hMRI toolbox – Import & Filters

Responsible developer: Evelyne Balteau
Group members: Martina Callaghan, Siawoosh Mohammadi, Antoine Lutti, Karsten Tabelow, Nikolaus Weiskopf, Lars Ruthotto.

1. DICOM to nifti conversion – hMRI_dicom_convert

hMRI_dicom_convert is based on spm_dicom_convert, and works identically if either *nii* or *img* format is used. The behaviour changes if *nii+* format is specified, where *nii+* is the standard nifti format with extended header. Note that the file extension is still **.nii* and that the extended nifti image format is compatible with standard nifti readers (MRIcron, SPM, FSL…).

Example:

```
files2convert = spm_select(Inf,'^*.IMA');
dcmhdr = spm_dicom_headers(files2convert);

files_converted = spm_dicom_convert(hdr,'all','flat','nii');
% is identical to:
files_converted = hMRI_dicom_convert(hdr,'all','flat','nii');
% but not to (extended header included):
files_converted = hMRI_dicom_convert(hdr,'all','flat','nii+');
```

2. The extended header structure

FORMAT OF EXTENDED HEADER

- extended headers are simple Matlab structures – hence quite flexible and modular
- the Matlab structure is converted into JSON (basically a text format transcription of the Matlab structure) to be stored in the nifti header
- the JSON transcription is easily readable and searchable if you open the nii file with a text editor
- JSON string ↔ Matlab structure conversion is easily done using JSONLAB (https://github.com/fangq/jsonlab.git)
- nifti files with extended header can be read without any trouble with any nifti reader.

CONTENT AND STRUCTURE OF EXTENDED HEADER

The structure is divided into the following two branches:

- **Acquisition parameters** (**hdr.acqpar**): this branch contains the *original DICOM header*. It should be kept unchanged and is created when importing DICOM images to nifti. Might be dropped once the processed image relies on several input images (e.g. when creating a map from mtflash and b1mapping acquisitions). Note that for confidentiality, the patient name, date of birth and DICOM filename (often containing the patient name) have been

removed, while patient age (years), sex, size and weight are kept. This can be modified one way or another. It is the basic, minimalist anonymization I've come up with for now.

- **History** (**hdr.history**): is a *nested* structure containing
  - **procstep**: a structure describing the current processing step and containing:
    - **descrip**: a brief description of the processing applied (e.g. dicom to nifti import, realign, create map, …)
    - **version**: version number of the processing applied (in github, retrieving the version number is not straightforward, needs further discussion)
    - **procpar**: processing parameters (relevant parameters used to process the data – e.g. for fieldmap-based EPI undistortion: EPI readout duration, TEs of the field mapping acquisition, ...). Basically all the parameters included into the matlab batch "job" *except from the input images*.
  - **input**. A cell array listing the input images used for the processing. Each input (hdr.history.input{i}) is a structure containing:
    - **filename**: the filename of the input image
    - **history**: the history structure from the input image
  - **output**: a structure containing
    - **imtype**: image type of the current output (e.g. R1, FA, MT, ADC, …)
    - **units**: either physical units (sec, 1/sec, …), standardized units (e.g. percent units = p.u.) or arbitrary units (a.u.)…

3. hMRI toolbox's version number

For traceability, it is essential to include information on the toolbox's version in the header of processed images. This is not as straightforwardly done as with Subversion, where a centralised repository is used. However, it is possible to retrieve the SHA1, author, date and message of the last commit in the current branch of the current repository. The following script:

```
version = hMRI_get_version;
```

returns this information as a string:

```
commit e7e8b3ff7755ba59df363996455be985268ea711
Author: ebalteau <e.balteau@ulg.ac.be>
Date:   Tue May 10 22:47:59 2016 +0200

    Initial commit

    Including:
    - reading and tidying DICOM header
    - read/write extended header
    - dicom to nifti conversion with extended header
```

and is used when filling up the version in the history subfield of the extended header (see *Content and Structure of extended header* above).

Since this applies not only the Import & Filters section but to the whole hMRI toolbox, the script hMRI_get_version will be located in the root directory of the hMRI toolbox.

hMRI_get_version calls the git command using the MATLAB-git wrapper from https://github.com/manur/MATLAB-git.git. The latter allows you to use the command line git instructions in Matlab (as long as Git is installed on your computer!). Make sure that the git.m script is in the Matlab path te excecute this script.

4. How to read/write extended headers

The extended header can be *read* using the function hMRI_get_extended_hdr (type help hMRI_get_extended_hdr in Matlab for detailed syntax). The function returns the Matlab structure described above.

