

GPI Quick Start Guide

Nicholas R. Zwart, Ph.D.

Summary:

This quick start guide is meant get the user exploring GPI networks as quickly as possible as well as providing introductory material such as basic GPI nomenclature and provides a brief overview of the user interface, keyboard and mouse buttons necessary to construct and execute GPI networks.

The figure below identifies most of the essential tools the user will need to construct a network, prototype algorithms and develop nodes.

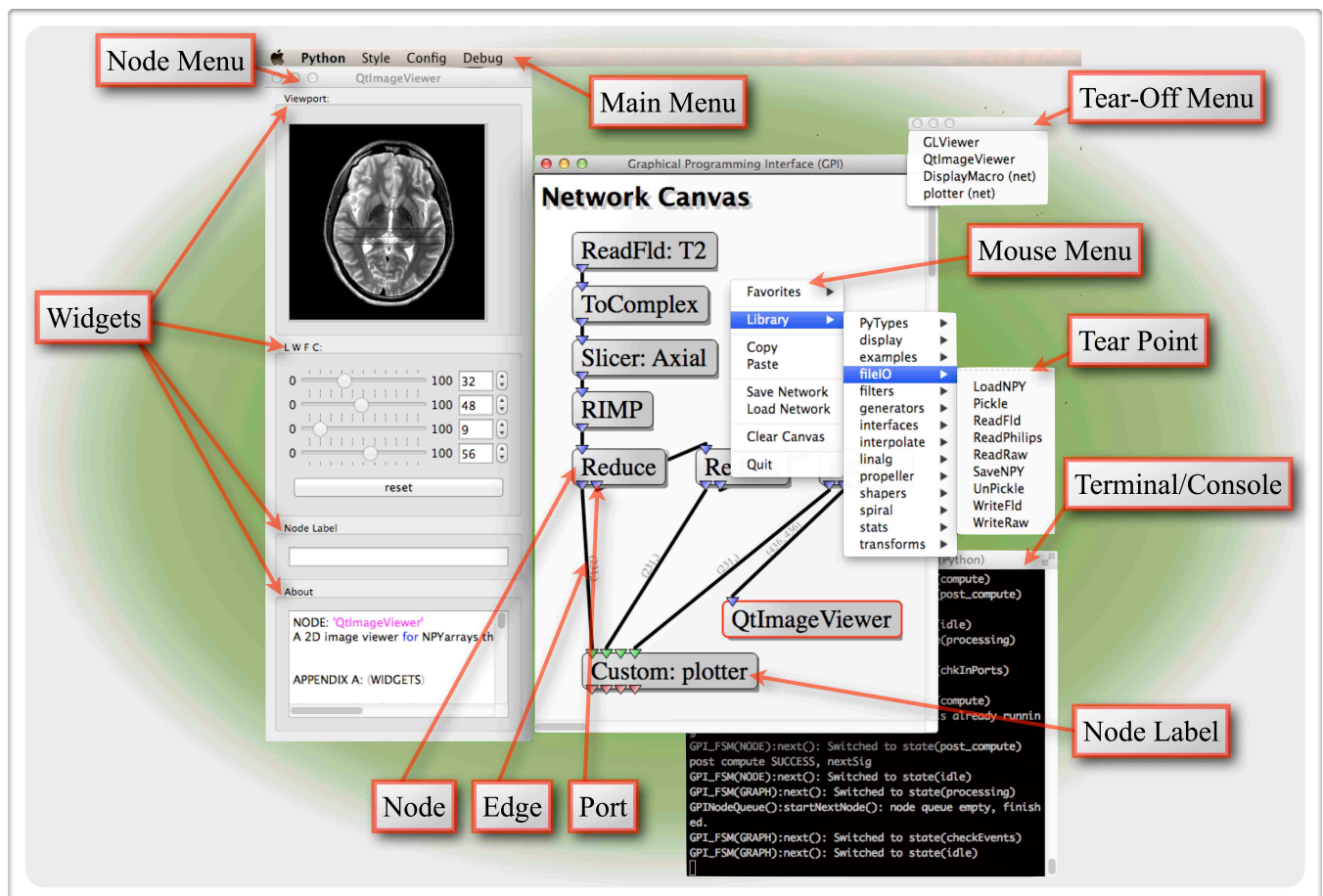


Fig. 1: A screen-capture of an active GPI session with superimposed labels identifying each UI component.

