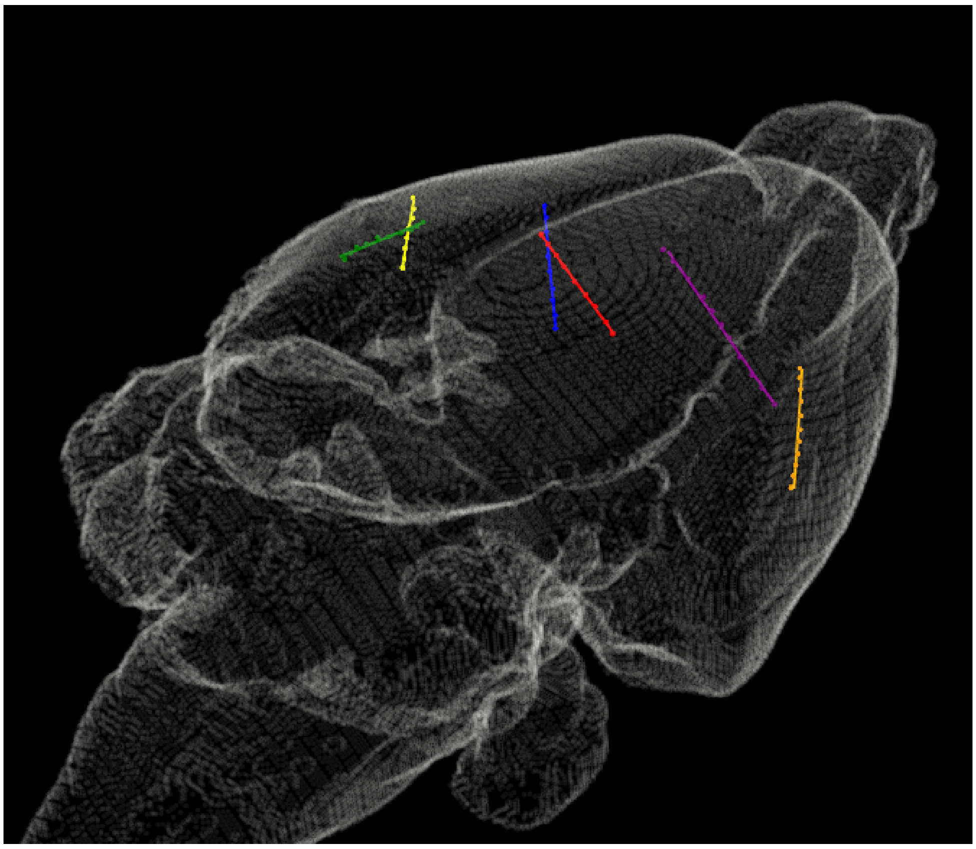
User manual

TRACER: a **T**oolkit for **R**econstructing **A**natomical **C**oordinat**E**s in **R**ats

v. 2021 -4

October 1, 2021



Introduction

TRACER is a python-based toolbox for visualizing user-defined anatmoical features such as the trajectories of recording electrodes (e.g Neuropixels) in 2D or 3D volumes. The anatomical delineations, 2D and 3D brain templates are sourced from the Waxholm Space atlas of the adult Sprague Dawley rat brain, version 4.0 (https://www.nitrc.org/projects/whs-sd-atlas). The TRACER toolbox consists of three main packages serving distinct functions: one for generating coordinates to target specific brain regions prior to surgery, one for registering electrode track locations post-hoc using histological images, and one for all-purpose 2D and 3D visualization of virus expression, anatomical tracers or lesions. NOTE: For Neuropixels we included the 175um tip + 9600um electrode surface length (20um * 960 electrodes *1/2 row per electrode).

