KIT
Karlsruhe Institute of Technology

GPU-based data analysis with the UFO framework

Matthias Vogelgesang

matthias.vogelgesang@kit.edu

PNI-HDRI Spring Meeting 2015

Introduction

ufo framework

- Streamed data processing using heterogeneous compute resources
- Pipelined and parallelized on multiple levels
- Suited for high-volume image processing (e.g. tomography)

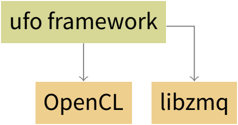
This talk

- Framework does *not* provide any functionality on its own
- Domain-specific tools and applications have to be developed
- This will be a quick tour what is possible and how it is done

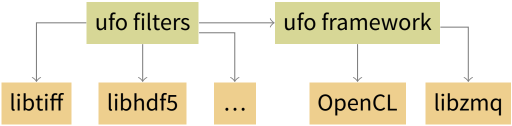
Recap

- Core framework written in C and OpenCL
- Large suite of pre-defined filters for high-throughput image processing
- User specifies workflow, framework takes care of the rest
- Open source (LGPL) and hosted at GitHub `github.com/ufo-kit`

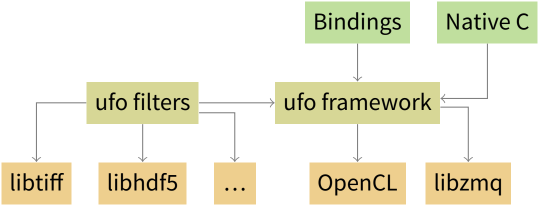
Components



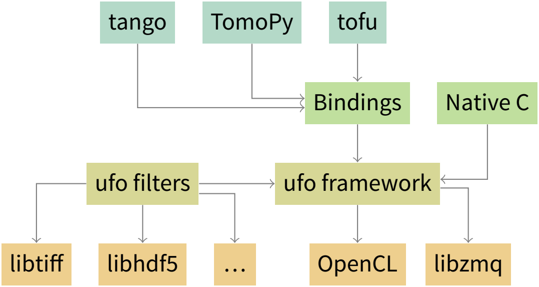
Components



Components

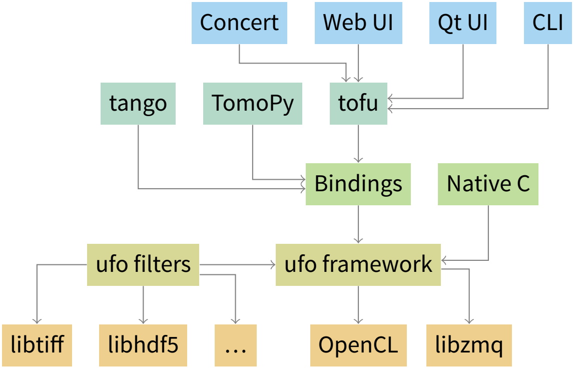

Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
...
OpenCL
libzmq

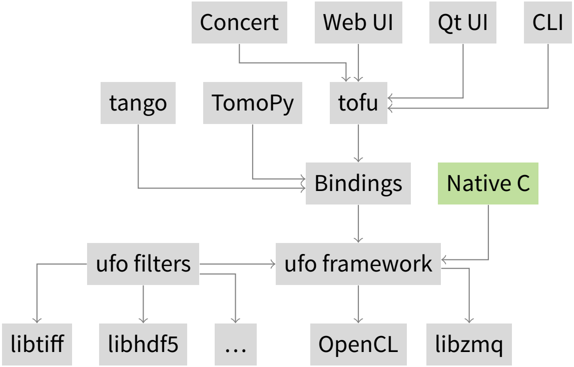
Components



tango
TomoPy
tofu
Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
...
OpenCL
libzmq

Components


Concert
Web UI
Qt UI
CLI
tango
TomoPy
tofu
Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
…
OpenCL
libzmq


