TRAINING
GRAPHICAL PROGRAMMING INTERFACE (GPI)

Nick Zwart
2014feb06

Keller Center
For Imaging Innovation

CONTENTS

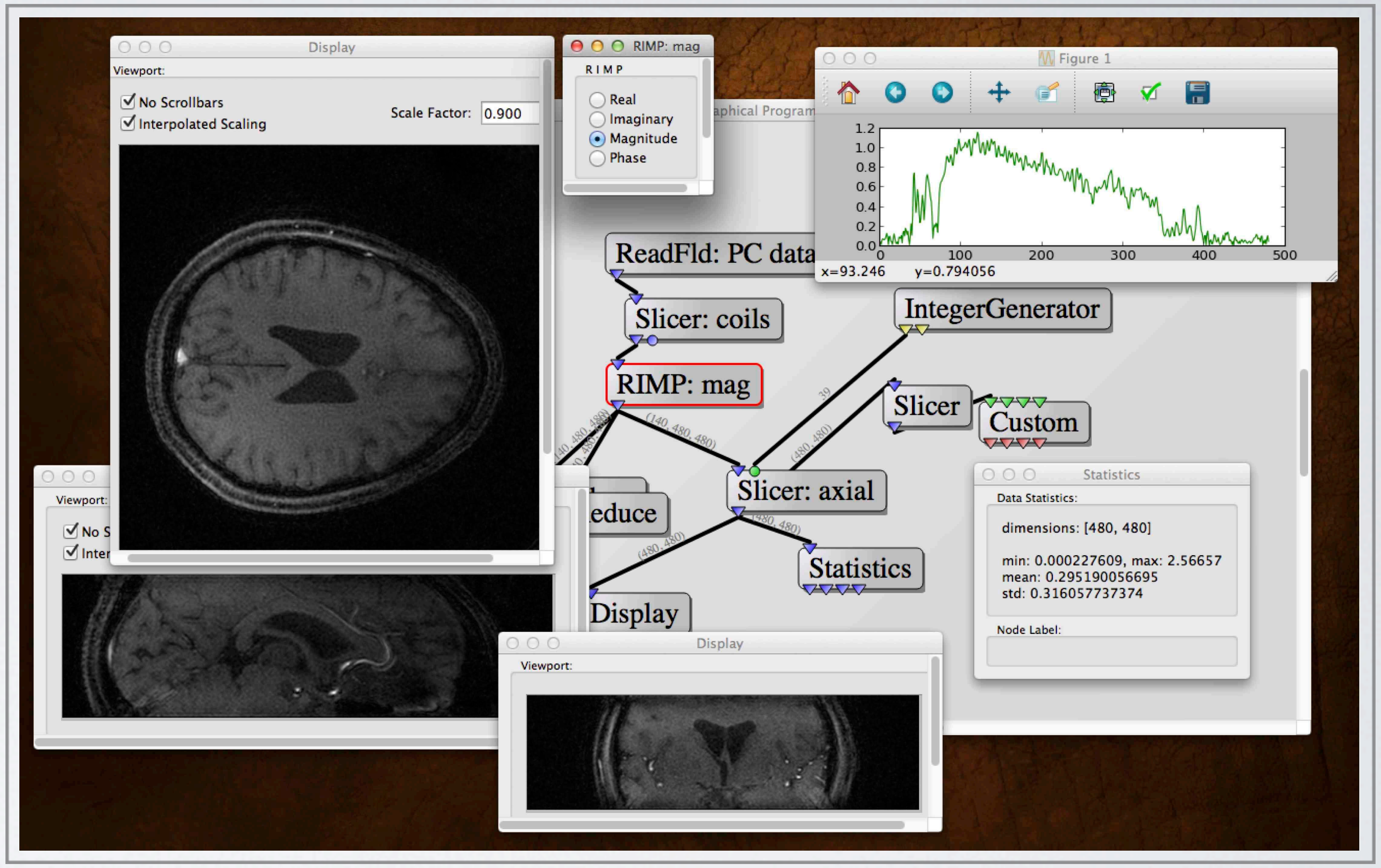
- GPI Introduction
  - KCII User Examples
- LESSON 1: GUI Familiarity
  - Introduce GUI Elements
  - Library/Node Intro (Demo a Few)
  - Play with Example Networks
  - Build Algorithm using Existing Nodes
  - User Data
- LESSON 2: Pure Python Nodes
  - Exercises (Use Existing Nodes as a Guide)
- LESSON 3: C++/R2 PyMods
  - Exercises
- Discussion

CHECKLIST

- Computer (50GB HDD, 8GB RAM, 4-Core Intel 64bit)
  - VMWare Fusion / VMPlayer ≥ 5.0
  - GPI Virtual Machine 0.1beta (via Download or USB Drive)
  - Mouse (trackpad and mapped keys are not setup)
  - NOTE: Upgrade Ubuntu at your own risk. GPI was only tested with the current revision of the installed libraries.
- Packet
  - Node Developer's Guide
  - Quick Start Guide
  - Training Slides

INTRODUCTION

- Modular
- Unit Testing
- Test Driven Development
- Generate Code On-The-Fly
- Abstractable (Sharing, R2)
- Multi-platform
- Reconstruction, Simulations, Pulse Sequence Development
- Data & Algorithm Analysis



Display

Viewport:

☑ No Scrollbars
☑ Interpolated Scaling

Scale Factor: 0.900

RIMP: mag

RIMP

○ Real
○ Imaginary
● Magnitude
○ Phase

Statistics

Data Statistics:

```
dimensions: [480, 480]

min: 0.000227609, max: 2.56657
mean: 0.295190056695
std: 0.316057737374
```

Node Label:

KCII LAB EXAMPLES

- Nick Zwart - Spiral CG Deblur (iterative)
- Dinghui Wang - Spiral Fat-Water (iterative)
- Zhiqiang Li - Spiral TSE
- Ryan Robison - System Characterization
- Yuchou Chang - PROPELLER
- Mike Schar - PROPELLER
- Jim Pipe - Spin Simulation

SPIRAL RECONSTRUCTION EXAMPLE

- k-Space Data
    - 2D & Plot Views
- Trajectory Coordinates
    - 3D-GL (Single Interleave)
    - Gridded Sample Density (MTF)
        - Cross-section
- Single Slice Recon (Accumulated)

