DeepMACT Handbook

This document serves as guidance to implement DeepMACT; please refer to the publication for more details: **Pan*, Schoppe*, Parra-Damas*, et al. Deep learning reveals cancer metastasis and therapeutic antibody targeting**

Please also refer to our other online resources and video documentations available at http://discotechnologies.org/.

Contents:

A) Overview of the DeepMACT pipeline

The DeepMACT pipeline is an end-to-end procedure that enables highly automated detection and characterization of tumor micrometastases in large volumetric scans of whole cleared mice. This pipeline comprises two core technologies: DISCO tissue clearing and deep learning. This handbook is intended to provide a step-by-step guideline to explain each step in detail.

Step 1: DISCO imaging of whole mice

Step 1.1: Animal preparation

Step 1.2: vDISCO tissue clearing with immuno-staining and signal enhancement

Step 1.3: 3D image acquisition with fluorescent light-sheet microscopy

Step 2: Deep learning-based quantification

Step 2.1: Data preparation

Step 2.2: Identification and segmentation of micrometastases with DeepMACT

Step 2.3: Metastasis-level analyses

A) Step-by-step instructions for applying the DeepMACT pipeline

Step 1.1: Animal preparation

Timing: 30-40 min per mouse + post-fixation (a few hours to overnight).

1. Anesthetize the animal with triple combination of metedominim 0,5mL + midazolam 5mL + fentanyl 1mL (MMF). The dosage depends on the animal´s weight (1ml/100g of body mass for mice).

2. Wait a few minutes for anesthesia to take complete effect and pinch the toe of the animal to make sure that the animal is fully anesthetized.

3. Perfuse the animal first at room temperature with 0.01 M Phosphate Buffer Saline (PBS) containing Heparin (25 U/ml, Ratiopharm GmbH, N68542.03) for 5-10 min until the blood is completely removed from the tissue (the liver will become yellow).

4. Switch the perfusion to fixative solution: 4% PFA in 0.01 M PBS and continue perfusion with 4% PFA for 15-20 min at a speed of 3 ml/min with a peristaltic pump (or 100-140mm Hg pressure on Leica Perfusion One system).

5. Remove the skin from the animals and clean the gut and stomach from feces (food and gut content may hinder the imaging): using scissors, make a few incisions of the gut in different regions and one of the stomach and with the help of a syringe with PBS gently flush the content out without dissecting out the gut or the stomach. A tiny piece from the back of the skull at the level of the occipital bone is removed by fine scissors to achieve better fixation of the brain.

6. Post-fix the mouse bodies in 4% PFA overnight at 4°C. Avoid long post-fixation because PFA might increase the autofluorescence over time. Before further processing for whole-body immunolabeling, wash the mouse bodies 2-3 times with PBS at room temperature or long term store the samples in PBS with 0.05% sodium azide at 4°C for up to some months.

Step 1.2: vDISCO tissue clearing with immuno-staining and signal enhancement

Preparation of vDISCO whole-body clearing solutions

1. Prepare decolorization solution: 25–30 vol% dilution of CUBIC reagent 1 in 0.01 M PBS. CUBIC reagent 1 was prepared with 25 wt% urea (Carl Roth, 3941.3), 25 wt% N,N,N',N'-tetrakis (2-hydroxypropyl)ethylenediamine (Sigma-Aldrich, 122262) and 15 wt% Triton X-100 in 0.01 M PBS.

2. Prepare decalcification solution: 10 wt/vol% EDTA (Carl Roth, 1702922685) in 0.01 M PBS, adjusting the pH to 8–9 with sodium hydroxide (Sigma-Aldrich, 71687).

3. Prepare permeabilization solution: permeabilization solution contains 1.5% goat serum (Gibco, 16210072), 0.5% Triton X-100 (AppliChem, A4975,1000), 0.5 mM of methyl-β -cyclodextrin (Sigma-Aldrich, 332615), 0.2% trans-1-acetyl-4-hydroxyl-proline (Sigma-Aldrich, 441562) and 0.05% sodium azide (Sigma-Aldrich, 71290) in 0.01 M PBS.

4. Prepare washing solution: washing solution contains 1.5% goat serum, 0.5% Triton X-100, 0.05% sodium azide in 0.01 M PBS.

5. Prepare DISCO clearing solutions: dehydration solutions consisted of a gradient series of tetrahydrofuran (THF; Sigma-Aldrich, 186562) in distilled water: 50 vol% THF, 70 vol% THF, 80 vol% THF, 100 vol% THF and 100 vol% THF; dichloromethane (Sigma-Aldrich, 270997) for delipidation; BABB (benzyl alcohol + benzyl benzoate 1:2; Sigma-Aldrich, 24122 and W213802) for refractive index matching.

Whole-body immunostaining, labeling and clearing

Establish the simplified transcardial-circulatory system as in **Figure 1**. *Here, we used a peristaltic pump (ISMATEC, REGLO Digital MS-4/8 ISM 834; reference tubing, SC0266) with one channel per sample.*

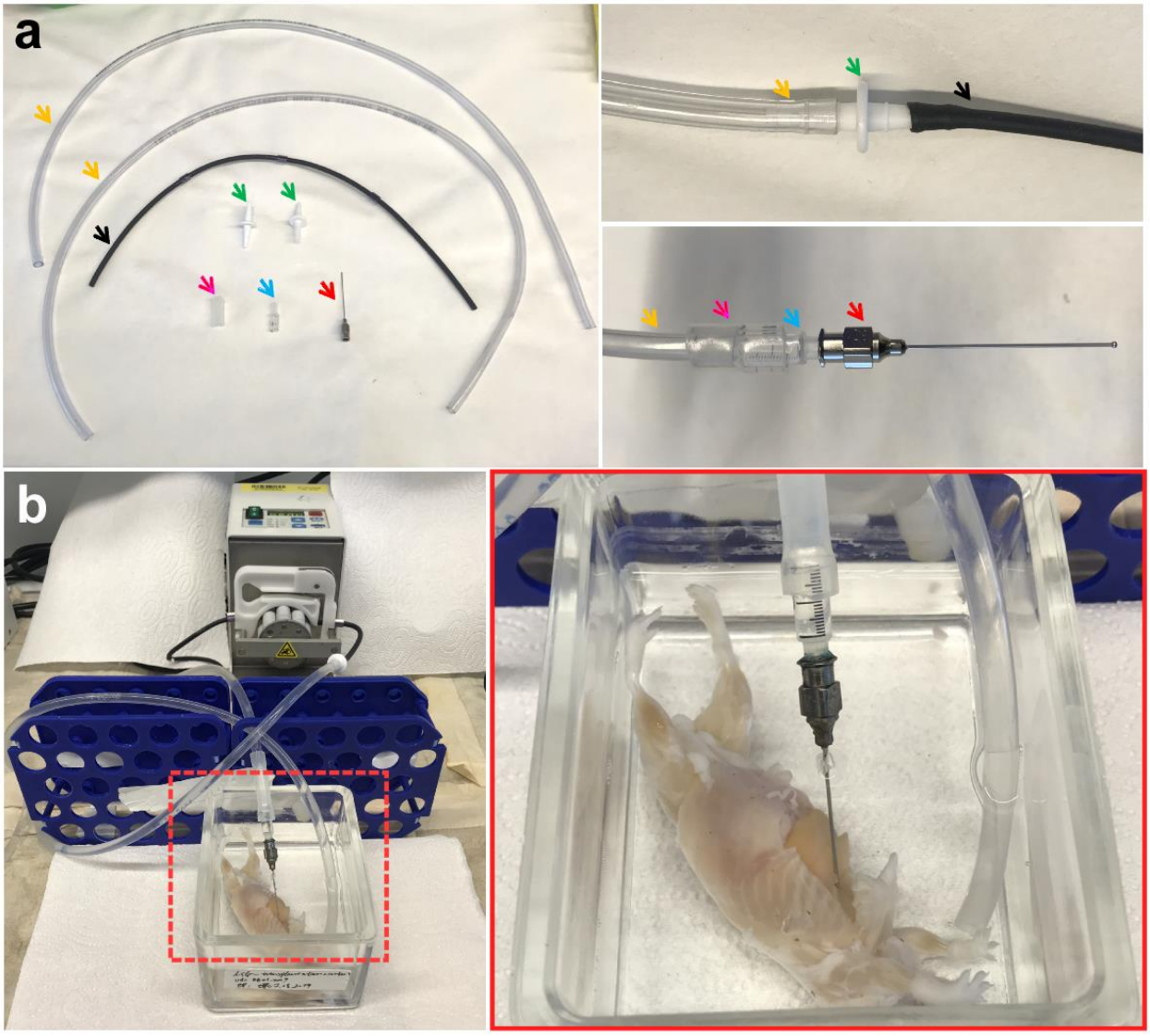

Figure 1: Establishing the transcardiac perfusion system.
(a) *Components of the pumping system: 1x reference tubing (ISMATEC, SC0266) (black arrow), 2x PVC tubing for extension (Omnilab, 5437920) (yellow arrow), 2x tubing connectors (Omnilab, 5434482) (green arrow), 1x soft tubing for connecting (magenta arrow), 1x needle connector cut from 1 ml syringe (Braun, 9166017V) (blue arrow), 1x transcardiac perfusion needle (Leica, 39471024) (red arrow). The zoomed-in pictures show the details of the connecting parts of tubing and needle respectively.* ***(b)*** *A completed setup of the transcardiac circulatory system. The zoomed-in picture shows that the perfusion needle is inserted into the left ventricle of the mouse and it is ready to start the pumping process.*

1. Connect the reference tubing to the peristaltic pump.

2. Insert the tubing connectors (Omnilab, 5434482) (green arrow) at each end of the reference tubing (black arrow).

3. Connect the tubing connectors (green arrow) with additional PVC tubing (Omnilab, 5437920) (yellow arrow).

4. Cut the head part of the 1 ml syringe (Braun, 9166017V) as a connector (blue arrow) for the perfusion needle and insert it into the outflow tubing (yellow arrow).

