Cartesian geometric representation with isometric dimensions

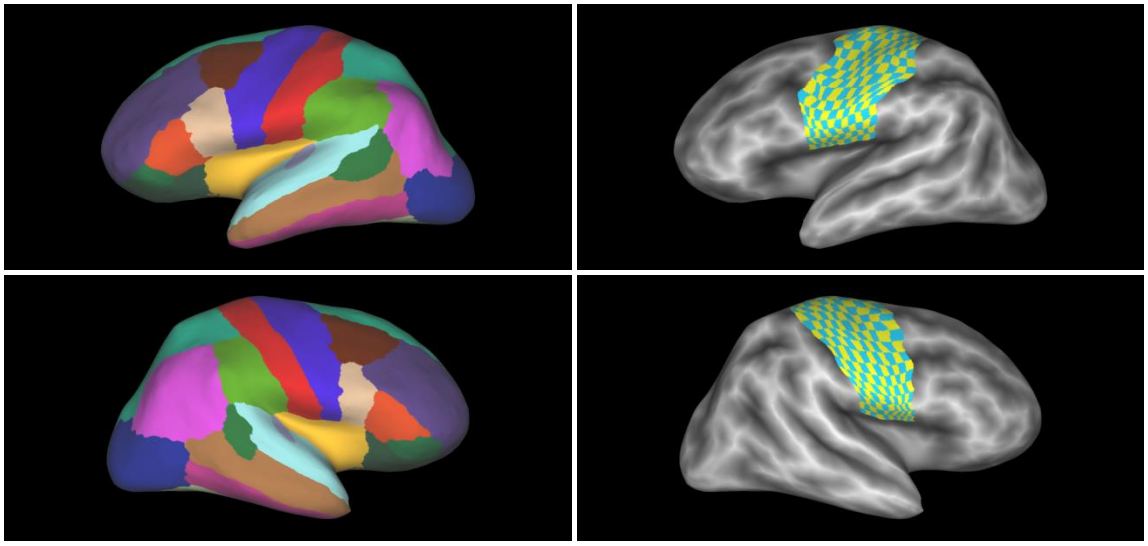
Manual & Tutorial

Mathijs Raemaekers

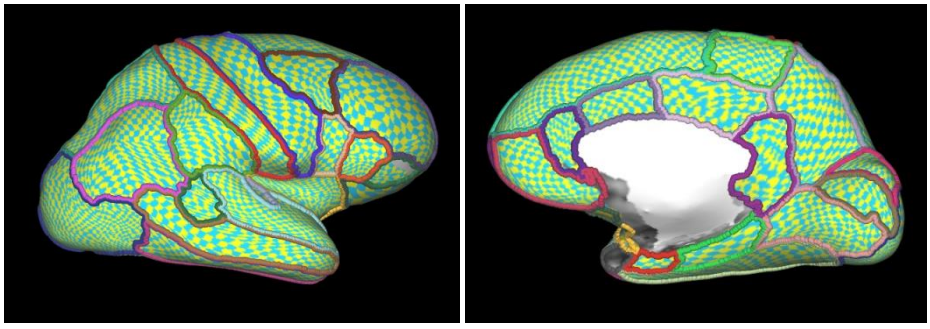
Index:

1. Introduction:

The FreeSurfer recon-all pipeline includes automatic assignment of neuroanatomical labels to the cortex. The Cgrid toolbox is a graphical user interface that allows users to fully automatically force Regions of Interest (ROIs) as defined by one of the FreeSurfer labeling schemes onto Cartesian grids, by assigning x and y coordinates to every vertex within a particular FreeSurfer ROI, or combination of ROIs. Here below you can see an example of a Cgrid, covering the precentral and postcentral ROI's from the FreeSurfer 'aparc' labeling scheme (shown in red and blue in the left 2 panels).



In essence, imposing the Cgrid provides the possibility to build rectangular representations of areas of the brain, which can facilitate processing and interpretation of MRI/fMRI/DTI data in numerous ways. In principle, Cgrids can be generated for all areas for which four borders can be defined, which includes virtually the entire surface area of the brain:



The toolbox can use the information from Cgrids to transform volumetric MRI/fMRI or surface based data into 2D matrices, which are stored in Cgrid-nifti-files, providing direct compatibility with many other software packages. This output is also immediately suitable for second-level analysis. In addition, the Cgrid information can be stored in 3D volumetric nifti format, providing the x and y coordinates for every voxel in the area of the brain to which the Cgrid is applied.

This manuscript includes a stepwise description on how to generate and apply Cgrids, using the sensorimotor cortex as example (including the precentral and postcentral ROI of the 'aparc' annotation), and explaining the full functionality of the toolbox in the process.

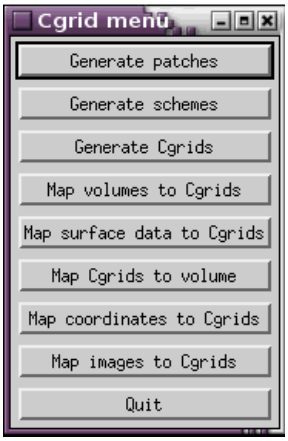
2. Downloading, installing, and running the Cgrid-toolbox

The Cgrid toolbox functions as an add-on to the FreeSurfer software package, meaning that it requires FreeSurfer to be fully installed and working on your system. The toolbox uses several FreeSurfer command line tools and uses the output of the FreesSurfer recon-all pipeline as input. Details on how to install FreeSurfer and can be found on:
https://surfer.nmr.mgh.harvard.edu/fswiki/DownloadAndInstall
The Cgrid-toolbox is written in the Interactive Data Language (IDL) and compiled so that it can run with the IDL virtual machine, so you don't need an IDL license. You do however need to have IDL (which includes the IDL virtual machine) installed on your system to be able to run it. You can download IDL for free from:
https://www.harrisgeospatial.com/
However, you do need to register, and your registration needs to be verified before you are able to download, which might take up to 24 hours. Instructions on how to install IDL you can find on:
https://www.harrisgeospatial.com/Support/SelfHelpTools/HelpArticles/HelpArticles-Detail/TabId/2718/ArtMID/10220/ArticleID/15066/Install-and-License-IDL-86.aspx
You can disregard the portion covering the license server, as you will be only using the virtual machine.
Once FreeSurfer and IDL are installed on your system, you can download 'cgrid.sav' file from the Cgrid repository at Github (https://github.com/mathijsraemaekers/Cgrid-toolbox), which contains toolbox, and store it somewhere on your system. You can start the toolbox, e.g. when *'cgrid.sav'* if stored in *'/home/user/'*, by typing from the console:
'idl -vm=/home/user/cgrid.sav'
To make starting the toolbox a bit more convenient, you can also add an alias to your .bashr e.g.:
alias cgrid='idl -vm=/home/user/cgrid.sav'
after which you can start the toolbox by just typing 'cgrid' from the console. After executing Cgrid from the console, a startup screen of the IDL virtual machine is shown, that you need to click on before the toolbox is started. Cgrids are generated based on the output of the FreeSurfer 'recon-all' pipeline, so before Cgrids can be generated for a particular subject, you need to run the full pipeline. Instructions for running the pipeline can be found on:
https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all
For now, Cgrid will only work properly on systems using little-endian for storage. We hope to expand compatibility with big-endian storage in the near future.

If you want to briefly check out the toolbox without going through the hassle described above, you can also download a pre-installed virtual machine that already includes FreeSurfer, the IDL virtual machine, and the Cgrid toolbox.
You can run the virtual machine using vitualbox, which you can download from https://www.virtualbox.org/.

3. Overview

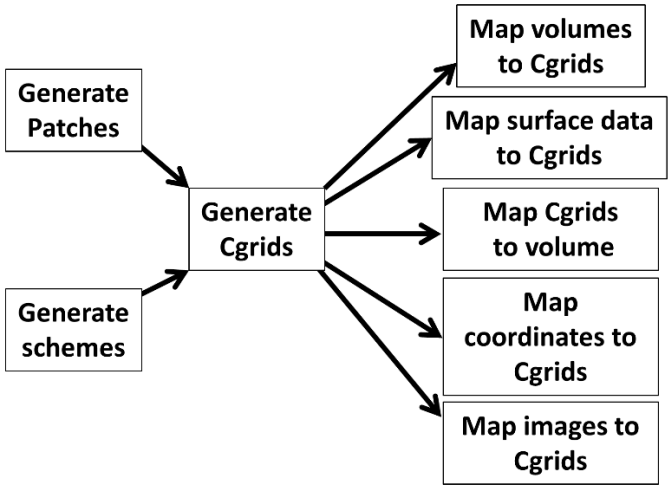
After you started the toolbox, you will see the Cgrid-menu:



The menu items represent the possible processing steps for generating and applying Cgrids and these include:

1. *'Generate patches':* extracting and flattening ROIs or combinations of ROIs and store as patch-file
2. *'Generate schemes':* creating and storing a set of instructions on how to transform the patches into a Cartesian grid
3. *'Generate Cgrids':* Applying the schemes to the patches, thereby assigning x and y coordinates to the vertices of the ROIs included in the Cgrid.
4. *Map volumes to Cgrids:* Mapping 3D or 4D volumetric MRI/fMRI data into Cartesian grids using the generated Cgrids, and storing it as Cgrid-nifti-files
5. *Map surface data to Cgrids:* Mapping FreeSurfer's surface data into Cartesian grids using the generated Cgrids, and storing it as Cgrid-nifti-files
6. *Map Cgrids to volume:* Putting the Cgrid x and y coordinates back in 3D space, and storing it in a volumetric format (e.g. nifti)
7. *Map coordinates to Cgrids:* Generate an ascii table with Cgrid coordinates for supplied 3D coordinates, which is useful for mapping e.g. electrodes positions to a Cgrid.
8. *Map images to Cgrids:* Map tif/jpg/png images to Cgrids on the cortical surface, storing it as a FreeSurfer ‘w’-file

The diagram below shows the general workflow of the toolbox.

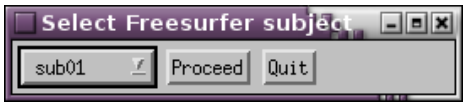


The different processing steps will be described in detail in the following chapters. During the processing steps, the Cgrid toolbox uses and generates a number of different file types including:

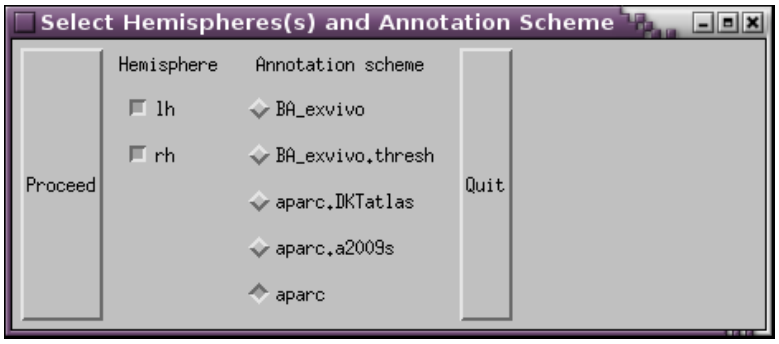
- P*atch-file* (lh/rh.*_flat.3d) contains a limited portion of a brain's surface reconstruction. The patch-file format is compatible with FreeSurfer. The patch-files generated by the Cgrid-toobox are flattened and are stored in the 'surf' folder of a subject's FreeSurfer recon-all output.
- A *scheme-file* (*.scheme) contains the instructions on how to impose the Cartesian Grid on a patch-file. Scheme-files are ascii files that can be generated with the Cgrid-toolbox, but can also be easily altered using a generic text editor. By default scheme-files are stored the $CGRID_SCHEMEDIR, as defined in your .bashrc.
- A *Cgrid-file* (lh/rh_*.cgrid) file is a three column ascii-file containing a vertex number and Cgrid x&y coordinates for every vertex within the surface area covered by the Cgrid. Cgrid-files are stored in the 'surf' folder of a subject's FreeSurfer recon-all output.
- A *Cgrid-nifti-file* (*.nii) is in principle a normal nifti-file that can be used with any nifti-compatible software package. Its x and y dimensions are the dimensions of the Cgrid, and the z-dimension (length 2) represents the left (z-coordinate=1) and the right (z-coordinate=2) hemisphere. Multiple volumes can be represented by the 4th dimension, just as with the regular nifti format.
- V*olumetric-cgrid-file* (e.g. *.nii, *.mgz) describe the Cgrid in volumetric space, including separate volumes for the Cgrid x and y-coordinates, and for the left and right hemisphere.
- *FreeSurfer weigth-file* (*.w) is an overlay file that can be displayed using e.g. FreeSufer's tksurfer, containing an tif/png/jpg file that is mapped to the surface of the brain.

4. Generate Patches

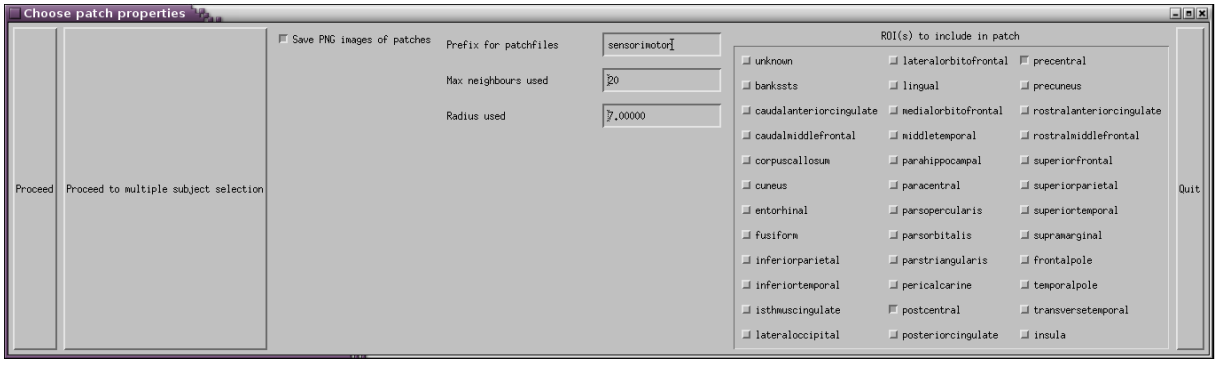
The first step in generating Cgrids is extracting and flattening portions from the cortical surface. The portions can exist of one or multiple ROIs as defined by one of the FreeSurfer parcellations (e.g. 'aparc', 'aparc.a2009s') that are generated as part of the FreeSurfer recon-all pipeline. If you want to generate patches for existing schemes, you can open a *'*.scheme'* file in a generic text editor, and look under 'Annotation scheme' and 'ROIs to include' to find out the requirements for the to be generated patch. You start the process of making patches by clicking on *'Generate patches'* in the Cgrid-menu. While flat patches can be generated for multiple subjects in a single batch, the process starts with the selection of a single 'example'-subject for which the available annotation schemes and ROIs therein are established:



Although under normal circumstances the same annotation schemes are present for all subjects, things can be different when subjects were run with different FreeSurfer versions, when subjects have abnormal anatomies, or when annotations schemes have been generated manually. The toolbox automatically detects all available subjects in the FreeSurfer SUBJECTS_DIR (as defined in your .bashrc). In this case we choose *'sub01'*, and press *'Proceed'*. In the following menu you select according to which from the available annotation schemes for your example subject you want to extract patches, and if you want to do this for the left, right, or both hemispheres:



Under normal circumstances, you will select both hemispheres (select one hemisphere when an appropriate surface reconstruction only exists for one hemisphere, or when you are impatient). In this case we use the select the *'aparc'* annotation scheme (containing the 'precentral' and 'postcentral' ROI) and include both hemispheres. After that, press *'Proceed'*. In the menu that follows you can name the patch, select the ROIs to include, and specify parameters for flattening:



To find out what each ROI comprises, you can do so by visualizing parcellations using FreeSurfer's 'tksurfer'. In addition, images of a cortical labeling in an example subject using the 'aparc' and 'aparc.DKT' annotation scheme are included in the addendum of this manual.

When selecting ROIs, keep the following in mind:

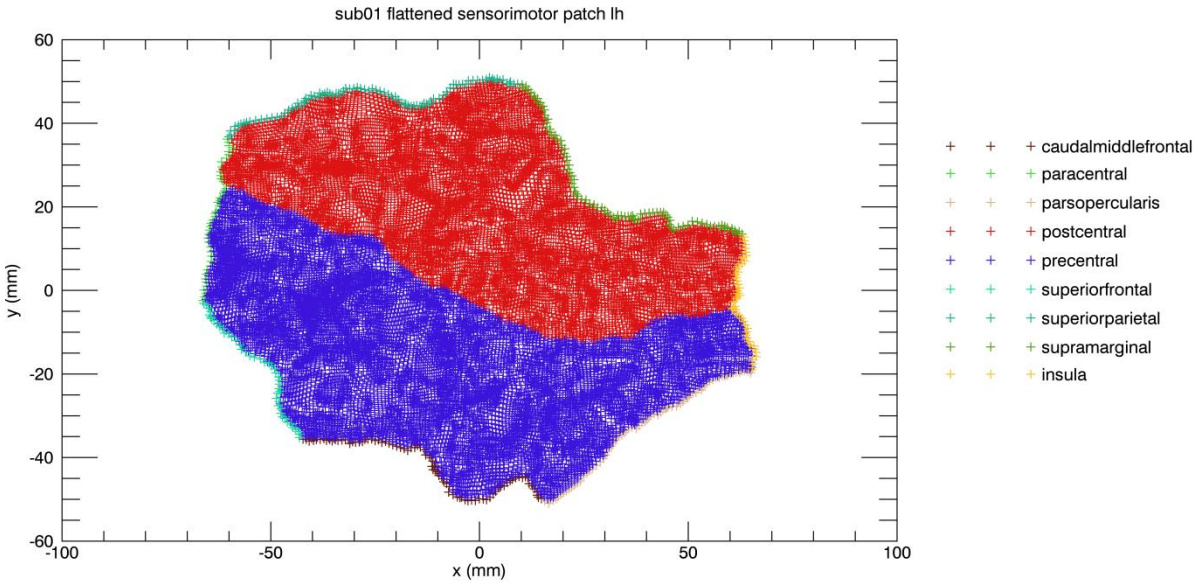
- The patch needs to include the entire surface area (all vertices) for which you want to generate the Cgrid
- Select ROIs that are adjacent in the brain
- The larger the patch, the more processing time

E.g., in this case we want to generate a Cgrid of the sensorimotor cortex, and thus include only *'precentral'* and *'postcentral'*. Including additional adjacent areas will only increase processing time.

