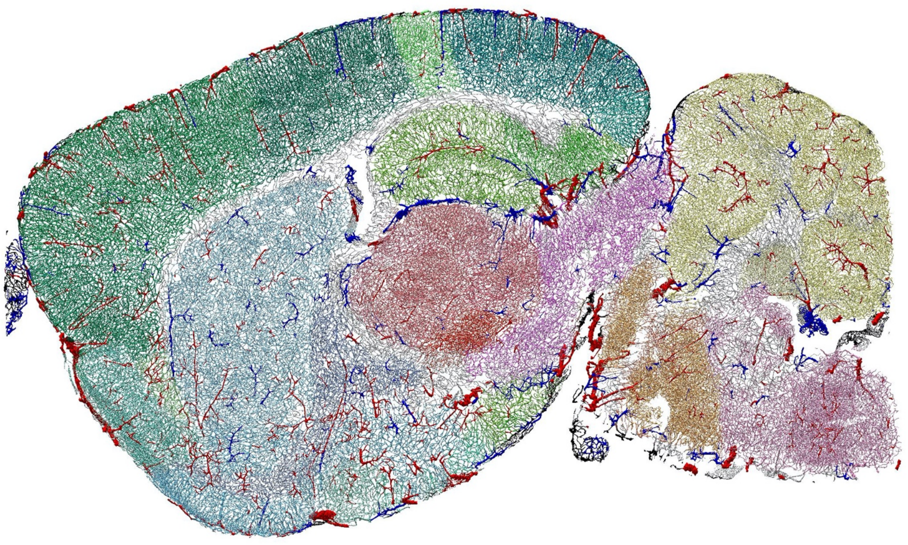


IDISCO AND TUBEMAP

Practical handbook for tissue clearing, light sheet microscopy and analysis of cerebral vascular datasets

This document provides users with step by step instruction to generate cerebral vascular datasets. It can also be used as a general introduction to tissue clearing with immunolabeling and light sheet imaging.

https://www.idisco.info

Version 0.9

Sample preparation with iDISCO+

This document is a comprehensive guide to implement an entire pipeline for vascular analysis of the mouse brain with iDISCO+, the LaVision Ultramicroscope II and TubeMap. Users are free to experiment with their own antibodies, different models of light sheet microscopes, or different analysis softwares. This guide can also be used as an introduction to the general frameworks guiding the combined use of tissue clearing, immunolabeling, light sheet microscopy and image analysis for other applications.

1. Brain fixation and dissection

Due to the long duration of the subsequent staining protocol (around 3 weeks) tissue fixation is required but must be quick enough to prevent the loss of antigenicity and high tissue background. In the case of the vasculature, the fixation method will change the apparent diameter of large vessels, especially veins. We tested 2 alternative protocols that we optimized for vascular analysis combined with immunolabeling: intracardiac PFA perfusion and a blood retention method.

· **Required material/equipment**:

- Chemical hood
- Peristaltic or gravity perfusion pump
- Pentobarbital-Euthasol Vet (Produlab Pharma B.V, Neederlands)
- Dissection tools (we use 26G perfusion needle, forceps, 2x precision dissection tweezers, regular tweezers, small and large scissors)
- 5mL Eppendorf tubes or equivalent
- Cold PBS (Phosphate buffer saline 1X pH7.4) [supplement with Sodium Azide 0.01%]
- Paraformaldehyde 4% in PBS (Electron Microscopy Sciences, USA) [Commercial solution 32%]

1.1 Intracardiac perfusion with PFA (recommended method)

Intracardiac PFA perfusion provides better immunostaining results than drop-fixation (higher signal and less antibody "spotting" in the tissue). This fixation method should be used by default if compatible with the experimental conditions.

1.1.1 Prepare the perfusion hood with all the tools and solutions needed. Load the peristaltic pump with cold PBS. Set the perfusion rate to a low flow rate (about 10mL/min). ⚠Low flow rate is important to prevent dilation of the brain ventricles and internal damages to the brain morphology.

1.1.2 Kill the mouse with an overdose of pentobarbital (intraperitoneal injection of 200mg pentobarbital for 100g of animal; that is 0,5mL of Euthasol/100g in case of using Euthasol Vet).

1.1.3 Once the animal is unresponsive, quickly open the thoracic cavity, expose the heart and cut the right atrium. Poke a small hole in the left ventricle just enough to introduce the tip of the perfusion needle and initiate the infusion of cold PBS (10mL should be enough to remove most of the blood from the circulatory system).

1.1.4 Switch to 4% PFA and infuse another 30mL.

1.1.5 For the dissection, use thin dissection tweezers, to carefully remove the skull starting from the occipital bone and moving forward to the rostral regions. Once the occipital, interparietal, parietal and frontal bones are removed, drop the brain in a 5mL Eppendorf tube with 4% PFA. ⚠ It is important to avoid touching the brain during the dissection: any scratch at the surface of the brain will impact the reconstructions and alignments during analysis. The use of scissors during the dissection is not recommended after the opening of the scalp.

1.1.6 Keep the brains in PBS - PFA 4% for 3 hours at room temperature.

1.1.7 Wash in PBS [Sodium Azide 0.01%] 3 times until the next day and keep them at 4ºC until further processing.

1.2 Alternative protocol: Blood retention method

As an alternative to brain perfusion, we also developed a method for vascular staining based on the immunolabeling of circulating IgG present in the blood. This method is very effective in younger animals, up to 6 weeks old. Older animals can also be prepared this way but may require longer fixation times. Bleeding during the dissection would deplete the vessel filling. Blood staining is a method of choice when the perfusion of the animal is impossible.

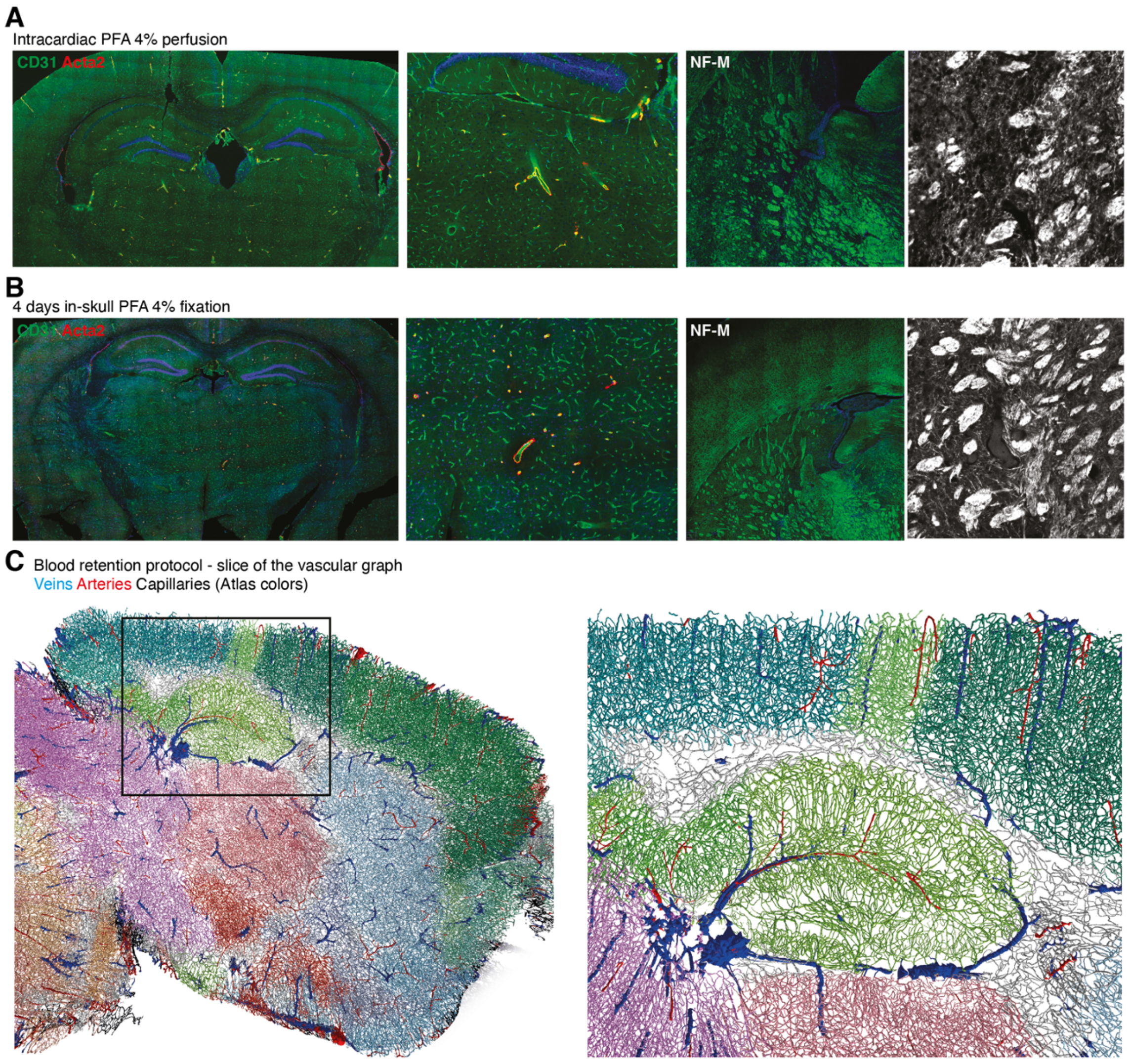

A
Intracardiac PFA 4% perfusion
CD31 Acta2
NF-M
B
4 days in-skull PFA 4% fixation
CD31 Acta2
NF-M
C
Blood retention protocol - slice of the vascular graph
Veins Arteries Capillaries (Atlas colors)

Figure 1 Comparison of intracardiac perfusion protocol with drop-fixation. ***A*** *Perfused brain, sectioned and immunolabeled for CD31, Acta2 and Neurofilament M.* ***B*** *Brain prepared with the blood retention protocol: the head was immersed intact in the fixative for 4 days before dissection. The brain was then sectioned and immunolabeled for CD31, Acta2 and NF-M. The immunoreactivity is maintained and tissue structure preserved, but the signal is quenched at the surface, as a consequence of the longer fixation time* ***C*** *Vascular graph obtained from a whole brain reconstruction done on drop-fixed sample. The immunolabeling done was rat anti-podocalyxin, donkey anti-mouse IgG and rabbit anti-Acta2. The reconstruction is nearly identical to the one obtained on perfused samples, except for the much larger vein radii.*

Procedure:

1.2.1 Kill the mouse with a CO_2 rising gradient.

1.2.2 Very carefully, remove the skin around the neck and dorsal regions of the head. To reduce the risk of bleeding, use precision tweezers to pinch the skin and detach it from the underlying tissues. Simultaneously, with the other hand use surgical scissors to carefully cut the skin in small pieces (cutting large pieces of skin at once increases the risk of bleeding).

Injuries in major vessels such as jugular veins or carotid arteries completely compromise the quality of the sample at this point.

1.2.3 Immerse the whole animal PFA 4% and keep at 4ºC for 24 hours.

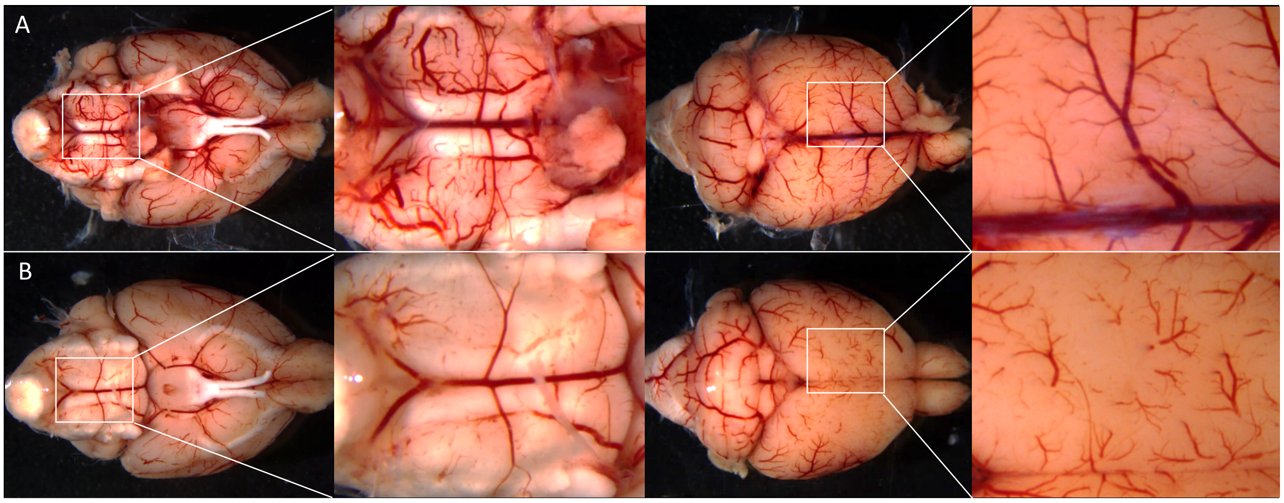
1.2.4 Use a shaving razor or surgical scissors to carefully cut the snout of the animal avoiding major compression to the skull. Then position the animal in a flat surface and remove the interparietal bone. With this step two "windows" are opened in the skull to facilitate PFA diffusion into the brain.

1.2.5 Place the whole animal back in the same 4% PFA solution for further fixation. Keep it at 4ºC for 48 hours.

1.2.6 In the third session (day 4 after sacrificing the animal), dissect the brain as described in step 1.1.5 Use magnifying glasses or a dissection microscope to take special care and avoid any bleeding.

Blood retention should be visible in large vessels and the brain parenchyma should have a pink homogeneous color (Fig. 1A). Brains with visible problems at this step of the protocol should be excluded (Fig 1B)

Fig 1. Dorsal and ventral view of a brain after the blood retention protocol. The upper row (**A**) shows an example of good



blood retention, while in the lower row (**B**) part of the blood was not retained.

2. iDISCO$^+$ processing

· **Required material/equipment**:

- Chemical hood
- Rotator/wheel for tube agitation
- Oven at 37ºC with agitation
- 5mL Eppendorf tubes (or any equivalent recipient resistant to organic solvents, glass is also suitable, but make sure the caps are resistant)
- Sodium Azide (Sigma-Aldrich, Germany). Prepare a stock at 10% w/v in distilled water and supplement all the buffers with a dilution 1:1000 of the stock (the final concentration of Sodium Azide in all your buffers should be 0.01%).
- PBS (Phosphate buffer saline 1X pH7.4) $^{\text{[Sodium Azide 0.01\%]}}$
- PBST (PBS supplemented with 0.2% Triton X-100) $^{\text{[Sodium Azide 0.01\%]}}$
- Permeabilization Solution [20% dimethyl sulfoxide (Sigma-Aldrich, USA), 5,75% Glycine (Sigma-Aldrich, USA) in PBST] $^{\text{[Sodium Azide 0.01\%]}}$
- PBST-gelatin [PBST supplemented with 0.2% w/v of porcine gelatin (Sigma-Aldrich, USA)] $^{\text{[Sodium Azide 0.01\%]}}$
- Methanol 100% (Sigma-Aldrich, France)
- Methanol dilutions in water. Prepare stock dilutions with Methanol 20%, 40%, 60% and 80%.
- Hydrogen Peroxide 30% (Sigma-Aldrich, Spain)
- Dichloromethane (Sigma-Aldrich, Germany)
- Primary antibodies [Acta2 (Abcam, Cat#ab5694); CD31 (R&D Systems, Cat#AF3628); Podocalyxin (R&D Systems, Cat#MAB1556)]
- Secondary antibodies [Donkey anti-Goat IgG, Alexa Fluor 647 (Thermo Fisher Scientific, Cat# A-21447; Chicken anti-Rat IgG, Alexa Fluor 647 (Thermo Fisher Scientific, Cat# A-21472); Donkey anti-Rabbit IgG, Alexa Fluor 555 (Thermo Fisher Scientific, Cat# A-31572)]
- Dibenzyl ether (Sigma-Aldrich, USA)

IMPORTANT: All the steps are done in the same 5mL Eppendorf tubes where the brain was preserved. Make sure to **completely fill the tubes** to the top reducing to the minimum the amount of air. The presence of air oxidizes the tissue affecting the final transparency, and therefore the subsequent imaging steps.

