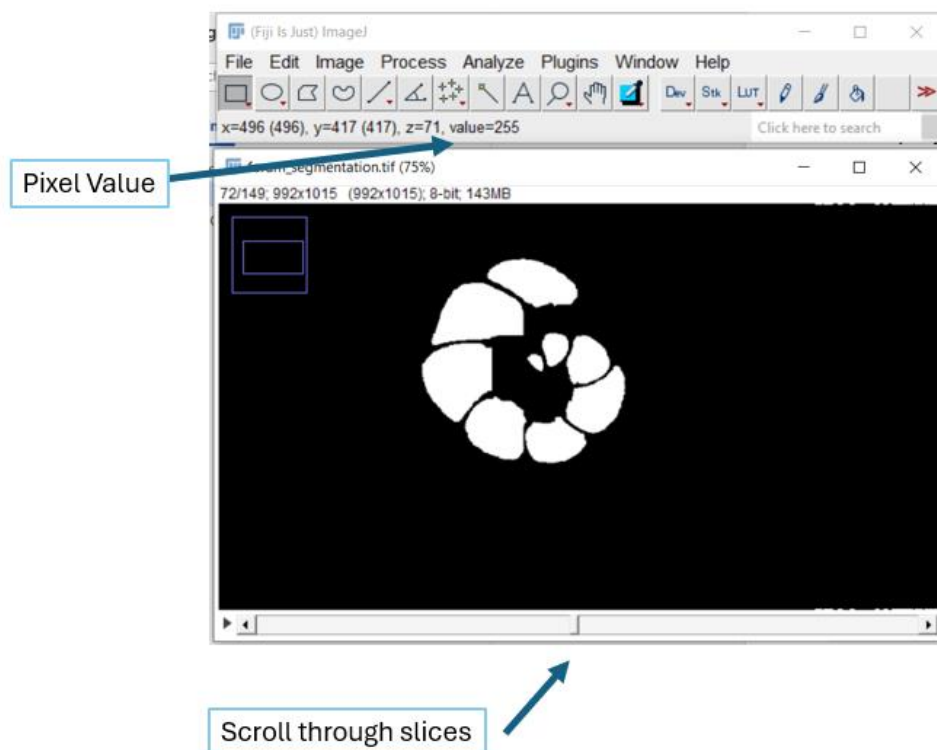


Segmenting chambers of Foraminifera

Examine your scan

First, identify your task: For the `foram_segmentation.tif` file, it is an AI generated segmentation of the whole foram chambers, and we want to separate it to individual chambers.

Visualise the scan you want to segment. Here we use Fiji (ImageJ) to open the file. Scroll through the scan, and have an understanding of the scan, and pixels values of different regions. As it's a binary segmentation, it only has values of 0 and 255.



Generate seeds

In Sprout, we shrink the regions to create separations, then we can segment them into different regions. Outputs of different levels of shrinking are named seeds.

VS code version:

Click open folder to open the root folder you downloaded (`nhm_seg_workshop`), then click `make_seeds.yaml` to edit.

Non VS code version:

Use any text editor to edit the `make_seeds.yaml`

Editing `make_seeds.yaml`

Set the file path correctly, as follows:

- `workspace: "."`
- `file_name: "data/foram_segmentation.tif"`

Set the output folder. Here, we specify `output/foram`, which is an `output/foram` folder in the root directory where the seeds will be saved.

Remaining parameters:

- `num_threads`: Set the number of threads based on your machine's capacity. If unsure, set it to 1.
- `target_thresholds`: Set the threshold for the Region of Interest (ROI). Generally, a larger threshold results in less ROI inclusion. For this case, we only need one threshold, set to 0, since there are only two pixel values.
- `ero_iters`: Set the number of erosion iterations for seed generation. More iterations reduce the retained regions' size. Here, we use 5.
- `segments`: The algorithm retains the top `<segments>` largest disconnected components. In this case, we use 20.

Run `make_seeds.py`









VS code version:

Click `make_seeds.py` and press `Ctrl+F5` to run the code.

Terminal version:

In the terminal, navigate to the root folder, then run `make_seeds.py` in the terminal.

There are outputs in the terminal, and seeds are saved in `output/foram` folder.

work > codes > nhm_seg_workshop > output > foram	
Name	St
 seed_ero_0_thre0_segs_20.tif	
 seed_ero_1_thre0_segs_20.tif	
 seed_ero_2_thre0_segs_20.tif	
 seed_log.json	

Examine seeds

Many candidate seeds were generated in the previous step. We need to select a good one and then grow it back to obtain the final segmentation. To choose a good seed, it's helpful to visualize it. We recommend using tools that support 3D visualization, such as AVIZO and DragonFly.

As a free, less computationally intensive alternative, we provide instructions for using MeshLab to visualize the meshes generated from seeds.

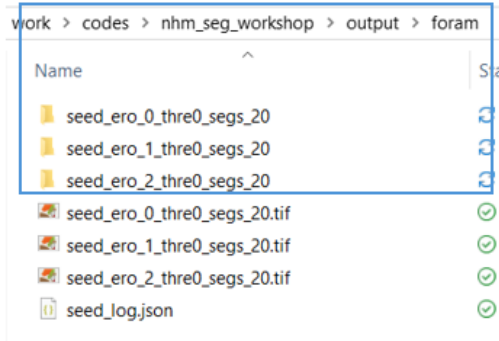
Edit the `make_mesh.yaml`.