Concert

Web UI

Qt UI

CLI

tango

TomoPy

tofu

Bindings

Native C

ufo filters

ufo framework

libtiff

libhdf5

…

OpenCL

libzmq

Builtin tools

Written in native C

- Tools written directly in C
- General purpose: `ufo-launch` and `ufo-runjson`
- (Domain-specific: laminographic reconstructor)

Builtin tools

Written in native C

- Tools written directly in C
- General purpose: `ufo-launch` and `ufo-runjson`
- (Domain-specific: laminographic reconstructor)

Launching linear pipelines

- Used for basic one-off jobs and specified on the command line
- Tasks separated by exclamation marks
- Parameterized with key-value property assignments

Launch examples

Read and write data

```
ufo-launch read path=folder/sino*.tif !
    write filename=multi.tif
```

Launch examples

Read and write data

```
ufo-launch read path=multi.tif !
    write filename=folder/single-%05i.tif
```

Launch examples

Read and write data

```
ufo-launch read path=folder/sino*.tif !
    write filename=output.h5:/raw
```

Launch examples

Downscale input data

```
ufo-launch read path=folder/sino*.tif !
    rescale factor=0.5 !
    write filename=output.h5:/raw
```

Launch examples

Apply OpenCL expressions

```
ufo-launch read path=folder/sino*.tif !
    rescale factor=0.5 !
    calculate expression="log(v)" !
    write filename=output.h5:/raw
```

Launch examples

Remove vertical stripes

```
ufo-launch read path=folder/sino*.tif !
    rescale factor=0.5 !
    calculate expression=”log(v)” !
    fft dimensions=2 ! filter-stripes ! ifft dimensions=2 !
    write filename=output.h5:/raw
```

Launch examples

Compute filtered backprojection

```
ufo-launch read path=folder/sino*.tif !
    rescale factor=0.5 !
    calculate expression=”log(v)” !
    fft dimensions=2 ! filter-stripes ! ifft dimensions=2 !
    fft ! filter ! ifft ! backproject !
    write filename=output.h5:/entry/data/data
```

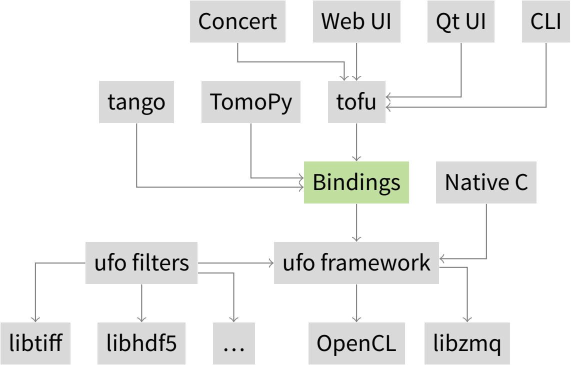
JSON representation

- `ufo-launch` can only execute linear pipelines
- More complex relationships must be expressed programmatically or a data structure
- We use a simple JSON format to serialize the data structure
- The structure can be executed via

```
$ ufo-runjson dataflow.json
```

JSON example

```
{
  "nodes": [
    {"plugin": "read", "name": "read",
     "properties": {"path": "folder/sino*.tif"}},
    {"plugin": "rescale", "name": "rescale",
     "properties": {"factor": 0.5}},
    {"plugin": "write", "name": "write",
     "properties": {"filename": "output.h5:/raw"}}
  ],
  "edges": [
    {"from": "read", "to": "rescale", "input": 0},
    {"from": "rescale", "to": "write", "input": 0}
  ]
}
```


Concert
Web UI
Qt UI
CLI
tango
TomoPy
tofu
Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
…
OpenCL
libzmq

Language bindings

Karlsruhe Institute of Technology

- JSON is a good format to freeze a data flow
- Further customization requires writing C code or bind to a scripting language
- Introspection mechanism allows for third-party language support
- Including JavaScript, Python, Ruby, Lua, Go, Haskell …

Language bindings

- JSON is a good format to freeze a data flow
- Further customization requires writing C code or bind to a scripting language
- Introspection mechanism allows for third-party language support
- Including JavaScript, Python, Ruby, Lua, Go, Haskell …
- However, our primary target for now is Python

