Fast Neurite Tracer

User Manual

Author: GOU Lingfeng
Contact: goulf@ion.ac.cn
Date: 24 Feb 2017
Web site: https://fast-neurite-tracer.sourceforge.io/

Version: 1.0

Table of Contents

1 Introduction

Fast Neurite Tracer (FNT) is a tool for semi-automated neurite tracing. FNT can handle big imaging data such as fMOST data. It can also process other types of light imaging data in general. FNT is designed to be

- accurate in tracing (each tracing step needs your confirmation),
- fast (operations using computer mouse is reduced with automatic searching),
- scalable (large volume data of tera-bytes in size are supported).

Other features include

- the results can be exported to SWC files,
- automatic detection of cycles during tracing,
- support for data of multiple channels,
- support for both 8-bit and 16-bit image data.

1.1 Definition of Terms

Some commonly used terms in FNT are defined here:

Tracing The reconstruction of neurons, especially neurites including both axons and dendrites.

Node A reconstructed neuron is represented by a collection of connected nodes in a tree structure. Nodes are associated with the properties including 3D positions, radii and types.

Edge Nodes connected in a linear structure, flanked by branching nodes or terminal nodes, are collectively called an edge. Note that a branching node is associated with multiple edges.

Vertex End points of edges, namely branching nodes or terminal nodes.

Putative Path Collection of nodes in a linear structure, returned by the execution of an algorithm, that is used to extend an existing edge or to create a new edge. A putative path is not included in the tracing results until being confirmed by the user.

Current Position The 3D location around which visualization and calculations are performed. It is located at the center of the 3D viewer and represented by a circle.

Current Edge If the current position is on an edge, then that edge is the current edge.

Target Position The 3D location picked by the user. It is represented by a cross.

Target Edge If the target position is on an edge, then that edge is the target edge.

2 Preparation

To get started, you need to load imaging data for tracing first. You can either choose and open a previously saved FNT file (**File/Open**) or create a new FNT file (**File/New**). Creating new FNT files requires the location of imaging data and an optional **affine transformation** matrix as input.
For small imaging data in a single file, you can specify the file path, or URL if it is in a remote computer, and open it directly. The file format can be either multi-image TIFF file or 3D NRRD file. Whether a file is small enough to be opened directly depends on the computer hardware and current computer state like memory usage. Some basic requirements are

- there is enough main memory for uncompressed 3D data,
- there is enough GPU memory for uncompressed 3D data,
- image size in each direction is not larger than `GL_MAX_3D_TEXTURE_SIZE`.

Big imaging data in a series of TIFF files need to be **converted** to small cubes. The location (path or URL) of the CATALOG file generated during conversion is used to load the big imaging data.
Multiple image files or CATALOG files can be combined by writing a new **CATALOG file**. FNT supports data of multiple channels in this approach.

2.1 Affine Transformation

Detailed explanation of affine transformation is available at its **Wikipedia page**. FNT uses affine transformation to specify image resolution or to align multiple image volumes. The followings are some examples of using affine transformation in FNT.
When creating a new FNT file with a CATALOG file as input, the transformation matrix is typically left unchanged (**A**). Usually, resolution information is specified in CATALOG file already and there is no need to do further transformation.
With a single image file, the affine transformation is typically used to specify image resolution. Many TIFF files provide no resolution information and some provide wrong resolution information. There is no standard way for TIFF files to provide Z resolution. Therefore FNT does not automatically obtain resolution information from file and affine transformation matrix is used to manually specify image resolution. Here the file `test.tiff` is specified with X/Y resolution 0.3 µm/pixel and Z resolution 1µm/pixel (**B**).