Opening an Example Network (MR Reconstruction):

GPI is packaged with a few example networks ranging from image post processing, MR spin simulation, 3D-GL visualization, etc... The following example focuses on a 2D spiral MR reconstruction which contains the basic building blocks for all non-cartesian reconstruction (i.e. gridding, sample density correction, fft, and rolloff). The network and data are packaged in the `/opt/gpi/doc` directory. The easiest way to navigate to doc is to open GPI as shown in Fig. 3 below.

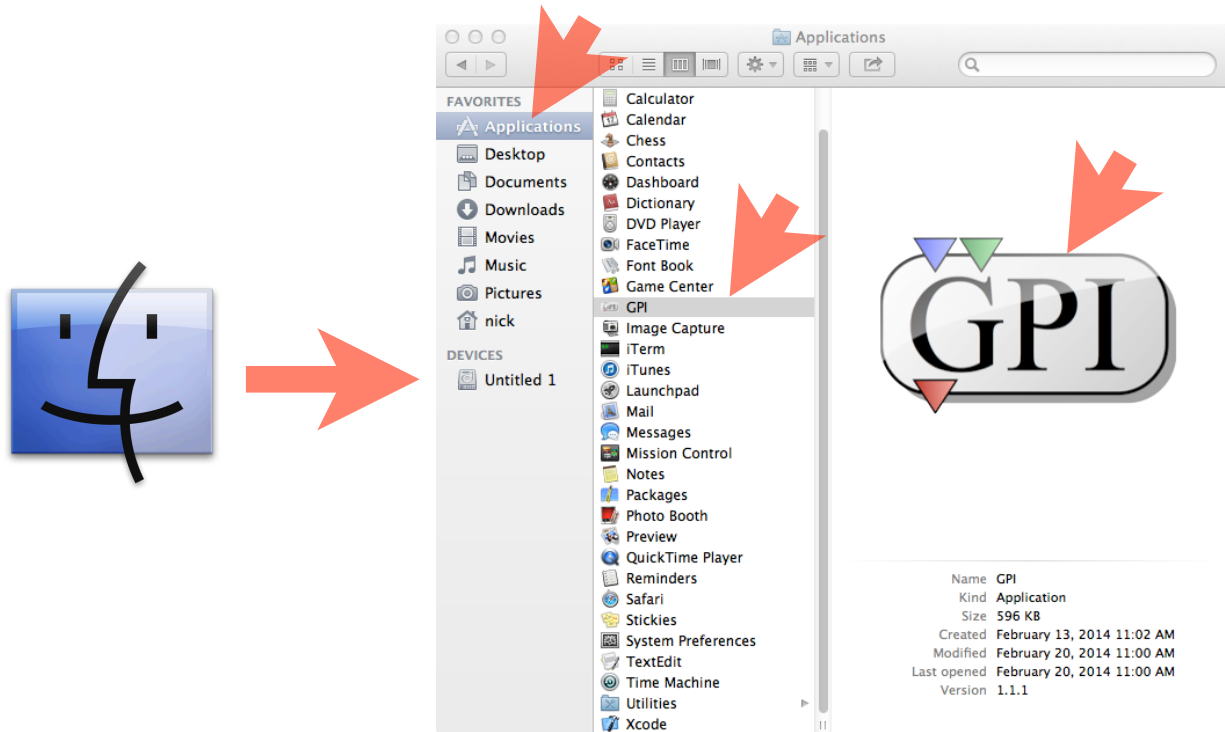



Fig. 3: Navigate to the GPI.app. The App can be stored in the dock for easy access and allows OSX to hold file associations for GPI such as .npy, .pickle, .net, .hdf5, .png, .jpg, .csv, etc...

Once GPI is open, head to the 'Help' menu, click 'Documentation' and this will bring up a new finder window containing the 'doc' directory. From the finder navigate to a network file named 'SpiralReconEx.net', it is located under the 'Training' folder, then drag it to the canvas.

NOTE: on Mavericks the Terminal.app sometimes steals focus so click the “” icon on the dock, then click on the GPI canvas to regain the GPI main menu.