♥

Binding interface

```python
# "ufo-runjson" in five lines

import sys
from gi.repository import Ufo

pm = Ufo.PluginManager()
g = Ufo.TaskGraph.read_from_file(pm, sys.argv[1])

sched = Ufo.Scheduler()
sched.run(g)
```

Binding interface

```
from gi.repository import Ufo

pm = Ufo.PluginManager()
read = pm.get_task('read')
rescale = pm.get_task('rescale')
write = pm.get_task('write')

read.set_properties(path='folder/sino*.tif')
rescale.set_properties(factor=0.5)
write.set_properties(filename='output.h5:/raw')

g = Ufo.TaskGraph()
g.connect_nodes(read, rescale)
g.connect_nodes(rescale, write)

sched = Ufo.Scheduler()
sched.run(g)
```

Improved Python integration

Global Interpreter Lock

- GIL would block Python interpreter during computation
- GIL is released during execution and insertion of data

Interfacing with NumPy

- C module converts between ufo and NumPy
- Alternatively data pointers can be re-used

High-level abstractions

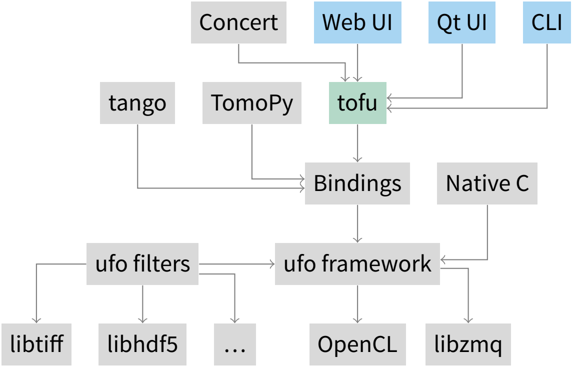
- `ufo` module wraps filters during import
- More magic but cleaner instantation and setup

High-level Python

```python
from ufo import Read, Write, Rescale

read = Read(path='folder/sino*.tif')
rescale = Rescale(factor=0.5)
write = Write(filename='output.h5:/raw')

# wait for execution to finish
write(rescale(read())).run().wait()
```

High-level Python

```python
from ufo import Read, Rescale

read = Read(path='folder/sino*.tif')
rescale = Rescale(factor=0.5)

# use result immediately
for image in rescale(read()):
    print(np.mean(image))
```


Concert

Web UI

Qt UI

CLI

tango

TomoPy

tofu

Bindings

Native C

ufo filters

ufo framework

libtiff

libhdf5

…

OpenCL

libzmq

TOFU reconstruction toolkit

Idea

- Move reconstruction-related code to single Python module
- Simplifies setup and execution of reconstruction pipelines using ufo
- Visualization widgets based on PyQtGraph

Focus

- Tomographic reconstruction with FBP, DFI and SART
- Laminographic reconstruction with FBP
- Manual and automatic axis alignment

Command line interface

- Offline reconstruction for power users
- Parameters are stored in a configuration

```
$ ufo-reconstruct init
$ vi reco.conf
$ ufo-reconstruct tomo
```

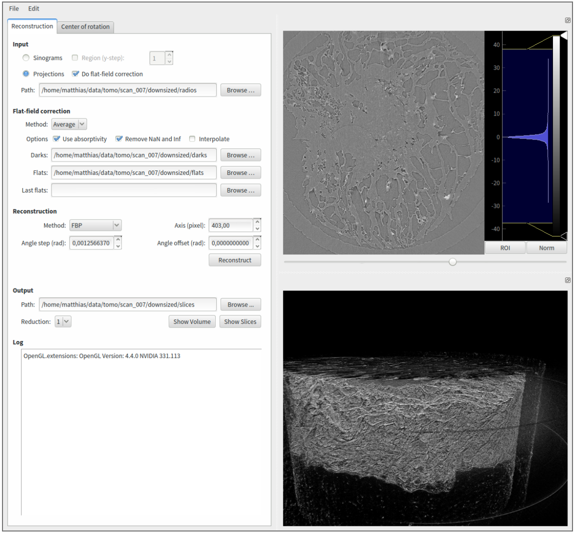
- …which can be overriden with command line arguments

```
$ ufo-reconstruct run --axis=234.5
```

