北大医学部保健学科 放射線技術科学専攻3年前期

核医学機器工学概論　C言語

6月10日　講義5　畳み込み演算　filter.zip　Ramp.zip

6月17日　講義5 復習

6月24日　講義6　断層画像再構成法　SimpleBackProjection.

7月 1日　期末試験 解答　成績

PET SPECT 画像再構成の原理　OSEMの原理

7月1日 期末試験　核医学機器工学概論
試験問題は、以下の 問題1 から問題10 のうち4問、
（画像サイズ等は変更して出題 = 丸暗記では答えられない）
および応用問題1問を出題します。（合計50点）

プログラム文終端の セミコロン ; の記述忘れに注意。
変数の型に注意。 キャストの記述忘れに注意。
配列の添字に注意。宣言範囲外の変数を入れないように。
括弧（ ）[ ] { } の使い分けに注意。混同しないように。

問題　1.
整数1からNまでの 合計 sum、平均 mean、分散Variation v
（v＝{ Σ{ (mean－i) ＊ (mean－i) } }／ N ）および
標準偏差 Deviation sd （ sd = sqrt(v); sqrt( )関数は平方根
を計算するC言語関数）を求めるCコードを記述して下さい。

```
void Main(void)      //  問題  1
{
   int          [          ] ;
   double       [          ] ;

   TextWindow(0,0,300,300);  Title("Calculate  Mean、SD");

   printf("¥n N = ") ;  scanf( [          ] );

   sum = 0 ; [          ] ;

   [                                        ]

      printf("¥n Sum         = %d" ,  sum   ) ;
      printf("¥n Mean        = %lf" , mean  ) ;
      printf("¥n Variation   = %lf" , v     ) ;
      printf("¥n Deviation   = %lf" , sd    ) ;
}
```

問題　2.

１画素2バイトの 256x256画素の 画像ファイルを読込んで
配列 img[260][260] に入力するプログラムを記述して下さい。

```
void Main(void)
{
    char   a, b, f[100] ;
    int      i,  j, img[260][260] ;
    
    
    GetFileName( f, 0 ) ; printf( “ file = %s ” , f ) ;
    
    
}
```

問題　3.

配列 img[ ][ ] に入力された 256x256画素の画像（1画素の実長は 1.695mm、画像サイズは 43.4 x 43.4cm ）の計数密度（count/cm2）を求めるプログラムを記述して下さい。ただし、カウントが 0 の部位は計数に加えないようにする。

```
//------------ Calculate count density ------------

 int            i,  j, total_count, pixel , img[260][260];
 double         area, density ;

 printf("¥n¥n Count density = %.2lf (count/cm2)", density );
```

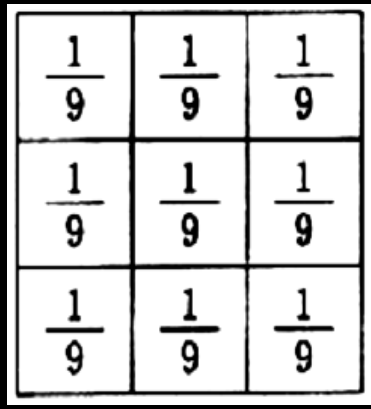
問題　4.

配列 x[260][260] に入力された 256x256画素の 画像に
3x3メディアンフィルタをかけて配列 y[260][260]に出力する
関数 median() を記述して下さい。

```
void median( int x[ ][260], int y[ ][260] )
{
        int   i, j, k, mi, mj, ni, nj, min, med[10] ;

}
```

問題　5.

配列 x[260][260] に入力された 256x256画素の画像に、右図に示す3x3平滑化フィルタ（移動平均フィルタ）をかけて配列 y[260][260]に出力する関数 smoothing() を記述して下さい。




```
void smoothing( int x[ ][260], int y[ ][260] )
{

}
```

問題　6.

