Date: Mon, 12 Mar 2018 14:18:54 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: "TSUCHIYAMA, Akira", MATSUNO Junya, Masayuki Uesugi
Subject: Re: FDK_ver.3_(xp2tg_3)_bug_fix

うえすぎさま、

GSJ/AIST のなかのです。返事が遅くなりました。

> 産総研の不調はその後いかがでしょうか。

インターネットへの接続はまだ全面解除されていません。
E-mail のやりとりは MS の outlook.office の WEB 経由のみ
Windows Update は解除されたが、Linux のものは制限付き
それ以外の HTTP(s) や SSH 接続は遮断されたまま。

どうしても必要だった www-bl20 への書庫ファイルのアップロードは奥さんの部門が持っている portable WiFi router を借りて行った。

> S/N 的にちょっと厳しいみたいなので、投影像あたりで処理をしてから再構成をしようと思い、
> fdk ではなく、xp2tg_3 を使っています。

仕様書 http://www-bl20.spring8.or.jp/~sp8ct/tmp/fdk.pdf に記した FDK 法を用いた画像再構成用のプログラムには以下の 3 種類のものがあります：

上杉君が言っている「fdk」は［3］の「hp_fdk」のことですね？

［1］以下の書庫ファイルに入っている xp2tg_[1-4]
http://www-bl20.spring8.or.jp/~sp8ct/tmp/fdk.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/fdk.zip
［2］以下の書庫ファイルに入っている fdk
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
［3］以下の書庫ファイルに入っている hp_fdk
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

これらのプログラムは入出力する「画像」の種類が異なります：

入力（投影画像もしくは測定した生の X 線強度画像）
[1] ぼくのプログラムでしか処理できない binary 形式の投影画像
[2] 浮動小数点数画素値の TIFF（float TIFF）の投影画像
[3] 測定で得た HiPic 形式の生の X 線強度画像
出力（FDK 法で再構成した 3 次元画像）
[1] 整数画素値の TIFF（integer TIFF）に変換したもの
[2,3] 起動時の指定に応じて integer もしくは float TIFF

ということで、投影像の前処理は float TIFF に変換して行った方が良いのでは？
Irfan view や ImageJ を使えば float TIFF を表示（および処理）できる。
[2] の「fdk」を使えばそれらの処理済みの投影画像から画像再構成できる。

> この xp2tg_3 に関して質問なのですが
> 1. 引数の単位はすべて mm でしょうか？(pixel 以外のやつ)

起動パラメータの値 A、B、Du、Ou、Dw と Ow の単位ですね？ これらすべてを同じ単位で指定するのであれば、それを mm としても問題ありません。ただし、その場合には再構成画像の値（CT 値）の単位が 1/mm になります。

> 2. 結果を tif_float で出力したいのですが、単純に sif.c を sif_f.c にして
> コンパイルすれば良いんですかね？（もちろんコールする関数は入れ替えます。…）

いいえ。TIFF ファイルへの書き込み用関数に渡す画像（画素値の 2 次元配列）の型が違うのでコードの（大幅な）書き換えが必要です。

integer TIFF の書き込み関数：

```
void   StoreImageFile(
    char   *path,          /* ファイルのパス名 */
    int    Nx,             /* 画像の横画素数 */
    int    Ny,             /* 画像の縦画素数 */
    int    BPS,            /* 整数画素値のビット数 */
    Cell   **cell,         /* 整数画素値の 2 次元配列 */
    char   *desc           /* TIFF image description の文字列 */
);
```

ただし、デフォルトの設定は「Cell = unsigned short」です。

float TIFF の書き込み関数：

```
void    StoreImageFile_Float(
        char    *path,          /* 整数画像の場合と同じ */
        int     Nx,             /* 整数画像の場合と同じ */
        int     Ny,             /* 整数画像の場合と同じ */
        FOM     **fom,          /* 浮動小数点数画素値の 2 次元配列 */
        char    *desc           /* 整数画像の場合と同じ */
);
```

ただし、デフォルトの設定は「FOM = double」です。

とり急ぎ、

On 2018/03/10 23:40 Kentaro UESUGI wrote:
>
> 中野さん
>
> 上杉です
>
> 産総研の不調はその後いかがでしょうか。（関西センターのユーザーさんからも悲鳴が届きました）
>
> MFX-CT ですがぼちぼち出来るようになってきました。
> S/N 的にちょっと厳しいみたいなので、投影像あたりで処理をしてから再構成をしようと思い、
> fdk ではなく、xp2tg_3 を使っています。で、この xp2tg_3 に関して質問なのですが
>
> 1. 引数の単位はすべて mm でしょうか？(pixel 以外のやつ)
> 2. 結果を tif_float で出力したいのですが、単純に
> sif.c を sif_f.c にしてコンパイルすれば良いんですかね？
> （もちろんコールする関数は入れ替えます。整数が必要なときはあとで t2t_float で変換する）
>
> あと土`山さんに質問なのですが、東北大の装置で撮ったデータは放射光のデータのように数値的に
> 何か処理していますか？ それとも、適当なコントラスト調整だけでしょうか？

Date: Wed, 14 Mar 2018 10:48:04 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: "TSUCHIYAMA, Akira", MATSUNO Junya, Masayuki Uesugi
Subject: Re: FDK_ver.3_(xp2tg_3)_bug_fix

うえすぎさま、

GSJ/AIST のなかのです。すみません。―昨日の E-mail に間違ったことを書きました。

On Mon, 12 Mar 2018 14:18:54 +0900 Tsukasa NAKANO wrote:

> > この xp2tg_3 に関して質問なのですが
> > 1. 引数の単位はすべて mm でしょうか？(pixel 以外のやつ)
>
> 起動パラメータの値 A、B、Du、Ou、Dw と Ow の単位ですね？ これらすべてを同じ単位で指定する
> のであれば、それを mm としても問題ありません。ただし、その場合には再構成画像の値（CT 値）の
> 単位が 1/mm になります。

起動パラメータのうちの Ou と Ow の値は画素幅単位の「センター値」でした。それら以外の実寸の値 A、B、Du（FDK 法の仕様書では δu になっている）と Dw（δw）の単位は mm で問題ありません。

それから、言い忘れていましたが、プログラム fdk と hp_fdk は X 線透過率や強度の画像ファイルの入力と画像再構成を並列に実行するので、それを行わないプログラム xp2tg_[1-4] よりも処理が高速なはずです。
とり急ぎ、

Date: Mon, 02 Apr 2018 16:06:18 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: "TSUCHIYAMA, Akira", MATSUNO Junya, Masayuki Uesugi
Subject: Re: FW: FDK_ver.3_(xp2tg_3)_bug_fix

うえすぎさま、

GSJ/AIST のなかのです。

(0)
AIST からのインターネットアクセスは 3/29 から制限付きで可能になりました：
　就業時間内（8:00〜17:00）のみ（夜間・休日はダメ）
　DHCP 接続している計算機からのみ（固定 IP のサーバ機はダメ）

転送速度は AIST-LAN の値 100 Mbps≒11 MB/sec. に規定されるので、総量が (17-8)*3600* 11/1000 ≒ 356 GB までのデータなら vrm.spring8.or.jp などのそちらの計算機からダウンロードできます。

(1)
FDK 法の仕様書 http://www-bl20.spring8.or.jp/~sp8ct/tmp/fdk.pdf に記したように、ぼくが書いた「FDK 法による cone beam CT の画像再構成プログラム」では cone beam CT として（最も）単純な装置の配置を仮定しています：
　[1] サンプル回転軸は検出器面上の縦軸（w 軸）と平行。
　[2] 点と仮定した光源と「照明中心」を結ぶ線上にサンプル回転軸がある。ただし、照明中心とは点光源から検出器面に下ろした垂線の足。

これらのうちの [1] は通常の parallel beam CT でも仮定しているので、cone beam CT でも高精度に実現できていると思います。問題は [2] で、その実現や確認は難しい。これはサンプルの載せ換えのたびに確認したい仮定ですが、その簡便な方法をちょっと思いつきません。実用的には、サンプルステージの移動に使うステッピングモータの再現性を信じるしかない？：
　[a] ステージは直交する u、v、w の 3 軸方向に移動できるとする。ただし、
　　u 軸方向：検出器面上の横軸と平行な方向
　　v 軸方向：検出器面に垂直な方向
　　w 軸方向：検出器面上の縦軸と平行な方向
　[b] 回転軸の位置に置いた真球のサンプルを撮影し、v 軸方向にステージを移動して拡大・縮小しても検出器上の像の形状が真円でかつその中心が動かないステッピングモータの表示値（公称値）u0 と w0 を調べる。
　[c] 本番のサンプルの撮影時もこれらの公称値 u0 と w0 が指すステッピングモータの位置を「ホームポジション」とする。

(2)

ところで、3/13 のつちやまさん宛の件名「FW: reticulite」の E-mail で紹介した「2013/5 に中村美千彦さんが東北大学の μ-focus（cone beam）CT 装置 scanXmate で測定したデータ」からぼくが書いた FDK 法のプログラム hp_fdk を使って reticulite の CT 画像を再構成してみました。scanXmate の照明中心の位置が不明なのでそれを様々に変えてみましたが、上杉君の場合と同様にまともな CT 画像を再構成できませんでした。そこで、以前に紹介した raysum 方式の cone beam CT simulator

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

を使って scanXmate で再構成した CT 画像を再撮影し、それによって得た X 線透過率画像を scanXmate で実際に撮影したものと比較しました。この E-mail に添付した 1305_xt.pdf を御覧下さい：

xt/0000.tif

scanXmate で実際に撮影した回転角が 0 度の方向の X 線透過率画像

xp/000.tif、xp/015.tif および xp/345.tif

scanXmate で再構成した CT 画像を cone beam CT simulator「cb_ss」で再撮影した回転角が 0、15 と 345（= -15）度の方向の X 線投影値画像を変換した X 線透過率画像。

1305_xt.pdf の実測した xt/0000.tif とそれをシミュレートした xp/000.tif を比較するとサンプルの全体像は概ね同じですが、上下中央の左端と右端の像が若干異なっています：

xt/0000.tif の上下中央の左端にある「カニの鋏」状の像

xp/000.tif にはないが、xp/345.tif にはある。

xt/0000.tif の上下中央の右端の上下 2 個あるコブ

xp/000.tif では「見え過ぎ」で、xp/015.tif のものと似ている。

このようなサンプルの透過像の左右端の違いを「サンプルの回転」と「cone beam CT の X 線光路の配置」を考慮して説明した図 scanXmate.pdf をこの E-mail に添付しました。結論だけを言うと、scanXmate の装置のレイアウトはぼくが FDK 法で仮定していた simple fan- or cone beam CT のものとは異なり、サンプル回転軸が「光源と照明中心を結ぶ線」の上に位置していないようです。

上杉君の cone beam CT 装置もこうなっているのでは？

(4)

サンプル回転軸が「光源と照明中心を結ぶ線」の上にない fan beam CT の画像再構成の手法については以前に考察済みです：

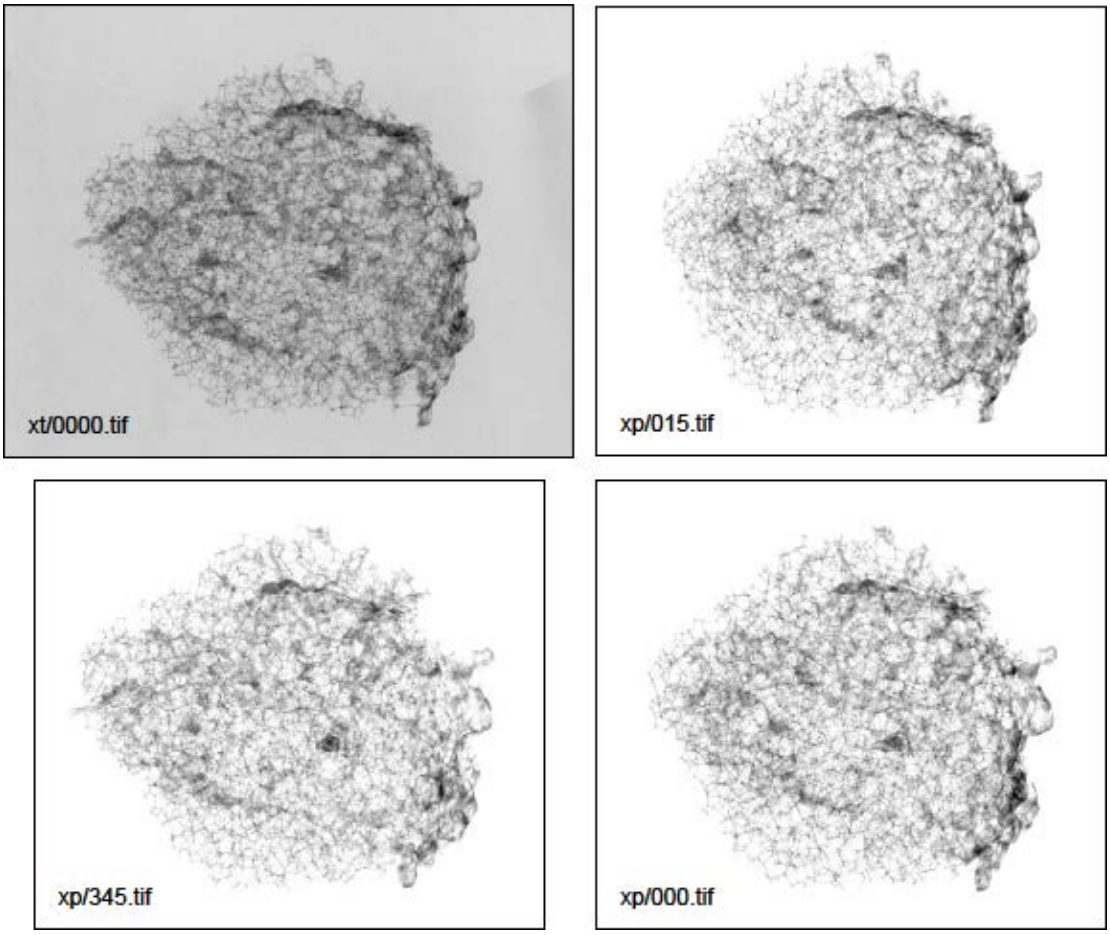
http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=3

この場合も CBP 法で画像再構成することが可能です。そして、cone beam CT に対する FDK 法は fan beam CT に対する CBP 法を（理論的な根拠無しに）拡張したものなので、上記の文書 fbct.pdf の最後に記した計算式で cone beam CT の画像再構成が可能かもしれません。それを行うプログラム hp_fdk_rc を書いてみましたが、...。今現在、その動作テストのためにサンプル回転軸が「光源と照明中心を結ぶ線」上にない cone beam CT をシミュレートするプログラム cb_ss_rc を書いています。

