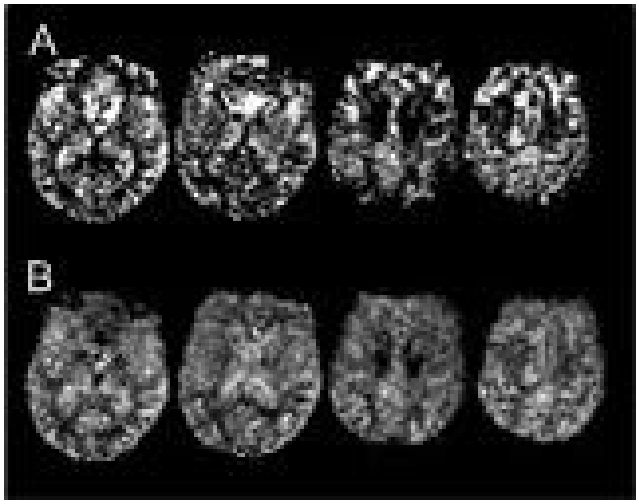


ASLtbx Manual

(May 28 2011)

http://cfn.upenn.edu/~zewang

Ze Wang

Dept. of Psychiatry, School of Medicine,
Department of Bioengineering
School of Engineering and Applied Science
University of Pennsylvania
zewang@mail.med.upenn.edu
redhatw@gmail.com

CONTENT

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 technique [1] . The toolbox is free for academic users, and can be obtained from http://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 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.

The users are supposed to know the basic idea of fMRI and ASL MRI. They should also know how to program in Matlab. Please cite the toolbox and the related papers if you are using it or modifying the code. We are happy to help and happy to collaborate.

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

ASLtbx is a collection of processing steps for ASL data. The premier version of the pipeline and the toolkit were first described in the following paper:
Ze Wang, Geoffrey Aguirre, Hengyi Rao, JiongJiong Wang, Anna R. Childress, John A. Detre, Empirical ASL data analysis using an ASL data processing toolbox: ASLtbx, Magnetic Resonance Imaging, 2008, 26(2):261-9.

and further enhanced in:
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).*

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

ASLtbx is implemented in Matlab m-scripts. It 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 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 (SPM2, SPM5, or SPM8 are preferred, though SPM99 can be still supported) 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.

2. Installing ASLtbx

2.1 Download ASLtbx

ASLtbx can be downloaded from the author's website (http://cfn.upenn.edu/~zewang) or through the website of the Center for Functional Neuroimaging. It's available as a zipped file named "ASLtbx.zip". The example datasets can be downloaded through the same web site following the steps illustrated below.

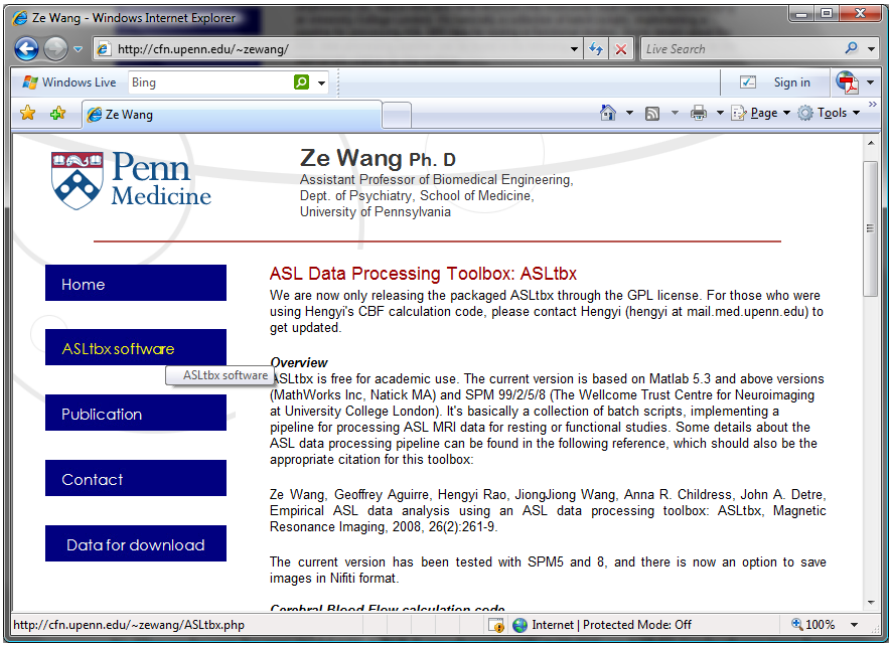


Figure 2.1. Screen capture showing the link to the download page.

