Advanced image registration

Riku Klén

28th August 2003

Turku PET Centre Image Processing Report Series

Report TPCIMG 0006

Contents

1 Problem

The problem is to translate MR and CT images into same postion. It is assumed that both files are in same file format (ECAT 6.3).

2 Solution

Solution is based on program `amirfit` written by Babak A. Ardekani in 1994. It uses few mathematical applications to improve the quality of the cost funtion. These application are explained in the following section.

Main intrest of this project was to find an accurate cost function. There for efficency of the program isn't very good. It can thou be improved by known algorithms such as Golden Section Search or Powells's method.

2.1 How to use the program?

The name of the program is *afit*. It can be used to fit ECAT format MR image into ECAT format CT image. Both images have to be transaxial and CT image has to be a transmission image. MR and CT image should match each other (meaning that the first and the last planes are at the same height and both images have same tilt angle). The program creates a new ECAT file. It takes three arguments:

`mri file`	name of MRI file
`cti file`	name of CTI file
`new mri file`	name of new fit MRI file

For example file `mri.img` is transaxial MR image of ECAT form and `cti.img` is transaxial CT (transmission) image of ECAT form. Fit MR image `newmri.img` can be created by

```
afit mri.img cti.img newmri.img
```

2.2 Additional features

There are two optional arguments to the program:

`iterations`	number of iterations made in matching (initially 1)
`mode`	which method and image type is used

User has an oppertunity to increase number of iterations which will consume more time and give more accurate result. This is done by the fourth (and first

optional) argument. User can also use an optional translation fitting method and other CT images than transmission image. This is not recommended. For mode there are four options:

`mode`	translation method	image type
0	normal	transmission image
1	normal	other image
2	linear	transmission image
3	linear	other image

For example file `mri.img` is transaxial MR image of ECAT form and `cti.img` is transaxial CT (transmission) image of ECAT form. Fit MR image `newmri.img` of 4 iterations can be created by

```
afit mri.img cti.img newmri.img 4
```

If `cti.img` is not a transmission image, the command line is

```
afit mri.img cti.img newmri.img 4 1
```

or

```
afit mri.img cti.img newmri.img 4 3
```

3 Analysis of the program

Before calculating values for the cost function the image needs to be processed. First the contour of the image is searched by using smoothening and gradient image. Then the image is made to consist of one piece (or made connected). Finally the image is clustered and these clustered images are matched.

Connected component means such set of points, that all points in the set are reachable from any other point. *4 connected point*[1] is a point that has a neighbour in north, south, west and east two dimensionally (See figure 1). Similarily is defined *6 connected point*. Now a point has a neighbour in all 6 directions north, south, west, east, up and down (See figure 1).

3.1 `head_contour.c`

File `head_contour.c` contains methods to make a contour image of an image. It uses file `ecat63.h` and has 5 methods.

First of all the image needs to be smoothened before it can be used in registration. Method `void boxcar_average(IMG*, IMG*)` is used to smoothen

[1] For more detailed mathematical theory of n connectivity see A. Rosenfeld and A. C. Kak, Digital Picture Processing, Second Ed., Vol. 2, page 206, Academic Press, 1982.

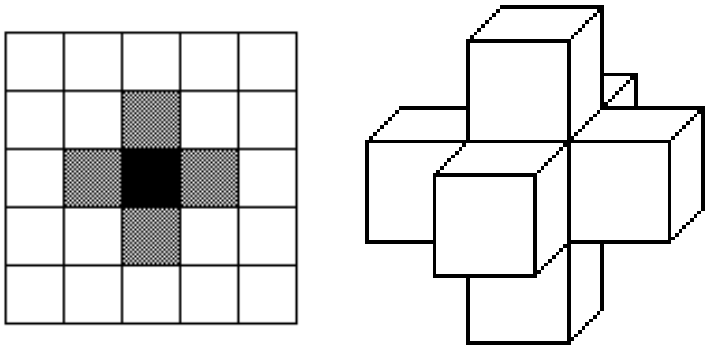

Figure 1: 4 connected pixel (coloured black) and 6 connected voxel (inside the knot).

the image. Each plane of the first frame is smoothened. The first argument is input image and the second argument is output image. Smoothening is done by calculating average value of 3×3 window for each pixel that is not on the edge of the image. The selected pixel is in the center of the window. Values of pixels on the edge of the image remains the same.

Once image is smoothened the contour of the image can be found. Method `void sobel_gradient(IMG*, IMG*)` is used to find pixels which value differs from the values of its surrounding pixels. The first argument is smoothened input image and the second argument is output image. So called *Sobel gradient* is calcualted using 3×3 window for every pixel that is not on the edge of the image. Values are calculated for each plane of the first frame. Consider matrix $\mathbf{A}$ as a plane of an image and a_{ij} as an element of the matrix or pixel of the image (i is row and j column). Now Sobel gradient can be calculated by

$$G(i,j) = \frac{|X| + |Y|}{2},$$

where

$$X = (a_{i+1,j+1} + 2a_{i+1,j} + a_{i+1,j-1}) - (a_{i-1,j+1} + 2a_{i-1,j} + a_{i-1,j-1})$$

and

$$Y = (a_{i+1,j+1} + 2a_{i,j+1} + a_{i-1,j+1}) - (a_{i+1,j-1} + 2a_{i,j-1} + a_{i-1,j-1}).$$

See figure 2 for graphical presentation of the gradient.