Qt GUI

- Offline reconstruction for regular users
- Shares configuration with command line version
- Uses PyQt and PyQtGraph widgets for visualization

Qt GUI



Input

○ Sinograms ☐ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf ☐ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

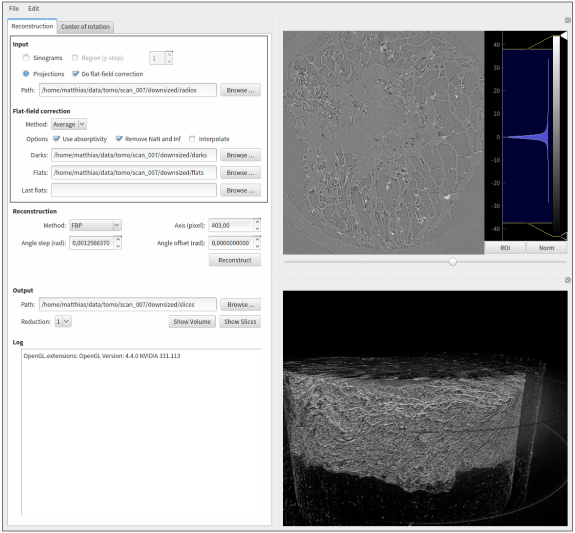
Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Qt GUI

Input



Input

○ Sinograms □ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf □ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

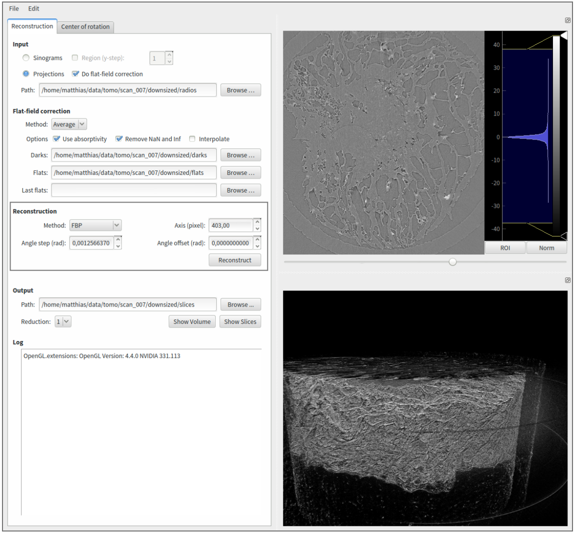
Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Qt GUI

Parameters



Input

○ Sinograms □ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf □ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

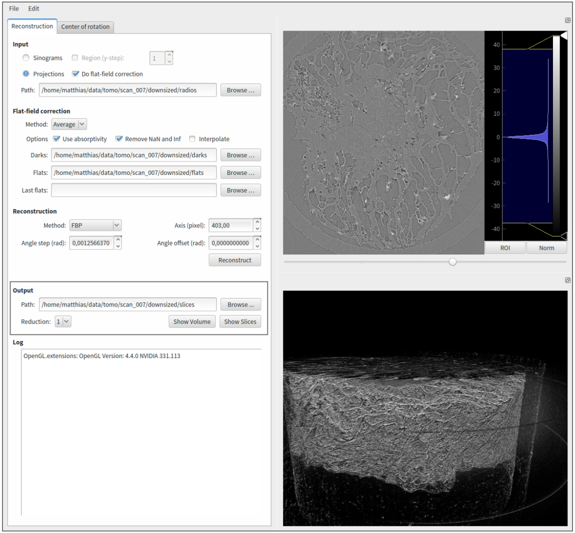
Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Qt GUI

Output



Input

○ Sinograms □ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf □ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

