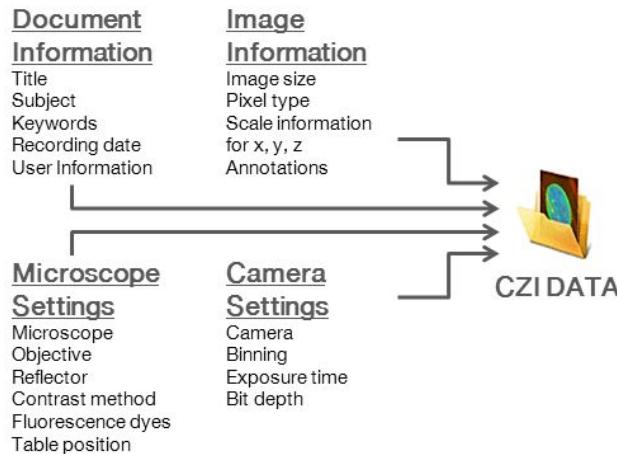


- Modern bioimages can be *huge*. It's not uncommon for a single image to contain hundreds of gigabytes of pixel data, making it infeasible to open the full image in one go.
- However, for visualization or analysis we often don't *need* all the pixels at once.
- For example, when viewing an image we only really need to access the pixels that are visible on screen at any moment – at the magnification at which they are being viewed.
- Pyramidal image files help overcome this by using two tricks:
 - They store the image as separate chunks (often called 'tiles', if they are 2D)
 - They store the same image at different resolutions in the same file



▪ <https://www.10xgenomics.com/resources/analysis-guides/performing-3d-nucleus-segmentation-with-cellpose-and-generating-a-feature-cell-matrix>

File Formats – Carl Zeiss Image (CZI)



Carl Zeiss Image Format:

<https://www.zeiss.com/microscopy/en/products/software/zeiss-zen/czi-image-file-format.html>

In addition to the image, the following metadata (and more) stored within the file:

- acquisition date, keywords, user comments
- microscope type and settings
- experiment settings
- dimensions types, scaling, pixel types
- used contrasts, objectives, filters
- stage (x,y) and focus (z) positions
- detector parameters such as exposure values, binning, bit depth, pixel dwell time
- automatic or user-defined graphic elements, annotations and display curves
- sample barcode information and label images
- analysis pipelines, results and objects, label images
- audit trail

- High Level Description: [CZI Image Format - High Level Description - Overview \(azure.com\)](#)
- CZI is the format for image storage
- The fundamental data storage schema is called “**Zeiss Integrated Software RAW**” data format
- Currently, CZI is the only ZISRAW based implementation
- For all the details please refer to: [DS_ZISRAW FileFormat - Overview \(azure.com\)](#)

For the CZI exist three main APIs allowing to read and write CZI images

- ZeissImgLib (.NET): to be published as Nuget package soon
- libCZI (C++): [ZEISS/libczi](#)
- pylibCZIrw: based on libCZI, available on PyPi as [pylibCZIrw](#) (PyPi) and internal repo: [RMS_PyPI_pylibCZIrw - Repos \(azure.com\)](#)
- <https://pypi.org/project/aicspylibczi/> - python package from Allen Cell Institute based on libCZI

Existing Readers:

- BioFormats:
<https://github.com/ome/bioformats/blob/develop/components/formats-gpl/src/loci/formats/in/ZeissCZIReader.java>
- Czifile: <https://github.com/cgohlke/czifile>

Around the the CZI exist other important tools useful for users and developers

- WBa.ImagingTool.exe (comes with binaries)
- CZICheck – Ceheck the CZI integrity
- CZICompress – CLI to compress CZI image data
- CZIShrink – UI for CZI Compress

File Formats – WBA.ImagingTool

Inspect subblocks and much more



WBA.ImagingTool V1.0.1

Image Files Refresh To CZI Tools Dump Create!

Thumbnail Preview Analyzer

Nr	SegmentType	Allocated	Used	Position	Next
1	FileHeader	512	512	00000000	00000220
2	SubBlockDirect	35.264	32.120	00000220	00008C00
3	AttachmentDir	768	650	00008C00	00008F20
4	Deleted	524.480	524.480	00008F20	00089000
5	SubBlock	1.069.120	1.069.094	00089000	0018E060
6	SubBlock	1.069.120	1.069.094	0018E060	002930C0
7	SubBlock	1.069.120	1.069.094	002930C0	00398120
8	SubBlock	535.712	535.685	00398120	0041ADE0
9	SubBlock	2.097.472	2.097.449	0041ADE0	0061AF40
10	SubBlock	535.712	535.685	0061AF40	0069D000
11	SubBlock	2.097.472	2.097.449	0069D000	0089D060
12	SubBlock	535.712	535.685	0089D060	00920A20
13	SubBlock	2.097.472	2.097.449	00920A20	00B20B80
14	SubBlock	292.640	292.637	00B20B80	00B602C0
15	SubBlock	1.216.928	1.216.901	00B602C0	00C91480
16	SubBlock	292.640	292.637	00C91480	00CD8BC0
17	SubBlock	1.216.928	1.216.901	00CD8BC0	00E01080
18	SubBlock	292.640	292.637	00E01080	00E49AC0
19	SubBlock	1.216.928	1.216.901	00E49AC0	00F72680
20	SubBlock	2.097.472	2.097.449	00F72680	011727E0
21	SubBlock	2.097.472	2.097.449	011727E0	01372340
22	SubBlock	2.097.472	2.097.449	01372340	01572AA0
23	SubBlock	2.097.472	2.097.449	01572AA0	01772C00
24	SubBlock	2.097.472	2.097.449	01772C00	01972D60
25	SubBlock	2.097.472	2.097.449	01972D60	01B72E00
26	SubBlock	203.104	203.085	01B72E00	01B8A480
27	SubBlock	2.097.472	2.097.449	01B8A480	01DA49A0
28	SubBlock	203.104	203.085	01DA49A0	01D06320
29	SubBlock	2.097.472	2.097.449	01D06320	01FD6480
30	SubBlock	203.104	203.085	01FD6480	02007E00
31	SubBlock	2.097.472	2.097.449	02007E00	02207F60
32	SubBlock	1.005.888	1.005.865	02207F60	022FD8C0
33	SubBlock	2.097.472	2.097.449	022FD8C0	024FDA20
34	SubBlock	1.005.888	1.005.865	024FDA20	025F3380
35	SubBlock	2.097.472	2.097.449	025F3380	027F34E0
36	SubBlock	1.005.888	1.005.865	027F34E0	0288E4A0
37	SubBlock	2.097.472	2.097.449	0288E4A0	02AE8FA0
38	SubBlock	1.005.888	1.005.865	02AE8FA0	028DE900
39	SubBlock	2.097.472	2.097.449	028DE900	02DDEA60
40	SubBlock	1.005.888	1.005.865	02DDEA60	02ED43C0
41	SubBlock	2.097.472	2.097.449	02ED43C0	030D4520
42	SubBlock	1.005.888	1.005.865	030D4520	031C9E80
43	SubBlock	2.097.472	2.097.449	031C9E80	033C9F60
44	SubBlock	97.568	97.551	033C9F60	033E1D20
45	SubBlock	203.104	203.085	033E1D20	034136A0
46	SubBlock	1.005.888	1.005.865	034136A0	03509000

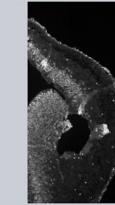
SegmentType SubBlock
MetadataSize 97
AttachmentSize 71.716
DataSize 1.144.832

Directory
Compression Uncompressed
FilePosition 11961024
FilePart 0
PixelType Gray16
PyramideType MultiSubblock

Dir	Start	Size	StSize	1/Skale	Scale	StartC
X	8.192	4.472	559	8.00	0.125	0.0
Y	0	8.192	1.024	8.00	0.125	0.0
C	0	1	1	1.00	1	0.0
S	0	1	1	1.00	1	0.0
B	0	1	1	1.00	1	0.0

