
Date: 2016/08/15 18:25:00
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: 土'山,上椙,松野,三宅,竹内
Subject: rhp

うえすぎさま、

GSJ/AIST のなかのです。予告していた CT-simulator の説明の前で何ですが、そのテストのために書いた parallel beam CT と cone beam CT 用の画像再構成プログラムを実際の X 線 CT 測定のデータを用いた画像再構成に使えるものに書き換えてみました。ここではその話をします。

(1)
CT simulator 用の画像再構成プログラムは浮動小数点数画素値の TIFF（float-TIFF）のファイルとして与えられた投影画像群を必要としますが、新しいプログラムは指定されたディレクトリに格納されている X 線 CT 測定のデータセット（テキストファイル output.log と、HiPic 画像 dark.img および "q*.img"）から画像再構成に必要な投影画像群を自動的に作成します。その処理を実行する汎用性の高い C 言語の関数のコード "rhp.[h,c]" を書きました：

関数 InitReadHiPic(char *dir, HiPic *hp)
　指定されたディレクトリの下の測定ログ（output.log）や HiPic 画像のファイル（dark.img と q*.img）の検索、ログの記述内容のスキャン、暗電流画像と入射 X 線強度（I0）画像の読み込みなどを行う。
ReadHiPic(HiPic *hp, int t)
　指定されたシーケンス番号の透過 X 線強度画像を読み出し、暗電流と I0 画像を使って補正した浮動小数点数画素値の X 線透過率画像（XT 画像）を作成する。
TermReadHiPic(HiPic *hp)
　関数 InitReadHiPic() が確保した計算機メモリを解放する。

(2)
これらの関数はインクルードファイル "rhp.h" の中で宣言されている型 HiPic の構造体の変数を引数として以下のようにして呼び出します。

```
#include "rhp.h"
...
HiPic   hp;             /* ポインタの宣言ではないことに注意 */
int     t, y, x;
...
InitReadHiPic( "CT測定のディレクトリ名", &hp );
...
for ( t = 0 ; t < hp.Nt ; t++ ) {
    ReadHiPic( &hp, t );
    for ( y = 0 ; y < hp.Ny ; y++ )
    for ( x = 0 ; x < hp.Nx ; x++ )
        /* 各画素の X 線透過率の値 hp.T[y][x] を使った処理 */
}
...
TermReadHiPic( &hp )
...
```

ただし、各画素の X 線透過率の値を格納する構造体 HiPic のメンバー **T はマクロ定数 FOM（float on memory）として宣言した型の浮動小数点数です。"rhp.h" に設定してある通り "rhp.c" のコンパイル時に FOM を自分で定義することが可能で、何も指定しなければ FOM = double です。この T 以外にも型 HiPic の構造体は多数のメンバーを擁しますが、普通に使用する分には上で例示した３個の型 int の整数値のメンバー（と **T）だけで十分だと思われます：

int Nx：XT 画像の横画素数
int Ny：XT 画像の縦画素数
int Nt：透過 X 線強度画像の枚数（== 作成可能な XT 画像の枚数）
FOM **T：XT 画像の各画素の浮動小数点数の X 線透過率の値

なお、X 線 CT の測定ログのファイル output.log には画像再構成で直接は使用しないサンプル回転軸の位置決め用の画像の情報も記録されています。"rhp.c" のコンパイル時にマクロ定数 ONLY_CT_VIEWS を「-DONLY_CT_VIEWS」として定義してやれば、その情報を読み飛ばす関数 InitReadHiPic() のコードになります。

(3)
"rhp.[h,c]" とそれらを使ったプログラムのソースファイル、および、それらのプログラムの 32 もしくは 64 ビット Windows 用の実行ファイルなどを以下の２個の書庫ファイルに入れておきました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.zip

これらの書庫ファイルに入れたテキストファイル install.txt の記述に従ってこれらの書庫ファイルに入れた６もしくは８個のプログラム（２個の画像再構成プログラムの Windows 用実行ファイルにはそれぞれ 32 ビット用と 64 ビット CPU 用のものがある）をインストールしてみて下さい。

(4)
書庫ファイルに入れたテキストファイル usage.txt に６個のプログラムの使用法などを記しておきました。その概略は以下の通りです。

プログラム hp2xt
　"rhp.[h,c]" の関数の使用例として書いた HiPic 画像のデータセットから X 線透過率画像を作成するプログラム。

cbp と fdk
　parallel beam CT と cone beam CT の画像再構成プログラム。

trim_float
　再構成画像の各スライスのトリミングを行うためのプログラム。
t2t_float
　float-TIFF 画像を integer-TIFF 画像に変換するプログラム
t2g_float
　float-TIFF 画像を animation-GIF に変換するプログラム

ただし、"*_float" はオマケで "rhp.[h,c]" を使っていません。CT-simulator 用の書庫ファイルに入れたものと同一のユーティリティプログラム３兄弟です。

(5)
画像再構成プログラム cbp と fdk で実際の X 線 CT 測定のデータを処理してみました。測定 160723e は先月のつちやまさんの FZP-CT 実験のもの、また、150303d は昨年３月に医療棟で行った cone beam CT の測定データです。

