FastSPECT II Image Reconstruction Suite (version 4.04)

Luca Caucci
caucci@email.arizona.edu

February 5, 2015

Abstract

This document provides brief descriptions of the programs that have been developed to perform maximum likelihood expectation maximization (MLEM) image reconstruction for FastSPECT II. Examples are provided as well.

Contents

1 Getting Started

1.1 Compiling the Code

The programs discussed in this brief manual have been developed and tested on a Linux machine on which CUDA 6.5 (`http://www.nvidia.com/object/cuda_home_new.html`) had previously been installed. Although some parts of the code (such as the function `get_exec_path(...)` in `fs2_util.cc`) are Linux-depended, it is possible to port the code to other compilers and operating system (such as Visual Studio on MS-Windows systems).

Under Linux, the programs can be compiled using the `make` utility and the provided `Makefile`. The programs require some libraries to be installed in the system. Such libraries include `dcmtk` (for handling DICOM files) and `jansson` (for reading the JSON configuration file). If these libraries are not installed in your system, please contact your system administrator. Some minor adjustments of the `Makefile` might be required to successfully compile the code.

The programs must be run on a system with one or more GPU devices installed. The present GPU implementations do require GPU devices with GPU capability 3.5 or higher. The programs search the system for GPU devices with the required capability.

1.2 The "###" Filename Format

Some of the programs that have been developed use collections of files whose names are of the form `proj_00.dat`, `proj_01.dat`, ..., `proj_15.dat`. To avoid having to list all these filenames in the command line, the "`###`" filename format convention has been developed. Whenever a list of filenames like the one above is required, the user will simply have to replace the numerical values with a sequence of `#` characters. The program will replace the sequence of `#` characters with the appropriate zero-padded integer number. The number of `#` characters will be equal to the number of characters in the zero-padded integer number. For example, the list of filenames `proj_00.dat`, `proj_01.dat`, ..., `proj_15.dat` above can succinctly be denoted as `proj_##.dat`. Please note that if fewer `#` characters than needed are used, the program behavior is undefined. The "`###`" filename format is also used to specify the name of the files that store the reconstructed data throughout the iterations of the reconstruction algorithms.

1.3 The "{...}" Filename Format

For the case of dynamic studies, it is common to have a few data sets (collected at different times) of the same animal. The filenames of these data sets might include one or more common parts plus a variable part, which, usually, denotes when the data were taken. To fit this need, the "`{...}`" filename format has been developed. In the "`{...}`" filename format, a comma-separated list of variable parts is specified between `{` and `}`. In some cases, it is also possible to combine the "`###`" and the "`{...}`" formats when specifying filenames. For example, suppose that LM data have been collected four times every 30 minutes and that the variable parts in the filenames are `30_min`, `1_hr`, `1_hr_30_min`,

and `2_hr`. With the "`{...}`" and the "`###`" filename format conventions, all the LM files will be denoted as, for example, `mouse_{30_min,1_hr,1_hr_30_min,2_hr}_##.dat`. In this example, `mouse_` and `_##.dat` are the common parts for the "`{...}`" filename format. This example also shows that it is possible to combine the "`{...}`" filename format with the "`###`" filename format. If there is only one possible variable part, the `{` and `}` characters can be omitted. Please notice that it might be necessary to enclose the filename in double quotes (i.e., the `"` character) to prevent the operating system to interpret the `{` and `}` characters.

1.4 The DICOM Standard

The Digital Imaging and Communications in Medicine (DICOM) is a standard for handling, storing, printing, and transmitting information in medical imaging. The current version of the programs developed for FastSPECT II image reconstruction support the DICOM standard for the storage of 3D images. The greatest advantage of DICOM files is that they allow storing a wide range of supplemental information along with the actual pixel data. In the current version of the FastSPECT II image reconstruction programs, the voxel size is stored in the DICOM files. DICOM files can easily be loaded into Amide (`http://amide.sourceforge.net/`) for visualization. The current version of the programs also sets the patient's name to the name of the file (purged of the file path) plus the date when the file was created. This facilitates handling multiple files that have been loaded into Amide.