Parameters we need to change for this case is: change workspace to the seeds folder, which is `./output/foram/`. Then `downsample_scale`: 10 to make sure the meshes are not too big.

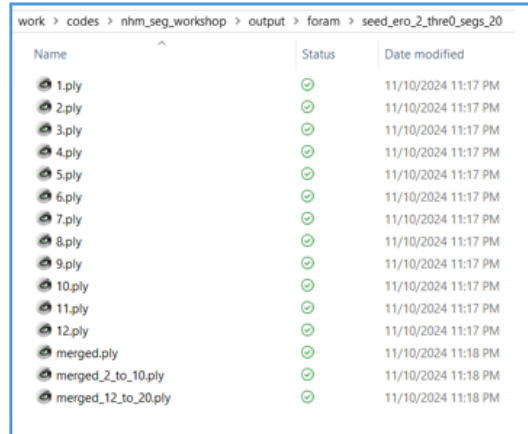
Next, run `make_mesh.py`. Folders matching the seed file names will be generated in `./output/foram/`.

Within each folder, meshes for individual classes will be created and named according to their class IDs (e.g., 1,2,3,4...). Additionally, merged meshes for all classes, as well as for every 10 classes, will be generated.

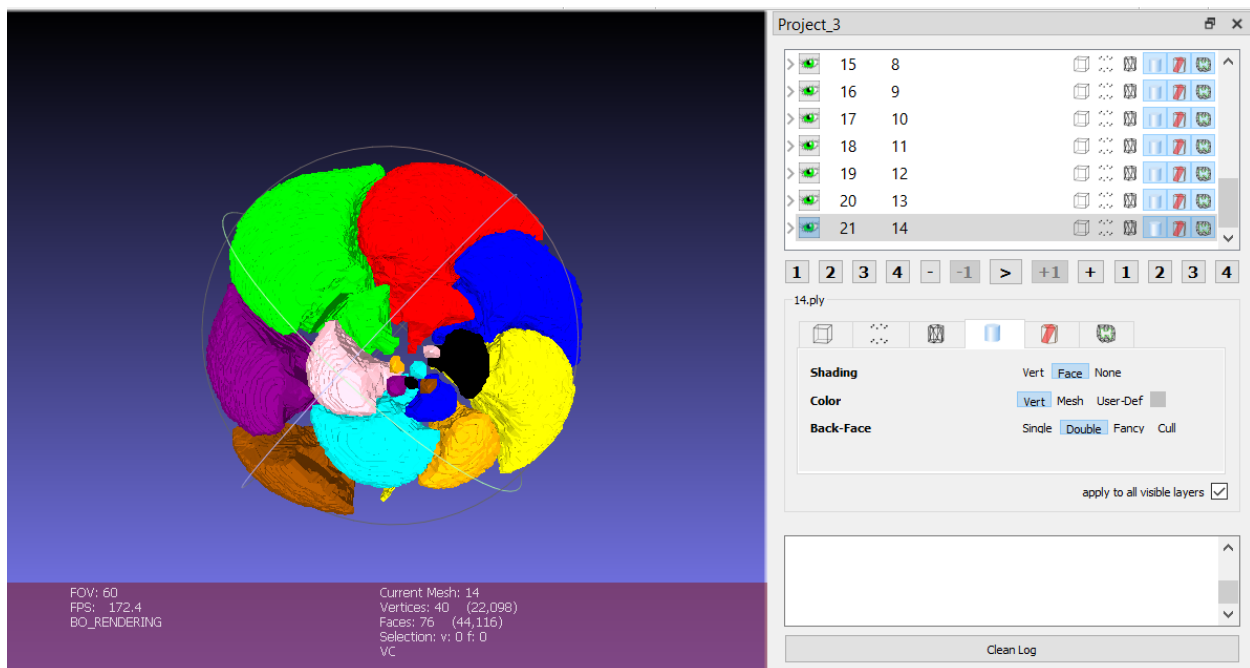
Mesh folders



Meshes



Open them in Meshlab. You can drag all separated meshes in one meshlab session, or you can view the merged meshes for quicker review.