MORE DETAIL

- README.pdf
  - docs/

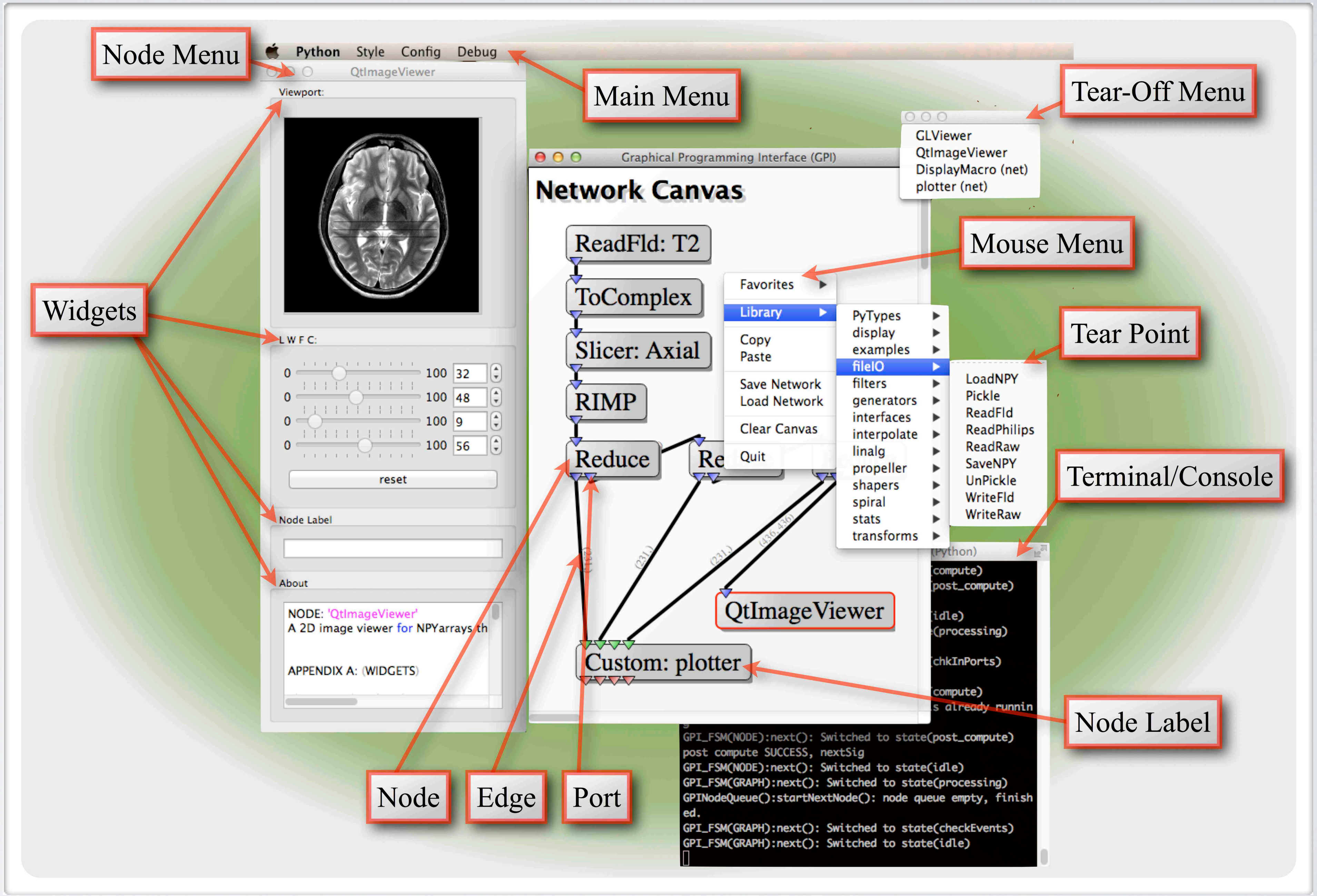
CONTENTS

- ~~GPI Introduction~~
  - ~~KCII User Examples~~
- LESSON 1: GUI Familiarity
  - Introduce GUI Elements
  - Library/Node Intro (Demo a Few)
  - Play with Example Networks
  - Build Algorithm using Existing Nodes
  - User Data
- LESSON 2: Pure Python Nodes
  - Exercises (Use Existing Nodes as a Guide)
- LESSON 3: C++/R2 PyMods
  - Exercises
- Discussion

GUI ELEMENTS

- QuickStart.pdf
  - docs/

Docs



NODE: 'QtImageViewer'
A 2D image viewer for NPYarrays th

APPENDIX A: (WIDGETS)

```
(compute)
(post_compute)

(idle)
(processing)

(chkInPorts)

(compute)
s already runnin
GPI_FSM(NODE):next(): Switched to state(post_compute)
post compute SUCCESS, nextSig
GPI_FSM(NODE):next(): Switched to state(idle)
GPI_FSM(GRAPH):next(): Switched to state(processing)
GPINodeQueue():startNextNode(): node queue empty, finish
ed.
GPI_FSM(GRAPH):next(): Switched to state(checkEvents)
GPI_FSM(GRAPH):next(): Switched to state(idle)
```

