

Open Application Development (OAD)

Tools for Smart Microscopy



ZEN is only part of the workflow



ZEN Connect

OAD

Scripting

Analysis

Machine Learning

Simplify

ZEN blue

Fiji

Extensions

Python

MATLAB

KNIME

Automation

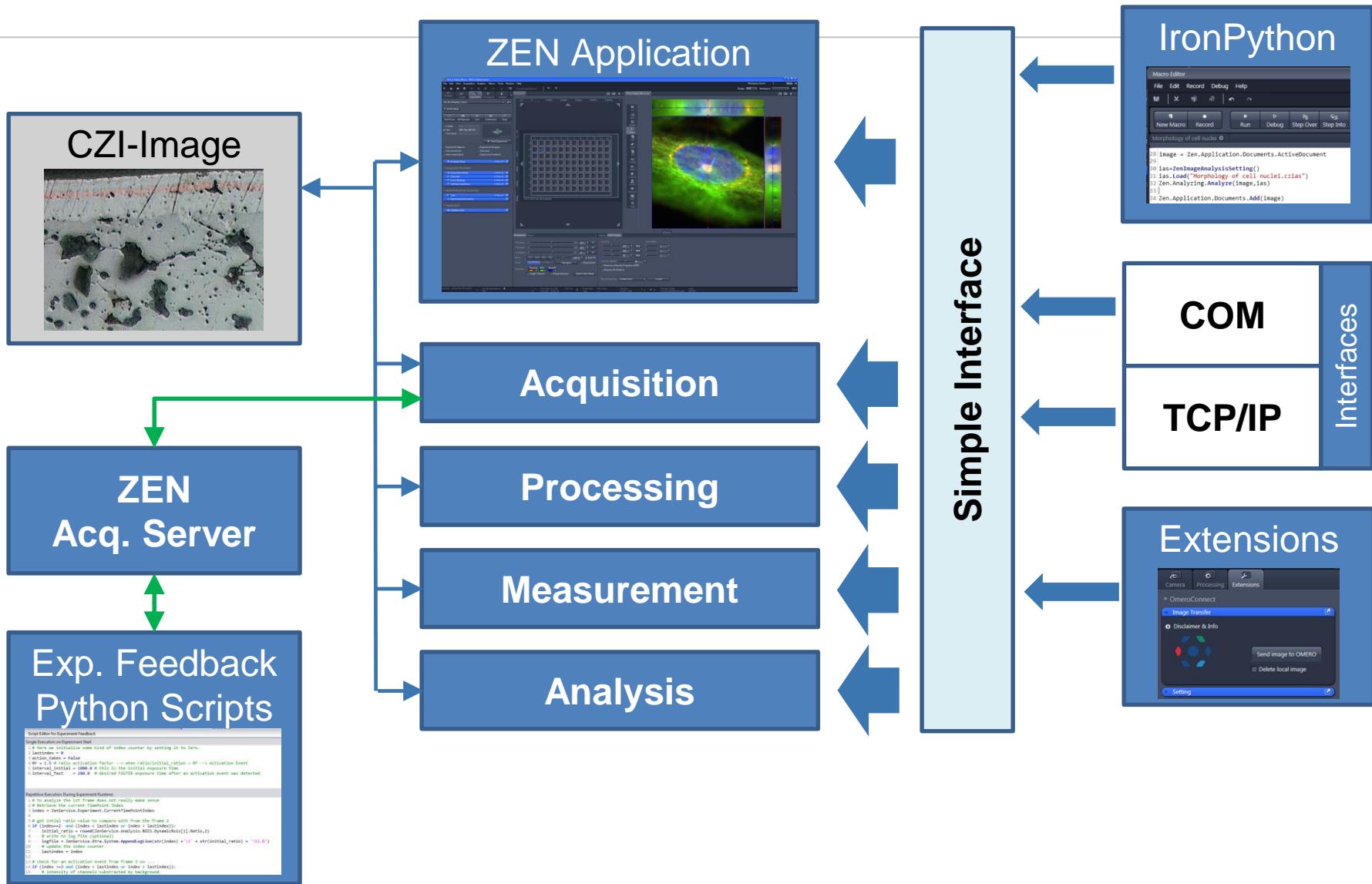
BioFormats

OAD – General Concept and Key Features



- **Open Application Development** (OAD) uses powerful **Python Scripts** to **simplify, customize** and **automate** your workflows.
- **Analyze** and **Exchange** data with applications like **Fiji, Python, Knime, CellProfiler, Icy, MATLAB, Excel** and ...
- The **CZI-API for .NET** (ZeissImgLib) and for **C++** (libCZI) and **BioFormats** (CZIReader) allow easy access to CZI files from many external applications. (OME-TIFF Export and Import in ZEN Blue is possible)
- **BioFormats Import** as a module inside ZEN Blue
- Create “smart” experiments with **Experiment. Feedback** and modify the acquisition **On-the-fly** based on **Online Image Analysis** and **External Inputs**

OAD – ZEN Interfaces



Carl Zeiss Image (CZI) File Format



- **CZI = Unified File Format for all ZEISS Light Microscopy Systems**
- Allows to JPG-XR compression **with lossless option**
- **XML structure of Metadata** developed with “an eye” on OME Meta data scheme (to facilitate compatibility)
- **Open up access** to your imaging **data** acquired with ZEISS Microscopes
- **CZIReader as part of OME-BioFormats-Reader** → ImageJ/Fiji, OMERO, KNIME, Python, CellProfiler, MATLAB, Icy etc.
- API: **ZeissImageLib** (.NET) and **libCZI** (C++) for commercial & non-commercial software

<http://www.zeiss.com/czi>

<http://github.com/zeiss-microscopy/libCZI>

libCZI – open-source crossplatform C++ library



The screenshot shows the GitHub repository page for `zeiss-microscopy/libCZI`. The repository is described as an "Open Source Cross-Platform C++ library to read CZI image files". It has 10 commits, 1 branch, 0 releases, 2 contributors, and is licensed under GPL-3.0. The latest commit was 13 hours ago. The repository features include image-processing, cross-platform, file-format, czi, microscopy, zeiss, open-source, c-plus-plus, and Manage topics.

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

Open Source Cross-Platform C++ library to read CZI image files

image-processing cross-platform file-format czi microscopy zeiss open-source c-plus-plus Manage topics

10 commits 1 branch 0 releases 2 contributors GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

zeissmicroscopy committed on GitHub Update README.md Latest commit 3a55c08 13 hours ago

File	Description	Time Ago
Doc	initial upload to GitHub	6 days ago
Src	adding .gitignore	5 days ago
images	Add files via upload	2 days ago
LICENSE	Initial commit	6 days ago
README.md	Update README.md	13 hours ago

README.md

libCZI

Open Source Cross-Platform C++ library to read CZI image files

libCZI is a library intended for providing read-only access to the information contained in CZI-documents.

It features:

- reading subblocks and get the content as a bitmap
- reading subblocks which are compressed with JPEG-XR

OAD – Examples and Script are available on GitHub



Screenshot of the GitHub repository page for `zeiss-microscopy / OAD`.

The repository summary shows:

- Code: 43 commits
- Issues: 0
- Pull requests: 0
- Projects: 0
- Wiki
- Insights
- Settings

Topics listed include: python, microscopy, scripting, automation, imaging, machine-learning, open-source, image-analysis, zen, zen-blue, acquisition, python-script, workflow, oad, image-processing.

Contributors: 2 contributors

Licenses: GPL-3.0

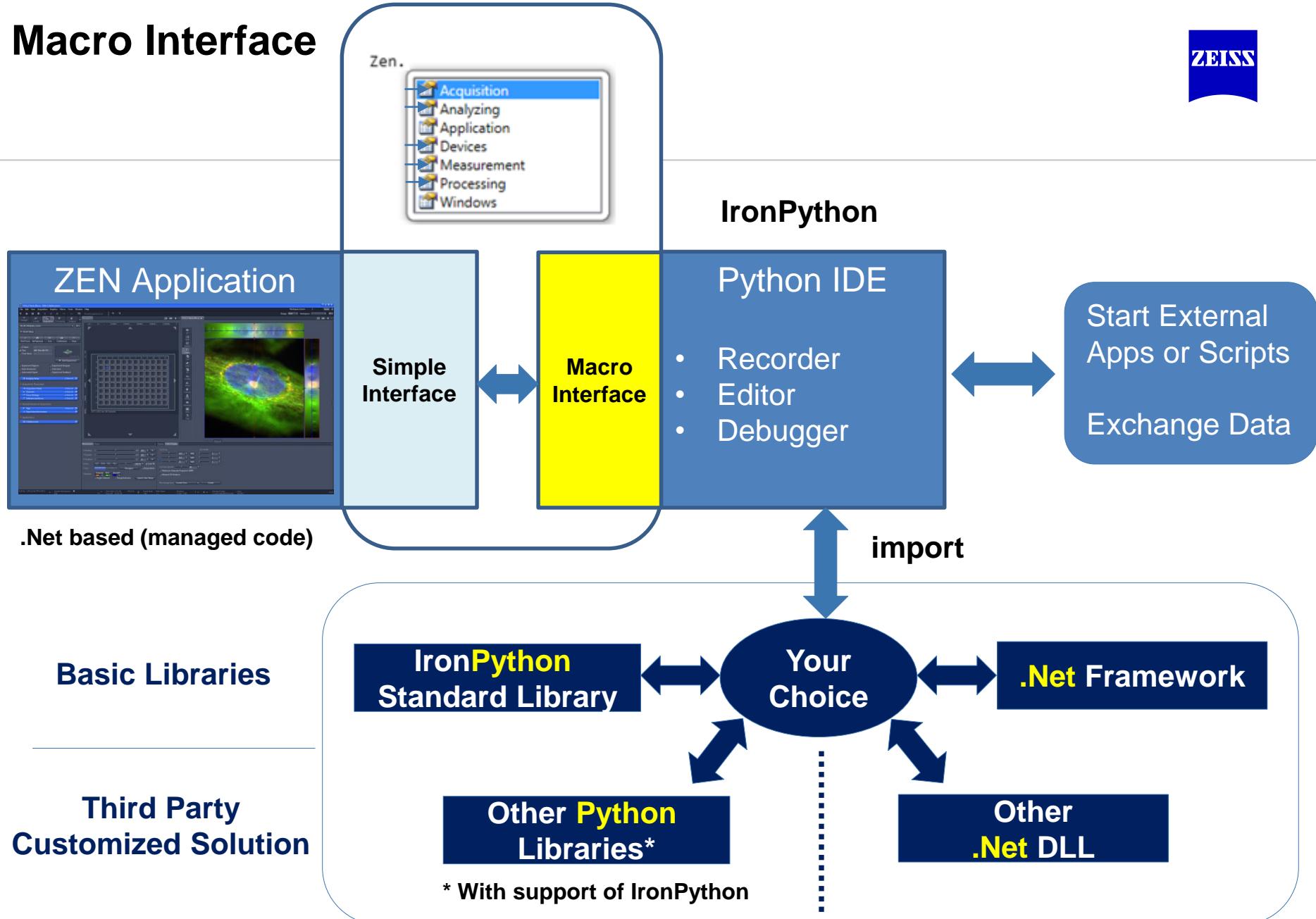
Branch: master

Latest commit: 8ab37be 23 hours ago

Repository contents:

- zeissmicroscopy updates
- Experiment_Feedback
- Image_Analysis_Settings
- Images
- Interfaces
- Scripts
- Testdata
- Tutorials
- Videos

Macro Interface



Macro Interface – General Remarks



- ZEN uses IronPython as its internal scripting language → this allows using ZEN and .NET functionality directly, but not SciPy, NumPy etc.
- Not everything can be recorded, but that does not mean, there is not command for a specific functionality
- script editor has an built-in help file
- there is a OAD forum
- Example and Scripts can be found on the DVD

The screenshot shows the Macro Editor window with the following details:

- Toolbar:** File, Edit, Record, Debug, Help (selected), New Macro, Record, Run, Debug, Step Over, Step Into, Step out, Pause, Stop, Reset.
- Help Menu:** Contents..., Macro Object Model..., Forum... (disabled).
- Script Area:** Wizard_RareEventDetection_DLLs (active tab) and morphology_cell_nuclei_data_transfer_excel (inactive tab).
- Code:**

```
"""
File: morphology_cell_nuclei_data_transfer_excel.czmac
Author: SRh + CSC
Date: 2017_05_19
Version: 0.2

Macro name: Morphology of cell nuclei and data transfer to Excel
Required files:
- morphology_of_nuclei.czci
- morphology_cell_nuclei.czias

LOAD IMAGE, SEGMENT NUCLEI, MEASURE SIZE OF NUCLEI
SEND DATA TO EXCEL AND GENERATE DATALIST AND CHART
"""

import clr
clr.AddReferenceByName('Microsoft.Office.Interop.Excel, Version=11.0.0.0, Culture=neutral,
from Microsoft.Office.Interop import Excel
excel = Excel.ApplicationClass()
from System.IO import Directory, Path, FileInfo
```

Macro Interface – Object Model Documentation



Macro ObjectModel Documentation

Ausblenden Suchen Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen

public class ZenAcquisitionLM : StandardDocumentableObject

The ZenAcquisitionLM type exposes the following members.

