Converting DICOM 3.0 into ECAT 6.3

Riku Klén

29th August 2003

Turku PET Centre Image Processing Report Series

Report TPCIMG 0003

Contents

1 Problem

Problem is to convert DICOM 3.0 (Digital Imaging and COmmunications in Medicine) files into ECAT 6.3 file. One DICOM file contains one image (plane). ECAT file contains many image (planes).

This conversion is needed because we want to compare MRI (Magnetic Resonance Imaging) and PET (Positron Emission Tomography) images.

2 Solution

The problem is solved by using files `MRIfilenames.h`, `MRIfilenames.c`, `dicom3.c` and `dicom2cti.c`. The two first ones are used to collect information about the names of DICOM files. The third one is used to read DICOM files and write ECAT files. The fourth one is the actual program that contains the `main` method.

Following files are also included in the program: `ecat63.h`, `ecat63.c`, `img.h` and `img.c`.

2.1 How to use the program?

Program *dicom2cti* converts DICOM 3.0 files into an ECAT 6.3 file. All the DICOM input files have to be in the same folder. DICOM files have to contain numbers and they can differ only in numbers. DICOM file names can't contain character `+`. Number can be at most 4 digits.

For example DICOM files `file01.pic`, `file02.pic` and `file04.pic` have similar beginning and end. They differ only in number. Note that number one can be written 1,01,001 or 0001. The DICOM files are read in order of the numbers (smallest number first and 1 is considered smaller than 001). Number zero has to be 0 (only one zero).

The program takes 2 arguments:

`dicom file`	similar part of DICOM file name
`cti file`	name of the ECAT file to be created

Let `MR001`,`MR002`,...,`MR031` be DICOM files in the same folder and we would like to make an ECAT file (named `ecatfile.img`) from them into the same folder. This can be done by

```
dicom2cti MR+ ecatfile.img
```

There is also one optional argument to the program. It is used if DICOM files are overview files (if the name of the file begins with OV). The third argument can

be any word. If the program is called with three arguments it will automatically consider the input DICOM files as overview files.

If we have files `OV001`,`OV002`,...,`OV011` in a same folder and we would like to make an ECAT file (named `ov.img`) from them into the same folder. This can be done by

```
dicom2cti OV+ ov.img 1
```

3 Analysis of the program

3.1 `MRIfilenames.h`

Header file `MRIfilenames.h` contains one constant `MAXIMUM_NUMBER_OF_PLANES` and one structure `struct fileNames`. Naturally `MAXIMUM_NUMBER_OF_PLANES` describes how mane planes (images) the program can handle at most.

Structure `struct fileNames` is used to store the information about the names of DICOM files. It hs seven members. Vector `short int numbers[]` is used to store actual numbers (0-9999) that were found in DICOM file names. Vector `short int zeros[]` is used to store the number of zeros that were found in DICOM file names before the actual number. Both vectors have lengthof`MAXIMUM_NUMBER_OF_PLANES`. If the number part of DICOM file is zero (this isn't recommended) it has to contain only one zero. Member `short int numberOfFiles` is used to store total number of DICOM files and `short int currentFile` tells how many files have been handdled. If all files have been handled then `currentFile` is greater than `numberOfFiles`. Member `char nameBegin` contains the beginning of the DICOM file name including path (part of the input name before +) and `char nameEnd` contains the end of the DICOM file name (part of the input name after +). Member `char name` contains the name of the DICOM file that has previously been handled (including path). If all DICOM files have been handled then it will be a empty word `""`.

type	name	definition
short int[]	numbers	actual number part of the DICOM file $(0-9999)$
short int[]	zeros	number of zeros before the actual number $(0-3)$
short int	numberOfFiles	number of DICOM files
short int	currentFile	number of files that have been handled
char	nameBegin	part of DICOM file name before charachter +
char	nameEnd	part of DICOM file name after charachter +
char	name	name of the DICOM file that have been handled

3.2 MRIfilenames.c

File `MRIfilenames.c` consists of two functions. They are used to handle information about DICOM files.

Function `void MRIfilenames(char*, fileNames*)` is used to scan the folder for potential DICOM files and make a list of them. The first argument is a pointer to a string that is input word of form $xx + yy$. The second argument is a pointer to a structure `fileNames`, which is used to store infomation about DICOM file names. Function has five members. Member `int intnumber` is a integer number $(0 - 9999)$ and `char number` is the same integer as vector. Members `char name` and `FILE *filename` are used to make possible filenames out of input string and to test if such a file exists. Member `int index` is an index to members `numbers` and `zeros` of input structure `fileNames`.