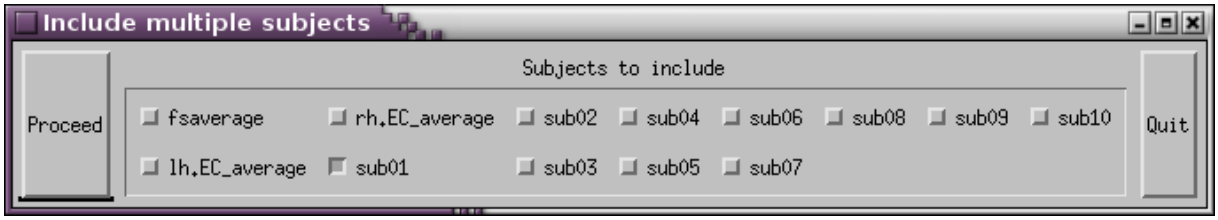
Furthermore, you need to specify the '*prefix'* for the patch-files. This will also be the prefix of any of the following files that you generate, so choose a sensible name. The flat patches will be stored in the 'surf' folder of the FreeSurfer recon-all output of the subject as rh/lh.*prefix*_flat.3d. These files are in the Freesurfer 'patch' format, and can be visualized using Freesurfer's 'tksurfer'.

With *'Max neighbours used'* and *'Radius used'* you can set the distances parameters of FreeSurfer's *'mris_flatten'* procedure. The higher the setting of *'Max Neighbours'*, the less the metric distortion in the resulting patch, but the longer the processing time. The 'Radius used' is typically set at 7 mm. For more details on the mris_flatten procudere see
https://surfer.nmr.mgh.harvard.edu/fswiki/mris_flatten

You can choose to store a graph image of the patch as 'png'-file, which will be stored in the 'surf' folder of the subject. This image is for visual inspection only. When visually inspecting the images after flattening, you may notice that patches include the selected patches, and the first order neighbors. This is necessary for later processing steps. The stored graphs can also help with defining borders of the Cgrid when making schemes (next chapter). You can see an example of a graph of a patch including 'precentral' and 'postcentral' of the 'aparc' annotation here below:



After the patch specification is complete, you can choose *'Proceed'* or *'Proceed to multiple subject selection'*. When you choose *'Proceed'*, the specified patches are generated for the single example-subject that you chose in the beginning. When you choose *'Proceed with multiple subject selection'*, you continue to a menu where you can select from all subjects in the FreeSurfer SUBJECTS_DIR for which you want to extract the defined patch:



The program will then run the patch extraction and flattening of all these subjects sequentially. Note that the chosen parcellation scheme and must be available for all the subjects that you select.

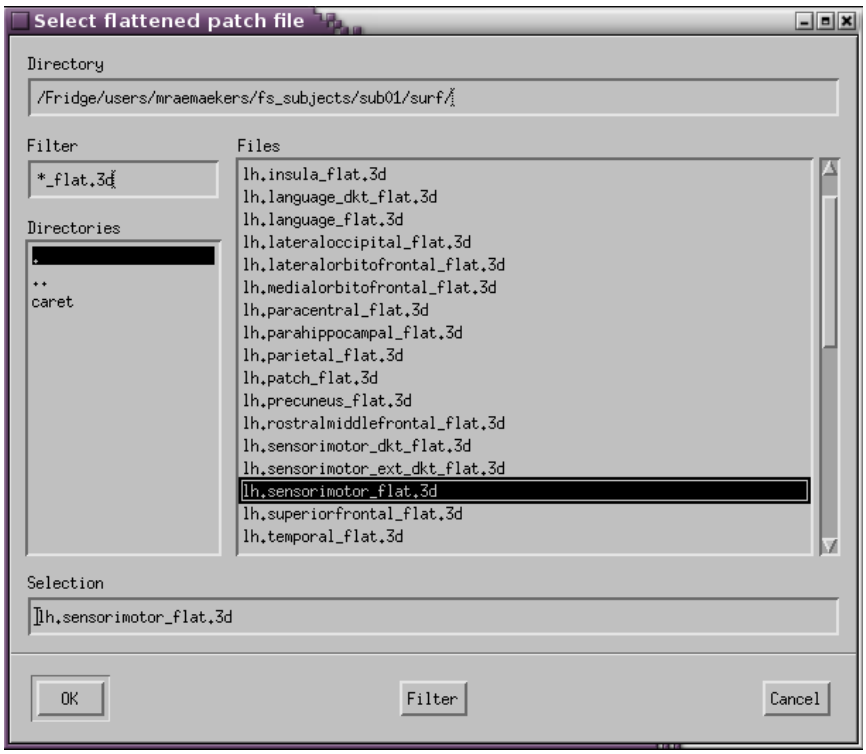
The generation of the flat patches can take roughly anywhere between 5 and 60 minutes per patch, depending on the size of the patch and your available hardware. The process runs on a single processing core, so if you have multiple cores available and are in a hurry, you can run several subjects/batches simultaneously.

5. **Generating schemes**

A Cgrid-scheme contains instructions on how the Cartesian grid is imposed on the generated flattened patches. Schemes are stored as *.scheme, which are basically ascii files and are by default stored in the $CGRID_SCHEMEDIR as defined in your .bashrc. Generating schemes is probably the most difficult part when generating Cgrids. Luckily, a scheme needs to be created only once, and can subsequently be applied to multiple subjects. In addition, several existing schemes are already available in the cgrid_schemes folder in the Github repository. If you want to share a scheme you developed, you can mail it to m.raemaekers-2@umcutrecht.nl, and it will be put in the repository.

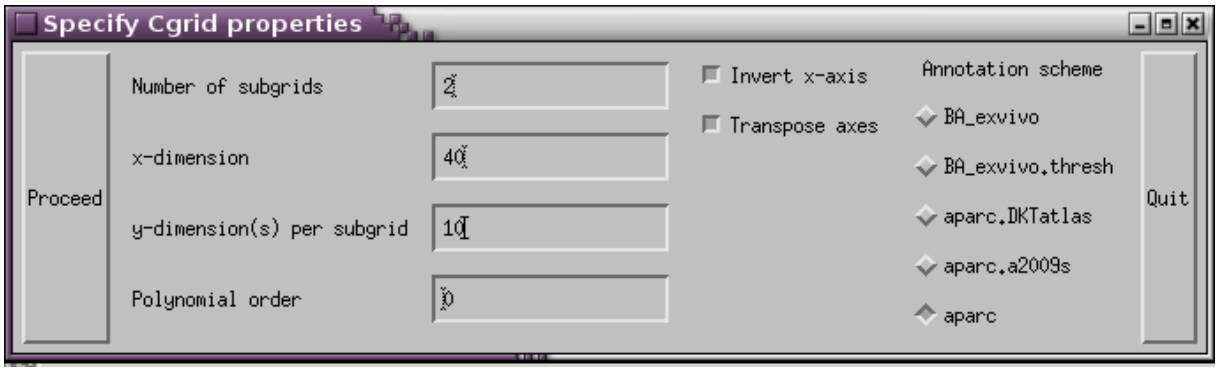
A Cgrid is composed of one or multiple subgrids that are independently generated, and subsequently concatenated. The central part of generating schemes is defining borders, where for each subgrid that is to be included in the Cgrid, an upper, lower, left and right border needs to be established. Each border is composed of one or a combination of borders between adjacent FreeSurfer ROI-pairs. You start the process of making patches by clicking on *'Generate schemes'* in the Cgrid-menu.

The first step when generating a scheme is selecting a representative flattened patch that contains the ROIs on which the GRID is to be imposed. These flattened patches are stored in the subject's 'surf' folder during 'Generate Patches':



Select a patch-file, from either the left (lh.*prefix*_flat.3d) or right (rh.*prefix*_flat.3d) hemisphere. In this case we select the *'lh.sensorimotor_flat.3d'*, which is a flattened patch including the 'precentral' and 'postcentral' ROI of the aparc annotation. We could just as well have chosen *'lh.sensorimotor_flat.3d'*. After selecting the patch-file, press *'OK'*.

In the following menu you need to specify a number of Cgrid properties:



The *'Number of subgrids*' designates the number of ROIs or combination of ROIs that you want to include in the Cgrid. If you want to apply the Cgrid to a single ROI, this number should be 1. When you want the Cgrid to include multiple ROIs, this number should equal to how many times you need to specify the 4 borders. For example, when making a Cgrid of the sensorimotor cortex based on the precentral and postcentral ROI of the 'aparc' annotation, you have two options:

- Specify the number of subgrids as 1, and define the 4 borders once for the combined ROI. In this case you generate one subgrid containing two ROIs, but information on the location of the precentral-postcentral border in the Cgrid is lost.
- Specify the number of subgrids as 2, and define the 4 borders twice, once for the precentral and once for the postcentral ROI. When doing this, you will generate two subgrids that are concatenated along the y-axis, and the edge where they are concatenated represents the border between the precentral and postcentral gyrus (i.e. the depth of the central sulcus).

In this case we choose 2 subgrids, meaning that we will be defining the 4 borders separately for the precentral and postcentral ROIs.

The *'x-dimension*' and *'y-dimension*' specify the number of resolution elements of each subgrid along the two axes. Note that the subgrids are concatenated along the y-axis, so the *'x-dimension'* of the separate subgrids must be the same, and needs to be specified only once. The *'y-dimension'* can vary between subgrids. If the *'y-dimension'* is specified as a single number, as in this case, all subgrids have the same number of elements along the y-dimension. To specify different y-dimensions for each subgrid, provide a space-separated integer for each subgrid e.g. *'6 4'*

'Polynomial order' specifies the polynomial order to fit to the borders. The default is 0, which lets the program determine the optimal polynomial order automatically. If for some reason you want to set this number manually, enter any integer larger than 0. Note that depending on the number of coordinates in a border, a high polynomial order (around >15) can cause erratic behavior of the algorithm, so beware.

The *'Transpose axes*' and *'Invert x-axis*' buttons let you control the orientation of the resulting Cgrids:

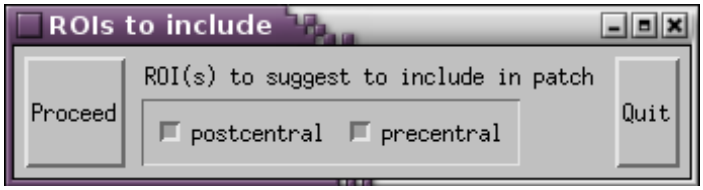
'Transpose axes' swaps the x and y-axes of the resulting Cgrid. E.g. when making a Cgrid of the sensorimotor cortex, you might want to concatenate GRIDS of the precentral and postcentral gyrus

(as in this example). As they need to be concatenated along the y-axis, the x-axis will run parallel to the central sulcus, thus horizontally. As in real life the central sulcus runs more or less vertically, this might not be an intuitive representation. By swapping the axes you can solve this issue.

'Invert x-axis' reveres the x-axis of the Cgrid. In case of defining Cgrids for the sensorimotor cortex, after swapping of the x and y-axis, the bottom of the Cgrid is the top of the brain, which is again not very intuitive. Inverting the x-axis solves this issue.

'Annotation scheme' sets the scheme from which the ROI-borders are extracted that define the Cgrid. Normally this is the same annotation scheme that you used when generating the flattened patches (but it does not have to be by definition). As the patch in this example consists of the precentral and postcentral ROIs from the 'aparc' annotation, that is the one we pick here.

After you press *'Proceed'* you will be shown a list of ROIs that are included in the selected patch according to the *'Annotation scheme'* that you selected in the previous menu.



The names of the selected ROIs will be included in the scheme-file as a reminder for users of what the patch-file on which the scheme is to be applied should include. Normally, only necessary ROIs are included in the representative patch that you selected at the start, and thus all should be selected (default). If the representative patch that you selected at the start includes more ROI's than necessary, you can unselect them here.

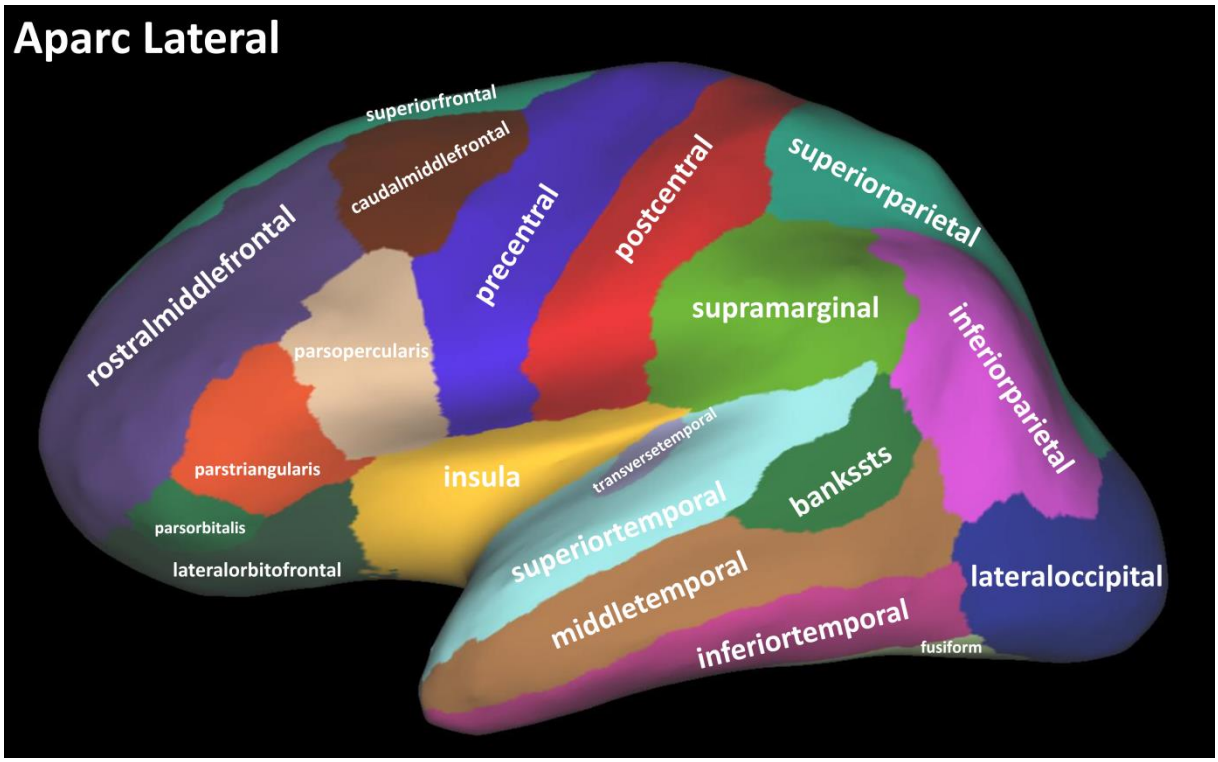
After you press '*Proceed*', you get to a menu with 4 tabs. Each tab will show all the borders that are present in the patch according to the specified annotation scheme, and the borders are designated by two ROI labels separated by an underscore. The order of the ROIs in the border definition is mostly, but not completely in alphabetical order. By selecting ROI-borers or combinations of ROI-borders you can specify the upper/lower/left/and right border of a subgrid, and you repeat this process for every subgrid in the Cgrid. Each border can be composed of one or more ROI-borders. Each ROI-border can be selected only once for each subgrid. The addendum of the manual contains pictures of an 'aparc' and a 'aparc.DKT' parcellation of a subject, that can help with defining borders. For inspecting other annotation schemes you can use FreeSurfer's 'tksurfer'.

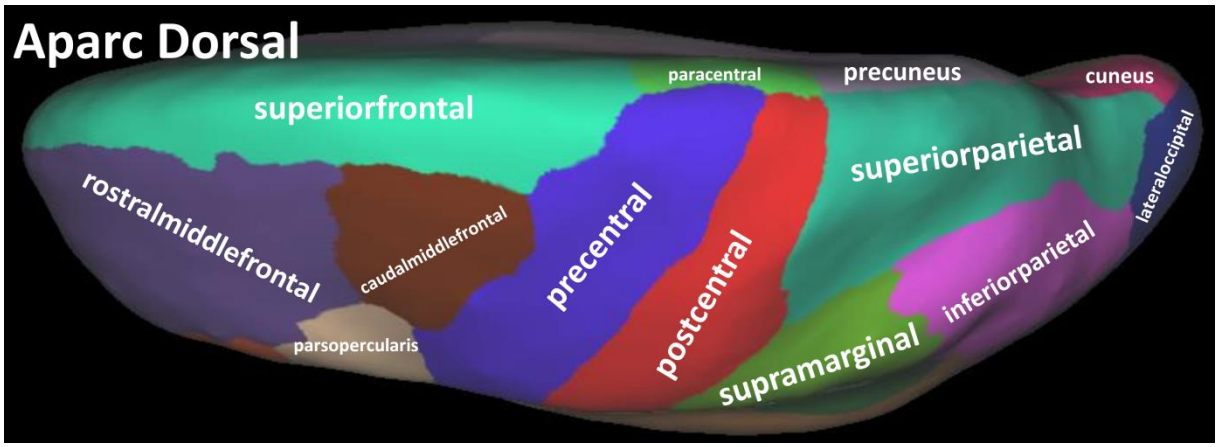
When selecting borders, keep the following in mind:

- The definition of the upper, lower, left, and right border is based on the perspective within the ***LEFT*** hemisphere. For the right hemisphere, the borders are automatically adjusted so that the final orientation of the grids is similar to those of the left hemisphere.
- Upper/lower/left/right do not refer to orientations in volumetric space, but instead to the orientations of the borders in the subgrid.

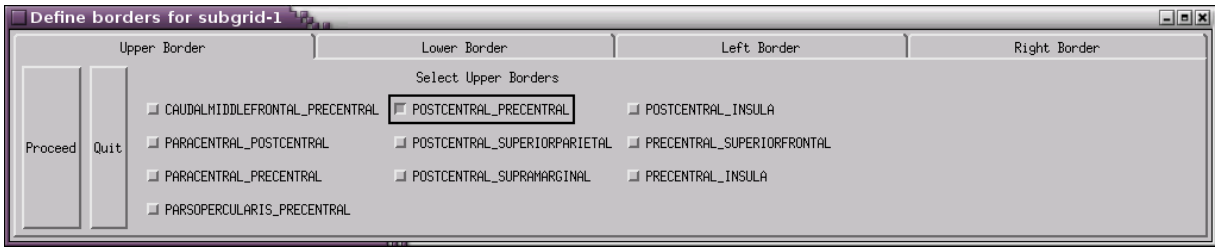
- Especially, for some more precise parcellation schemes ('aparc.a2009s') it can happen that the appropriate composition of borders for an intended Cgrid is different from subject to subject. E.g. a left border may consist of one ROI-border in one subject, and two ROI-borders in another subject. When you want to make a Cgrid scheme that you want to apply to multiple subjects, it is best to base it on the subject where the subgrid borders consist of the most ROI-borders. If an ROI-border is not detected in a particular subject, it is simply ignored, as long as the subgrid border can still be defined.