Constructors

Name	Description
ZenAcquisitionLM	

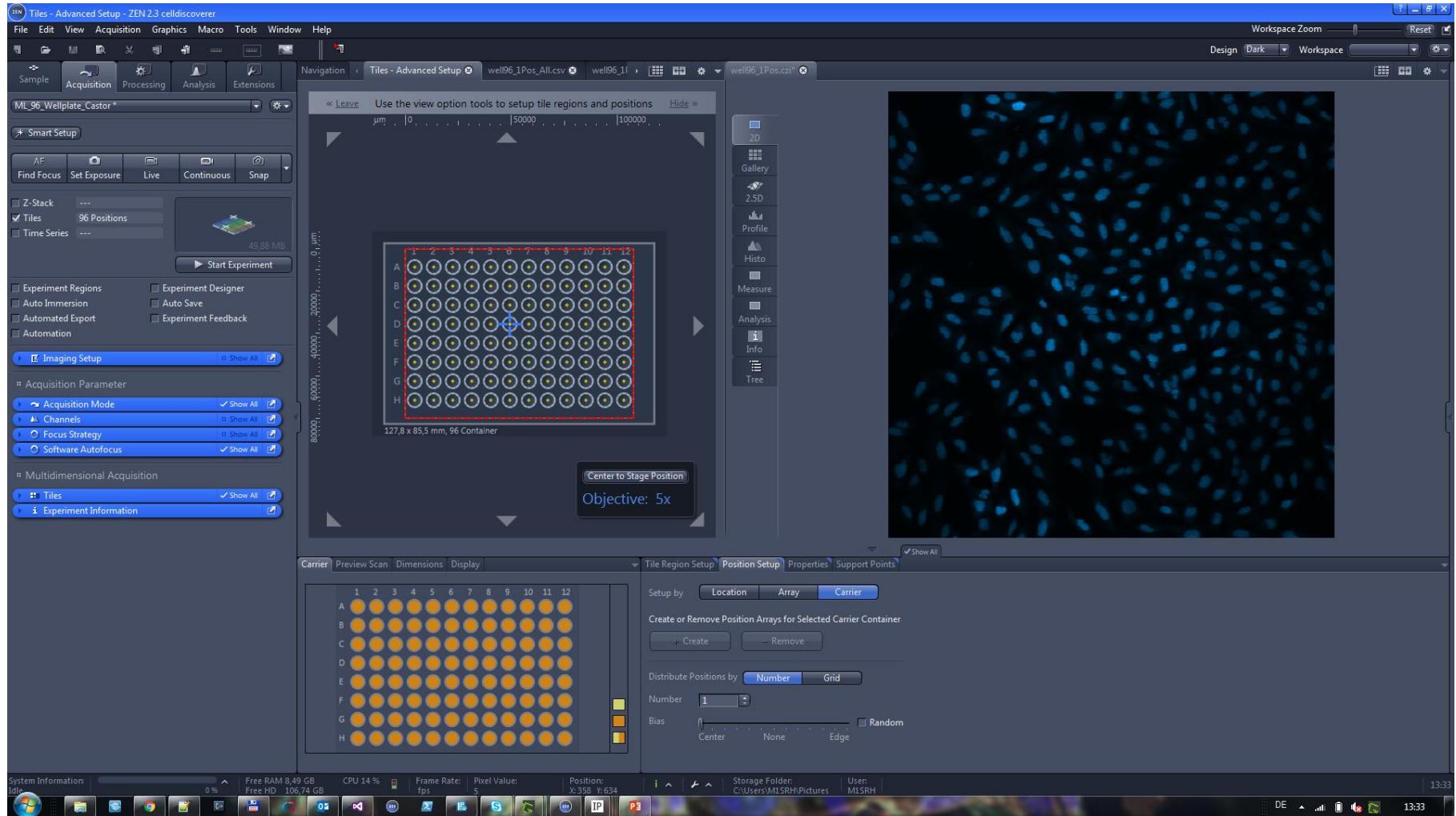
Top

Methods

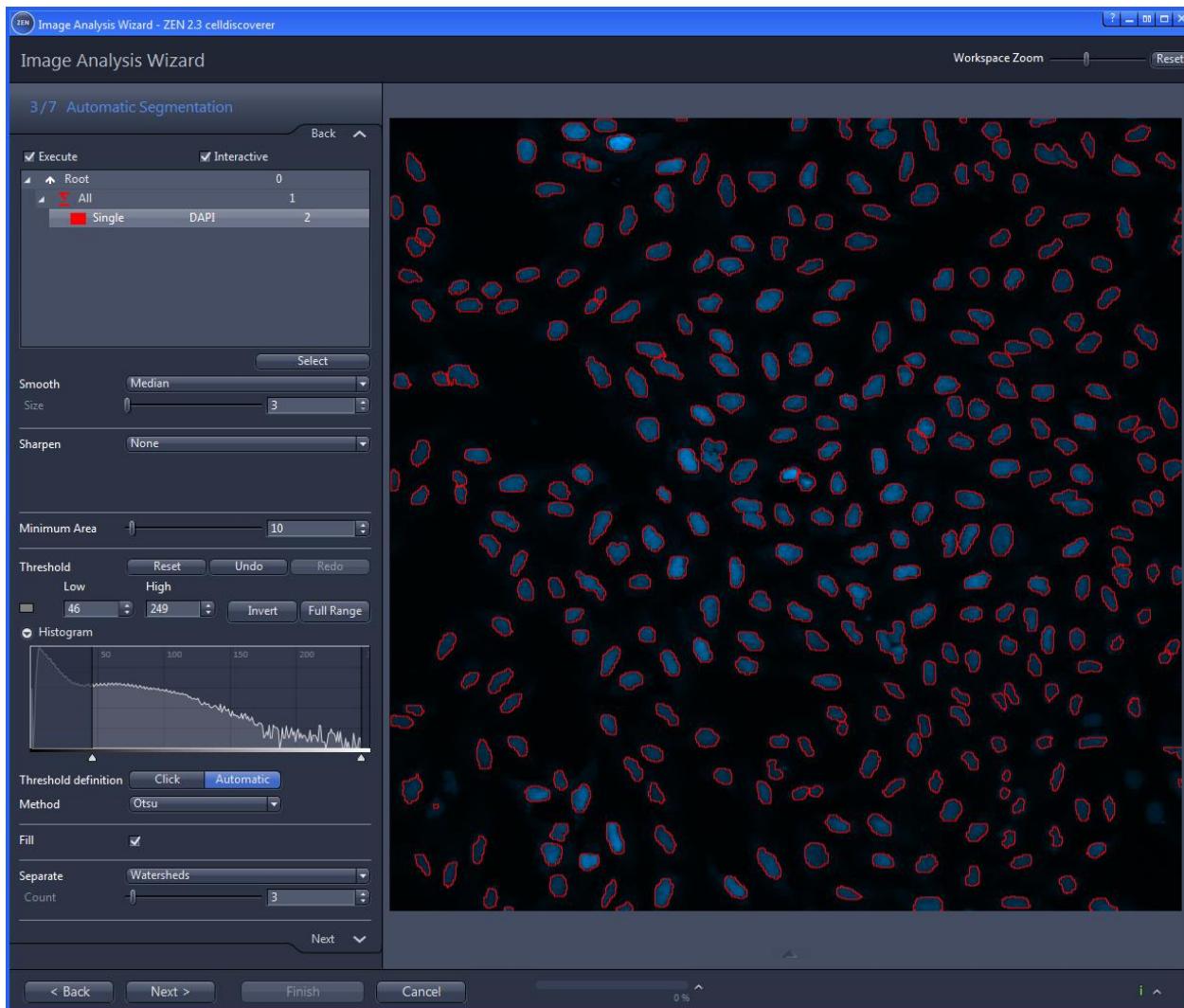
Name	Description
AcquireImage()	Acquires an image.
AcquireImage(Boolean)	Acquires an image and displays it in the ZEN application (if true).
AcquireImage(ZenExperiment)	Acquires an image of the experiment.
AutoExposure()	Calculates the exposure time.
AutoExposure(ZenExperiment)	Calculates the exposure time for all camera in the experiment.
Execute	Executes the specified experiment.
ExecuteMultiBlockImages	Executes this experiment.
ExecuteMultiImages	Executes this experiment.
FindAutofocus()	Finds the autofocus.
FindAutofocus(ZenExperiment)	Automatically finds the focus for all camera in the experiment.
FindSurface	Finds the surface.
IsLive	Returns the live state.
RecallFocus	Finds the surface + offset.
StartContinuous()	Starts the continuous. Shows the continuous image in the document collection.
StartContinuous(ZenExperiment)	Starts the continuous. Shows the continuous image in the document collection.

... yes, it is still an "old-school" CHM-file

Scripting - Automated Image Acquisition

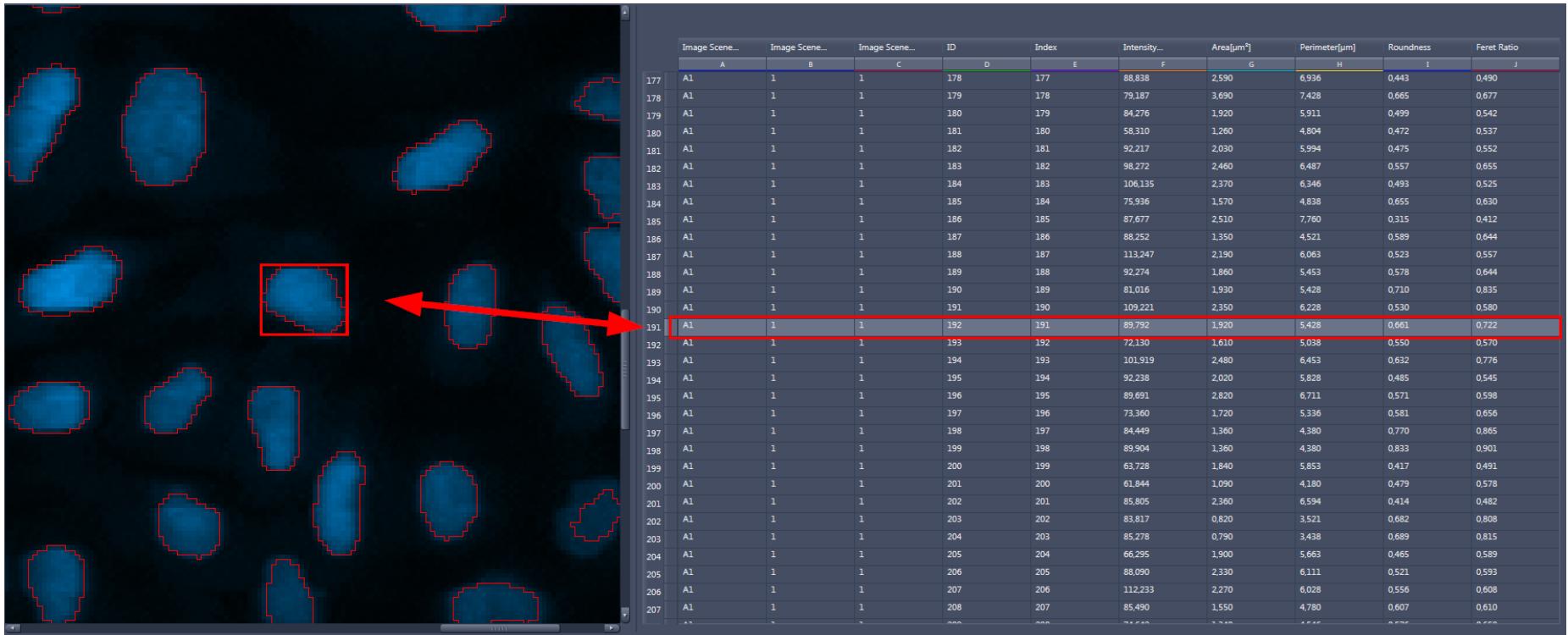


Scripting - Automated Image Analysis



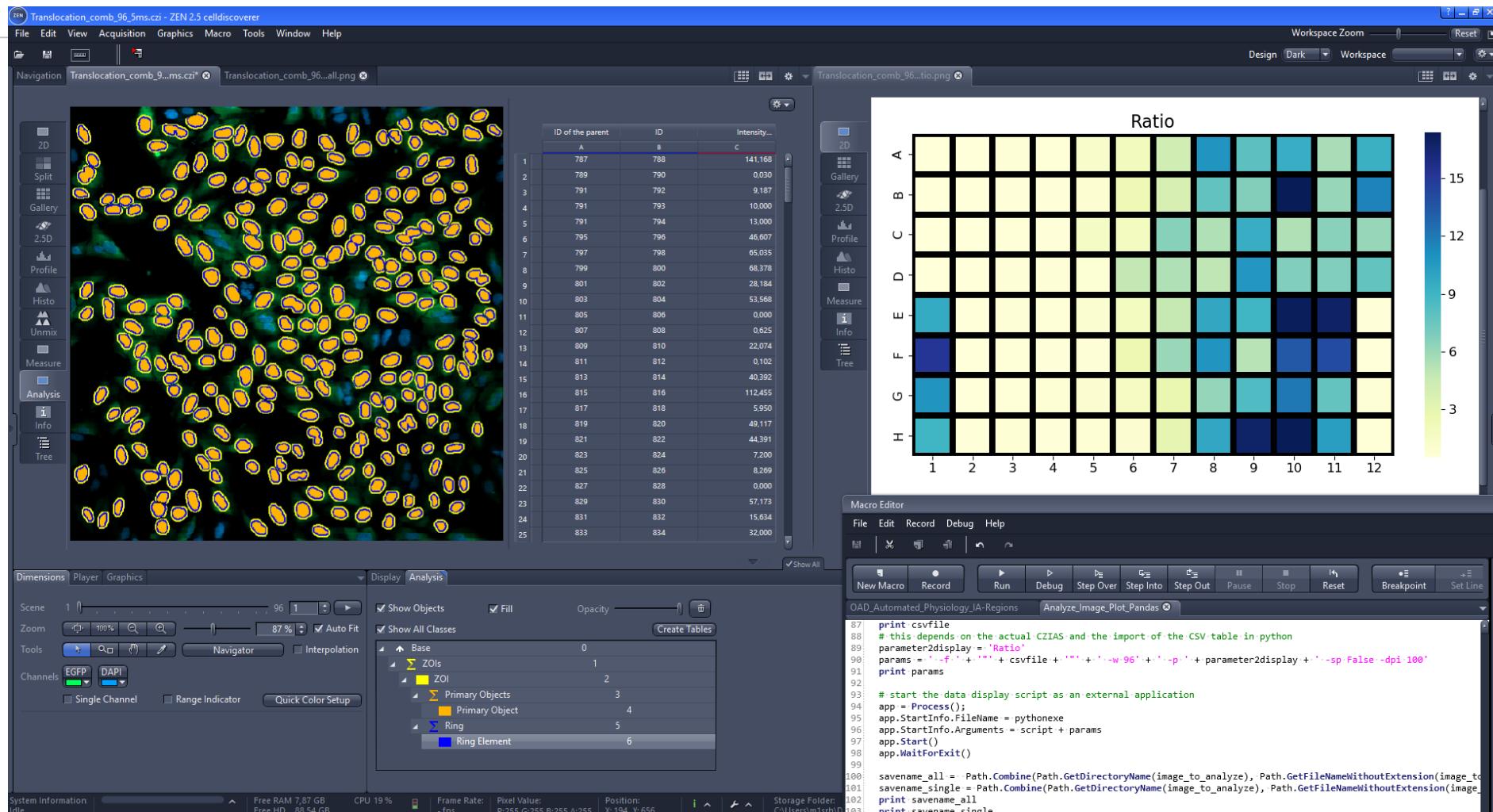
- Automated image analysis is built-in.
- Online Image Analysis is using the same pipelines.
- ZEN can do “hierarchical” measurements.
- **But where required OAD allows integration of (your own) external image analysis.**

Scripting - Access to Image Analysis Parameters



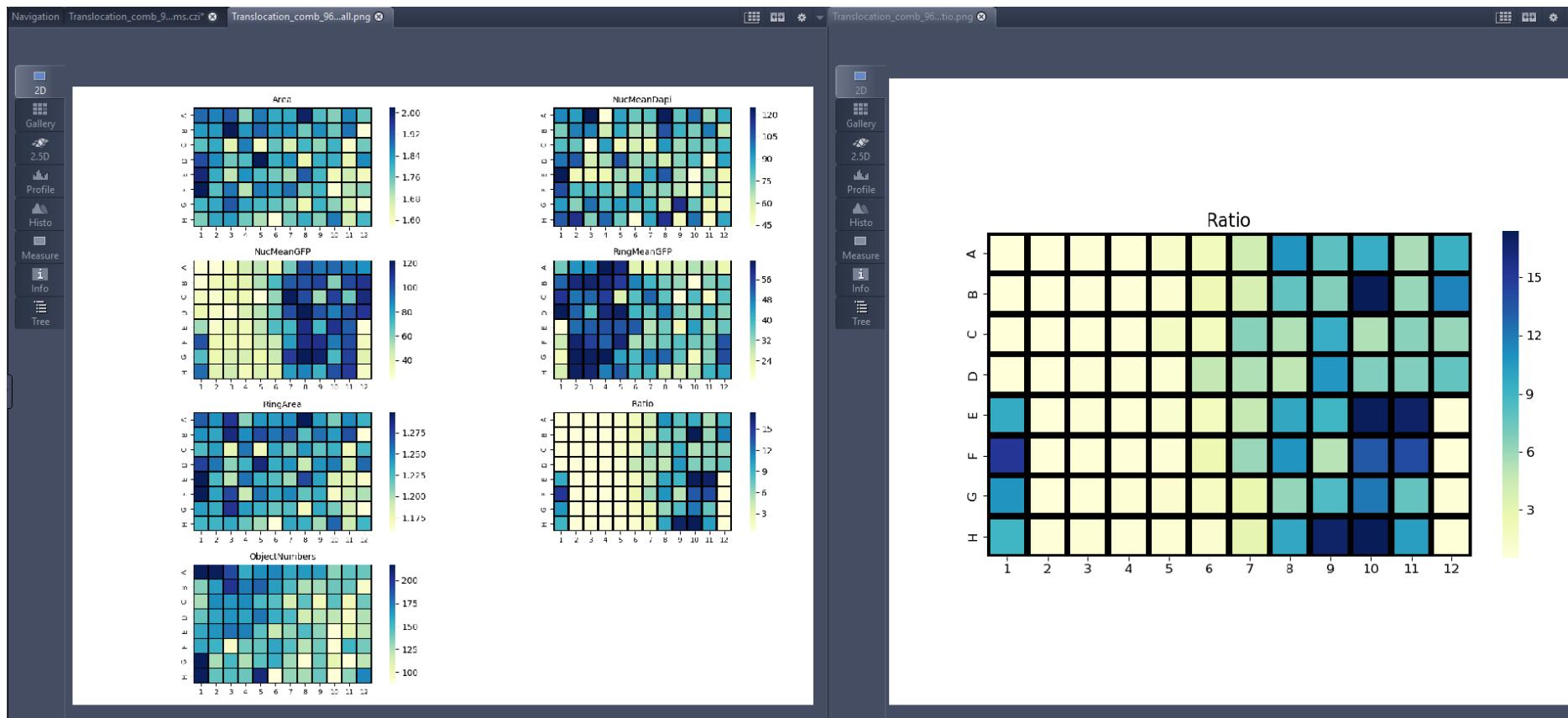
- Parameters for single objects are measured.
- Data Tables can be exported as CSV easily for usage in external applications.

OAD – Automated Image Analysis



- Scripting is used to automate the image analysis and to trigger the external visualization

OAD – Use external tools to visualize your results



- Data Tables can be exported easily for usage in external applications
- Those images are imported back into ZEN on demand

External SW is our friend – Python, MATLAB, R ...



