Intro to FreeCT_wFBP

John Hoffman

October 29, 2015

While we have demonstrated that our software *can* be used to obtain high-quality reconstructions, we do not guarantee reconstruction quality in any general sense. FreeCT_wFBP should not, under any circumstances, be used for diagnostic or other clinical work.

Contents

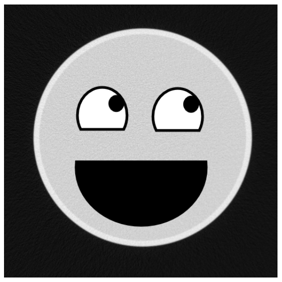
1 Developer's Note

We have done our best to write a software package that is easy to use *and* easy to understand, however it is unlikely that users will get satisfactory results without some baseline familiarity with CT reconstruction. Analytic CT reconstruction, while relatively simple in concept, is in practice a complex and highly-sensitive process. In some parameters, seemingly miniscule errors can introduce systemic problems that can dramatically reduce or destroy image quality. We highly recommend reading the publications referenced in section 2 prior to learning to use FreeCT_wFBP. This will help users become familiar with the necessary parameter inputs and potential complications that can be experienced with the software.

Similarly, a user must also be familiar with the geometry of the system from which they are attempting to reconstruct data. This includes, but is not limited to things such as detector spacing and/or fan angle increment, gantry geometries, tube anode angle, and other, often proprietary, information. For clinical scanners, this data sometimes is available publicly via the internet, or must be obtained through agreements with a manufacturer. Knowledge of 3rd generation CT scanners can also be of assistance in deducing or deriving some of these parameters. We've found that we just sometimes have to experiment to get optimal results.

If you experience trouble with the software please don't hesitate to contact us at **freect.project@gmail.com**. It is through user feedback that FreeCT_wFBP will improve. We will make every effort to support the software from a "quality control" standpoint (bug fixes, instructions on software installation and use, etc.); we cannot guarantee that we will be able to support every request for assistance in solving reconstruction quality issues since there's just too darn many things that can cause problems and only one FreeCT development team! The exception to this is with the samples provided via the FreeCT website.

Thanks for your interest in FreeCT and happy reconstructing!



2 Introduction

FreeCT_wFBP grew out of a need for fast, customizable diagnostic CT reconstruction software. We have striven to make the code easy to read, use, and modify with the intent of facilitating education and research. It is an implementation of the reconstruction methods outlined in:

- K. Stierstorfer, A. Rauscher, J. Boese, H. Bruder, S. Schaller, and T. Flohr, “Weighted FBP—a simple approximate 3D FBP algorithm for multislice spiral CT with good dose usage for arbitrary pitch,” Phys. Med. Biol., vol. 49, no. 11, pp. 2209–2218, Jun. 2004.
- T. G. Flohr, K. Stierstorfer, S. Ulzheimer, H. Bruder, a N. Primak, and C. H. McCollough, “Image reconstruction and image quality evaluation for a 64-slice CT scanner with z-flying focal spot.,” Med. Phys., vol. 32, no. 8, pp. 2536–2547, 2005.

An outline and validation of FreeCT_wFBP has been published (submitted: 10/25/2015, under review as of 10/29/2015) as a technical note in Medical Physics:

- (Citation will be provided if/when accepted)

FreeCT_wFBP is currently most easily used under the Ubuntu distribution of GNU/Linux, however all other Linux distributions should work as well but will require a more manual approach to building and installing the software (i.e. utilizing the provided makefiles instead of the install script).

3 Quick Start Guide (Ubuntu/Debian)

This is the default, recommended installation. It requires root privileges (i.e. “sudo”) to work without modification, however it is most likely to produce desired results.

These instructions are only valid for systems running Ubuntu/Debian. Although it is likely that much of the installation script will work on other systems, we cannot guarantee that everything will work without modification.