5. Connect the transcardiac perfusion needle (Leica, 39471024) (red arrow) and fix the inflow tubing in the glass chamber (Omnilab, 5163279) using adhesive tape. Keep a certain level of solution in the glass chamber to ensure that the sample is covered by solutions during the entire process. In general, since the maximum volume of the standard chamber is about 250 ml, 200 ml of appropriate solution would be sufficient for labeling.

6. Place the whole mouse body in the glass chamber and keep the inflow tubing underneath the surface of the solution. Start the pumping circulation until air bubbles are completely eliminated from the tubing system.

7. Insert the perfusion needle into the heart of the animal through the same pinhole made by sample preparation and remove the remaining PBS to expose the heart. Then, put a drop of superglue (Pattex, PSK1C) at the pinhole where the needle was inserted inside the heart and leave it for several minutes for solidification. This will help to maintain the pumping pressure (160–230 mmHg, or 45–60 rpm) during the labeling.

8. After setting this active pumping system, start the circulation of appropriate solutions one by one as indicated in **Figure 2**. Stop the pumping temporarily when changing the labeling solutions between steps.

9. Firstly, circulate the decolorization solution for 2-3 days at room temperature with one round of refreshment. The solution will turn from clear to yellowish and the spleen and liver become lighter in color (indicating that the blood heme was extracted).

10. Next, after washing with 0.01 M PBS for 3 hours 3 times, perfuse the decalcification solution for 2 d at room temperature and again wash the sample with 0.01 M PBS for 3 hours 3 times.

11. After this, perfuse the animals with 200 ml of permeabilization solution overnight at room temperature and then circulate the immunostaining solution containing 35 µl of signal-enhancing nanobody (Chromotek RFP or GFP signal-enhancing nanobody, stock concentration 0.5–1 mg/ml), corresponding to 17.5–35 µg in 200 ml (0.088–0.175 µg/ml), 1:6000 in dilution, and/or 300 µl of propidium iodide (PI) (Sigma-Aldrich, P4864, stock concentration 1 mg/ml). The immunostaining solution should be filtered through a 0.22 µm syringe filter (Sartorius, 16532) before use. During the circulation, another 0.22 µm syringe filter is connected to the entrance of the inflow tubing to prevent the accumulation of dye aggregates into the mouse body. This immunolabeling step will last for 6 days and is followed by a passive labeling step conducted in the same staining solution by adding extra 5 µl of signal-enhancing nanobody with gentle shaking at 37 °C.

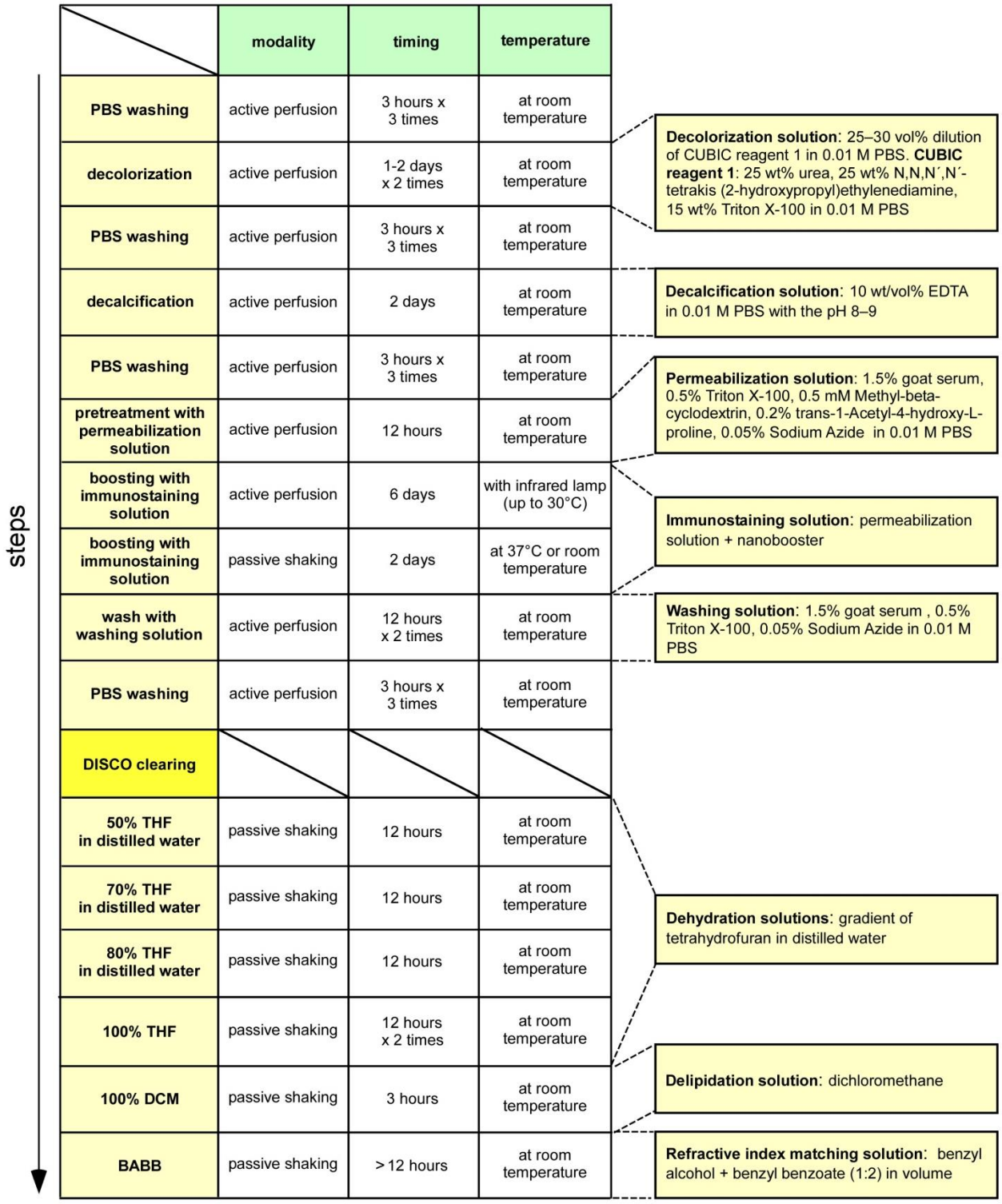

Figure 2: Pipeline for vDISCO whole-body immunolabeling and clearing protocol.
The timing for each step can be shortened or extended based on the age of the animals to achieve better labeling and clearing efficacy. Multiple staining such as propidium Iodide nuclear staining in addition to nanobody enhancement can be applied during whole-body circulation. After light-sheet microscopy imaging, the organs of interest can be dissected and rehydrated for further histological staining and high-resolution confocal microscopy.

12. Place the mouse back in the circulation system after the previous passive labeling step, perfuse with washing solution for 1 day with one-time refreshment, and 0.01 M PBS for 3 hours 3 times at room temperature.

13. After the staining, clear the animals at room temperature in dehydration and clearing solutions in the same glass chamber, kept on a shaking rocker (IKA, 2D digital) with gentle rotation in a fume hood. For dehydration, the mouse body will be incubated in 100 ml of the following gradient series of THF in distilled water (12 h for each step): 50 vol% THF, 70 vol% THF, 80 vol% THF, 100 vol% THF and again 100 vol% THF, followed by 3 h in dichloromethane and finally in BABB for refractive index matching. During all incubation steps, the glass chamber is sealed with parafilm and covered with its glass cover and aluminum foil.

Please also refer to our detailed online resources under: http://discotechnologies.org/vDISCO/

Please also refer to our video demonstrations:

https://www.youtube.com/watch?v=rNDxnY4vpSU

https://www.youtube.com/watch?v=wtEjonrw1Ig

Step 1.3: 3D image acquisition with fluorescence light-sheet microscopy

Optional: 2D epifluorescence stereomicroscopy imaging and stitching

The cleared whole mouse body was kept in the original staining chambers with BABB and put under a stereo-fluorescent microscope (Zeiss AxioZoom EMS3/SyCoP3). For whole-body 2D scanning, 1× long working distance (WD) air objective lens (Plan Z 1×, 0.25 NA, WD = 56 mm) was used with a zoom factor of 7×. For high resolution imaging of individual metastasis, higher zoom factor can be applied up to 112x.

1. Put the cleared sample straight and start scanning covering the entire mouse body. Move the sample slowly to prevent any unexpected sample movement and make sure to leave some overlap for image stitching.

2. Save and export the images with 'Merged channels image' mode with RGB arrangement of different imaging channels, e.g. GFP channel (signal from background) with green, RFP channel (signal from 6A10 antibody) with red, and far red channel (signal from nanobody enhanced tumor metastasis) with blue.

3. Stitch the images with Adobe Photoshop CS6 software by using function File/Automate/Photomerge. Select the 'Reposition' function and unselect the 'Blend Images Together'.

4. Adjust the positions of each image to achieve the best stitching outcome and save the stitched image as TIFF or JPEG file.

5. Load the stitched image into ImageJ and extract the signal from each channel by using the function Image/Color/Split Channels. For further details, please see **Figure 3**.