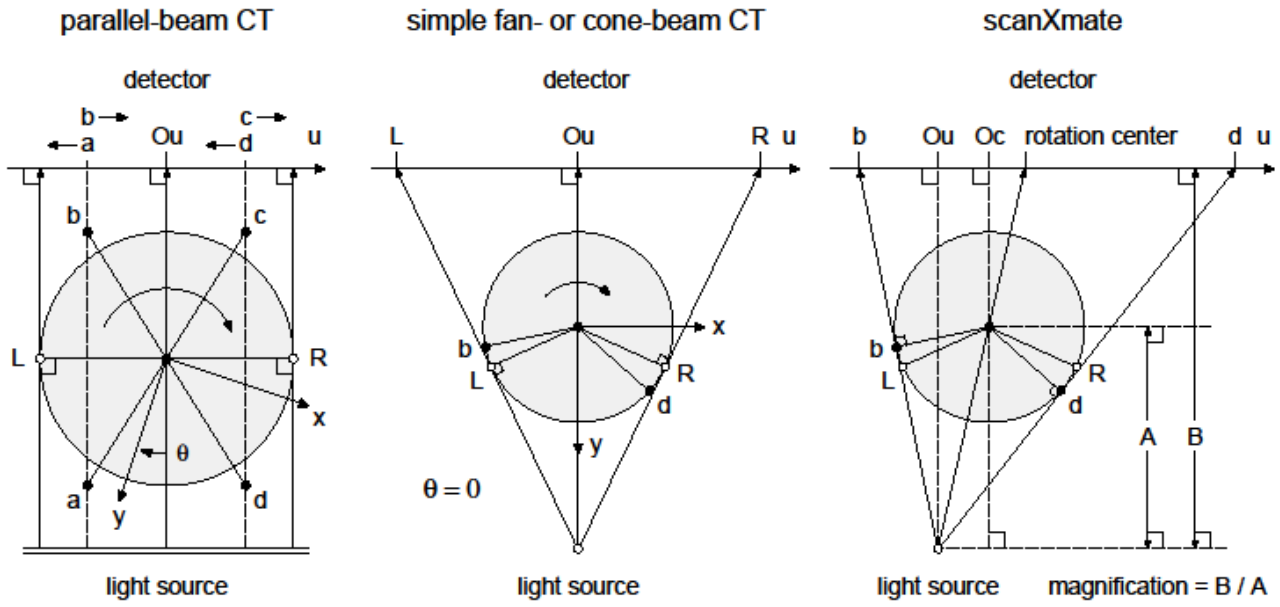
とり急ぎ、

On Fri, 30 Mar 2018 21:08 +0900 Kentaro UESUGI wrote:
>
> 中野さん
>
> 上杉です
>
> どうにも画像再構成がうまく行きません。
> (中心付近は良いのだが、周辺部に行くほどおかしな像になります)
> そちらにデータ一部をお送りして検討して頂きたいのですが、ＨＤＤの送付先教えて
> 頂けませんでしょうか？ （まだ産総研から外へのアクセスは難しいのですよね？）
>
> Kentaro UESUGI wrote:
>>
>> 中野さん
>>
>> 上杉です
>>
>> 報告遅れました。先週の終わりくらいに無事にこちらの変換ソフトの改造が完了しました。
>> 無事に再構成は出来ています。tif_float に変換するのは意外と便利だなと言う印象です。
>> （2017 年にいくつかアホな計測方法の試験をしたのですが、それの検証に使えそう）
>> もしかしたら SPring-8 の方もこの方式に変えちゃうかもしれませんね。

添付ファイル 1305_xt.pdf



添付ファイル scanXmate.pdf



Date: Wed, 18 Apr 2018 12:26:06 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: MATSUNO Junya, Masayuki Uesugi, "TSUCHIYAMA, Akira"
Subject: modified_FDK

うえすぎさま、

GSJ/AIST のなかのです。4/2 の E-mail でコメントした
　fan beam CT（FB-CT）や cone beam CT（CB-CT）において
　サンプル回転軸が「光源と照明中心を結ぶ線」の上にない場合を
　シミュレートする FB-CT および CB-CT simulator のプログラムと、
　それらで得たものや CB-CT の実測データ用の画像再構成プログラム
に関する話の続きです。結論を先に言うと、
　サンプル回転軸が「光源と照明中心を結ぶ線」の上にない場合も
　　http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=4
　に記した式で FB-CT や CB-CT の画像再構成を行うことができる
です。

On Mon, 02 Apr 2018 16:06:18 +0900 Tsukasa NAKANO wrote:

> サンプル回転軸が「光源と照明中心を結ぶ線」の上にない fan beam CT の画像再構成の手法については
> 以前に考察済みです：
>　　http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=3
> この場合も CBP 法で画像再構成することが可能です。そして、cone beam CT に対する FDK 法は fan
> beam CT に対する CBP 法を（理論的な根拠無しに）拡張したものなので、上記の文書 fbct.pdf の最後に
> 記した計算式で cone beam CT の画像再構成が可能かもしれません。それを行うプログラム hp_fdk_rc を
> 書いてみましたが、...。今現在、その動作テストのためにサンプル回転軸が「光源と照明中心を結ぶ線」
> 上にない cone beam CT をシミュレートするプログラム cb_ss_rc を書いています。

(0)
4/2 の E-mail 以降に以下のことを行ないました。

　注
　　ここでは FB-CT と CB-CT の照明中心（点と仮定した光源から検出器面に下ろした垂線の足；
　　illumination center）を IC と略記します。

　[0] parallel beam CT（PB-CT）、FB-CT および CB-CT の simulator や、それらが出力した X 線投影
　　値画像から画像再構成を行う既存の複数のプログラムのコードのそれぞれに修正を加えました。
　[1] PB-CT の新しい simulator プログラム pb_float を書きました。

[2] サンプル回転軸が「光源と IC を結ぶ線」の上にない FB-CT と CB-CT の simulator プログラム fb_float と cb_float を新たに書きました。なお、4/2 の E-mail ではこの新しい CB-CT simulator プログラムを cb_ss_rc と呼んでいましたが、その名前を cb_float に改めました。

[3] fb_float や cb_float で得た X 線投影値画像から画像再構成を行うため、ぼくが導出した FB-CT 用の CBP 法の式やそれを拡張した CB-CT 用の FDK 法（modified FDK 法）の式などを既存の画像再構成プログラム fb_cbp と fdk に組み込みました。

[4] CB-CT の装置で実測した HiPic 形式のデータファイルから画像再構成を行う既存のプログラム hp_fdk を廃して、4/2 の E-mail で言及した modified FDK 法で画像再構成を行うプログラム hp_fdk_rc を hp_fdk に改名しました。ただし、今回排したプログラム（hp_fdk_org と呼びます）のコードのファイルを hp_fdk.c.org に改名して残しておきました。

(1)

以上のような修正・作成したプログラムのコードなどの一式を以前に紹介した以下の書庫ファイルに入れておきました。

注

下記の書庫ファイル radon.taz に入れてある 3 つのコードはそれぞれ書庫ファイル raysum.taz の中の同名のものと同内容のファイルです。

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

fdk/hp_xp.c	A
fdk/xp2tg_[0-4].c	A

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

radon/pb_cbp.c	A
radon/fb_cbp.c	B
radon/fdk.c	B

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

raysum/cb_ss.c	B
raysum/pb_float.c	D
raysum/fb_float.c	D
raysum/cb_float.c	D
raysum/pb_cbp.c	A
raysum/fb_cbp.c	B
raysum/fdk.c	B

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

rhp/hp_fdk.c.org	B
rhp/hp_fdk.c	C

ただし、上記のコードのファイル名の後の記号 A～D の意味は以下の通りです。

A: これらにはプログラムの起動法に影響しない修正を加えた。特に、書庫ファイル "fdk.*" の中のものは error message の修正を行っただけ。
B: 新機能の追加などの大修正を加えたが、それを有効にする起動法をオプション扱いにしたので、以前のものと同じ方法で起動することも可能。
C: これは書庫ファイル "rhp.*" に入っている前述の hp_fdk のコードのみ。新機能の追加などによりプログラムの起動法が以前のものとは異なる。
D: 新たに書いたプログラムのコード。

(2)

新たに書いた PB-CT、FB-CT および CB-CT simulator 用のプログラム "*_float" の起動法とそれらが標準エラー出力に書き出す画像再構成用のパラメータは以下の通りです。以前に一連の E-mails
http://www-bl20.spring8.or.jp/~sp8ct/tmp/raysum.pdf
で紹介した旧版の CT simulator プログラム pb_3d、fb_3d および cb_ss の起動法なども併記しましたので、新旧のプログラムが取り扱うパラメータの違いに注意して下さい。

PB-CT simulator

```
旧      pb_3d  TG/  NBS  interval  {views  {start  end}}  XP_format > XP.log
        →      Nu    Nw    views    Du      Ou
新      pb_float  TG/  NF  interval  {views  {start  end}}  RA0  XP_format > XP.log
        →      Nu    Nw    views    (改行)
               Du    Ou    RA0
```

FB-CT simulator

```
旧      fb_3d  TG/  NBS  interval  SSD  ¥
               {views  {start  end}}  XP_format  > XP.log
        →      Nu    Nw    views    SSD    Du      Ou
新      fb_float  TG/  NF  interval  SSD  ORC  ¥
                {views  {start  end}}  RA0  XP_format  > XP.log
        →      Nu    Nw    views    (改行)
               SSD   ORC   Du      Ou      RA0
```

CB-CT simulator

```
旧      cb_ss  TG/  NBS  lxy  lz  SSD  ¥
               {views  {start  end}}  XP_format  > XP.log
        →      Nu    Nw    views    SSD    Du      Ou      Dw      Ow
新      cb_float  TG/  NF  lxy  lz  SSD  ORC  OSC  ¥
                {views  {start  end}}  RA0  XP_format  > XP.log
        →      Nu    Nw    views    (改行)
               SSD   ORC   Du      Ou      Dw      Ow      RA0
```

これらの起動パラメータと画像再構成用パラメータの意味は以下の通りです。

TG/

撮影する 3 次元画像のスライスのファイルが入っているディレクトリの名前。旧版の simulator プログラムは integer TIFF の画像しか処理できないが、新版は integer TIFF と float TIFF の両方を処理できる。

NBS

3 次元画像を構成するスライス画像のそれぞれのファイル名（name）と、その画像上の整数画素値 i と実際の値 f の関係式 f = base + step × I の係数値 base と step を各行に記した NBS file の名前。ハイフン "-" を指定すると TG/ の下のファイルすべてを base = 0 かつ step = 1 のスライス画像と見なした 3 次元画像の読み込みを行う。

NF

3 次元画像のスライス画像のファイル名を列記した name file の名前。"-" を指定すると TG/ の下のファイルすべてをスライス画像と見なす。

interval、Ixy および Iz

3 次元画像を構成する画素の辺長の「実寸」の値。

SSD

光源とサンプル回転軸の間の「実寸」の距離（source-sample distance）。

ORC

「光源と IC を結ぶ線」から測ったサンプル回転軸の位置の「実寸」のオフセットの値。旧版の FB-CT および CB-CT simulator では ORC = 0 を仮定している。なお、FB-CT と CB-CT の top view の装置の概念図 http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=3 ではこの値を記号「C」で表していたが、それは他でも使っていたのでここでは「ORC」にした。

OSC

「光源と IC を結ぶ線」から測った「撮影する 3 次元画像の中央のスライスの位置」のスライス幅を単位とするオフセットの値。旧版の CB-CT simulator では OSC = 0 を仮定している。

views、start および end

サンプルの 180 度（PB-CT の場合）もしくは 360 度（FB-CT と CB-CT）回転の総ステップ数とそのステップの初期値および終了値。

RA0

サンプル回転角の度単位の初期値。旧版の simulators では RA0 = 0 を仮定している。

XP_format

撮影した投影値画像を入れる float TIFF のファイル名のフォーマット。

XP.log

撮影した投影値画像それぞれのサンプル回転角に応じた投影番号（0 〜）、投影値の最小値と最大値の 3 個の値がタブコード区切りで書き込まれるログのテキストファイルの名前。

Nu、Du と Ou

撮影した投影値画像の横画素数、画素の横方向の辺長の実寸値とそれを単位とする検出器面上の IC の横座標値。ただし、PB-CT simulator が出力する Ou の値はいわゆる「センター値」である。

Nw、Dw と Ow

撮影した投影値画像の縦画素数、画素の縦方向の辺長の実寸値とそれを単位とする検出器面上の IC の縦座標値。ただし、PB-CT と FB-CT simulators では Du と同じ値であることを仮定している Dw や、画像再構成の処理に不要な Ow の値を出力しない。

注

ぼくが書いた FB-CT および CB-CT simulators は現実にはありえない、サンプル回転軸の位置にある検出器で測定した投影値画像を出力します。そのため、Du と Dw は起動パラメータとして指定した 3 次元画像の画素の辺長 Ixy と Iz のそれぞれと同じ値です。また、そのデータから画像再構成を行う際には「光源と検出器間の距離（source-detector distance；SDD)」として SSD と同じ値を指定してやる必要があります。

(3)

CT simulators が出力した投影値画像とパラメータから画像再構成を行う改造版のプログラムの起動法とそれらが標準エラー出力に書き出す値は以下の通りです。ただし、PB-CT 用プログラムの起動法や処理内容は以前の E-mails

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

で紹介したものと同じです。また、FB-CT と CB-CT 用のものでは「光源と IC を結ぶ線」から測ったサンプル回転軸のオフセット値 ORC をオプション扱いにしてあるので、従来と同じ起動法で実行することも可能です。なお、CB-CT 用のプログラム fdk に対して起動パラメータの layer1、layer2、RA0 と TG_format のすべての指定を省略すると画像再構成の処理をせずに標準エラー出力への値の書き出しだけを行います。これにより出力される「画像再構成が可能なスライスの総数 Nz」に応じた 0≦ layer1 ≦ layer2＜Nz を満たす layer1 と layer2 の値を起動パラメータとして指定して下さい。

PB-CT simulator が出力したデータから CBP 法で画像再構成

```
pb_cbp  XP/  NF  Dr  Or  {layer1  layer2}  RA0  TG_format  > TG.log
→       Nxy      Nz       Dr
```

FB-CT simulator が出力したデータから CBP 法で画像再構成

```
fb_cbp  XP/  NF  SSD  SDD  {ORC}  Du  Ou  ¥
        {layer1  layer2}  RA0  TG_format  > TG.log
→       Nxy      Nz       Dxy
```

CB-CT simulator が出力したデータから modified FDK 法で画像再構成

```
fdk  XP/  NF  SSD  SDD  {ORC}  Du  Ou  Dw  Ow  ¥
          {{layer1 layer2}  RA0  TG_format}  > TG.log
→         Nxy      Nz       Dxy      Dz
```

これらのパラメータと標準エラー出力に書き出される値の意味は以下の通りです。

XP/

投影値画像の float TIFF が入っているディレクトリの名前。

NF

投影値画像の float TIFF のファイル名を列記した name file の名前。ディレクトリ XP/ の下にサンプル回転角に応じたファイル名の float TIFF だけが入っているなら NF としてハイフン "-" を指定すれば良い。

Dr と Or

PB-CT で撮影した投影画像の画素の横幅の実寸の値と「センター値」。Dr は再構成するスライス画像の正方形画素の x および y 方向の辺長になり、標準エラー出力に書き出される。

SSD および SDD

光源とサンプル回転軸の実距離（source-sample distance)、および、光源と投影値の検出器の実距離（source-detector distance)。

ORC

「光源と IC を結ぶ線」から測ったサンプル回転軸の位置のオフセットの実寸の値。指定を省略すると ORC = 0 と見なす。

Du と Ou

FB-CT および CB-CT で撮影した投影画像の画素の横幅の実寸の値と、それを単位とする投影画像上の IC の横座標値。

Dw と Ow

CB-CT で撮影した投影画像の画素の縦幅の実寸値とそれを単位とする投影画像上の IC の縦座標値。

layer1 と layer2

再構成するスライス画像の番号（0 〜）の下限と上限の値。これらの値の指定を省略すると可能なすべてのスライス画像を再構成する。

RA0

サンプル回転角の度単位の初期値。

TG_format

再構成したスライス画像の float TIFF のファイル名のフォーマット。

TG.log

再構成したスライス画像それぞれのスライス番号（0 〜)、それらの上の再構成した CT 値の最小値と最大値の 3 個の値がタブコード区切りで書き込まれるログのテキストファイルの名前。

Nxy と Nz

再構成した正方形のスライス画像の横縦画素数とスライスの総数。

Dxy = Du × SSD / SDD と Dz = Dw × SSD / SDD

再構成した FB-CT もしくは CB-CT 画像上のスライス面内の正方形画素の x および y 方向の辺長とスライスの幅。

(4)

CB-CT 装置で実測した HiPic 形式のデータファイルから画像再構成を行う新旧のプログラムの起動法とそれらが標準エラー出力に書き出す値は以下の通りです。先の fdk とは異なり hp_fdk では起動パラメータ ORC の指定を省略できません。

```
旧  hp_fdk_org  HiPic/  SSD  SDD  Du  Ou  Dw  Ow  ¥
                {{layer1  layer2}  RA0  {BPS}  TG_format}  > TG.log
    →           Nxy     Nz      Dxy     Dz
新  hp_fdk  HiPic/  SSD  SDD  ORC  Du  Ou  Dw  Ow  ¥
                {{layer1  layer2}  RA0  {BPS}  TG_format}  > TG.log
    →           Nxy     Nz      Dxy     Dz
```

これらのパラメータと標準エラー出力に書き出される値の意味は以下の通りです。前述のプログラム fdk のものと概ね同じなので、それとの違いだけを記します。

HiPic/

CB-CT 装置で実測したデータファイル一式（ログファイル output.log と HiPic 形式の暗電流、入射および透過 X 線強度の画像 "*.img"）が入っているディレクトリの名前。

BPS

この値を指定すると再構成したスライス画像を整数画素値の integer TIFF のファイルに格納する。ただし、BPS として正の値を指定すると再構成した CT 値をスライスごとに最小値と最大値で正規化した整数値に変換し、負の値なら 3 次元再構成画像全体の最小値と最大値で CT 値を正規化する。なお、BPS に値 0 を指定した場合やその指定を省略した場合にはスライス画像を float TIFF のファイルに格納する。