2.1Tissue preprocessing/permeabilization

2.1.1 Return the fixed brains (either from the regular or the blood retention method) to room temperature.

2.1.2 Dehydration in Methanol: incubate the brains in crescent concentrations of Methanol diluted in distilled water. For whole-brain samples the extend each wash for a minimum of 1 hour. Use Methanol 20%, 40%, 60%, 80% and Methanol 100%, followed by a second wash in Methanol 100% of 2 hours. During dehydration keep the tubes in agitation using a rotor (approx. 6 rpm) at room temperature.

2.1.3 For lipids extraction, incubate the samples in a mixture of Dichloromethane/Methanol in proportion 2:1 (Dichloromethane 66%-Methanol 33%). Keep the tubes in this solution overnight in agitation at room temperature.

2.1.4 Wash the samples twice in Methanol 100% (4 hours each wash).

2.1.5 To reduce the autofluorescence from the tissue and remaining blood, bleach the brains in 5% H_2O_2 in Methanol (dilute 1:5 the stock solution containing 30% H_2O_2). Keep the samples at 4ºC overnight without agitation.

2.1.6 Take your samples out of the fridge and let them recover room temperature.

2.1.7 Rehydrate the samples by incubating them in decrescent concentrations of Methanol diluted in distilled water. You can start directly with Methanol 60% (the bleaching solution is already a ~80% Methanol solution), then Methanol 40% and 20%. During rehydration keep your samples in agitation at room temperature.

2.1.8 Wash the samples in PBS$^{[\text{Sodium Azide } 0.01\%]}$ twice for 15 min.

2.1.9 Wash the samples 1 hour in PBST$^{[\text{Sodium Azide } 0.01\%]}$.

2.1.10 Incubate the samples in Permeabilization Solution$^{[\text{Sodium Azide } 0.01\%]}$ for 24 hours at 37ºC with agitation.

2.2 Immunostaining

2.2.1 Block by incubating the samples in PBST-gelatin$^{[\text{Sodium Azide } 0.01\%]}$ for 24h at 37ºC with agitation.

2.2.2 Incubation with primary antibodies. For standard staining use the following antibody mixture:

Rabbit anti-smooth muscle actin/Acta2 (Abcam, Cat#ab5694) – Dilution 1:1000
Goat anti-CD31/PCAM-1 (R&D Systems, Cat#AF3628) Dilution 1:1500
Rat anti-Podocalyxin (R&D Systems, Cat#MAB1556) Dilution 1:1000
PBST-gelatin$^{[\text{Sodium Azide } 0.01\%]}$

Incubate the samples with the primary antibodies for 10 days (whole brain) or 7 days (Hemisphere) at 37ºC with agitation

2.2.3 Wash the primary antibodies with PBST$^{[\text{Sodium Azide } 0.01\%]}$. Wash them twice for 1 hour each, and then overnight. Do all the washes in agitation at room temperature.

2.2.4 Incubate the samples with secondary antibodies. Use the following antibody mixture:

Donkey anti-Goat IgG, Alexa Fluor 647 (Thermo Fisher Scientific, Cat# A-21447) – Dilution 1:500
Chicken anti-Rat IgG, Alexa Fluor 647 (Thermo Fisher Scientific, Cat# A-21472) – Dilution 1:500
Donkey anti-Rabbit IgG, Alexa Fluor 555 (Thermo Fisher Scientific, Cat# A-31572) – Dilution 1:500
PBST-gelatin$^{[\text{Sodium Azide } 0.01\%]}$

Keep the samples with the secondary antibody for 7 days at 37ºC with agitation and protected from light.

2.2.3 Wash the secondary antibodies with PBST$^{[\text{Sodium Azide } 0.01\%]}$. Wash them twice for 1 hour each, and then overnight. Do all the washes in agitation at room temperature protecting the samples from light.

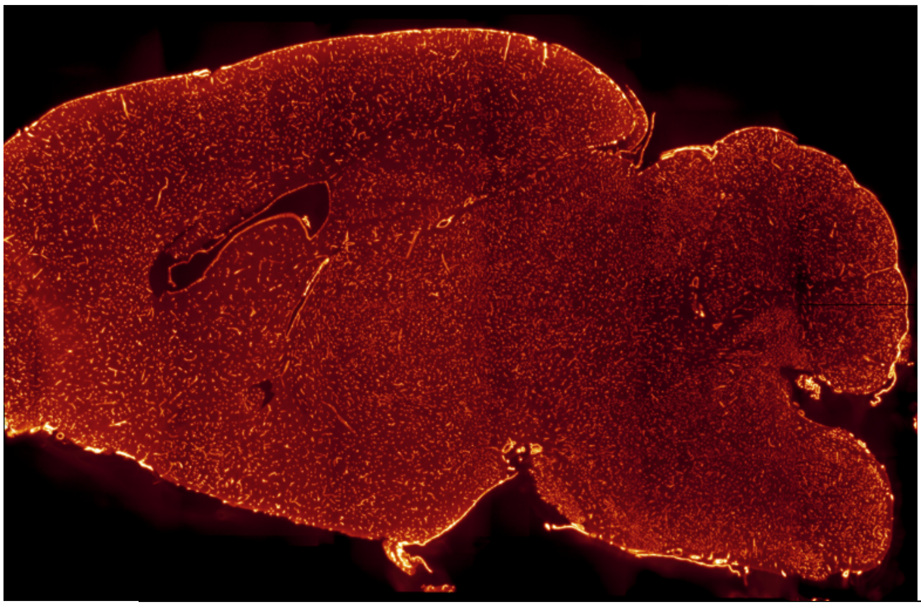
· Alternatives for vascular immunostaining

In some cases, the standard combination of antibodies may not be compatible with the specifics of the experimental design, of might be unavailable. We tested the following alternative antibodies:

***All-vessels* staining**

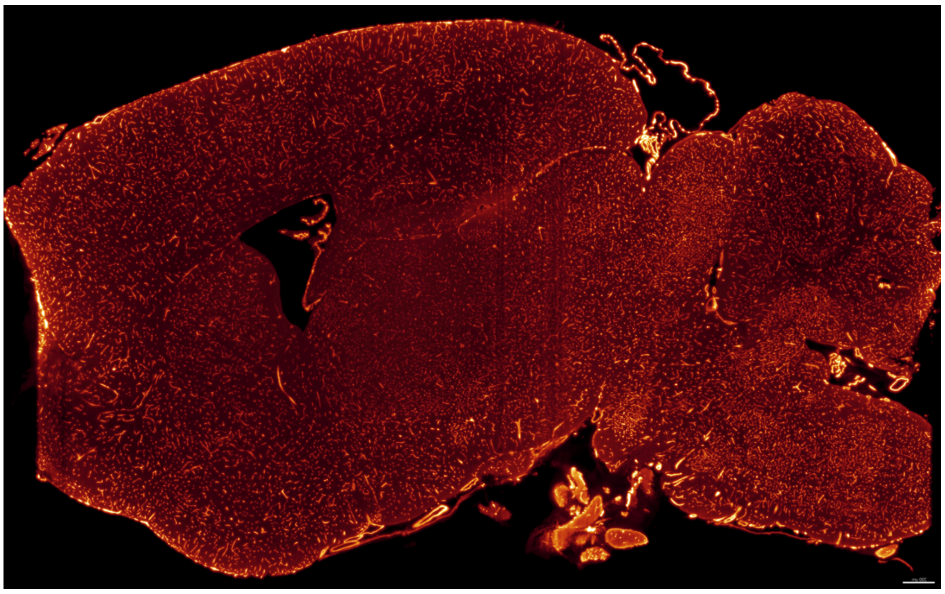
Combining multiple antibodies against different vascular epitopes ensures bright and complete staining of the capillary net. The default protocol uses two primary antibodies, a monoclonal Rat anti-Podocalyxin with a polyclonal Goat anti-CD31. However, the following combinations were also validated:

All vessels in Goat (Fig.2): Similar performances than the standard protocol.



Goat anti-CD31/PCAM-1 (R&D Systems, Cat#AF3628) Dilution 1:1500
Goat anti-Podocalyxin (R&D Systems, Cat#AF1556) – Dilution 1:1500

All vessels Rat/Rabbit/Goat (Fig.3): Better signal to noise, but less hosts flexibility for counterstain.



Goat anti-CD31/PCAM-1 (R&D Systems, Cat#AF3628) Dilution 1:1500
Rat anti-Podocalyxin (R&D Systems, Cat#MAB1556) – Dilution 1:1000
Rabbit anti-Collagen IV (Abcam, Cat#19808) – Dilution 1:1000

Arterial staining

Alternative antibodies were also tested to address arterial staining. Although all of them target smooth muscle cells to ensure the compatibility with TubeMap arterial detection parameters, two different antigens were tested: αSMA/Acta2 (Fig. 4A-B) and Transgelin/SM22 (Fig. 4C-D).

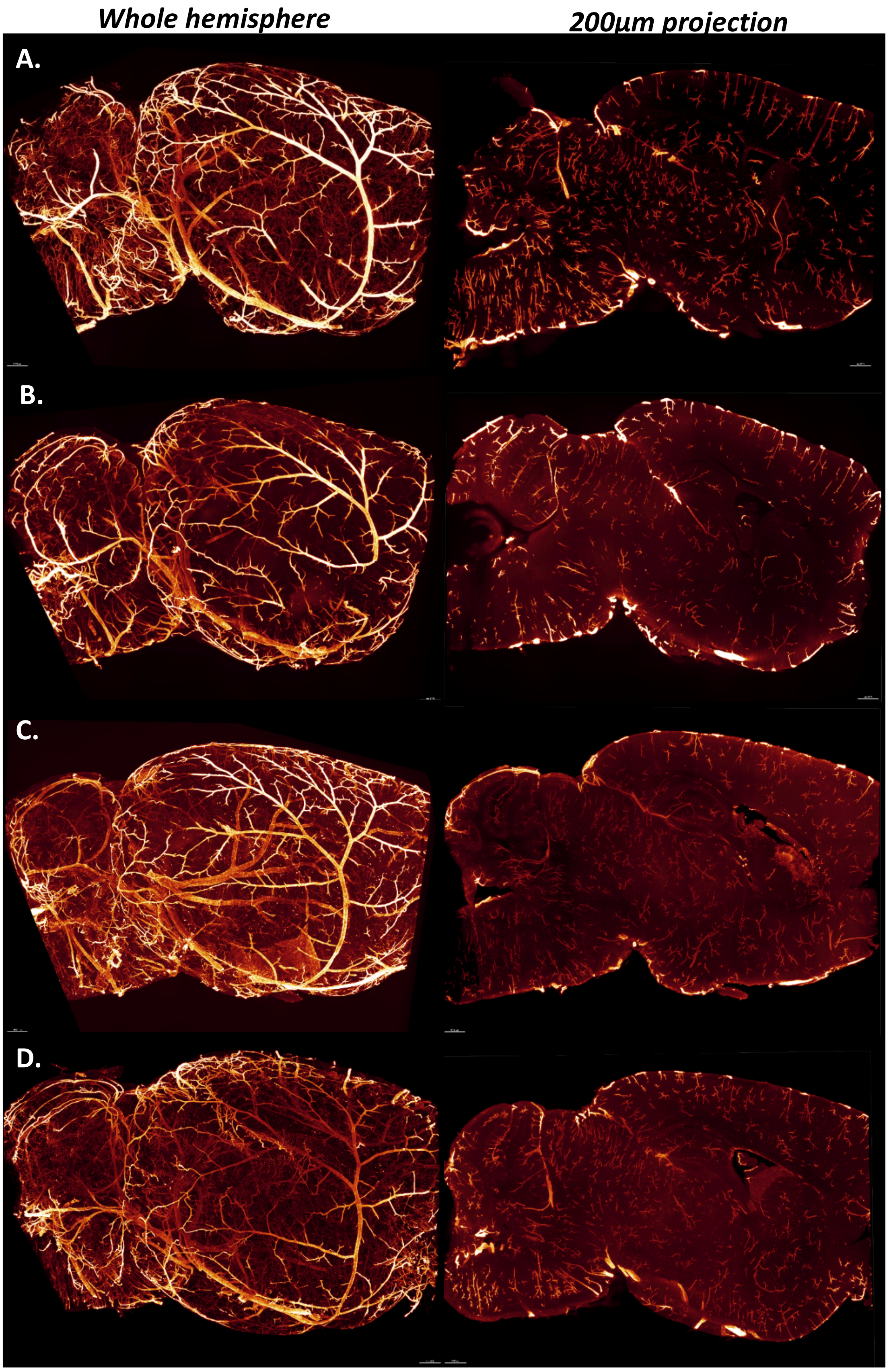


Fig. 4. Examples of four alternative antibodies for arterial staining. **A** - Rabbit anti-SM22/Transgelin antibody from Abcam (Cat#ab14106) diluted [1:1500]; **B**- Rabbit anti-SM22/Transgelin antibody from Proteintech (Cat#10493-1-AP) diluted [1:300]; **C** – Goat anti-αSMA/Acta2 antibody from Abcam (Cat#21027) diluted [1:1500]. **D** - Goat anti-αSMA/Acta2 antibody from Novus (Cat#NB300-978) diluted [1:1500].

2.3 Clearing

2.3.1 Dehydrate the samples as in step 2.1.2. but in this case extend the second wash in Methanol 100% overnight. Protect the samples from light.

2.3.2 Wash the samples in a mixture of Dichloromethane/Methanol in proportion 2:1 (Dichloromethane 66%-Methanol 33%) for 3 hours with agitation, at room temperature and protected from light.

2.3.4 Wash the samples twice in Dichloromethane 100% 15 minutes each. Keep the samples in agitation at room temperature and protected from light.

2.3.5 Transfer to Benzyl Ether for optical clearing (diffraction index equilibration). At this step the brain becomes transparent. The sample will have a pale-yellow hue (Fig. 5A). Browning indicates oxidation (Fig. 5B), which can be probably mitigated by making sure that tubes are always completely filled during the whole protocol, that PFA fixation times are not too long, and dehydration steps not too short. A milky appearance indicates a contamination of water in the sample, which can be caused by insufficient times in the dehydration solutions (Fig. 5C).

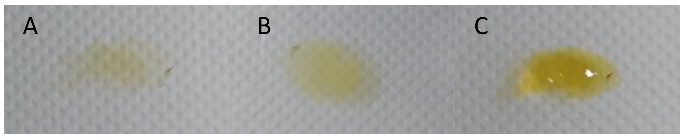


Fig. 5. Examples of a successfully processed brain (A), incomplete dehydration (B) and a slightly oxidized brain (C).