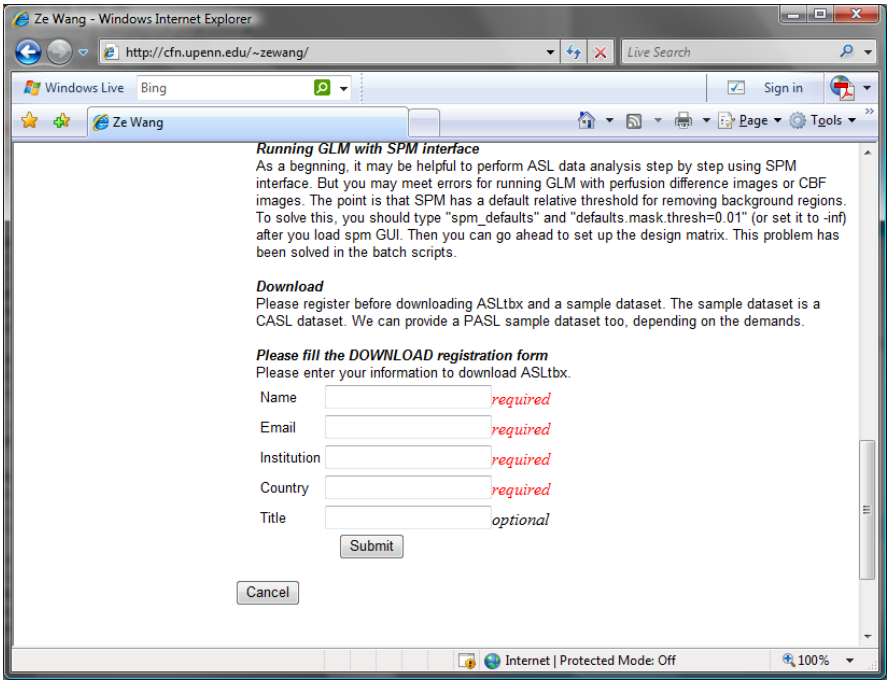


Figure 2.2. Registration page.

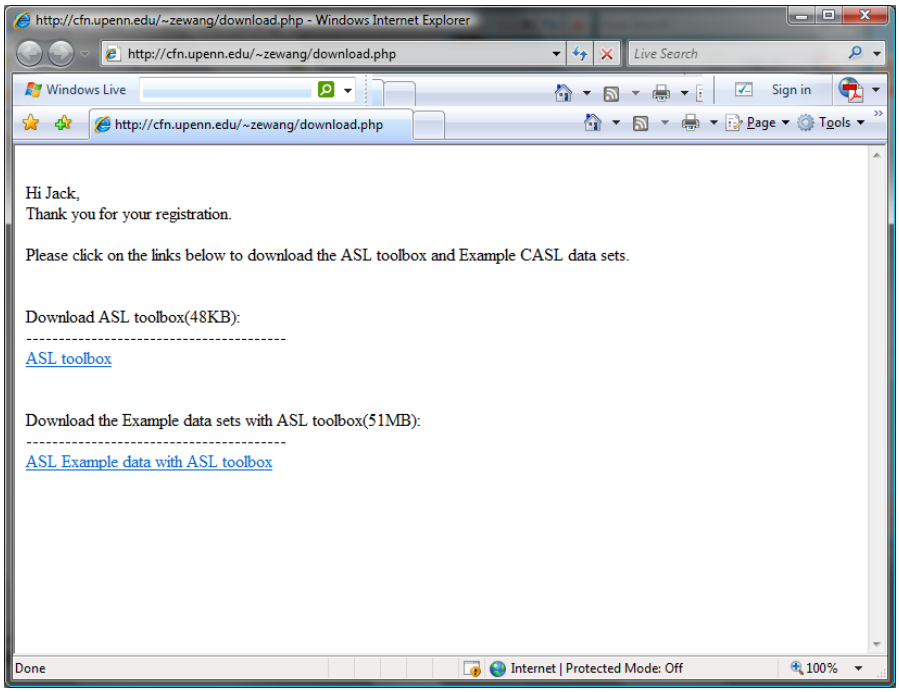


Figure 2.3. The real download page.