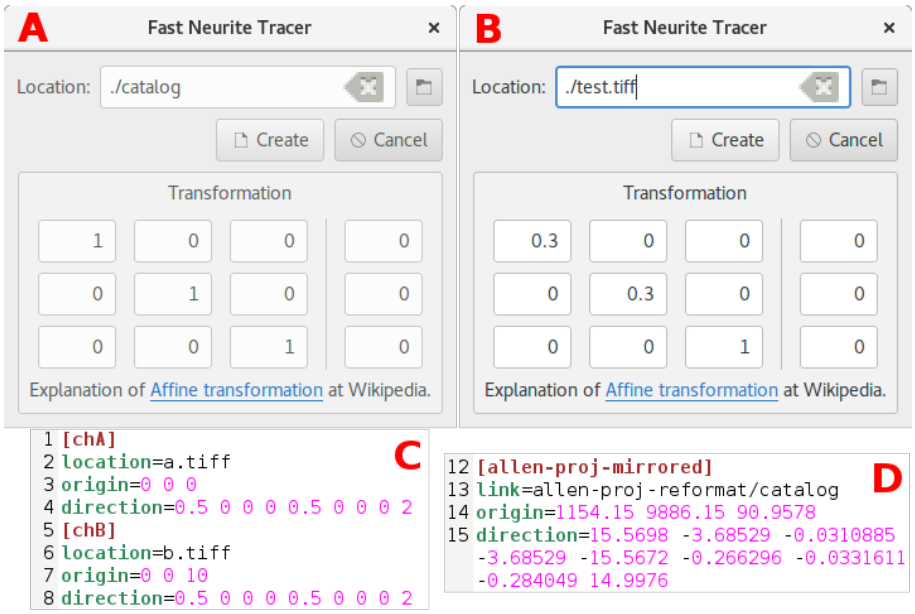

Figure 2.1 *Affine transformation examples.* **A** and **B** are examples of creating new FNT files. **C** and **D** are examples of writing CATALOG files.

Suppose TIFF file `a.tiff` has voxel size of 0.5µm/0.5µm/2µm. File `b.tiff` is the same as `a.tiff` except that the first five slices are missing. They can be aligned and combined if the correct transformation matrix is given in the CATALOG file (**C**).
In a real example, some neurons are traced on one hemisphere of the mouse brain, while the injection sites for most of Allen Brain Atlas (ABA) projection experiments are on the other hemisphere. To compare the traced results with ABA projection experiments, we can obtain the mirror image of all reformatted ABA projection density images by the transformation. The transformation matrix given here is calculated from a mirror plane (**D**).
Note that, in CATALOG files, the **`origin`** property corresponds to the column vector on the right side of the dialog box and the **`direction`** property corresponds to the 3x3 matrix on the left side in column-major order.

2.2 Big Data Conversion

Big imaging data that cannot be loaded at a time is split to small cubes using the `fnt-slice2cube` command. Run `fnt-slice2cube --help` for help on detailed usage. The following is a simple example.
Suppose we have the following files

```
$ ls *.tif
image1.tif image2.tif image3.tif image4.tif image6.tif
```

The resolution for each file is 0.3µm/pixel and each slice is 2µm thick.
First create a list of file names

```
$ ls *.tif > list
$ cat list
image1.tif
image2.tif
image3.tif
image4.tif
image6.tif
$ wc list
 5  5 55 list
```

There are 5 files in the list, but 6 files are expected (`image1.tif` to `image6.tif`).
So use a text editor and add one line to remedy the missing file

```
$ vi list
$ cat list
image1.tif
image2.tif
image3.tif
image4.tif
BLACK
image6.tif
```

Then start the conversion

```
$ fnt-slice2cube -i list -r 0.3:0.3:2 -c 256:256:6 -d 16:16:1 -o
cubes
```

It is important to set the right cube sizes and downsample factors. If the cube sizes are too big, they may fail to be loaded into the graphics card. If the cube sizes are too small, too many small files will be generated and that lowers file system efficiency.

