Primitive Tractography User Manual (incomplete)

Correspondance: Daniel Deiros Hernández ([ddeir094@uottawa.ca](mailto:ddeir094@uottawa.ca))

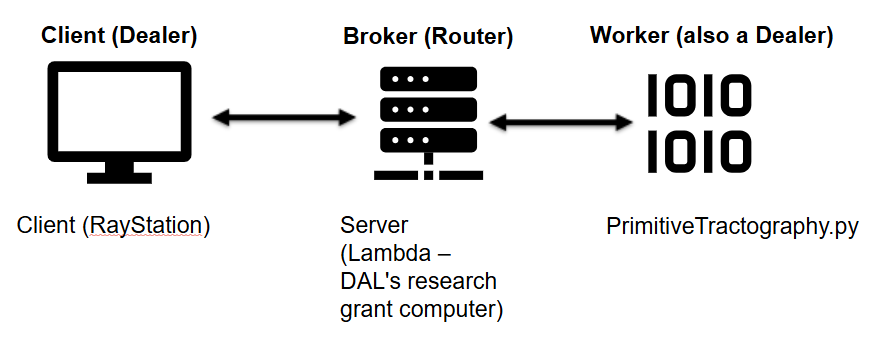
# Introduction

This manual intends to guide users looking to learn a bit more about the process behind the Primitive Tractography scripts, or refresh seasoned veterans looking for a refresher. Please redirect any questions not answered to your satisfaction to my email at [ddeir094@uottawa.ca](mailto:ddeir094@uottawa.ca).

# Overview

To start, *Primitive Tractography* refers to a collection of scripts and subscripts written in Python. Included in these are PrimitiveTractography\_Client.py, PrimitiveTractography\_Server.py, and PrimitiveTractography.py. For simplicity, PrimitiveTractography\_Client.py will be referred to as the client, PrimitiveTractography\_Server.py will be referred to as the server, and PrimitiveTractography.py will retain its name when being referred to.

To explain the basic layout of the code, a user runs the client, which defines a specific base directory via the get\_base\_dir() function in Preliminaries.py and sends this folder path to the server (Lambda computer) where PrimitiveTractography.py is called and looks for dMRI data in that folder under a folder called “Combined”. As it proceeds through the tractography pipeline, new folders and files are saved to the base folder, where RayStation (client) can then access the WMPL map and create the CTV. Refer to figure 1 below to gain some idea of these connections. You may ignore the labels above the images if you do not quite understand them. They are terms used to define the components of this system for ZMQ (a Python package used to connect the client to the server) purposes.



### Figure 1. Simple diagram illustrating connection between the three main scripts in use.

Now, a user may run the client via RayStation or directly via Python, however, running via Python means that some functions may not be accessed, such as accessing the registrations. It is recommended that a user first run the complete script via RayStation, so that necessary DICOM files are automatically uploaded to the proper folder. Otherwise, these DICOM files (including the CT image set, MR image set (with FA), and RT Struct) would need to be manually uploaded by the user onto the specific folder required. However, this is possible.

Running the script via Python directly will not create any CTVs in RayStation, so, if one simply desires to view the Fury plots, it is certainly possible to do this via Python. Simply running the client will detect whether RayStation is available or not.

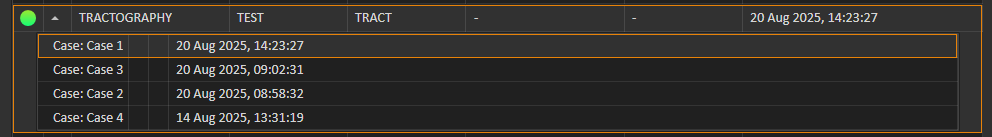
# RayStation Access/ Folder Locations

This section will briefly explain how one may access the test patients on RayStation. Also, it will go over where the dMRI scans and the tracts are currently being held

Currently, one must go to the RayStation research app as indicated below in figure 2.

### Figure 2. RayStation research app.

This is available via the Citrix workspace. Next, all of the current cases that have been identified to have dMRI data are on a “patient” called “TEST TRACTOGRAPHY” as indicated in figure 3 below.



### Figure 3. Tractography patient in RayStation research.

Currently, cases 1-3 have dMRI data available. This data is currently stored in the V drive under the path *V:\Common\Staff Personal Folders\DanielH\DICOM\_Files\TractographyPatient* where then case 1, for example, is stored under *Case 1 RS*. In this folder, the *Combined* folder contains the raw dMRI data we need to start the tractography. **This folder must not be deleted**.

Once the user has opened the case they would like to perform tractography on, they can go to scripting available on the left tab, click “Script creation...” and open the PrimitiveTractography\_Client.py script available via *V:\Common\Staff Personal Folders\DanielH\RayStation\_Scripts\Tractography*.