3. Image acquisition – Light sheet microscopy

· **Required material/equipment**[1]:

- LaVision Ultramicroscope II
- Laser lines: OBIS-561nm 100mW, OBIS-639nm 70mW with a 2nd generation LaVision beam combiner.
- Filters 525/50; 595/40 and 680/30.
- LVMI-Fluor 4X/O.3 WD6 LaVision Biotec objective.
- 1.3X LaVision objective (discontinued, use the MI-plan 1.1X 0.1NA instead)
- Isolation table or/and active vibration filtration device
- PC equipped with SSD drives to speed up the acquisition.
- Glass Petri dish
- Benzyl Ether

3.1 Montage

3.1.1 Position the sample in the holder. The midline of the brain should be parallel to the table, and the lateral side should face up, as shown in Fig. 6 You can use the olfactory bulbs to secure the rostral side of the brain, and a screw in the caudal region. The head of the screw should be positioned in the Vermis or in between the cerebellum and the brainstem. Finally, before positioning the holder in the microscope stage, make sure that the sample is secure, any movement during the acquisition will force to repeat the process.

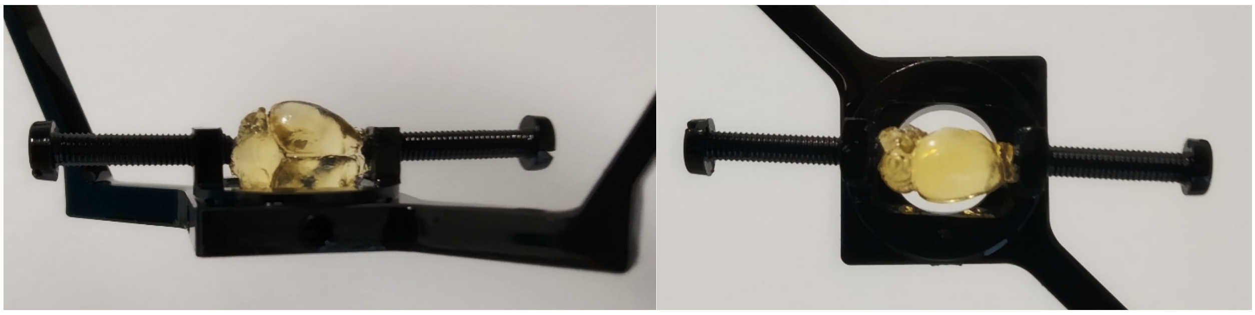


Fig. 6. Brain positioning in the lightsheet holder.

3.1.2 Put the holder into the pool previously filled with DBE (Ethyl Cinnamate) can be used alternatively in the chamber)

3.1.3 Mount the 1.3X objective

3.1.4 Initiate the camera, microscope, light source, computer and start Imspector.

3.1.5 Imspector should automatically open the following 5 windows: “Measurement Wizard”, UltraII, xyz-Table Z, xyz-Table Visual XY, Neo Life and a large previsualization window called “Document1.

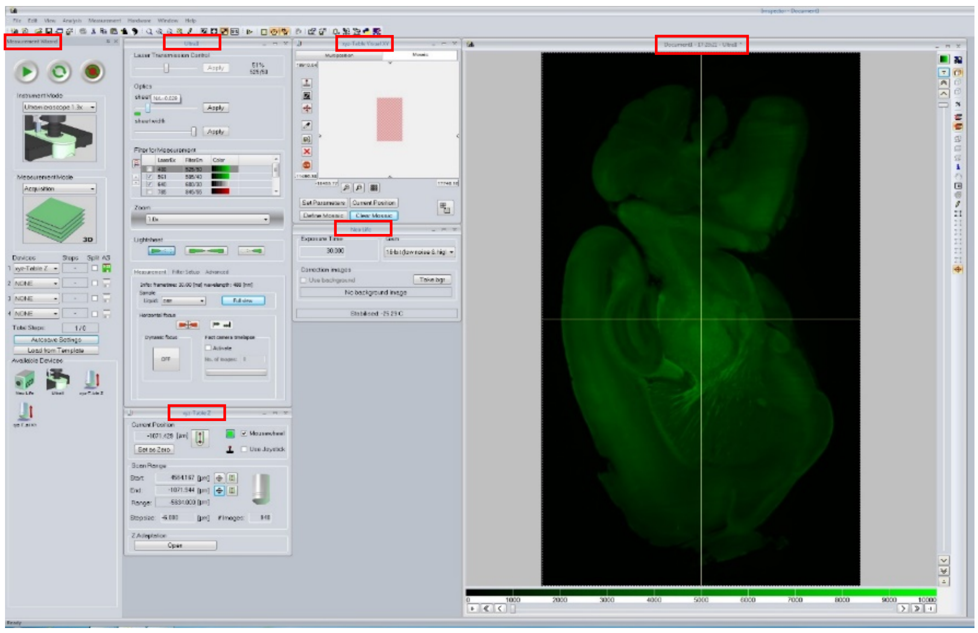


Fig 7. Inspector layout with and main panels

3.2 Autofluorescence imaging

3.2.1 The first scan is the autofluorescence of the sample at low magnification (1.3x objective) using blue light (488 excitation wavelength) to acquire the alignment reference channel. This information will be used by TubeMap to align the brain to a template.

3.2.2 Start with the “Measurement Wizard”: Select the 1.3x objective, Devices (xyz-Table Z to do a single Z stack), press the “save” button (“AS”) on the right to save each individual plane, and finally, set the folder and file names in “Autosave Settings”.

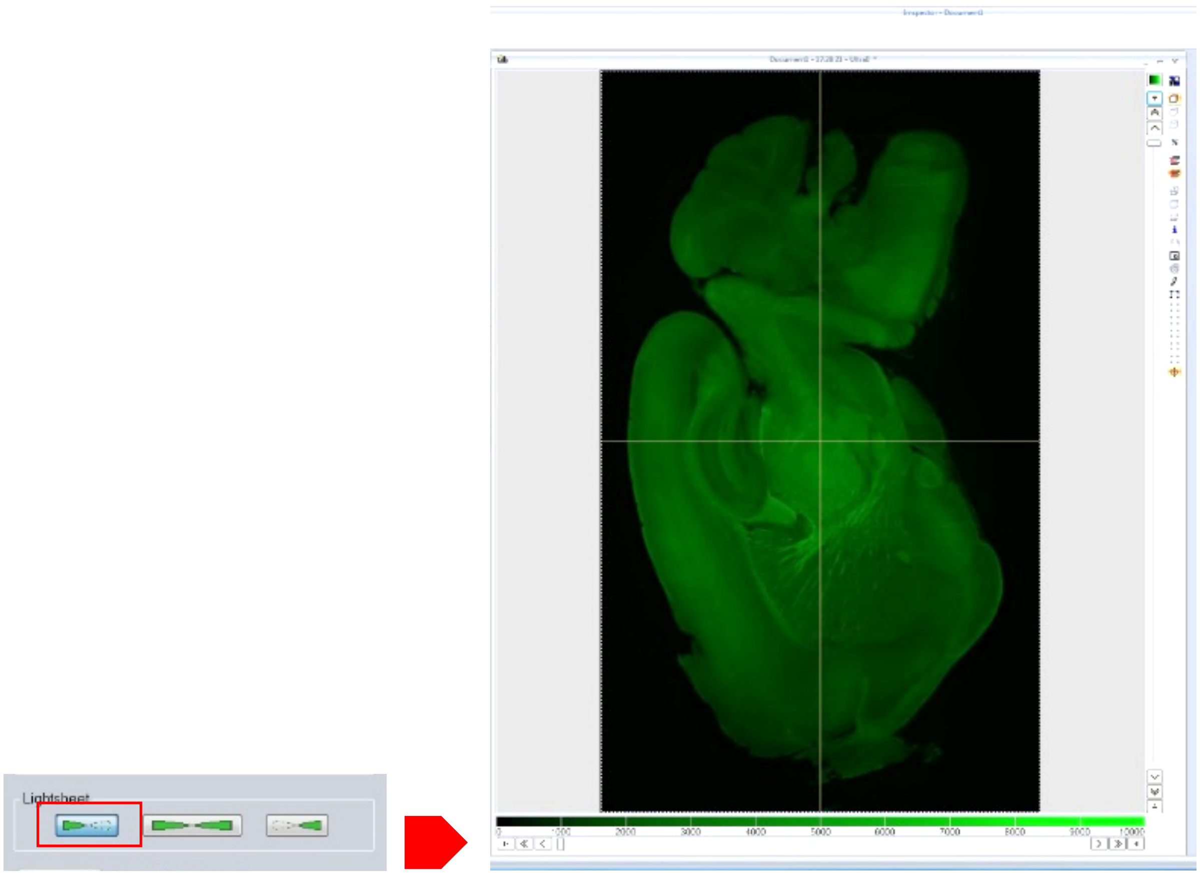


Fig 8. Selection of the illumination side. Three options are available: left, dual illumination or right. Select the side corresponding to the orientation of the cortex.

3.2.3 In the panel “UltraII”: set the “sheet NA” to 0.03 and the “sheet width” to 100%. Press apply in each of them to save the changes. In the filter panel click on the 488 laser/filter set. Keep the zoom at 1.0X and select the illumination side. To limit shadows, the light should enter the brain through the cortex. In this example the left laser was used.

3.2.4 To set the beginning and end of the stack, go in the live mode (Fig. 9A). While in the live mode, keep moving the sample up and down constantly to avoid bleaching. To boost the previsualization signal, shorten the dynamic range at the bottom of the preview panel (double click in the value, 65000 by default, and reduce it to 10000 for ex.) (Fig. 9B). Then set the beginning and the end of the acquisition (Fig. 9C-D). Set the step size to 6µm in this case (Fig. 9E). ⚠ It is important, when setting up parameters, to turn off the laser so it doesn’t continuously illuminate a single plane. The antibody signal should be resilient to prolonged illumination, but the autofluorescence signal is extremely sensitive to bleaching.

3.2.5 It is important to crop the field of view to limit the black areas around the brain. If there is a large volume of the image that is outside of the brain, it will decrease the quality of the alignment later. Go to the plane where the brain is the widest (usually at the level illustrated in Fig.9). Select F to draw a Region Of Interest (ROI) around the brain. Go to live mode and quickly check that the tissue stays inside the box throughout the range of the z movement. Then, crop with G.

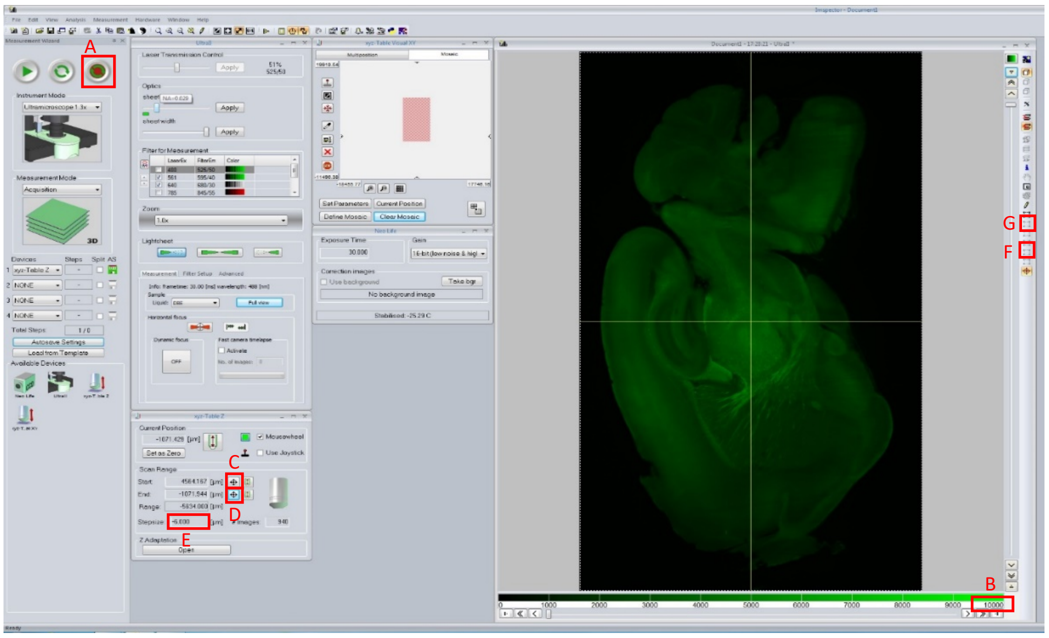


Fig. 9. Key parameters to set the acquisition of a single z-stack. A-Live mode, B-Dynamic range of the display, C-Top Position for the stack, D- Bottom Position E-Step size F ROI G Crop

3.2.5 Adjust the laser power and exposure time to ensure a good signal (get the darker regions of the brain to an intensity level of at least 1000). On our system, 30% laser power and 30ms acquisition time are usual for autofluorescence, but this may vary depending on the laser combiner of your system.

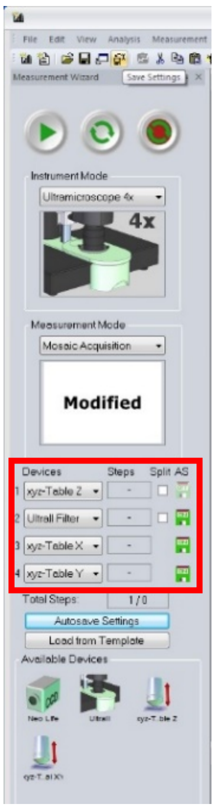
3.2.5 Launch the acquisition pressing the "play" button. This acquisition should take around 5-10min and generate a single folder containing the TIF files of the individual Z planes (around 1000 files for an adult brain). The size of the autofluorescence folder is about 5-7Gb.

3.3 Vasculature imaging

3.3.1 The second acquisition is the high-resolution scan of the vessels and arteries. This scan will be done with a 4x objective, as a mosaic of small fields of view (FOV) in which the plane of illumination is the thinnest and most homogeneous possible. Of note, we don't use horizontal dynamic focusing, at it is very slow on this microscope!

3.3.2 Change the lens to the 4x objective. It is possible to use the version of the microscope referred to as the "Zoom body" instead, but corrected dipping caps should be used. If available, the "Objectives" version of the microscope is strongly recommended.

3.3.3 In the "Measurement Wizard" panel: select the 4x objective and select in order the devices xyz-TableZ, UltraII Filter, xyz-TableX and xyz-TableY (the acquisition will be done by scanning first the Z planes, then changing the filter and reacquiring the same stack, and then moving in X or in Y if at the end of the row) (Fig. 10). Select the saving option for all the devices except xyz-Table Z. Finally, set the "Autosave Settings". ⚠ It is important to keep the "auto save" (green floppy disks icons) as indicated to insure the metadata and file structure are correct for the stitching later.



Instrument Mode: Ultramicroscope 4x

Measurement Mode: Mosaic Acquisition

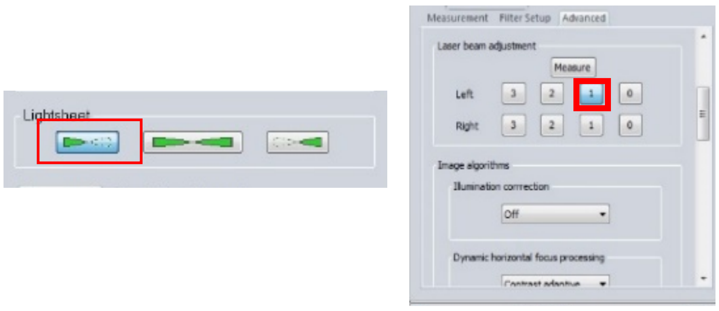
Modified

	Devices	Steps	Split	AS
1	xyz-Table Z	-	☐	
2	UltraII Filter	-	☐	
3	xyz-Table X	-		
4	xyz-Table Y	-		

Total Steps: 1 / 0

Fig. 10. Generic parameters for a mosaic acquisition. For subsequent successful analysis with TubeMap, pay special attention to the saving configuration (highlighted in red).

