Quality Control Functions Manual

Author: Bryce Johnson
Date: 8/26/19
Email: joh14192@umn.edu
Project Duration: May 2019-Aug 2019
Address: Waisman Center - 1500 Highland Ave, Madison, WI 53705
Advisors: Prof. Bradley Christian (bchristian@wisc.edu), Dr. Tobey Betthauser (tjbetthauser@medicine.wisc.edu)

Brief Outline:

I. Abstract

- i. Background:
  1. **Overall Goal**: Develop software tools that aid in the quality monitoring of dynamic PET images from the Horizon Scanner.
- ii. Approach:
  1. **Specific Aims:**
     - a. Create master script to perform quality checks across studies
     - b. Sort dicom header data.
     - c. Convert dicom to nifti
       - i. maintain image space
       - ii. recover lost header data during conversion
     - d. Motion Correction of a single nifti study
     - e. Define acceptable rigid motion parameters
- iii. Methods:
  1. Required Programming Environment/Tools
     - a. **Matlab r2018b update 2**
     - b. **SPM12 required**
     - c. **MacOS environment**
- iv. Results:
  1. **Primary Result: wc_master_qc** --> A master script that can work across studies that performs three main quality checks:
     - a. wc_dicom_petct_sorter4_0 -->sorts dicom metadata fieldnames into categories based on series attributes. Produces 4 csv files. [default: no sort unless specified]
     - b. wc_dicom2nii: converts dicom slices which are 3D image into a 3D nifti volume, while also producing an associated html file of metadata lost from a dicom slice.
       - i. wc_dicom2origin -->rewrites dicom files to maintain same image space as nifti image in AMIDE. [default is to not perform this]

c. wc_motion_corr_4D --> will perform an spm realignment of the dynamic PET series to a weighted mean image and return 6 rigid motion parameters , then checks if parameters are greater than the acceptable amount, will flag if so.
d. wc_write2error --> any errors/ flags thrown in any of the the above quality checks will write to a csv file.

2. **Secondary Result: wc_def_norm_movement** --> used to define the acceptable 6 rigid motion parameters.

v. Conclusions/Future

1. **Conclusion**: The quality control check consisted of three main parts: 1. sorting metadata produced by unique dicom series, 2. successful conversion of dicom to nifti while saving dicom metadata information that was lost into a nicely printed html file. 3. check for motion in dynamic PET series from a single study. These quality checks are meant to flag possible errors that come from the massive throughput of PET data, not correct any errors in the files.
2. **Future:** Future works include redefining acceptable motion parameters, displaying slices in html file, template production, method for testing overlay between modalities, and debugging.

II. Introduction

a. Motivation :

i. High throughput PET scanning for research at the Waisman Center
ii. Needs a computerized system to monitor the quality of data during preprocessing steps, must consider:
   1. Metadata lost during dicom to nifti conversion
   2. Preservation of image space during dicom to nifti conversion
   3. Motion of study participant while in scanner and other quality metrics
iii. Computational resources are abundant, but human resources are limited

b. Goals

i. Overall Goal: Develop software tools that aid in the quality monitoring of PET images from the Horizon scanner at the Waisman Center.
ii. Specific Aims:
   1. Recursively get directory specific file ids and decompress all files. Identify dicom files without using the ending identifier (i.e. '.dcm')
   2. Sort Metadata information from different series in a single dicom study
   3. Convert DICOM slices into NIfTI images
      a. Display metadata from a single DICOM slice in an html file
      b. Change coordinate space of dicom image so it overlays with corresponding nifti image in AMIDE

4. Perform motion correction to dynamic PiB PET series frames, using only rigid body rotations
    a. Take a weighted mean of dynamic PET frames for reference image in realignment
5. Define normal motion to be used in motion correction (above)
6. Write a master script that works across studies to perform quality control functions without failing due to any errors.

III. Methods/ Approach Tools

a. Programming Environment/Tools
    i. MATLAB programming environment (requires **Matlab r2018b update 2**)
        1. Functions written at Waisman Center: wc_*.m
        2. Functions downloaded from internet: print2html.m, cell2csv.m
        3. Must have Image Analysis Toolbox provided by MATLAB (dicominfo, dicomread, etc)
        4. Internal variables that are stored across program runs are stored in .mat files (i.e. error_path.mat : stores full path to where error file should be written)
    ii. **SPM12 required**, used for a large amount of the preprocessing
    iii. **MacOS** operating system is the environment this has been tested with. If want to run in windows and fails, email : joh14192@umn.edu to help identify issue

b. Specific Aims
    i. Get directory file ids for a specific file type
        1. Recursively find all the file ids in a given directory
        2. Decompress all files that have the extension '.tar','.tgz','.bz2', and '.gz' , continue to do so until all are decompressed, unless limit.
        3. Filter out specified files by reading metadata information, return a cell array containing file ids for specified file type
    ii. Sort metadata information from dicom series
        1. Took metadata from each dicom slice into a cell array
        2. Found unique Series UIDS
        3. Use logical arrays to…
            a. Identify which field names are local to all series
            b. Identify which field names are local to a single series
            c. Identify changing fieldnames in PET data
    iii. Convert dicom to nifti
        1. Get dicom headers for each slice
            a. Filter out series which don't have fieldname 'SliceThickness' as they are not 3D volumes
        2. Use SPM to convert each series to a Nifti image
            a. If more than one nifti image is created it is a dynamic PET series, merge the files into one 4D image with multiple frames.

b. Nifti file output hierarchy based off study number, with subfolders of series names.

3. Display metadata from a single DICOM slice in an html file
4. Compress the html header file with the nii image in the '.tar.gz' format.
5. Delete all the converted nifti images and header html file.
6. Shift all the dicom files so that they appear at the origin when brought up in the AMIDE image viewer, also flip data matrix to account for coordinate system change. Nifti and dicom file will now overlay in AMIDE image viewer.

iv. **Motion Correction of a single nifti study**

1. Find out which file is the 4D image in the nifti study
2. Find the weighted mean of the 4D pet series by multiplying each frame by its corresponding frame rate then dividing by the sum of all the frame rates. This puts larger emphasis on frames which took counts over a longer period of time.
3. Realign all frames in dynamic pet series to weighted mean image , get the rigid motion parameters.
4. If defining normal has already been done check to see if any frames motion is larger than the standard deviation. If so flag it and write file name to the error csv file.

v. **Defining normal motion parameters**

1. Make sure that Nifti file tree is in the proper format for many studies, get the paths for each study
2. Do motion correction of a every Nifti study (above). Save rigid motion parameters into a nx6 double array
3. Get the standard deviation for the [x, y, z, roll, pitch, yaw] rigid motion parameters. Save to a .mat file

vi. **Master Script**

1. Sort the dicom files,
2. Convert dicom to nifti
3. Perform motion correction
4. If an error occurs in the code use try -catch statements to catch the error and write the error and file to a csv file, with the specified directory or file where the error occurred .