Download and install the CUDA toolkit (and driver) Download and install the CUDA toolkit 6.5 (and driver if using a GPU for reconstruction) from:

https://developer.nvidia.com/cuda-toolkit-65

then follow the instructions for configuring your linux environment found here (adjusting any version numbers as needed):

http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux/#environment-setup

See section 4.1 for more information about permanently configuring your environment for CUDA.

Download and run the fct_install script Download the desired "fct_install_ubuntu" script from the FreeCT website. Ensure that the script is executable with the following command:

```
user@node:~/Downloads/$ chmod +x fct_install_ubuntu
```

Run the script with:

```
user@node:~/Downloads/$ ./fct_install_ubuntu
```

Note that you should *not* call the script with "sudo"; you will be prompted when the script requires root permissions.

(Optional) Download and run the samples It is highly recommended that you download and install the samples from the website. Samples include all of the raw projection data used in the FreeCT Medical Physics publication, as well as the parameter files required for reconstruction. This will allow you to familiarize yourself with the structure of FreeCT_wFBP's parameter files and command line calls, without the burden of configuring everything for yourself.

This subsection will be updated with a step by step guide on downloading and running samples.

4 Building the software manually from source

If the install script is not cooperating with your system, or if you are using a distribution other than Ubuntu or Debian (or a derivative thereof), it's likely the the software will still work, you just need a more manual approach to compiling. Please note that all of the steps below (with the exception of the CUDA Toolkit installation) are performed automatically by the installation script. We highly encourage users to take advantage of it, if possible.

4.1 Get the CUDA Toolkit

First, ensure that you have downloaded and installed the CUDA toolkit (https://developer.nvidia.com/cuda-downloads). It is not necessary to have installed the CUDA driver to build the software, however if you wish to run FreeCT_wFBP on your NVIDIA GPU, the corresponding CUDA driver is required. You will likely need to add the follwing to your ~/.bashrc file (google "editing bashrc" if you are unfamiliar with this file):

```
export PATH=/usr/local/cuda/bin:${PATH}
export MANPATH=/usr/local/cuda/man:${MANPATH}
if [[ "\x${LD_LIBRARY_PATH}" != "" ]]
then
        export LD_LIBRARY_PATH=\
          /usr/local/cuda/lib64:\${LD_LIBRARY_PATH}
else
        export LD_LIBRARY_PATH=/usr/local/cuda/lib64
fi
```

4.2 Get FFTW3 development libraries

The best way to install the development libraries is via your package manager. Under Ubuntu, this is done with the following command:

```
user@node:~$ sudo apt-get install libfftw3-dev
```

Unfortunately, the fftw3.h header file has a typedef statement that doesn't directly support the CUDA compiler. Fortunately it's a very simple fix. We have provided a copy of the patched header file with the source code which is available at: https://github.com/FreeCT/FreeCT_WFBP/blob/master/resources/patches/fftw3.h. To patch the header with the FreeCT-provided file use the following command (type in as one line, we have broken it up here for readability):

```
user@node:~$ sudo wget -O /usr/include/fftw3.h
  https://github.com/FreeCT/FreeCT_WFBP/blob/master/resources/patches/fftw3.h
```

If you wish to edit the file yourself, we are changing line 362 (may vary slightly) from:

```
&& !(defined(__ICC) || defined(__INTEL_COMPILER)) \
```

to

```
&& !(defined(__ICC) || defined(__INTEL_COMPILER) || defined(__CUDACC__)) \
```

Note that you will need to launch your editor using "sudo" if editing the line manually.

4.3 Get the source code

You will first need the FreeCT_Reader source code as this is where the code is held to read raw data files (this piece of software is kept separate from the rest since it is licensed differently). This can be downloaded from can be downloaded as a zip file from the GitHub project page (see section 7), or using wget:

```
wget https://github.com/FreeCT/FreeCT_Reader/archive/master.zip
```

The source code for FreeCT_wFBP can be downloaded as a zip file from the GitHub project page (see section 7), or using wget:

```
wget https://github.com/FreeCT/FreeCT_WFBP/archive/master.zip
```

Go ahead and extract it to the directory of your choosing.