3.3.4 In the UltraII panel: change the sheet NA to the maximum (0.14). Select the 640nm filter set, and set the laser with at 100%. Press "Apply" to save the changes. Repeat with the 561nm set. Select both channels for acquisition in the tick boxes, make sure other filters are unselected. In the panel "Measurement"/"Filter Setup"/"Advanced", go to "Advanced" and select a single laser illumination from the adequate side (Fig. 11). You can press the live mode to verify that instead of three, now you have a single light sheet (the image will go darker compared to 3 light sheets).



Illumination correction: Off

Dynamic horizontal focus processing: Contrast adaptive

Fig. 11. Example for left side single laser illumination.

3.3.5 In the xyz-Table Z panel change the step size to 1.6µm, while keeping the same beginning and end planes that previously selected for the autofluorescence scan. Of note, it is not mandatory for the TubeMap alignment to maintain exactly the same brain position, and it is also possible to perform each scan on different imaging sessions.

3.3.6 Define a ROI by clicking in the square icon in the previsualization tools panel (Fig. 12A). Draw a square of 400x1300µm. To easily define the size of the ROI, you can move the square into the upper left corner of the screen, which is the point 0,0 in the coordinates of the navigator. Once the square has the right size, position it in the middle of the image. You can also use the “crosshair” tool to center the position (Fig. 12B). ⚠ At this point, make sure that the illumination of your image is consistent: check the vessels at the top and the bottom of the frame. They should have a similar aspect and intensity. If the vessels at the center top or bottom look blurry or dimmer than the ones on the other end, your microscope needs to be realigned.

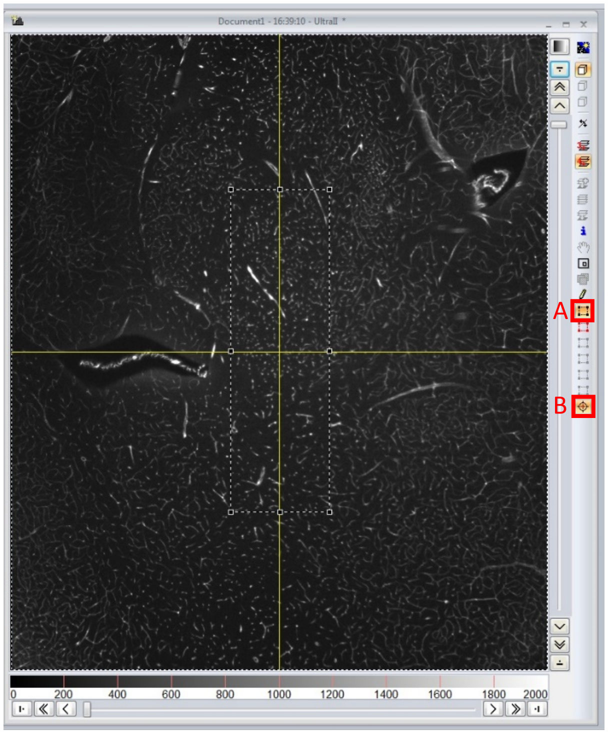


Fig. 12. Functions to draw a ROI A: ROI, B: Crosshair

3.3.7 Center the light sheet. This is a critical step, the thickness of the light sheet is heterogeneous, and only the center of the light sheet should be used. To find the center of the lightsheet go into live mode. In the “Advanced” panel from the “UltraII” window, move down to find the “sheet motor calibration” tool (Fig. 13A). Move the slider to visually identify the center of the lightsheet (Fig. 15). The light sheet is centered if most capillaries in the ROI appear as points (cross-sections of the vessels), instead of strands (indicating a thicker illumination that would allow to see more of the vessel).

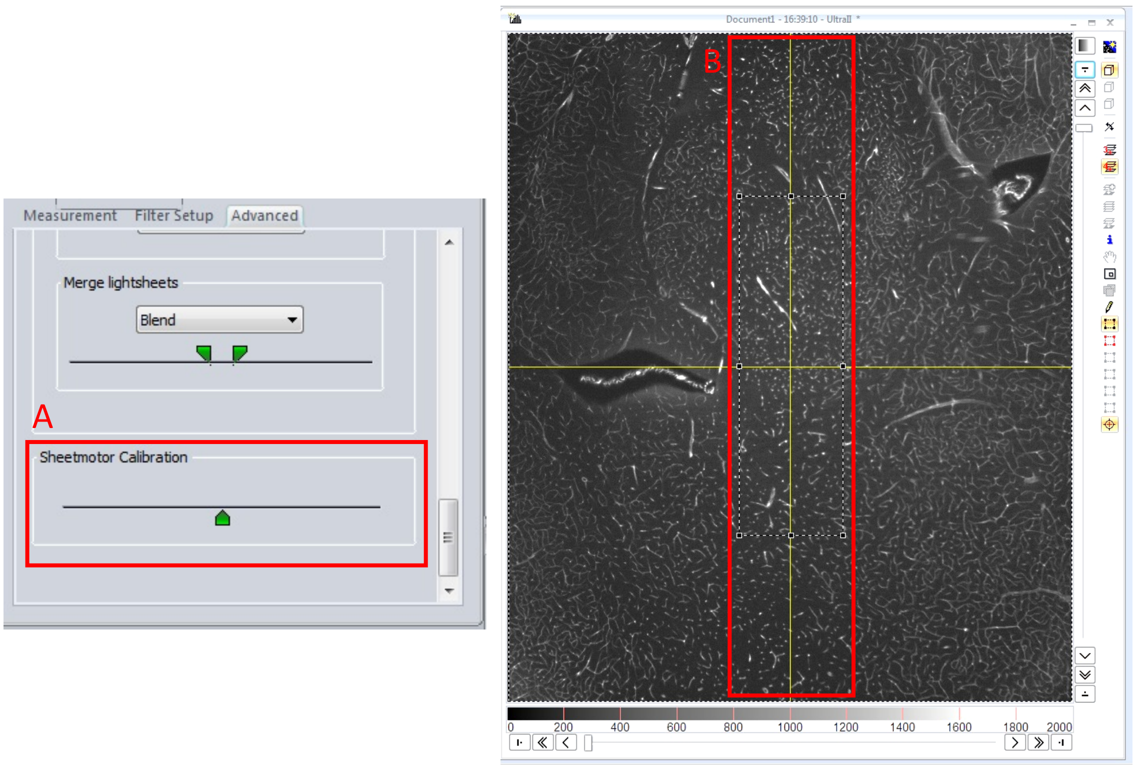


Fig. 13. Lightsheet motor tool to align the center of the laser beam to the ROI.

3.8.7 Once the center of the light sheet and the position of the ROI previously drawn are aligned in the middle of the preview window, crop the ROI by pressing the "subregion" button in the previsualization tools bar (Fig. 14).

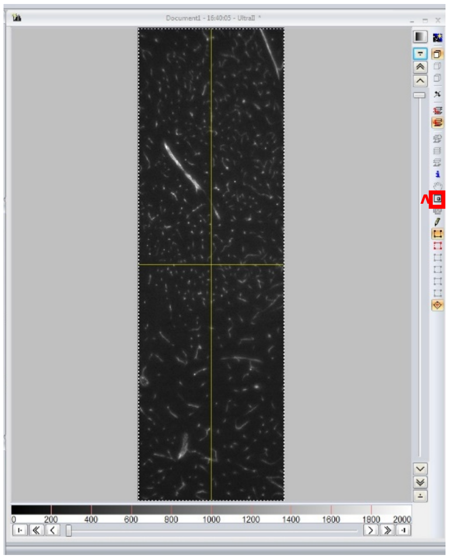


Fig. 14. Tool to crop a ROI.

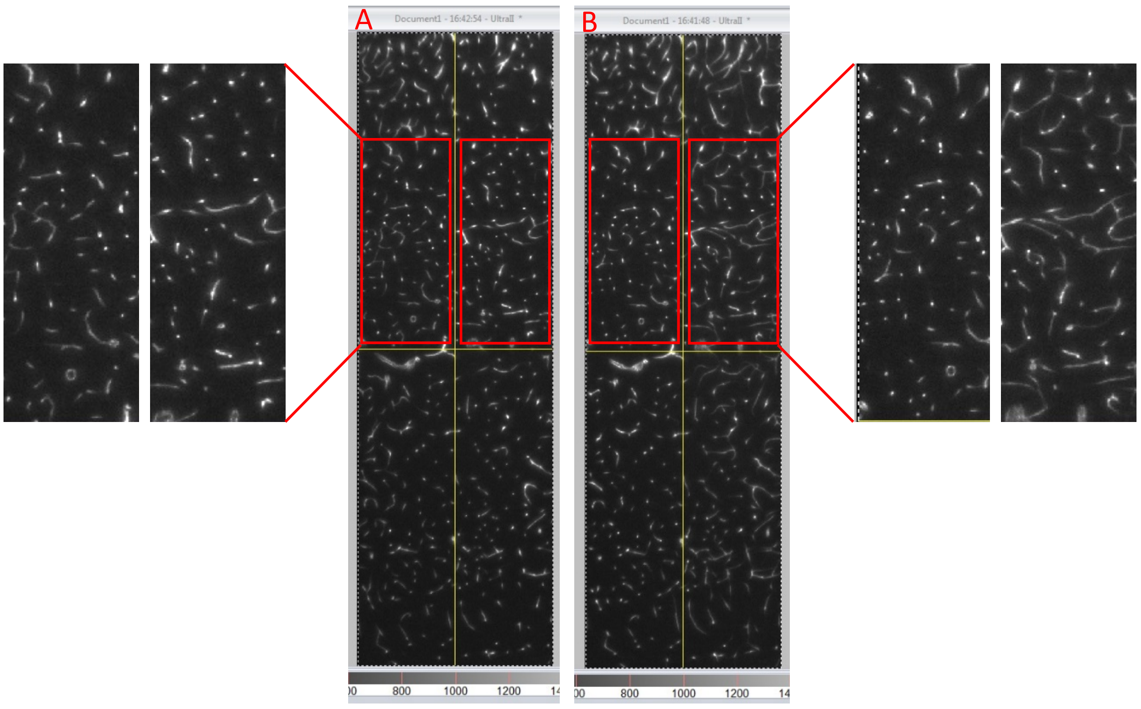

A
Document1 - 16:42:54 - UltraII *
800
1000
1200
B
Document1 - 16:41:48 - UltraII *
800
1000
1200

Fig. 15. Examples or good (**A**) and bad (**B**) lightsheet centering. The asymmetry in the capillary appearance in B is visible from the presence of many continuous vessels on 1 side of the ROI. This points to a left shift in the light sheet centering.

3.8.9 Set the acquisition time taking into account the full dynamic range and avoiding saturation in the surface. Check that the signal is strong enough in the deep regions (in the thalamus for ex. the signal should be above 1000 while the background should be under 500).

3.8.10 Adjust the chromatic correction for both channels. Go in live mode with one channel and precisely set the focus using the "chromatic correction" tool (Fig. 16). Press "set" to save and change the illumination wavelength to repeat the same process with the second channel.

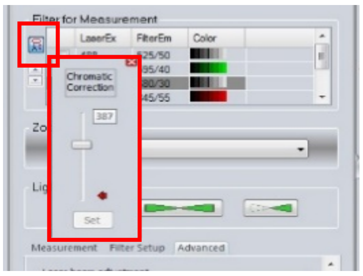
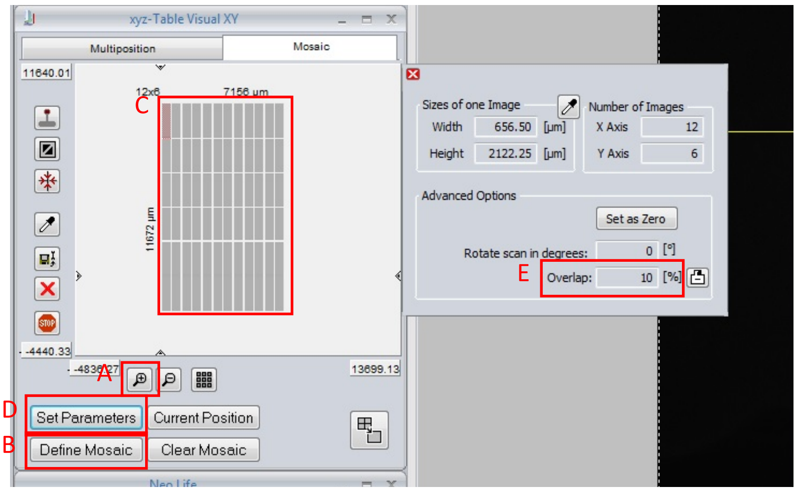


Fig. 16. Chromatic correction tool.

3.8.11 Set the mosaic: Zoom in into the panel "xyz-Table Visual XY" until you see the rectangle corresponding to your ROI (Fig. 17A). Click "Define a Mosaic" (Fig. 17B) and expand the gray grid to a 12x6 mosaic (the size of the grid may vary in between samples but should be around 11/13 columns by 6 rows) (Fig. 17C). Click in "Set Parameters" (Fig. 17D) and specify a 10% overlap in between tiles (Fig. 17. E). Go into live mode and adjust the position of the gray grid to the position of the brain. Double clicking on a grid tile changes the stage physical position, while dragging the gray grid moves its virtual position. The red tile indicates the active stage position.

⚠ Due to the range limitations of the stage, it is possible to set the mosaic outside of the boundaries of normal motor movement. If the limit of movement is reached, the position of the red tile doesn't reflect the real stage position anymore. The grey mosaic may then need to be repositioned. Usually, a mechanical ticking sound indicates that the motor has reached its limit.



Sizes of one Image
Width 656.50 [µm]
Height 2122.25 [µm]

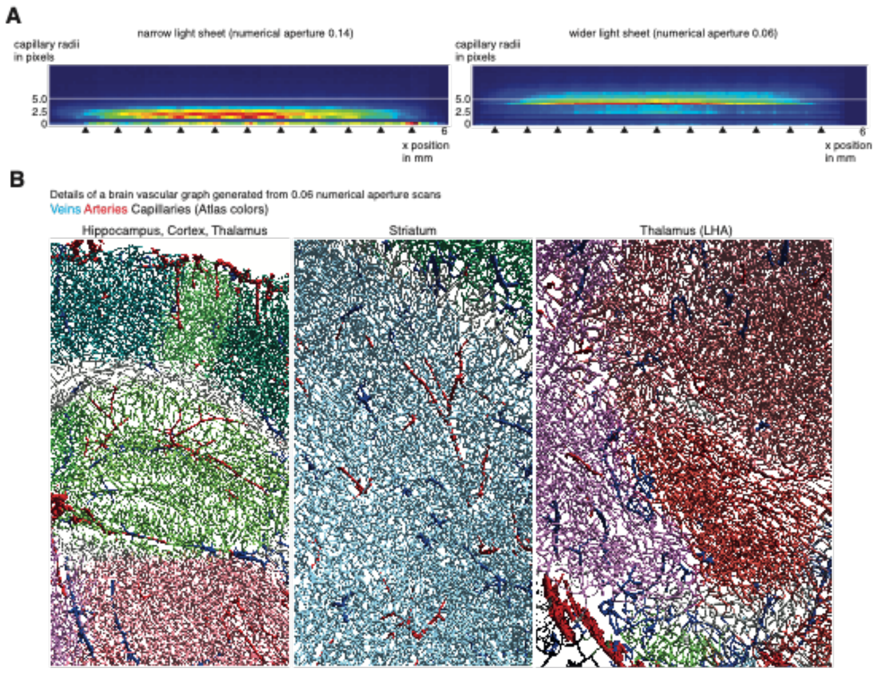
Number of Images
X Axis 12
Y Axis 6

Advanced Options
Set as Zero
Rotate scan in degrees: 0 [°]
Overlap: 10 [%]

Fig. 17. Mosaic configuration.

