

# Experiment Feedback



**Dr. Marion Lang**  
Product Management  
2018-10-05

# ZEN is only part of the workflow



Scripting

**OAD**

Python

MATLAB

Excel

**ZEN blue**

KNIME

**Analysis**

Fiji

Automation

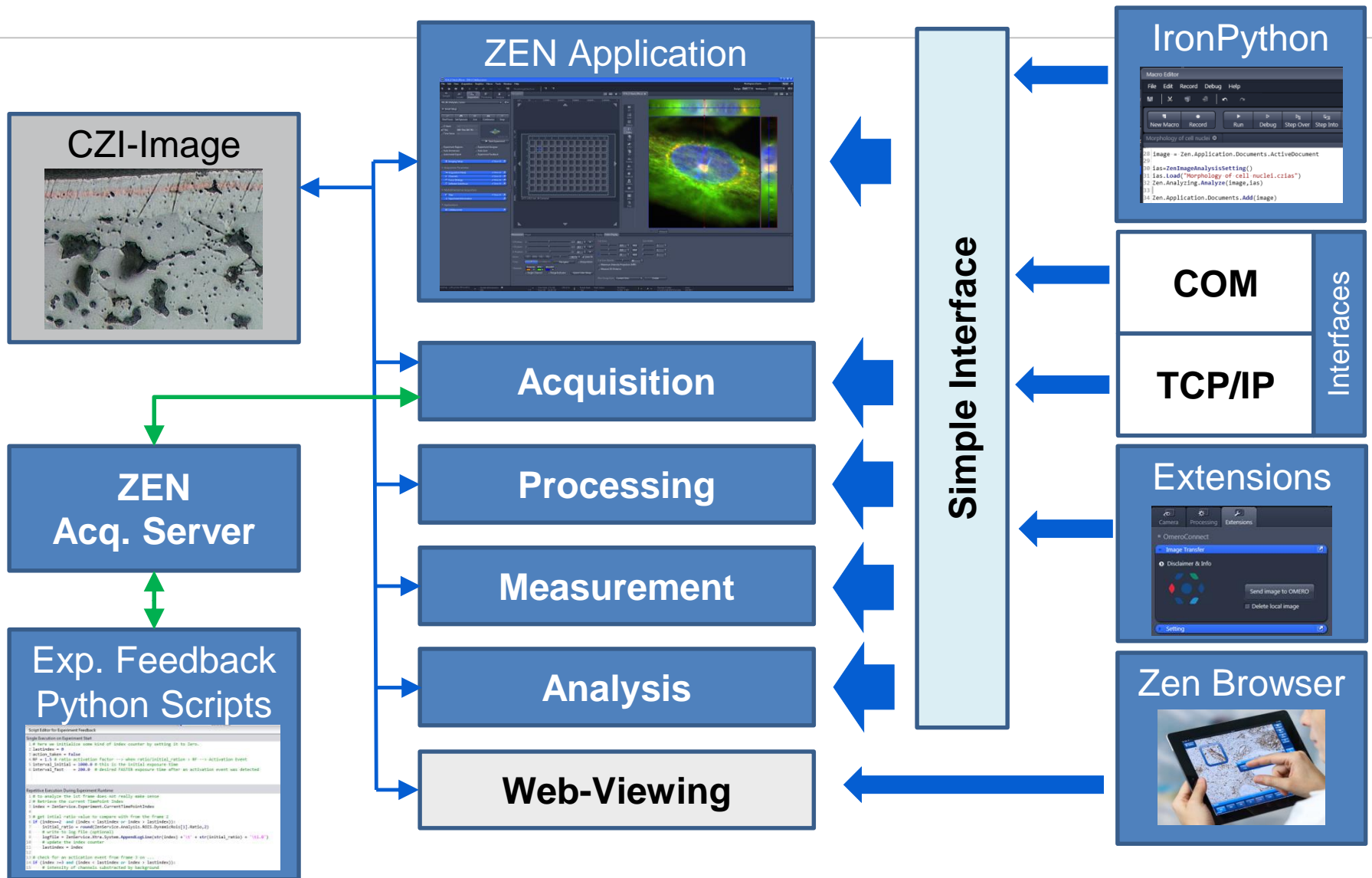
Machine Learning

**Extensions**

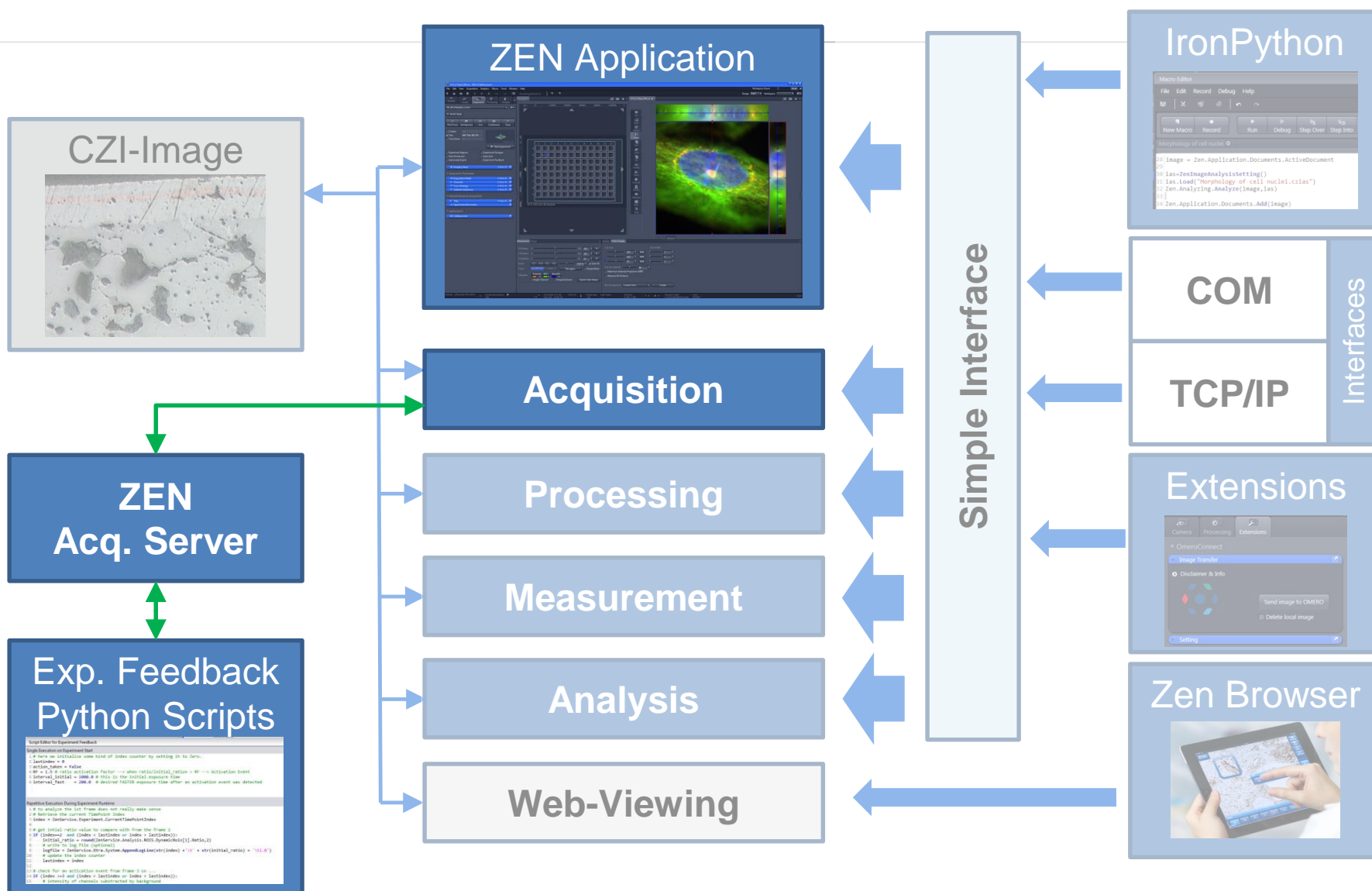
**Simplify**

- 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