If the downsample factors are too small, the downsampled channel will be too big to be loaded. If the downsample factors are too big, the resolution of the downsampled channel will be too low. Besides, the cube sizes should better be chosen such that the resulting cubes have similar physical lengths at X/Y/Z directions. The downsample factors should better be chosen such that downsampled image has similar resolutions at X/Y/Z directions.
It is also possible to do simple conversion, without splitting or downsampling, by specifying `-d 1:1:1` in the command line arguments. A series of single-image TIFF files cannot be directly opened by FNT as a single volume. In addition, the extra channels in a multi-channel multi-image TIFF file are ignored by FNT. In these two cases, simple conversion helps.

2.3 CATALOG File

FNT uses a CATALOG file to organize multiple channels in a dataset. It is an **INI file** with restricted properties.

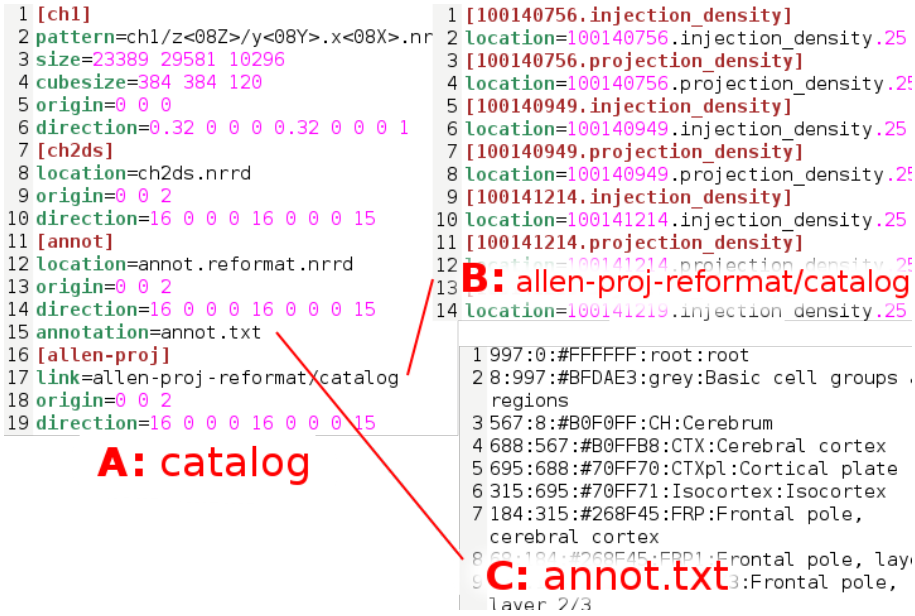

Figure 2.2 *CATALOG file examples.* **A** is a CATALOG file containing 4 sections. **B** is the CATALOG file referred to in section 4 (allen-proj) of **A**. **C** is the annotation file refered to in section 3 (annot) of **A**.

The section for a single image file has the **`location`** property, which is the path or URL of the image file (section 2 in **A**).

The section for converted big imaging data has the **pattern** property (section 1 in **A**). The pattern, when substituted with X/Y/Z coordinates, provides the paths or URLs of small cubes. For example, `ch1/z<08Z>/y<08Y>.x<08X>.nrrd` denotes the file `ch1/z00000120/y00004096.x00000256.nrrd` when substituted with X/Y/Z coordinates . The pattern is similar to that used by the C function **printf**. Size information for the whole volume and for each small cube is also required. The properties **size** and **cubesize** are for this purpose.
Section containing the link to another CATALOG file require the **link** property (section 4 in **A**). The link is a path or URL to the target CATALOG file (**B**).
Each section can have optional **origin** and **direction** properties. If omitted, **origin** has the default value of and **direction** has the default value of (i.e. the identity matrix). An affine transformation is specified with these two keys and is applied to the image (or images if the section is a link) in this section.
Images with integer values can contain the **annotation** property (section 3 in **A**). It is a path or URL to a plain text file containing annotations to different values (**C**). This example shows an annotation file converted from ABA's annotation JSON file. The columns from left to right are structure ID, parent structure ID, color, structure abbreviation and name.