Contents

1 Requirements

To start working with TRACER you will need:

- TRACER package downloaded on your local computer from `https://github.com/Whitlock-Group/TRACER`
- Images of individual histological sections from rats (coronal, sagittal, or horizontal) in .jpg format.
- Python 3.8.5 or above (Terminal or Python IDE, e.g. Spyder, Pycharm, VS).
- A computer mouse with scroll wheel and a functional keyboard.
- The Waxholm Space rat brain atlas files can be downloaded from the NITRC website (`https://www.nitrc.org/projects/whs-sd-atlas`). The files you will need are:
  - `WHS_SD_rat_T2star_v1.01.nii.gz`
  - `WHS_SD_rat_brainmask_v1.01.nii.gz`
  - `WHS_SD_rat_atlas_v4.nii.gz`
  - `WHS_SD_rat_atlas_v4.label`

2 Setup of working environments

It is recommended that non-experts use an Anaconda in Python, which will help with downloading the necessary packages and updating existing ones. However users can also use PyCharm or any other Python IDEs. TRACER requires users to install Numpy, Scipy, matplotlib, vedo, opencv-python, Pillow, scikit-image, scikit-spatial, nibabel, nilearn, tabulate, keyboard and mplcursors packages as dependencies. The required version specifications of these packages are listed in the file 'requirements.txt' in TRACER.

2.1 Anaconda environments

Anaconda can be downloaded for free from their website (https://www.anaconda.com). Once installed Anaconda will already include Spyder. To create a new environment in Anaconda:

- go to `Environments` on the left panel,
- click `create` in the bottom left,
- name your environment and in `Packages` select both Python (3.8) and R.

After a new environment is created, a set of packages will need to be installed in order to use TRACER, which is done with the following steps:

- open the terminal by clicking the arrow near the name of you environment,
- navigate to the folder where requirements.txt is
- run `pip install -r requirements.txt`

2.2 Other environments

In PyCharm, users can install the dependence packages accordingly through project interpreter. To access project structure, open Settings/Preferences dialog by pressing Ctrl+Alt+S or by choosing File | Settings for Windows and Linux or PyCharm | Preferences for macOS, then expand the Project node, and select Project Interpreter. Detailed instructions can be found at `https://www.jetbrains.com/help/pycharm/configuring-python-interpreter.html`.

In the terminal, users can go into the directory where TRACER located and run the command `pip install -r requirements.txt` directly to install the dependencies.

2.3 Installation of TRACER

TRACER can be downloaded from GitHub page as a .zip file or cloned to user location directories. Users can select the desired directory to store TRACER. TRACER can also be installed directly through the Terminal by using command lines in Spyder and PyCharm. Users can also install TRACER through version control in PyCharm by following the steps shown below: Pycharm help page.

```
# example shows how to clone TRACER to local directory "./Workspace"
git clone https://github.com/Whitlock-Group/TRACER.git ./Workspace
# example shows how to install TRACER using pip in terminal
pip install git+git://github.com/Whitlock-Group/TRACER.git
```

After TRACER is downloaded/cloned to local directories or installed, users can import TRACER to check if everything is set up correctly.

```
# if TRACER is installed directly or saved in the current working
    directory
from TRACER import *
# if TRACER is saved in other directories
from the_directory.TRACER import *
```

3 TRACER API

3.1 Load Atlas

The Waxholm Space rat brain atlas v4.0 can be loaded through the command given below after importing TRACER,

```
from TRACER import *
atlas = AtlasLoader(atlas_folder= 'folder_path')
```

where the argument `folder_path` is the path of the folder where the Waxholm atlas files will be saved.

3.2 Generating surgical coordinates prior to surgery

This feature of TRACER was originally designed for targeting electrodes to desired regions in the brain, but it works equally well to get coordinates for injecting virus, tracers or making focal lesions. The outputs include a table showing the inclination and insertion distance necessary to target the specified brain regions, as well as visualizations of the insertion trajectory. For planning an insertion, create a 'probes' folder where output files can be saved and run the following commands:

```
from TRACER import *
atlas = AtlasLoader(atlas_folder='folder_path', atlas_version = "v3")
presurgery = ProbesInsertion(atlas, probe_folder)
```

