Instructions for applying CODA

Last updated by Ashley Kiemen, December 2023

This document gives basic usage instructions to create 3D tissue and cellular labelled datasets from serial histological images using CODA, as described in Kiemen et al, “CODA: quantitative 3D reconstruction of large tissues at cellular resolution”, (2022), *Nature Methods*.

You will need the following programs installed:

MATLAB (including toolboxes such as the image processing toolbox, deep learning toolbox, and the resnet50 model**)**, **Aperio ImageScope**, **FIJI ImageJ**

Codes are available at the following GitHub:

https://github.com/ashleylk/CODA/tree/main/update%2012-13-2023

Here, we discuss application to a sample dataset “lungs”, containing 150 serial histological images. Download the sample dataset (serial images and sample annotations) here:

https://drive.google.com/drive/folders/1K-wY_ArVGbEhebQD4AjOeERwx6-4Fw3G?usp=sharing

Images are .ndpi format and were scanned at 20x magnification (approximately 0.5 micron / pixel resolution), spaced 10 micron apart. The images are saved in a hypothetical folder:

```
pth='\\Users\Ashley\Documents\lungs';
```

Sections: (for the most part, these must be followed in order)

Section 1. Notes on Filenames

Filenames for each image should be created such that tissue sections are read consecutively by Matlab. Therefore, include zero-padding in numerical indices.

CORRECT FILENAMES:

lungs_001.ndpi

lungs_003.ndpi

…

lungs_011.ndpi

INCORRECT FILENAMES (no zero padding):

lungs_1.ndpi

lungs_3.ndpi

…

lungs_11.ndpi

Section 2. Create downsampled copies of high-resolution images

Create downsampled tif images from the high-resolution .ndpi files. There are several ways to do this.

1. The function **create_downsampled_tif_images** will create downsampled copies of the .ndpi files by directly loading each high-resolution images in tiles and down sampling it to the desired pixel resolutions.

   First, decide the resolution of the images you want to create. Here, we create images of 1 micron / pixel, 2 microns / pixel, and 10 micron / pixel resolution:

   ```
   ds=[1 2 10];
   ```

   Next, decide on the name of the output folders for each of the downsampled images you create. Here, we will save the images downsampled to 1 micron / pixel in a folder named "10x", the images downsampled to 2 micron / pixel in a folder named "5x," and the images downsampled to 10 micron / pixel in a folder named "1x."

   ```
   subfolders=["10x" "5x" "1x"];
   ```

   Finally, call the function:

   ```
   create_downsampled_tif_images(pth,ds,subfolders);
   ```

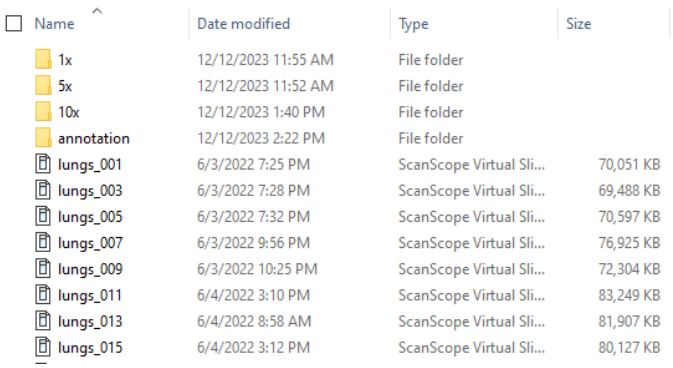
   Using this function, you will make two subfolders within the original folder containing the .ndpi images. One subfolder named "10x" containing the 20x images downsampled by a factor of 2. The other subfolder named "1x" containing the 20x images downsampled by a factor of 20. Most calculations will be performed on these tif images. Note: here we use 10x and 1x for example, but other resolutions could be created as desired.

   ```
   pth10x=[pth,'10x'];

   pth1x=[pth,'1x'];
   ```

   ****Note: If this code fails due to memory constraints on your computer, try python Openslide.**

   Sample folders in Windows explorer:



Name	Date modified	Type	Size
1x	12/12/2023 11:55 AM	File folder	
5x	12/12/2023 11:52 AM	File folder	
10x	12/12/2023 1:40 PM	File folder	
annotation	12/12/2023 2:22 PM	File folder	
lungs_001	6/3/2022 7:25 PM	ScanScope Virtual Sli...	70,051 KB
lungs_003	6/3/2022 7:28 PM	ScanScope Virtual Sli...	69,488 KB
lungs_005	6/3/2022 7:32 PM	ScanScope Virtual Sli...	70,597 KB
lungs_007	6/3/2022 9:56 PM	ScanScope Virtual Sli...	76,925 KB
lungs_009	6/3/2022 10:25 PM	ScanScope Virtual Sli...	72,304 KB
lungs_011	6/4/2022 3:10 PM	ScanScope Virtual Sli...	83,249 KB
lungs_013	6/4/2022 8:58 AM	ScanScope Virtual Sli...	81,907 KB
lungs_015	6/4/2022 3:12 PM	ScanScope Virtual Sli...	80,127 KB

Section 3. Calculate registration on low-resolution tissue images

Calculate nonlinear image registration on the low-resolution tif images. For most images, a resolution of 1x should be sufficient. If the 1x results are not well registered, or registration of small tissues (such as needle biopsies) is desired, try calculating registration at a higher resolution such as 2x or 4x.