測定 160723e の画像再構成

```
# sample = Fo-1 (forsterite)
# X-ray energy = 8 keV
# width of XP-images = 800 pixels
# height of XP-images = 780 pixels
# views (steps of 180-degree rotation) = 1800
# Dr (detector interval) = 41.1 nm = 41.1e-7 cm
# Cr (center value) = 412

setenv THREADS 8        # Linux tcsh
export THREADS=8        # Linux bash
set THREADS=8           # Windows command prompt
```

```
        mkdir tg
        cbp raw 41.1e-7 412 0 tg/%03d.tif > tg.log
                775     780     41.1e-7

        xv tg/*.tif             # tg/ : 775*775*780 pixels

    150303d
        # sample = ?
        # X-ray energy = ? (polychromatic X-ray)
        # width of XP-images = 2048 pixels
        # height of XP-images = 2048 pixels
        # views (steps of 360-degree rotation) = 1200
        # SSD (source-sample distance, A) = 534.5 mm
        # SDD (source-detector distance, B) = 534.5 mm
        # Du (detector interval along horizontal direction) = 6.5 um = 6.5e-3 mm
        # Cu (center value along horizontal direction) = 1024
        # Dw (detector interval along vertical direction) = 6.5 um = 6.5e-3 mm
        # Cw (center value along vertical direction) = 1024

        setenv THREADS 8        # Linux tcsh
        export THREADS=8        # Linux bash
        set THREADS=8           # Windows command prompt

        mkdir tg
        fdk raw 534.5 534.5 6.5e-3 1024 6.5e-3 1024 0 tg/%04d.tif > tg.log
                2045    2023    6.500000e-03    6.500000e-03

        xv tg/*.tif             # tg/ : 2045*2045*2023 pixels
```

とりあえず以上です。

```
    P.S. 1
        "rhp.[h,c]" ではディレクトリの読み取りに "dirent.h" を使っています。
        MS C++ compiler や Windows の Intel C++ compiler では "dirent.h"
        を導入しておく必要がありますよね？
    P.S. 2
        昨年３月の cone beam CT 実験の直後に書いた FDK 法のプログラム
        "xp2tg_*" とは異なり、新しいプログラム fdk は画像再構成の処理と
        同時に X 線透過率画像の読み込みを行います（そのため、環境変数
        THREADS で指定した値に１を加えた個数のスレッドを実行します）。
        fdk は "xp2tg_*" に比べて少しだけ高速になりました。
```

```
----------------
2016/8/30
----------------
```

HiPic 画像のデータセットから X 線投影画像を作成するプログラム hp2xp を書いた。usage.txt に記したように、その使用法は hp2xt のものと概ね同じ。また、そのソースコードや Windows 用実行ファイルを "rhp.[h,c]" 用の書庫ファイル rhp.taz と rhp.zip に追加した。

```
-------------------------------------------------
Date:           2016/09/01 14:04:25
From:           Tsukasa NAKANO
To:             Kentaro Uesugi
Cc:             土'山,上椙,松野,三宅,竹内,星野
Subject:        rhp_オマケ_speed_of_cbp_and_fdk
Attached:       cbp+fdk.txt
-------------------------------------------------
```

うえすぎさま、
GSJ/AIST のなかのです。HiPic のデータセットの読み込みを行う "rhp.[h,c]" に関する 8/15 の E-mail （後に転記しました）で紹介したプログラム cbp と fdk による画像再構成の処理時間を測ってみました。その処理内容と結果はこの E-mail に添付したファイル cbp+fdk.txt に記した通りです。SPring-8 にある計算機 vrm で実行した cbp の時定数は 0.3430 nano sec. で、これは CPU 用の画像再構成プログラムとしてはかなり高速です。また、こちらの主力計算機 gsjgix で実行した fdk は従来の最速版プログラム xp2tg_3 の２割以上高速になっていました。つちやまさんのところに導入する cone beam CT-scanner にはこれを使えば OK ですね。とり急ぎ、

```
On Mon, 15 Aug 2016 18:25:00 +0900 Tsukasa NAKANO wrote:

> (5)
> 画像再構成プログラム cbp と fdk で実際の X 線 CT 測定のデータを処理して
> みました。測定 160723e は先月のつちやまさんの FZP-CT 実験のもの、また、
> 150303d は昨年３月に医療棟で行った cone beam CT の測定データです。
> ...
```