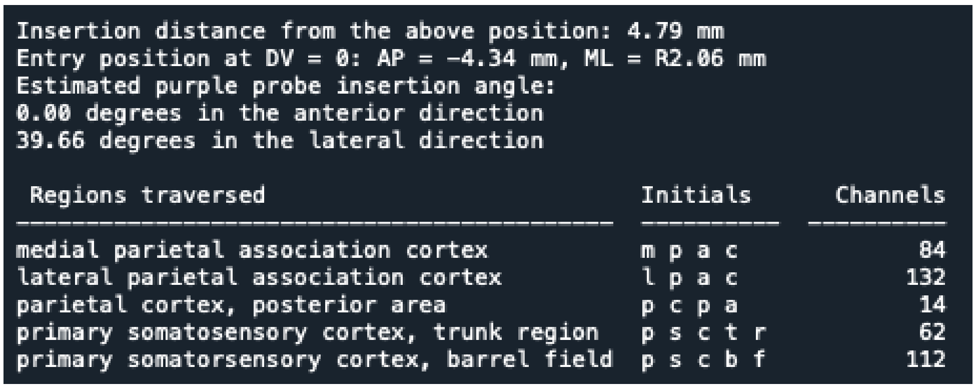
NOTE: Refer to example_code.py for example lines of code showing how to use class functions to run TRACER.

3.2.1 Planning probe insertion

The class file **ProbesInsertion** is essential to plan electrode insertions before any surgeries. Once the script is run the Waxholm atlas will be displayed. The user can then scroll through the slices in the atlas to find the slice containing the region(s) of interest. The commands displayed on the console can be used to view boundaries and define the desired trajectory of the probe (or injection needle, etc.) through the brain. This process can be repeated to include a total of 6 probes.

3.2.2 Visualizing the probe trajectory in the brain

Once the probe trajectory points have been registered and saved, the user can generate a table summarizing the details of the planned implant in the probe folder and save as a .pkl file, shown in Figure 1.



```
Insertion distance from the above position: 4.79 mm
Entry position at DV = 0: AP = -4.34 mm, ML = R2.06 mm
Estimated purple probe insertion angle:
0.00 degrees in the anterior direction
39.66 degrees in the lateral direction

 Regions traversed                          Initials      Channels
-------------------------------------------  ----------   ----------
medial parietal association cortex          m p a c             84
lateral parietal association cortex         l p a c            132
parietal cortex, posterior area             p c p a             14
primary somatosensory cortex, trunk region  p s c t r           62
primary somatosensory cortex, barrel field  p s c b f          112
```

Figure 1. Example output table showing details provided by TRACER once a probe insertion has been planned. This includes the travel length of the probe in the brain, the insertion angle, regions contacted and the number of channels in each region.

2D and 3D visualizations (Figure 2) of the probe trajectory can be generated using the following lines of code in the console:

```
vis3d_presurgery = vis_inserted_probes(atlas, probe_folder = 'path')
vis3d_presurgery.vis2d() #gives readout of probe implants and 2D
    rendering of the planned probe implant
vis3d_presurgery.vis3d() #3D rendering of the planned probe implant
```

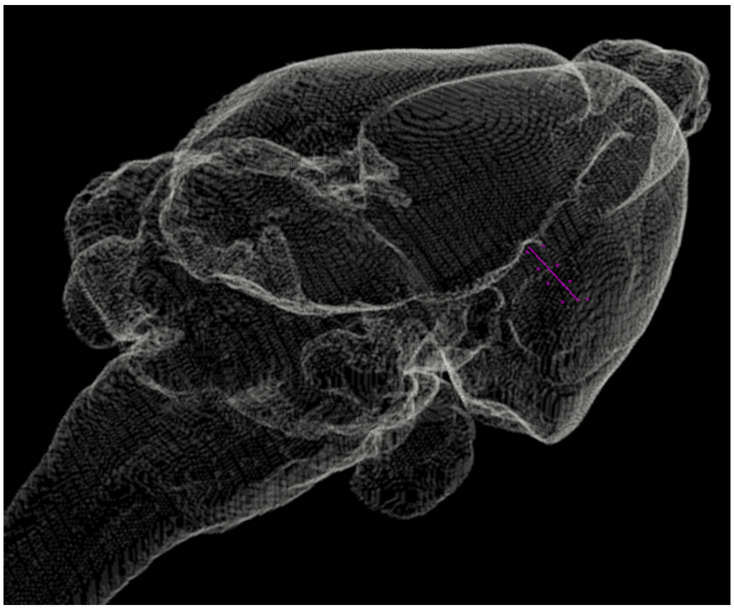


Figure 2. 3D rat brain showing the planned trajectory of the probe. In the API it is possible to rotate and zoom in on areas of interest.

