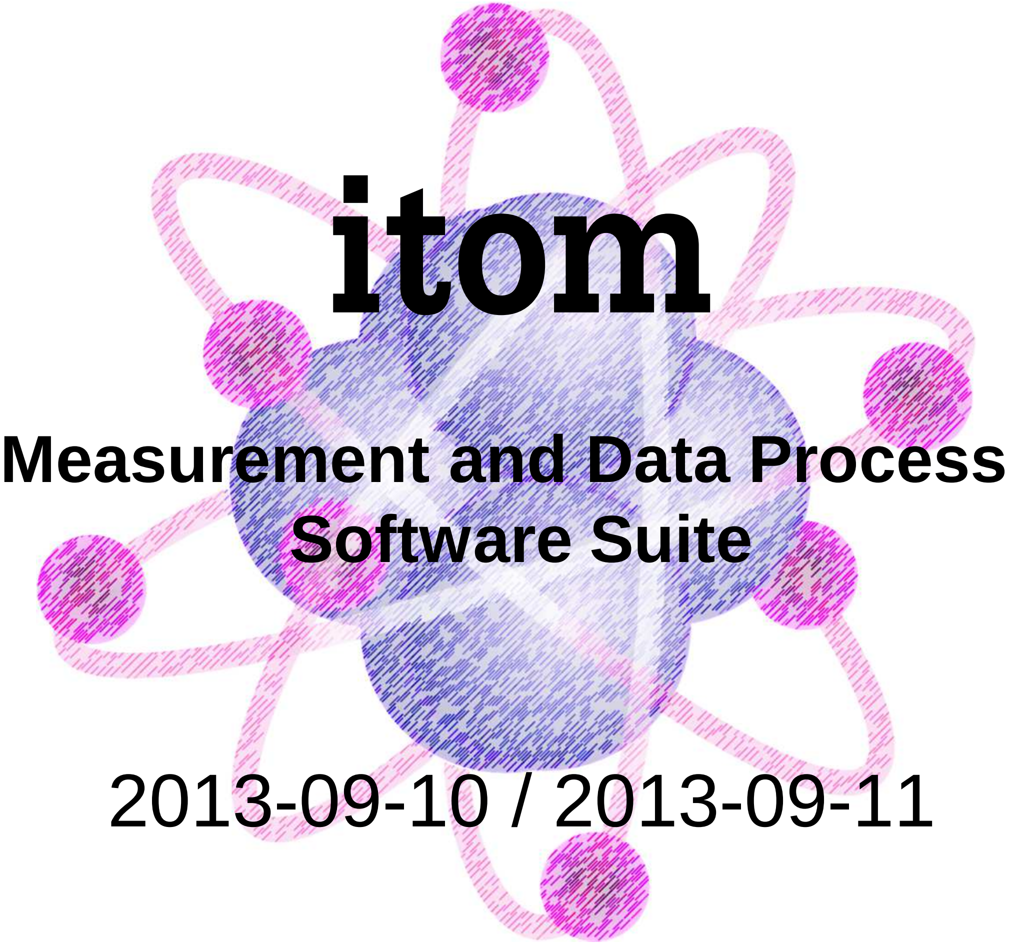


itom

A Measurement and Data Processing Software Suite

2013-09-10 / 2013-09-11

What this tutorial is about

- Introduction about **itom**
  - Why did we develop **itom**?
  - Main features
  - Python and its most important modules
  - **itom**’s plugin system
- Show-Cases
  - Macroscopic fringe projection
  - Software-Plugin: GUI for GPU based ray tracer MacroSim
  - Commercial confocal microscope from TWIP Optical Solutions
- Hands-on exercises
  - We develop an example to calculate the offset between two images, acquired with your webcam and create a user-developed GUI

Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

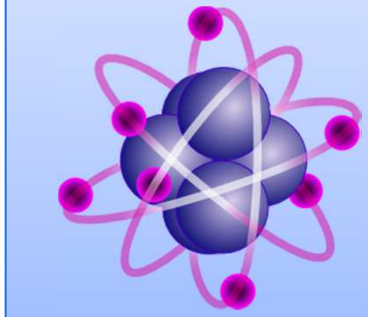
Motivation

Matlab

+ Data processing
+ Extensive math libraries
– Integration of hardware
– User defined interface

Labview

+ Easy generation of GUIs
+ Excellent hardware support
– Limited data processing and analysis
– No unified hardware interfaces



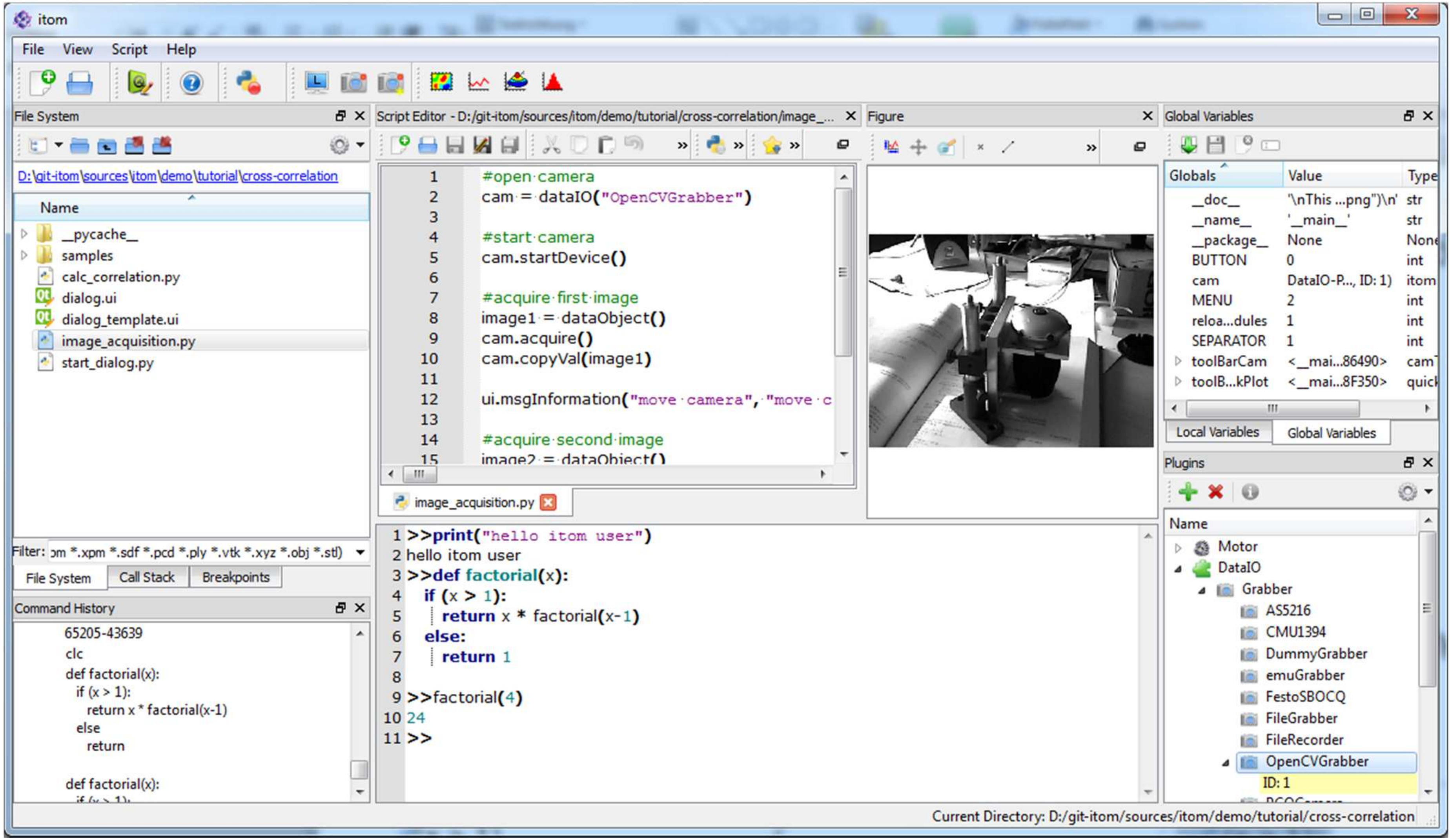
itom

- Fast, well-established, easy to use scripting language (Python)
- Homogeneous hardware integration
- Automation of measurement systems
- Fast data processing and analysis
- Easy to customize

Requirements and Solutions

Requirements	Solution
Fast, performant implementation	C++
Modern, user-friendly interface, independent of hardware platform	Qt-Framework (Windows, Linux, Mac OS)
Fully integrated scripting language (fast, robust, easy to learn, extensive existing libraries, well documented and supported)	Python (Version 3) incl. numerous libraries (numpy, scipy, scikit-image, matplotlib, …)
Easy, flexible, homogenous integration of hardware support (motors, cameras, AD converter, ...) and algorithms	Plugin-System
Using well-known, time-proven, free software libraries where possible	OpenCV, PointCloudLibrary, Qscintilla, Qwt, …

itom


itom
File
View
Script
Help
File System
Script Editor - D:/git-itom/sources/itom/demo/tutorial/cross-correlation/image_...
Figure
Global Variables
Plugins
Command History
File System
Call Stack
Breakpoints
Local Variables
Global Variables
image_acquisition.py
Filter: om *.xpm *.sdf *.pcd *.ply *.vtk *.xyz *.obj *.stl)
Current Directory: D:/git-itom/sources/itom/demo/tutorial/cross-correlation

D:\git-itom\sources\itom\demo\tutorial\cross-correlation

- Name
- _pycache_
- samples
- calc_correlation.py
- dialog.ui
- dialog_template.ui
- image_acquisition.py
- start_dialog.py

```
#open camera
cam = dataIO("OpenCVGrabber")

#start camera
cam.startDevice()

#acquire first image
image1 = dataObject()
cam.acquire()
cam.copyVal(image1)

ui.msgInformation("move camera", "move c

#acquire second image
image2 = dataObject()
```

```
1 >>print("hello itom user")
2 hello itom user
3 >>def factorial(x):
4     if (x > 1):
5         return x * factorial(x-1)
6     else:
7         return 1
8
9 >>factorial(4)
10 24
11 >>
```

Globals	Value	Type
__doc__	'\nThis ...png")\n'	str
__name__	'__main__'	str
__package__	None	None
BUTTON	0	int
cam	DataIO-P..., ID: 1)	itom
MENU	2	int
reloa...dules	1	int
SEPARATOR	1	int
toolBarCam	<__mai...86490>	cam
toolB...kPlot	<__mai...8F350>	quick

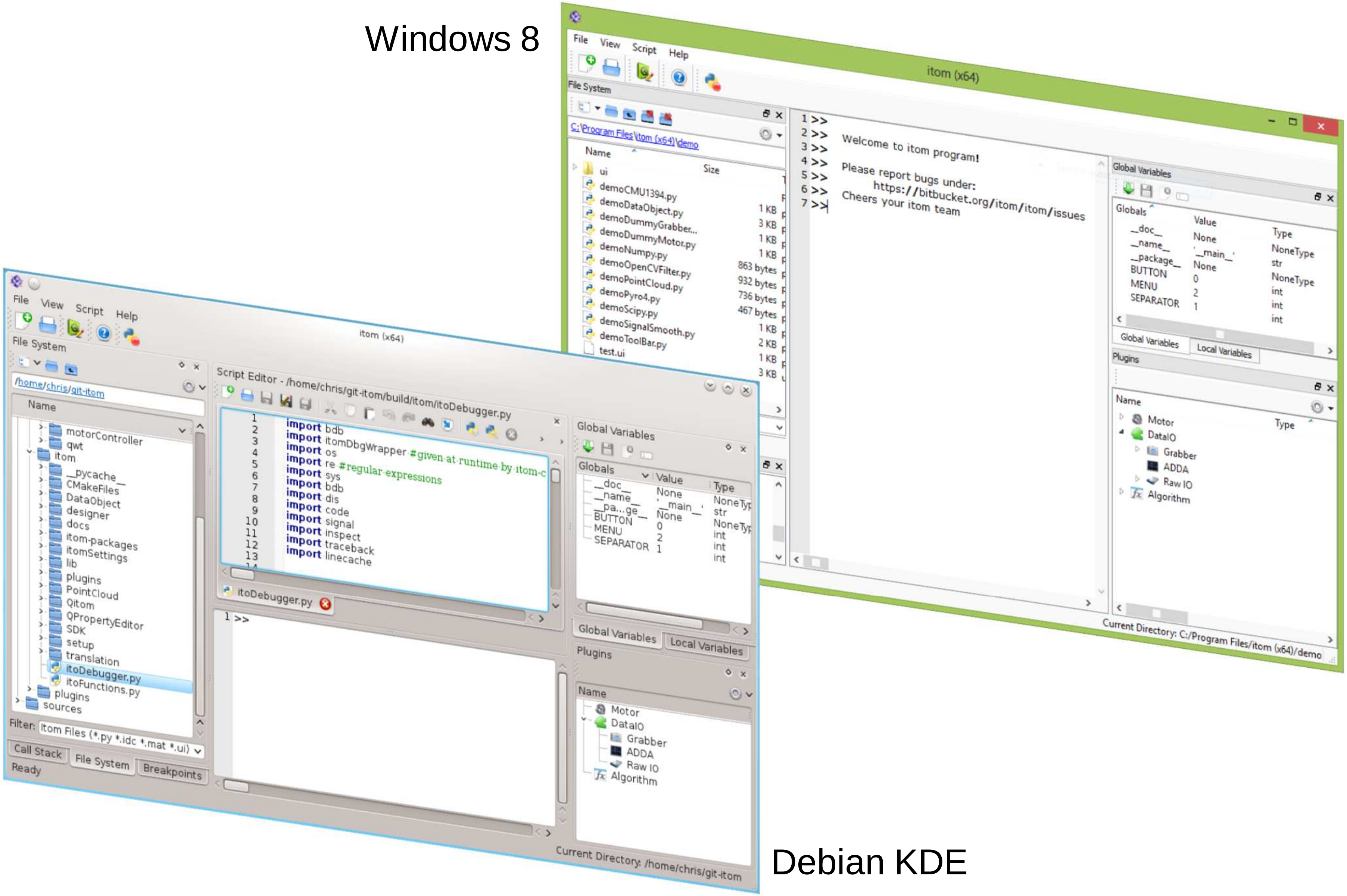
- Name
- Motor
- DataIO
  - Grabber
    - AS5216
    - CMU1394
    - DummyGrabber
    - emuGrabber
    - FestoSBOCQ
    - FileGrabber
    - FileRecorder
    - OpenCVGrabber
      - ID: 1
    - PCOCamera

```
65205-43639
clc
def factorial(x):
  if (x > 1):
    return x * factorial(x-1)
  else
    return

def factorial(x):
  if (x > 1):
```

Windows 7

itom

Windows 8



```
>>
>> Welcome to itom program!
>>
>> Please report bugs under:
>>     https://bitbucket.org/itom/itom/issues
>> Cheers your itom team
>>
```

```python
import bdb
import itomDbgWrapper #given at runtime by itom-c
import os
import re #regular expressions
import sys
import bdb
import dis
import code
import signal
import inspect
import traceback
import linecache
```

itom (x64)
File
View
Script
Help
File System
C:\Program Files\itom (x64)\demo
Script Editor - /home/chris/git-itom/build/itom/itoDebugger.py
itoDebugger.py
Global Variables
Local Variables
Plugins
Motor
DataIO
Grabber
ADDA
Raw IO
Algorithm
Call Stack
Breakpoints
Ready
Current Directory: C:/Program Files/itom (x64)/demo
Current Directory: /home/chris/git-itom

Debian KDE

itom – main features

Scripting
- Integrated Python programming environment
- Almost full Python functionality
- Controlling **itom** by specific Python module

Plugins
- C++ libraries (e.g. dll)
- Hardware and algorithm integration
- Integration of complex dialogs and windows

GUI
- Intuitive
- Optimized for implementation of measurement systems
- Ability to integrate customized user interfaces

Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

Python

- Open-Source scripting language (very liberal BSD-license)
- Implemented in C
- Developed and supported since 1991
- Supports object-oriented, functional and imperative programming paradigms
- Version 3.2 or newer supported
- Fully integrated core component of itom
- Vast number of third-party modules available for free
- Scripts are precompiled and cached for faster execution
- Integrated Python-debugger

Python

- Variables have an Python internal type, mainly: int, float, complex
- Casting uses the functions *int(), float()…*
- Assignment: *a=1* *a,b=1,2*
- Comparison operators: ==, >, <, <=, >=, !=
- Bitwise-Operators: &, |, ~, ^
- Basic arithmetic: a = a+1, a += 1, a=a**2
- Operators also work on many non-basic types (arrays, lists, dictionaries…)

Example: Factorial

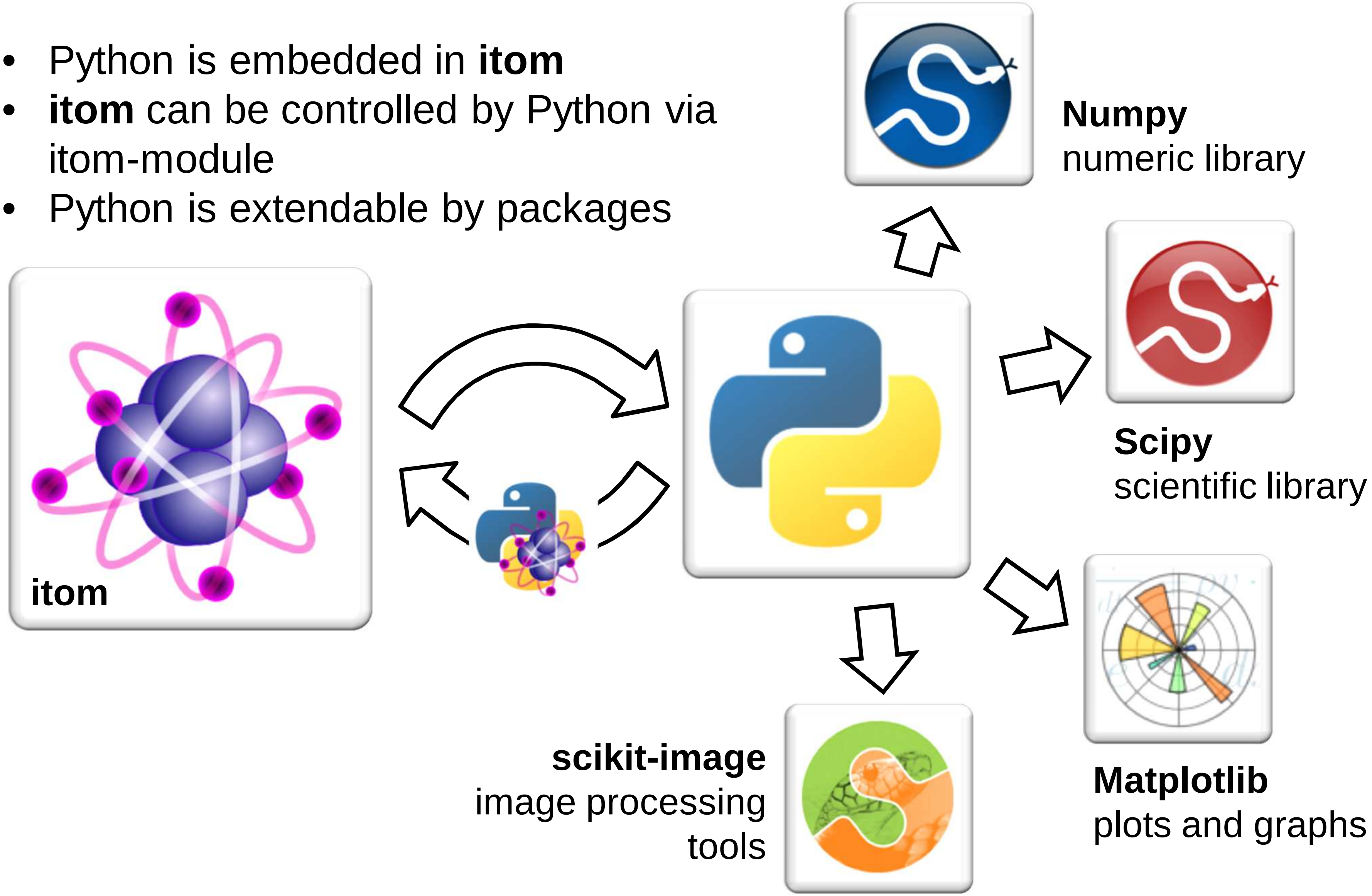
```
function ret = factorial(x)
   if(x > 1)
      ret = x * factorial(x-1);
   else
      ret = 1;
   end
end
```

```
int factorial(int x)
{
   if (x > 1) {
      return x * factorial(x-1);
   } else {
       return 1;
   }
}
```

```
def factorial(x):
   if (x > 1):
      return x * factorial(x-1)
   else:
      return 1
```