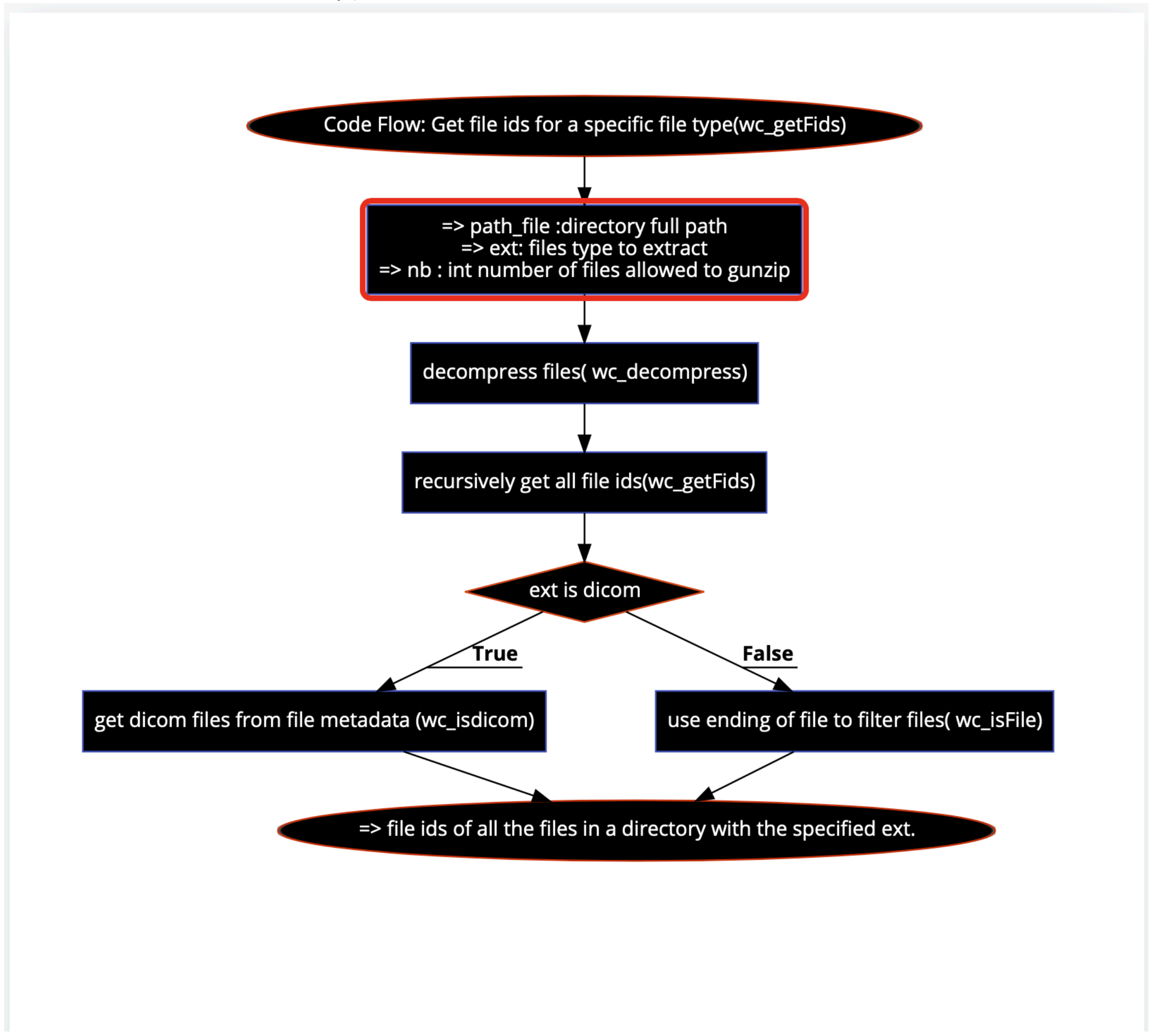
IV. Results

a. Flow for each Aim

i. **Get directory file ids for a specific file type**

1. Description: outputs all files in a directory of a certain file type. Will uncompress all .tar, bz2, and .gz, and .bz2 files. Will keep

going until every file has been opened so be careful. (Infinite Loop)


Code Flow: Get file ids for a specific file type(wc_getFids)
=> path_file :directory full path
=> ext: files type to extract
=> nb : int number of files allowed to gunzip
decompress files(wc_decompress)
recursively get all file ids(wc_getFids)
ext is dicom
True
False
get dicom files from file metadata (wc_isdicom)
use ending of file to filter files(wc_isFile)
=> file ids of all the files in a directory with the specified ext.