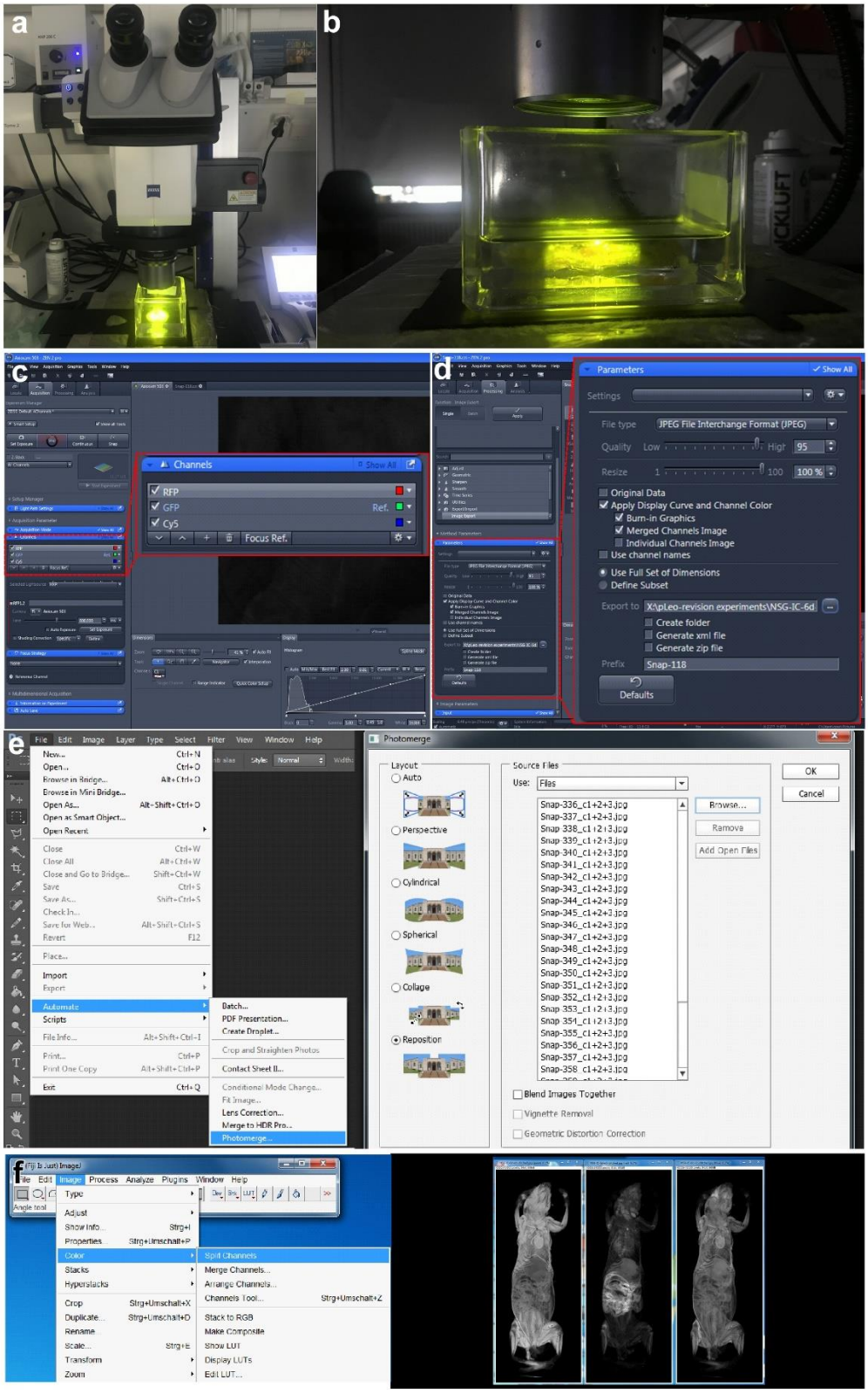

Figure 3: Pipeline for epifluorescence imaging (optional step).
(a-b) *Place the transparent intact mouse body under an epifluorescence stereomicroscope (Zeiss AxioZoom EMS3/SyCoP3).* ***(c)*** *Manually move the sample and image the sample with three channels allocated to the red, green and blue channels in RGB mode.* ***(d)*** *Export the files with 'Merged channels image' mode.* ***(e)*** *Stitch the images with Adobe Photoshop CS6 by using the*

'Photomerge' function. ***(f)*** *After exporting the stitched image, extract the signal from each channel by using the 'Split Channels' function in FIJI.*

3D light-sheet microscopy imaging and stitching

After 2D epifluorescence imaging, 3D light-sheet microscopy scanning of transparent whole mouse body can be carried out. Here, we used Ultramicroscope II (LaVision BioTec) with extended stage movement and a 1.1× NA 0.1 MI PLAN objective (1.1×, 0.1 NA, WD = 17 mm).

1. Mount the sample on the sample holder as indicated in **Figure 4**.

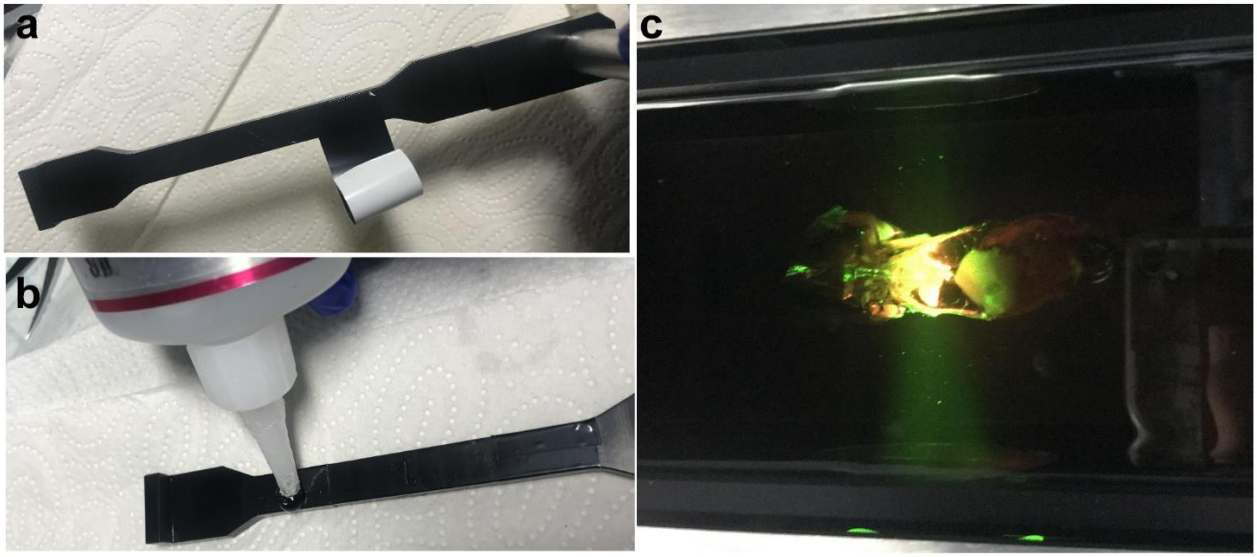

Figure 4: Pipeline for light-sheet microscopy imaging.
(a) *Stick black tape to the sample holder as a base for mounting the sample.* ***(b)*** *Put some drops of glue on the sample holder and hold the cleared whole mouse body until the glue sets.* ***(c)*** *Put the sample into the imaging chamber and start light-sheet scanning.*

2. To cover the entire sample, we scan 3x8 tiles with a 25% overlap. Due to the limited working distance of the objective, we first image the sample from one side and then from the other side. These two separate data set are subsequently stitched by a custom-made macro in FIJI. For details, please see **Figure 5**.

3. Then the series of stitched images are fused together as a whole body data set by using Vision4D (Arivis AG). First, convert and load the images as required and flip the data set imaged from the dorsal side (**Figure 6**). Then, find and set the fusion parameters (the same spots in two different image data sets) by using the Place New Objects/Marker function. After setting three landmarks, use the Fusion function and 10% scale for prototyping (**Figure 7**).

4. After checking that the fusion outcome is appropriate, set the scale back to 100% and start the final fusion process. The final series of the whole-body data set can be exported from the 100% fused data set as TIFF files. Rename the images again and these series of images from the nanobody enhanced tumor metastasis channel or 6A10 antibody channel are ready for deep learning algorithm analysis (**Figure 7**).

Please also refer to our video demonstration: https://www.youtube.com/watch?v=Y7VFAoUo8Fs