配列 x[260][260] に入力された 256x256画素の画像の最大画素値を整数値で出力する関数 get_int_max() を記述して下さい。

```
          get_int_max( int x[ ][260] )
 {
 }
```

問題　7.

１次元データを g(x)、フィルタ関数を h(x)、フィルタ処理後のデータを l(x)、それぞれのフーリエ変換を G(f)、H(f)、L(f) として、(x は実空間座標、f は周波数)
畳み込みの定理 を 文章と数式を用いて 説明して下さい。

スリットを取りはずすと（スリット間隔を無限に狭くすると）
任意の座標 x における フィルムの濃度 $l(x)$ は

$$l(x) = \int_{-\infty}^{\infty} f(n)\, h(x-n)\, dn$$

Convolution 積分
（たたみ込み積分）

これを $l(x) = f(x) * h(x)$ と表わす。

$h(x)$ を convolution 関数という。

$f(x)$ が $h(x)$ のために $l(x)$ に ボケてしまったことを意味する。

たたみ込みの定理

$h(x-n)$ の Fourier 変換を $H(f)$ とすると

$$h(x-n) = \int H(f)\, e^{j(2\pi f(x-n))}\, df \quad (\text{逆 Fourier 変換})$$

これを $l(x) = \int f(n)\, h(x-n)\, dn$ に代入すると

$$l(x) = \int f(n) \left[\int H(f)\, e^{j(2\pi f x)}\, e^{-j(2\pi f n)}\, df \right] dn$$

$$= \int \left[\int f(n)\, e^{-j(2\pi f n)}\, dn \right] H(f)\, e^{j(2\pi f x)}\, df$$