Saved probes can also be reloaded by running the line:

```
presurgery.reload_probes('Probe1') #Probe1 is an example name given to
    the saved probe
```

The script file `probe_insertion.py` provides information about the probe insertion angle, regions traversed by the electrode, the location and the number of recording channels in each region. Close and open a new console every time you run this script.

3.3 Reconstructing probe trajectories

This function allows the user to register, visualize and recover quantitative details about recording electrodes using the tracks left in histological tissue sections. Run `%matplotlib inline` line of code in your console before running the entire script.

3.3.1 pre-processing histological images

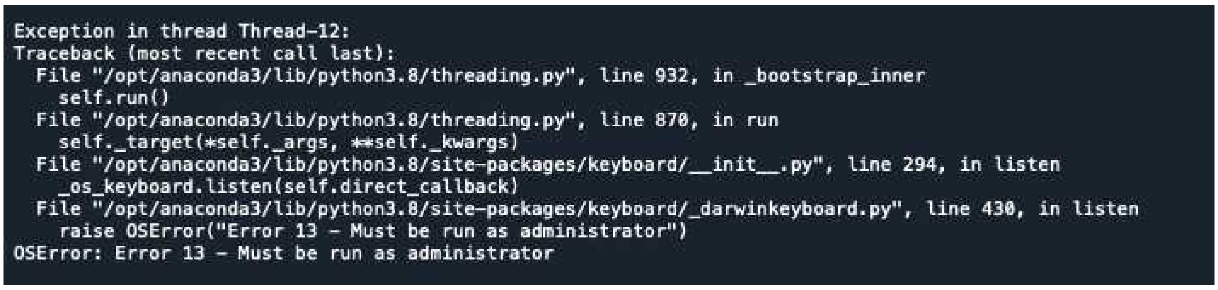
This step is for optimizing the brightness and contrast of histology images prior to probe registration. The pre-processing script is executed by the command:

```
pre-process_hist = pre-process_histology(histology_folder='path')
```

Users should be sure to save all relevant brain slices in the same folder, designated by the 'path' name assigned by the user. It is recommended to give the images consecutive names (e.g. histology01, histology02, etc).

NOTE: if you have multiple images, each image will be individually shown in the script to edit.

NOTE: Mac users should make sure Python has the security authorization to access input from the keyboard, otherwise it may cause an error (see Figure 3). In this case, use the `sudo spyder` command in your terminal to open your desired IDE (Spyder is recommended since it was used to develop the software).



```
Exception in thread Thread-12:
Traceback (most recent call last):
  File "/opt/anaconda3/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
  File "/opt/anaconda3/lib/python3.8/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/opt/anaconda3/lib/python3.8/site-packages/keyboard/__init__.py", line 294, in listen
    _os_keyboard.listen(self.direct_callback)
  File "/opt/anaconda3/lib/python3.8/site-packages/keyboard/_darwinkeyboard.py", line 430, in listen
    raise OSError("Error 13 - Must be run as administrator")
OSError: Error 13 - Must be run as administrator
```

Figure 3. Potential error for Mac users when running `pre-process_histology.py`.

Use the command `sudo spyder` and follow the steps below to run the `pre-process_histology.py`:

- Add the correct path for your folder:
  `histology_folder = Path('/Users/....../Histology')`
- Choose the plane of view with the keyboard letters (**'c'** for coronal, **'s'** for sagittal, **'h'** for horizontal).
- Once the histology image has been adjusted using the controls in the console press **'s'** to save.
- The processed histology image can be found in the folder "Processed", which is created as a subfolder in the "Histology" folder.

