Representation of diffusion MRI data: color coding, glyphs, tractography and beyond

In the previous chapters we have focused on voxel-wise quantitative information extracted from diffusion volumes, but we have more or less neglected orientation and tract-based information (except for the color-coded maps in DTI). In this chapter we will introduce the advanced graphics capabilities of `dmrimatlab`, both for DTI and the non-Gaussian representations presented in the previous chapter. Our `dmrimatlab` bases all these representations in the same function `plotdmri3d`:

```
help plotdmri3d
```

As you can see, it has plenty of options that can result somehow overwhelming. However, we will try to illustrate some usual use cases.

TO DO: Clear all variables and figures to start from zero (*Ctrl+ENTER*):

```
clear;
close('all');
```

For this chapter we will use an *ad hoc* data set that, as opposed to those we have used before:

- Comprises all slices of the acquisition, so that its resolution is the same for all axes.
- Contains pre-processed data instead of raw attenuation signals.

```
whos -file test_data4
```

```
  Name           Size                     Bytes  Class      Attributes

  bi           253x1                       2024  double
  dti          110x110x66x6            38332800  double
  gi           253x3                       6072  double
  ijk2xyz        4x4                        128  double
  labs         110x110x66                798600  uint8
  lpar         110x110x66               6388800  double
  lperp        110x110x66               6388800  double
  mask         110x110x66                798600  logical
  shodf        110x110x66x28           178886400  double
```

```
load test_data4
```

Note we have a variable `dti` with the tensor volume computed for the subject. Besides, `lpar`, `lperp` are the micro-structure parameters of MiSFIT convolution kernel, and `shodf` are the SH coefficients of MiSFIT's ODF. The variable `ijk2xyz` is used to relate the orientations of the raster scan of the voxels with the actual anatomical orientations of the subject (necessary to properly interpret voxel-wise orientations). Besides, we have a variable `labs` which contains manual segmentations of certain white matter regions.

1- Represent image slices in 3-D

The function `plotdmri3d` is meant to be acumulative, i.e., several elements will be added to the same figure with each call. Let's start creating a blank figure to draw in (*Ctrl+ENTER*):

```
close(figure(1));
hf1 = figure(1);
set(hf1,'Position',[1,1,800,800]);
% Get the current axes of the newly created figure:
ha = axes(hf1);
```

A "slice" is a 2-D from of a 3-D volume. One limitation of `plotdmri3d` is that it can only represent slices directly taken from the raster scan arrangement of the volume (no oblique slices are allowed). We can define an axial slice by fixing a constant value of the third index:

```
[szI,szJ,szK] = size(mask); % The field of view
i1 = [1,szI]; % All the FOV for the 1st index
j1 = [1,szJ]; % All the FOV for the 2nd index
k1 = [round(szK/2),round(szK/2)]; % Just the central slice
```

A sagittal slice means the first index is kept constant:

```
i2 = [round(szI/2),round(szI/2)];
j2 = [1,szJ];
k2 = [1,szK];
```

A coronal slice means the seond index is kept constant:

```
i3 = [1,szI];
j3 = [round(szJ/2),round(szJ/2)];
k3 = [1,szK];
```