添付ファイル cbp+fdk.txt：

```
    execution time of "cbp" on vrm (Intel Xeon E5-2609v2 @ 2.50 GHz)

        setenv THREADS 8
        mkdir 160723e_tg
        stop_watch cbp 160723e_raw \
                       41.1e-7 412 0 \
                       160723e_tg/%03d.tif > 160723e_tg.log
            775 780     41.1e-7
            289.272059  # execution time in sec.

        --------
        memory usage = 1.8 GB
        time constant = 289.272059*10^9/1800/775/775/780 = 0.3430 nano sec.

    execution time of "fdk" on gsjgix (Intel Xeon E5-2687W @ 3.10 GHz)

        setenv THREADS 8
        mkdir 150303d_tg
        stop_watch fdk 150303d_raw \
                       534.5 534.5 6.5e-3 1024 6.5e-3 1024 0 \
                       150303d_tg/%04d.tif > 150303d_tg.log
            2045        2023    6.500000e-03    6.500000e-03
            2906.282495 # execution time in sec.

        --------
        memory usage = 31 GB
        time constant = 2906.282495*10^9/1200/2045/2045/2023 = 0.2826 nano sec.

        --------
        cf. execution time of "xp2tg_3"
            dataset     host    time in sec.
            150303a     ebisu   6426.935867
            150303b     vrm     9253.550598
            150303d     gsjvix  5402.920478
            150303e     gsjgix  4315.926146
            150303f     gsjgix  3854.464696
            150312a     ebisu   6657.838441
            150312b     vrm     8882.170843
            150312c     gsjvix  5332.430261
            150312d     gsjgix  3917.036423
            150312e     gsjgix  8628.554696     # 1200*2 views
```

2017/3/22

プログラム cbp と fdk のそれぞれを hp_cbp と hp_fdk に改名した。さらに、再構成した画像を浮動小数点数画素値の float-TIFF のファイルではなく、通常の整数画素値の integer-TIFF のファイルに格納することを起動時に選択できるようにこれらのコードを書き換えた。起動時にパラメータ BPS（bit per sample；画素値のビット数）を指定すると、hp_cbp と hp_fdk は再構成した浮動小数点数の CT 値を正規化・整数化して integer-TIFF のファイルに格納する。

hp_cbp と hp_fdk の起動法
```
hp_cbp HiPic/ Dr Or {layer1 layer2} RA0 {BPS} TG_format
hp_fdk HiPic/ A B Du Ou Dw Ow {layer1 layer2} RA0 {BPS} TG_format
```

cf. 以前の cbp と fdk の起動法
```
cbp HiPic/ Dr Or {layer1 layer2} RA0 TG_format
fdk HiPic/ A B Du Ou Dw Ow {layer1 layer2} RA0 TG_format
```

起動パラメータ BPS の指定なし、もしくは BPS として値０を指定した場合には hp_cbp と hp_fdk は従来通り再構成画像を float-TIFF のファイルに格納する。それに対して、BPS として０以外の整数値を指定すると、hp_cbp と hp_fdk は再構成した CT 値を以下の方法で正規化し、BPS の値の絶対値 |BPS| に応じたビット数の整数画素値に変換して integer-TIFF のファイルに書き込む。

BPS として正の整数値を指定した場合
　スライスごとに、そこでの最小値と最大値で CT 値を正規化する。
BPS が負の整数値の場合
　再構成した３次元画像全体の最小値と最大値で CT 値を正規化する。

つまり、hp_cbp や hp_fdk に起動パラメータ BPS として負の値を指定すれば、CT 値と整数画素値の対応関係がスライスすべてで同じ３次元 CT 画像（BPS が値 -16 なら、これはいわゆる「word 画像」）を得ることができる。

Date:　　2017/04/04 16:15:42
From:　　Tsukasa NAKANO
To:　　Kentaro Uesugi, MATSUNO Junya
Cc:　　Masayuki Uesugi, "TSUCHIYAMA, Akira"
Subject:　　imagej_open_ITEX
Attached:　　open_ITEX.java,
　　HandleExtraFileTypes.java（以下の［注］参照）

［注］
　添付ファイル open_ITEX.java と HandleExtraFileTypes.java は単純なテキストファイルです。それぞれの名前に ".txt" を付けたものを以下の場所に置いてあるので、ここでは表示しませんでした。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/Open_ITEX.java.txt
http://www-bl20.spring8.or.jp/~sp8ct/tmp/HandleExtraFileTypes.java.txt

うえすぎさま、
まつのさま、

GSJ/AIST のなかのです。SPring-8 の実験で使っている浜ホトのソフトウェア HiPic が出力する ITEX 形式の画像（"*.img"）を ImageJ で読み込むための plug-in を書きました。その JAVA のソースコード "open_ITEX.java"（単純なテキストファイルです）をこの E-mail に添付します。ImageJ で以下のようにすればインストールできます。

　ImageJ を起動
　Plugins → install で open_ITEX.java を選択
　コンパイル後の保存場所としてディレクトリ「Input-Output」を選択

ただし、これだけではファイルの drag & drop で ITEX 形式ファイル "*.img" を開くことはできません。ImageJ のディレクトリ plugins/Input-Output/ の下に置いてある画像形式の判定用の JAVA のコード HandleExtraFileTypes.java を書き換える必要があります。その上の適切な場所（後述）に以下の２行（１行にまとめても良いです）を埋め込むだけです。

```
if (name.endsWith(".img") && buf[0]==73 && buf[1]==77)
return tryPlugIn("Open_ITEX", path);
```

ぼくの手元の ImageJ 用の HandleExtraFileTypes.java をこの E-mail に添付しましたが、これはそちらの ImageJ（Fiji？）で使えるかどうかわかりません。そちらで使っている HandleExtraFileTypes.java を眺めて「拡張子 ".img" のファイルを処理しているコード」を探し出し、それよりも前の適当な場所に上記の２行を埋め込む方が良いと思います。

