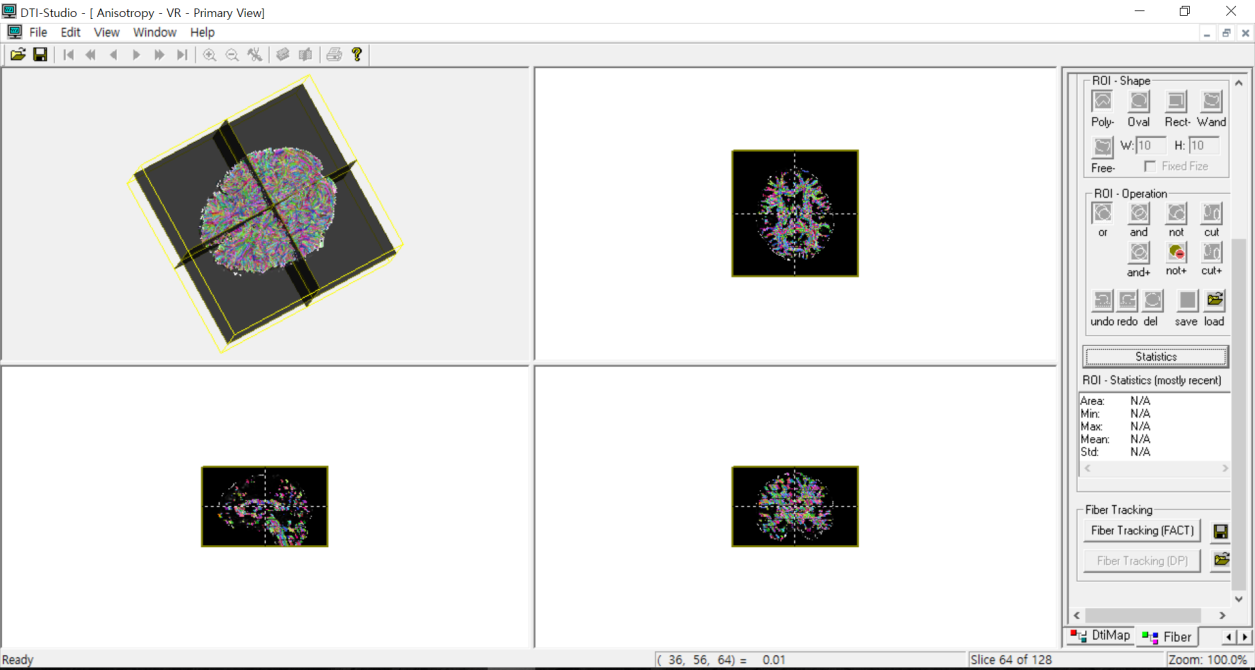
**BM Imaging Report 2**

2014250136 류제성

DWI image를 이용하여 Structure Connectivity를 알아보았다. 그 중에서도 FACT algorithm을 이용하여 Fiber로 추정되는 위치를 찾고, Voxel별 FA value값을 찾아 Connectivity를 알아보는 과정이었다. 우선 DTI studio를 이용, 주어진 gradient를 주고, Tensor를 Fitting하고, 일정 parameter으로 Fiber tracking을 진행하고, FA value 및 fiber data를 저장하였다. 그 자료들을 활용한 Connectivity matrix를 만드는 과정은 D.code 를 참고하면 될 것이다.



a) DTI studio 이용하여 tensor fitting 후 fiber tracking한 image의 모습

cf)

FA(Fractional anisotropy): tensor eigenvalue 값들의 nonuniformity. 0과 1사이에 분포하며 FA값이 크다는 것은 그 방향으로 강하게 향해 있는 것이다.

RA(relational anisotropy): 상대적인 비등방성.

VR(volume ratio): 부피비. fiber 길이 구할 때 voxel단위로 생각할 시 고려하였다.

radial diffusivity : average of 2 lesser eigenvalues. 방사 확산성.

PDD(principle diffusion direction): deterministic tractography에서 fiber 진행방향. FA value 값과 관련.