1. The quality of the registration depends on first calculating the tissue space in each image, as this allows removal of noise (shadows and lines that aren't tissue within the whole slide image). The presence of noise in the images during registration can affect the codes determination of registration accuracy (QC step), as this is determined through pixel-to-pixel cross correlation. So first, create logical images indicating locations containing tissue with the function **calculate_tissue_ws**. This function has two methods of calculating the tissue space. First try with method 1:

   calc_style=1;

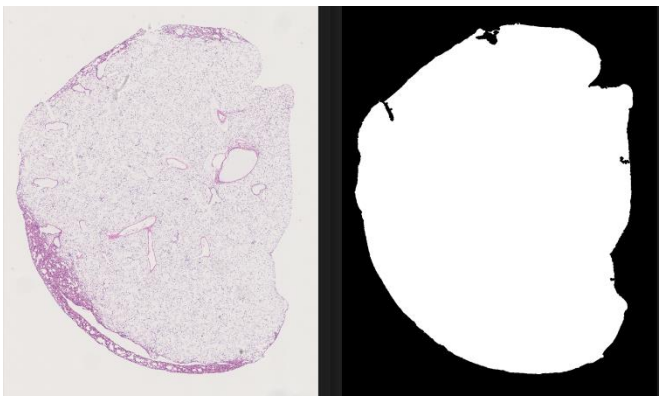
   calculate_tissue_ws(pth1x, calc_style);

   This function will create logical images with the same filenames as the images in pth1x, saved in a subfolder named 'TA'. Check that these tissue images look correct (are not white over background, nontissue space in the image, only white on tissue regions). If they do not look correct, delete the images inside the folder named 'TA' and try again, using method 2:

   calc_style=2;

   calculate_tissue_ws(pth1x, calc_style);

   Sample H&E images with a corresponding 'good' TA image:



Next, apply the CODA nonlinear image registration to the low-resolution tif images. Note, this function can be applied without first creating the tissue masks with **calculate_tissue_ws.** The registration code will by default use method 1 to create the tissue masks. But it is best practice to first calculate the tissue masks and validate that they are accurate to ensure the best registration quality is achieved.

calculate_image_registration(pth1x);

This function will create output folders containing the globally registered images, the elastically registered images, and the registration transformation metadata:

pth1xG=[pth1x,'registered'];

pth1xE=[pth1x,'registered\elastic registration'];

pthdata=[pth1x,'registered\elastic registration\save_warps\';

Validate the image registration by loading the elastically aligned, downsampled z-stack in ImageJ. These validation images will be saved in a subfolder within the folder containing the low-resolution tif images:

pth_validate_regisration=[pth1x,'registered\elastic_registration\check'];

Notes on debugging registration: If your results are bad and you would like to re-register with different parameters, first delete the 'registered' folder (inside pth1x) created by **calculate_image_registration**. If you do not delete this subfolder the registration code will skip all images that were previously registered.

- If the elastically registered images are too jiggly, try reducing szE and/or diE inside **calculate_image_registration**
- If the elastically registered images are too smeared, try increasing szE and/or diE inside **calculate_image_registration**
- If the registration is taking too long for one image (>5 min), try reducing the resolution of the images, reduce the szE or diE inside **calculate_image_registration** and/or try a computer with higher RAM

Section 4. Deep learning multi-labelling of tissue structures using training on manual annotations

In this section, we describe steps to create a basic semantic segmentation algorithm using CODA. CODA uses a modified resnet50 network adapted for semantic segmentation using an implementation of DeepLab. Here, we describe how to generate training datasets of manual annotations for a set of images, how to format the data for deep learning, and how to train and apply a CODA deep learning model.

1. First, choose the biological structures you wish to segment in your images. Semantic segmentation algorithms must classify every pixel of every image with a label, so your list must be exhaustive. For example, in lung histology you could choose to annotate:

    a. Bronchioles

    b. Alveoli

    c. Vasculature

    d. Metastases

    e. Nonexpanded lung

    f. Background

    g. Stroma

    Here, populations like fibroblasts and immune cells may be annotated inside of the 'stroma' layer. Background can encompass nontissue space as well as non-target noise within the histological images including red blood cells, shadows, and dust visible in the scanned images.

2. Next, select some original resolution (.ndpi or .svs) images to annotate. Try first choosing 7 training images and 1 testing image, then add images as necessary until your model performs acceptably (>90% quantitative accuracy + passes visual inspection) on an independent (not seen in training) testing image. Put images you wish to annotate in a separate folder named 'pthannotations':

    pthannotations=[pth,'annotations']; % put images here to annotate for training

    Inside of pthannotations, create a subfolder named pthim where you copy a corresponding high-resolution (here 10x) tif image of each image that you are annotating.

    pthim=[pthannotations,'10x']; % copy a tif image here for each training image

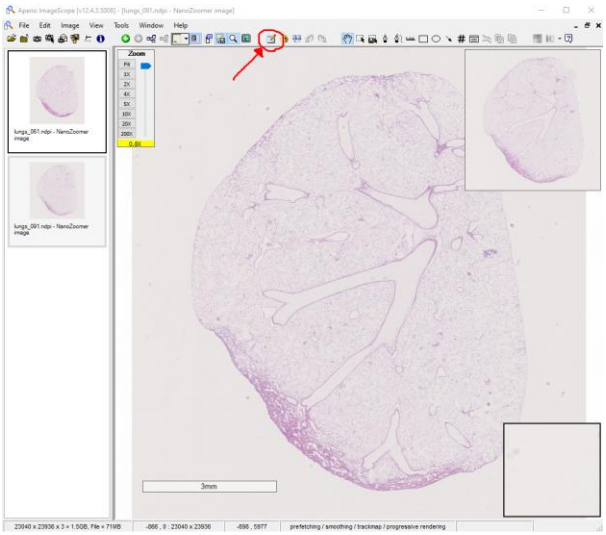
    Notes on the resolution you choose to train your model at: This choice highly depends on the resolution of structures you wish to label. If you want to quantify 'bulk' structures, low resolution (5x or lower) should be sufficient. If you want small structures such as small vasculature, small tubules, a medium resolution (~10x) should be sufficient. If you want to label individual cellular-sized strucutres, try ≥20x (you will need a computer with very high RAM and will have to very tediously annotate for this).