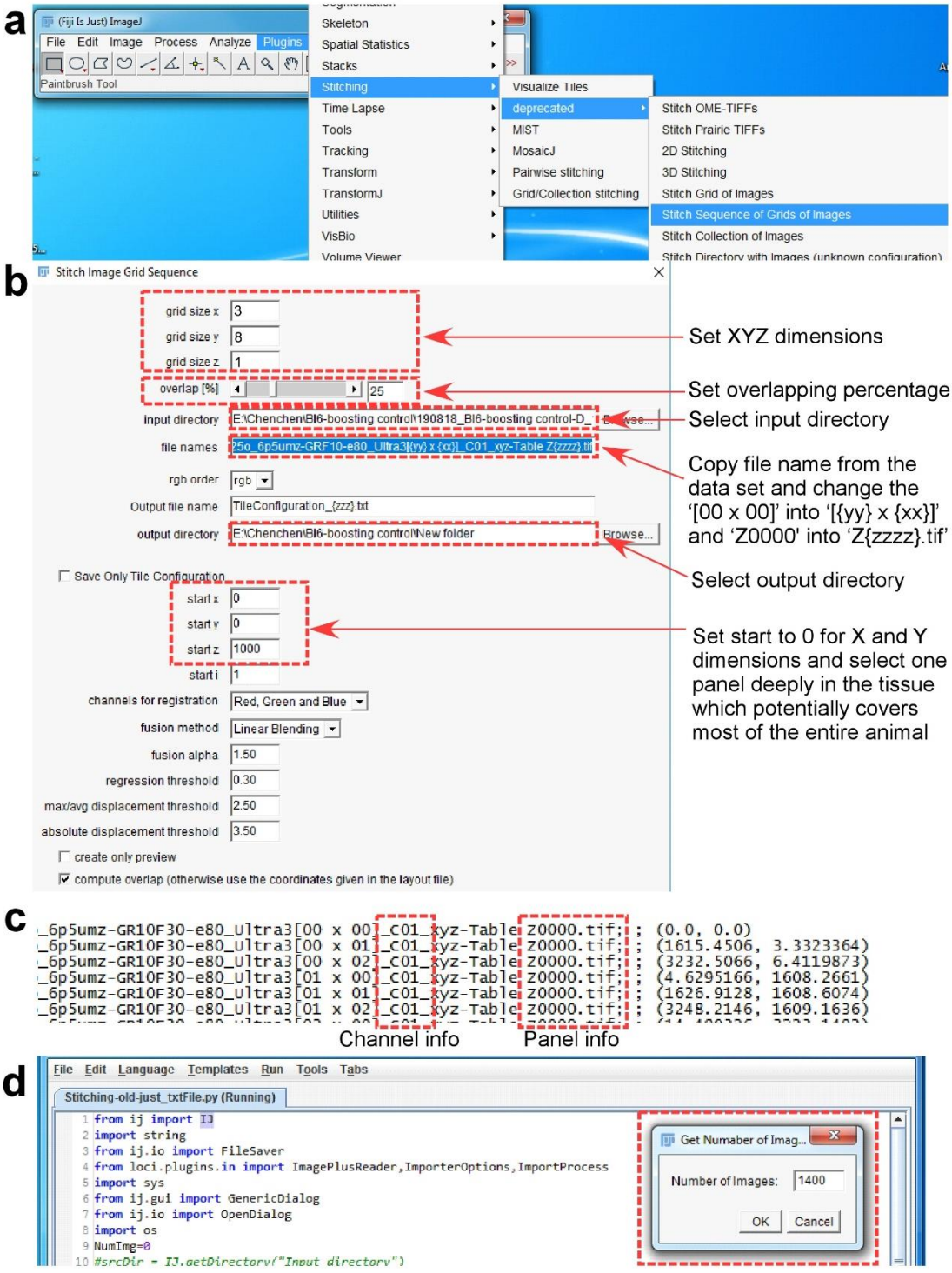

Figure 5: Pipeline for FIJI 2D stitching.
(a) *Open FIJI, go to Plugins/Stitching/deprecated/Stitch sequence of Grids of Images.* ***(b)*** *Select one panel deep in the tissue containing images from all tiles to get the correct parameters for stitching. Set the stitching window as indicated and click 'OK' to proceed.* ***(c)*** *The stitching parameter will be saved in the file 'TileConfiguration_{zzz}.txt.registered' in the input folder. Rename this file, deleting the '.registered' part. Open the renamed file and change the panel numbers for every tile back to 0000. Save the changes and put this txt file into a new folder for stitching the current channel. Change the channel info and place the txt files in the respective folders for stitching the other channels.* ***(d)*** *Load the custom macro and open the txt file containing the stitching parameters. Run the macro and set the panel number. Click 'OK' to proceed and the stitching will be down automatically.*

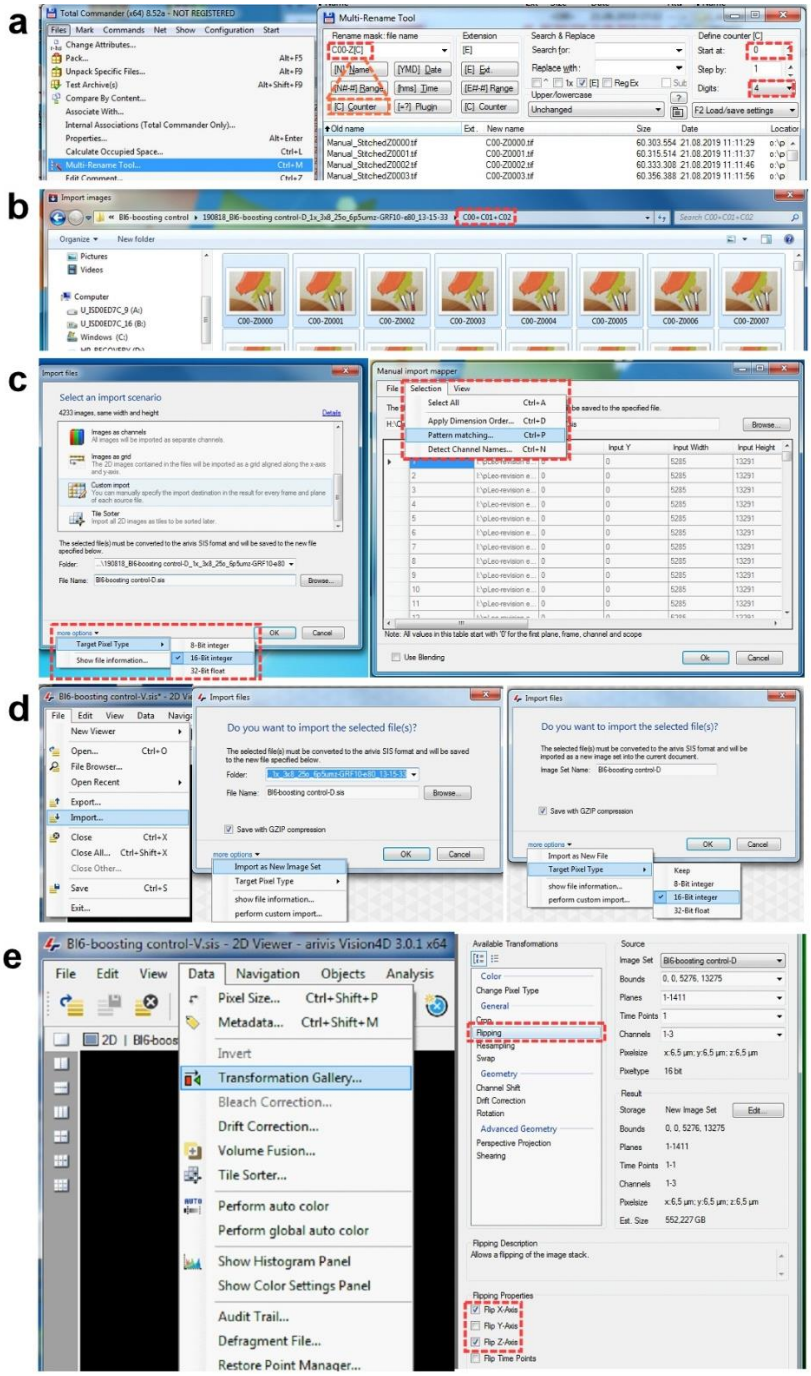

Figure 6: Pipeline for Arivis 3D stitching – part 1

(a) *Rename the FIJI-stitched files in TotalCommander. The file name should contain the information of the channel and panel. Use the 'Counter' function, set the 'Start at' as 0 and 'Digits' as 4.* ***(b)*** *Put all the renamed images from different channels into the same folder and open the Arivis converter. Click 'Add Files', select and open all the images. At this point, a window 'Assume same structure for all files' will appear, click 'Yes' to continue.* ***(c)*** *Select 'Custom import' and match the pixel type of the original images. Then, check the 'Pattern matching' function to make sure that the files will be loaded by the channel names and panel names. Click 'OK' to start the conversion.* ***(d)*** *After converting the data sets from both the ventral and the dorsal sides, open the one from the ventral side and import the other one from the dorsal side. It is important to select 'Import as New Image Set' and match the pixel type of the original images. Then click 'OK' to load the data set from the other side.* ***(e)*** *After setting the correct 'Pixel Size' of both ventral and dorsal data sets, select the data set from the dorsal side and click 'Transformation Gallery'. Select 'Flipping' and choose the properties 'Flip X-Axis' and 'Flip Y-Axis'. Click 'OK' to proceed.*