Python - Packages

- Python is embedded in **itom**
- **itom** can be controlled by Python via itom-module
- Python is extendable by packages



itom

Numpy
numeric library

Scipy
scientific library

Matplotlib
plots and graphs

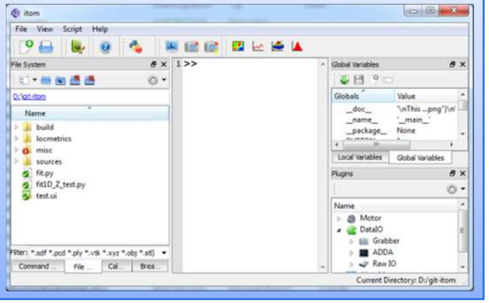
scikit-image
image processing tools

Python-Module *itom*

„The bridge between Python and itom“

```
>> from itom import * <<
```

- Add menus and toolbars to *itom* GUI and connect them with Python methods
- Plots arrays/matrices and camera live images

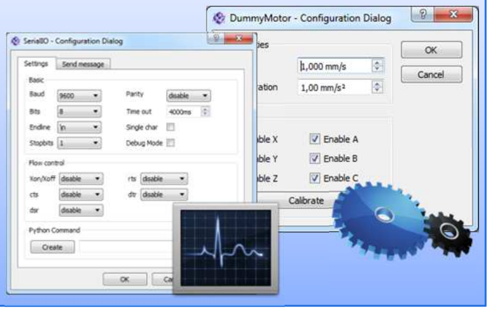


Filter: *.sdf *.pcd *.ply *.vtk *.xyz *.obj *.stl)

Globals	Value
__doc__	'\nThis ...png')\n'
__name__	'__main__'
__package__	None

Current Directory: D:/git-itom

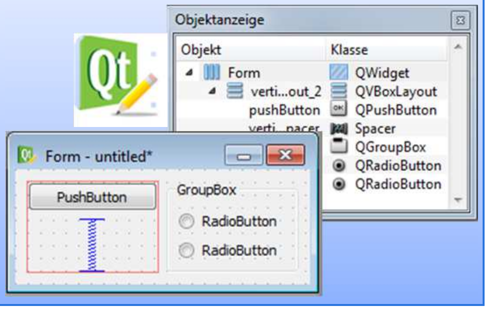
- Control hardware plugins (*dataIO*, *actuator*)
- Call algorithms from software plugins
- Online help for plugins



Baud 9600 | Parity disable
Bits 8 | Time out 4000ms
Endline \n | Single char □
Stopbits 1 | Debug Mode □
Xon/Xoff disable | rts disable
cts disable | dtr disable
dsr disable

1,000 mm/s
1,00 mm/s²
☑ Enable A
☑ Enable B
☑ Enable C

- Build GUIs at runtime with WYSIWYG design tool
- Connect widget’s signals to python methods
- Change properties of widgets by script commands



Objekt	Klasse
Form	QWidget
verti...out_2	QVBoxLayout
pushButton	QPushButton
verti...pacer	Spacer
	QGroupBox
	QRadioButton
	QRadioButton

Numpy

Numeric package

- Support of large, multi-dimensional arrays
- Large library of mathematical functions and operators
- **itom**’s own array object is compatible to Numpy arrays.

Example: Solve Ax=b

```
from numpy import *
from numpy.linalg import solve

# The system of equations we want to solve for (x0,x1,x2):
# 3 * x0 + 1 * x1 + 5 * x2 = 6
# 1 * x0 + 8 * x2 = 7
# 2 * x0 + 1 * x1 + 4 * x2 = 8

a = array([[3,1,5],[1,0,8],[2,1,4]])
b = array([6,7,8])
x = solve(a,b)
print(x) # This is our solution
[-3.28571429 9.42857143 1.28571429]
```

- Array creating and manipulation
- Binary operations
- Linear algebra
- Masked arrays
- Polynomials
- Random Sampling
- Sorting, Searching, Counting
- Fourier Transforms
- …

Scipy

Scientific Algorithms

- Extension for *numpy*
- Provide more functions from the field of numeric, statistic and optimization
- Itself extendable by *scikits*

- Optimization
- Linear Algebra
- Integration
- Interpolation
- FFT
- Signal Processing
- ODE Solvers
- Optimization
- Basic image processing
- Sparse Matrices

Example:

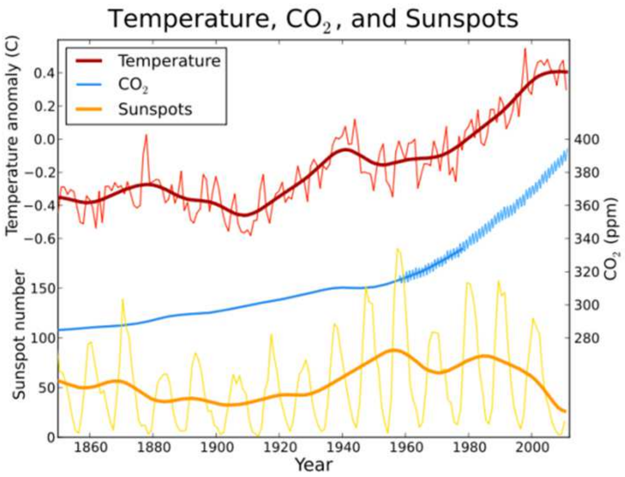
Find root of $x + 2cos(x) = 0$ around $x = 0.3$

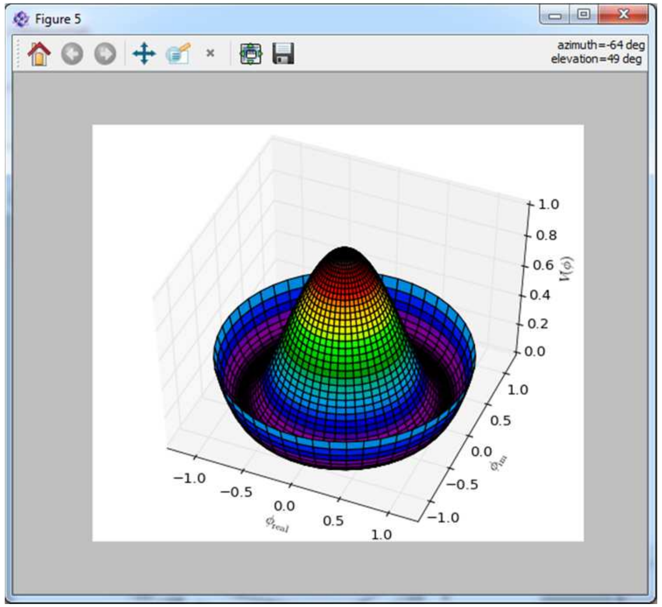
```
import numpy as np
from scipy.optimize import root

def func(x):
    return x + 2 * np.cos(x)

sol = np.root(func, 0.3)
sol.x
>>> array([-1.02986653])
sol.fun
>>> array([ -6.66133815e-16])
```

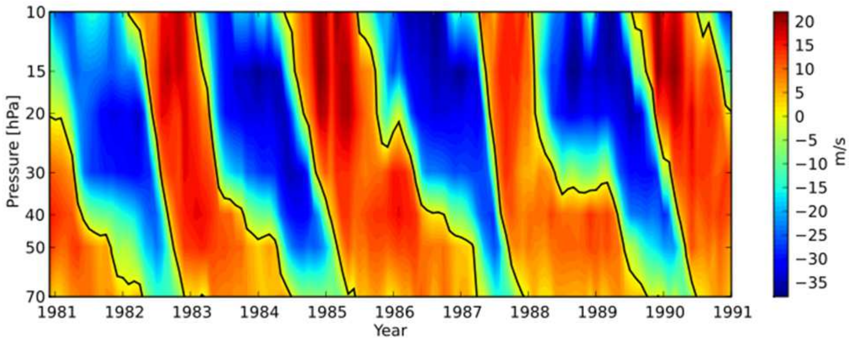
Matplotlib

Plotting package

- Python package for math plots
- Based on *numpy*
- Syntax close to Matlab
- Export in various image formats: png, pdf, eps…
- Fully integrated in *itom*
- Can be integrated in custom GUIs







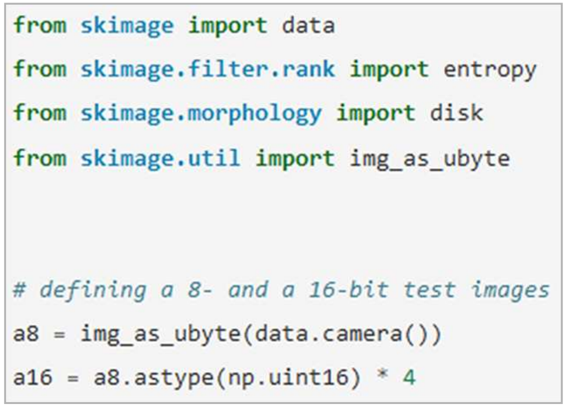
scikit-image

Image processing package

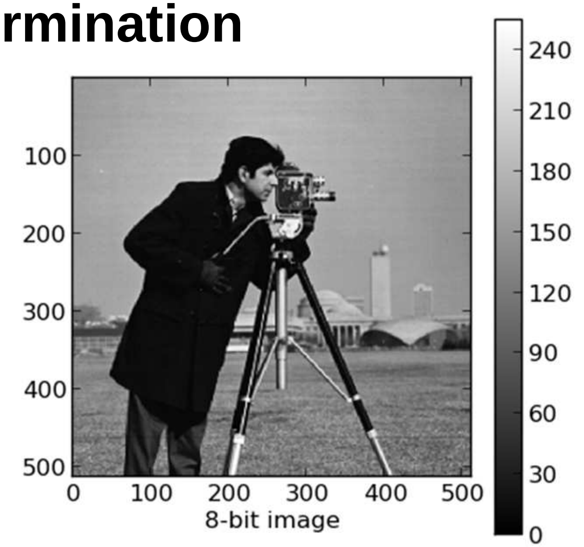
- Based on Numpy arrays
- Algorithms written in Python and C
- Uses Matplotlib for plotting results

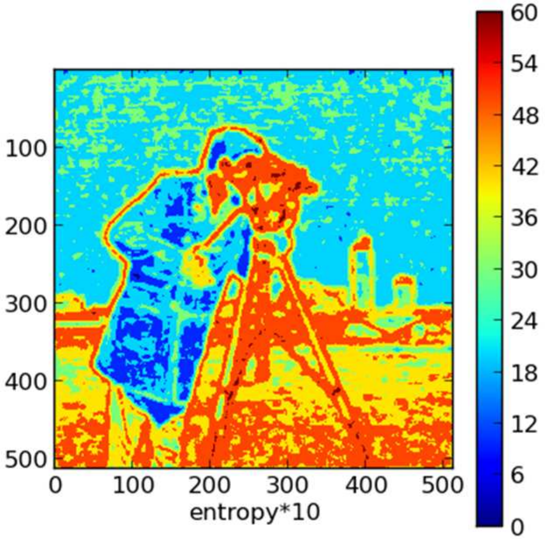
- Segmentation
- Transformation
- Morphology
- Measure
- IO
- Image filtering
- Rank filters
- Feature detection

Example: Entropy determination



```
from skimage import data
from skimage.filter.rank import entropy
from skimage.morphology import disk
from skimage.util import img_as_ubyte


# defining a 8- and a 16-bit test images
a8 = img_as_ubyte(data.camera())
a16 = a8.astype(np.uint16) * 4
```





Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

itom Plugin System

- Plugins extend the basic functionalities of **itom**. Each plugin is a C++ library (.dll, .so)
- Every Plugin implements one of three basic interface classes (*DataIO, Actuator, Algorithm*)
- Plugins (e.g. camera, motor stages…) can be instantiated from Python or directly through the itom GUI

DataIO

- Cameras
- A/D-Converters
- Serial Bus

Actuator

- Motors
- Multi-Axes Machines

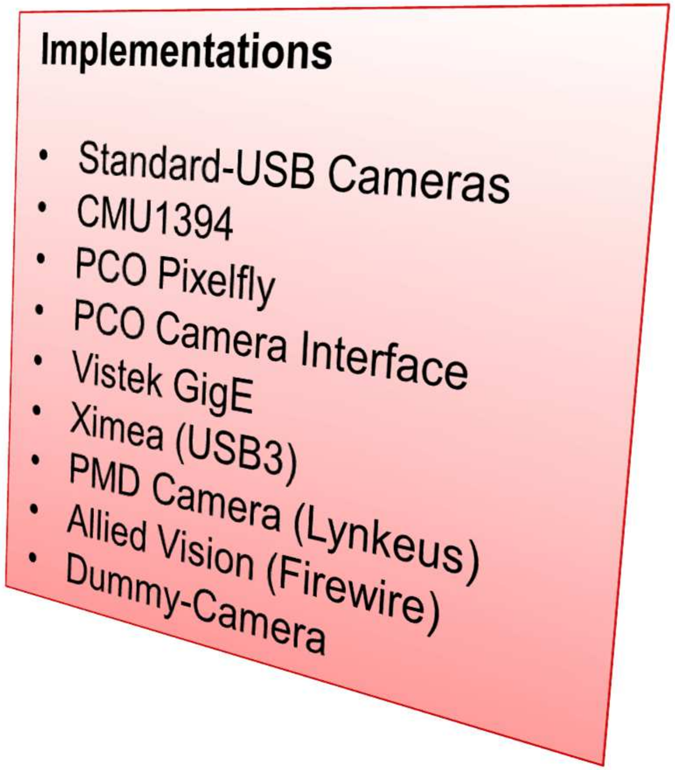
Algorithm

- Algorithms
- Data Filters
- Complex GUIs

Interface “dataIO + Grabber”

Primary functionality

- getParam(..) → read a parameter
- setParam(..) → set a parameter
- startDevice() → start camera
- stopDevice() → stop camera
- acquire() → take a picture
- getVal(..) / copyVal(..) → load image from camera into itom/Python
- …



Implementations

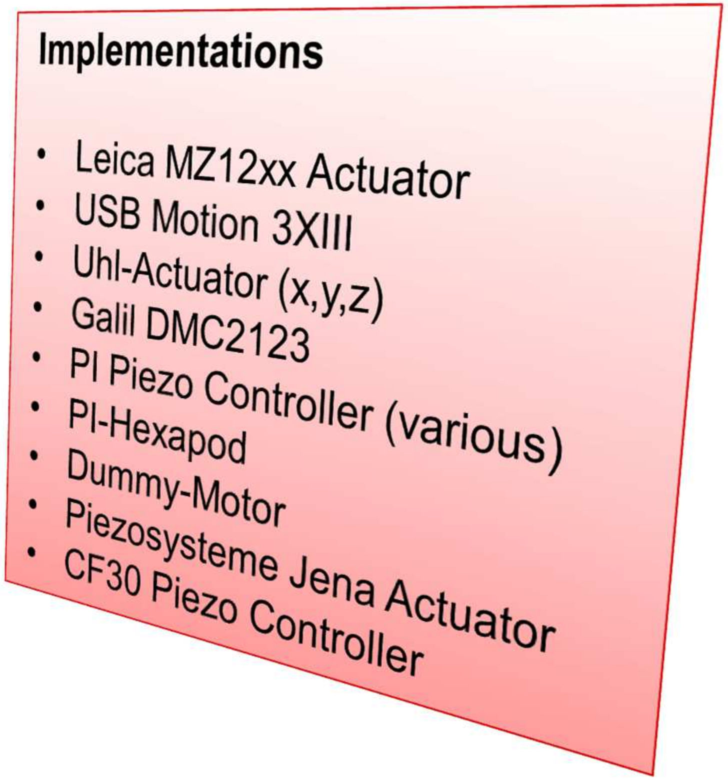
- Standard-USB Cameras
- CMU1394
- PCO Pixelfly
- PCO Camera Interface
- Vistek GigE
- Ximea (USB3)
- PMD Camera (Lynkeus)
- Allied Vision (Firewire)
- Dummy-Camera

Live images from the camera can be displayed in separate windows or integrated into custom GUIs

Interface “actuator”

Primary Functionality

- getParam(..) → read Parameter
- setParam(..) → set Parameter
- getStatus(..) → get status per axis
- getPos(..) → read current position
- setPosAbs/Rel() → move to position
- …



Implementations

- Leica MZ12xx Actuator
- USB Motion 3XIII
- Uhl-Actuator (x,y,z)
- Galil DMC2123
- PI Piezo Controller (various)
- PI-Hexapod
- Dummy-Motor
- Piezosysteme Jena Actuator
- CF30 Piezo Controller

Signals about position and status of the actuator can be linked to and processed by the GUI.