Metadata
<METADATA>
<AttachmentSchema>
<DataFormat>CHUNKCONTAINER</DataFormat>
</AttachmentSchema>
</METADATA>

Image Mask



File Formats – CZICompress

Compress CZI Images Files from the commandline

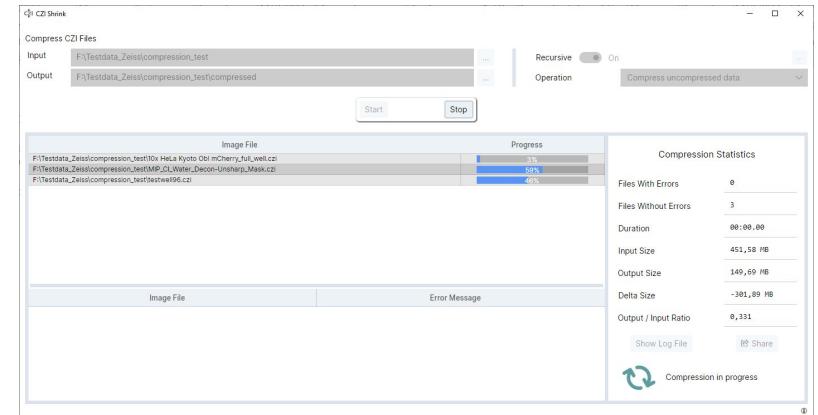
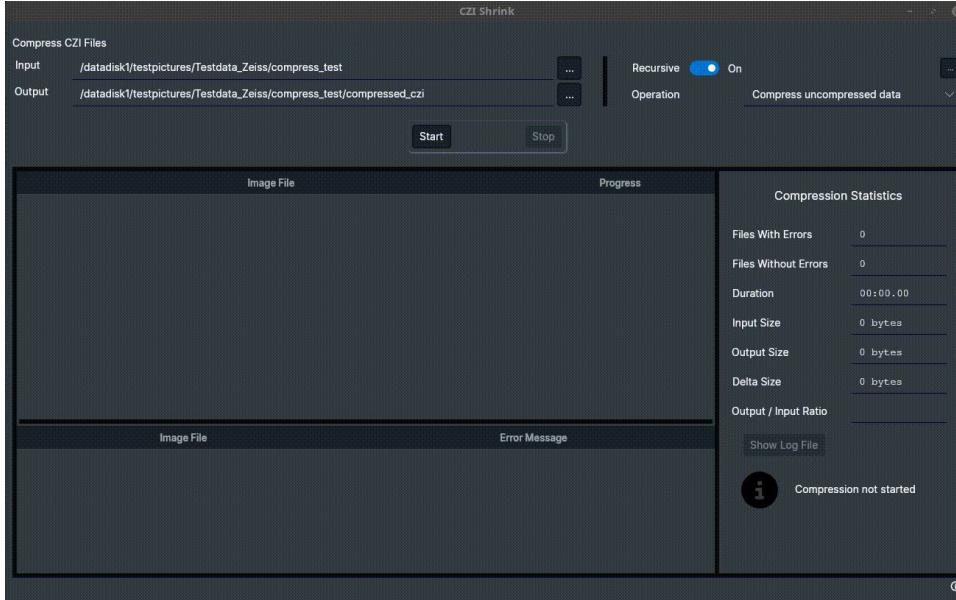
A screenshot of a terminal window titled "Tilix: sebi06@sebastian-tuxedo: /datadisk1/testpictures/Testdata_Zeiss/compress_test". The terminal shows the command "(base) sebi06@sebastian-tuxedo:/datadisk1/testpictures/Testdata_Zeiss/compress_test\$". The main area of the terminal is dark gray with a subtle grid pattern.

<https://github.com/ZEISS/czicompress>

- compress or decompress a single CZI file
- versatile
- scriptable
- run in headless/server environments
- run in cron jobs
- cross-platform (focus on linux-x64 and win-x64)

File Formats – CZIShrink

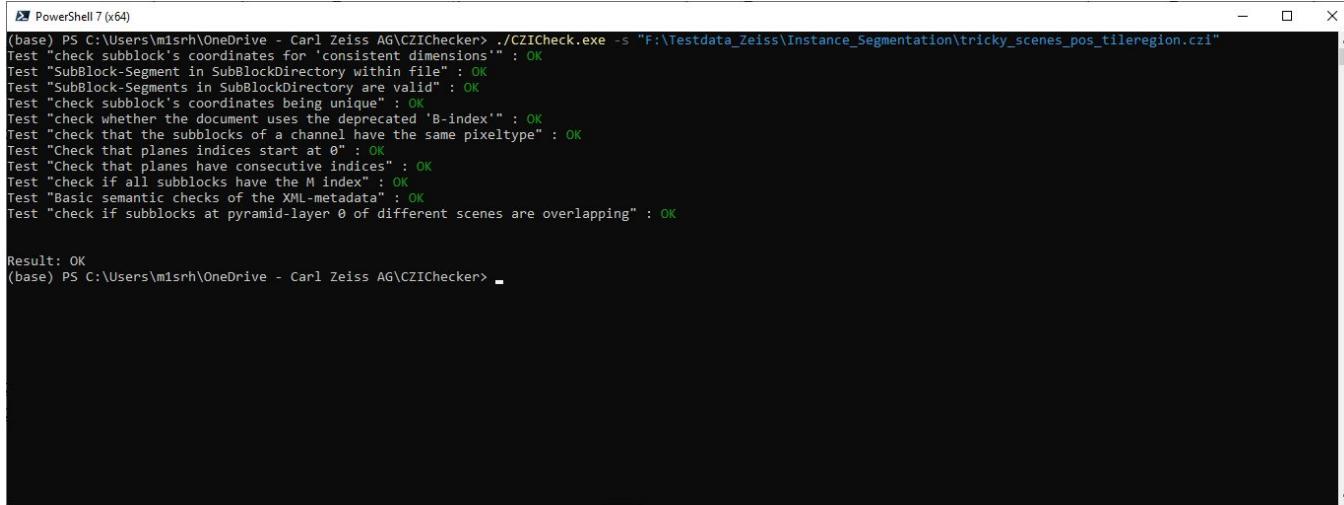
Compress CZI Images Files from a cross-platform UI (not released yet)



- Cross Platform GUI App
- Developed, tested and released on Win-x64 and Linux-x64
- Designed to work with large CZI collections
- Multi-threaded processing
- Strictly non-destructive

File Formats – CZI Tools

Check CZI for internal errors



The screenshot shows a PowerShell window titled "PowerShell 7 (x64)". The command run is `./CZICheck.exe -s "F:\Testdata_Zeiss\Instance_Segmentation\tricky_scenes_pos_tilerregion.czi"`. The output displays several test results, all of which are marked as "OK". The tests include checking subblock coordinates, segment validity, uniqueness, deprecated B-index usage, pixeltype consistency, plane indices, M index presence, XML metadata, and overlapping subblocks across different scenes. The final result is also marked as "OK".

```
(base) PS C:\Users\m1srh\OneDrive - Carl Zeiss AG\CZIChecker> ./CZICheck.exe -s "F:\Testdata_Zeiss\Instance_Segmentation\tricky_scenes_pos_tilerregion.czi"
Test "check subblock's coordinates for 'consistent dimensions'" : OK
Test "SubBlock-Segment in SubBlockDirectory within file" : OK
Test "SubBlock-Segments in SubBlockDirectory are valid" : OK
Test "check subblock's coordinates being unique" : OK
Test "check whether the document uses the deprecated 'B-index'" : OK
Test "check that the subblocks of a channel have the same pixeltyp" : OK
Test "Check that planes indices start at 0" : OK
Test "Check that planes have consecutive indices" : OK
Test "Check if all subblocks have the M index" : OK
Test "Basic semantic checks of the XML-metadata" : OK
Test "check if subblocks at pyramid-layer 0 of different scenes are overlapping" : OK

Result: OK
(base) PS C:\Users\m1srh\OneDrive - Carl Zeiss AG\CZIChecker> -
```