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 3d for resting pCASL. Each dataset has preset scripts showing the acquisition parameters and settings for processing.
The following instructions are for the ASL fMRI data set, and can be easily adapted to analyze brain state ASL data such as long video cue condition vs neutral condition. Fewer pre-processing steps are required for processing resting ASL data. Please check the resting data and the associated scripts for the details. We will add instructions for processing resting data later.

2.4.1 Data structure and parameters

Go back to the download page to get the example dataset used in this tutorial. Unzip the data into a new directory (say, “Z:\ASL\ASL_Example_data “) and copy the folders (“batch_scripts” “sub1” “sub2” “sub3”) into it. The example dataset contains four folders, “batch_scripts” , “sub1” , “sub2” and “sub3” . The first contains batch scripts customized for processing the 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, 192×256 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. Acquisition parameters were: FOV=22 cm, 64×64×12 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

During the CASL scan, a block design with two interleaved conditions was used. 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.

The scripts in folder “batch_scripts” are ready to be used for processing the example dataset. 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 prepare the imaging data. Otherwise, it will be a headache for modifying the scripts.

3 ASL data preprocessing using SPM GUI

ASLtbx can process images that are in the same orientation and have the same voxel size. It is our recommendation that the raw images should be preprocessed with the following steps before calculating the quantitative CBF.

3.1 Reorient the images

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. The comparisons were made in spm5. Since the same dataset might be processed ten or more times before publication, it is convenient to put down the coordinators 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 made to the pipeline.

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.