In [6]:

```
# calculate the number of actual measurement parameters for single objects
num_param = len(df_single.columns) - num_nonmp
print('Number of Object Parameters: ', num_param)

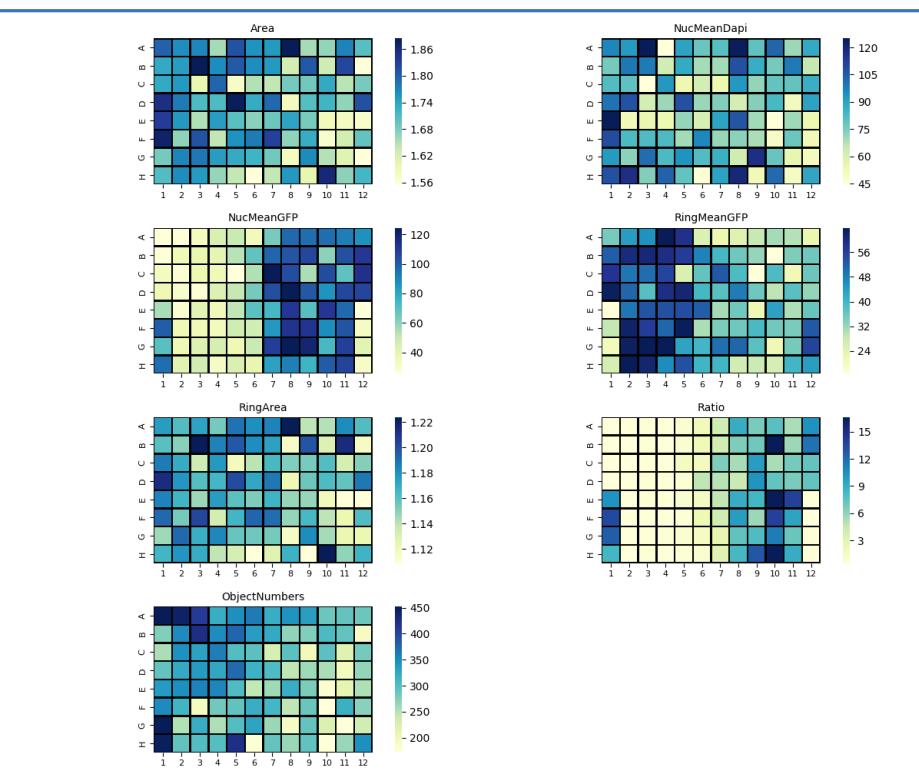
# show part of dataframe
df_single[:6]
```

Slide Type ▾

Number of Object Parameters: 6

Out[6]:

	ParentID	ID	WellID	RowID	ColumnID	Index	Area	NucMeanDapi	NucMe
1	291.0	292.0	A1	1.0	1.0	1.0	2.61	134.620690	86.
2	293.0	294.0	A1	1.0	1.0	2.0	1.52	106.664474	36.
3	295.0	296.0	A1	1.0	1.0	3.0	2.92	173.167808	9.5
4	297.0	298.0	A1	1.0	1.0	4.0	1.14	110.403509	41.9
5	299.0	300.0	A1	1.0	1.0	5.0	1.77	79.909605	0.0
6	301.0	302.0	A1	1.0	1.0	6.0	2.46	146.914634	52.1



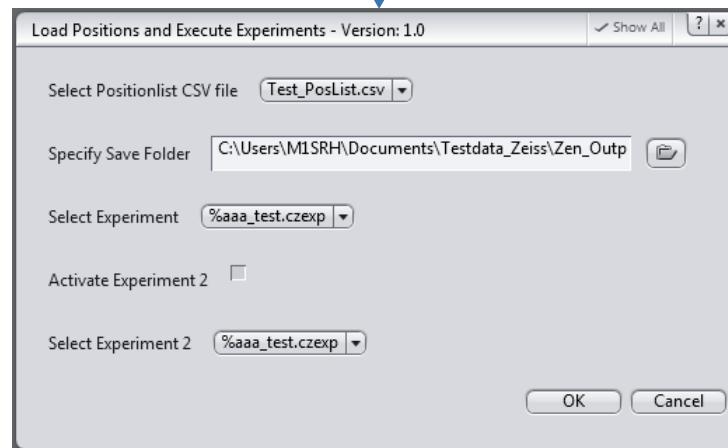
OAD Workflows – Dialogs to make life easy



- Create Position List with Tiles & Positions and export coordinates as CSV file
- Re-Import position list, specify **Multiblock Experiment(s)** and Save Folder
- Move to 1st XYZ position
- Run experiment(s) for all experiments and all blocks and move to next position



Write OAD script to create simple GUI



OAD - Adaptive Feedback Microscopy?

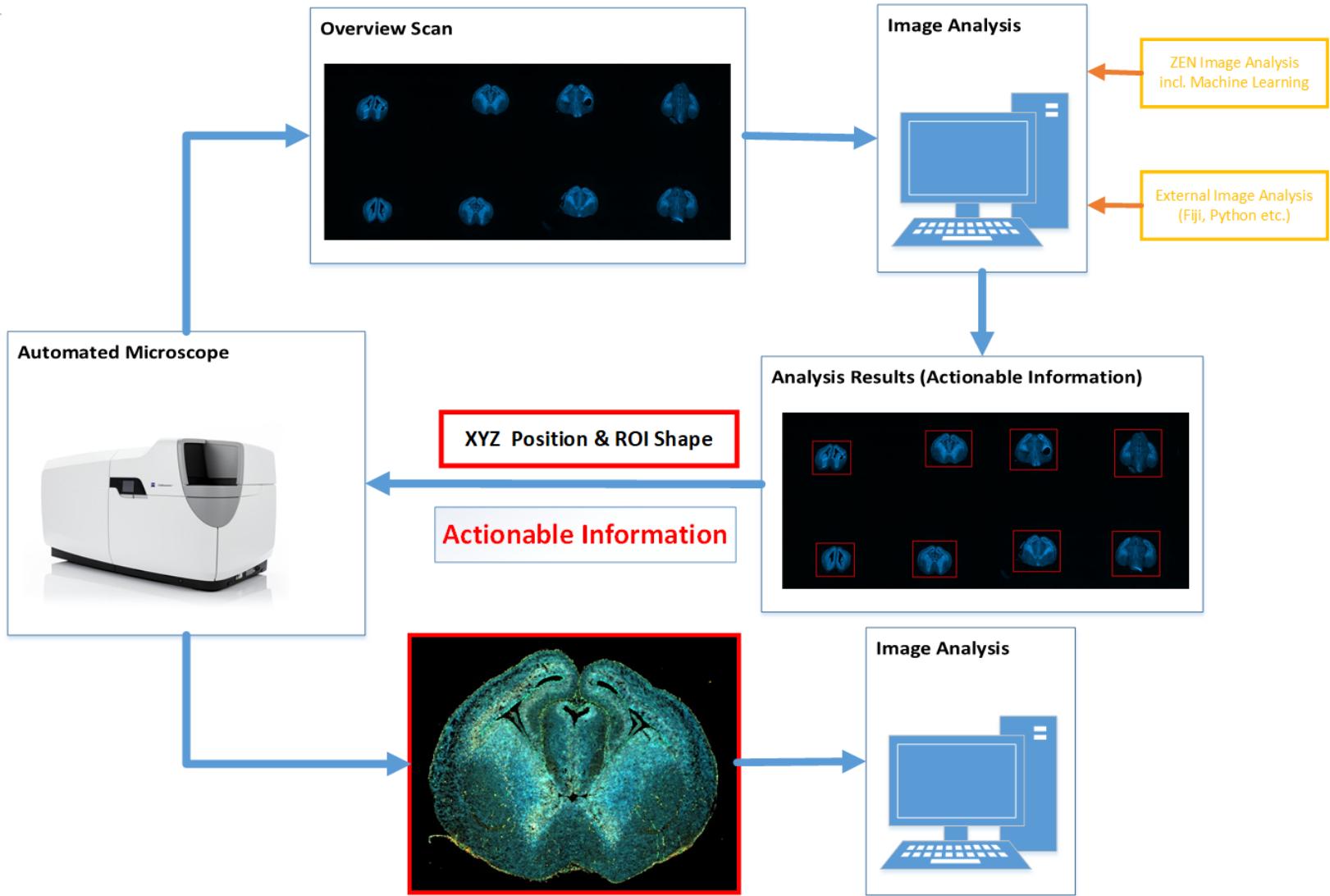


There are obviously different ways how to define **Adaptive Feedback Microscopy**, but in general one or more of the points below apply:

- **Automatically guide the system to the correct places inside a sample**
- **React on changes inside the sample during a running acquisition**
- **Optimized acquisition parameters based on the analysis of the current sample**
- **React on external signals from the “outside” and adapt the acquisition or the workflow based on those**
- **Send signal to the “outside” based on online or offline image analysis to modify the sample and continue with the workflow**

Adaptive Feedback Microscopy

Use Image Analysis to Automate the workflows



Interfaces and Tools for Adaptive Feedback Microscopy



- ZEN Image Analysis to extract Actionable Information
 - classical segmentation and Machine Learning methods can be used
- OAD Python Environment
 - powerful and flexible python scripts to automate workflows
 - Call external software and exchange data
 - Create dialogs and User Interfaces for Ease-of-Use
 - Automate Image Analysis workflows
- Experiment Feedback
 - Online Image Analysis during running experiments
 - Modification of Experiments on runtime
 - Sending or receiving signals to the “outside”
 - Powerful combinations with Experiment designer for heterogeneous acquisition workflows
- TCP/IP Interface to control ZEN from the “outside” or vice versa

Adaptive Feedback Microscopy

General Considerations



- What is the actual **nature of the desired feedback** and upon what **event** it should be triggered?
- What exactly is the **Actionable Information** to be extracted?
- On what **timescale** this feedback is required?
- Is **Online Image Analysis** available and is it sufficient to detect the feedback event?
- Which **interfaces** can be used to communicate with **external image analysis tools** or **external devices**?
- What is right **choice of hardware** and is it ready to be automated?
- What could go potentially **wrong** inside such an **automated workflow** and what be the **consequences**?

Adaptive Feedback Microscopy

Data Format Considerations



Especially in the case where external image analysis is needed the “painless” exchange of data becomes crucial!

- CZI can be read easily by many open source and commercial software packages
 - Fiji, ImageJ, Python, KNIME, Icy
 - MATLAB, Imaris, Arivis, ORS
 - ...
- Constant exchange with BioFormats team to keep **their** CZIReader up-to-date
- ZEN has option for BioFormats import to read 3rd party images (paid module)
- Zeiss offers two open available APIs to read CZI on any platform
 - libCZI (C++) for cross-platform applications (Windows, Linux, MacOS)
 - ZeissImgLib (c#) for Windows only
 - Python wrapper for libCZI is in progress

What is the right system for Adaptive Feedback?



Well, it depends from your application ...



... but all motorized ZEN Blue systems can be used

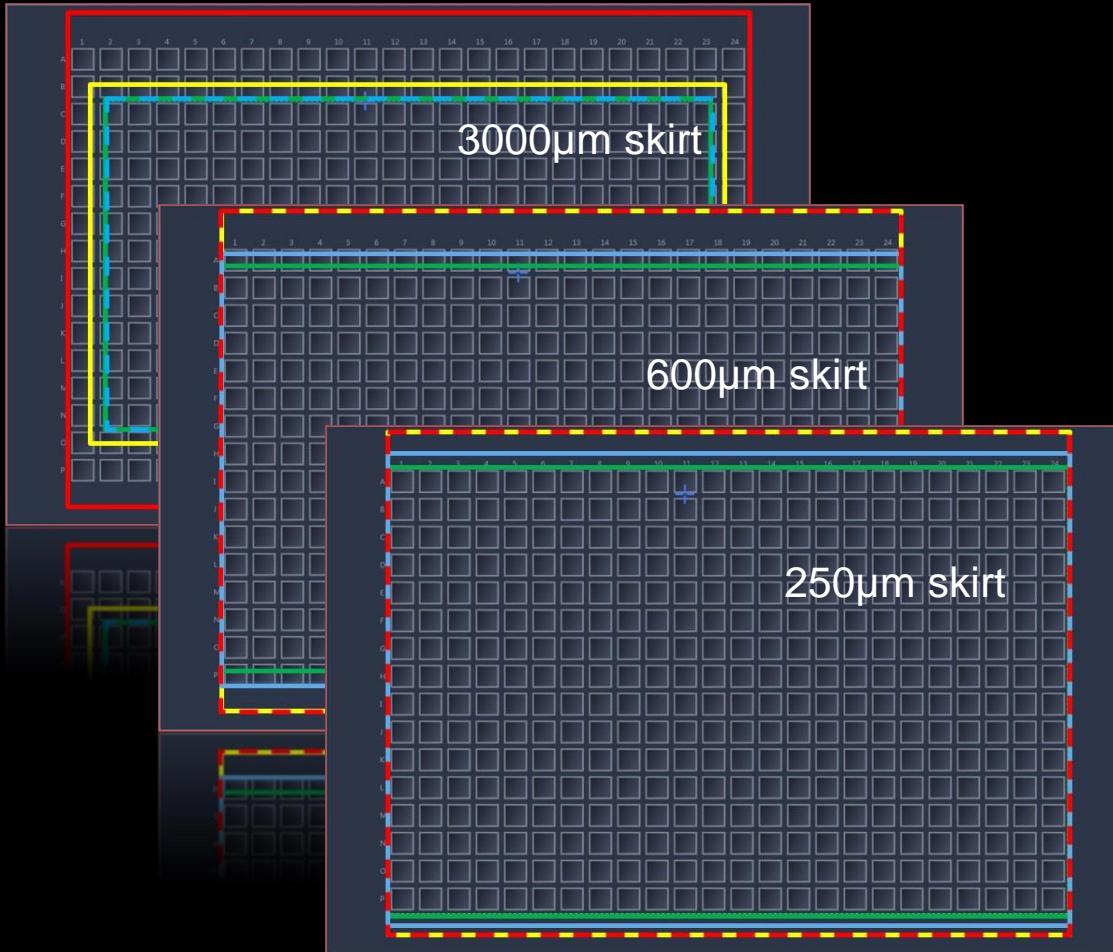
Adaptive Feedback Microscopy – Potential Pitfalls

When things go wrong ...



Adaptive Feedback Microscopy – Potential Issues

Adaptive Lens Guard of Celldiscoverer 7



- 2,5x / 0.12
- 5x / 0.25
- 10x / 0.35

- 10x/0,35
- 20x/0,7
- 40x/0,7

- 10x / 0.5
- 20x / 0.8
- 40x / 0.95

- 25x / 1.2
- 50x / 1.2
- 100x / 1.2

Celldiscoverer 7

Ready for Adaptive Feedback Microscopy and Automation



Unique Optical Concept

Unparalleled Sensitivity



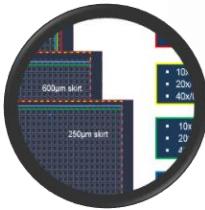
AutoCorrection

Ideal Image Quality



AutoIncubation

Enables High throughput



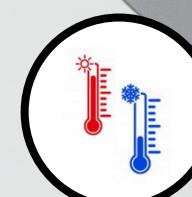
Adaptive LensGuard

Protect Your Precious Lenses



Open ZEN blue SW

Celldiscoverer 7 is Part of
Your Workflow



Auto-Carrier Recognition

Optimized Imaging Conditions



Rock-solid Incubation

Reliable long-term Timelapse
Experiments.

Phase Gradient Contrast

Adaptive Label-free Imaging

Adaptive Feedback Microscopy

Make use of scriptable Software Autofocus



SWAF.py

```
1 def runSWAF_special(expname,
2                     delay=5,
3                     searchStrategy='Full',
4                     sampling=ZenSoftwareAutofocusSampling.Coarse,
5                     relativeRangeIsAutomatic=False,
6                     relativeRangeSize=500,
7                     timeout=0):
8
9     # get current z-Position
10    zSWAF = Zen.Devices.Focus.ActualPosition
11
12    # run intial SWAF with the DetailScan experiment settings
13    SWAF_exp = Zen.Acquisition.Experiments.GetByName(expname)
14    # set DetailScan active and wait for moving hardware due to settings
15    SWAF_exp.SetActive()
16    time.sleep(delay)
17
18    # set SWAF parameters
19    SWAF_exp.SetAutofocusParameters(searchStrategy=searchStrategy,
20                                     sampling=sampling,
21                                     relativeRangeIsAutomatic=relativeRangeIsAutomatic,
22                                     relativeRangeSize=relativeRangeSize)
23
24    zSWAF = Zen.Acquisition.FindAutofocus(SWAF_exp, timeoutSeconds=timeout)
25    SWAF_exp.Close()
26
27    return zSWAF
28
29 #####
30
31 zSWAF = runSWAF_special(DetailExpName,
32                         delay=hwdelay,
33                         searchStrategy='Full',
34                         sampling=ZenSoftwareAutofocusSampling.Coarse,
35                         relativeRangeIsAutomatic=False,
36                         relativeRangeSize=SWAF_beforeDT_range,
37                         timeout=0)
```

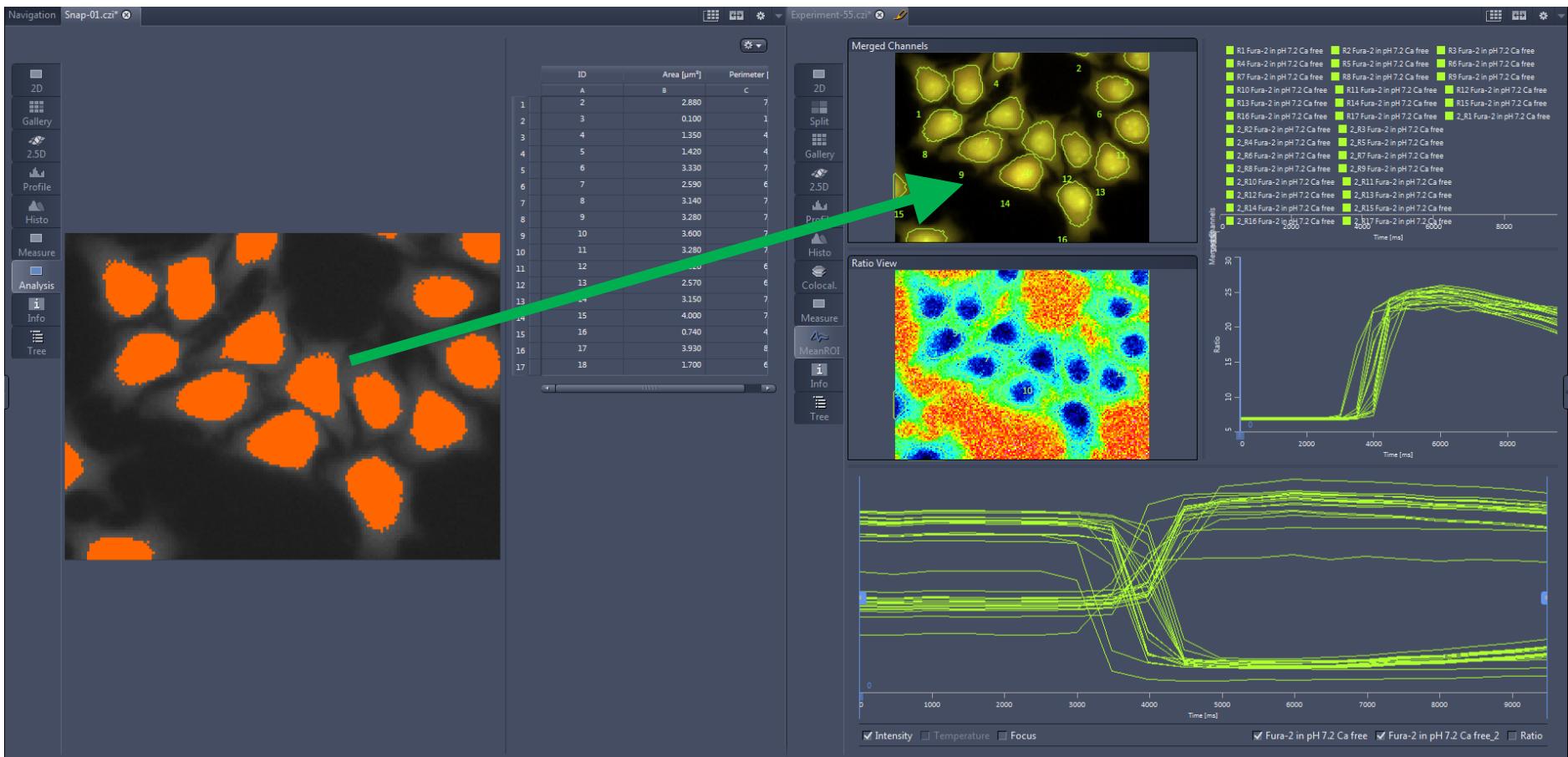
- SWAF is now completely scriptable
- Define special SWAF runs via python scripting
- Use it to define a safe starting point for a Focus Strategy