Interface “algo”

‚Algo‘ plugins define

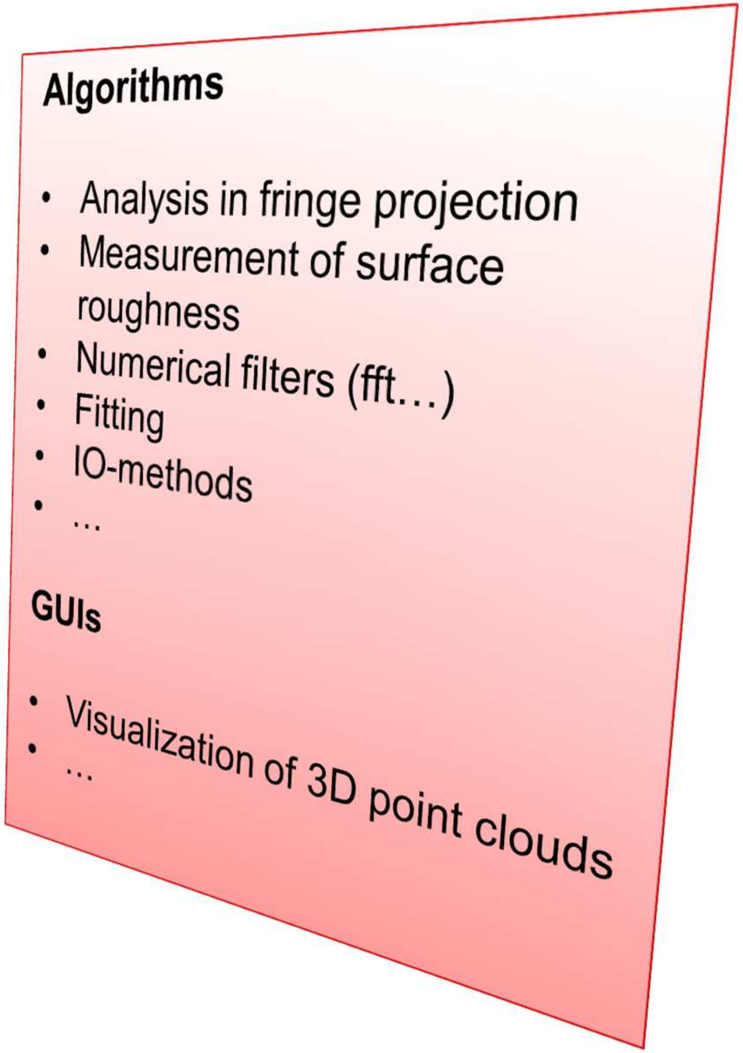
- Numerical algorithms
- GUI elements

Call:

- From a Python script
- By other plugins

Each method is defined by :

- Mandatory parameters (Type, description…)
- Optional parameters
- Return values



Algorithms

- Analysis in fringe projection
- Measurement of surface roughness
- Numerical filters (fft…)
- Fitting
- IO-methods
- …

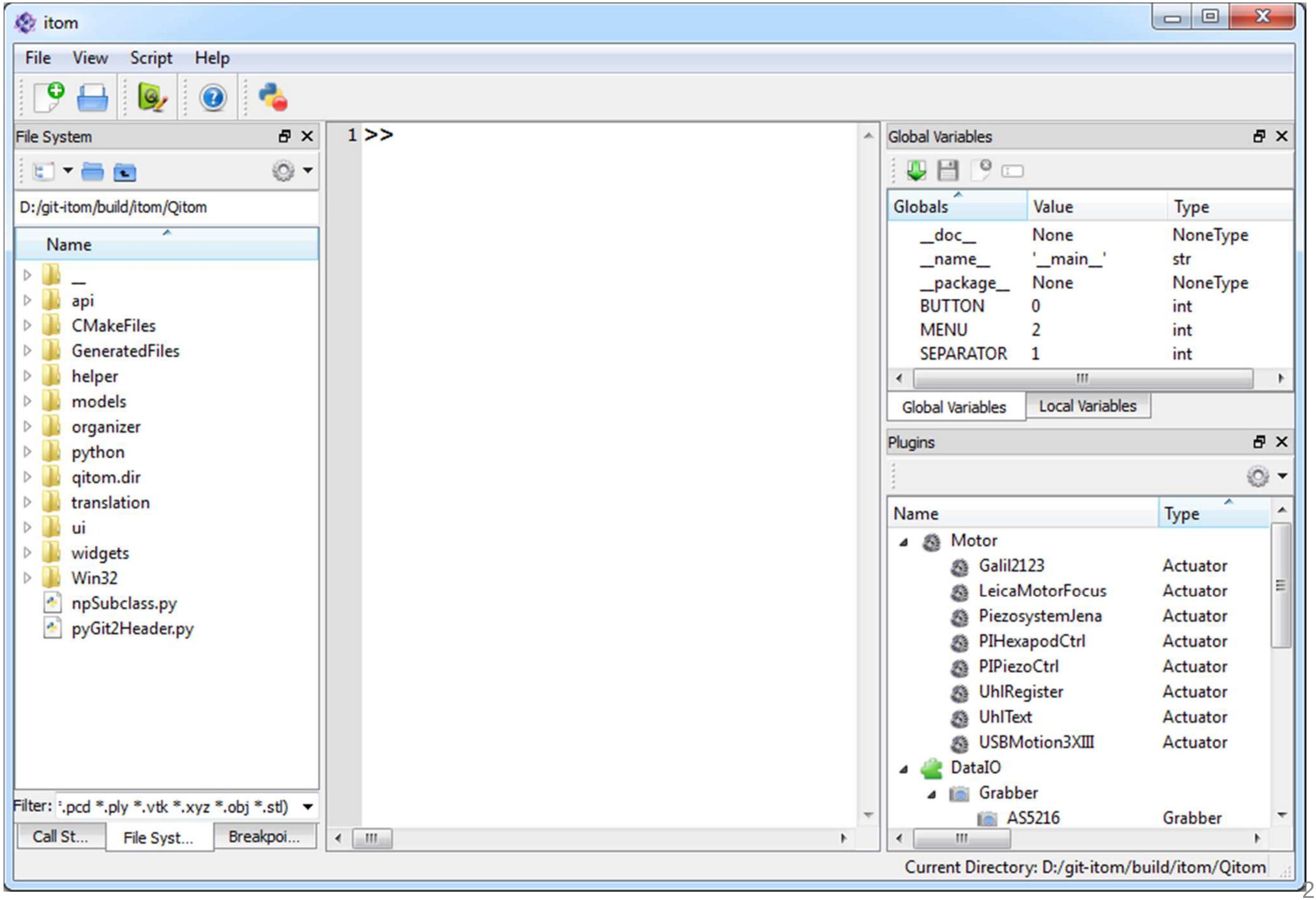
GUIs

- Visualization of 3D point clouds
- …

Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

GUI



D:/git-itom/build/itom/Qitom

Name

- _
- api
- CMakeFiles
- GeneratedFiles
- helper
- models
- organizer
- python
- qitom.dir
- translation
- ui
- widgets
- Win32
- npSubclass.py
- pyGit2Header.py

Filter: *.pcd *.ply *.vtk *.xyz *.obj *.stl)

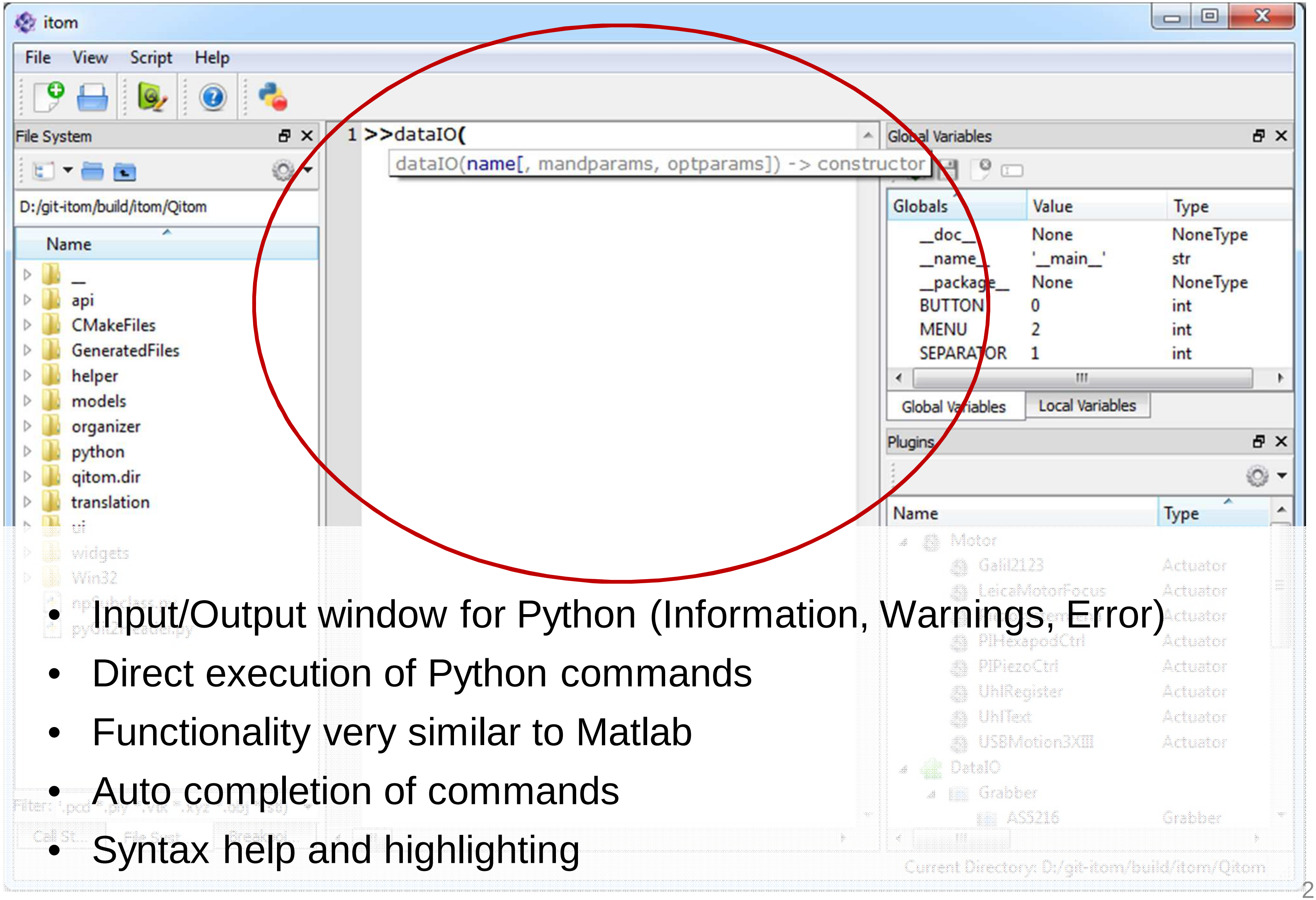
```
1 >>
```

Globals	Value	Type
__doc__	None	NoneType
__name__	'__main__'	str
__package__	None	NoneType
BUTTON	0	int
MENU	2	int
SEPARATOR	1	int

Name	Type
Motor	
Galil2123	Actuator
LeicaMotorFocus	Actuator
PiezosystemJena	Actuator
PIHexapodCtrl	Actuator
PIPiezoCtrl	Actuator
UhlRegister	Actuator
UhlText	Actuator
USBMotion3XIII	Actuator
DataIO	
Grabber	
AS5216	Grabber

Current Directory: D:/git-itom/build/itom/Qitom

GUI – Command Line

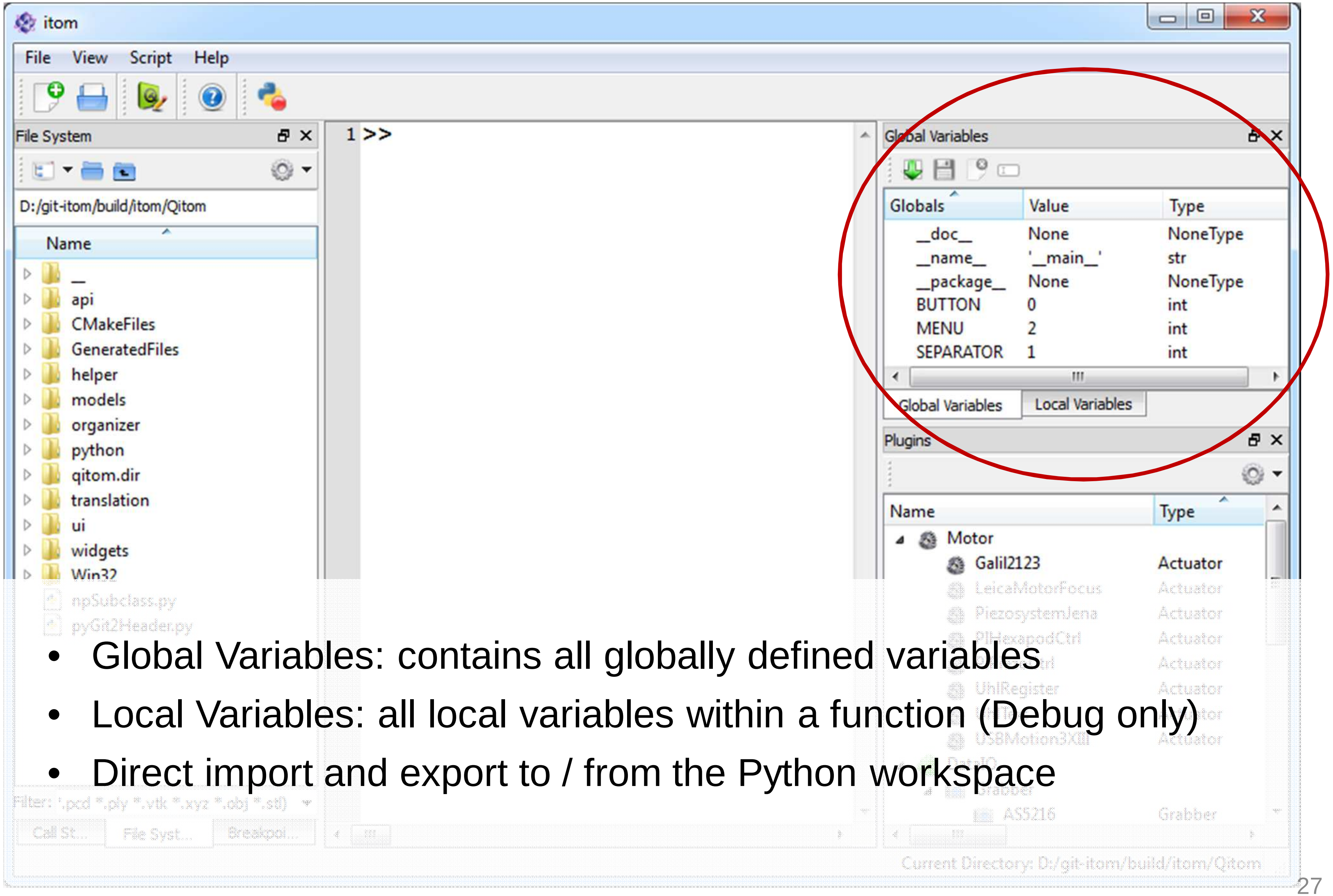


```
>>dataIO(
dataIO(name[, mandparams, optparams]) -> constructor
```

Globals	Value	Type
__doc__	None	NoneType
__name__	'__main__'	str
__package__	None	NoneType
BUTTON	0	int
MENU	2	int
SEPARATOR	1	int

- Input/Output window for Python (Information, Warnings, Error)
- Direct execution of Python commands
- Functionality very similar to Matlab
- Auto completion of commands
- Syntax help and highlighting

GUI - Workspace

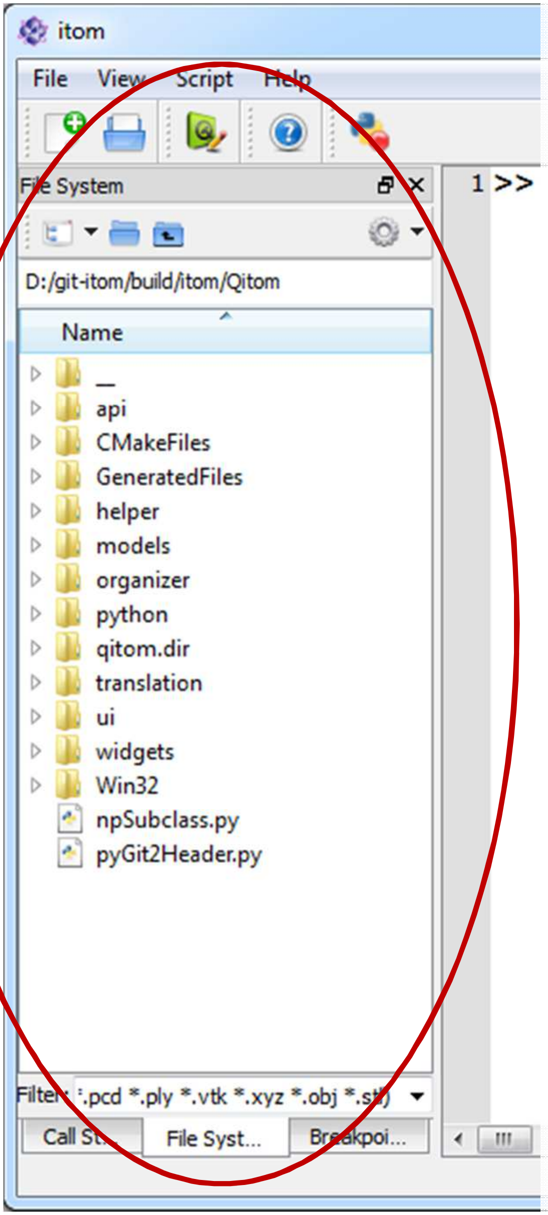


Globals	Value	Type
__doc__	None	NoneType
__name__	'__main__'	str
__package__	None	NoneType
BUTTON	0	int
MENU	2	int
SEPARATOR	1	int

Name	Type
Motor	
Galil2123	Actuator
LeicaMotorFocus	Actuator
PiezosystemJena	Actuator
PIHexapodCtrl	Actuator
	Actuator
UhlRegister	Actuator
	Actuator
USBMotion3XIII	Actuator
DataIO	
Grabber	
AS5216	Grabber

- Global Variables: contains all globally defined variables
- Local Variables: all local variables within a function (Debug only)
- Direct import and export to / from the Python workspace

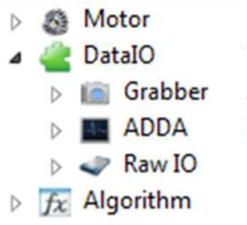
GUI – File System



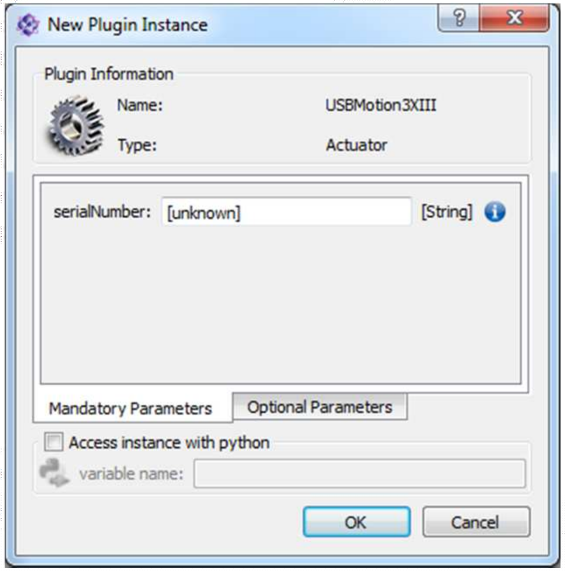
- Access and administration of all scripts and files that can be opened in *itom*
- The default main directory is the current working directory (similar to Matlab)
- Double click on a .py Python script will open it in the scripting window
- Double click on supported file types will load them into workspace

GUI – Plugins

- List of all available Plugins
- Sorted by category



- Allows direct instantiation of hardware plugins



Plugin Information

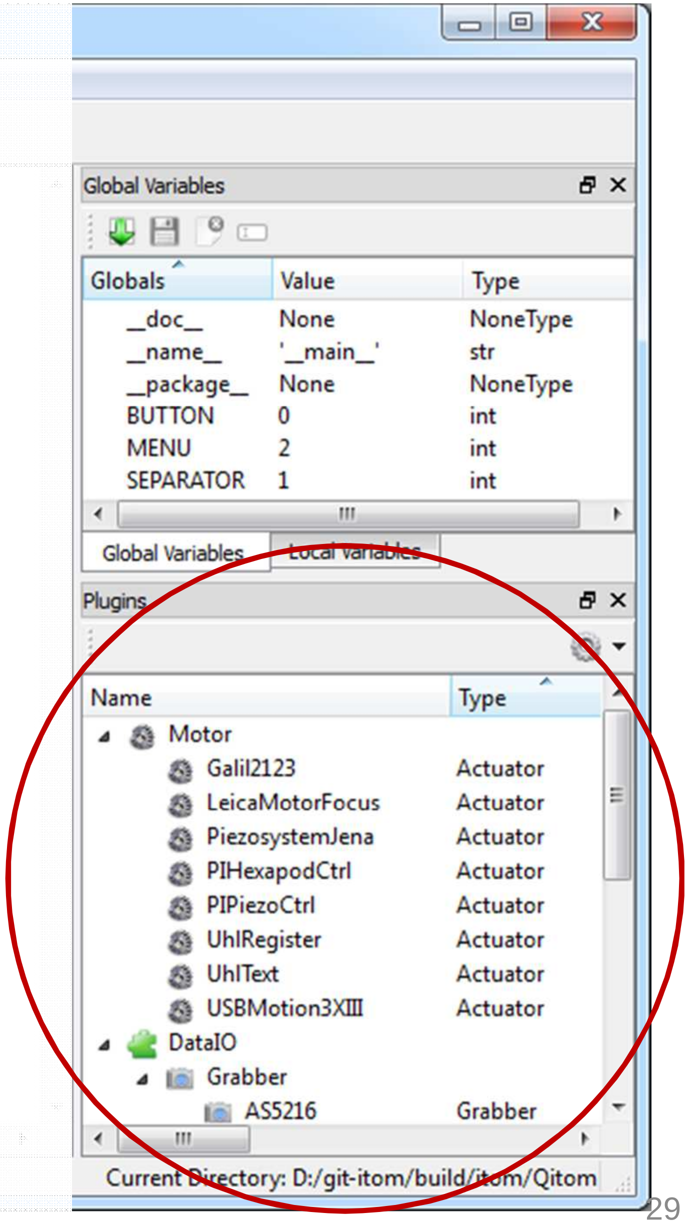
Name: USBMotion3XIII

Type: Actuator

serialNumber: [unknown] [String]