3. Inside of pthannotations, copy one additional 20x (.ndpi or .svs) image to a subfolder named 'pthtest,' with a corresponding high-resolution tif image saved in 'pthtestim'.

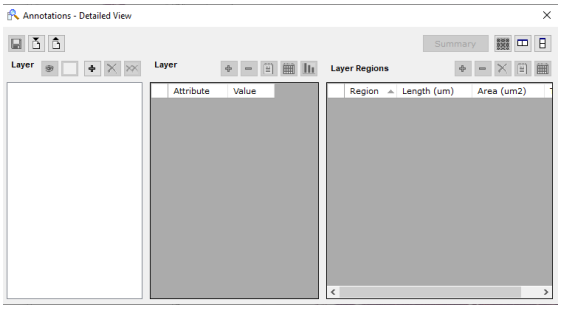
    pthannotations_test=[pthannotations,'testing image']; % put image here to annotate for testing

    pthtestim=[pthannotations_test,'10x']; % copy a tif image here for each testing image

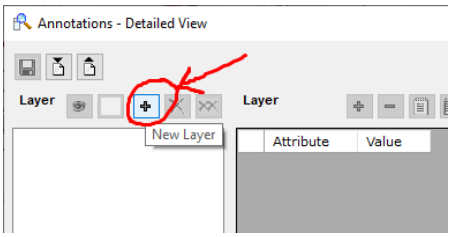
4. For each image you wish to annotate, open the file in **Aperio ImageScope**. To generate the xml files that will contain annotation data, select the Annotations button:



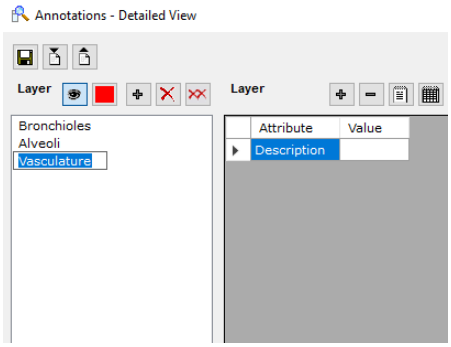
This will open the Annotations window:



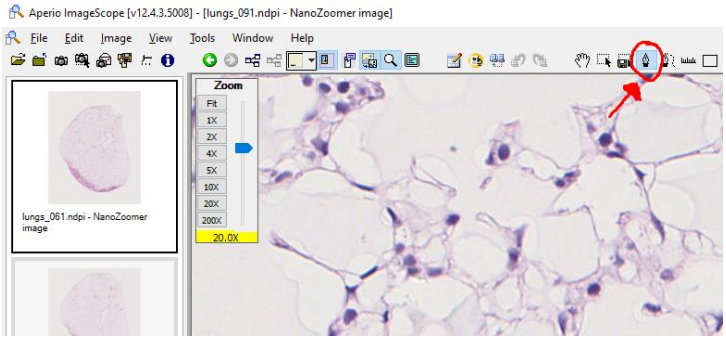
Generate annotation layers by pressing the plus button, and rename layers by hovering over the text that says ‘Layer 1’, and clicking left, then right, then left on the mouse in quick succession (I don’t know why this works).



Make all layers for all structures you want to annotate in your sample. It is VERY IMPORTANT that all layers exist in the exact same order in all training and testing images you annotate. Create a layer even for structures that are not present in all images you annotate. If a layer is present in one image, it must be present in all images.

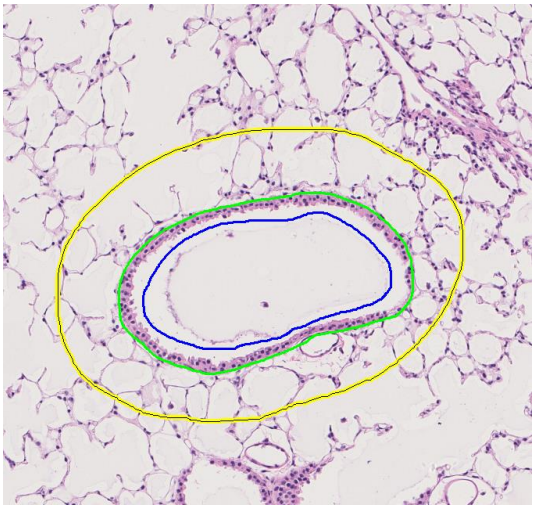


5. Next, create your annotations by selecting the pen tool in the ImageScope taskbar. Aim for ~25 annotations of each structure on each image. This is not always possible for rare structures.



