Technical Note

Working with DICOM and TAG files in MATLAB

Davi Marco Lyra Leite

www-scf.usc.edu/∼leite

University of Southern California, Los Angeles, USA

May 22, 2014

1 Introduction

In studies where the researcher is not responsible for the data acquisition, sometimes it is difficult to have access to raw data. In these cases, people might have to perform their work directly using the DICOM and TAG files. Hence, it is important to learn how to input DICOM and TAG files into MATLAB in order to perform the desired tasks correctly.

This tutorial aims to explain how to read and organize the datasets for both the images and tags. It also explains how to sort the data in order to match the object of study. In addition, attached to this tutorial, there is a function called `tagread`, used to read the *.tag files.

Section 2 explains how to work with DICOM files, reading the images and necessary, and sorting the stacks of slices from different files. Section 3 deals with TAG files and provides a general framework for sorting both tags and slices.

2 Working with DICOM files in MATLAB

MATLAB has built-in functions to read DICOM files and obtain header information. Therefore, it is not necessary to download new packeges to perform these taks. The functions are `dicomread`, and `dicominfo`. Their usage is straightforward and is described below.

2.1 Reading DICOM files in MATLAB

In order to read the DICOM files to obtain the images, the following syntax is used:

```
imagename = dicomread(filename);
```

in this command, *imagename* receives the gray-scale DICOM image in *int16* precision, presenting the matrix size defined by the scan protocol. This matrix contains only magnitude information of the image from the DICOM, without the header.

Figure 1 shows a DICOM image read into MATLAB using `dicomread`. It is possible to see the lack of header information like scan protocol and patient's data, differing from a common DICOM viewer such as ImageJ or Osirix.

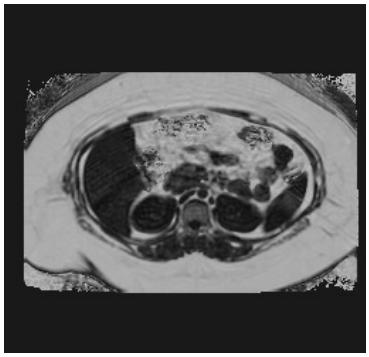

Figure 1: Example of a DICOM image read in MATLAB.

2.2 Reading DICOM header in MATLAB

In order to obtain the header information from a DICOM file, we use another function called `dicominfo`. The syntax is as follows:

```
header = dicominfo(filename);
```

The variable *header* is a struct containing the other information from the DICOM file: slice loaction, protocol, patient's data, etc. It is extremely useful to have this data when working with multiple DICOM files corresponding to a 3D volume, because the file's names might not correspond to their spatial position.

For example, if one wants the slice location, the person just have to type the following line:

```
header.SliceLocation
```

and hit return. The output will be something like:

```
>> header.SliceLocation
ans =
    -14.8420
```

For a list of the elements in the struct, one can write `header` and hit return, or enter `header.` and hit tab to see a list of possible completions.

2.3 Sorting 3D volumes from multiple DICOM files

As described previously, the names of DICOM files may not correspond to their saptial location. Therefore, if someone wants to read correctly a 3D volume in MATLAB, it is necessary to sort the slices. In order to perform this task, it is possible to use the following script:

```
% Initializing datasets
unsortedsliceposition = zeros(numberofslices,2);
unsortedstack = zeros(yvoxels,xvoxels,numberofslices);
sortedstack = zeros(yvoxels,xvoxels,numberofslices);

% Reading the data
```

```
for zz = 1:numberofslices
    unsortedstack(:,:,zz) = dicomread(filenames{zz}); % Images
    hdr = dicominfo(filenames{zz}); % Header information
    unsortedsliceposition(i,1) = hdr.SliceLocation; % Getting slice position
end

% Sorting the slice positions
unsortedsliceposition(:,2) = 1:numberofslices;
sortedsliceposition = sortrows(unsortedsliceposition,1);

% Sorting the slices and tags
for zz = 1:numberofslices
    sortedstack(:,:,zz) = unsortedstack(:,:,sortedsliceposition(zz,2));
end
```

Figure 2 presents an example of unsorted and sorted slices organized using an approach similar to the previous script. The two images correspond to coronal planes of a fat-fraction 3D volume. On the left, we see the unsorted slices, organized according to the file name. On the right, the slices were sorted according to their corresponding spatial location.