# OAD – ZEN Interfaces

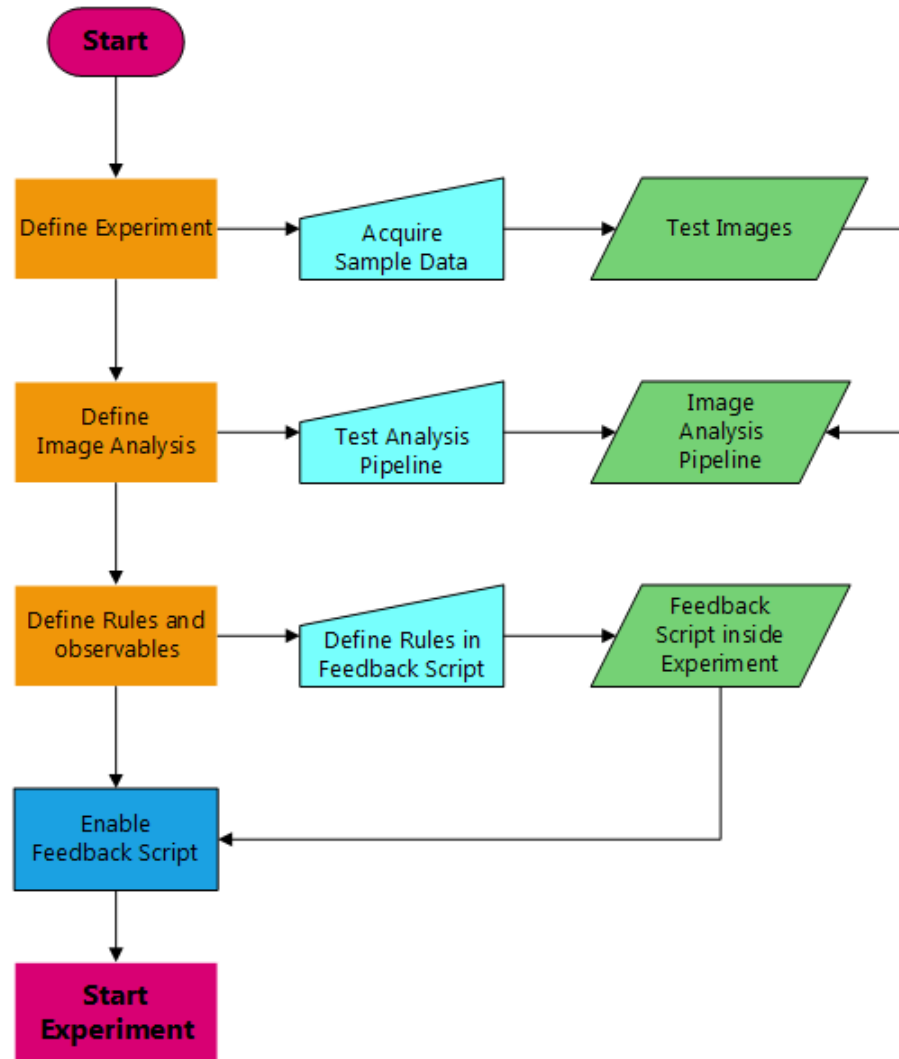


# OAD – ZEN Interfaces



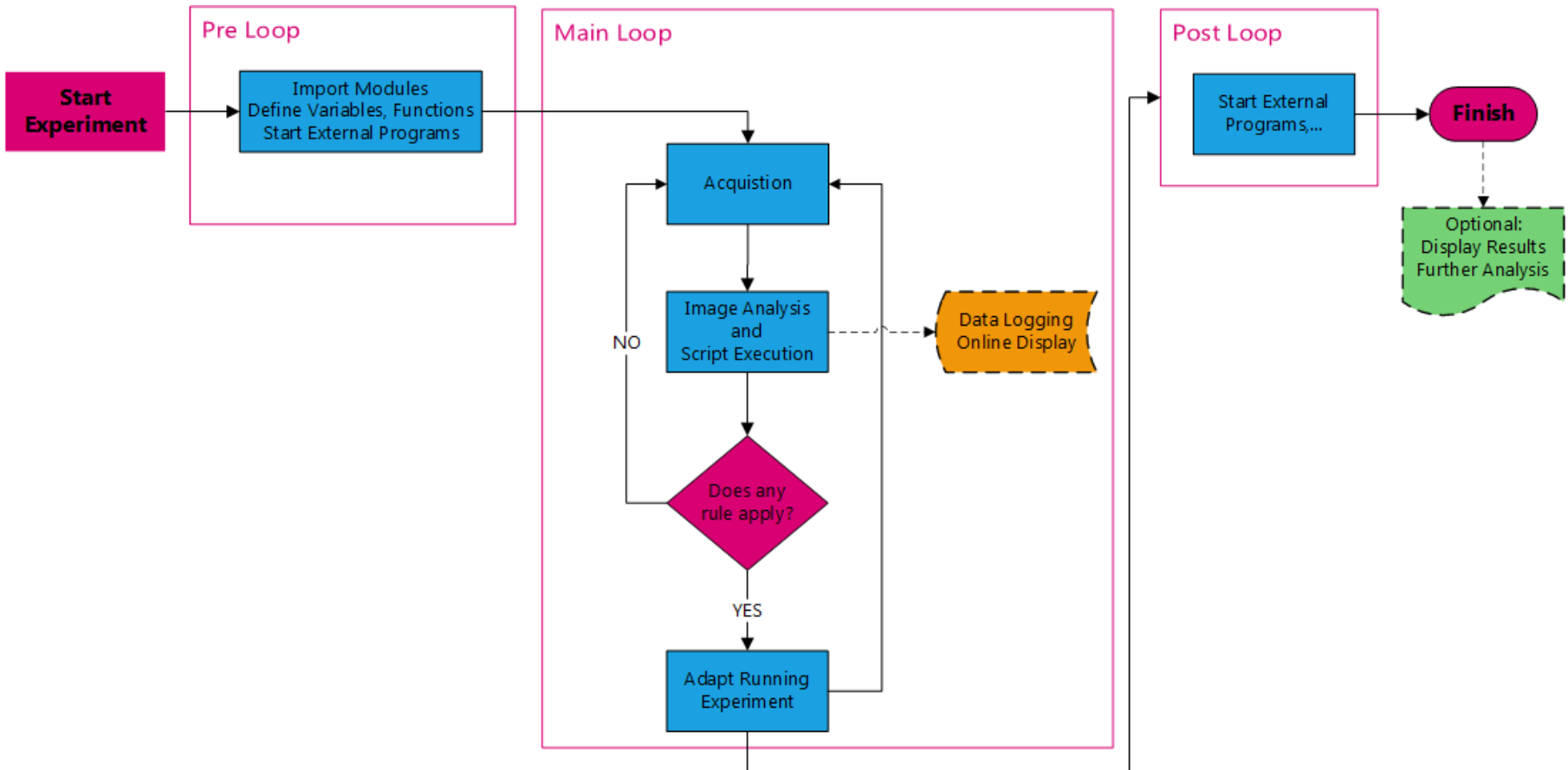
- **Adaptive Acquisition Engine:**  
**Modify running experiments** using Python scripts
- Access the **current system status** & results from **online image analysis** on runtime during the experiment