Grow it back

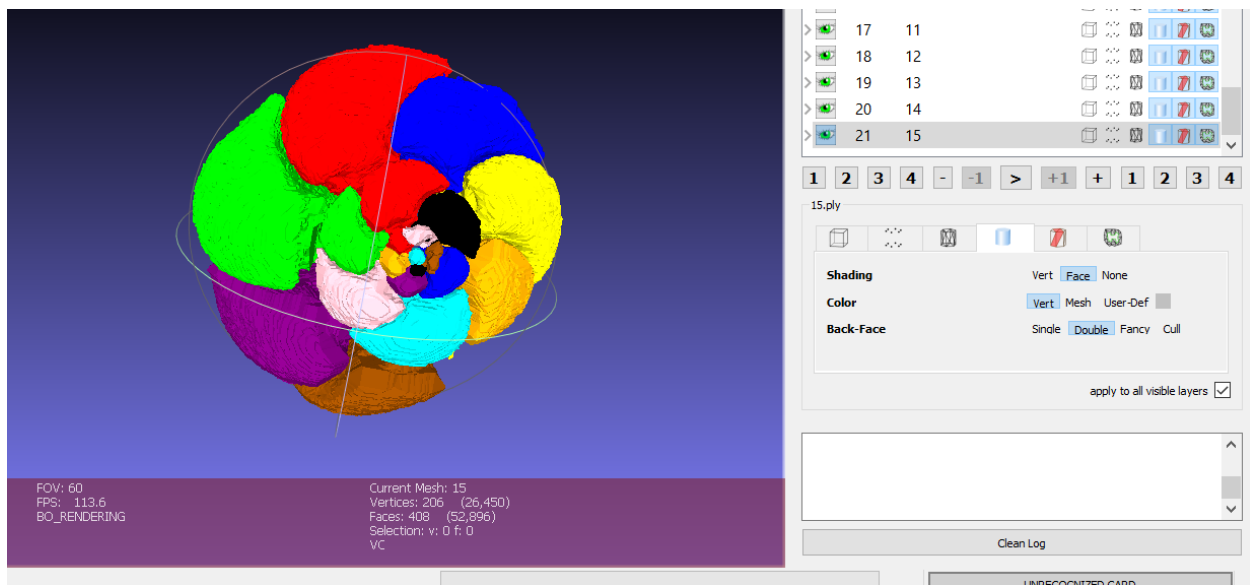
Seeds are typically eroded, covering smaller areas than the target regions. For example, the chambers from the seeds may appear much smaller than the actual chambers. To restore these to their original size, follow these steps:

Edit make_grow_result.yaml

- `img_path`: The path to the original input file, which in this case is `foram_segmentation.tif`.
- `seg_path`: The path to the selected seed file.
- `thresholds`: The region threshold you want to grow to; here, 0 will work fine.
- `dilate_iters`: The number of growth iterations; we'll use 10.
- `output_folder`: The folder for the output; set this to `output/foram/grow`.

Run `make_grow_result.py`. The grown results will be generated and saved in the `output/foram/grow` folder.

You can then use `make_mesh.py` to generate meshes and view them in MeshLab to check the results.



Segmenting bones from a dog limb

Examine your scan

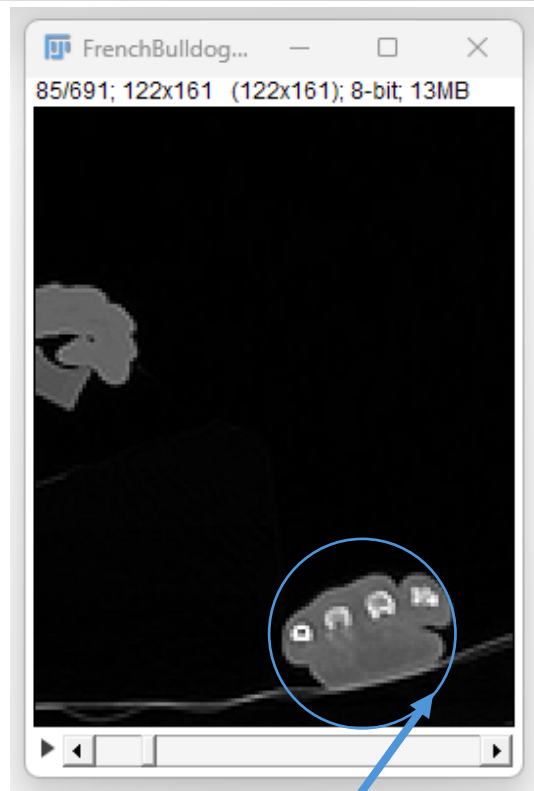
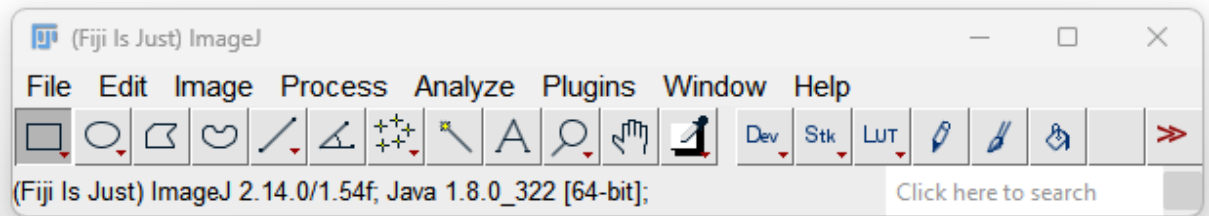
In this case, the input data is a CT scan of a dog (French Bulldog), cropped to include only the right forelimb and surrounding material. The process of segmenting this scan is very similar to the foram except this time the thresholds are important, rather than erosion. Unlike the foram data, which is a binary segmentation (black and white and has 0 and 255 values), this image data contains different grey values.