　ImageJ の plugins にはファイル名の拡張子 ".img" だけで画像形式を決めてしまうもの（Nifti_Reader）があるので、そのチェックの前に ITEX 形式画像かどうかをチェックしないとダメです。なお、ITEX 形式画像のチェックではファイル名の拡張子だけでなく、ファイルの先頭に埋め込まれている ITEX 形式に固有の「マジックナンバー」も調べます。

　この E-mail に添付した HandleExtraFileTypes.java では、116 行目に文字列「.img」がありました。その部分（コメント行でした）の直前に前記の２行を埋め込みました。

HandleExtraFileTypes.java を書き換えた後は、前記の open_ITEX.java の場合と同様にして、それを ImageJ のディレクトリ plugins/Input-Output/ の下にインストールして下さい。とり急ぎ、

Date: 2017/04/07 15:29:00
From: Tsukasa NAKANO
To: Kentaro Uesugi, MATSUNO Junya
Cc: Masayuki Uesugi, "TSUCHIYAMA, Akira"
Subject: Fw: imagej_open_ITEX
Attached: open_ITEX.java,
161121d_q0002.txt（これら２ファイルの表示は省略）

うえすぎさま、
まつのさま

なかのです。4/4 の E-mail で紹介した ImageJ で ITEX 形式画像を読み込むための plug-in「Open_ITEX.java」を書き換えましたので、それを送付します。ITEX 画像のファイルに埋め込まれているコメントを 「Image → Show Info」で閲覧できるようにしました。なお、その Window（Info Window）で「File → Save As」とすれば、コメントを含む画像の情報をテキストファイルに書き込むことができます。昨年 11 月の SPring-8 で撮った測定 161121d の q0002.img の情報を書き込んだファイル 161121d_q0002.txt をこの E-mail に添付します。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.pdf#page=48

とり急ぎ、

Date: 2017/04/08 13:12:36
From: Tsukasa NAKANO
To: Kentaro Uesugi, MATSUNO Junya
Cc: Masayuki Uesugi, "TSUCHIYAMA, Akira"
Subject: Re: imagej_open_ITEX

うえすぎさま、
まつのさま、

なかのです。何度もすみません。ImageJ で ITEX 形式画像を読み込むための plug-in "Open_ITEX" を修正（エラーチェックを強化）しました。そして、今後も書き換えるかもしれないので、そのコードを SPring-8 の FTP サイトに置くことにしました。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/Open_ITEX.java.txt

ファイル名の拡張子 ".txt" を取り除いてお使い下さい。なお、そのコードの後に JAVA のコメントとして ImageJ の画像形式判定用 plug-in
ImageJ/plugins/Input-Output/HandleExtraFileTypes.java
への Open_ITEX の設定の仕方を埋め込んでおきました。とり急ぎ、

Date: 2017/06/23 19:49:11
From: Tsukasa NAKANO
To: Kentaro Uesugi, MATSUNO Junya
Cc: "TSUCHIYAMA, Akira", Masayuki Uesugi
Subject: ImageJ_macros_rhp.txt

うえすぎさま、
まつのさま、

GSJ/AIST のなかのです。４月４、７、８日に差し上げた E-mails で紹介した ITEX 形式の画像を読み込むための plug-in "Open_ITEX" を用いて、SPring-8 の通常の X 線 CT 実験で得た測定画像のシーケンス（dark.img とログファイル output.log に記録されている順の "q*img"）を「image stack」としてまとめて読み込み表示する ImageJ の「マクロ言語」のコード "rhp.txt" を書きました。

http://www-bl20.spring8.or.jp/sp8ct/tmp/IJ_m_rhp.txt

このテキストファイルを既存の ImageJ のマクロ言語のスクリプト専用のディレクトリ "ImageJ/macros/" の下に（ファイル名を "rhp.txt" に改名して）コピーしておけば、以下のような端末からの入力で実行できます（ImageJ のメニューからも実行できると思いますが、その仕方をぼくは知りません）。

X 線 CT の測定データー式が入っているディレクトリを選択する場合
```
ImageJ  -macro  rhp
```

測定データー式が入っているディレクトリ（HiPicDir）を指定する場合
```
ImageJ  -macro  rhp  HiPicDir
```

なお、image stack にまとめた個々の画像のファイル名を表示する方法がわからなかったので、それ（シーケンス番号ごとの画像のファイル名）を output.log の残りの情報とともにログウィンドウに表示するようにしてあります。