☐ Access instance with python

variable name:



Global Variables

Globals	Value	Type
__doc__	None	NoneType
__name__	'__main__'	str
__package__	None	NoneType
BUTTON	0	int
MENU	2	int
SEPARATOR	1	int

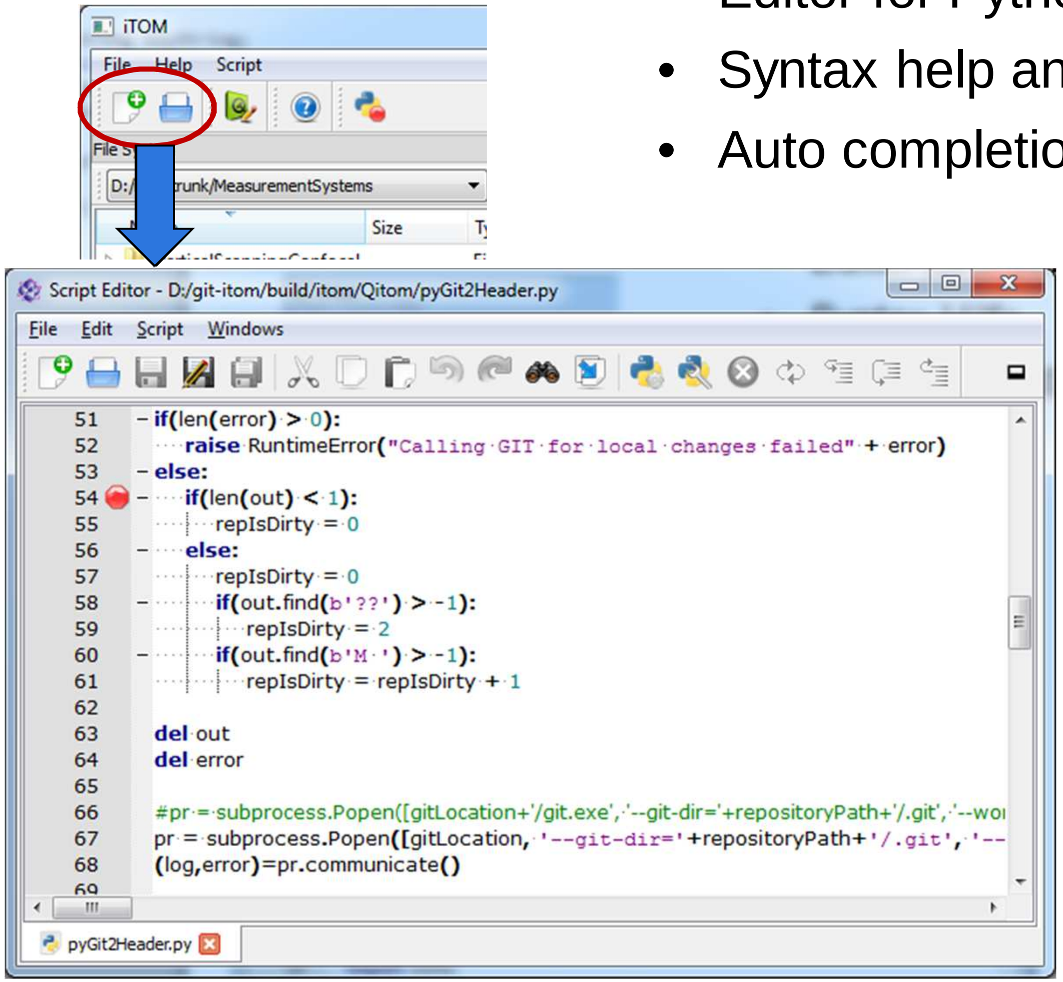
Plugins

Name	Type
Motor	
Galil2123	Actuator
LeicaMotorFocus	Actuator
PiezosystemJena	Actuator
PIHexapodCtrl	Actuator
PIPiezoCtrl	Actuator
UhlRegister	Actuator
UhlText	Actuator
USBMotion3XIII	Actuator
DataIO	
Grabber	
AS5216	Grabber

Current Directory: D:/git-itom/build/itom/Qitom

Scripting window

- Editor for Python scripts
- Syntax help and highlighting
- Auto completion

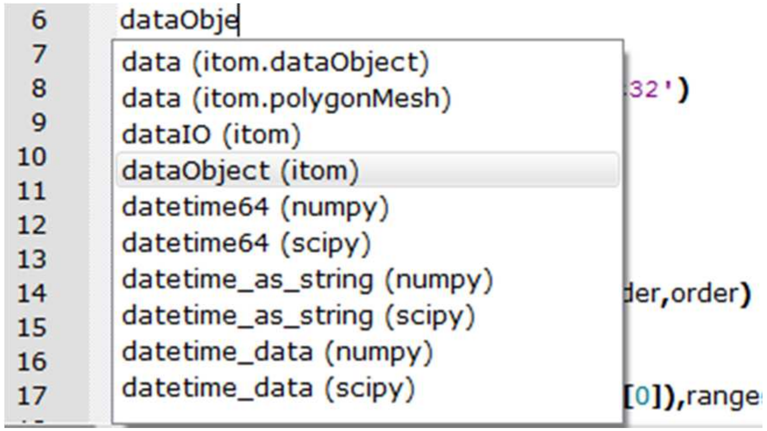

iTOM
File Help Script
File S
D:/ runk/MeasurementSystems
Size
Script Editor - D:/git-itom/build/itom/Qitom/pyGit2Header.py
File Edit Script Windows
pyGit2Header.py

```
if(len(error) > 0):
    raise RuntimeError("Calling GIT for local changes failed" + error)
else:
    if(len(out) < 1):
        repIsDirty = 0
    else:
        repIsDirty = 0
        if(out.find(b'??') > -1):
            repIsDirty = 2
        if(out.find(b'M ') > -1):
            repIsDirty = repIsDirty + 1

del out
del error

#pr = subprocess.Popen([gitLocation+'/git.exe','--git-dir='+repositoryPath+'/.git','--wo
pr = subprocess.Popen([gitLocation, '--git-dir='+repositoryPath+'/.git', '--
(log,error)=pr.communicate()
```

- Standard editor functionality
- Tabbing of multiple scripts
- Dockable into the main GUI
- Executes Scripts
- Full debugging functionality

Syntax Help and Auto Completion

- Auto completion
  (selection item with tab-key)

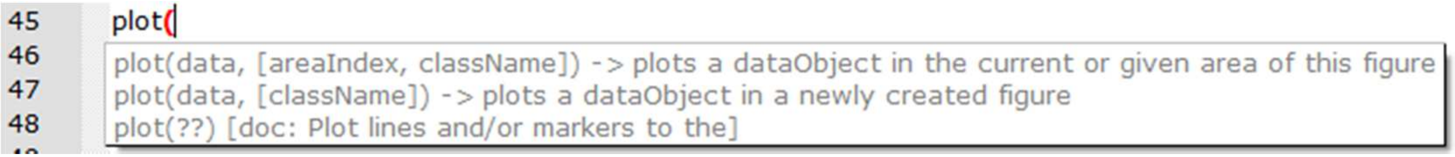


```
dataObje
```

```
data (itom.dataObject)
data (itom.polygonMesh)
dataIO (itom)
dataObject (itom)
datetime64 (numpy)
datetime64 (scipy)
datetime_as_string (numpy)
datetime_as_string (scipy)
datetime_data (numpy)
datetime_data (scipy)
```

```
32')
der,order)
[0]),range
```

- Syntax help



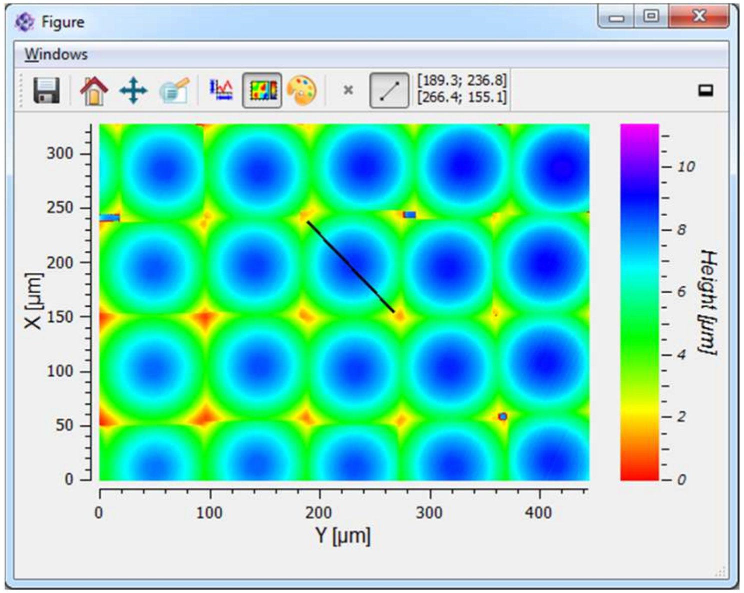
```
plot(
```

```
plot(data, [areaIndex, className]) -> plots a dataObject in the current or given area of this figure
plot(data, [className]) -> plots a dataObject in a newly created figure
plot(??) [doc: Plot lines and/or markers to the]
```

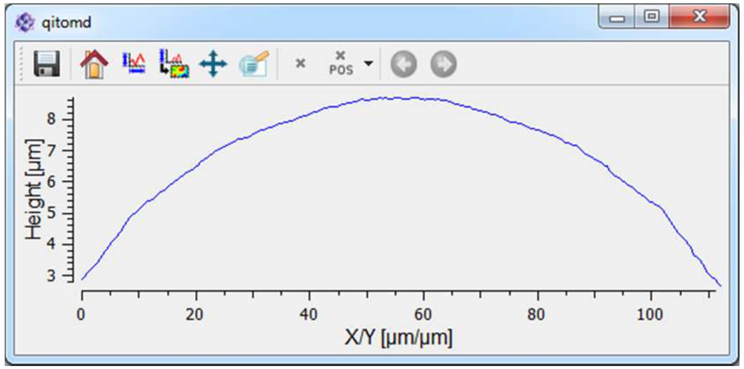
Set various syntax-files (for important Python modules) in itom’s property editor in order to enable these features.

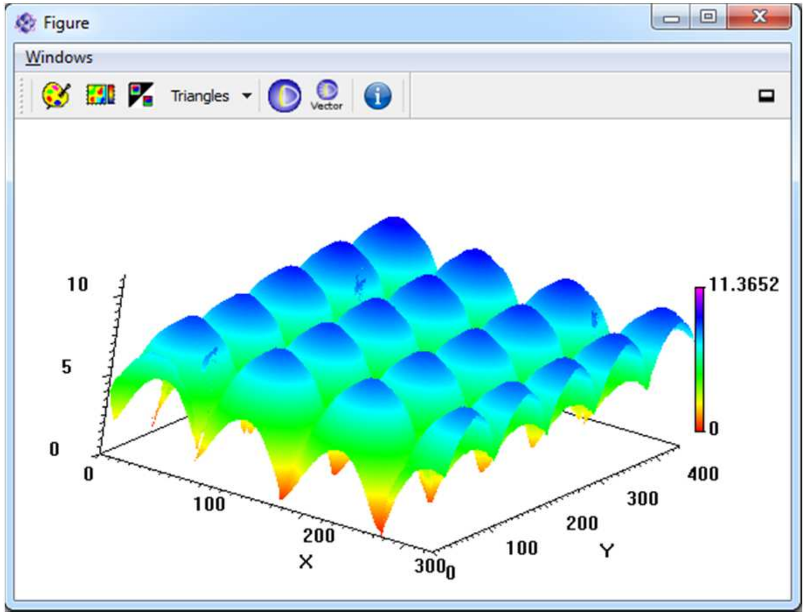
Plots

- 1D, 2D, 2.5D plots
- Custom windows can be implemented
- Displayed in
  - A separate window
  - Docked into the main GUI
  - Integrated into a custom GUI

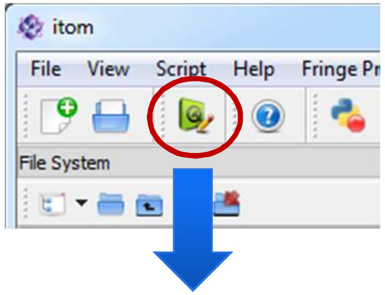


dependent 1D-line plot

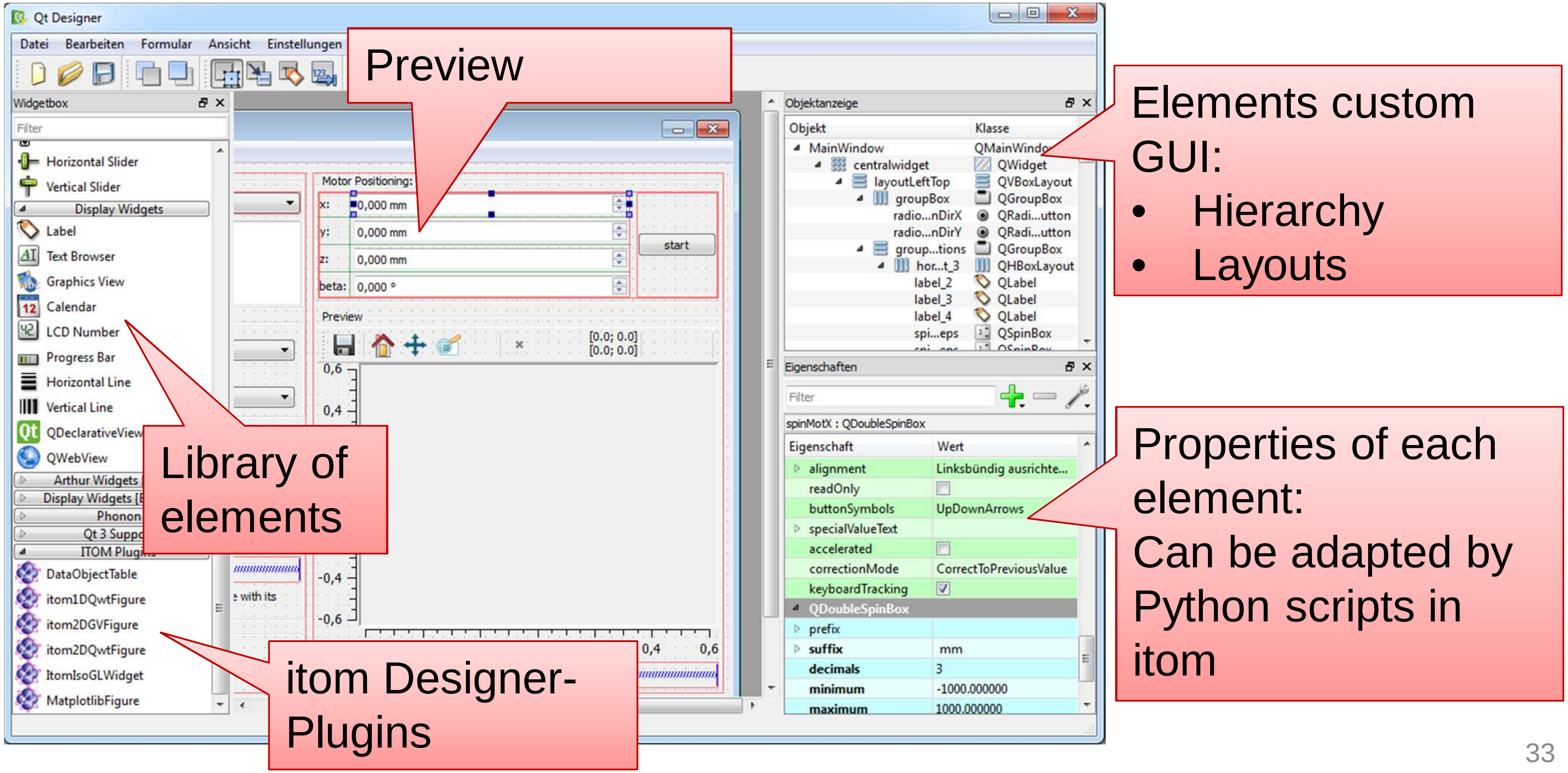




Custom GUIs (Qt Designer)



- Design of custom GUIs in the external Qt Designer WYSIWYG tool (drag&drop).
- Events created by the GUI (button click) can be linked to Python functions



Eigenschaft	Wert
alignment	Linksbündig ausrichte...
readOnly	□
buttonSymbols	UpDownArrows
specialValueText	
accelerated	□
correctionMode	CorrectToPreviousValue
keyboardTracking	☑
QDoubleSpinBox	
prefix	
suffix	mm
decimals	3
minimum	-1000,000000
maximum	1000,000000

Preview

Library of elements

itom Designer-Plugins

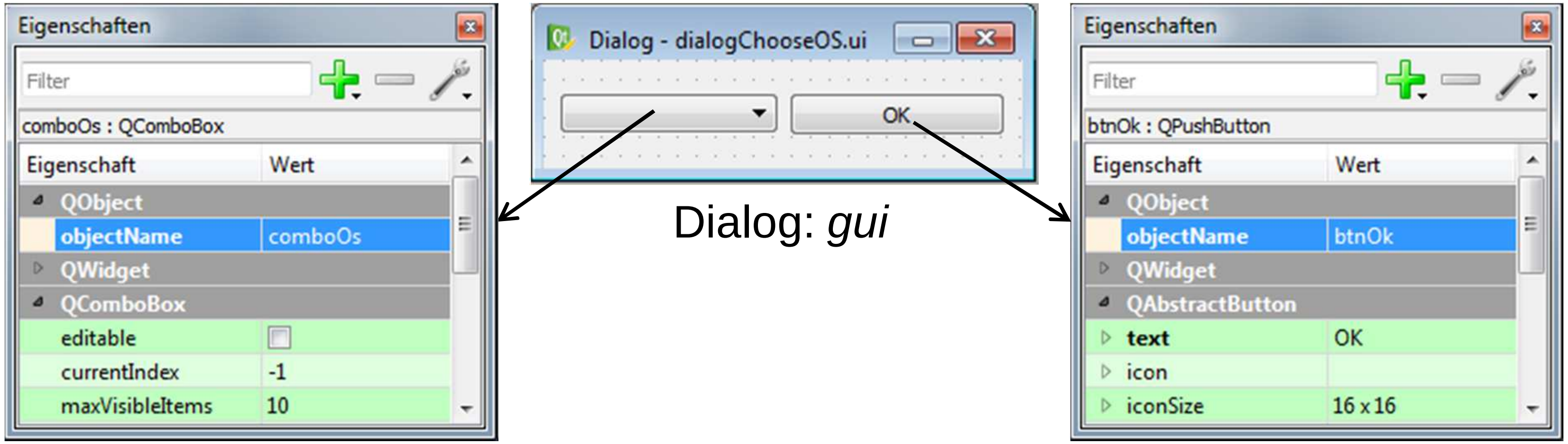
Elements custom GUI:

- Hierarchy
- Layouts

Properties of each element:
Can be adapted by Python scripts in itom

Custom GUIs (Qt Designer)

Dialog design with Qt Designer:



Eigenschaften

Filter

comboOs : QComboBox

Eigenschaft	Wert
QObject	
objectName	comboOs
QWidget	
QComboBox	
editable	□
currentIndex	-1
maxVisibleItems	10

Dialog: *gui*

Eigenschaften

Filter

btnOk : QPushButton

Eigenschaft	Wert
QObject	
objectName	btnOk
QWidget	
QAbstractButton	
text	OK
icon	
iconSize	16 x 16

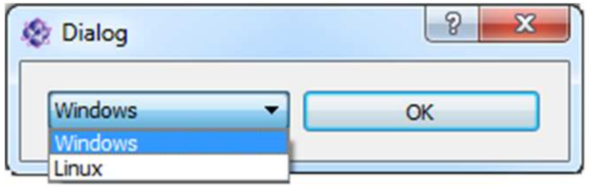
Script logic with python:

1. Access properties

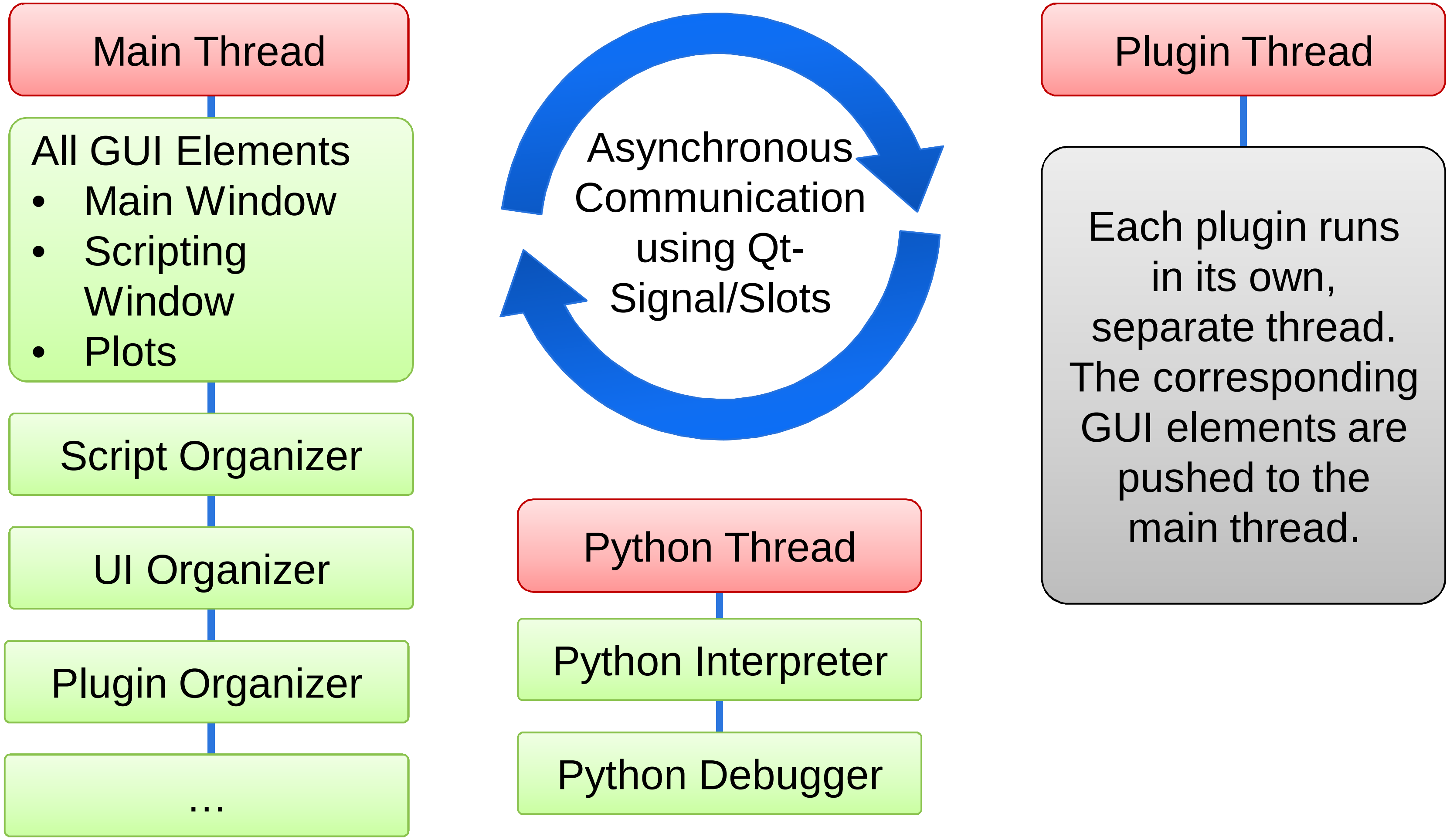
```
gui.btnOk[“text”] = “OK”
gui.comboOs.call(“addItems”, [“Windows”,”Linux”])
```

2. Connect signals with Python methods

```
def clickMe():
    print(“operating system”, gui.comboOs[“currentText”])

gui.btnOk.connect(“clicked()”, clickMe)
```



Multithreading



Main Thread

All GUI Elements
- Main Window
- Scripting Window
- Plots

Script Organizer

UI Organizer

Plugin Organizer

…

Asynchronous Communication using Qt-Signal/Slots

Python Thread

Python Interpreter

Python Debugger

Plugin Thread

Each plugin runs in its own, separate thread. The corresponding GUI elements are pushed to the main thread.

Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**'s Built-in Array Class
- Documentation and Help

License

- **itom** (main application) is “Open Source” (**LGPL**)

- **itom-SDK** (resources common to the main application and plugins) are distributed under the **LGPL-licence** + **itom-exception**. The itom exception allow the inclusion and linking of additional components independent of those components licensing against all data included in the SDK.

- **Plugins** can be subject to any (including proprietary) licenses. The ITO offers a number of generic plugins under the **LGPL**.

- **Designer-Plugins** (plots…), similarly, can be subject to any licenses.

Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

Data Object

Goal:

- Different basic types of data (including complex)
- Processing of large, multi-dimensional data sets (series of images)
- Compatible with Matlab, Numpy, OpenCV

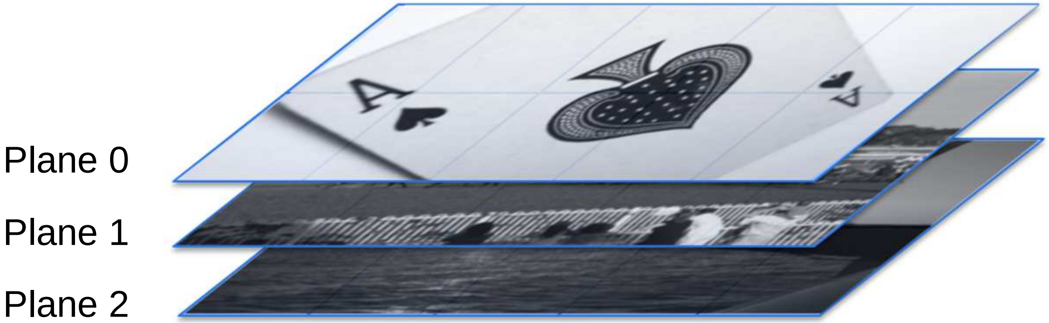
Implementation:

- *DataObject* very similar to OpenCV data structures
- Basic data types supported: *int8, uint8, int16, uint16, int32, uint32, float, double, complex(float), complex(double)*
- *DataObject* supports tags (axes units, descriptions, title…)

Data Storage

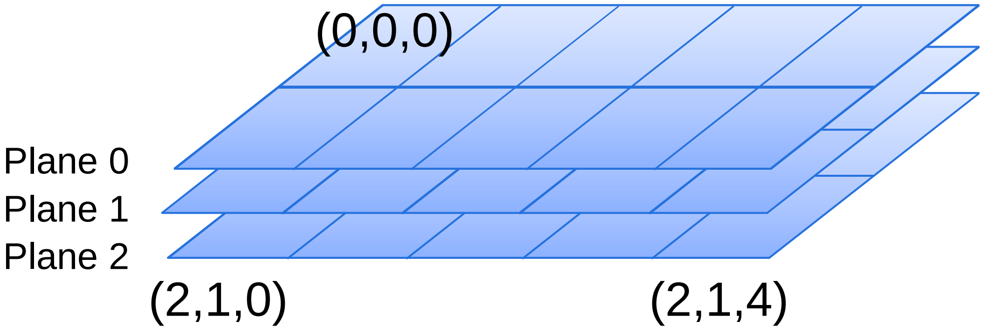
Series of 2D-images

3D data stack

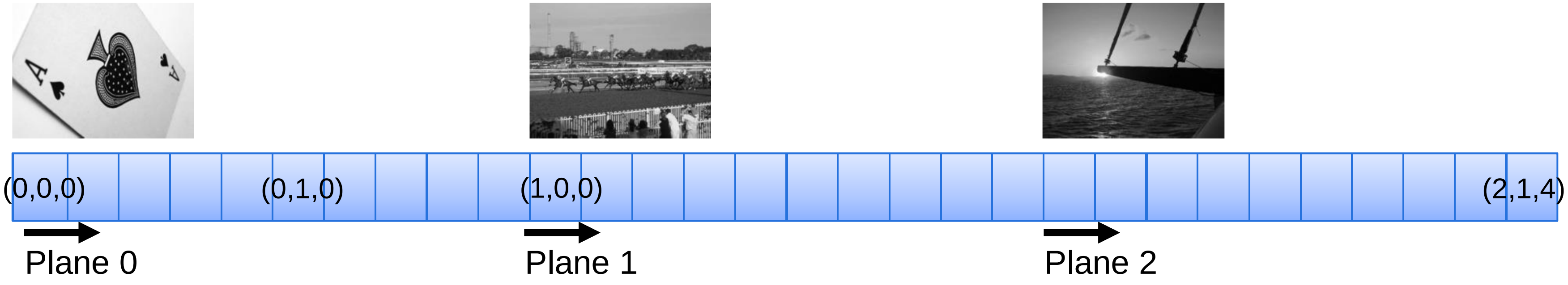


Data Storage

Assume: Series of 2D-images *(3 x 2 x 5)*



C / Matlab: continuous chunk of memory

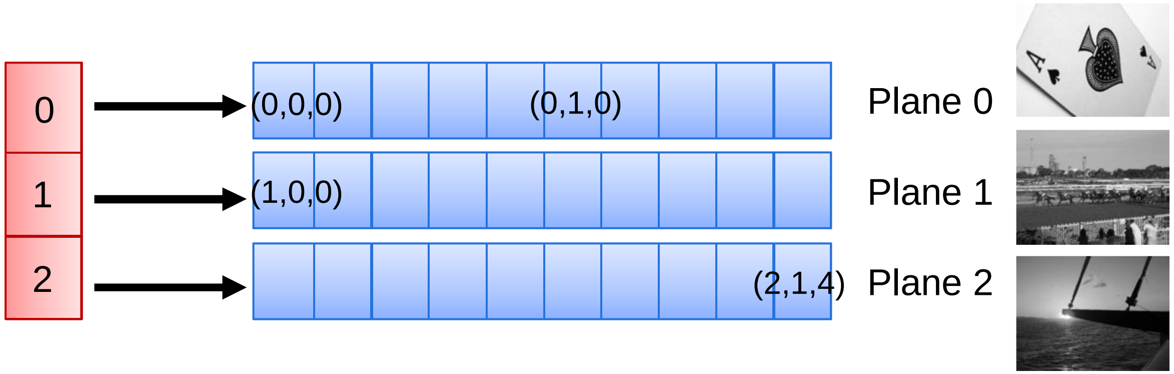


+ Uniform, quick and easy access to multi-dimensional arrays

– Memory allocation error for „big“ arrays

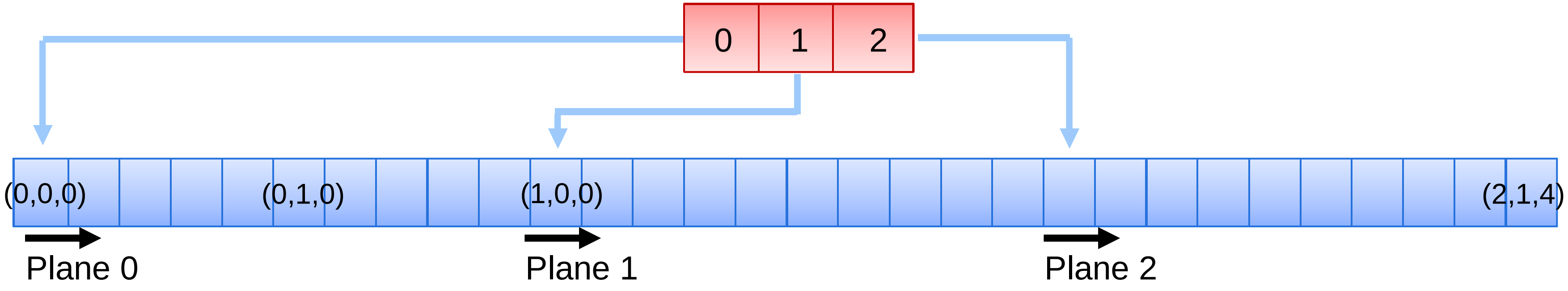
Data Storage

DataObject:



+ Less allocation errors due to distributed chunks of memory
– Slightly more complex access to memory

DataObject (continuous): Compatibility to C-style arrays

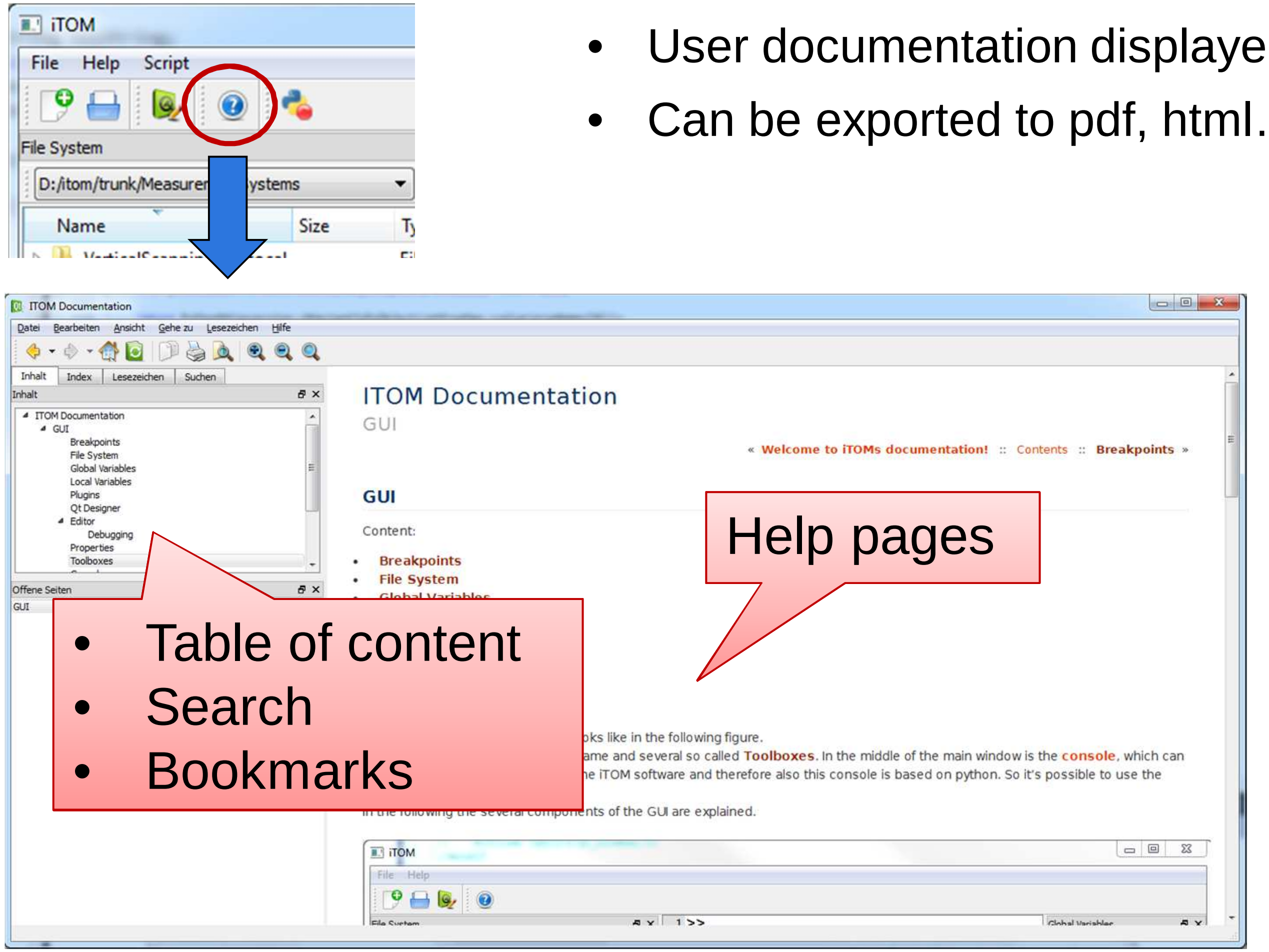


Agenda

- Motivation. Why **itom**?
- Features
- Script Language Python
- Modular Plugin System
- The Graphical User Interface
- Licensing
- DataObject – **itom**’s Built-in Array Class
- Documentation and Help

User Documentation

- User documentation displayed with Qt Assistant
- Can be exported to pdf, html…



ITOM Documentation

GUI

« Welcome to iTOMs documentation! :: Contents :: Breakpoints »

GUI

Content:

- Breakpoints
- File System

oks like in the following figure.
ame and several so called **Toolboxes**. In the middle of the main window is the **console**, which can
e iTOM software and therefore also this console is based on python. So it's possible to use the
ents of the GUI are explained.