(5)

サンプル回転軸が「光源と IC を結ぶ線」の上にない CB-CT の画像再構成を modified FDK 法で行うプログラム fdk と hp_fdk の改造箇所について少しだけ説明しておきます。FDK 法の式としては以下の 2 箇所を書き換えただけです。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=4

convolution の式

A を A + C / B・u に変えただけ

back-projection の式

R を C + R に変えただけ

ただし、

C == プログラム fdk や hp_fdk の起動パラメータ ORC

さて、C or ORC が 0 ではない CB-CT では「斜め側方」からサンプルを透視することになります。また、CB-CT simulator に非 0 のパラメータの値 OSC を指定すると「斜め上方や下方」からサンプルを透視します。そのような CB-CT の画像再構成が可能な領域（imaging area）の計算法は ORC = OSC = 0 の場合とは異なるので、それに関する画像再構成プログラムの改造が必要でした。

http://www-bl20.spring8.or.jp/~sp8ct/tmp/fdk.pdf#page=3

具体的には、この円筒形の imaging area の円断面の径の計算法の改造は比較的容易でしたが、円筒形の軸方向の広がり（円筒の高さ）の計算法がややこしい。その様相をこの E-mail に添付した fdk_sv.pdf に示しまし

たが、長くなるので詳細な説明は省略します。

(6)
改造・作成した CT simulators と画像再構成用プログラムの動作テストを行ないました。その概要は以下の通りです（こちらの計算機 gsjgix と SPring-8 の計算機 vrm で実行した際の秒単位の総処理時間も付記しました)。なお、新版の FB-CT と CB-CT simulators を使ったテスト（後述）で指定した下記のパラメータの値 ORC と OSC を示す図 040711j_layout.pdf をこの E-mail に添付しましたので御覧下さい（この図の縦横比は実際のものと同じにしてありますが、投影値画像を撮影した検出器の位置はウソです)。

test	TG/	ORC	OSC	gsjgix	vrm
p0	int	-	-	185.853991	338.554912
p1	float	-	-	184.302456	296.814511
f0	int	-	-	357.383837	575.756993
f1	float	0	-	356.550424	570.490445
f2	float	-250	-	363.228020	566.324333
f3	float	+250	-	361.377428	564.584241
c0	int	-	-	2027.906158	2603.624537
c1	float	0	0	2027.706307	2612.467212
c2	float	-250	0	2087.107840	2658.142258
c3	float	+250	0	2087.060191	2679.681836
c4	float	0	-180	2392.980325	2972.296295
c5	float	0	+180	2375.836098	2991.676787
c6	float	-250	-180	2425.175506	3034.510317
c7	float	+250	-180	2430.369639	3040.075901
c8	float	-250	+180	2432.920568	3039.831314
c9	float	+250	+180	2412.255314	3038.574547

名前［p,f,c][0-9］の 1 文字目が［p,f,c］ のテストは［P,F,C]B-CT 用のプログラムを対象としたものです。そして、 2 文字目が "0" のテストでは旧版の CT simulator を実行し、新版でもそれらと同じ結果が得られることを "1" のテストで確認しました（新版は旧版の上位互換プログラムです)。なお、名前の 2 文字目が "[2-9]" のテストも新版の CT simulators を使っています。

いずれのテストでも SPring-8 BL20B2 の測定 040711j で得た byte 画像
http://www-bl20.spring8.or.jp/~sp8ct/tmp/040711j.gif
をトリミングした画像（後述）を CT-simulator で（パラメータ ORC や OSC の値を変えて）撮影し、それによって得た
PB-CT では 360 投影 / 180 度回転
FB-CT と CB-CT では 720 投影 / 360 度回転
の投影値画像からプログラム［p,f]b_cbp や fdk で画像再構成を行いました。

測定 040711j の byte 画像のトリミングは画素数を削減するためではなく TIFF の image description として埋め込まれていた 8 ビット画素値と CT 値の対応関係に関する情報を消去するために行いました。そのようにした integer TIFF を float TIFF として新版の CT simulators に渡すと 8 ビットの整数画素値を浮動小数点数の CT 値と見なします（スライス画像ごとに警告を発しますが）。なお、具体的には、テストに先立ち 1000^2×720 画素の byte 画像の画素値 125 以上の画素すべてがちょうどおさまる 775×734×706 画素の直方体領域を切り出し、そのスライス画像をディレクトリ tg/の下に格納しました。

最終的な再構成画像によってもとの byte 画像の 8 ビット画素値を再現するため、CT simulators や画像再構成プログラムの起動パラメータとして実寸の長さではなく画素幅単位の値を指定しました。そのため、投影値画像の画素値は「画素幅」と「もとの byte 画像の 8 ビット画素値」の積を単位とする値になっています。後述する X 線透過率画像の作成時にはこの投影値画像の画素値に

画素の辺長の実寸（0.000583 cm）×

byte 画像の 8 ビット画素値 1 階調ごとの CT 値の増分（0.02930016 1/cm）

を乗じて補正した後に「X 線透過率 ＝ exp(- 投影値)」の計算を行いました。

(7)

上記のテストの結果をまとめた 040711j_test.pdf と 040711j_hg.pdf をこの E-mail に添付しました。040711j_test.pdf の各ページにテスト[p,f,c][0-9]のそれぞれに関する以下の記載や画像を並べました（合計 16 ページ）：

上段：入力したコマンドと（コメントアウトした）標準エラー出力の値

下左：CT simulator で撮影したサンプル回転角が 0 度方向の X 線透過率画像

下中：CT simulator の投影値画像から再構成した中央のスライスの画像

下右：再構成した 3 次元画像の縦（xz）断面の中央のスライスの画像

ただし、X 線透過率画像では値域 0 ～ 1 の透過率を、また、再構成画像ではもとの byte 画像の画素値 0 ～255 の範囲の値を同じグレースケールで表しています。

040711j_hg.pdf にはもとの 3 次元 byte 画像の画素値ヒストグラム（緑線）と再構成した 3 次元画像の画素値ヒストグラム（赤もしくは青線）を示しました。比較のため、byte 画像のヒストグラムをすべてのグラフに転写しました。また、紙面の関係上、テスト p[0,1] などではヒストグラムをひとつのグラフにまとめてプロットしました。なお、グラフ中央の黒の縦線の位置はもとの byte 画像の画素値 125 で、測定 040711j ではこれをしきい値として物体像を識別したので、その位置のヒストグラムの高さが大きく違っていると大問題です。

今回行ったテストの結果から色々なことがわかります。何よりも重要なのはこの E-mail の最初に記した結論です：

サンプル回転軸が「光源と照明中心を結ぶ線」の上にない場合も
http://www-bl20.spring8.or.jp/~sp8ct/tmp/fbct.pdf#page=4
に記した式で FB-CT や CB-CT の画像再構成を行うことができる。

ORC や OSC の値が非 0 の CB-CT で撮影したデータから modified FDK 法で再構成した画像には「幾何学的な歪み」は生じないようです。しかしながら、近似的な画像再構成法である FDK 法では「鉛直方向の CT 値の変動」は不可避です。

(8)
このようなテストに用いたファイルはすべて SPring-8 の計算機 vrm のディレクトリ/media/ssd/040711j/の下に置いてあります。ただし、総量が 113 GB 程度もあるので、そのうちに消去する予定です。

(9)
CB-CT 装置で実測したデータ用の画像再構成プログラム hp_fdk_org と hp_fdk の動作テストも行いました。ただし、適当な実測データがないので、サンプル回転軸が「光源と IC を結ぶ線」の上にある CB-CT 装置を用いた測定 150303d のものを使いました（hp_fdk の起動パラメータ ORC として値 0 を指定すれば OK)。こちらの計算機 gsjgix で行った処理の記録を以下に貼り付けました。ただし、最初に実行した hp_fdk_old は hp_fdk_org（hp_fdk.c.org）の改造前のコードからコンパイルした、以前の E-mails
http://www-bl20.spring8.or.jp/~sp8ct/tmp/rhp.pdf#page=3
で紹介した従来の CB-CT 用画像再構成プログラムです。

```
# execution test of hp_fdk_old, hp_fdk_org and hp_fdk

    ln  -s  /work2/tsukasa/xct/150303/150303d/raw  150303d_raw
    setenv  THREADS  8

    mkdir  150303d_old
    stop_watch  hp_fdk_old  ¥
                150303d_raw  ¥
                534.5  534.5  ¥
                6.5e-3  1024  6.5e-3  1024  0  ¥
                150303d_old/%04d.tif  >  150303d_old.log
#   2045    2023    6.500000e-03    6.500000e-03
#   2555.964645

    mkdir  150303d_org
    stop_watch  hp_fdk_org  ¥
                150303d_raw  ¥
                534.5  534.5  ¥
                6.5e-3  1024  6.5e-3  1024  0  ¥
                150303d_org/%04d.tif  >  150303d_org.log
#   2045    2023    6.500000e-03    6.500000e-03
#   2526.298687
```

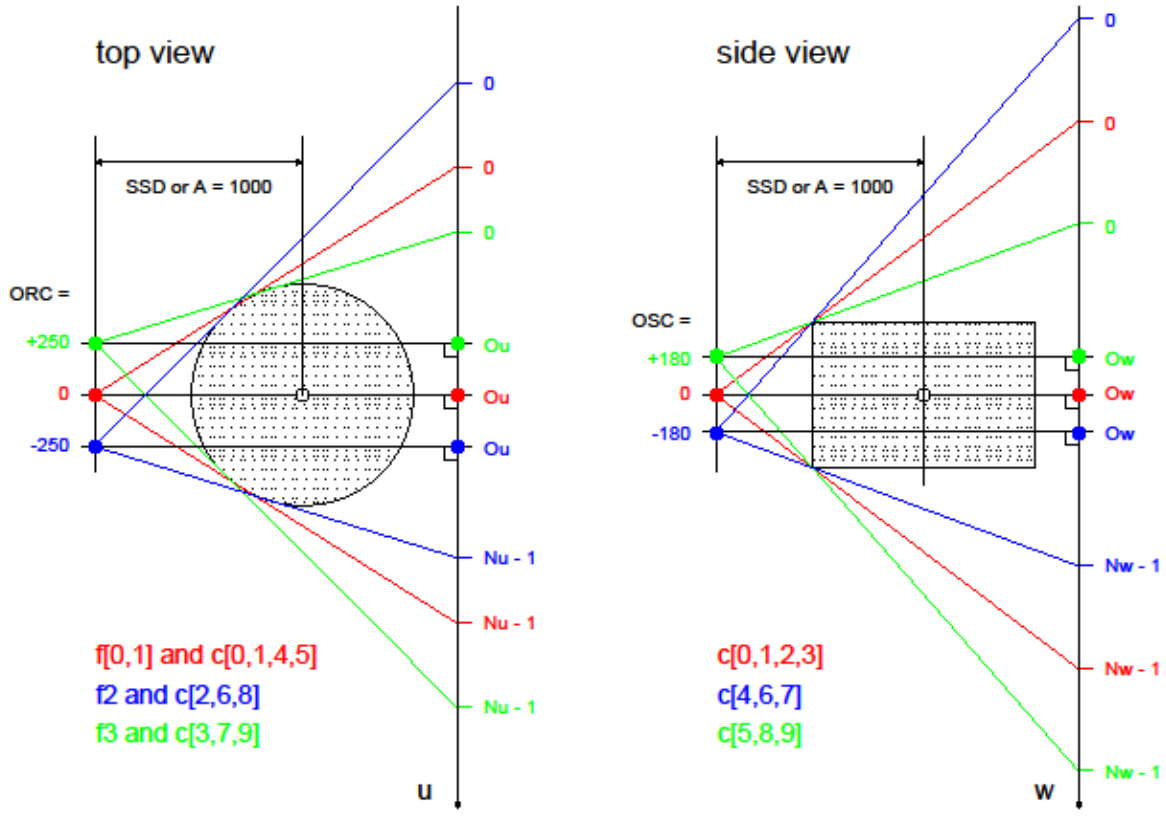
```
    mkdir   150303d
    stop_watch   hp_fdk   ¥
                 150303d_raw   ¥
                 534.5   534.5   ¥
                 0   ¥                              ← ORC = 0
                 6.5e-3   1024   6.5e-3   1024   0   ¥
                 150303d/%04d.tif   >   150303d.log
#   2045     2023     6.500000e-03     6.500000e-03
#   2542.649171

    cksum   150303d_old.log   150303d_org.log   150303d.log
#   2137442633 47442 150303d_old.log
#   2137442633 47442 150303d_org.log
#   2137442633 47442 150303d.log
    ← すべて同じ CRC checksum なので、３個の再構成画像は同一（多分）
```

非常に長い E-mail になりました。とりあえず以上です。

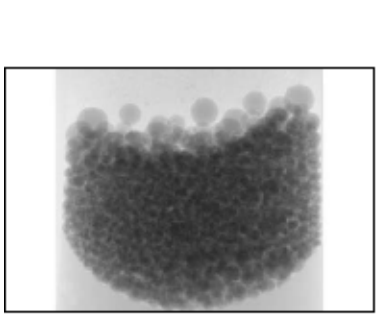
添付ファイル fdk_sv.pdf：省略

添付ファイル 040711j_layout.pdf

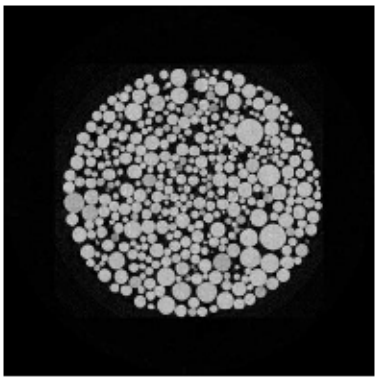


添付ファイル 040711j_test.pdf（page 1 of 16）

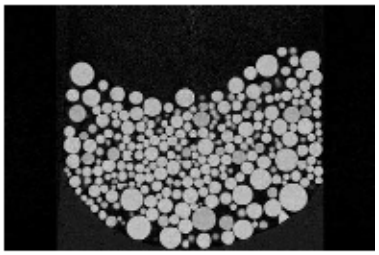
```
p0 :
    setenv  THREADS  8
    mkdir  p0_xp  p0_tg
    pb_3d  tg/  -  1  360  p0_xp/%03d.tif  >/dev/null
    #      1068   706    360    1      533.5
    pb_cbp  p0_xp/  -  1  533.5  0  p0_tg/%03d.tif  >/dev/null
    #      1067   706    1
```



p0 : xp2xt / 000



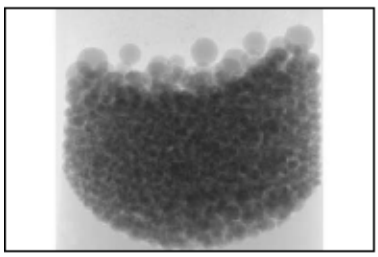
p0 : tg_xy / 353



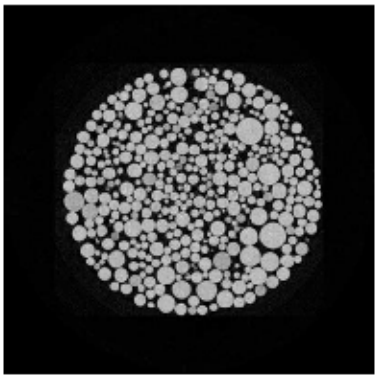
p0 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 2 of 16）