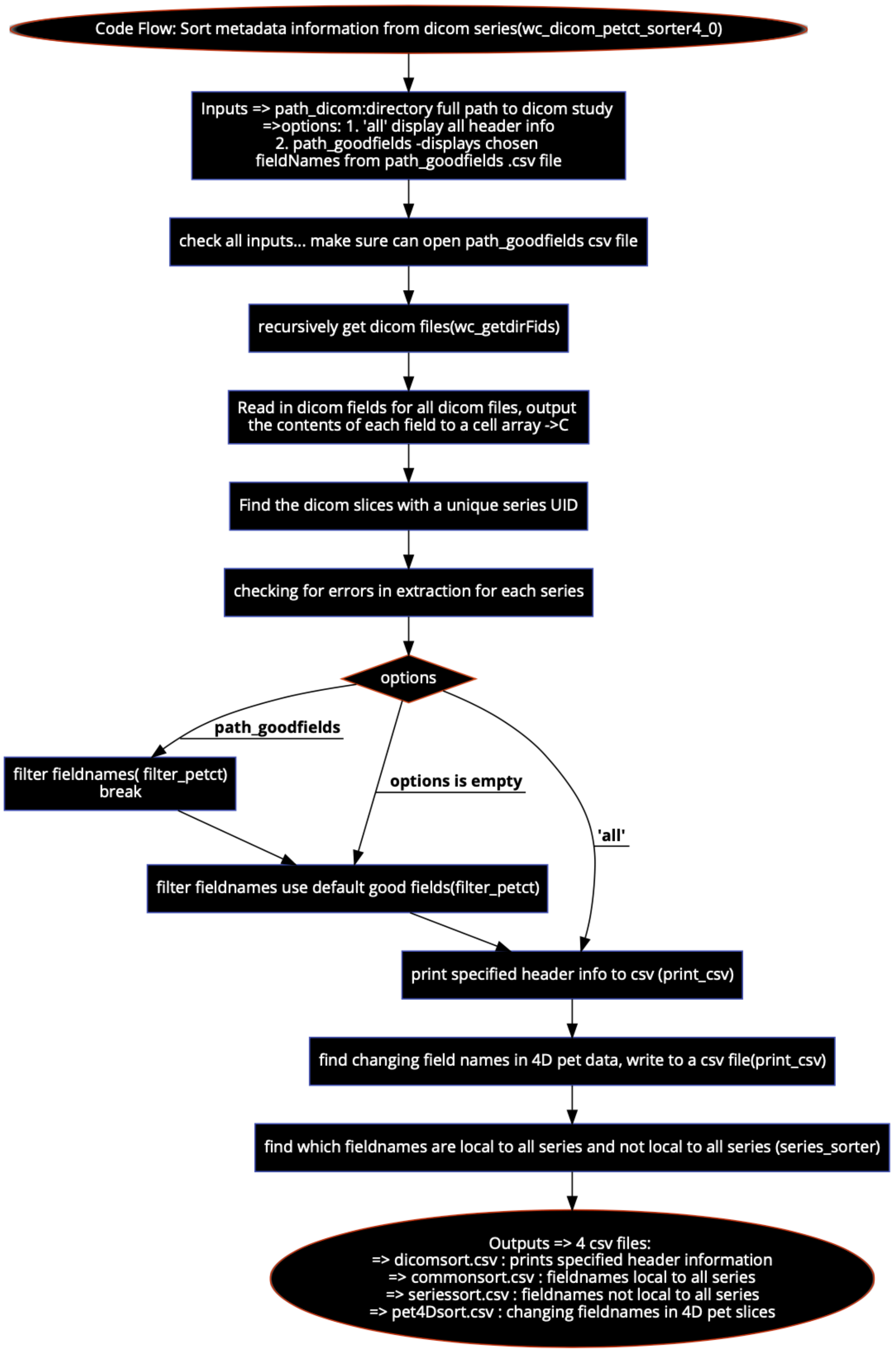
2. Inputs :
    a. path_file : character vector of full path to directory which files are located [default: pwd]
    b. ext: file type to extract, (ie ‘nii’) [default: dicom]
    c. nb: integer, numbr of files allowed to gunzip, done to save space of disk in case this function is called to the wrong directory. If gunzip all use Inf. [default: 6]
3. Function hiearchy
    a. wc_getdirfids
        i. wc_getFids
        ii. wc_decompress
            1. wc_getFids
        iii. wc_isdicom
        iv. isFile
4. Outputs:

a. fids : cell array of character vectors containing full paths to the specified file types in that directory.

ii. Sort metadata information from dicom series

1. Note: This function is clunky,slow, and suseptible to bugs. Unless necessary don’t use.

2. Description: Will display all or specified field names from different dicom series.
3. Inputs :
    a. path_dicom : directory to a single dicom study containing different dicom series. File heriarchy in directory doesn’t matter, but **must** consist of **one study**.