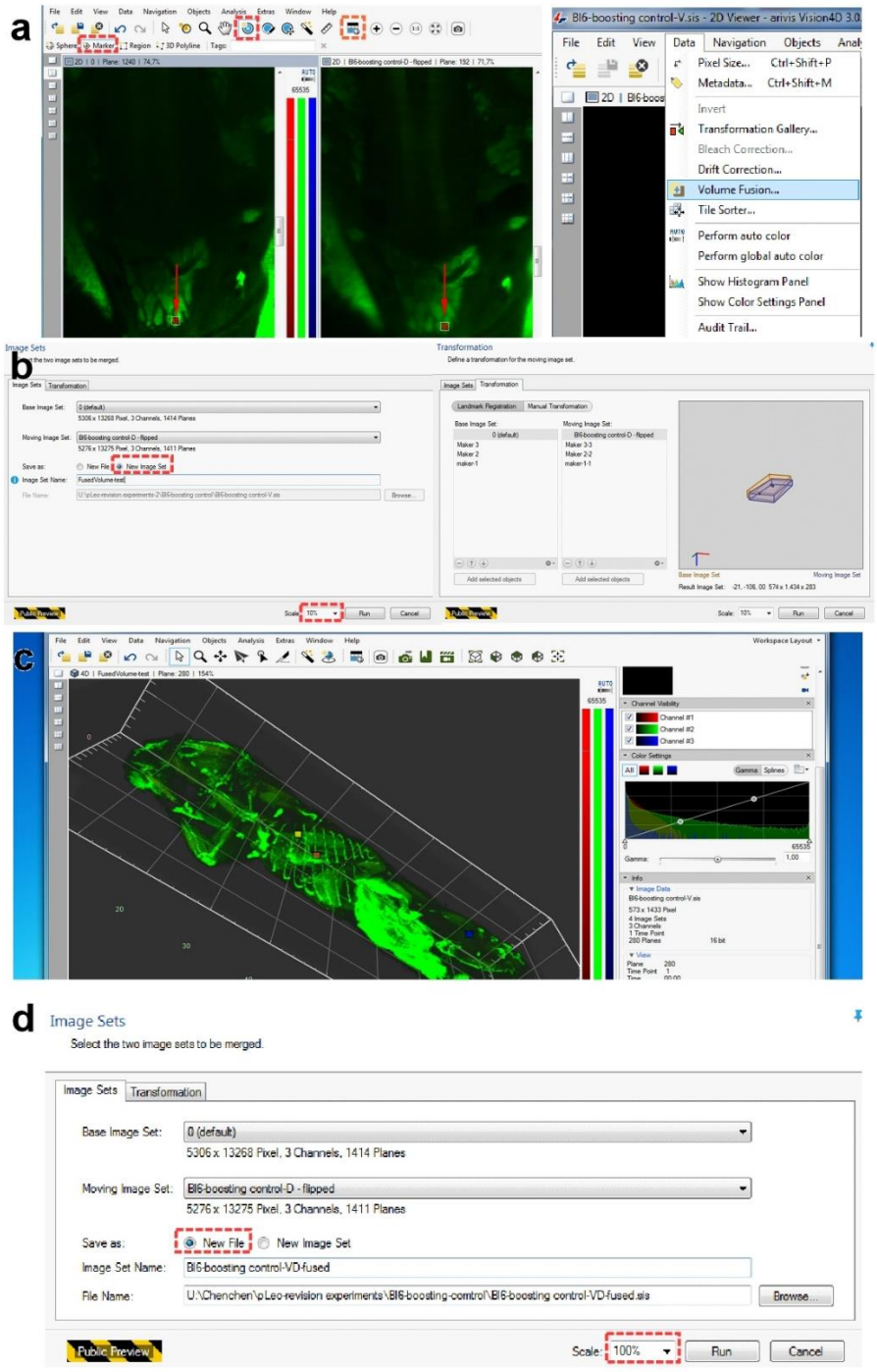

Figure 7: Pipeline for Arivis 3D stitching – part 2.
(a) *Add 'Marker' to the same structure from the ventral data set and flipped dorsal data set and click 'Volume Fusion'.* ***(b)*** *First, set the fused scan as a 'New Image Set' and down scale to 10% for prototyping. Choose 'Landmark registration' for transformation, load and pair the landmarks from two data sets respectively. Then, an illustration of the positions of the fused data set will appear on the right side of the window.* ***(c)*** *Click 'Run' and the data set will be fused automatically. Load the stitched data in 3D mode and the joint landmarks will be indicated by small squares. It is important to check if it is necessary to optimize the fusion parameter at this step. Use the ribs as reference points to make sure that the two separate data sets are fused perfectly.* ***(d)*** *If the fusion parameters are suitable, save as a 'New File' and set the Scale back to 100% to generate the final stitched file. After completing this step, export the file as series of images using 'TIFF Exporter'. After renaming the files with TotalCommander, the images are ready for pre-processing of deep learning algorithm analysis.*

Step 2.1: Data preparation

DeepMACT solves the 3D task of identifying and segmenting metastases in a volumetric scan via the three possible 2D maximum intensity projections (XY, XZ, and YZ). Depending on the scan size, this works best on the level of smaller subvolumes; i.e. the entire volume is not processed at once but subvolume-wise. This serves two purposes: first, one can optimize the size of the subvolumes so that the 2D projections show a useful representation of the data (no excessive overlap of metastases); second, this will also be computationally less expensive, enabling the running of DeepMACT on a normal workstation with a standard GPU. We provide a Python script that automates this step (**CutVolume.py**). This script takes a large volume (a stack of TIFF files; see **Figure 8**, left) and subdivides it into smaller subvolumes (which are saved in the Nifti file format; see **Figure 8**, middle); all parameters, such as subvolume size, can be chosen freely. Please note that no further pre-processing (such as data normalization) is necessary. Subsequently, each subvolume must be projected along the three dimensions, creating 3 views per subvolume (**Figure 8**, right). These projections can be saved to files and the triplets of subvolume projections represent the input to the DeepMACT network.

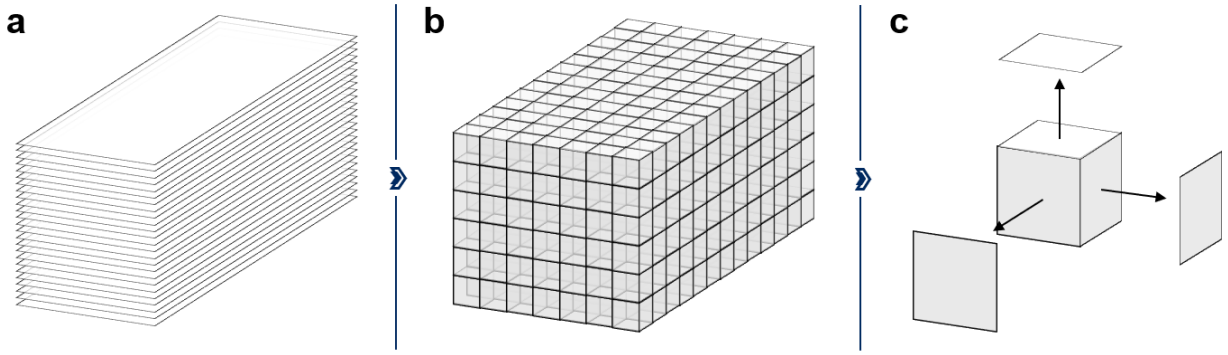

Figure 8: Visualization of different stages of data preparation.
(a) *The 3D stitched scan has been exported as a stack of TIFF files (see the previous step).* ***(b)*** *Running the CutVolume.py Python script divides the volume into a set of subvolumes, which are saved as Nifti files.* ***(c)*** *In a last step, please create 3 orthogonal maximum-intensity-projections per subvolume.*

In our case, we choose to have subvolumes of 350px edge length, which corresponds to 3.5mm or 2.3mm, depending on the magnification used (0.63x and 1.1x, respectively). Also, subvolumes overlap by 50px in all three dimensions (see implementation in CutVolume.py), which is larger than any micro-metastasis we found. Thus, this ensures that any metastasis is fully captured by at least one subvolume to avoid artifacts of divided metastases at subvolume interfaces. Subsequent re-assembly with concatenation rules out double-counting, which must be ensured if subvolumes overlap. Please note that subvolume size and subvolume overlap are design parameters that can be easily adapted to optimize for datasets with different resolutions, metastasis sizes, and signal-to-background ratio. DeepMACT is a fully convolutional network that can process inputs of arbitrary (square) size.

The user may want to consider only presenting those subvolumes to the network that are actually within the mouse body. A full body scan of a mouse will inevitably contain subvolumes outside the mouse as well, which do not contain useful data to analyze.

Step 2.2: Identification and segmentation of micro-metastases with DeepMACT

For convenience, we have provided a fully functional demo version of DeepMACT implemented in PyTorch, which can be downloaded from http://discotechnologies.org/DeepMACT/. This allows DeepMACT users to identify and segment metastases right away to see how it is working. To this end, we have provided data from a full body scan of a tumor-bearing mouse, a pre-trained version of the deep neural network, and a Python demo script. Please refer to the demo script (**demo.py**) for detailed comments on the working mechanisms.

A fully functional online demo of DeepMACT is available on CodeOcean.com (see **Figure 9**). To run DeepMACT online, no other software besides a web browser is needed. This allows the users to get acquainted with DeepMACT easily before downloading and setting it up for their own purposes. You can access the online version here: https://codeocean.com/capsule/8c13691f-7f9a-4af4-8522-c26f581c9e83/tree?ID=a8ba18d2bf5046b08fafe2d6a42bfd7a

Please note that you need to create an account with Code Ocean in order to be able to run the demo.