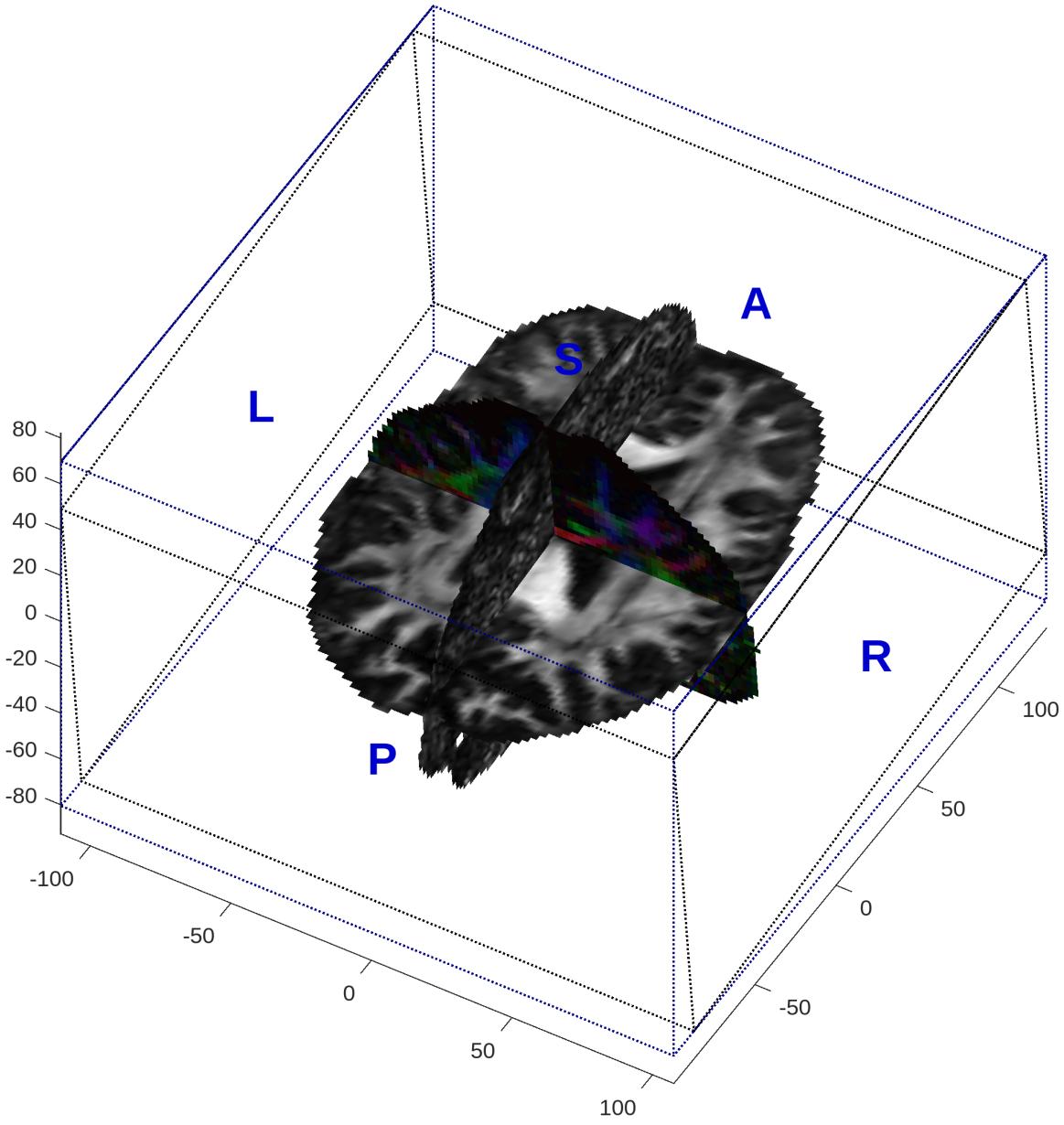
As you can see, slices are defined as four "window corners" plus a singleton dimension. The function `plotdmri3d` is primarily meant to represent orientation information, so that its basic input is a SH volume. But, what if I am only interested in DTI, so I don't have any SH volume? Well, good news is that you can easily translate a DTI volume to a SH-compatible volume just like:

```
dtiSH = hot2sh( dti, 'mask', mask );
```