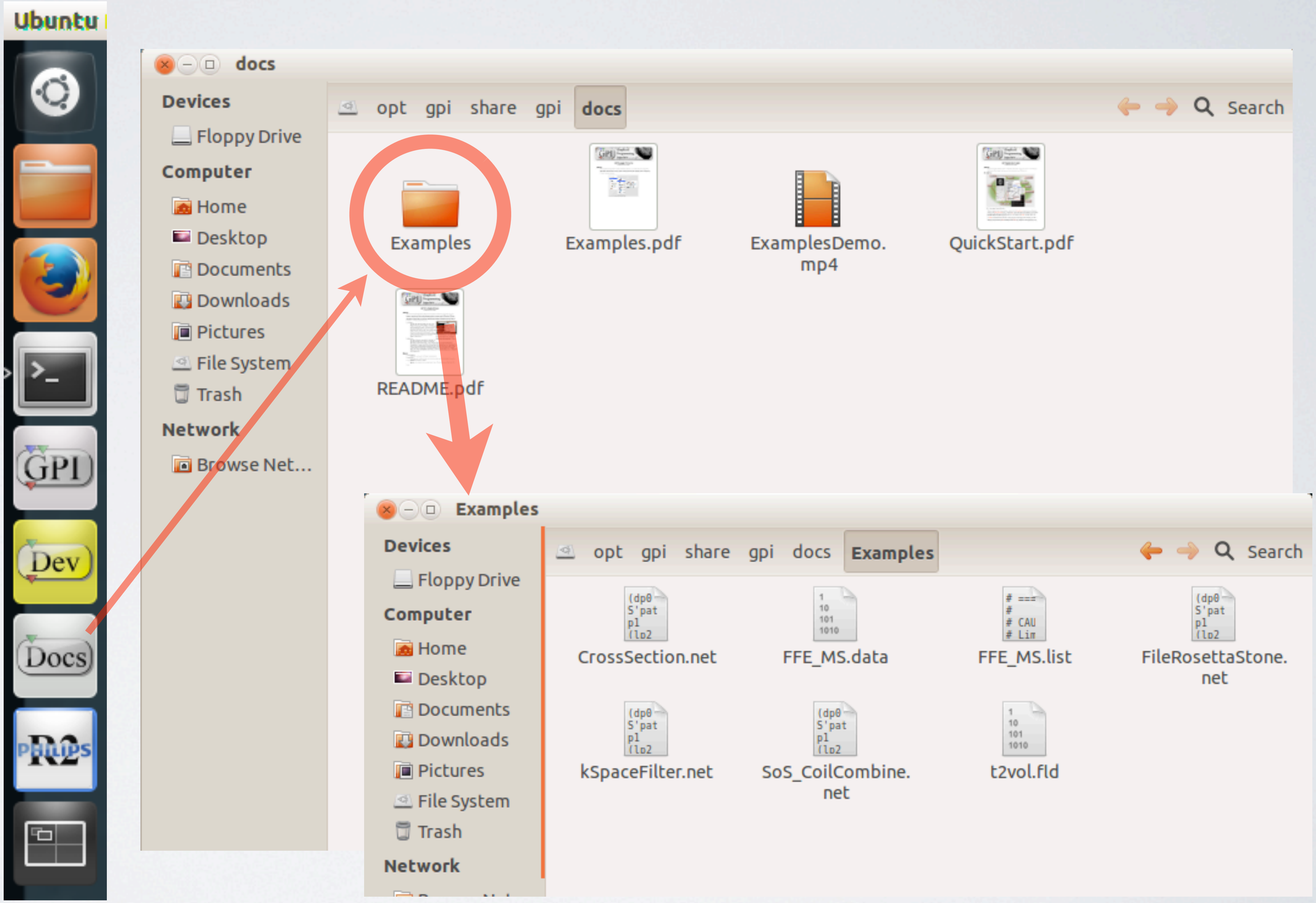
ELEMENTS TO COVER

- Network Canvas
  - Pan/Zoom
  - Multiples
- Nodes
  - Widgets
    - Label
    - About
  - Ports
  - Edges
  - Hover
- Mouse Menu
  - Tear Points
  - Library
- Main Menu
  - Log Level
- Terminal/Console

START EXERCISES

EXAMPLE NETWORKS

- Examples.pdf
  - docs/



CROSS SECTION

- Reader - OS info, dimensions
- Vector Vs. Complex
- Reduce - Mask: Slice & Complement
- ImageDisplay - Zoom, Interpolate
- Plotter - Port Independence
- IntegerLoop w/ Slicer
    - Widget Ports

Task:

Add vertical cross sections.

FILE ROSETTA STONE

- File Types
  - .data/.list, .raw, .npy, .fld, .pickle
  - Drag'n Drop
- Numpy is the Medium

Task:

Read in an array and save it to a different file format.

K-SPACE FILTER

- Shapes
- Elem_Math
- Combine
- ImageDisplay - RIMP Features

Task:

Make a low-pass filter.

GL MACRO NODE

- Macros
  - Widget Layout
  - Node Labels
- Dimensions
- Custom - Combine Code
- GL Widget / GL Objects

Task:

Rotate, translate, and zoom rendered objects, scene and lighting.

Add a GL Object. Hint: look at the 'GLObjects' node's input and output to see the data type they require. Then explore the node-menu of GL-objects.

Change the trajectory by modifying the MacroNode widgets. Practice pulling the widgets out of the menu and putting new ones in.

SOS COIL COMBINE

- Macros
  - Widget Layout
- Dimensions
- Collapse - RMS

Task:

Select different slices. Measure object diameter in pixels.

SPIN SIMULATOR

- Spyn - Spin Generator
- Bloch - Simulator
- 1D & 3D Visualization
    - 3D Movies
        - Save ARGB Frames as ‘.raw’
        - ImageJ - Stack > Movie
        - Examples/SpinSim.m4v

Task:

Apply a symmetric RF pulse and observe the result.

SPIRAL RECONSTRUCTION

- 2D or 3D Spiral (2D Example)
- .data/.list/.txt
- Non Cartesian Recon
  - Coordinates
  - Density Correct (SDC)
  - Gridding & Rolloff

Task:

Scroll through reconstructed stack of images.