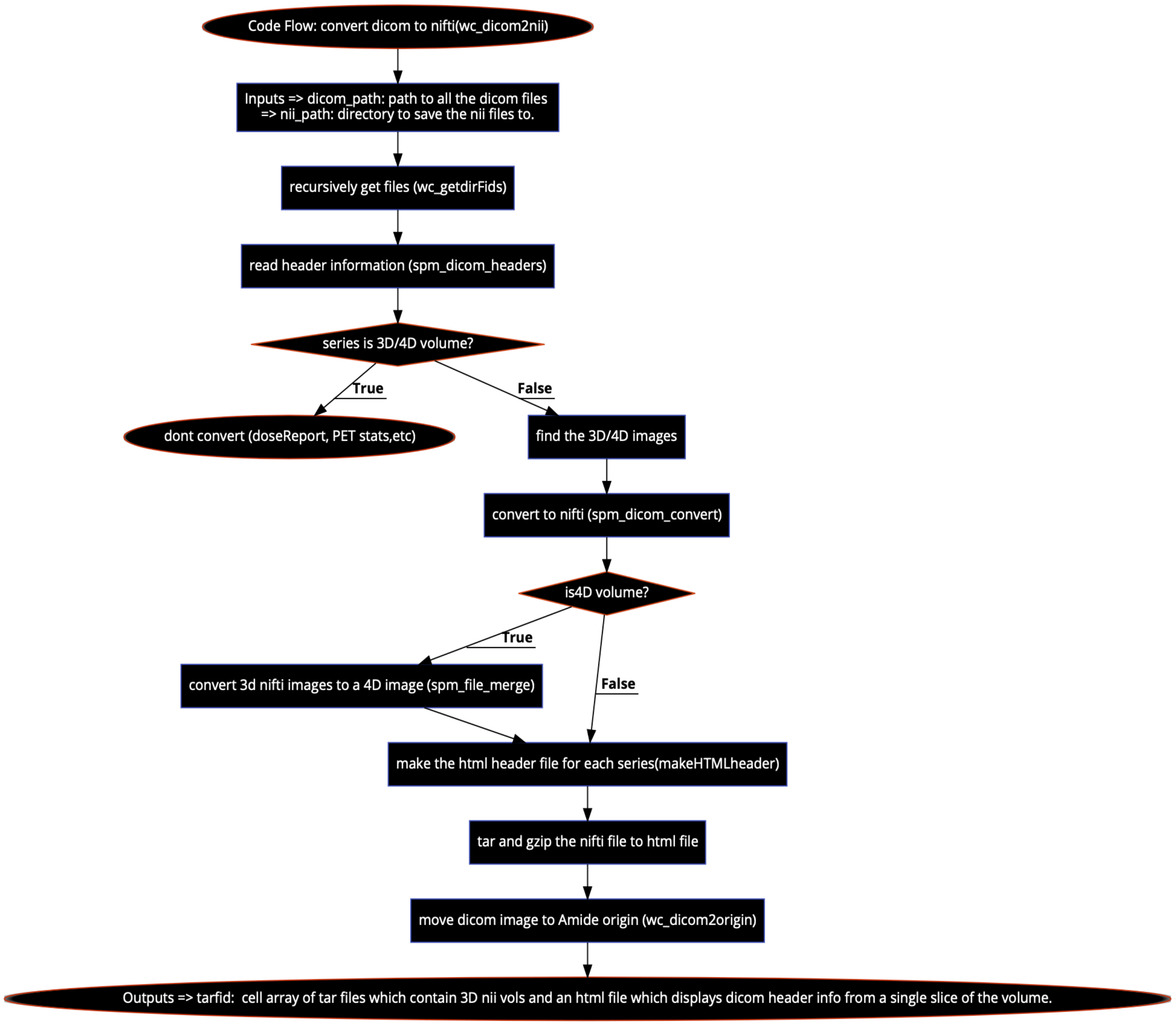


Code Flow: Sort metadata information from dicom series(wc_dicom_petct_sorter4_0)
Inputs => path_dicom:directory full path to dicom study =>options: 1. 'all' display all header info 2. path_goodfields -displays chosen fieldNames from path_goodfields .csv file
check all inputs... make sure can open path_goodfields csv file
recursively get dicom files(wc_getdirFids)
Read in dicom fields for all dicom files, output the contents of each field to a cell array ->C
Find the dicom slices with a unique series UID
checking for errors in extraction for each series
options
path_goodfields
options is empty
'all'
filter fieldnames(filter_petct) break
filter fieldnames use default good fields(filter_petct)
print specified header info to csv (print_csv)
find changing field names in 4D pet data, write to a csv file(print_csv)
find which fieldnames are local to all series and not local to all series (series_sorter)
Outputs => 4 csv files: => dicomsort.csv : prints specified header information => commonsort.csv : fieldnames local to all series => seriessort.csv : fieldnames not local to all series => pet4Dsort.csv : changing fieldnames in 4D pet slices

b. Options: The second input parameter can vary
    i. 'all' will display all fieldnames from every series
    ii. goodfields.csv : is the character array of the full path to a csv file containing all the fieldnames which should be included in output header file
    iii. [] : input nothing will trigger the default header fieldnames==>'Filename','ImageType','Modality','StudyInstanceUID','SeriesInstanceUID','SeriesNumber','SeriesDescription','NumberOfTimeSlices','RadiopharmaceuticalInformationSequence_Item_1_Radiopharmaceutical','NumberOfSlices'

4. Function hierarchy
    a. wc_dicom_petct_sorter4_0
        i. wc_getdirfids()
        ii. series_sorter
        iii. print_csv
        iv. filter_petct
5. Outputs:
    a. Four csv files:
        i. dicomsort.csv : prints specified field names for uniques Series
        ii. commonsort.csv: prints field names common to all series
        iii. seriesort.csv: prints field names not common to all
        iv. pet4Dsort.csv: prints fieldnames changing between 4D dicom slices

iii. Convert dicom to nii


Code Flow: convert dicom to nifti(wc_dicom2nii)
Inputs => dicom_path: path to all the dicom files
=> nii_path: directory to save the nii files to.
recursively get files (wc_getdirFids)
read header information (spm_dicom_headers)
series is 3D/4D volume?
True
False
dont convert (doseReport, PET stats,etc)
find the 3D/4D images
convert to nifti (spm_dicom_convert)
is4D volume?
True
False
convert 3d nifti images to a 4D image (spm_file_merge)
make the html header file for each series(makeHTMLheader)
tar and gzip the nifti file to html file
move dicom image to Amide origin (wc_dicom2origin)
Outputs => tarfid: cell array of tar files which contain 3D nii vols and an html file which displays dicom header info from a single slice of the volume.

1. Description: takes in a path to a dicom study and then converts all the series in that study to nii files. It produces a 4D image image if one is found. Also changes pixel and metadata of dicom image so it overlays with nifti image in amide.
2. Inputs:
   a. path_dicom: : directory to a single dicom study containing different dicom series. File heriarchy in directory doesn't

matter, but **must** only be one study. Don't be missing slices. Spm will throw an error.

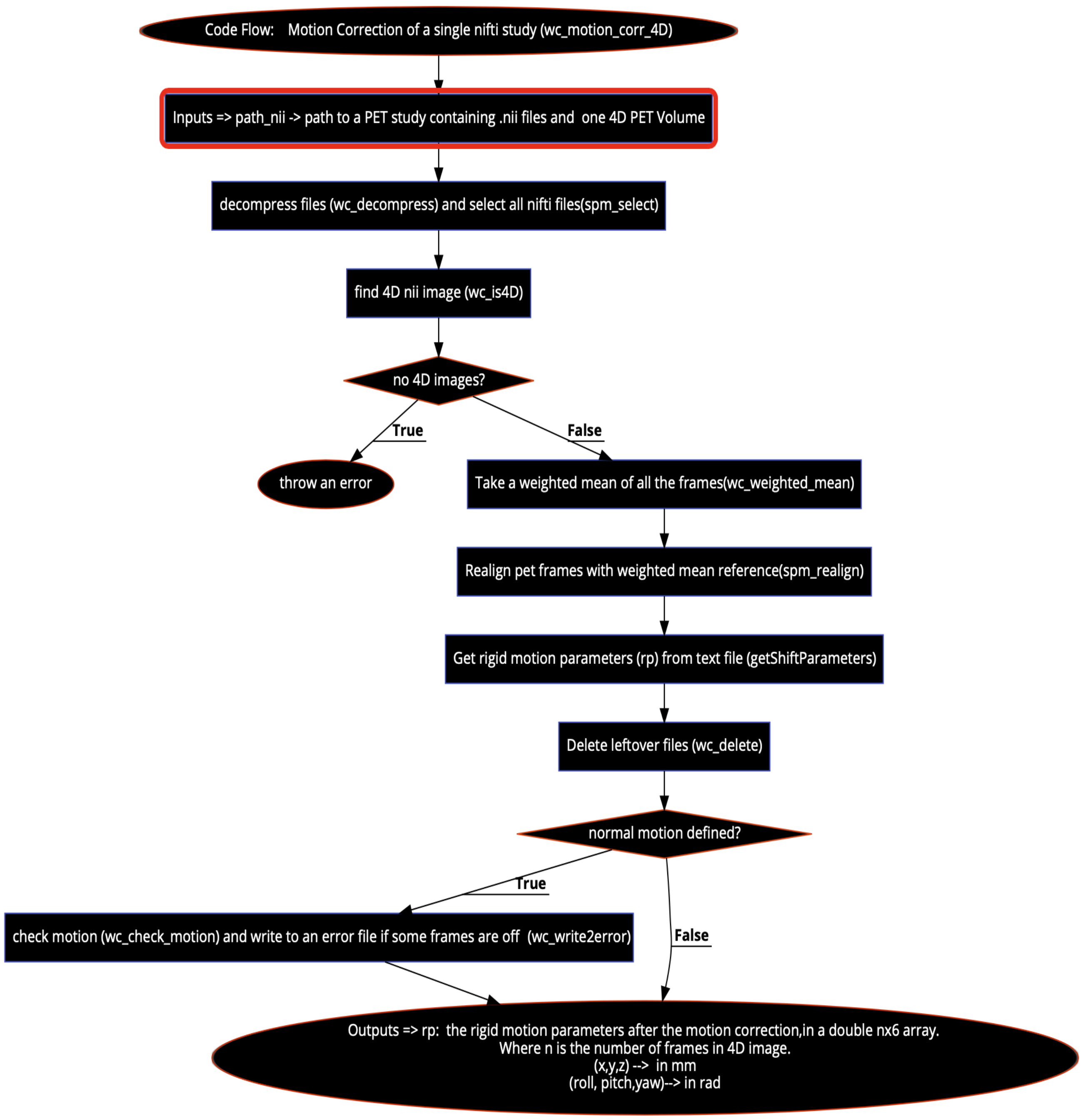
- b. nii_path: directory to save nii files to.
- c. doAmideReslice : T/F boolean on whether to reslice dicom images to overlay with nifti in amide. Not recommended if dont want to change original dicom images. However, code in wc_dicom2origin could be improved to account for this, i.e. output to a different directory.[default: false]