3.3.2 Navigating the Waxholm atlas and registering probe tracks

The probe registration step uses images saved previously from pre-processing. It begins by the user scrolling to the appropriate brain slice in the Waxholm atlas, overlaying the user's histology section onto the atlas template, then registering the probe location. Remember to have the correct path to the pre-processed images and other relevant folders (see below).

- Close the console where `pre-process_Histology.py` was running and open a new one.

```
register_probes = ProbesRegistration(atlas,
    processed_histology_folder='path')
probe_folder = 'path_probes'
```

- Add the path:
  `processed_histology_folder =`
  `Path('/Users/..../Histology/processed').`
- The atlas section and user's histology section will appear on the screen, after which the user specifies overlay points by clicking **'t'** that the software uses to align and overlay the two sections.
- Choose at least 4 anchor points **in the same order** for both images (Figure 4A and B). Accuracy in selecting the points will determine how well the histology image is fit onto the atlas template. The numbering is intended to help the user maintain the same the order of points clicked on atlas and histology slices.
- Press **'h'** to match histology and atlas sections, and a new figure will appear with the histology and atlas slices overlaid (Figure 4C)
- Press **'p'** to visualize the regional boundaries of the Waxholm atlas (Figure 5A).
- After having pressed **'h'** and **'p'** it is possible to either visualize colors overlaid on the different regions by pressing **'v'** or start registering the probe track by pressing **'r'**.

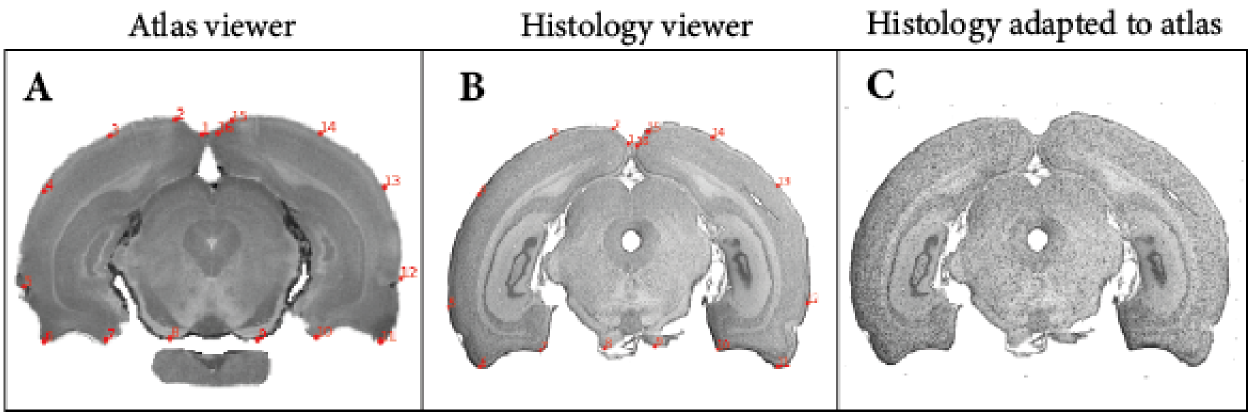


Figure 4. **A** shows the numbered points clicked on the atlas slice, **B** shows the user-loaded, Nissl stained histology slice, and **C** shows the histology slice warped and overlaid on top of the atlas section.