<https://github.com/ZEISS/czicheck>

- Check CZI for internal errors
- For Devs: Use it to make sure that newly created CZI (inside project XYZ) pass the check !
- For testers: Use it to verify that the CZI used testing is not causing the issue one tries to track down

File Formats – CZI Dimension Identifiers



Identifier or Letter	Name of dimension	Explanation & Remarks
X	Width	width of the image in X dimension in [pixel]
Y	Height	height of the image in Y dimension in [pixel]
C	Channel	number of channels
Z	Slice	number of z-planes
T	Time	number of time points
M	Mosaic	index of tile for compositing a scene
S	Scene	contiguous regions of interest in a mosaic image
R	Rotation	
I	Illumination	SPIM direction for LightSheet
B	Block	acquisition block
H	Phase	e.g. Airy detector fibers
V	View	e.g. for SPIM

Tile Region

- consists out of many individual tiles (which may overlap)
- can have arbitrary shape
- a TileRegion also is tagged as a **scene** (S-Index) and counts as one scene

Tiles

- individual tiles (M-index inside CZI) refers to individual acquisition tiles inside a bigger Tile Region
- overlap can be adjusted inside the experiment
- a tile itself can have multiple channels and slices

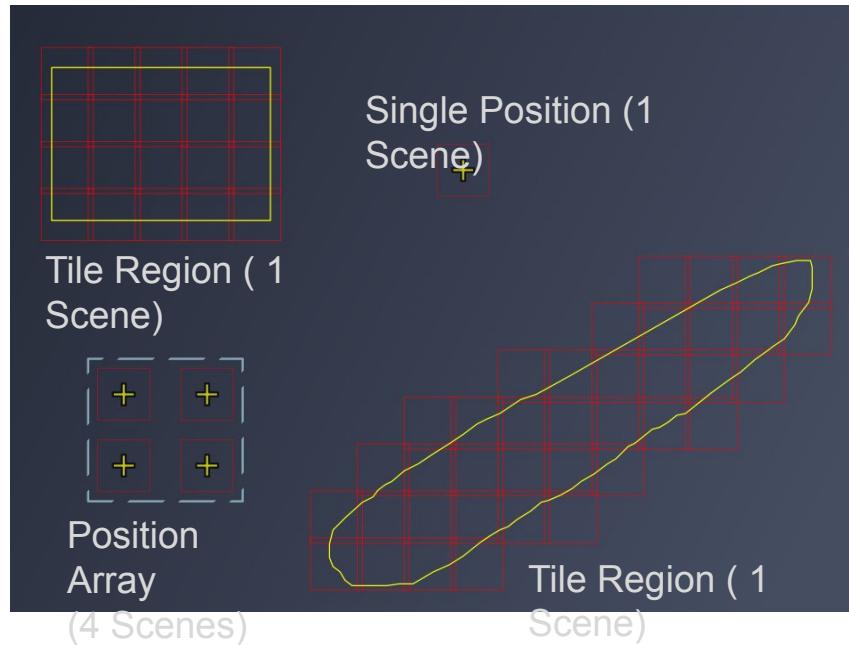
Positions

- single multi-dimensional stack acquired at a specific XYZ position
- a single positions counts as one scene
- position arrays are groups of positions

Scenes

- tile regions or single positions (inside an array) get the tag scene
- scene tag can be used to "show" only specific scenes etc.

File Formats – CZI TileRegions, Positions and Scenes



File Formats – CZI TileRegions, Positions and Scenes



The image shows a software interface for managing tile regions and positions. On the left, there are four panels labeled D6, D7, E6, and E7:

- D6: A dark gray square with a small yellow crosshair icon in the center.
- D7: A dark gray square containing a red grid and a yellow outline of a irregular shape, representing a tile region.
- E6: A dark gray square with a large blue rectangular selection box and a small yellow crosshair icon near the bottom-left corner.
- E7: A dark gray square containing a 5x5 grid of yellow crosshair icons, with a dashed blue rectangle highlighting a 3x3 subgrid in the bottom-right corner.

On the right is a central window titled "Tiles". It has two sections:

- Tile Regions:** A table with columns: Name, Category, Tiles, Z (μm). It contains two entries:

Name	Category	Tiles	Z (μm)
D6	Default	25	111,8
TR1	Default	338	111,8
- Positions:** A table with columns: Name, X (μm), Y (μm), Z (μm), Category. It contains one entry:

Name	X (μm)	Y (μm)	Z (μm)	Category
P1	49561,6	44659,2	111,8	Default

Each section has a "Verify" button at the bottom right. The entire interface has a dark theme with blue highlights.

File Formats – CZI Image Bounds



Image Bounds

This term is often used to specify the total size of the dimensions of an CZI. In that context the term **bounding box** is often used. To illustrate the concept, consider the following CZI image:

- 2 Scenes **SizeS = 2**
- 2 Time Points **SizeT = 2**
- 25 (5x5) Tiles per Tile Region **SizeM = 25**
- 3 Slices **SizeZ = 3**
- 2 Channels **SizeC = 2**
- X (Width) total **SizeX = 94170**
- Y (Height) total **SizeY = 4170**

The screenshot shows a software interface for viewing CZI file metadata. The top bar displays three files: "scenes.czi", "mouse_brain_service_2x2...ery.czi", and "S=2_5x5Tiles_T=2_Z=3_C=2.czi". The main area is a tree view of the file structure with the following details:

- Experiment**:
 - HardwareSetting
 - ImageScaling
 - CustomAttributes =
- Information**:
 - Measurement
 - User =
 - Application
 - Document
- Image**:
 - SizeX = 94710
 - SizeY = 4710
 - SizeS = 2
 - SizeM = 25
 - OriginalCompressionMethod = Uncompressed
 - OriginalEncodingQuality = 100
 - AcquisitionDateAndTime = 2019-01-23T16:19:42.171810Z
 - SizeC = 2
 - SizeZ = 3
 - SizeT = 2
 - ComponentBitCount = 12
 - PixelType = Gray16