Following step is used to remove noisy voxels. This is done by method `void binary_mask(IMG*, IMG*)`. It removes all the voxels that have value less than

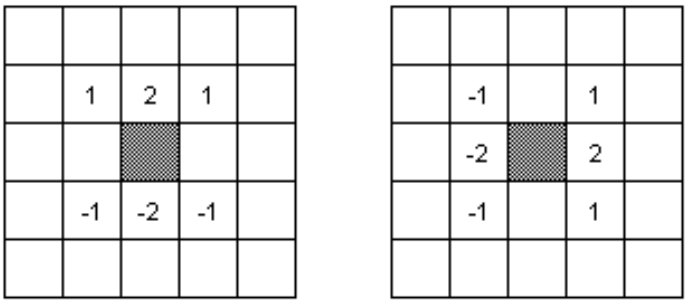

Figure 2: Variables X (left) and Y in Sobel gradient.

15 percent of the maximum value. Other voxels get value 1. Removing means that voxel gets value 0 and maximum value means maximum value of the whole image (all planes). The first argument is input image and the second argument is output image. The input image should be an image that shows the Sobel gradients.

Finally needs to be checked that which connected component of the masked image contains largest number of voxels. Method `int find4cc(IMG*, short int, short int, short int, short int)` is a recursive method to label a 4 connected component. The first argument is masked input image and the second argument is output image. The third argument is number of plane, the fourth is number of row, the the fifth is number of column and the last argument is the label of the 4 connected component. The label has to be greater or equal to 2 and all the voxels that are not connected get value 0.

Method `void mask_4cc(IMG*)` is used to make the contour image. It uses previously presented method and the only argument is a pointer to a binary masked image. The program will change the input image. All the voxels that belong to the largest 4 connected component (a 4 connected component that has to be greatest number of voxels) get value 1. If the largest 4 connected component has holes they are also filled with value 1. All the other voxels get value 0. If there occurs an error during this function, it migth be possible to fix it by unlimiting the stack size of the computer. This is done by `unlimit stacksize`.

3.2 `cluster.c`

Once connectivity is cheked then the image is clustered. File `cluster.c` contains one method `void cluster(IMG*, IMG*, IMG*, short int)`. It is used to cluster image plane by plane. The first argument is a pointer to a connected component image (that has only voxels valued 0 or 1) and the second argument is a pointer to a MR image from which the connected component image has been made of. The third argument is a pointer to output image and the last argument is number of classes. Every pixel will get an integer value from 0 to number of classes. The orginal values are scaled to these integer values so that the minimum value will become 0 and the maximum value will become the number of classes.

3.3 `register.c`

When orginal images are processed to clustered images the registration can begin. File `cluster.c` contains 6 methods of which only three are used.

Methods `void mask_6cc(IMG*)` and `int find4cc(IMG*, short int, short int, short int, short int)` can be used to make a contour image. They work similarily as methods `mask_6cc(...)` and `find4cc(...)`. If there occurs an error during these function, it migth be possible to fix it by unlimiting the stack size of the computer. This is done by `unlimit stacksize`. These methods are not used in the current version.

Methods `float transaxial(IMG*, IMG*)`, `float sagittal(IMG*, IMG*)` and `float coronal(IMG*, IMG*)` are the cost functions. One for each possible view. For all three methods the first argument is a pointer to a clustered PET image and the second argument is a pointer to a clustered MR image. Returned value is a value that tells how well the images are matched. The smaller the value is the better the images are matched. Actually the returned value is opposite value to the value of the number of pixels of PET image that fit in the MR image.

Method `void center(IMG*, IMG*, point*)` calculates the difference of center points of two images. The first argument is a pointer to a k-mean MR image, the second argument is a pointer to a k-mean transmission CT image and the third argument is a pointer to a point where the difference is stored. It moves the MR image 5 pixels along and against all axis and tests which of these images fits best into the CT image. This method is used in the optional matching method (linear).

3.4 `fit.c`

File `fit.c` contains the actual program. It uses files `cluster.c`, `head_contour.c`, `slicer.c`, `fit.c`, `register.c` and `point.c` aswell as some files to support ECAT format.

File `fit.c` consists only of the main method. The program works when called with 3 arguments as presented in section 2.1. First the arguments are checked and then k-mean images are made. Then the actual fitting begins. The k-mean MR is fit into k-mean transmission CT image and the transformation parameters are stored. First the k-mean MR image is transformed and then rotated three dimesionally. Finally the orginal MR image is transformed with the calculated parameters.

4 Testing the program

File `amirfit.c` contains a test function `void test()`. It can be used in testing the program by adding line

```
test();
```

in the beginning of the main method. Once the line is added the program will print additional lines on the screen when used.