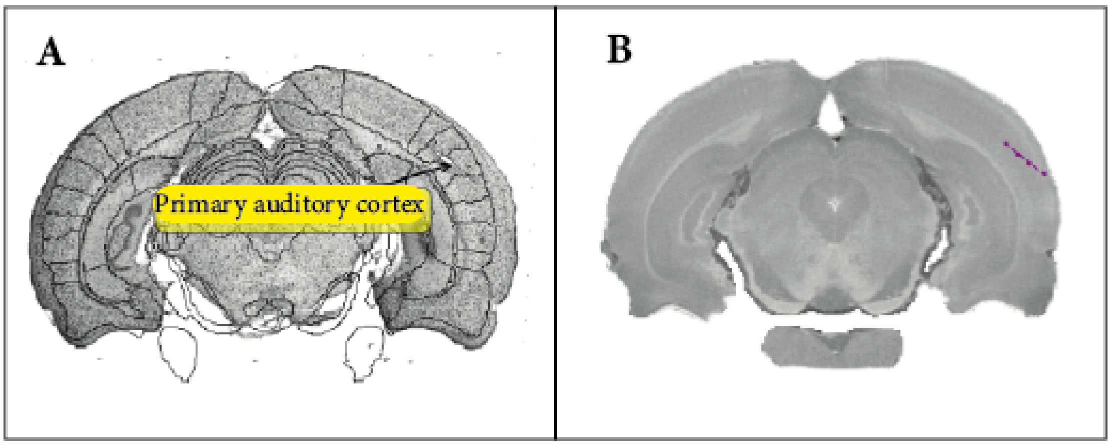


Figure 5. **A** shows the histological section from Figure 4 with activated grid lines delineating distinct anatomical regions. **B** shows the estimated probe trace in primary auditory cortex.

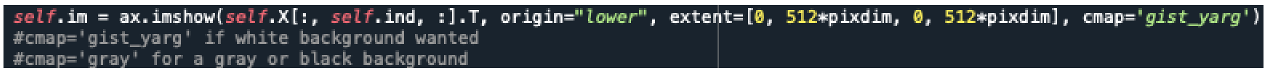
- Click the points on the probe tracks.
- When registering probe tracks, it is important that the first and the last point clicked correspond, respectively, to the proximal-most and distal-most visible points of the probe. This applies for all histological images which contain any portion of a probe track.
- Press **'c'** to delete probe points.
- Press **'w'** to visualize the probe in 2D (Figure 5B).
- Press **'e'** to save the histology with marked probe points and proceed to register on the next slice. The slices will appear in order according their name.
- In cases with multiple histological images, the succeeding slice can be made to appear bu using the following line of code in the console:

  ```
  register_probes.plot_hist()
  ```

  And continue the same procedure of marking and saving is to be followed for each histological image.
- For more options (e.g. **'g'** - to activate gridlines, or **'n'** - to add a new probe) follow the instructions on the console.

NOTE: Be sure that one of the windows for the Atlas viewer, Histology viewer or Combined histology + atlas viewer is selected when using a keyboard, otherwise the the software will not respond to keystrokes.

NOTE: The background colour of the image or the atlas can be changed by doing the following: go to line 21 (Figure 6) in the `index_tracker.py` file and change the 'cmap' colour to either white or black depending on your image.



```
self.im = ax.imshow(self.X[:, self.ind, :].T, origin="lower", extent=[0, 512*pixdim, 0, 512*pixdim], cmap='gist_yarg')
#cmap='gist_yarg' if white background wanted
#cmap='gray' for a gray or black background
```

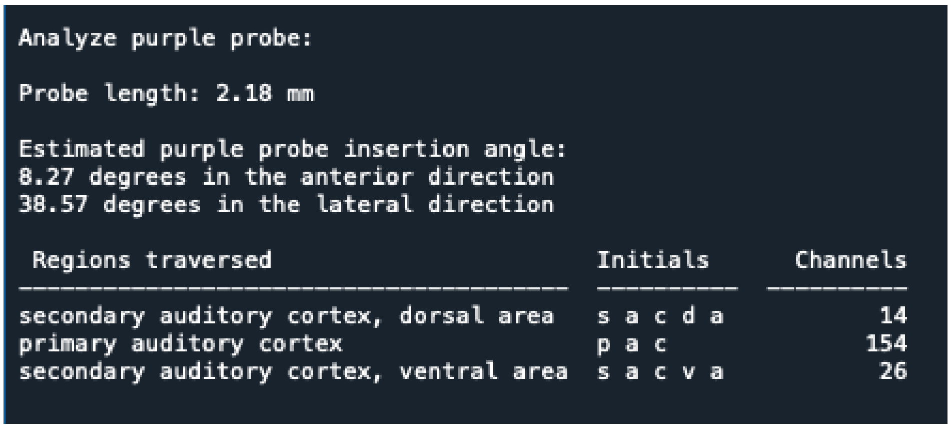
Figure 6. Code lines in which background colours can be changed.