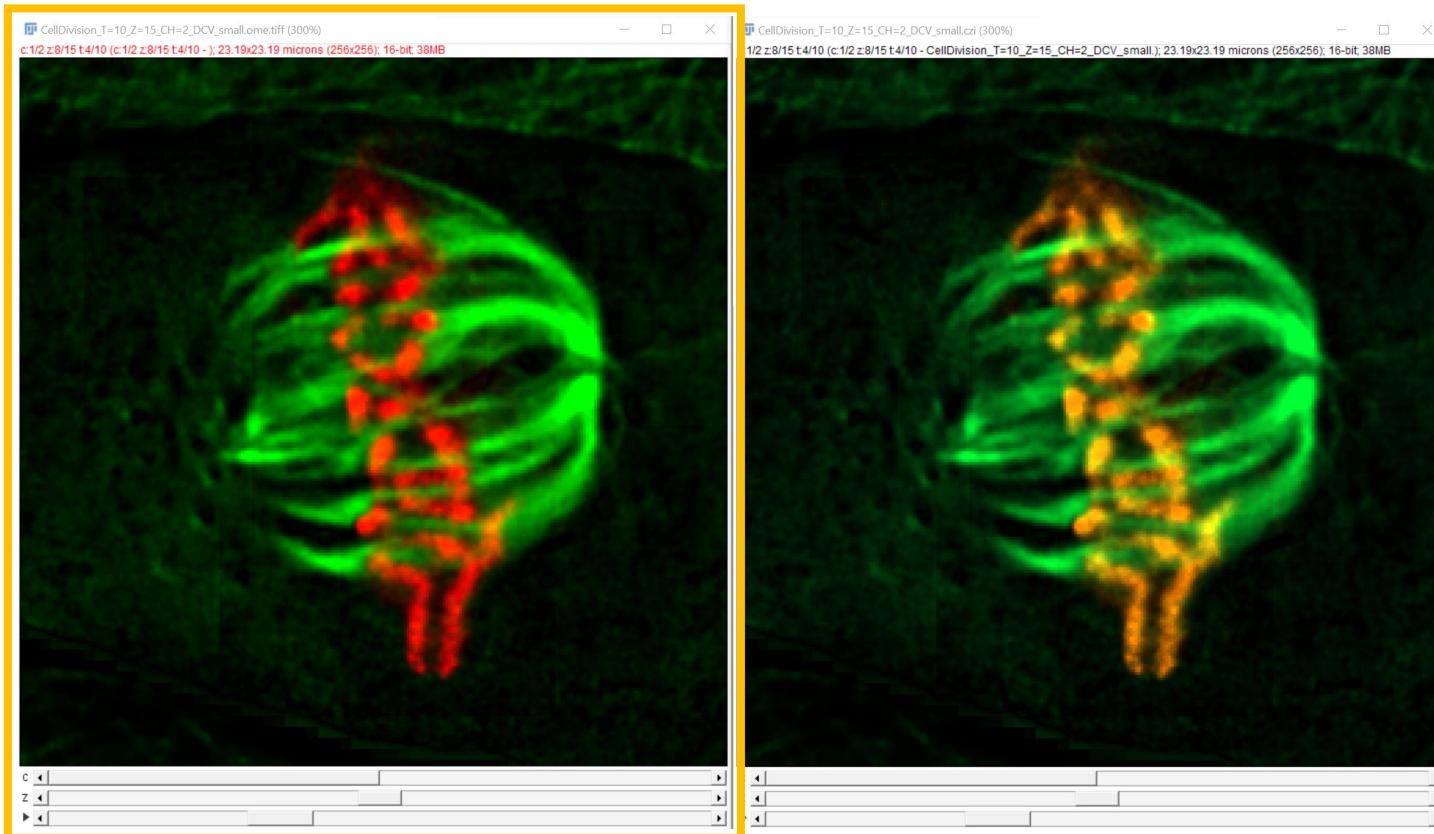
On the left, there is a vertical toolbar with icons for 2D, Split, Gallery, Ortho, 2.5D, 3D, Tomo3D, Profile, Histo, and Unmix.

File Formats – CZI Image Bounds



- Open Microscopy Environment (OME):
 - <https://www.openmicroscopy.org/>
 - <https://ome-model.readthedocs.io/en/stable/ome-tiff/specification.html#>
 - <https://github.com/ome/ome-model-documentation/blob/12e87cc5f2c3714a22b06af8163f4a60566dc87a/ome-tiff/specification.rst>
- “def-facto” standard format for analyzing microscopy images (but will not really developed any further)
- is supported by many applications and languages (Java, Python, C++, MATLAB, R)
- already “quite old” and has important limitations
 - working with large, multi-dimensional datasets is tricky (or even impossible)
 - additional dimension apart from T-C-Z-Y-X have limited support
- main library to work with is BioFormats: <https://www.openmicroscopy.org/bio-formats/>
- **ZEN can easily read & write OME-TIFF**

File Formats – OME-TIFF in Fiji (BioFormats)



File Formats – OME-TIFF in Napari



The screenshot shows the Napari software interface with the following components:

- Top Bar:** File, View, Plugins, Window, Tools, Help.
- Left Sidebar (layer controls):** Includes sliders for opacity (1.0), contrast limits, auto-contrast (once or continuous), gamma (1.0), color map (green), blending (additive), interpolation (linear), depiction (volume), and rendering (mip).
- Layer List:** Shows two layers: "0 :: Image:1 :: LED470" and "0 :: Image:1 :: LED555".
- Bottom Toolbar:** Includes icons for zoom, transform, selection, and navigation.
- Right Panel (OME Tree Widget):** Displays the hierarchical structure of the OME-TIFF file:
 - CellDivision.T=10_Z=15.CH=2.DCV_small.ome.tif
 - images
 - instruments
 - Instrument:1
 - detectors
 - filter_sets
 - filters
 - id: Instrument:1
 - light_emitting_diodes
 - LightSource:1
 - LightSource:2
 - microscope
 - kind: microscope
 - model: Castor.Stand
 - type: Inverted
 - objectives
 - Objective:1
 - id: Objective:1
 - immersion: Water
 - kind: objective
 - lens_na: 1.2
 - model: Plan-Apochromat 50x/1.2
 - nominal_magnification: 50.0
 - working_distance: 800.0