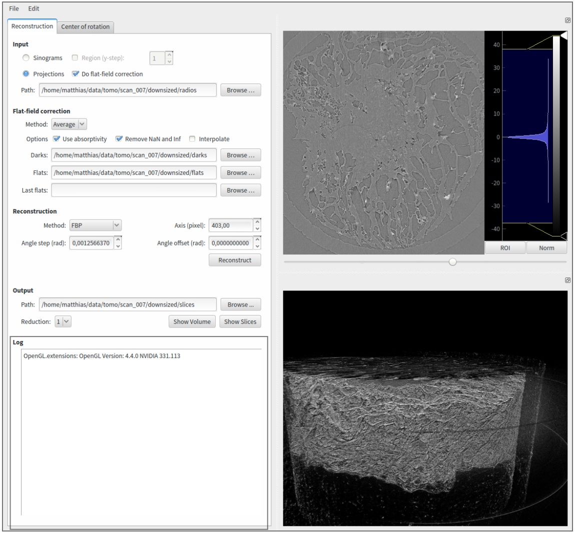
Path: /home/matthias/data/tomo/scan_007/downsized/slices

Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Qt GUI



Input

○ Sinograms □ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf □ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

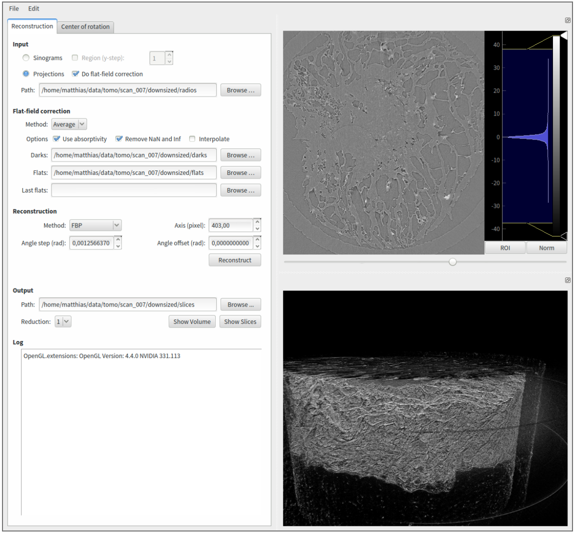
Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Log

Qt GUI



Input

○ Sinograms ☐ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf ☐ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

