

Arterial Spin Labeling Perfusion MRI Signal Processing Toolbox (ASLtbx)

(Mar 18 2018)

<https://cfn.upenn.edu/~zewang/ASLtbx.php>

Zhengjun Li, Sudipto Douli, Ze Wang

redhatw@gmail.com

CONTENT

1. INTRODUCTION	4
1.1 THEORY AND BACKGROUND	4
1.2 SOFTWARE AND HARDWARE REQUIREMENT	5
1.3 IMAGE FORMAT	5
1.4 LICENSE	6
2. INSTALLING ASLTBX.....	6
2.1 DOWNLOAD ASLTBX.....	6
2.2 UNZIP AND INSTALL ASLTBX	7
2.3 ADD ASLTBX PATH TO MATLAB SEARCH PATH	7
2.4 SAMPLE DATA	8
2.4.1 Data structure and parameters	8
2.4.2 Functional Stimuli.....	8
2.4.3 Batch processing using the customized scripts.....	9
3 ASL DATA PREPROCESSING USING SPM GUI	9
3.1 REORIENT THE IMAGES	9
3.1.1 Set or find the origin in SPM GUI mode	9
3.1.1.1 Reset the orientations	11
3.1.1.2 Find the Anterior Commissure	11
3.1.1.3 Reorient images	11
3.2 MOTION CORRECTION.....	11
3.3 COREGISTRATION BETWEEN ASL IMAGES AND THE STRUCTURE IMAGES	11
3.4 SMOOTHING.....	12
4 PROCESSING ASL DATA IN BATCH MODE	12
4.1 THE PIPELINE FILE AND DATA SETTING FILE.....	12
4.2 RESETTING AND SETTING ORIGINS	12
4.3 MOTION CORRECTION.....	13
4.4 REGISTRATION	13
4.5 SMOOTHING.....	13
4.6 GENERATING A BRAIN MASK FOR EXCLUDING OUT-OF-BRAIN VOXELS	13
4.7 CBF QUANTIFICATION	13
4.8 INDIVIDUAL LEVEL GENERAL LINEAR MODEL	13
4.9 GROUP LEVEL ANALYSIS	15
5 CBF QUANTIFICATION IN ASLTBX	15
5.1 GUI MODE	16
5.2 COMMAND LINE.....	18

5.5 DETERMINING ASL DATA ACQUISITION PARAMETERS25
6. OUTPUT26
7. ACKNOWLEDGEMENT.....28

1. Introduction

This is a brief tutorial to the Arterial Spin Labeled Perfusion MRI data processing toolbox (ASLtbx), a MATLAB (Mathworks Inc.) and SPM (Wellcome Department, UCL) -based toolkit for processing ASL data acquired with either the pulsed ASL (PASL) or continuous ASL (CASL) or the pseudo-CASL technique¹. The toolbox is free for academic users, and can be obtained from <https://cfn.upenn.edu/~zewang/> under the GPL license. The original GPL license and the file header should be included in any modified versions. Example datasets for resting ASL and functional ASL with customized settings are also available through the website. Both the toolbox and the sample data are not allowed for any commercial use without formal permission from the University of Pennsylvania. We (the author and University of Pennsylvania, the same for the following) are not and will not be responsible for any use which is made of this package. By providing exemplar data, no references or gold standards for the images, blood measures, and any kind of resultant outcomes are implied. We further disclaim any liability and accuracy of the outcomes arising from using this package. We are not responsible for any data interpretations. All the code and the data are provided **as they were**.

Some basic concepts about fMRI and ASL MRI are assumed from ASLtbx users before running it. Basic Matlab script programming skills will be required too. **Please cite the toolbox and the related papers for any use of the toolkit.**

For the users who are only interested in cerebral blood flow (CBF) quantification, section 3 and 4 can be skipped.

1.1 Theory and background

Arterial spin labeled (ASL) perfusion MRI is a noninvasive technique for quantitating cerebral blood flow (CBF) using magnetically labeled arterial spins as an endogenous tracer [1, 2]. The methodological scheme of ASL is analogous to that used in PET or SPECT scanning [3]. Depending on how the blood water is labeled, ASL techniques can be roughly divided into pulsed ASL (PASL) [4-7] or continuous ASL (CASL) [1, 2, 8-10]. PASL provides lower SNR than CASL as well as increased physiological noise [11], but it is easier to implement. The state-of-art ASL labeling technique is the pseudo-CASL (pCASL) [10]. In ASL, perfusion signal is typically determined by pair-wise comparison with separate images acquired with control labeling.

ASLtbx is a collection of Matlab scripts. The premier version of the pipeline and the toolkit were first described in the following paper:

- 1) *Ze Wang, Geoffrey Aguirre, Hengyi Rao, JiongJiong Wang, Anna R. Childress, John A. Detre,*

¹ All references can be easily found in pubmed and in the code, though a list will be provided in later version.

Empirical ASL data analysis using an ASL data processing toolbox: ASLtbx, Magnetic Resonance Imaging, 2008, 26(2):261-9.

and further enhanced in:

- 2) *William T. Hu*, Ze Wang*, Virginia M.-Y. Lee, John Q. Trojanowski, John Detre, Murray Grossman, Distinct Cerebral Perfusion Patterns in FTLD and AD, Neurology, 2010 Sep 7;75(10):881-8. (contributed equally).*
- 3) *Ze Wang, Improving Cerebral Blood Flow Quantification for Arterial Spin Labeled Perfusion MRI by Removing Residual Motion Artifacts and Global Signal Fluctuations Magnetic Resonance Imaging, Magnetic Resonance Imaging 30(10):1409-15, 2012.*

A recent abstract might be interested for the users regarding various quantification approaches:
Yufen Chen, Ze Wang, John A. Detre, Impact of equilibrium magnetization of blood on ASL quantification, ISMRM 2011 annual meeting, pg. 300.

ASLtbx is partly based on SPM, a Matlab software package for brain imaging visualization and analysis that can be freely downloaded from the University College London (<http://www.fil.ion.ucl.ac.uk/spm/>). The current version of cerebral blood flow (CBF) quantification function: `asl_perf_subtract` uses several input/output functions provided by SPM, though we are implementing an independent widget in C++.

1.2 Software and Hardware Requirement

ASLtbx runs under core MATLAB (The MathWork, Inc., Natick, MA), version 5.3 or higher. Since it needs a few functions from SPM, SPM (SPM8 and SPM12 are preferred, though SPM5 can be still supported. We won't further support SPM99 and SPM2 anymore since 2013) should be installed and added to MATLAB search path. Please refer to SPM's website to get more information. Depending on the dataset size, greater than 256 MB memory might be required. Otherwise, there are no additional specific hardware requirements for running ASLtbx.

1.3 Image format

Both 3D and 4D NIFTI format and 3D Analyze images are acceptable for current version. Nifti format is recommended now.

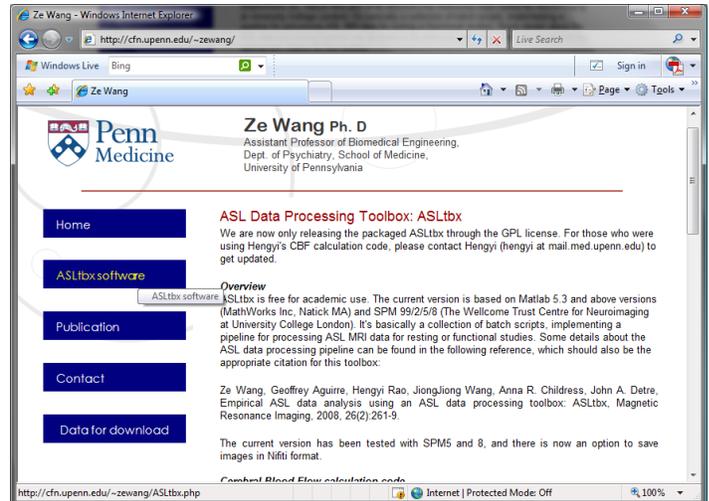


Figure 2.1. A snapshot of the download webpage.

1.4 License

The University of Pennsylvania reserves all rights regarding ASLtbx. ASLtbx is free for academic users. You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Commercial or industrial users should contact the University and the author for the use of ASLtbx.

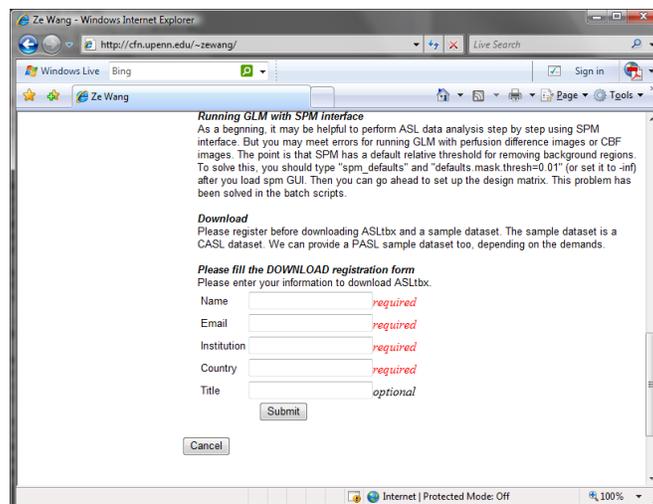
ASLtbx is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with ASLtbx. If not, see <<http://www.gnu.org/licenses/>>.