Notes on annotating:

- The quality of your segmentation model depends on the quality of your annotations. Zoom-in to high magnification to annotate, and try to annotate very cleanly along the edges of structures.
- Try to make all of annotations roughly the same size (do not make huge background annotations and tiny tissue annotations).
- If your annotations are overlapping, they must follow consistent 'laws.' One annotation layer must always dominate the other layer, so that MATLAB understands how to interpret overlapping masks. Consider creating a list of 'nesting order' before you begin annotating to organize which annotations are the 'bottom layer' up to the 'top layer.' For example, to annotate bronchiole inside of alveoli in lung histology, the alveolar annotation (yellow) will encircle the bronchiole annotation (green). A background annotation (blue) will circle noise inside the bronchiole. Here, the background layer is 'dominant' over the bronchiole layer, and the bronchiole layer is 'dominant' over the alveolar layer, in cases of overlap.



6. When you've finished making annotations, you are ready to set up your MATLAB training function. This package requires several variable definitions that are listed inside the top section of the function '**train_image_segmentation**.' For the sample lung dataset, this function is filled out and saved as **train_image_segmentation_lung**. To create the inputs for this function you will need:

   The subfolder containing '.xml' files with your training annotation information (created by ImageScope)

pthannotations=[pth,'annotations']; % put images here to annotate for training

The subfolder containing '.xml' files with your testing annotation information (created by ImageScope)

pthannotations_test=[pthannotations,'testing image']; % put image here to annotate for testing

The subfolder containing high-resolution tif images corresponding to each annotated training image

pthim=[pthannotations,'10x']; % copy a tif image here for each training image

The subfolder containing high-resolution tif images corresponding to each annotated testing image

pthtestim=[pthannotations_test,'10x']; % copy a tif image here for each testing image

The subfolder containing the full dataset of high-resolution tif images that you want to classify with your model after training is finished

pthclassify=[pth,'10x'];

The downsample factor of your high-resolution tif images compared to the images you annotated in ImageScope (this is the same number you gave as input to the function **create_downsampled_tif_images** to create your high-resolution tif images).

umpix=2;

The date that your model was trained. This will define the output folder name where the trained model will be saved, such that unique folders are generated for different iterations of models made.

nm='12_15_2023';

The size (in pixels) of the tiles you want to create for model training. By default this is set at 700, resulting in creation of 700 x 700 x 3 sized RGB tiles. Depending on your GPU memory, you may be able to increase this (model performance will be better for larger tiles) or you may need to decrease this.

sxy=700;

The number of large training images you want to create. Each of these large images will be chopped into ~100 training tiles. A good number is around 15 but go lower or higher if you have very few (~5 images) or very many (>20 images) annotations.

ntrain=15;

7. The variable '**WS**' is the most complicated to create, as it requires you to think critically about the annotation layers you created. Inside of this variable, you will define how to order your layers, whether to combine layers, and whether to keep or remove whitespace from your layers. For simplicity, **WS** is split into four components:

    a. **annotation_whitespace** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. For each position within **annotation_whitespace**, indicate with 0 if you wish to remove the whitespace from your annotation (for example if you annotated a blood vessel and with to remove the luminal space from your annotation), indicate with 1 if you wish to keep only the whitespace in your annotation (for example if you annotated fat and wish to remove the nonwhite lines dividing separate fat cells from your annotation), and indicate with 2 if you wish to keep both whitespace and nonwhite space within your annotation.

    For example, for the four classes:

    1. Bronchioles (remove whitespace [contains lumen])
    2. Alveoli (remove whitespace [keep only the webbing])

3. Vasculature (remove whitespace [lumen of vasculature])
4. Metastases (remove whitespace [whitespace around the edges of the cancer cells])
5. Nonexpanded lung (remove whitespace [keep only the webbing])
6. Background (keep both [this class contains whitespace + noise like shadows])
7. Stroma (remove whitespace [keep only the thin collagen fibers])

annotation_whitespace=[0 0 0 0 0 2 0];

b. **add_whitespace_to** is a matrix of size [1 2]. The first number defines the annotation layer to add whitespace to when it was removed from another class (where **annotation_whitespace** = 0) – this is usually the background class. The second number defines the annotation layer to add nonwhitespace to when it is removed from another class (where **annotation_whitespace**=1) – this is usually the stroma class. For example, for the four classes listed above, add whitespace to class 6 (background), and add nonwhitespace to class 6 (also background as this is not applicable to this model).

add_whitespace_to=[6 6];

c. **nesting_order** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. This variable allows you to define how overlapping annotations should be processed. Numbers on the left side of this variable are 'below' annotations listed to the right of them. For example, in the example above, we annotated a bronchiole inside of an alveolar annotation. This means bronchioles must be 'above' alveoli in the nesting order. Let's assume that stroma is the 'bottom layer,' followed by alveoli, nonexpanded tissue, cancer, vasculature, bronchioles, and background.

nesting_order=[7 2 5 4 3 1 6]; % stroma, alveoli, nonexpanded, cancer, vessels, bronch., backgrd

In contrast, if none of your annotations overlap, nesting_order can be sequential:

nesting_order=[1 2 3 4 5]; % no particular nesting order

d. **combine_classes** is a matrix of size [1 N] where N is the number of annotation layers you created in ImageScope. This variable allows you to re-order or combine multiple classes. For example, if you originally annotate the seven classes listed above but decide to combine the alveoli and nonexpanded tissue classes your variable would look like:

combine_classes=[1 2 3 4 2 5 6];

You now will create a deep learning model that has only 6 classes.

These four defined variables will be combined into the variable '**WS**' in MATLAB.