# Experiment Feedback Preparation



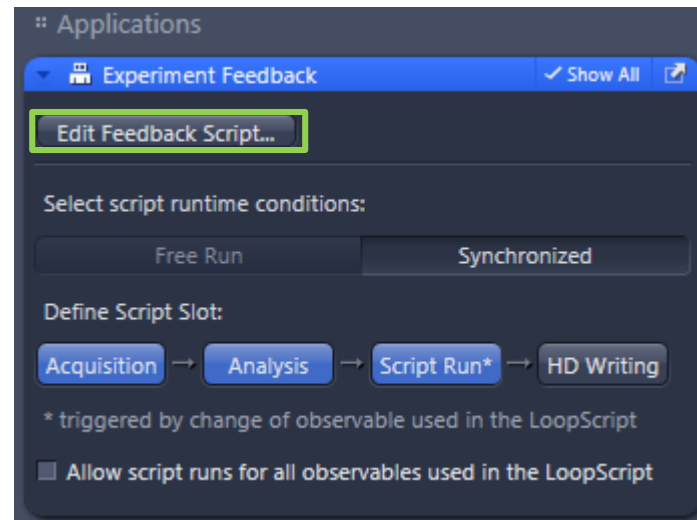
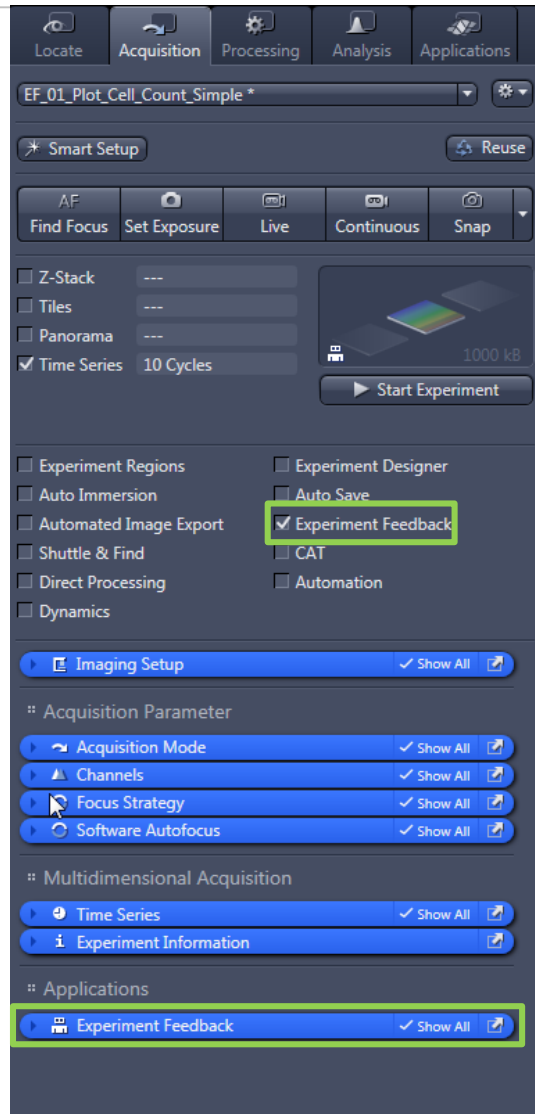
# Experiment Feedback

## Running the Experiment





# Create a Feedback Experiment



# 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

**Tools** **Debug**

**Available Observables**

Analysis

Experiment

Hardware

Environment

**Observe**

**Available Actions**

Experiment Actions

Hardware Actions

Extra actions

**Actions**

**Editor Tools**

Examples & Templates

**Validate Script**

The script is okay.