ALGORITHM EXERCISE

COIL COMBINE

$$\hat{C}_n = C_n \otimes K$$

- Estimate B_1^- for each coil.

$$B_n = \frac{\hat{C}_n}{\sqrt{\sum^n \hat{C}_n^2}}$$

- Remove B_1^- sensitivity profile

  from coil images.

$$\hat{I}_n = C_n B_n^*$$

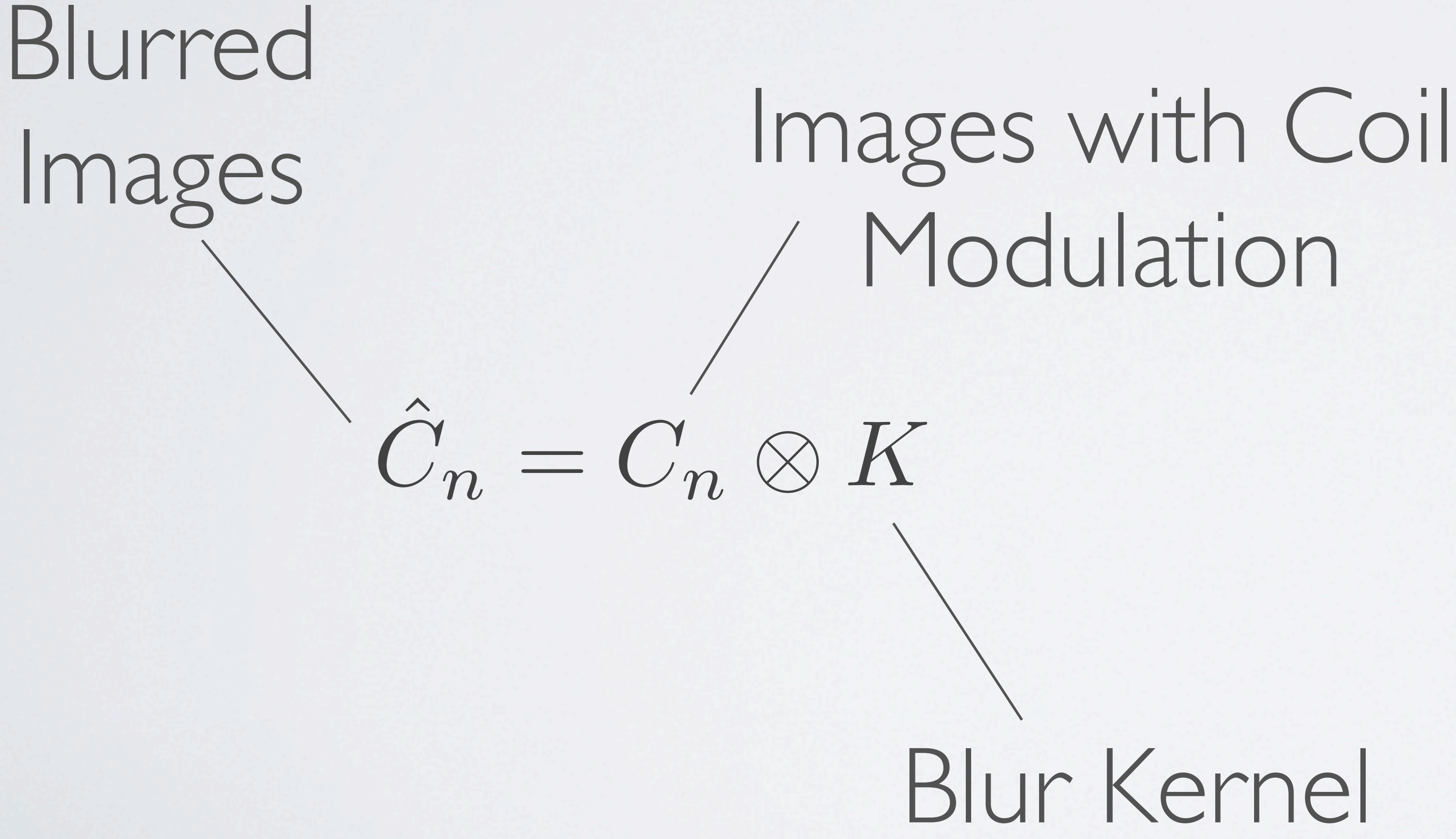
- Phase preserving coil combine.

$$I = \sum^n \hat{I}_n$$

ALGORITHM EXERCISE

COIL COMBINE

Step 1: Blur Coil Images



Blurred Images

Images with Coil Modulation

$$\hat{C}_n = C_n \otimes K$$

Blur Kernel

Nodes:

LoadNPY (Examples/MultiCoil.npy)
Elem_Math
Shapes
FFT_NUMPY

ALGORITHM EXERCISE

COIL COMBINE

Step 2: Remove Image Modulus.

Complex Coil Sensitivity

Blurred Images

$$B_n = \frac{\hat{C}_n}{\sqrt{\sum^n \hat{C}_n^2}}$$

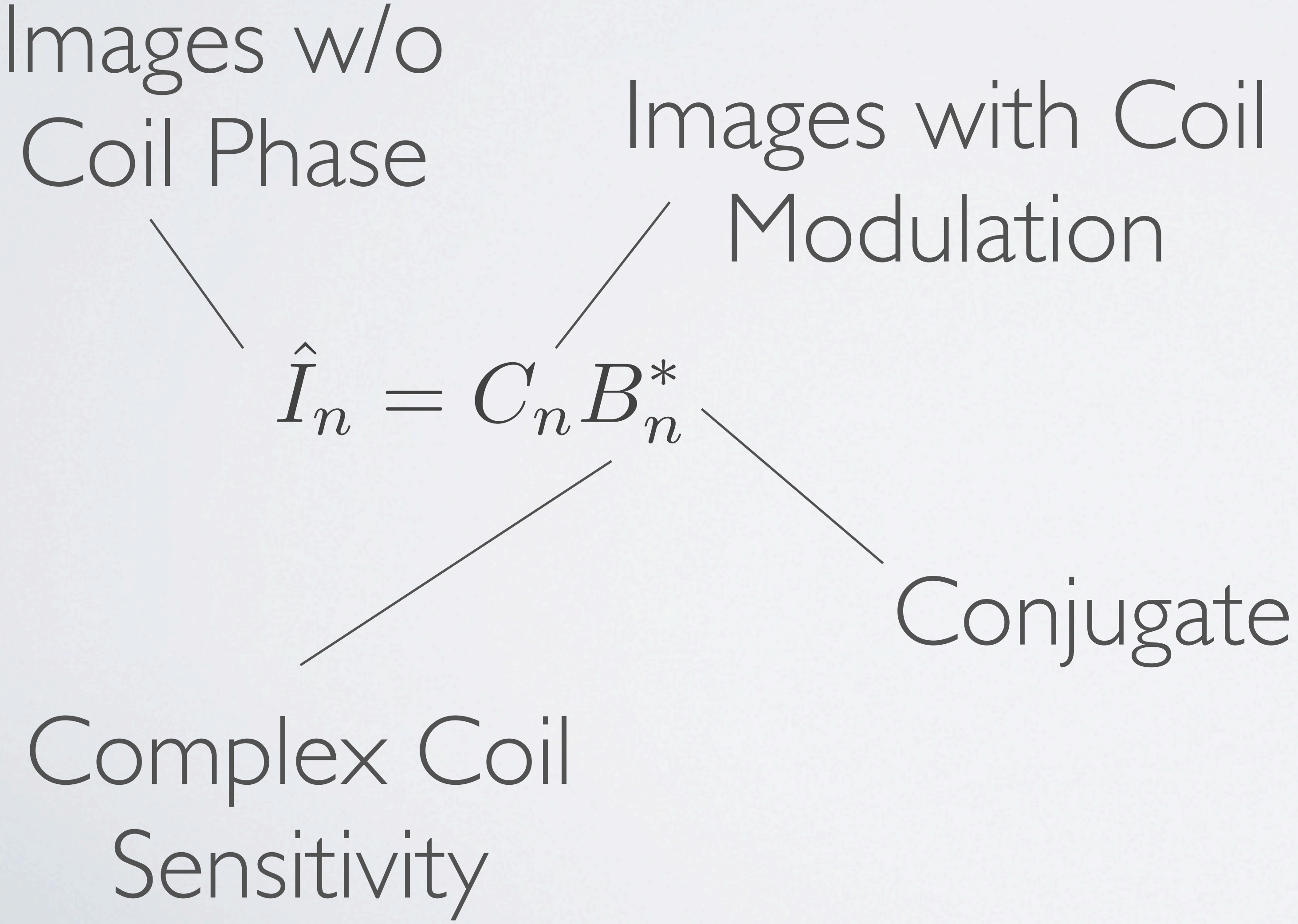
Nodes:

RIMP
Collapse
Dimensions
ValueBounds
Elem_Math

ALGORITHM EXERCISE

COIL COMBINE

Step 3: Remove B_1^- phase from coil images.



Images w/o Coil Phase

Images with Coil Modulation

$$\hat{I}_n = C_n B_n^*$$

Conjugate

Complex Coil Sensitivity

Nodes:

Elem_Math

ALGORITHM EXERCISE

COIL COMBINE

Step 4: Phase preserving coil combine.

Final Image

Images w/o Coil Phase

$$I = \sum^{n} \hat{I}_n$$

Nodes:

Collapse

TRY READING **YOUR** DATA

- Use ReadPhilips Node
  - data/list
- Your Data or Our Supplemental Data

CONTENTS

- ~~GPI Introduction~~
  - ~~KCII User Examples~~
- ~~LESSON 1: GUI Familiarity~~
  - ~~Introduce GUI Elements~~
  - ~~Library/Node Intro (Demo a Few)~~
  - ~~Play with Example Networks~~
  - ~~Build Algorithm using Existing Nodes~~
  - ~~User Data~~
- LESSON 2: Pure Python Nodes
  - Exercises (Use Existing Nodes as a Guide)
- LESSON 3: C++/R2 PyMods
  - Exercises
- Discussion

NODE DEVELOPER CHALLENGE

Pick a node description from the list, claim your node by signing up at the front, write it before the end of class and it will be included in the GPI core with
your name as author.

Or follow along with your own node idea.

DIRECTORY STRUCTURE

Format:

```
<library>/GPI/<mynode>_GPI.py
```

Library Name
in Mouse Menu

Node Name
on Canvas

Required Suffix

Examples:

```
mathematics/GPI/RIMP_GPI.py
```

Dev

```
~/src/ex1/GPI/e1_widgets_GPI.py
```

DIRECTORY STRUCTURE

- Nodes can’t have same names (regardless of library)
- Libraries can’t have same names
  - import <library>.<name>
  - <library>/<name>.so

ENVIRONMENT VARIABLES

- ~/.bashrc
  - GPI_PATH (path to the gpi start script)
  - GPI_LIBRARY_PATH (colon delimited list of library paths)
  - PATH += $GPI_PATH (for the make.py or mk commands)
  - EDITOR (full path to desired code editor)

START EXERCISES

Dev → ex1/GPI/e1_widgets_GPI.py

ex1/GPI/e1_widgets_GPI.py

- Drill Down the Library Menu **or** Search
- Ctrl - Right Mouse Click (Open the Code)
- NodeDevGuide.pdf (Widget Attributes)
- Terminal Window
    - `mk e1_widgets_GPI.py`
- ‘about’ widget

ex1/GPI/e2_ports_GPI.py

- Navigate to the Code (Use Drag & Drop)
- Port Specifications
  - NodeDevGuide.pdf (Type Attributes)
  - `import numpy`
- Shapes & Statistics Nodes
  - Check InPort Enforcement
  - Observe OutPort Errors (for Wrong Shape)
  - Check Shape with Statistics, Edge, Port-Hover

ex1/GPI/e3_validate_GPI.py

- validate()
- Return Codes
  - Use Shapes as Input
  - Practice Resolving Warn & Error States
- Update Logger Message Based on Code Reqs

ex1/GPI/e4_logger_GPI.py

- Main Menu > Debug > Log Level
  - Test Against Chosen Level in Node
  - Terminal Window Output
    - Time, Line #, Code, etc...
  - Messages Demonstrate Appropriate Info

ex1/GPI/e5_events_GPI.py

- Observe Event Types
  - Widget, Port, Init
- NOTE: Latest Event is Kept
  - All Events will be Kept in Future Release

ex1/GPI/e6_import_GPI.py

- Pure-PyMODs
  - import Dev Modules in compute()
  - Reinforce Library Directory Structure
  - Identify Kernel Compute Code
    - Relocate to Module

YOUR NODE PROJECT

Take the rest of the time to work on a node from the challenge list or your own algorithm.