Adaptive Feedback Microscopy

Convert Image Analysis Results into Actionable Information



- Create regions for Dynamics automatically based on IA results



Adaptive Feedback Microscopy

Convert Image Analysis Results into Actionable Information



S Sample Acquisition Processing Analysis Extensions

OAD_Smart_Physiology3 *

Smart Setup Reuse

AF Find Focus Set Exposure Live Continuous Snap 00:00:09s 1.46 MB Start Experiment

Z-Stack Tiles Time Series 20 Cycles

Experiment Regions Auto Immersion Automated Image Export Automation Dynamics

Experiment Designer Auto Save Experiment Feedback Dispensing

Imaging Setup Show All

Acquisition Parameter

- Acquisition Mode Show All
- Channels Show All
- Focus Strategy
- Software Autofocus Show All
- Experiment Regions Show All

Enable Import

Keep Tool Auto Color Edit Regions in Current Image

Type ID Acquisition Analysis

Dimension X Y W H

System Information: file Free RAM: 6.7 GB CPU 17 % Frame Rate: Pixel Value: Position: Storage Folder: User: 6.7 GB 87.62 GB fps X:3223 Y:45621 C:\Users\m1srh\Document M1SRH

Navigation μm -10000 0 10000 20000 30000 40000 50000 60000 70000

Macro Editor File Edit Record Debug Help

New Macro Record Run Debug Step Over Step Into Step Out

OAD_Automated_Physiology_IA

```
122 # get all required objects to be ready to start the regions
123 exp = ZenExperiment()
124 exp.Load(expname, ZenSettingDirectory.User)
125 exp.SetActive()
126
127 # get all required objects to be ready to start the regions
128 exp = CreateRegions(exp, snap, iasname,
129 ..... expblock=0,
130 ..... accuracy='high',
131 ..... color=Colors.Yellow,
132 ..... minpoints=10,
133 ..... acquisition=True,
134 ..... bleaching=False,
135 ..... analysis=True)
136
137 # run experiment with the added regions
138 output = Zen.Acquisition.Execute(exp)
139
140 # close the experiment
141 exp.Close()
```

Watch Message Clear all

127,8 x 85,5 mm, 96 Container

Adaptive Feedback Microscopy

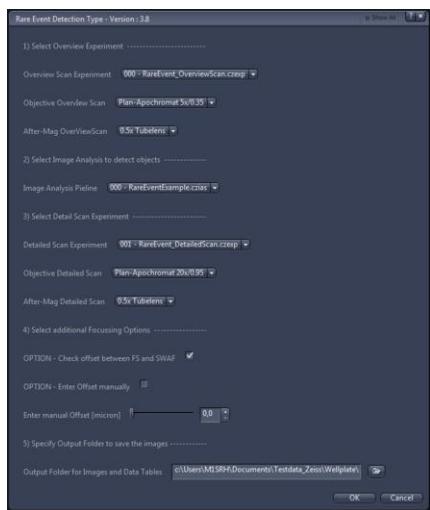
Convert Image Analysis Results into Actionable Information



- commands to modify experiments based on Image Analysis results interactively or using python scripting
- Create regions for Dynamics (Online Ratio) automatically based on segmented objects
- Use Image Analysis results (detected regions) and convert them to
 - **Experiment Regions**
 - **FRAP Regions**
 - **New Tile Regions**

Adaptive Feedback Microscopy

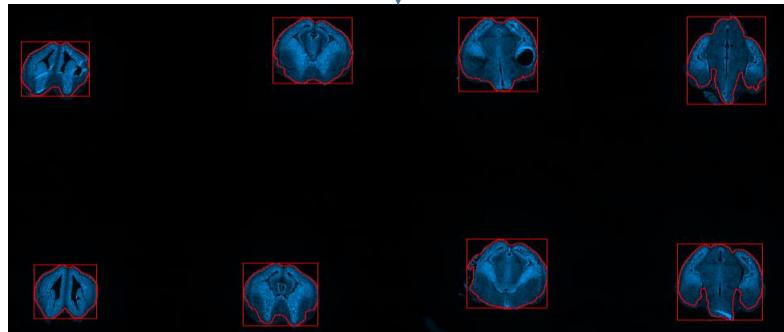
Complete workflow scheme



ID	Bound Center X Stage[µm]		Bound Center Y Stage[µm]		Bound Width[µm]	Bound Height[µm]
	A	B	C	D		
1	2	40.786,876	10.236,734	3.014,505	2.511,325	
2	3	47.835,969	10.371,678	2.996,208	2.799,510	
3	4	30.956,571	10.934,324	2.584,515	2.085,909	
4	5	56.540,981	10.604,970	2.996,208	3.320,987	
5	6	31.372,838	19.403,757	2.392,392	2.035,591	
6	7	48.176,759	18.745,048	3.046,525	2.639,407	
7	8	56.309,976	19.044,669	3.238,649	2.909,295	
8	9	39.556,373	19.508,967	2.831,530	2.383,243	

OAD – Cycle through list, modify experiment and acquire Detailed Scan

OAD - Automated Acquisition and Image Analysis

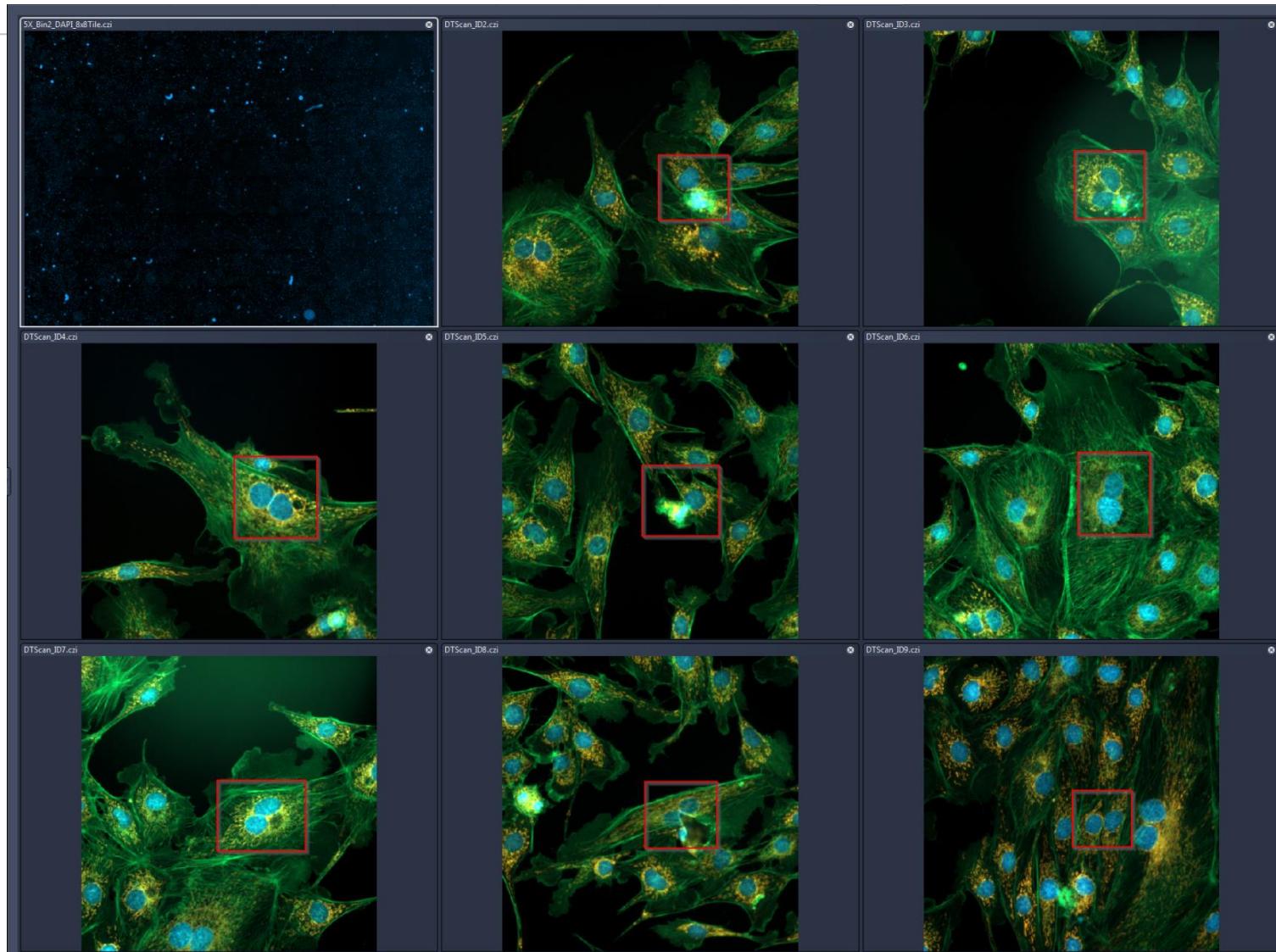


Adaptive Feedback Microscopy

Detect “Rare” objects automatically



Adaptive Feedback Microscopy



Adaptive Feedback Microscopy

Run Experiment



ZEN ZEN 2.3 system

File Edit View Acquisition Graphics Macro Tools Window Help

Workspace Zoom Reset

Design Dark Workspace

OverViewScan.czr Multi-Image-01*

OverViewScan.czr

Drag item from Images and Documents gallery

Macro Editor

New Macro Record Run Debug Help

Run_3D_Object_Detection_Fiji_Headless_special_short Run_Fiji_Headless_special_short

```
1: #!/usr/bin/python
2: File: Run_Fiji_Headless_special.czmac
3: Author: Sebastian Rhode
4: Date: 2017_12_12
5: """
6: version = 0.3
7:
8: import sys
9: # adapt this path depending on your system
10: sys.path.append(r'c:\Users\W15RH\OneDrive - Carl Zeiss AG\Projects\OAD\External_Python_Scrip
11: import ConvertTools as ct
12: import jsontools as jt
13: import Fijitools as ft
14: import cPickle as pickle
15: import time
16: from System.Diagnostics import Process
17: from System.IO import Directory, Path, File, FileInfo
18:
19:
20: # clear output console
21: Zen.Application_MACROEDITOR.ClearMessages()
22:
23: # define image
```

Watch Message

Name	Value	Type

Name Run_Fiji_Headless_special

Keywords

Description

Toolbar Configuration

Scaling: 1 px/pix (measured)

System Information: Idle

Free RAM 11,17 GB CPU 7 %

Frame Rate: 0 % Pixel Value: 0 %

Free HD 157,06 GB

Fps

Pc

C:\Users\W15RH\Documents\ZEN2.3\

Adaptive Feedback Microscopy

Run External Analysis



ZEN 2.3 system

File Edit View Acquisition Graphics Macro Tools Window Help

Workspace Zoom Reset

Design Dark Workspace

OverViewScan.czi Multi-Image-01*

C:\ProgramData\Oracle\Java\javapath\java.exe

```
09:10:23,336 |-INFO in ch.qos.logback.classic.joran.JoranConfigurator@245a060f - Registering current configuration as sa fe fallback point

SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]
[INFO] Overriding BIOP Run Macro...; identifier: command:ch.epfl.biop.macrorunner.B_Run_Macro; jar: file:///::/Users/m1srh/Do cuments/Fiji/plugins/BIOP/B_Run_Macro-1.0-0.SNAPSHOT.jar
[INFO] Overriding Save Images Tiff Without Prompt; identifier: script:ZeissIntegration\ale.ImageA_Tiff_Without_Prompt.java; jar: file:/c:/users/m1srh/documents/Fiji/jars/scijava-common-2.5.0.jar
Mär 15, 2018 9:10:26 AM java.util.prefs.WindowsPreferences <init>
WARNING: Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x80000002. Windows RegCreateKeyEx(... ) r eturned error code 5.
[INFO] Fiji Script Directory : c:\Users\m1srh\Documents\Fiji\scripts
[INFO] Start Fiji Image Analysis ...
[INFO] Fiji Python Script : c:\Users\m1srh\Documents\Fiji\scripts\GuidedAcq_fromZEN.py
[INFO] Filename : c:\Output\Guided_Acquisition\OverViewScan.czi
[INFO] BinFactor : 1
[INFO] RankFilter : MEDIAN
[INFO] Filter Radius : 3
[INFO] Threshold Method : Triangle
[INFO] Threshold Background : black
[INFO] Min. Particle Size [pixel] : 10000
[INFO] Min. Circularity : 0.01
[INFO] Max. Circularity : 0.99
[INFO] Add Particles to ROI Manager : False
[INFO] Save Particles as Image : True
[INFO] Save Format : ome.tif
[INFO] Save Results : True
[INFO] Headless Mode : True
[INFO] ----- START IMAGE ANALYSIS -----
```

Start External Application
(here Fiji in headless mode)

and read results in ZEN.

Scaling: 1 px/pix (measured) System Information: Idle Free RAM 11,17 GB CPU 7% Frame Rate: 0% Pixel Value: Fps: Description Toolbar Configuration

Adaptive Feedback Microscopy

Read Results and take Action



ZEN OverViewScan_RESULTS - ZEN 2.3 system

File Edit View Acquisition Graphics Macro Tools Window Help Workspace Zoom Reset Design Dark Workspace

OverViewScan.czi Multi-Image-01* OverViewScan_PA.ome.tif* OverViewScan_RESULTS*

OverViewScan.czi

OverViewScan_PA.ome.tif

Scaling: 1 px/px (measured) System Information: Free RAM 11,26 GB CPU 17% Frame Rate: Pixel Value: 0 % Free HD 157,06 GB fps

Macro Editor

File Edit Record Debug Help

New Macro Record Run Debug Step Over Step Into Step out Pause Stop Reset

Run_3D_Object_Detection_Fiji_Headless_special_short Run_Fiji_Headless_special_short

```
1
2 File: Run_Fiji_Headless_special.czmac
3 Author: Sebastian Rhode
4 Date: 2017_12_12
5 """
6 version = 0.3
7
8 import sys
9 # adapt this path depending on your system
10 sys.path.append(r'c:\Users\W15RH\OneDrive -- Carl Zeiss AG\Projects\OAD\External_Python_Scripts')
11 import ConvertTools as ct
12 import jsontools as jt
13 import FijiTools as ft
14 import clr
15 import time
16 from System.Diagnostics import Process
17 from System.IO import Directory, Path, File, FileInfo
18
19
20 # clear output console
21 Zen.Application.MacroEditor.ClearMessages()
22
23 # define image
```

Watch Message

Name	Value	Type

Name: Run_Fiji_Headless_special
Keywords:
Description:
Toolbar Configuration

This screenshot shows the ZEN 2.3 software interface for Adaptive Feedback Microscopy. The main window displays two sets of microscopy images: 'OverViewScan.czi' and 'OverViewScan_PA.ome.tif'. The 'OverViewScan.czi' images show grayscale brain slices, while the 'OverViewScan_PA.ome.tif' images show the same slices with specific regions highlighted in white. A 'Macro Editor' panel is open, showing a Python script for 'Run_3D_Object_Detection_Fiji_Headless_special_short'. The script includes imports for various tools and defines a variable 'version'. The 'Macro Editor' also shows a 'Run_Fiji_Headless_special_short' macro entry. A 'Watch' and 'Message' table is present at the bottom of the editor. The top menu bar includes options like File, Edit, View, Acquisition, Graphics, Macro, Tools, Window, Help, and workspace settings. System information at the bottom indicates free RAM of 11,26 GB and free HD space of 157,06 GB.