**Tools**

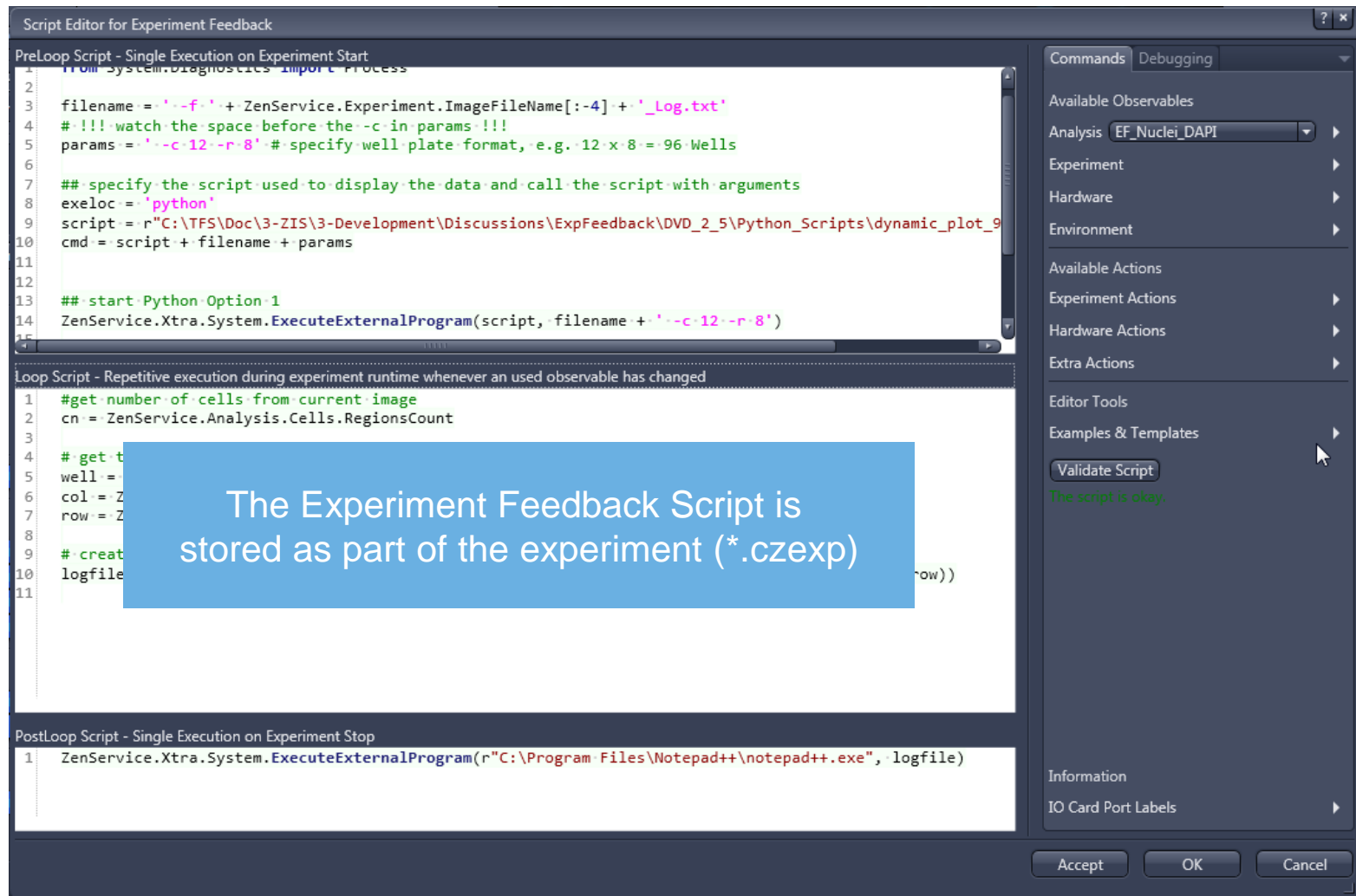
**Select Image Analysis Pipeline**

**Information**

IO Card Port Labels

**Accept** **Ok** **Cancel**

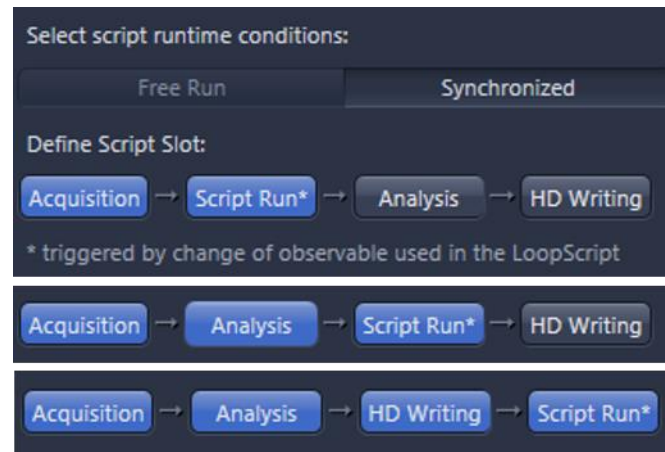
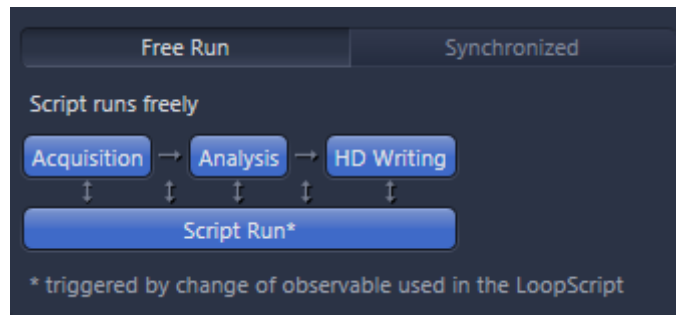
# Experiment Feedback Adaptive Acquisition Engine



# Experiment Feedback: Run Options



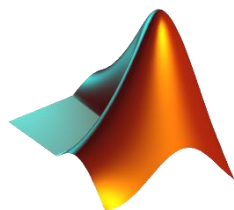
- Free Run
- Synchronized Run



# Sample Macros

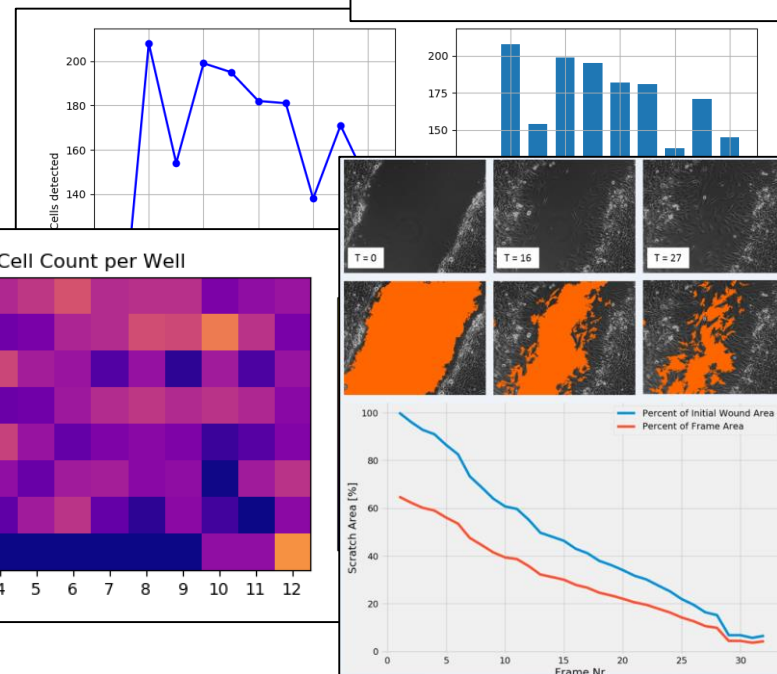


- Count cells, write logfile and start external data display
- Acquire tile images until a Total Number of Objects is Reached
- Online data display
- Acquire Image Data, Open in Fiji and Apply a Macro
- Jump to next well
- Adapt exposure time during acquisition
- Modify the blocks of an experiment
- Time-lapse per Z-Plane
- Automatic event detection
- Online dynamics
- Online scratch assay
- Online tracking