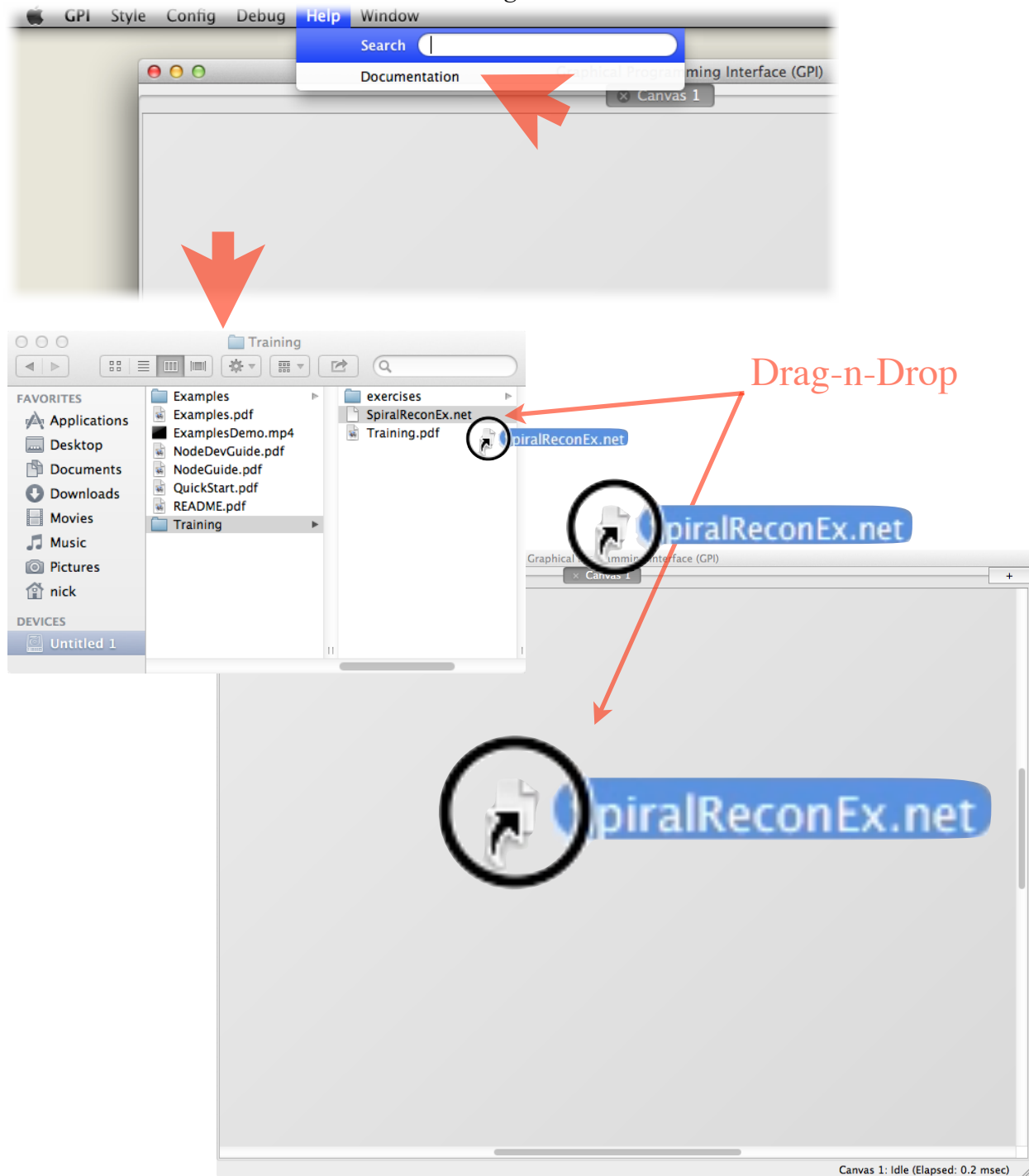


Fig. 4. From the 'doc' folder, network, data examples and training materials can be drag-n-dropped onto the canvas.

The instantiated spiral network should look like Fig. 5. The areas boxed in red are analysis tools that can be explored by right-clicking on the node (e.g. ImageDisplay and Matplotlib). The rest of the network contains the usual non-Cartesian reconstruction elements (Grid, SDC, FFT and Rolloff). The ReadPhilips node converts many of the Philips proprietary file formats, that are produced by the scanner, into information dictionaries (viewable by dictionquery) and data arrays (as Numpy arrays). The only node specific to spiral reconstruction is the SpiralCoords node, which generates spiral trajectory/ coordinate points for gridding k-space data.

This same basic network is essentially used for other types of reconstruction such as PROPELLER, radial, FLORET, 3D spherical spiral, and 3D stack of spirals. With few exceptions, the only element that differentiates these reconstructions is the coordinate generating node. This framework allows total flexibility for applications to introduce additional processing at any point within the network.