$$= \int \boxed{G(f)}\, H(f)\, e^{j(2\pi f x)}\, df$$

$l(x)$ の Fourier 変換を $L(f)$ とすると

$l(x) = \int L(f)\, e^{j(2\pi f x)}\, df$, よって $\boxed{L(f) = G(f)\, H(f)}$

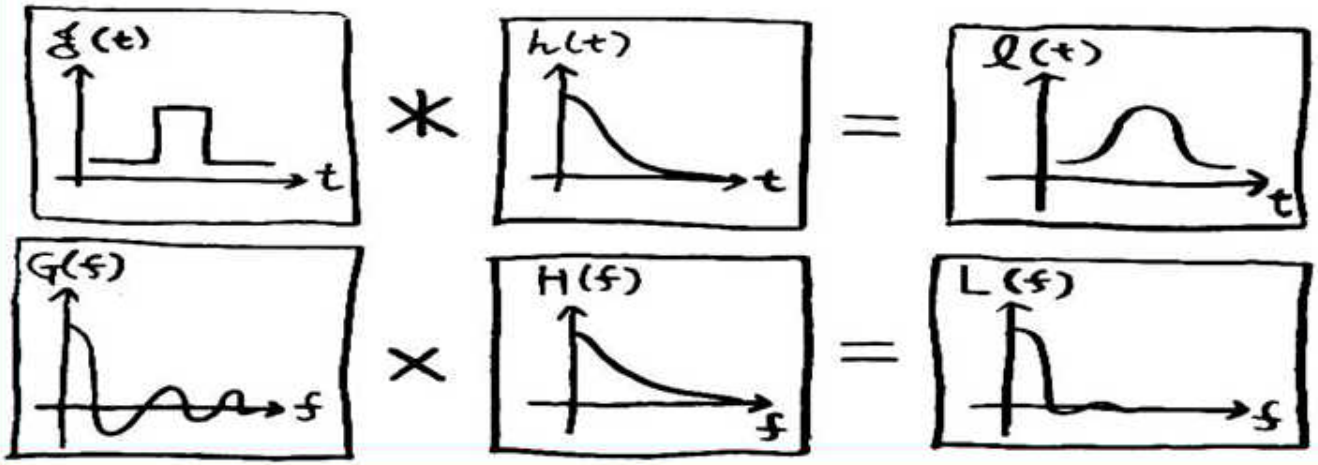
Date 20

位置(x)や時間(t)の関数gが装置の特性hによって
ボケてlになることは，

曲線gの細かい変動（高周波成分）がhによって
減衰して大ざっぱな曲線lになることを示す．

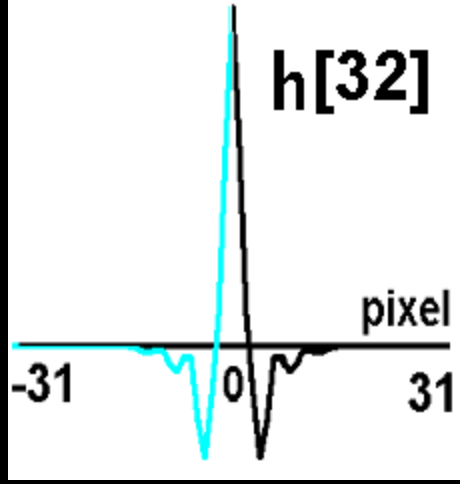
つまり周波数空間において，G(f)はH(f)によって
高い周波数の成分だけ減衰させられて
高周波成分の少なくなったL(f)になる．

convolutionが周波数空間において乗算として
示されることは，fが大きい値のときにH(f)は小さい
値をとるfilterで，そのためG(f)にH(f)を掛けた
L(f)は高周波成分が小さくなって，そのために
lは大ざっぱな曲線にボケると解釈できる．



問題　8.

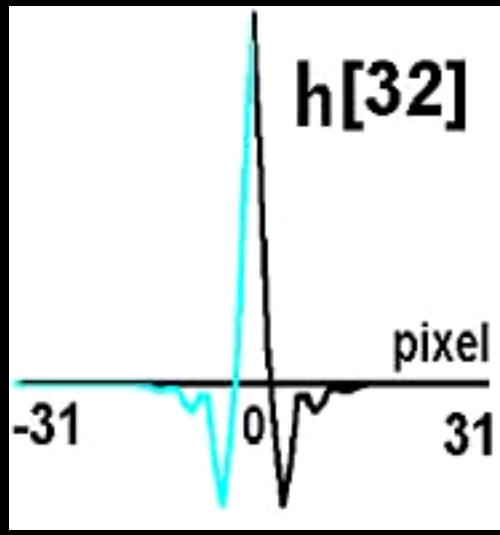
データ数６４の１次元データ g[0]～g[63] と、
データ数３２の実空間フィルタ h[0]～h[31]
がある（ h[0] が原点）。



データ g に フィルタ h を畳み込んで
配列 I [0]～ I [63] に書き込むプログラムを記述して下さい。

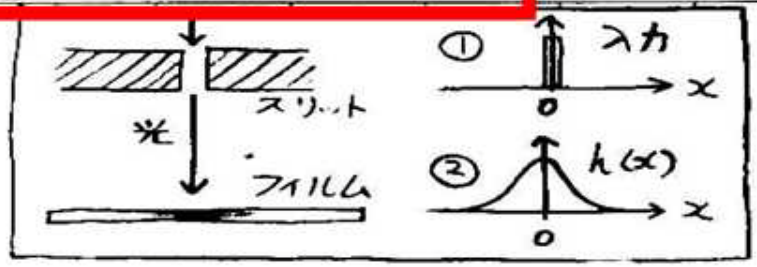
```
//---------- Convolution h[ ] into g[ ]  ------------

    int         i , j ;
    double      g[64] , h[32] , I[64] , gh ;
```