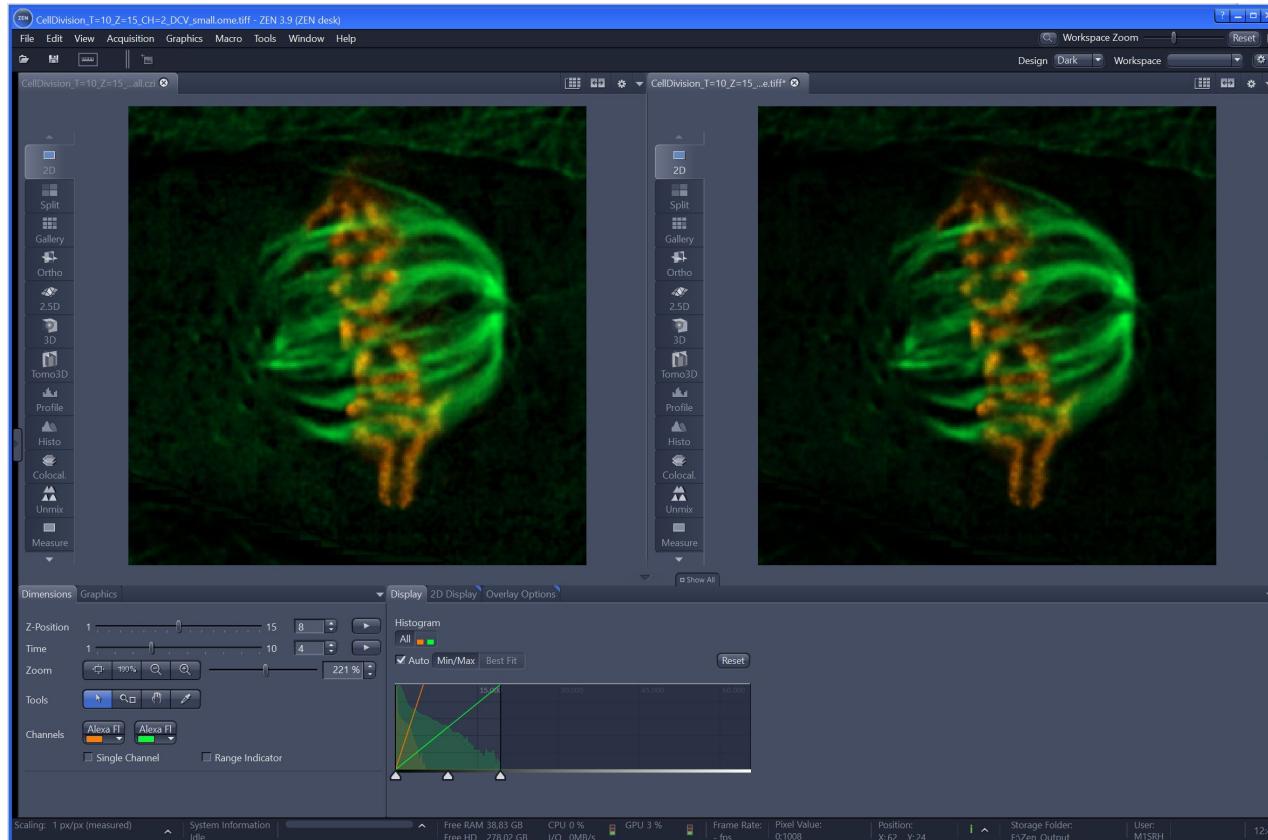
File Formats – OME-TIFF in Fiji (BioFormats)



```
OME Metadata - CellDivision_T=10_Z=15_CH=2_DCV_small.ome.tif

<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06 http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">
  <Instrument ID="Instrument:1">
    <Microscope Model="Castor.Stand" Type="Inverted"/>
    <LightEmittingDiode ID="LightSource1"/>
    <LightEmittingDiode ID="LightSource2"/>
    <Detector ID="Detector:1" Model="AxioCam 506m"/>
    <Objective ID="Objective:1" Immersion="Water" LensNA="1.2" Model="Plan-Apochromat 50x/1.2" NominalMagnification="50.0" WorkingDistance="800.0" WorkingDistanceUnit="μm"/>
    <FilterSet ID="FilterSet:1">
      <FilterSet ID="FilterSet:2">
        <Filter ID="Filter:1">
        <Filter ID="Filter:2">
        <Filter ID="Filter:3">
        <Filter ID="Filter:4">
        <Filter ID="Filter:5">
        <Filter ID="Filter:6">
        <Filter ID="Filter:7">
        <Filter ID="Filter:8">
      </FilterSet>
    </FilterSet>
    <Image ID="Image:0" Name="">
      <AcquisitionDate>2016-02-12T09:41:02.491</AcquisitionDate>
      <Description/>
      <InstrumentRef ID="Instrument:1"/>
      <ObjectiveSettings ID="Objective:1" Medium="Air" RefractiveIndex="1.33"/>
      <Pixels BigEndian="false" DimensionOrder="XYCZ" ID="Pixels0" Interleaved="false" PhysicalSizeX="0.09057667" PhysicalSizeXUnit="μm" PhysicalSizeY="0.09057667" PhysicalSizeYUnit="μm" PhysicalSizeZ="0.32" PhysicalSizeZUnit="μm" SignificantBits="16" SizeC="2" SizeT="10" SizeX="256" SizeY="256" SizeZ="15" Type="uint16">
        <Channel AcquisitionMode="WideField" Color="-8519423" ContrastMethod="Fluorescence" EmissionWavelength="568.0" EmissionWavelengthUnit="nm" ExcitationWavelength="553.0" ExcitationWavelengthUnit="nm" Fluor="Alexa Fluor 555" ID="Channel:0:0" IlluminationType="Epifluorescence" Name="LED555" SamplesPerPixel="1">
          <DetectorSettings Binning="1x1" ID="Detector:1"/>
          <FilterSetRef ID="FilterSet:1"/>
          <LightPath/>
        <Channel AcquisitionMode="WideField" Color="16724991" ContrastMethod="Fluorescence" EmissionWavelength="517.0" EmissionWavelengthUnit="nm" ExcitationWavelength="493.0" ExcitationWavelengthUnit="nm" Fluor="Alexa Fluor 488" ID="Channel:0:1" IlluminationType="Epifluorescence" Name="LED470" SamplesPerPixel="1">
          <TiffData/>
          <Plane DeltaT="0.0" DeltaTUnit="s" PositionX="16977.1523" PositionXUnit="reference frame" PositionY="18621.4883" PositionYUnit="reference frame" PositionZ="1113.21" PositionZUnit="reference frame" TheC="0" TheT="0" TheZ="0"/>
          <Plane DeltaT="0.3220184" DeltaTUnit="s" PositionX="16977.1523" PositionXUnit="reference frame" PositionY="18621.4883" PositionYUnit="reference frame" PositionZ="1113.21" PositionZUnit="reference frame" TheC="1" TheT="0" TheZ="0"/>
          <Plane DeltaT="60.0354347" DeltaTUnit="s" PositionX="16977.1523" PositionXUnit="reference frame" PositionY="18621.4883" PositionYUnit="reference frame" PositionZ="1113.21" PositionZUnit="reference frame" TheC="0" TheT="1" TheZ="0"/>
          <Plane DeltaT="60.361454" DeltaTUnit="s" PositionX="16977.1523" PositionXUnit="reference frame" PositionY="18621.4883" PositionYUnit="reference frame" PositionZ="1113.21" PositionZUnit="reference frame" TheC="1" TheT="1" TheZ="0"/>
      </Pixels>
    </Image>
  </Instrument>
</OME>
```

File Formats – OME-TIFF in ZEN



File Formats – OME-ZARR and OME-NGFF

What is ZARR?



Zarr is motivated by the need for a simple, transparent, open, and community-driven format that supports high-throughput distributed I/O on different storage systems.

- Zarr data can be stored in any storage system that can be represented as a key-value store, including most commonly POSIX file systems and cloud object storage but also zip files as well as relational and document databases.

Features:

- Chunk multi-dimensional arrays along any dimension
- Store arrays in memory, on disk, inside a Zip file, on S3, etc.
- Read and write arrays concurrently from multiple threads or processes.
- Organize arrays into hierarchies via annotatable groups

File Formats – OME-ZARR and OME-NGFF

The future of microscopy image file formats (for processing)?

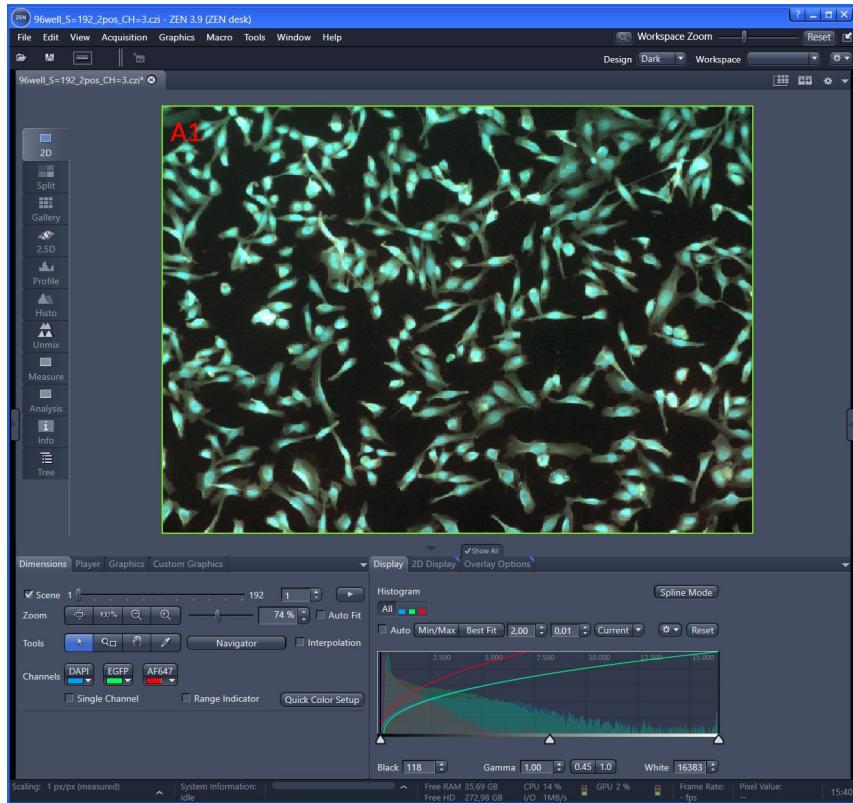
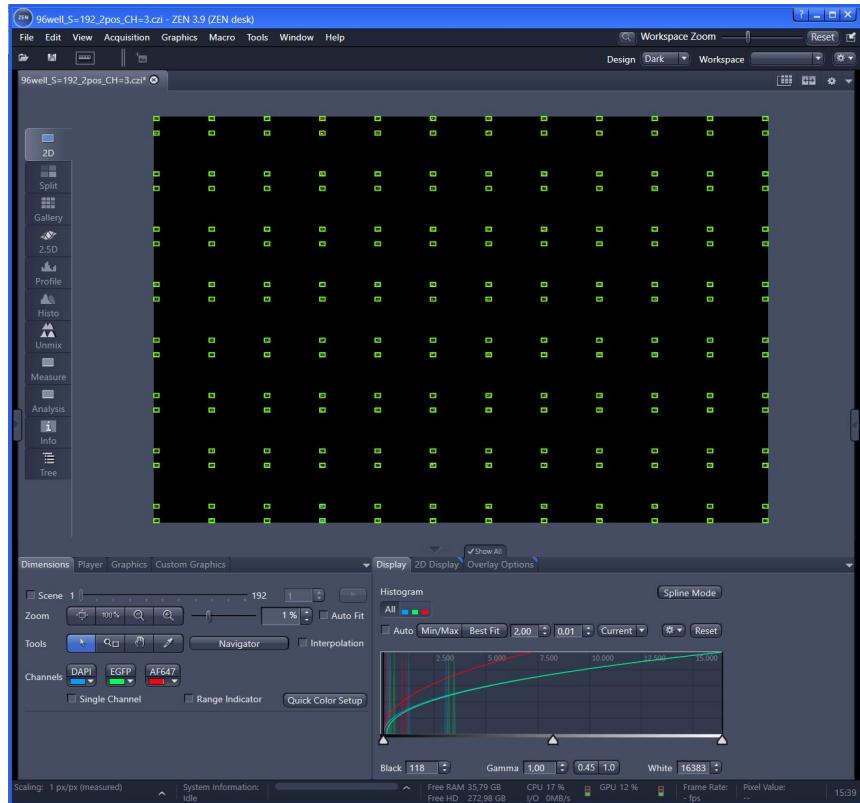