3. File hiearchy:
   - a. wc_dicom2nii
     - i. get_headers
       1. wc_getdirfids
       2. spm_dicom_headers
     - ii. convert2nii
       1. spm_dicom_convert
       2. spm_file_merge
       3. makeHTMLheader
          - a. makefileName
          - b. dicominfo
          - c. print2html
       4. tar /gzip
     - iii. wc_dicom2origin
       1. wc_getdirfids
       2. spm_dicom_headers
       3. dicominfo
       4. dicomread
       5. dicomwrite
4. Output:
   - a. tarfid: cell array of tar files which contain 3D nii vols and an html file which displays dicom header info from a single slice of the volume.

iv. Motion correction of a single nifti study

1. Description: Checks the alignment of nii 4D files. Will write files which are bad to an error catch .csv file. If any of the motion parameters are beyond the accepted movement amount defined by xyzrpy.mat from wc_def_norm_movement. Will not check if it is aligned if xyzrpy.mat file doesn't exist with the variable xyzrpy 1x6 double array.
2. Inputs:
   - a. path_nii : path to a PET study containing .nii files and one 4D PET nii file

b. opts: a struct definng the possible options for motion checking;
   i. opts.nb: number of acceptable studies to for defining normal parameters [default: 10]


Code Flow: Motion Correction of a single nifti study (wc_motion_corr_4D)
Inputs => path_nii -> path to a PET study containing .nii files and one 4D PET Volume
decompress files (wc_decompress) and select all nifti files(spm_select)
find 4D nii image (wc_is4D)
no 4D images?
True
throw an error
False
Take a weighted mean of all the frames(wc_weighted_mean)
Realign pet frames with weighted mean reference(spm_realign)
Get rigid motion parameters (rp) from text file (getShiftParameters)
Delete leftover files (wc_delete)
normal motion defined?
True
check motion (wc_check_motion) and write to an error file if some frames are off (wc_write2error)
False
Outputs => rp: the rigid motion parameters after the motion correction,in a double nx6 array.
Where n is the number of frames in 4D image.
(x,y,z) --> in mm
(roll, pitch,yaw)--> in rad

   ii. opts.stdev: number of standards of deviations the rotation parameters can be before writing to the error file. [default: 3]
   iii. opts.percent: percent of the rigid motion parameters [default: 10]

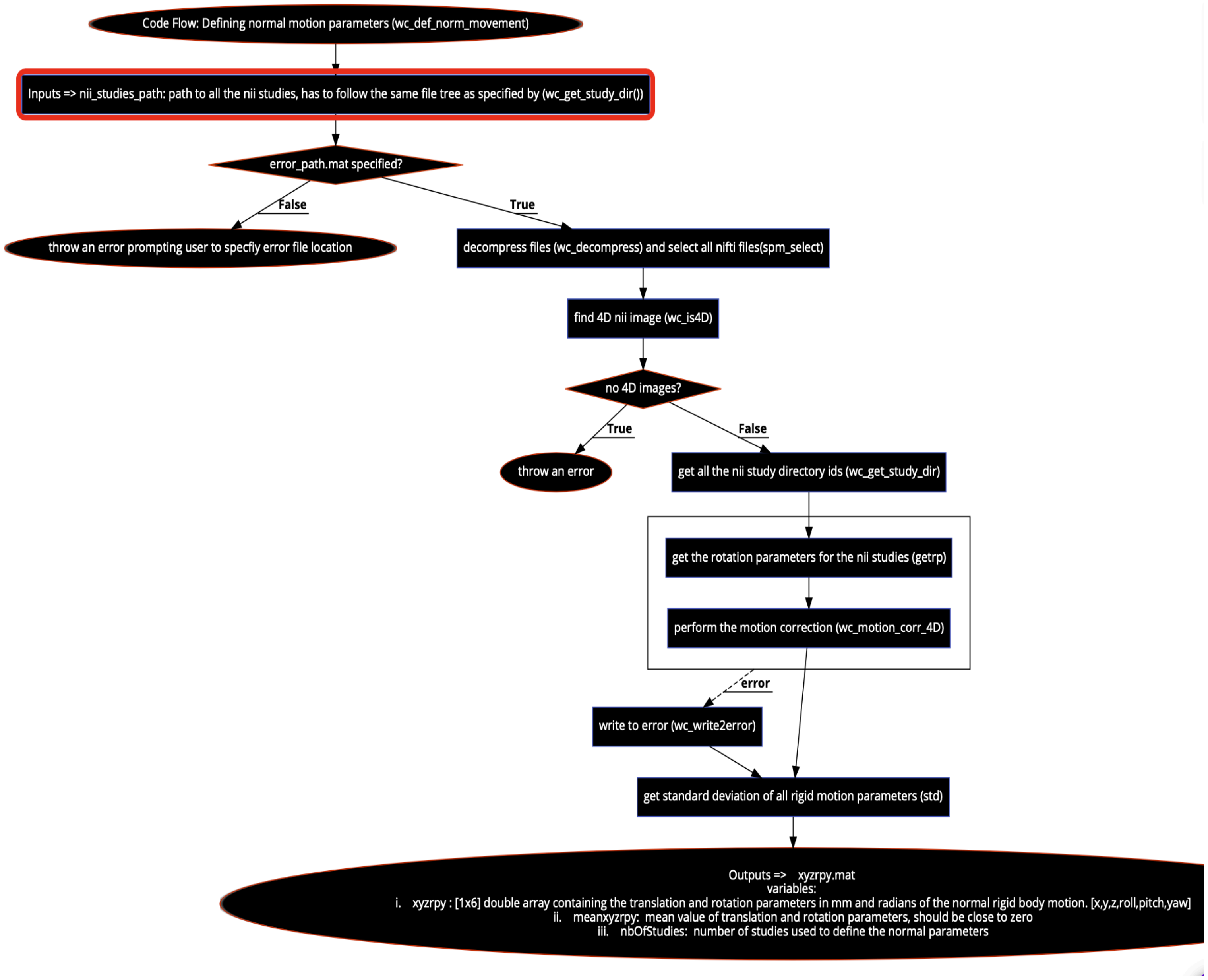
            iv. opts.doMotion: a T/F value in which true means it will check the motion of the rigid motion parameters against the normal values. (wc_check_motion). This value is mainly here if running wc_def_norm_movement and xyzrpy.mat is already defined. [Default:true]
            v. opts.xyzrpy_path: path to xyzrpy .mat file,[default: will just look to matlab path]