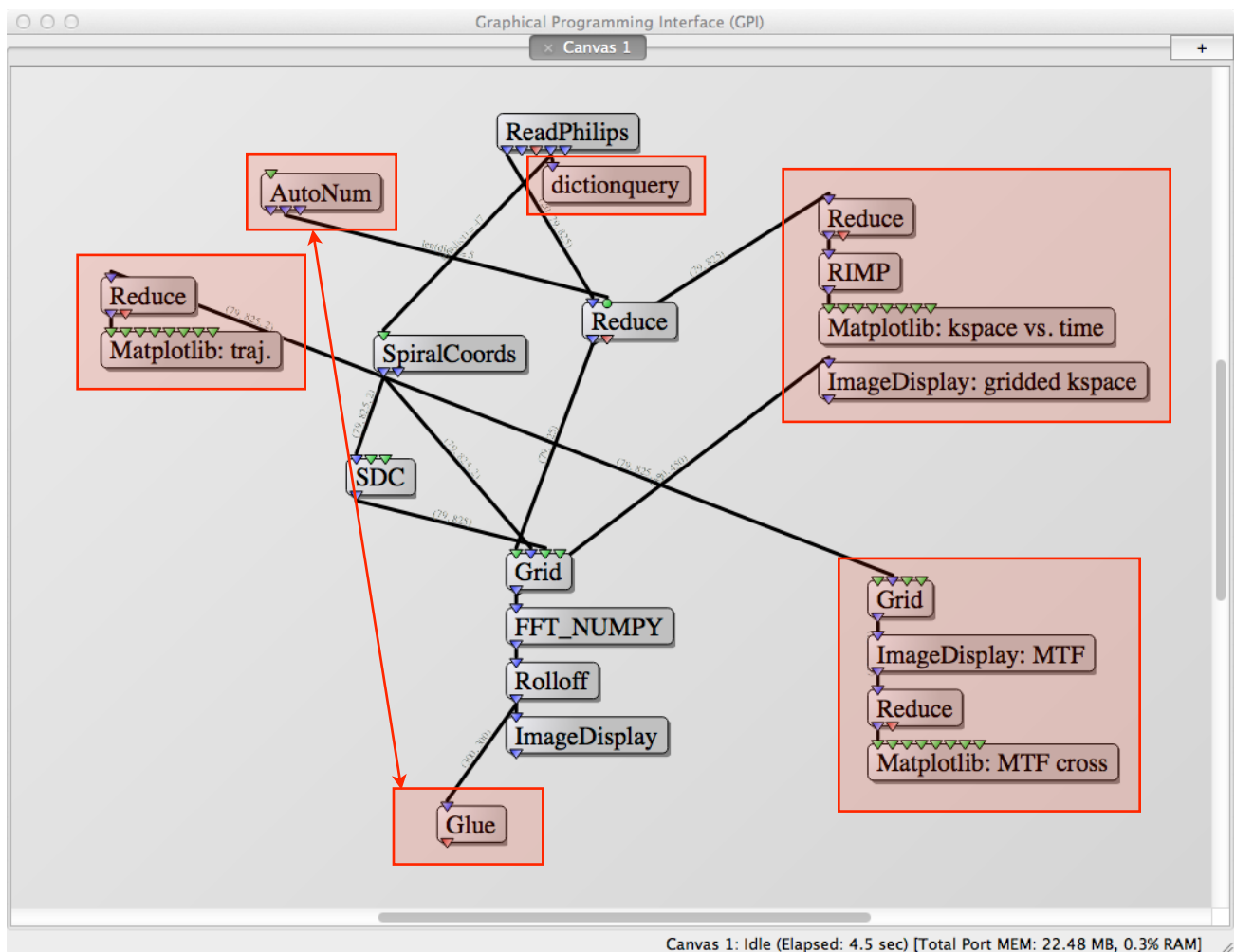


Fig. 5. Example spiral reconstruction network with analysis tools connected at various points of interest.

User Interface:

GPI network manipulation tools are mostly accessible within the **Mouse Menu** which can be opened by right-clicking on the **Network Canvas** (Fig. 1). From this menu, nodes can be instantiated, copy/paste, saved/loaded as a network, and the canvas can be completely cleared. **Node** code, network description and data files can also be drag and dropped on the canvas from a file browser. This will instantiate the **Nodes** associated with the code, description or data that has been dropped.