2. Installing ASLtbx

2.1 Download ASLtbx

ASLtbx can be downloaded from the author's website (<https://cfn.upenn.edu/~zewang/>). It's available as a zipped file named "ASLtbx.zip". The example datasets can be downloaded through the same web site following the steps as illustrated below. It is highly recommended to try one of the few sample datasets with customized scripts and modify the scripts accordingly to incorporate your own data. Please spare a minute to type the required information in the form. All recorded information will be kept confidential. We will only automatically pull out your emails and distribute messages when big updates are made. **Please don't fill non-words or nonsense in the registration boxes.**



The screenshot shows a web browser window titled "Ze Wang - Windows Internet Explorer" with the address bar displaying "http://cfn.upenn.edu/~zewang/". The page content includes a section titled "Running GLM with SPM interface" with explanatory text. Below this is a "Download" section with instructions to register. The main part of the page is a registration form titled "Please fill the DOWNLOAD registration form" with the instruction "Please enter your information to download ASLtbx." The form contains five input fields: "Name" (required), "Email" (required), "Institution" (required), "Country" (required), and "Title" (optional). A "Submit" button is located below the "Title" field, and a "Cancel" button is located below the "Submit" button. The browser's status bar at the bottom indicates "Internet | Protected Mode: Off" and "100%".

Figure 2.2. The ASLtbx registration webpage.

Thank you for your registration!

Please click on the links below to download ASL toolbox, manual, or example datasets.

ASL toolbox for SPM5 and 8:

[ASL toolbox](#)

ASL toolbox for SPM12:

[ASLtbx2](#)
[ASL toolbox Manual](#)

Sensorimotor CASL fMRI data with scripts based on SPM5/8:

[CASL fMRI data](#)

Sensorimotor CASL fMRI data with scripts based on SPM12:

[CASL fMRI data with ASLtbx2](#)

Resting ASL datasets with customized scripts:

[CASL example data with customized ASLtbx for SPM5/8](#)
[PCASL example data with customized ASLtbx scripts \(based on SPM12\)](#)
[Pulsed ASL\(PASL\) example data with customized batch scripts\(51MB, SPM5/8\)](#)
[PASL example data acquired using Siemens product 2D PASL sequence. Scripts are based on ASLtbx with SPM12.](#)

Resting PASL data with scripts based on ASLtbx2 and latest outlier cleaning:

[PASL example data with ASLtbx2 and latest outlier cleaning](#)

Figure 2.3. The download webpage with links to the scripts and sample data.

2.2 Unzip and install ASLtbx

Copy the zip file into a directory, for example, "Z:\ASL". Decompress the zip file and move all the .m files to a directory, for example, "Z:\ASL\ASLtbx".

2.3 Add ASLtbx path to Matlab search path

Suppose the toolkit is installed in Z:\ASL\ASLtbx, in Matab command window, type:

[addpath Z:\ASL\ASLtbx](#)

to add the ASLtbx path to Matlab search path.

2.4 Sample data

We have now provided 3 datasets: one for fMRI using CASL, another for resting PASL, and the 3rd for resting pCASL. Each dataset has preset scripts showing the acquisition parameters and settings for processing.

The following instructions are for the CASL fMRI data set. Similar procedures can be used for analyzing brain state ASL data such as long video cue condition vs neutral condition as well. Fewer pre-processing steps are required for processing resting ASL data. Please check the resting data and the associated scripts for the details.

2.4.1 Data structure and parameters

The sample data are downloadable from the same webpage as that for ASLtbx scripts. To start, you can unzip the data into a new directory (say, "Z:\ASL\ASL_Example_data ") and copy the decompressed content into the new work directory. The example CASL fMRI data file contains four sub-folders, "batch_scripts", "sub1" , "sub2" and "sub3" . The first one contains customized batch scripts for this particular example data, the other three folders contain the three subjects' example data. Each subject's folder has a folder named "anat_anlz" holding the structural image and a folder named "func_anlz" holding the functional images.

The structure image was acquired using a 3D MPRAGE sequence with scan parameters as: FOV=250 mm, TR/TE=1620/3 ms, 192x256 matrix, 160 slices with thickness of 1 mm. Functional images were acquired using an amplitude modulated continuous ASL (CASL) perfusion imaging sequence optimized for 3.0 T (Wang 2005, Radiology). Acquisition parameters were: FOV=22 cm, 64x64x12 matrix, Bandwidth=3 kHz/pixel, Flip angle=90°, TR=3 s, TE=17 ms, Slice thickness=6 mm, Inter-slice space=1.5 mm, Labeling time=1.6 s, Post label delay time=800 ms. Seventy-two label/control image pairs were acquired for each subject.

2.4.2 Functional Stimuli

The functional experiment was performed following a block design with two interleaved conditions. During the "on" condition, visual stimuli with an 8-Hz reversing black and white checkerboard were presented periodically with duration of 72 s. Subjects were also instructed to perform a self-paced, right-hand-only, finger-tapping task during visual stimuli. The "off" condition consisted only of a blank screen with a crosshair in the middle of the screen.

2.4.3 Batch processing using the customized scripts

To run the data analysis job, you can simply start Matlab, and switch to the subfolder “batch_scripts”, and type “batch_run” in Matlab command line and hit enter. Please see section 4 for more details.

Note: for your own data, it is better to organize them in a similar way or at least using a consistent structure when preparing the imaging data. Or you have to change the scripts accordingly.

3 ASL data preprocessing using SPM GUI

ASLtbx can process images that are in the same orientation and have the same voxel size. We recommend to use the following steps to preprocess ASL data before doing CBF quantification.

3.1 Reorient the images

(This step is obsolete for SPM8 and above versions. I have noticed that registrations using the latest SPM8 after resetting origins to be the center but without manually setting origins to be the AC-PC line are pretty well. So you might be able to avoid this tedious part if you have updated your SPM8. However, you should still reset the origins. FSL is totally free of this step.)

In SPM, the image origin is set to be the AC-PC line. Though SPM8 might not require this, setting origins are required in ASL data processing as we noticed remarkable difference when the origins were set to be the center (we noticed this difference in spm5). Since the same dataset might be processed ten or more times before publication, it is convenient to put down the coordinates of the AC-PC line in a m-file so the whole dataset can be re-processed anytime and the same results can be obtained if no changes would be made to the processing pipeline. The following steps should be performed for different type of images (such as structural images, functional images, and DTI images etc).

3.1.1 Set or find the origin in SPM GUI mode

Click “Display” button and open the structure image or the first image of the functional series.

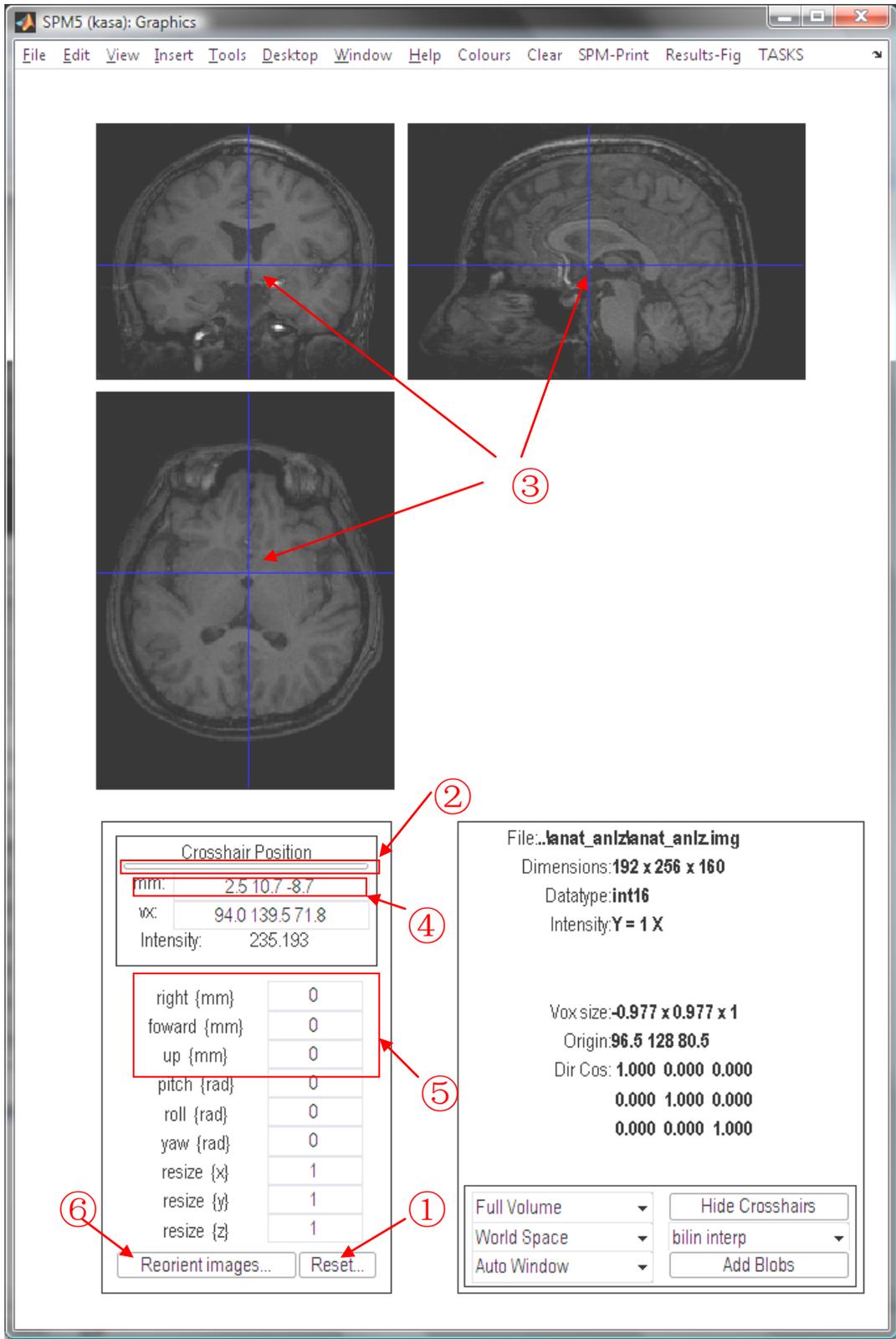


Figure 3.1. Illustration for how to find and set up image orientation using SPM GUI.