- OME-TIFF combines 2D TIFF format with metadata cast in XML in the TIFF header
- appropriate for many applications, where the plane-based access pattern is appropriate
- N-dimensional formats like HDF5 (“Hierarchical Data Format”) provide much more flexibility and allow chunking across different dimensions chosen by the user
 - depend on fast random access to the entire file □ not provided by large scalable cloud-based storage
- OEM-ZARR is basically ZARR + OME meta specifications
- specification is called **OME-NGFF (OME-Next Generation File Format)**

OME-ZARR is a cloud-optimized, chunked format that functions as a common API for both desktop, cluster, and web-based tools as well as national and international repositories.

File Formats – OME-ZARR and OME-NGFF

Application example: Annotated hierarchies for Plate data

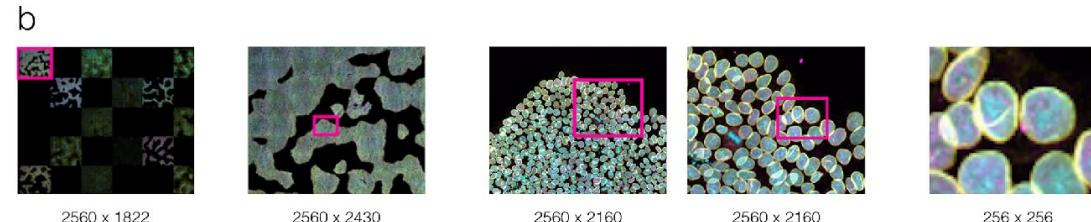
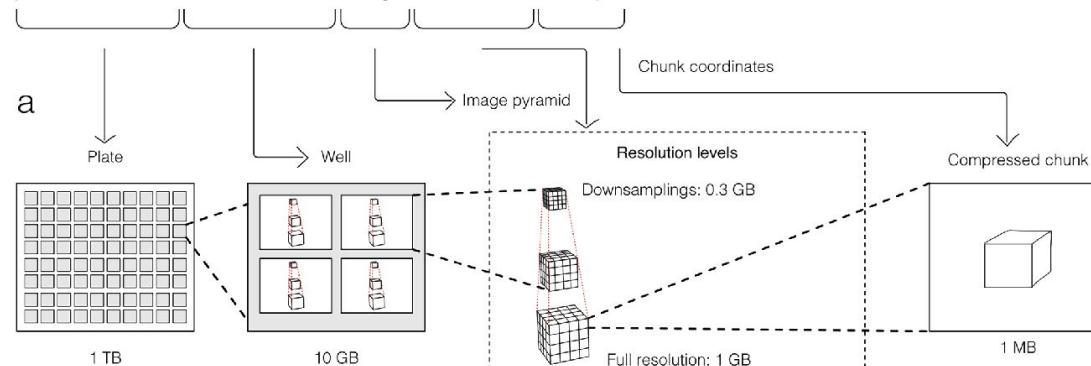


File Formats – OME-ZARR and OME-NGFF

Application example: Annotated hierarchies for Plate data



plate.ome.zarr/row/col/well/image/resolution/t/c/z/y/x



- top-level directory can be 1TB plate with > 100000 pixel in XY
- lowest level directory has chunks of N-dimensional data of 1MB
- Amount of pixel at different zoom levels to be loaded by the client is manageable
- Chunks can be compressed
- Metadata can be attached to each group and array separately in web-readable JSON files

File Formats – OME-ZARR and OME-NGFF

Application Example: Annotated hierachies for Plate data



NGFF-Converter

File Edit View Help

Output Folder <Same as input>

Datei Start Freigeben Ansicht

96Well_Ch=1_50Wells_3x3_pro_Well.zarr

96Well_Ch=1_50Wells_9P_pro_Well.zarr

96well_S=192_2pos_CH=3.zarr

A B C D E F G H OME .zattrs .zgroup

11 Elemente

+ Add files - Remove Selected × Remove All ✓ Clear Completed Show Logs

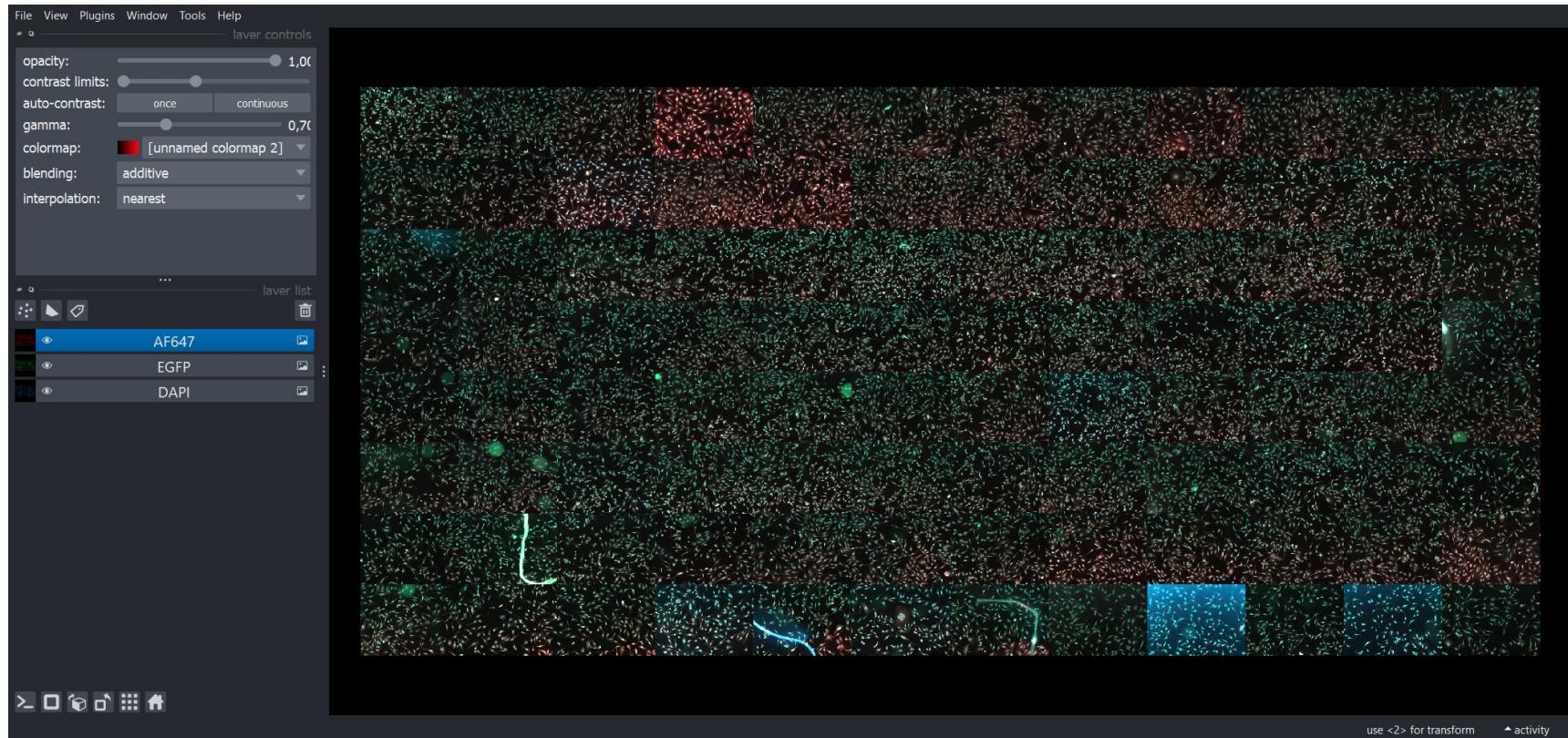
Found and added 1 supported file(s)