3.8.12 Launch the scan by clicking in the "play" button. You should get around 300000 to 400000 planes for such a scan. This acquisition takes around 14 hours and should generate a single folder in your directory containing the TIF files of the different stacks, one for each channel and each position. The size of the vasculature scan is about 450Gb.

Effect of the light sheet numerical aperture



Effect of the light sheet aperture on the resolution of the capillary reconstruction and its homogeneity. ***A*** *Distribution of capillary radii along the x axis. Tile centers are indicated with arrowheads. A brain was acquired with a narrow light sheet numerical aperture of 0.14 (left) or a wide light sheet at 0.06 (right). The resolutive light sheet shows finer capillaries (below 2.5 pixels of radius), but is slightly more resolutive at the center of each tile due to the conservative tiling layout. A wider light sheet gives isotropic capillaries throughout the brain, but with a much larger radii, at 4 pixels.* ***B*** *Graph reconstructed from a lower light sheet NA (0.06): while the resolution is inferior to the graphs generated at high NA, it is still appropriate for most brain regions, except the densest ones (an example of the thalamic LHA is given).*

TubeMap

TubeMap is a python pipeline to generate vascular graphs from light sheet scans. It is part of the ClearMap environment, which contains the following tools dedicated to the analysis of light sheet 3D datasets: Wobbly Stitcher, TubeMap itself, StarMap (formerly ClearMap) for the detection of cell nuclei, and ClearPath, for the segmentation of axon tracks (coming soon).

4. Installation and systems requirement

- ClearMap can be executed on any Python environment, Mac, Windows or Linux. We provide the tools for an easy setup on a Linux Debian system for now (Ubuntu or Mint for instance). Easy installation on other systems (Mac or Windows) is coming soon. We use Ubuntu 18.04LTS.
- At least 256Gb of system RAM is strongly recommended, although not necessary. For best performances we use workstations with 512Gb of system RAM.
- A video card with CUDA capabilities (from nVidia) with at least 12Gb of video RAM is necessary for the tube filling image processing. We use Quadro cards with 24Gb of video RAM.
- It is recommended to have around 2Tb of available disk space to run the pipeline on 1 brain. This space is for temporary files.

For ClearMap installation, follow these steps:

- Open a terminal, create a folder to install ClearMap, enter this folder and download ClearMap:

```
> mkdir ClearMap
> cd ClearMap
> git clone https://www.github.com/ChristophKirst/ClearMap2.git
```

- We recommend using Anaconda for an easy installation: download and install Anaconda from here: https://www.anaconda.com/
- To setup the ClearMap environment, open a terminal and go to the ClearMap folder. Setup the environment with the following command:

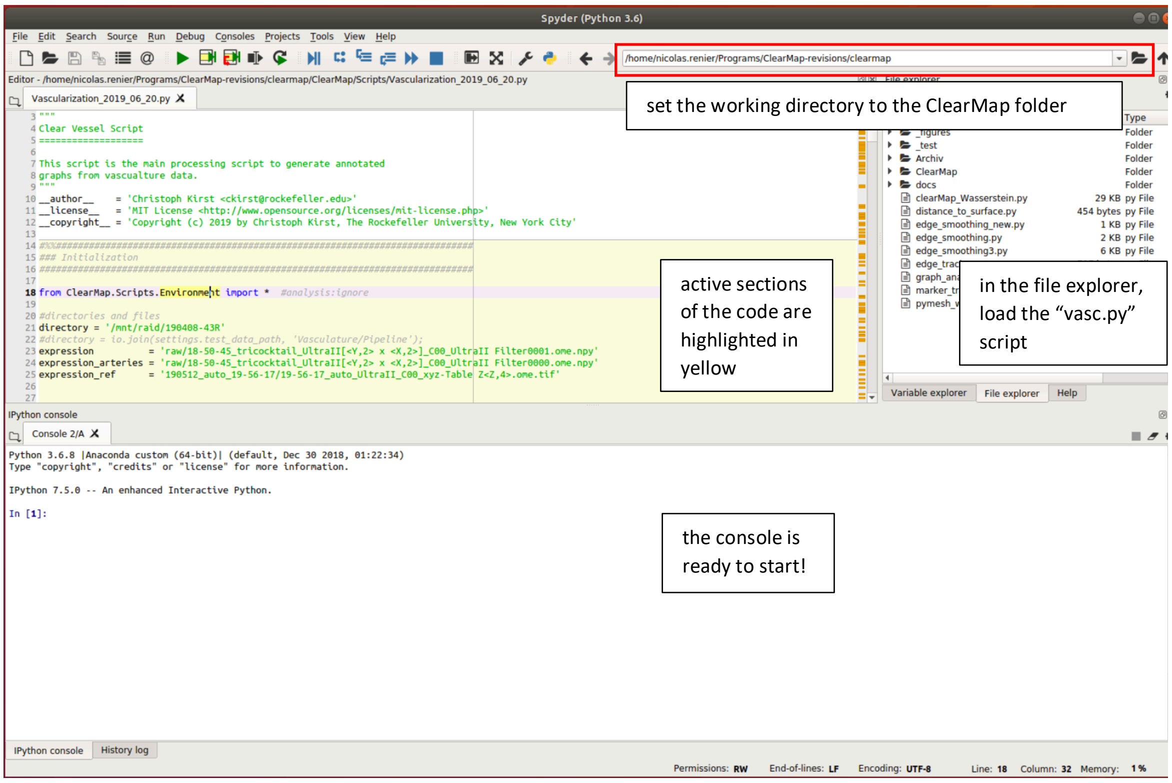
```
> conda env create -f ClearMap2.yml
```

This will automatically download everything else you need to run ClearMap 2.

- Launch the environment and the code interface:

```
> conda activate Clearmap2
(Clearmap2) > spyder
```

- Check that everything is running correctly. You should have a window that looks like this:


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
set the working directory to the ClearMap folder
active sections of the code are highlighted in yellow
in the file explorer, load the "vasc.py" script
the console is ready to start!
Variable explorer
File explorer
Help
IPython console
History log

```
/home/nicolas.renier/Programs/ClearMap-revisions/clearmap
```

Editor - /home/nicolas.renier/Programs/ClearMap-revisions/clearmap/ClearMap/Scripts/Vascularization_2019_06_20.py

Vascularization_2019_06_20.py

```
"""
Clear Vessel Script
===================

This script is the main processing script to generate annotated
graphs from vascualture data.
"""
__author__    = 'Christoph Kirst <ckirst@rockefeller.edu>'
__license__   = 'MIT License <http://www.opensource.org/licenses/mit-license.php>'
__copyright__ = 'Copyright (c) 2019 by Christoph Kirst, The Rockefeller University, New York City'

#%%############################################################################
### Initialization
###############################################################################

from ClearMap.Scripts.Environment import *  #analysis:ignore

#directories and files
directory = '/mnt/raid/190408-43R'
#directory = io.join(settings.test_data_path, 'Vasculature/Pipeline');
expression          = 'raw/18-50-45_tricocktail_UltraII[<Y,2> x <X,2>]_C00_UltraII Filter0001.ome.npy'
expression_arteries = 'raw/18-50-45_tricocktail_UltraII[<Y,2> x <X,2>]_C00_UltraII Filter0000.ome.npy'
expression_ref      = '190512_auto_19-56-17/19-56-17_auto_UltraII_C00_xyz-Table Z<Z,4>.ome.tif'
```

```
                                   Type
  _figures                         Folder
  _test                            Folder
  Archiv                           Folder
  ClearMap                         Folder
  docs                             Folder
  clearMap_Wasserstein.py   29 KB  py File
  distance_to_surface.py 454 bytes py File
  edge_smoothing_new.py      1 KB  py File
  edge_smoothing.py          2 KB  py File
  edge_smoothing3.py         6 KB  py File
  edge_trac
  graph_ana
  marker_tr
  pymesh_w
```

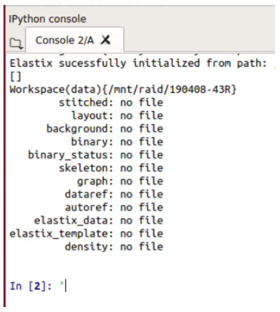
Console 2/A

```
Python 3.6.8 |Anaconda custom (64-bit)| (default, Dec 30 2018, 01:22:34)
Type "copyright", "credits" or "license" for more information.

IPython 7.5.0 -- An enhanced Interactive Python.

In [1]:
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 18   Column: 32   Memory: 1 %

- You have 3 ways to interact with the script:
    - Place the text cursor anywhere on a script section. The whole section text becomes pale yellow. Press 'ctrl + enter' to start the whole section
    - If you press the 'F9' key, only the current line (or current selection of lines) is executed
    - You can also type commands directly in the console
- Check that the imports are correct: load the 'vasc.py' script. Make sure the working directory is set to the ClearMap folder that was downloaded automatically. Click anywhere in the first section called "Initialization". Press 'ctrl + enter' to start the whole section. The first time only the section is started, the code will be compiled. It may take about 30 minutes. Many warning messages will appear, ignore them.

Once all imports are done and compiled, you should get this:



```
Elastix sucessfully initialized from path:
[]
Workspace(data){/mnt/raid/190408-43R}
       stitched: no file
         layout: no file
     background: no file
         binary: no file
  binary_status: no file
       skeleton: no file
          graph: no file
        dataref: no file
        autoref: no file
    elastix_data: no file
elastix_template: no file
        density: no file


In [2]: '|
```

ClearMap details the content of the "workspace", in this example '/mnt/raid/190408-43R'. This is simply the folder that contains our data, and where all the results will be written. The output also lists all the different types of result files that have been generated so far. At the beginning, there are no results yet to show!

5. Stitching and alignment

We implemented a specific data stitcher for ClearMap, called 'Wobbly Stitcher' to correct for common problems encountered when stitching image files generated by large scale imaging, which result in vessels duplications between adjacent tiles.

3.4 Setting the file locations and format

- Open the main script 'vasc.py', and first set the paths to the working directory and files:

```
directory = '/…/sample-01'
expression = 'raw/18-50-45_UltraII[<Y,2> x <X,2>]_C00_UltraII Filter0001.ome.npy'
expression_arteries = 'raw/18-50-45_UltraII[<Y,2> x <X,2>]_C00_UltraII Filter0000.ome.npy'
expression_ref      = 'auto/19-56-17_auto_UltraII_C00_xyz-Table Z<Z,4>.ome.tif'
```

`directory` refers to the folder where the data are located, and where all results will be written.

`expression` refers to the filename of the vessel channel acquisition. It is with the format '`18-50-45_UltraII[02 x 01]_C00_UltraII Filter0001.ome.tif`' where here as an example 02 and 01 refer to the row and column respectively. In the script, we change the particular names of these file with a generic expression to describe the structure of the file name. Thus, `[02 x 01]` becomes `[<Y,2> x <X,2>]` where Y and X are the descriptors of the positions, and '2' refers to the fact that the position is described with 2 digits in the filename. Note that in the script, the expression extension is `.npy` and not `.tif` as we'll be working with this format for subsequent steps.

`expression_arteries` refers to the filename of the arteries channel acquisition. It is handled the same way as `expression` except the channel name at the end.

`expression_ref` refers to the filename of the autofluorescence channel acquisition. Here, the generic expression is similar to the one of the mosaic acquisitions, but in this case, each file is a different plane, and not a different tile position. `19-56-17_auto_UltraII_C00_xyz-Table Z0001.ome.tif` is an example of one of these files. In the generic expression of the script, we replace `0001` with `<Z,4>` because it describes z planes, and 4 digits (note that we keep the `.tif` extension here).

Tip: you don't have to type in the complex filenames yourself! In Linux Ubuntu, you can click on the file whose path you want to get in the file explorer and 'copy' ('ctrl + c'), and then go back on the script and 'paste' ('ctrl + v'), and the full path of the file will be put in.

The file paths in the script omit the path to the work directory (set at `directory`), which is automatically added to all paths. Here, `raw/ or auto/` refers to the acquisition folder that was created by the microscope.

- Then, set the alignment files to match the field of view of your brain acquisition:

```
annotation_file = io.join(resources_directory, 'annotation_25_full.nrrd')
template_file   = io.join(resources_directory, 'template_25.tif')
```

It is important that the alignment template and annotation file match the field of view of your acquisition. By default, they cover the whole brain, but if you acquired a smaller region of the brain (for instance only one hemisphere), you need to prepare new versions of these files to match the coverage of the brain from your acquisition. Also, the orientation in space of these images in these files needs to match the orientation of the brain during your acquisition. You can use Fiji (Image J) to reorient or crop these files to match your acquisition. The file themselves are located in the `'ClearMap/Ressources/Atlas'` folder. You can save new version of these cropped / reoriented files in the same folder and adjust the name here.

- You're all set with the settings! Now we're converting the tif files from the microscope to a faster file format to speed up the stitching. This step is not necessary, but will greatly speed up the generation of the final mosaic. Run the section ('ctrl + enter'):

```
#%%############################################################################
### Tile conversion
###############################################################################

io.convert_files(ws.file_list('expression', extension='tif'), extension='npy', processes=12,
verbose=True)
```

This step should take about 3 minutes. At the end, you will have a duplicate of all the original tif files with the npy format. It is recommended to delete the original tif files from the local disk to free up space.

3.5 Alignment and tile fusion

- Start the alignment: the next long section in the script is the alignment and stitching. You can start it all at once, but first there are a few parameters to set:

```
layout = stw.WobblyLayout(expression=ws.filename('expression'), tile_axes=['X','Y'],
overlaps=(40,130))
```

`overlaps=(40,130)` is the value in pixels of the expected overlaps for the tiles. If you used the tiling parameters described in this guide, they should be (40,130), because we had set a 10% overlap of a (400,1300) pixels rectangles for the tiles. Change these values if needed. Also, because the microscopes will usually have a delay in their real positions, it is often "safe" to set this overlap a bit larger than the theoretical value. On our system, we set is to (45,155) instead of (40,130), because we noticed the stage had a systematic error of 5 pixels in x and 15 pixels in y. However, this tweak is not necessary.

- The first step is an alignment along the z axis of all tiles. It works by projecting the border pixels and scoring the cross-correlation when moving the tiles relative to each other with a certain range. This is set this way:

```
st.align_layout_rigid_mip(layout, depth=[45, 155, None], max_shifts=[(-30,30),(-30,30),(-
20,20)], ranges = [None,None,None], background=(400, 100), clip=25000, verbose=True,
processes='!serial')
```

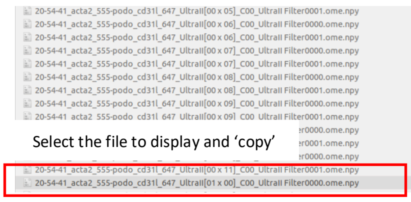
`depth=[45, 155, None]` : this is the depth of the projection used at the border. Just repeat what you put for the overlap before.

`max_shifts=[(-30,30),(-30,30),(-20,20)]` set the range of movement freedom for the tiles from their starting position. The starting position is the one defined previously as `overlaps`. The range is set for x, y and z. On our system, ±30 is a safe value for x and y. The larger the value, the more memory is necessary to compute the displacement.

`background=(400, 100)` here, set the value of the background pixels. This is the value of the black from outside the brain. You can check this easily by opening a tile for inspection. To quickly visualize a tile, you can use the ClearMap data explorer by typing in the console:

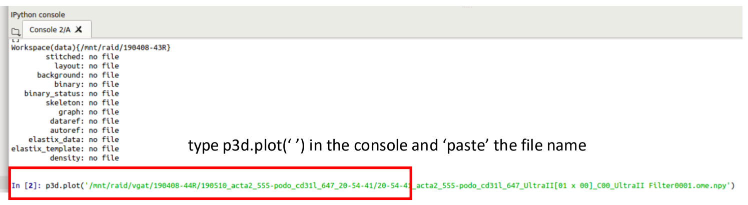
```
p3d.plot('/path_to_a_tile_file.npy')
```

- The function p3d.plot() is an important one to quickly check visually a file, or an array in memory:



```
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 05]_C00_UltraII Filter0001.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 06]_C00_UltraII Filter0000.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 06]_C00_UltraII Filter0001.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 07]_C00_UltraII Filter0000.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 07]_C00_UltraII Filter0001.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 08]_C00_UltraII Filter0000.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 08]_C00_UltraII Filter0001.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 09]_C00_UltraII Filter0000.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 09]_C00_UltraII Filter0001.ome.npy
er0000.ome.npy
er0001.ome.npy
er0000.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[00 x 11]_C00_UltraII Filter0001.ome.npy
20-54-41_acta2_555-podo_cd31l_647_UltraII[01 x 00]_C00_UltraII Filter0000.ome.npy
```

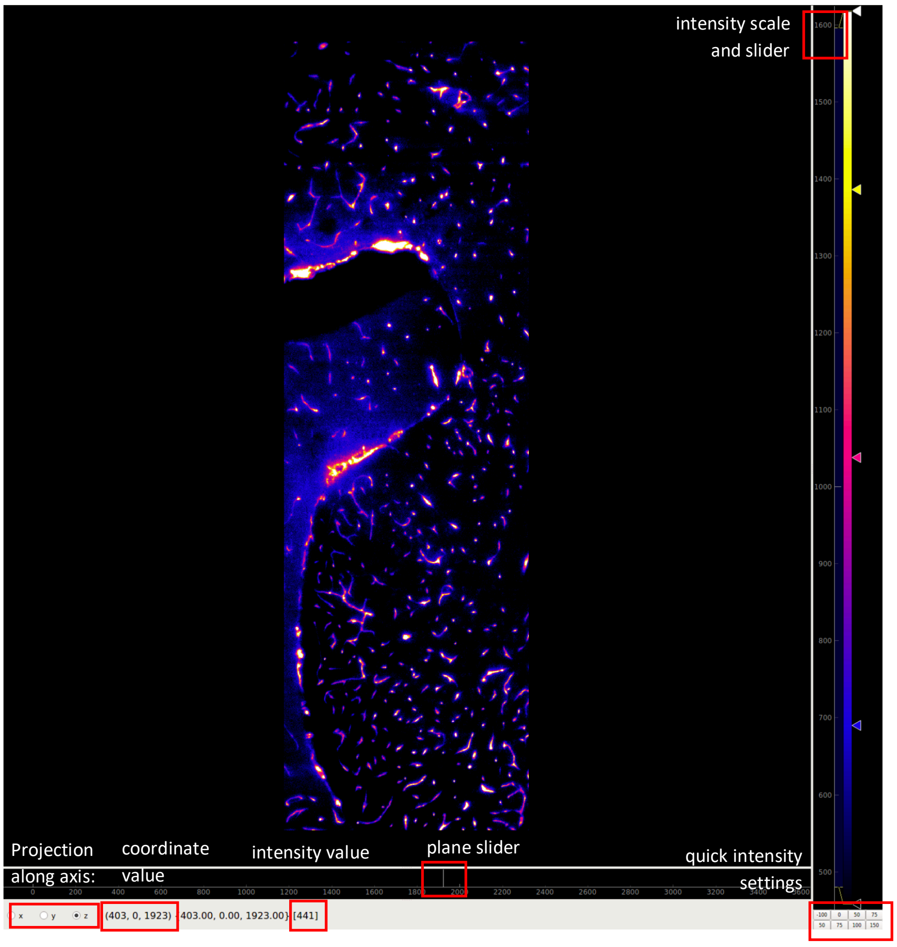
Select the file to display and 'copy'



```
Workspace(data){/mnt/raid/190408-43R}
        stitched: no file
          layout: no file
      background: no file
          binary: no file
   binary_status: no file
        skeleton: no file
           graph: no file
         dataref: no file
         autoref: no file
     elastix_data: no file
elastix_template: no file
         density: no file

In [2]: p3d.plot('/mnt/raid/vgat/190408-44R/190510_acta2_555-podo_cd31l_647_20-54-41/20-54-41_acta2_555-podo_cd31l_647_UltraII[01 x 00]_C00_UltraII Filter0001.ome.npy')
```

type p3d.plot(' ') in the console and 'paste' the file name

The internal ClearMap data visualization window opens:



- After the alignment first step, a layout file (extension lyt) is created to store the progress. This step is where most traditional alignment tools stop. The second step of the alignment in Wobbly Stitcher is to correct for plane displacements during the acquisition, ensuring a more robust alignment. This is controlled by this command in the script:

```
stw.align_layout(layout, axis_range = (None, None, 3), max_shifts =  [(-30,30),(-
15,15),(0,0)], axis_mip = None,
           validate = dict(method='foreground', valid_range = (200, None), size = None),
           prepare = dict(method='normalization', clip = None, normalize=True),
           validate_slice = dict(method='foreground', valid_range = (200,20000), size = 1500),
           prepare_slice = None,
           find_shifts = dict(method='tracing', cutoff = 3*np.sqrt(2)),
           processes = '!serial', verbose = True)
```

The important parameters to set for this function are:

`max_shifts = [(-30,30),(-15,15),(0,0)]` This is the freedom of movement granted to move the individual planes of each tile relative to the first alignment position. On our system, most of the jitter occurs in the x direction, so it is set to ±30.

`validate_slice = dict(method='foreground', valid_range = (200,20000), size = 1500)` This is the most important parameter. Tiles at the border of the sample are challenging to place, because a lot of planes in these positions are outside the sample. Therefore, empty black planes outside the sample can easily be misplaced, and as a domino effect, this misplacement ripples through all connected tiles. To prevent tiles at the border of the scan to affect others, we set a minimal threshold under which the plane is not aligned, nor placed. Using the visualization tool as indicated above, find the lowest intensity level of a central tile in the brain, usually deep in the scan. Set this minimal intensity value in `valid_range = (200,20000)`. The other number (here 20000) is the highest value encountered. If the value is set too high, this could lead to dropped planes inside the brain, which would appear as black tiles in the middle of the mosaic.

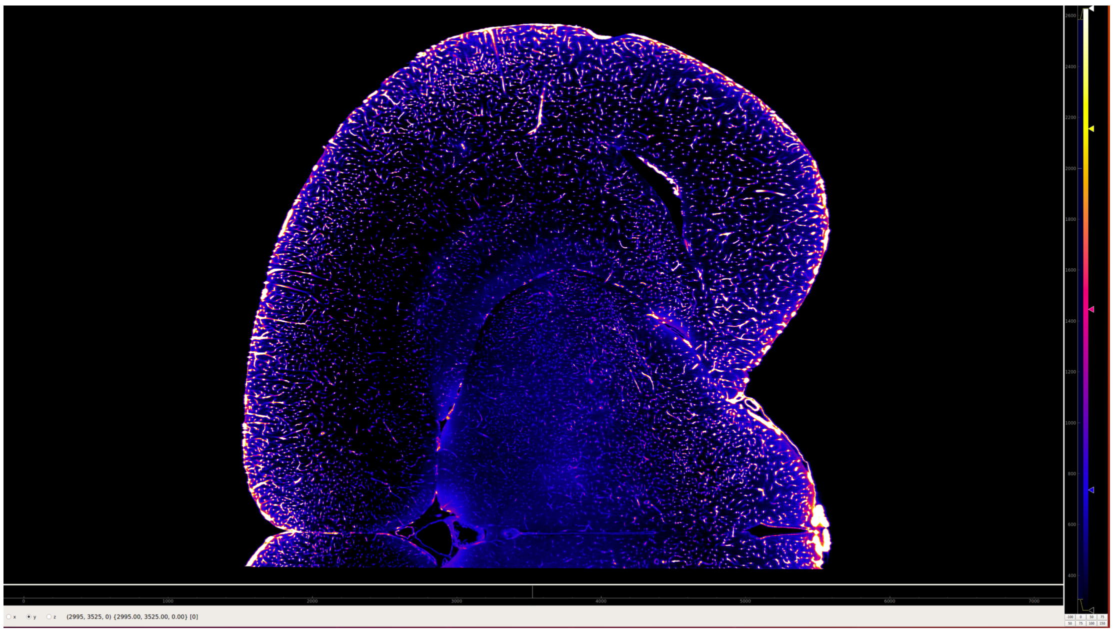
- The other functions in the stitching don't have parameters you need to modify for now, so once these are set, you can run the whole stitching section of the script directly. The whole process should take around 1 hour. At the end of the process, you should have a couple of layout files (lyt extension), and two large image files: `data_stitched.npy` and `data_stitched_arteries.npy`, each of about 200Gb. It is time to check the results!
- To plot files that are part of the workflow of the pipeline, you can still use the `p3d.plot('…')` command seen before, but there is a shortcut command so you don't have to go back finding those files on the disk. The command can be typed in the console:

```
ws.plot('stitched')
```

For the vessel channel. For the artery, we add a postfix:

```
ws.plot('stitched', postfix='arteries')
```

The result of the command will look like this:



You can use the mouse and click left to pan the view, or you can click right and drag to zoom in and out. Inspect the stitching, move the plane slider to look at other regions. If you see black regions in the mosaic, it is likely that the minimal intensity set in `valid_range` is too high. If you see misalignment, you can increase the search radius of `max_shifts` (from both the first `st.align_layout_rigid_mip` function of the second `stw.align_layout` function. If the mosaic looks good, you can delete the original individual tiles from the local disk to free up space.

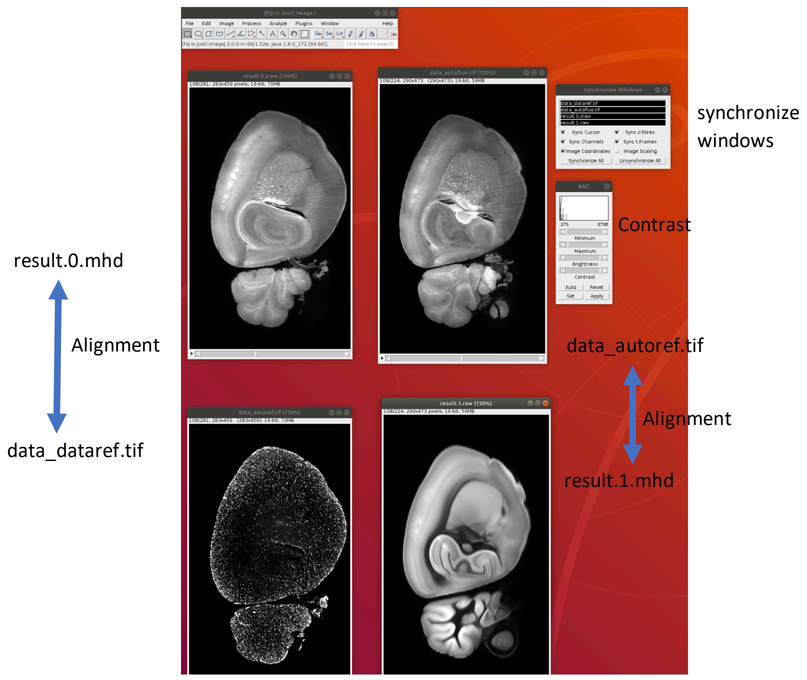
3.6 Alignment to the Atlas

- Now that the mosaic is stitched, the next section of the script will align the scan with the reference atlas. As mentioned above, make sure the template and annotation files are in the same orientation as the brain scan, and that their field of view covers more or less the same regions. In the resampling and alignment cell, there are a few parameters to check:

```
resample_data_parameter = {
    "source_resolution" : (1.625,1.625,1.6),
    "sink_resolution" : (25,25,25),
    "processes": None,
    "verbose":True,
    };

resample_ref_parameter = {
    "source_resolution" : (5,5,6),
    "sink_resolution" : (25,25,25),
    "processes": None,
    "verbose":True,
    };
```

The first `"source_resolution" : (1.625,1.625,1.6)` is the original image resolution (x, y, z). The sink-resolution is the resolution of the Atlas. We use the 25µm version of the annotation. The second source resolution, `"source_resolution" : (5,5,6)` refers to the autofluorescence scan with the 1.3X objective. If you're using other lenses, or different step sizes during the acquisition, change this accordingly. The alignment step should last for about 5 to 10 minutes, and generate two files: `data_dataref.tif` and `data_autoref.tif`, as well as 2 folders: `elastix_ref_to_template` and `elastix_data_to_ref`. These are the result of the 2 consecutive alignments that have been performed: the first one aligns the autofluorescence scan with the mosaic, the second aligns the autofluorescence scan with the Atlas template.

- Then, check the quality of the alignment. The easiest is to use Fiji (Image J). Open the following files with Fiji: `data_dataref.tif` and `data_autoref.tif`. These are the original we need to align. Then go into the `elastix_data_to_ref` folder. This is the alignment of the data file to the autofluorescence. Open the `result.0.mhd file`. Go then to the folder `elastix_ref_to_template`. This is the result of the alignment of the autofluorescence to the Atlas template. Open the `result.1.mhd` file in Fiji. Organize the files as follows:



You can find the contrast adjustment panel in Image -> Adjust -> Brightness/Contrast. The synchronize windows tool can be found in Analyze -> Tools -> Synchronize windows. Click on “Synchronize all”, and travel through the stacks. With the mouse pointer, and make sure each aligned stack are well in sync with each other: make sure the outline of the brain is aligned, as well as the internal structures. Only the aligned data (images organized vertically here) have to match. If the alignment is good, you are then ready for the image processing steps.

6. Image processing

The goal of the image processing step is to turn the 16bit intensity levels image of vessels and arteries into a “binary” image, in which each pixel either belongs to a vessel or not. This step is essential to get a good quality reconstruction. The center line of the vessels will be evaluated from this binary image, and it necessitates the vessels to appear as filled tubes. Therefore, the image processing has two steps: In the first step, we create the first binary mask from the vessels and arteries images. Then in the second step we clean this binary image by removing holes in the vessels and noise in the background.

3.7First binarization

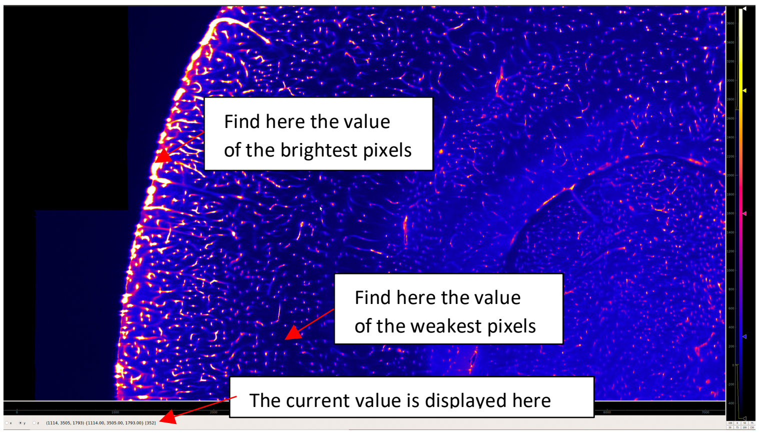
- Start the vessels binarization by executing the next section in the script:

```
source = ws.filename('stitched')
sink   = ws.filename('binary')

processing_parameter = vasc.processing_parameter.copy()
processing_parameter['processes'] = 10
processing_parameter['as_memory'] = True

binarization_parameter = vasc.binarization_parameter.copy()
binarization_parameter['clip']['clip_range'] = (150,20000)

vasc.binarize(source, sink, binarization_parameter=binarization_parameter,
processing_parameter=processing_parameter)
```

Here there are two important parameters to set:

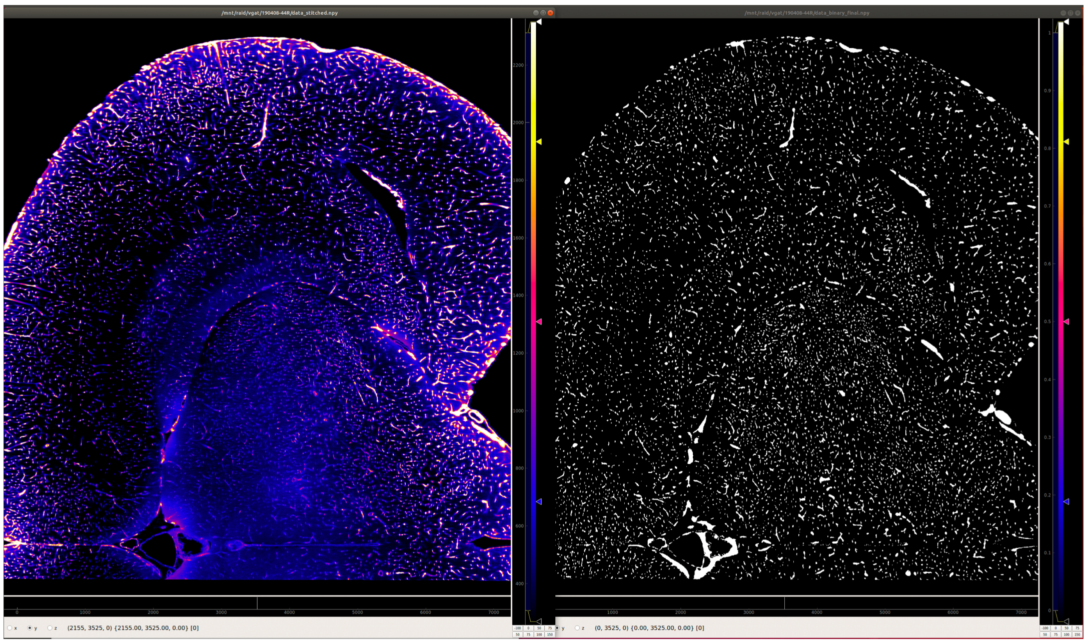
`processing_parameter['processes'] = 10` This defines how many processors work at the same time. The more processors work, the faster the process will be, but the more system (RAM) memory you need. You can start with 10 and check if you get a memory error. If you do, you can decrease the number of processors. If it works fine, you can test with more processors.

`binarization_parameter['clip']['clip_range'] = (150,20000)` Set here the range for the vessel detection. To help you find this range, open the main scan mosaic with the command `ws.plot(‘stitched’)`. Look for the weakest intensities in the pixels inside the brain (ignore the pixels outside the brain). Set the minimum based on the lowest intensities you see. For the higher intensity, look at the large vessels at the surface. You should set to the limit to a value that still excludes the halo.


Find here the value of the brightest pixels
Find here the value of the weakest pixels
The current value is displayed here

Tip: the lowest signal intensity in the whole brain is often found, counterintuitively, in the hippocampus of in the deep cerebellar nuclei

- The vessels binarization should take around 4 to 5 hours. At the end, you can verify the results by typing the command `ws.plot('binary')`. You can also open two files at the same time and synchronize them to inspect the quality of the binarization. For this, use the command `p3d.plot(['path_to_file1.npy','path_to_file2.npy'])`. The `[…]` brackets here will open two side by side synchronized windows. You can also do: `p3d.plot([['path_to_file1.npy','path_to_file2.npy']])` (note the `[[…]]` brackets). This will open a single window, with each file overlaid with different colors. Plot the results (here as an example, with separate windows): `p3d.plot([ws.filename('stitched'), ws.filename('binary')])`



If there is too much noise in the binarization (white pixels everywhere), you should put a higher threshold for the clipping. If there are black regions without detected vessels, you should lower the threshold.

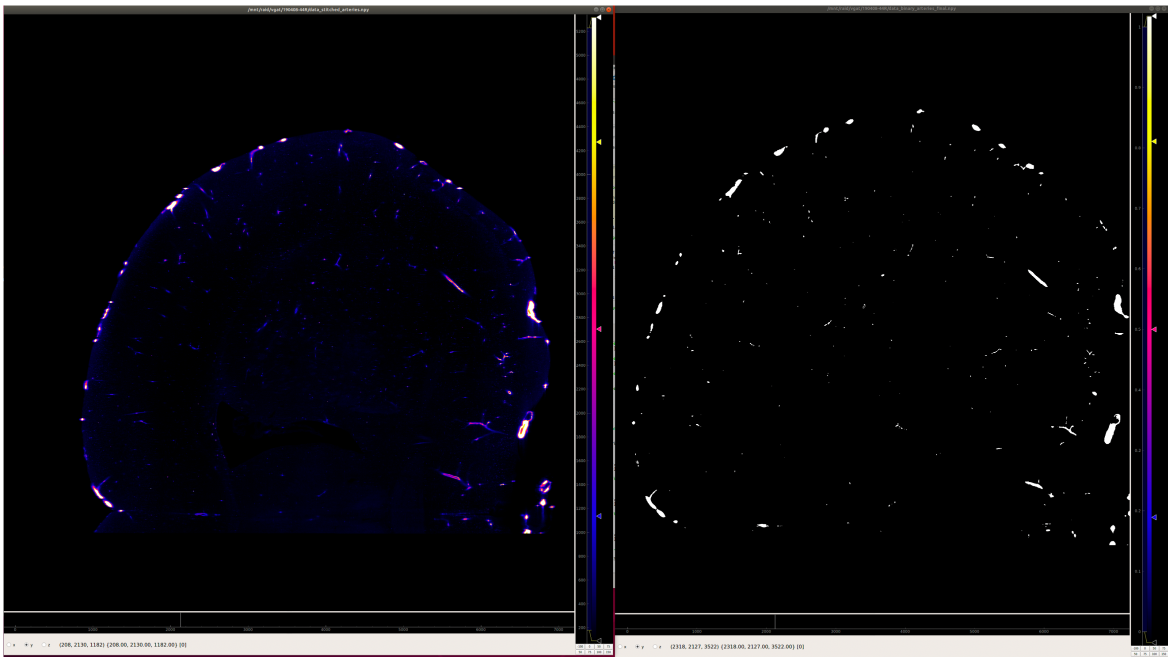
- Start the arterial binarization, which is very similar:

```
source = ws.filename('stitched', postfix='arteries');
sink   = ws.filename('binary', postfix='arteries');
```

```
processing_parameter = vasc.processing_parameter.copy();
processing_parameter['processes'] = 15;
processing_parameter['as_memory'] = True;

binarization_parameter = vasc.binarization_parameter.copy();
binarization_parameter['clip']['clip_range'] = (1000, 8000)
binarization_parameter['deconvolve']['threshold'] = 1000
binarization_parameter['equalize'] = None;
binarization_parameter['vesselize'] = None;

vasc.binarize(source, sink, binarization_parameter=binarization_parameter,
processing_parameter=processing_parameter);
```

The only difference is that the minimal threshold has to be set for both the `['clip']['clip_range'] = (1000,8000)` and the `['deconvolve']['threshold'] = 1000`. This step should take about 2h30.

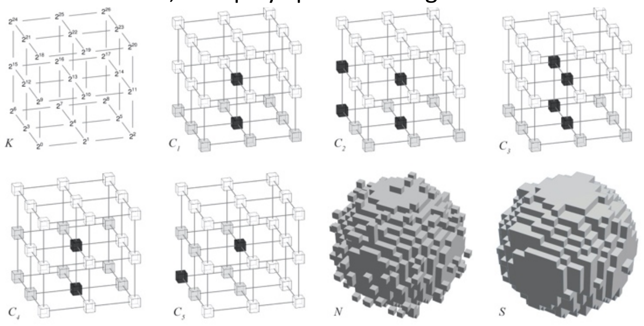
- Open the result with a dual window as above: `p3d.plot([ws.filename('stitched', postfix='arteries'), ws.filename('binary', postfix='arteries')])`

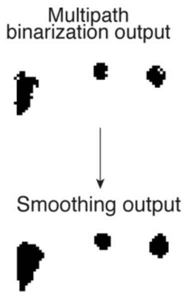


- You're now done with the first binarization, the most time-consuming part of the pipeline!

3.8 Binary smoothing

This step will remove the "bad pixels" from the first binarization: they can be from noise, or a few black pixels in the middle of a vessel, or "spiky" pixels on larger vessels:





The smoothing in TubeMap is highly optimized for large datasets and works with a pre-calculated table of possibilities for pixel removal.

- To start the smoothing, run the next section of the script for vessels:

```
source = ws.filename('binary');
sink   = ws.filename('binary', postfix='smoothed');

postprocessing_parameter = vasc.postprocessing_parameter.copy();

vasc.postprocess(source, sink, postprocessing_parameter=postprocessing_parameter,
as_memory=True)
```

And do the same for arteries:

```
source = ws.filename('binary', postfix='arteries');
sink   = ws.filename('binary', postfix='arteries_smoothed');

postprocessing_parameter = vasc.postprocessing_parameter.copy();

vasc.postprocess(source, sink, postprocessing_parameter=postprocessing_parameter,
as_memory=True)
```

- This step should take about 40 minutes. The effect will not be striking, but if you inspect the details of the result (as in the previous figure), you should see the smoothed vessel shapes. You can now delete the original binary files to free up space, but you should keep the mosaic 'stitched' files, at least the artery one, which will be important for the graph construction later. The vessels 16bit 'stitched' file will not be used anymore in the pipeline.

3.9 Vessel filling

The vessel filling is necessary to find the center line of vessels in datasets generated with immunolabeling, as the vessels are empty. This is done by a neural network, and is simply started by the next code section:

```
vesselFill(inputFolder,nbIteration,DNNFeat,False,saveFile,sampling,patchShape,stepsize,cuda)

io.writeData(os.path.join(directory,'data_binary_06.npy'),(io.readData(saveFile)>0.60))
```

There are no parameters to set for the neural network, except eventually the threshold for the probabilities. The output of the Neural Network is indeed a probability value for each pixel to belong or not to the inside of a vessel. Here, we set the probability threshold of being inside a vessel to 0.6: `io.readData(saveFile)>0.60`

This step should take about 1h per channel (1 for arteries, and 1 for vessels), and produces a new image where the artery binary is filled, and another one where the vessels are filled. These two images are then added to generate the final vessel detection image, which has a pixel coverage of all vessels (indeed, a few large arteries such as the Middle Cerebral Artery are invisible in the CD31 + Podocalyxin channel).

You can plot the result of the final binary files as shown above. If you're happy with the result, you can delete the original binary files and the network probability files, to keep the network output binary files.

7. Graph construction

The graph construction is the easiest part of the pipeline, as there are no parameters to set at the beginning. Then, the arterio-venous labeling is the most important part to obtain correct results.

3.10 Graph construction

- The first step is to start the skeletonization, which will create a vessel center line. Start this section in the script. This should take about 1h.

```
binary   = ws.filename('binary', postfix='final')
skeleton = ws.filename('skeleton')
skl.skeletonize(binary, sink=skeleton, delete_border=True, verbose = True)
```

- The second step is to build the initial raw graph, that transforms the pixels from the center line into a chain of vertices linked by edges. This should take about 20 minutes.

```
graph_raw = gp.graph_from_skeleton(ws.filename('skeleton'), verbose=True)
graph_raw.save(ws.filename('graph', postfix='raw'))
```

- Once this initial graph is built, we then measure the radius of each edge on this graph, based on the binary file. This should take about 1 minute.

```
coordinates = graph_raw.vertex_coordinates()
radii, indices = mr.measure_radius(ws.filename('binary', postfix='final'), coordinates, value
= 0, fraction = None, max_radius = 150, return_indices=True, default = -1)

graph_raw.set_vertex_radii(radii)
```

- Then, we measure the colocalization of each vessel (graph edge) with the arterial Acta2 marker based on the binary mask we generated in the previous steps. We also add for each edge the value of the arterial signal level in the original scan from the microscope. This should take 1 minute.

Adding the binary mask information from the Acta2 channel on the graph:

```
binary_arteries = ws.filename('binary', postfix='arteries_final')
coordinates = graph_raw.vertex_coordinates()
radii = graph_raw.vertex_radii()
radii_measure = radii + 10
expression = me.measure_expression(binary_arteries, coordinates, radii, method='max')
graph_raw.define_vertex_property('artery_binary', expression);
```

Adding the raw Acta2 signal:

```
artery_raw = ws.filename('stitched', postfix='arteries')
coordinates = graph_raw.vertex_coordinates()
radii = graph_raw.vertex_radii()
radii_measure = radii + 10
expression = me.measure_expression(artery_raw, coordinates, radii_measure, method='max')
graph_raw.define_vertex_property('artery_raw', np.asarray(expression.array, dtype=float))
```

- The graph is then cleaned and “reduced”. During the cleaning, bad edge connections between vertices at branch points are corrected, and during the reduction, all the vertices not connected to at least 3 edges (ie, not a branch point), are removed from the graph. There are not completely removed, but stored as an edge_geometry property. This step should take about 30 minutes:

```
graph_cleaned = gp.clean_graph(graph_raw,
   vertex_mappings = {'coordinates'       : gp.mean_vertex_coordinates,
                      'radii'             : np.max,
                      'artery_binary'     : np.max,
                      'artery_raw' : np.max}, verbose=True)

#%% Graph reduction

def vote(expression):
  return np.sum(expression) >= len(expression) / 1.5;

graph_reduced = gp.reduce_graph(graph_cleaned, edge_length = True,
                          edge_to_edge_mappings =    {'length'          : np.sum},
                          vertex_to_edge_mappings = {'artery_binary'   : vote,
                                                     'artery_raw'.     : np.max,
                                                     'radii'           : np.max},
                          edge_geometry_vertex_properties = ['coordinates', 'radii',
'artery_binary', 'artery_raw'],
                          edge_geometry_edge_properties = None,
                          return_maps = False, verbose=True)
```

Save the reduced graph:

```
graph_reduced.save(ws.filename('graph', postfix='reduced'))
```

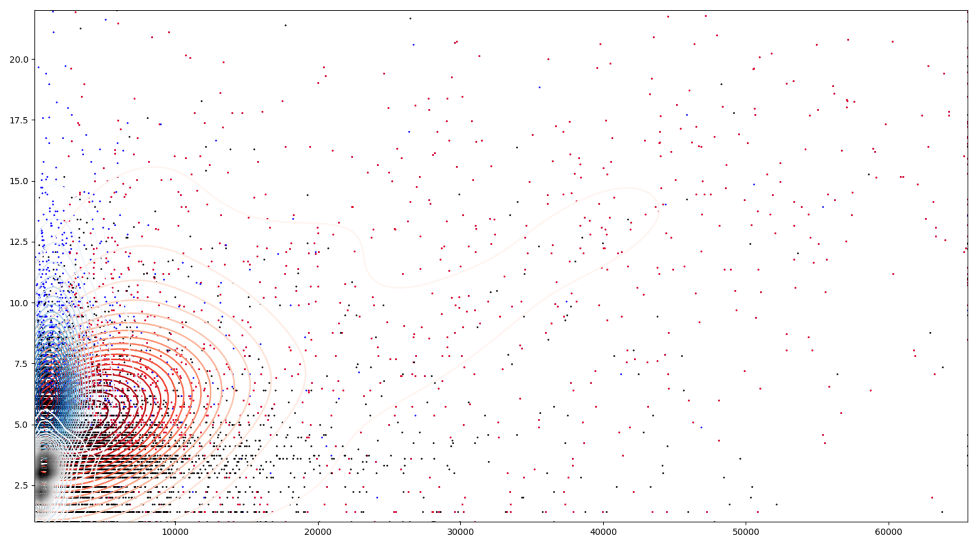
3.11 Graph Registration and Labeling

Now that we constructed the graph, we have to annotate the vessel branches according to their location in the brain and determine their identity.

- Annotate the graph with the template: run the atlas annotation part of the script. There are no parameters to set here as well.