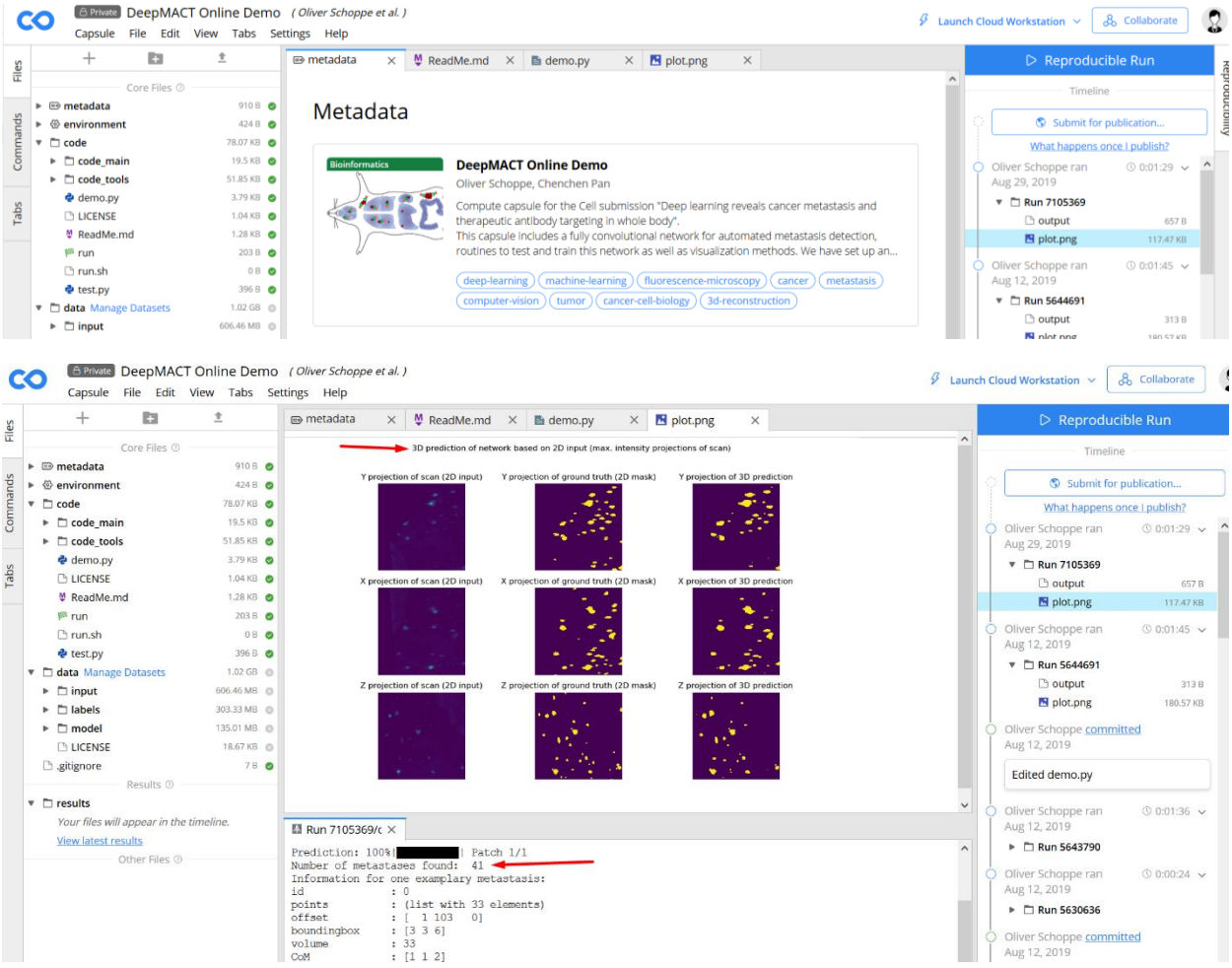

Figure 9: Online demonstration of DeepMACT
Besides downloading all code and data from discotechnologies.org, the user can also choose to run a fully functional online version of DeepMACT on CodeOcean.com. Running DeepMACT on the data provided yields predictions of 3D segmentations of metastases (upper red arrow) as well as an explicit list of the individual metastases that were found (lower red arrow). To run DeepMACT online, no other software besides a web browser is needed. This allows the users to get acquainted with DeepMACT easily before downloading and setting it up for their own purposes.

Please note that this demo is not restricted to the data set we have provided. The user may directly use it to run predictions on their own data as the algorithm is already trained. Thus, the user can use DeepMACT right away without the need to train the model (i.e. no manual data labeling is needed). To do this, just replace the demo data with the custom data and feed it to DeepMACT as shown in the demo. For a given volumetric scan, the data should be provided as three 2D maximum intensity projections (one projection along each of the three possible axes). An example implementation of such a data loader is provided in **data_functions.py** → volumetric_loader(). Feeding such data to the predict_3D() function of the DeepMACT model will return a binary segmentation, as well as a list of blobs that were identified as metastases.

```
# Load triples of projections from list of IDs of subvolumes
testLoader = data_functions.volumetric_loader(settings, ListOfSubvolumeIDs)
# Output: binary 3D volumes & list of metastases
binary_volumes, metastasis_list = model.predict_3D(testLoader)
```

In our data, we found that DeepMACT had a substantially higher detection rate of micrometastases than a human annotator (82% versus 71%). However, the prediction may also contain false positives. We thus recommend to further refine the results with the help of visual inspection (see the demo.py file for a possible implementation of the visualization). We typically reviewed in all 100-1000 metastasis candidates within 30-60 minutes for a whole-body scan of a mouse with high tumor burden. If desired, the user can also run DeepMACT with a higher sensitivity (by reducing the default threshold in the settings dictionary from 50% to lower values) to further increase the detection rate. Please note that even without manual review we found DeepMACT to be on a par with a human annotator in terms of F1 score, which quantifies prediction performance in terms of detection rate and false-positive rate.

Please take a look at the appendix in chapter C) to see instructions on how to modify DeepMACT if desired. For instance, the users can retrain the model to their own data or also replace the core network by an alternative version with 3D convolutions that works directly on volumetric data instead of projections.

Step 2.3: Metastasis-level analyses

The output of DeepMACT comprises binary segmentation volumes as well as an explicit list of metastases. For each identified metastasis, DeepMACT provides a dictionary data structure with pre-computed information such as a unique ID, its 3D position within the subvolume (in terms of a bounding box as well as the coordinates of its center of mass), its volume (count of voxels), and a characterization of its shape. This serves as a basis for any further analysis on the level of individual metastases the user may want to perform.

```
1 # Visualization of a typical dictionary for a detected metastasis
2 print(metastasis_list[0]) # Print first metastasis in list
OUT
> metastasis_id   : 0
> list_of_points  : (list with 150 elements)
> offset_in_subvol: [  4 234 184]
> boundingbox     : [ 6  6 11]
> volume_vx       : 150
> CenterOfMass    : [2 2 5]
> max_diameter_vx : 10.2
> characterization:
>  |-> compactness: 0.27
>  |-> sphereness : 0.65
>  |-> stringness : 0.35
>  |-> skewness   : 0.06
> subvolume_id    : 2982
```

For our study, we chose to compute further parameters such as the Euclidian distance to the nearest metastasis. Furthermore, we segmented major organs of interest such as the kidneys, lungs, brain, or liver in 3D using the Fiji ROI manager. This enables, for each individual metastasis, to automatically determine whether it is located in one of these organs by checking whether its 3D location falls into the 3D segmentation of the organs. Also, you may want to analyze the 3D distribution of any other fluorescent signals acquired during the scan. For instance, we traced the distribution of therapeutic antibodies with a voxel-to-voxel correspondence in an independent acquisition channel. In combination with the binary 3D segmentations provided by DeepMACT, this allowed us to quantify the distribution of therapeutic antibody within the metastatic tissue and in each metastasis' local environment – enabling the identification of metastases the antibody targeted successfully, as well those that it failed to target.

C) Appendix

This section provides step-by-step instructions for adapting the DeepMACT approach to fit custom needs. Again, please note that it may not be necessary to adapt anything in order to run DeepMACT on the user's data (see above). If the data looks similar to the data presented in our publication, the user may use the pre-trained DeepMACT network we provided directly. However, DeepMACT can also be applied to very different data. In short, the user can choose to (re-)train a network on own labels and/or even use a different architecture for the network. We provide a set of tools that allow the user to

a) pre-process the data
b) automatically label the data based on filter-based detectors
c) refine these labels manually using an interactive user interface.

Furthermore, we provide a standard architecture for training and explain how the user can train this or other networks on their own data.

Hard- and software requirements for DeepMACT

DeepMACT is stand-alone, easy to implement, does not require any proprietary software, and can be run on a normal workstation. No costly cloud computations are needed.

From a hardware perspective, we recommend a workstation with 32-64 GB RAM and a GPU with 12GB memory (consider the NVIDIA GPU Grant program, which sponsors GPUs for researchers). Also, it may be convenient to use an SSD instead of a hard drive for faster processing through reduced loading times.

From a software perspective, DeepMACT is based on Python 3 and PyTorch; it can be run on any operating system. To utilize the GPU (instead of CPU) for training or prediction, please make sure that the right NVIDIA driver for your GPU as well as Cuda and CuDNN are installed. Beyond that, DeepMACT makes use of a few standard Python libraries such as Numpy or OpenCV, as listed in the individual Python scripts. All this software if freely available online.

Create your own training data (optional)

Automatic pre-labeling (optional).

As described in detail in the publication, we built a custom filter-based metastasis detector. This not only serves as a baseline for performance comparison but may also be used as a tool to automatically create preliminary labels for your data. In short, we define a spatial filter kernel that resembles the typical blob-like shape and size of metastases to identify and locate potential metastases (see left panel of Figure 10). In a next step, we try to segment the metastases around their identified locations using dynamic region growing (middle panel). Together, this may already provide a first draft of annotations (right panel). We provide this tool (**CreateFilterbasedSegmentations.py**) to enable efficient labeling of custom data. Please consider adjusting the definition of the spatial filter kernel and other parameters to suit your data. The task solved by this tool is computationally expensive; to speed up, this tool thus runs on the level of subvolumes (see above) and supports parallel processing of several subvolumes at once on multi-core CPUs.