The example described in the following section describes the border selections for a Cgrid of the sensorimotor cortex, which includes the 'precentral' and 'postcentral' gyrus of the 'aparc' annotation. The four borders are defined separately for the precentral and postcentral gyrus. In this example, the borders for the precentral gyrus are defined first. As the subgrids are concatenated along the y-axis, the subgrid that is defined first will be at the bottom. It is of course also possible to start with the postcentral gyrus, but that would swap left/right and upper/lower definitions that are shown further downwards and the resulting Cgrids will be oriented differently. The pictures below show the relevant pictures of the 'aparc' annotation that you need for specifying the borders. The full range of viewpoints for the 'aparc' and the 'aparc.DKT' parcellations are in the addendum.

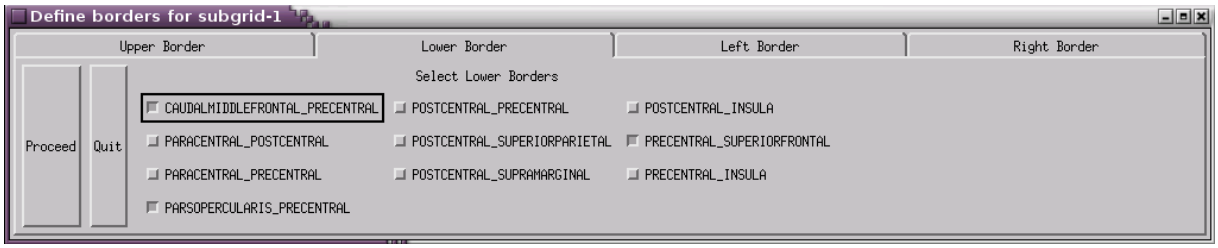




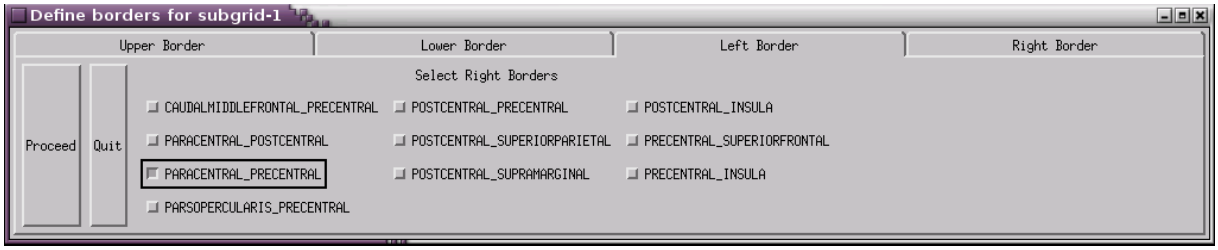
The upper border of the precentral gyrus is represented by the POSTCENTRAL_PRECENTRAL border:



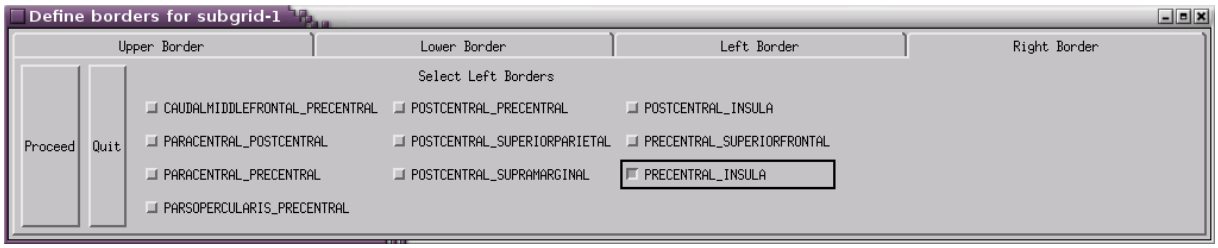
The lower border of the precentral gyrus is represented by 3 ROI-borders, including CAUDALMIDDLEFRONTAL_PRECENTRAL, PARSOPERCULARIS_PRECENTRAL, PRECENTRAL_SUPERIORFRONTAL



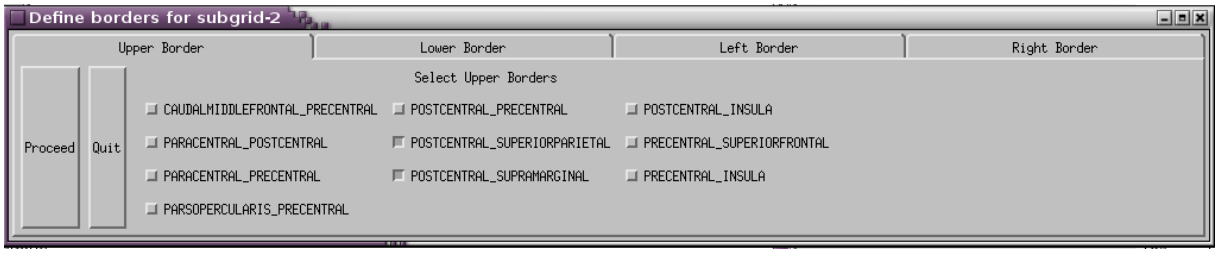
The left border of the precentral gyrus is represented by the PARACENTRAL_PRECENTRAL border:



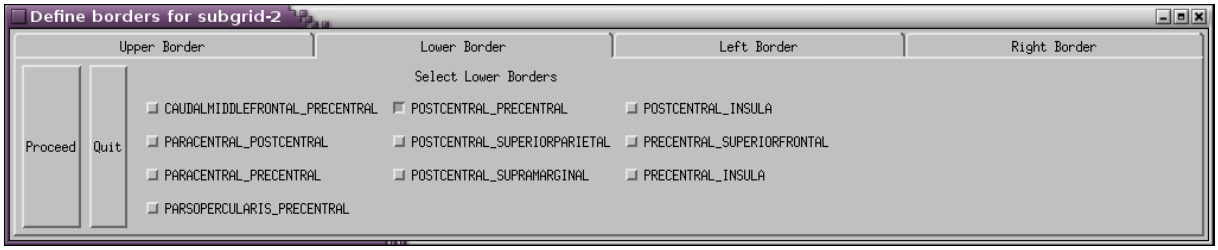
The right border of the precentral gyrus is represented by the PRECENTRAL_INSULA border:



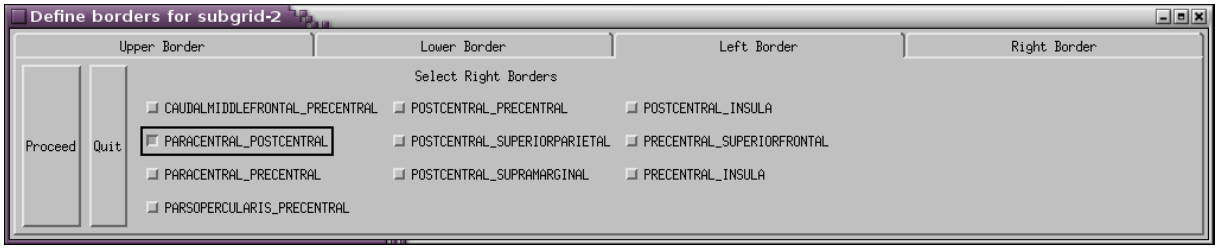
The upper border of the postcentral gyrus is represented by 2 ROI-borders including the POSTCENTAL_SUPERIORPARIETAL and and POSTCENTRAL_SUPRAMARGINAL:



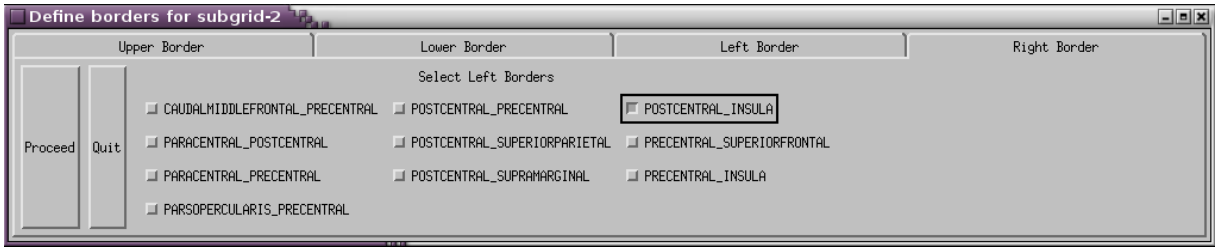
The lower border of the postcentral gyrus is represented by the POSTCENTAL_PRECENTRAL border:



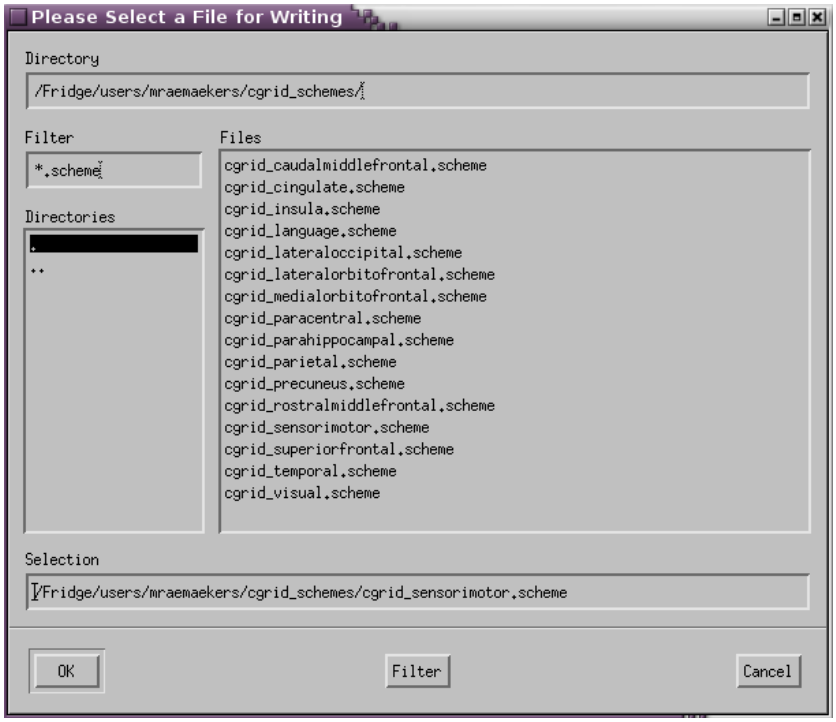
The left border of the postcentral gyrus is represented by the PARACENTRAL_POSTCENTRAL border:



The right border of the postcentral gyrus is represented by the POSTCENTRAL_INSULA border



After you defined the borders for each GRID, you can press '*Proceed*', and you can save to composed scheme under an intuitive name. For convenience, it's best to store them in the designated scheme folder (defined as $CGRID_SCHEMEDIR in your .bashrc).

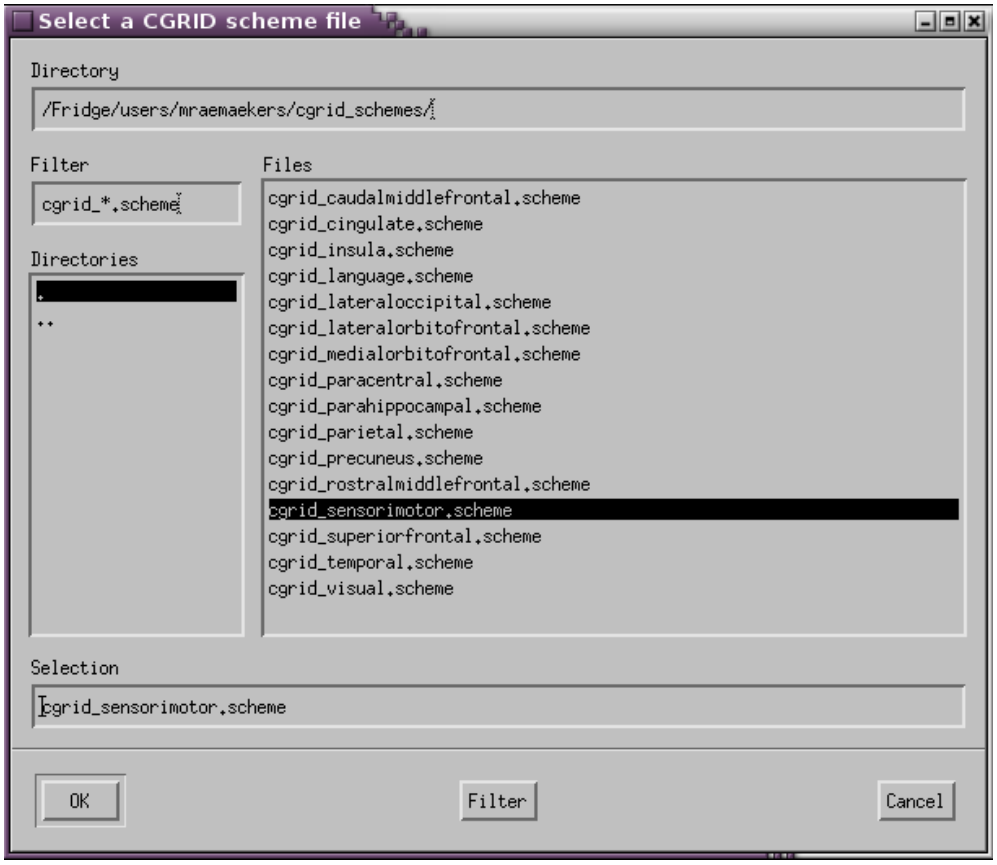


Note that scheme-files are simple ascii files that can also be composed and altered using a simple text editor. In some cases this might be more convenient than using the toolbox, especially when you want to adjust small things such as the resolution of the Cgrid. You can see the content of the scheme-file that was generated on the next page.

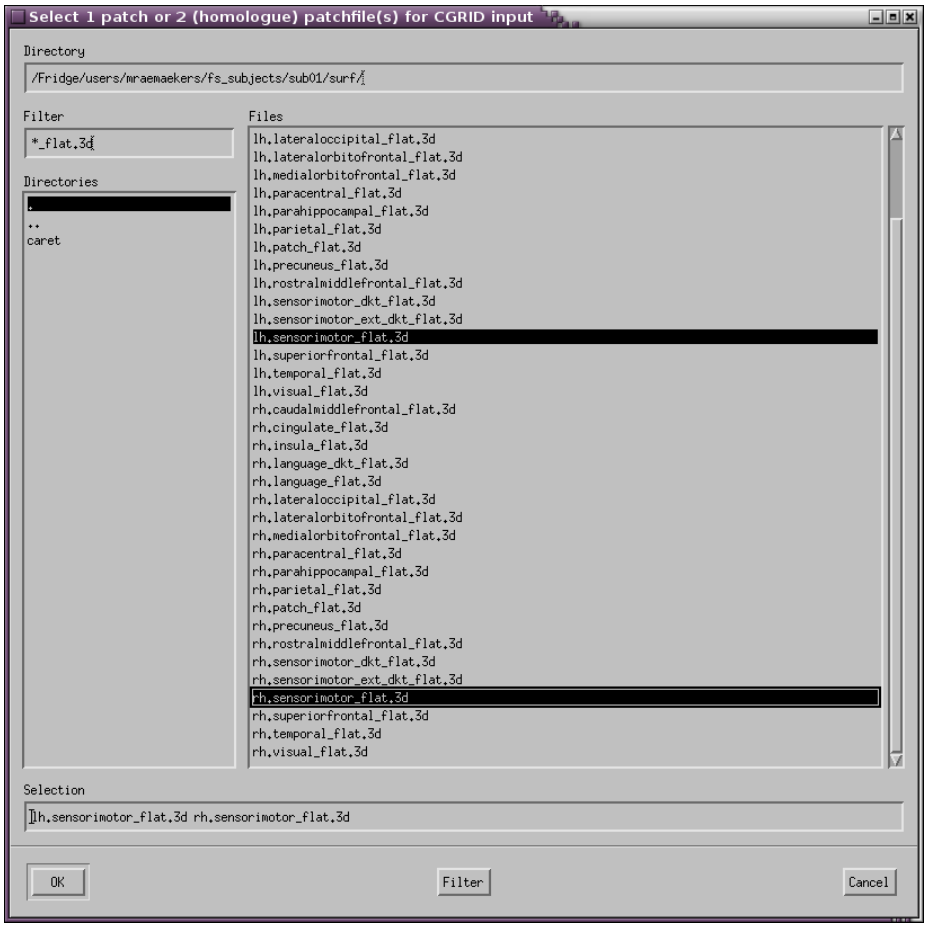
```
Number of subgrids:
2
Annotation scheme:
aparc
ROIs to include:
precentral,postcentral
x-dimension:
40
y-dimension(s):
10
Polynomial order:
0
Invert x-axis:
yes
Transpose axes:
yes
subgrid-1 upper border:
POSTCENTRAL_PRECENTRAL
subgrid-1 lower border:
CAUDALMIDDLEFRONTAL_PRECENTRAL
PARSOPERCULARIS_PRECENTRAL
PRECENTRAL_SUPERIORFRONTAL
subgrid-1 left border:
PARACENTRAL_PRECENTRAL
subgrid-1 right border:
PRECENTRAL_INSULA
subgrid-2 upper border:
POSTCENTRAL_SUPERIORPARIETAL
POSTCENTRAL_SUPRAMARGINAL
subgrid-2 lower border:
POSTCENTRAL_PRECENTRAL
subgrid-2 left border:
PARACENTRAL_POSTCENTRAL
subgrid-2 right border:
POSTCENTRAL_INSULA
```

6. **Generate Cgrids**

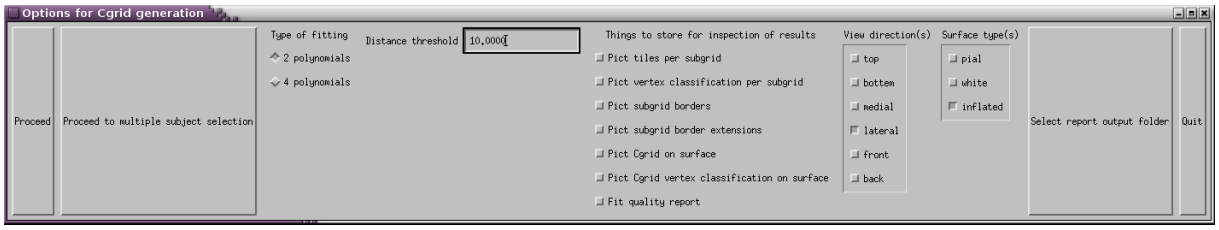
During the generation of the Cgrids, you are applying the schemes to the patch-files, thereby assigning Cgrid x and y coordinates to every vertex within the borders specified by the scheme-file. You start the process of making Cgrids by clicking on *'Generate Cgrids'* in the Cgrid-menu. The first step is the selection of the scheme that you want to apply. By default, the scheme-files are stored in $CGRID_SCHEMEDIR. In this example we generate Cgrids using the scheme stored as *'cgrid_sensorimotor.scheme'.*



After pressing *'OK'*, you can select the patches of a subject that you want to apply the scheme to. The patch-files were stored in the subject's 'surf' folder during 'Generate Patches' as lh/rh.prefix_flat.3d. You can select 1 or 2 patch-files. When you select two patches, they should be homologues (same area in the left and right hemisphere). You can select multiple patch-files by holding the control button during mouse clicks. In this case we select *'lh.sensorimotor_flat.3d'* and *'rh.sensorimotor_flat.3d'*. You can include multiple subjects for batch operations at a later stage.