8. Finally, create variables defining the names and RGB colors for the final tissue structures in your model. In our sample model, we have four classes (after we combined the classes fat and background using the combine_classes variable). For this we define classNames and cmap.
ClassNames is a string variable containing the names of each class. Note this variable cannot contain spaces in names, use underscore instead ("blood_vessels" instead of "blood vessels").

classNames=["bronchioles" "alveoli" "vasculature","cancer","nonexpanded","whitespace","stroma"];

cmap is a matrix of size [N 3] for N final classes in your deep learning model. Each row of this matrix defines the RGB color of one class of your deep learning model in 8-bit space. For example:

cmap=[150 099 023;... % 1 bronchioles (brown)

```
023 080 150;...    % 2  alveoli     (dark blue)
150 031 023;...    % 3  vasculature  (dark red)
199 196 147;...    % 4  cancer       (v dark purple)
023 080 150;...    % 5  nonexpanded  (dark blue)
255 255 255;...    % 6  whitespace   (white)
242 167 227];...   % 7  collagen     (light pink)
```

With these inputs, you are ready to train your model. If you call the function **train_image_segmentation**, this will train a deep learning model (saved in a folder inside **pthannotations**), test that model using the annotations inside **pthannotations_test** and will classify the images in the folder **pthclassify**.

If you do not receive satisfactory results, add additional annotations (either entirely new training images or additional annotations on the same training images) until satisfactory results are achieved.

The workflow in this section will create a subfolder inside **pthannotations** containing the trained deep learning model and training tiles. These tiles can take up large amounts of disk space, so delete them (but keep your model inside the folder 'net.mat').

```
pthmodel=[pthannotations,nm];
```

This workflow will also create a subfolder inside the high-resolution tif image path named **['classification_',nm]**, where nm is the date of the deep learning model training. Inside this subfolder will be the segmented, high-resolution tif images.

```
pthclassified=[pth10x,'classification_',nm];
```

Inside of this subfolder will be a second folder named '**check_classification**,' containing color labelled versions of the classified files. These colorized classified images are handy for visualization of the results and qualitative validation of model performance across the dataset.

```
pthcheckclassification=[pth10x,'classification_',nm,'check_classification'];
```

Section 5. Register the deep learning labelled images

Using the registration transformations calculated in low-resolution (as described in Section 3), register the segmentation masks of the high-resolution tif images generated in Section 4.

1. The function requires the path containing the classified, high-resolution images from Section 4 (**pthclassified**), the path containing the image registration information from Section 3 (**pthdata**), the scale between the high-resolution classified images and the low-resolution registration images (here for registration of 1x images and classification of 10x images, scale=10), and the pixel number of the background class in the segmentation model built in Section 4 (here, nwhite=3).

```
pthim=pthclassified;

pthdata=[pth,'1x\registered\elastic registration\save_warps'];

scale=10;

padnum=3;

apply_image_registration(pthim,pthdata,scale,padnum);
```

   This function will create an subfolder named 'registeredE' inside the **pthclassified** folder that contains the registered, classified images.

```
pthclassifiedE=[pthclassified,'registeredE'];
```

Section 6. Construct 3D tissue matrix

Next, we will create a 3D quantifiable matrix from the tissue labels using the registered, segmented images created in Section 5. **vol** will be a 3D matrix containing the registered, labelled data. To call this function, you need to locate the subfolder containing the registered classified images output in Section 5, define a location to save the output matrix, define the desired resolution of the volume matrix (relative to the current resolution of the images, and identify the RGB color map used in the deep learning model in Section 4.

pthclassifiedE is the subfolder with the registered, classified images created in Section 5:

```
pthclassifiedE=[pthclassified,'registeredE'];
```

Define a subfolder where you would like to save the volumetric data:

```
pthvolume=[pth,'lung_data'];
```

set **sk** to 4 so that 10x image (1 um / pixel resolution) are downsampled to 4 micron / pixel:

```
sk=4;
```

nwhite is the background class from the deep learning model. Here, let's use 6:

```
nwhite=6;
```

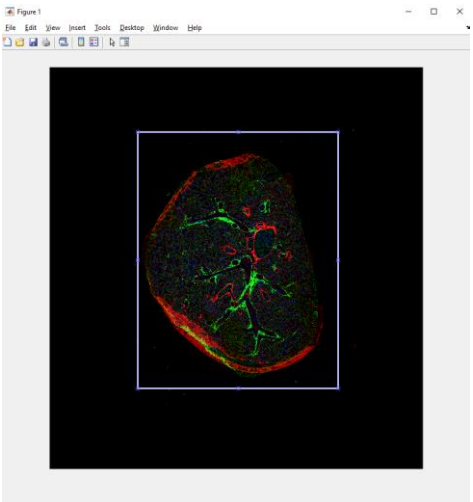
cmap is the matrix containing RGB triplets for each class in the deep learning model, defined in Section 4.

```
cmap=[];

build_tissue_volume(pthclassifiedE,pthvolume,sk,nwhite,cmap);
```

First, the function will display to you a concatenated image containing the first, center, and last classified image in your image stack. Here, you will manually drag a rectangle to surround the tissue in the image, then double-click. This allows you to crop out excess whitespace from your 3D matrix to save RAM.