Adaptive Feedback Microscopy

Read Results and take Action



Navigation

μm | -10000 | 0 | 10000 | 20000 | 30000 | 40000 | 50000 | 60000 | 70000 | 80000 | 90000 | 100000 | 110000 | 120000

1 2 3 4 5 6 7 8 9 10 11 12

A A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12

B B1 B2 B3 B4 B5 B6 B7 B8 B9

C C1 C2 C3 C4 C5 C6 C7 C8 C9

D D1 D2 D3 D4 D5 D6 D7 D8 D9

E E1 E2 E3 E4 E5 E6 E7 E8 E9

F F1 F2 F3 F4 F5 F6 F7 F8 F9

G G1 G2 G3 G4 G5 G6 G7 G8 G9

H H1 H2 H3 H4 H5 H6 H7 H8 H9

127,8 x 85,5 mm, 96 Container

Macro Editor

File Edit Record Debug Help

New Macro Record Run Debug Step Over Step Into Step Out Pause Stop Reset Breakpoint Set Line R

```
RunFiji_Headless_GuidedAcq_Example
```

83 # start Fiji script in headless mode
84 app = Process()
85 #app.StartInfo.FileName = IMAGEJ
86 app.StartInfo.FileName = "java"
87 app.StartInfo.Arguments = fijistr
88 app.Start()
89 # wait until the script is finished
90 app.WaitForExit()
91 excode = app.ExitCode
92
93 print("Exit Code:", excode)
94 print("Fiji Analysis Run Finished.")
95
96 # read metadata JSON -- the name of the file must be specified correctly
97 md_out = jt.readjson(jsonfilepath)
98 print("ResultTable:", md_out['RESULTTABLE'])
99 # extract the relevant data
100 OVScan_CenterX = md_out['CENTERX']
101 OVScan_CenterY = md_out['CENTERY']
102 OVScan_Width = md_out['WIDTH_MICRON']
103 OVScan_Height = md_out['HEIGHT_MICRON']

Watch Message

Clear all

System Information

Free RAM 8.31 GB CPU 31% Frame Rate: Pixel Value: Position: Storage Folder:

Free HD 88.41 GB fms X:109462 Y:1747 C:\Users\mishra

Experiment Feedback

Important Features



- **Adaptive Acquisition Engine** allows modifying **running experiments** using Python scripts.
- Python Scripts can access the **current system status** and results from **Online Image Analysis** on runtime during the experiment.
- **Data Logging** or starting an **External Application** (Python, Fiji, MATLAB, ...), directly from within the imaging experiment is possible.

Experiment Feedback

Important Features



- Create dynamic acquisition-experiments
- Observe parameters during acquisition via online image analysis
- Monitor the status of the microscope and/or the sample
- Automatically react on changes of the sample or other parameters
- Modify the hardware/experiment parameters during the acquisition
 - increase integration time when the sample bleaches
 - stop the acquisition after a certain number of objects was detected
- Create custom log-files, integrate data logging in the ZEN experiment
- Start external applications (Python, Matlab,....)
- Display measurement results already during image acquisition

Experiment Feedback

New Features in ZEN 2.5



- New commands to update tiles and position list
- Integrated logging window for full control over the feedback scripts
- Updated and extensive tutorial with example for Online Assays

Script Editor for Experiment Feedback

PreLoop Script - Single Execution on Experiment Start

```
1
frame = ZenService.Experiment.CurrentTimePointIndex
2
logfile = ZenService.Xtra.System.AppendLine(str(frame))
3
ZenService.Xtra.System.WriteLine(str(frame))
```

Loop Script - Repetitive execution during experiment runtime whenever an used observable has changed

```
1
frame = ZenService.Experiment.CurrentTimePointIndex
2
logfile = ZenService.Xtra.System.AppendLine(str(frame))
3
ZenService.Xtra.System.WriteLine(str(frame))
4
ZenService.Xtra.System.WriteLine(str(frame))
5
ZenService.Xtra.System.WriteLine(str(frame))
6
ZenService.Xtra.System.WriteLine(str(frame))
7
```

PostLoop Script - Single Execution on Experiment Stop

```
1
```

Output Messages

```
Observable Experiment.CurrentTimePointIndex: Value changed from 0 to 1
AppendLogLine(1, Experiment-228_Log)
AppendLogLine(1, Experiment-228_Log)
WriteDebugOutput: 1
Observable Experiment.CurrentTimePointIndex: Value changed from 1 to 2
AppendLogLine(2, Experiment-228_Log)
AppendLogLine(2, Experiment-228_Log)
WriteDebugOutput: 2
Observable Experiment.CurrentTimePointIndex: Value changed from 2 to 3
AppendLogLine(3, Experiment-228_Log)
AppendLogLine(3, Experiment-228_Log)
```

Tools Debug

Output Messages

User Messages

Write to Output

Auto Messages

Observable Change

Action Called

Action Executed

Action Warning

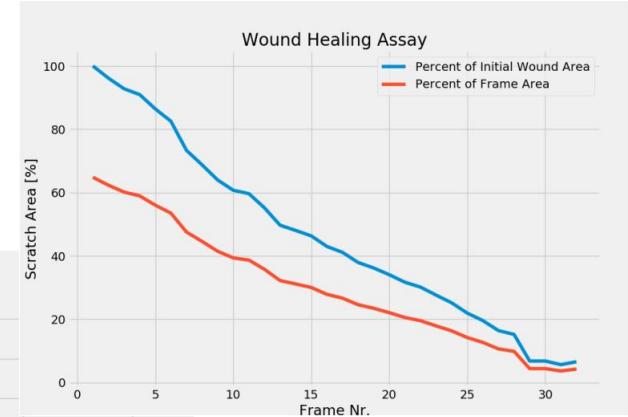
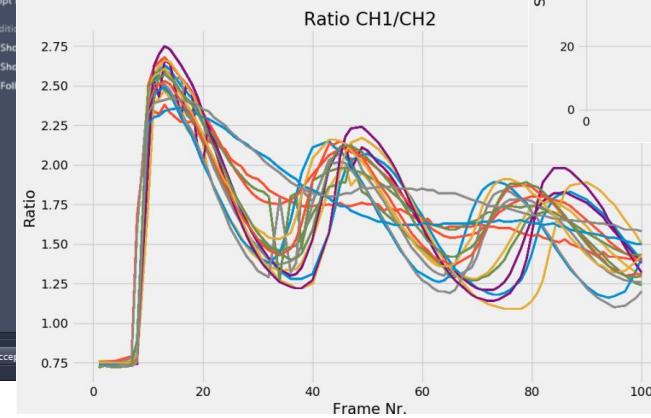
Exception

Script

Additions

Additional Output

Follow Log

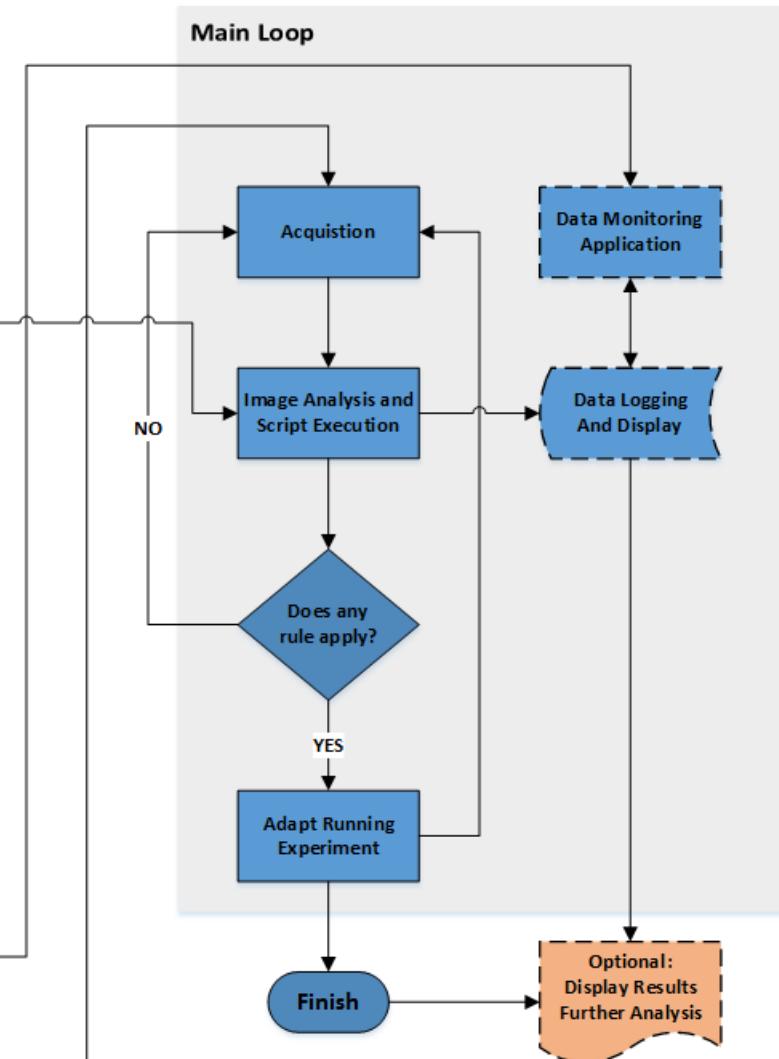
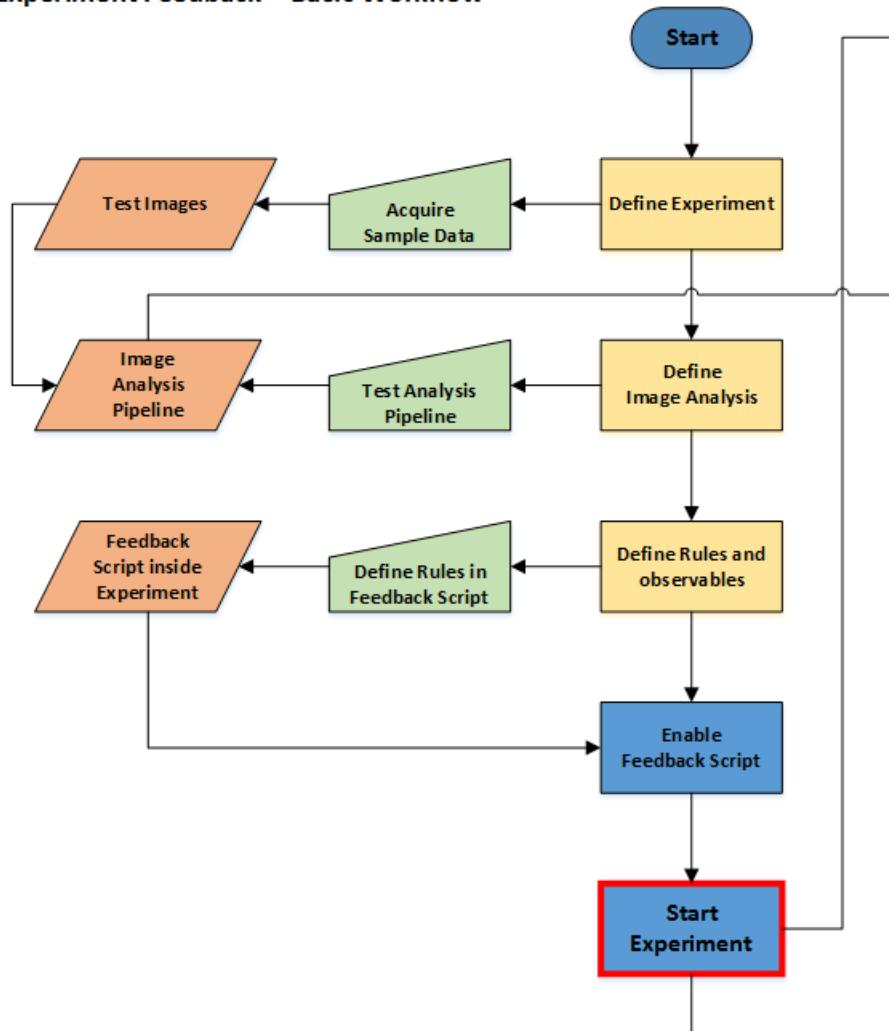


Experiment Feedback

General Workflow



Experiment Feedback – Basic Workflow



Experiment Feedback

Adaptive Acquisition Engine



Script Editor for Experiment Feedback

PreLoop Script - Single Execution on Experiment Start

```
1
Import modules, define functions or variables at
the beginning of the experiment
```

Loop Script - Repetitive execution during experiment runtime whenever an used observable has changed

```
1
Will be executed during the experiment inside a
loop automatically only when containing
observables

- React upon results from online image analysis.
- Take action upon external signals.
- Modify the experiment on-the-fly

```

PostLoop Script - Single Execution on Experiment Stop

```
1
Define actions to be executed when the
experiment stops, e.g. display the data logfile.
```

Available Observables

- Analysis
- Experiment
- Hardware
- Environment

Available Actions

- Experiment Actions
- Hardware Actions
- Extra actions

Editor Tools

- Examples & Templates

Validate Script
The script is okay.

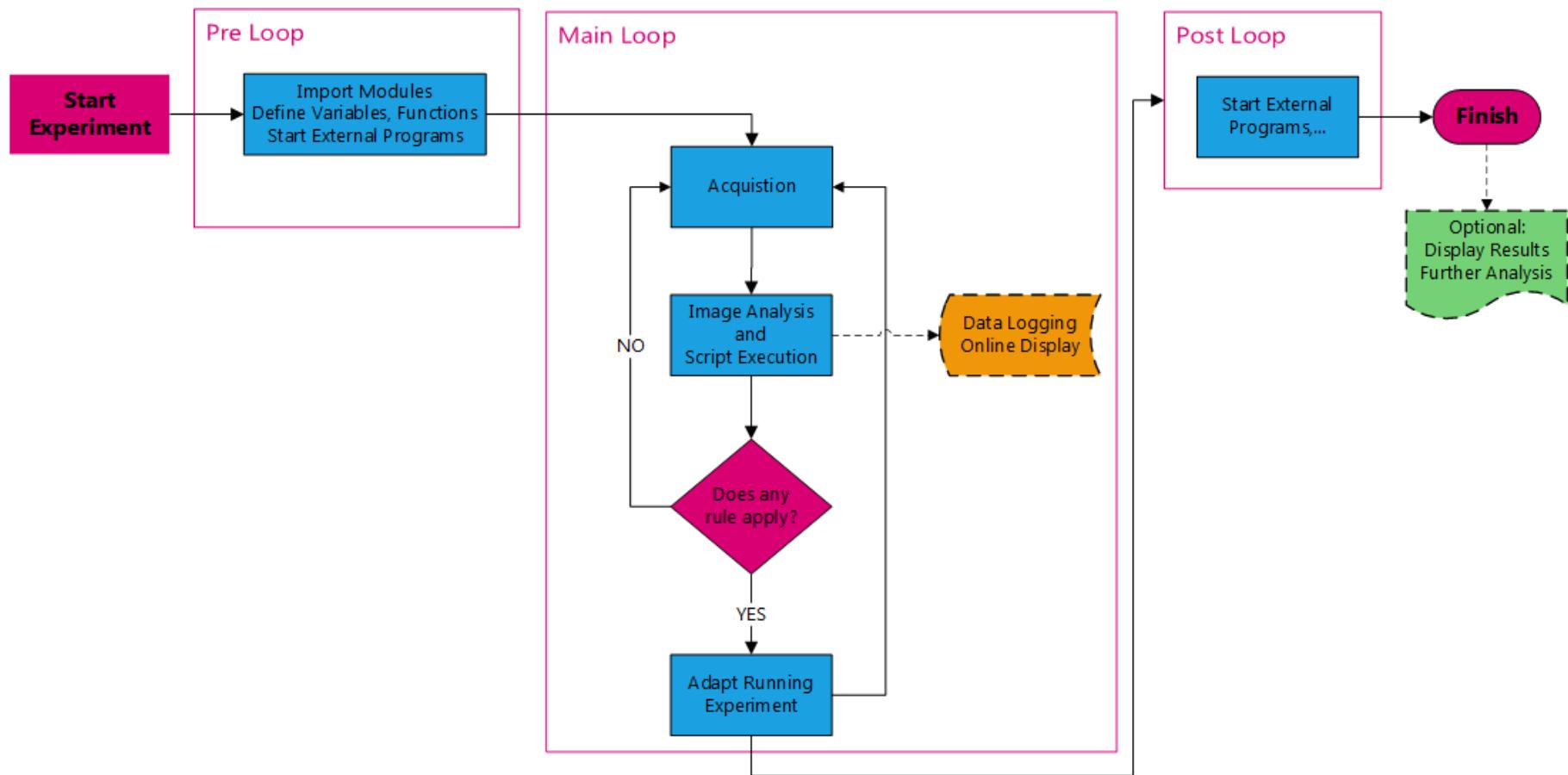
Tools Actions Observe

Select Image Analysis Pipeline

Accept Cancel

Experiment Feedback

Adaptive Acquisition Engine



Experiment Feedback

Adaptive Acquisition Engine



Script Editor for Experiment Feedback

Single Execution on Experiment Start

```
1 filename = 'f' + ZenService.Experiment.ImageFileName[:-4] + '_Log.txt'
2 # !!! watch the space before the -c in params!!!
3 params = '-c 12 -r 8' # specify well plate format, e.g. 12 x 8 = 96 Wells
4 script = r'c:\Users\MSRH\Documents\Projects\ExpFeedback_Current\Python_Scripts_for_Data_Display\dynamic.py'
5 # this is a external python application
6 ZenService.Xtra.System.ExecuteExternalProgram(script, filename + params)
```

Repetitive execution during experiment runtime whenever an used observable has changed

```
1 # get number of cells from current image
2 cn = ZenService.Analysis.AllCells.RegionsCount
3 # get the current well name, column index, row index and position index
4 well = ZenService.Analysis.AllCells.ImageSceneContainerName
5 col = ZenService.Analysis.AllCells.ImageSceneColumn
6 row = ZenService.Analysis.AllCells.ImageSceneRow
7 # create logfile
8 logfile = ZenService.Xtra.System.AppendLine(str(well) + '\t' + str(cn) + '\t' + str(col) + '\t' + str(row))
```

Single Execution on Experiment Stop

```
1 # open logfile
2 ZenService.Xtra.System.ExecuteExternalProgram(logfile, r'C:\Program Files (x86)\Notepad++\notepad++.exe')
```

Available Observables

- Analysis (Count_Cells_DAPI_96well)
- Experiment
- Hardware
- Environment

Available Actions

- Experiment Actions
- Hardware Actions
- Extra actions

Editor Tools

Examples & Templates

Validate Script

The script is okay.

