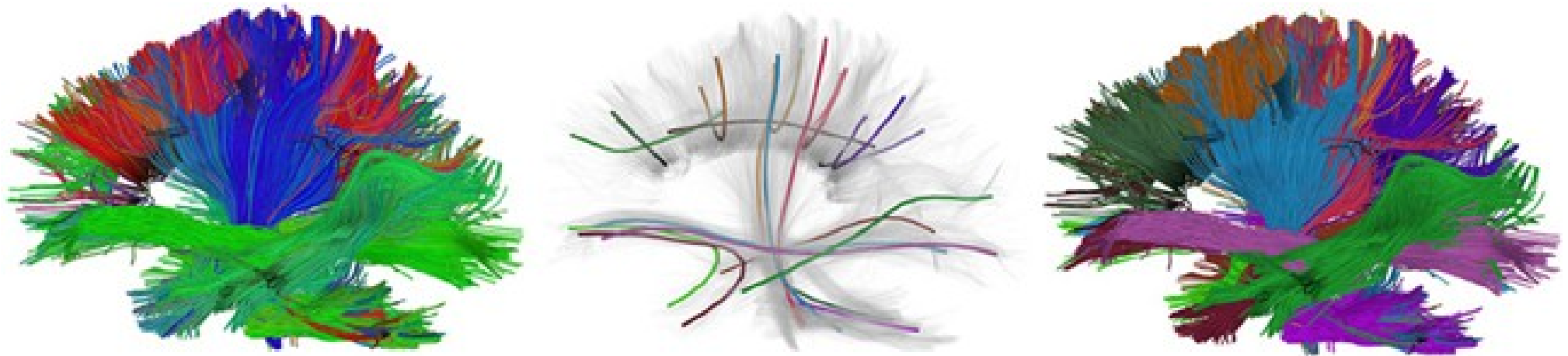
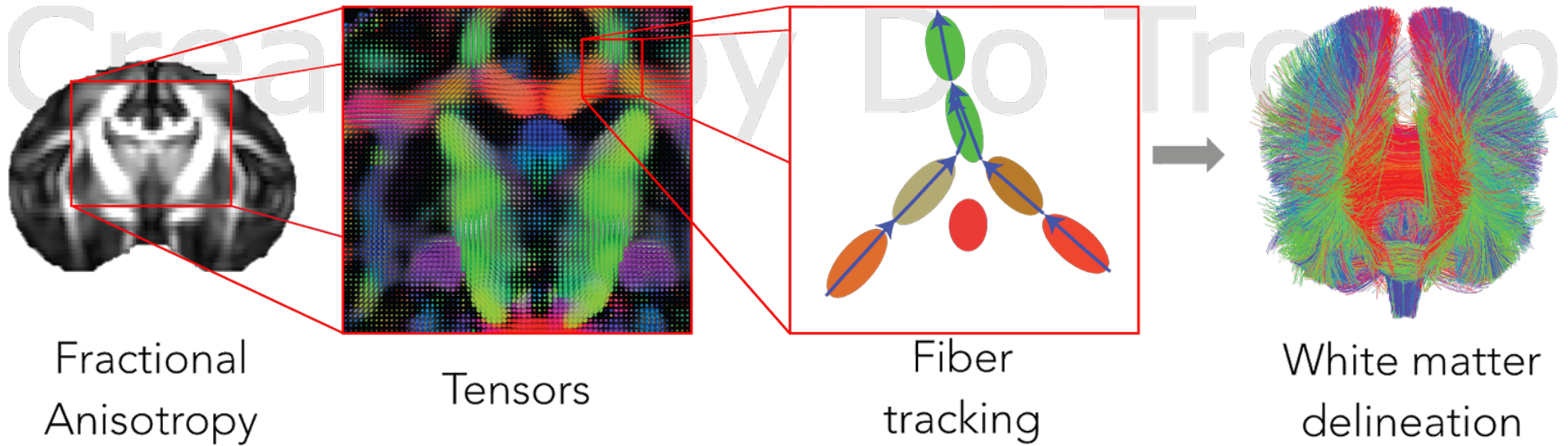