Generate seeds

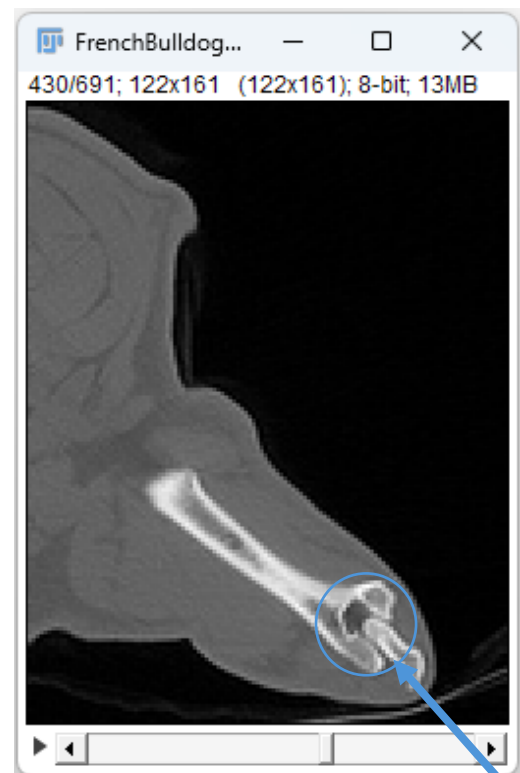
This process is the same as for the foram but with different parameters. We are also using a different version of the seed-making code called `make_seeds.py`, which makes many different seeds. Instead of using erosion we will use thresholds (differences in grey value) to separate the parts.

Choosing thresholds

- Open the stack in Fiji (imageJ) (drag and drop the stack):

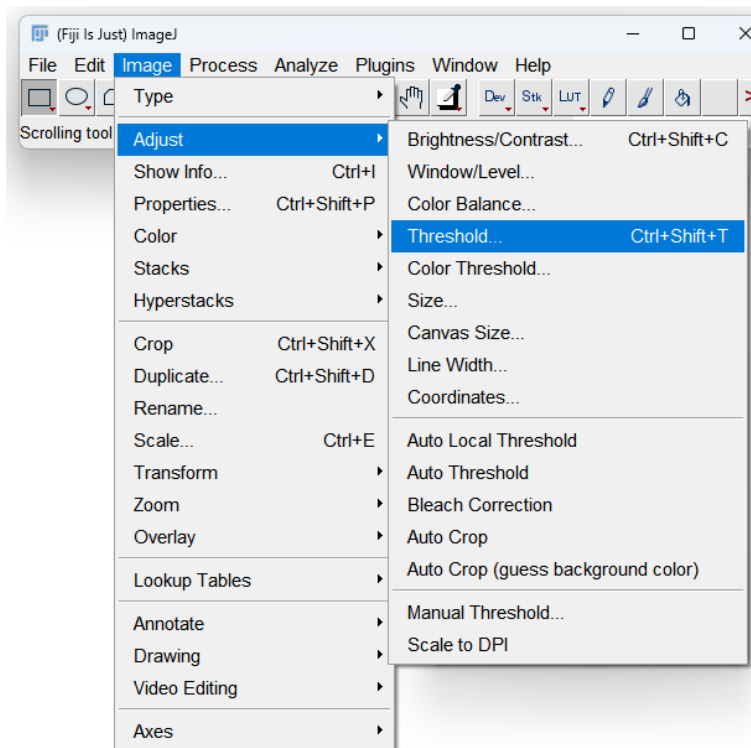


X,Y slice through the foot, showing the metatarsals. Whiter shades are bone, grey is soft tissue, and black is the background



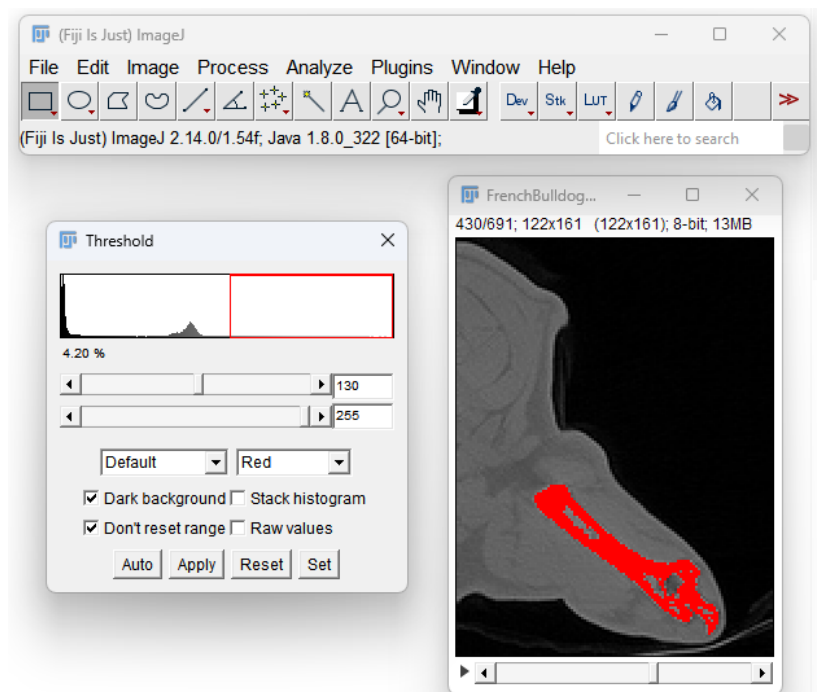
X,Y slice through the elbow joint. Note how the two bones (humerus and ulna) are separated visually by a band of grey – this threshold separation is what we will exploit with SPROUT to separate the bones.