If input string doesn't contain + then function is exited. Otherwise the beginning (xx) and the end (yy) of input string are defined. Next are checked if there exists a file of form $xx + yy$ and if so it is added to the list. This is done for all possible + $(0, 1, 01, 001, 0001, 2, 02, \ldots, 9999)$. The list is ordered so that the smallest number is first and the greatest last. If two numbers differ only in the number of zeros before the actual number then the one that has more zeros is considered greater. For example number 001 is considered greater than 1. Finally members `numberOfFiles`, `currentFile` and `name` of input structure `fileNames` are initialized.

Function `char *NextFileName(fileNames*)` returns a pointer to a vector that contains next DICOM file name. If there are no more DICOM files a pointer to empty vector is returned. The only argument of the function is a pointer to structure `fileNames`, that contains information about DICOM files.

3.3 dicom3.c

File `dicom3.c` is used to read DICOM files and create an ECAT file. It uses functions of `MRIfilenames.c`. It consists of one class structure and various reading function.

Both DICOM files and ECAT files consists of header files and image data. DICOM header has been marked with tags. A tag is of form (g,e) where g (group) and e (element) are numbers of 4 digits in hexadecimal form.

Class structure `mri_header` is used to store header data from DICOM files. Once data has been read it will be stored into a ECAT file. Below there is information about the members of the structure.

name	tag	name	tag
scan_day	(0008, 0020)	plane_separation	(0018, 0088)
scan_month	(0008, 0020)	dimension_1	(0028, 0010)
scan_year	(0008, 0020)	dimension_2	(0028, 0011)
scan_hour	(0008, 0030)	dimension_3	(0028, 0012)
scan_minute	(0008, 0030)	pixel_size	(0028, 0030)
scan_second	(0008, 0030)	slice_width	(0018, 0050)
patient_name	(0010, 0010)	manufacturer_data_read	(0008, 0070)
patient_id	(0010, 0020)	model_data_read	(0008, 1090)
study_description	(0008, 0060)	plane_number	(0020, 0013)

type	name	definition
short int	scan_day	scan day $(1-31)$
short int	scan_month	scan month $(1-12)$
short int	scan_year	scan year (positive)
short int	scan_hour	scan hour $(0-23)$
short int	scan_minute	scan minute $(0-59)$
short int	scan_second	scan second $(0-59)$
char[32]	patient_name	name of the patient
char[16]	patient_id	identification number of the patient
char[32]	study_description	description of the study
float	plane_separation	space between slices (cm)
short int	dimension_1	number of rows $(1-512)$
short int	dimension_2	number of columns $(1-512)$
short int	dimension_3	number of frames
float	pixel_size	size of pixels (cm)
float	slice_width	width of slices (cm)
short int	manufacturer_data_read	is data read, $0 =$ no
short int	model_data_read	is data read, $0 =$ no
short int	plane_number	number of image

Method `void ReadOff(short*, short*, int*, FILE*)` reads next header tag but doesn't write it anywhere. The first argument is a pointer to the group, the second argument is a pointer to the element, the third argument is a pointer to a value that tells the length of the element and the last argument is a pointer to structure `mri_header` in which the information could be stored.

Similarily works methods `void ReadGroup#(short*, short*, int*, FILE*)`, but this time information is actually stored in the structure if it is necessary.

Method `char *ReadMriFile(char*, struct mri_header*)` is used to read

DICOM file header. It returns a pointer to image data which is stored after the header information. The first argument is a pointer to the DICOM file name and the second argument is a pointer to `struct mri_header` in which DICOM header data will be saved. It uses previously presented reading methods.

Method `int ReadDicom(char[]*)` is the actual program. The argument is a pointer to a vector. The first element of the vector is a pointer to the name of DICOM file and the second element is a pointer to the name of ECAT file. Method uses method `char *ReadMriFile(...)`, both functions implemented in file `MRIfilenames.c` and methods presented in `ecat63.h`.

Method `int ReadDicomOverview(char[]*)` works similarily to method `int ReadDicom(char[]*)`. It is used for overview files.

3.4 `dicom2cti.c`

File `dicom2cti.c` contains only `main` method. If number of the arguments is uncorrect (not equal to two or three) a usage message is printed. Otherwise function `ReadDicom` or `ReadDicomOverview` is called depending on the number of arguments. The first argument is the name of DICOM file in $xx+yy$ form and the second argument is the name of ECAT file. The optional third argument can be any word.

4 Testing the program

Test DICOM files `MR0001,...,MR0009` have been converted into file `koe.img`. This is done by

```
dicom2cti MR+ koe.img.
```

File `dicom3.c` has a method `test()` that can be use to test the program. It will print few extra lines on the screen when running the program. To use the method line

```
test();
```

needs to be added in the very beginning of the method `ReadDicom` in file `dicom3.c`.