3.1.1.1 Reset the orientations

Click “Reset..” button (as shown as ① in figure 3.1) and select all the images to be processed including the structure images and the functional images. This would retain the current voxel sizes and sets the origins of the images to the center of the volumes and set all the rotations back to zero.

3.1.1.2 Find the Anterior Commissure

Display one of the origin reset images. Click the horizontal bar (② in figure 3.1) to check if the origin, which is indicated by the crosshair in the three views, is in the center of each view. If not, please do step 3.1.1.1 again.

Click around the three views until the crosshair is at the Anterior Commissure(③ in figure 3.1).

Illustrations for finding the Anterior Commissure can be also got from the following webpage: <http://imaging.mrc-cbu.cam.ac.uk/imaging/FindingCommissures>.

The offset between the center of the volume (the current origin after “reset”) and the Anterior Commissure (the location of the cursor) will show in the box “mm” (④ in figure 3.1).

3.1.1.3 Reorient images

Once you have moved the cursor to the desired “origin” place, you will notice the offset between the current cursor position to current “origin” of the displayed MR image (which is the center of the volume in this example). Now, you can reorient the images by entering the **NEGATIVE**s of the x, y, z coordinates (shown in the “mm” box) to the boxes named “right”, “forward”, “up” (⑤ in figure 3.1), respectively.

Then click the “Reorient images...” button (⑥ in figure 3.1) and select the images to which the new origins should be applied to. This will change the affine transformation matrix in the image header in SPM > 5.

3.2 Motion correction

Motion correction for ASL MRI data is slightly different from that used for BOLD fMRI. ASL MRI should either be performed for the control and label image series separately [12] or using the amended motion correction procedure, which is included in `batch_realign.m` by default [13]. The adapted ASL specific motion correction method can be found in `spm_realign_asl.m`. Specifically, the rigid body transform was used to estimate the motion timecourses for all ASL control and label images as a first step. The spin labeling and control labeling time paradigm (the zigzag pattern) is then regressed out from the motion timecourses. The residuals are used as the final motion parameters for correcting motions from the acquired label and control images.

3.3 Coregistration between ASL images and the structure images.

ASL images should be coregistered to anatomical images so they can be later normalized to the MNI space (or local template space or any other standard space) for group analysis. The target image and

source image should be selected as the T1 image and the mean ASL raw image, respectively. T2 weight structure images can be used as well.

3.4 Smoothing

The raw ASL images should be smoothed before CBF calculation to prevent noise propagation. **A second smoothing can be applied after CBF calculation and spatial normalization if there are large inter-brain structure variations noticed after spatial normalization.** Users can use SPM GUI to do spatial smoothing or use the `batch_smooth.m` script in this package.

4 Processing ASL data in batch mode

We strongly recommend using scripts to process ASL data though the same procedures can be implemented using the SPM GUI. Users can try the sample data and the pre-set scripts to get a feeling for using and adapting scripts for the whole pipeline. For each dataset, users should be able to run through all the steps listed in “`batch_run.m`”.

4.1 The pipeline file and data setting file

The file “`batch_run.m`” defines a sequential ASL data processing flow, which generally consists of motion correction, coregistration, smoothing, CBF calculation, normalization, GLM analysis, group analysis, etc. Different projects (like the resting study) may not need all of the steps. All data parameters, data path, settings for analysis are saved in a global structure specified in “`par.m`” (par means parameter).

The parameter setting file can be verified by typing “par” in Matlab console and each field of the global structure: PAR can be checked by typing the name accordingly (for example, typing “`PAR.subjects{:}`” should list all the subjects involved in the current dataset).

4.2 Resetting and setting origins

Three files are involved here. The file: “`batch_reset_orientation`” will reset the origin to center of each volume and set all the rotations back to zero. This will modify the header file of each image directly; “`batch_generateorigintable`” is to create an origin table to store the relative coordinators of the AC-PC line to the center of the volume. After this step, you will need to open the “`origintable.m`” to fill the coordinators you find after origin reset (see Fig. 3.1 where you should put down the coordinators 2.5 10.7 -8.7 in the origin table as `OPAR.subs(x).t1org = [2.5 10.7 -8.7]`). Then “`batch_setorigin`” will use the offsets recorded in `origintable.m` to find the correct origins. It will directly change the header file of each image. (Note the reset orientation script has been updated. Old version might have problem for sagittal acquired images)

Do not run “`batch_setorigin`” more than once unless you have run `batch_reset_orientation` in advance. **For many users, these 3 steps (reset, generate origin table, and fill in the origin coordinators) are a little bit confusing. However, you have to do this unless we found easier solutions.** Please try these 3 steps

with the sample data several times and display the images to get a feeling of these steps.

4.3 Motion correction

Depending on the setting, “batch_realign.m” will realign ASL images to the reference volume. The reference volume can be set to be the first image or the mean image (by default). Please read the code comments for how to change the setting. After reslicing, new images with “r” in the beginning of the filename and a mean image (starts with “mean”) will be generated.

4.4 Registration

“batch_coreg.m” coregisters the realigned ASL images to each subject’s structure image. It will apply the transformation by modifying the header file of each functional image directly.

4.5 Smoothing

“batch_smooth.m” uses the SPM Gaussian smoothing kernel to smooth the realigned and coregistered functional images to reduce noise. New images with prefix “sr” will be created.

4.6 Generating a brain mask for excluding out-of-brain voxels

“batch_create_mask.m” creates a mask based on the mean of the functional images which is generated in step 4.5. The mask image will be named by “mask_perf_cbf.img” and saved in each subject’s functional folder. This mask is used to exclude extra-cranial voxels. Users can define their own mask.

4.7 CBF quantification

“batch_perf_subtract.m” calls “asl_perf_subtract” to calculate CBF (see section 5 for more details about the latter function). Depending the options chosen, CBF image series, delta perfusion signal image series, mean CBF images (“meanCBF_*.img”) will be generated in each subject’s directory. Below is a mean CBF image of sub2 in the example dataset.

4.8 Individual level general linear model

This step is to assess the individual level effects, which can be then taken into the group level for a group level analysis, mimicing the two-stage random effect analysis model designed for BOLD fMRI analysis. For brain state analysis such as before and after taking medicine/treatment or long duration video condition vs neutral video condition, users

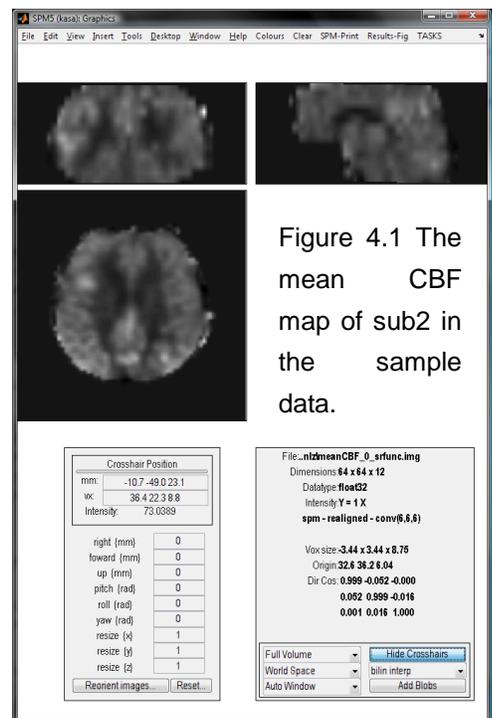


Figure 4.1 The mean CBF map of sub2 in the sample data.

can use this two-stage model as well. Or you can alternatively take the mean CBF maps into the group level ANOVA model, just like analysis in PET imaging. Theoretically, these two models are equivalent, but the former one allows the users to assess each individual's response with statistical inference, which might be important for future experiment plan. But the cons are you have to go back to the data in order to extract the CBF difference in the significant areas.

“**batch_model.m**” runs the General Linear Model (GLM) on the subtracted images to get the tap-rest effect in the brain perfusion. Some people may want to use the pure boxcar function as the reference in GLM. In the toolbox, this can be done by turning on line 73 “SPM.xBF.name='Fourier set' and commenting out line 74 “SPM.xBF.name='hrf”.

“**batch_contrasts.m**” will produce the contrast maps between the two contrasting conditions. The result images and SPM.mat will be saved in a directory called “glm_cbf” (specified in par.m) which is located in each subject's folder. Below is a snapshot of the GLM analysis results for sub2.