3. Function hierarchy
    a. wc_motion_corr_4D
        i. wc_decompress
            1. wc_getFids
        ii. spm_select
        iii. wc_is4D
        iv. wc_weighted_mean
            1. spm_vol
            2. spm_select
            3. writeExp
            4. spm_imcalc
        v. spm_realign
        vi. getShiftParameters
            1. spm_fileparts
            2. dlmread
        vii. wc_check_motion
            1. wc_check_xyzrpy
            2. wc_write2error
        viii. wc_delete
4. Outputs:
    a. rp: the adjustment parameters after the motion correction,in a double nx6 array. Where n is the number of frames
        i. (x,y,z) --> in mm
        ii. (roll, pitch,yaw)--> in rad

v. Defining normal motion parameters

1. Description: define the normal movement across many nii studies by finding the standard deviation of movement parameters in the x,y, and z direction of returned from wc_motion_corr_4D across multiple studies.
2. Inputs:
    a. nii_studies_path: path to all the nii studies, has to follow the same file tree as specified by wc_get_study_dir()
3. Function Hierarchy
    a. wc_def_norm_movement

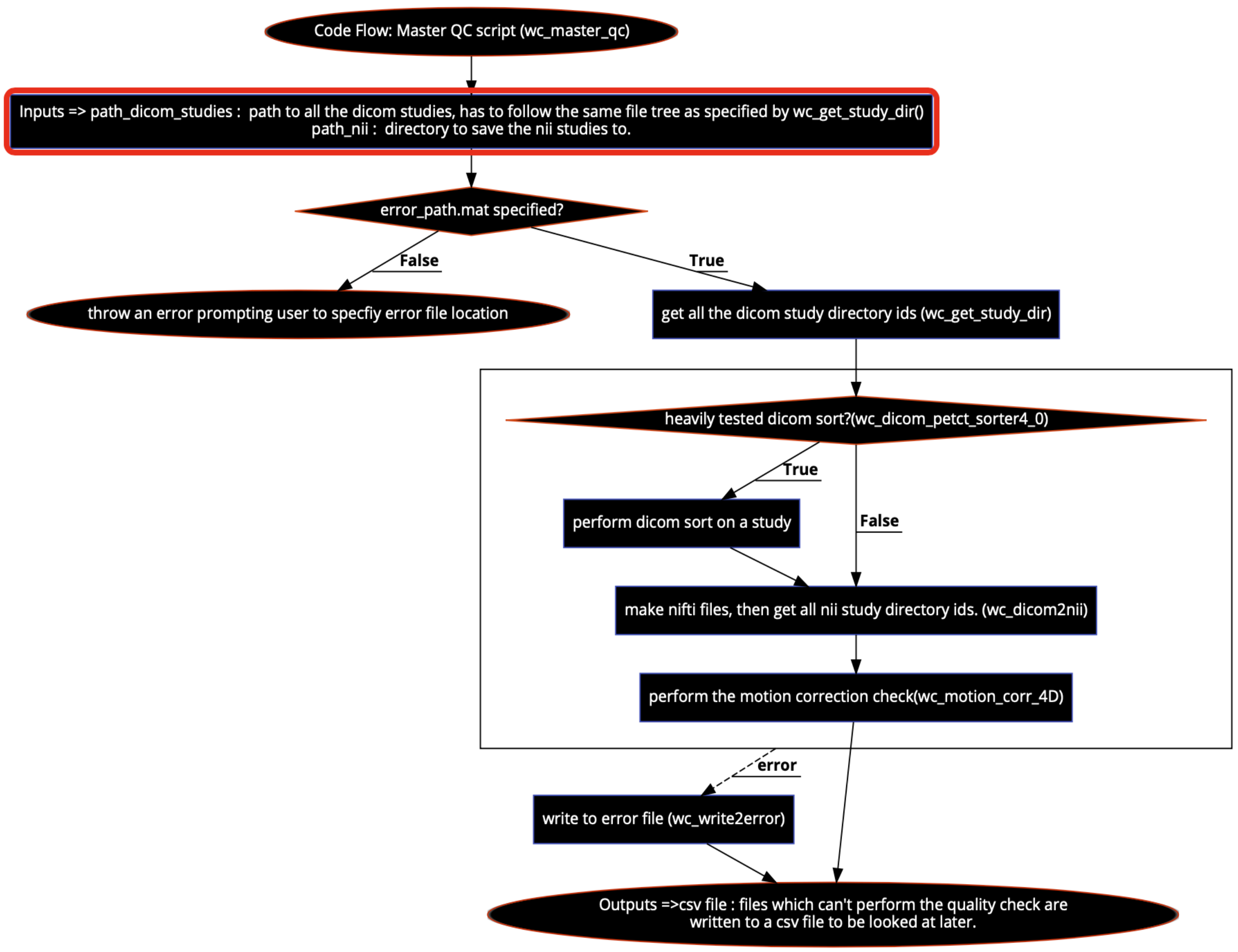
i. wc_check_error_path
    1. wc_isdicom
ii. getrp
    1. wc_motion_corr_4D
    2. wc_write2error
iii. std


Code Flow: Defining normal motion parameters (wc_def_norm_movement)
Inputs => nii_studies_path: path to all the nii studies, has to follow the same file tree as specified by (wc_get_study_dir())
error_path.mat specified?
False
True
throw an error prompting user to specfiy error file location
decompress files (wc_decompress) and select all nifti files(spm_select)
find 4D nii image (wc_is4D)
no 4D images?
True
False
throw an error
get all the nii study directory ids (wc_get_study_dir)
get the rotation parameters for the nii studies (getrp)
perform the motion correction (wc_motion_corr_4D)
error
write to error (wc_write2error)
get standard deviation of all rigid motion parameters (std)
Outputs => xyzrpy.mat
variables:
i. xyzrpy : [1x6] double array containing the translation and rotation parameters in mm and radians of the normal rigid body motion. [x,y,z,roll,pitch,yaw]
ii. meanxyzrpy: mean value of translation and rotation parameters, should be close to zero
iii. nbOfStudies: number of studies used to define the normal parameters