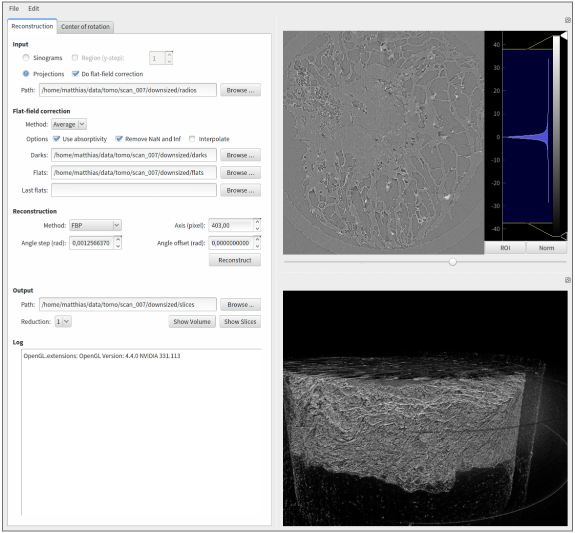
Reduction: 1

Log

```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Slices

Qt GUI



Input

○ Sinograms □ Region (y-step): 1

● Projections ☑ Do flat-field correction

Path: /home/matthias/data/tomo/scan_007/downsized/radios

Flat-field correction

Method: Average

Options ☑ Use absorptivity ☑ Remove NaN and Inf □ Interpolate

Darks: /home/matthias/data/tomo/scan_007/downsized/darks

Flats: /home/matthias/data/tomo/scan_007/downsized/flats

Last flats:

Reconstruction

Method: FBP

Axis (pixel): 403,00

Angle step (rad): 0,0012566370

Angle offset (rad): 0,0000000000

Output

Path: /home/matthias/data/tomo/scan_007/downsized/slices

Reduction: 1

Log

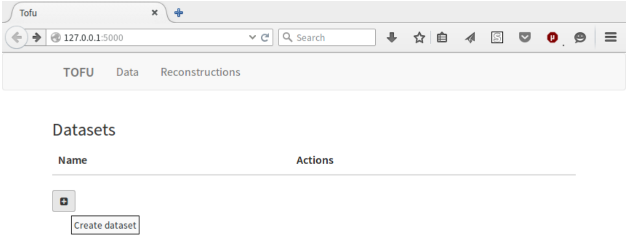
```
OpenGL.extensions: OpenGL Version: 4.4.0 NVIDIA 331.113
```

Volume

Web UI

- Offline reconstruction for regular users
- Simplifies deployment and maintenance
- Uses Flask backend, Bootstrap frontend and WebGL for basic visualization

Web UI prototype

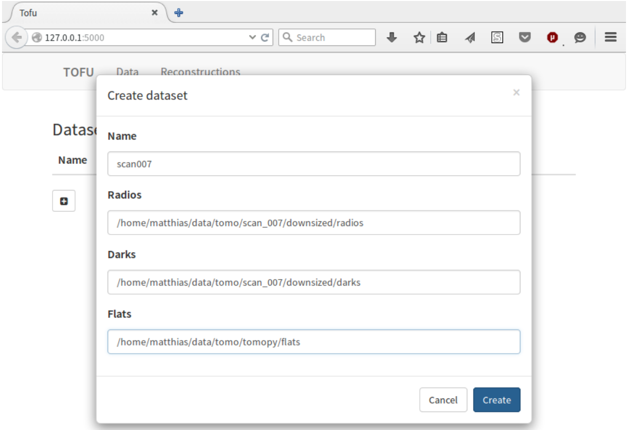


Datasets

Name	Actions

Create dataset from experiment data

Web UI prototype



Create dataset

Name