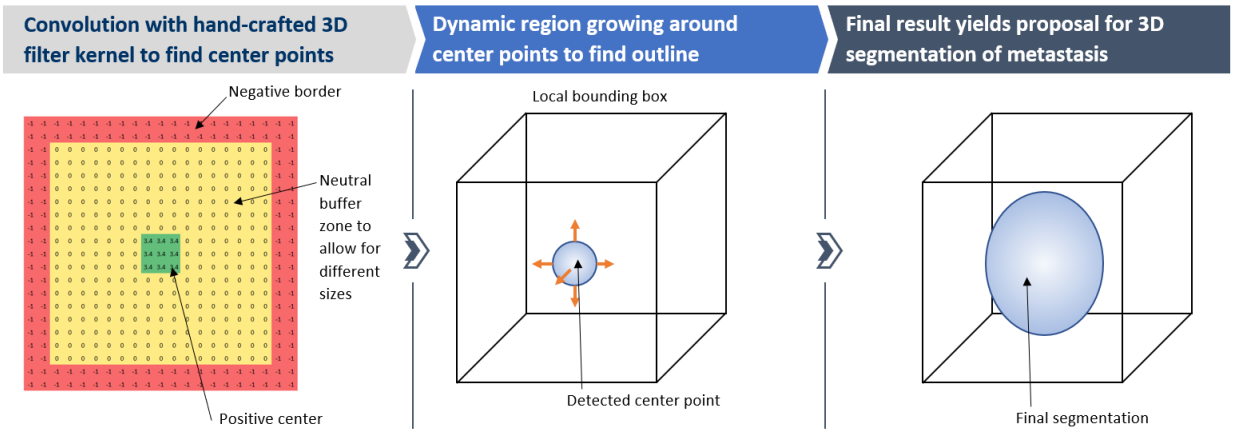

Figure 10: Schematic of filter-based detection with subsequent local region growing.
The filter-based segmentation is an optional step to create labels for additional training data. First, the scan volume is convolved with a hand-crafted 3D filter kernel sensitive to blob-like signal peaks (left panel). Subsequent binarization yields center points of potential metastases. In a next step (middle panel), dynamic region growing is performed to find the final outline of the metastasis candidate around the previously detected center point(s). The region is grown in order to include all voxels with a signal strength 4 standard deviations above the mean signal within the local bounding box. Please note that this step is optional and not required to generate additional training data for DeepMACT, as it can be done manually as well (see the next step).

Manual creation/refinement of labels. Automatic, filter-based segmentations miss many metastases and furthermore also contain a substantial number of false positives. Thus, manual refinement is essential. Screening large volumes for small metastases and segmenting them would usually require manual analysis of each slice of the volume, a task that we estimated would take several months or more. Thus, we created an interactive labeling software (**TumorLabelingTool.py**) that substantially speeds up this task. After having created automatic labels (see above), this tool will allow you to iterate over all subvolumes of the data set. For each subvolume, the user can remove false positives and add previously overlooked metastasis with a single click. The user can analyze unclear cases with the help of various visualization methods. The size of the segmentation can be increased or decreased as desired and on a case-by-case basis. Furthermore, the user can flag special subvolumes and/or special metastases within a subvolume for later analysis. Also, the tool tracks the name of the annotator and each individual action for later meta-analysis (e.g., to compare annotation variability among a group of experts). A **detailed manual for the TumorLabelingTool** and all its functionalities is provided in a separate document, which is part of the download folder provided online. Please note that the tool (with the help of the manual) can be operated by anyone capable of interpreting the data and does not require any coding skills as all user interaction takes place via a graphical user interface (see **Figure 11**).

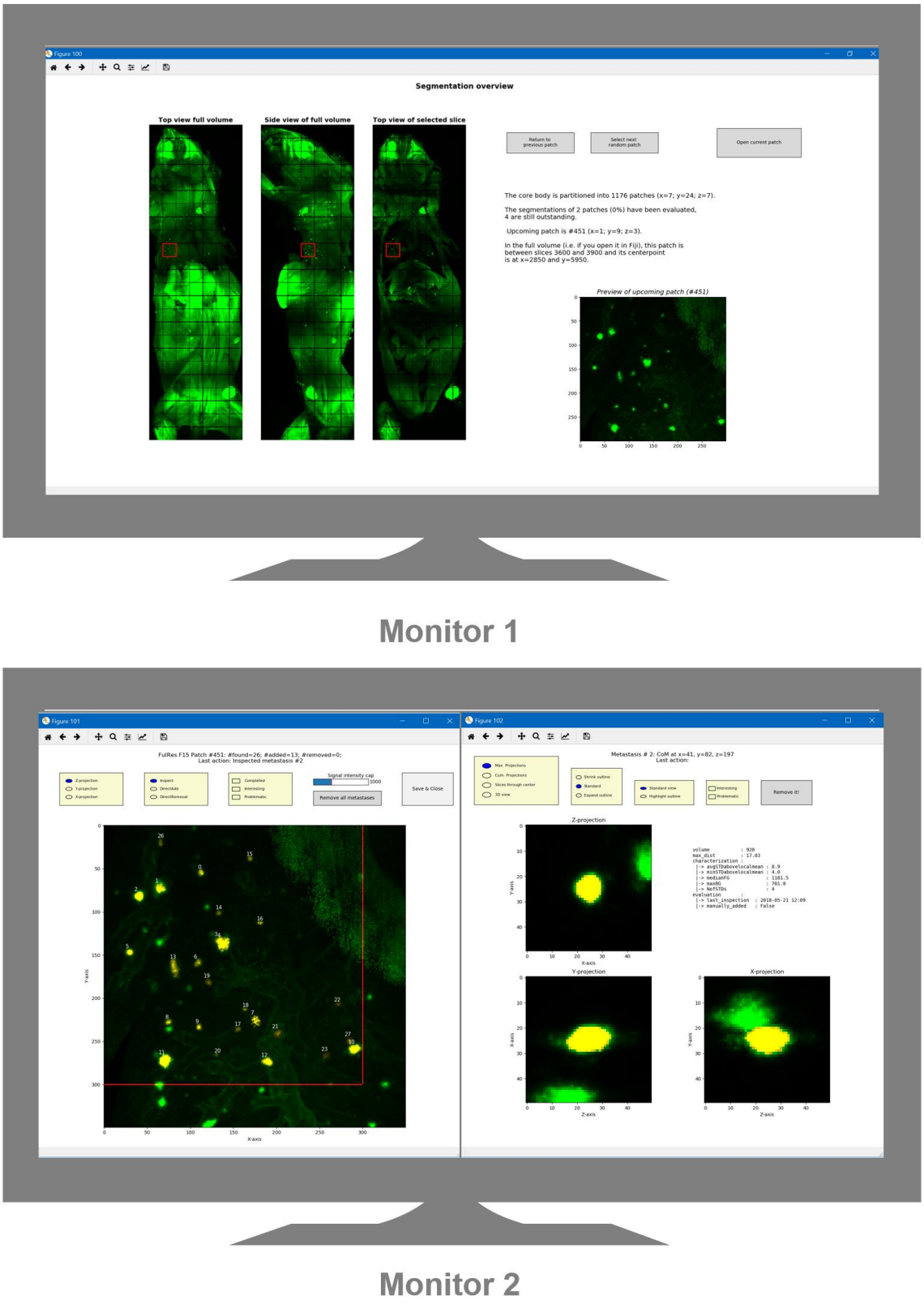

Figure 11: User interface of the optional Tumor Labeling Tool.
This tool can be used to annotate additional training data in a fast and efficient manner, if desired by the user. Please refer to the Tumor Labeling Tool Manual (separate document) for detailed instructions. Please note that creating additional training data is optional and not required to run DeepMACT.

Converting data sets for DeepMACT. Once the raw data is cut into subvolumes and each subvolume has been manually segmented, it is ready to be converted into training data for the algorithm. For every subvolume, please create three 2D maximum intensity projections of the scan and three 2D maximum intensity projections of the segmentation, yielding 6 files per subvolume. If the user does not want to train an algorithm but just wants to segment their data set using the pre-trained version of DeepMACT we provided, the user can skip the labeling steps above and just create three projections of the scan.

Training DeepMACT on the user's own data (optional)

Again, please note that we already provide a trained version of DeepMACT. Thus, training is not needed to run DeepMACT. However, if the user wants to train a network to custom data, here is how to do it: first, define an architecture for a 2D segmentation task. We use a modified version of U-net and provide an implementation of this architecture for your convenience (**code_main/architecture.py**). Second, define a PyTorch data loader that would read in the pairs of 2D projections (scan and segmentation). (The 3 projections per subvolume can be treated as independent training samples for training purposes. We recommend training a single network on all data, rather than training independent networks for each projection axis, unless the imaging method used yields highly un-isotropic data.) We have provided demo data loaders in **code_main/data_functions.py**. As a loss function, we recommend weighted binary cross-entropy, wherein the weights account for the class imbalance (more background than foreground) and may be learned as a hyper-parameter. We have provided an implementation in **code_main/helper_functions.py**. Then, define a training schedule for this 2D segmentation task and train the network on all available training data. We recommend the Adam optimizer with a learning rate scheduler that reduces the learning rate on plateaus (see **code_main/model_builder.py**). For reference: we found 50-100 epochs sufficient for a training set of about 3x1000 pairs of 2D projections (300x300 pixels) but this may depend on the data. This may last around 45 minutes on a Titan XP GPU. We recommend tuning/learning all hyper-parameters such as the learning rate via (k-fold) cross-validation (see code example below and also please refer to the demo script).

```
# Train a model using cross-validation for 50 epochs using 2D projections
trainLoader = data_functions.projection_loader(settings, Subvolume_IDs_Train_Set)
validationLoader = data_functions.projection_loader(settings, Subvolume_IDs_Val_Set)
for epoch in range(0,50):
    model.net.train() # switch to training mode
    training_loss = model.train(trainLoader)
    model.net.eval() # switch to evaluation mode
    validation_loss = model.validate(validationLoader)
    model.lr_scheduler.step(validation_loss, epoch) # Reduce learning rate, if needed
# After training in 2D, retrieve a 3D prediction on a 3D volume
testLoader = data_functions.volumetric_loader(settings, Subvolume_IDs_Test_Set)
binary_volumes, metastasis_list = model.predict_3D(testLoader)
```

The training and validation function itself are trivial as it can follow any standard implementation suitable for 2D binary segmentation. The data loader functions and the three-dimensional prediction

function is of course already pre-implemented and provided in the **code_main/model_builder.py** file.