- Select Image → Adjust → Threshold...

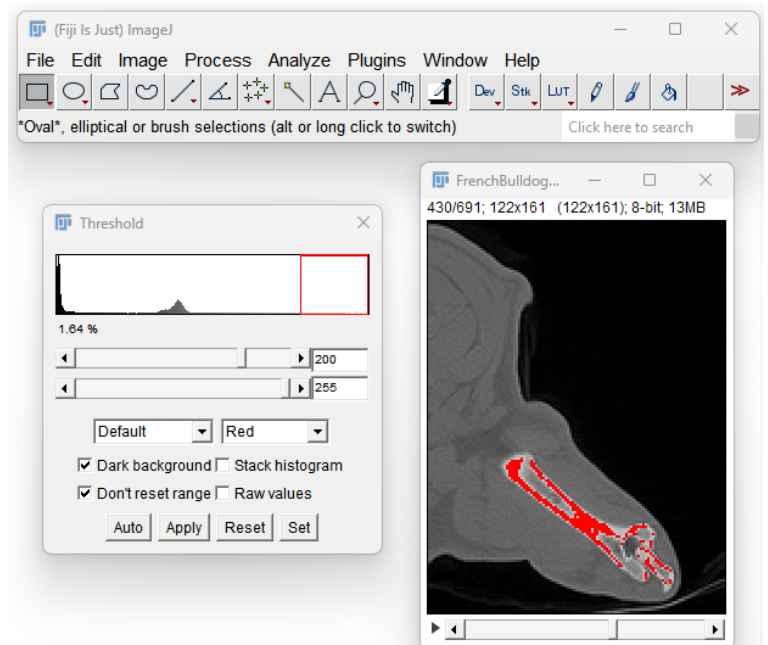


- Change the top bar to select different thresholds, see how the red (selected parts of the image) change:

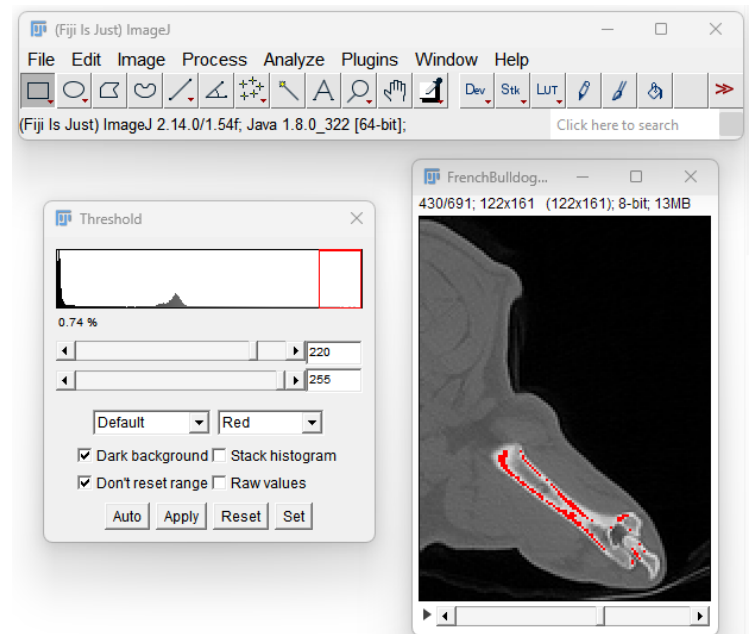
A threshold of ~130 selects all of the bone and very little of the soft tissue.



A threshold of ~200 selects less of the bone, creating separation of the humerus and ulna.



A threshold of ~220 selects even less of the bone – total separation of the elements



- With this process we can decide on a range of thresholds across which separation appears.
- We are going to set multiple thresholds so we do not need to spend a long time on this step, separation appears around a threshold of ~200 so let's set seed thresholds of [180,190,200,210,220]

Edit the .yaml and run seed generation:

VS code version:

Click open folder to open the root folder you downloaded (nhm_seg_workshop), then click make_seeds.yaml to edit.

Non VS code version:

Use any text editor to edit the make_seeds.yaml

Editing make_seeds.yaml

Set the file path correctly, as follows:

- workspace: "."
- file_name: "data/FrenchBulldog_lowres.tif"

Set the output folder. Here, we specify output/dog, which is an output/dog folder in the root directory where the seeds will be saved.

Remaining parameters:

- num_threads: Set the number of threads based on your machine's capacity. If unsure, set it to 1.
- target_thresholds: Set the threshold for the Region of Interest (ROI). Generally, a larger threshold results in less ROI inclusion. For this case, we will make seeds using a number of thresholds and then pick the one which creates the best separation. Let's try changing the target thresholds to [180,190,200,210,220]
- ero_iters: Set the number of erosion iterations for seed generation. More iterations reduce the retained regions' size. Here, as we are using thresholding to separate the bones we will set this to 0,
- segments: The algorithm retains the top <segments> largest disconnected components. In this case, we use 20.

Run make_seeds.py

VS code version:

Click make_seeds.py and press Ctrl+F5 to run the code.