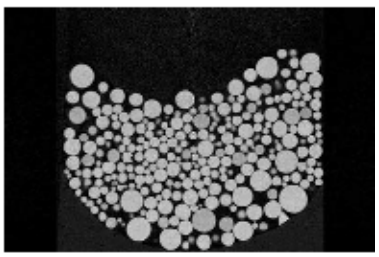
```
p1 :
    setenv  THREADS  8
    mkdir  p1_xp  p1_tg
    pb_float  tg/  -  1  360  +0  p1_xp/%03d.tif  >/dev/null
    #      1068   706    360
    #      1      533.5  +0
    pb_cbp  p1_xp/  -  1  533.5  +0  p1_tg/%03d.tif  >/dev/null
    #      1067   706    1
```



p1 : xp2xt / 000



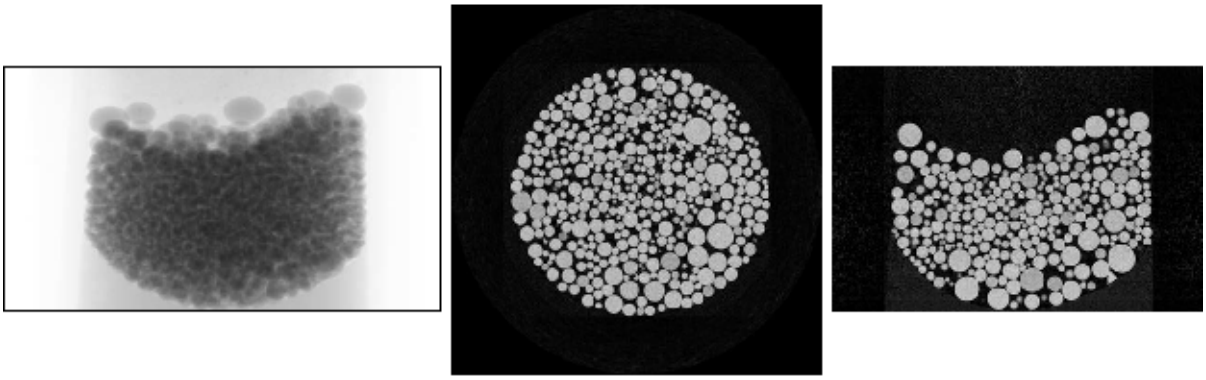
p1 : tg_xy / 353



p1 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 3 of 16）

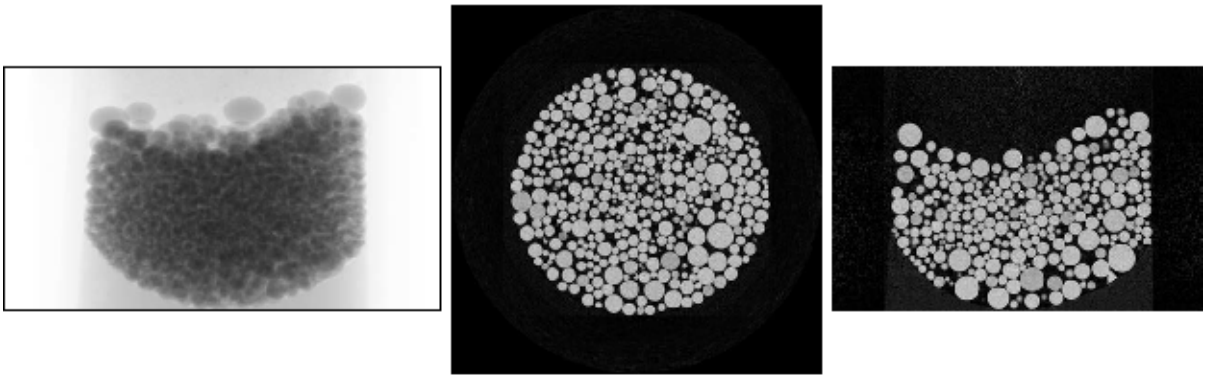
```
f0 :
    setenv THREADS 8
    mkdir f0_xp f0_tg
    fb_3d tg/ - 1 1000 720 f0_xp/%03d.tif >/dev/null
    #     1263   706    720    1000   1      631
    fb_cbp f0_xp/ - 1000 1000 1 631 0 f0_tg/%03d.tif >/dev/null
    #     1067   706    1.000000e+00
```



f0 : xp2xt / 000　　f0 : tg_xy / 353　　f0 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 4 of 16）

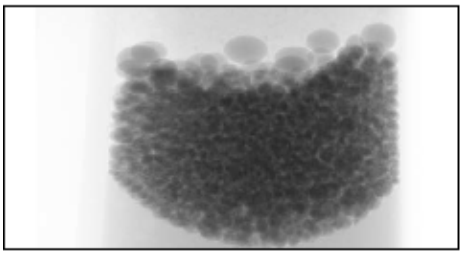
```
f1 :
    setenv THREADS 8
    mkdir f1_xp f1_tg
    fb_float tg/ - 1 1000 +0 720 +0 f1_xp/%03d.tif >/dev/null
    #     1263   706    720
    #     1000   +0     1      631.109150     +0
    fb_cbp f1_xp/ - 1000 1000 +0 1 631.109150 +0 f1_tg/%03d.tif >/dev/null
    #     1067   706    1.000000e+00
```



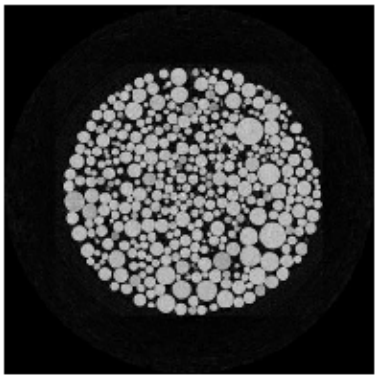
f1 : xp2xt / 000　　f1 : tg_xy / 353　　f1 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 5 of 16）

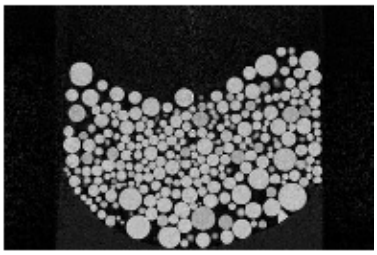
```
f2 :
    setenv THREADS 8
    mkdir f2_xp f2_tg
    fb_float tg/ - 1 1000 -250 720 +0 f2_xp/%03d.tif >/dev/null
    #     1317   706    720
    #     1000   -250   1      1007.683762    +0
    fb_cbp f2_xp/ - 1000 1000 -250 1 1007.683762 +0 f2_tg/%03d.tif >/dev/null
    #     1067   706    1.000000e+00
```



f2 : xp2xt / 000



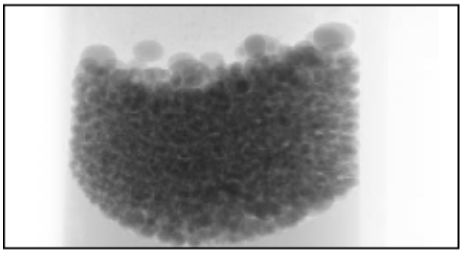
f2 : tg_xy / 353



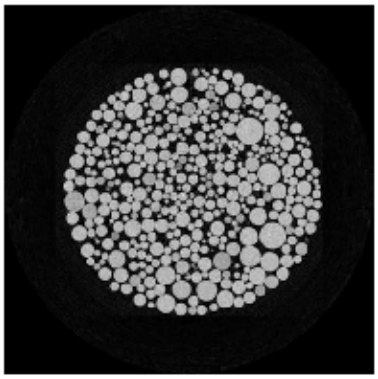
f2 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 6 of 16）

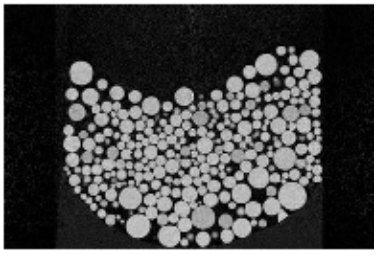
```
f3 :
    setenv THREADS 8
    mkdir f3_xp f3_tg
    fb_float tg/ - 1 1000 250 720 +0 f3_xp/%03d.tif >/dev/null
    #     1317   706    720
    #     1000   250    1      308.534382     +0
    fb_cbp f3_xp/ - 1000 1000 250 1 308.534382 +0 f3_tg/%03d.tif >/dev/null
    #     1067   706    1.000000e+00
```



f3 : xp2xt / 000



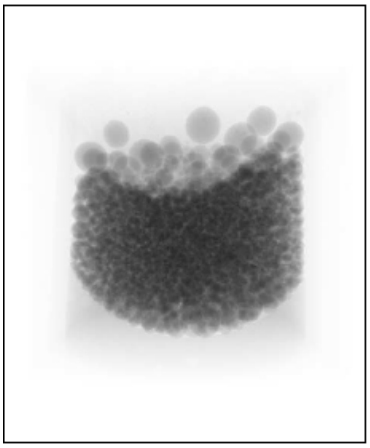
f3 : tg_xy / 353



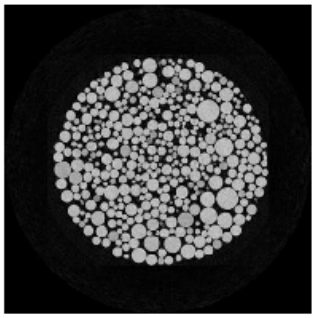
f3 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 7 of 16）

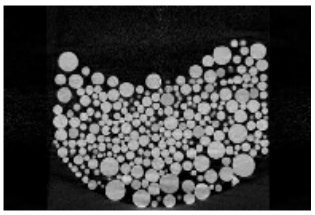
```
c0 :
    setenv THREADS 8
    mkdir c0_xp c0_tg
    cb_ss tg/ - 1 1 1000 720 c0_xp/%03d.tif >/dev/null
    #     1263   1515   720    1000   1      631    1      757
    fdk c0_xp/ - 1000 1000 1 631 1 757 0 c0_tg/%03d.tif >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c0 : xp2xt / 000



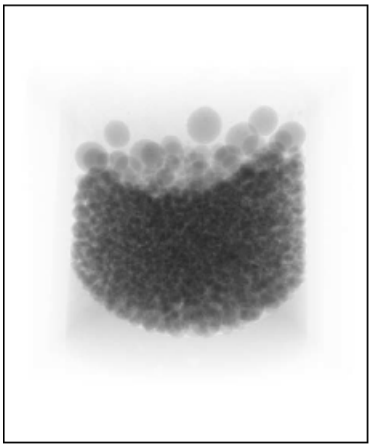
c0 : tg_xy / 354



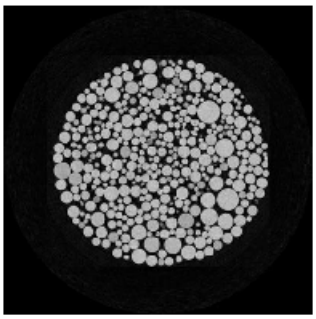
c0 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 8 of 16）

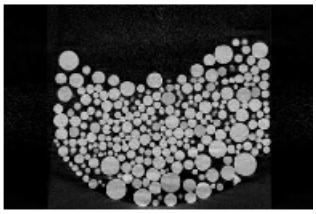
```
c1 :
    setenv THREADS 8
    mkdir c1_xp c1_tg
    cb_float tg/ - 1 1 1000 +0 +0 720 +0 c1_xp/%03d.tif >/dev/null
    #     1263   1515   720
    #     1000   +0     1      631.109150     1      757.037916     +0
    fdk c1_xp/ - 1000 1000 +0 1 631.109150 1 757.037916 +0 c1_tg/%03d.tif >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c1 : xp2xt / 000



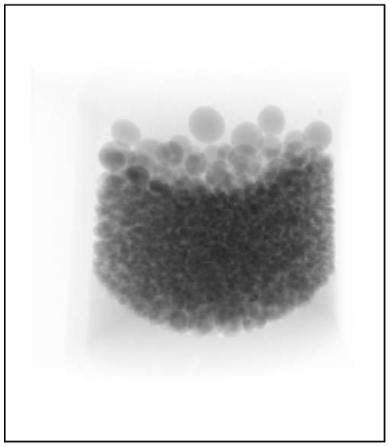
c1 : tg_xy / 354



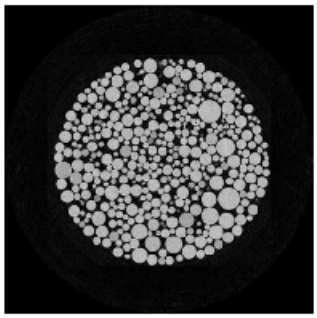
c1 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 9 of 16）

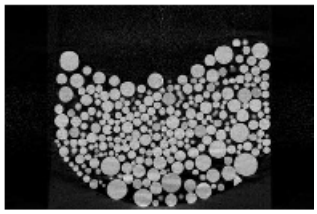
```
c2 :
    setenv  THREADS  8
    mkdir  c2_xp  c2_tg
    cb_float  tg/  -  1  1  1000  -250  +0  720  +0  c2_xp/%03d.tif  >/dev/null
    #      1317    1515    720
    #      1000    -250    1       1007.683762     1       757.037916      +0
    fdk  c2_xp/  -  1000  1000  -250  1  1007.683762  1  757.037916  +0  c2_tg/%03d.tif  >/dev/null
    #      1067    708     1.000000e+00    1.000000e+00
```



c2 : xp2xt / 000



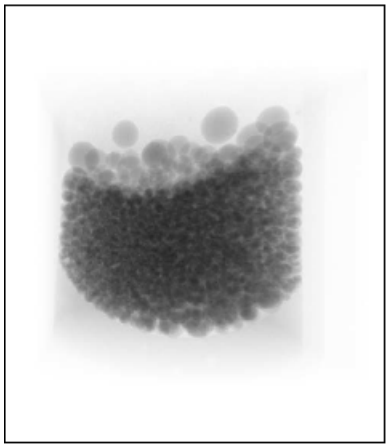
c2 : tg_xy / 354



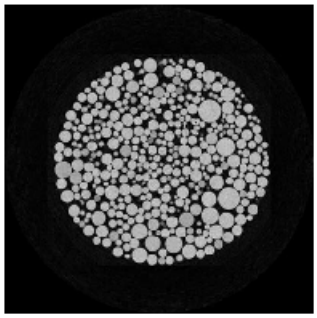
c2 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 10 of 16）

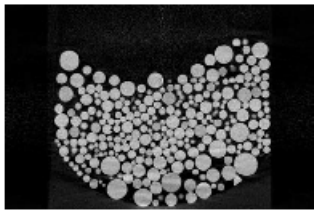
```
c3 :
    setenv  THREADS  8
    mkdir  c3_xp  c3_tg
    cb_float  tg/  -  1  1  1000  250  +0  720  +0  c3_xp/%03d.tif  >/dev/null
    #      1317    1515    720
    #      1000    250     1       308.534382      1       757.037916      +0
    fdk  c3_xp/  -  1000  1000  250  1  308.534382  1  757.037916  +0  c3_tg/%03d.tif  >/dev/null
    #      1067    708     1.000000e+00    1.000000e+00
```



c3 : xp2xt / 000



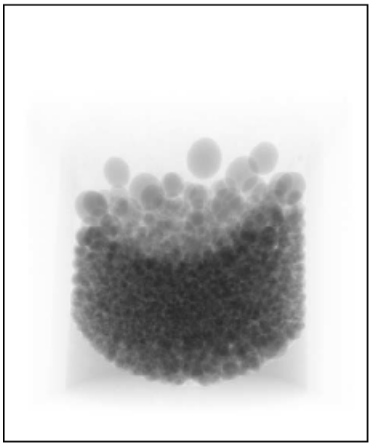
c3 : tg_xy / 354



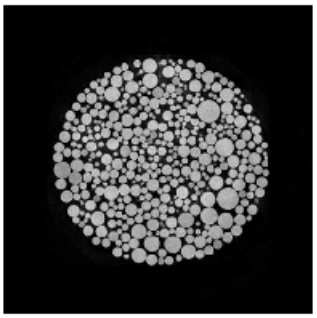
c3 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 11 of 16）

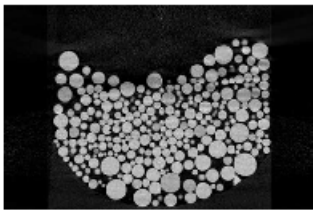
```
c4 :
    setenv THREADS 8
    mkdir c4_xp c4_tg
    cb_float tg/ - 1 1 1000 +0 -180 720 +0 c4_xp/%03d.tif >/dev/null
    #     1263   1515   720
    #     1000   +0     1      631.109150     1      1143.062916     +0
    fdk c4_xp/ - 1000 1000 +0 1 631.109150 1 1143.062916 +0 c4_tg/%03d.tif >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c4 : xp2xt / 000



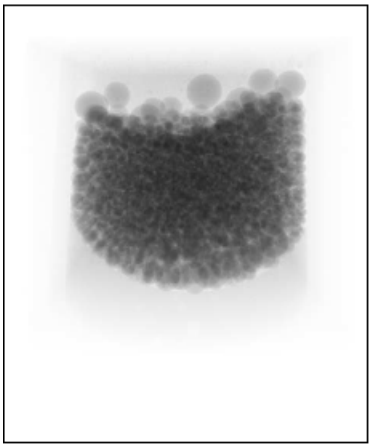
c4 : tg_xy / 354



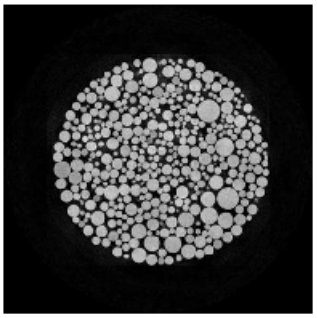
c4 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 12 of 16）

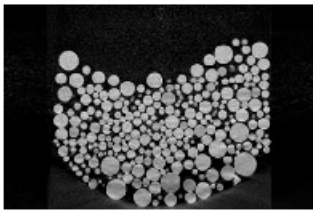
```
c5 :
    setenv THREADS 8
    mkdir c5_xp c5_tg
    cb_float tg/ - 1 1 1000 +0 180 720 +0 c5_xp/%03d.tif >/dev/null
    #     1263   1515   720
    #     1000   +0     1      631.109150     1      371.012916     +0
    fdk c5_xp/ - 1000 1000 +0 1 631.109150 1 371.012916 +0 c5_tg/%03d.tif >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c5 : xp2xt / 000