Before one runs this, they must ensure the server, which is in the same *Tractography* folder as the client above, is running. To do this, remote into the Lambda computer, with TOHGOCCD222976, then open the server via VSCode or your preferred IDE, and run it. You should see “Waiting for client message...” in the terminal. Alternatively, if the server was already running, then the last message should be that the previous client missed a heartbeat and is assumed to be disconnected. This is expected when the script is finished. It is recommended to restart the server (i.e. shut it down and run it again) every once in a while. Sometimes I have received strange errors involving incorrect addresses when the server is running for too long. To shut down the server, simple click into the terminal and press Ctrl+C. This should print the message “Shutting down server from user (KeyboardInterrupt)”. Once the server is shut down, one may run it again.

When you have confirmed the server is running, you may run the client. First, the client will check to see if we have the ROIs required, if not, then it will collect them from RayStation before connecting to the server. Once connected to the server, the server will run PrimitiveTractography.py script. The client first tells the server what the base directory is, which, for case 1, would be *V:\Common\Staff Personal Folders\DanielH\DICOM\_Files\TractographyPatient\Case 1 RS*, then the server uses this folder to extract the raw dMRI data, the RayStation ROIs, and save data there too.

# Tractography

There are several key parameters that need to be defined to input into DiPy functions. This section intends to explain their use and the current value, while going through the pipeline and functions in place.

## Median\_otsu()

This function pre-processes the dMRI image set to remove noise.

## TensorModel() & White Matter Mask

This function applies a tensor model using DTI to the gradients. From this model we apply .fit() to fit the data and extract the FA. From this FA, mapped across the images, we use an FA threshold of 0.15 to create a white matter mask. So any voxel with an FA higher than 0.15 is a part of this mask. 0.15 is the same value from the paper. Increasing this threshold will result in the generation of less tracts, as well as them stopping earlier (since we use the white matter mask in the generation process).

## CsaOdfModel()

Applies the CSA ODF model to the gradients. Currently using a spherical harmonic order of 4 (2 orders below example, I think 6 was giving me issues but should be tested at 6 again).

## Peaks\_from\_model()

Applies model to our data and returns peaks and metrics required to generate tracts. We input default\_sphere (from example), a relative\_peak\_threshold of 0.5 (from DiPy: Only return peaks greater than relative\_peak\_threshold \* m where m is the largest peak). The effect of increasing the relative\_peak\_threshold decreases the number of tracts we see. The paper uses 0.5. We also apply a minimum separation angle of 15 degrees (same as paper). For two peaks within this range, we choose the higher peak.

## ThresholdStoppingCriterion()

Defines where we stop the tract. Currently set at FA=0.15 (same as paper and as white matter mask threshold).

## Random\_seeds\_from\_mask()

Generates seeds on a mask. We input seeds\_count=1 and make seed\_count\_per\_voxel=True such that we have 1 seed generated per voxel. Paper did the same. For deterministic tracking, 1 seed per voxel is really the max we can do from my understanding (a second seed in the same voxel would produce the same tract).

## Eudx\_tracking() & Streamlines()

Applies EuDX tracking. Uses a step size of 0.5 mm (our voxels are 1.5 mm long, step size should be smaller than this). Step size is default value, I haven’t toyed around with it but making it bigger should make our tracts less accurate. We also apply a maximum angle of 60 degrees like the paper. From my understanding this is the maximum angle that a tract may change between steps. This initializes the EuDX algorithm while Streamlines() applies it.

# CTV Maker

This section outlines some parameters inputted into the DTI CTV Maker script.

First, we set the MyThreshold value to the lowest number from the WMPL map, which should be –1024. This MyThreshold we determine the PL maps with specific thresholds we want. For example, for tracts within 1 cm, we set a LowThreshold of MyThreshold and a HighThreshold of MyThreshold+10 (10 mm).

Next, we set MaxNumPts=2500. This number is 2000 in the paper. This simple tells RayStation the max number of points it is allowed to use when contouring the PL maps.

Penultimately, AreaThreshold is set to 1 like in the paper. This is the smallest area a contour may be. Any area smaller than this is ignored.

Finally, I have set an expandingVar of 0.5 mm like the paper. This means we expand 0.5 mm in all directions from the end of the tract. Hence, it is important to understand that the ROIs which say CTV DTI 2 cm may have a maximum distance of 2.5 cm from the GTV. Note that this isn’t necessarily likely, since this would suggest a perfectly straight tract for 2 cm. However, it is common to see several “bubbles” which make the anisotropic “2 cm” peak above the isotropic 2 cm.