Terminal version:

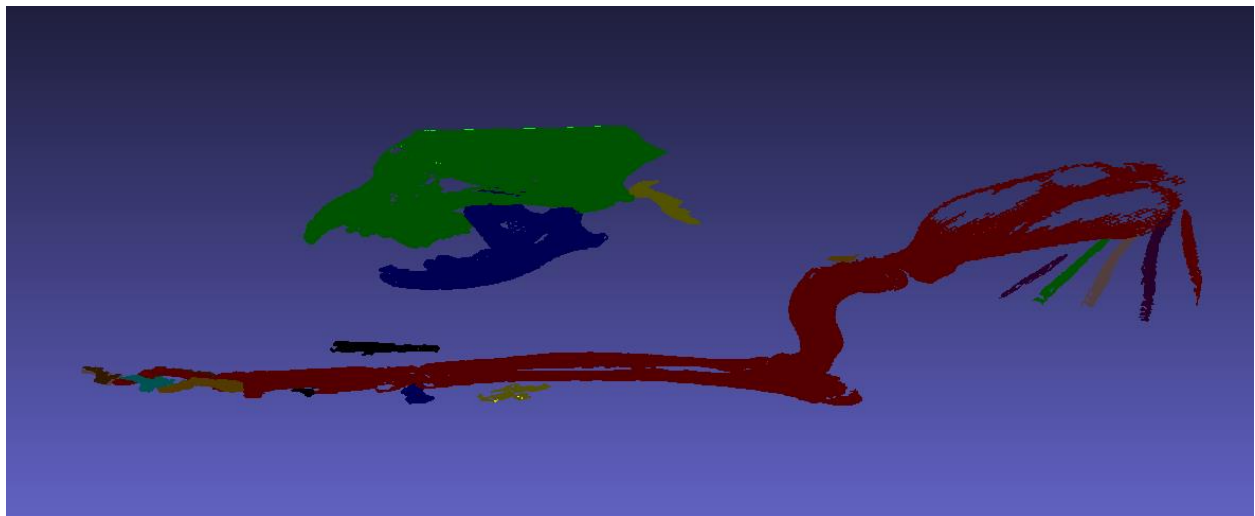
In the terminal, navigate to the root folder, then run `make_seeds.py` in the terminal.

There are outputs in the terminal, and seeds are saved in `output/dog` folder.

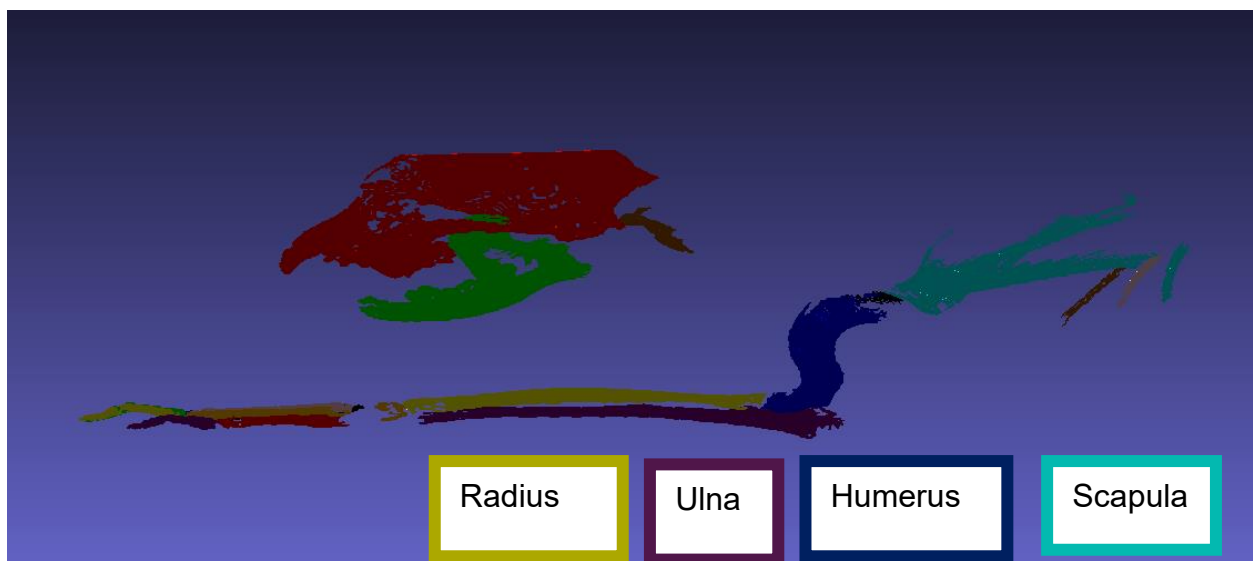
Examine seeds

Either look at the seeds in AVIZO or follow the same steps as before to make meshes of the seeds and visualize in Meshlab. Just make sure to change the input and output folders to `output/dog/seeds_ball`. This time set the downsampling to 1, the scans are already very downsampled.

If we look at the meshes you can see that the lower threshold seeds do not separate the limb bones:



As the threshold increases, we see more separation, and merged seed made using the highest threshold we've used (210) looks like this:



The scapula, humerus, radius, ulna and foot bones are now separated. This is a good seed for growing.