1.5 The "`2dlm`" Data Format

The "`2dlm`" data format has been developed to store 2D camera positions of interactions estimated from PMT data. The file format assumes that FastSPECT II is equipped with sixteen cameras. A "`2dlm`" file begins with a header of sixteen 32-bit unsigned integer numbers $\ell_1, \ldots, \ell_{16}$. These numbers denote the number of 2D attribute vectors estimated for each camera. The header is followed by sixteen blocks of 2D (x, y) position estimates, stored as single-precision little-endian floating-point numbers. The number of position estimates stored in the n^{th} block (thus corresponding to the n^{th} camera) is ℓ_n. The units of x and y are millimeters. The following few lines of MatLab code will read a file that follows the "`2dlm`" data format and will display each camera's block of 2D position estimates as a scatter plot.

```
close all;
clear all;
filename = 'MOBY_proj.2dlm';
det_min = 0.00;
det_max = 79 * 1.50;
LM_data = cell(16, 1);
fid = fopen(filename, 'r');
num = fread(fid, 16, 'uint32');
for camera = 1:16
```

```
    LM_data{camera} = fread(fid, [2, num(camera)], 'float');
end
fclose(fid);
figure('Position', [100, 100, 800, 600]);
for camera = 1:16
    subplot(4, 4, camera);
    x = LM_data{camera}(2, :);
    y = LM_data{camera}(1, :);
    if num(camera) > 0
        plot(x, y, '.', 'MarkerSize', 1);
        axis square;
    end
    xlim([det_min, det_max]);
    ylim([det_min, det_max]);
    set(gca, 'YDir', 'reverse');
    title(['Camera ', num2str(camera)], 'FontWeight', 'Bold');
end
```

1.6 The `fs2_conf.json` Configuration File

Some important parameters (such as the names of the files containing calibration data used for the ML search of 2D position of interaction from LM data), are listed in the configuration file `fs2_conf.json`. The structure of this file follows the JSON standard. Information about the JSON standard can be found at `http://json.org/`.

2 An Example

In this section we will provide an example that shows how the programs we developed can be used to reconstruct 3D data from raw photomultiplier tube (PMT) outputs acquired with FastSPECT II [Che06, FWC+04]. Detailed description of the programs is reported in the next section. The folder `example` contains PMT data for a bone scan of a simulated MOBY phantom[1] [STF+04]. The user can access these data by changing the current directory to `example`:

```
cd example
```

2.1 Preprocessing the PMT Data

The first step consists of processing [FHB05, HCK+10, BHM+09] the raw PMT data to get 2D event position estimates on each camera. This can be accomplished as follows:

```
../get_data_proj MOBY_PMT_###.dat MOBY_proj.2dlm
```

This command generates the file `MOBY_proj.2dlm` which, for each camera, lists 2D event positions on the camera face. The content of this file can be visualized with the MatLab script discussed in Section 1.5. The script is also contained in the `MATLAB` folder. If we run it on the "`2dlm`" file `MOBY_proj.2dlm`, a set of sixteen diagrams will be shown. One of them has been reported in Figure 1.

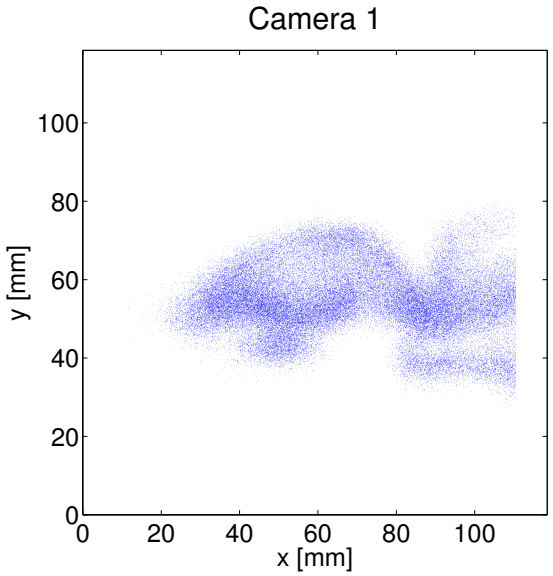

Figure 1: Plot of 2D event positions for Camera 1

[1]The MOBY phantom is a 4D mouse whole body phantom and it is distributed with a small licensing fee. Please refer to `http://www.hopkinsradiology.org/DMIP/Research/xcat` for more information on how to obtain the MOBY phantom.