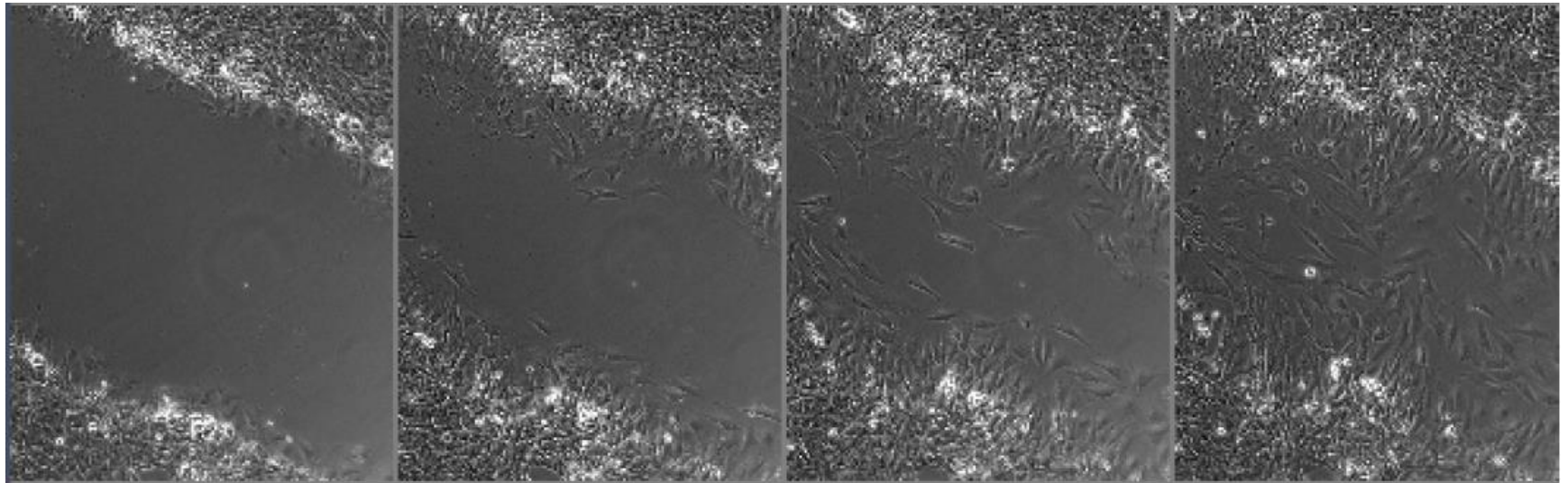


Experiment-59\_Log.txt - Editor

Datei	Bearbeiten	Format	Ansicht	?
Tile	Cells	Total	PosX	PosY
1	282.0	282.0	49324.0	35324.0
2	231.0	513.0	49381.6	35324.0
3	206.0	719.0	49439.2	35324.0
4	174.0	893.0	49496.8	35324.0
5	188.0	1081.0	49554.4	35324.0
6	201.0	1282.0	49612.0	35324.0
7	180.0	1462.0	49612.0	35381.6
8	175.0	1637.0	49554.4	35381.6
9	178.0	1815.0	49496.8	35381.6
10	151.0	1966.0	49439.2	35381.6
11	156.0	2122.0	49381.6	35381.6
12	170.0	2292.0	49324.0	35381.6
13	144.0	2436.0	49324.0	35439.2
14	176.0	2612.0	49381.6	35439.2
15	220.0	2832.0	49439.2	35439.2
16	193.0	3025.0	49496.8	35439.2



# Online Scratch Assay Data

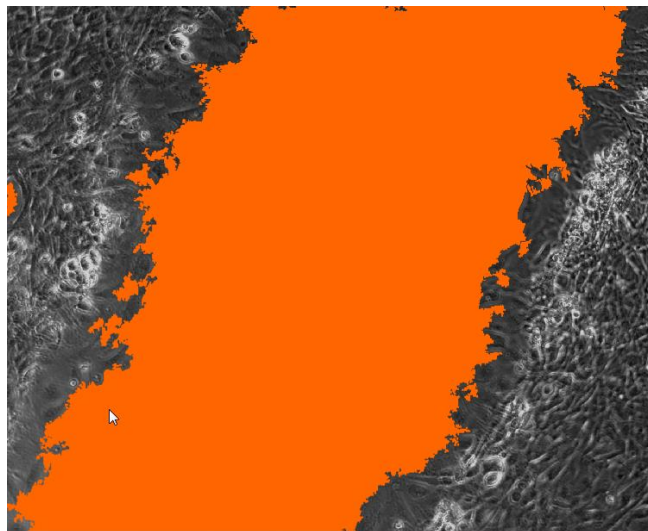
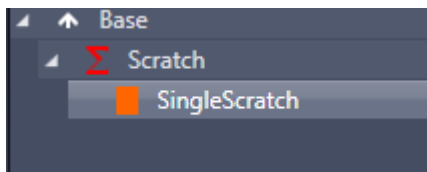


time

# Online Scratch Assay Image Analysis Setting



Classes      →      Segmentation      →      Features (Scratch)



Feature Selection	
Selected Features	
Name	Display
Image Scene Container Name	<input type="checkbox"/>
Image Scene Column Index	<input type="checkbox"/>
Image Scene Row Index	<input type="checkbox"/>
Image Stage Position X Image Center	<input type="checkbox"/>
Image Stage Position Y Image Center	<input type="checkbox"/>
Area Percentage	<input type="checkbox"/>
Area	<input type="checkbox"/>
Count	<input type="checkbox"/>
Image Index Time	<input type="checkbox"/>
Image Relative Time	<input type="checkbox"/>

# Online Scratch Assay Pre-Script



```
from System.Diagnostics import Process

filename = ZenService.Experiment.ImageFileName[:-4] + '_Log.txt'

exeloc = 'python'

script = r'C:\...\Python_Scripts\EF_ScratchAssay.py'

cmd = script + ' -f ' + filename

## start Python Option 1
#ZenService.Xtra.System.ExecuteExternalProgram(script, ' -f ' + filename)

## start Python Option 2
app = Process();
app.StartInfo.FileName = exeloc
app.StartInfo.Arguments = cmd
app.Start()
```



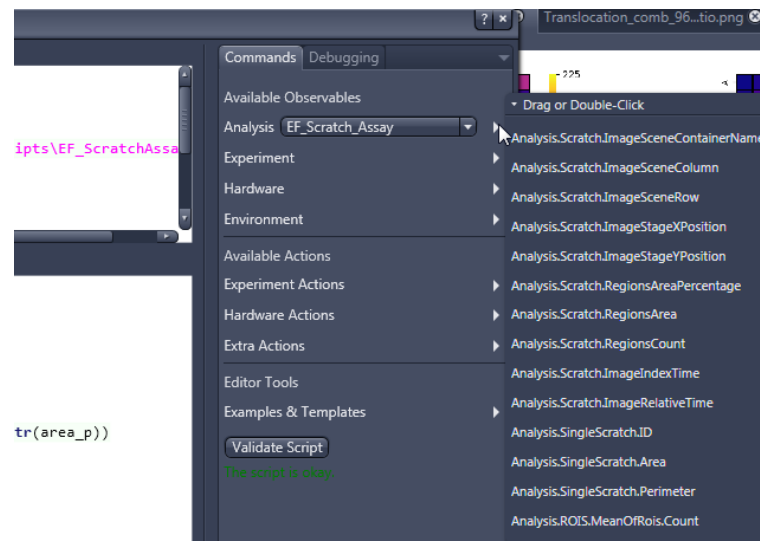
# Online Scratch Assay Loop-Script



```
# get the current well name, column index, row index and position index
frame = ZenService.Experiment.CurrentTimePointIndex

# get area parameters for the scratchnumber of cells from current image
area_t = ZenService.Analysis.Scratch.RegionsArea
area_p = ZenService.Analysis.Scratch.RegionsAreaPercentage

# create logfile
logfile = ZenService.Xtra.System.AppendLogLine(str(frame)+'\t'+str(area_t) + '\t'+
    str(area_p))
```



