VTK Assignment - White Matter Tractography

Sam Alws[1]

[1] Vanderbilt University, Nashville TN 37235, USA

Abstract. This paper describes a white matter tractography tool created using VTK, ITK, and C++. The tool has a simple user interface and runs efficiently using a queue-based algorithm.

Keywords: Tractography, Fractional Anisotropy, Tensor Image, ITK, VTK, Medical Imaging.

1 User Interface

1.1 Display

The tool displays two views at once. On the left side of the screen is a fractional anisotropy image of the provided brain scan. On the right side are the paths created by the tractography algorithm (described in section 3). They are color-coded based on the seed they grew from. See section 5 for screenshots of the display.

1.2 User Interaction

Left-click and hold on either side of the screen to rotate the image. Right-click on a point in the left side of the image to add a point for tractography.

2 Command Line Interface

The tool must be run from the command line. It takes up to 5 command line arguments, and outputs using stdout and by creating image files.

2.1 Command Line Arguments

The tool's command line arguments are the following, in order:

1. The input image, a tensor image to perform tractography on.
2. The input label, a segmentation image which determines the starting points for tractography.
3. The step size, delta (more information on this later).
4. The minimum fractional anisotropy value, minFA (more information on this later).
5. The maximum number of iterations, maxIter (more information on this later).

Arguments 2 through 5 are optional; their values are blank file (“”), 3, 0.2, and 300 respectively.

2.2 Advantages of this Interface

This interface allows for a lot of customization without having to edit the source code. It also allows users who don't need any customization to run the command with a minimal list of arguments.

2.3 Example

Here is an example run of the tool:
./VTK-HW DTIBrain.nrrd segmentation.seg.nrrd 2.5 0.25 100
This command does the following:

- Sets DTIBrain.nrrd as the input tensor image.
- Sets segmentation.nrrd as the input label image.

- Chooses 2.5 as delta, 0.25 as minFA, and 100 as maxIter.

Here is an example run using default parameters:
./VTK-HW DTIBrain.nrrd
This command does the following:

- Sets DTIBrain.nrrd as the input tensor image.
- Chooses 3 as delta, 0.2 as minFA, and 300 as maxIter (default values).

3 Algorithm

3.1 Tractography Algorithm

The tractography algorithm works as follows.

First, read input files and initialize variables (in particular, the command line arguments delta, minFA, and maxIter). Next perform fractional anisotropy on each tensor pixel in the input image; this can be done very easily using ITK's TensorFractionalAnisotropyImageFilter; no default settings need to be changed, aside from setting the input image. [2]

Next, set up a queue, voxelsQueue, which will keep track of the points which need to be looked at, their colors, and their iteration numbers; and a segmentation image, alreadyTouched, initialized to all zeros, with the same dimensions as the input image and input label, which will indicate which points have visited. For each point labeled "1" in the input label (a segmentation image file), add this point to the queue and mark it as "1" on the output image. [3] Next, do the following every frame:

Set up a new queue, nextVoxelsQueue, and initialize it to be blank. Do the following until voxelsQueue is empty:

First, pop off a single point from voxelsQueue. Determine the current iteration number by reading the value of this point from the output image, and make sure that it is not over maxIter; also, determine the fractional anisotropy number at this point using the output of the filter from before, and make sure that it is not under minFA. Next, determine the principal eigenvector of the input image at this point (more information on this later). Next, multiply the principal eigenvector by delta, the step size; create new points addedVoxel and subbedVoxel by adding and subtracting the multiplied eigenvector onto the original point (and rounding to the nearest voxel). For both of these points, check that they are not out-of-bounds of the image, check that they are blank (marked "0") in the output image, add the points and their colors (same as current color) and iteration numbers (equal to the current iteration number plus 1) to newVoxelsQueue, mark them with 1 in alreadyTouched, and draw them in the VTK display.

Once voxelsQueue is empty, replace voxelsQueue with nextVoxelsQueue and return.

3.2 Advantages of this Approach

Advantages of Local Iteration Numbers

There are multiple advantages to using iteration numbers that increase locally (i.e. tied to a point, and using the current point's number to determine the next points' numbers) instead of globally (i.e. by having a global variable outside of the loop which increases by 1 for each new point added). First of all, each point's number can be used to determine intuitively how "far" it is from the initial starting point – indicating how many delta-sized steps one would need to walk from the initial state to the given point, rather than only indicating the number of steps taken internally by the algorithm, which is more of an implementation detail than a particularly useful metric. In addition, it ensures that iteration numbers don't need to get excessively high in order to get a competent output image – the total global iteration number can get in the tens of thousands when the local iteration number is in the hundreds, and in theory the global iteration number is $O(2^n)$ in comparison to the local iteration number.