4.4 Building and installing the reader library

Prior to building FreeCT_wFBP, you will need the reader library (and corresponding headers) installed on your system path. Once you have the source code downloaded, open a terminal window and navigate to the FreeCT_Reader project directory and run the "make" command. For example, if we have stored the code in ~/Code/FreeCT_Reader:

```
user@node:~$ cd ~/Code/FreeCT_Reader/
user@node:~/Code/FreeCT_Reader$ make
```

If everything is configured properly, then your output from the “make” command should look something like the following:

```
user@node:~/Code/FreeCT_Reader$ make
mkdir -p src/obj
make -C src ../libfct_read.a
make[1]: Entering directory ‘/home/john/Code/FreeCT_Reader/src’
g++ -g -c -o obj/binary.o binary.c
g++ -g -c -o obj/ctd.o ctd.c
g++ -g -c -o obj/dicom.o dicom.c
g++ -g -c -o obj/ptr.o ptr.c
g++ -g -c -o obj/siemens_ima.o siemens_ima.c cd obj && ar rcs ../../libfct_read.a *.o
make[1]: Leaving directory ‘/home/john/Code/FreeCT_Reader/src’
mkdir -p build
cp include/* build/
cp libfct_read.a build/
rm -f libfct_read.a
user@node:~/Code/FreeCT_Reader$
```

If the code has built properly, a directly called “build” will have been created. Inside there should be a file called “libfct_read.a” and several C header files (*.h). By default, the library is installed in /usr/lib/ and the headers are placed in /usr/include/fct/ however this will require root/sudo permissions. This is the recommended way to install this library and should only be done differently if users understand the linux compile toolchain fairly well. Please see “troubleshooting” at the end of this section if you do not have root permissions or have access to an administrator.

To install the libary type:

```
user@node:~/Code/FreeCT\_Reader/$ sudo make install
```

Enter your administrator password when prompted. You should see the following output if the library has successfully installed:

mkdir -p /usr/include/fct mkdir -p /usr/include/fct/include/ cp build/libfct_read.a /usr/lib/ mv build/fct_read.h /usr/include/fct/ cp build/*.h /usr/include/fct/include/

4.5 Building and installing FreeCT_wFBP

Building and installed the wFBP source code is a very similar process to the reader source code. First, navigate to the FreeCT_wFBP source directory and run the “make” command:

```
user@node:~/Code/FreeCT_wFBP$ make
make -C src ../test
make[1]: Entering directory ‘/home/user/Code/FreeCT_wFBP/src’
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/main.o main.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/setup.o setup.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/interp.o interp.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/rebin_filter.o rebin_filter.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/rebin_filter_cpu.o rebin_filter_cpu.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/backproject.o backproject.cu
nvcc -I../include/ -I/home/user/Public/reader_cuda/ -L/home/user/Public/reader_cuda/ -g -c -o obj/backproject_cpu.o backproject_cpu.cu
(cd obj && nvcc -L/home/user/Public/reader_cuda/ -o ../../test /home/user/Public/reader_cuda/read_raw_file.o *.o -lfftw3)
make[1]: Leaving directory ‘/home/user/Code/FreeCT_wFBP/src’
user@node:~/Code/FreeCT_wFBP$
```

If the compile process has worked properly, you should end up with an executable file called “FreeCT_wFBP” in the source directory.

While FreeCT_wFBP can now be used in this directory, we recommend using the “install” make target to install FreeCT_wFBP system-wide, and create a bash run script to call FreeCT_wFBP from any location with one command. Run the “make install” command as follows:

```
user@node:~/Code/FreeCT_wFBP$ sudo make install
```

The default installation directory is "/usr/local/FreeCT/FreeCT_WFBP/" however writing to this directory will require root privelges (i.e. the use of "sudo" in the command). This is perhaps not ideal for all users in which case the INSTALL_PATH environment variable should be changed at the beginning of the makefile to where the user wants. If you do not have access to an administrator account, we recommend changing line 1 of the makefile to (substituting your system user name for "username"):

```
INSTALL_PATH?=/home/username/Applications/
```

and changing line 2 to:

```
SCRIPT_PATH?=/home/username/bin/
```

The user will have to ensure that their $PATH variable contains "/home/username/bin/" in order to use the script generated by the makefile a general linux command.