And now we are ready to plot the slices:

```
% The sagittal slice. At the very least, we need to pass the SH volume and
% the variables describing the slice. We will use the reference dtiSH
% volume also as the image to plot the background. For the axial slice we
% will plot the FA
plotdmri3d( dtiSH, i1, j1, k1, ...
    'mask', mask, ... % This makes non-masked voxels transparent
    'origin', ijk2xyz(1:3,4), ... % The anatomical (rea-world units) origin
    'direction', ijk2xyz(1:3,1:3), ... % The real-worl orientation
    'space', 'RAS', ... % Anatomical space
    'ha', ha, ... % Don't forget to tell the function where to plot!!!
    'glyphs', false, ... % Dont's show voxel-wise orientation glyphs
    'bgimage', 'fa', ... % This slice will be an FA map
    'bgalpha', 0.1 ... % Slice transparency
```

```
    );
% For the sagittal slice we will represent a volume "manually computed"
% outside the function, for example Westin's "planar coefficient":
[~,~,~,l1,l2,l3] = dti2spectrum( dti, 'mask', mask );
cp = spectrum2scalar( l1, l2, l3, 'scalar', 'cp', 'mask', mask );
fa = spectrum2scalar( l1, l2, l3, 'scalar', 'fa', 'mask', mask ); % To be
used later on
plotdmri3d( dtiSH, i2, j2, k2, ...
    'mask', mask, ... % This makes non-masked voxels transparent
    'origin', ijk2xyz(1:3,4), ... % The anatomical (rea-world units) origin
    'direction', ijk2xyz(1:3,1:3), ... % The real-worl orientation
    'space', 'RAS', ... % Anatomical space
    'ha', ha, ... % Don't forget to tell the function where to plot!!!
    'glyphs', false, ... % Dont's show voxel-wise orientation glyphs
    'bgimage', cp, ... % This slice will be an CP map
    'bgalpha', 0.1 ... % Slice transparency
    );
% And for the coronal slice we will represent a color-code
plotdmri3d( dtiSH, i3, j3, k3, ...
    'mask', mask, ... % This makes non-masked voxels transparent
    'origin', ijk2xyz(1:3,4), ... % The anatomical (real-world units) origin
    'direction', ijk2xyz(1:3,1:3), ... % The real-world orientation
    'space', 'RAS', ... % Anatomical space
    'ha', ha, ... % Don't forget to tell the function where to plot!!!
    'glyphs', false, ... % Dont's show voxel-wise orientation glyphs
    'bgimage', 'color', ... % This slice will be a color-code map
    'bgalpha', 0.1 ... % Slice transparency
    );
% Scale the axis so that they represent real-world dimensions:
axis('equal');
% Rotate:
view(30,45);
% Allow manual rotation (drag the mouse):
rotate3d('on');
```



QUIZ:

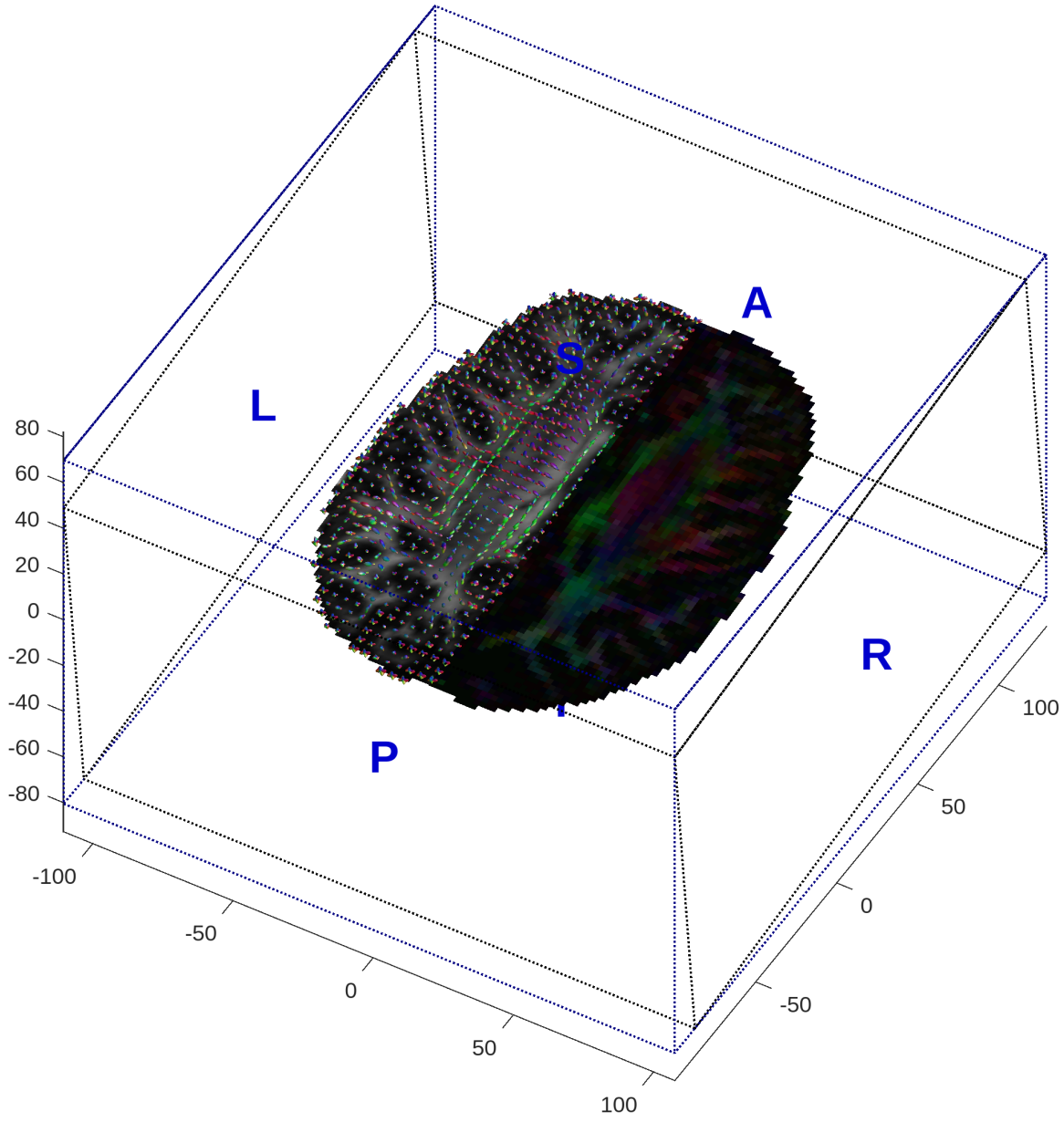
- *What is the size of the matrix ijk2xyz? What does each part of the matrix represent?*
- *You can see two different "bounding boxes" in the picture, one in blue the other in black. Can you guess what each of these boxes represent?*