2.2 Performing MLEM Reconstruction

The data in `MOBY_proj.2dlm` can be reconstructed with the `mlem` program, which is a GPU implementation of the maximum likelihood expectation maximization (MLEM) method [SV82, DLR77]. A possible usage is as follows:

```
../mlem MOBY_proj.2dlm up_4 MOBY_recon_###.dcm 30
```

In our case, we have used the option `up_4` to get a 4× upsampling of the original PSF and we decided to run 30 iterations of the MLEM algorithm. The reconstructed data will be saved in DICOM format in the files `MOBY_recon_`n`.dcm`, in which n takes values `005`, `010`, ..., `030`.

2.3 Removing MLEM Artifacts

The MLEM algorithm is notorious for creating artifacts near the edges of the field of view (FOV) [SMTP87]. For this reason, the command `trim_FOV` has been developed to remove such artifacts near the edges of the field of view. An example of usage is shown below:

```
../trim_FOV MOBY_recon_030.dcm MOBY_recon_trimmed_030.dcm 0.85 0.85 0.10
```

The final result—file `MOBY_recon_trimmed_030.dcm`—can easily be visualized with a variety of tools. For example, we can use AMIDE (`http://amide.sourceforge.net/`) to obtain a 3D rendering such as the one shown in Figure 2.

Figure 2: AMIDE 3D rending of a bone scan of a simulated MOBY phantom (click on the image to play clip)

3 Description of Reconstruction Codes

3.1 The `get_data_proj` Program

Description

This program performs 2D ML position estimation from PMT outputs of list-mode FASTSPECT II data and generates files that contain blocks of 2D position estimates, one block for each camera.

Usage

```
./get_data_proj LM_filenames proj_filenames
```

where:

- `LM_filenames`
  Filenames (in "`###`" format, and, optionally, "`{...}`" format) of the list-mode FASTSPECT II files.
- `proj_filenames`
  Filenames (in "`{...}`" format) of the "`2dlm`" files for the projection list-mode data.

3.2 The `mlem` Program

Description

This program reconstructs 2D projection data via the MLEM iterative algorithm. The 3D reconstructed data are written to one or more files.

Usage

```
./mlem proj_filenames num_FOV_voxels recon_filenames num_iter
```

where:

- `proj_filenames`
  Filenames (in "`{...}`" format) of the "`2dlm`" files storing the projection list-mode data.
- `num_FOV_voxels`
  Number of voxels for the FOV. This parameter can be "`no_up`" for no FOV upsampling; "`up_2`" for 2× FOV upsampling; "`up_4`" for 4× FOV upsampling; "`up_8`" for 8× FOV upsampling; or also "`AxBxC`" (where `A`, `B`, and `C` are integer numbers) for FOV upsampling to A, B, and C samples along each dimension.
- `recon_filenames`
  Filenames (in "`###`" format, and, optionally, "`{...}`" format) of the DICOM files for the reconstructed data. The data will be saved every five iterations, as well as at the end of the whole reconstruction process.

- `num_iter`
  Number of MLEM iterations to be performed.

3.3 The `mlem_move_bed` Program

Description

This program reconstructs 2D projection data via the MLEM iterative algorithm. The 3D reconstructed data are written to a file. It is assumed that two or more bed positions are used during the scan and the data for these bed positions are provided to the program. The only bed movement available is moving inside or outside the imaging system (X axis).

Usage

```
./mlem_move_bed num_bed_pos delta_pos proj_filenames
        num_FOV_voxels recon_filenames num_iter
```

where:

- `num_bed_pos`
  Number of bed positions (must be at least 2).

- `delta_pos`
  List of comma-separated non-negative integer numbers representing the number of steps between two consecutive bed positions. Please note that the step size is in units of the upsampled field of view (for example, if the original PSF was acquired on a 2-mm grid and the option "`up_4`" is used, then a unit of bed motion corresponds to 0.50 mm). Notice that the number of integer numbers in the comma-separated list has to be one less than the number of bed positions.