As **Nodes** are instantiated on the canvas they briefly turn dark grey indicating that they are running their kernel (compute() function) code. As **Nodes** are connected and **Widgets** are modified they automatically trigger compute() events which flow down connections of the network. This execution flow can be paused by selecting 'Pause' from the **Mouse Menu**. Networks and **Widgets** can be modified in pause mode and will start execution when un-paused.

If GPI is launched from a terminal, all **Node** and **Network Canvas** information will be echoed to the terminal window. This can aid in debugging runtime errors. Additionally, the **Nodes** and **Ports** have a color code to indicate their status (Fig. 6). The background color code is as follows: idle (light grey), running (dark grey), error (red), warning (yellow). Error states do not allow the **Node** to be run again (the **Node** can be copied and pasted to retry). The warning state (like the idle state) will allow the **Node** to reenter the event-loop. The input **Ports** can be blue or green for required or optional input respectively. The output **Ports** are blue (indicating success), yellow (unchanged), and red (no data and no downstream event).

The **Network Canvas** has a tabbed interface allowing more canvases to be instantiated and run simultaneously. **Nodes** can then be copied and pasted between canvases allowing the user to easily consolidate emergent or spontaneous design changes. Layouts, spawned from the **Mouse Menu**, can also be used to tie **Widgets** from various **Nodes** together, making a single unified 'control panel' for networks on a single canvas. Finally, macro-**Nodes** can also be used to aggregate multiple **Nodes** into a single representation on the canvas. A macro-**Node** is instantiated from the **Mouse Menu**. Connections between the 'Input' and 'Output' **Nodes** are enveloped within the macro when either **Node** is double-clicked.

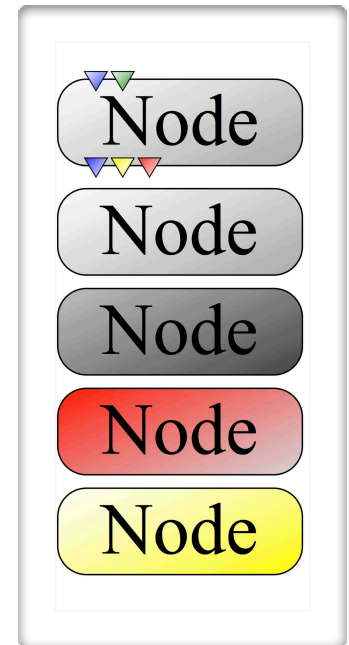


Fig. 6: Node and Port color code.

Mouse & Keyboard Command List:

This section lists how to interact with the display items shown in Fig. 1.

Network Canvas:

- Left Click:** select **Nodes** or drag to select multiple **Nodes**
- Middle Click + Drag:** move the **Network Canvas** relative to view
- Scroll Wheel:** zoom in and out of the **Network Canvas**
- Right Click:** **Mouse Menu**
- Tab:** cycle **Node** selection
- Ctrl/⌘+O:** organize selected **Nodes**
- Ctrl/⌘+L:** load a network
- Ctrl/⌘+S:** save a network
- Ctrl/⌘+C:** copy selected **Nodes**
- Ctrl/⌘+V:** paste selected **Nodes**
- Ctrl/⌘+P:** toggle the pause state of the canvas (execution will halt until un-paused)
- Spacebar:** open **Node Menu** for selected **Nodes**
- Delete/Backspace:** delete selected **Nodes**
- Up, Down, Left, Right Arrows:** move selected **Nodes** across the **Network Canvas**

Node Menu:

- Left Click:** select **Widget**
- Left Double Click:** show/hide **Widget**
- Middle Click + Drag:** grab **Widget** and place in a layout or macro-node menu
- Right Click:** open **Widget-Port** menu
- Tab:** cycle focus through **Widgets**

Node:

- Left Click:** select **Node**
- Left Click+Drag:** move **Node**
- Right Click:** **Node Menu**
- Ctrl/⌘+Right Click:** open the **Node**'s python code.
- Hover:** display compute() wall time and total outport memory used

Macro-Node:

- Left Double Click:** open/close the macro

Edge:

- Right Click:** **Edge** delete menu

Port:

- Middle Click:** raise all connectable **Ports**
- Middle Click+Drag:** draw connection **Edge** to another **Port**
- Hover:** display **Port** type and held data information