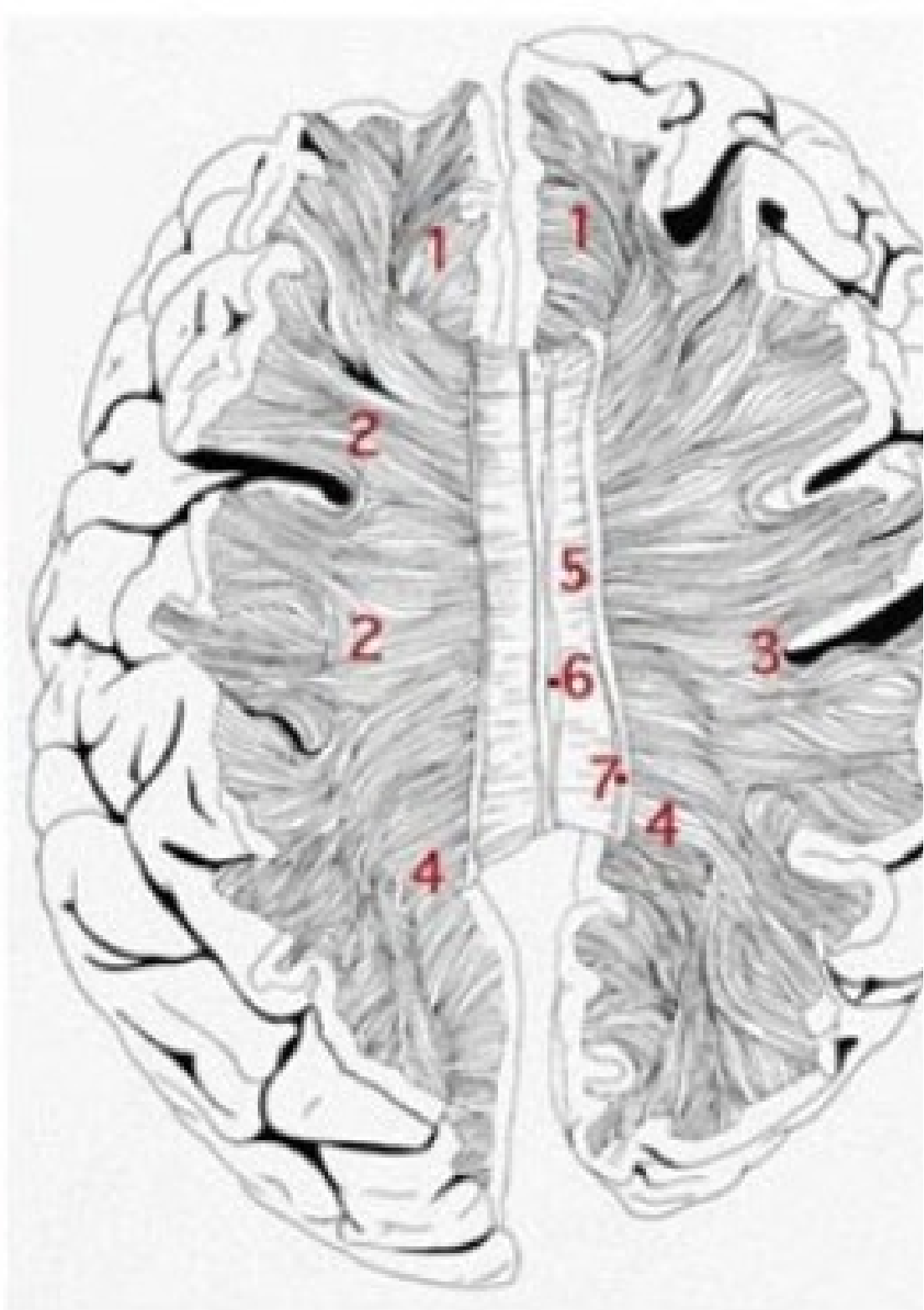
Tractography tutorial



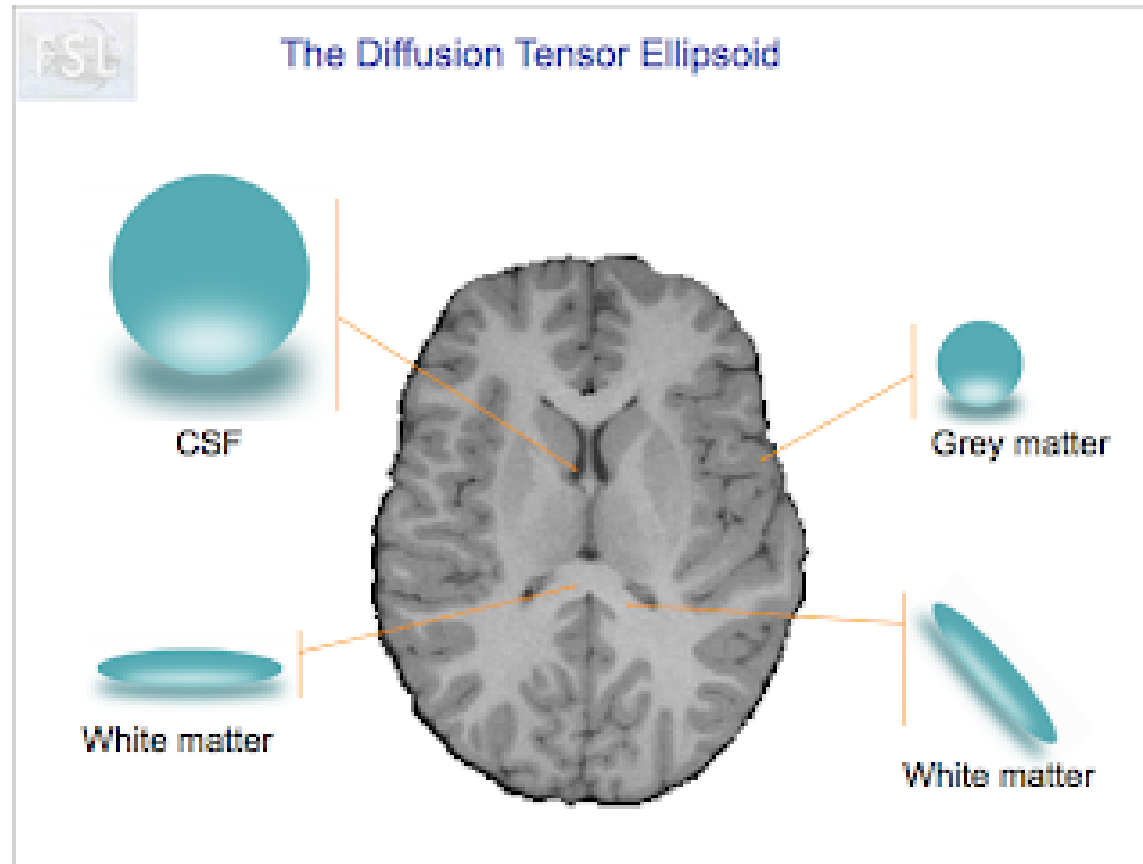
Dr. Alessandro Crimi

Overview





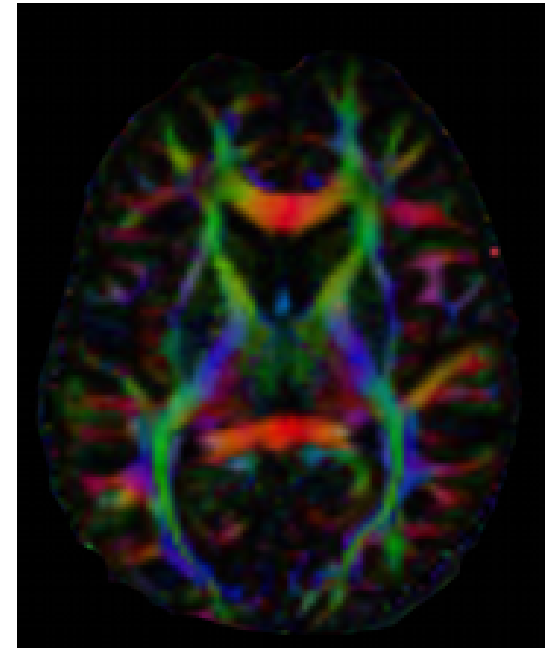
Axons are what we want



$$\underline{D} = \begin{bmatrix} \text{axial} & \text{axial} & \text{axial} \\ \text{sagittal} & \text{sagittal} & \text{sagittal} \\ \text{coronal} & \text{coronal} & \text{coronal} \end{bmatrix} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \rightarrow \text{Ellipsoid}$$

Diffusion files

- 1) A dwi data file.
- 2) A bvals files contains a scalar value for each applied gradient, corresponding to the respective b-value.
- 3) A bvecs contains a 3x1 vector for each gradient, indicating the gradient direction. The entries in bvals and bvecs are as many as the number of volumes in the dwidata file. The i th volume in the data corresponds to a measurement obtained after applying a diffusion-sensitising gradient with a b-value given by the i th entry in bvals and a gradient direction given by the i th vector in bvecs.