After pressing *'OK'*, you get to the following menu that allows you to set a few options for generating the Cgrids:



The *'Type of fitting'* specifies how the x-coordinates of the subgrids are established. When selecting 2-polynomials, x-coordinates are established by subdividing the fitted polynomials along the upper and lower border (including also the interpolated polynomials), in portions of equal length. When selecting 4-polynomials, the x-coordinates are established by fitting and interpolating polynomials along the left and right border. While the 4-polynomials approach seems more elegant, using this technique results in shifts in the x-coordinate at borders between different subgrids of a Cgrid. As a result, the Cgrid will not run smoothly across subgrid borders. In addition, the 4 polynomial approach can fail when the left and right border have an angle of more than 90 degrees. Therefore, apply the following rules of thumb when choosing between 2 or 4 polynomial fitting:

- Use 4-polynomail fitting for Cgrids containing only a single subgrid.
- Use 2-polynomial fitting for Cgrids with multiple subgrids or when 4-polynomial fitting fails.

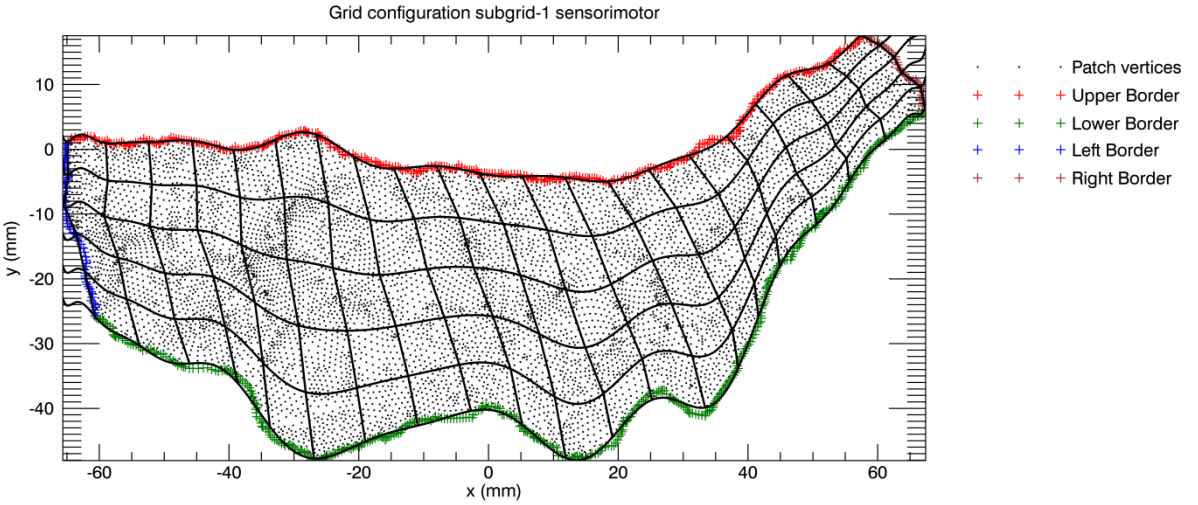
With *'Distance threshold'* you can choose to exclude coordinates or clusters of coordinates from borders that have a minimum distance larger than the specified threshold from the main cluster. This is sometimes necessary when there are 'island' in the border, which usually reflects errors in the

annotation or in the flattening. A low value (<2 mm) will remove all suspected 'erroneous' islands, but has a risk of also removing parts of the real/intended border.

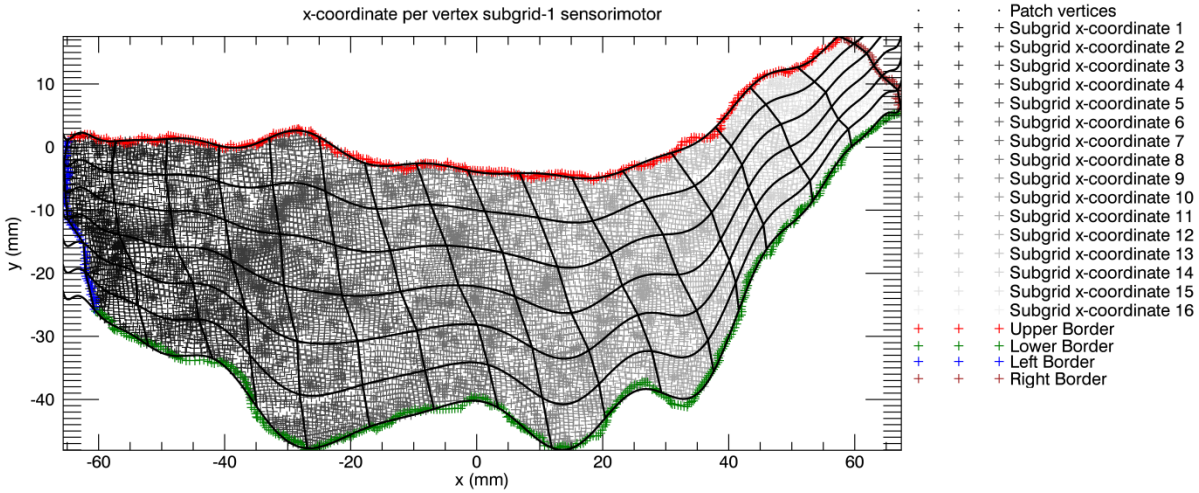
Under *'Things to store for inspection of results'* you can select graphs and a text-file that can be used for inspection of the results. While it's certainly not necessary to save them all all the time, it is advisable to always have some form of quality check of the resulting Cgrids, especially when you are applying newly created schemes. Note that the graphs and pictures shown in the section below are using a lower Cgrid resolution as defined in the example in *'Generate schemes'* to make the images easier to comprehend. Also note that flat patches for each subgrid are shown after a rotational optimization step, so their orientation differs from those in the patch-file.

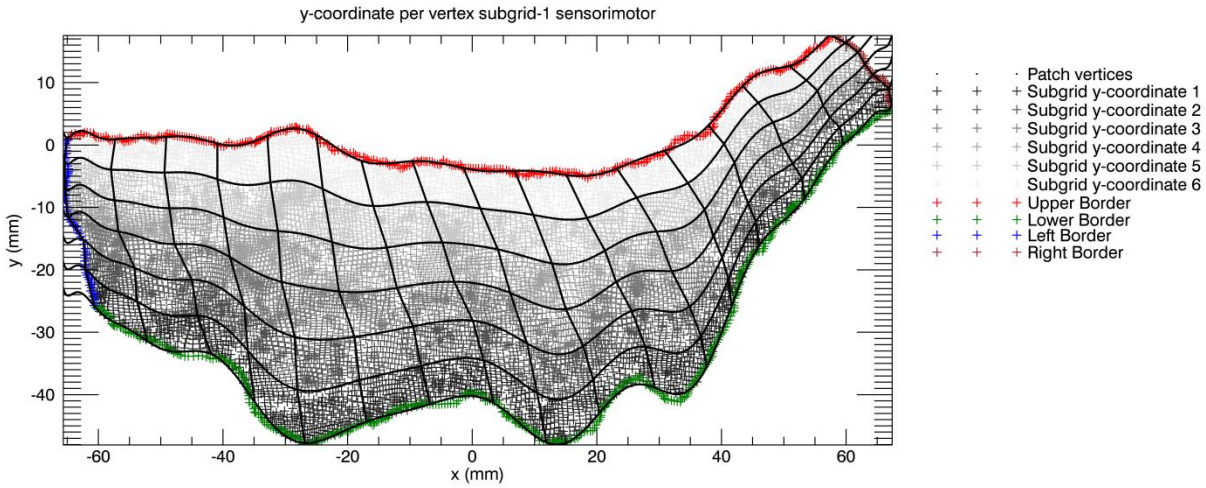
The following can be stored:

1. *'Pict tiles per subgrid'*: A graph for each subgrid of the Cgrid including the borders, vertices, and grid-tiles. If you want to create only a single plot for visual inspection, choose this one. An example (of a left 'precentral' subgrid) is shown below. The graphs are stored as *'subject_lh/rh_prefix_subgrid#.png'*.

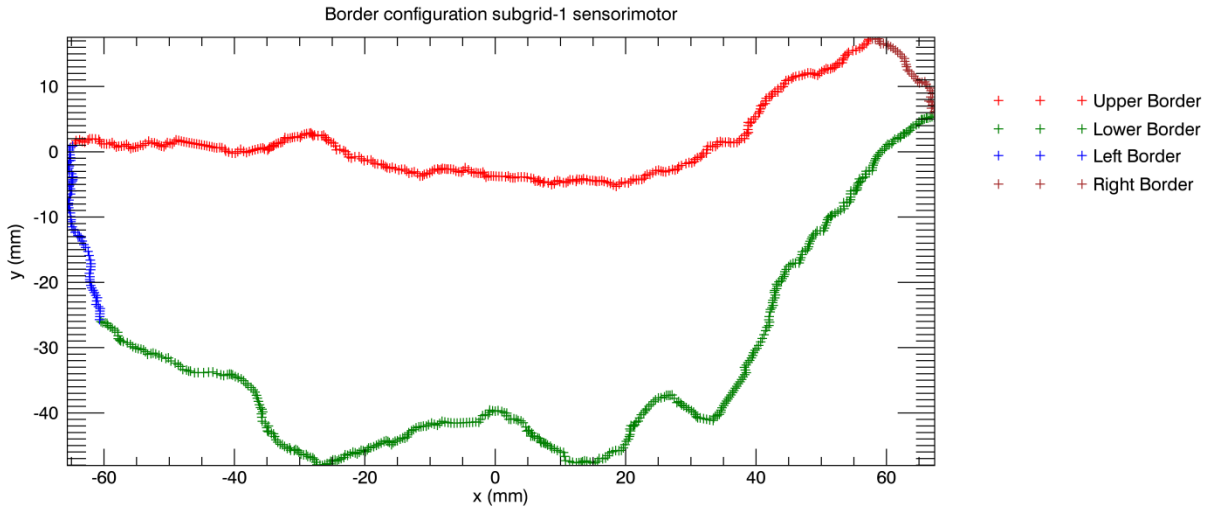


2. *'Pict vertex classification per subgrid'*: Two graphs for each subgrid in the Cgrid, one for the subgrid's x-coordinates and one for the subgrid's y-coordinates. The two graphs show a gray-scale color coding of the subgrid-coordinates for every vertex in the patch. The graphs are stored *as 'subject_lh/rh_prefix_subgrid#_x/yclass.png'*.

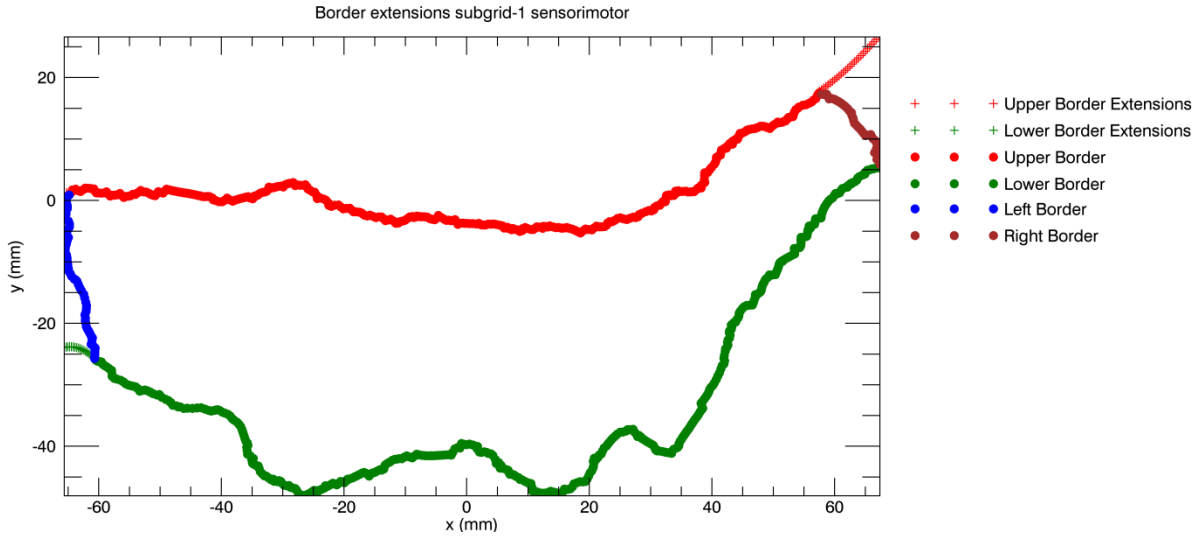




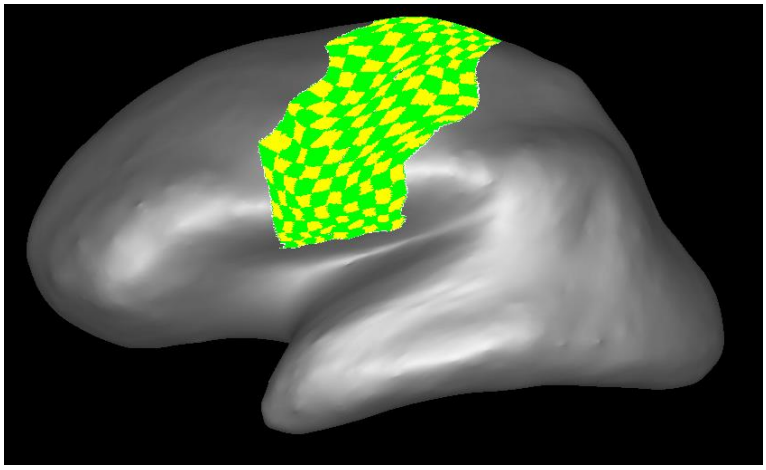
3. *'Pict subgrid borders'*: A simple graph depicting the upper, lower, left and right border of each subgrid in the Cgrid. The graphs are stored as *'subject_lh/rh_prefix_subgrid#_border_configuration.png'*.



4. *'Pict subgrid border extensions'*: The Cgrid toolbox uses a polynomial fitting and interpolation procedure that requires the upper and lower borders to be defined across the full x-range of the subgrid. As this is far from always the case, the upper and lower borders often need to be extrapolated. The 'plot border extension' shows the result of this extrapolation procedure, which are stored as *'subject_lh/rh_prefix_subgrid#_border_extensions.png'*. Note that for the 4-polynomial fitting procedure, both the upper/lower and the left/right borders are extrapolated, and these extrapolations are stored in separate graphs.

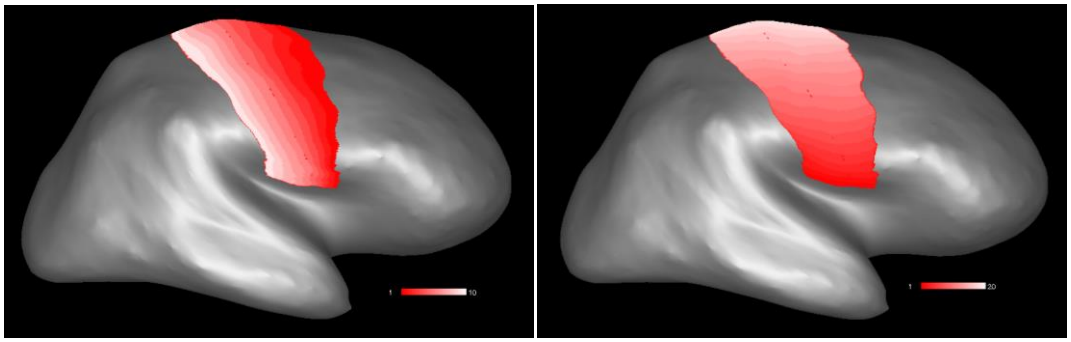


5. *'Pict Cgrid on surface'*: Create a picture of the Cgrid on the cortical surface represented as a chessboard pattern. Note that the 3D visualization of the toolbox has some limitations, which makes the chessboard pattern a little messy for higher resolution Cgrids, so only useful for visualizing low-resolution grids. Using the *'View direction for surface view(s)'* and the *'Surface type for surface view(s)'* panels on the right you can specify the view directions and types of surface used when generating the pictures. The pictures are stored as *'subject_lh/rh_prefix_cgrid_surface_surftype_viewdir.png'.*



6. *'Pict Cgrid vertex classification on surface'*: Create pictures of the Cgrid x and y-coordinates plotted on the cortical surface. Two pictures are stored for each Cgrid, one for the x and one for the y-coordinate. These pictures can be helpful for determining the final orientation of the Cgrids when experimenting with new schemes. Using the *'View direction for surface view(s)'* and the *'Surface type for surface view(s)'* panels on the right you can specify the views direction and types of surface used when generating the pictures. The pictures are stored as *'subject_lh/rh_prefix_cgrid_surface_surftype_viewdir_x/yclass.png'.*

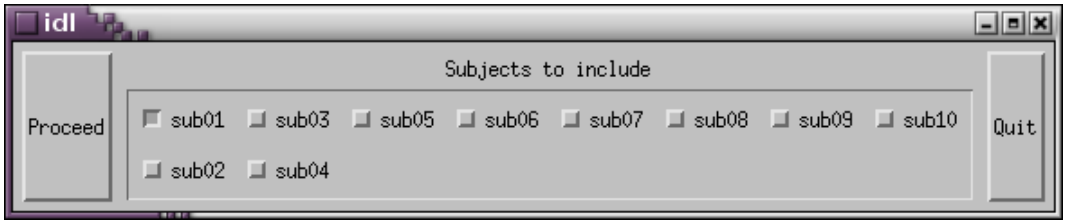
x-coordinate: y-coordinate



7. *'Fit quality report'* creates a text-file with the root mean square of error of the polynomial fit of the border, averaged for all fitted borders for each subgrid. These values can be more or less interpreted as the standard error of the position estimation within each subgrid. When it is small relative to the sizes of the tiles, the effect of this error is low. Together with visual inspection of the plot of the tiles it can be used to estimate how well the fitting procedure performed.