2- Represent orientation glyphs

In the previous chapter we have mentioned that advanced diffusion representations are able to disentangle complex micro-structural arrangements such as fibers crossing, bending or kissing. However, this capability is by no means evident from the RTxP maps, which are otherwise qualitative similar to any scalar map derived from DTI. Let's demonstrate that MiSFIT is actually able to resolve fiber crossings by representing the variable `shodf` previously loaded:

```
close(figure(2));
hf2 = figure(2);
set(hf2,'Position',[1,1,800,800]);
% Get the current axes of the newly created figure:
ha = axes(hf2);
% We will show just an axial slice (hope you have a decent graphics card).
% Half the slice will show just DTI-bsed color code:
i1 = [1,floor(szI/2)];
j1 = [1,szJ];
k1 = [round(szK/2),round(szK/2)] + 5; % Move several pixels upwards
% The other half will show a background with orientation glyphs overimposed
% to the FA map:
i2 = [floor(szI/2),szI];
j2 = j1;
k2 = k1; % Move several pixels upwards
% The first half is the same as in the previous image:
plotdmri3d( dtiSH, i1, j1, k1, ...
    'mask', mask, ... % This makes non-masked voxels transparent
    'origin', ijk2xyz(1:3,4), ... % The anatomical (rea-world units) origin
    'direction', ijk2xyz(1:3,1:3), ... % The real-worl orientation
    'space', 'RAS', ... % Anatomical space
    'ha', ha, ... % Don't forget to tell the function where to plot!!!
    'glyphs', false, ... % Dont's show voxel-wise orientation glyphs
    'bgimage', 'color', ... % This slice will be a color-code map
    'bgalpha', 0.1 ... % Slice transparency
    );
% The second half needs some more arguments. Besides, note that we are
% using MiSFIT-computed shodf as the reference volume to be able to resolve
% crossing fibers
plotdmri3d( shodf, i2, j2, k2, ...
    'mask', mask, ... % This makes non-masked voxels transparent
    'origin', ijk2xyz(1:3,4), ... % The anatomical (rea-world units) origin
    'direction', ijk2xyz(1:3,1:3), ... % The real-worl orientation
    'space', 'RAS', ... % Anatomical space
    'ha', ha, ... % Don't forget to tell the function where to plot!!!
    'bgimage', 'fa', ... % Used only for the background
    'bgsh', dtiSH, ... % Use the tensor representation for the background
    'bgalpha', 0.1, ... % Background transparency
    ... % THESE ARE THE NEW ARGUMENTS WE NEED:
    'glyphs', true, ... % We DO WANT glyphs!
    'glyphspc', [2;2;2], ... % Plot glyphs every two voxels at each direction
    'glyphsc', 2, ... % Scale applied to all glyphs
    'glyphscvar', 'ga' ... % Scale the glyphs depending on their anisotropy
```

```
    );
% Scale the axis so that they represent real-world dimensions:
axis('equal');
% Rotate:
view(30,45);
% Add some light effects to make things look cooler:
light
lighting('phong');
% Allow manual rotation (drag the mouse):
rotate3d('on');
```