```
#%% graph atlas registration

def transformation(coordinates):
  coordinates = res.resample_points(
                  coordinates, sink=None, orientation=None,
                  source_shape=io.shape(ws.filename('binary', postfix='final')),
                  sink_shape=io.shape(ws.filename('dataref')));

  coordinates = elx.transform_points(
                  coordinates, sink=None,
                  transform_directory=ws.filename('elastix_data', prefix=''),
                  binary=True, indices=False);

  coordinates = elx.transform_points(
                  coordinates, sink=None,
                  transform_directory=ws.filename('elastix_template', prefix=''),
                  binary=True, indices=False);

  return coordinates;

graph.transform_properties(transformation=transformation,
                           vertex_properties = {'coordinates' : 'coordinates_atlas'},
                           edge_geometry_properties = {'coordinates' : 'coordinates_atlas'},
                           verbose=True);

def scaling(radii):
  resample_factor = res.resample_factor(
                      source_shape=io.shape(ws.filename('binary', postfix='final')),
                      sink_shape=io.shape(ws.filename('dataref')))
  return radii * np.mean(resample_factor);

graph.transform_properties(transformation=scaling,
                           vertex_properties = {'radii' : 'radii_atlas'},
                           edge_properties   = {'radii' : 'radii_atlas'},
                           edge_geometry_properties = {'radii' : 'radii_atlas'})

#%% annotation

ano.set_annotation_file(annotation_file)
def annotation(coordinates):
  label = ano.label_points(coordinates, key='order');
  return label;

graph.annotate_properties(annotation,
                          vertex_properties = {'coordinates_atlas' : 'annotation'},
                          edge_geometry_properties = {'coordinates_atlas' : 'annotation'});
```

- Detect the large veins: to detect large veins, we use two criteria. Large veins have a large radius, but they also have a low expression of the Acta2 marker comparatively to arteries of similar radius. This is visible in this plot:



Here, each dot represents a vessel annotated. Blacks are capillaries, blues veins and reds arteries. The plot here show the distribution of the radius (in pixel) in function of the Acta2 signal level from the raw stitching images. Large veins (radii above 10 for instance), have a low Acta2 expression predominantly, while arteries high Acta2 expression. We first look at a “safe” quadrant to identify our first veins, so we pick a high radius and a low maximal Acta2 expression. Some veins don’t express detectable amounts of Acta2, so we set the minimum to 0:

```
vein_large_radius = 8.5
vein_artery_expression_min = 0;
vein_artery_expression_max = 2200;

radii  = graph.edge_property('radii');
artery_expression = graph.edge_property('artery_raw');

vessel_large  = radii >=  vein_large_radius;

vein_expression = np.logical_and(artery_expression >= vein_artery_expression_min,
                                 artery_expression <= vein_artery_expression_max);

vein_large = np.logical_and(vessel_large, vein_expression)
```

Then we set our first arteries. Arteries have been defined in a previous step as vessels that have Acta2 expression in the binary mask. However, this definition can include two wrong annotations: either isolated vessel branches, caused by noise in the arterial binary file, or large veins that express low levels of Acta2. We will therefore “cleanup” the initial arterial annotation from the binary mask by setting the minimum number of branches an artery must have, that we set here to 3:

```
min_artery_size = 3;

artery = graph.edge_property('artery_binary');
graph_artery = graph.sub_graph(edge_filter=artery, view=True);
graph_artery_edge, edge_map = graph_artery.edge_graph(return_edge_map=True)

artery_components, artery_size =
graph_artery_edge.label_components(return_vertex_counts=True);
remove = edge_map[np.in1d(artery_components, np.where(artery_size < min_artery_size)[0])];
artery[remove] = False;

artery = np.logical_and(artery, np.logical_not(vein_large))

graph.define_edge_property('artery', artery)
```

After having cleaned the arterial annotation, we might have new veins to detect that had been wrongly assigned an arterial identity before, and we can expand a bit our initial pool of veins:

```
vein_big_radius = 8.5

radii  = graph.edge_property('radii');
artery = graph.edge_property('artery');
big_vessel  = radii >=  vein_big_radius;

vein = np.logical_and(np.logical_or(vein_large,big_vessel), np.logical_not(artery))

graph.define_edge_property('vein_big', vein);
```

Now that we have our seeds placed for veins and artery, let's trace the vessels down to smaller radii to finish the labeling. Let's do the arteries first:

```
artery_trace_radius = 5
artery_expression_min = 500
distance_threshold = 15
max_artery_tracing = 5

radii = graph.edge_property('radii')
artery = graph.edge_property('artery')
artery_expression = graph.edge_property('artery_raw')
distance_to_surface = graph.edge_property('distance_to_surface')

def continue_edge(graph, edge):
  if distance_to_surface[edge] < distance_threshold or vein[edge]:
    return False;
  else:
    return radii[edge] >= artery_trace_radius and artery_expression[edge] >=
artery_expression_min

artery_traced = gp.trace_edge_label(graph, artery, condition=continue_edge,
max_iterations=max_artery_tracing)

graph.define_edge_property('artery', artery_traced)
```

There are 4 important parameters to set here:

`artery_trace_radius = 4.5` is in pixels the radius where the tracing will stop. At 4.5, we are still well above the size for most capillaries (about 14µm diameter).

`artery_expression_min = 500` is the minimum signal level we are looking for from the Acta2 staining. 500 in our data is very low, and close to the background signal. Still, we are not tracing Acta2 negative branches here.

`distance_threshold = 15` During the Atlas alignment step, the distance of the vertices of our graph to the brain surface is measured in pixels. We are avoiding tracing the vessels in the surface, as the vein-artery segmentation is not good enough there. 15 is the number of pixels in the Atlas coordinate system, which is about 375µm.

`max_artery_tracing = 5` means that we are stopping the trace if it is longer than 5 branches away from the starting point. This is to avoid propagating the trace everywhere, but this usually doesn't happen, so that parameter could be set much higher.

Once the arteries are completed, we now also complete the veins similarly via tracing. First, let's add more seed points, now that the arterial annotation is complete, we can lower the minimum radius for vein labeling without too much risk, and set the new vein radius to 6 pixels:

```
vein_big_radius = 6

radii  = graph.edge_property('radii')
artery = graph.edge_property('artery')
big_vessel  = radii >=  vein_big_radius

vein = np.logical_and(np.logical_or(vein_large,big_vessel), np.logical_not(artery))
```

```
graph.define_edge_property('vein_big', vein)
```

And let’s trace from there:

```
vein_trace_radius = 4.5;
max_vein_tracing = 5;
min_distance_to_artery = 1;

radii = graph.edge_property('radii');
artery = graph.edge_property('artery');
vein_big  = graph.edge_property('vein_big');
artery_expanded = graph.edge_dilate_binary(artery, steps=min_distance_to_artery);

def continue_edge(graph, edge):
  if artery_expanded[edge]:
    return False;
  else:
    return radii[edge] >= vein_trace_radius;

vein = gp.trace_edge_label(graph, vein_big, condition=continue_edge,
max_iterations=max_vein_tracing);

graph.define_edge_property('vein', vein);
```

The arterial tracing parameters function the same way as the vein tracing parameters explained above.

- That’s it! If you’ve reached this point, you’ve successfully reconstructed your first brain vascular graph! Let’s save the results:

```
graph.save(ws.filename('graph', postfix='labeled'))
```

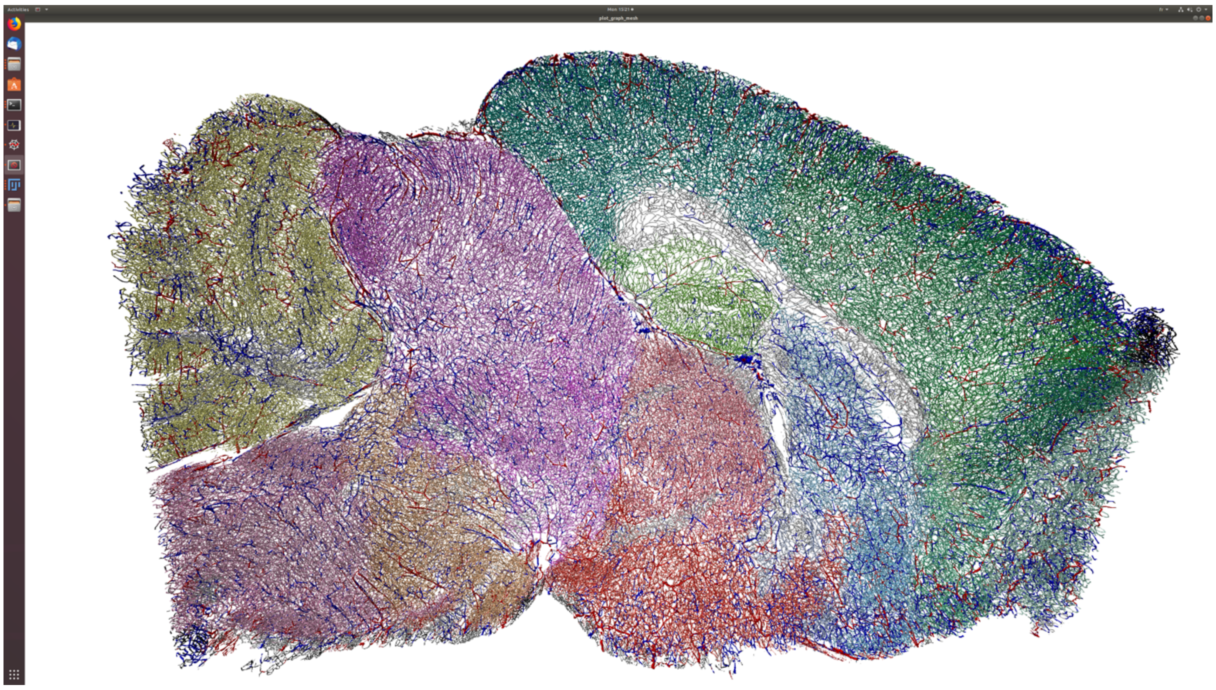
You can plot a thick slice of your graph to check that it looks correct. In this example, the coordinates are in the Atlas coordinate system. You can look at the annotation file with Fiji to help you retrieve coordinates of interest. Here, we select `slice(1,300), slice(50,480), slice(165,175))` which covers a whole sagittal plane with a thickness of 250µm in z.

```
gs = graph.sub_slice((slice(1,300), slice(50,480), slice(165,175)),
coordinates=’coordinates_atlas’)
```

You can plot all the vessels.

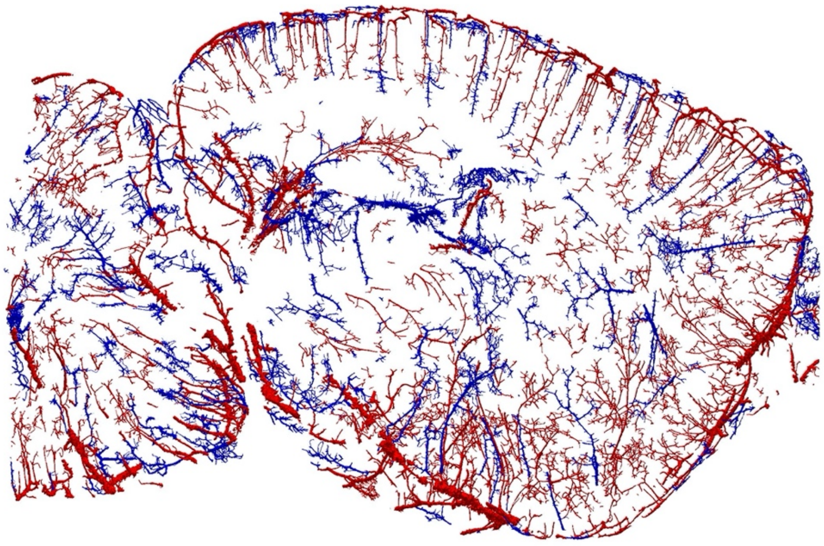
```
edge_vein_label = gs.edge_property('vein');
edge_artery_label = gs.edge_property('artery');

vertex_colors = ano.convert_label(gs.vertex_annotation(), key='order', value='rgba');

connectivity = gs.edge_connectivity();
edge_colors = (vertex_colors[connectivity[:,0]] + vertex_colors[connectivity[:,1]])/2.0;
edge_colors[edge_artery_label >0] = [0.8,0.0,0.0,1.0]
edge_colors[edge_vein_label   >0] = [0.0,0.0,0.8,1.0]

p = p3d.plot_graph_mesh(gs, edge_colors=edge_colors, n_tube_points=3);
```

In the window that opens, it may take around 3 to 10 minutes to render the 3d graph, just wait for the empty window to fill. Once the 3d graph appear, you can manipulate it with the mouse to zoom in and out (by holding the right button), to pan (by holding the shift key and the left mouse button), to rotate (holding the left mouse button) or change the perspective (hold the shift key and the right mouse button).



Or plot only veins and arteries:

```
edge_vein_label = gs.edge_property('vein');
edge_artery_label = gs.edge_property('artery')

edge_filter=np.logical_or(edge_vein_label,edge_artery_label)
gsrt = gs.sub_graph(edge_filter=edge_filter)

edge_vein_label = gsrt.edge_property('vein');
edge_artery_label = gsrt.edge_property('artery')

vertex_colors = ano.convert_label(gsrt.vertex_annotation(), key='order', value='rgba');
```



3.12 Now what?

There are many different ways to visualize and analyze vascular graphs. Specific applications are detailed in companion scripts that come with TubeMap, and replicate the analysis done in the TubeMap manuscript.