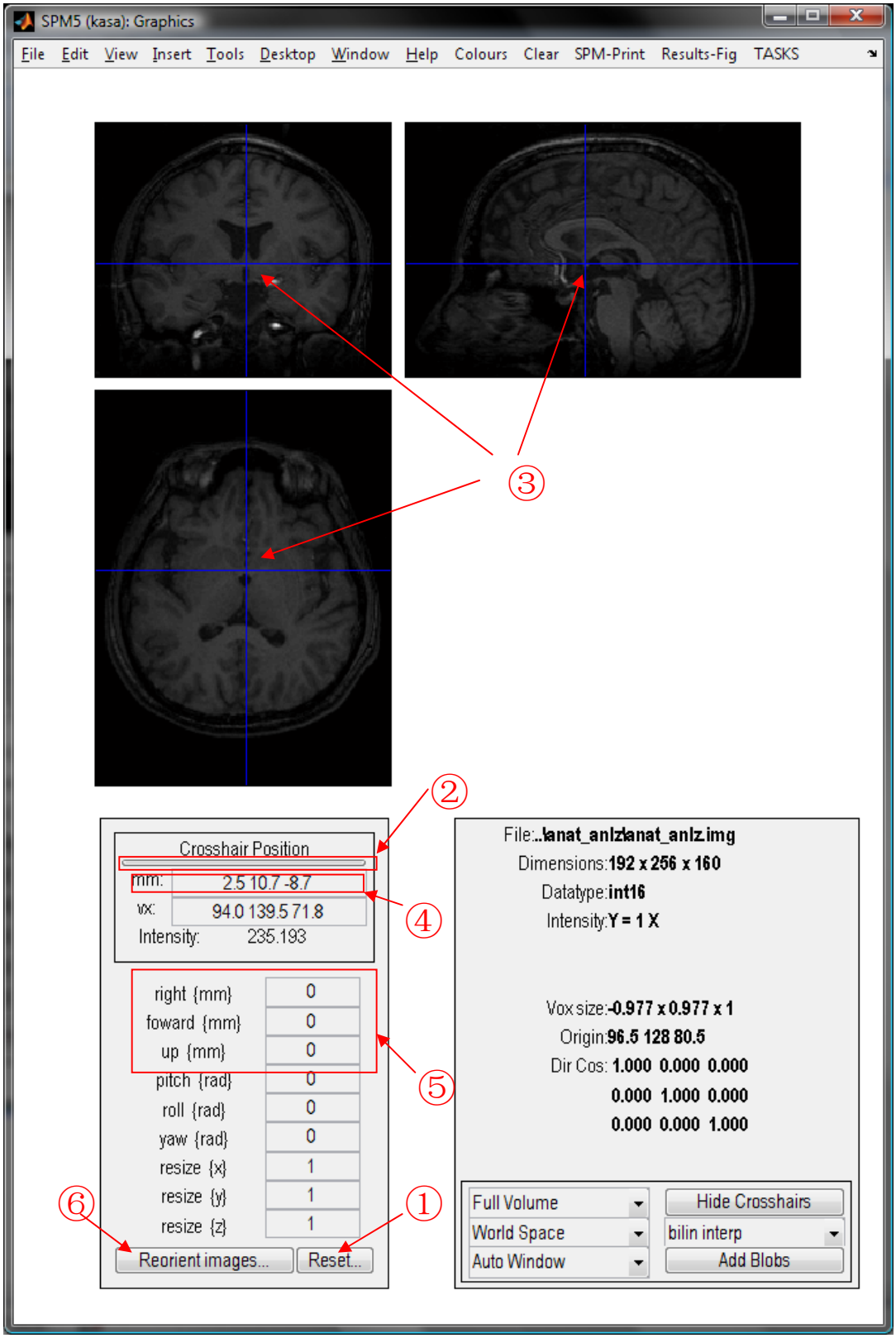


Figure 3.1. Illustration for how to find and set up image orientation using SPM GUI.

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 the offset between the center of the volume and the Anterior Commissure, you can reorient the images by entering the **NEGATIVE** of 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. The same procedures should be performed for both structural and functional images.

3.2 Motion correction (NEW)

ASL data should be motion corrected for the control and label series separately. But that requires additional coregistration between these two series. We have adapted the motion correction function provided in SPM to avoid treating the zig-zagged spin labeling paradigm as additional motions. See batch_realign.m for the details.

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 standard

MNI 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

It is the author's flavor to use scripts for processing ASL data though every step can be implemented using the GUI. Users can try the sample data and the preset scripts to get familiar with 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 pipeline file "batch_run.m" shows the sequential ASL data processing flow, including 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 setting file can be verified by typing "par" in Matlab console and each filed can be checked by typing the name accordingly (like typing "PAR.subjects{:}").

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.

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 solution. Please try these 3 steps with the sample data several times to get an idea.

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. Please read the code comments for how to change the setting. To use the updated motion correction routine, Matlab 7 is required. 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

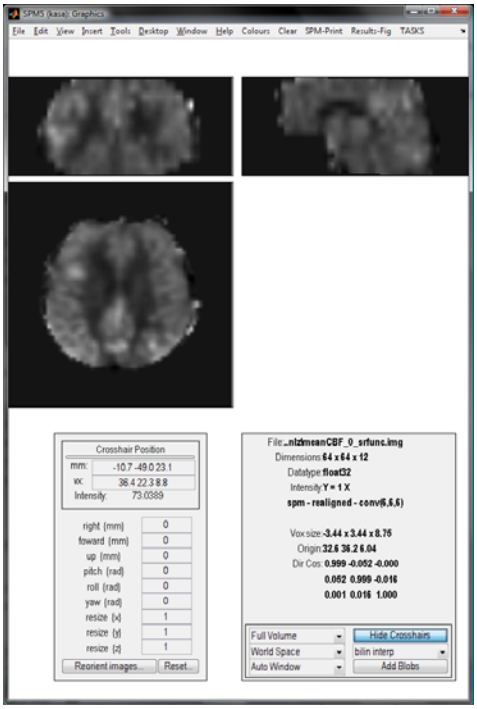
“batch_coreg.m” coregisters the realigned ASL images to each subject’s structure image. It will apply the transformation by modifying the header file of each functional image directly.

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 are produced in step 4.5. The mask image will be called “mask_perf_cbf.img” and will be saved in each subject’s functional folder. This mask file is used to exclude the outliers during the ASL perfusion subtraction. If it is not provided, a default mask will be created in the quantification code. This brain mask is a rough estimate using a simple intensity threshold method. Users can use their own created using a better way.



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.

Figure 4.1 The mean CBF map of sub2.