4. Outputs:
    a. xyzrpy.mat

i. xyzrpy : [1x6] double array containing the translation and rotation parameters in mm and radians of the normal rigid body motion. [x,y,z,roll,pitch,yaw]
ii. meanxyzrpy: mean value of translation and rotation parameters, should be close to zero
iii. nbOfStudies: number of studies used to define the normal parameters

vi. Master QC script


Code Flow: Master QC script (wc_master_qc)
Inputs => path_dicom_studies : path to all the dicom studies, has to follow the same file tree as specified by wc_get_study_dir()
path_nii : directory to save the nii studies to.
error_path.mat specified?
False
True
throw an error prompting user to specfiy error file location
get all the dicom study directory ids (wc_get_study_dir)
heavily tested dicom sort?(wc_dicom_petct_sorter4_0)
True
False
perform dicom sort on a study
make nifti files, then get all nii study directory ids. (wc_dicom2nii)
perform the motion correction check(wc_motion_corr_4D)
error
write to error file (wc_write2error)
Outputs =>csv file : files which can't perform the quality check are written to a csv file to be looked at later.

1. Descripton: this is the master script to perform all quality control

checks across studies. Make sure you have the proper dicom file tree for your dicom studies as shown in wc_get_study_dir (below). error_path.mat should also contain a character vector with one variable name error_path with where the files that contain errors should be written. Prior to xyzrpy.mat should also contain a variable xyzrpy contianing the normal motion defined by wc_def_norm_movement

2. Inputs:
    a. path_dicom_studies : path to all the dicom studies, has to follow the same file tree as specified by wc_get_study_dir()
    b. path_nii : directory to save the nii studies to.
    c. opts: a structure containing flags on what qc processes to do and paths to .mat files if necessary.
        i. ->opts.doMotion: boolean TF saying whether to do a motion correction. [default: true]
        ii. ->opts.stdev: the amount of standard deviation that is acceptable from normal parameters without flagging. [default: 3]
        iii. ->opts.dosort: boolean saying whether the function wc_dicom_petct_sorter4_0, to sort the metadata should be called, as it may contain bugs and errors. [default: false]
        iv. ->opts.headerSort: how much of header to display, look to wc_dicom_petct_sorter4_0 for options. [default: no inputs which is defaults in wc_dicom_petct_sorter4_0]
        v. ->opts.error_path: path to error_path.mat file [default: will look to matlab path]
        vi. ->opts.xyzrpy_path: path to xyzrpy .mat file,[default: will just look to matlab path]
        vii. opts.doAmideReslice : T/F boolean on whether to reslice dicom images to overlay with nifti in amide. Not recommended if dont want to change original dicom images. However, code in wc_dicom2origin could be improved to account for this, i.e. output to a different directory.[default: false]

Function hierarchy
    d. wc_master_qc
        i. wc_check_xyzrpy
        ii. wc_check_error_path
        iii. wc_get_study_dir
            1. wc_isdicom
        iv. dcmsort

1. wc_dicom_petct_sorter4_0
2. wc_write2error

v. makeNii
1. wc_dicom2nii
2. wc_write2error
3. wc_get_study_dir

vi. motionCorr
1. wc_motion_corr_4D
2. wc_write2error

3. Output: csv file : files which can't perform the quality check are written to a csv file specified by error_path.mat

b. Description of helper functions:

i. **wc_dicom2origin:** Put dicom image at the AMIDE origin for the x,y, and z coordinates and flip left-right the pixel data of each slice. This will cause the dicom image to overlay with the converted nifti image when it is in AMIDE. Look to function to view the "theory" on it.
1. input:
   a. dicom_path: path to all the dicom files of one study
2. output: new dicom slices with changed ImagePositionPatient field and pixel data flipped.

ii. **wc_check_motion:** Will take in the rigid motion parameters from single dynamic pet study, check them against the normal defined motion parameters from xyzrpy.mat, and then flag any frames that have more than the accepted amount of motion.
1. Input:
   a. rp: nx6 double array with rigid motion parameters produced by spm_realign from wc_motion_corr_4D.
   b. P: a character array of spm file ids(spm_select). See wc_weighted_mean for example.
   c. opts: a struct definng the possible options for error, see wc_check_xyzrpy file for more description.
2. output:
   a. csv file produced by wc_write2error that flags down which frames have larger than the accepted amount of motion.

iii. **wc_check_xyzrpy**: checks the normal motion parameters to see if they are feasible values. Also defines how many standards of deviation the xyzrpy can be. xyzrpy.mat must also be on the matlab path.
1. inputs:
   a. opts: a struct definng the possible options for error checking
      i. opts.nb: number of acceptable studies to for defining normal parameters [default: 10]

            ii. opts.stdev: number of standards of deviations the rotation parameters can be before writing to the error file. [default: 3]
            iii. opts.percent: percent of the rigid motion parameters that the mean can be from zero, [default: 10]
            iv. ->opts.xyzrpy_path: path to the xyzrpy.mat file, if not specified file must be on matlab path.
    2. output:
        a. xyzrpy: 1x6 double array of the accepted standard deviation for each parameter

iv. **wc_check_error_path**: a function to check the error path specified by error_path.mat. error_path.mat must contain a single variable error_path which has the path to where to store the error catch csv file from wc_write2error.
    1. Inputs: opts: either a struct containting a field name 'error_path' which has the directory to which to store the error_path.mat file to, or a character vector containing the directory. If not input it will just look at the matlab path. [default: will look to matlab path]

v. **wc_decompress :** decompresses files which are tarred or gzipped.
    1. input
        a. file_path: the path to where the files are located [default:pwd]
        b. nb: the nb of files that can can be gunzipped, [default: 6]
        c. fids: instead of using wc_getFids(), specify the files to decompress.
    2. output: NONE

vi. **wc_delete:** does a delete of all specified ending files. returns back a cell array of files it just deleted.
    1. NOTE: BE VERY CAREFUL WITH THIS FUNCTION, COULD LEAD TO DELETING OF USEFUL DATA
    2. input:
        a. path_file: directory to which to delete file
        b. ext: file extensions which to delete [default: '.html','.nii','.tar','.txt','.mat']
        c. nb: number of files allowed to delete [default:19] , safety check so you don't loose everything in the case of a bad call.
    3. output:
        a. badfids : cell array of files just deleted

vii. **wc_get_study_dir:** gets the study directories from each dicom study, also checks to make sure it is in the following format:

ParentDirectory/Study1/

```
./Study2/
./Study3/
.
.
.
./StudyN/
```

***Will throw an error if not in this format so be careful.
*The choice to use this file organization was made as it makes sure the
*user has studies properly organized and not overlaping.