4.6 Compiling and Installation Troubleshooting

Code will not compile First, have you installed and properly configured the CUDA toolkit, including configuring the system paths as described in section 4.1? We may sound like a broken record, but this is an essential step. You may need to restart your system after installation to ensure that you are using the newly installed driver.

Second, have you installed **and patched** libfftw3-dev? If not, please refer back to section 4.2.

Finally, have you compiled and installed the FreeCT_Reader library? If not, please refer back to section 4.4 above. **If you modified the installation location of the library** (not recommended), you will need to either (1) modify the FreeCT_wFBP source makefile (located in FreeCT_wFBP/src/) library arguments or (2) add the reader library/header install locations to your compiler's library and include search paths.

Lots of issues can cause code to not compile, however we have found most problems arise from linking external libraries. The best advice we can give is to look carefully at error messages coming out of the compiler. When the problem lies in linking an external library, often someone else has experienced them before in which case Google is your best resource for working out these issues.

Code will not install First, ensure that you are running "make install" with root permissions, or have configured your paths to the appropriate, non-system directories (this is done by editing the makefile in the top-level source directory, FreeCT_WFBP/makefile).

Code has installed but will not run Ensure that the directory to which you installed your run script is on the $PATH variable of your shell, and that the script correctly points to the installation directory.

Other issues? If you are following the instructions above on a Linux system and still experiencing problems, please contact us at freect.project@gmail.com with as much information as possible about the error you are experiencing. We want FreeCT_wFBP to be as smooth of an experience as possible and will do our best to help you work through any issues.

5 Using FreeCT_wFBP

While FreeCT_wFBP is relatively simple when considered against other reconstruction software, it will still appear somewhat complex to users new to the reconstruction process. We encourage users to *read this entire section before attempting any non-sample reconstructions* to ensure a base level of familiarity with how FreeCT_wFBP expects data and how FreeCT_wFBP is configured and called.

It will help immensely if the user is familiar with standard descriptors of diagnostic CT geometry (see the references in section 2 for a good starting point).

5.1 Projection data storage format

Modern diagnostic CT projection data can be thought of as a stack of two-dimensional images with dimensions $N_{channels} \times N_{rows}$ each acquired at some tube angle. We call each of these "images" a projection. This results in a three-dimensional array of size $N_{channels} \times N_{rows} \times N_{projections}$.

Throughout FreeCT_wFBP, our projection data is stored in memory indexing across channels first, then rows, then projections, however all arrays are single dimensional. Put another way, the channels are stored with stride 1, rows are stored with stride $N_{channels}$ and projections are stored with stride $N_{channels} \times N_{rows}$.

For example, if we wish to fetch the projection pixel from channel i, row j, and projection k, the array index for these coordinates would be:

$$index(i, j, k) = (N_{channels} * N_{rows}) * k + (N_{channels}) * j + i$$

and we would then reference our array with

$$raw\_array[index(i, j, k)] = projection\ ray\ value$$

It is important to understand this concept, since this is also how FreeCT_wFBP expects projection data to be stored.

FreeCT_wFBP currently offers support for projection data stored as binary files of floating point data, as well as an open format raw data file type from the Mayo Clinic. See section 5.1.2 for more details on this data format.

5.1.1 Binary files

Binary files should be stored on disk indexing first across channels, then rows, then projections, as described above. **All data should be written as single-precision float data**. The system on which FreeCT_wFBP was developed utilizes the little-endian convention and while there is no explicit endianness specified in the code, a user should make sure that the endianness of the data matches the endianness of the system used to compile FreeCT_wFBP.