また、rhp.txt を書く前に「指定したディレクトリの下にある一連の TIFF 画像などを image stack にまとめるマクロ言語コード is.txt」を書きました（実質的には１行だけなので、コードと言うほどのものではないですが）。

http://www-bl20.spring8.or.jp/sp8ct/tmp/IJ_m_is.txt

この is.txt のインストール法と起動法は rhp.txt のものとまったく同じです。こちらも御利用下さい。とり急ぎ、

2017/7/5

SPring-8 での X 線 CT 実験の測定データセットの処理用コード "rhp.[h,c]" と直接は関係しない話だが、ここに記しておく。float-TIFF 画像を integer-TIFF 画像に変換するプログラム t2t_float にそれらの画素値ヒストグラムのデータを抽出する機能を追加した。integer-TIFF 画像用のファイルやディレクトリのパス名として「その先頭にハイフン "-" を付けたもの」を指定すると t2t_float は画素値ヒストグラムのテキストデータを標準出力に書き出す。

t2t_float の従来の起動法（新しい t2t_float でも使用可能）
```
t2t_float  org.tif  {base  step}  BPS  new.tif
t2t_float  org/  nameFile  {base  step}  BPS  new/
```
→
タブコード区切りで以下の３個の値が並んでいる行を出力する。
[1] integer-TIFF 画像のパス名もしくはファイル名
[2,3] 画素値 F と I の対応関係を決める係数値 base と step
F = base + step・I
ただし、
F：float-TIFF 画像の上の浮動小数点数の画素値
I：integer-TIFF 画像の上の整数画素値（0≦ I<2^BPS)

t2t_float で画素値ヒストグラムを抽出する場合
```
t2t_float  org.tif  base  step  BPS  -{new.tif}
t2t_float  org/  nameFile  base  step  BPS  -{new/}
```
→
タブコード区切りで以下の４個の値が並んでいる行を出力する。
[1] integer-TIFF 画像の上の整数画素値、I
[2,3] ヒストグラムの bin の下端もしくは上端の値、
F[1,2] = base + step・(I [-,+] 1/2)
[4] その bin に落ちる画素の総数

このように t2t_float でヒストグラムを抽出する場合は float- と integer-TIFF 画像の画素値の対応関係を決める係数値 base と step の指定を省略できない。その反面、integer-TIFF 画像用のパス名の文字列としてハイフンだけを指定すると、画像ファイルを作らずにヒストグラムを調べることができる。

なお、最新版の t2t_float のソースコードの類は以下の６個の書庫ファイルのそれぞれに入っている。

以下のものには 32 bit Windows 用の実行ファイルも入れた：
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.zip
see
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.pdf

以下のものには 64 bit Windows 用の実行ファイルも入れた：
http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.zip
see
http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.pdf

以下のものには Windows 用の実行ファイルを入れなかった：
http://www-bl20.spring8.or.jp/~sp8ct/tmp/radon.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/raysum.taz
see
http://www-bl20.spring8.or.jp/~sp8ct/tmp/radon.pdf
http://www-bl20.spring8.or.jp/~sp8ct/tmp/raysum.pdf

Date: 2017/08/22 19:26:12
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: 松野,三宅,土'山,上椙,竹内,星野
Subject: rhp.[h,c]+hp_rc
Attached: rhp_rev.txt

うえすぎさま、

GSJ/AIST のなかのです。SPring-8 の（通常の）CT 実験で得た測定データから X 線透過率の画像を取り出す C 言語のコード "rhp.[h,c]" に関する話をします。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.pdf

(1)
この E-mail に添付した rhp_rev.txt の前半に記したもののような "rhp.[h,c]" を組み込んだプログラムの実行の前に下記の３個の環境変数のそれぞれの設定を行うと有効になる機能を "rhp.[h,c]" に追加しました。

RHP_O
コマンドラインから指定したディレクトリの下のファイル output.log からではなく、この環境変数に設定されたファイルから CT 測定のログデータを読み出す。ディレクトリ名を含むパス名を指定する必要がある。また、ハイフン "-" を設定すると標準入力からログデータを読み出す。ただし、その場合には下記の「４個のパラメータの数値が並んでいる行」ではない行（空行を含む）に遭遇するとログデータの終わりと見なす。
[1] 入射および透過 X 線強度画像のファイル番号
[2] 測定時刻（単位は秒）
[3] サンプル回転角（単位は度）
[4] 入射もしくは透過 X 線強度画像の識別用の０もしくは１
RHP_D
コマンドラインから指定したディレクトリの下の暗電流画像のファイル名を設定する。これは通常は dark.img だが、ぼくは大昔に d.img としていたこともあった。従来の "rhp.[h,c]" はこれら両方のファイルを受け付けていたが（ただし、両方がある場合は dark.img を優先）、新しい "rhp.[h,c]" は RHP_D によって陽に指定しない限り d.img を暗電流画像のファイルとして処理しない。
RHP_Q
コマンドラインから指定したディレクトリの下の入射および透過 X 線強度画像のファイル名（通常は "q*.img"）の先頭の１文字を設定する。

RHP_D と RHP_O を用いたプログラムの実行例を添付ファイル rhp_rev.txt の後半に示しました。なお、端末からの環境変数の具体的な設定法は以下の通りです。

環境変数 RHP_Q に文字 "a" を設定する方法
Windows のコマンドプロンプトで実行する場合：set RHP_Q=a
UNIX の C-shell：setenv RHP_Q a
UNIX の bash：export RHP_Q=a