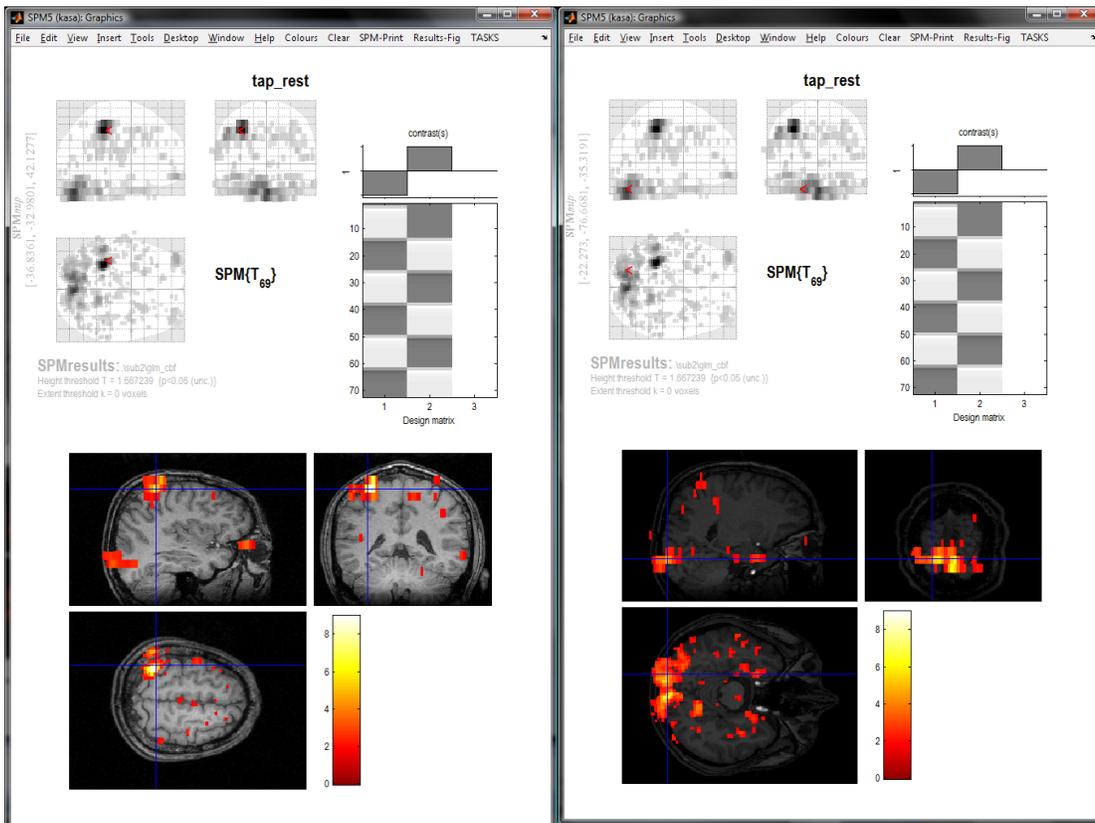


Figure 4.2 GLM analysis results of one representative subject (sub2 of the sample data). Task activations are expected to be in left motor cortex (as shown in the left image) and visual cortex (as shown in right the image) as a result of performing the visual and right hand finger tapping task.

4.9 Group level analysis

Three or 4 steps are involved.

“`batch_2nd_cp_confiles.m`” copies each subject’s result map (named as “con*.img”) to a group directory called “group_anasmallPerf_sinc\tap_rest” under the “ASL_Example_data” directory.

“`batch_usegment_spm5.m`” uses SPM’s unified segmentation method to normalize the subjects’ result maps to standard MNI space for group analysis. The images’ names are prefixed with “w” after normalization.

“`batch_smooth_wconfiles.m`” smoothes the normalized images to reduce the variability induced by inter-brain structure differences. Images like “swcon*.img” will be generated.

“`batch_2nd_glm.m`” does the 2nd level group analysis on the normalized and smoothed images to get the group effect of the experiment design. Result will be saved in a directory called “group_anasmallPerf_sinc\tap_rest\RFX” under the “ASL_Example_data” directory. Below is a snapshot of the group level results for the CASL finger tapping data.

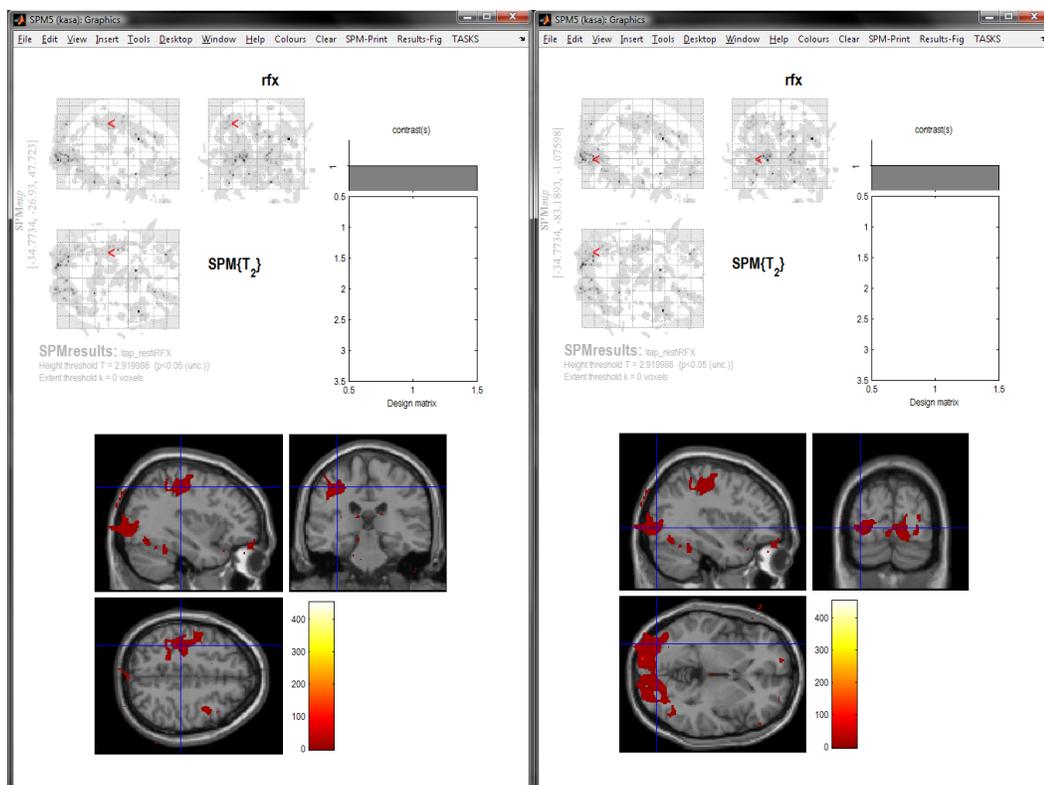


Figure 4.3 Group level results for the CASL fingertapping data.

5 CBF quantification in ASLtbx

CBF quantification in ASLtbx can be performed either in GUI mode or batch mode. The quantification

model is described in [asl_perf_subtract.m](#). Differences might be seen in the equations as compared to those in the cited papers due to the different units used or minor adaptations made. Please refer the documentation in the code, and the papers cited in the code. I'm happy to discuss this in email if users are interested. But I assume the questioners read through the equations and the cited papers before raising questions to me. Otherwise, it is just easy for the questioners but not for me unfortunately since I have all kinds of ASLtbx emails every day.

Before started, make sure SPM and ASLtbx are included into the Matlab searching path, please refer to section 1.2 and 2.3 for more information about this.

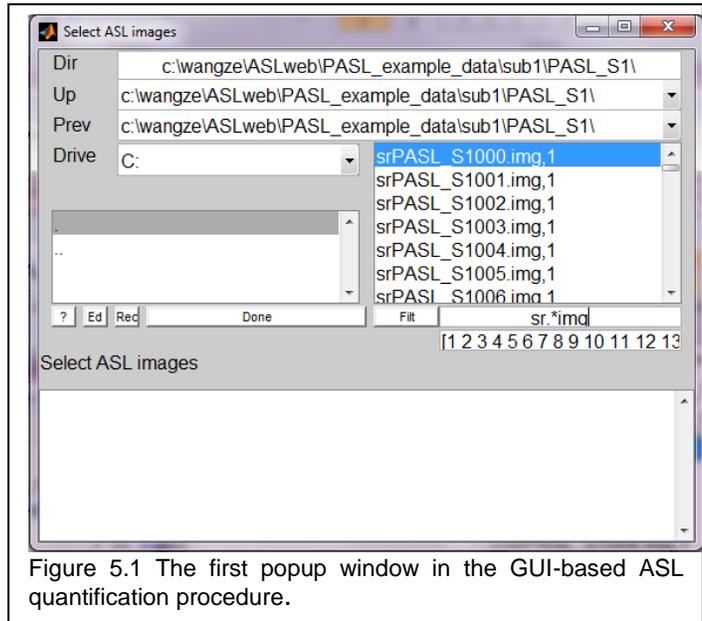


Figure 5.1 The first popup window in the GUI-based ASL quantification procedure.

5.1 GUI mode

In Matlab command window, simply type:

[asl_perf_subtract](#)

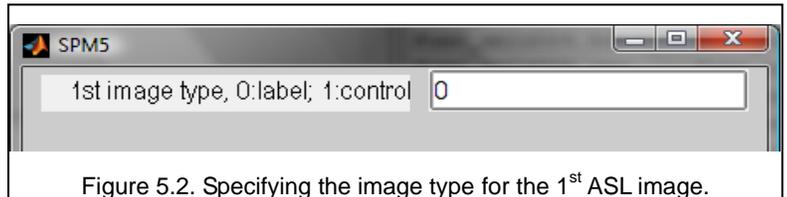


Figure 5.2. Specifying the image type for the 1st ASL image.

and hit the enter key. You will see a popup Window (Fig. 5.1) asking for selecting ASL images. The images should be preprocessed and have the same orientation and voxel size. The latest version now allow you to select 4D Nifti images.