Advantages of Using a Queue

By using a queue to keep track of points that need to be visited, we are performing a breadth-first search instead of a depth-first search (which would happen if we used a stack). Because of this, every point in the queue will have an iteration number which is either the same as the first element or one more than the first element. Thus, each point's iteration number will be the lowest possible number of steps needed to reach it, and the algorithm will never be able to assign a point with a lower number (and have to revisit it). Also, once the iteration number of one point being visited reaches maxIter, the loop can exit immediately, instead of checking out all other elements in the queue, since it is known that all the other points in the queue will also have maxIter as their iteration number.

Advantages of Marking a Point Before Enqueueing it

By marking a point on the output image before enqueueing it (rather than while visiting it), we can ensure that any given point will never get added to the queue twice. If the algorithm tries to enqueue it again, it will find that the point is already marked (even though it hasn't been visited yet) and will not add it to the queue again. This reduces both the space and the time needed to run the program.

3.3 Principal Eigenvector Calculation

Principal eigenvector calculation is done individually on each pixel visited. It is done with the help of the itk::DiffusionTensor3D::ComputeEigenAnalysis method. [2] This method returns a 3-element array of eigenvalues, and a 3-by-3 matrix of eigenvectors. The principal eigenvector can be finding the largest element in the list of eigenvalues, and choosing the corresponding column in the matrix of eigenvectors. Each eigenvector in the matrix is a unit vector, so it only needs to be multiplied by delta and rounded to the nearest integer in order to determine the step vector to the next pixel to visit.

4 Build Instructions

The tool can be built relatively easily using cmake. In order to do this, the user must download the source code, make a separate build folder, cd into the build folder, run ccmake ../[Source Code], and run make. An output file, VTK-HW, will be made. Note that VTK, ITK, and cmake are dependencies of this tool.

5 Parameters and Results

5.1 Results

The following figure shows the tool's tractography output given different inputs.