Sample concatenated image of the first, center, and last image with manually selected rectangle to crop out excess space:



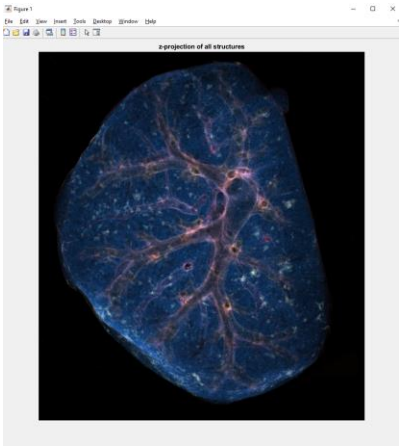
Next, the function will load all serial images, crop them, downsample them using the variable **sk**, and build them into the matrix **vol**. A z-projection will be created and displayed showing a combination of all non-background classes. This z-projection will be saved inside the folder **pthvolume**.

Additionally, this function will create a .mat file named **volume.mat** saved inside the subfolder **pthvolume**. Inside this file will be a variable named **vol** containing the volumetric tissue labels, **rr** containing the cropping information defined

inside the function, **pthclassifiedE** the folder containing the registered classified images used to create the volume, **imlist** containing the filenames of the images comprising the 3D matrix, and **sk** the downsample factor between the images in **imlist** and the data in **vol**.

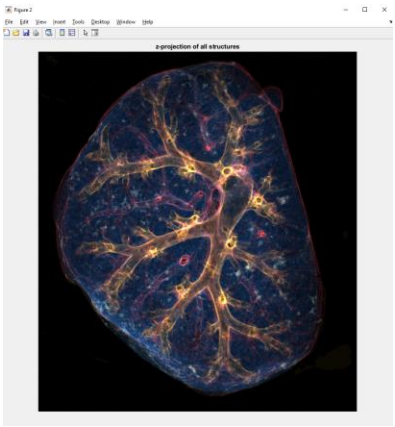
Sample z-projection of all non-background tissue with default contrast:

contrast=[1 1 1 1 1 1 1];



Sample z-projection of all non-background tissue with contrast to emphasize bronchioles and vasculature:

contrast=[3 0.8 2 1 0.8 1 1]



Note: When determining what resolution to make your volumetric matrix, there are a few considerations:

- You may want to make your 3D matrix isometric (same resolution in x, y, and z) to simplify 3D quantifications. In this case, define **sk** such that your classified images will be downsampled to the spacing between adjacent histological images
- You may want to consider the size of structures you are trying to quantify. If you need to maintain classification of very fine structures (such as thin blood vessels), down sampling too much may eliminate the connectivity between these thin objects.

Section 7. Nuclear coordinate generation

In this section, generate nuclear coordinates on the high-resolution H&E images through color deconvolution and identification of two-dimensional intensity minimums in the hematoxylin channels of the images (corresponding to the dark blue nuclei). This calculation is done on the high-resolution tif images saved inside pth10x.

Note on cell detection: It is important that the cell coordinates are generated on unregistered images, as warping caused by the registration process can cause inaccuracies in nuclear detection performed on the registered images. Instead, detect coordinates on the unregistered images, then apply the registration transformations to the cell coordinates to determine their positions in registered space.

1. First, generate a mosaic image containing tiles from nine randomly chosen high-resolution images. This will be the image used to optimize the parameters for the cell detection algorithm. Given the path to the high-resolution images, the function will randomly choose nine files (filenames can overlap if the folder contains fewer than nine files). The low-resolution version of each image will be loaded and displayed to the user, prompting the user to click on a region containing tissue. This will be repeated nine times, after which the high-resolution mosaic image will be generated.

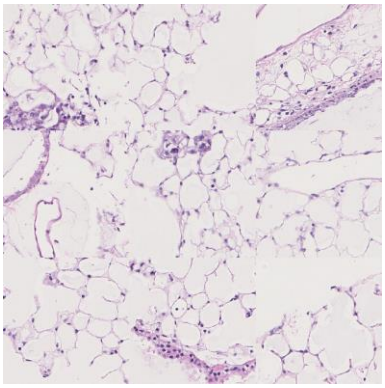
   Note: When manually selecting regions for the mosaic image, select regions with various morphologies to ensure that your cell detection algorithm is robust.

   make_cell_detection_mosaic(pth10x);

   This will create a subfolder containing the mosaic image:

   pthmosaic=[pth10x,'cell_detection_validation'];

   Sample mosaic image:

2. Manually count the nuclei on the mosaic image to generate ground-truth coordinates. Given the path to the mosaic image, this function will display the image to the user and prompt the user to zoom in. Zoom to a region where nuclei are clear, then press 'spacebar.' Click on each nucleus in the zoomed region, then press 'z' to zoom or scroll to another region in the image. Continue until all nuclei in the mosaic have been annotated. At any time, exit the code by pressing 'z' and selecting continue later. When you recall the code it will automatically continue from where you left off. When you are finished, press 'z' then select 'Quit'.

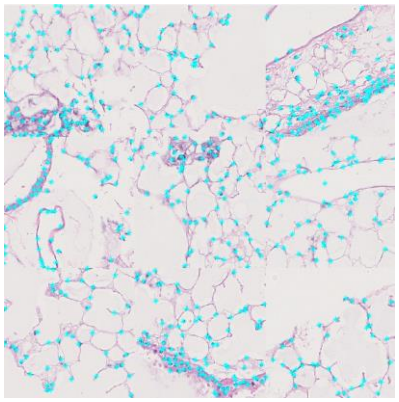
   For a mosaic image comprised of 9 200x 200 micron^2 10x magnification tiles, manual annotation of nuclei for a medium dense tissue should take 30 – 45 minutes for a trained user.

   manual_cell_count(pthmosaic);

   This will create a subfolder named 'manual detection' containing the manually identified coordinates inside a mat file in a variable named 'xym.'

   pth_mosaic_manul_coords=[pthmosaic, manual detection'];

Sample mosaic image with manual nuclear coordinates overlaid:



3. Next, determine the optimal intensity cutoff and minimum spacing between nuclei to most maximize the true positives and minimize the false positives and false negatives in automatic determination of nuclear coordinates. This is done through comparison of the manually generated cell coordinates to several variations of automatically generated coordinates.