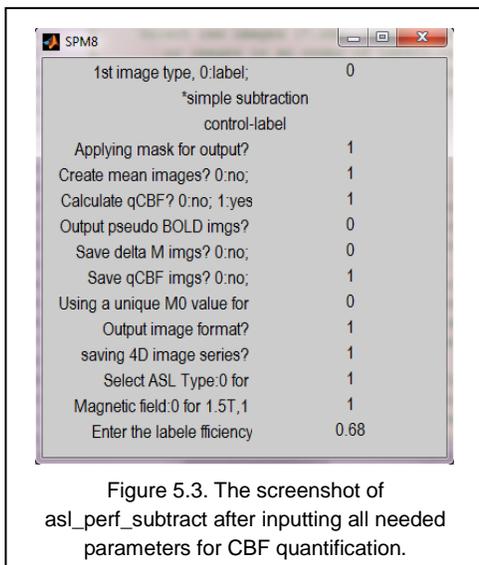


Figure 5.3. The screenshot of `asl_perf_subtract` after inputting all needed parameters for CBF quantification.

After you select the sources images, press the **“Done”** button if Fig. 5.1 you will get another pop-up window (Fig. 5.2), which will take ASL imaging acquisition parameters and quantification options that are needed for CBF calculation.

To use a unique M0b (M0 blood) for CBF quantification,

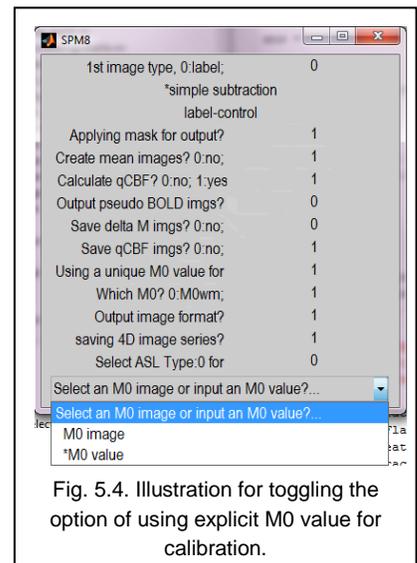


Fig. 5.4. Illustration for toggling the option of using explicit M0 value for calibration.

you should choose 1 for the question “Using a unique M0 value for all voxels?”. Then later on, you will see another questionnaire asking whether you will directly provide an M0i (M0wm or M0csf) value (see the bottom part in Fig. 5.4) or let the script to get the M0i from an M0 image.

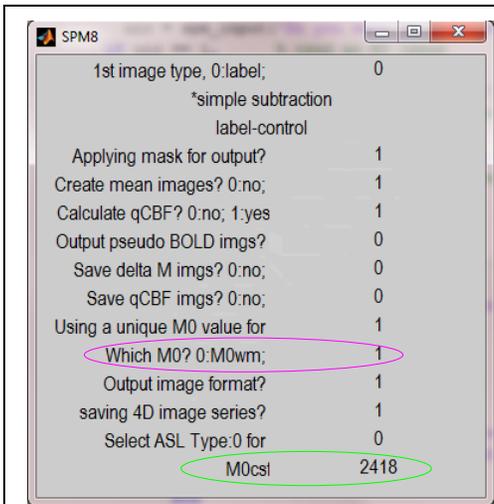


Fig. 5.5. Option for choosing the M0csf-based calibration.

If you have explicitly extracted M0i (M0wm or M0csf) and you want to use it for CBF quantification in `asl_perf_subtract`, then you need to type 1 in the question “Using a unique M0 value for all voxels?” (see the purple ellipse in Fig. 5.5) and then select “M0 value” in “Select an M0 image or input an M0 value?” (see Fig. 5.4), then you will get an input window for you to type in the M0i value you have (see the green ellipse in Fig. 5.5).

When you select using a unique M0b for quantification and select “M0 image” in the questionnaire “Select an M0 image or input an M0 value”, you will see another option (see the bottom part of Fig. 5.6) to choose whether you will draw an ROI in the

M0 image you selected to extract the M0i (M0csf or M0wm, depending on the option you choose in the purple ellipse in Fig. 5.5) value or select an ROI mask image. If you choose “yes” for that item, you will see a popup window showing one slice of the selected M0 image (Fig. 5.7) and in Matlab command line, you can see instructions for choosing the image slice to be shown. The slice number has to be within 1 – maximum slice number of the M0 image. If you found a good slice easy for drawing the ROI, you can type 0 in Matlab command line to finish the slice selection process. Then you can start to draw an ROI in the popup window. For example, in Fig. 5.7, the ROI should be drawn in the ventricles. Double click to finish ROI drawing.

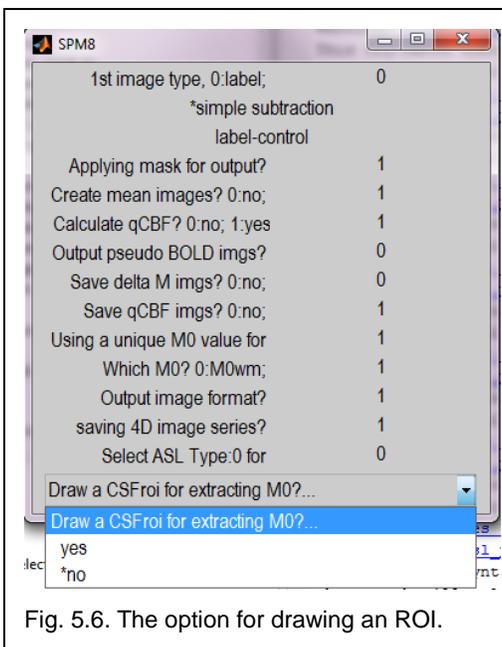


Fig. 5.6. The option for drawing an ROI.

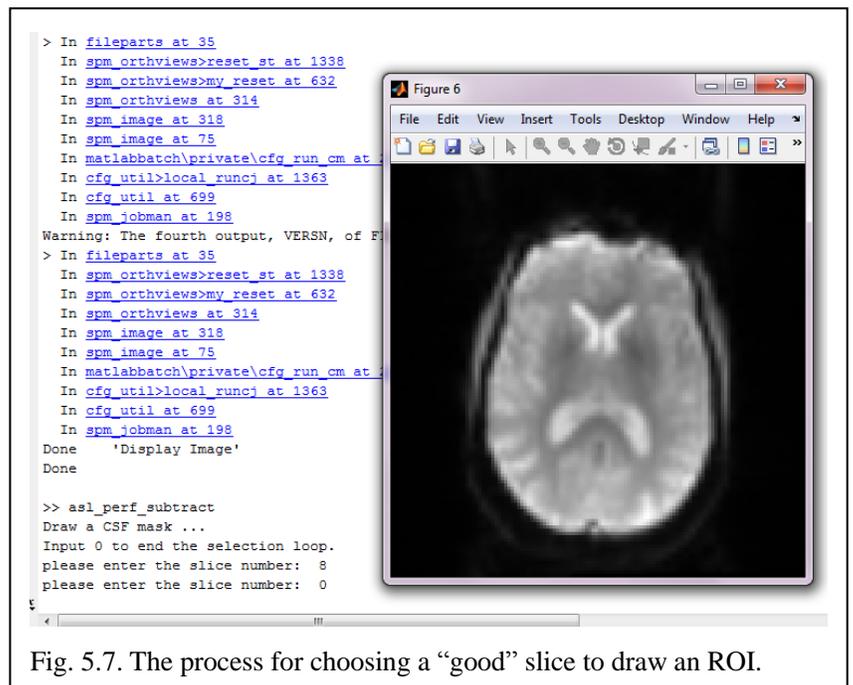


Fig. 5.7. The process for choosing a “good” slice to draw an ROI.

Once you have all those parameters set up, `asl_perf_subtract` will start to calculate all the designated outputs like the perfusion difference signal, CBF, pseudo BOLD, mean perfusion difference, mean BOLD, mean CBF etc. All those outputs will be saved in the same folder as the source images. You will also get a text file saving the global perfusion difference and CBF signal (see next section for more details).

5.2 CBF quantification using `asl_perf_subtract` in Matlab command line

The main function `asl_perf_subtract` can be also called in Matlab command line without involving any GUI if you know how to use all the arguments. The function prototype is defined as:

```
[perfnm, glcbf] = asl_perf_subtract(Filename, FirstimageType, SubtractionType, SubtractionOrder, Flag, Timeshift, AslType, labeff, MagType, Labeltime, Delaytime, Slicetime, TE, M0img, M0seg, maskimg, M0csf, M0wm)
```

The following instructions can be also found in the beginning of the file `asl_perf_subtract.m` or simply by typing “`help asl_perf_subtract`” in Matlab.

Arguments:

Filename: a list of input image series. Both 3D and 4D NIfTI format and 3D Analyze images are supported now. This list can be achieved by using SPM’s file selection function. For example, the following code will get the list of all the `.img` files names with `srfunc_anlz*` and pass it to the variable “Filename”:

```
Filename=spm_select('ExtFPList','Z:\ASL\ASL_Example_data\sub1\func_anlz', ['^sr'  
'func_anlz' '*.img$']);
```

FirstimageType – an integer variable indicates the type of the first image.