If you're writing binary data from MATLAB, pay close attention to the fact that MATLAB uses column major indexing (i.e. columns of an array are stored linearly in memory). This means that your matrix in MATLAB would have dimensions $N_{channels} \times N_{rows} \times N_{projections}$ and each projection when displayed would be a tall, thin image for current diagnostic CT scanners. See Appendix A for more information about using MATLAB to manipulate and then save raw data.

5.1.2 Open format DICOM raw data

FreeCT_Reader has implemented a basic version of this reader, however through correspondence with the Mayo Clinic group, we have learned that the "standard" has recently changed. A journal article clarifying the standard has been submitted and we will update the reader function to reflect these changes upon its publication.

5.1.3 Proprietary data formats

FreeCT_WFBP and FreeCT_Reader do not support any proprietary data formats. We encourage users to utlitilize the open-format DICOM-based standard for raw data or binary file formats as described above.

5.2 PRM files

PRM files (short for parameter files) are how the user configures a reconstruction and the only argument that is passed to the FreeCT_wFBP executable aside from program execution options. At this point in the FreeCT_wFBP project, it is important that a user understand all aspects of the PRM files since they must be manually created for each reconstruction, however in the future we hope to have a more automated method of creating them.

A standard PRM file is a carefully formatted plain text file and will look like the following:

```
% Sample PRM file for reconstruction with FreeCT\_wFBP

RawDataDir:     /home/user/raw_data/
RawDataFile:    my_test_data.bin
Nrows:          16
CollSlicewidth: 1.2
StartPos:       245.0
EndPos:         200.0
SliceThickness: 1.2
TubeStartAngle: 180.0
PitchValue:     19.2
AcqFOV:         500.0
ReconFOV:       250.0
ReconKernel:    3 % Sharp kernel
Readings:       60000
Xorigin:        0.0
Yorigin:        0.0
Zffs:           0
Phiffs:         1
Scanner:        1
FileType:       1
FileSubType:    0
RawOffset:      0
Nx:             512
Ny:             512
```

Some important things to note about PRM file structure:

- Each parameter should be separated by a newline character,
- Each line takes the form "ParameterIdentifier: \tab(s) ParameterValue"
- All whitespace between the identifier and its value should be tabs or spaces. Multiple tab characters are OK between a parameter identifier and its value.
- Parameter files are commentable with "%" in the manner shown above. Do not place comments before or in between parameter/value specifications. They should be placed on their own lines, or at the end of a line.
- Parameter identifiers are case-sensitive, and
- FreeCT_wFBP expects all physical distance measurements to be in millimeters.

Some "sanity checking" is performed to make sure the software was able to configure settings properly from the given input file. If FreeCT finds a parameter out of a reasonable range (e.g. "Nrows" is zero or negative), it will not run. This usually indicates a parameter that was unable to be properly parsed.

Table 1 lists each parameter and gives a sample value and explanation of the parameter. All parameters listed in table 1 are required, however they may appear in any order in the PRM file. Several sample PRM files and raw data files are available on the FreeCT website.

5.3 Running a reconstruction

FreeCT_wFBP is run with the following command structure:

```
fct_wfbp [options] input_prm_file
```

So for example, to run the no flying focal spot sample provided on the FreeCT website, the command is:

```
user@node:~/Code/FreeCT_wFBP$ fct_wfbp samples/prms/n_ffs.prm
```

Current options available to the user are:

```
-v                Verbose mode.
-t                Testing mode. Will write a number of *.ct_test files to
                  the desktop at key points in the reconstruction. This is
                  useful for debugging and diagnosing
                  reconstruction problems. These are binary files of floats
                  stored in the format described in section 5.
--timing          Timing mode will output timing information to stdout
                  during reconstruction.
--benchmark       Saves hidden binary file of timing information for the
                  reconstruction to the desktop.  Works, but still under
                  development.  Not recommended for use as of 10/28/2015.
--no-gpu          Runs FreeCT_wFBP entirely on the CPU. If you do not have a
                  GPU installed in your machine, or your GPU is not capable
                  of running CUDA code, FreeCT_wFBP MUST be run
                  using this switch.
--device=i        Specify which CUDA device on which to run FreeCT_wFBP
                  (not necessary for most uses).
```