3 Browsing Data

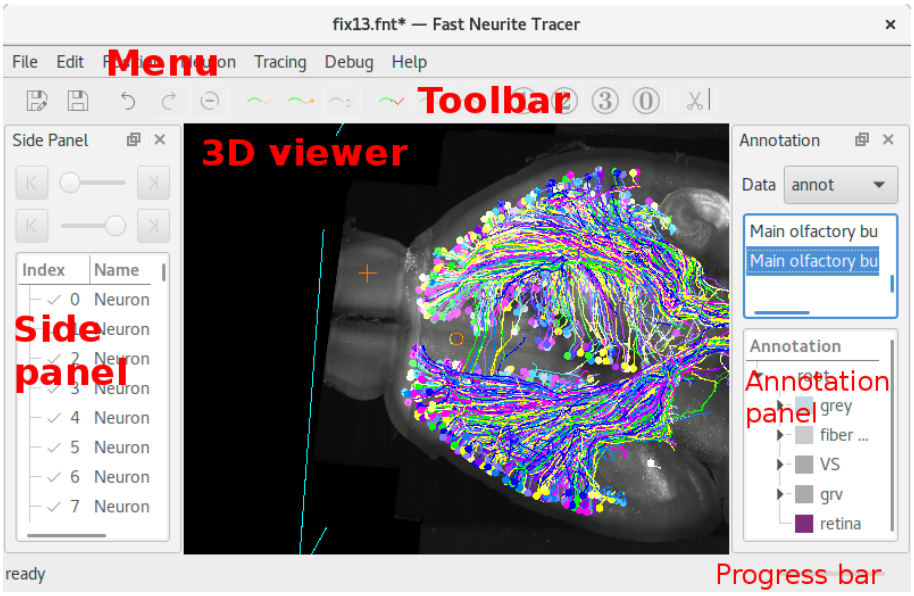

Figure 3.1 *FNT user interface.*

Neurite tracing functionality is built upon interactive browsing of imaging data. The following sections explain how to browse imaging data in FNT.

3.1 Zooming

To zoom in, scroll the mouse wheel upwards when the mouse cursor is in the 3D viewer. To zoom out, scroll downwards. Note that scrolling in 2D mode has no zooming effect.

3.2 Rotation

To rotate the view, push down left mouse button in the 3D viewer, move the mouse to the direction of rotation, and release left mouse button afterwards.

3.3 Visualization

The 3D viewer shows the maximum intensity projection of volumetric imaging data by default. Visualization in 2D mode is activated by holding the **Shift** key. A cross-section of the 3D volume is shown in this mode. To show the subsequent neighboring cross-sections, scroll the mouse wheel.

3.4 Change Current Position

FNT handles big imaging data by only showing the data cubes around the current position. The current position is indicated by a circle at the center of the 3D viewer. To browse data at another position, change the current position to a new position. If the X/Y/Z coordinates of the new position is known, they can be entered in a dialog box to change the current position (**Position/Goto position**). If the target position exists, it can be assigned to the current position (**Position/Goto target**). To jump to the soma of a neuron, select the neuron in the side panel and execute **Neuron/Goto soma**.
If the current position is on an edge, moving the first slider in the side panel will change the current position to neighboring positions on the edge. The two buttons besides the slider are shortcuts for jumping to two vertices of the edge.
The current position might be changed automatically after undoing/redoing or after changing the state of a vertex.

3.5 Pick Target Position

The target position is indicated by a cross. It serves as an input for many tracing and editing operations.
The current position can also be assigned to the target position (**Position/Pick current**). If the target position is on an edge, moving the second slider in the side panel will change the target position to neighboring positions on the edge. The two buttons besides the slider are shortcuts for picking two vertices of the edge. To pick the soma of a neuron, select the neuron in the side panel and execute **Neuron/Pick soma**.
In 3D mode, left mouse clicking on an edge will pick the node of the edge under mouse cursor. If mouse cursor is not on an edge when clicking, a line segment appears. Rotate the view and perform another mouse click near the line segment. The two line segments determine a 3D position and that position is picked as the target position. An example is shown in **A**. In 2D mode, a position on the currently shown cross-section is picked with just one mouse click (**B**).