Accept Cancel

Experiment Feedback Example

Count objects or measure parameters online



C:\Users\MLSRH\Documents\testdata_Zeiss\Zen_Output\temp\Experiment-27.czi - ZEN system 2012

File Edit View Acquisition Graphics Macro Tools Window Help

Workspace Zoom Reset

Design Light Workspace

Locate Acquisition Processing Analysis Reporting

Experiment Manager

Count_Cells_96well_Heatmap_dynamic_QWT

Smart Setup Show all Tools

AF Find Focus Set Exposure Live Continuous Snap

Z-Stack Tiles Panorama Time Series 72 of 96 Stop II Pause Experiment

All Channels

Setup Manager

Light Path Settings Automation

Experiment Feedback Enable Script...

Acquisition Parameter

Acquisition Mode Channels Focus Strategy Focus Devices

Multidimensional Acquisition

Experiment Designer Time Series Information on Experiment Auto Save

Applications

Physiology Correlative Microscopy Shuttle and Find

Dimensions Player Graphics

Time 1 72 72 Zoom 100% 109% Auto Fit

Tools Navigator Interpolation

Channels DAPI Single Channel Range Indicator Quick Color Setup

Histogram Spline Mode

Auto Min/Max Best Fit 2.00 0.01

Black 2 Gamma 0.80 0.45 1.0 White 222

Scaling 1.0 µm/px (Theoretical) System Information Experiment Completion... Free RAM 4.32 GB CPU 41% Frame Rate: 0 fps Pixel Value: 15 Position: X:615 Y:623 Storage Folder: C:\Users\MLSRH\Document User: mlsrh

ZEISS 14:54 23.04.2013 DE

Images and Documents Container 1 Experiment-27.czi 28.13 MB

Macro Record Run Stop Selection Preview Properties Macro Editor...

Online Data Display

96 Well Cell Count

Well A-H Well 1-12

Update speed = 2.0 (Hz)

Monitor running

The screenshot shows the ZEN system 2012 software interface. On the left, the 'Experiment Manager' panel displays a 'Time Series' of 72 frames. The 'Experiment Feedback' section is enabled. The 'Acquisition Parameter' and 'Multidimensional Acquisition' sections are visible. The 'Applications' section includes 'Physiology' and 'Correlative Microscopy'. The main workspace features a large image of a 96-well plate with blue fluorescence, overlaid with a heatmap titled '96 Well Cell Count'. Below the image is a histogram for the 'DAPI' channel, showing a distribution with a fitted curve. The bottom status bar provides system information like RAM usage and frame rate.

Experiment Feedback Example

Count objects or measure parameters online



Screenshot of the ZEN 2 Advanced Setup software interface showing the 'Tiles - Advanced Setup' window.

The main window displays a 12x8 grid of 96 well positions. A red dashed box highlights a specific region of 12 wells (A1-A3, B1-B3). A blue crosshair is positioned over the well B2-B3.

Below the grid, a status bar indicates: **Scaling: 0.10 µm/pixel (Theoretical)**, **System Information: File**, **Free RAM: 10.8 GB**, **CPU: 4 %**, **Frame Rate: 1/s**, **Pixel Value: Position X: Y:**, **Storage Folder: C:\Users\MLIRH\Documents**, **User: MLIRH**.

The left sidebar shows the 'Smart Setup' tab selected, with various experimental parameters listed:

- AF**: Find Focus, Set Exposure, Live, Continuous, Snap
- Tiles**: 96 Positions (selected)
- Time Series**: ...
- Experiment Regions**: ...
- Auto Immersion**: ...
- Automated Export**: ...
- Scripting**: ...
- Imaging Setup** (selected):
 - Acquisition Parameter**:
 - Acquisition Mode**: Show All
 - Channels**: Show All
 - Focus Strategy**: Show All
 - Software Autofocus**: Show All
 - Multidimensional Acquisition**:
 - Tiles**: Show All
 - Experiment Information**: Show All
 - Applications**:
 - Experiment Feedback**: Show All
 - Edit Feedback Script...**

Under 'Edit Feedback Script...', the 'Unbound Observer' tab is selected. A note states: "Determine spot of script run within the acquisition queue". Below this, a flowchart shows: **Acquisitions** → **Analysis** → **Script Run*** → **HD Writing**. A note at the bottom says: "* when triggered by used Observable".

The bottom right corner of the software window shows the date and time: **DE 09/05 09:05**.

Experiment Feedback

Configure analysis offline and use it online



ZEN Experiment-42 Regions - ZEN 2.3 celldiscoverer

File Edit View Acquisition Graphics Macro Tools Window Help

Navigation Experiment-42.czi* Experiment-42 Region*

Workspace Zoom Design Dark Workspace

2D
Gallery
2.5D
Profile
Histo
Measure
Analysis
MeanROI
Info
Tree

Experiment-42 Regions*

ID	Area[µm²]	Perimet
A	B	
1	7.618.900	806.835

	Image Stage...	Image Stage...	Area...	Area[µm²]	Count[#]	Image Index...	Image...
	D	E	F	G	H	I	J
1	49.500.000	35.500.000	70.056	10.642.240	1	1	00:00:00
2	49.500.000	35.500.000	68.675	10.432.420	1	2	00:00:05.2274772
3	49.500.000	35.500.000	67.755	10.292.680	1	3	00:00:05.9344065
4	49.500.000	35.500.000	67.406	10.239.700	2	4	00:00:06.9373062
5	49.500.000	35.500.000	65.667	9.975.510	2	5	00:00:07.9362063
6	49.500.000	35.500.000	65.002	9.874.510	2	6	00:00:08.9291070
7	49.500.000	35.500.000	58.697	8.916.700	1	7	00:00:09.9320067
8	49.500.000	35.500.000	57.366	8.714.450	1	8	00:00:10.9319067
9	49.500.000	35.500.000	55.730	8.465.950	1	9	00:00:11.9418057
10	49.500.000	35.500.000	54.279	8.245.570	2	10	00:00:12.9367062
11	49.500.000	35.500.000	53.015	8.053.480	3	11	00:00:13.9466052
12	49.500.000	35.500.000	52.037	7.905.010	4	12	00:00:14.9455053
13	49.500.000	35.500.000	49.753	7.557.980	2	13	00:00:15.9454053
14	49.500.000	35.500.000	50.558	7.680.260	2	14	00:00:16.9503048
15	49.500.000	35.500.000	49.978	7.592.160	2	15	00:00:17.9392059
16	49.500.000	35.500.000	50.154	7.618.900	1	16	00:00:18.9271071
17	49.500.000	35.500.000	49.309	7.490.480	2	17	00:00:19.9350063
18	49.500.000	35.500.000	48.122	7.310.280	2	18	00:00:20.9409057
19	49.500.000	35.500.000	46.822	7.112.720	4	19	00:00:21.9438054
20	49.500.000	35.500.000	46.218	7.021.060	3	20	00:00:22.9417056
21	49.500.000	35.500.000	45.531	6.916.560	2	21	00:00:23.9476050
22	49.500.000	35.500.000	45.639	6.933.110	1	22	00:00:24.9315066
23	49.500.000	35.500.000	45.798	6.957.260	3	23	00:00:25.9574040
24	49.500.000	35.500.000	44.430	6.749.390	2	24	00:00:26.9433054
25	49.500.000	35.500.000	44.363	6.739.210	3	25	00:00:27.9432054
26	49.500.000	35.500.000	44.364	6.739.390	1	26	00:00:28.9401057
27	49.500.000	35.500.000	42.900	6.516.900	3	27	00:00:29.9400057
28	49.500.000	35.500.000	43.890	6.667.320	3	28	00:00:30.9419055
29	49.500.000	35.500.000	38.556	5.857.130	6	29	00:00:31.9388058
30	49.500.000	35.500.000	39.094	5.938.780	4	30	00:00:32.9387058
31	49.500.000	35.500.000	37.088	5.634.000	3	31	00:00:33.9366060
32	49.500.000	35.500.000	36.984	5.618.720	4	32	00:00:34.9335063

System Information: 0 % Free RAM: 9.54 GB Free HD: 52.38 GB CPU: 24 % Frame Rate: -- Pixel Value: -- Position: X: - Y: - Storage Folder: C:\Users\M1SRH\Documents User: M1SRH DE ? Windows 10 13:53

Experiment Feedback

Configure analysis offline and use it online



ZEN Experiment-42.czi - ZEN 2.3 celldiscoverer

File Edit View Acquisition Graphics Macro Tools Window Help

Sample Acquisition Processing Analysis Extensions

EF_Scratch_Assay_dynamic.py3

Smart Setup

AF Find Focus Set Exposure Live Continuous Snap

Z-Stack ...

Tiles ...

Time Series 19 of 32 Stop 00m:17s Pause Experiment

Experiment Regions

Auto Immersion

Automated Export

Automation

Dynamics

Imaging Setup Show All

Acquisition Parameter

Acquisition Mode Show All

Channels Show All

Focus Strategy Show All

Software Autofocus Show All

Multidimensional Acquisition

Time Series Show All

Experiment Information Show All

Applications

Experiment Feedback Show All

Edit Feedback Script...

Select script runtime conditions

Free Run Synchronized

Define Script Slot

Acquisition → Analysis → Script Run → HD Writing

* when triggered by used observable inside the LoopScript

System Information Next in Experiment Complete 59% Free RAM 10,46 GB Free HD 52,44 GB CPU 37% Frame Rate: 1000 fps Pixel Value: Position: X: Y: Storage Folder: C:\Users\MLSRH\Documents User: MLSRH DE ? 13:49

Navigation Experiment-42.czi

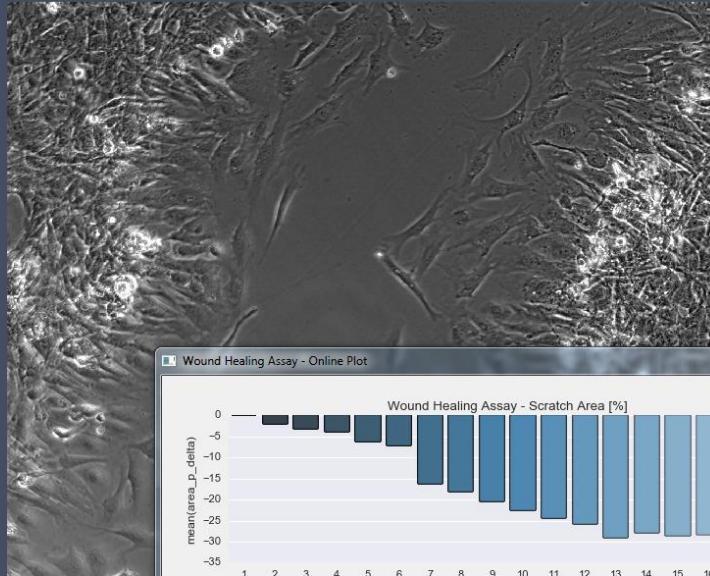
Design Dark Workspace

Workspace Zoom Reset

Images and Documents

Devices

Celldiscoverer Show All Stage Show All Focus Show All Definite Focus Incubation Auto Immersion Macro Show All



Wound Healing Assay - Online Plot

mean(area_P_delta)

Wound Healing Assay - Scratch Area [%]

frame	mean(area_P_delta)
1	-5
2	-6
3	-7
4	-8
5	-9
6	-10
7	-11
8	-12
9	-13
10	-14
11	-15
12	-16
13	-17
14	-18
15	-19
16	-20
17	-21
18	-22

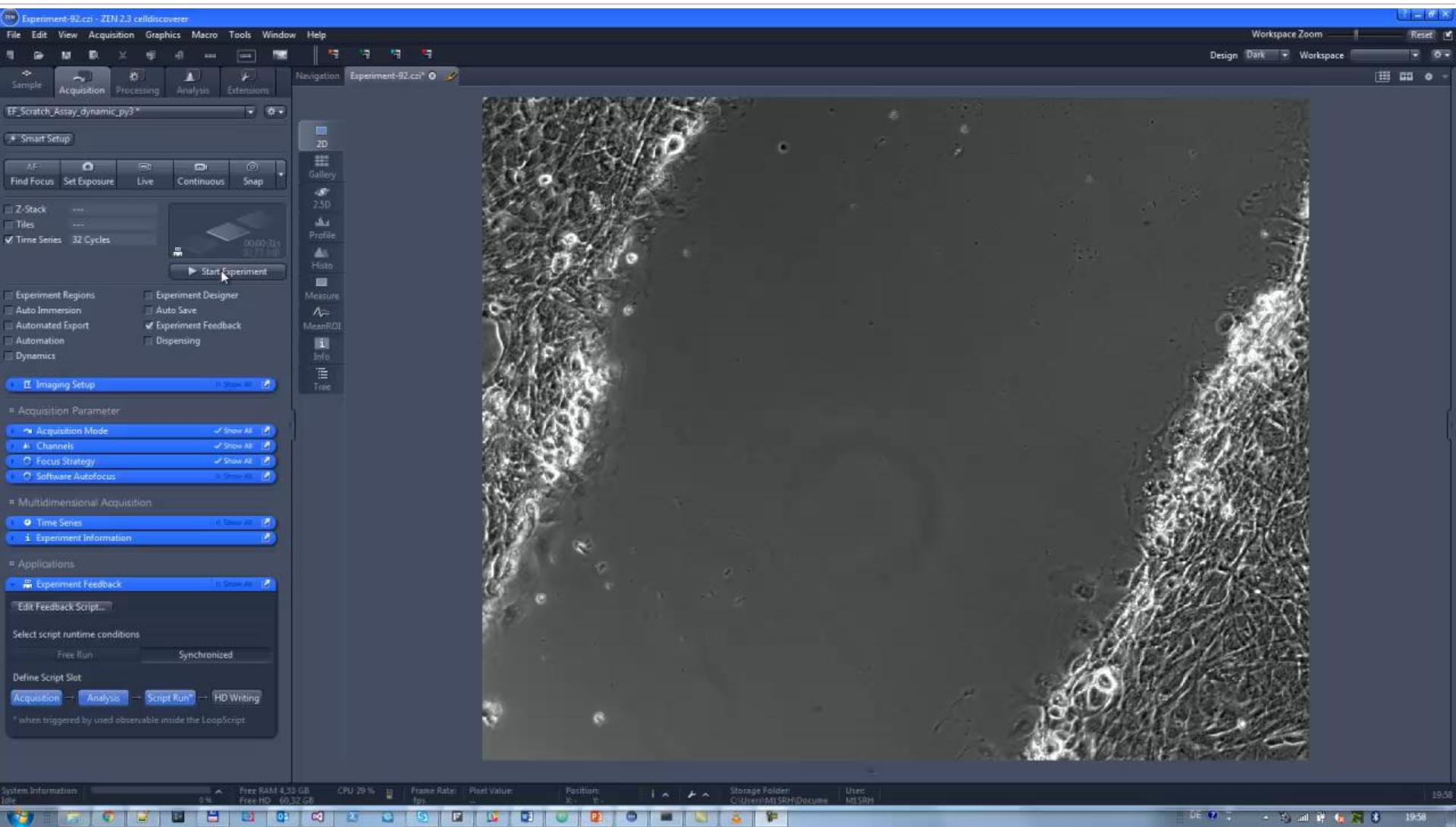
Wound Healing Assay - Scratch Relative Delta [%]

frame	mean(area_P_delta)
1	-2
2	-3
3	-4
4	-5
5	-6
6	-7
7	-8
8	-7
9	-6
10	-5
11	-4
12	-3
13	-2
14	-1
15	0
16	1
17	2
18	3