[1]Note that for non-binary files, we use the the table positions specified in the raw data file. For binary files, we assume the first projection occurs at table position 0.0 and the final projection occurs at table position $\frac{N_{proj}}{N_r} z_{rot}$ where N_{proj} is the total number of projections in the scan, N_r is the number of projections per rotation, and z_{rot} is the distance (in millimeters) the table moves per rotation.

Parameter	**Sample value(s)**	**Description**
RawDataDir:	/home/user/raw__data__dir/	Full path to directory containing raw projection data. Ensure that the path ends in the file separator for your system.
RawDataFile:	n__ffs.IMA	Case-sensitive file name of raw data file
Nrows:	16	Number of rows in the collimation value (e.g. if collimation is $16 \times 1.2mm$, $64 \times 0.6mm$, Nrows is 16 and 64 respectively)
CollSlicewidth:	1.2	The second term of your collimation, in millimeters. Note this is collimated slice width in your acquisition, not reconstruction slice thickness.
StartPos:	250	Location of the first slice to be reconstructed. [1]
EndPos:	265	Location of final slice to be reconstructed.[2] May be lower, higher or equal to the start position.
SliceThickness:	5.0	Reconstructed slice thickness and, currently, reconstruction pitch
TubeStartAngle:	180.0	Acquisition angle, in degrees, of the first projection in the raw data. This parameter is ignored for file formats that contain tube angle information (i.e. we read this paramter when using binary files, but not the DICOM raw data type.)
PitchValue:	19.2	Table travel per gantry rotation.
AcqFOV:	500.0	Acquisition field of view. For most clinical scanners this is 500 mm.
ReconFOV:	250.0	Reconstruction field of view. Reconstruction grid height and width, centered on Xorigin and Yorigin (see below).
ReconKernel:	-1, 1, 2, or 3	Reconstruction kernel selection. Current offerings are (-1) experimental kernel, (1) smooth (2) medium (3) sharp/ramp. Offerings will be expanded in the future.
Readings:	31456	Total number of projections contained in the raw data file. This is to ensure that we don't try and reconstruct outside of the available data.
Xorigin:	0.0, -12.5, 30.0, etc.	Spatial location of the horizontal reconstruction center in millimeters. Use for zoned reconstructions on objects away from the center of the scan. Should be less than ReconFOV/2.
Yorigin:	0.0, -12.5, 30.0 , etc.	Spatial location of the vertical reconstruction center in millimeters. Should also be less than ReconFOV/2.
Zffs:	0, or 1	Flag to tell FreeCT__wFBP if scan was acquired using Z flying focal spot. 0 means no z ffs was used, 1 means it was used.
Phiffs:	0, or 1	Flag to tell FreeCT__wFBP if scan was acquired using Phi flying focal spot. 0 means phi ffs was not used, 1 means it was used.
Scanner:	1,2, filename, full file path	1, and 2 correspond to hardcoded scanner geometries for the Siemens Definition AS, and Sensation 64 respectively. If user passes a filename, FreeCT__wFBP will fetch the scanner geometry from FreeCT__wFBP/resources/scanners/filename.
FileType:	0,1,2, or 3	Type of raw data we are reading. (0) Binary data (1) Unsupported proprietary format 1 (2) Unsupported proprietary format 2 (3) Unsupported proprietary format 3, and (4) DICOM open-format.
FileSubtype:	0	Parameter reserved for future use. Leave set to 0.
RawOffset:	23564	Byte offset to projection data. Only necessary if a binary file type contains a fixed header.
Nx:	512	Width of the reconstruction volume in pixels. Should be a multiple of 32. Standard clinical reconstructions are 512x512.
Ny:	512	Height of the reconstruction volume in pixels. Should be a multiple of 32. Standard clinical recosntructions are 512x512.