- `proj_filenames`
  Filenames (in "`{...}`" format) of the "`2dlm`" files storing the projection list-mode data for each bed position. Notice that the order of the tokens in the "`{...}`" format does matter.

- `num_FOV_voxels`
  Number of voxels for the FOV. This parameter can be "`no_up`" for no FOV upsampling; "`up_2`" for 2× FOV upsampling; "`up_4`" for 4× FOV upsampling; "`up_8`" for 8× FOV upsampling; or also "`AxBxC`" (where `A`, `B`, and `C` are integer numbers) for FOV upsampling to A, B, and C samples along each dimension.

- `recon_filenames`
  Filenames (in "`###`" format) of the DICOM files for the reconstructed data. The data will be saved every five iterations, as well as at the end of the whole reconstruction process.

- `num_iter`
  Number of MLEM iterations to be performed.

3.4 The `trim_FOV` Program

Description

This program trims the edges of the FOV of reconstructed data, so that MLEM artifacts near the edges of the FOV are eliminated. The tapering function has the following form:

$$f(r, R_0, W) = \begin{cases} 1 & \text{if } r < R_0 - W/2, \\ \frac{1}{2}\left\{\sin\left[\pi\left(1 + \frac{r-R_0}{W}\right)\right] + 1\right\} & \text{if } R_0 - W/2 \le r \le R_0 + W/2, \\ 0 & \text{if } r > R_0 + W/2. \end{cases}$$

Usage

```
./trim_FOV inputs outputs ell radius width
```

where:

- `inputs`
  Filenames (in "`{...}`" format) of the DICOM input files.

- `outputs`
  Filenames (in "`{...}`" format) of the DICOM output files.

- `ell`
  Fraction (relative to X the center of the field of view at which the voxel intensities are set to zero. As this parameter is lowered (from its maximum value of 1.00 towards 0.00) more and more slices are set to zero. A good value is 0.85.

- `radius`
  Relative radius at which the middle part of the tapering filter is applied. A good value is 0.85.

- `width`
  Relative width of the tapering filter. By setting this parameter to a small value, a step-function-like tapering filter can be obtained. A good value is 0.10.

References

[BHM+09] Harrison H. Barrett, William C. J. Hunter, Brian William Miller, Stephen K. Moore, Yichun Chen, and Lars R. Furenlid. Maximum-likelihood methods for processing signals from gamma-ray detectors. *IEEE Transactions on Nuclear Science*, 56(3):725–735, June 2009.

[Che06] Yi-Chun Chen. *System Calibration and Image Reconstruction for a new Small-Animal SPECT System.* PhD thesis, University of Arizona, Tucson, AZ, 2006.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[FHB05] Lars R. Furenlid, Jacob Y. Hesterman, and Harrison H. Barrett. Real-time data acquisition and maximum-likelihood estimation for gamma cameras. In *14^{th} IEEE-NPSS Real Time Conference*, pages 498–501, Stockholm, Sweden, June 2005.

[FWC+04] Lars R. Furenlid, Donald W. Wilson, Yi-Chun Chen, Hyunki Kim, Philip J. Pietraski, Michael J. Crawford, and Harrison H. Barrett. FastSPECT II: A second-generation high-resolution dynamic SPECT imager. *IEEE Transactions on Nuclear Science*, 51(3):631–635, June 2004.

[HCK+10] Jacob Y. Hesterman, Luca Caucci, Matthew A. Kupinski, Harrison H. Barrett, and Lars R. Furenlid. Maximum-likelihood estimation with a contracting-grid search algorithm. *IEEE Transactions on Nuclear Science*, 57(3):1077–1084, June 2010.

[SMTP87] Donald L. Snyder, Michael I. Miller, Lewis J. Thomas, and David G. Politte. Noise and edge artifacts in maximum-likelihood reconstructions for emission tomography. *IEEE Transactions on Medical Imaging*, 6(3):228–238, October 1987.

[STF+04] W. Paul Segars, Benjamin M. W. Tsui, Eric C. Frey, G. Allan Johnson, and Stuart S. Berr. Development of a 4D digital mouse phantom for molecular imaging research. *Molecular Imaging and Biology*, 6(3):149–159, 2004.

[SV82] L. A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, 1(2):113–122, October 1982.