Jupyter Notebooks

First, install Ipython:

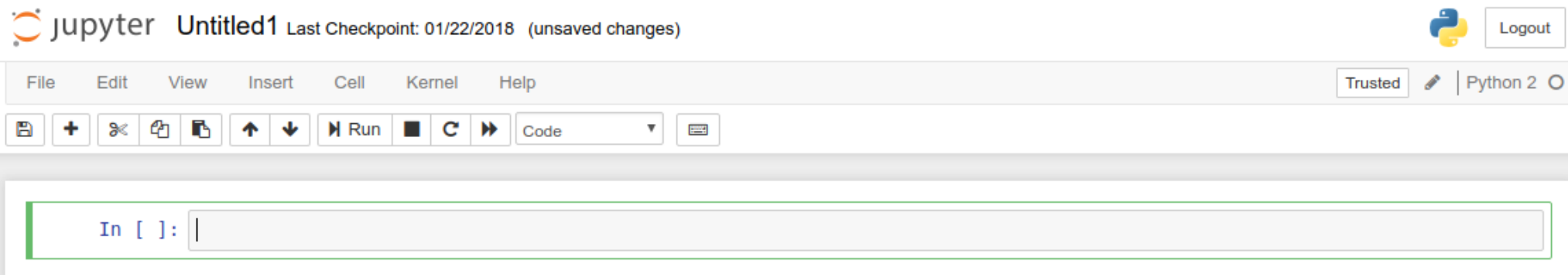
```
sudo apt-get -y install ipython ipython-notebook
```

Now we can move on to installing Jupyter Notebook:

```
sudo -H pip install jupyter
```

Running Jupyter Notebook

```
jupyter notebook
```



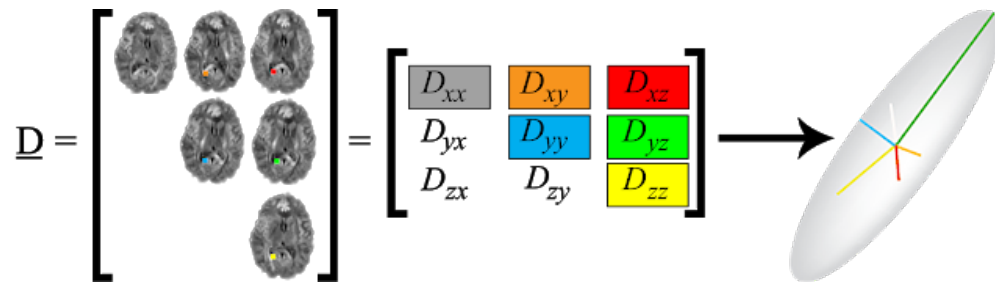
Load data

```
import dipy
import numpy as np
import nibabel as nib

fimg = "dti_fm_bet.nii.gz"
img = nib.load(fimg)
data = img.get_data()
affine = img.get_affine()
header = img.get_header()
voxel_size = header.get_zooms()[:3]
mask, S0_mask = median_otsu(data[:, :, :, 0])
fbval = "../.../bvals"
fbvec = "../.../bvecs"

bvals, bvecs = read_bvals_bvecs(fbval, fbvec)
gtab = gradient_table(bvals, bvecs)
```

We use the fractional anisotropy (FA) of the DTI model to build a tissue classifier.