Table 1: Summary of parameter identifiers and their values for FreeCT__wFBP v0.0.2

To run the same no flying focal spot sample case with all options activated (i.e. verbose, testing, and no-gpu) the command would be:

```
user@node:~/Code/FreeCT_wFBP$ ./test -v -t --no-gpu samples/prms/n_ffs.prm
```

We do NOT currently support the ability to mix switches (i.e. ./test -vt –no-gpu prms/file.raw will not work since "-vt" is not a valid option).

Reconstruction data is currently output to the desktop in two files: image_data.txt and the value of "RawDataFile:" in your PRM file, with ".img" appended. Thus, if you entered "n_ffs.IMA" for your raw data file, your output file will be "n_ffs.IMA.img" saved to the Desktop. This is the primary reconstruction file containing only the slices requested and in the order they were requested is the latter of the two (n_ffs.IMA.img in our example). Image_data.txt is more of a debugging file than something that should be relied on for future releases. FreeCT_wFBP reconstructs in volumes of $N_x \times N_y \times 32$ voxels for efficiency/gpu reasons, however depending on the configuration, some of these slices may be thrown out in the final "*.img" file. Image_data.txt records ALL of the slices reconstructed (i.e. the number of slices will always be a multiple of 32), however not necessarily in the correct order. This can also be useful for debugging.

5.4 Scripting with FreeCT_wFBP

There are a myriad of different possibilities for scripting with FreeCT_wFBP. We have used FreeCT_wFBP extensively with MATLAB, and Bash. Any language that provides an API for system calls should work well with FreeCT_wFBP.

6 Raw Data

Raw data is a complex and difficult issue in diagnostic CT. Assuming the user can access the raw data file, all manufacturers currently use proprietary data formats that are often encoded in not-trivial ways. Through agreements with manufacturers, several groups (including our own) have worked to reverse engineer a very small subset of these data types for use in our own research, however due to the proprietary nature of the work, they are currently unavailable for release.

There is an ongoing effort from the Mayo Clinic to develop a library of freely available diagnostic CT projection data sets which will be stored in an open-format raw data file type based on the DICOM standard. Direct reading from this file-type is currently under development for FreeCT_wFBP and will be released in the near future as both part of the FreeCT_wFBP package, as well as an independent C library for other use. FreeCT_wFBP also supports the reading of raw projection data stored as a binary file of float data. More information on how data should be stored for reading can be found in section 5.

7 The GitHub Repository

Users wishing to develop with the code may want to use Git. The directory can also be cloned using Git (https://git-scm.com/):

```
git clone git@github.com:FreeCT/FreeCT_WFBP.git
```

Currently, all code can be found on Github at https://github.com/FreeCT/. For both the FreeCT_Reader and FreeCT_wFBP repositories, there are three main branches that are maintained:

1. Master - Contains “production” code that should be almost entirely stable. The goal is that this contains scientifically accurate code, that can be reliably used for research. This will be the best starting place for new users.
2. Develop - Contains mostly stable code, however will also contain new feature additions, some experimental work, etc. This is where you should go for the most recent developments in the code, however without the headache of truly experimental code. Most people interested in developing/forking FreeCT should work from this branch.
3. Nightly - Will contain work that is likely not suitable for all audiences. It is largely just to sync work between computers. This will truly have the latest updates, but may also contain modifications to otherwise stable code for testing purposes. No effort is made to ensure that this will run on anyone’s computer. This branch should be ignored by all but the most adventurous users.