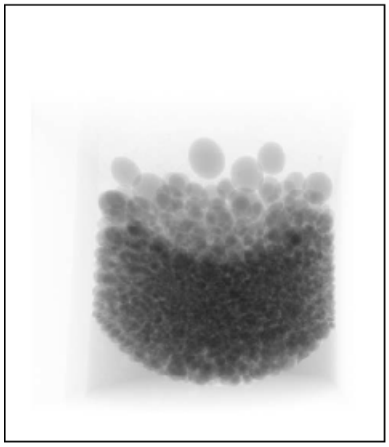
c5 : tg_xy / 354



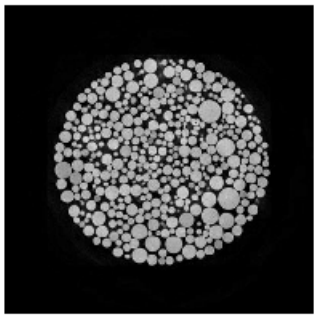
c5 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 13 of 16）

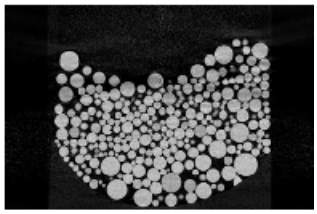
```
c6 :
    setenv  THREADS  8
    mkdir  c6_xp  c6_tg
    cb_float  tg/  -  1  1  1000  -250  -180  720  +0  c6_xp/%03d.tif  >/dev/null
    #     1317   1515   720
    #     1000   -250   1     1007.683762    1     1143.062916    +0
    fdk  c6_xp/  -  1000  1000  -250  1  1007.683762  1  1143.062916  +0  c6_tg/%03d.tif  >/dev/null
    #     1067   708    1.000000e+00    1.000000e+00
```



c6 : xp2xt / 000



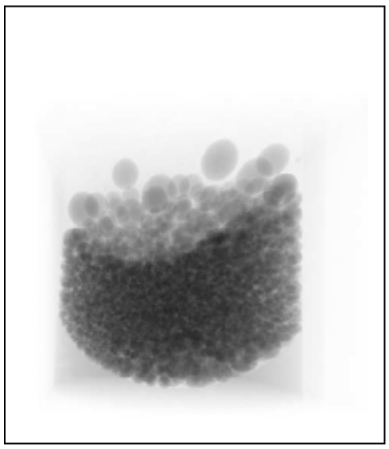
c6 : tg_xy / 354



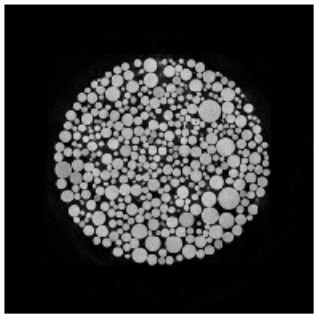
c6 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 14 of 16）

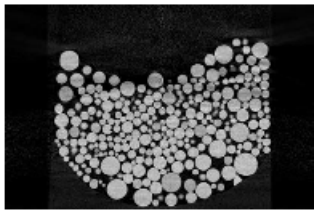
```
c7 :
    setenv  THREADS  8
    mkdir  c7_xp  c7_tg
    cb_float  tg/  -  1  1  1000  250  -180  720  +0  c7_xp/%03d.tif  >/dev/null
    #     1317   1515   720
    #     1000   250    1     308.534382     1     1143.062916    +0
    fdk  c7_xp/  -  1000  1000  250  1  308.534382  1  1143.062916  +0  c7_tg/%03d.tif  >/dev/null
    #     1067   708    1.000000e+00    1.000000e+00
```



c7 : xp2xt / 000



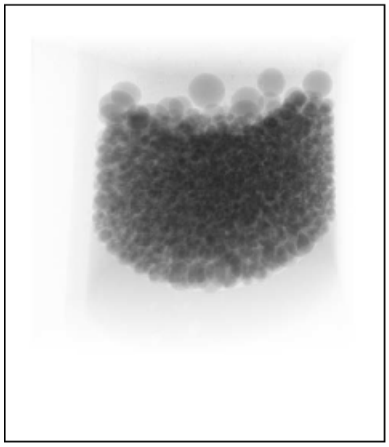
c7 : tg_xy / 354



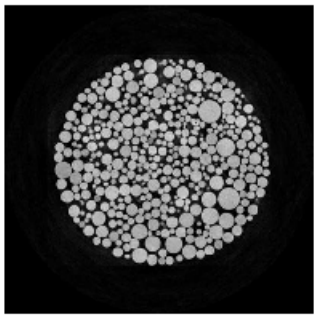
c7 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 15 of 16）

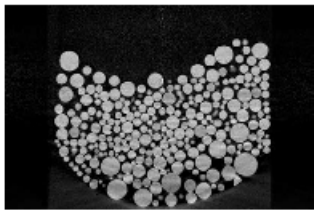
```
c8 :
    setenv  THREADS  8
    mkdir  c8_xp  c8_tg
    cb_float  tg/  -  1  1  1000  -250  180  720  +0  c8_xp/%03d.tif  >/dev/null
    #     1317   1515   720
    #     1000   -250   1     1007.683762    1     371.012916     +0
    fdk  c8_xp/  -  1000  1000  -250  1  1007.683762  1  371.012916  +0  c8_tg/%03d.tif  >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c8 : xp2xt / 000



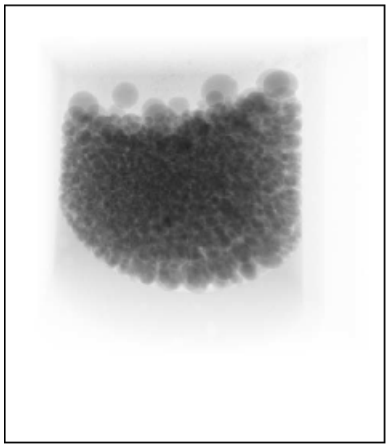
c8 : tg_xy / 354



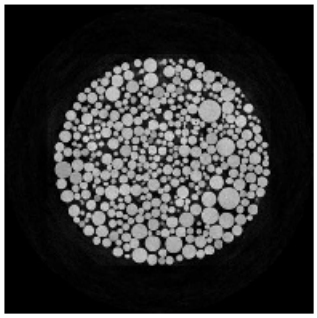
c8 : tg_xz / 0533

添付ファイル 040711j_test.pdf（page 16 of 16）

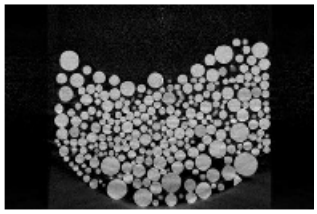
```
c9 :
    setenv  THREADS  8
    mkdir  c9_xp  c9_tg
    cb_float  tg/  -  1  1  1000  250  180  720  +0  c9_xp/%03d.tif  >/dev/null
    #     1317   1515   720
    #     1000   250    1     308.534382     1     371.012916     +0
    fdk  c9_xp/  -  1000  1000  250  1  308.534382  1  371.012916  +0  c9_tg/%03d.tif  >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c9 : xp2xt / 000

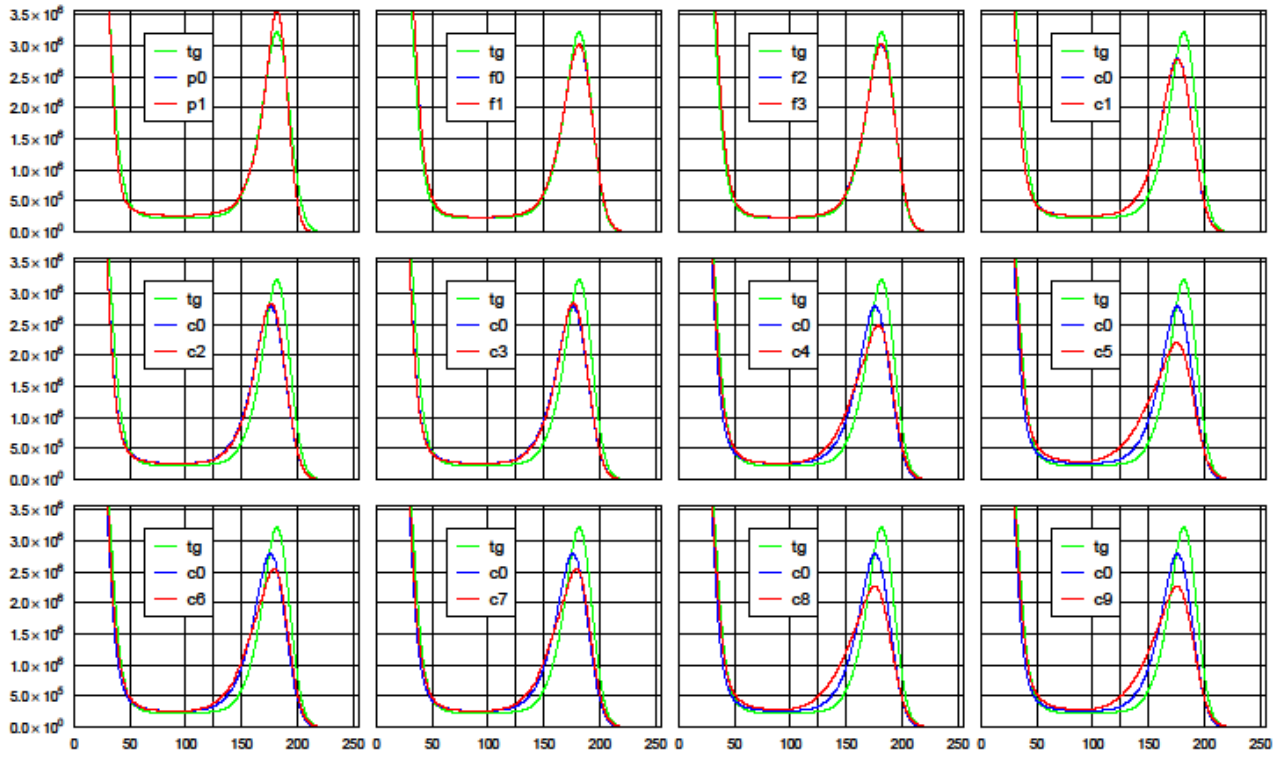


c9 : tg_xy / 354



c9 : tg_xz / 0533

添付ファイル 040711j_hg.pdf



Date: Thu, 26 Apr 2018 19:29:33 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: MATSUNO Junya, Masayuki Uesugi, "TSUCHIYAMA, Akira", Miyake
Subject: CB_sphere

うえすぎさま、

GSJ/AIST のなかのです。球形のサンプルを使って CT をせずに CB（cone beam）の IC（照明中心 = 点と仮定した光源から検出器面に下ろした垂線の足）の位置を決める方法を思いつきました。実はこの手法は 4/18 の E-mail で紹介した CB-CT simulator で撮影？した「ビーズ球」の X 線透過率の画像を眺めていて気づきました。4/18 の E-mail に添付した 040711j_test.pdf のテスト「c7」のページだけを抜き出した c7.pdf をこの E-mail に添付します。その X 線透過率の画像の上に新たに IC の位置を示す赤線と青線を引きました（それらの交点が IC の位置です）。この画像の上の IC から離れた位置にあるビーズ球は CB で撮影したために著しく歪んだ像（楕円）になっており、それらの長軸はどれも IC の方を向いています。

このことを数式で確かめました。この E-mail に添付した cb_sphere.pdf に記した通り CB で撮影した球の投影像は傾いた楕円となり、以下が成り立ちます：

投影像の楕円の長軸の傾きは球の中心の位置で決まる。
その長軸を延長した線は常に IC を通る。

念のために cb_float（新しい CB-CT simulator プログラム）で球の像を撮影し、その投影値を cb_sphere.pdf に記した式を使って計算した値と比較しました。この E-mail に添付した cb_s+e.pdf がその結果です。球の中心の位置（正確には、cb_float の起動パラメータの ORC と OSC の値）として 3 通りの値を指定して以下の 3 種類の画像を作成しました（その処理には新たに書いたプログラム cb_sphere を使いましたが、その説明は省略します)。

simulated：cb_float で撮影した球の投影画像

exact：cb_sphere.pdf の式で計算した球の投影値の画像

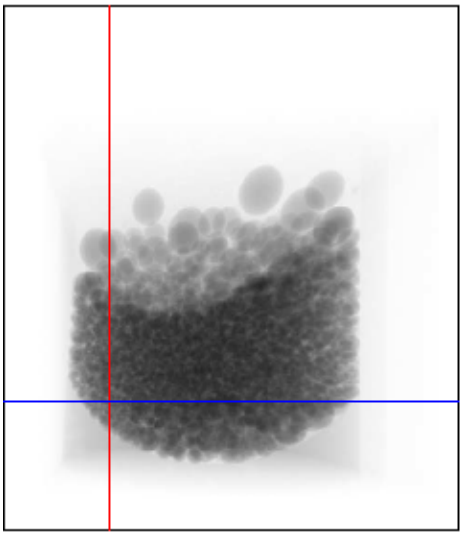
simulated - exact：上記の 2 画像の差分画像

それぞれの画像上の赤線と青線の交点が IC の位置です。また、緑色の線は球の中心の位置から計算した楕円の長軸方向を示しています。simulated と exact の投影値は概ね一致しており、球の投影像は予想通りの楕円になっています。

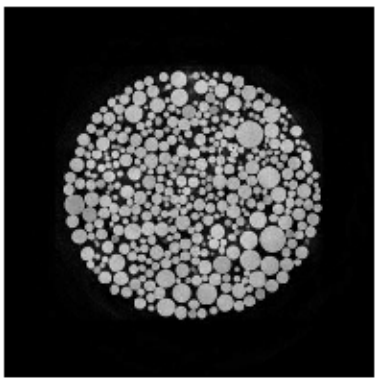
と言うことで、球を撮影した X 線透過率（もしくは投影値）画像を使えば CT をせずに CB-CT 装置の IC の位置を推定できそうです。ただし、光源と IC を結ぶ線に近い位置に球の中心がある場合には円に近い投影像になるので、超精密な位置決めはできないかもしれません。また、画像にノイズが多い場合も同様だと思われます。とり急ぎ、