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
"bioformats2raw.layout" : 3,
"plate" : {
  "columns" : [ { "name" : "1" },
    { "name" : "2" },
    { "name" : "3" },
    { "name" : "4" },
    { "name" : "5" },
    { "name" : "6" },
    { "name" : "7" },
    { "name" : "8" },
    { "name" : "9" },
    { "name" : "10" },
    { "name" : "11" },
    { "name" : "12" },
    { "name" : "" },
    "wells" : [ {
      "field_count" : 2,
      "rows" : [ {
        "name" : "A"
      },
      { "name" : "B" },
      { "name" : "C" },
      { "name" : "D" },
      { "name" : "E" },
      { "name" : "F" },
      { "name" : "G" },
      { "name" : "H" }
    ],
      "version" : "0.4"
    }
  ]
}, {
  "plate" : {
    "columns" : [ { "name" : "" },
      "wells" : [ {
        "path" : "A/1",
        "rowIndex" : 0,
        "columnIndex" : 0
      },
      {
        "path" : "A/2",
        "rowIndex" : 0,
        "columnIndex" : 1
      },
      {
        "path" : "A/3",
        "rowIndex" : 0,
        "columnIndex" : 2
      },
      {
        "path" : "A/4",
        "rowIndex" : 0,
        "columnIndex" : 3
      },
      {
        "path" : "A/5",
        "rowIndex" : 0,
        "columnIndex" : 4
      },
      {
        "path" : "A/6",
        "rowIndex" : 0,
        "columnIndex" : 5
      },
      {
        "path" : "A/7",
        "rowIndex" : 0,
        "columnIndex" : 6
      },
      {
        "path" : "A/8",
        "rowIndex" : 0,
        "columnIndex" : 7
      },
      {
        "path" : "A/9",
        "rowIndex" : 0,
        "columnIndex" : 8
      },
      {
        "path" : "A/10",
        "rowIndex" : 0,
        "columnIndex" : 9
      },
      {
        "path" : "A/11",
        "rowIndex" : 0,
        "columnIndex" : 10
      },
      {
        "path" : "A/12",
        "rowIndex" : 0,
        "columnIndex" : 11
      }
    ]
  }
}
```

LN:1 Col:1 Pos:1 Windows (CR/LF) UTF-8 INS LN:1 Col:1 Pos:1 Windows (CR/LF) UTF-8 INS

File Formats – OME-ZARR and OME-NGFF

Application Example: Visualize in Napari

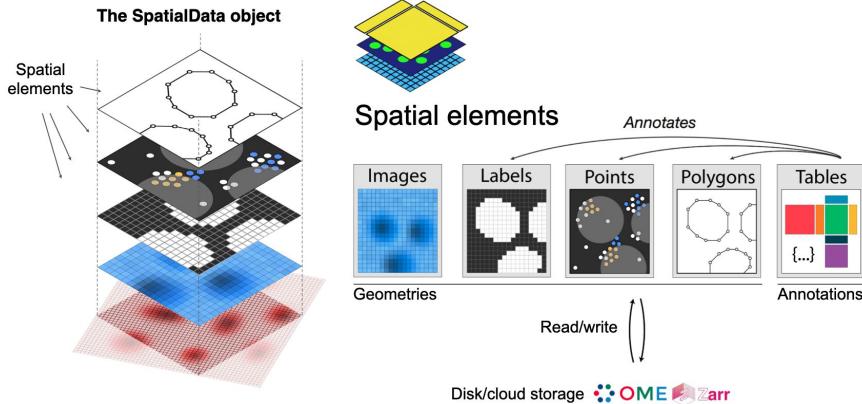


File Formats – OME-ZARR and OME-NGFF

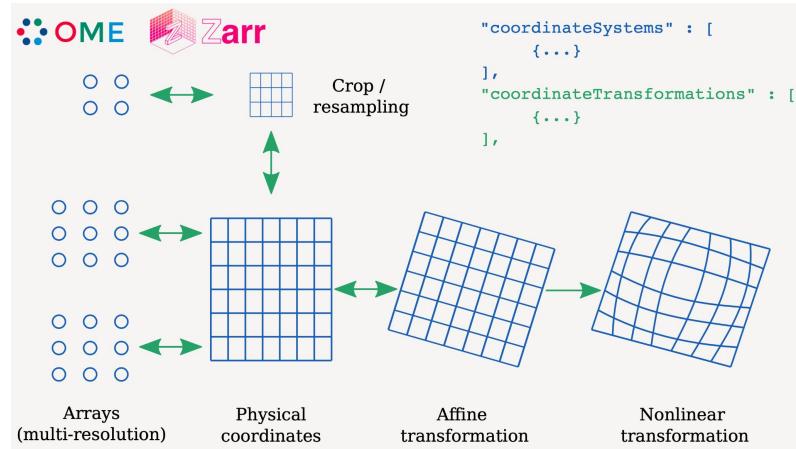
Additional Spatial Data and Tables



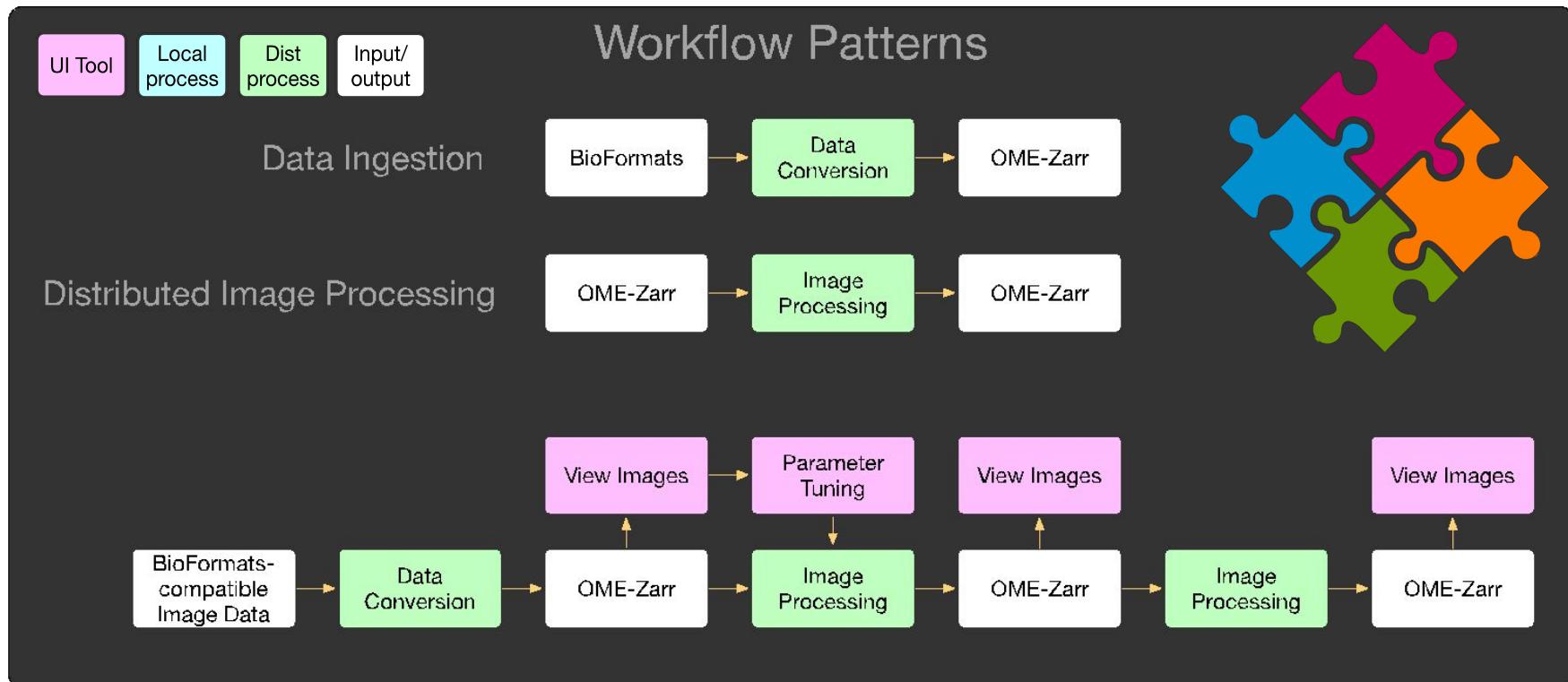
Tables (i.e. SpatialData)



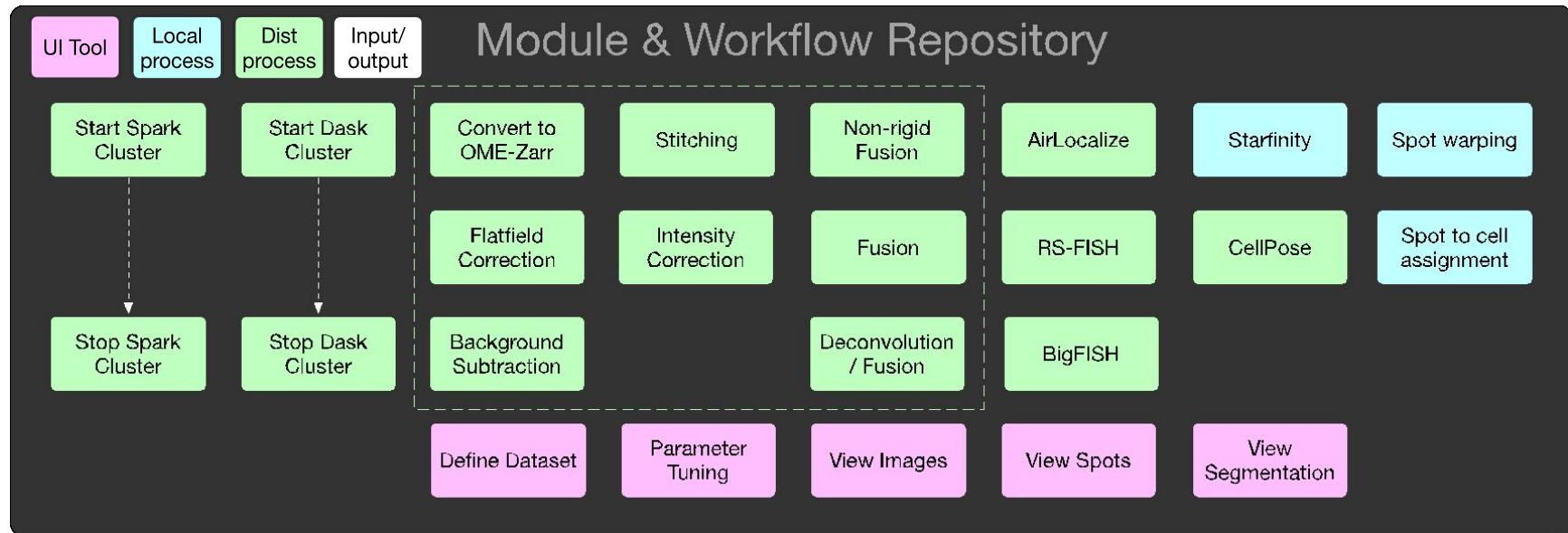
Transforms



OME-ZARR – How is it used already