- 0:label; 1:control; for the sequence (PASL and CASL) distributed by the Center for functional neuroimaging at the University of Pennsylvania, the first image is set to be label.

SubtractionType – an integer indicates which subtraction method will be used

-0: simple subtraction; 1: surround subtraction; 2: sinc subtraction.

For control-label: if the raw images are: (C1, L1, C2, L2, C3...), the simple subtraction are: (C1-L1, C2-L2...) the surround subtraction is: ((C1+C2)/2-L1, (C2+C3)/2-L2,...), the sinc subtraction is: (C1.5-L1, C2.5-L2...) if the raw images are: (L1, C1, L2, C2...), the simple subtraction are: (C1-L1, C2-L2...) the surround subtraction will be: (C1-(L1+L2)/2, C2-(L2+L3)/2,...), the sinc subtraction will be: (C1-L1.5, C2-L2.5...) and vice versa for using the label-control subtraction order.

SubtractionOrder – an integer indicates the subtraction orientation: 1: control-label; 0: label-control

Note: a gold stand to get the correct subtraction order is to check the CBF value in grey matter. If most grey matter voxels have negative CBF values, you should switch to the opposite subtraction order. Usually, for CASL, you should choose control-label, and for the FAIR based PASL data, you should select label – control. When background suppression is applied, the subtraction order may need to be flipped as well.

Flag - flag vector composed of [MaskFlag, MeanFlag, CBFFlag, BOLDFlag, OutPerfFlag, OutCBFFlag, QuantFlag]

MaskFlag – an integer variable indicates whether perfusion images will be masked or not. Masking

is recommended to remove the background noise and those non-perfusion regions. - 0: no mask; 1: mask

MeanFlag – an integer variable indicates whether various mean images will be created - 0: not save mean image; 1: produce mean image

CBFFlag - 1: will calculate CBF value, 0: no CBF calculation

BOLDFlag – 1 or 0 indicate extracting the pseudo BOLD images or not.

OutPerfFlag – save the perfusion difference images? 1 yes, 0:no.

OutCBFFlag: write CBF signal to disk or not?

QuantFlag: using a unique M0 value for the whole brain? 1:yes, 0:no. To understand this better, you can read our ISMRM 2011 abstract: [Chen et al., Impact of equilibrium magnetization of blood on ASL quantification, Proc of ISMRM 2011 #300](#). The paper is now in submission.

ImgFormatFlag: saving images in Analyze format (0) or Nifti format (1).

4DFlag: saving output images in 3D or 4D Nifti format (1)?

M0wmcsfFlag: when QuantFlag is on (1), you can choose either using M0csf (M0wmcsfFlag=1) or M0wm (M0wmcsfFlag=0) to estimate M0b and subsequently do CBF quantification.

Labeff - labeling efficiency, 0.95 for PASL, 0.68 for CASL, 0.85 for pCASL, this can be determined by simulations or in-vivo measurement.

MagType - indicator for magnet field strength, 1 for 3T, 0 for 1.5T.

Timeshift - a value between 0 and 1 to shift the labeled image forward or backward; only valid for sinc interpolation.

AslType - 0 means PASL, 1 means CASL or pCASL

Labeltime - time for labeling arterial spins (sec).

Delaytime - delay time for labeled spins to enter the imaging slice, this argument should be set to be T12 (or T1) in QUIPSS2. The actually post-labeling delay is usually shorter than that in CASL/PCASL.

Slicetime - time for getting one slice, which can calculated through dividing (minimum TR – labeling time – post labeling delay) by the # of slices in 2D. The minimum TR can be obtained by blocking out the spin labeling section and set the post label delay to 0 in the sequence. The slice time contains excitation time (including the total slice selective gradient time, refocusing gradient time), fat or other saturation time, slice data acquisition time (#phase encoding lines/bandwidthperpixel, #number of phase encoding lines usually is the same as the image dimension along y since phase encoding is generally applied along y direction in 2D imaging. For 64x64 imaging matrix, it is 64.). **More details have been added for determining the slice acquisition time.**

M0img - M0 image acquired with short TE and long TR.

M0seg - segmented white matter M0 image needed for automatically calculating the PASL cbf images.

maskimg - a predefined mask image, for background suppression data, please specify a mask or change the corresponding threshold in the code.

Outputs:

perfnm: perfusion image number.

Example for using the ASLtoolbox in command line:

Suppose we have selected a series of preprocessed ASL images and save their names into a variable “imgs” and we have an M0 image whose name is saved in “M0img”, a CSF mask image whose name saved in “M0csfroi”, a WM mask image whose name saved in “M0wmroi”, M0csf value and M0wm value saved in “m0csf” and “m0wm” respectively. The following command lines will illustrate different quantification approaches by choosing different input options of asl_perf_subtract.

```
1. asl_perf_subtract(imgs,  
    0, % first ASL image in the series is label image  
    0, % subtraction type: simple subtraction  
    1, % subtraction order: control - label  
    [1 1 1 0 0 1 0], % MaskFlag: 1, will mask out the background voxel  
    ... MeanFlag: 1, will output mean images (mean PERF, mean CBF, and mean BOLD)  
    ... CBFFlag: 1, will calculate CBF (if you just want do a subtraction, set it to  
    0)  
    ... BOLDFlag: 0, not save the pseudo bold  
    ... OutPerfFlag: 0, not save the perfusion difference image series  
    ... OutCBFFlag: 1, save CBF image series  
    ... QuantFlag: 0, use mapwise M0 calibration  
    ... ImgFormatFlag: not set, will take the default value of 0 (using Analyze format)  
    ... D4Flag: not set, will take default value of 0, meaning not saving images in 4D  
    format  
    ... M0wmcsfFlag: not set, will be disabled.  
    0.5, % time shift for since interpolation. Useless here because the subtraction type is set to be  
    simple subtraction.  
    1, % CASL  
    0.68, %label efficiency  
    1, % Magtype 1 means 3T  
    1.6, % labeling time  
    1, % delay time  
    45, % slice timing for 2D acquisition. Should be 0 for 3D ASL  
    17, % TE, only be used when QuantFlag is 1  
    [], % M0 image, CASL usually does not take additional M0 image, it uses M0 control  
    [], % M0 roi mask image, not used here  
    [], % whole brain mask, if provided, this mask image will be used to remove background voxels.  
    [], % M0csf, not relevant here  
    [] % M0wm, not used here  
);
```

Suppose the ASL images are CASL data, this line will run a simple subtraction based quantification.

```
2. asl_perf_subtract(imgs,  
0, % first ASL image in the series is label image  
0, % subtraction type: simple subtraction  
1, % subtraction order: control - label  
[1 1 1 0 0 1 1 1 1], %MaskFlag: 1, will mask out the background voxel  
... MeanFlag: 1, will output mean images (mean PERF, mean CBF, and mean BOLD)  
... CBFFlag: 1, will calculate CBF (if you just want do a subtraction, set it to  
0)  
... BOLDFlag: 0, not save the pseudo bold  
... OutPerfflag: 0, not save the perfusion difference image series  
... OutCBFFlag: 1, save CBF image series  
... QuantFlag: 1, use a unique M0 for calibration  
... ImgFormatFlag: 1, will use Nifti format for saving the results  
... D4Flag: 1, will save images in 4D format  
... M0wmcsfFlag: not set, will set to 1 (using M0csf).  
0.5, % time shift for since interpolation. Useless here because the subtraction type is set to be  
simple subtraction.  
1, % CASL  
0.68, %label efficiency  
1, % Magtype 1 means 3T  
1.6, % labeling time  
1, % delay time  
45, % slice timing for 2D acquisition. Should be 0 for 3D ASL  
17, % TE, only be used when QuantFlag is 1  
[], % M0 image, CASL usually does not take additional M0 image, it uses M0 control  
[], % M0 roi mask image, not used here  
[], % whole brain mask, if provided, this mask image will be used to remove background voxels.  
m0csf, % M0csf,  
[] % M0wm, not used here  
);  
This line will run a simple subtraction based M0csf based CBF quantification for a CASL data  
(suppose this is the case).
```

```
3. asl_perf_subtract(imgs,  
0, % first ASL image in the series is label image  
0, % subtraction type: simple subtraction  
1, % subtraction order: label - control  
[1 1 1 0 0 1 1 1 1], %MaskFlag: 1, will mask out the background voxel  
... MeanFlag: 1, will output mean images (mean PERF, mean CBF, and mean BOLD)
```

```

... CBFFlag: 1, will calculate CBF (if you just want do a subtraction, set it to
0)
... BOLDFlag: 0, not save the pseudo bold
... OutPerfFlag: 0, not save the perfusion difference image series
... OutCBFFlag: 1, save CBF image series
... QuantFlag: 1, use a unique M0 for calibration
... ImgFormatFlag: 1, will use Nifti format for saving the results
... D4Flag: 1, will save images in 4D format
... M0wmcsfFlag: not set, will set to 1 (using M0csf).

```

0.5, % time shift for since interpolation. Useless here because the subtraction type is set to be simple subtraction.

0, % PASL

0.9, %label efficiency

1, % Magtype 1 means 3T

1.6, % labeling time, useless for PASL

1, % delay time

45, % slice timing for 2D acquisition. Should be 0 for 3D ASL

17, % TE, only be used when QuantFlag is 1

[], % M0 image, CASL usually does not take additional M0 image, it uses M0 control

[], % M0 roi mask image, not used here

[], % whole brain mask, if provided, this mask image will be used to remove background voxels.

m0csf, % M0csf,

[] % M0wm, not used here

);

This line will run a simple subtraction based M0csf based CBF quantification for a PASL data (suppose this is the case).