QUIZ:

- *The slice we have chosen contains a great part of the centrum semiovale, where many important fiber bundles are known to cross each other. If you have some expertise on neuroanatomy, you may try to identify some of them.*
- *One important property of Spherical Harmonics is that they draw progressive refinements of the spherical signal with their successive degrees. You can easily "reduce" the representation given by* `shodf` *by*

keeping just its first few coefficients. In particular, try to do the following operations and re-run the code substituting `shodf` with each resulting `shodf_reduced`. What happens for L=2?

```
% This corresponds to SH up to order L=4:
shodf_reduced = shodf(:,:,:,1:15);
% This corresponds to SH up to order L=2:
shodf_reduced = shodf(:,:,:,1:6);
```

QUIZ:

Would you be able to draw a coronal slice with glyphs representing MiSFIT's ODFs?

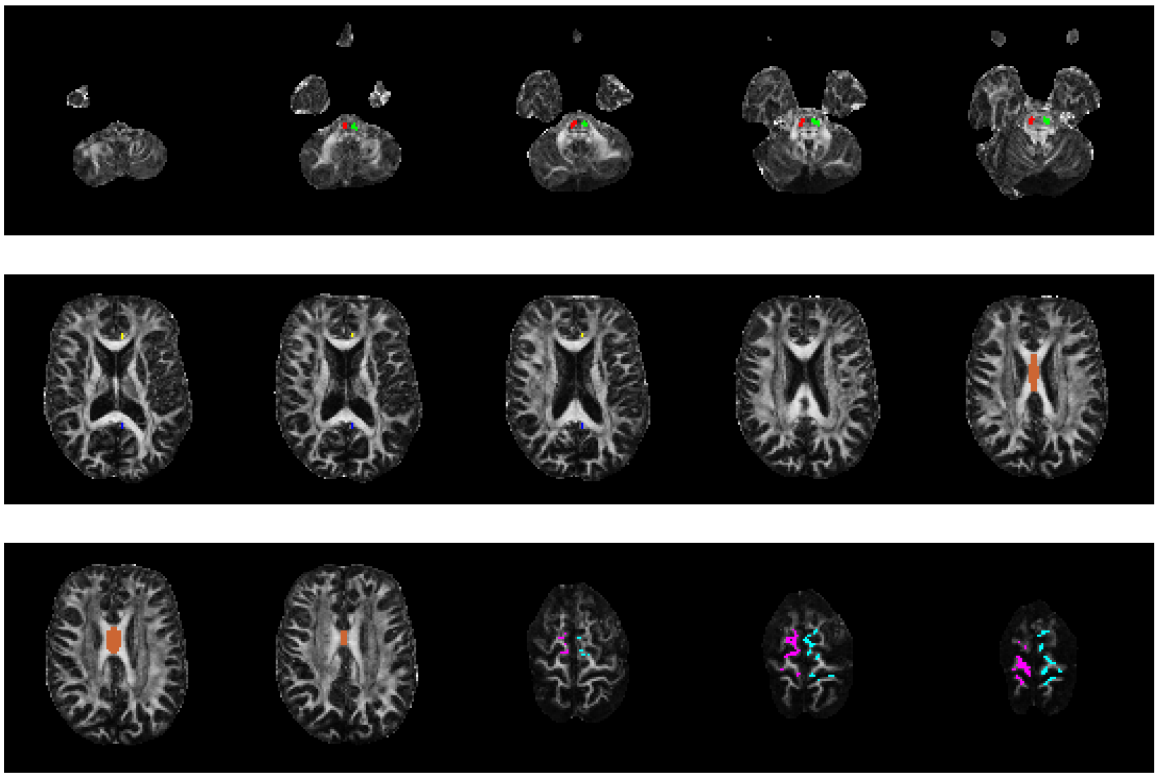
3- Computing and representing tractographies

The highest level front-end for the representation of diffusion MRI data is pobably tractography or fiber-tracking. With this technique, one starts with a set of "seeds" or fiducial locations within the field of view of the image and traces streamlines following the main diffusion directions at each visited location until a stop criterion is met (this explanation is a heavy over-simplification of the literature on fiber-tracking, but it will serve for our purposes). To know what we are talking about, let's check the "labelmap" we have loaded with the test data in this chapter, i.e. the volume `labs`. It is just a volume with the same size as the diffusion data that contains integer values tagging certain regions of interest we have manually delineated within the white matter (*Ctrl+ENTER*):

```
unique(labs)',
```

```
ans = 1×8 uint8 row vector
   0   1   2   3   4   5   6   7
```

```
close(figure(3));
hf3 = figure(3);
set(hf3,'Position',[1,1,900,600]);
plotOverlaidLabelmap(labs,fa);
```



