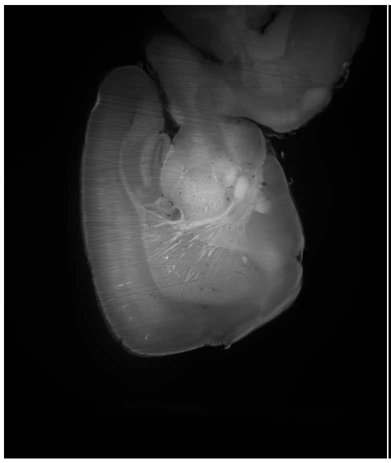
DeepCount Protocol

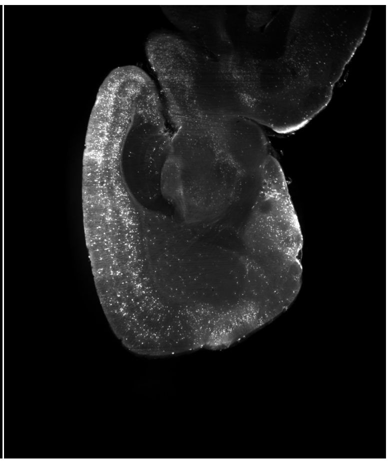
Step 1: Imaging

In our protocol, we image TRAPed cells in the 640nm channel and autofluorescence in the 488nm channel. This generates two stacks of individual TIF images separated into two folders, one for each channel (examples below). We will be using the 640nm channel to extract the cells and the 488nm channel to align the brains to an atlas, which allows automated quantification by brain region.

488nm (autofluorescence)



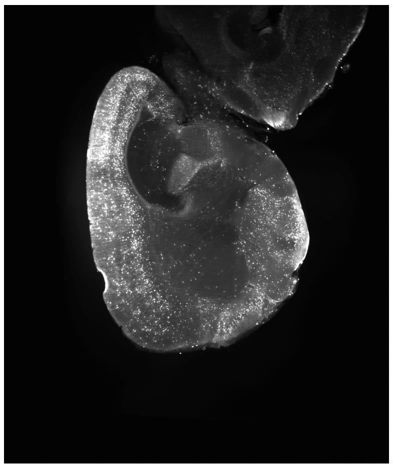
640nm (cells)



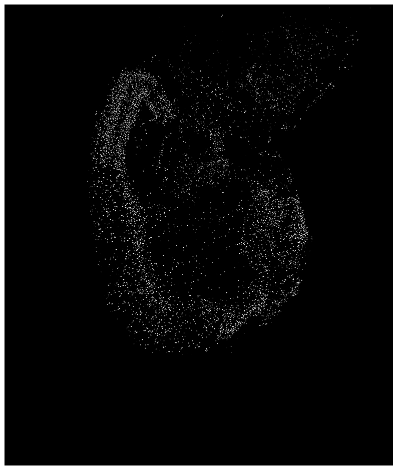
Step 2: TRAILMAP

TRAILMAP is the Deep-Learning pipeline used to convert the raw 640nm image stack to an image stack of the same size containing a map of the probability that each pixel contains a cell. Pixel values of 1 indicate high likelihood that a cell is contained in that region, and pixel values of 0 indicate low likelihood. This probability is calculated using a 3D convolutional network that has been trained to recognize cells.

640nm (cells)



Probability map (TRAILMAP output)



a. Install TRAILMAP & install necessary dependencies (if this is already done skip to part b)
    i. Clone GitHub repository to drive
    ii. Open Anaconda Prompt (install Anaconda if not already installed)
    iii. Create an environment for TRAILMAP and install dependencies using the following line:
    ```
    conda create -n trailmap_env tensorflow-gpu=2.1 opencv=3.4.2 pillow=7.0.0
    ```
    *for troubleshooting see readme on github
b. Select model to use for cell segmentation
    i. From the GitHub repository, open segment_brain_batch.py in a python or text editor and in line 18 change the path to the location of the model weights you would like to use
c. Activate TRAILMAP environment & enter TRAILMAP directory
    i. Open Anaconda prompt
    ii. Activate environment using the following line:
    ```
    conda activate trailmap_env
    ```
    iii. Enter directory using the following, using the actual directory in place of the one shown:
    cd *C:/Users/Michael/Documents/TRAILMAP*
    **if you are on the D drive you will need to type D: and click enter
d. Run TRAILMAP inference step on the 640nm channel of the brain of interest
    i. Enter the following line, replacing input_folder1+ with the directories for the brain(s):
    python segment_brain_batch.py *input_folder1 input_folder2 input_folder3*
    ii. This outputs the cell probability map to the same directory as the original folder, but with “seg-” added to the beginning of the folder name.

Step 3: Scale brain to a 10µm space and convert to 8-bit using ImageJ

From this point forward, we will want our brains in a 10µm space (where each voxel is 10µmx10µmx10µm), as this is the resolution of the atlas we use for registration. We perform this downscaling step in ImageJ using values calculated based on our imaging parameters, and if you would like to do this in batch we have provided a macro for this purpose. In addition, this macro converts the files to 8bit as opposed to 16- or 32- bit, as 8-bit images are best compatible with future steps. If you would like to do this manually, use the Image->Scale and Image->Type functions in ImageJ, but the instructions below are for the use of a macro.