4. asl_perf_subtract(imgs,

0, % first ASL image in the series is label image

0, % subtraction type: simple subtraction

1, % subtraction order: label - control

[1 1 1 0 0 1 1 1 1 0], % MaskFlag: 1, will mask out the background voxel

```

... MeanFlag: 1, will output mean images (mean PERF, mean CBF, and mean BOLD)

```

```

... CBFFlag: 1, will calculate CBF (if you just want do a subtraction, set it to
0)

```

```

... BOLDFlag: 0, not save the pseudo bold

```

```

... OutPerfFlag: 0, not save the perfusion difference image series

```

```

... OutCBFFlag: 1, save CBF image series

```

```

... QuantFlag: 1, use a unique M0 for calibration

```

```

... ImgFormatFlag: 1, will use Nifti format for saving the results

```

```

... D4Flag: 1, will save images in 4D format

```

```

... M0wmcsfFlag: 0, using M0wm.

```

0.5, % time shift for since interpolation. Useless here because the subtraction type is set to be simple subtraction.

0, % PASL

0.9, %label efficiency

1, % Magtype 1 means 3T

1.6, % labeling time, useless for PASL

1, % delay time

45, % slice timing for 2D acquisition. Should be 0 for 3D ASL

17, % TE, only be used when QuantFlag is 1

[], % M0 image, CASL usually does not take additional M0 image, it uses M0 control

[], % M0 roi mask image, not used here

[], % whole brain mask, if provided, this mask image will be used to remove background voxels.

m0csf, % M0csf, not relevant here

m0wm % M0wm, not used here

);

This line will run a simple subtraction based M0wm based CBF quantification for a PASL data (suppose this is the case).

5. asl_perf_subtract(imgs,

0, % first ASL image in the series is label image

0, % subtraction type: simple subtraction

1, % subtraction order: label - control

[1 1 1 0 0 1 1 1 1 0], % MaskFlag: 1, will mask out the background voxel

... MeanFlag: 1, will output mean images (mean PERF, mean CBF, and mean BOLD)

... CBFFlag: 1, will calculate CBF (if you just want do a subtraction, set it to 0)

... BOLDFlag: 0, not save the pseudo bold

... OutPerfFlag: 0, not save the perfusion difference image series

... OutCBFFlag: 1, save CBF image series

... QuantFlag: 1, use a unique M0 for calibration

... ImgFormatFlag: 1, will use Nifti format for saving the results

... D4Flag: 1, will save images in 4D format

... M0wmcsfFlag: 0, using M0wm.

0.5, % time shift for since interpolation. Useless here because the subtraction type is set to be simple subtraction.

0, % PASL

0.9, %label efficiency

1, % Magtype 1 means 3T

1.6, % labeling time, useless for PASL

1, % delay time

45, % slice timing for 2D acquisition. Should be 0 for 3D ASL

17, % TE, only be used when QuantFlag is 1

```

M0img, % M0 image,
M0wmroi, % M0 roi mask image, not used here
[], % whole brain mask, if provided, this mask image will be used to remove background voxels.
m0csf, % M0csf, not relevant here
m0wm % M0wm, not used here
);

```

This line will run a simple subtraction based M0wm based CBF quantification for a PASL data (suppose this is the case). Note: although you have provided M0img and M0wmroi, the M0wm will be directly taken from m0wm provided in the end. The same thing will happen for M0csf if the corresponding argument is explicitly provided.

6. `asl_perf_subtract(imgs,`
 - `0,` % first ASL image in the series is label image
 - `0,` % subtraction type: simple subtraction
 - `1,` % subtraction order: label - control
 - `[1 1 1 0 0 1 1 1 1 0],` % MaskFlag: 1, will mask out the background voxel
 - `... MeanFlag: 1,` will output mean images (mean PERF, mean CBF, and mean BOLD)
 - `... CBFFlag: 1,` will calculate CBF (if you just want do a subtraction, set it to 0)
 - `... BOLDFlag: 0,` not save the pseudo bold
 - `... OutPerfFlag: 0,` not save the perfusion difference image series
 - `... OutCBFFlag: 1,` save CBF image series
 - `... QuantFlag: 1,` use a unique M0 for calibration
 - `... ImgFormatFlag: 1,` will use Nifti format for saving the results
 - `... D4Flag: 1,` will save images in 4D format
 - `... M0wmcsfFlag: 0,` using M0wm.
 - `0.5,` % time shift for since interpolation. Useless here because the subtraction type is set to be simple subtraction.
 - `0,` % PASL
 - `0.9,` %label efficiency
 - `1,` % Magtype 1 means 3T
 - `1.6,` % T11 (700 or 800 msec) if this value is greater than 1 or smaller than 0.5, it will be disregarded and the default (700) will be used.
 - `1,` % delay time. Note delay time in QUIPSS should be set to TI2 or the inversion time (time from the 180 inversion pulse to the excitation pulse of the imaging sequence. In the code TI will be calculated as: $TI = \text{delay time} + \text{slice time}$
 - `45,` % slice timing for 2D acquisition. Should be 0 for 3D ASL
 - `17,` % TE, only be used when QuantFlag is 1
 - `M0img,` % M0 image,
 - `M0wmroi,` % M0 roi mask image, not used here
 - `[],` % whole brain mask, if provided, this mask image will be used to remove background voxels.
 - `[],` % M0csf, not relevant here

[] % M0wm, not used here

);

This line will run a simple subtraction based M0wm based CBF quantification for a PASL data (suppose this is the case). M0wm will be taken from the M0 image using the provided WM mask.

5.5 Determining ASL data acquisition parameters

Users should note that ASL acquisition parameters may be defined differently in different ASL sequences and it is then necessary to contact your vendors or the sequence providers about this. The following text and graphs are based on Siemens platform. For those who are using GE product sequence, you will only get a M0 image and a subtracted perfusion difference image.

Slice timing for 2D: If you open the ASL scan protocol in the scanner console, you will be able to find a similar interface in the right bottom of the screen to that shown in Fig. 5.8. Click the window after “TR” (as pointed by the blue arrow in Fig. 5.8), you will find a long horizontal colorbar in green at the bottom of the interface. The number under the colorbar on the left end (pointed by the purple arrow in Fig. 5.8) is the minimum TR. Number of slices can be also found in the same interface in the second window on the left top corner. Slice timing can be then calculated using:

$$(\text{minimum TR} - \text{labeling time} - \text{post labeling time}) / (\text{number of slices})$$

For example, in the protocol shown in this subsection, the slice time will be: $(3230 - 1406 - 1500) / 14 = 23.14$ ms. Note, this is just an example, and it should not be treated as a standard protocol. For those who are using Siemens product PICORE-based PASL sequence, slice timing can be obtained by: $(\text{minimum TR} - 50\text{ms} - \text{T12}) / (\text{number of slices})$, where 50 ms is the duration of the pre-saturation pulses.

Labeling time and post labeling time: Case a) in

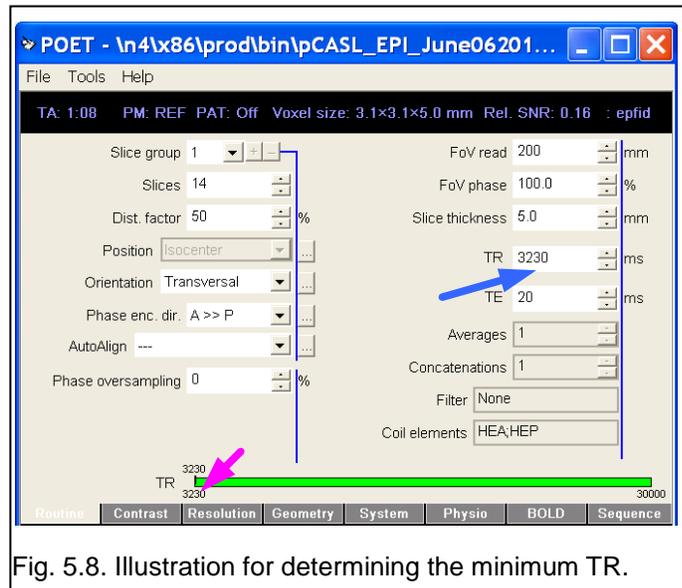


Fig. 5.8. Illustration for determining the minimum TR.