**A. Observations of tractography of various parameter changes**

정상인환자(01\_01)의 image를 기준으로 tractography를 관찰하였다.

i)Start point, end point change

FA value start, end point를 0부터 1 사이에 0.2씩 증가시키면서 Fiber tracking을 하고 fiber 개수와의 연관성을 살펴 보았다. turning angle은 45도로 일정하게 유지했다.

|  |  |
| --- | --- |
| Start&end  FA value | data |
| 0.2 |  |
| 0.4 |  |
| 0.6 |  |
| 0.8 |  |

지정한 FA value 값이 커질수록 우선 Fiber의 개수가 현저히 줄어드는 양상이다. 예를 들면 FA value를 0.2로 지정했다면 FA값이 0.2 이상인 것들을 Fiber로 보겠다는 의미이기 때문에, 그만큼 Fiber로 추출되는 지점이 많다는 것이므로 당연한 결과이다. 이외에도 Maximum length, Average length 등과 같은 값도 Fiber 개수에 따라 감소하는 모습을 보인다.

ii)Turning angle change

Starting, ending point의 FA value를 0.2로 고정시키고 turning angle을 15도,30도,45도,60도,75도로 변화시켜 보았다.

|  |  |
| --- | --- |
| Turning angle(degree) | data |
| 15 |  |
| 30 |  |
| 45 |  |
| 60 |  |
| 75 |  |

turning angle이 커질수록 Fiber의 개수는 증가한다. 예를 들면 turning angle이 45도라고 하면 이전 진행 방향 기준 45도 이하로 꺾이는 것으로 보이는 지점만 Fiber가 이어지는 것으로 보겠다는 것이고, turning angle이 증가할수록 심하게 꺾여도 Fiber로 보겠다는 의미이다. 대체로 Maximum length, average length도 angle을 높일수록 증가하는 양상을 보인다.

i),ii)를 통해 Parameter를 변화시켜 가며 tractography를 관찰한 결과, 설정하는 값에 따라 tractography의 Fiber의 개수가 달라졌다. 이는 parameter는 설정하기 나름이며 Fiber를 직접적으로 tracking할 수 없고 추론을 통해 확인하는 것과 통하는 의미라고 볼 수 있다.

**B. images of structural connectivity**

|  |  |  |
| --- | --- | --- |
|  | Normal(01) | Abnormal(02) |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

한눈에 봐도 확연히 차이가 난다. 90\*90 matrix를 이용하여 connenctivity를 나타낸 그래프이기에 대각성분이 모두 0이고 symmetric하다. favalue 평균값의 합, 즉 weight값이 클수록 밝은 색을 띠게 구분되는 그래프인데, 정상인의 뇌가 AD 또는 SVaD를 앓고 있는 환자의 뇌보다 connectivity에 대응되는 FA value weight가 대체로 큰 것을 알 수 있다.

**C. Qualitative comparison of degree distributions between normal and abnormal subjects**

|  |  |  |
| --- | --- | --- |
|  | Normal(01) | Abnormal(02) |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

한눈에 들어오는 가장 큰 차이는 degree가 [0,80] 범위에 있는 node의 개수이다. 정상인은 degree가 80이하인 node 개수가 40을 웃도는 반면, 환자는 50을 훌쩍 넘는다. 이는 환자 군에 degree가 낮은 node가 많다는 의미이므로, 결국 connectivity 가 떨어진다는 의미이다. degree는 node에 연결된 edge의 개수를 의미하기 때문이다. 이 때문에 degree distribution은 뉴런들이 연결된 정보를 아는 데 필수적이다.

한마디로, degree distribution에서 환자 군의 degree 값이 대체로 더 낮게 분포하고 있으므로, 정상인에 비해 connectivity가 약하다고 분석된다.

D. Code

# -\*- coding: utf-8 -\*-  
import matplotlib.pyplot as plt  
import numpy as np  
import nibabel as nib  
import mpmath  
  
  
'''  
Written by Seung-Wook Kim on May 4, 2018.  
  
Reference:  
 https://www.mristudio.org/wiki/faq  
'''  
  
class fiber():  
 def \_\_init\_\_(self):  
 self.nLength=0  
 self.nSelStatus=[]  
 self.rgbColor=[]  
 self.nSelLenStart=0  
 self.nSelLenEnd=0  
 self.pxyzChain=[]  
  
class Fibers():  
 def \_\_init\_\_(self):  
 self.FiberNr=0  
 self.FiberLenMax=0  
 self.FiberLenMean=0  
 self.ImgWidth=0  
 self.ImgHeight=0  
 self.ImgSlice=0  
 self.PixelSizeWidth=0  
 self.PixelSizeHeight=0  
 self.SliceThickness=0  
 self.SliceOrientation=0  
 self.SliceSequencing=0  
 self.Fiber=[]  
  