CONTENTS

- ~~GPI Introduction~~
  - ~~KCII User Examples~~
- ~~LESSON 1: GUI Familiarity~~
  - ~~Introduce GUI Elements~~
  - ~~Library/Node Intro (Demo a Few)~~
  - ~~Play with Example Networks~~
  - ~~Build Algorithm using Existing Nodes~~
  - ~~User Data~~
- ~~LESSON 2: Pure Python Nodes~~
  - ~~Exercises (Use Existing Nodes as a Guide)~~
- LESSON 3: C++/R2 PyMods
  - Exercises
- Discussion

DIRECTORY STRUCTURE

Node Format:

```
<library>/GPI/<mynode>_GPI.py
```

Library Name in Mouse Menu

Node Name on Canvas

Required Suffix

PyMOD Format:

```
<library>/<mymod>_PyMOD.cpp
```

Library Name in Mouse Menu

import Name

Required Suffix

```
import library.mymod
```

DIRECTORY STRUCTURE

Python Package:

```
<library>/__init__.py
```

```
import library.mymod
```

PYTHON FUNCTION INTERFACE (PYFI)

- Translates Numpy (.py) to R2 (.cpp)
  - Wraps Original Data Segment (No Copying)
- Types
  - double, long (int64_t)
  - R2 Array Types
    - float, double, int32_t, int64_t, complex<float>, complex<double>

START EXERCISES

Dev → ex2/GPI/e1_hello_GPI.py

ex2/GPI/e1_hello_GPI.py

- Terminal Window
  - `mk e2_module_PyMOD.cpp (or e2_module)`
  - `ls -l`
  - <module>.so Binary File
- Module Re-Uptake
- Look at PyFI Code

ex2/GPI/e2_PosArgs_GPI.py

- Positional Arguments
  - Pointers
  - Code: Top-Bottom
  - Py-Function: Left-Right
- Multiple Outputs are Returned in a Tuple

ex2/GPI/e3_arrays_GPI.py

- R2 Arrays
    - Declaration & Initialization
    - Operations
    - Destructor
    - coutv()
    - Copy Mode Output

ex2/GPI/e4_ArrayBounds_GPI.py

- R2 Arrays
    - Declaration & Initialization
    - Bounds
    - Pre-Allocate (PYFI_SETOUTPUT_ALLOC())
    - Observe Output Order

ex2/GPI/e5_KeywordArgs_GPI.py

- Default Arguments
  - Array Defaults
- Positional First, then Keyword Args

ex2/GPI/e6_errors_GPI.py

- Python Exceptions <> PyFI Errors
    - PYFI_ERROR()
- Use Python Docs for Exception Handling

ex2/GPI/e7_kernels_GPI.py

- Kernel Code
  - `<algorithm-name>_kernel.cpp`
- Portability Guidelines
- Task:
  - Pass dimensions from python (using a numpy array).
  - Generate a new R2 Array inside the kernel function.
  - Pass the newly generated array back to python.

ex2/GPI/e8_threads_GPI.py

- Observe Threaded `stdout` in the Terminal
- Test w/o Threading Interface (e.g. R2 Portability)

YOUR NODE PROJECT

Take the rest of the time to work on a node from the challenge list or your own algorithm.

CONTENTS

- ~~GPI Introduction~~
  - ~~KCII User Examples~~
- ~~LESSON 1: GUI Familiarity~~
  - ~~Introduce GUI Elements~~
  - ~~Library/Node Intro (Demo a Few)~~
  - ~~Play with Example Networks~~
  - ~~Build Algorithm using Existing Nodes~~
  - ~~User Data~~
- ~~LESSON 2: Pure Python Nodes~~
  - ~~Exercises (Use Existing Nodes as a Guide)~~
- ~~LESSON 3: C++/R2 PyMods~~
  - ~~Exercises~~
- Discussion