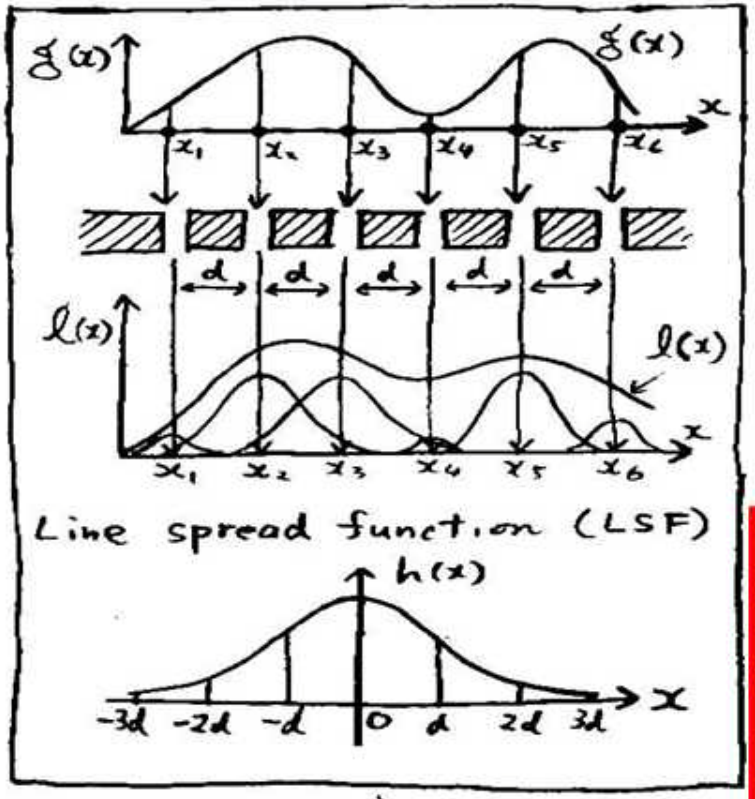
データ数64の1次元データ g[0]～g[63] と、
右図に示すようなデータ数32の実空間フィルタ
h[0]～h[31] がある（ h[0] が原点）。
データ g に フィルタ h を畳み込んで
配列 I [0]～ I [63] に書き込むプログラムを
記述して下さい。




```
//---------- Convolution h[ ] into g[ ] ----------

    int             i , j ;
    double          g[64] , h[32] , I[64] , gh ;

    for( i = 0 ; i <=63 ; i++ ){                    //   i <64 と書いても同じ

        gh = 0.0 ;

        for( j = 0; j <=31;  j++ ){  if(  i+j <=63 ) gh +=  g[ i+j ] * h[ j ] ;  }

        for( j = 1; j <=31;  j++ ){  if(  i-j >= 0 ) gh +=  g[ i-j ] * h[ j ] ;  }

        I[i] = gh ;
    }
```

④ Convolution





スリットを通したビーム①がフィルム上で②のようなh(x)の濃度分布の像をつくるとする。

x軸に沿って明るさがg(x)である線状の被写体を，間隔dのスリットを通してフィルムに写す。

フィルムのx_4の位置での濃度は

$$l(x_4) = g(x_4)h(0) + g(x_5)h(-d) + g(x_6)h(-2d) + g(x_3)h(d) + g(x_2)h(2d) + g(x_1)h(3d)$$

任意の座標x_iにおける$l(x_i)$は

$$l(x_i) = g(x_i)h(0) + g(x_{i+1})h(x_i - x_{i+1}) + g(x_{i+2})h(x_i - x_{i+2}) + \cdots + g(x_{i-1})h(x_i - x_{i-1}) + g(x_{i-2})h(x_i - x_{i-2}) + \cdots$$

$$= \sum_{n=-\infty}^{\infty} g(x_n)h(x_i - x_n)$$

g に h を畳みこんだ結果を l とする。座標 i における l の値 l[i] は、

l[i] = g[i]*h[0] + g[i+1]*h[−1] + g[i+2]*h[−2] + g[i+3]*h[−3] + ・・・
　　+ g[i−1]*h[1] + g[i−2]*h[2] + g[i−3]*h[3] + ・・・

ここで フィルタ h は偶関数（左右対称）なので、h[−j] = h[j] を代入し、

l[i] = g[i]*h[0] + g[i+1]*h[1] + g[i+2]*h[2] + g[i+3]*h[3] + ・・・
　　+ g[i−1]*h[1] + g[i−2]*h[2] + g[i−3]*h[3] + ・・・