   First, deconvolve the mosaic image to get its hematoxylin channel. If the images are H&E, use the ImageJ default optical densities for H&E images:

   This function generates output images containing the hematoxylin and eosin (or DAB) channel images:

   pthmosaicH=[pth10x,'Hchannel'];

   pthmosaicE=[pth10x,'Echannel'];

4. Next, optimize the parameters by iteratively generating nuclear coordinates on the mosaic image and comparing them to the manual cell detection.

   get_nuclear_detection_parameters(pthmosaic)

   This will generate a subfolder named 'automatic detection' containing the optimal automatically generated cell coordinates, and a subfolder named 'optimization params' containing the determined parameters.

   paramsfile=[pthmosaic,'automatic_detection\optimized_params.mat'];

   Now, apply the color deconvolution to the entire high-resolution dataset using the calculated parameters. We first deconvolve the images to get their hematoxylin channel. If the images are H&E, use the ImageJ default optical densities for H&E images:

   stain_type=1;

   If the images are IHC, use the ImageJ default optical densities for H DAB images:

   stain_type=2;

   deconvolve_histological_images(pth10x,stain_type);

   This function generates output images containing the hematoxylin and eosin (or DAB, for IHC) channel images:

   pthH=[pth10x,'Hchannel'];

   pthE=[pth10x,'Echannel'];

5. Finally, apply the cell detection to the hematoxylin channel of the high-resolution images using the optimized parameters:

```
pthparams=[pth10x,'cell_detection_validation\automatic_detection'];

cell_detection(pthH,pthparams);
```

The workflow in this section will create a subfolder inside the high-resolution hematoxylin-channel tif image folder named ‘cell_counts’. Within ‘cell_counts’ will be a mat file corresponding to each tif image. Inside each mat file will be a variable named ‘xy’ with coordinates for each cell found in the corresponding image.

```
pthcoords=[pth10x,'cell_counts\'];
```

Section 8. Register the nuclear coordinates

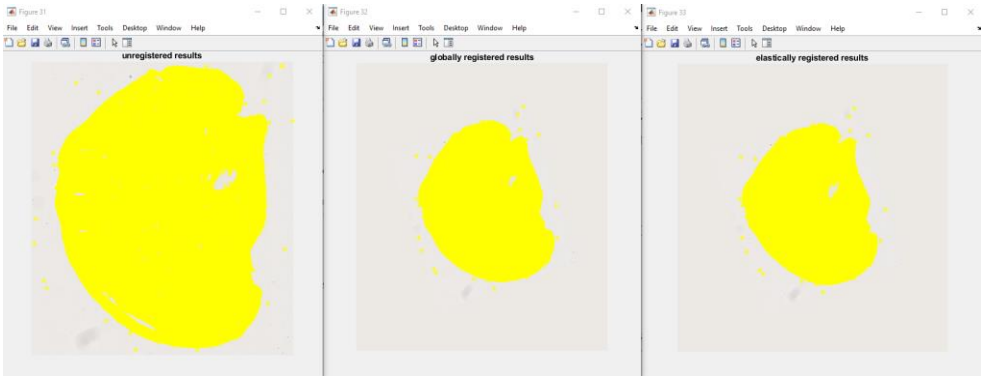
Using the registration transformations calculated in low-resolution (as described in Section 3), register the nuclear coordinates generated in Section 6.

1. As input, this function needs the path to the low-resolution images that were registered in Section 3 (pth1x), the path containing the coordinates calculated in Section 5 (pthcoords), and the scale factor between these images. For cell detection performed in 10x and registration calculated in 1x,

```
scale=10;

register_cell_coordinates(pth1x,pthcoords,scale);
```

Sample well registered cell coordinates overlap with unregistered, global, and elastically registered H&E:



*if the yellow dots do not line up over the tissue, there is a problem in your code.

This function will create a subfolder inside pthcoords named 'cell_coordinates_registered.' Inside that subfolder will be a mat file corresponding to each tif image. Inside each mat file will be a variable named 'xy' (unregistered cell coordinates), 'xyg' (globally registered cell coordinates), and 'xye' (elastically registered cell coordinates), all saved at the same low-resolution as the original registration images used in Section 3.

```
pthcoordsE=[pthcoords,'cell_coordinates_registered'];
```

Section 9. Construct 3D cell matrix

Next, create **volcell**, a matrix containing the nuclear labels in a volumetric matrix the same size and resolution of the tissue matrix constructed in Section 6. To call this function, you need to define the folder containing the registered, classified images created in Section 5, the folder containing the registered cell coordinates created in Section 8, the folder containing the tissue matrix created in Section 6, and the scale between the high-resolution, classified images and the high-resolution images used for cell detection (this is probably 1).