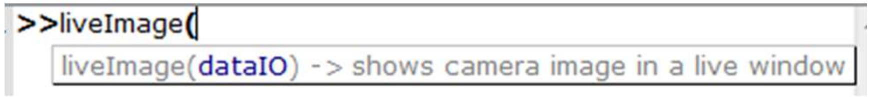
- Table of content
- Search
- Bookmarks

Help pages

itom.bitbucket.org/latest/docs

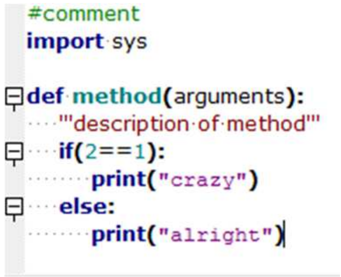
Additional User Help within Python

1. Syntax help and auto completion in the Python editor



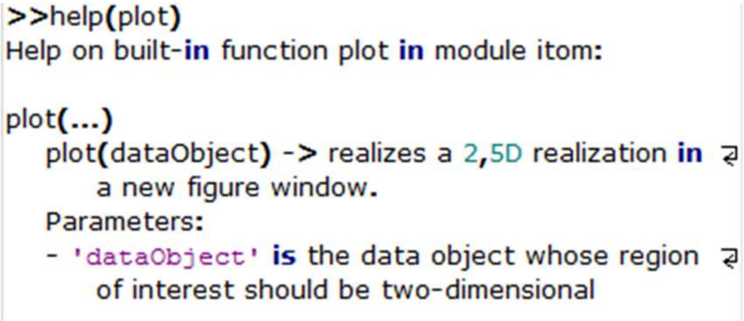
```
>>liveImage(
  liveImage(dataIO) -> shows camera image in a live window
```

2. Customizable, context sensitive syntax highlighting



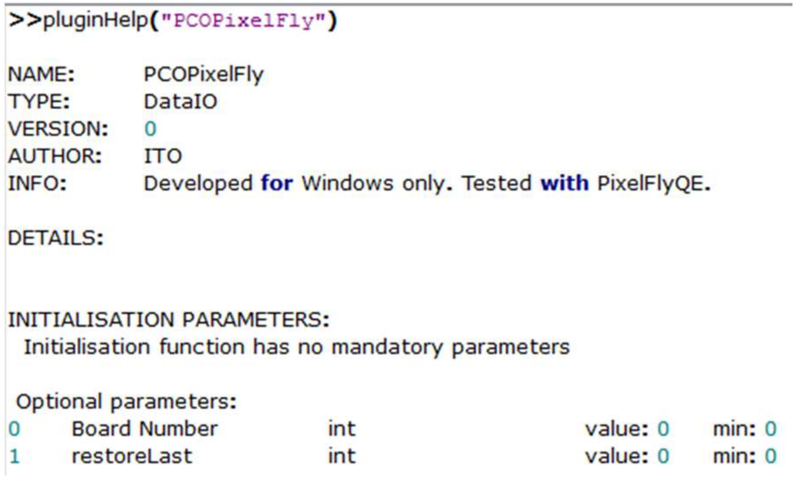
```
#comment
import sys

def method(arguments):
    '''description of method'''
    if(2==1):
        print("crazy")
    else:
        print("alright")
```

3. Python-internal help system using the command *help(…)*



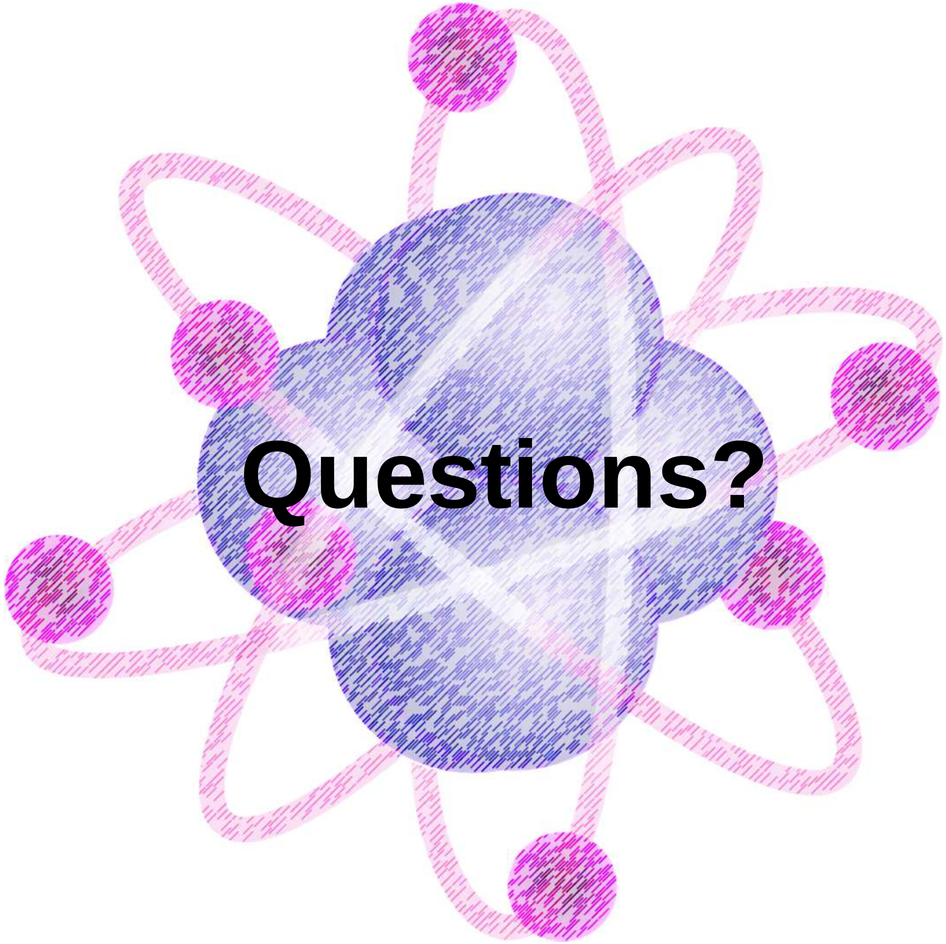
```
>>help(plot)
Help on built-in function plot in module itom:

plot(...)
    plot(dataObject) -> realizes a 2,5D realization in
        a new figure window.
    Parameters:
    - 'dataObject' is the data object whose region
        of interest should be two-dimensional
```

4. Additional information and help about available plugins or algorithms using the commands *pluginHelp(…), filterHelp(…), widgetHelp(…)*

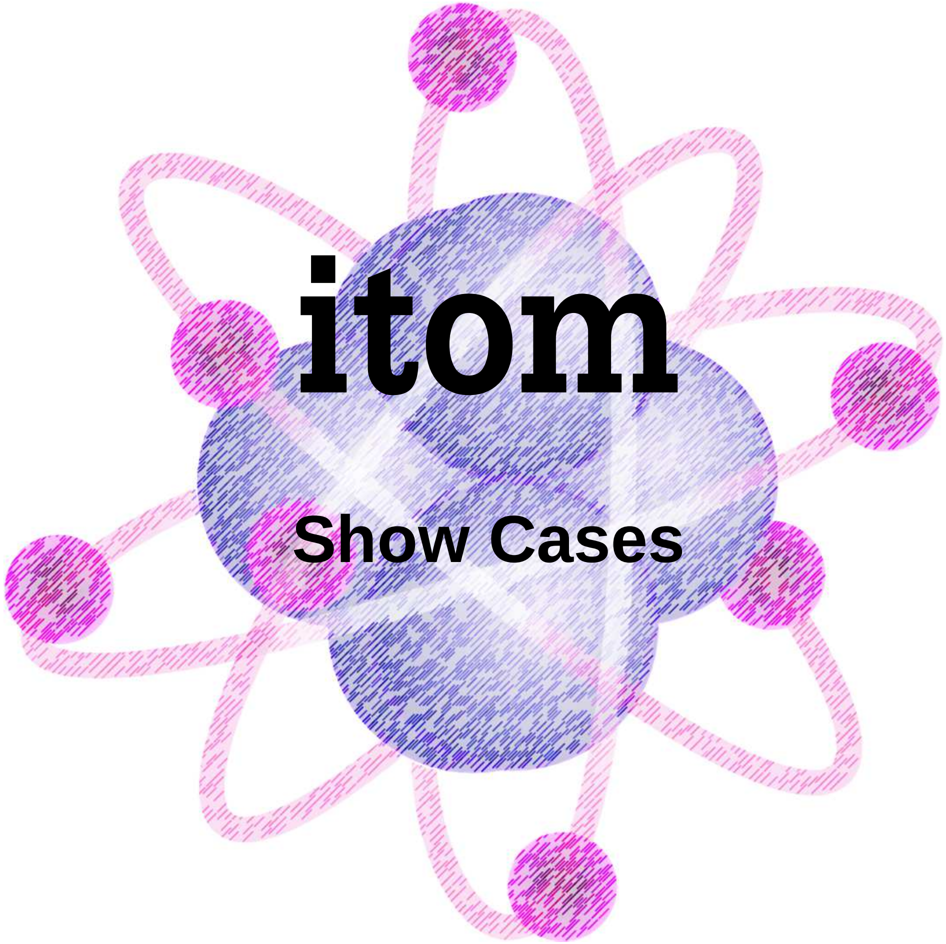


```
>>pluginHelp("PCOPixelFly")

NAME:       PCOPixelFly
TYPE:       DataIO
VERSION:    0
AUTHOR:     ITO
INFO:       Developed for Windows only. Tested with PixelFlyQE.

DETAILS:


INITIALISATION PARAMETERS:
 Initialisation function has no mandatory parameters

 Optional parameters:
0    Board Number        int                    value: 0    min: 0
1    restoreLast         int                    value: 0    min: 0
```

itom



Questions?



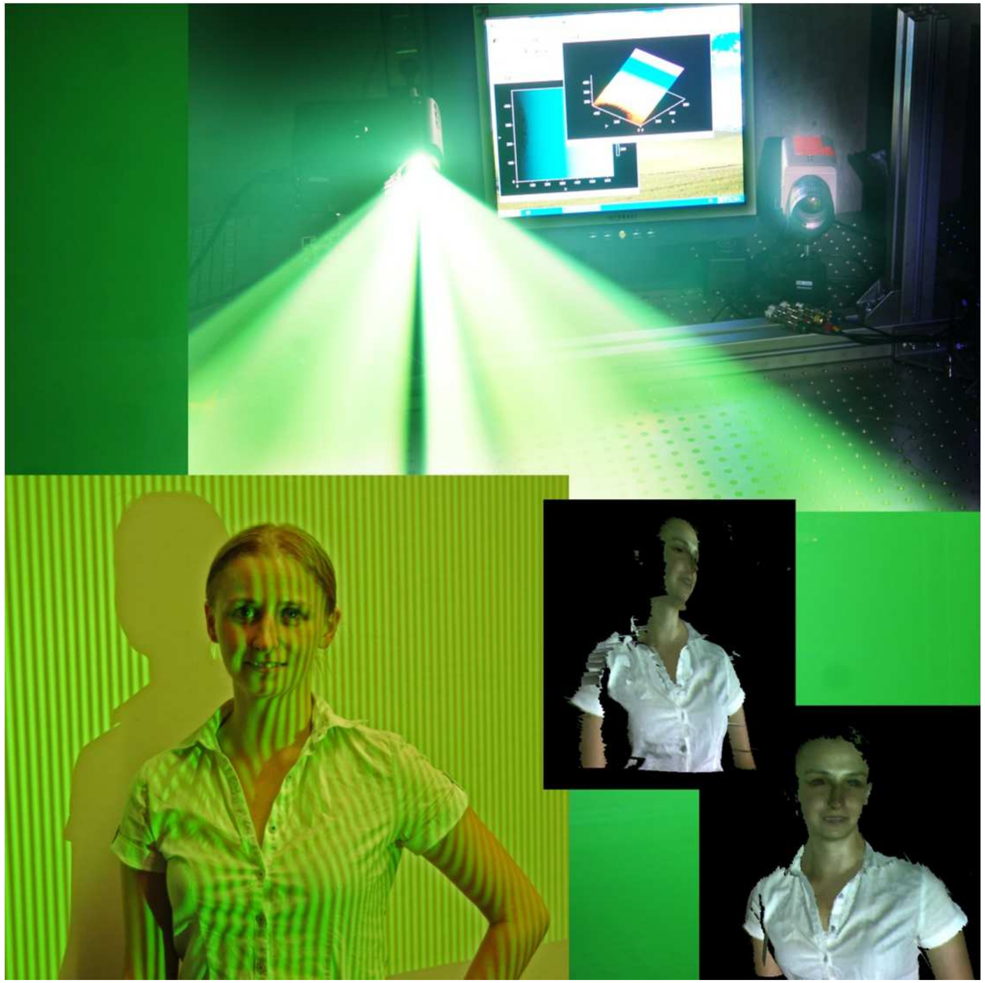
itom

Show Cases

Show-Case I: Fringe Projection

Situation

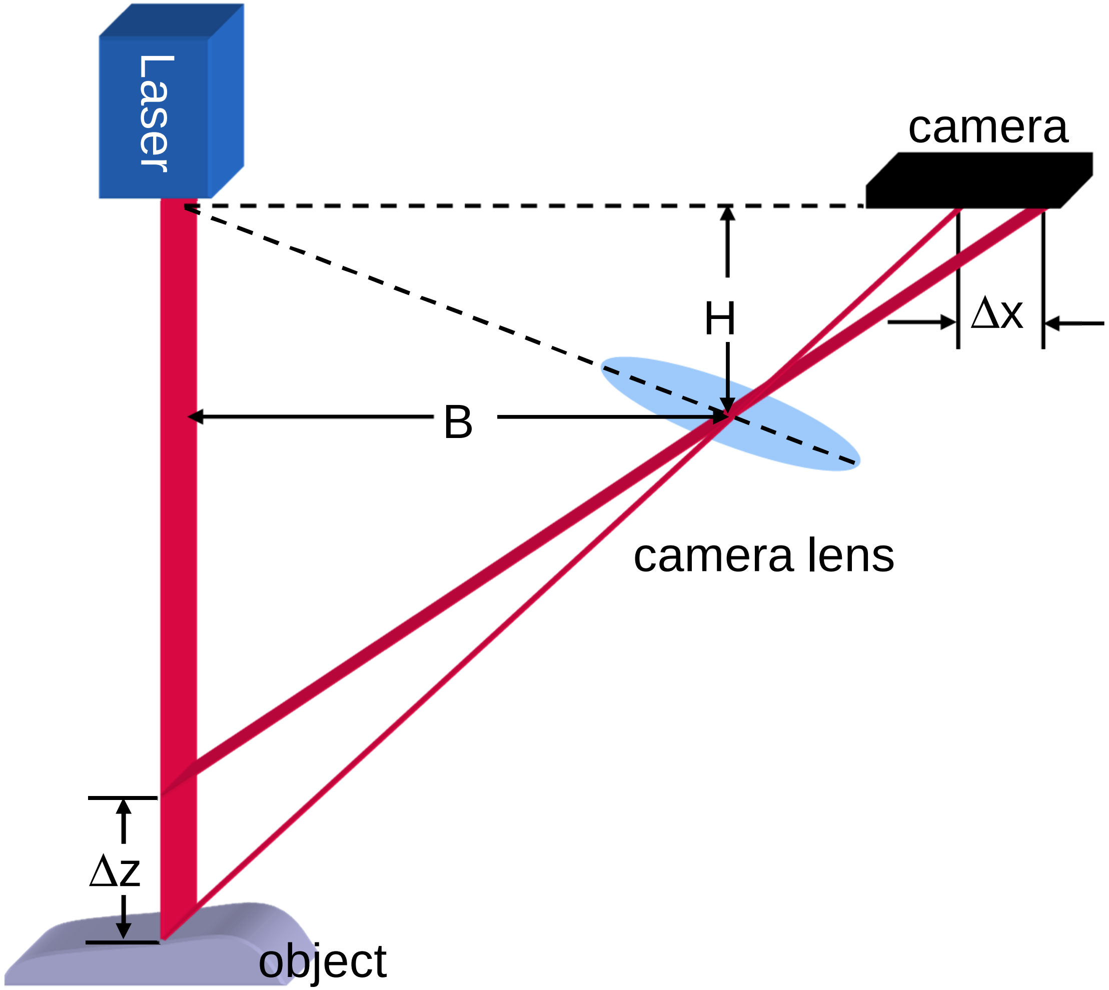
A flexible fringe projection setup (structured light) for student projects and public presentation is been developed

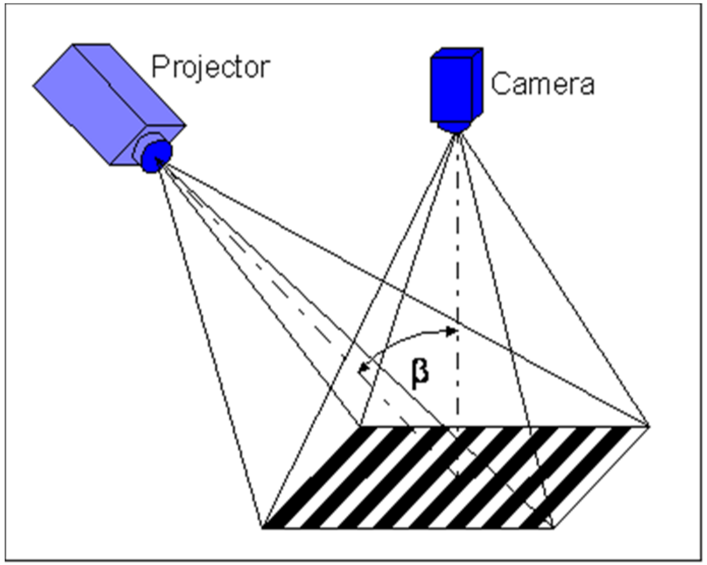


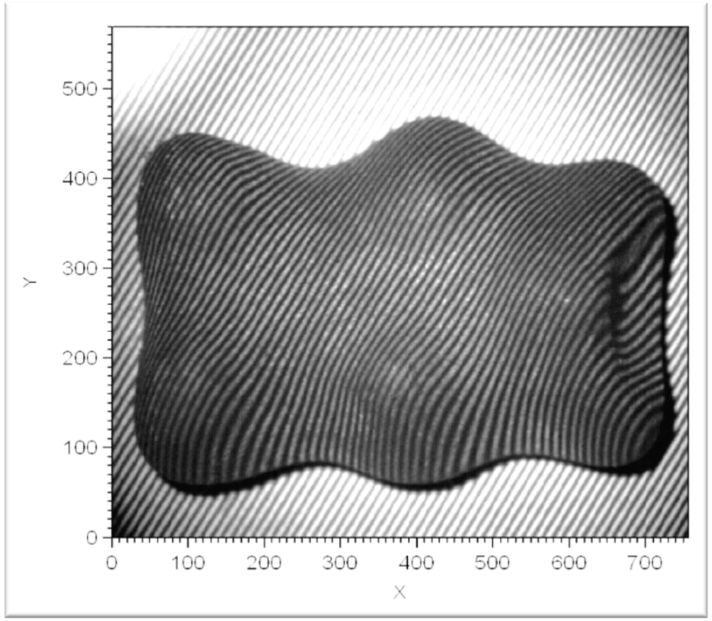
Objective

- Provide a GUI for such a system to demonstrate the function
- Allow students to run batch processes for system characterization
- Provide flexibility to change between several evaluation or calibration methods and hardware components.