3.3.3 Generating data tables and visualizing probe tracks in 2D or 3D

Once the probe track has been defined and saved, a data table can be generated containing detailed physical data regarding the probe angle, the location and number of electrodes in the various regions the probe traversed (Figure 7). In addition, the probe can be rendered in a 2D annotated slice or in a 3D brain volume. Both the data table and visualizations are generated by running the following lines of code individually:

```
vis3dres = vis_registered_probes(atlas,probe_folder ='path')
vis3dres.vis3d()
vis3dres.vis2d()
```

In the processed folder on your 'path' you will then find a `.txt file` containing the details for the probe you have currently visualized (Figure 7).



```
Analyze purple probe:

Probe length: 2.18 mm

Estimated purple probe insertion angle:
8.27 degrees in the anterior direction
38.57 degrees in the lateral direction

 Regions traversed                      Initials      Channels
--------------------------------------  ----------  ----------
secondary auditory cortex, dorsal area  s a c d a           14
primary auditory cortex                 p a c              154
secondary auditory cortex, ventral area s a c v a           26
```

Figure 7. Data table listing physical parameters for the placement of the probe as well the estimated number of recording channels in each sub-region through which the probe passed.

In addition, 2D and 3D visualizations showing the reconstructed probe are generated and can be found in the `probe_folder` that is created in TRACER by the user before starting the probe reconstruction (Figure 8). At present, anatomical annotations from the Waxholm Space atlas are applied only in 2D slices.

The process can be repeated many times, but the console should be closed and a new one opened each time the script is run.

3.4 Visualization of viral expression or other anatomical features in 3D

TRACER can be used to visualize virus expression or any 2D or 3D regions of interest in the brain, including tracer injections, labeled fibers or lesions. We use the example of virus expression here, but the procedure is the same for any type of anatomical reconstruction.

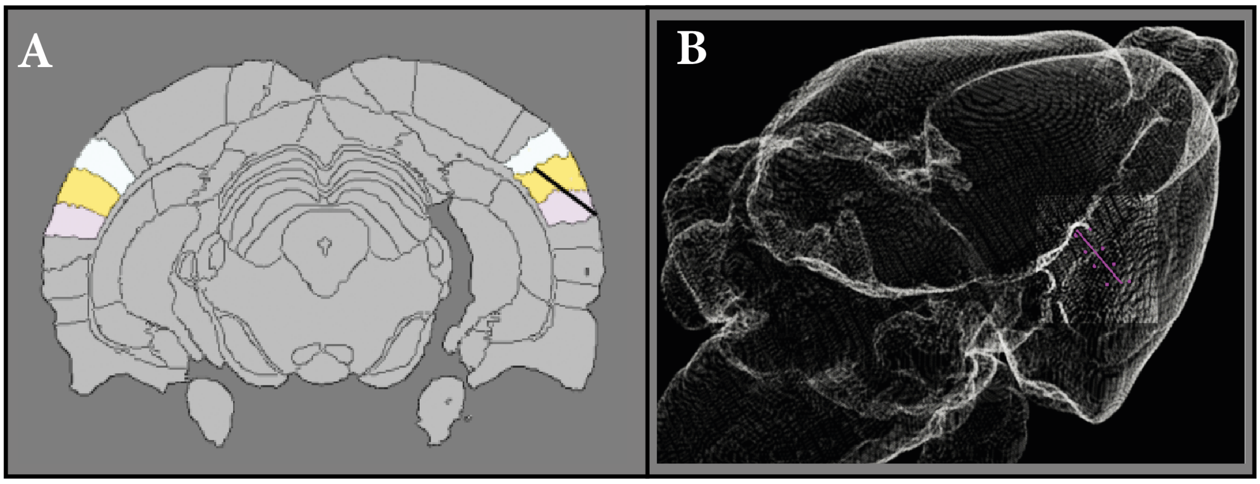


Figure 8. **A**. Reconstruction of the distal 2mm of a Neuropixels probe traversing primary and secondary auditory cortex. **B**: 3D brain volume showing the reconstructed electrode passing through the right auditory cortex.