'''  
use read\_fiber to load the fibers from .dat file.  
output.Fiber contains the fiber informations.  
'''  
def read\_fiber(file):  
 new\_fiber = Fibers()  
 fid=open(file)  
  
 for i in range(2):  
 chk1=np.fromfile(fid,np.dtype(np.int32),1)  
   
 new\_fiber.FiberNr = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.FiberLenMax = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.FiberLenMean = np.fromfile(fid,np.dtype(np.float32),1)  
 new\_fiber.ImgWidth = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.ImgHeight = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.ImgSlice = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.PixelSizeWidth = np.fromfile(fid,np.dtype(np.float32),1)  
 new\_fiber.PixelSizeHeight = np.fromfile(fid,np.dtype(np.float32),1)  
 new\_fiber.SliceThickness = np.fromfile(fid,np.dtype(np.float32),1)  
 new\_fiber.SliceOrientation = np.fromfile(fid,np.dtype(np.int32),1)  
 new\_fiber.SliceSequencing = np.fromfile(fid,np.dtype(np.int32),1)  
   
 for i in range(19):  
 chk2=np.fromfile(fid,np.dtype(np.int32),1)  
   
 for i in range(new\_fiber.FiberNr[0]):  
 n=fiber()  
 n.nLength = np.fromfile(fid,np.dtype(np.int32),1)  
 for j in range(3):  
 chk3=np.fromfile(fid,np.dtype(np.int32),1)  
   
 for j in range(n.nLength[0]):  
 pxyz = np.fromfile(fid,np.dtype(np.float32),3)  
 n.pxyzChain.append(pxyz)  
   
 new\_fiber.Fiber.append(n)  
  
 return new\_fiber  
  
'''  
use load\_img to load .img files you saved from DTI Studio  
'''  
def load\_img(file):  
 fid=open(file,'rb')  
 data=np.fromfile(fid,np.dtype('float32'))  
 fa=data.reshape((70,128,128))  
   
 return fa  
  
'''  
use load\_nifti to load given aal.nii.gz file.  
if you don't use this function and load the image yourself,   
then the image direction and dimension might be wrong with others.  
'''  
def load\_nifti(file):  
 aal=nib.load(file)  
 aimg=aal.get\_data()  
 aimg=np.rot90(aimg,3)  
   
 cimg=np.zeros([70,128,128],dtype=np.int32)  
 for i in range(70):  
 cimg[i,:,:]=aimg[:,:,i]  
   
 return cimg  
  
for q in range(1,3):  
 for w in range(1,6):#01\_01~02\_05까지  
 fadata=load\_img('../scripts/DWI\_data/0%d\_0%d\_Image.img'%(q,w))  
 aaldata=load\_nifti('../scripts/DWI\_data/0%d\_0%d\_aal.nii.gz'%(q,w))  
 fiberdata=read\_fiber('../scripts/DWI\_data/0%d\_0%d\_Fiber.dat'%(q,w)) # fa,fiber,aal data 불러오기  
  
 aal\_value = open('../scripts/DWI\_data/aal\_values.txt')#aal value txt 불러오기  
 aal\_list=[]  
 for line in aal\_value :  
 a=int(line.strip())  
 aal\_list.append(a) #각성분을list로 만듦  
  
  
 start\_list=[0]\*len(fiberdata.Fiber)  
 end\_list=[0]\*len(fiberdata.Fiber) #fiber시작,끝점의 intensity값 list  
 start\_point = [0] \* len(fiberdata.Fiber)  
 end\_point = [0] \* len(fiberdata.Fiber)#fiber길이 구하기위한 시작,끝점의 위치값 list  
 length\_list=[0]\*len(fiberdata.Fiber) #길이 list  
 fa\_sum\_list=[0]\*len(fiberdata.Fiber) #fa값의 합 list  
 fa\_avg\_list=[0]\*len(fiberdata.Fiber) #fa값의 평균 list 즉 fa 합/길이  
  
 '''fiber시작점 intensity, 위치 구하는 for문 '''  
 for I in range(len(fiberdata.Fiber) ):  
 for i in range(len(fiberdata.Fiber[I].pxyzChain)):  
 AAL = aaldata[int(fiberdata.Fiber[I].pxyzChain[i][2])][int(fiberdata.Fiber[I].pxyzChain[i][1])][int(fiberdata.Fiber[I].pxyzChain[i][0])]  
 idx=0  
 for j in range(90):  
 if AAL == aal\_list[j]:  
 idx=AAL  
 if idx!=0:  
 start\_list[I]=idx #intensity 반환  
 start\_point[I]=i #위치 반환  
 break  
  