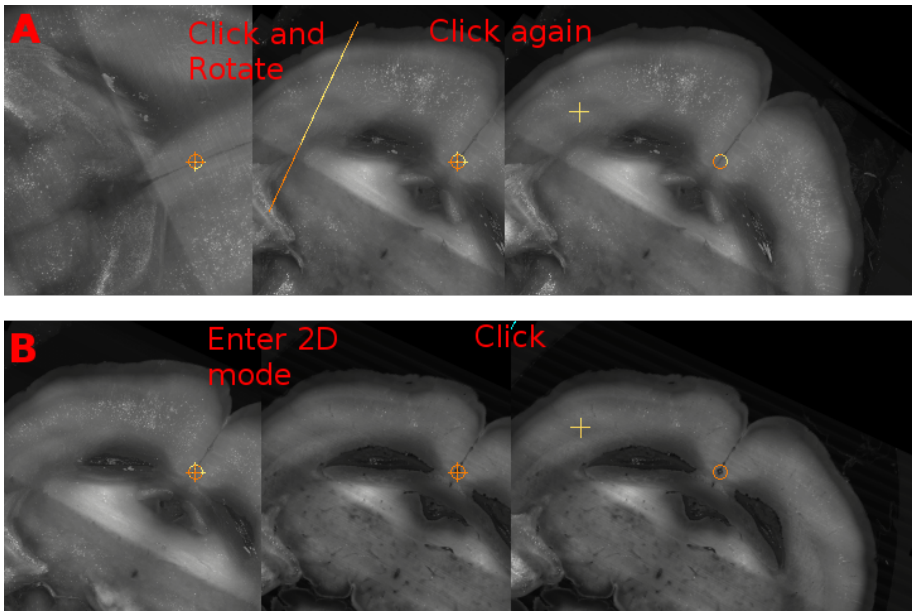

Figure 3.2 *Examples of picking target position.* **A** shows how to pick a position in 3D mode. **B** shows an example in 2D mode.

The target position might be changed automatically after undoing/redoing.

3.6 Data Loading

By default, FNT automatically loads imaging data cubes near the current position. Automatic data loading can be disabled by unchecking **Position/Load data auto**. After that, data can be manually loaded by executing **Position/Load data**. Progress of data loading is shown in the status bar below the window. Data loading can be cancelled by executing **Edit/Abort**.

3.7 Annotation

If there is an annotation channel (a channel with the **`annotation`** property), it can be activated by selecting it in the annotation panel. After that, the annotation data at the target position will be shown in the annotation panel.

4 Tracing

The main tracing process consists of three steps: finding a putative path, examining it and accepting it. Accepted paths are used to create new edges or extend existing edges. Edges can be modified or deleted. It is possible to import neurites from SWC files or FNT files (**Edit/Import SWC|FNT**). All these operations can be undone (**Edit/Undo**). Don't forget to save the tracing results before closing (**File/Save**).