添付ファイル c7.pdf

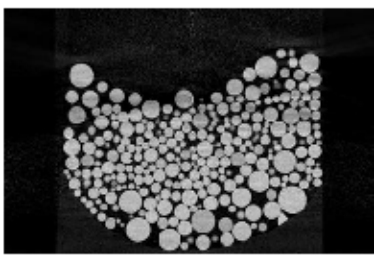
```
c7 :
    setenv THREADS 8
    mkdir c7_xp c7_tg
    cb_float tg/ - 1 1 1000 250 -180 720 +0 c7_xp/%03d.tif >/dev/null
    #     1317   1515   720
    #     1000   250    1     308.534382    1     1143.062916    +0
    fdk c7_xp/ - 1000 1000 250 1 308.534382 1 1143.062916 +0 c7_tg/%03d.tif >/dev/null
    #     1067   708    1.000000e+00   1.000000e+00
```



c7 : xp2xt / 000



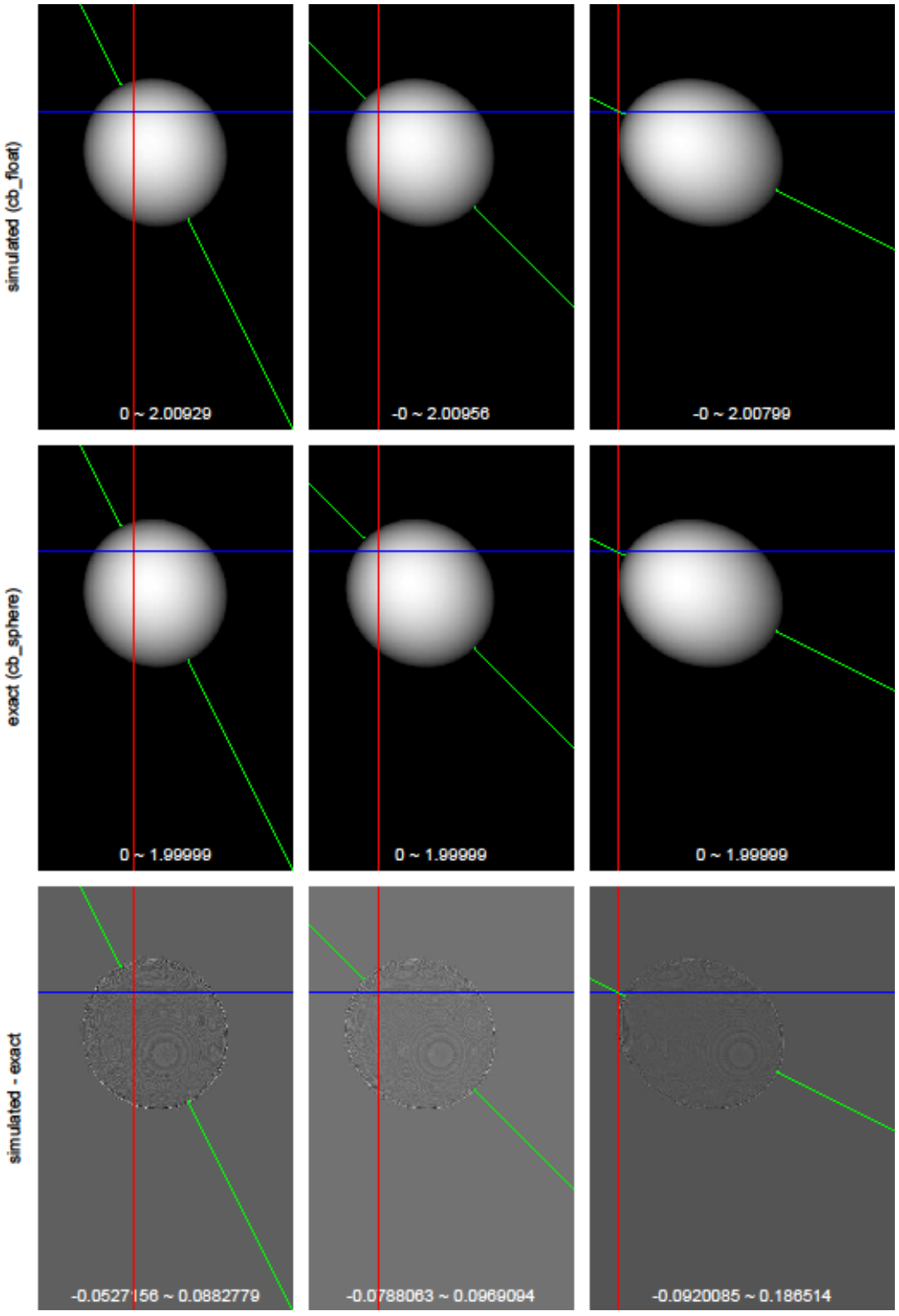
c7 : tg_xy / 354



c7 : tg_xz / 0533

添付ファイル cb_sphere.pdf：5/31 の E-mail に添付した改訂版を御覧下さい。

添付ファイル cb_s+e.pdf



Date: Fri, 25 May 2018 12:29:47 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: MATSUNO Junya, Miyake, "TSUCHIYAMA, Akira"
Subject: RE: Fwd: Re: 質問

うえすぎさま、

GSJ/AIST のなかのです。東北大学の scanXmate のパラメータについて岩下さんに質問してくれてありがとうございます。ただ、岩下さんは scanXmate（cone beam CT）の画像再構成に詳しくないようです。つまり、cone beam CT では照明中心と回転軸の位置関係が肝ですが、それについての言及がありません。また、パラメータ StageAxis[X,Dx] と StageAxis[Y,Dy] の説明を読む限り scanXmate では回転軸や検出器面が光路に直交していることになりますが、それを仮定してぼくが再構成した画像は全然ダメでした（そちらで開発中の装置で上杉君が経験したものと同様に、回転軸から離れた位置にある像を正常に再構成できなかった)。と言う訳で、scanXmate の詳細な幾何学的情報は企業秘密？ とり急ぎ、

---------- Original Message ----------
差出人: Kentaro UESUGI
送信日時: 2018 年 5 月 25 日 10:12
宛先: 中野司
件名: Fwd: Re: 質問

ようやく昨日メールを出しました。すぐ返事がいただけました。うえすぎ

iwashita wrote:
>
> Date: Fri, 25 May 2018 09:54:31 +0900
> Subject: Re: 質問
> From: iwashita
> To: Kentaro UESUGI
>
> 上杉さま
>
> 有限会社ホワイトラビットの岩下です。お世話になっております
>
> 装置の軸方向は
> X 線管から検出器へ向かう方向 X 軸
> 上下方向 Z 軸
> X,Y に直交する軸 Y 軸
> と設定されています

> StageAxisX、StageAxisY、StageAxisZ
>　　ステージ回転中心位置（X, Y, Z）です
>
> 回転中心に試料の中心（関心領域の中心）を持っていくため、回転するモーターの上に、
> モータが二個載っています。これが、dx,dy 軸（回転角 0 度の時に、dx 軸と X 軸の方向が
> 一致している。dy も同様）です
>
> StageAxisDX、StageAxisDY
>　　dx, dy の値
> StageRange
>　　X 線焦点から検出器までの長さ
> Xoffset
>　　X 線焦点から X 軸原点（X 線照射窓付近に設定されている）までの長さ
> StageMagnification
>　　StageMagnification = StageRange / (StageAxisX + Xoffset)
>　　注：StageAxisX は、「X 軸原点から回転中心までの長さ」です
> SaveMatrixW
>　　検出器の横方向ピクセル数
> SaveMatrixH
>　　検出器の縦方向ピクセル数
> PixelSize
>　　検出器一素子の大きさ
>
> 単位はすべて、mm です
>
> .sxm ファイルは、撮影条件の保存・読み込みでも使用しますので
> 再構成には必要ない情報も多数含んでいます
>
> 以上です。Molcer には秘密の工夫がいっぱいです、特に高速化で。よろしくお願いします
>
> On 2018/05/24, at 22:17, Kentaro UESUGI wrote:
>
> > 岩下さま
> >
> > 上杉です
> >
> > ちょっと質問があります。もるさーの事じゃないんで申し訳ありませんが・・・
> > (モルサーは時々使っていますが、難しいことはまだしていません。
> > きれいにレンダリング像が出てくるので感心しています。ヒミツ の工夫がたくさんありそう)
> >

> > 本題ですが、東北大のコムスキャンのＣＴ装置で撮ったデータなのですが、
> > 色々見ていてパラメーターが気になりました。
> > 下記のパラメーターの意味をご存じでしたら教えて頂けませんでしょうか？
> >
> > StageAxisX
> > StageAxisY
> > StageAxisZ
> > StageAxisDX
> > StageAxisDY
> > StageRange
> > StageMagnification
> > SaveMatrixW
> > SaveMatrixH
> > Xoffset
> > PixelSize
> ----------

Date: Thu, 31 May 2018 14:49:01 +0900
From: Tsukasa NAKANO
To: Kentaro Uesugi
Cc: MATSUNO Junya, Masayuki Uesugi, "TSUCHIYAMA, Akira", Miyake
Subject: Fw: CB_sphere

うえすぎさま、

GSJ/AIST のなかのです。4/26 の E-mail で紹介した
　cone beam で撮影した均質な球の像の投影値（ファイル cb_sphere.pdf）
に関する話の続報です。

(1) 先日の SIXM 実験の場で A4 の出力紙だけを差し上げた cb_sphere.pdf の改訂版をこの E-mail に添付します。

(2) 先日の SIXM 実験の場で（その時点で入手できていた）要旨だけを紹介した cb_sphere.pdf のと同様な議論をしている論文の PDF「Clackdoyle_2011_Phys._Med._Biol._56_003.pdf」を入手できたので、この E-mail に添付します。その Figure 2 と式（3）より、この論文は cb_sphere.pdf の記号 L0, U と V の値に対してまったく同じ計算式を導出してます。しかしながら、（難しそうなのでぼくとしては挑戦を躊躇している）球の投影像の重心の解析的な計算式の導出は行っていないようです。

とり急ぎ、

添付ファイル cb_sphere.pdf

2018 / 5 / 9

cone beam で撮影した均質な球の像の投影値

照明中心（IC）が検出器上の座標値 (X, Y) の原点だとする。

球の表面： $(x-x_c)^2+(y-y_c)^2+(z-A)^2=R^2$

X 線光路： $\begin{pmatrix} x \\ y \end{pmatrix}=\frac{z}{B}\cdot\begin{pmatrix} X \\ Y \end{pmatrix}$

球の表面と X 線光路の交点の z 座標値

$$\left.\begin{aligned} a&=(X/B)^2+(Y/B)^2+1 \\ b&=x_c\cdot X/B+y_c\cdot Y/B+A \\ c&=x_c^2+y_c^2+A^2-R^2 \end{aligned}\right\}\ \text{として、}\quad \begin{aligned} &a\cdot z^2-2\cdot b\cdot z+c=0 \\ &\rightarrow (z_1-z_2)^2=4\cdot(b^2-a\cdot c)/a^2 \end{aligned}$$

球の像の投影値、p

$$p^2\propto(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2=a\cdot(z_1-z_2)^2=4\cdot(b^2-a\cdot c)/a$$

以下の座標回転により $p=0$ に対応する球の像の外形が楕円であることがわかる。

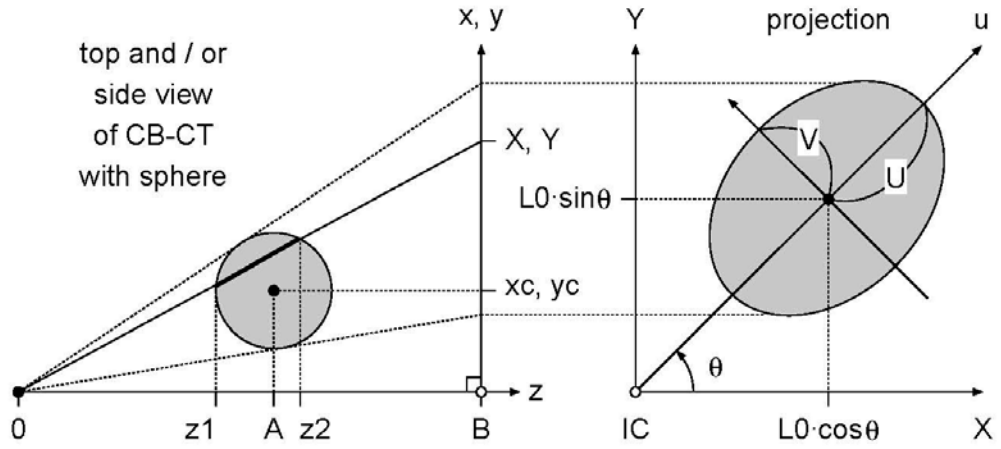
$$\left.\begin{aligned} l_c^2&=x_c^2+y_c^2 \\ \begin{pmatrix} x_c \\ y_c \end{pmatrix}&=l_c\cdot\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \end{aligned}\right\}\ \text{として、}\quad \begin{pmatrix} u \\ v \end{pmatrix}=\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix}=\frac{1}{l_c}\cdot\begin{pmatrix} x_c & y_c \\ -y_c & x_c \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix}$$

$$\rightarrow 0=b^2-a\cdot c=\frac{(x_c\cdot X+y_c\cdot Y+A\cdot B)^2-(X^2+Y^2+B^2)\cdot c}{B^2}=\frac{R^2\cdot c}{c-l_c^2}\cdot\left\{1-\left(\frac{u-L_0}{U}\right)^2-\left(\frac{v}{V}\right)^2\right\}$$

$$\text{ただし、}\quad \begin{pmatrix} L_0 \\ U \\ V \end{pmatrix}=\frac{B}{c-l_c^2}\cdot\begin{pmatrix} A\cdot l_c \\ R\cdot\sqrt{c} \\ R\cdot\sqrt{c-l_c^2} \end{pmatrix}=\frac{B}{A^2-R^2}\cdot\begin{pmatrix} A\cdot\sqrt{x_c^2+y_c^2} \\ R\cdot\sqrt{x_c^2+y_c^2+A^2-R^2} \\ R\cdot\sqrt{A^2-R^2} \end{pmatrix}$$

p を座標値 (u, v) の関数として表すと、

$$\left(\frac{p}{2\cdot R}\right)^2\propto\frac{c}{c-l_c^2}\cdot\left\{1-\left(\frac{u-L_0}{U}\right)^2-\left(\frac{v}{V}\right)^2\right\}\cdot\frac{B^2}{u^2+v^2+B^2}=\left(\frac{U}{V}\right)^2\cdot\left\{1-\left(\frac{u-L_0}{U}\right)^2-\left(\frac{v}{V}\right)^2\right\}\cdot\frac{B^2}{u^2+v^2+B^2}$$



添付ファイル Clackdoyle_2011_Phys._Med._Biol._56_003.pdf

http://iopscience.iop.org/article/10.1088/0031-9155/56/23/003/pdf

Date: Fri, 05 Oct 2018 14:33:12 +0900
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: jmatsuno, uesugi, atsuchi, miya
Subject: MFX-CT

うえすぎさま、

GSJ/AIST のなかのです。7/19 から「クモ膜下出血」で入院していましたが、9/20 に退院して自宅に戻り、10/1 より GSJ/AIST での研究生活を再開しました。体調は悪くないです。10/24 からの SPring-8 実験には参加するつもりでいます。

8/5 の E-mail に記されていた MFX-CT の再構成画像の問題を考えてみました。まず、SPring-8 の計算機 vrm のディレクトリ

/media/disk/bl20b2/2018.07.25_alignment/[600,1800]prj_2x2/001/rh/

の下に置いてあった再構成画像を眺めると、サンプル像の輪郭線が二重になっていました。これは通常の PB（parallel beam）CT では発生しない CB（cone beam）CT 特有の現象で、4/18 の E-mail で紹介した最新版の CB CT 用の画像再構成プログラム hp_fdk（および fdk）に適切な非ゼロの起動パラメータ ORC の値を指定してやれば解消するかもしれません：

CB CT の画像再構成で使う起動パラメータ ORC

この E-mail に添付した 040711j_layout.pdf の左側の top view の図に示したようなサンプル回転軸と「光源と照明中心を結ぶ線」の実距離

最新の hp_fdk が入っている書庫ファイルとその起動法

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

```
hp_fdk  HiPic/  SSD  SDD  ORC  Du  Ou  Dw  Ow  ¥
        {{layer1  layer2}  RA0  {BPS}  TG_format}  > TG.log
```

おまけ：最新の fdk が入っている書庫ファイルとその起動法

http://www-bl20.spring8.or.jp/~sp8ct/tmp/radon.taz
http://www-bl20.spring8.or.jp/~sp8ct/tmp/raysum.taz

```
fdk  XP/  NF  SSD  SDD  {ORC}  Du  Ou  Dw  Ow  ¥
     {{layer1 layer2}  RA0  TG_format}  > TG.log
```

MFX-CT に適した hp_fdk の起動パラメータ ORC の値の探索用 C-shell script "test.0500"（この E-mail に添付した test.0500.txt）と、その値を指定して hp_fdk で画像すべてを再構成する C-shell script "run.byte"

（run.byte.txt）を vrm のディレクトリ /media/disk/tsukasa/180725/ の下に置いておきました。また、それらを実行するため、ディレクトリ

/media/disk/bl20b2/2018.07.25_alignment/[600,1800]prj_2x2/001/raw/

の下に置いてあった 600 投影と 1800 投影の各測定データをディレクトリ

/media/disk/tsukasa/180725/[600,1800]/raw/

の下にハードリンクしました。なお、test.0500 と run.byte は投影数には依存しないので、ディレクトリ [600,1800]/ の中から実行すれば OK です。

test.0500 はスライス番号が 0500 の画像を ORC の値を変えて再構成します：

ORC = { -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2 } × 1.3e-3

ただし、 1.3e-3 は X 線検出器の水平方向の画素幅（Du）。

これら 9 枚の画像を並べたものを PostScript 画像として出力します：

chdir 600 ; csh ../test.0500 > 0500.ps ; chdir ..

chdir 1800 ; csh ../test.0500 > 0500.ps ; chdir ..