8 Dependencies

One of the purposes in developing our own code was to remove the need for large quantities of external libraries. There are currently only two hard dependencies:

1. The Nvidia CUDA toolkit (https://developer.nvidia.com/cuda-downloads) - The CUDA toolkit is required to build the current version of FreeCT_wFBP, however the code can be run exclusively on a CPU without the CUDA driver (more information on this later). We plan to provide a CPU-exclusive version in the future that can be compiled with gcc.
2. FFTW3 (http://www.fftw.org/) - FFTW is a C-based fourier transform library utilized in the CPU filtering code. It is available though most, if not all, linux package managers. It can be built from source as well however paths will most likely need to be corrected in the makefile.

Furthermore, this code was developed under linux (Ubuntu 14.04 LTS) and relies on several core unix “includes.” These should all be available on any POSIX-like or POSIX-compliant system and are:

1. sys/types.h
2. pwd.h
3. unistd.h
4. math.h
5. complex.h
6. stdlib.h
7. stdio.h
8. cstdarg
9. regex.h

Note that sys/types.h, pwd.h, and unistd.h are used for path independence in the compiled version of the code. If these files are not available, hard coding of system paths (or another method of obtaining paths) should allow the user to port the software to any system.

9 Project Organization

The project contains three main directories: src, include, and resources.

- FreeCT_WFBP/src: contains the *.cu source files for the project. This includes host code for the GPU-based functions, and any other CPU code.
- FreeCT_WFBP/include: contains C header files and CUDA header files. C header files (*.h) are standard function prototypes and structure definitions. CUDA header files (*.cuh) contain the GPU kernel code. There is one C header file for every source file and one CUDA header file for source files that contain GPU code (these files are rebin_filter.cu, and backproject.cu with corresponding cuda headers rebin_filter.cuh and backproject.cuh).
- FreeCT_WFBP/resources: contains static data files such as scanner geometries and reconstruction kernels that are used in the reconstruction process. No code used explicitly by the FreeCT_wFBP program lives here.
- FreeCT_WFBP/doc: Contains the documentation for free CT, including this document.

Source files are broken up by reconstruction step performed. The "int main()" function is contained in *src/main.cu* which then makes the calls to all other functions. The flow of the code is as follows:

1. Parse input file and configure scanner geometry and reconstruction parameters: *include/recon_structs.h, include/setup.h, and src/setup.cu*
2. Extract raw data from file: *include/setup.h, include/setup.cu,* FreeCT_Reader (our library dedicated to reading CT projection data).
3. Parallel rebinning: *include/rebin_filter.h, include/rebin_filter_cpu.h, include/rebin_filter.cuh, src/rebin_filter.cu, and include/rebin_filter_cpu.cu*
4. Filtering: *include/rebin_filter.h, include/rebin_filter_cpu.h, include/rebin_filter.cuh, src/rebin_filter.cu, and include/rebin_filter_cpu.cu*
5. Backprojection: *include/backproject.h, include/backproject.cuh, include/backproject_cpu.h, src/backproject.cu, src/backproject_cpu.cu*
6. Write data to output file: *include/setup.h, src/setup.cu*

10 Possible future developments paths

The work listed here is work that we would like to do to grow FreeCT_wFBP, for which however we do not have specific plans. If you're potentially interested in developing for FreeCT, these would be so great things to work on to have the best chances of getting your code included in the project!

General:

- Speed, speed, speed! If you can get it running faster (while preserving results of course), we'll ALWAYS take a look at your work.
- Cross platform (other Linux/Unix, MacOS, Windows) implementation, perhaps with libraries like Boost (http://www.boost.org/),

- Multicore CPU implementation
- Documentation improvements, or instructions for installing on other systems.

Reconstruction kernels.

- Let's get as many as we can, as similar to what is used clinically as is possible!
- On-the-fly calculation and saving for new scanner geometries.

Multi-GPU work:

- Multi-GPU implementation (i.e. utilize mutliple GPUs for one reconstruction)
- GPU "job-scheduling" (i.e. run one reconstruction per GPU on a computer system with multiple GPUs).

A Appendix A: Saving Projection Data from MATLAB to binary files for use with FreeCT_wFBP

Under construction!