4.1 Finding

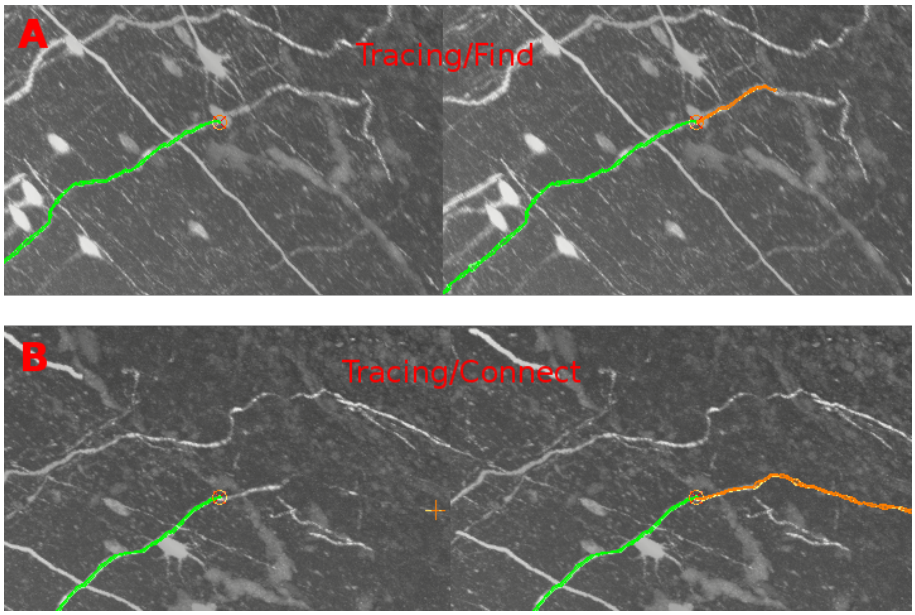

Figure 4.1 *Examples of finding putative paths.* **A** shows how to find a putative path starting from the current position. **B** shows how to find a putative path connecting the current position and the target position.

Putative paths can be found in two approaches. The first approach is to execute **Tracing/Find** and obtain a putative path starting from the current position (**A**). In more difficult cases, pick a target position on the neurite being traced and execute **Tracing/Connect**. A putative path connecting the current position and the target position will be found (**B**).
Node properties of the found putative path can be changed by executing **Tracing/Refine**. For example, the Rayburst refine method can be used to estimate node radius. In addition, a refine method can be chosen to run automatically for every putative path.

The progress of finding, connecting or refining is shown in the status bar below the window. These actions can be cancelled when needed (**Edit/Abort**).

4.2 Examining

The correctness of a putative path can be evaluated by comparing it to real image signals of neurite structure. For difficult cases, operations involving zooming, rotation and 2D visualization can often help resolve complex neurite structure under human perception.

4.3 Accepting

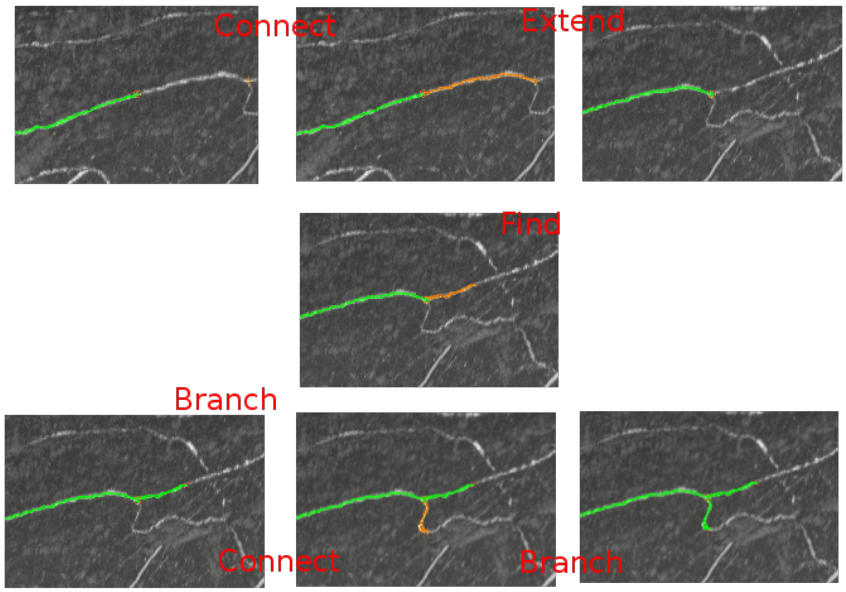

Figure 4.2 *Examples of accepting paths.* Note the difference between **Extend** and **Branch**.

If the putative path is examined to be incorrect, just ignore it and find another one. If it is correct, add it to the tracing results by executing either **Tracing/Branch** or **Tracing/Extend**.
Extending operation is slightly different from branching operation in that the current position is changed to the new end point after the operation. Extending operations are for tracing part of a linear structure while branching operations are for adding initial segments of branches at a branch point.