# Online Scratch Assay Python Script



```
import matplotlib.animation as animation
from matplotlib import style
import optparse
import numpy as np

# configure parsing option for command line usage
parser = optparse.OptionParser()

parser.add_option('-f', '--file',
                  action="store", dest="filename",
                  help="query string", default="No filename passed")

# read command line arguments
options, args = parser.parse_args()
savename = options.filename[:-4] + '.png'
print('Filename: ', options.filename)
print('Savename: ', savename)

# define plot layout
style.use('fivethirtyeight')
fig = plt.figure(figsize=(10,8))
ax1 = fig.add_subplot(1,1,1)
```

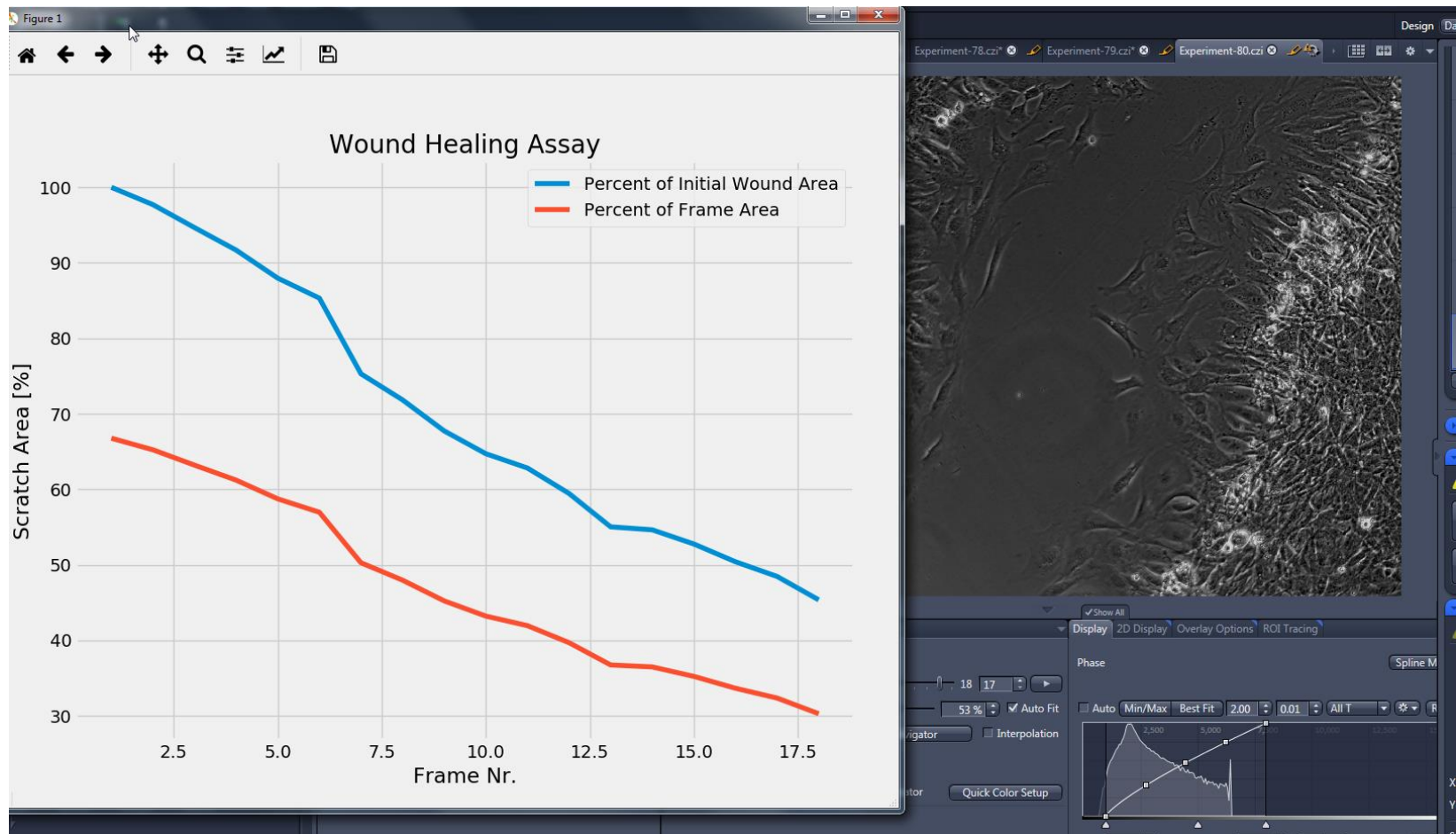
# Online Scratch Assay

## Python Script: Animate



```
def animate(i):  
    try:  
        graph_data = np.genfromtxt(options.filename, delimiter='\t')  
        xs = graph_data[:,0] # get frame Nr.  
        ys1 = graph_data[:,1] # get absolute Scratch area  
        ys2 = graph_data[:,2] # get Scratch area in percent of Frame Area  
        ys1_max = np.max(ys1, axis = 0) #get maximum Scratch Area  
        ys1_percent = ys1/ys1_max*100 #normalize Scratch Area  
  
        #labels and legend for plot  
        ax1.clear()  
        plt.title('Wound Healing Assay')  
        plt.xlabel('Frame Nr.')  
        plt.ylabel('Scratch Area [%]')  
        ax1.plot(xs, ys1_percent, label = 'Percent of Initial Wound Area')  
        ax1.plot(xs, ys2, label = 'Percent of Frame Area')  
        ax1.legend(loc='upper right')  
  
        #save plot  
        plt.savefig(savename)  
  
    except:  
        print('No file loaded')  
  
ani = animation.FuncAnimation(fig, animate, interval=1000, repeat=False)  
plt.show()
```

# Online Scratch Assay Online Plotting

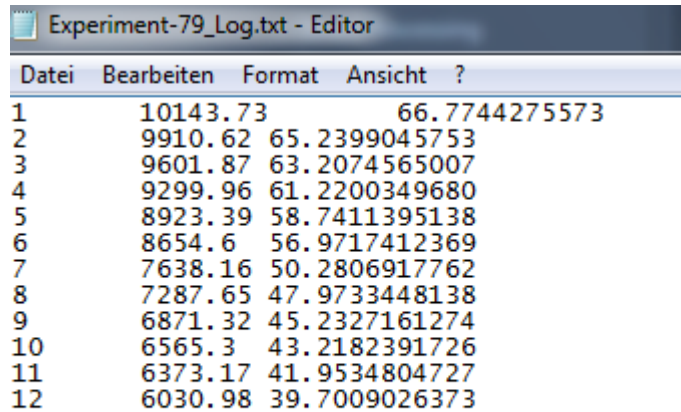


# Online Scratch Assay Post-Script



```
# open logfile
```

```
ZenService.Xtra.System.ExecuteExternalProgram(logfile, r'C:\...\notepad++.exe')
```