$$FA = \sqrt{\frac{1}{2} \frac{\sqrt{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}}}$$

```
ten_model = TensorModel(gtab)
ten_fit = ten_model.fit(data, mask)
```

```
from nibabel import trackvis as tv
from dipy.tracking.streamline import
set_number_of_points
from dipy.segment.mask import median_otsu
from dipy.io import read_bvals_bvecs
from dipy.core.gradients import gradient_table
from dipy.reconst.dti import TensorModel
from dipy.reconst.dti import fractional_anisotropy
from dipy.reconst.dti import color_fa
from dipy.reconst.shm import CsaOdfModel
from dipy.data import get_sphere
from dipy.reconst.peaks import peaks_from_model
from dipy.tracking.eudx import EuDX
from dipy.tracking.utils import density_map
from dipy.tracking import utils
```

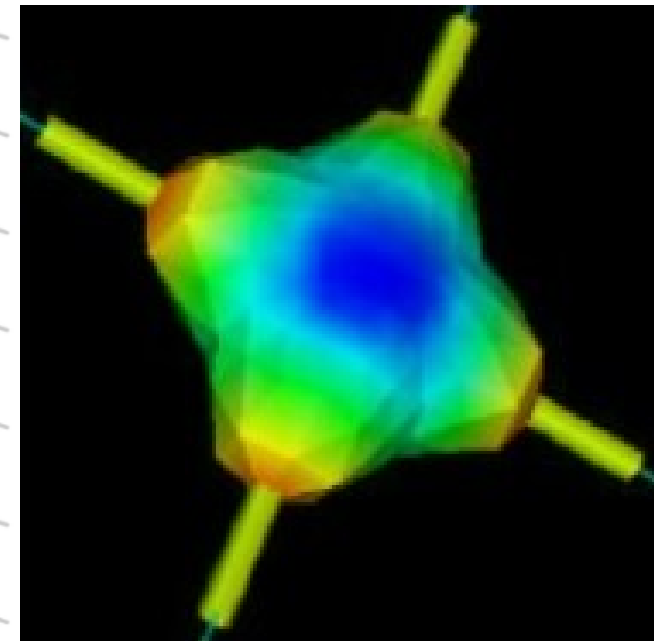
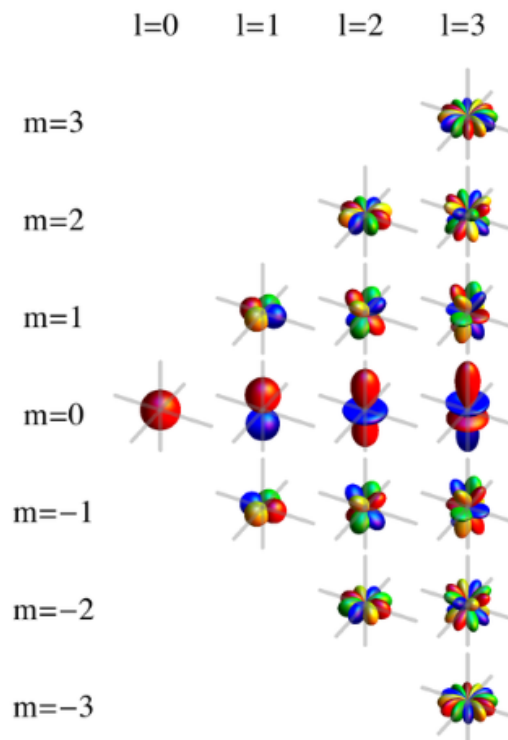
```
fa = fractional_anisotropy(ten_fit.evals)
cfa = color_fa(fa, ten_fit.evecs)
```


Constant Solid Angle

- The DTI model is limited by the “Gaussian assumption” → We use spherical harmonics and Orientation Distribution Functions (ODF).
- So, in order to get the main direction of diffusion we fit the data to a Constant Solid Angle ODF Model. The peaks of an ODF are good estimates for the orientation of tract segments at a point in the image:

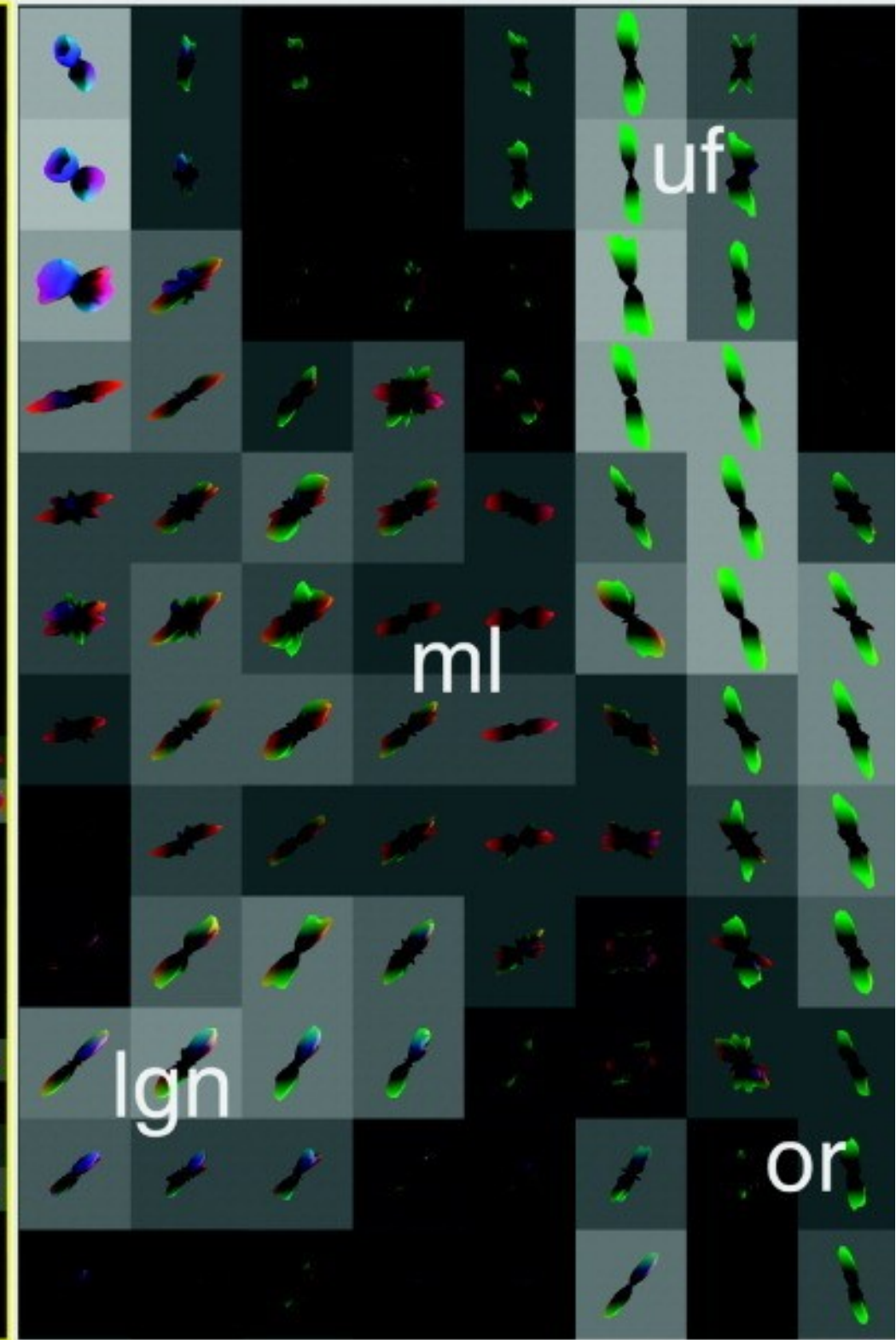
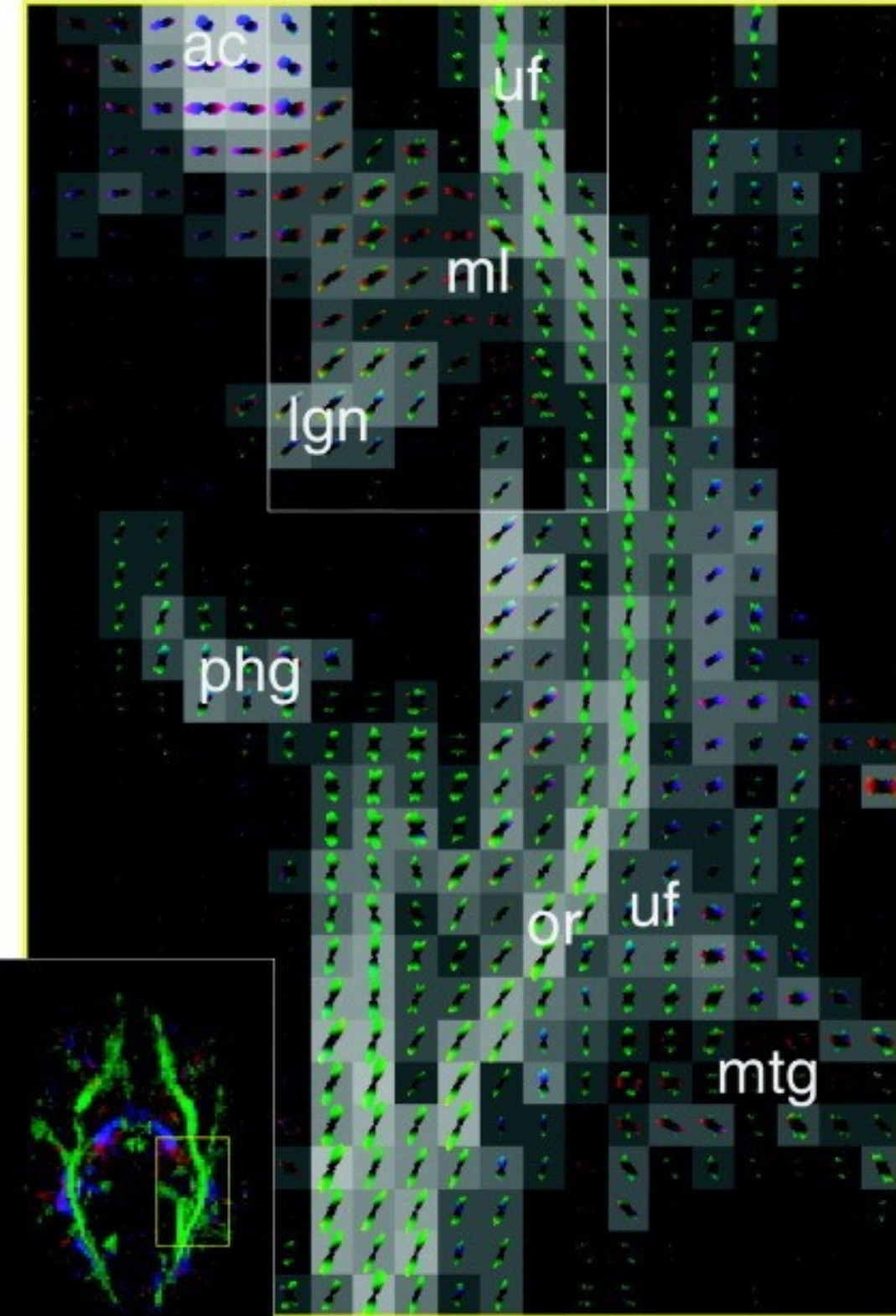
```
csamodel = CsaOdfModel(gtab, 6)
sphere = get_sphere('symmetric724')
sphere = get_sphere('symmetric724')

pmd = peaks_from_model(model=csamodel,
    data=data,
    sphere=sphere,
    relative_peak_threshold=.5,
    min_separation_angle=25,
    mask=mask,
    return_odf=False)
```