Remarks on training in 2D versus prediction in 3D. DeepMACT solves the 3D task of localization and segmentation via 2D projections. This is beneficial for several reasons, a few of which we want to list here for better understanding of the approach. First, it only needs 2D training data, which is exponentially faster and easier to generate by human annotators. Thus, if the user generates 2D training data with a software of choice, this is sufficient for DeepMACT (our "TumorLabelingTool" creates 3D training data for the user's convenience; DeepMACT is trained on 2D projections of these data). Second, processing large volumes in 3D would be exponentially more time-consuming and would require much more powerful hardware. Exploiting 2D maximum intensity projections is much faster and less resource-intensive, enabling the user to run DeepMACT on standard workstations. Third, in our comparisons it yields better results compared to 3D processing. This may be counter-intuitive, but the following considerations help to can provide an explanation of why this is the case. Training a 3D network requires substantially more training data, which is typically not available. Training a 2D network with a given annotated data set will converge much faster and at a lower final loss. Thus, training is happening in 2D and only in 2D. However, taking the segmentation results from the 3 projections of a given subvolume allows DeepMACT to recombine this information in 3D space, which yields a 3D segmentation for all metastases. Please note that this approach is robust even if any metastases may overlap in a given projection, as this overlap can be resolved with the help of the other two projections. In summary, DeepMACT is trained in 2D but can locate and segment in 3D reliably.

Alternative: Replace DeepMACT with 3D convolutional network (optional)

Please note that, as mentioned above, we generally find DeepMACT to be superior in terms of speed as well as prediction performance over 3D networks. However, there may be cases in which the user may wish to perform 3D convolutions on volumetric data instead of working with 2D maximum intensity projections – for instance, if the density of micro-metastases is in the order of hundreds per 1 mm^3. For such cases, we provide two possible implementations with corresponding pre-trained models (see **Figures 12 and 13**). The corresponding files for 3D networks can be found in the folder "alternatives" of the demo package, which is available for download at http://discotechnologies.org/DeepMACT.

3D alternative #1:

This architecture closely resembles the architecture of DeepMACT but replaces 2D with 3D convolutions. This heavily increases model complexity and increases the number of network parameters by a factor of about 4. In order to be able to train it / run it on a standard 12GB GPU, we removed the deepest layer and thus have 5 en- and de-coding units (instead of 6) with a maximum number of feature channels of 512 (instead of 768; see **Figure 12**). This model has a very high capacity but consequently also needs a substantial amount of training data to be trained successfully.

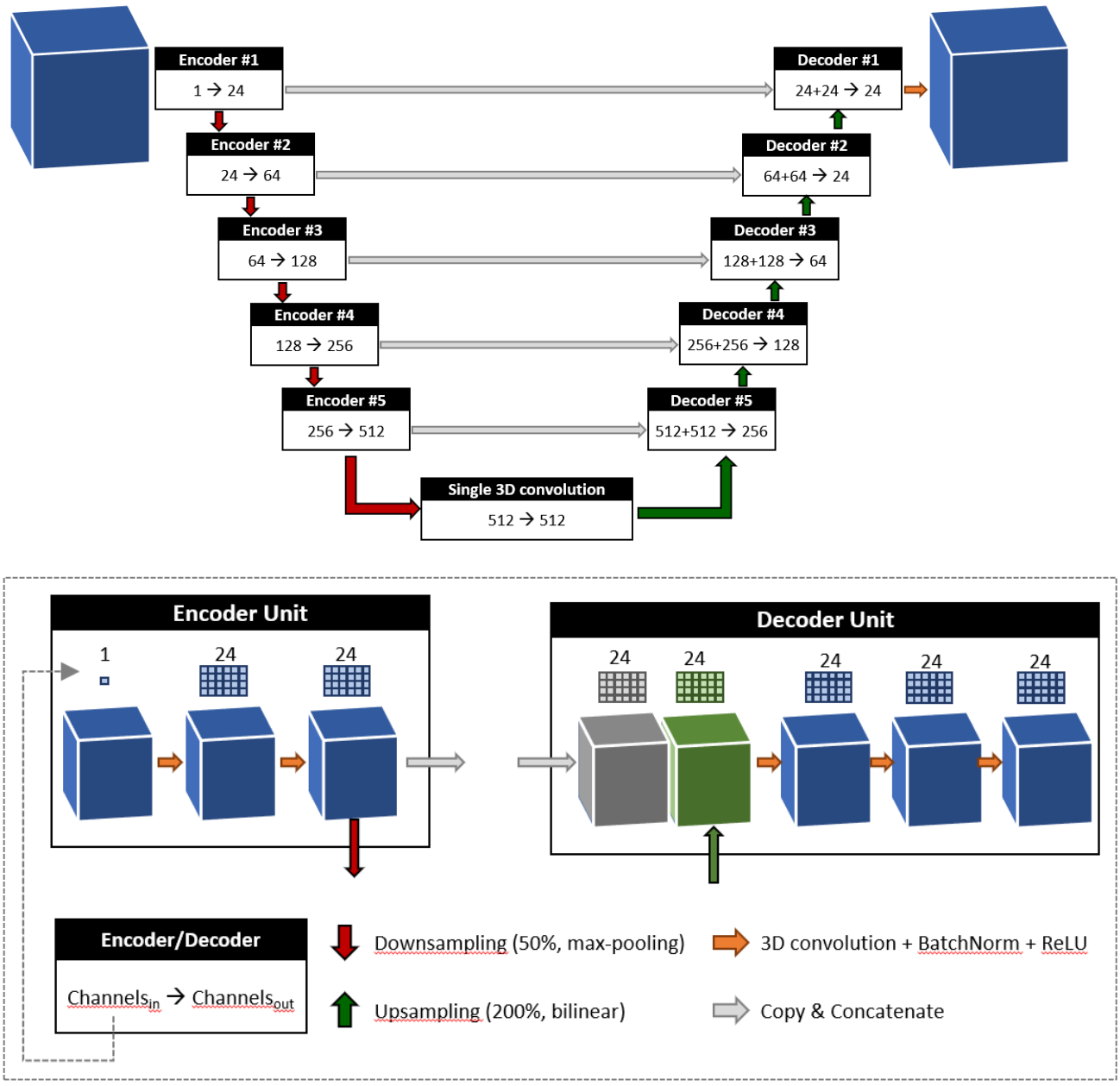

Figure 12: Architecture of a possible 3D U-net implementation.
This architecture is a 3D version of U-net and structurally very similar to the network architecture of our DeepMACT pipeline. The most important difference is the switch to 3D. The input and output of the network is a 3D subvolume; consequently, all convolutional steps (orange arrows; see the legend in the dashed box for details) operate fully in 3D. Please note that this network architecture has 5 instead of 6 encoding/decoding layers, which is necessary to meet memory requirements of standard GPUs (such as Nvidia Titan Xp). The boxes above the volumes in the legend visualize the number of feature channels for the input and output of each encoding and decoding unit.

3D alternative #2:

We also provide a leaner 3D version. For this second alternative, we further trimmed the network to 3 en- and de-coding units with a maximum number of feature channels of 48. This network thus requires less training data but may not have the capacity to learn more complex tasks.

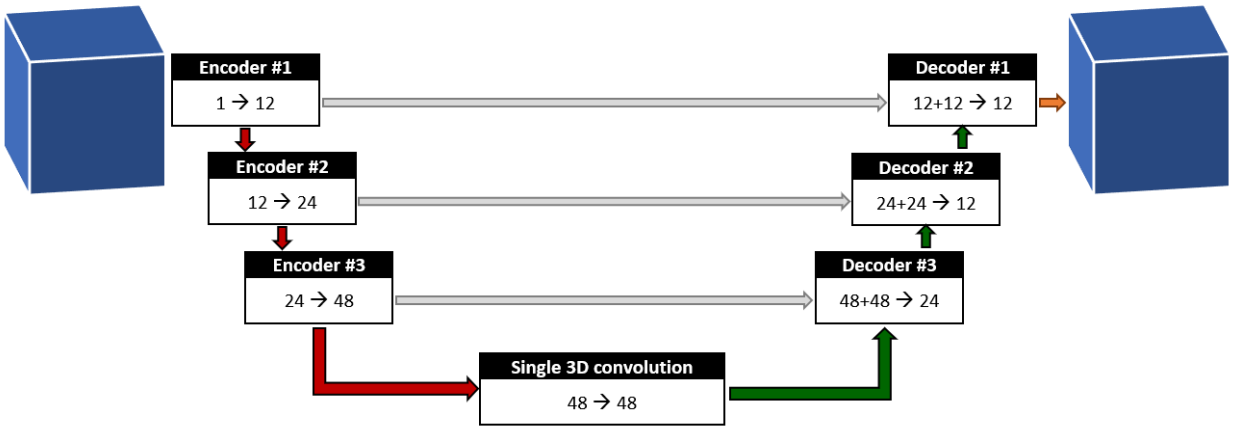

Figure 13: Alternative implementation of a 3D U-net
*This alternative architecture is substantially less complex and thus, has less parameters than alternative #1 (see **Figure 12**). Other than the reduced complexity, the working mechanisms are identical and the annotations are analogous to those in **Figure 12**.*

Volumetric data loader

Since both 3D versions do not work with projections but directly on volumetric data, we also provide a modified PyTorch data loader implementation that directly feeds 3D subvolumes to the network instead of triplets of projections. The data loader is part of the folder "alternatives" of the demo package, which is available for download at http://discotechnologies.org/DeepMACT.