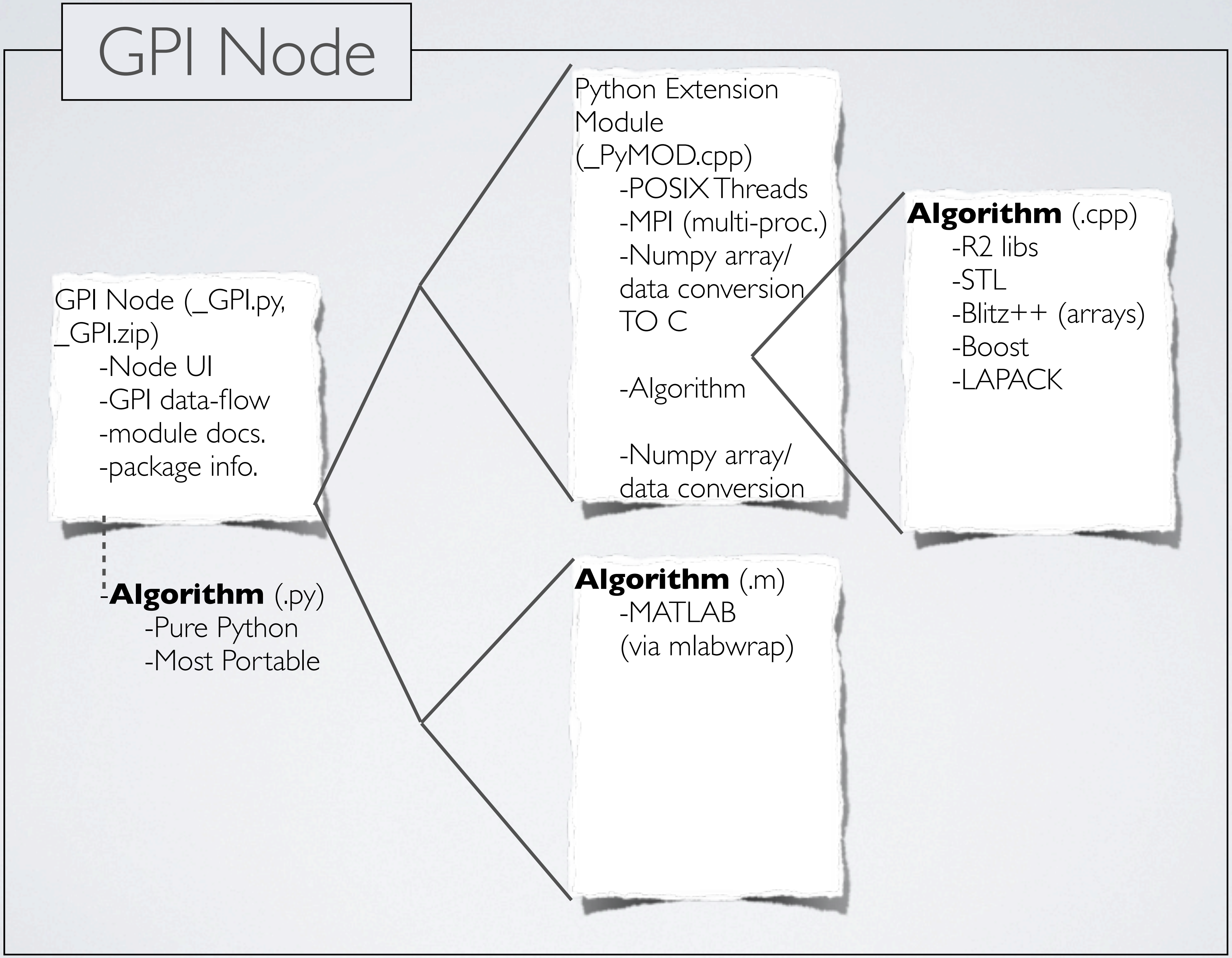
DISCUSSION

- Questions
- Future Features
- Communication & Feedback
  - Site Contact

APPENDIX

- Extra notes about GPI

CODE STRUCTURE



GPI Node

GPI Node (_GPI.py, _GPI.zip)
- -Node UI
- -GPI data-flow
- -module docs.
- -package info.

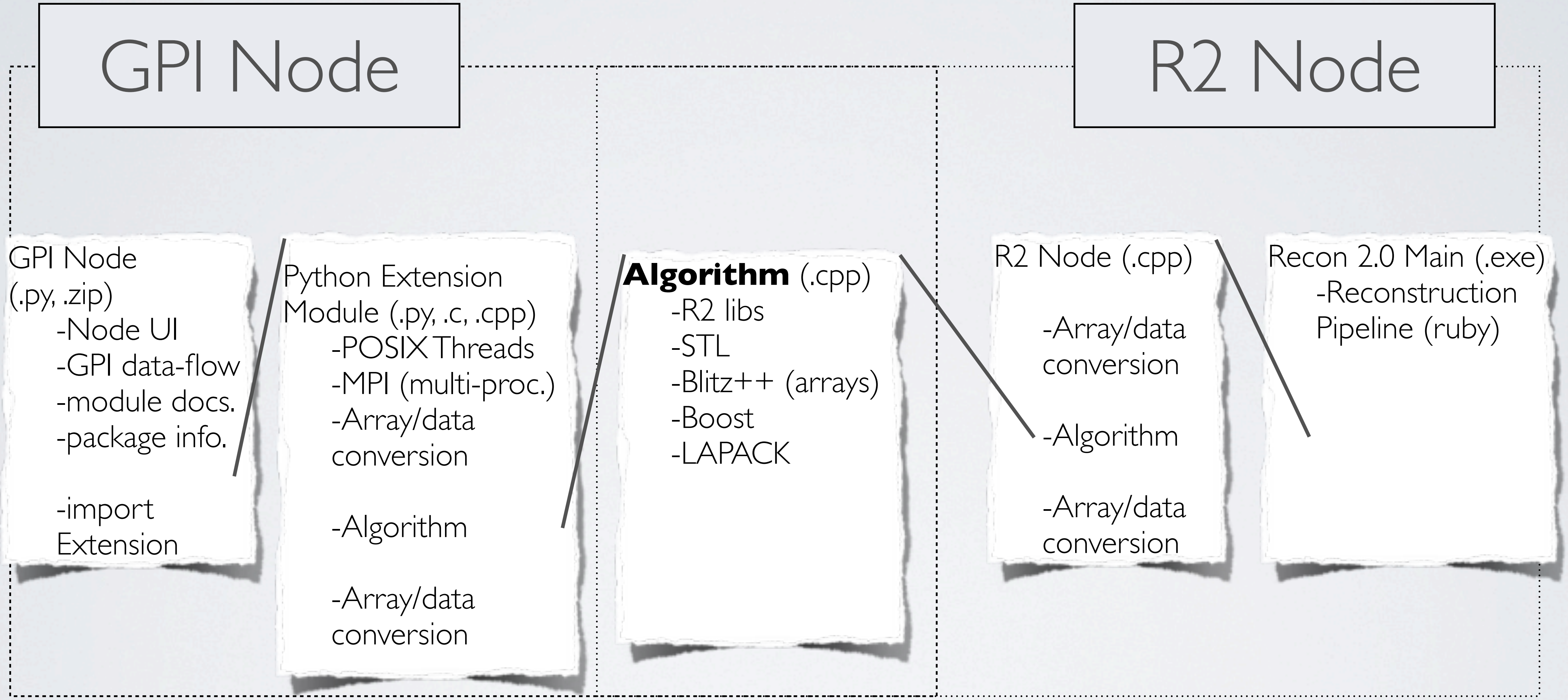
Algorithm (.py)
- -Pure Python
- -Most Portable

Python Extension Module (_PyMOD.cpp)
- -POSIX Threads
- -MPI (multi-proc.)
- -Numpy array/ data conversion TO C
- -Algorithm
- -Numpy array/ data conversion

Algorithm (.cpp)
- -R2 libs
- -STL
- -Blitz++ (arrays)
- -Boost
- -LAPACK

Algorithm (.m)
- -MATLAB (via mlabwrap)

CODE STRUCTURE



GPI Node

R2 Node

GPI Node (.py, .zip)
- -Node UI
- -GPI data-flow
- -module docs.
- -package info.
- -import Extension

Python Extension Module (.py, .c, .cpp)
- -POSIX Threads
- -MPI (multi-proc.)
- -Array/data conversion
- -Algorithm
- -Array/data conversion