この E-mail に添付した 0500.pdf に上の 2 枚の PostScript 画像を並べました。これより、投影数によらず ORC = 1・1.3e-3 を指定すればまともな再構成画像になることがわかります。

run.byte は ORC = 1・1.3e-3 を指定した 3 次元画像再構成を行います：

chdir 600 ; csh ../run.byte ; chdir ..

chdir 1800 ; csh ../run.byte ; chdir ..

これにより以下のディレクトリとファイルが作成されます：

byte.log：再構成したスライス画像それぞれの CT 値の最小値と最大値

byte/：すべてのスライス画像で同じ明暗にした 8 ビット画素値の画像

byte.csv：byte 画像のヒストグラム（この E-mail に添付した hg.pdf）

byte.tif：byte 画像の browse 画像（この E-mail に添付した byte.pdf）

とりあえず以上です。

Tsukasa NAKANO wrote at 2018 年 8 月 7 日 10:36

>

> 上杉さま、

> GSJ/AIST のなかのです。体調を崩しているので、お返事は少々時間をください。

>

> --------

> 差出人: Kentaro UESUGI

> 送信日時: 2018 年 8 月 5 日 2:48

> 宛先: 中野司

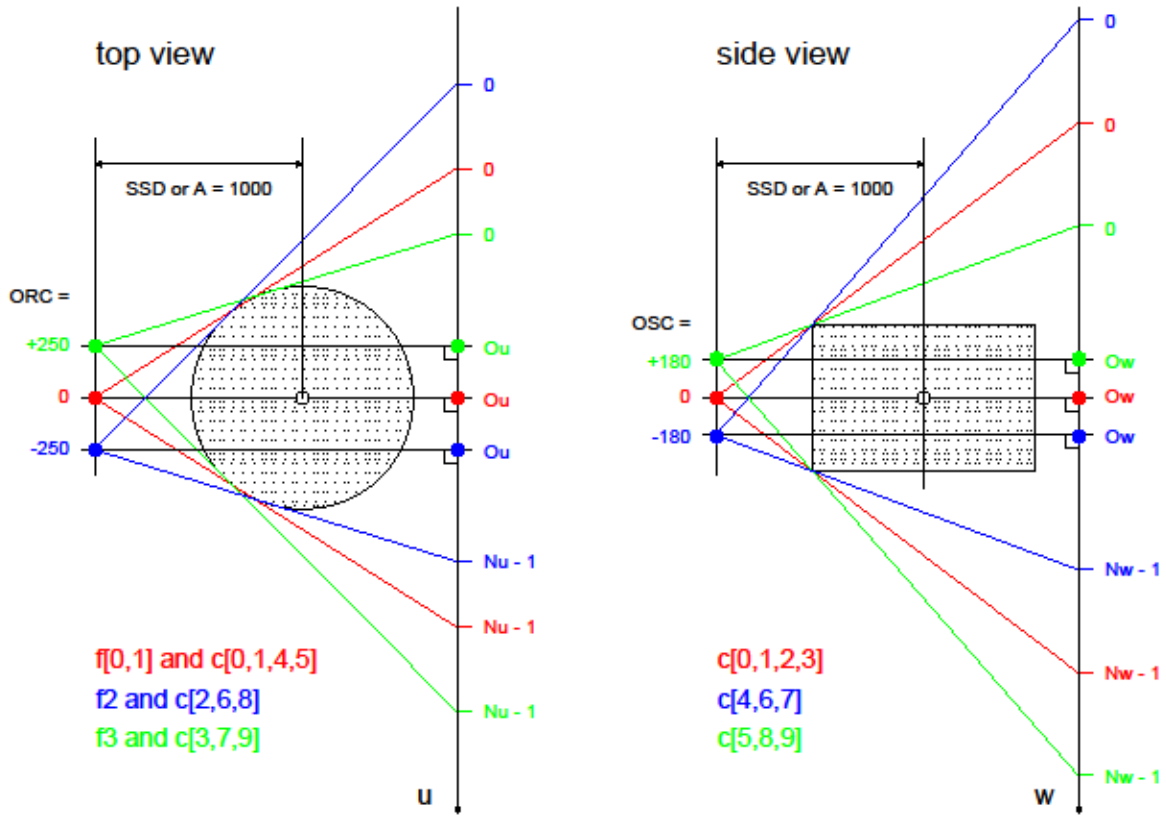
> CC: jmatsuno; uesugi; atsuchi; miya

> 件名: Re: Fw: CB_sphere

>

> 中野さん
>
> 上杉です
>
> MFX-CT ですが、ほど近いところまでは行ったのですが、
> 微妙に再構成画像がおかしな事になっています。
> データを見て検討していただけませんでしょうか？
> vrm の /home/bl20b2/001.zip に置きました。
> ユーザー名は bl20b2 です。パスワードの法則は bl47xu と一緒です。
> tiff float の投影像と、いくつかのパラメーターで再構成した画像が
> ディレクトリごとに分けてあります。また、その際に使った
> バッチファイルも入っています。再構成は旧版の fdk を使っていますが、
> もしかしたら、新版のやつで位置ズレを直さないとダメなのかもしれません。
> ただ、光軸中心と回転中心は大体画像の真ん中あたりにはいます。
> ((511,511) から 1-2pixel 以内と思っています)

添付ファイル 040711j_layout.pdf（4/18 の E-mal に添付したものと同じ画像です）



添付ファイル test_0500.txt

```
#
set layer=`basename $0 | cut -d. -f2`
set c_ds=(`seq -f %g -2 0.5 2`)
set ratio=0.99
#
set a=8.5
set b=29.37
set d=1.3e-3
set ou=511.0
set ow=511.0
set ra0=0
#
setenv THREADS 8
#
set fs=8
set dpi=500
set pile=(-n -3 3 -o $fs $fs)
#
set tmp=/dev/shm/${layer}.$$
#
mkdir ${tmp}.0
foreach c_d ($c_ds)
  hp_fdk raw $a $b `echo $c_d $d | awk '{ printf "%e¥n",$1*$2 }'` ¥
                $d $ou $d $ow $layer $layer $ra0 ${tmp}.0/$c_d >& /dev/null
end
#
@ line = `( t_f2txt ${tmp}.0 - %lf - | ¥
                awk -v R=$ratio ¥
                    'NR>=2 { for (f=1; f<=NF; f++) if ($f>0) { print $f; L+=R }} ¥
                     END { print int(L+1) > "/dev/stderr" }' | ¥
                sort -n >${tmp}.1 ) |& tail -1`
set pv=`sed -n ${line}p ${tmp}.1`
#
set step=`echo $pv / 255 | bc -l`
echo /Helvetica findfont $fs scalefont setfont >! ${tmp}.1
foreach c_d ($c_ds)
  set tiff=${tmp}.0/$c_d
  set nm=(`pig $tiff | cut -f1-2` `pid $tiff`)
```

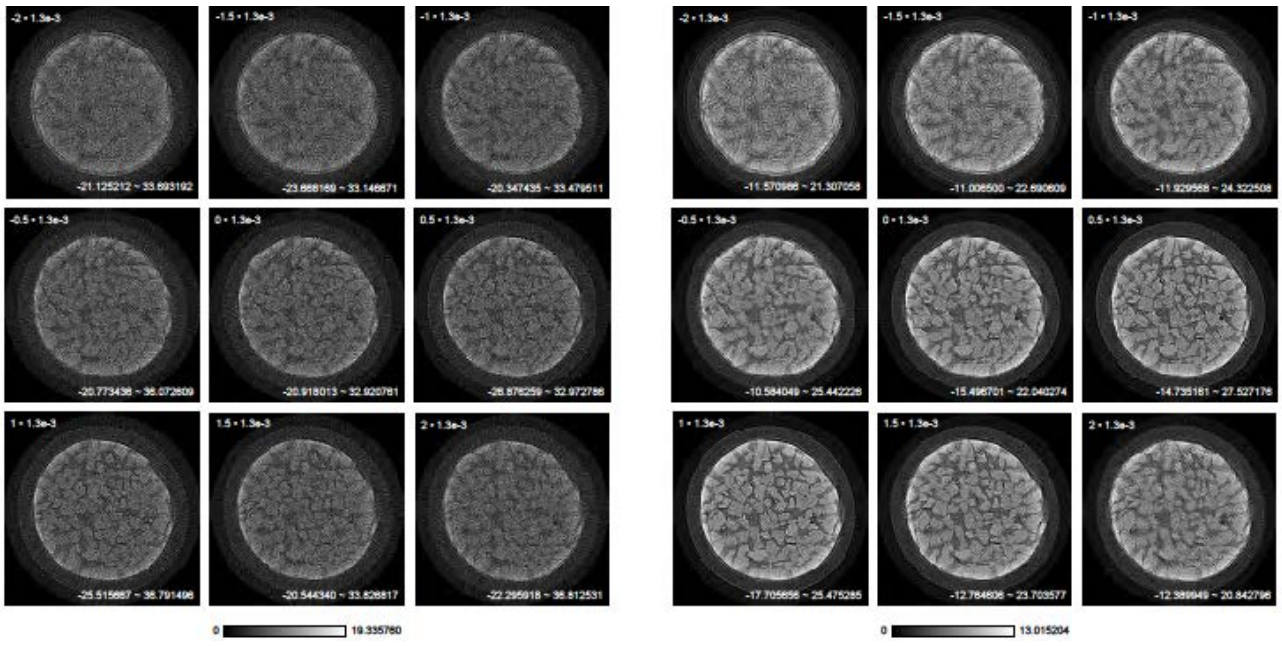
```
    t2t_float $tiff 0 $step 8 $tiff >/dev/null
    ( t2ps_P $tiff $dpi - - | sed s/¥^%%BoundingBox:/%%PageBoundingBox:/g ; ¥
      echo 1 setgray ; ¥
      echo ¥($c_d ¥¥¥¥267 ${d}¥) ; ¥
      echo $fs 0.5 mul $nm[2] $dpi div 72 mul $fs 1.375 mul sub moveto show ; ¥
      echo ¥($nm[3] ¥~ ${nm[4]}¥) ; ¥
      echo dup stringwidth pop $fs 0.5 mul add $nm[1] $dpi div 72 mul sub neg ¥
          $fs 0.625 mul moveto show showpage ) >>${tmp}.1
  end
  rm -r ${tmp}.0
  #
  ( epspile $pile ${tmp}.1 ; ¥
    bar_gs.csh `echo 640 / $dpi ¥* 72 | bc -l` $fs $fs 0 $pv ) | ¥
  gspile -n 1 2 -o 0 `echo $fs ¥* 2 | bc -l` ; rm ${tmp}.1
```

添付ファイル run.byte.txt

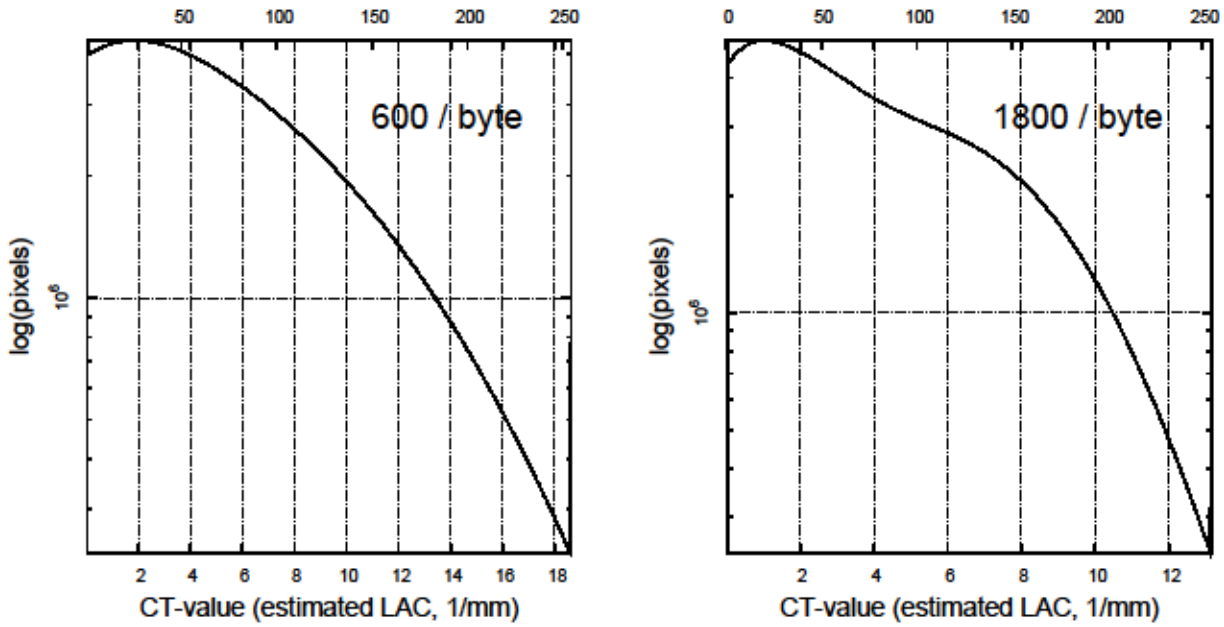
```
  #
  set ratio=0.99
  #
  set a=8.5
  set b=29.37
  set c=1.3e-3
  set d=1.3e-3
  set ou=511.0
  set ow=511.0
  set ra0=0
  #
  setenv THREADS 8
  #
  set nxy=(3 4)
  set fn=Helvetica
  set fs=`expr ¥( 8 ¥* 500 + 72 - 1 ¥) / 72`
  #
  set tmp=/dev/shm/byte.$$
  #
  mkdir byte
  set hp_fdk=(hp_fdk raw $a $b $c $d $ou $d $ow)
  $hp_fdk $ra0 byte/%0`$hp_fdk |& awk '{ print length($2-1) }'`d.tif >& byte.log
  #
```

```
set bs=(`awk 'NR==2 { min=$2; max=$3 } ¥
                NR>=3 { if ($2<min) min=$2; if ($3>max) max=$3 } ¥
                END { printf "%s %e¥n",min,(max-min)/65535 }' byte.log`)
set pv=`t2t_float byte - $bs 16 - | ¥
        awk -v R=$ratio -v B=$bs[1] -v S=$bs[2] ¥
            '$2+$3>=0 { C[$1]=O+$4; O=C[$1] } ¥
             END { O*=R; for (I=0; C[I]<O; I++) ; printf "%f¥n",B+S*I }'`
t2t_float byte - 0 `echo $pv / 255 | bc -l` 8 -byte | tr ¥¥t , > byte.csv
#
mkdir $tmp
foreach csv (`awk -v Nx=$nxy[1] -v Ny=$nxy[2] ¥
                    'NR==1 { L=$2-1; F="%0" length(L) "d,%s,%s¥n" ; ¥
                            N=Nx*Ny; D=int(L/N); O=int((L-D*(N-1))/2) }  ¥
                     NR>=2 { if (($1-O)%D==0) printf F,$1,$2,$3 }' byte.log`)
  set ssv=(`echo $csv | tr , ' '`)
  cp byte/${ssv[1]}.tif ${tmp}/$ssv[1]
  ( echo $fn $fs 255 LU $ssv[1] ; ¥
    echo $fn $fs 255 RD $ssv[2] ¥~ $ssv[3] ) | add_label ${tmp}/$ssv[1]
end
pile_gray -$nxy -$fs -$fs 255 byte.tif ${tmp}/* ; rm -r $tmp
#
bar_gs.csh 640 $fs $fs 0 $pv $tmp
pile_gray 1 2 0 -`expr $fs ¥* 2` 255 byte.tif byte.tif $tmp ; rm $tmp
```