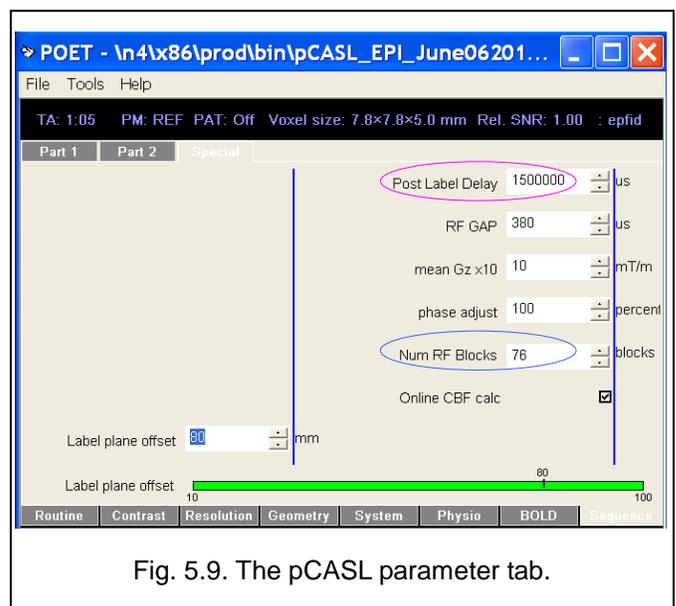


Fig. 5.9. The pCASL parameter tab.

the UPENN pCASL or the old CASL sequence distributed by Danny JJ Wang, labeling time is defined by “Num RF Blocks” (as marked by the blue circle in Fig. 5.9). Each RF block lasts for 18500 us, so the labeling time for 76 RF blocks set in the protocol shown on the right (Fig. 5.9) is $18500\text{us} \times 76 = 1.406\text{ sec}$. Post label delay can be directly got from the window marked with the purple ellipse. Case b) in Siemens product PASL (PICORE) sequence, you will be able to find "inversion time 1" and "inversion time 2", which is called TI1 and TI2 in the Q2tips technique developed by Eric Wong. The post labeling delay time to be used for `asl_perf_subtract`, the program for ASL CBF quantification in ASLtbx, should be TI2. For those who are using the FAIR-based PASL sequence distributed by Danny Wang, the post labeling time to be passed into `asl_perf_subtract` should be: Post IR Delay + Post Inf. Sat Delay (you can find these two parameters in the “special” tag in the protocol as shown below. For Q2tips, “labeling time” in `asl_perf_subtract` is used to pass TI1 (in sec) into the code (note the default TI1 is 0.7 sec, if labeling time is a number >0.5 and <1 , TI1 will be set to be $\text{TI1} = \text{labeling time}$); Case c) in GE product sequence (developed by David Alsop), labeling time and post labeling delay time can be found in the research mode. Post labeling delay time can be directly found in the protocol tab.



Parameters that may change slice time and therefore the minimum TR. In case you want to reduce the readout time to reduce off-resonance induced distortion, you can use several strategies to do that: 1. Use parallel imaging, 2. Use partial Fourier acquisition, 3. Increase bandwidth and reduce echo spacing. The 3rd approach is not recommended for general users because severe ghosting artifacts might result unless you can do off-line reconstruction. Typical bandwidth for 3T Trio can be from 2232 to 3004 Hz/pixel. Remember to reduce TE after you choose any of the above steps.

6. Output

Dependent on the data and options chosen, there will be a few intermediate files and final output files. Below is list of possible files seen in the exemplar data. You may also see some files named by an extension of “.mat”. Those are created by SPM to save the spatial transform and origin of the imaged object. CBF values: you will see a message like “The mean CBF is 56.81 (58.10)”. The second one (inside the parentheses) is from the whole brain (defined by the mask or a threshold of 0.2 of the maximum intensity of the mean control image); the first one is from the whole-brain as well but the whole brain mask has been refined to exclude outliers.

Output images:

Images written to the disk: Perfusion images, BOLD images, and CBF images (if the flags are turned on accordingly).

The mean of each perfusion difference image and CBF image will be saved to a txt file: `global....txt`. There will be 4 columns in that text file: the first two columns are the global mean perfusion difference

and CBF using an outlier excluding brain mask; the second 2 columns are the global mean perfusion difference and CBF value of the entire brain using the brain mask either provided by the argument “masking” or generated inside the code when no such argument is provided. Unless later noticed, we don’t suggest using the first 2 columns since the “outliers” there were defined using an arbitrary threshold.

All the output files will be saved in the same directory as the input files.

Depending on which version of `asl_perf_subtract.m` you are using, you will see one line of output message in Matlab saying “The mean CBF is xxx” or “The mean CBF is xxx (xxx)”. The former means whole brain mean CBF within the brain mask; in the latter case (in earlier versions of `asl_perf_subtract.m`), the first number reported is after removing outliers (empirically defined with a range of -40 to 150) and the second one (in parentheses) is the whole brain mean calculated from the brain mask provided or implicitly calculated in the code. Because the outlier definition is too arbitrary, the first value is not recommended for analysis. The 2nd one is not ideal as well if the brain mask contains skull or CSF. But users can always use finer masks to calculate whole brain mean or grey matter mean CBF after CBF quantification.

There is also a txt file (often named by “globalsgxxx.txt”. see below in the table) recording the global CBF and perfusion signal for each timepoint. In the old version, you will find 4 columns saved in an order of: mean perfusion difference signal from the outlier-cleaned mask, mean CBF from the outlier-cleaned mask, mean perfusion difference signal from the whole-brain mask, and mean CBF from the whole-brain mask. In the newer version, you will only see two columns, which are the mean CBF and the mean perfusion difference signal from the whole brain mask, respectively. For the same reason given above, the 2nd and the 4th columns are more informative. Note that only two columns may be recorded in `globalsg.txt` in later version of `asl_perf_subtract.m`.

filename	descriptions
ASL.nii	The 4D raw data saved in an order of label-control-label ...
rASL.nii	Output of motion correction.
meanASL.nii	Output of motion correction. The average of all images.
rp_ASL.txt	Output of motion correction. Motion timecourses. There are 12 columns. The first 6 ones are the output of SPM motion correction: x,y,z translations and 3 rotations; the last 6 are the zigzag pattern cleaned motion timecourses.
ASLflt_rASL.nii	Output of temporal filtering.
sASLflt_rASL.nii	Output of smoothing.
meanCBF_0_srASLflt_rASL.nii	Output of CBF quantification. The mean CBF map. “_0_” means simple subtraction.
cmeanCBF_0....	Outlier cleaned mean CBF map.
meanPERF_0_srASLflt_rASL.nii	Output of CBF quantification. The mean perfusion difference map (nonquantitative).
wmeanCBF_0_srASLflt_rASL.nii	Mean CBF map warped into the MNI space.
cbf_0_srASLflt_rASL.nii	Output of CBF quantification. The CBF map time series. Each control/label image pair has a corresponding CBF map saved in this

	file.
brainmask.nii	A rough brain mask by hard thresholding if FSL BET is not available.
globalsg_0.txt	The first and second column are the whole brain mean of the perfusion difference image and the CBF map at each timepoint.

7. Acknowledgement

Developing and maintaining ASLtbx is purely voluntary. There is no financial support directly for this endeavor. I would thank Dr. Zhengjun Li for setting up the website, polishing the scripts, and drafting the first version of this manual; thank Dr. Sudipto Douli for various code testing and releasing the SCORE scripts. I would also thank many users and developers for providing valuable feedbacks and suggestions especially for the various compatibility issues. The earliest version of the CBF quantification code was translated from an IDL code developed at Upenn with contributions from Danny JJ Wang, Geoffrey K. Aguirre, and David C. Alsop.

1. Detre, J.A., et al., *Perfusion imaging*. Magnetic Resonance in Medicine, 1992. **23**: p. 37-45.
2. Williams, D.S., et al., *Magnetic resonance imaging of perfusion using spin inversion of arterial water*. Proceedings of the National Academy of Sciences, 1992. **89**(1): p. 212-216.
3. Detre, J.A., et al., *Tissue specific perfusion imaging using arterial spin labeling*. NMR Biomed, 1994. **7**(1-2): p. 75-82.
4. Edelman, R.R., et al., *Qualitative mapping of cerebral blood flow and functional localization with echo-planar MR imaging and signal targeting with alternating radio frequency*. Radiology, 1994. **192**(2): p. 513-20.
5. Kim, S.G., *Quantification of relative cerebral blood flow change by flow-sensitive alternating inversion recovery (FAIR) technique: application to functional mapping*. Magn Reson Med, 1995. **34**(3): p. 293-301.
6. Wong, E.C., R.B. Buxton, and L.R. Frank, *Implementation of quantitative perfusion imaging techniques for functional brain mapping using pulsed arterial spin labeling*. NMR Biomed, 1997. **10**(4-5): p. 237-49.
7. Golay, X., et al., *Transfer insensitive labeling technique (TILT): application to multislice functional perfusion imaging*. Journal of magnetic resonance imaging : JMRI, 1999. **9**(3): p. 454-61.
8. Alsop, D.C. and J.A. Detre, *Reduced transit-time sensitivity in noninvasive magnetic resonance imaging of human cerebral blood flow*. J Cereb Blood Flow Metab, 1996. **16**: p. 1236-1249.
9. Alsop, D.C. and J.A. Detre, *Multisection Cerebral Blood Flow MR Imaging with Continuous Arterial Spin Labeling*. Radiology, 1998. **208**: p. 410-416.
10. Dai, W., et al., *Continuous flow-driven inversion for arterial spin labeling using pulsed radio frequency and gradient fields*. Magn Reson Med, 2008. **60**(6): p. 1488-97.
11. Wu, W.C., et al., *Physiological modulations in arterial spin labeling perfusion magnetic resonance imaging*. IEEE Trans Med Imaging, 2009. **28**(5): p. 703-9.
12. Wang, Z., et al., *Empirical optimization of ASL data analysis using an ASL data processing toolbox: ASLtbx*. Magn

Reson Imaging, 2008. **26**(2): p. 261-269, PMC2268990.

13. Wang, Z., *Improving Cerebral Blood Flow Quantification for Arterial Spin Labeled Perfusion MRI by Removing Residual Motion Artifacts and Global Signal Fluctuations*. Magnetic Resonance Imaging, 2012(in press).