(2)
CT 実験のデータから "rhp.[h,c]" を使って取り出した
回転角０の透過率画像を変換した X 線投影画像 P0 と
回転角 180 度の X 線投影画像を左右反転した画像 P180
を用いて CT 測定したサンプルの回転中心の位置（rotation center、RC；いわゆる「センター値」）を推定するプログラム hp_rc を書きました。既存の C-shell script "check.rc"（および、それを用いた HASPET 用の "center.csh"）と同じ処理を Windows の上で実行できます。その起動法は以下の通りです。

```
hp_rc  HiPic/  {y1  y2}  Rx  Ry  {RMSD.tif}
```

ただし、

HiPic/
: CT の測定データのファイル（output.log、dark.img と "q*.img"）が入っているディレクトリの名前

y1 と y2
: P0 と P180 の y（スライス）方向の範囲。これらの指定を省略すると投影画像の y 方向全域のデータを使って RC を推定する。

Rx と Ry
: P0 と P180 の重複領域における RMSD(Ox,Oy) のマッピング（後注参照）の範囲を決める x（横）および y（縦）方向の０～１のファクターの値。投影画像の横と縦の画素数をそれぞれ Nx と Ny とすると、hp_rc は

```
Ox = - Rx × Nx ～ + Rx × Nx
Oy = - Ry × Ny ～ + Ry × Ny
```

の範囲の RMSD(Ox,Oy) をマッピングして RC を推定する。

RMSD.tif
: RMSD(Ox,Oy) のマップの画像を格納する float-TIFF のファイルの名前。

注
: check.rc と同様に、hp_rc は P0 に付随した座標系における P180 の原点の位置（オフセット値 Ox と Oy）を変えながら両画像が重複している領域の画素の X 線投影値の「差の自乗の平均値の平方根（RMSD）」をマッピングし、その値 RMSD(Ox,Oy) が最小になっているオフセット値 Ox0 と Oy0 を探し出す。その配置で P0 と P180 が一致していると見なすと、RC の推定値 RC0 は以下の式で表されるはずである。

```
RC0 = ( Nx - 1 - Ox0 ) / 2
```

最終的には hp_rc は以下の２行・６個の値を標準出力に書き出します。なお、これら各行の３番目の推定値が１もしくは２番目の「RMSD マップの範囲を示す値」に等しい（RC0 = RC1 or RC0 = RC2 or Oy0 = Oy1 or Oy0 = Oy2）場合は hp_rc による RC の推定は失敗です。

１行目
: [1] RC1 = (Nx - 1 - Rx × Nx) / 2
: [2] RC2 = (Nx - 1 + Rx × Nx) / 2
: [3] RC の推定値、RC0 = (Nx-1 - Ox0) / 2

２行目
: [1] Oy1 = - Ry × Ny もしくは Oy1 = - Ry × (y2 - y1 + 1)
: [2] Oy2 = + Ry × Ny もしくは Oy2 = + Ry × (y2 - y1 + 1)
: [3] RMSD(Ox,Oy) が最小になっていた Oy の値、Oy0

添付ファイル rhp_rev.txt の後半に hp_rc、check.rc と center.csh による RC の推定処理の実例を示しました。ただし、ここで使った大昔の測定データ 040711j では暗電流画像のファイル名が d.img なので、それを環境変数 RHP_D に設定する必要があります。また、hp_rc の実行の例として環境変数 RHP_O に "-" を指定して標準入力からログデータを読み込ませる手法を試しています。hp_rc は回転角が０と 180 度の X 線投影画像しか使わないので、それらの計算に必要十分な output.log の最初（head）と最後（tail）の２行づつを標準入力から hp_rc に流し込んでいます。なお、HASPET 用の center.csh は測定データのファイルがディレクトリ raw の下にあることを仮定しているので、そうなるように 040711j_raw を raw にシンボリック・リンク（ln -s）しました。

とりあえず以上です。

添付ファイル rhp_rev.txt：

```
# programs using "rhp.[h,c]"

    http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.taz
    http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.zip

        rhp/
            hp2xt hp2xp hp_rc
            hp_cbp hp_fdk                                   # "*_[32,64].exe"

    http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.taz
    http://www-bl20.spring8.or.jp/~sp8ct/tmp/oct.zip

        oct/
            oct_cmp_0 oct_cmp_180 oct_xy
            oct_xt oct_xt_sa oct_xt_sbs
            oct_xp oct_xp_sa oct_xp_sbs
            oct_sg oct_sg_sa oct_sg_sbs                     # "*_[32,64].exe"
            oct_tg oct_tg_sa oct_tg_sbs                     # "*_[32,64].exe"
            oct_sg2tg_t oct_sg2tg_sa_t oct_sg2tg_sbs_t  # "*_[32,64].exe"
        oct/cuda/
            oct_sg2tg_g oct_sg2tg_sa_g oct_sg2tg_sbs_g  # for CUDA GPU

# estimation of rotation center

    wget http://www-bl20.spring8.or.jp/\~sp8ct/tmp/040711j_raw.taz
    tar xzf 040711j_raw.taz

    setenv RHP_D d.img
    hp_rc 040711j_raw 0.5 0.25
        249.5       749.5   498.5
        -180        180     -1

    setenv RHP_O -
    ( head -2 040711j_raw/output.log ; \
      tail -2 040711j_raw/output.log ) | hp_rc 040711j_raw 0.5 0.25
        249.5       749.5   498.5
        -180        180     -1

    check.rc 040711j_raw 0.5 0.25
        d.img = 040711j_raw/d.img
        i.img = 040711j_raw/q001.img
        t.img = 040711j_raw/q002.img at angle = 000.5000
        d.img = 040711j_raw/d.img
        i.img = 040711j_raw/q398.img
        t.img = 040711j_raw/q399.img at angle = 180.5000
        249.5       749.5   498.5   -180    180     -1

    ln -s 040711j_raw raw
    center.csh
        d.img = raw/d.img
        i.img = raw/q001.img
        t.img = raw/q002.img at angle = 000.5000
        d.img = raw/d.img
        i.img = raw/q398.img
        t.img = raw/q399.img at angle = 180.5000
        249.5       749.5   498.5   -180    180     -1
    cat center.log
        249.5       749.5   498.5   -180    180     -1
```