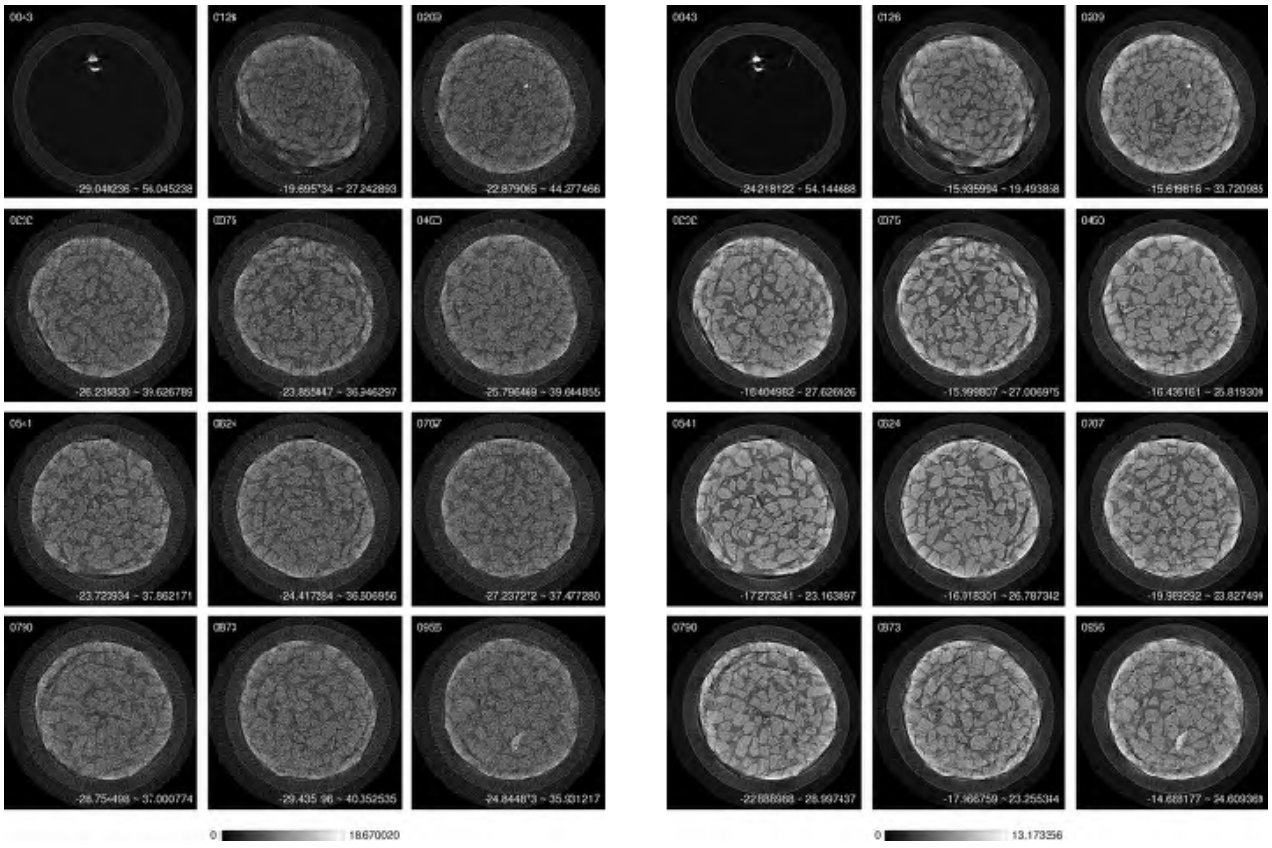
添付ファイル 0500.pdf



添付ファイル hg.pdf



添付ファイル byte.pdf



Date: Fri, 05 Oct 2018 18:02:19 +0900
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: jmatsuno, uesugi, atsuchi, miya
Subject: MFX-CT_訂正

うえすぎさま、
みなさま、

GSJ/AIST のなかのです。先程お送りした MFX-CT の E-mail に間違いを見つけました。CB CT の画像再構成プログラム hp_fdk の起動パラメータ ORC の値の計算に間違った「X 線検出器の水平方向の画素幅、Du」の値を記していました。

誤：1e-3
正：1.3e-3

C-shell scripts "test.0500" と "run.byte" では正しい値 1.3e-3 を使っているので問題なしですが、先の E-mail の文中ではそれを 1e-3 としていました（4 個の値 1e-3 を 1.3e-3 に修正しました）。すみません。
とり急ぎ、

Date: Thu, 11 Oct 2018 15:02:33 +0900
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: jmatsuno, uesugi, atsuchi, miya
Subject: MFX-CT_追加

うえすぎさま、

GSJ/AIST のなかのです。上杉君が言っていることを理解できないので、先週の E-mails に記した画像再構成プログラム hp_fdk と fdk の追加の説明をします。

```
>>      hp_fdk  HiPic/  SSD  SDD  ORC  Du  Ou  Dw  Ow  ¥
>>              {{layer1  layer2}  RA0  {BPS}  TG_format}  > TG.log
>>      fdk     XP/  NF  SSD  SDD  {ORC}  Du  Ou  Dw  Ow  ¥
>>              {{layer1 layer2}  RA0  TG_format}  > TG.log
```

(1)
hp_fdk と fdk はディレクトリ HiPic/と XP/のそれぞれの下の異なる形式の測定データから同一の FDK 法のアルゴリズムによって CB-CT の画像再構成処理を行うプログラムです。ただし、ぼくの計算機環境では

書庫ファイル

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

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

に入っている fdk のソースコードをコンパイルしていません。fdk ではわかりづらいので、それを xp_fdk のようなファイル名に変更すべきだと思っています。できることなら上杉君もそのようにしていただけると助かります。

(2)

計算機 vrm のディレクトリ

/media/disk/bl20b2/2018.07.25_alignment/[600,1800]prj_2x2/001/

を見ると MFX-CT の測定データは hp_fdk 用の HiPic/ に相当する raw/ のようですね。そして、fdk（xp_fdk）を実行するためには raw/ の下のデータを変換した X 線投影値画像をディレクトリ XP/（ここではそれを raw_f/ とします）の下に入れておく必要があります：

mkdir raw_f/ ; hp2xp raw/ - raw_f/%04d.tif > raw_f.log

ただし、この変換には長い処理時間を要します：

前記の投影数が 600 のデータの変換：61.4 秒（vrm）/ 29.9 秒（gsjgix）

前記の投影数が 1800 のデータの変換：182.6 秒（vrm）/ 187.8 秒（gsjgix）

また、raw_f/ のデータ量は raw/ の約 2 倍なので、xp_fdk による画像再構成の処理時間は hp_fdk よりも長くなります。上杉君が言うように raw_f/ を用いた位相回復やリング偽像の軽減処理は有望ですが、現時点ではそれは不要では？ hp_fdk で単純な CB-CT の画像再構成を高速に行う方が良いように思えます。

(3)

MFX-CT のサンプル回転軸の位置を変えた場合、hp_fdk などに起動パラメータとして指定する SSD（source-sample distance）と ORC はそれに応じた値になります。ただし、SSD の誤差は CT 値の一様な誤差になるだけなので再構成画像上では目立ちません。一方、ORC に誤差があると再構成画像上に偽像が発生します。それを調べる C-shell script "test.0500" を先週の E-mal に添付しました。問題は test.0500 の処理の遅さです。このスクリプトは ORC の値を変えた 9 枚の再構成画像の CT 値の値域を同じにする処理を行っていますが、それに時間がかかり過ぎです。そのためのコードを除去した C-shell script "run.0500.txt" を作成しました。run.500.txt は ORC の値を変えて再構成した画像をディレクトリ 0500/ の下に書き込みます。

chdir 600 ; csh ../run.0500.txt ; chdir ..

chdir 1800 ; csh ../run.0500.txt ; chdir ..

こちらの計算機 gsjgix を使うとこれらの総処理時間は 20 秒以下でした。なお、ディレクトリ 0500/の下の 9 枚の再構成画像のファイル名はそれらに指定した ORC の値を Du の値 1.3e-3 で割った値です。

>> ORC = { -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2 } × 1.3e-3

>> ただし、 1.3e-3 は X 線検出器の水平方向の画素幅（Du）。

とりあえず以上です。

On Sat, 06 Oct 2018 0:50 Kentaro UESUGI wrote:
>
> 中野さん
> 上杉です
>
> ちょっと今はビームラインなので、ちゃんと確認できませんが、ORC は振って 1 番良いところを
> 探したんだったと思います。(0.5 ずつ±3 画素分くらいは行ったような)
> ＢＬが一段落したらもう 1 回やってみます。
> 位相回復処理を挟む都合で hp_fdk ではなくて fdk を使います。あと、リング偽像を軽減する新しい
> やり方も試したいとか、色々あり img から直で再構成する機会が減りそうな予感がしています。
> かっこよく言うと pre-process なんですかね。
>
>> GSJ/AIST のなかのです。7/19 から「クモ膜下出血」で入院していましたが、
>> 9/20 に退院して自宅に戻り、10/1 より GSJ/AIST での研究生活を再開しました。
>> 体調は悪くないです。10/24 からの SPring-8 実験には参加するつもりでいます。
>
> おお。先週あたりにあまりに何もないので「生きてますか」ってジョークで送ろうかと思いついた
> のですが、あんまりジョークになってなかったですね。アブナイアブナイ。後学のためと武勇伝を
> 楽しみにしていますが、無理はしないで下さい。

添付ファイル run.0500.txt

```
#
set layer=`basename $0 | cut -d. -f2`
#
set a=8.5
set b=29.37
set d=1.3e-3
set ou=511.0
set ow=511.0
set ra0=0
#
setenv THREADS 8
#
mkdir $layer
foreach c_d (-2 -1.5 -1 -0.5 0 0.5 1 1.5 2)
  hp_fdk raw $a $b `echo $c_d $d | awk '{ printf "%e¥n",$1*$2 }'` ¥
                $d $ou $d $ow $layer $layer $ra0 ${layer}/$c_d >& /dev/null
end
```

Date: Mon, 15 Oct 2018 16:56:39 +0900
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: jmatsuno, uesugi, atsuchi, miya
Subject: MFX-CT_処理時間

うえすぎさま、

GSJ/AIST のなかのです。投影数が 600 と 1800 のそれぞれの CT 画像すべての再構成に要する hp_fdk の処理時間を調べてみました。ただし、先々週の E-mail で紹介した run.byte を使用すると画像再構成以外の処理に時間を浪費するので、そのコードを除去した画像再構成専用の C-shell script "run.tg"（この E-mai に添付した run.tg.txt）を実行しました。

起動パラメータ（$a、$b、$c、$d、$ou、$ow と $ra）の値の設定後、run.tg は以下のようにして hp_fdk を実行します：

```
mkdir  tg
hp_fdk  raw/  $a  $b  $c  $d  $ou  $d  $ow  $ra0  tg/%04d.tif >& tg.log
```

ディレクトリ tg/の下には再構成した 32 ビット画素値の CT 画像が入ります。また、tg.log は run.byte の byte.log と同一の内容のテキストファイルです。

run.tg はディレクトリ raw/ の下の測定データの投影数には依存しないので、run.byte などと同様にディレクトリ 600/ や 1800/ の外部に置けば OK です。そして、プログラム stop_watch などを使えば run.tg の処理時間（== CT 画像すべての再構成に要した時間）を知ることができます：

```
chdir  600 ;  stop_watch  csh  ../run.tg  ;  chdir  ..
chdir 1800 ;  stop_watch  csh  ../run.tg  ;  chdir  ..
```

こちらにある計算機 gsjgix と SPring-8 の vrm での処理時間（単位は秒）は以下のようになりました：

計算機	投影数 600	1800
gsjgi	174.895511	452.504740
vrm	508.221846	1385.177955

とり急ぎ、

添付ファイル run.tg.txt

```
#
set a=8.5
set b=29.37
set c=1.3e-3
set d=1.3e-3
set ou=511.0
set ow=511.0
set ra0=0
#
setenv THREADS 8
#
mkdir tg
hp_fdk raw $a $b $c $d $ou $d $ow $ra0 tg/%04d.tif >& tg.log
```

Date: Fri, 26 Oct 2018 16:47:27 +0900
From: Tsukasa NAKANO
To: Kentaro UESUGI
Cc: Akira TSUCHIYAMA, MATSUNO Junya, Miyake, Masayuki UESUGI
Subject: MFX-CT_Windows

うえすぎさま、

GSJ/AIST のなかのです。ぼくの手元にある Windows 8.1 機（ノート PC、DBR73）を使って投影数が 600 と 1800 の MFX-CT 画像すべての再構成を行いました。この E-mail に添付したバッチファイル tg.bat（tg.bat.txt）に記されているように、その処理内容は以前に差し上げた C-shell script "run.tg"のものと概ね同じです。ただし、4 GB 以上のメモリを使うので書庫ファイル

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

に入っている 64 ビット CPU 用の hp_fdk_64.exe を実行しました。それぞれの投影数の画像の再構成の処理時間は以下の通りです：

	計算機	投影数 600	1800
>	gsjgix	174.895511	452.504740
>	vrm	508.221846	1385.177955
	DBR73	1542.0461268	3182.8373144

ぼくの「病後の後始末」はこれで終わりです。以上のような MFX-CT の再構成画像をつちやまさんに見せましたか？ とり急ぎ、

添付ファイル tg.bat.txt

```
@echo off

set a=8.5
set b=29.37
set c=1.3e-3
set d=1.3e-3
set ou=511.0
set ow=511.0
set ra0=0

set THREADS=8

mkdir tg
..¥hp_fdk_64 raw %a% %b% %c% %d% %ou% %d% %ow% %ra0% tg¥%%04d.tif > tg.log 2>&1
```

Date: 2018 / 11 / 16
Subject: MFX-CT_余談

みなさま、

GSJ/AIST のなかのです。プログラム hp_fdk を使った MFX-CT の画像再構成に関する余談です。

(1)
最近購入した Windows 10 搭載のノート PC（Panasonic CF-SV73FRQR；計算機名 CF-SV）を使って投影数が 600 と 1800 のそれぞれの CT 画像すべてを再構成してみました：

```
chdir 600 && ..¥stop_watch_64 ..¥tg.bat && chdir ..
chdir 1800 && ..¥stop_watch_64 ..¥tg.bat && chdir ..
```

その秒単位の処理時間は以下の通りで、新しい計算機 CF-SV は Windows 8.1 搭載の DBR73（東芝 dynabook R73/W6M）よりも高速ですが Linux 機（gsjgix や vrm）に比べると全然低速です。

>>	計算機	投影数 600	1800
>>	gsjgix	174.895511	452.504740
>>	vrm	508.221846	1385.177955
>	DBR73	1542.0461268	3182.8373144
	CF-SV	1353.4108329	2439.9146559

(2)

このような Linux 機と Windows 機で実行した hp_fdk の処理速度の大きな違いについて考えます。これは使用した CPUs の処理速度の違いだけで生じたものではありません。hp_fdk は

ディレクトリ raw/ に入っているサンプル回転角ごとの測定した X 線強度画像を読み込み、
それらから得た X 線投影値の Convolution Back-Projection（CBP）演算により
３次元 CT 画像のスライス画像すべての再構成

を行いますが、その処理に使った外部記憶装置

Linux 機では計算機の内臓ハードディスク（HDD）
Windows 機ではネットワーク接続した装置（NAS；Buffalo TeraStation WS-QV8.0TL/R5）

の処理速度の違いが重要です。ただし、hp_fdk は X 線強度画像の読み込みごとに CBP 演算による画像再構成処理を行うので、ディレクトリ raw/ の下のファイルの読み込みとそれらに対する CPUs の処理の速度を分離することができません。そこで、そのすべての処理が終了した後にディレクトリ tg/ の下に書き込んだ CT 画像のファイルそれぞれの作成時刻を調べてみました。それらの最大値から最小値を引き算した tg/ の下の画像すべての作成時間（単位は秒）は以下の通りです。

host	views	hp_fdk	tg/	hp_fdk – tg/
---- HDD				
gsjgix	600	174.895511	61	113.895511
gsjgix	1800	452.504740	37	415.504740
vrm	600	508.221846	71	437.221846
vrm	1800	1385.177955	69	1316.177955
---- NAS				
DBR73	600	1542.0461268	799	743.046127
DBR73	1800	3182.8373144	924	2258.837314
CF-SV	600	1353.4108329	798	555.410833
CF-SV	1800	2439.9146559	782	1657.914656
---- NAS				
gsjgix	600	332.659452	152	180.659452
gsjgix	1800	696.030986	157	539.030986

(3)

このように「Linux機＋HDD」による画像 tg/ の書き込みは「Windows機＋NAS」の場合よりも10 倍以上高速です（おかしな値も発生していますが、その原因は不明です）。また、新たに行った「gsjgix（Linux機）＋NAS」によるhp_fdk の処理時間でも同様です。そして、hp_fdk の処理時間と画像 tg/ の作成時間の差の値「hp_fdk – tg/」からわかるように、CF-SV の処理速度は Linux 機 vrm のものに近い値です。つまり、hp_fdk による画像再構成処理では高速な CPUs だけではなく高速な外部記憶装置を使うことが重要です。

とりあえず以上です。