So the regions of interest are placed in (if you cannot see them, try to undock the figure):

- Labels 1 (red) and 2 (green) are placed in the middle cerebellar peduncle, right and left hemispheres.
- Label 3 (blue) is approximately placed over the posterior cingulum.
- Label 4 (yellow) is approximately placed over the anterior cingulum.
- Label 5 (orange) is placed right in the middle of the corpus callosum.
- Labels 6 (magenta) and 7 (cyan) are placed in the uppermost part of the corticospianl tract.

These are just labelmaps tagged in the voxel space. To feed the tractography algorithm, we need to convert them to actual real-world points with a function that takes a labelmap as an input and outputs acual seeds, hence its name is `labelmap2seeds`:

```
% The only mandatory input is the labelmap itself, but it is very likely
% that we will have to use some of its options:
seeds1 = labelmap2seeds( labs, ...
    'label', 1, ... % Select the region we are interested in (MCP, right)
    'nSeeds', 50, ... % How many seeds do we need?
    ... % NOTE THIS PARAMETER IS CRUCIAL!!!:
    'ijk2xyz', ijk2xyz ... % What is the actual spatial orientarion of the
voxels?
    );
```

```
% Repeat for the other hemisphere:
seeds2 = labelmap2seeds( labs, ...
    'label', 2, ...
    'nSeeds', 50, ...
    'ijk2xyz', ijk2xyz );
```

Now we need a function that takes diffusion MRI data and outputs a tractography. Since our tractography will be by now DTI-based, the function to use is called `dti2tractography`:

```
% The minimum required inputs are the DTI volume and
% the seeding poits, but it is likely that you will
% need to fix some of the optional parameters:
paths1 = dti2tractography( dti, seeds1, ...
    'mask', mask, ...
    ... % THIS PARAMETER IS AGAIN CRUCIAL!!!:
    'ijk2xyz', ijk2xyz );
% Repeat for the other hemisphere:
paths2 = dti2tractography( dti, seeds2, ...
    'mask', mask, ...
    ... % THIS PARAMETER IS AGAIN CRUCIAL!!!:
    'ijk2xyz', ijk2xyz );
```

You can check that the variables output by `dti2tractography` are so-called "cell arrays", i.e abstract vectors each of whose positions can be more or less anyhthing:

```
paths1,
```

paths1 = 100×1 cell

	1
1	3×18 double
2	3×112 double
3	3×25 double
4	3×120 double
5	3×40 double
6	3×41 double
7	3×8 double
8	3×45 double
9	3×19 double
10	3×71 double
11	3×110 double
12	3×68 double
13	3×48 double
14	3×37 double

	1
15	3×22 double
16	3×124 double
17	3×111 double
18	3×15 double
19	3×26 double
20	3×99 double
21	3×6 double
22	3×64 double
23	3×105 double
24	3×41 double
25	3×135 double
26	3×15 double
27	3×71 double
28	3×15 double
29	3×26 double
30	3×112 double
31	3×36 double
32	3×61 double
33	3×39 double
34	3×27 double
35	3×30 double
36	3×118 double
37	3×12 double
38	3×7 double
39	3×72 double
40	[2.6781,2.2956,1.8897;17.6289,17.6222,17.6270;-51.9106,-52.2326,-52.5245]
41	3×52 double
42	3×16 double
43	3×26 double
44	3×62 double

	1
45	3×45 double
46	3×16 double
47	3×85 double
48	3×45 double
49	3×77 double
50	3×63 double
51	[ ]
52	3×67 double
53	3×20 double
54	3×74 double
55	3×25 double
56	3×129 double
57	3×47 double
58	3×19 double
59	3×13 double
60	3×61 double
61	3×42 double
62	3×6 double
63	3×12 double
64	3×130 double
65	3×7 double
66	3×39 double
67	3×37 double
68	3×110 double
69	3×5 double
70	3×55 double
71	3×113 double
72	3×31 double
73	3×9 double
74	3×6 double
75	3×136 double
76	3×9 double
77	3×49 double