1. input:
   a. path_studies: a character vector specifying the path to all dicom studies.
2. output:
   a. dirids: A cell array of character vectors specifying the path to each study under path_studies

viii. **wc_getFids:** recursively gets all the files in a give directory, including files in subdirectories.
   1. input
      a. directory: directory of all the files which would like to get file ids
   2. output
      a. fids: all file ids in that directory

ix. **wc_is4D:** get the 4D files from a cell array of files, will return a logical array of the files that are 4D. Will throw an error if no 4D files are found, or more than one 4D file is found. (Uses metadata to determine dimension.)
   1. input:
      a. fids: cell array of character vectors of file ids
   2. output
      a. fid4D: logical array of which file is a 4D nii file

x. **wc_isCT:** takes in a path to a nii study and then returns a logical array of which files are 3D CT images.
   1. input
      a. path_nii: path to the nii study
   2. output
      a. CT: logical array of which files are 3D CT images.

xi. **wc_isdicom:** sees which files of the files ids are dicom files. by looking at the metadata in the dicom file.
   1. input
      a. fids : cell array of file ids : no directories
   2. output

a. dicoms: logical array of which files are dicoms

xii. **wc_weighted_mean:** computes the weighted mean of dynamic PET series, then produces a nifti image of it

1. input
    a. fid: file id of the 4D pet file which a sum is to be taken of. Given in the form of a single character vector.
2. output
    a. frameRates: A column vector of doubles defining the frame rates for each frame (in seconds). The numel must be the same as the number of frames, otherwise, an error will be thrown. [Default: 'PiB' ~ 17 frames; first 5 @ 2mins, last 12 @5mins]

xiii. **wc_write2error:** writes to a csv file a list of files which data was not processed correctly, as opposed to writing an error to the user. Can be useful when processing large amounts of data.(try -- catch statements)

1. input:
    a. fids: file ids in a cell array of character vectors
    b. errorStruct: MException object
    c. msg: message to display on error file as to why those files failed if known. [Default: 'Following files were not processed properly:']
    d. error_path: path to the directory to save the error file to as a charactor vector. [Default: finds error_path.mat file and uses that directory]
    e. fileName: name of error file as a character vector. [Default: errorcatch.csv]
2. output
    a. csv file specifying the errors that occured for specific files. Saved to path specified by error_path.mat file.

V. Summary/Future

a. Summary

i. These functions began the development of automated Quality Control check for the Horizon Scanner at the Waisman Center. The quality control check consisted of three main parts: 1. Sorting of dicom metadata from unique series slices 2. Conversion of dicom slices into nifti volumes which overlay in Amide, with an associated html header file from an associated dicom slice 3. Check dynamic PET series to see if any frames had a large amount of motion, as defined by the standard deviation from many dynamic PET images. These automated quality checks are designed to help catch any errors that could occur from the a large amount of PET data produced by the horizon scanner.

b. Drawbacks

        i. However, these quality checks are not designed to correct nor to always correctly identify errors, they merely flag images which look to have an error in preprocessing. This filters down the number of images the user will have to look at during inspection which reduces the amount of man power needed for a quality control check.

c. Future
    i. Bugs in current program
        1. Program can be adjusted to work for MK radiotracer, currently cannot convert any as it conflicts with the PiB dicom to nifti conversion, due to file hierarchy confliction from spm_dicom_convert, outputs both tracers to the same nifti study folder.
        2. Use slover (in SPM) to overlay the CT and a single PET modality and then display the associated 2D image volume in the header html file. Make sure to view all slices.
        3. Adjust the wc_dicom2nii code to find if a slice is apart of a 3D volume, there must be a fieldname which gives the dimensions of the pixel data, just couldn't find it. [Currently using isfield('SliceThickness') to determine whether the slice is 3D. If a 2D field contains this such as the Topogram it will still convert.]
        4. Could find a way to automatically email the error csv file to the user so they can just get an email the next morning.
        5. Normal motion parameters were only done for random PiB studies, not ones known to have little motion. Also, only 11 studies were used. (maybe try spm_smooth to counteract the low study count)
        6. Edit code for wc_dicom_petct_sorter4_0 , it is hard to read and even harder to edit. Many more functions need to be included. Also more bug checking for errors.
        7. Code for sort function and wc_dicom2origin to change the output directory of the csv and dicom files.
    ii. Additions to program
        1. Algorithm for if there is motion is quite weak.
        2. Make an algorithm to determine how good overlay is between CT and PET modalities, or use another method.
        3. Make a template for horizon scanner so that can easily extract ROI for qc purposes. This is an email from tobey betthauser describing the starting steps for the template:
        4. Brad just mentioned something about implementing that into his downs syndrome research? Maybe bring up next time you start to develop the project?

At Aug 23,2019 10:23am Tobey wrote:

As for the template, this should be a template for many subjects. The idea is that we can have a common reference space for each tracer that will allow some quick QC measures using only PET data that do not rely on needing additional anatomical data (e.g. MRI or CT). The template image for each tracer could be made in different ways, but here is one thought.

1) Perform initial QC of alignment between CT and PET images
2) If those are in alignement, Use the CT to get a rough segmentation of the brain (SPM has tools for this)
3) Intensity normalize the PET data to the whole brain (i.e. divide the average activity concentration in the PET image by the average concentration in the entire brain ROI generated from the CT)
4) Spatially normalize the PET images to a template space (use CT to guide the spatial normalization to the template space, alternatively we could register the PET to MRI and use that, but probably not needed for this purpose)
5) Perform 1-4 in scans from many subjects
6) Take the mean of all the images that should now be in template space.

· Alternatively, you could just create an SUV image (PET image divided by injected activity x body weight) and use that to create the template.

The following paper from one of Brad's previous graduate students should also give you some extra information about how to create a PET template:
https://www.ncbi.nlm.nih.gov/pubmed/29752653 // very helpful source

end EMAIL

Possible Additions to QC program:

1. Algorithm for if there is motion is quite weak.
2. Make an algorithm to determine how good overlay is between CT and PET modalities, or use another method.
3. Make a template for horizon scanner so that can easily extract ROI for qc purposes. This is an email from tobey betthauser describing the starting steps for the template:
4. Brad just mentioned something about implementing that into his downs syndrome research? Maybe bring up next time you start to develop the project?