The function hMRI_set_extended_hdr (type help hMRI_set_extended_hdr in Matlab for detailed syntax) allows you to *write* (insert, modify, overwrite) the extended header into a nifti file. In general, the extended header is initialized during the DICOM to nifti convertion and further modified as the processing progresses.

Example 1: modify an existing header

NB: a very basic example, not the most common usage of hMRI_set_extended_hdr but should help getting familiar with handling extended headers...

```
% select nii file with (or without) extended header
niifile = spm_select(1,'^*.nii');
% read the extended header
hdr = hMRI_get_extended_hdr(niifile);

% modify the header of an existing nii file
modif_hdr = hdr{1};
modif_hdr.history.procstep.descrip = 'original file with modified header';
modif_hdr.history.procstep.version = hMRI_get_version;
modif_hdr.history.procstep.procpar = [];
modif_hdr.history.input{1}.filename = niifile;
modif_hdr.history.input{1}.history = hdr{1}.history;
modif_hdr.history.output.imtype = hdr{1}.history.output.imtype;
modif_hdr.history.output.units = hdr{1}.history.output.units;
hMRI_set_extended_hdr(niifile, modif_hdr);
```

Example 2: save a newly computed R1 map with extended header

NB: I have to admit that I haven't tried this example on real data. So it might need some debugging and updating as the implementation of the hMRI toolbox will progress with the map generation part. Use it (cautiously) as a (hopefully good) starting point :).

```
% initialize nifti object with the map's file name
Ni = nifti('map_file_name.nii');
```

```
% ... fill up the fields and data section of the nifti object as usual...
create(Ni);
% create a new structure for the extended header
output_hdr = struct('history', struct('procstep',[],'input',[],'output',[]));
% NB: since it is no longer an original image from the scanner, the field
% 'acqpar' is dropped.
output_hdr.history.procstep.descrip = 'map creation';
output_hdr.history.procstep.version = hMRI_get_version;
hdr.history.procstep.procpar = struct; % ... insert job parameters here
for cinput = 1:number_of_input_images
    % assuming input_file_name is a list of files used for the processing:
    output_hdr.history.input{cinput}.filename = input_file_name(cinput,:);
    input_hdr = hMRI_get_extended_hdr(input_file_name(cinput,:));
    output_hdr.history.input{cinput}.history = input_hdr{1}.history;
end
output_hdr.history.output.imtype = 'R1 map';
output_hdr.history.output.units = '1/sec';
hMRI_set_extended_hdr('map_file_name.nii', output_hdr);
```

5. How to retrieve specific parameters from the extended header

This section deals with the *acqpar* subfield of the extended header, i.e. the parameters retrieved from the DICOM header.

Since the extended header is a structure, searching for a specific parameter is often equivalent to searching for the corresponding field name. For that reason, the function findFieldName has been implemented to recursively search for a specific field name (exact of partial match, case sensitive or not). The “hMRI_get_extended_hdr_val” script makes use of findFieldName to sort out the parameters and return the desired value. More refined versions of “hMRI_get_extended_hdr_val” will develop with time to deal with hardware/software/sequences variations – if possible, while the front end of “hMRI_get_extended_hdr_val” should keep unchanged.

Below are listed a series of common parameters and how they can be retrieved from the extended header. Note that these examples will need to be adapted according to the version of the MRI hardware, software and sequences.

Retrieve the whole header

```
hdr = hMRI_get_extended_hdr('my_extended_nifti_file.nii');
```

Excitation flip angle [deg]

```
fa = hMRI_get_extended_hdr_val(hdr{1},'FlipAngle');
```

Repetition time [ms]

```
TR = hMRI_get_extended_hdr_val(hdr{1},'RepetitionTime');
```

Echo time(s) [ms] – and array of TEs if multi-echo sequence

```
TE = hMRI_get_extended_hdr_val(hdr{1},'EchoTime');
```

Magnetization transfer (MT) pulse = 1/0 according to pulse switched on/off

```
MT = hMRI_get_extended_hdr_val(hdr{1},'MT');
```

Field strength B0 [T]

```
B0 = hMRI_get_extended_hdr_val(hdr{1},'FieldStrength');
```

Center frequency [Hz]

```
freq = hMRI_get_extended_hdr_val(hdr{1},'Frequency');
```

Protocol Name

```
ProtName = hMRI_get_extended_hdr_val(hdr{1},'ProtocolName');
```

Sequence Name

```
SeqName = hMRI_get_extended_hdr_val(hdr{1},'SequenceName');
```

RF spoiling phase increment (warning: sequence dependent, see comments in the code)

```
RFSpoilIncr =
        hMRI_get_extended_hdr_val(hdr{1},'RFSpoilingPhaseIncrement');
```

All WIP parameters (returns a structure with two fields alFree and adFree containing arrays of long and double values respectively)

```
WIP = hMRI_get_extended_hdr_val(hdr{1},'WipParameters');
```