'Select report output folder' lets you specify where all the files created under *'Things to store for inspection of results'* are stored. By default they are stored in a subject's 'surf' folder of the FreeSurfer recon-all output, but here you can select an alternative. Note that the Cgrid files themselves (*'*.cgrid'* files) will always be stored in the subject's 'surf' folder of the FreeSurfer output.

When you are ready, you can choose *'Proceed'* or *'Proceed to multiple subject selection'*. When you choose *'Proceed'*, the Cgrids and requested pictures/data are generated for the patch-files that you selected at the start, and the resulting Cgrids will be stored in the subject's 'surf' folder of the FreeSurfer output as *'lh/rh_prefix.cgrid',* where the prefix is the prefix defined when generating patches (in this case *'sensorimotor'*). The '*.cgrid' files are tab delimited 3 column ascii files with the vertex number, the Cgrid x-coordinate, and the Cgrid y-coordinate for every vertex in the Cgrid. When you choose *'Proceed with multiple subject selection'*, the toolbox will search through the FreeSurfer $SUBJECTS_DIR to find all subjects for whom the selected patch-files are present, and will display them in the following selection menu:



After you press *'Proceed'*, the Cgrids and requested pictures/data are generated all subjects and patch-files that you selected.

The generation of Cgrids can take between seconds and several minutes for each subject, dependent on the surface area the Cgrid is covering, the number of subgrids, the required dimensions, and the available hardware. Note that the generation of graphs can substantially increase processing time.

7. Mapping volumes to Cgrids

With mapping volumes to Cgrid you can map the data from 3D or 4D nifti files onto the Cgrid, and subsequently store them in Cgrid-nifti-format. A Cgrid-nifti-file (*.nii) is in principle a normal nifti-file that can be used with any nifti-compatible software package. Its x and y dimensions are the dimensions of the Cgrid, and the z-dimension (length 2) represents the left (z-coordinate=1) and the right (z-coordinate=2) hemisphere. Multiple volumes can be represented by the 4th dimension, just as with the regular nifti format. Note that Cgrid-nifti-files of different subjects but based on the same Cgrid scheme are suitable as input for second-level statistics. The mapping to Cgrid occurs by first mapping volumetric data to the cortical surface, and subsequently assign values to the Cgrid by averaging across the vertices in each tile. Note that the contribution of each vertex to the mean of the tile is weighted by its corresponding surface area.

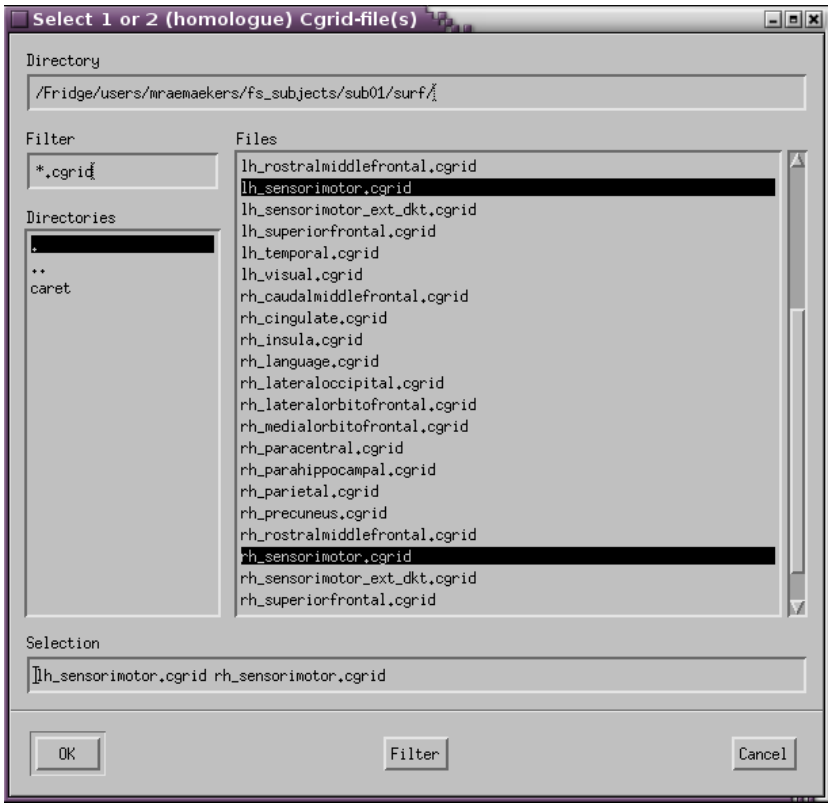
A prerequisite of the mapping to the surface to go successful is that the volumetric images are aligned with the FreeSurfer output. This is the case when the images you want to map are aligned with the 'T1.mgz' or 'brain.mgz' volumes in a subject's 'mri' folder of the FreeSurfer recon-all output. Note that it is not necessary that the images you want to map to Cgrid are resliced to these images (have the same dimensions) as e.g. the T1.mgz image, they only need to be aligned.

There are several software packages that can be used for aligning images. Here I will provide a brief outline of how to do this with SPM12.

- Convert the *'T1.mgz'* to nifti in the console by going to the 'mri' folder of a subject and type: *'mri_convert T1.mgz T1.nii'*
- Start SPM12 and select *'Coregister Estimate'* from the menu
- Select the newly created *'T1.nii'* as reference scan
- Select the nii image you want to map to Cgrid as *'Source Image'*. This can be any kind of image (T1, T2, T2*, B0) provided that it has anatomical contrast, so a statistical image will not do. Statistical images can however be provided under 'Other Images'.
- Optionally you can also select *'Other images'* that you want to map to Cgrid. These should already be aligned with the *'Source Image'*
- Press run (the green triangle)

Note that SPM12 does not create new files, but instead alters the header of the 'Source Image' and the 'Other files' so that they are now aligned with the FreeSurfer output. Keep in mind that the image registration is slightly more delicate when mapping data to the cortical surface then for volumetric analysis, as it requires alignment of the relatively thin layer of grey matter. Misalignment of images can thus have substantially larger effects. This issue is especially relevant when the images you want to map are geometrically distorted relative to the *'T1.mgz'*. In that case, use methods to remove these distortions whenever possible. When you are ready for mapping volumes to Cgrids, click on *'Map volumes to Cgrids'* in the Cgrid-menu.

You start the process of mapping volumetric data to a Cgrid by first selecting the Cgrid-files of a subject. The Cgrid-files are stored in the subject's 'surf' folder during 'Generate Cgrids' as *'lh/rh_prefix.cgrid'*. You can select 1 or 2 Cgrid-files. When you select two Cgrid-files, they should be homologues (of the same area in the left and right hemisphere). You can select multiple Cgrid-files by holding the control button during mouse clicks. In this case we select *'lh_sensorimotor.cgrid'* and *'rh_sensorimotor.cgrid'*. You can include multiple subjects for batch operations at a later stage.

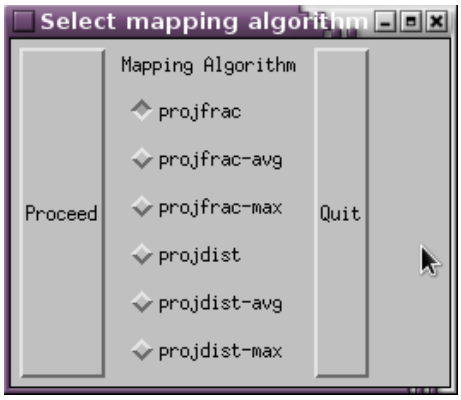


After pressing *'OK'* you can choose the algorithm for mapping the volumetric data to the surface. The possible algorithms are those provided by FreeSurfer's *'mri_vol2surf'*, which the Cgrid toolbox uses for mapping volumetric data to the surface. Documentation on FreeSurfer's 'mri_vol2surf' can be found on: https://surfer.nmr.mgh.harvard.edu/fswiki/mri_vol2surf.

The algorithm determines the way in which data is sampled from the volumes in order to assign values to the vertices of the cortical surface.

- Using one of the *'projfrac'* algorithms, you define the sample locations as fractions of the distance between the grey-white matter boundary and the pial surface (direction of the pial surface is positive)
- Using one of the *'projdist'* algorithms, you define the sample locations as distance in mm from the grey-white matter boundary (direction of the pial surface is positive)
- Using the *'projfrac'* or *'projdist'* algorithm you sample from a single distance
- Using an *'avg'* algorithm you sample from multiple points across a specified range that you subsequently average.
- Using a *'max'* algorithm you sample from multiple points across a specified range and pick the highest value.

There is no a-priori optimal algorithm, but the safest to use is *'projfrac'*. When the voxel-size of the volumes to be mapped is small (e.g. <1.5) you can also use *'projfrac-avg'*.

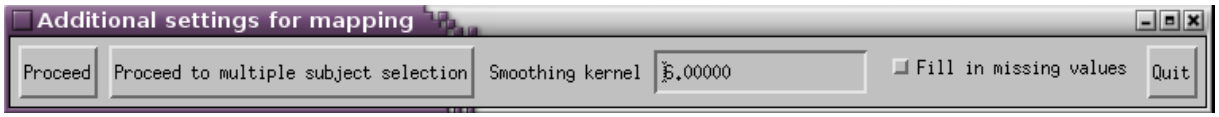


After pressing *'Proceed'*, you specify the details of the algorithm that you selected. When you selected *'projfrac'* you need to specify the fractional distance from where to sample, which is normally 0.5, meaning the middle of the gray matter:

When you selected *'projfrac-avg'* or *'projfrac-max'*, you select the range across which you want to sample, and the number of samples is determined by the *'Stepsize'*. In the settings below you sample across the entire thickness of the gray matter in steps of 0.1 the fraction of the gray matter thickness (thus 11 steps in total).

The principle for the *'dists'* algorithms is identical, except that you specify the algorithm's details in 'mm' instead of fractional distance.

After specifying the algorithm's details, you can specify a few additional things.

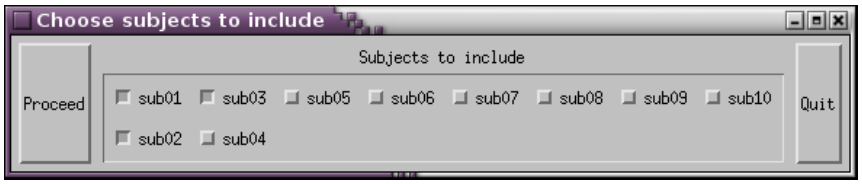


'Smoothing Kernel' specifies the FWHM in mm of the kernel used for smoothing across the cortical surface before mapping the data to the Cgrid. Applying a smoothing kernel is necessary when you want to perform second level analysis on the Cgrid-nifti files. When you don't want to perform smoothing, simply set the *'Smoothing kernel'* to 0. Note that it is strongly recommended that if you need to do smoothing, to do it at this stage (along the surface), and not on the resulting Cgrid-nifti-files, as they can be heavily geometrically distorted. When you decide to do smoothing on the Cgrid niti's anyway, make sure you are not using isometric smoothing, as you will be blending the left and right hemisphere. Instead set the FWHM of the smoothing kernel along the z-dimension to 0.

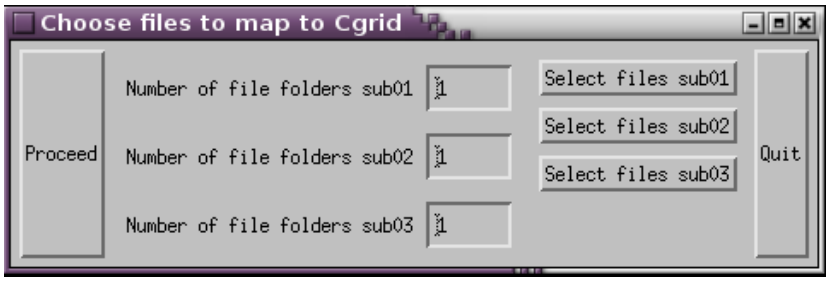
'Fill in missing values' is an option to assign values to Cgrid tiles that do not contain any vertices. While high resolution Cgrids are recommended as they discard the fewest information, the smaller the Cgrid tiles, the higher the chance on empty tiles. By default, empty tiles are assigned a value of 0.

This can however be inconvenient, e.g. when you want to perform second level analysis on the Cgrid-nifti files. Missing values across all subjects will then result in many missing values in the analysis. The *'Fill in missing values'* uses a slightly smoothed copy of the data to assign values to empty tiles. Note that this assignment only addresses empty Cgrid tiles. Tiles that get no values assigned due to the absence of volumetric data for the vertices included in the tiles, remain unaffected.

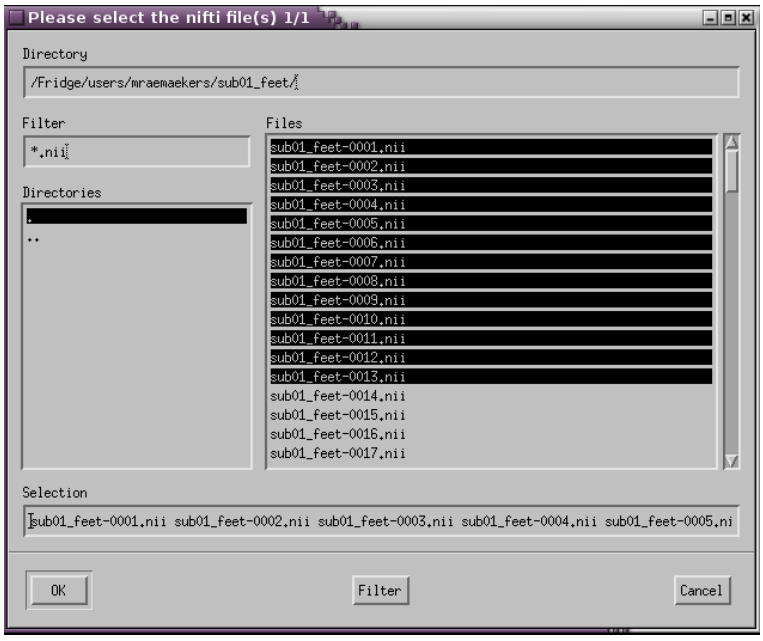
When you are ready, you can choose *'Proceed'* or *'Proceed to multiple subject selection'*. When you choose *'Proceed'*, you immediately continue to the file-selection procedure (see further downward). When you choose '*Proceed to multiple subject selection',* the toolbox will search through the FreeSurfer $SUBJECTS_DIR to find all subjects for whom the selected Cgrid-files are present, and will display them in the following selection menu:



After selecting e.g. three subjects, press *'Proceed',* and we can start with the file selection procedure.



Here you can specify for each subject from how many different folders the nifti files for mapping are to be selected. The file selection widget does not allow selecting files from different folders, therefore the selection needs to be done for every folder separately. Thus if possible, move all nifti files of a subject to a single folder first. When you specified the number of folders for a subject, press 'Select files *', and you can start selecting the nifti files.



You can select multiple nifti files by pressing control during mouse clicks, or by holding the mouse button. For especially long timeseries (e.g. >100 images) it is recommended that you merge 3D nifti files into 4D nifti files. This merging can be done e.g. using FSL's 'fslmerge' or using SPM. The mapping goes substantially quicker per volume with 4D files then with separate 3D files. Note that selected nifti's do not need to be of the same resolution, they only need to be aligned with the FreeSurfer's T1.mgz or brain.mgz. After pressing *'OK'* you will go either to a new file selection widget (to select files from another folder), or go back to the menu for selecting nifti files for other subjects. After having selected the files for every subject, press *'Proceed',* and the mapping starts.
The Cgrid-nifti-files will be stored in the same folders as the volumetric nifti's that were used as input. The Cgrid nifti-files carry the same name as the input, but have a *'_cgrid_prefix.nii'* extension. When the input files are numbered, as with e.g. fMRI data, the extension is put before the number.

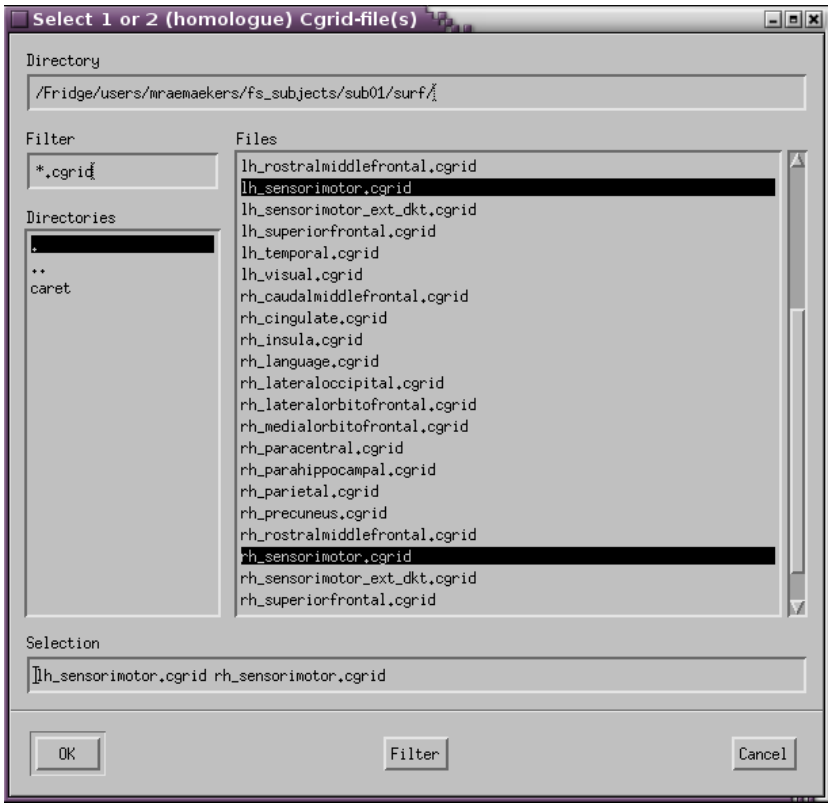
Mapping of volumetric data to Cgrid takes a few seconds for every 3D nifti volume, so with long fMRI timeseries, this process can be quite lengthy. Processing time can however be radically shortened by using 4D nifti files.