これを C言語で表す。h の要素数 j は h[0] から h[31] まで。
g の 要素数 は g[0] から g[63] までなので、g の 添字を表す i+j は
63 以下、i−j は 0 以上の場合だけ加算する条件式を入れる。

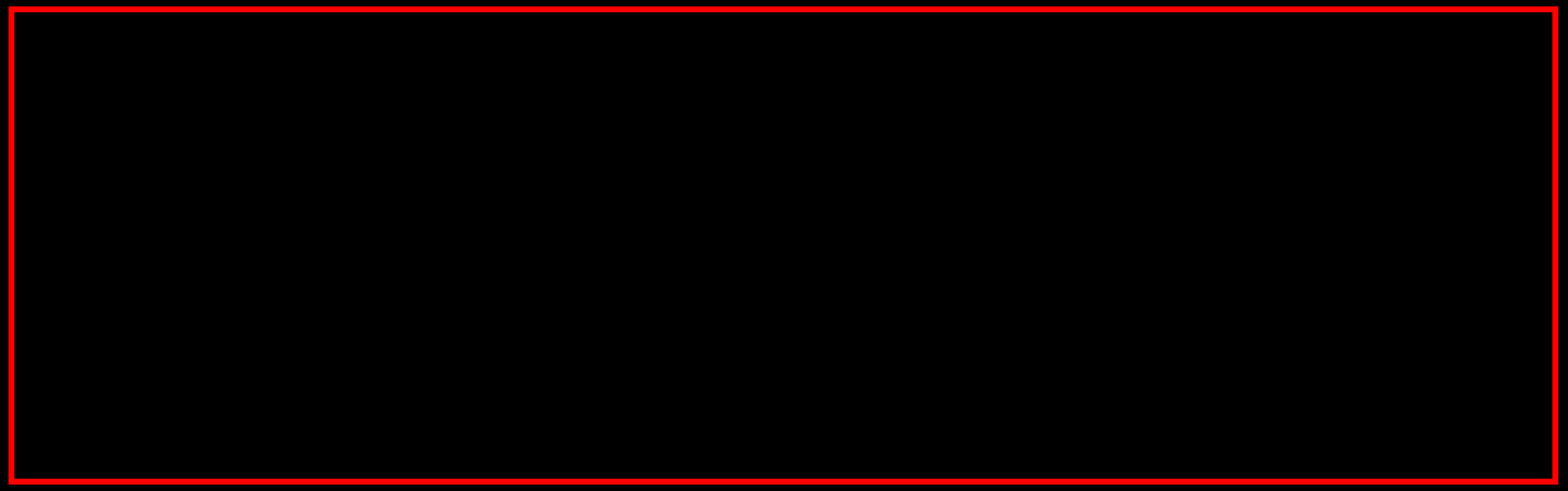
```
gh = 0.0 ;
for( j = 0; j <=31;  j++ ){  if(  i+j <=63 ) gh +=  g[ i+j ] * h[ j ] ;  }
for( j = 1; j <=31;  j++ ){  if(  i−j >= 0 ) gh +=  g[ i−j ] * h[ j ] ;  }
l [ i ] = gh ;
```

これを i が 0 から 63 まで繰り返すと、l [0] から l [63] が計算される。

問題　９.

64x64画素の 画像 x[ ][ ] を 角度 T（度） 回転させて
配列 y[ ][ ] に格納するプログラムを記述して下さい。
ただし座標が整数値であることに伴う、回転後に画素値が
割り当てられない座標が生じる誤差は無視してよい。

```
//-------------------- Image   Rotation ----------------------

   int          i , j , ir , jr ;
   double    I , J , T , rot , x[64][64] , y[64][64] , PAI = 3.1415926 ;
```