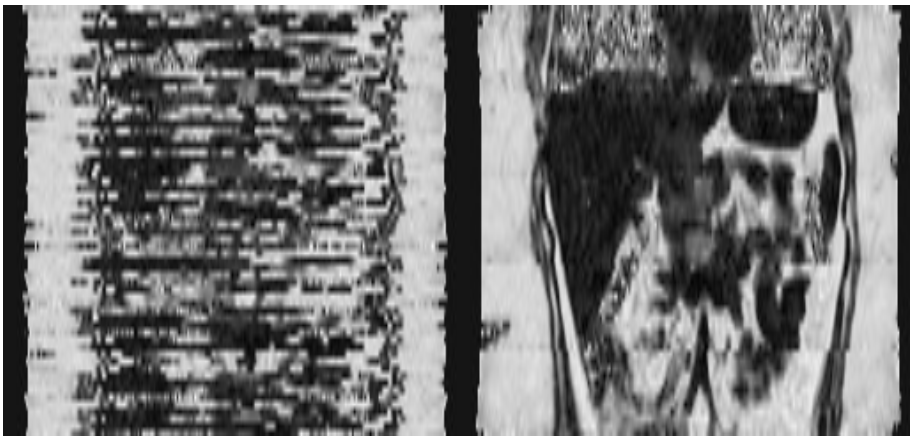

Figure 2: Coronal place of a 3D fat-fraction dataset. In the image on the left, the slices are sorted according to their file's names (MR000000, MR000001, and so on). On the right, the slices are sorted using the spatial location information obtained from the header of the DICOM files.

If the acquisition is completely sequential, probably the slice position will have correspondence to the file's name. However, it is not guaranteed. Therefore, it is always useful to perform the sorting task as routine before preprocessing the datasets.

3 Working with TAG files in MATLAB

Differently from DICOM case, MATLAB does not present a built-in set of functions to read *.tag files. Thus, we need to use the commands `fopen`, `fread` and `fclose` in order to perform

this task.

The *.tag files are composed by a header followed by various tag numbers. With the image size, we can identify the size of the header and obtain the gray-scale image corresponding to the tags from manual segementation. In addition, due to sliceOmatic's and MATLAB different orientations, it is necessary to rotate the image 90 degrees clockwise and flip it horizontally to match the corresponding DICOM file.

In order to facilitate this process, I wrote a function to read the tags and output the gray-scale image containing the corresponding data, called `tagread`. It is attached to this tutorial. This is the syntax of the function:

```
tagname = tagread(filename,[sizey,sizex,sizez])
```

Where *sizex*, *sizey* and *sizez* are the number of pixels the image will have in x, y and z planes, respectively. The output is a gray-scale image with the tags and the same orientation as the DICOM files as shown in Figure 3.

In Figure 3, it is possible to see a slice of fat-fraction dataset from a DICOM file and the corresponding tags obtained from manual segmenation using sliceOmatic, both imported to MATLAB.

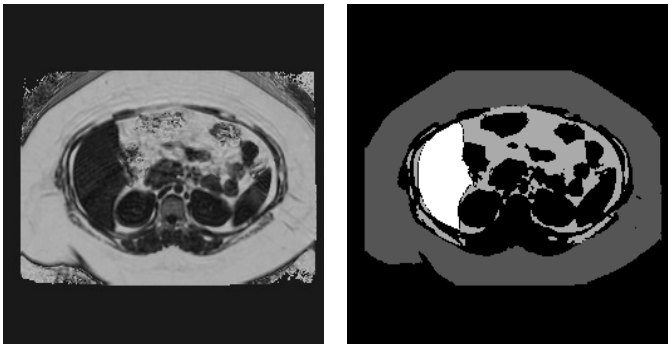

Figure 3: Axial slice, on the left, and corresponding tags, right.

3.1 Sorting tags

If the *.tag file contains tags from multiple slices, the function `tagread` will have as output a stack of tags correctly sorted. However, if all tags are not in the same file, we need to read each tag indenpendetely and perform a sorting task similar to the one used for the DICOMs. Since the slice position of each tag is the same of the corresponding DICOM file, we might use the same `sortedslieceposition` to sort the tags.

An example of complete sorting routine – accounting for both images and tags – is given below:

```
% Initializing datasets
unsortedsliceposition = zeros(numberofslices,2);
unsortedstack = zeros(yvoxels,xvoxels,numberofslices);
sortedstack = zeros(yvoxels,xvoxels,numberofslices);
unsortedtags = zeros(yvoxels,xvoxels,numberofslices);
```

```
sortedtags = zeros(yvoxels,xvoxels,numberofslices);

% Reading the data
for zz = 1:numberofslices
    unsortedstack(:,:,zz) = dicomread(filenames{zz}); % Images
    hdr = dicominfo(filenames{zz}); % Header information
    unsortedsliceposition(i,1) = hdr.SliceLocation; % Getting slice position
    unsortedtags(:,:,zz) = dicomread(tagnames{zz},[yvoxels,xvoxels]); % Reading the tags
end

% Sorting the slice positions
unsortedsliceposition(:,2) = 1:numberofslices;
sortedsliceposition = sortrows(unsortedsliceposition,1);

% Sorting the slices and tags
for zz = 1:numberofslices
    sortedstack(:,:,zz) = unsortedstack(:,:,sortedsliceposition(zz,2));
    sortedtags(:,:,zz) = unsortedtags(:,:,sortedsliceposition(zz,2));
end
```

The outputs of this sorting routine are sorted stacks of tags and images. It is needed to note that if the number of tags is not the same as the total amount of slices, some simple modifications in the routine are necessary to add black slices in the stack to complete it.