2017/12/1

SPring-8 の X 線 CT 実験で得た HiPic 画像のデータセットから X 線 total absorption (TA) の float-TIFF 画像を作成するプログラム hp_ta を書いた。

起動法
hp_ta HiPic/ TA.tif
起動パラメータ
HiPic：HiPic 画像のデータセットが入っているディレクトリの名前
TA.tif：TA の float-TIFF 画像のファイル名
標準出力に書き出される２個の数値
TA の最小値と最大値

TA の画像上の x 座標値は X 線 CT のサンプル回転角のステップ番号に、また、y 座標値は測定画像の y 座標値 == 再構成画像のスライス番号に対応している。

x 座標値 = 0 〜 Nx-1
左端（x = 0）：回転角０度
右端（x = Nx-1）：回転角 180 度
y 座標値 = 0 〜 Ny-1
上端（y = 0）：測定画像の上端もしくは上端のスライスの位置
下端（y = Ny-1）：測定画像の下端もしくは下端のスライスの位置

そして、その各画素には回転ステップごとの X 線投影値画像上の横一列の投影値の和が入っており、これは理想的には再構成画像のスライスの上の estimated LAC（CT 値）の総和に等しいはずである。

hp_ta のソースや 32 ビット Windows 用実行ファイルを "rhp.[h,c]" 用の書庫ファイルに追加しておいた。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.zip

Date: 2017/12/05 13:19:10
From: Tsukasa NAKANO
To: 上杉,松野,三宅,土'山
Cc: 竹内,杉本,北山,松本め,田口,野口,松本と,奥村し
Subject: HR-CT_steady_vertical_creep
Attached: ta_dz.pdf（以下の［注］参照）

［注］
添付ファイル ta_dz.pdf はこの E-mail の最後に記した書庫ファイルの中に入っているので、そちらをご覧下さい（本文も改変しました）。

みなさま、

GSJ/AIST のなかのです。先月の FZP-CT 実験で起きていたような測定サンプルの「定常な鉛直方向のクリープ」を補正して half-rotation (HR) CT の画像再構成を行うプログラムを書きました。これらは 11/23 と 11/29 の E-mails で紹介した C-shell scripts「測定番号.txt」において既存のプログラム群で実行していた補正処理を組み込んだ「CBP engine」によって画像再構成を行うプログラムです。「本番用」のものに加えてサンプル回転軸の位置（rotation center、RC）を変えた「テスト用」のプログラムを用意しました：

テスト用の再構成プログラム
```
hp_stg_[t,g]  HiPic/  Dr  RC_base  step  count  {Dz}  layer  RA0  STG/ > STG.log
```

本番用の画像再構成プログラム
```
hp_tg_[t,g]  HiPic/  Dr  RC  {Dz} {layer1  layer2}  RA0  TG/ > TG.log
```

ただし、CPU 用のマルチスレッドプログラム "*_t" と CUDA GPU 用の "*_g" のそれぞれの実行前に以下の環境変数の設定が必要です。

"*_t"
```
setenv  CBP_THREADS  スレッド数          # ログイン shell が tcsh
export  CBP_THREADS=スレッド数           # bash
set  CBP_THREADS=スレッド数              # Windows のコマンドプロンプト
```

"*_g"（複数の CUDA GPU を搭載している計算機の場合のみ）
```
setenv  CUDA_GPU  GPU 番号（０〜）
export  CUDA_GPU=GPU 番号（０〜）
set  CUDA_GPU  GPU 番号（０〜）
```

また、プログラムの起動パラメータの意味は以下の通りです：

HiPic/
HR-CT の測定データファイル（output.log、dark.img と "q*.img"）が入っているディレクトリの名前

Dr
測定画像の正方形画素もしくは再構成画像の立方体画素の辺長

RC_base、step および count
hp_stg_[t,g] は
RC = RC_base + step * { 0 〜 count - 1 }
のそれぞれの値を用いた１枚のスライス画像の再構成を順次行う。

RC
hp_tg_[t,g] はこの RC の値を用いて複数のスライス画像を再構成する。

Dz
サンプル回転角が０度と 180 度の X 線投影値画像のペアの「縦ズレ量」。
既存の C-shell script "check.rc" や "center.csh"、もしくは新しいプログラム hp_rc で推定した値を指定すれば良い：
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.pdf#page=5