8. Mapping surface data to Cgrids

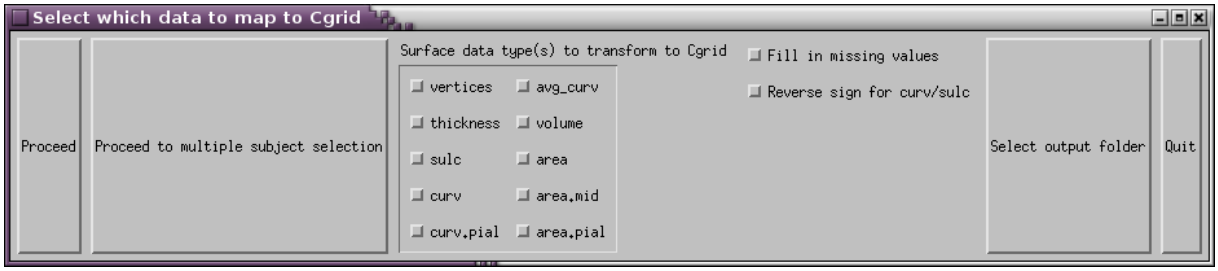
By mapping surface data to Cgrid, you can put several types of surface based information onto the Cgrid, and store it as a Cgrid-nifti-file. A Cgrid-nifti-file (*.nii) is in principle a normal nifti-file that can be used with any nifti-compatible software package. Its x and y dimensions are the dimensions of the Cgrid, and the z-dimension (length 2) represents the left (z-coordinate=1) and the right (z-coordinate=2) hemisphere. Note that Cgrid nifti files of different subjects but based on the same Cgrid schemes are suitable for second-level statistics.

By default, the FreeSurfer recon-all pipeline generates several types of surface based data including amongst others curvature, cortical depth, and cortical thickness that you might want to map to a Cgrid. E.g., the curvature and cortical depth in Cgrid space may be used as background for projecting fMRI results in Cgrid-nifti-file format upon. Furthermore, this mapping can provide some important information on generated Cgrids such as the number of vertices per tile and the total surface area or volume per tile. You can begin the process of mapping surface data to Cgrids by clicking on *'Map surface data to Cgrids'* in the Cgrid-menu.

You start the process of mapping surface data to a Cgrid by first selecting the Cgrid-files of a subject. The Cgrid-files are stored in the subject's 'surf' folder during 'Generate Cgrids' as *'lh/rh_prefix.cgrid'*. You can select 1 or 2 Cgrid-files. When you select two Cgrid-files, they should be homologues (same area in the left and right hemisphere). You can select multiple Cgrid-files by holding the control button during mouse clicks. In this case we select *'lh_sensorimotor.cgrid'* and *'rh_sensorimotor.cgrid'*. You can include multiple subjects for batch operations at a later stage.



After pressing *'OK'*, you can choose which type of surface data you want to map to Cgrid.



You can choose to generate the following output:

- *'vertices'* maps the number of vertices in each CGRID-tile
- *'thickness'* maps the mean gray matter thickness of the vertices within each Cgrid-tile
- *'sulc'* maps the mean sulcal depth of the vertices within each Cgrid-tile
- *'curv'* maps the mean white-matter-surface curvature of the vertices within each Cgrid-tile
- *'curv-pial'* maps the mean pial-surface-curvature of the vertices within each Cgrid-tile
- *'avg_curv'* maps the template-curvature of the vertices within each Cgrid-tile
- *'volume'* maps the total gray-matter volume of the vertices within each Cgrid-tile
- *'area'* maps the total white matter surface area of the vertices within each Cgrid-tile
- *'area.mid'* maps the total surface 'layer 4' surface area of the vertices within each Cgrid-tile
- *'area.pial'* maps the total pial surface area of the vertices within each Cgrid-tile

Note that for the curvature, sulcal depth, and thickness metrics, the contribution of each vertex to the mean of the tile is weighted by its corresponding surface area.

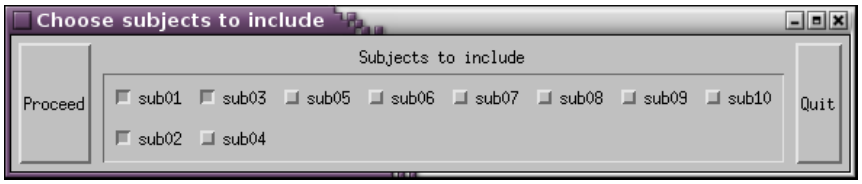
'Fill in missing values' is an option to assign values to Cgrid tiles that do not contain any vertices. While high resolution Cgrids are recommended as they discard the fewest information, the smaller the Cgrid tiles, the higher the chance on empty tiles. By default, empty tiles are assigned a value of 0. This can however be inconvenient, e.g. when you want to perform second level analysis on the Cgrid-nifti files. Missing values across all subjects will then result in many missing values in the analysis. The *'Fill in missing values'* uses an iterative process to fill in the values of empty tiles by averaging the values of the non-zero first order neighboring tiles.

'Reverse sign for curv/sulc' multiplies any generated Cgrid-nifti-file that includes curvature or sulcal depth information with -1. The reason for this option is that while the FreeSurfer output is correct regarding the sign of the curvature and sulcal depth (e.g. sulci have a large cortical depth and therefore have a high value), it does not connect very well with our intuition that sulci should be dark and gyri should be bright. Reversing the sign can make results more easily interpretable when e.g. you want to use the output as visual background for fMRI results.

'Select output folder' lets you specify where all the created Cgrid-nifti-files should be stored. By default they are stored in a subject's 'surf' folder of the FreeSurfer output, but here you can select an alternative.

When you are ready, you can choose *'Proceed'* or *'Proceed to multiple subject selection'*. When you choose *'Proceed'*, you immediately map the surface data to the Cgrid for the selected subject. When

you choose '*Proceed to multiple subject selection',* the toolbox will search through the FreeSurfer $SUBJECTS_DIR to find all subjects for whom the selected Cgrid-files are present, and will display them in the following selection menu:

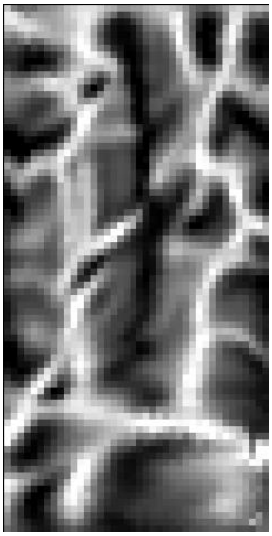


Mapping of surface data to Cgrid is not a lengthy process, taking only a few seconds every surface type. Some examples of the results of a Cgrid of the sensorimotor cortex can be seen below:

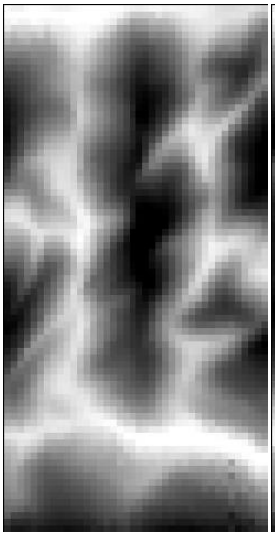
The figures below show pictures of the Cgrid-nifti-files of some of the possible outputs.

Curvature Left sensorimotor

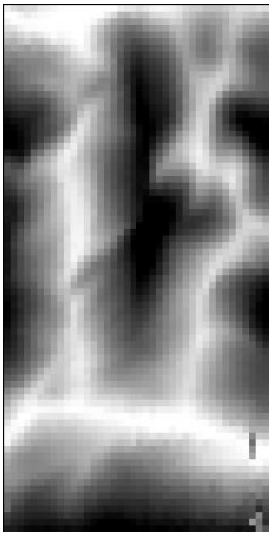
Curvature right sensorimotor



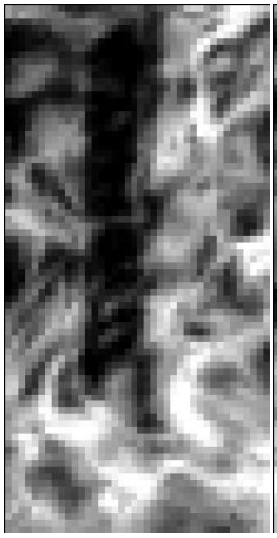
Sulcal depth left sensorimotor



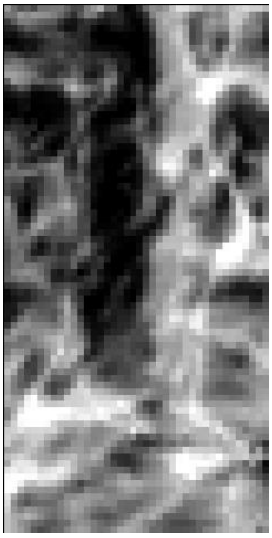
Sulcal depth right sensorimotor



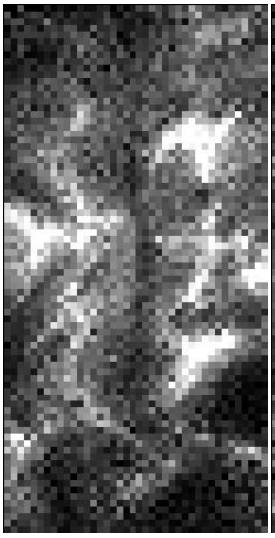
Thickness Left sensorimotor



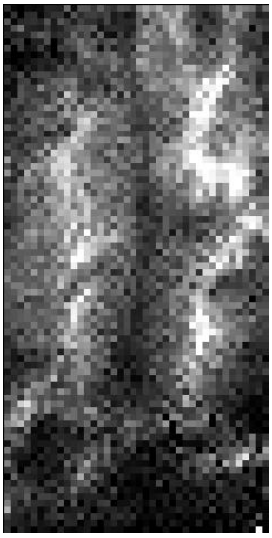
Thickness right sensorimotor



Area.mid left sensorimotor



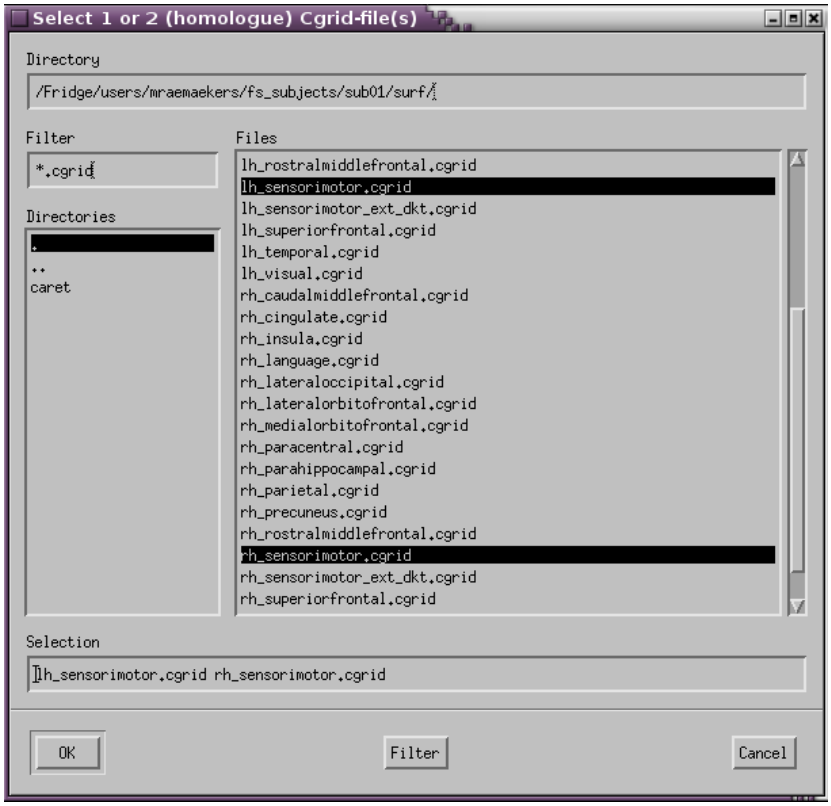
Area.mid right sensorimotor



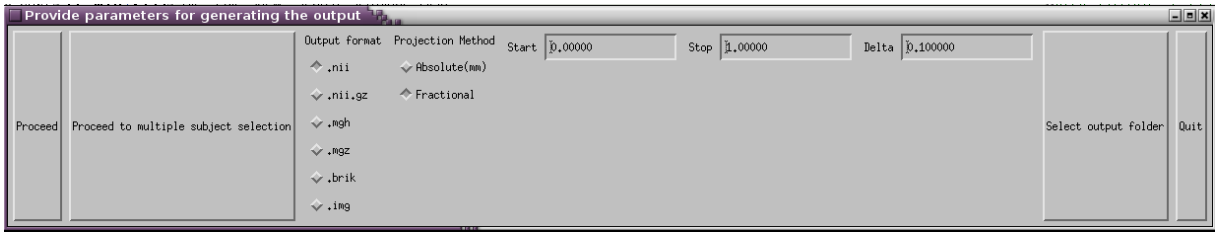
9. Mapping Cgrids to volume

By mapping a Cgrid to a volume, you use the generated Cgrid-files to assign x and y-coordinates to a volume, and storing it in a volumetric format. Separate volumes are created for the x-coordinate and the y-coordinate, and for the left and right hemisphere. The reference volume for mapping the Cgrid coordinates is the *'T1.mgz'* in a subject's *'mri'* folder of the FreeSurfer output. Under normal circumstances, it is recommended to analyze and visualize the data in Cgrid space itself (thus as Cgrid-nifti files), as this provides the benefits of a Cartesian grid, and direct compatibility with second level analysis. However, there are circumstances when projecting back to volumetric space is necessary. For example, when you would like to use Cgrid coordinates to define inclusion areas for DWI tractography, or when you want to use the toolbox to simply define limited portions of FreeSurfer ROIs. Due to limits in FreeSurfer algorithms, it's not possible to map Cgrids with x or y dimensions larger than 100 to a volume. To begin mapping Cgrids to volume click on *'Map Cgrids to volume'* in the Cgrid-menu.

You start the process of mapping Cgrids to a volume by first selecting the Cgrid-files of a subject. The Cgrid-files are stored in the subject's 'surf' folder during 'Generate Cgrids' as *'lh/rh_prefix.cgrid'*. You can select 1 or 2 Cgrid-files. When you select two Cgrid-files, they should be homologues (same area in the left and right hemisphere). You can select multiple Cgrid-files by holding the control button during mouse clicks. In this case we select *'lh_sensorimotor.cgrid'* and *'rh_sensorimotor.cgrid'*. You can include multiple subjects for batch operations at a later stage.



After pressing *'OK'*, you can specify the algorithm for mapping the Cgrid to the volume and the output.



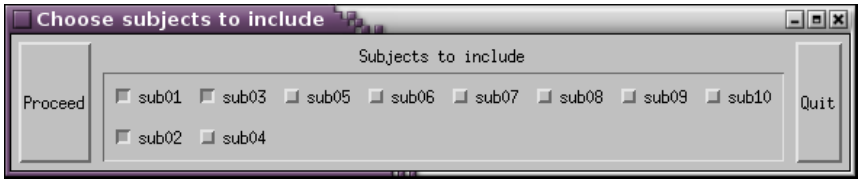
The *'Output format'* specifies how the volumetric results are stored. The default is nifti (*'nii'*), but as this can be a little rough on disk space, other options are available. You can always change nifti output afterwards using FreeSurfer's *'mri_convert'*.

The *'Projection Method'* sets the unit in which is specified how voxels are labeled by the vertices in the Cgrid. By using *'Fractional'*, voxels are labeled according to distances specified as a fraction of the gray matter thickness. When using *'Absolute'*, the labeling is done based on absolute distances in mm. The *'Start'*, *'Stop'*, and *'Delta'* parameters define the locations in volumetric space, and thus the voxels, that a vertex within the Cgrid will label. *'Start'* should always be a smaller number than *'Stop'*. *'Delta'* should always be a positive value, and smaller than the difference between *'Start'* and '*Stop'*.

In this example the method is *'Fractional'*. The *'Start'* set at 0, means that the labeling starts at the gray-white matter boundary. A *'Stop'* at 1 means that it stops at the pial surface. A *'Delta'* 0f 0.1 means that the labeling is performed in steps of 0.1 (thus 11 steps in total) along the surface normal vector of the vertex. A smaller *'Delta'* means more complete labeling, but also increases processing time. For using the Cgrid toolbox for creating inclusion/exclusion areas for DWI research, it can be helpful to also assign labels just below the gray-white matter boundary. This can be done by setting the *'Start'* at a negative value.

'Select output folder' lets you specify where all the created Cgrid-volumetric-nifti-files should be stored. By default they are stored in a subject's *'mri'* folder of the FreeSurfer output, but here you can select an alternative.