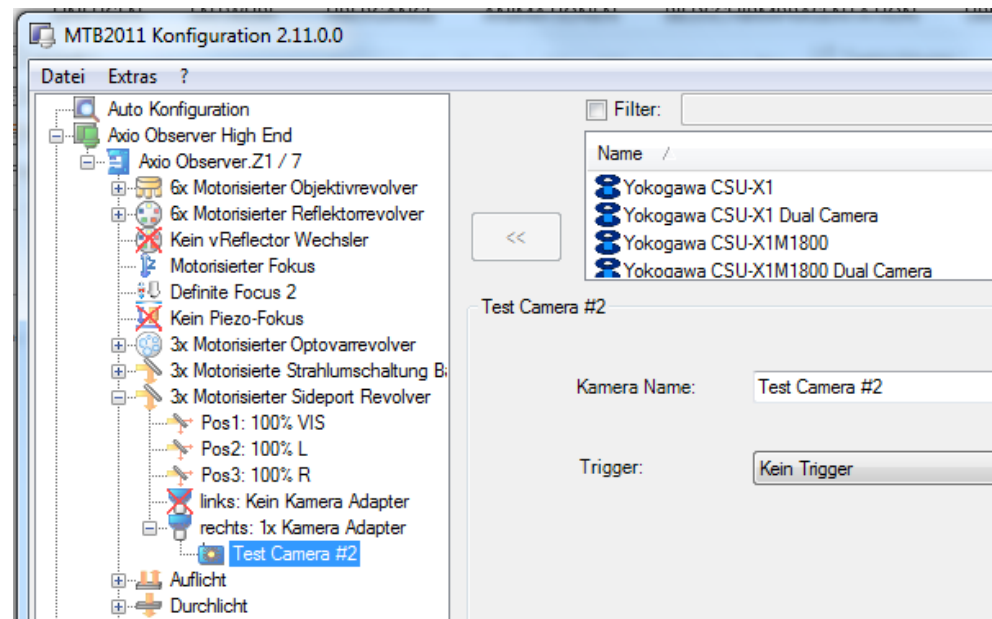
Datei	Bearbeiten	Format	Ansicht	?
1	10143.73		66.7744275573	
2	9910.62	65.2399045753		
3	9601.87	63.2074565007		
4	9299.96	61.2200349680		
5	8923.39	58.7411395138		
6	8654.6	56.9717412369		
7	7638.16	50.2806917762		
8	7287.65	47.9733448138		
9	6871.32	45.2327161274		
10	6565.3	43.2182391726		
11	6373.17	41.9534804727		
12	6030.98	39.7009026373		

Copy **Active Configuration.xml** and **CZIS\_Cameras.xml** to  
C:\ProgramData\Carl Zeiss\MTB2011\2.11.0.0