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 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.

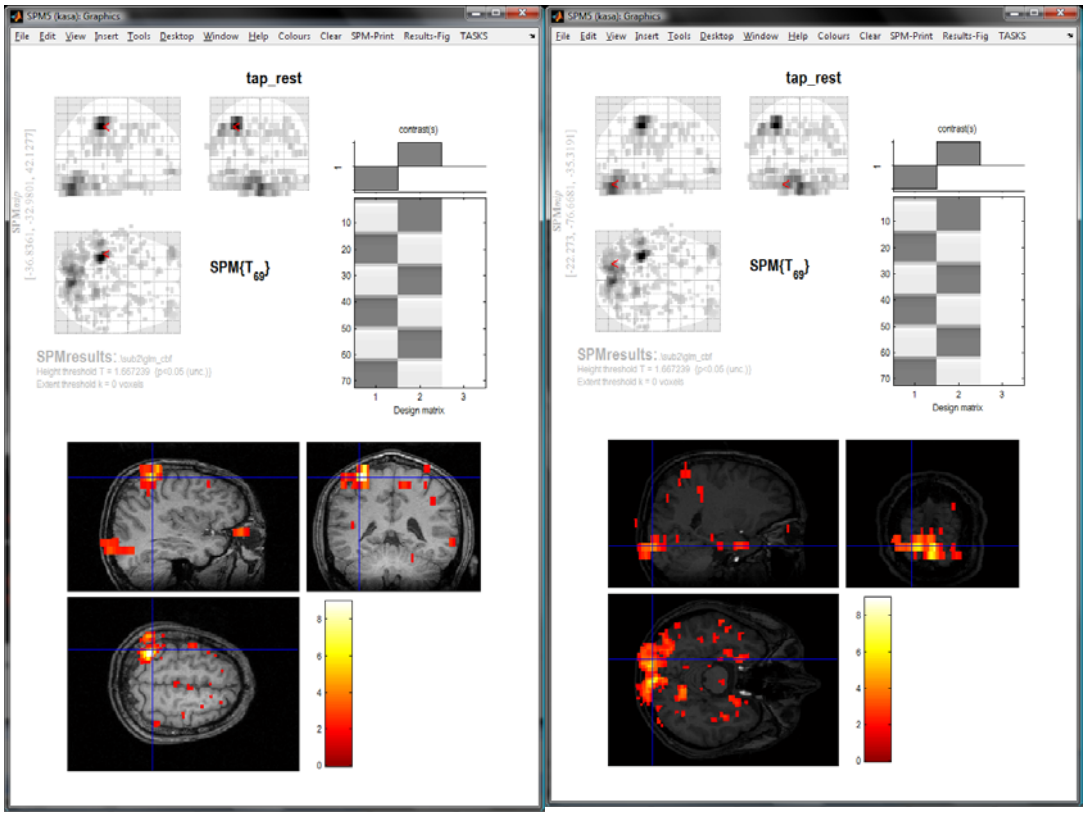


Figure 4.2 GLM analysis results of sub2. Activations in left motor cortex (as shown in the left image) and visual cortex (as shown in right the image) in response to the visual stimulation and right hand finger tapping task.

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.