For further reading:

J. Cohen-Adad, et al. (2008). "Detection of multiple pathways in the spinal cord using q-ball imaging". NeuroImage, 42, 739-74
D. Tuch (2004) Q-ball imaging, MRM 52.



Perform Tractography

```
#Deterministic tractography
```

```
eu = EuDX(a=fa, ind=pmd.peak_indices[..., 0],  
seeds=2000000, odf_vertices=sphere.vertices, a_low=0.01)
```

```
affine = eu.affine
```

```
csd_streamlines= list(eu)
```

```
#Remove tracts shorter than 30mm
```

```
from dipy.tracking.utils import length
```

```
csd_streamlines=[t for t in csd_streamlines if length(t)>30]
```

Save as a trackvis file

```
#Trackvis
```

```
hdr = nib.trackvis.empty_header()
```

```
hdr['voxel_size'] = img.get_header().get_zooms()[:3]
```

```
hdr['voxel_order'] = 'LAS'
```

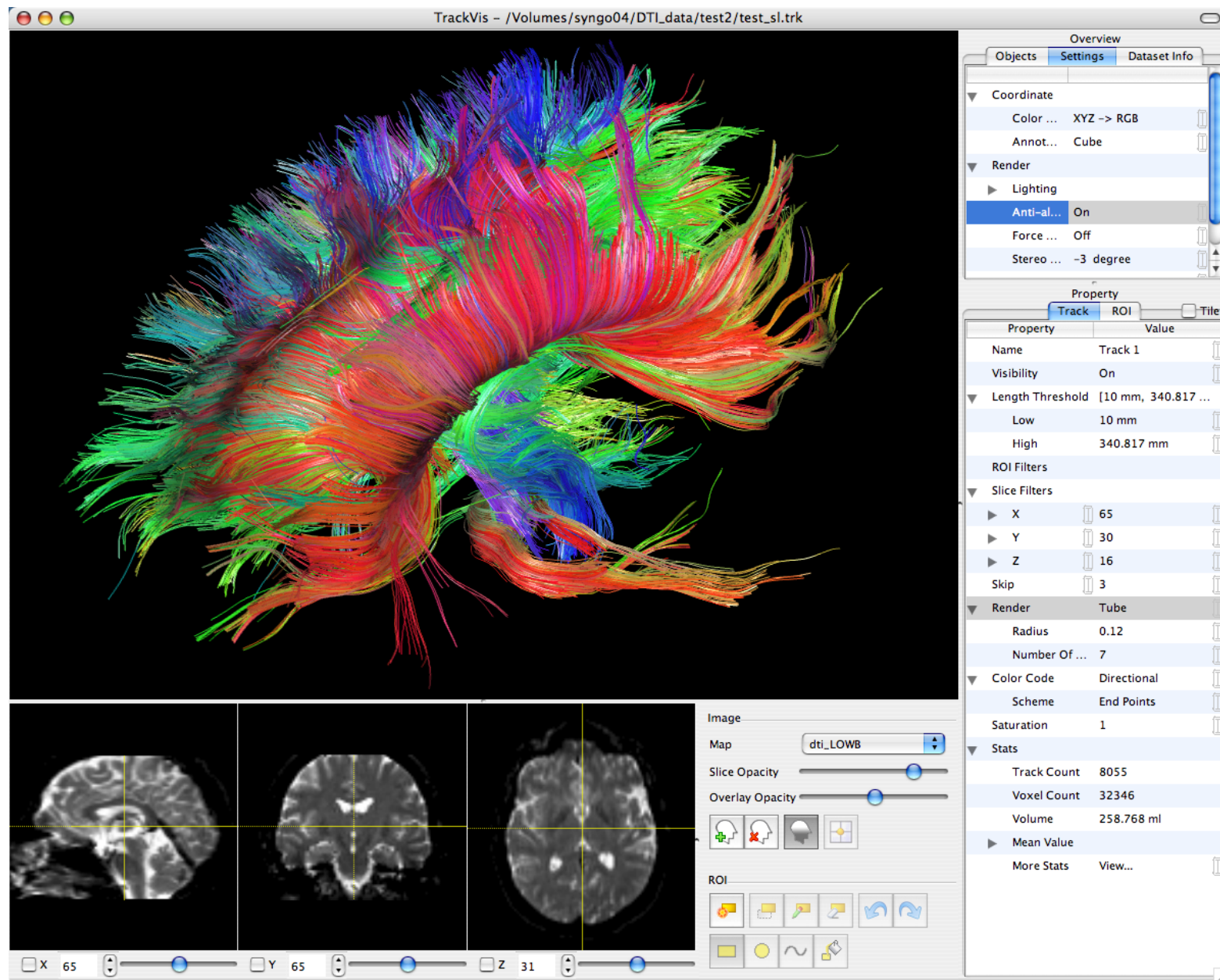
```
hdr['dim'] = fa.shape
```

```
tensor_streamlines_trk = ((sl, None, None) for sl in  
csd_streamlines)
```

```
ten_sl_fname = 'tensor_streamlines.trk'
```

```
nib.trackvis.write(ten_sl_fname, tensor_streamlines_trk, hdr,  
points_space='voxel')
```


Now open Trackvis



Thank you for your attention