These steps must be performed on both the 488nm raw channel and the probability map from trailmap, as these are the two images that will feed into the next step of the pipeline.

a. Open ImageJ
b. Click “plugins” → “macros” → “edit” and open the macro file for brain scaling (macro 1)

    i. *** macros located in the directory: Whole Brain > CellCounting > 'ImageJ Macro…'*

c. In the first line, add the path for the first **.tif file** of each brain you want to analyze to the array, separating them by commas
    i. You will want to scale both the TRAILMAP output and the 488nm channel for each brain.
    ii. NOTE: macros are very finicky; you may need to change all the \ in the directories to \\
d. Run the macro using ctrl-R
    i. *** if error: image not open, open virtual stacks and then run macro*
e. This will output the brain to a separate folder in the same directory, with "_scaled" on the end

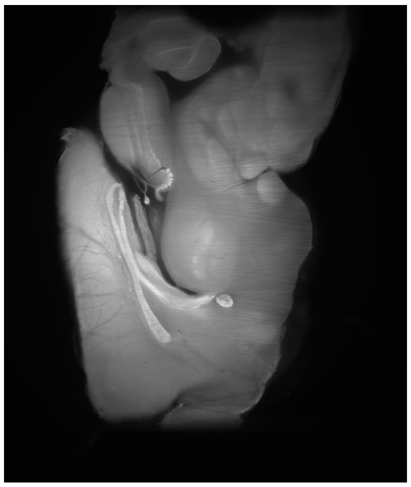
Step 3b (optional step to improve alignment): Manually rotate each brain in ImageJ

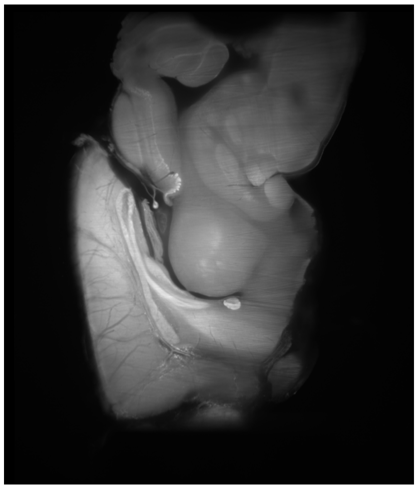
We have found that elastix, the program used to align the brain with an atlas, particularly struggles with alignment of brains that require large degrees of rotation around the X, Y, or Z axis. To account for this, we incorporated a step where we manually rotate the brains to be flush with the atlas using an ImageJ plugin called TransformJ. In our imaging protocol, we image slightly past the midline, which allows visualization of blood vessels along the midline in the autofluorescence channel. Our goal in this rotation is to make those midline veins all appear in the same z plane, thus making the brain flush with the atlas.

a. Install ImageScience's TransformJ (https://imagescience.org/meijering/software/transformj/)
    i. In ImageJ, go to help->update
    ii. Click "add update site," check "ImageScience," and close. After restarting ImageJ, TransformJ should be installed.
b. Open 10um scaled 488nm image in ImageJ
c. Assess rotation of image by scrolling to the midline and checking whether all blood vessels are visible in the same z plane or if they are rotated. If they are rotated, assess which direction the image needs to be rotated to flatten them into a single plan.
d. Click plugins->ImageScience->TransformJ->TransformJ Rotate
e. Select approximate degrees of rotation for each axis.
    i. Positive rotation in the X axis brings top of screen toward you
    ii. Positive rotation in the Y axis brings right side of screen towards you
    iii. Positive rotation in the Z axis brings top of screen to the right (clockwise)
f. Click "ok" to run the rotation. Check if the blood vessels are now aligned. If not, repeat the process with different angles of rotation until they all appear in the same plane. **Record the angles of rotation used for the final image.**
g. Make a substack of the rotated image in the z plane (image->stacks->tools->create substack) to crop out black borders that do not contain brain tissue. **Record the slices that were included in the substack.**
h. Save the image as "10umrc.tif" (10um image, rotated and cropped).

i. Open the 10um segmented 640nm image in ImageJ and apply the **exact same transformation**.

Midline blood vessels before alignment | Midline blood vessels after manual alignment





Step 4: Register brain autofluorescence channel (488nm) to atlas using Elastix

Alignment/registration is the process of taking your raw data and warping it such that it is aligned with a standardized reference atlas. This is performed in a command-line-based program called Elastix (https://elastix.lumc.nl/). The first step involves aligning the autofluorescence channel to an atlas image, and the next step involves applying the exact same transformation to the channel containing the label of interest, which is done in Transformix (a subsection of the Elastix program). Once your brain is aligned to an atlas, you then know which groups of pixels correspond to each brain region, and this information can be used for quantification and visualization.