 '''fiber끝점 intensity, 위치 구하는 for문 '''  
 for k in range(len(fiberdata.Fiber)):  
 for i in range(len(fiberdata.Fiber[k].pxyzChain)-1,-1,-1):  
 AAL2 = aaldata[int(fiberdata.Fiber[k].pxyzChain[i][2])][int(fiberdata.Fiber[k].pxyzChain[i][1])][int(fiberdata.Fiber[k].pxyzChain[i][0])]  
 idx2=0  
 for j in range(90):  
 if AAL2 == aal\_list[j]:  
 idx2=AAL2  
 if idx2!=0:  
 end\_list[k]=idx2 #intensity 반환  
 end\_point[k]=i #위치 반환  
 break  
  
 '''시작점,끝점 위치 이용해서 fiber길이 구하는 for문 '''  
 for m in range(len(fiberdata.Fiber)):  
 length=0  
 for n in range(start\_point[m],end\_point[m]):  
 z\_length=abs(fiberdata.Fiber[m].pxyzChain[n][2]-fiberdata.Fiber[m].pxyzChain[n+1][2])  
 y\_length=abs(fiberdata.Fiber[m].pxyzChain[n][1]-fiberdata.Fiber[m].pxyzChain[n+1][1])  
 x\_length=abs(fiberdata.Fiber[m].pxyzChain[n][0]-fiberdata.Fiber[m].pxyzChain[n+1][0])#z,x,y좌표 각각 차이  
 length+=mpmath.sqrt((1.718\*x\_length)\*\*2+(1.718\*y\_length)\*\*2+(2\*z\_length)\*\*2)  
 if float(length)>=20: #20mm이상인것만 반환  
 length\_list[m]=float(length)  
  
 '''fa 합 및 fa평균 구하는 for문 '''  
 for m in range(len(fiberdata.Fiber)):  
 fa\_sum = 0  
 for n in range(len(fiberdata.Fiber[m].pxyzChain)):  
 fa\_sum+=fadata[int(fiberdata.Fiber[m].pxyzChain[n][2])][int(fiberdata.Fiber[m].pxyzChain[n][1])][int(fiberdata.Fiber[m].pxyzChain[n][0])]  
 fa\_sum\_list[m]=fa\_sum #fa sum  
 if length\_list[m]!=0:  
 fa\_avg\_list[m]=fa\_sum\_list[m]/length\_list[m]#length로 나눠 fa평균  
  
 '''fa value 행렬 만들기 '''  
 matrix = np.zeros([90,90],dtype=np.float32) #90\*90 영행렬 만듦  
 for n in range(len(fiberdata.Fiber)):  
 for i in range(90):  
 if start\_list[n]==aal\_list[i]:  
 I=i  
 break  
 else:  
 continue #intensity 값 이용하여 행렬 세로 성분 해당하는 값 반환  
 for j in range(90):  
 if end\_list[n]==aal\_list[j]:  
 J=j  
 break  
 else:  
 continue #intensity 값 이용하여 행렬 가로 성분 해당하는 값 반환  
 favalue=fa\_avg\_list[n]  
 if favalue!=0:  
 a=matrix[I][J]  
 a+=favalue  
 matrix[I][J]=a #행렬에 fa값 더해가기  
 matrix[J][I]=a #행렬 symmetric 하게 만들어주기  
 for i in range(90):  
 a=matrix[i][i]  
 a=0  
 matrix[i][i]=a #행렬 대각 성분 0으로 만들어주기  
 plt.imshow(matrix)  
 plt.colorbar()  
 plt.clim(0,80) #이미지를 확인하기에 적절한 값으로 80선택  
 plt.show() #connectivity diagram 띄움  
  
 '''degree distribution 확인하는 그래프 코드'''  
 fa\_degree=[]  
 for i in range(90):  
 line\_sum = 0  
 for j in range(90):  
 line\_sum += matrix[i][j]  
 fa\_degree.append(line\_sum) #합들 모두 구해서 append  
  
 histogram = plt.figure()  
 plt.hist(fa\_degree, bins=5, range=[0, 400])  
 plt.yticks(range(0, 55, 5)) #결과값 나오기에 적절한 값들로 설정했음  
 plt.title("degree distribution")  
 plt.xlabel("degree")  
 plt.ylabel('number of node')  
 plt.show() #막대그래프 띄움