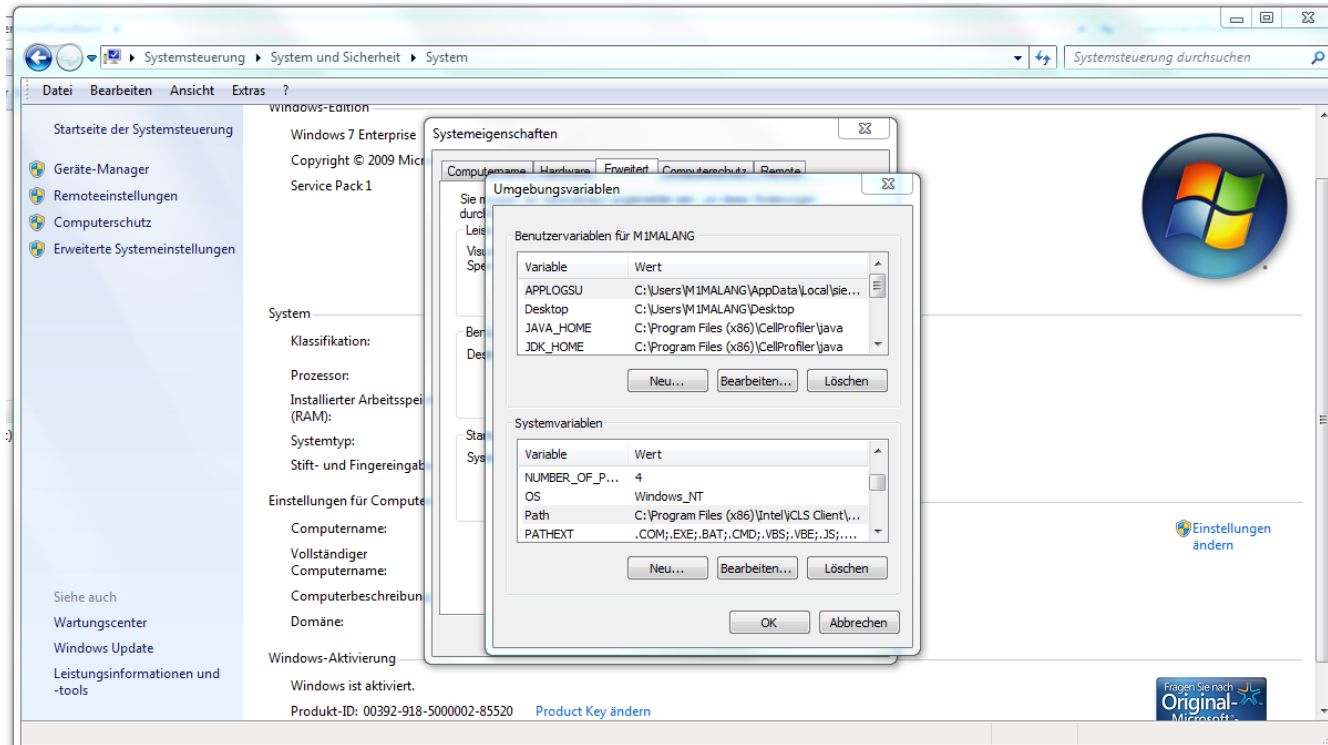
**Start MTB → Check “Simulate”**

C:\Program Files\Carl Zeiss\MTB 2011 - 2.11.0.0\MTB Configuration\MTBConfig.exe

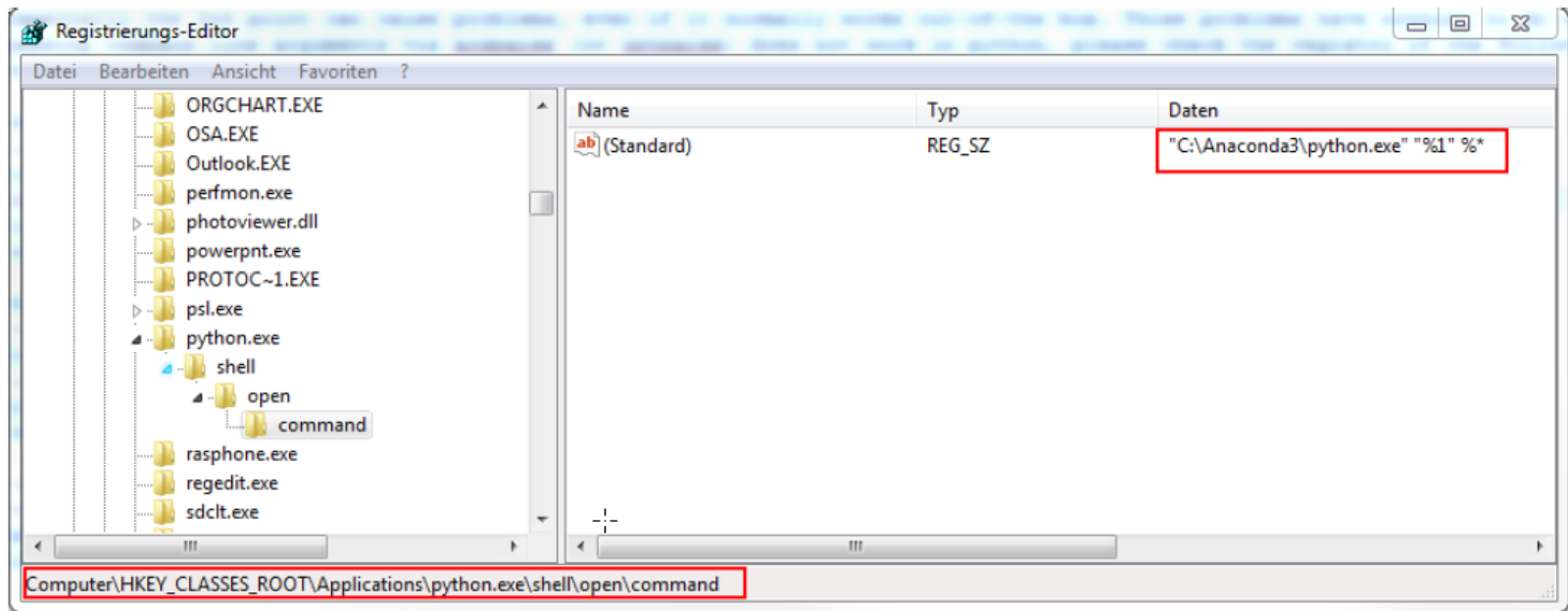
Check that there is a demo camera



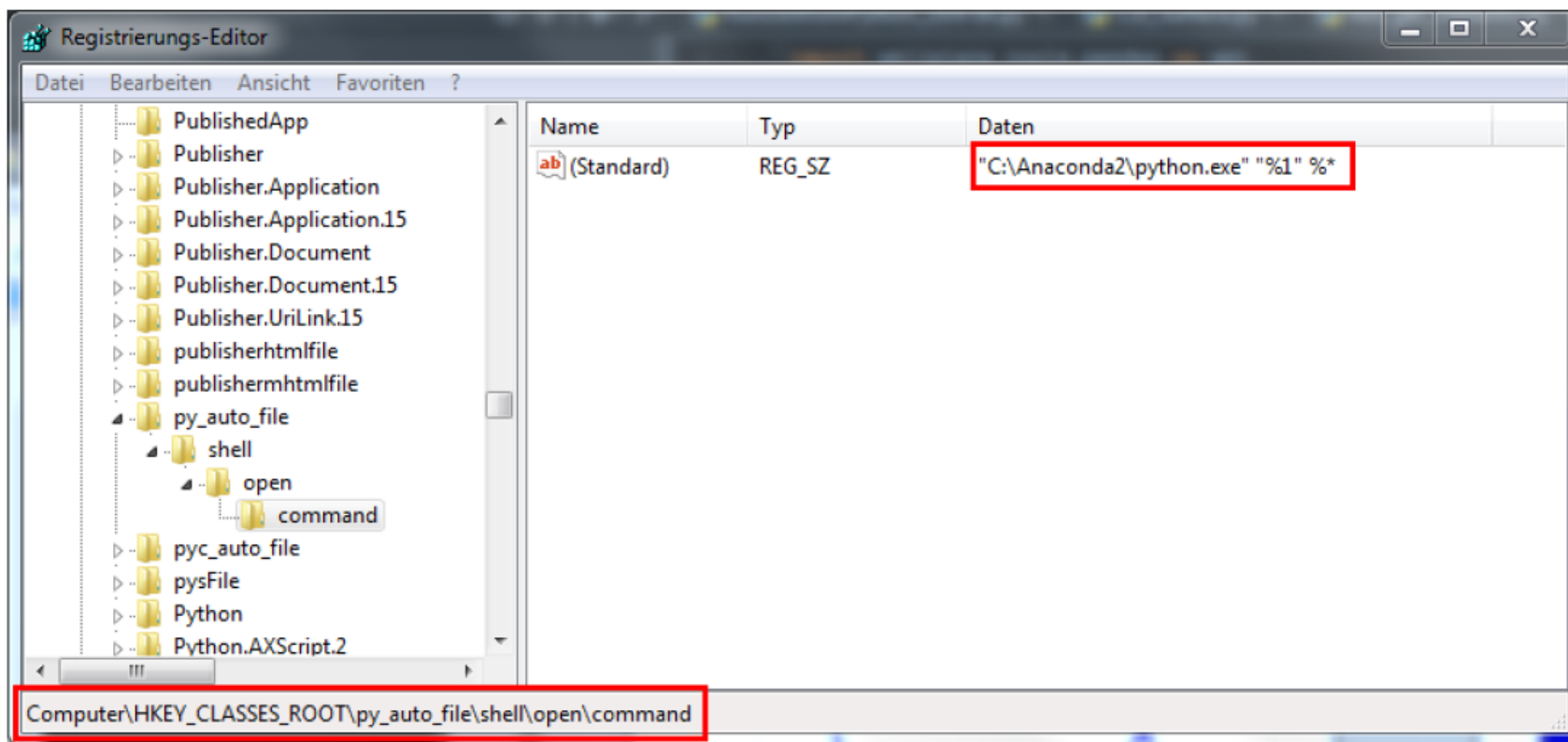
# Add Python to the PATH variable



Alternatively: In the script specify the whole path to the python.exe







## Copy Demo

To C:\Users\YourUsername\Documents\Carl Zeiss\ZEN\Documents

- Content of „Exp with Scripts“ → Experiment Setups
- Content of „Image Analysis“ → Image Analysis Settings

To a local folder:

- Test Images
- Python Scripts

# Simulate the Acquisition



Model Specific Reset

General

Camera Identifier

Total Magnification

Orientation

Lamp Intensity

Stage Position

Image Path

Image Mode

TestImage Mode

Perform Function

Acquire

Experimental Image File

HardwareImage Plug-In

Simulate Exposure ☒ On

Camera Bit Depth

Camera Bias

Display Current Image ☒ On

Current Image