scan007

Radios

/home/matthias/data/tomo/scan_007/downsized/radios

Darks

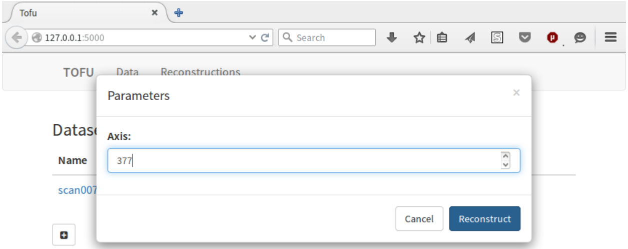
/home/matthias/data/tomo/scan_007/downsized/darks

Flats

/home/matthias/data/tomo/tomopy/flats

…by specifying paths for now.

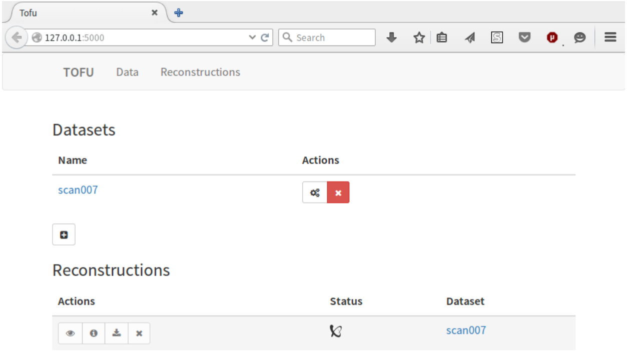
Web UI prototype



Axis: 377

Start a reconstruction

Web UI prototype



Datasets

Name	Actions
scan007	⚙ ✕

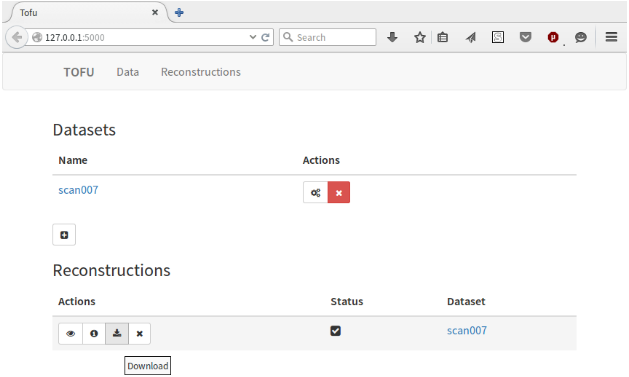
⊞

Reconstructions

Actions	Status	Dataset
👁 ℹ ⤓ ✕		scan007

and wait for reconstruction to finish.

Web UI prototype



Datasets

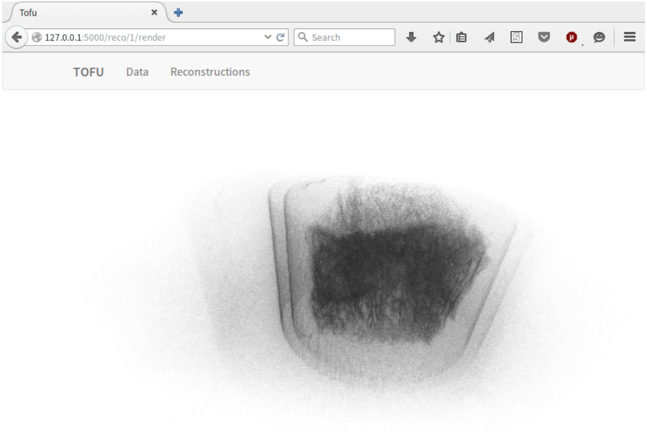
Name	Actions
scan007	

Reconstructions

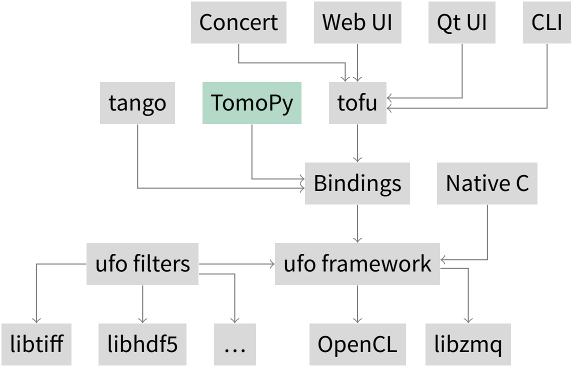
Actions	Status	Dataset
	☑	scan007

Download result

Web UI prototype



…or visualize it.


Concert
Web UI
Qt UI
CLI
tango
TomoPy
tofu
Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
…
OpenCL
libzmq

TomoPy

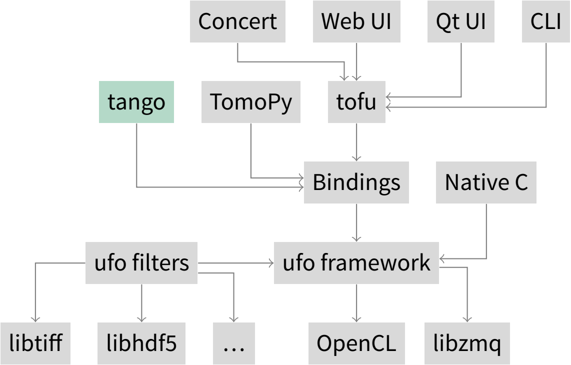
- TomoPy is APS' Python reconstruction toolkit
- The `ufo` module can hook into TomoPy
  - We can re-use existing I/O and pre-processing code
  - TomoPy's reconstruction speed can be improved

TomoPy Standard Demo