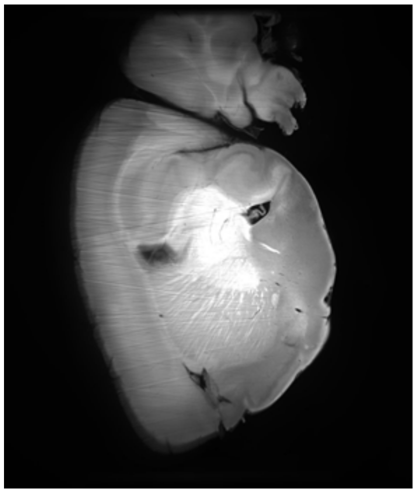
a. Open Anaconda Prompt. If elastix is not installed, follow the directions in the elastix manual.
b. Ensure that you have the 10um cropped atlas and the affine and bspline parameter files
c. Run the following command, replacing the placeholders with the correct path, using affine as parameter 1 and bspline as parameter 2

```
elastix -f directory/fixedImage.ext -m directory/movingImage.ext -out outputDirectory -p directory/parameterFile1.txt -p directory/parameterFile2.txt
```

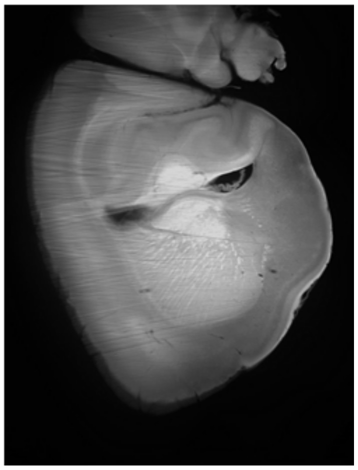
   i. The fixed image is the cropped 10um atlas, and the moving image is the scaled, rotated, and cropped 488nm channel. The “moving” image will be registered to match the fixed image.

d. This will output a registered brain as well as some new parameter files to the specified directory.
e. Verify alignment by opening “result.mhd” alongside the average_template file and checking to see that the aligned 488nm channel looks the same as the average template.

Pre-alignment



Post-alignment



Step 5: Register the TRAILMAP output (probability map) to the atlas using Transformix

As previously mentioned, once our autofluorescence channel is aligned to the atlas, we must then align the probability map to the atlas as well. We do this in **Transformix**, a subsection of the Elastix program, by applying the exact same transformation to the probability map as we did the autofluorescence channel.

a. Locate the "TransformParameters.1.txt" file in the elastix output directory
b. Within Anaconda prompt, run the following command, replacing the placeholders with the correct path
   transformix -in *directory/inputImage.ext* -out *outputDirectory* -tp *directory/TransformParameters.1.txt*
    i. The input image is the scaled, rotated, and cropped TRAILMAP output image file (e.g. .tif).
c. This will output the transformed cell probability map as "result.mhd"

Step 6: Convert .mhd file to 8-bit .tif file using ImageJ

f. Open ImageJ
g. Click "plugins" → "macros" → "edit" and open the macro file for this step (macro 2)
h. In the first line, add the path to the **.mhd file** from the transformix output of each brain to the array, separating them by commas
    i. NOTE: macros are very finicky; you may need to change all the \ in the directories to \\
i. Run the macro using ctrl-R
j. This will output the brain as FP.tif to the same folder as the .mhd file

Step 7: Reduce cells to single pixels using MATLAB

A critical step in the process of whole-brain cell counting is ensuring that all labeled cells, regardless of differences in appearance, are counted as one single cell. The approach we have elected to use for this is reduction of each individual cell to one single pixel, which will be followed by counting the number of labeled pixels in each region. We do this by calculating local 3D maxima, as this is well-positioned to parse out cells that are right next to each other as well as distant cells. To avoid over-counting large cells, we then reduce any connected maxima to single pixels by analyzing whether pixels are connected to each other. The output of this process is a map of the brain where pixel values of 1 indicate a cell and pixel values of 0 indicate background.

a. Open maxpoints.m in MATLAB
b. Change the directory in “folders” in line 2 to the folder containing FP.tif
c. Click run. This should output maxpoints.tif, where each cell is represented by a single pixel

Step 8: Quantify the Cell Counts by region

Now that we have a file (maxpoints.tif) that contains the location of each individual cell as single pixels, we can simply count the number of pixels with an intensity of 1 in each brain region, and this will tell us the number of labeled cells in that region. We can normalize this by region volume, total number of cells, neither, or both. The output of this step is an excel file with the cell counts normalized in several different ways, including the raw cell counts per region.

a. Open pointstocounts.m in MATLAB
b. Change “regions” and “annotated” in lines 3 and 5 to match the file paths of the atlas .nrrd file and the annotation .csv file, respectively.
c. Change “folders” in line 19 to the directory containing maxpoints.tif from the previous step
d. Click run. This will output CellCounts.xlsx to the same directory as FP.tif
e. If you would like to create a heatmap, use heatmap.m and add the paths to cellCounts.xlsx and the aba_ontology.csv file (to ensure the heatmap is generated in ABA order).

NOTE: This guide isn’t yet complete for creating a cropped/collapsed atlas.