3.4.1 Navigating the atlas and reconstructing viral expression in 3D

As with recording probe reconstructions, viral volume reconstructions use images that have been pre-processed for optimal contrast and brightness. The pre-processing script is the same as the one used for probe track reconstructions, and is executed by the command:

```
pre-process_hist = pre-process_histology(histology_folder='path')
```

Be sure to have the correct paths to pre-processed images. More details on the pre-processing step can be found above in section **3.3.1**.

After pre-processing, the user selects the Waxholm atlas slices which correspond to those containing virus expression, after which virus expression is registered by running the script `virus_registration.py`. The entire process is carried out by following the steps below:

- Always load the atlas first, then run the following line of code to start registering virus expression:

  ```
  register_virus = VirusRegistration(atlas,processedhistfolder='path')
  ```

- Add the path:

  `processed_histology_folder = Path('/Users/.../histology/processed')`

- Two windows will appear on the screen; one containing slices from the Waxholm atlas and the other with the image of the user's tissue. Begin the process of overlaying atlas and user slices by clicking **'t'**, which activates anchoring points for the two images.

- Choose at-least 4 points in the same order for both the atlas and histological slices. The anchor points are numbered to help keep the same ordering of the points.

- Press **'h'** to match the histological section onto the atlas (shown above in (Figures 4A and B).

- Press **'a'** to visualize the regional boundaries of the Waxholm atlas (see Figure 5A).

- After pressing **'h'** and **'a'** the user has the option to fill in the Waxholm delineations by pressing **'v'**, or to start marking the perimeter around the viral expression by pressing **'r'**.

- click points to delimit the region containing the virus.

- Press **'c'** to delete clicked points.
- Press **'e'** to save when finished, and add a name for the file.
- If you have multiple slices to mark, once the current section is saved and closed, the next slice appears on the desktop automatically. Follow the same marking procedure as above for each histology image in the series.
- For more options (e.g. press **'d'** – to delete the most recent transform point, or **'w'** – for a detailed analysis of the perimeter of virus expression) follow the instructions on the console.

NOTE: Be sure that one of the windows for the Atlas viewer, Histology viewer or Combined histology + atlas viewer is selected when using a keyboard, otherwise the the software will not respond to keystrokes.

3.4.2 Visualizing virus in the 3D atlas

Now that the relevant sections have been marked and saved, they can be rendered in 2D or combined in a 3D Waxholm brain volume by running the following lines of code individually:

```
vis3d_virus = vis_registered_virus(atlas, probe_folder)
vis3d_virus.vis2d() #generates annotated 2D slice showing where the
    virus is expressed
vis3d_virus.vis3d() #generates 3D brain volume with virus expression
```

The `vis3d_virus.vis2d()` script also provides detailed information about the anatomical coordinates and regions showing viral expression, which is saved in a `info.txt` file. Figure 9 shows an example of a 3D viral reconstruction. Close the console every time after this script is run.

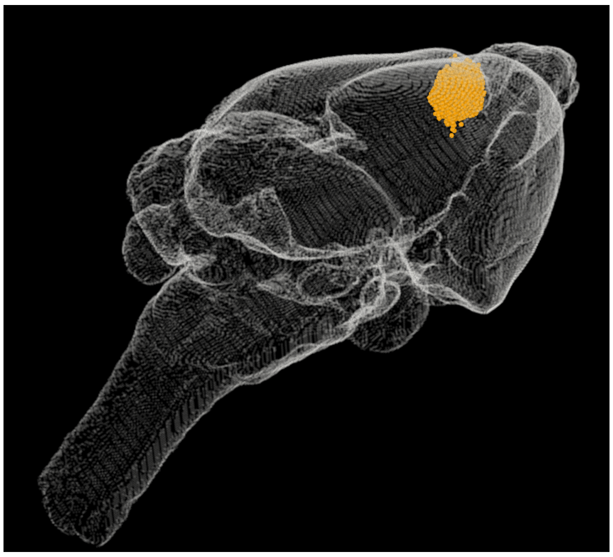


Figure 9. 3D brain volume showing the spatial extent of viral expression in rat frontal cortex.