1. First, define the folders containing the registered, classified images, the registered nuclear coordinates, and the tissue volume matrix:

   pthclassifiedE=[pthclassified,'registeredE'];

   pthcoordsE=[pthcoords,'cell_coordinates_registered'];

   pthvolume=[pth,'lung_data'];

2. Next, define the scale between the high-resolution images used for tissue classification in Section 4 and the high-resolution images used for cell detection in Section 7. If the classification and cell detection were applied to the same resolution images (this is likely), the scale is 1.

   scale=1;

   Finally, nwhite is the whitespace class from the deep learning model. Here, let's again use nwhite=6 (the background class number from the sample lung model.

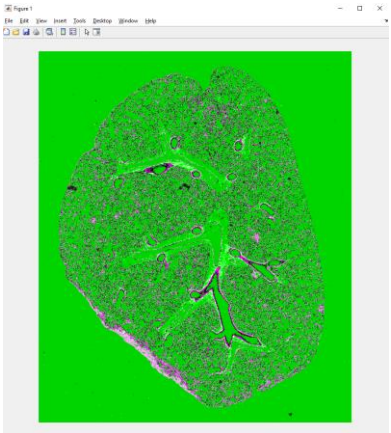
   nwhite=6;

3. Now, call the make_volcell function:

   build_cell_volume(pthclassifiedE,pthcoordsE,pthvolume,scale,nwhite)

   This function will create a 3D matrix containing nuclear coordinates. The matrix will be at the same resolution as the tissue volume matrix, and automatically crop out the same background space that was manually selected for removal in Section 6. To validate that the function creates a variable **volcell** that is the same resolution and crop as the tissue matrix **vol**, the function will display two images. First, the function will display an overlay of the center image from vol and **volcell**. Second, the function will display an overlay of the z-projections of **vol** and **volcell**. If either of these images do not appear overlayed (you see two separate, unaligned tissues), there was a problem with the construction of either **vol** or **volcell**.

Sample z-projection showing successful overlay of the center image of 3D tissue matrix and 3D cell matrix to validate your two variables (if you see two, unaligned structures, something is wrong):



Sample z-projection showing successful overlay of 3D tissue matrix and 3D cell matrix to validate your two variables (if you see two, unaligned structures, something is wrong):

Section 10: Notes on visualization

Beyond the scope of this guide, consider **volshow** or **patch**

Section 11: Notes on quantification

Primarily beyond the scope of this guide, as most spatial calculations must are custom to the biological question. Some basic considerations:

To smooth your data. Your segmentation algorithm and 3D volumetric matrix will likely contain small false positive and false negative data. Try smoothing your volumetric data with the following.

vol = volumetric matrix of size [m n z] containing 6 labels for 5 tissue structures plus background

ws = 6; % background class for volumetric matrix

to lightly smooth the tissue labels inside vol:

```
volS=vol; % create a smoothing variable
for b=1:max(vol(:))
    tmp=vol==b;
    tmp=imclose(tmp,strel('sphere',2)); % morphological closing to fill small holes
    tmp=imopen(tmp,strel('sphere',1)); % morphological opening to remove small noise
    tmp=bwareaopen(tmp,500); % delete 3D objects that are fewer than 500 voxels;
    volS(volS==b)=ws;
    volS(tmp==1)=b;
end
```

Note: be VERY CAUTIOUS using **imopen** to smooth thin structures (glands and vasculature), as **imopen** may eliminate "true positive" thin walls. Consider skipping **imopen** when you smooth

To quantify volumes. volume of tissue class 2 (use **vol** or your smoothed **volS**):

```
sxy=4; % resolution of 'vol' matrix in xy in units of micron / pixel
sz=4; % distance between serial histological sections
volume_type_2=sum(vol(:)==2)*sx*sx*sz; % volume of type 2 in units of micron^3
volume_type_2=volume_type_2/(10^9); % volume of type 2 in units of mm^3
```

To quantify cellularity. number of cells in class 2 (use **vol** or your smoothed **volS**):

```
tmp=double(volcell).*double(vol==2);
cell_type_2=sum(tmp(:));
```

For more complex quantifications:

Consider the functions **bwdist**, **regionprops3**, and custom functions you create.

Happy Coding!

Section 12: Shorthand for calling functions

To downsample ndpi or svs images to 10x, 5x, and 1x tifs:

create_downsampled_tif_images or try Openslide in python

To calculate registration on the low resolution (1x) images

Calculate the tissue area and background pixels:

calculate_tissue_ws

Calculate the registration transforms:

calculate_image_registration

To build a 3D tissue volume using sematic segmentation:

First, generate manual annotations in Aperio imagescope

Second, apply the deep learning function to train a model and segment the high resolution (5x or 10x) images:

train_image_segmentation

To apply the registration to segmented images:

apply_image_registration

To build a 3D tissue matrix from registered, classified images:

build_tissue_volume

To build a 3D cell volume containing nuclear coordinates:

Build a mosaic image containing regions of many whole-slide images for cell detection optimization:

make_cell_detection_mosaic

Manually annotate the mosaic image to get the 'ground-truth' number of cell nuclei:

manual_cell_count

Determine cell detection parameters using the manual annotations on the mosaic image:

get_nuclear_detection_parameters

Deconvolve the high-resolution (5x or 10x) H&E images before applying the cell detection algorithm:

deconvolve_histological_images

Detect cells on the hematoxylin channel of the high-resolution images:

cell_detection

Apply the registration to the cell coordinates:

register_cell_coordinates

Build a 3D cell coordinate matrix corresponding to the 3D tissue matrix:

build_cell_volume