この値の指定を省略すると Dz = 0 と見なされ、HR-CT 用の既存の画像再構成プログラムと同等な処理が行われる。

layer
hp_stg_[t,g] はこの位置のスライス画像を RC を変えて再構成する。
layer1 と layer2
hp_tg_[t,g] は layer1 ～ layer2 の位置のスライス画像を再構成する。
そして、これらの指定を省略すると可能なすべてのスライスを処理する。

RA0
サンプル回転角の度単位の初期値。通常は０を与えれば良い。この値の指定により画質の低下なしでスライス画像を面内回転することができる。

STG/
hp_stg_[t,g] が作成した１個の sinogram の画像「layer.tif」と、RC の値を変えて再構成した複数の画像のファイル「layer_RC.tif」を格納するディレクトリの名前。
STG.log
hp_stg_[t,g] が標準出力に書き出す「STG/ の下の各画像に関する値」が並んでいる行をリダイレクトするテキストファイルの名前：
１行目（３個の数値が並んでいる）
[1] sinogram のスライス番号（layer）
[2,3] その sinogram の上の投影値の最小値と最大値
２行目以降（４個の数値が並んでいる）
[1] 再構成画像のスライス番号（layer）
[2] その処理に使用した RC の値
[3,4] その画像の上の estimated LAC の最小値と最大値
TG/
hp_tg_[t,g] が再構成した複数のスライス画像のファイル "*.tif" を格納するディレクトリの名前。
TG.log
hp_tg_[t,g] が標準出力に書き出す「TG/ の下の各画像に関する３個の値」が並んでいる行をリダイレクトするテキストファイルの名前：
[1] 再構成画像のスライス番号
[2,3] その画像の上の estimated LAC の最小値と最大値

なお、hp_stg_[t,g] と hp_tg_[t,g] が出力するスライス画像のファイルはいずれも浮動小数点数の画素値の float TIFF です。また、先月の実験の場で話したように、０でない Dz の値を指定した場合のスライスの位置は通常の測定画像上の y 座標値が指すものとは異なります。下記の書庫ファイル中の ta_dz.pdf を御覧下さい。その上の灰色の長方形は左端がサンプル回転角０、右端が 180 度の X-ray total absorption（TA）の画像の領域で、３次元画像と見なした測定画像を水平側方から見たものに相当します。定常な鉛直方向のクリープの補正を行わなかった Dz = 0 の再構成画像ではスライス番号 z は測定画像の y 座標値に一致しますが、それを行うと z 軸は y 軸は別モノになります。可能なすべてのスライスを再構成するように hp_stg_[t,g] と hp_tg_[t,g] では ta_dz.pdf に示したような z 座標値（スライス番号）の設定の仕方にしました。そのため、測定画像の鉛直（y）方向の画素数を Ny とすると、
再構成が可能なスライスの枚数は Nz = Ny + | Dz |
であり、Dz ≠ 0 の場合、
z = 0 ～ | Dz | - 1 と z = Nz - | Dz | ～ Nz - 1
の範囲のスライスの画像は不完全なデータから再構成したものになります。

hp_stg_[t,g] および hp_tg_[t,g] のソースと Windows 用実行ファイルを以前に紹介した "rhp.[h,c]" 用の書庫ファイルに入れておきました：

http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.zip

---- 以下省略 ----

Date: 2021/08/20 16:07
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: 土'山，上椙，松野，三宅，竹内
Subject: SPring-8_XCT_documents

うえすぎさま、

GSJ/AIST のなかのです。つまらないことですが、あなたが書いた SPring-8 の X 線 CT 関連の「注意事項」のページ

http://www-bl20.spring8.or.jp/xct/soft/attention.html
（ページの表題は「ソフトの取り扱いに関して」）

の「CBP 法による画像最構成と一部の tiff 画像入出力に関するサブルーチンは旧地質調査所の中野 司 博士によって開発されたものを用いています」の部分に以下の２種類の説明書（PDFs）を引用していただけますか ?

(1)
CBP 法による X 線 CT 画像の再構成

http://www-bl20.spring8.or.jp/~sp8ct/tmp/CBP.A4.pdf
もしくは
http://www-bl20.spring8.or.jp/~sp8ct/tmp/CBP.A5.pdf
（これらは印刷のページ割り当て法が違うだけです）

(2)
float TIFF の書き込み（および読み込み）用サブルーチン

http://www-bl20.spring8.or.jp/~sp8ct/tmp/float.pdf

これらのうちの (1) は SPring-8 での X 線 CT 実験が始まる前に書いた（20 世紀の）文章です。やや時代遅れの部分もありますが、SPring-8 の X 線 CT 用に書いた僕のコードはすべてこれを基本としています。また、(2) はハヤブサ（初号機）の試料解析の頃に書いたもので、僕の「晩年」のコードで大活躍しています。

それから、あなたはこれを使っていないと思いますが、以下の説明書の PDF も引用していただけるとありがたい。

(3)
X 線 CT 用に撮影した HiPic 形式の画像群から X 線透過率の画像群を抽出するサブルーチン

http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.pdf

よろしく御検討下さい。とり急ぎ、