This is a GPU-accelerated version of the hepatic artery vessel simulation code. The following are step-by-step directions to build and execute the C++ code. For a complete description of the algorithm, see manuscript:

Whitehead JF, Laeseke, PF, Periyasamy S, Speidel MA,Wagner MG. In silico simulation of Hepatic arteries: An open-source algorithm for efficient synthetic data generation. Med Phys. 2023;1-14. https://doi.org/10.1002/mp.16379

Please cite all uses of this algorithm using the above reference. Any questions, comments, or suggestions may be emailed to joewhitehead1000@gmail.com

Section 1. Installation and Build

1) Download required software to run:

Visual studio 2017 v. 15.0 64-bit (x64) (available at: https://visualstudio.microsoft.com/vs/older-downloads/)

Eigen v. 3.4.0 (available at: https://gitlab.com/libeigen/eigen/-/releases)

CUDA v. 11.5 (available at: https://developer.nvidia.com/cuda-11-5-0-download-archive)

Vessel generation code (https://github.com/jewhiteh/HepaticVesselGeneration)

2) Unzip vessel generation code ("HepaticVesselGeneration-main.zip"). Place downloaded file folders ("eigen-3.4.0" and "CUDA 11.5") in the HepaticVesselGeneration-main folder. Make sure to have unzipped "eigen-3.4.0."

3) In the "HepaticVesselGeneration-main\Vessel_sim_GPU" folder, unzip "BloodDemandMap.7z." (May require download of 7zip (https://www.7-zip.org/)

4) Start Visual studio and go to File -> open -> Project/Solution. Navigate to the "HepaticVesselGeneration-main/Vessel_sim_GPU" folder and open "Vessel_sim_GPU.sln.

5) There are three macros defined at the top of the main.cpp file.

5.1) #define constant_seed: set to one to have a constant random number seed (same tree generated everytime)

5.2) #define Intersection_check: set to one to not have intersection vessels (longer computation time). NOTE: only vessel centerline trees are guaranteed to note intersect, not post-polynomial interpolated vessels.

5.3) #define GPU: set to one for GPU-accelerated tree generation

6) To ensure use of a multi-threaded CPU implementation please ensure openMP support is set to true

Project -> properties -> C/C++ -> Language -> Open MP Support -> **Yes (/openmp)**

7) Hit Build -> Build Solution.

8) You should now be able to execute the code by any of the following:

5.1) Debug -> Start Without Debugging to build the solution. (ctr + F5)

5.2) Run the executable from file explorer. The executable is in Vessel_sim_GPU\x64\Release\Vessel_sim_GPU.exe

Section 2. Using the Code

The code uses the blood demand maps in the "HepaticVesselGeneration-main/Vessel_sim_GPU/BloodDemandMap" folder. Ten initial blood demand maps have been provided. If you would like to use your own blood demand map, you can replace the files in this directory (NOTE: Use the same file name). The algorithm will always use the "Phantom.dat" blood demand map. If you wish to use one of the other 9 blood demand maps provided, simply rename them to "Phantom.dat." For example, "Phantom9.dat" is renamed to "Phantom.dat". For information on how to use your own blood demand map, see Section 3. Phantom Creation.

1) Execute the code. (see Section 1). A terminal window should pop up with the following input parameters. Input your desired parameters.

   Number of trees to build: This is the number of unique hepatic vasculatures the program should create. (Example: "10").

   Number of Endpoints per tree: This is the number of terminal vessel segments per vessel tree. Typical numbers range from 10k to 100k. More endpoints improved the level of detail of the vessel tree at the cost of increased computation time. (Example: "10000").

   Size of X Y and Z dimension of the blood demand map [voxels] (Example: 512 512 512): This is the size of the blood demand map in voxels. For the one provided with the code, input (512 512 512).

   Blood demand map isotropic voxel resolution [mm] (Example: 0.77): This is the isotropic resolution (non-isotropic resolutions not supported) of the blood demand map in millimeters.

   Desired output resolution of vessels [mm] (Example: 0.3): This is the desired output resolution in millimeters of the hepatic vessel tree. A smaller number improved the resolution of the tree at the cost of increasing file size upon write out.

2) Upon execution the code should tell you which branch it is on "Starting branch: X of Y." Once finished generating the initial tree, it will check if it needs to resample any endpoints. The code may need to be resampled if an endpoint could not be connected to the tree (due to causing a vessel intersection). Finally, the code will do one last intersection check.

3) When the Code is finished, the folder "HepaticVesselGeneration-main/Vessel_sim_GPU/CLines" will be populated with text files "radiiX" "seed_X" and "TreeX". X is the tree number. Files radii and Tree are the size and location of each vessel, respectively. Each row in radii and tree correspond to the same vessel branch. For example. Row 22 of radii.txt and Tree.txt are the size and position of the 22nd branch in the vessel tree. The Tree.txt and Radii.txt files are in units of pixels with a resolution specified in Step 1: "Desired output resolution of vessels [mm]." The seed_X.txt file is the random number seed used to generate that vasculature.

   The position of a centerline vessel branch in the Tree.txt file is specified as a space delimited list of x, y, z-coordinates. For example, row 1 of the Tree.txt file is the position for the centerline of the first branch. Points are then listed as X1 Y1 Z1 X2 Y2 Z2 X3 Y3 Z3 etc., where 1, 2, 3 are the first, second, and third point along the vessel centerline. The radius for the vessel branch can then be found in the corresponding row in "Radii.txt".

For example: To read “Tree.txt” into MATLAB, you can call the readTree.mat script using the path to Tree.txt file as an argument. If the second argument is set to 1, an example plot of the tree will also be plotted. For Example:

```
tree = readTree(“Tree.txt”,1);
```

Section 3. Initialize your own blood demand map

Ten initial blood demand maps are provided in the BloodDemandMap folder. Each Phantom.dat file contains a 4D volume (512 x 512 x 512 x 10 unit8). Along the fourth dimension, volumes 1-8 correspond to the Couinaud segment of the liver. For example, volume at index 3 (Phantom(:, :, 3)) would be the third Couinaud segment of the liver. The volume at index 9 represents the possible locations for the proper hepatic to enter the liver volume (where the first source of the hepatic artery will enter the liver). Finally, the volume at index 10 is the entire liver, where blood demand ranges from 0 – 255, where a value of 0 represents no blood demand (impossible for an endpoint to be chosen there) and a value of 255 represents the highest blood demand. Note that the summation of volumes at indices 1-8, should equal the volume at index 10.

To initialize your own blood demand map, you must follow the above format. You may delineate your own 3D liver volume into Couinaud segments and save them as a 4D volume. Alternatively, a function “GenerateLiverVolumn” is provided in the “HepaticVesselGeneration-main/MatlabScripts” folder that will take a 3D binary liver volume (0 – background, 1 – liver), and save a file “PhantomCustom.dat” that may be used as a blood demand map.