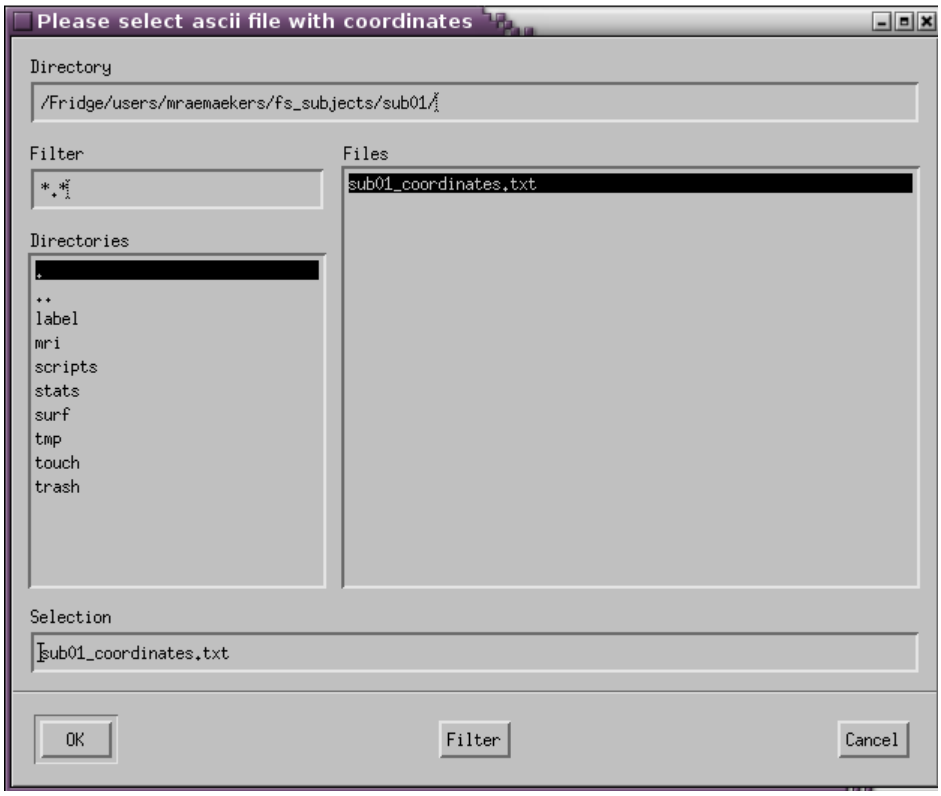
When you are ready, you can choose *'Proceed'* or *'Proceed to multiple subject selection'*. When you choose *'Proceed'*, you immediately map the Cgrid data to the volume for the selected subject. When you choose '*Proceed to multiple subject selection',* the toolbox will search through the FreeSurfer $SUBJECTS_DIR to find all subjects for whom the selected Cgrid-files are present, and will display them in the following selection menu:



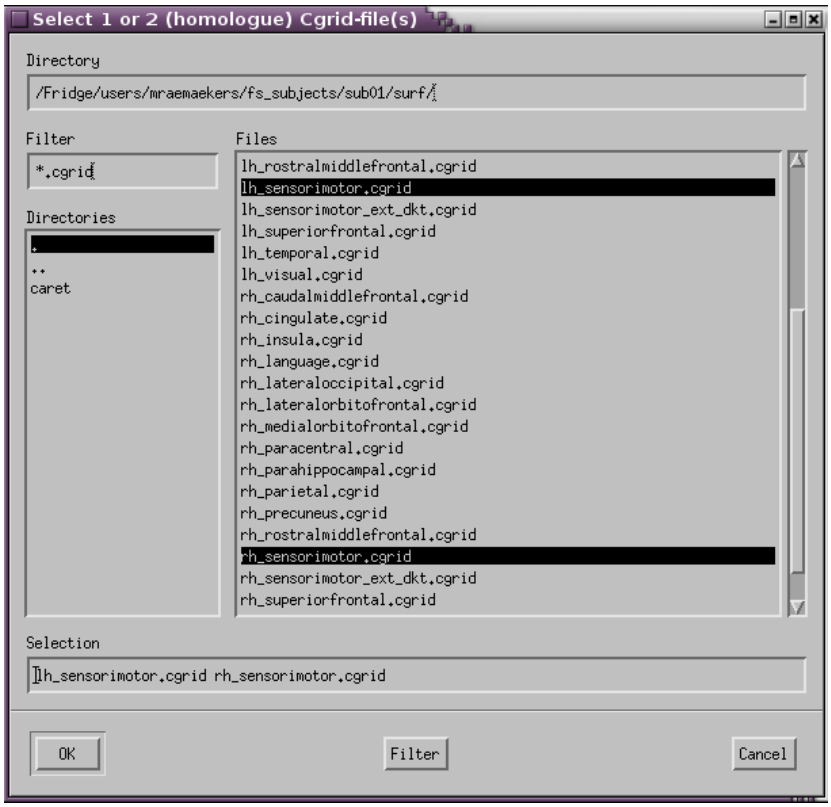
Mapping to a volume takes several minutes per subject, depending on available hardware, the size of the Cgrid, and the number of steps (defined by the *'Delta'*) when labeling voxels based on the values of vertices.

10. Mapping coordinates to Cgrids

By mapping coordinates to Cgrids, you can find Cgrid coordinates and other information for a list of coordinates in 3D space. This function is relevant e.g. when you would like to establish Cgrid coordinates for ECOG electrode positions. The 3D coordinates should be provided as a three column 'tab' delimited ascii-file, with a row for each coordinate. The position of the 3D-coordinates should be defined according to the volume space of the subject's *'T1.mgz'* in the *'mri'* folder of the FreeSurfer recon-all output (so not in FreeSurfer's RAS coordinates). The output is an ascii table containing a coordinate number, the original supplied coordinates, the RAS (Right, Anterior, Superior) coordinates, the vertex to which the coordinate is projected and corresponding coordinates, the distance along which the coordinate is projected, and the Cgrid x and y-coordinate. To begin mapping coordinates to Cgrid, click on *'Map coordinates to Cgrids'* in the Cgrid-menu. The process starts by selecting the ascii-file with the coordinates.



After that, you select the Cgrid-files for acquiring the Cgrid coordinates for each 3D coordinate. The Cgrid-files are stored in the subject's 'surf' folder during 'Generate Cgrids' as *'lh/rh_prefix.cgrid'*. You can select 1 or 2 Cgrid-files. When you select two Cgrid-files, they should be homologues (of the left and right hemisphere). You can select multiple Cgrid-files by holding the control button during mouse clicks. In this case we select *'lh_sensorimotor.cgrid'* and *'rh_sensorimotor.cgrid'*.

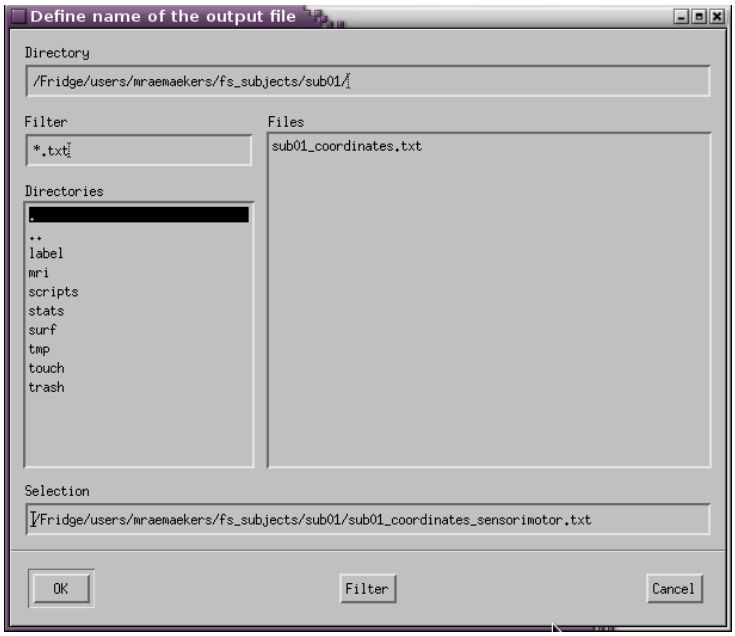


After you selected the Cgrid-files, press *'OK'*, you will get some additional options for storing data.

By selecting an *'Annotation scheme'*, you can supplement the output table with the labels of the vertices to which the 3D coordinates are projected. You can select multiple schemes.

With *'Store Cgrid nifti'* you can store the electrode positions in a Cgrid-nifti- file, which consists of the number of 3D coordinates projected to each Cgrid position. Normally this content of the nifti file consists of zero's and ones, but with low resolution Cgrids, or high density electrode grids, it can happen that multiple electrodes are mapped to the same Cgrid coordinate. The nifti file is stored in the same folder and under the filename as the output table, only with a '.nii' instead of a '.txt' extension. To see where the electrodes are located, you can use the generated nifti-file as an overlay for e.g. a 'sulc' Cgrid-nifti-file that you generated with *'Map surface data to Cgrids'*.

After selecting *'Proceed'*, you can select the filename under which to store the table with the projected coordinate information. By default it is stored in the same folder and under the same filename as the 3D coordinates, but including an extension with the prefix of the used Cgrid.

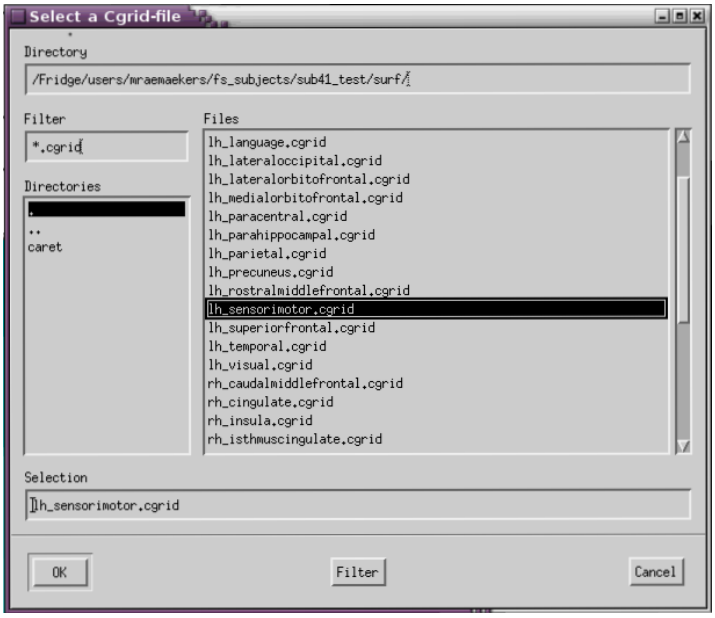


Mapping electrodes usually does not take longer than a few seconds.

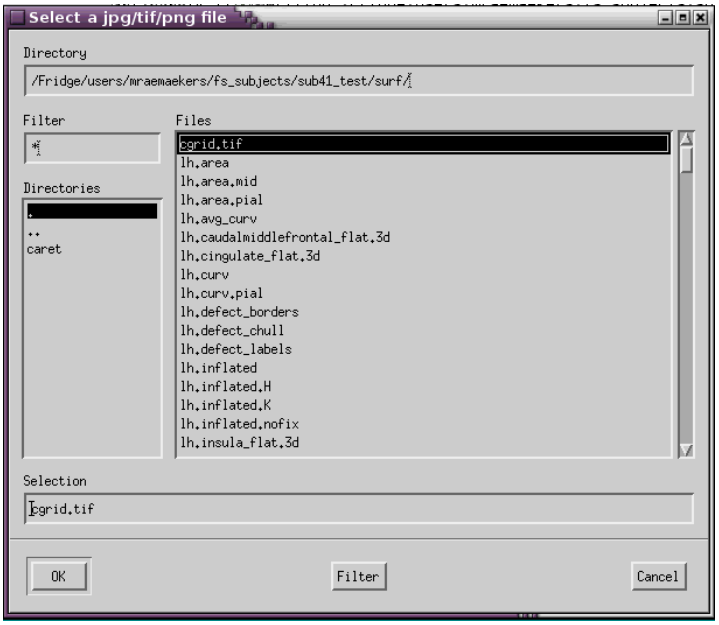
11. Mapping images to Cgrids

Using mapping images to Cgrid you can use generated Cgrids to map jpg/png/tif images to the surface of the brain. While not the central feature of the toolbox, it can be a useful tool for creating artwork, or to create images for visually inspect the geometric features of the created Cgrids. When mapping images to Cgrids, the images are first resampled to the dimension of the Cgrid, converted to greyscale if necessary, and scaled to values between -128 and 128. Subsequently every vertex of a particular Cgrid tile is assigned to corresponding image value. Note that the quality of the results is limited by the number of vertices or the spatial resolution of the Cgrid. This normally means that details get lost, depending on the content of the image.

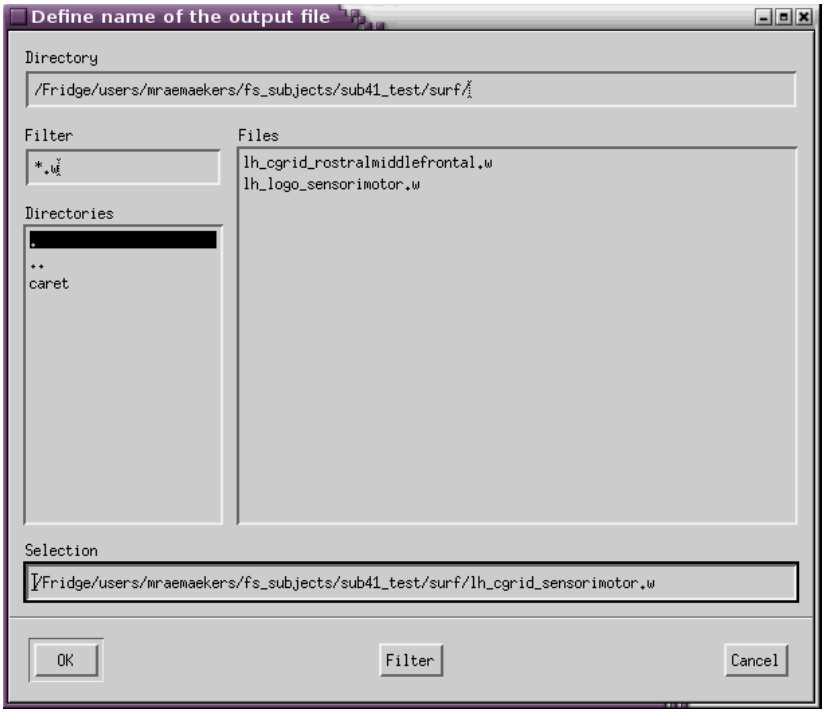
You start the process of mapping an image to a Cgrid by first selecting a Cgrid-file of a subject. The Cgrid-files are stored in the subject's 'surf' folder during 'Generate Cgrids' as *'lh/rh_prefix.cgrid'*. In this case we select *'lh_sensorimotor.cgrid'*.



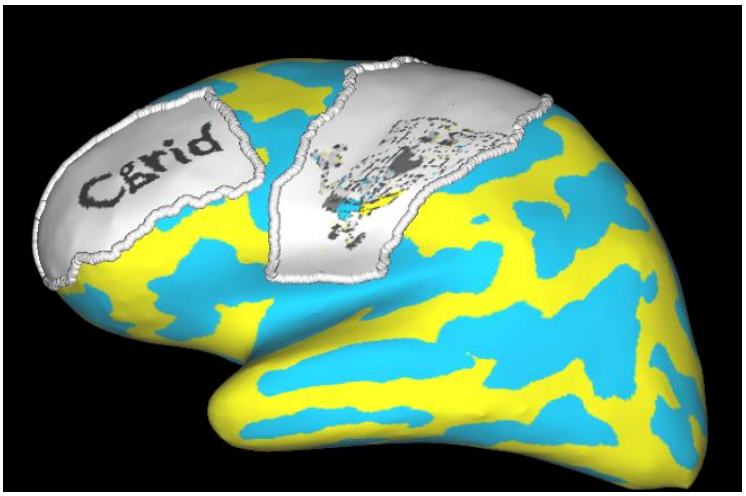
After pressing 'OK', you can select a jpg, png, or tif tile that you want to map to the selected Cgrid.



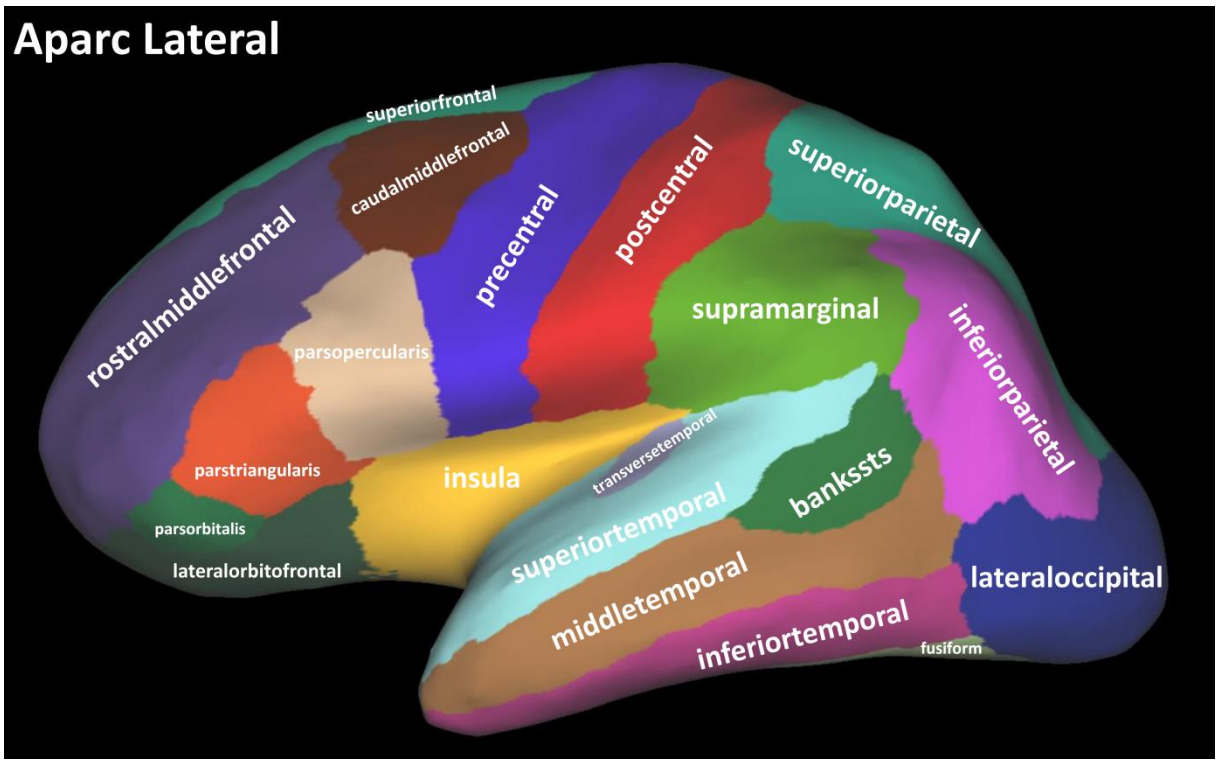
After pressing ‘OK’, you can specify the name of the ‘w’ file that contains the image mapped to the Cgrid.