Grow it back

In order to create good separation between the bones, the seeds do not contain all of the bone, so we have to grow the seed back using growing. As with the forams:

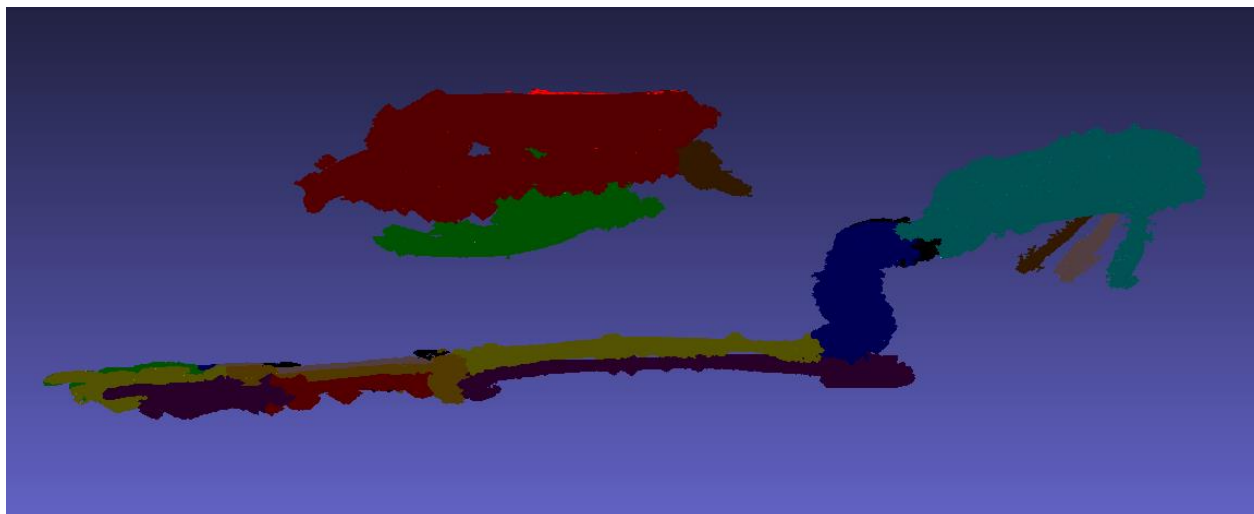
Edit `make_grow_result.yaml`

- `img_path`: The path to the original input file, which in this case is `data/FrenchBulldog_lowres.tif`.
- `seg_path`: The path to the selected seed file. We're going to use `seed_ero_0_thre210_segs_20.tif`
- `thresholds`: The region thresholds. We want to incrementally grow starting from high thresholds, let's use `[175,150,125,100]`
- `dilate_iters`: The number of growth iterations; we're setting four thresholds so we need to set a number of iterations for each of these, let's use `[20,20,10,10]`
- `output_folder`: The folder for the output; set this to `output/dog/grow`.
- `save_interval`: 10 add this so that we will only save growths for every 10 iterations

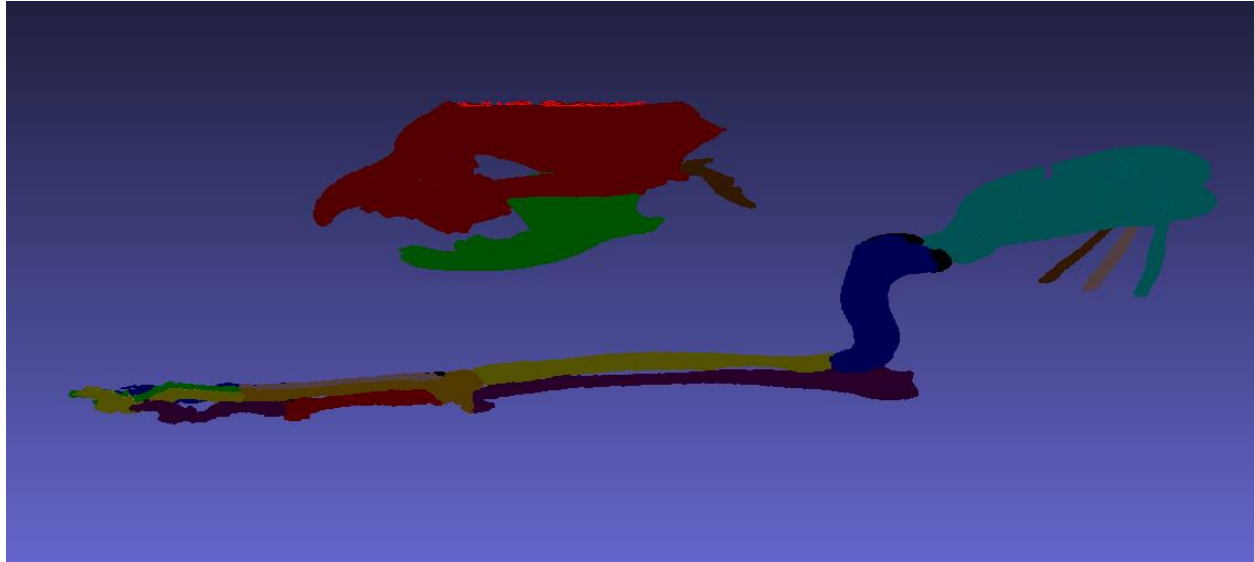
Run `make_grow_result.py`. The grown results will be generated and saved in the `output/dog/grow` folder.