Black 1000 Gamma 0,80 0,45 1,0 White 8000

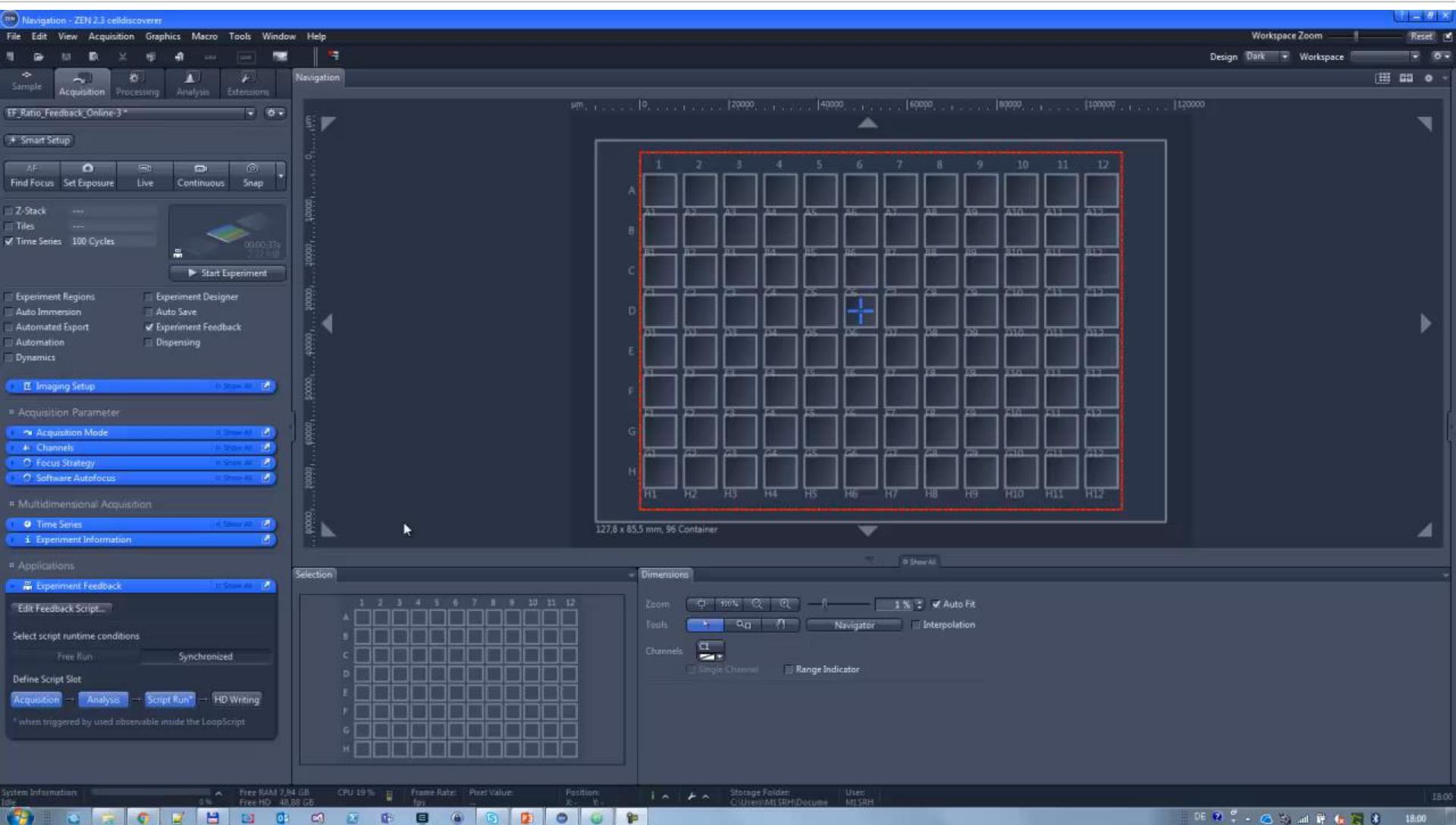
Experiment Feedback

Scratch Assay Online

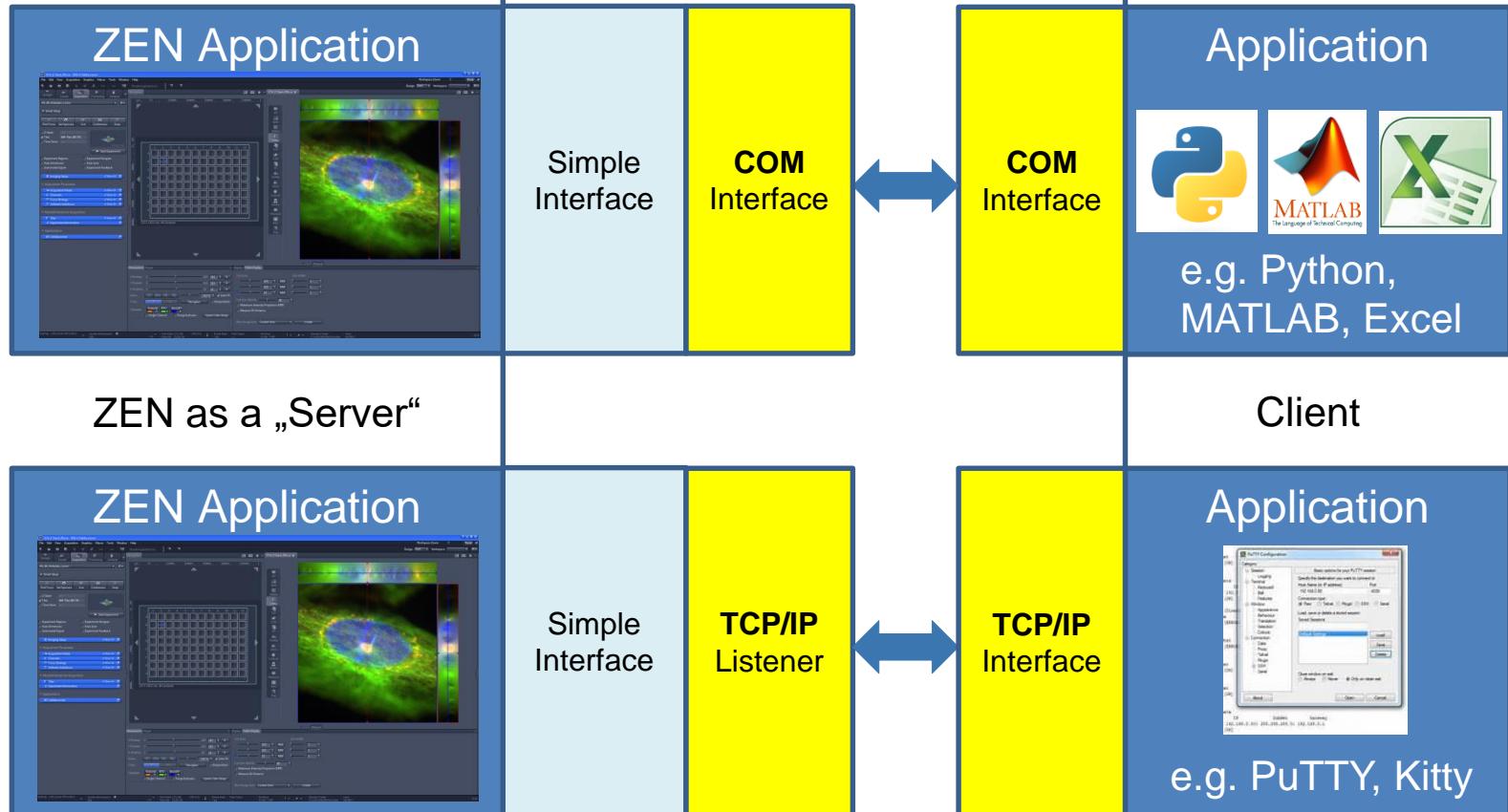


Experiment Feedback

Online Ratio



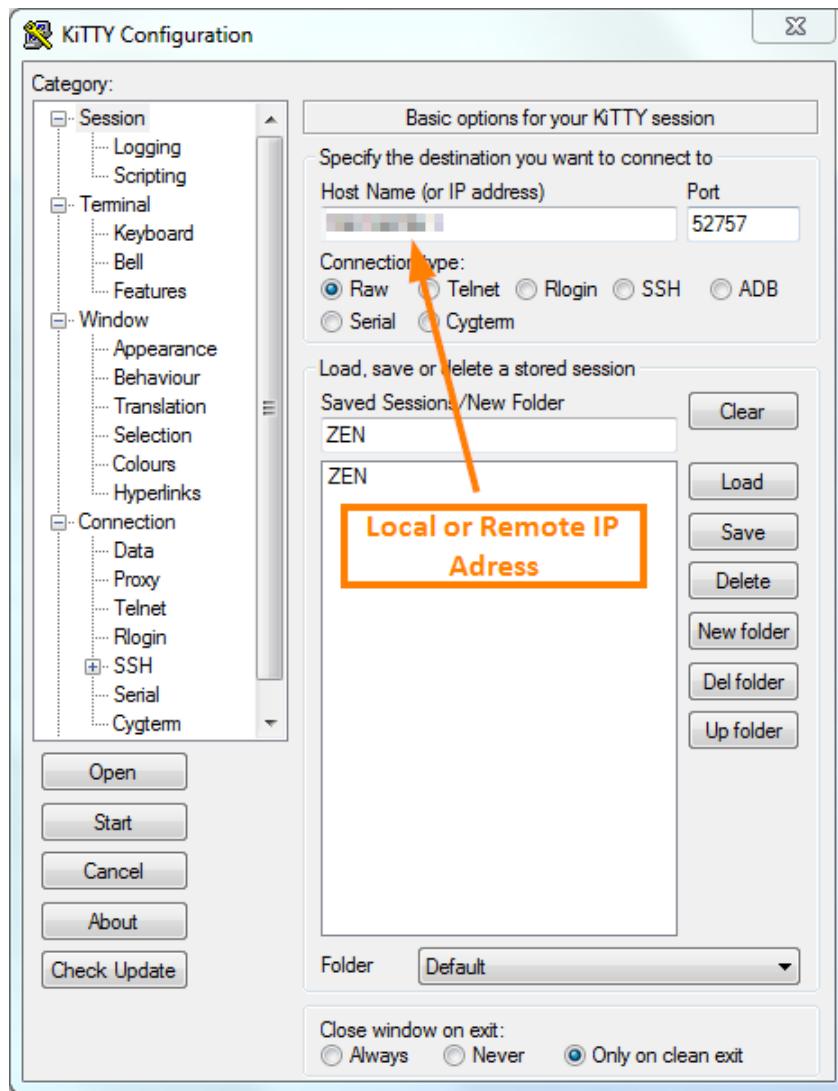
COM & TCP/IP Interface



- **TCP / IP is a general interface** to the ZEN Blue software platform allowing to **combine** ZEN functionality with **any software** on **any platform**
- can be used TCP / IP **locally** as an alternative to the COM interface (not every software has COM interface and Microsoft does not allow COM between two .NET applications for unknown reasons ...)
- can be used TCP / IP to **drive ZEN from another computer** that has an **arbitrary operating system** and **any programming language**
- **both systems have to be connected via a TCP / IP network** and the 2nd client system must be able to send and receive messages via TCP / IP

TCP / IP Interface

Simple Example



- **Kitty** is just an example for a simple client application
- local or remote IP address must be entered
- Connection must be configured correctly
- In case of the remote use case, firewalls etc. must be configured accordingly

TCP / IP Interface

Simple Example



```
Welcome to ZEN PythonScript
HELP
Commands are:
HELP           writes this help message
LIST <name>      list current/<name> macro
DEL            deletes current macro
STATUS          prints python status
PWD             print working directory
LS              lists current directory
RESET           restarts python engine
LOGFILE <file>|off    set logfile to <file>
LOGGING on|off   toggles logging to file
ECHO on|off     toggles echo of lines while debugging
VERBOSE on|off   toggles verbose mode
RUN  <name>       executes current/<name> macro
NEW  <name>       starts new macro and enters edit mode. Use END to exit editing
APPEND starts append mode on current macro. Use END to exit editing
END exits edit mode
LOAD <filename>      open macro file from disk
LOADUSER <filename>    open macro file from users Macro directory
LOADWORKGROUP <filename>  open macro file from users workgroup directory
SAVE <filename>       save script file to disk
EVAL  <python expr>    evaluates expression
CD <path>          change directory for macro files
BREAK <line>         Toggle breakpoint at <line> or current position
DEBUG <line>         executes current macro in debug mode, run until <line>, then break
CONTINUE           continues execution
STEP              steps into next statement
NEXT              executes next line
STOP              terminates script execution. Only works in DEBUG mode
LINE <line>         list lines
Ok
```

- send single OAD commands
- Load or create new OAD macros
- execute entire OAD macros

TCP / IP Interface

Simple Example: Open Image



ZEN 2.3 system

File Edit View Acquisition Graphics Macro Tools Window Help

Workspace Zoom | Reset

Design Dark Workspace

KiTTY Configuration

Category:

- Session
- Logging
- Scripting
- Terminal
 - Keyboard
 - Bell
 - Features
- Window
- Appearance
- Behaviour
- Translation
- Selection
- Colours
- Hyperlinks
- Connection
 - Data
 - Proxy
 - Telnet
 - Rlogin
 - SSH
 - Serial
 - Cygetm

Basic options for your KiTTY session

Specify the destination you want to connect to

Host Name (or IP address) 192.168.56.1 Port 52757

Connection type:
 Raw Telnet Rlogin SSH ADB
 Serial Cygetm

Load, save or delete a stored session

Saved Sessions/New Folder ZEN

Open Start Cancel About Check Update

Close window on exit:
 Always Never Only on clean exit

ZEISS

Carl Zeiss Professional Workstation

Projects & ToDo

GTD PGP process handling.pdf

ZEN 2.3

MTB 2.7.0.5 ZEN ATOMIC Build

ZEN ZEN Current ZENCoreM...

ZEN Main

ZENCoreM... Current ZEN CopyCure...

MTB 2.9.0.0 MTB 2.8.0.12

Zen 2 Core

ZEN core ZEN 2 core MTB 2.4.0.8 Kurzellet...

Scaling: 1 px/px (measured)

System Information: Idle

Free RAM 9.89 GB CPU 8 %

Frame Rate: 1205 fps

Pixel Value

kitty

Tools

Pencil oleview Autolt Debugger Texmaker ScreenCast... v2.0

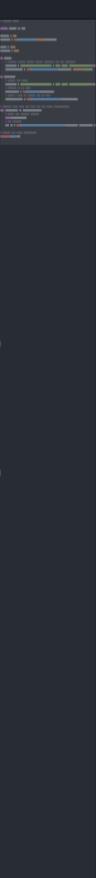
VS2015 Admin RapidEE RX64 3.3.3 Atom Oracle VM VirtualBox

Scripts

ExplorerRes... Fiji Update KillZen Update_Fiji... Scite Script Editor Kill_JEN+R... Kill_Fiji.bat KillPython... Downgrade...

TCP / IP Interface

Control ZEN from the „outside“



```
tcpip_runescript.py
1 # coding: utf-8
2
3 import tcpoad as tcp
4
5 timeout = 100
6 zentcpip = tcp.tcp_open_port(timeout=timeout)
7
8 runpy = True
9 runczmac = True
10
11 if runpy:
12     # specify python script (*.py) containing the OAD commands
13     inputfile = r'c:\Users\m1srh\OneDrive - Carl Zeiss AG\Python_Projects\ZEN_TCPIP\
14     oadcommandlist = tcp.create_oadlist_frompy(inputfile, extension='.py')
15
16 if runczmac:
17     # define OAD macro
18     inputfile = r'c:\Users\m1srh\OneDrive - Carl Zeiss AG\Python_Projects\ZEN_TCPIP\
19     # convert it to text
20     oadpyscript = tcp.read_macro(inputfile)
21     # create a list of tuples out of that
22     oadcommandlist = tcp.create_oadlist_fromczmac(oadpyscript)
23
24 # iterate over list and send the OAD macro line-by-line
25 for oadcommand in oadcommandlist:
26     # print the current command
27     print(oadcommand)
28     # and execute
29     rt, an = tcp.tcp_eval_expression_and_wait_for_ok(zentcpip, oadcommand, timeout)
30
31 # finish and close connection
32 zentcpip.close()
33
```

```
tcpoad.py
7 Date: 06.11.2017
8 Version. 1.1
9
10 Some useful TCP OAD helpers
11 """
12
13 import os
14 import telnetlib
15 import time
16 import sys
17 from xml.etree.cElementTree import Element, ElementTree
18
19
20 > def tcp_open_port(timeout=60, port=52757):=
21
22
23
24 > def tcp_eval_expression.telnet, expression):
25
26     # Evaluate the given python expression within ZEN
27     telnet.write(("EVAL " + expression).encode('ascii'))
28
29
30
31 > def tcp_eval_expression_and_wait_for_ok.telnet, expression, timeout):=
32
33
34 > def tcp_read_all.telnet):=
35
36 > def tcp_read_answer.telnet, timeout=0.1):=
37
38
39 > def tcp_clear_all.telnet):=
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
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
```

zo05n002q M1SRH 14:03:31 pm

+ C:\Users\m1srh\OneDrive - Carl Zeiss AG\Python_Projects\ZEN_TCPIP\tcpoad.py 1:1

1 CRLF 1 deprecation UTF-8 Python 0 files

Sending single commands or whole scripts from the „outside“

ZEN Remote via COM Interface – Python & MATLAB



The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Python_ZEN_Connection - [C:\Users\MI1SRH\Documents\Spyder_Projects\Python_ZEN_Connection] - ...\\RunZenfromPython.py - PyCharm ...
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Includes icons for file operations, search, and navigation.
- Project Tree:** Shows the project structure with files: RunZenfromPython.py, BioFormatsTools.py, and ReadAnalyzeImageData.py.
- Code Editor:** Displays the Python script `RunZenfromPython.py`. The code uses the `win32com` library to interact with ZEN software via COM. A yellow rectangular selection highlights the following code block:

```
## Import the ZEN OAD Scripting into Python
Zen = win32com.client.GetActiveObject("Zeiss.Micro.Scripting.ZenWrapperLM")
```

- Code Editor Features:** Line numbers, syntax highlighting, and a vertical scroll bar.
- Bottom Status Bar:** Event Log, Python Console, Terminal, and status indicators (1:1, CRLF, UTF-8).