4.4 Traversal

A neuron has a tree structure. A neuron tree can be completely traversed in a depth-first manner. FNT allows depth-first traversal by recording the state of each vertex. If all vertices of a traced neuron are in finished state, the neuron is considered to be completely traced. The following rules can ensure the completeness of tracing results.
When tracing of a branch comes to an end, change the state of the vertex at the terminal by executing **Tracing/Done**. This operation will set the state of the vertex to finished. If there are unfinished vertices, the current position will be automatically changed to the nearest unfinished vertex so that one can immediately start tracing the unfinished branch.
At a branch point, trace the initial segment of each branch before continue tracing one of the branches to the end. The end points of all the initial segments are all marked in unfinished state so that they will be automatically accessed at a latter time. See the figure in the previous section for an example of tracing at branch points.

4.5 Deletion

Edges can be deleted by executing **Tracing/Delete**. If both the current position and the target position are on the same edge, the part of the edge flanked by the two positions is deleted. Otherwise if the target position is on an edge, the target edge is deleted. If only the current position is on an edge, the current edge is deleted.

4.6 Modification

Type of a node can be modified by executing **Edit/Change node type**, **Tracing/Mark [1-3]** or **Tracing/Clear mark**. Node radius can be modified by executing **Edit/Resize node**. These operations modify the node at the target position.
Neurite type of an edge can be modified by executing **Edit/Change neurite type**. The modification is applied to the whole branch starting from soma associated with the target position.

4.7 Neurons

To create a neuron, pick the soma position and execute **Neuron/New neuron**. The list of all neurons are shown in the side panel. Every neuron is assigned a name and

can be renamed by executing **Neuron/Rename neuron**. A neuron can be deleted without removing its associated edges (**Neuron/Remove neurons**) or purged by removing all associated edges (**Neuron/Purge neurons**).

4.8 Shortcut Keys

Frequently used operations located in the **Tracing** menu can be activated with just one key stroke to speed up tracing.

Key	Operation
F	Find
C	Connect
R	Refine
D	Done
B	Branch
Space	Extend
1	Mark 1
2	Mark 2
3	Mark 3
0	Clear mark
X	Delete

4.9 Miscellaneous

There are additional useful functions under the **File/Misc** sub-menu, including

- saving the results of traced neurons to SWC files,
- obtaining statistics of traced neurons,
- annotating soma of traced neurons.

The sub-menu can be further extended by including new plug-ins.

5 Configuration

There are several modifiable settings in FNT in the **Options** dialog box (**File/Options**).

5.1 Cache

Remote imaging data are cached in computer memory. The amount of memory used for cacheing can be configured under **Data/Cache/Memory usage**. Parallel fetching of remote data is supporting by configuring **Data/Cache/Cache parallelism**. Pre-caching (**Data/Cache/Pre-cache parallelism**) can be used to reduce waiting time while loading the data. It can be disabled by setting the value to zero.

5.2 Loading

The amount of data loaded is configured by **Data/Loading/Radius**. If this value is too small, more frequent data loading is needed, and the maximal length of putative paths is shortened. On the other hand, bigger value requires more main memory and more video memory and increases time for data loading.

5.3 Channels

FNT allows the visualization of at most 9 data channels at the same time. Selected channels for visualization are divided to three groups that can be configured in **Data/Channel**. Channels in the same group are visualized in the same color. Each channel is covered by preceding channels in the same group and groups are blended together without covering. Tracing operations use channel 1 in group 1. The example shows how three channels including original data, downsampled data and one of ABA's projection experiment, are visualized simultaneously.
Transfer function for visualization can be modified for each channel. Use the sliders or text entries to modify it. To expand the range, enter an out-of-range value in the text entries. To shrink the range, click **Shrink**. Use **Reset** to restore the default range and reset the transfer function. Note that changing transfer function might result in different computation outcomes.

