Procedure for processing and analysing uCT images of mouse fetal hearts.

Updated November 2022

===============================================================================

Tools:

1. IrfanView (windows).

2. MATLAB (windows but perhaps better in the linux operating system). Note that there are known latency issues with 2021/2022 versions of MATLAB (particularly the editor) when used remotely through an ssh -X login. The last version to not have these issues is: /hpc/matlab/R2020b/bin/matlab.

3. cmgui (windows).

4. For access from windows to a remote linux operating system, MobaXterm is a good option.

5. For looking at stacks of images, ImageJ is a good option - slightly tricky to use from linux (easier from windows). From windows would need to VPN into the university system and have the /hpc drive mounted. This is fairly straight forward but could be a little slow depending on internet connection. Note: it may not be possible to have the /hpc drive mounted for machines outside of the ABI.

===============================================================================

Platforms:

A quite good desktop/laptop computer with graphics card, access to the ABI high-performance computing systems running the linux operating system.

===============================================================================

Steps:

1. Use batch processing in IrfanView to convert images to 8 bit png data (pixels have integer data values between 0 and 255). The images are originally captured in 16 bit (data range 0 to 65535) so considerably better at resolving small intensity differences. However, each voxel (a 3D pixel) in the image is storing two times the information (16/8) so it makes the images big to work with. It is also not clear that there is any information in the 16 bit images that is not mostly captured in the 8 bit anyway. However, it would be good to run a test calculation to confirm this. Images are stored in png format as this is a good compressed but lossless format (unlike e.g. jpeg which is a lossey compression).

2. Run ResliceOrthogonalToLV.m (MATLAB) script code to load in png (or bmp), rotate and reslice raw images so that the long axis of the LV is vertical (in z) and the RV is on the left when viewed from above. There are comments in the script about using Fiji/ImageJ to determine a tight bounding box and slice range. The image vs data coordinates are shown and how they are used is described. Issues that arise tend to come from the specification of these ranges and from inadvertedly swapping the i and j indices. The ij ranges and slice range are manually set to be tight around the tissue region and the top of the valve plane. The script generates a 3D scatter plot that can be used to confirm the placements of the new image axes and that the ventricular cavities have been correctly segmented. The tissue mask is used to determine cropping so that the new ij dimensions and starting slice are within 20 pixels of the nearest tissue. The code is parallelised. Note the code also requires GeneralRotateResample.m to run.

Note that there is now a specification in the file of PermuteOrder and FlipDim1 and FlipDim2 to enable the raw images to be reoriented if for example the ventricles are imaged on their side rather than verticle. At present this has only been necessary for H1C5H. The default PermuteOrder is [1,2,3] and both FlipDim1 and FlipDim2 empty arrays.

3. Run ThreeDMaskuCT\_Parallel.m (MATLAB) script code to load in 8 bit png images (resliced into common position), filter and find a boundary between tissue and background. The typical slice image range is [1:1420] to ensure the heart image set is 1.4 mm from apex to cut-off (the 20 extra pixels are padding before the apex). This is a simple implementation as the uCT images are consistent intensity throughout, and between image sets (set up at imaging time). The top of the image stack is padded with 20 blank slices.

A number of directories are specified: 1. the input image folder, 2. top level mask folder, 3. top level display folder, and 4-7. subfolder names for the ventricular tissue mask, the LV cavity mask, the RV cavity mask and the masked images with background removed. The display folder has downsampled masks and images for iso-surface rendering in cmgui. The script is set up to segment the images in parallel blocks to save memory and time. It takes of the order 5 or so minutes to do this (with 12 processors and considerable memory). The segmented images are written to file and then re-read to do whole mask cleaning where only the largest region (for ventricular tissue) is retained or the two largest regions (for the ventricular cavity, with largest being LV and second largest RV) and the final versions written to file, along with downsampled images and the necessary Box.exnode and Box.exelem files for subsequent display. The Box.ex\* files are written to the top level of the display folder. The image writing and re-reading helps with memory management in the parallel environment. Overall the complete process of constructing the masks required for assessing tissue and cavity volumes, cavity surface area/complexity etc and the masked tissue images that are used to extract subsequent fibre orientations takes approximately 10-20 minutes (with 12 processors and considerable memory) on the HPC machines. Dependency: WriteSingleVolumeElement.m.

4. Manual examination of mask to ensure the ventricle epicardium is clear/clean. The atrial appendages can hang over the base/valve plane of the ventricles and can interfere with the masking of the ventricles (although typically this does not happen with the orthogonal reslicing and truncation at 1.4 mm). One way (fairly quick) to remove issues such as appendages is to sample a few mask images (e.g. 10-15 equispaced images copied and stored in a directory, e.g. Images\_H1C1H/KeyImages) from when an appendage appears attached to the ventricles up to the final image in the cropped stack. A simple drawing tool with a black single pixel pen can be used to put an approximate line between the appendage and the ventricles. Then a black flood fill of the ventricles removes them but leaves the appendage. Flood fill any holes in the appendage mask and save in same directory.

5. If manual clean up has been used, combine and interpolate the sampled masks using the InfillMask.m script code. For validation, the stack of interpolated masks is written into a directory, e.g. Images\_H1C1H/KeyImagesFilled. Mask and masked tissue are loaded (e.g. from Images\_H1C1H/Mask and Images\_H1C1H/Masked) and the in-filled sampled masks are applied to the correct sequence of image slices. Only the largest connected mask (assumed to be the ventricles) is retained. The cleaned mask and masked tissue are written to directories, e.g. Images\_H1C1H/MaskClean and Images\_H1C1H/MaskedClean. The script also requires MaskMorph.m.

6. Run diffusion tissue extrapolation using the DiffusionTissueExtrapolation.m script. This is now parallelised and completes the process in around 20 minutes (with 12 processors and considerable memory). The Masked images (or the cleaned masked images if that was necessary) are used. Diffusion extrapolation images are written to directory, e.g. Images\_H1C1H/MaskedExtrapolated. The parameters in DiffusionTissueExtrapolation.m should be applicable to all problems. The processing is fairly computational intense so look for a machine that has spare capacity. []

7. Use the diffused intensity image to compute intensity gradients and form structure tensor components at a sequence of scales for eigen analysis and helix angle extraction.

8. Use structure tensor data and mask clean to compute streamline representations of the fibre directions for display.

9. Display results visually in cmgui. Note: write a script to load in ventricle and chamber masks, downsample and 8 bit them for display in cmgui. Perhaps also provide a downsampled version of the uCT image for rendering on the surfaces. Perhaps also do a cut volume image stack. All these steps are straight forward given the reorientation of the orginal images. This would also be a good place to generate the box exnode and exelem files for each heart to use for display.

Note: might be helpful to have a way to cut the masks for visualisation - although this may be one off and can be done in an adhoc way; it is simply zeroing out part of a mask. For example with the orientation of the LV/RV the mask could be cut through the centroid plane (this centroid information may need to be passed from step 2?).

===============================================================================

Quantitative measures:

Image information content (Shannon entropy)

Fibre orientations

Helix angles

Fractional anisotropy (measure of how anisotropic the texture information is)

Cavity/lumen surface to volume ratios

Cavity/lumen edge length to area ratios

What other measures are missing?

Visualisation data:

Mask surfaces

uCT and helix angles on surfaces

Streamlines