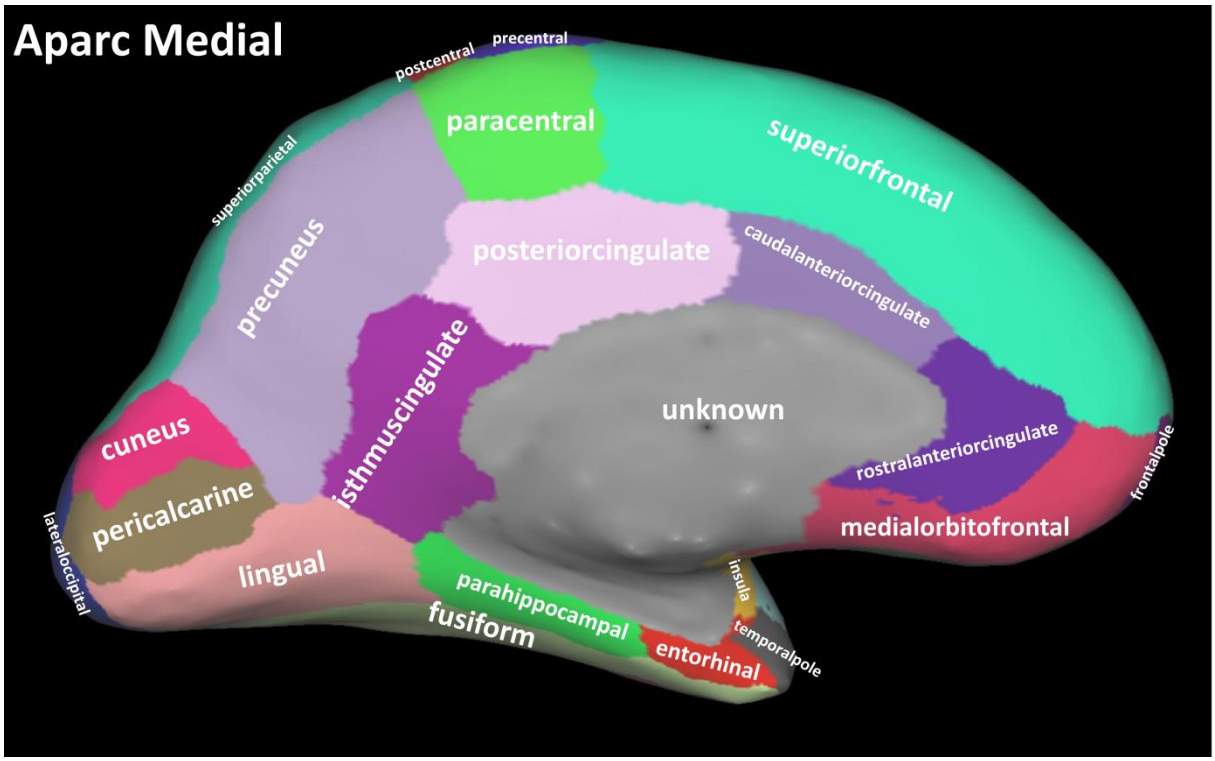


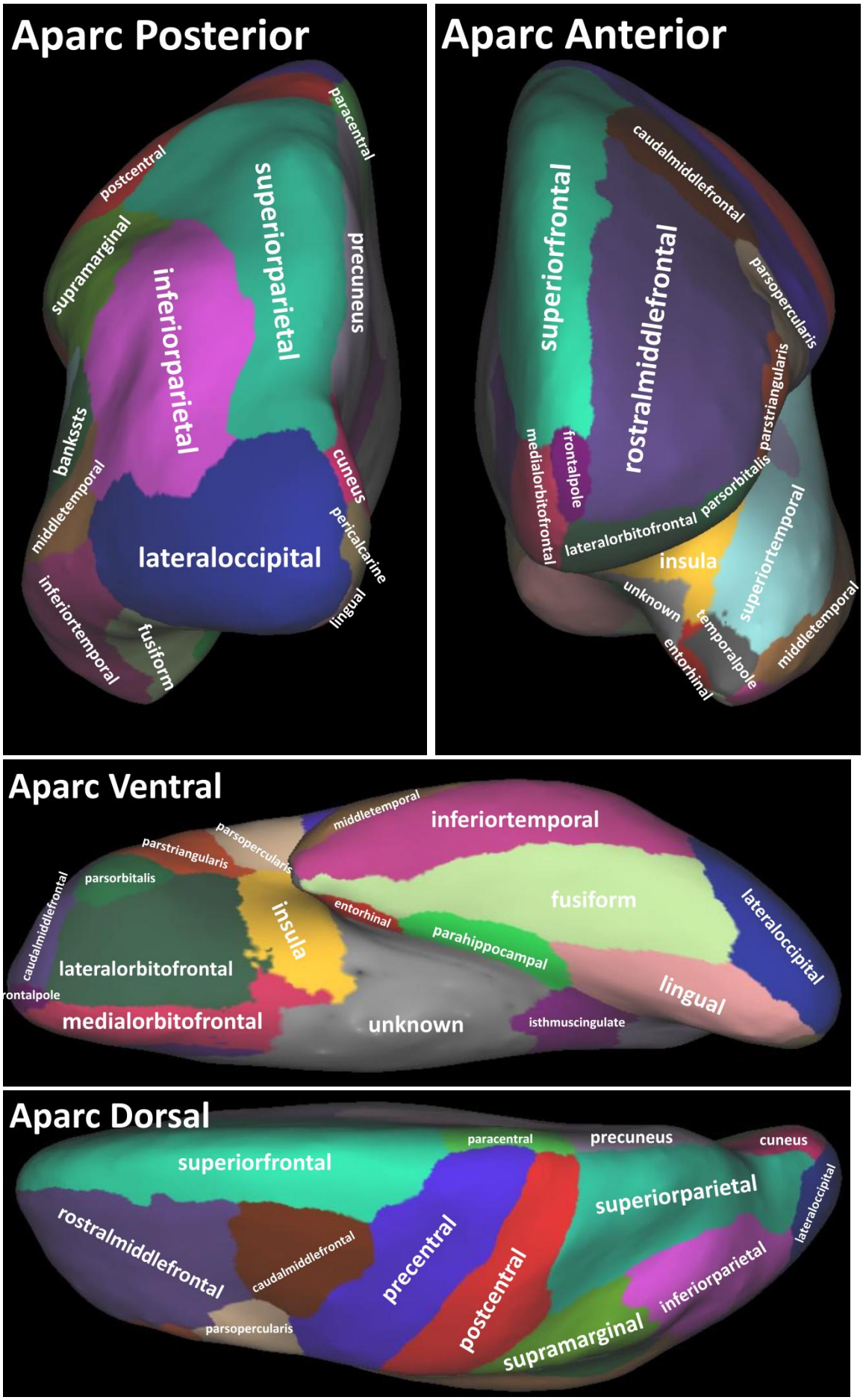
After the procedure, you can visualize the mapped image by starting e.g. FreeSurfer’s tksurfer, load the created ‘w’ file as overlay, and use a suitable color palette for display.

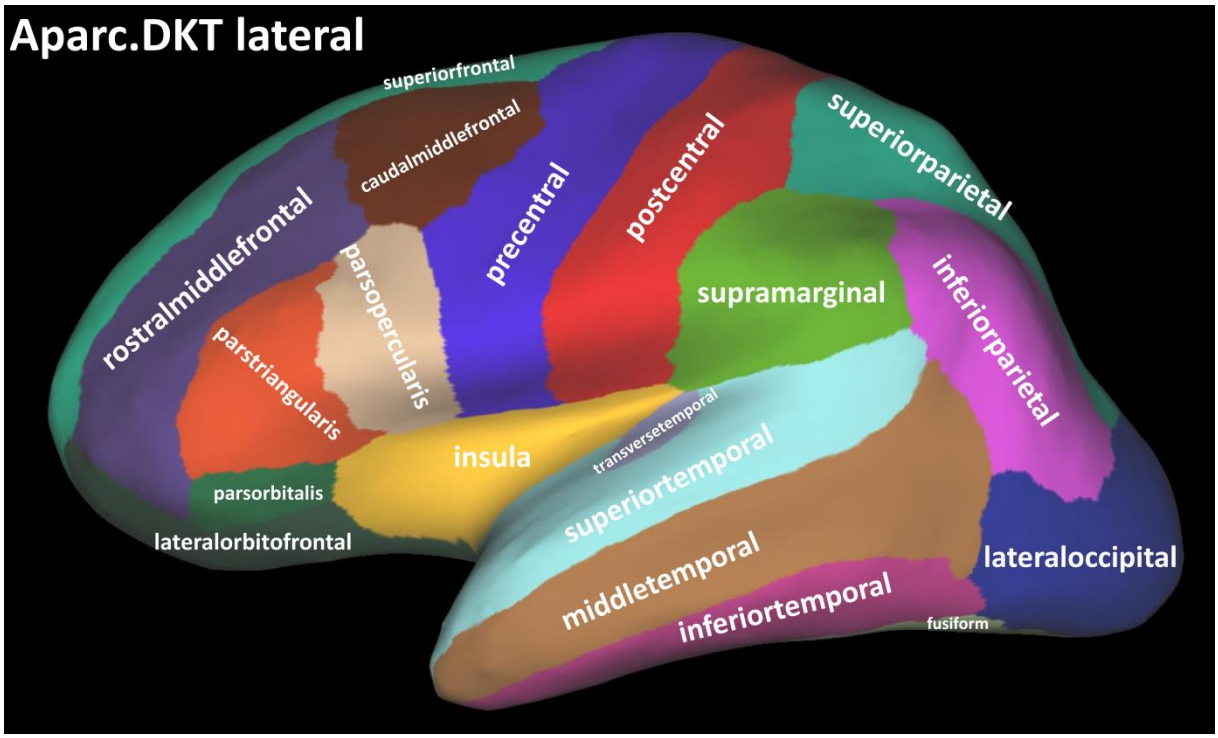


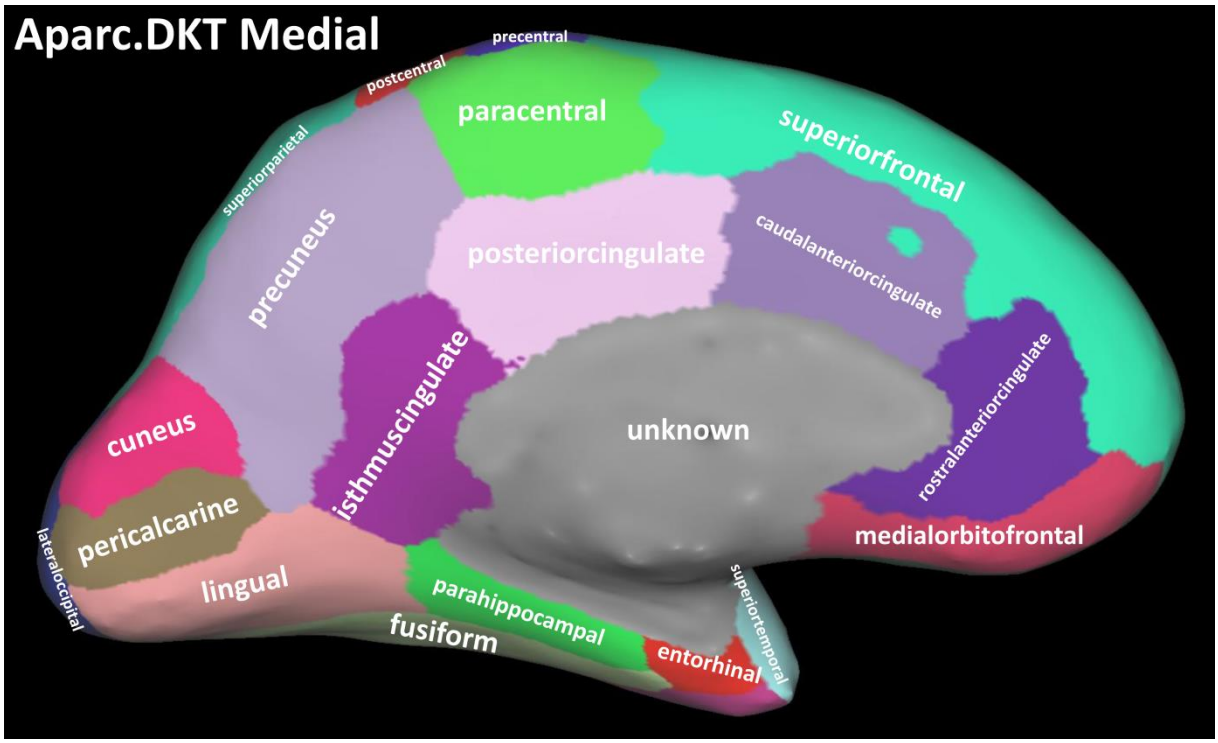
Addendum

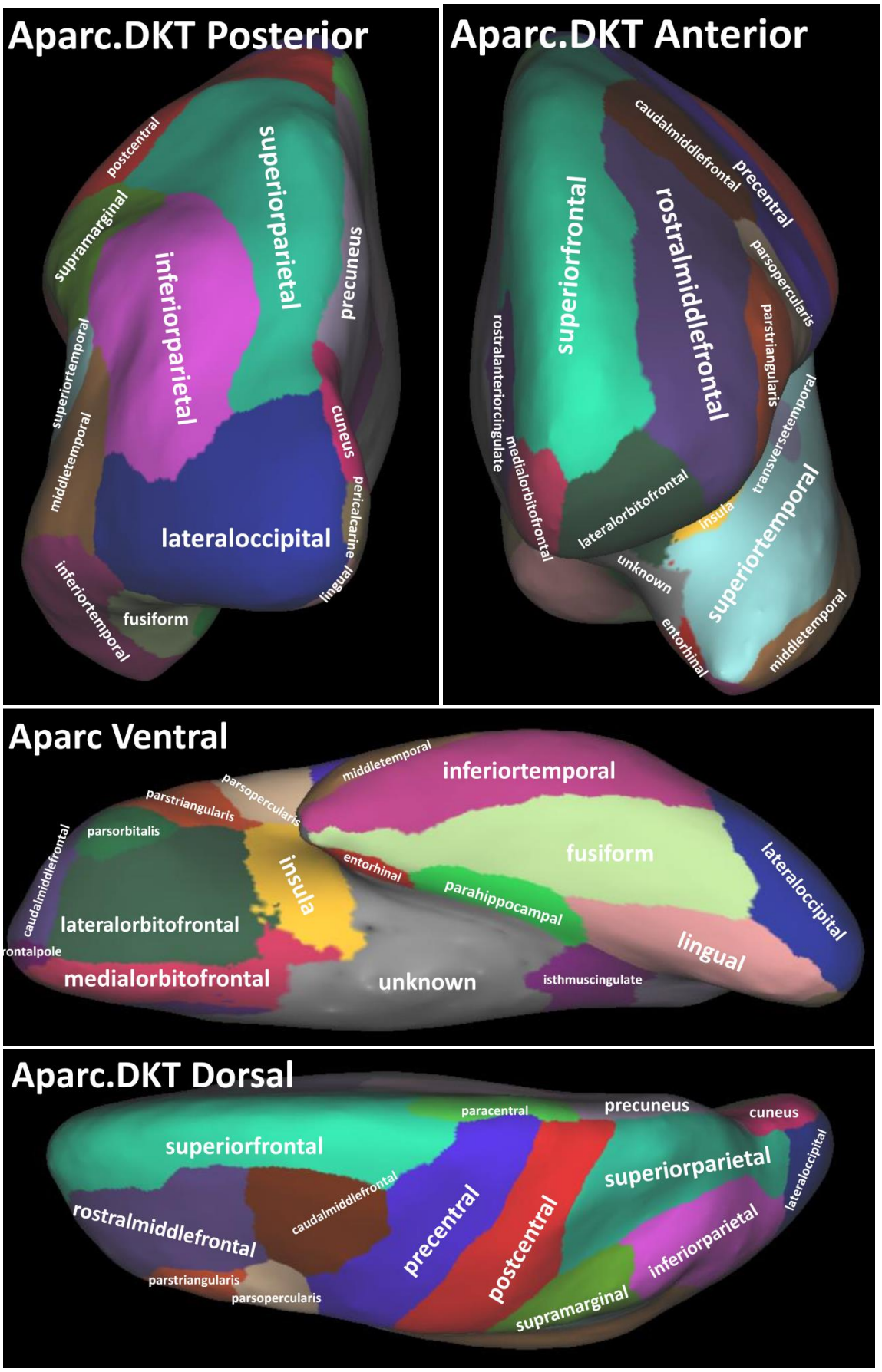

Aparc Lateral
superiorfrontal
caudalmiddlefrontal
rostralmiddlefrontal
precentral
postcentral
superiorparietal
supramarginal
inferiorparietal
parsopercularis
parstriangularis
insula
transversetemporal
bankssts
parsorbitalis
lateralorbitofrontal
superiortemporal
middletemporal
inferiortemporal
lateraloccipital
fusiform


Aparc Medial
postcentral
precentral
paracentral
superiorfrontal
superiorparietal
precuneus
posteriorcingulate
caudalanteriorcingulate
isthmuscingulate
unknown
cuneus
rostralanteriorcingulate
frontalpole
pericalcarine
medialorbitofrontal
lateraloccipital
lingual
insula
parahippocampal
fusiform
temporalpole
entorhinal


Aparc Posterior
paracentral
postcentral
supramarginal
superiorparietal
precuneus
inferiorparietal
bankssts
middletemporal
cuneus
pericalcarine
lateraloccipital
inferiortemporal
fusiform
lingual
Aparc Anterior
caudalmiddlefrontal
superiorfrontal
rostralmiddlefrontal
parsopercularis
parstriangularis
medialorbitofrontal
frontalpole
parsorbitalis
lateralorbitofrontal
insula
superiortemporal
unknown
temporalpole
middletemporal
entorhinal
Aparc Ventral
middletemporal
inferiortemporal
parsopercularis
parstriangularis
parsorbitalis
caudalmiddlefrontal
fusiform
lateraloccipital
entorhinal
insula
parahippocampal
lateralorbitofrontal
lingual
medialorbitofrontal
unknown
isthmuscingulate
Aparc Dorsal
paracentral
precuneus
cuneus
superiorfrontal
superiorparietal
lateraloccipital
rostralmiddlefrontal
caudalmiddlefrontal
precentral
postcentral
inferiorparietal
parsopercularis
supramarginal


Aparc.DKT lateral
superiorfrontal
caudalmiddlefrontal
rostralmiddlefrontal
precentral
postcentral
superiorparietal
parsopercularis
supramarginal
inferiorparietal
parstriangularis
insula
transversetemporal
superiortemporal
parsorbitalis
lateralorbitofrontal
middletemporal
lateraloccipital
inferiortemporal
fusiform


Aparc.DKT Medial
precentral
postcentral
paracentral
superiorfrontal
superiorparietal
precuneus
posteriorcingulate
caudalanteriorcingulate
isthmuscingulate
cuneus
unknown
rostralanteriorcingulate
pericalcarine
medialorbitofrontal
lateraloccipital
lingual
parahippocampal
superiortemporal
fusiform
entorhinal


Aparc.DKT Posterior
postcentral
supramarginal
superiorparietal
precuneus
inferiorparietal
superiortemporal
middletemporal
cuneus
pericalcarine
lateraloccipital
lingual
inferiortemporal
fusiform
Aparc.DKT Anterior
caudalmiddlefrontal
precentral
parsopercularis
superiorfrontal
rostralmiddlefrontal
parstriangularis
rostralanteriorcingulate
transversetemporal
medialorbitofrontal
lateralorbitofrontal
insula
unknown
superiortemporal
entorhinal
middletemporal
Aparc Ventral
middletemporal
inferiortemporal
parstriangularis
parsopercularis
parsorbitalis
caudalmiddlefrontal
insula
entorhinal
fusiform
lateraloccipital
parahippocampal
lateralorbitofrontal
rontalpole
lingual
medialorbitofrontal
unknown
isthmuscingulate
Aparc.DKT Dorsal
paracentral
precuneus
cuneus
superiorfrontal
superiorparietal
lateraloccipital
caudalmiddlefrontal
rostralmiddlefrontal
precentral
postcentral
inferiorparietal
parstriangularis
parsopercularis
supramarginal