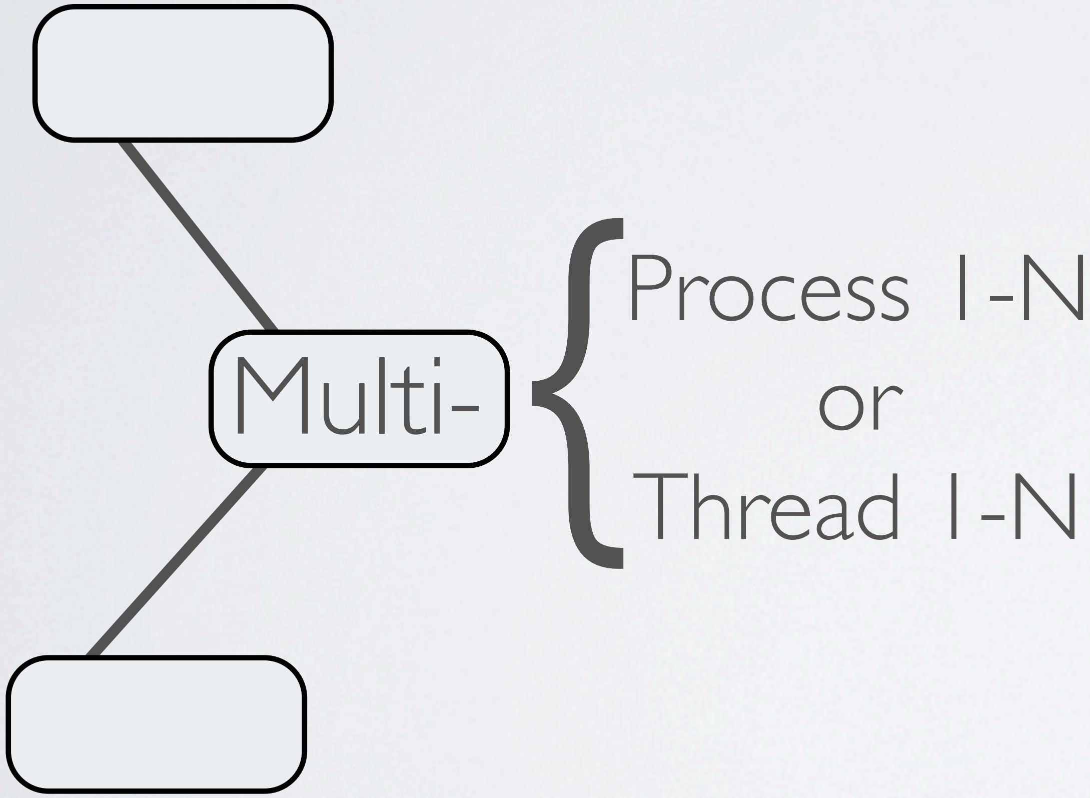
Algorithm (.cpp)
- -R2 libs
- -STL
- -Blitz++ (arrays)
- -Boost
- -LAPACK

R2 Node (.cpp)
- -Array/data conversion
- -Algorithm
- -Array/data conversion

Recon 2.0 Main (.exe)
- -Reconstruction Pipeline (ruby)

GPI VS. R2 TOPOLOGY

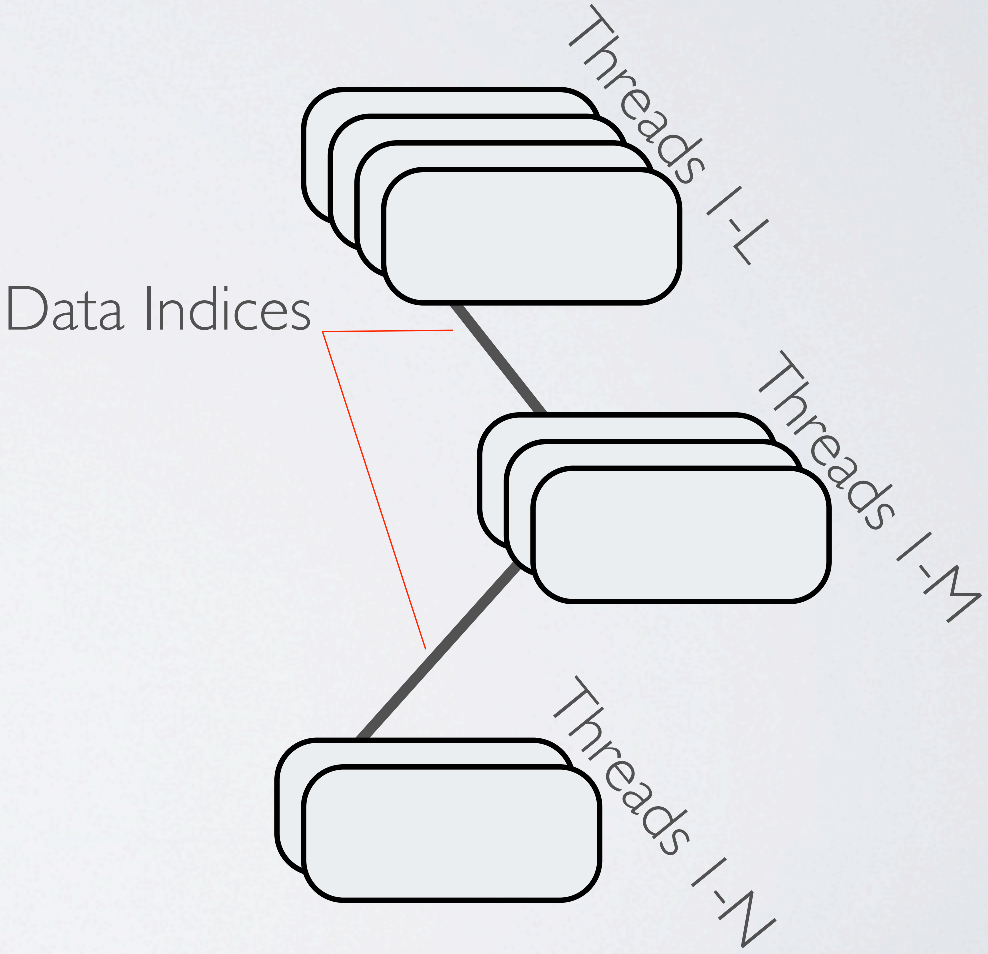
GPI Multi-Processing
POSIX, TBB, MPI...



Multi-

Process 1-N
or
Thread 1-N

R2 Multi-Processing
TBB



Threads 1-L

Data Indices

Threads 1-M

Threads 1-N