OME-ZARR – How is it used already



- Reusable modules and subworkflows
- Each module is encapsulated into a BioContainer
(<https://biocontainers.pro/>)
- Module repository: <https://github.com/JaneliaSciComp/nextflow-modules>

Kudos & Links



- <https://www.zeiss.com/microscopy/en/applications.html>
- <https://www.ibiology.org/online-biology-courses/microscopy-series/>
- <https://www.youtube.com/watch?v=d8Tqoo0S6qc>
- <https://github.com/CamachoDejay>
- <https://www.tuwien.at/phy/iap/biophysics/team/gerhard-schuetz>
- <https://www.microscropyu.com/tutorials>
- <https://petebankhead.gitbooks.io/imagej-intro/content/>
- https://downloads.micron.ox.ac.uk/lectures/micron_course_2020/lecture-20-introduction-bioimage-analysis.pdf
- <https://www.edx.org/course/image-processing-and-analysis-for-life-scientists>
- <http://www.imaging-qit.com/olympus-website-bioimage-data-analysis>
- <https://bioimagebook.github.io/>
- https://www.biodip.de/w/images/6/67/2011-10-Basics_of_Imaging_Processing_Course.pdf
- https://github.com/dwaithe/ONBI_image_analysis
- https://github.com/quiwitz/Python_image_processing
- <https://neubias.github.io/training-resources/>
- <https://www.scribbr.com/methodology/reliability-vs-validity/>
- <https://www.scribbr.com/methodology/reproducibility-repeatability-replicability/>
- <https://protocolsmethods.springernature.com/posts/intelligent-microscopy-work-smarter-not-harder>
- https://www.nature.com/articles/s41592-022-01589-x?utm_campaign=related_content&utm_source=MTHDS&utm_medium=communities
- <https://www.qu.se/en/core-facilities/centre-for-cellular-imaging/smart-microscopy>
- <https://www.nature.com/articles/s41592-023-01912-0>
- <https://www.nature.com/articles/s41592-019-0708-0>
- <https://www.eurobioimaging.eu/service/feedback-microscopy-FDBKM>
- https://neubias.github.io/training-resources/multidimensional_image_basics/index.html
- <https://forum.image.sc/t/outcomes-of-the-next-generation-bioimage-analysis-workflows-hackathon/88733>
- <https://link.springer.com/article/10.1007/s00418-023-02209-1>
- <https://doi.org/10.1038/s41592-022-01482-7>
- <https://doi.org/10.48550/arXiv.2101.09153>
- <https://doi.org/10.1101/2021.03.31.437929>
- <https://doi.org/10.1038/s41592-021-01326-w>
- <https://biocontainers.pro>
- <https://github.com/JaneliaSciComp/nextflow-modules>
- <https://doi.org/10.1016/j.cell.2021.11.024>