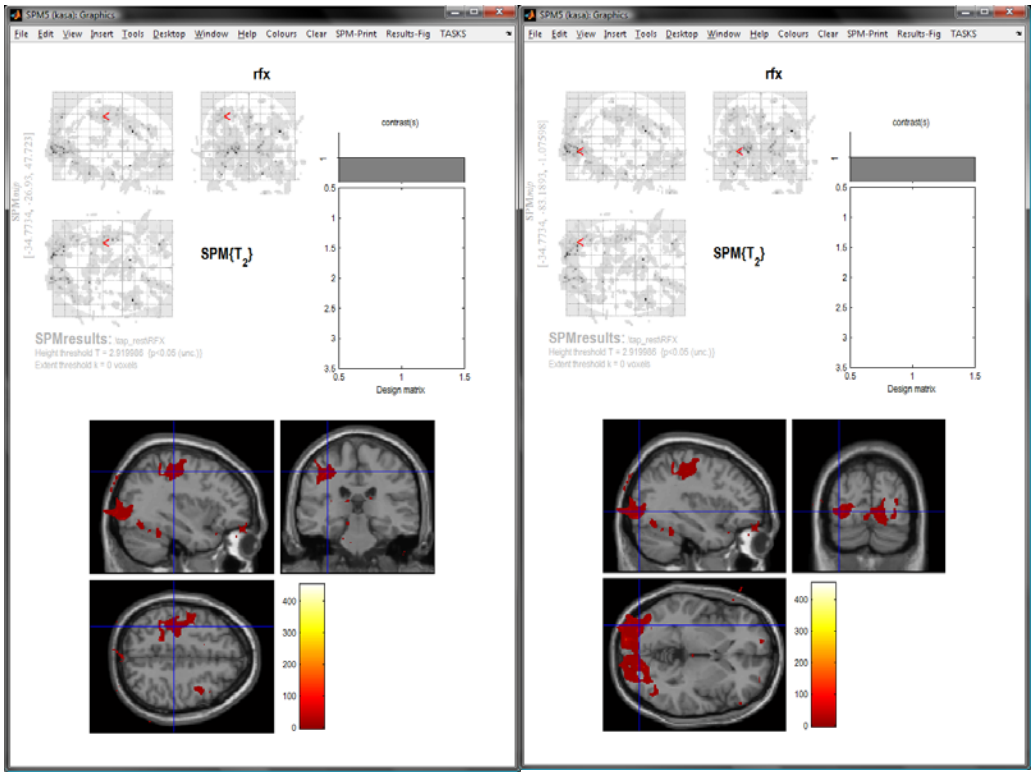
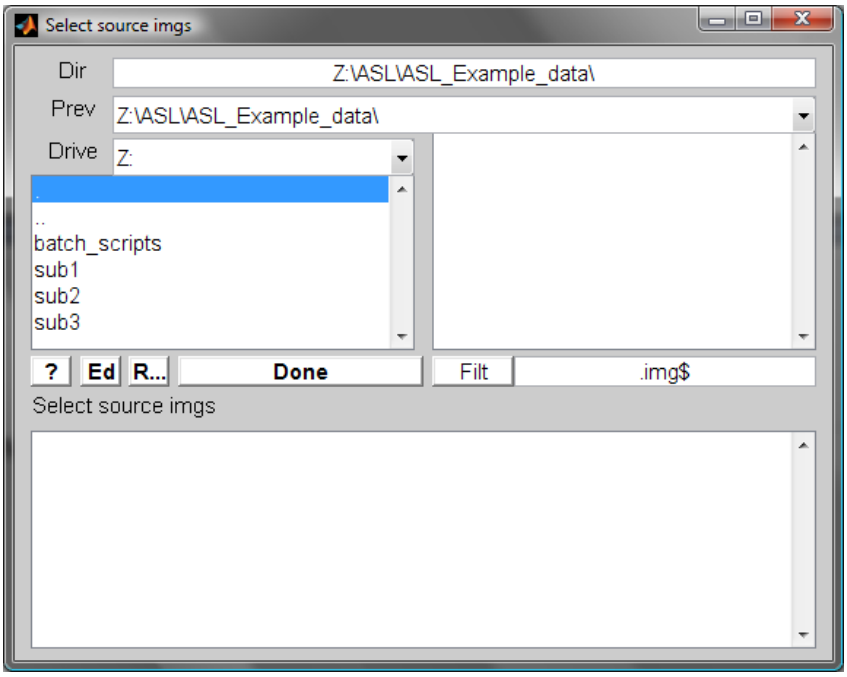


Figure 4.3 Group level results for the CASL fingertapping data.

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 adaptions 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.

5.1 GUI mode

In Matlab command window, simply type:

asl_perf_subtract

to call function in GUI, which will call a SPM function for selecting the input functional images. The images should be preprocessed and have the same orientation and voxel size.