ZEN Remote via COM Interface – Python & MATLAB



The image displays two side-by-side windows illustrating the integration of ZEN software with Python.

Left Window: Tiles - Advanced Setup - ZEN 2 system

- Top Bar:** File, Edit, View, Acquisition, Graphics, Macro, Tools, Window, Help.
- Main Area:** Shows a grid of 96 tiles labeled A through H and 1 through 12. A blue dot highlights tile D, row 5, column 6. Dimensions are indicated as 127.7 x 85.5 mm, 96 Container.
- Bottom Panel:** Carrier, Dimensions, Display. It shows a smaller grid of 96 tiles and includes tabs for Tile Region Setup, Position Setup, Properties, and Support Points. Under Tile Region Setup, it says "Setup by Contour Predefined Carrier". Buttons for Create and Remove are present, along with a Fill Factor slider set to 10.00 %.
- System Information:** Includes scaling (Automatic), pixel size (0.10 µm/px Theoretical), system RAM (2.67 GB), CPU usage (26%), frame rate (-fps), pixel value (-), position (X:-75956 Y:-2946), and storage folder (C:\Users\MI5F).

Right Window: Python_ZEN_Connection - [C:\Users\MI5RH\Documents\Spyder_Projects\Python_ZEN_Co...]

- Toolbar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Project:** Python_ZEN_Connection, RunZenfromPython.py
- Code Editor:** Spyder editor showing Python script content:

```
# coding: utf-8
# Import ...
try:
    os.makedirs(savefolder)
except OSError:
    if not os.path.isdir(savefolder):
        raise

## Define place to store the CZT file
savefolder = 'C:\\Python_ZEN_Output\\'
## check if the folder already exists
try:
    os.makedirs(savefolder)
except OSError:
    if not os.path.isdir(savefolder):
        raise

## Import the ZEN CAD Scripting into Python
Zen = win32com.client.GetActiveObject("Zeiss.Micro.Scripting.ZenWrapper")

## Define the experiment to be executed
ZEN_Experiment = "ML_96_Wellplate_Observer.cexp"

## run the experiment in ZEN and save the data to the specified folder
exp = Zen.Acquisition.Experiments.GetByName(ZEN_Experiment)
img = Zen.Acquisition.Execute(exp)

## Show the image in ZEN
Zen.Application.Documents.Add(img)

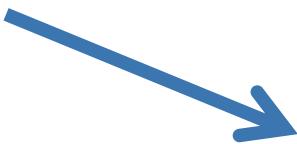
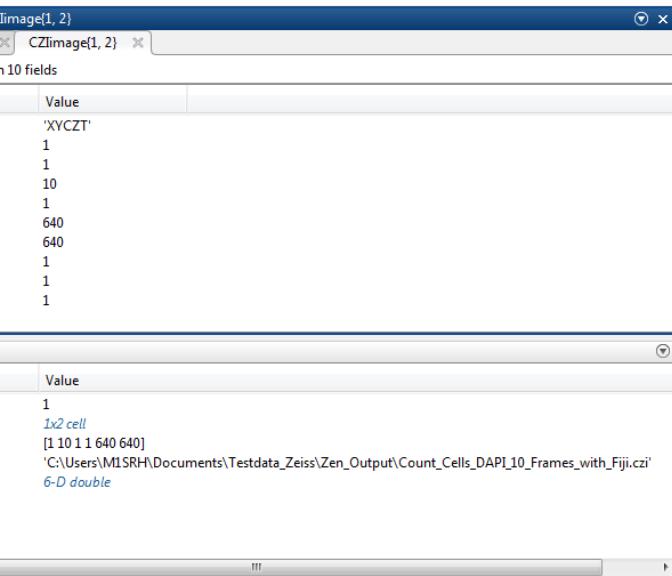
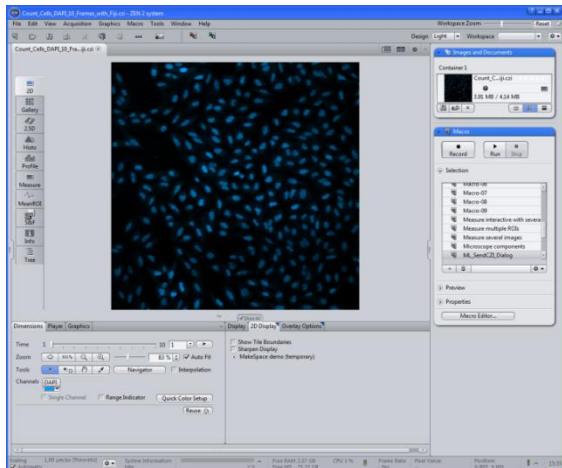
## Use the correct save method - it is polymorphic...
filename = savefolder + img.Name
img.Save_2(filename)

## set the actual CZT image data using Python wrapper for Bioformats

```

- Run:** RunZenfromPython

Run MATLAB code from within ZEN



Macro Editor

File Edit Record Debug Help

New Macro Record Run Debug Step Over Step Into Stop Reset

Analyse an image using an external mask ML_SendCZI_Dialog

```
33
34 ## get the input values and store them
35 cziname = result.GetValue('czi')
36
37 try:
38     MATLAB = Marshal.GetActiveObject('MATLAB.Application.8.5')
39     print 'ZEN-MATLAB bridge is OK.'
40     MLOK = True
41 except:
42     print 'MATLAB not running'
43
44 if MLOK == True:
45
46     ## get current active document
47     CZIfolder = os.path.dirname(CZIdict[cziname])
48     print 'Transfer:', CZIdict[cziname]
49
50     ## create MATLAB variables
51     MATLAB.execute("filename = '"+CZIdict[cziname]+"'")
52     MATLAB.execute("CZIimage = ReadImage6D(filename);")
53
54     print 'Done'
```

Watch Message

Clear all

ZEN-MATLAB bridge is OK.

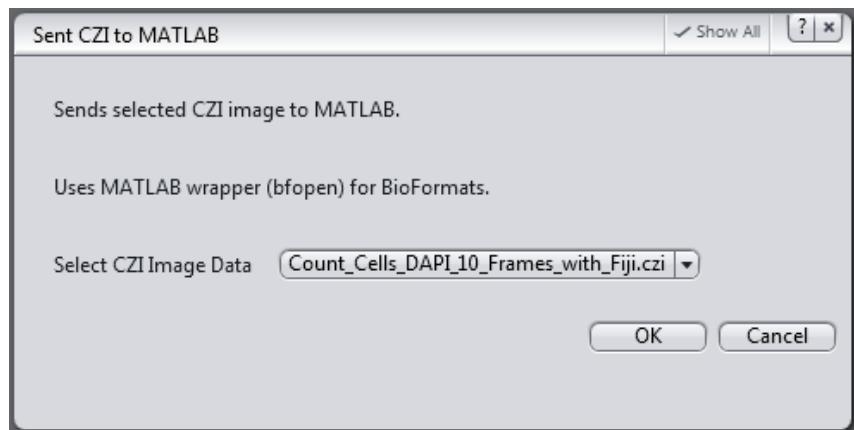
Transfer: C:\Users\MSRH\Documents\Testdata_Zeiss\Zen_Output\Count_Cells_DAPI_10_Frames_with_Fiji.czi

Done

Run MATLAB code from within ZEN



OAD Macro → Dialog



Variables - CZIimage{1, 2}

Field	Value
SeriesCount	1
SizeC	1
SizeT	10
SizeZ	1
SizeX	640
SizeY	640
ScaleX	1
ScaleY	1
ScaleZ	0
DimOrder	'XYCZT'

Workspace

Name	Value
ans	[1 1 10 640 640]
CZIimage	1x2 cell
dimorder	'XYCZT'
filename	'C:\Users\M1SRH\Do...
image	6-D double

Command Window

```
>> image = CZIimage{1};  
>> size(image)  
  
ans =  
  
1 1 10 640 640  
  
fx >>
```

Run MATLAB code from within ZEN

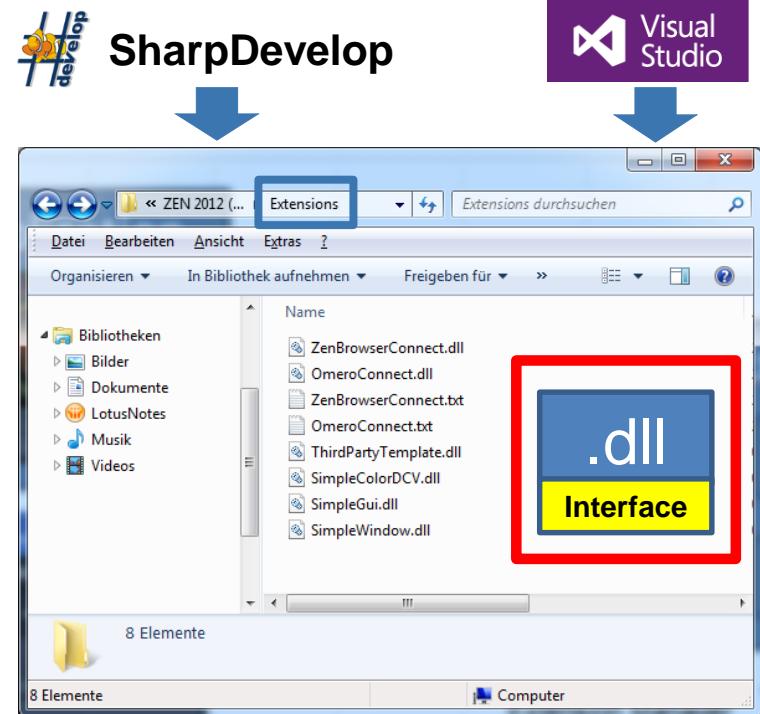
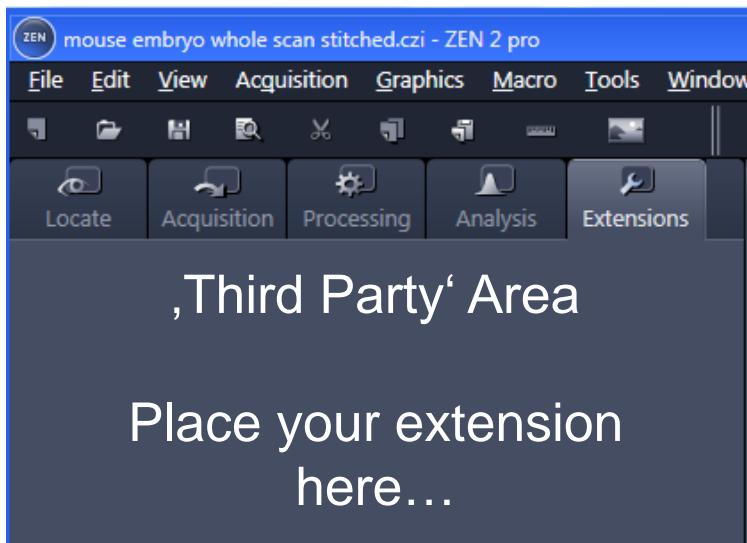


The image shows two software interfaces running side-by-side. On the left is the ZEN 2.3 software interface, which displays a fluorescence microscopy image of a cell with red and green channels. The ZEN interface includes various acquisition and analysis tools on its toolbar and a detailed control panel at the bottom for dimensions, channels, and other settings. On the right is the MATLAB R2015b interface, showing the workspace window with variables like 'ans' set to 1. Below it is the command window where MATLAB commands are being entered and executed. The MATLAB interface also features a central workspace browser and various toolbars and menus.

ZEN Extensions Interface for Advanced Topics



ZEN - LeftToolArea



ZEN - Extension Manager

Name	Description	ID	Assembly
OmeroConnect	OmeroConnect Free Demo Extension	61cd1a4c-8498-4275-b443-37191d60ad7a	OmeroConnect
SimpleColorDCV	SimpleColorDCV Free Extension	ead3f6a8-2202-4757-9ab4-5f37efebaa28	SimpleColorDCV
SimpleGui	SimpleGui Free Demo Extension	9ea07019-eedc-4101-9547-d26c156397d7	SimpleGui
SimpleWindow	SimpleWindow Free Demo Extension	21ac8925-c6b2-44bd-bcf-beb51b42d9c6	SimpleWindow
ThirdPartyTemplate	ThirdPartyTemplate Free Demo Extension	13a96a71-62f2-44bd-89c9-d1a66a74d9ee	ThirdPartyTemplate
ImageExtension	Image processing integration	fd82738e-e54f-4321-bcb7-58fe8e44cc1	Zeiss.Micro.Image.Plugin
TCP Script Service	Control ZEN scripting via TCP-server	155a8729-fa45-4242-9dd5-84c16fb4b50e	Zeiss.Micro.ScriptEditor
ZenBrowserConnect	ZenBrowserConnect Free Demo Extension	61cd1a4c-8498-4275-b443-37191d60ad7a	ZenBrowserConnect

Advanced Topics – ZEN Extensions



```
namespace MyFirstExtension
{
    using Zeiss.Micro;
    using Zeiss.Micro.Reflection;
    using Zeiss.Micro.Scripting;
    using Zeiss.Micro.Application.Configuration;
    using Zeiss.Micro.Diagnostics;

    [ApplicationExtension("6CB9D0FC-0EFA-4B74-A8D4-17411DB835CF", "MyFirstExtension",
        ExtensionFlags.Free | ExtensionFlags.SetupUI, "First Extension for Demo")]

    internal class MyFirstExtension : ApplicationExtension
    {
        readonly ZenWrapperLM zen = ZenWrapperLM.Instance;

        const string ExtensionName = "MyFirstExtension";
        const string MyControlName = "First Extension";
        MyControlViewModel myControlViewModel;

        // The functions to Initialize/Uninitialize the Extension are responsible to add and remove ZEN-specific WPF windows.
        // Therefore we have to add as many controls and their Action Classes to the project as we will need.
        protected override void InitializeCore(object application,
                                               ExtensionInitializeMode extensionInitializeMode,
                                               object parameters)
        {
            this.myControlViewModel = new MyControlViewModel();
            // add extension to ZEN when activated inside the extension manager
            UIToolItem toolAction = zen.Windows.AddToolWindow(
                ExtensionName, MyControlName, "MyFirstExtension; MyFirstExtension.MyControl", this.myControlViewModel);
            // this parameter manages whether the control will be expanded when the extension is started
            toolAction.IsExpanded = true;
        }

        protected override void UninitializeCore(ExtensionUninitializeMode extensionUninitializeMode)
        {
            // add extension to ZEN when activated inside the extension manager
            zen.Windows.RemoveToolWindow(ExtensionName, MyControlName);
        }
    }
}
```

Advanced Topics – ZEN Extensions GUI Design



MyFirstExtensio.cs MyControl.xaml X ATOMIC - Bugs [Results] Source Control Explorer

100% Design XAML

Zeiss_Micro_CoreControls:ExtendedUserControl

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Zeiss_Micro_CoreControls:ExtendedUserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="350" d:DataContext="{d:DesignInstance {x:Type MyFirstExtension:MyControlViewModel}}">
3   <Grid>
4     <Button Content="Button" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="115,46,0,0" Width="161" Height="64" />
5     <Zeiss_Micro_CoreControls:EditableComboBox HorizontalAlignment="Left" Margin="79,198,0,0" VerticalAlignment="Top" Width="172">
6       <CheckBox Content="Checkbox 1"/>
7     </Zeiss_Micro_CoreControls:EditableComboBox>
8     <Zeiss_Micro_CoreControls:FileChooserControl HorizontalAlignment="Left" Margin="40,147,0,0" VerticalAlignment="Top" Width="211"/>
9   </Grid>
10 </Zeiss_Micro_CoreControls:ExtendedUserControl>
```

100 %

ZEN Extensions - Examples



The screenshot displays the ZEN 2 software interface with several extension examples open:

- OmeroConnect**:
 - Image Transfer**: Includes a "Send image to OMERO" button and a checkbox for "Delete local image".
 - Setting**: Shows the path to OMERO.dropbox folder as C:\Omero\OMERO_dropbox, with a "Save Setting" button.
- SimpleGui**:
 - Minimal User Interaction**: Contains a "Text" input field, two checkboxes for "Check A" and "Check B", and an "Ok" button.
 - SimpleWindow**:
 - Action**: Includes a "Show Image" button.
- ImageJ**:
 - Apply**
 - Method**
 - Recently used**: A list of recently used items.
 - Search**: A search bar.
 - jars**, **macros**, **AutoRun**, **Sebi_Macros**, **tools**, **toolsets**, **Zen_Test**, **Batch BioFormats**: Hierarchical lists of files.
 - Parameters**: A section with a "Show All" checkbox.
- ZenBrowserConnect**:
 - Image Transfer**: Includes a "Send image to ZenBrowser" button and a checkbox for "Delete local image".
 - Setting**: Shows the path to ZenBrowser drop folder as C:\ZENDB\drop\00000001_Category\00000002_Document Clas:, with a "Save Setting" button.
- Rapp OptoElectronic**:
 - Toolbox**: Includes "Image Transfer" and "Import ROIs" buttons.

3rd party usage is appreciated



We make it visible.

www.zeiss.com/czi

www.zeiss.com/zen-oad

Tutorials

Application Notes

Examples

<https://github.com/zeiss-microscopy/OAD>

<http://github.com/zeiss-microscopy/libCZI>