	1
78	3×33 double
79	3×60 double
80	3×13 double
81	3×7 double
82	3×12 double
83	3×25 double
84	3×119 double
85	3×123 double
86	3×4 double
87	3×34 double
88	3×61 double
89	3×81 double
90	3×12 double
91	3×12 double
92	3×219 double
93	3×112 double
94	3×36 double
95	3×9 double
96	3×38 double
97	3×31 double
98	3×41 double
99	3×15 double
100	3×93 double

In our case, `paths1` has 100 entries because we have used 100 seeding points to generate it (we passed 'nSeeds', 100 when we called `labelmap2seeds`). Each of these entries is a $3 \times N$ matrix, where the algorithm stacks the succesive `x`, `y` and `z` coordinates of the traced fibers (in real world coordinates). Hence, it suffices to use Matlab's plot3 command to draw these streamlines. But first, to give a little bit of context, we will show first axial, sagittal and coronal slices of the FA:

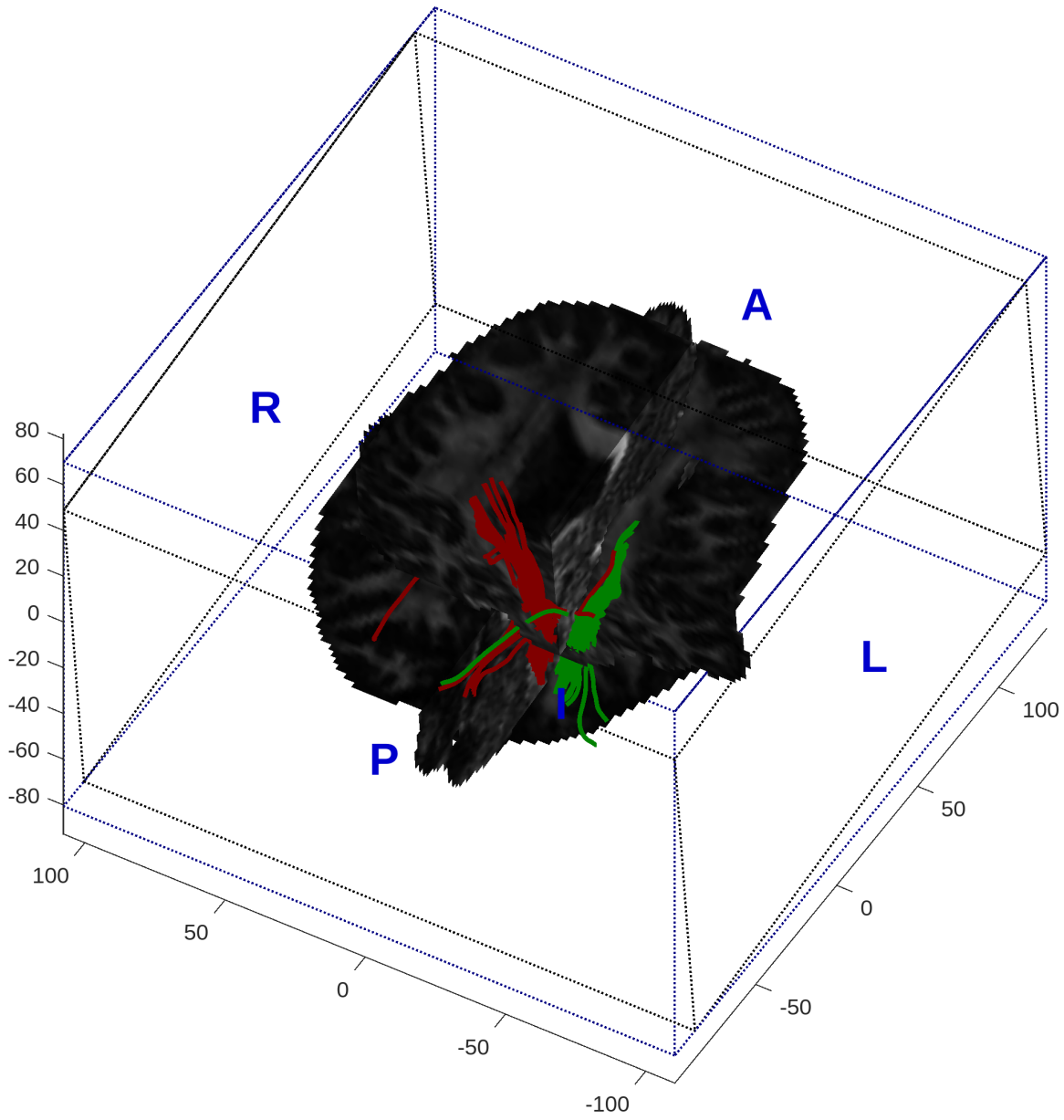
```
% Axial:
i1 = [1,szI];
j1 = [1,szJ];
k1 = [round(szK/2),round(szK/2)];
% Sagittal:
i2 = [round(szI/2),round(szI/2)];
j2 = [1,szJ];
k2 = [1,szK];
```

```
% Coronal:
i3 = [1,szI];
j3 = [round(szJ/2),round(szJ/2)];
k3 = [1,szK];
% Make the plots:
close(figure(4));
hf4 = figure(4);
set(hf4,'Position',[1,1,800,800]);
% Get the current axes of the newly created figure:
ha = axes(hf4);
for ax=1:3
    ix = eval(sprintf('i%d',ax));
    jx = eval(sprintf('j%d',ax));
    kx = eval(sprintf('k%d',ax));
    plotdmri3d( dtiSH, ix, jx, kx, ...
        'mask', mask, ...
        'origin', ijk2xyz(1:3,4), ...
        'direction', ijk2xyz(1:3,1:3), ...
        'space', 'RAS', ...
        'ha', ha, ...
        'glyphs', false, ...
        'bgimage', 'fa', ...
        'bgalpha', 0.1 ...
        );
end
```