Parameters for EPI undistortion

- Short and long echo times from dual-echo field mapping:

```
TE = hMRI_get_extended_hdr_val(hdr_gre_field_mapping{1},'EchoTime');
```

- EPI phase encoding direction (ROW/COL)

```
PEDir =
      hMRI_get_extended_hdr_val(hdr_epi{1},'PhaseEncodingDirection');
```

- EPI phase encoding direction positive (e.g. A>>P is positive (1), P>>A is negative (-1))

```
PEDirPos =
  hMRI_get_extended_hdr_val(hdr_epi{1},'PhaseEncodingDirectionSign');
```

- Total EPI readout duration (warning: tricky for home-made sequences with uncompleted headers)

```
epiROdur = hMRI_get_extended_hdr_val(hdr{1},'epiReadoutDuration');
```

- other parameters should be kept hard coded – I think

Parameters for processing B1 maps (al_b1mapping)

- Nominal FA values (betas)

```
beta = hMRI_get_extended_hdr_val(hdr{1},'B1mapNominalFAValues');
```

- Mixing time (TM) – NB: not sure how the TM relates to the TEs in the sequence (in our case – Prisma/Liège, TM = 33.8 ms while TE1 = 39.06 and TE2 = 19.53 ms…?? Although TE2 doesn't seem to be used in the sequence – maybe it is a 'dummy' TE due to the 2-contrasts setting?

```
TM = hMRI_get_extended_hdr_val(hdr{1},'B1mapMixingTime');
```

- Number of nominal values to use for each voxel: must be either hard coded or a user-dependent parameter in the GUI or derived from the data themselves (quality, SNR, …?)

DWI parameters

The diffusion parameters can be retrieved following two distinct ways.

1. reading all the directions and b-values at once from CSASeriesHeaderInfo subfield

> *NB: the current implementation assumes that Free diffusion mode is used, with user-defined set of diffusion directions.*

- Diffusion directions: list of directions as defined in the *.dvs file (normalised or not)

```
DiffDir =
          hMRI_get_extended_hdr_val(hdr{1},'AllDiffusionDirections');
```

- b-values: list of b-values as defined in the Diff tab of the UI

```
bVal = hMRI_get_extended_hdr_val(hdr{1},'AllBValues');
```

2. reading the individual directions and b-values for each image from CSAImageHeaderInfo subfield

> *NB: this implementation is expected to work more generally. I've noticed that the sign of the y and z components of the direction vector have their sign inverted, does it rings a bell to anybody why it is so? Or has anybody noticed they needed to invert signs in their processing (it actually rings a bell to me as something fellows mentioned to me, but I'm not processing much DWI myself). The respective orientation of patient/image/scanner axes is probably relevant too :(… Help from DWI expert required here ;)!*

- Diffusion direction: normalised vector

```
DiffDir = hMRI_get_extended_hdr_val(hdr{1},'DiffusionDirection');
```

- b-values: corresponding b-value

```
bVal = hMRI_get_extended_hdr_val(hdr{1},'BValue');
```

There is, to my knowledge, no way to retrieve the δ, Δ and G values describing the timing and amplitude the diffusion gradients :(...

Other parameters – TODO

A few parameters have been mentioned that are obviously not available in the header. These parameters will have to either stay hard-coded or made user-dependent through the batch processing GUI (tbx_cfg_* script and sub-scripts). Other parameters have been mentioned that are not clear to me, and probably other parameters are still missing. So this list can be considered as a TODO list...

- PD map calculation: calibration value taken to convert the flattened signal intensity maps into PD values. Set to 69 in current MTprot.m (from P. Tofts book): Y=Y/mean(A_WM(A_WM~=0))*69;

- asd (mentioned by Antoine) ??

- ...

6. Dependencies and list of files

DEPENDENCIES

- SPM(12): spm_dicom_headers to read the DICOM header
- JSONLAB: https://github.com/fangq/jsonlab.git for JSON – Matlab structure conversion of the extended header
- MATLAB-git: https://github.com/manur/MATLAB-git.git to use the command line git instructions in Matlab (as long as Git is installed on your computer!). Used by hMRI_get_version (see above).

LIST OF FILES IN DIRECTORY "FILTER"

Main scripts

- hMRI_dicom_convert
- hMRI_get_extended_hdr
- hMRI_set_extended_hdr
- hMRI_get_extended_hdr_val

Utilities scripts

- eb_read_ASCII
- eb_read_phoenix
- eb_spm_tidycsa
- findFieldName
- get_numaris4_val
- init_extended_hdr

ADDITIONAL FILE IN TOOLBOX's ROOT DIRECTORY

- hMRI_get_version