Show-Case I: Triangulation



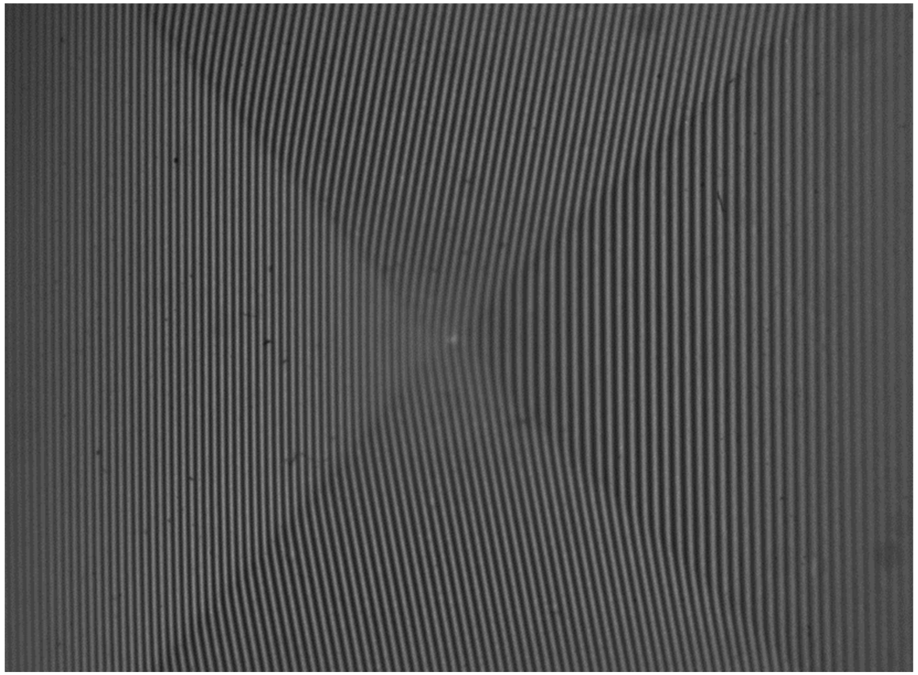




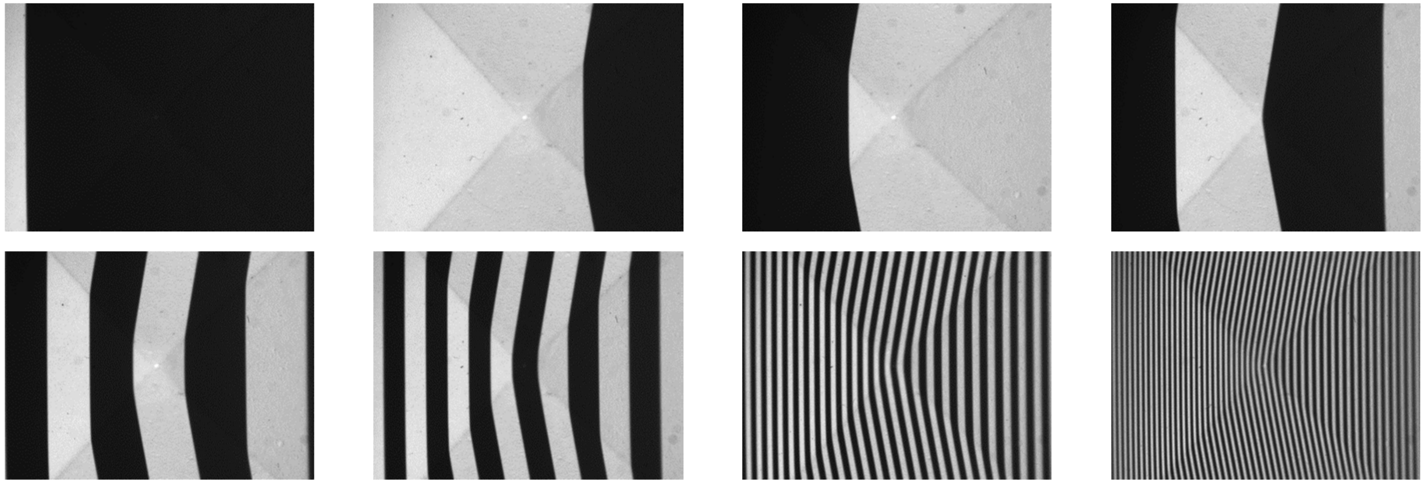
Show-Case I: Structured Illumination

LCOS-Display or DMD-Projector

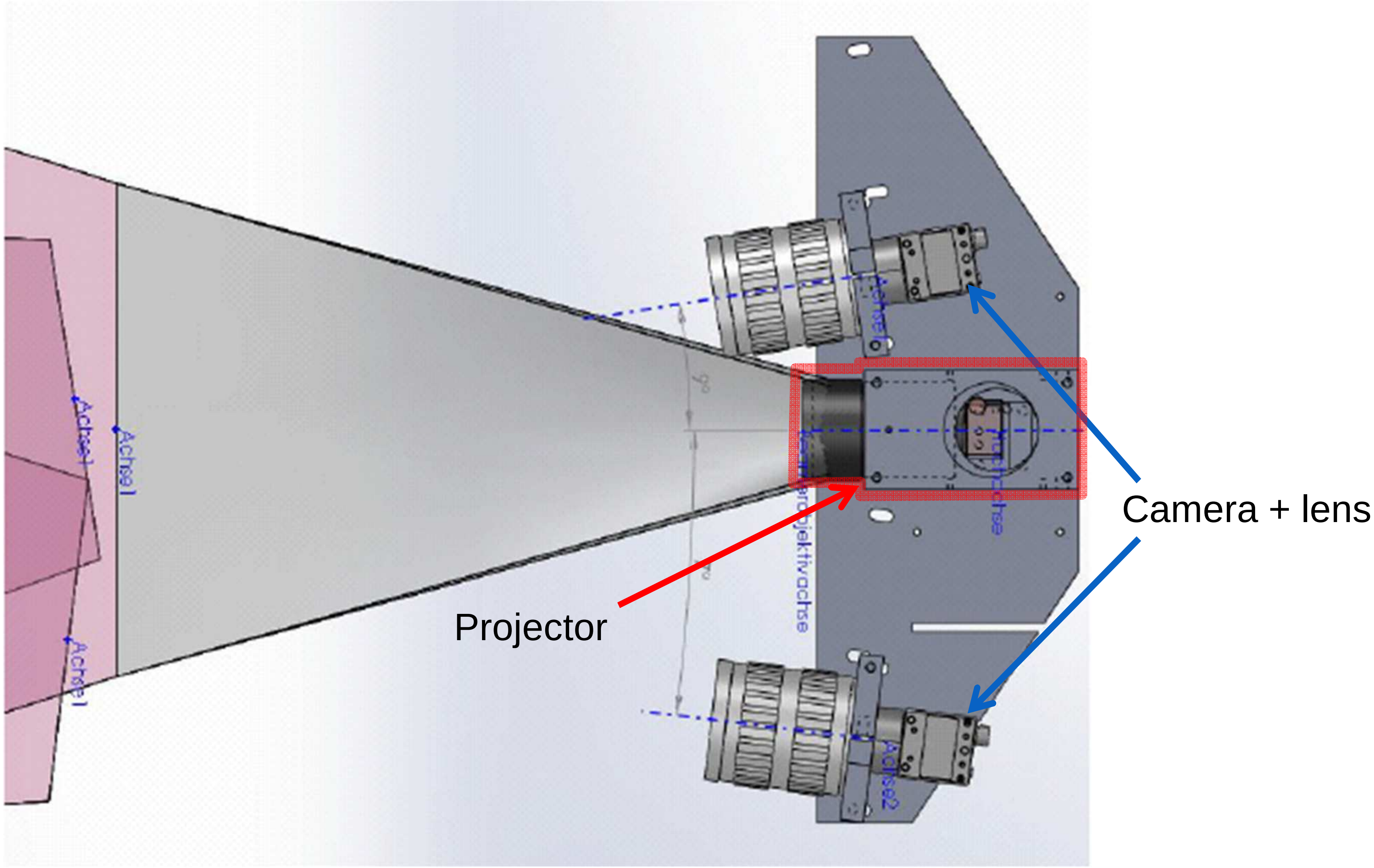
Cosine-fringes (mod 2PI)



Grey-Code → absolute coding



Show-Case I: Basic Set-up



Projector

Camera + lens

Show-Case II: MacroSim

Situation

- An open source GPU based ray-tracing tool has been developed at ITO
- The native tool is command-line based

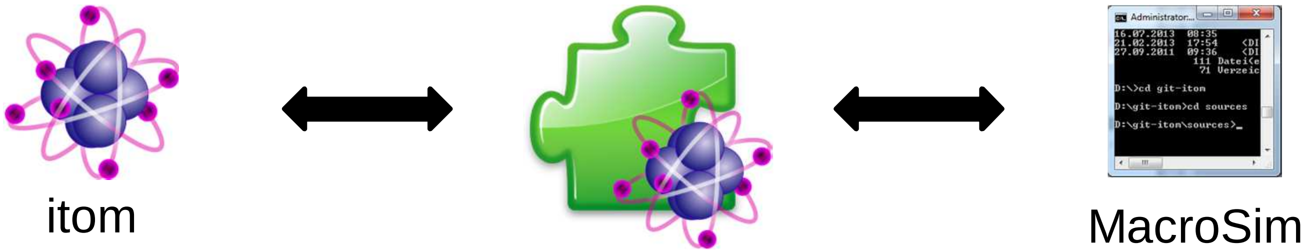
Objective

- Provide a GUI for MacroSim in order to simplify the creation of new scenes and execute simulations
- For the future it should be possible to run both the real setup and its corresponding simulation with the same tool.

Show-Case II: MacroSim

Solution

- Create an **itom** software plugin that provides its own GUI and communicates with the tool **MacroSim**



```
16.07.2013  08:35
21.02.2013  17:54    <DI
27.09.2011  09:36    <DI
             111 Datei(e
              71 Verzeic

D:\>cd git-itom

D:\git-itom>cd sources

D:\git-itom\sources>_
```

itom

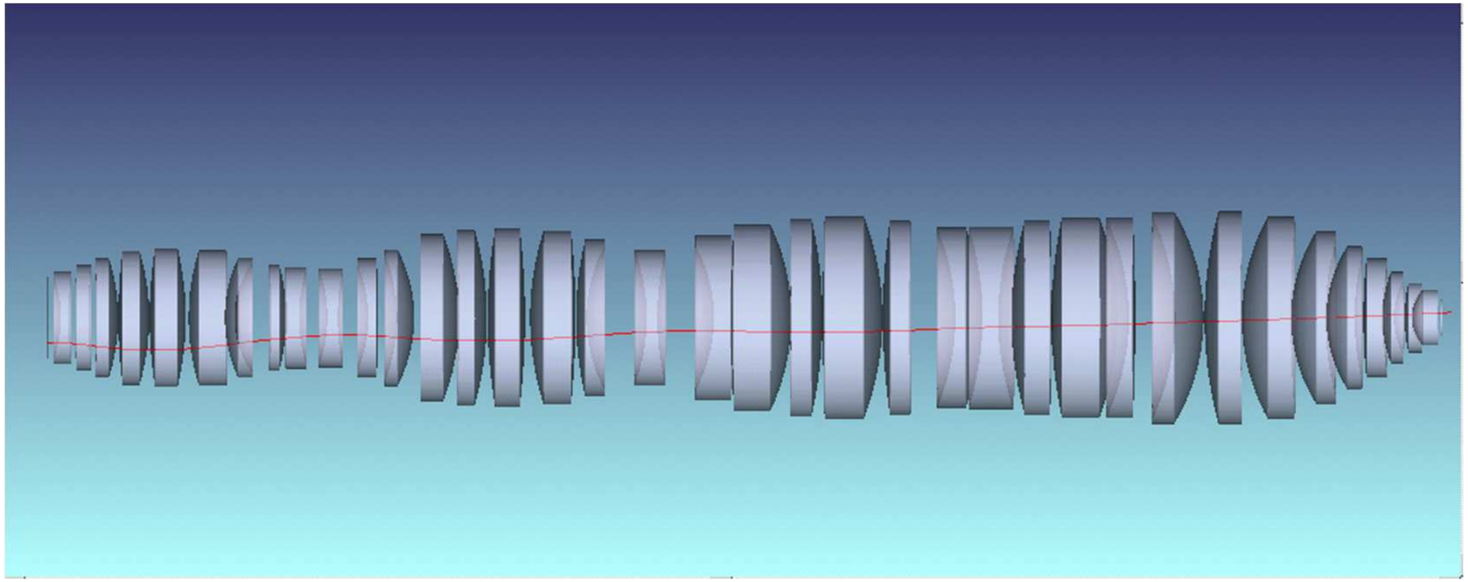
MacroSim
Plugin

MacroSim
Tracer

- ✓ MacroSim can use functionalities contained in itom
- ✓ Tracer can also be started by Python
- ✓ Batch execution possible using appropriate Python script
- ✓ Results of tracer are available in itom

Raytracing: A versatile tool

Raytracing is perfectly linear



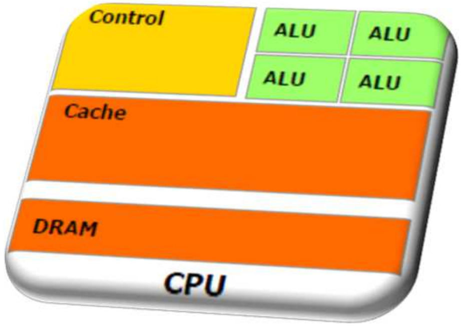
→ Raytracing is perfectly parallelizable

Modell adaptiert von US Patent Publication US 6522484 B1, K-H Schuster, Carl-Zeiss-Stiftung, (1999)

Parallelization of Raytracing

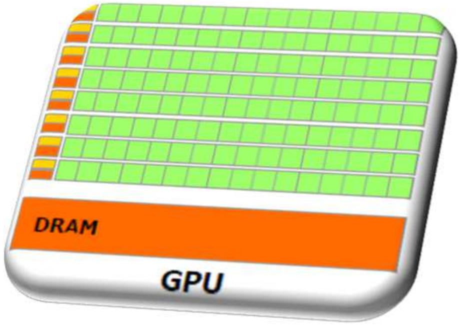
- **CPU-Parallelization**

- very flexible
- straightforward implementation
- More than 4 cores quickly become expensive



- **GPU-Parallelization**

- Restriction to Thread Coherence
- Specific Implementation
- Standard GPUs come with 200-500 cores

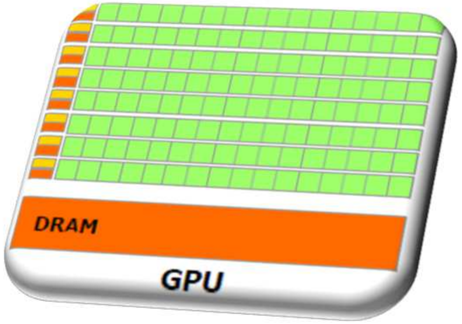


Images taken from CUDA programming guide 3.2

Parallelization of Raytracing

- **GPU-Parallelization**

- Restriction to Thread Coherence
- Specific Implementation
- Standard GPUs come with 200-500 cores

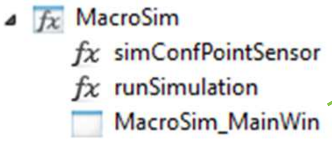


- **GPU accelerated Tool: MacroSim**

- Based on nVidia® OptiX™ acceleration engine
- Plugin to ITOs itom software
- imports glass catalog from Zemax®
- Published under GPL at https://bitbucket.org/itom/macrosim
- „An open source GPU-accelerated ray tracer for optical simulation“, submitted for publication to Optical Engineering.

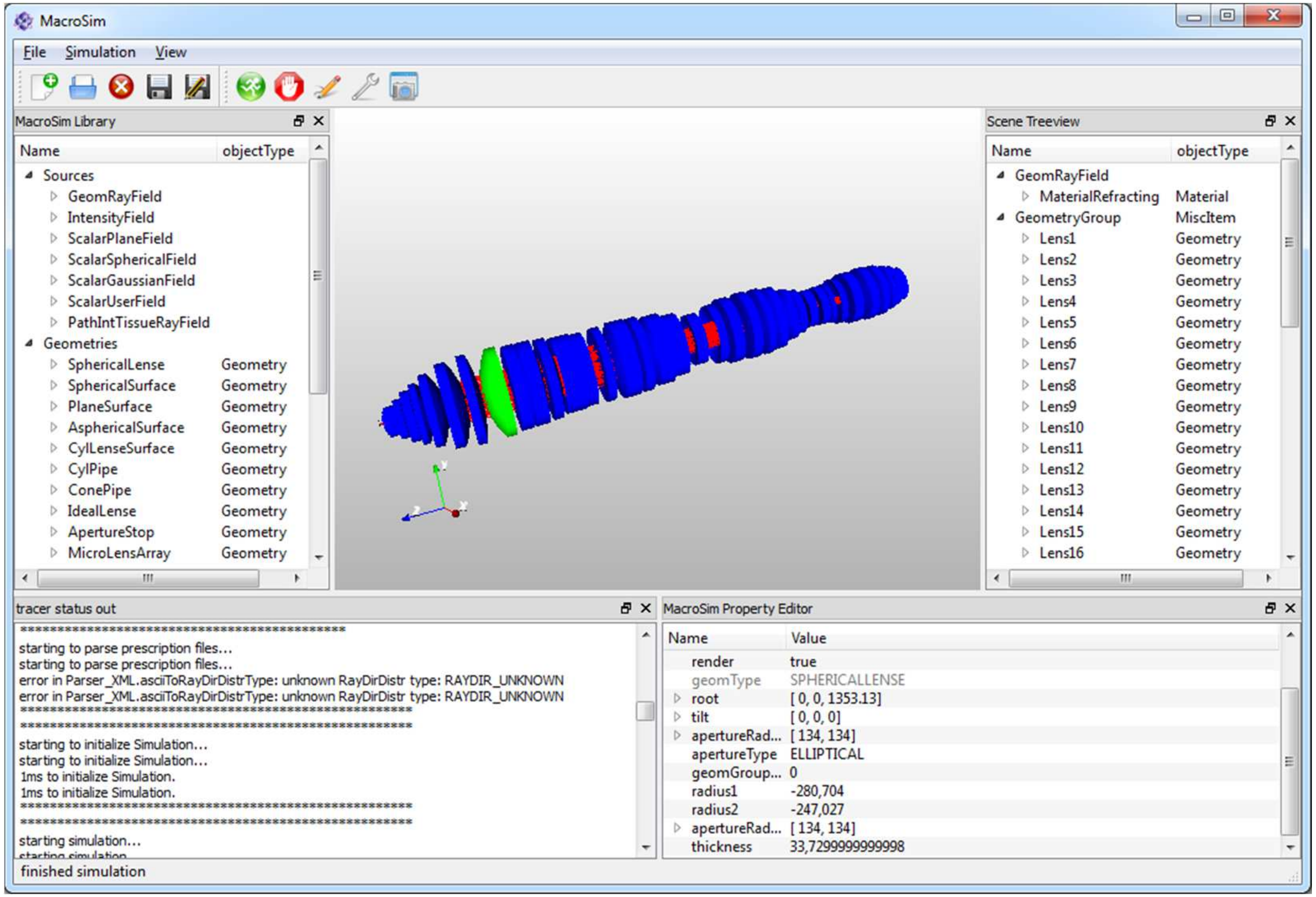
Images taken from CUDA programming guide 3.2

Parallelization of Raytracing



- MacroSim
  - simConfPointSensor
  - runSimulation
  - MacroSim_MainWin

MacroSim Plugins contains one GUI and some callable functions



MacroSim Library

Name	objectType
Sources	
GeomRayField	
IntensityField	
ScalarPlaneField	
ScalarSphericalField	
ScalarGaussianField	
ScalarUserField	
PathIntTissueRayField	
Geometries	
SphericalLense	Geometry
SphericalSurface	Geometry
PlaneSurface	Geometry
AsphericalSurface	Geometry
CylLenseSurface	Geometry
CylPipe	Geometry
ConePipe	Geometry
IdealLense	Geometry
ApertureStop	Geometry
MicroLensArray	Geometry

Scene Treeview

Name	objectType
GeomRayField	
MaterialRefracting	Material
GeometryGroup	MiscItem
Lens1	Geometry
Lens2	Geometry
Lens3	Geometry
Lens4	Geometry
Lens5	Geometry
Lens6	Geometry
Lens7	Geometry
Lens8	Geometry
Lens9	Geometry
Lens10	Geometry
Lens11	Geometry
Lens12	Geometry
Lens13	Geometry
Lens14	Geometry
Lens15	Geometry
Lens16	Geometry

tracer status out

```
******************************************
starting to parse prescription files...
starting to parse prescription files...
error in Parser_XML.asciiToRayDirDistrType: unknown RayDirDistr type: RAYDIR_UNKNOWN
error in Parser_XML.asciiToRayDirDistrType: unknown RayDirDistr type: RAYDIR_UNKNOWN
**************************************************
**************************************************
starting to initialize Simulation...
starting to initialize Simulation...
1ms to initialize Simulation.
1ms to initialize Simulation.
**************************************************
**************************************************
starting simulation...
starting simulation
```