And now we are ready to actually plot the streamlines:

```
for n=1:length(paths1)
    streamline = paths1{n}; % Retriebe an entry of the cell array
    if(~isempty(streamline))
        % Empty streamlines can appear if something goes wrong
        plot3( ha, ... % Plot in the same previous axes
            streamline(1,:), ... % x
            streamline(2,:), ... % y
            streamline(3,:), ... % z
            'Color', [.5,.0,.0], ...
            'LineWidth', 2 );
    end
end
% Repeat for the other hemisphere:
for n=1:length(paths2)
    streamline = paths2{n}; % Retriebe an entry of the cell array
    if(~isempty(streamline))
        % Empty streamlines can appear if something goes wrong
        plot3( ha, ... % Plot in the same previous axes
            streamline(1,:), ... % x
            streamline(2,:), ... % y
            streamline(3,:), ... % z
            'Color', [.0,.5,.0], ...
```

```
            'LineWidth', 2 );
    end
end
% Scale the axis so that they represent real-world dimensions:
axis('equal');
% Rotate:
view(150,-45);
% Add some light effects to make things look cooler:
light
lighting('phong');
% Allow manual rotation (drag the mouse):
rotate3d('on');
drawnow;
title('Drag the mouse!');
```



QUIZ:

- *Can you try to delineate the corpus callosum? (use label 5 of the labelmap) And the cingulum? (labels 3 and 4).*
- *Can you make the algorithm trace the streamlines further than it currently does?* ***HINT:*** *check the '`threshold`' argument of the `dti2tractography` function.*

```
help dti2tractography
```

Go back to index page

Previous

Auxiliar functions

Don't worry too much about this section. It is just a shortcut to represent the labelmap

```
function plotOverlaidLabelmap(labels,fa)
%slices = 7:4:54;
slices = [ ...
    3, 5, 7, 9, 11, ...
    30, 31, 32, 34, 35, ...
    36, 37, 51, 53, 55 ];
colors = [1,0,0;0,1,0;0,0,1;1,1,0;1,0,1;0,1,1;.8,.4,.2];
tiledlayout( 3, 5, 'TileSpacing', 'none' );
RGB = zeros( size(fa,2), size(fa,1), 3 );
for k=1:length(slices)
    nexttile;
    R  = fa(:,:,slices(k));
    G  = fa(:,:,slices(k));
    B  = fa(:,:,slices(k));
    LB = labels(:,:,slices(k));
    for l=1:7 % For each label
        R(LB==l) = colors(l,1);
        G(LB==l) = colors(l,2);
        B(LB==l) = colors(l,3);
    end
    RGB(end:-1:1,:,1) = R';
    RGB(end:-1:1,:,2) = G';
    RGB(end:-1:1,:,3) = B';
    imshow(RGB);
end
drawnow;
end
```