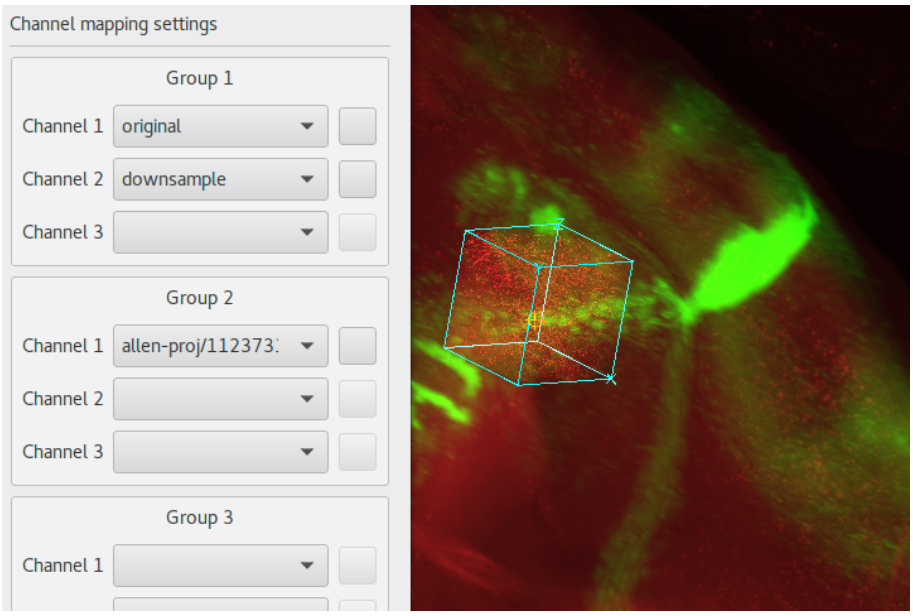

Figure 5.1 *Examples of visualizing multiple channels.* The left side shows the settings in the **Options** dialog box. On the right side is the 3D visualization of the three selected channels.

5.4 Display

Magnification factor of the 3D viewer can be modified with **Display/General/Magnification**. Values larger than 1 mean magnifying and can be used to reduce GPU usage at the cost of reducing quality. Values less than 1 mean minifying and can be used to produce high-quality visualization.
Most colors used in FNT can be customized in **Display/Color**. **Color[0-7X]** have different representations depending on the current color mode. Use the checkbox to change color mode.
Total number of slices for volume rendering with maximal intensity projection can be modified (**Display/Volume/Total slices**). Larger value results in better visualization quality. The other setting is the number of slices actually shown (**Shown slices**). Smaller value lowers GPU usage and can also be used to reduce the complexity of visualized image.

5.5 Computation

Progress information during computation can be displayed in the 3D viewer at a given interval (**Compute/General/Update interval**). The process of computation can be cancelled (**Edit/Abort**).

Different algorithms can be used for each type of computation. Parameters for a specific algorithm can be modified. The **Refine (auto)** computation is performed for each putative path automatically. To disable it, choose the dummy algorithm. The set of algorithms can be further extended by including new plug-ins.

6 Tools

FNT includes several programs of different utilities:

- `fnt-slice2cube`, to convert TIFF slices to small 3D cubes;
- `fnt-dist`, to align neurons and compute the dissimilarities between neurons based on the topology and geometry of their tracing results;
- `fnt-fromswc`, to convert SWC files to FNT files;
- `fnt-toswc`, to convert FNT files to SWC files;
- `fnt-join`, to join multiple FNT files into one output FNT file;
- `fnt-split`, to split one FNT file to multiple output FNT files;
- `fnt-lint`, to find putative tracing errors in FNT files.

Run `COMMAND --help` to see the usage.

7 Miscellaneous

7.1 Files and Directories

Under Linux or Mac OS, FNT uses the `.fnt` directory under home directory to store private data. Under Windows, FNT uses the `fnt` directory under user directory for that purpose.

7.2 License

FNT is licensed under **GNU GPL version 3 or later**.

7.3 Author

- GOU Lingfeng <**goulf@ion.ac.cn**>