You can then use `make_mesh.py` to generate meshes and view them in MeshLab to check the results.

Look at the result for the mesh fully grown to a threshold of 100:



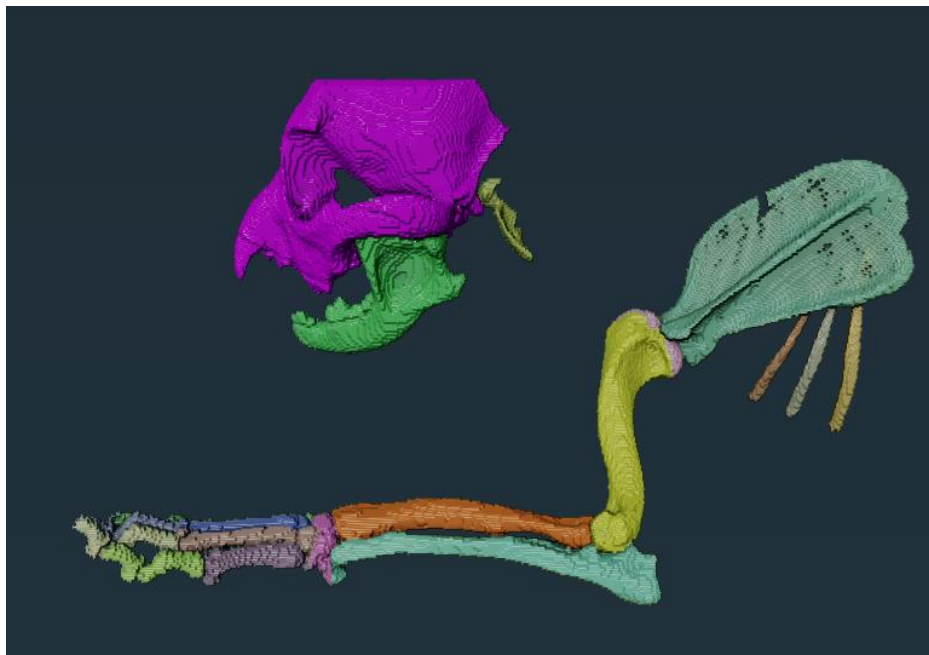
This is fuzzy and chunky, that is because it is overgrown! The threshold is too low. This is not a problem, however, because we have the other iterations where the seed is less grown. This is why setting save intervals is useful. Let's look instead at the result grown to a 120:



This is good!

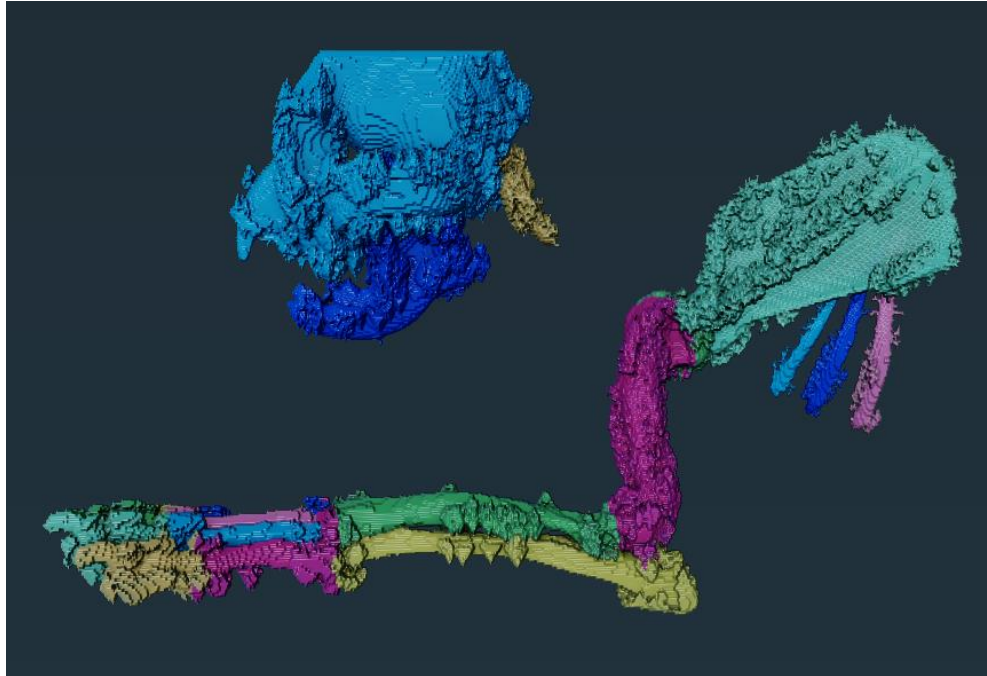
You may have noticed that the dimensions of this dog seem wrong, that's because meshlab assumes that the x, y, z dimensions are all the same, but in these vet scans the z dimension is different.

If we use Avizo to show the same mesh, but edit the dimensions:



Much better.

In Avizo it is also easier to see just how overgrown the mesh grown to a threshold of 100 is:



Batch processing: A pipeline for dogs

If you have a set of scans that you want to process in this way, and they all have similar thresholds for seed generation and growth, you can process them all in one go using the SPROUT pipeline.