MacroSim Property Editor

Name	Value
render	true
geomType	SPHERICALLENSE
root	[0, 0, 1353.13]
tilt	[0, 0, 0]
apertureRad...	[134, 134]
apertureType	ELLIPTICAL
geomGroup...	0
radius1	-280,704
radius2	-247,027
apertureRad...	[134, 134]
thickness	33,7299999999998

finished simulation

Interaction with itom

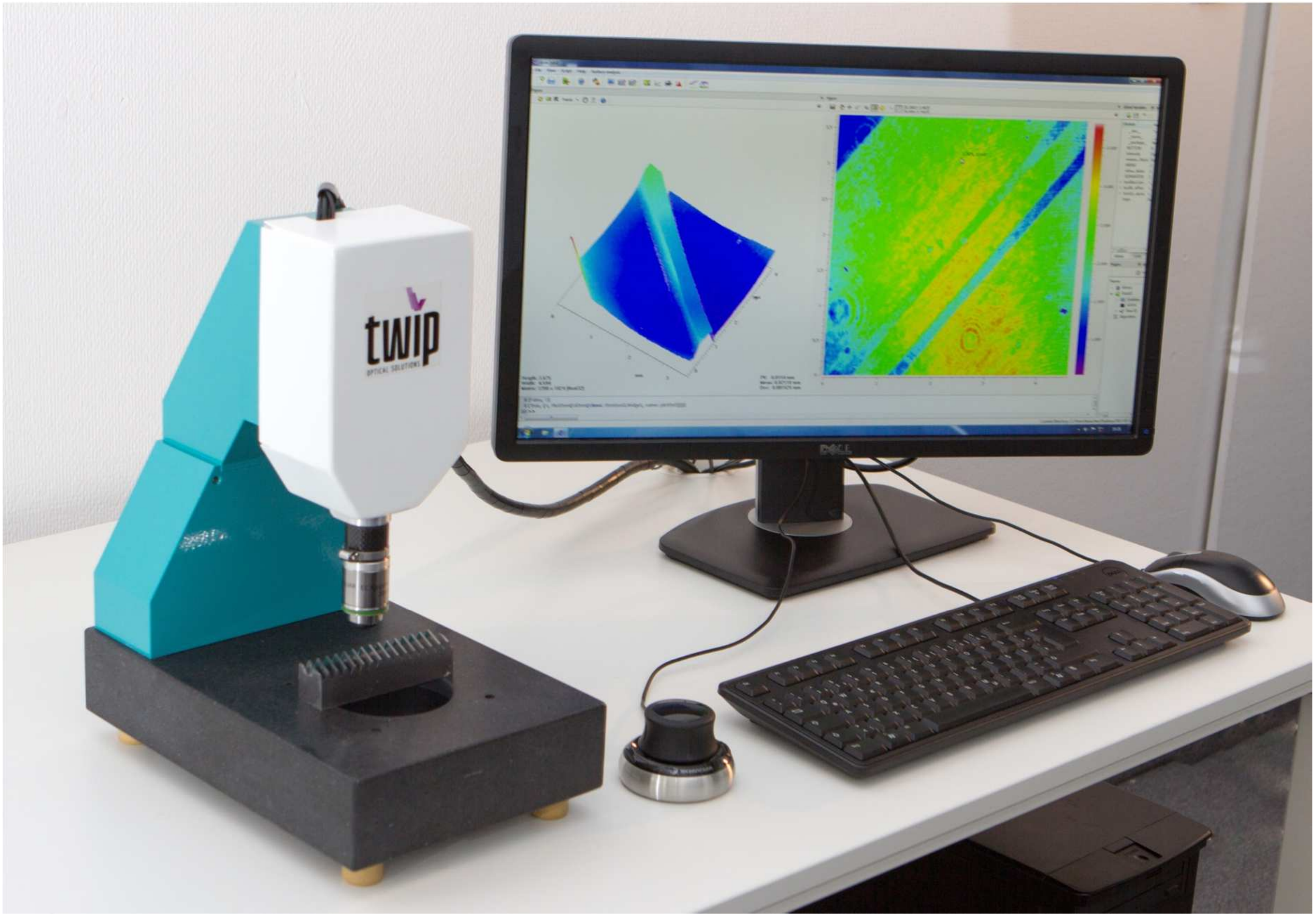
GUI based

- Start MacroSim GUI by Python command (*createNewPluginWidget*)
- Start simulation manually
- GUI emits a signal with the final detector matrix (dataObject)
- Connect a Python function to this signal (called when simulation done)

Script based

- Optional: Start MacroSim GUI and create scene (XML-file)
- Call function *runSimulation* of MacroSim plugin and pass XML-file (simulation is executed)
- The function finally returns the detector matrix as dataObject

Show-Case III: Confocal Microscopy



Show-Case III: Confocal Microscopy

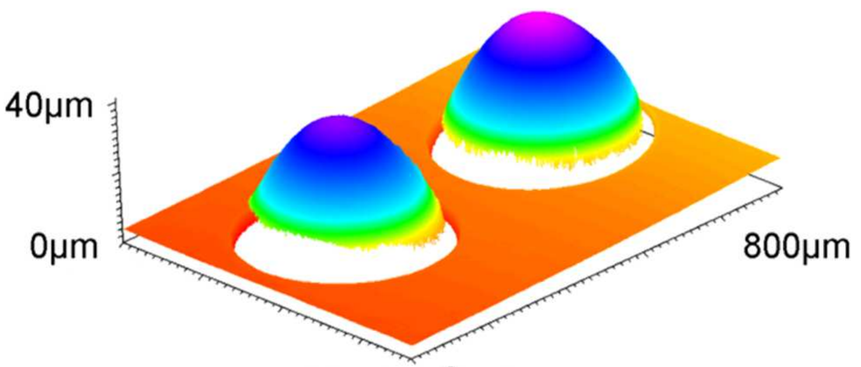
Situation

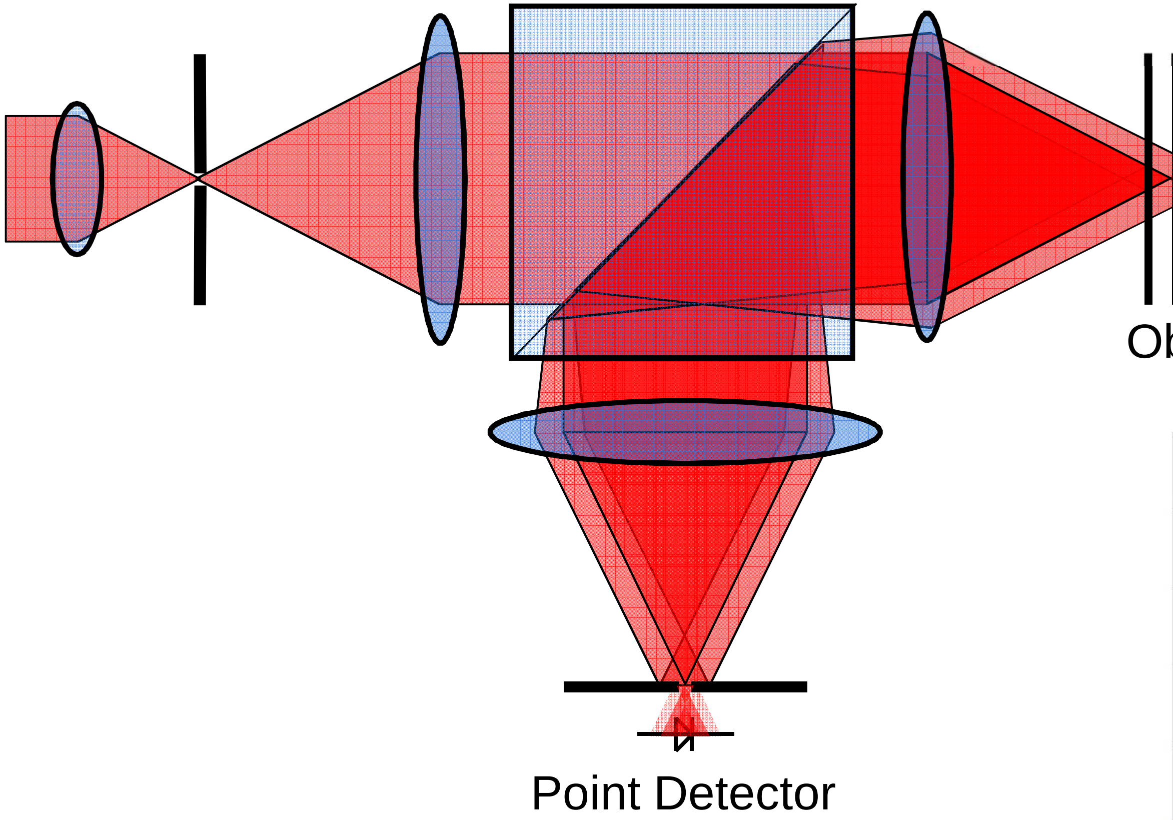
- A confocal microscope is being developed by Twip Os (spin-off of ITO)

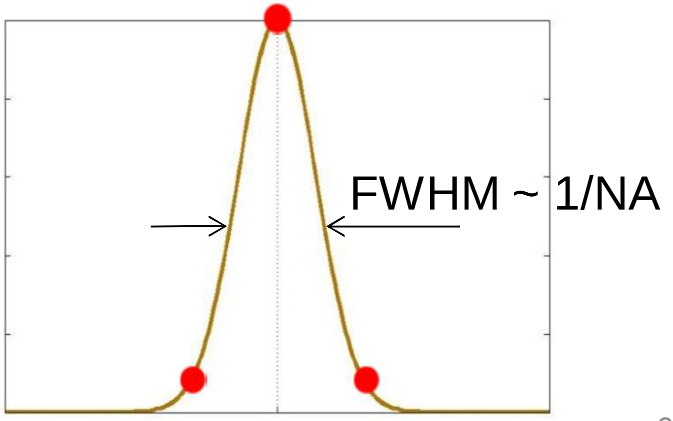
Objective

- **itom** should be used to…
  - control the measurement process
  - provide a user-friendly control panel
  - visualize the results
  - provide functionality for data evaluation (roughness, alignment, geometrical fitting…)

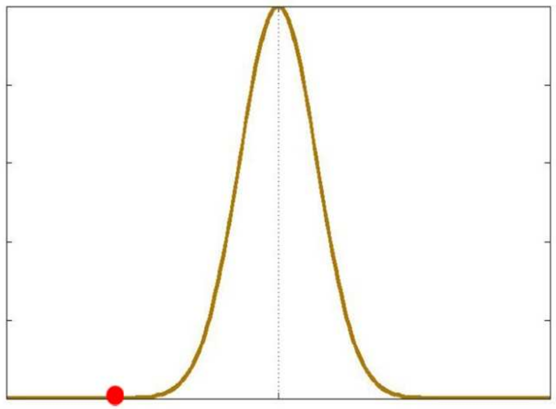
Show-Case III: Confocal Microscopy

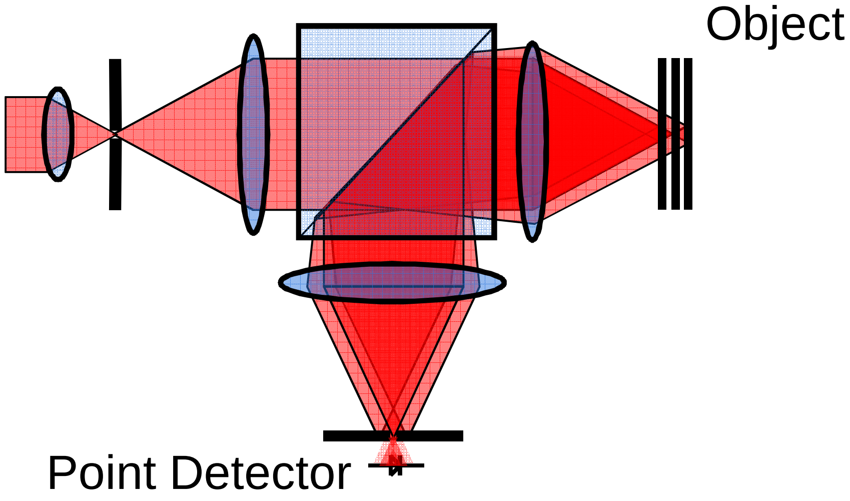


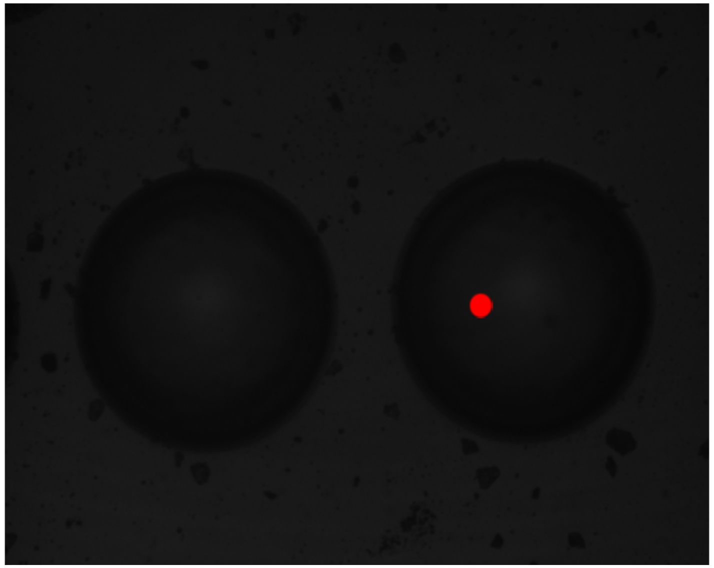


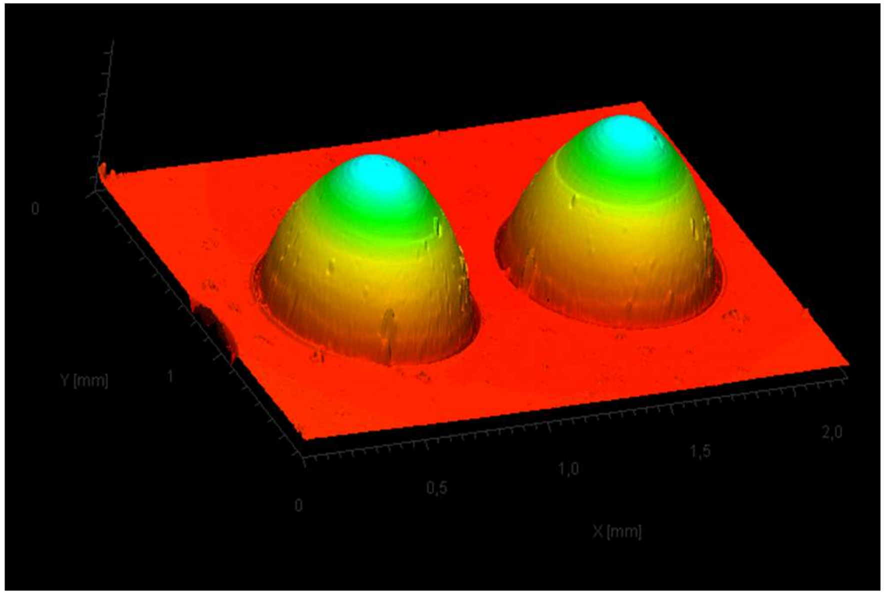


Show-Case III: Confocal Microscopy



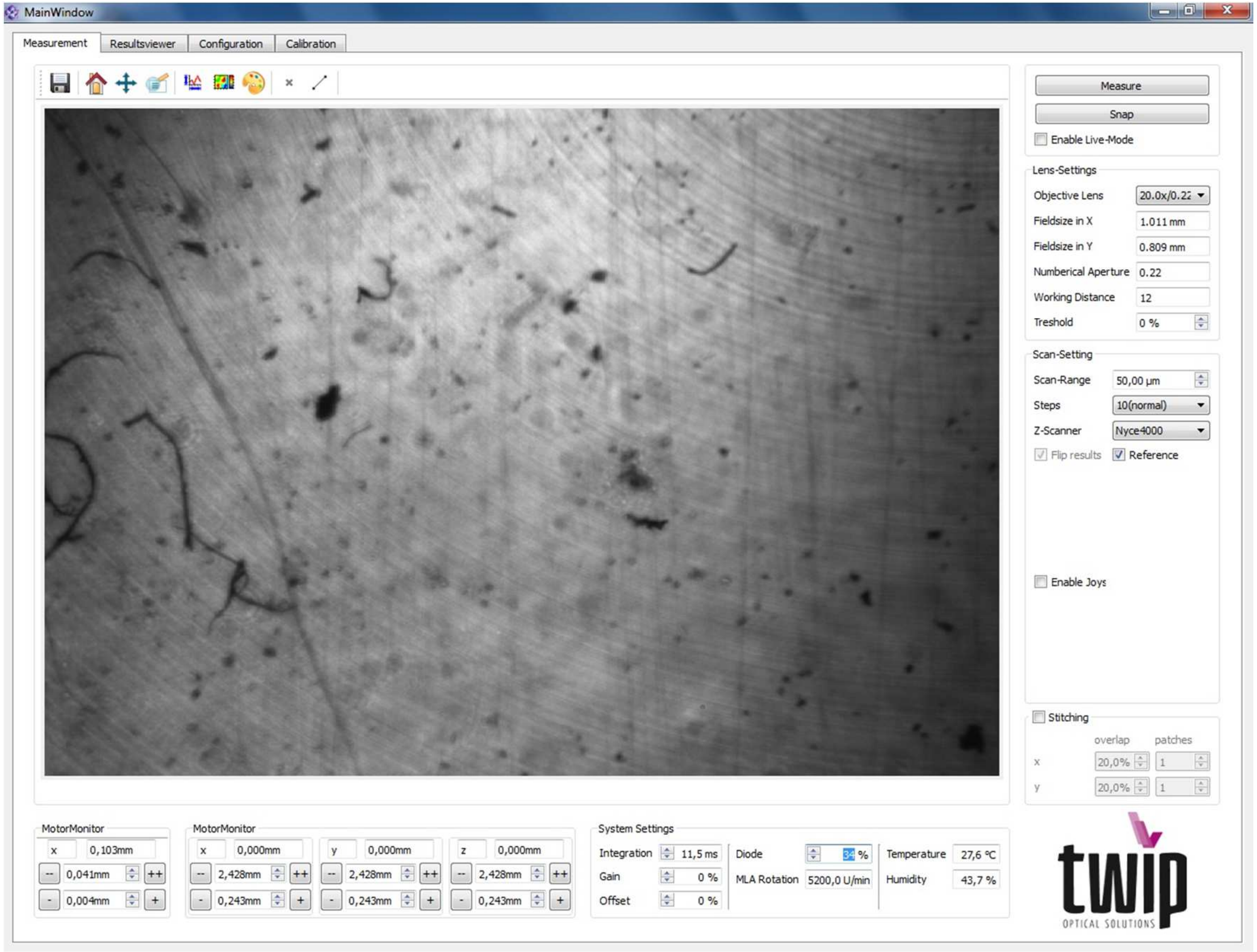






Show-Case III: Confocal Microscopy

GUI



□ Enable Live-Mode

Lens-Settings

Objective Lens	20.0x/0.22
Fieldsize in X	1.011 mm
Fieldsize in Y	0.809 mm
Numberical Aperture	0.22
Working Distance	12
Treshold	0 %

Scan-Setting

Scan-Range	50,00 µm
Steps	10(normal)
Z-Scanner	Nyce4000

☑ Flip results ☑ Reference

□ Enable Joys

□ Stitching

	overlap	patches
x	20,0%	1
y	20,0%	1

MotorMonitor

x	0,103mm
--	0,041mm ++
-	0,004mm +

MotorMonitor

x	0,000mm	y	0,000mm	z	0,000mm
--	2,428mm ++	--	2,428mm ++	--	2,428mm ++
-	0,243mm +	-	0,243mm +	-	0,243mm +

System Settings

Integration	11,5 ms	Diode	34 %	Temperature	27,6 °C
Gain	0 %	MLA Rotation	5200,0 U/min	Humidity	43,7 %
Offset	0 %				