```python
import tomopy

data, white, dark, theta = tomopy.xtomo_reader('demo/data.h5')

d = tomopy.xtomo_dataset()
d.dataset(data, white, dark, theta)
d.normalize()
d.correct_drift()
d.phase_retrieval()
d.correct_drift()
d.center = 661.5
d.gridrec()

tomopy.xtomo_writer(d.data_recon, 'tmp/test_', axis=0)
```

TomoPy + UFO

```python
import tomopy
import ufo.tomopy   # new

data, white, dark, theta = tomopy.xtomo_reader('demo/data.h5')

d = tomopy.xtomo_dataset()
d.dataset(data, white, dark, theta)
d.normalize()
d.correct_drift()
d.phase_retrieval()
d.correct_drift()
d.center = 661.5
d.ufo_fbp()         # changed

tomopy.xtomo_writer(d.data_recon, 'tmp/test_', axis=0)
```


Concert
Web UI
Qt UI
CLI
tango
TomoPy
tofu
Bindings
Native C
ufo filters
ufo framework
libtiff
libhdf5
…
OpenCL
libzmq

TANGO integration

- tango provides access to arbitrary “device” servers
- tango has a Python interface …

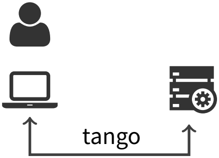
TANGO integration

- tango provides access to arbitrary “device” servers
- tango has a Python interface …
- …use tango as a means for remote computing

Approach

Protocol

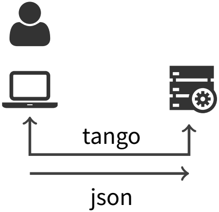
- Server listens for compute requests



Approach

Protocol

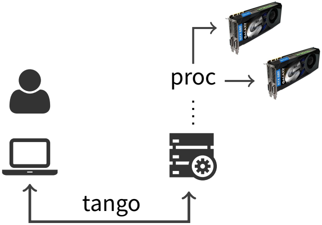
- Server listens for compute requests
- Client sets the `json` attribute and calls the `Run` or `RunContinuous` command



Approach

Protocol

- Server listens for compute requests
- Client sets the `json` attribute and calls the `Run` or `RunContinuous` command
- The server spawns a new compute process identified by a process id



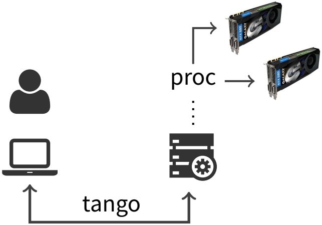
Approach

Protocol

- Server listens for compute requests
- Client sets the `json` attribute and calls the `Run` or `RunContinuous` command
- The server spawns a new compute process identified by a process id

Execution models

1. Single-run processes (“fire and forget”)
2. Continuous processes (update description and re-run)



Single-run processes

Interface

```
process = PyTango.DeviceProxy('hzgctkit/process/1')
process.json = "{ ... }"

pid = process.Run()
print(process.Running(pid))   # still running?
print(process.jobs)           # list of active jobs, e.g. [7041]

process.Wait(pid)
print(process.ExitCode(pid))  # return code of job
```

Single-run processes

Interface

```
process = PyTango.DeviceProxy('hzgctkit/process/1')
process.json = "{ ... }"

pid = process.Run()
print(process.Running(pid))   # still running?
print(process.jobs)           # list of active jobs, e.g. [7041]

process.Wait(pid)
print(process.ExitCode(pid))  # return code of job
```

Remarks

- Simple to use and understand
- No extended use of resources

Continuous processes

Interface

```
pid = process.RunContinuous()
process.Continue(pid)        # trigger execution
process.json = "{ ... }"     # update description
process.Continue(pid)
process.Stop(pid)            # terminate process
```

Continuous processes

Interface

```
pid = process.RunContinuous()
process.Continue(pid)       # trigger execution
process.json = "{ ... }"    # update description
process.Continue(pid)
process.Stop(pid)           # terminate process
```

Remarks

- Enables continuous exploration
- Resources are allocated as long as process is running
- Forgetting to call `Stop` leaks resources
- Real concurrency *still not* solved yet

Outlook

Karlsruhe Institute of Technology

Status

- The ufo framework provides various integration points
- All presented tools are open sourced and free for anyone to use

Plans

- Use tofu for the TomoPy integration
- Finish web GUI and merge with the data portal (see Andreas' talk)
- Integrate with other user frontends (DPDAK?)