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

Go to one subject's ASL image folder and select the preprocessed (realigned and smoothed) ASL images:

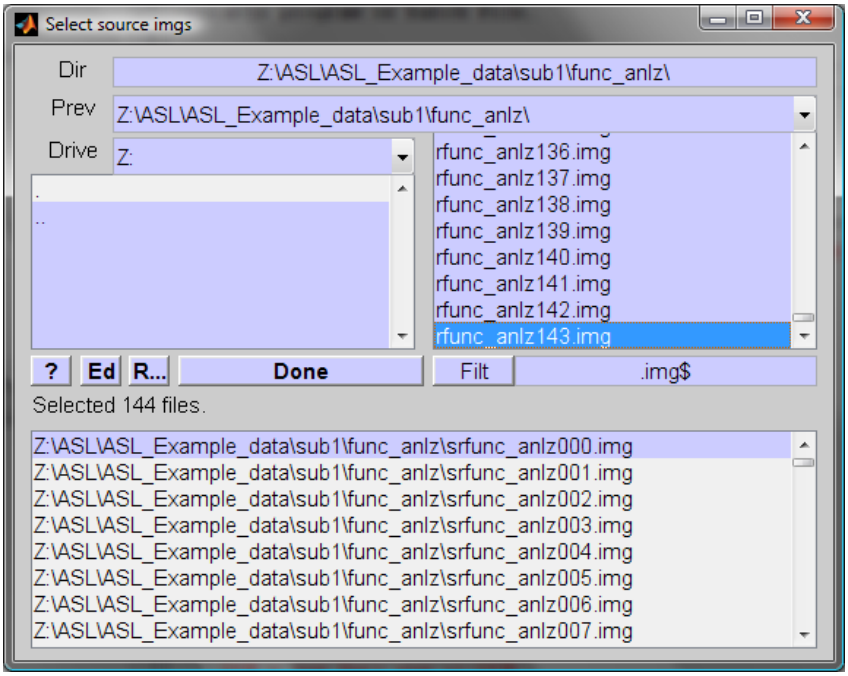


Figure 5.2 Step 2: select the preprocessed ASL images.

Press Done and you will get another pop-up window (Fig. 5.3) for inputting parameters or choosing different options:

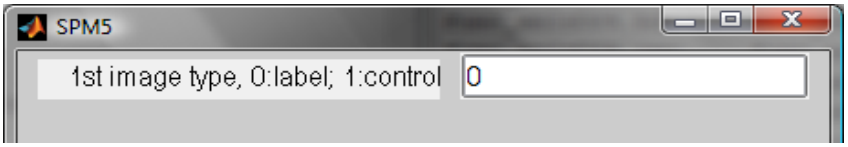


Figure 5.3. Input window for defining the image type of the 1st ASL image.

Set the values for other arguments. See section 5.2 for more details for the input arguments.

Figure 5.4. A snapshot of parameter settings for the sample data.

After you input all the required parameters. ASLtbx will start calculating the perfusion difference images or absolute perfusion images.

5.2 Command line

ASLtoolbox can also be called as a function in Matlab console:

```
[perfnum] = asl_perf_subtract(Filename, FirstimageType, SubtractionType,...
                              SubtractionOrder, Flag, …
```

Timeshift, AslType, labeff, MagType, …
Labeltime, Delaytime, Slicetime, TE, M0img, M0seg, maskimg)

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 subtractioin.

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.

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 as the TI2 (the second interval) in QUIPSS.

Slicetime - time for getting one slice, which can calculated through dividing the minimum TR 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.).

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:

perfnum: perfusion image number.

Example for using the ASLtoolbox in command line:

For processing the realigned and smoothed sub1's CASL data in example data set, the following two lines in MATLAB will select the images and do the CBF quantification:

```
Filename=spm_select('ExtFPList','Z:\ASL\ASL_Example_data\sub1\func_anlz', ['^sr' 'func_anlz' '.*\.img$']);
asl_perf_subtract(Filename,0, 0, ...
                1, [1 1 1 0 0 1 0], 0.5, 1, 0.68, 1,...
                1.6, 0.8, 45, 17, [],[],[]);
```

Output images:

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

Global perfusion difference signals and CBF values will be saved to a txt file.

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

6 License

7 Acknowledgement

Developing and maintaining ASLtbx is purely voluntary. There is no financial support directly for this endeavor. I would thank Zhengjun Li, a PhD candidate in my lab, who has helped a lot for setting up the website, polishing the scripts. He also drafted the first version of this manual. The earliest version of the CBF quantification code was translated from an IDL code developed at Upenn with contributions from Danny JJ Wang, Geoffrey Aguirre, and David Alsop.