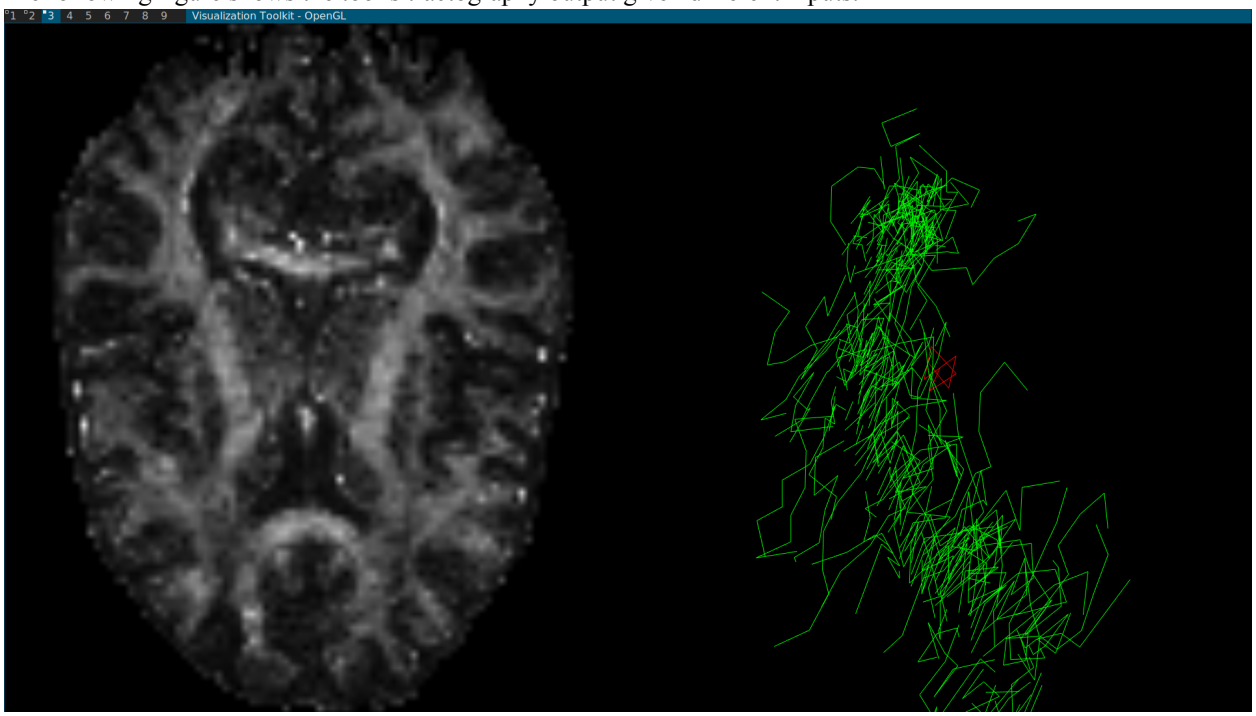

Figure 1. Tractography Output with Input Segmentation

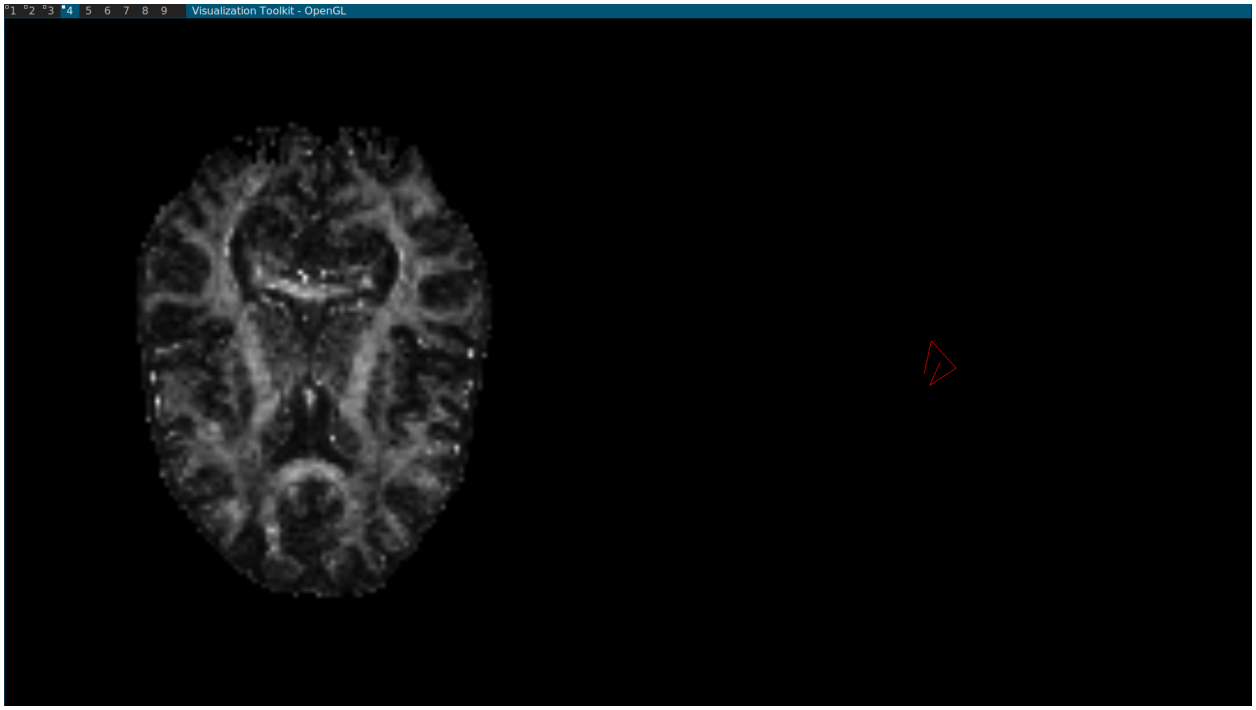

Figure 2. Tractography Output With No Segmentation Inputted

5.2 Choice of Parameters

Parameter values were chosen by experimentation. A value of 3 was chosen for delta; higher and lower values of delta resulted in shapes with more holes. A value of 0.2 was chosen for minFA; lower values resulted in too many voxels being selected, and higher values resulted in barely any voxels being selected. A value of 300 was chosen for maxIter; this iteration number is never reached in this example. This is because letting the algorithm run in full without exiting early results in a good image.

References

1. Mori S, van Zijl PC. Fiber tracking: principles and strategies - a technical review. NMR Biomed. 2002 Nov-Dec;15(7-8):468-80. doi: 10.1002/nbm.781. PMID: 12489096.
2. ITK Documentation, https://itk.org/Doxygen/html/, last accessed 2021/12/08.
3. C++ Reference, https://en.cppreference.com/w/, last accessed 2021/12/08.
4. VTK Documentation, https://vtk.org/documentation/, last accessed 2021/12/08.