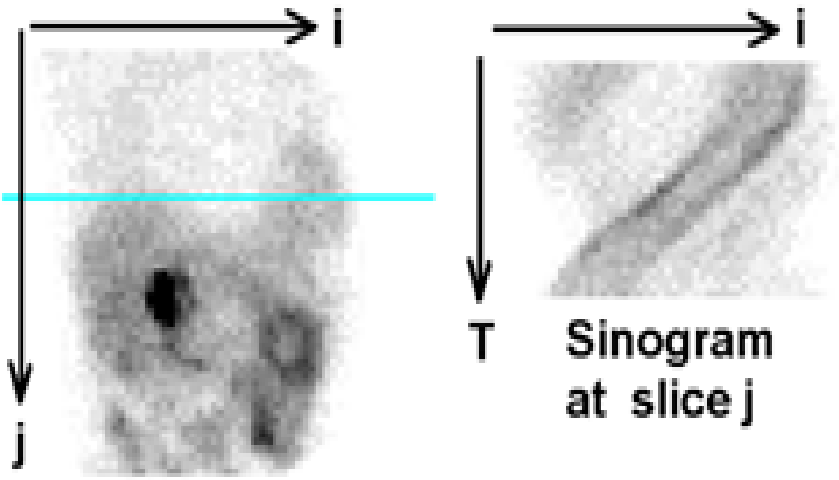


問題　10.
32方向から撮像されたプロジェクション Proj[i][j][T]
(i、j は画像の横、縦座標、Tは撮像方向) から、
フィルタ逆投影再構成法で SPECT像を作成する手順を、
以下のキーワードを用いて(不要なキーワードもある)、
文章、式および図などで説明してください。

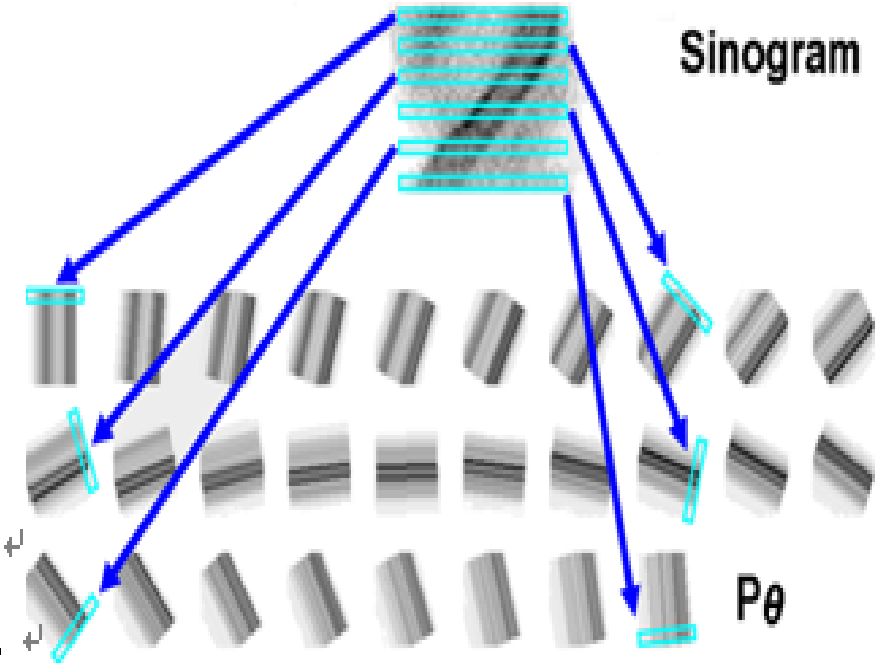
実空間フィルタ h、周波数空間フィルタ H、
1次元フーリエ変換、2次元フーリエ変換、
1次元逆フーリエ変換、2次元逆フーリエ変換、
畳み込み、実空間、周波数空間、実数成分、虚数成分、
2次元透視画像 Pθ、サイノグラム、
Rampフィルタ、 Shepp＆Loganフィルタ、
ナイキスト周波数

SPECTのProjection image Proj[i][j][T]からスライスjの再構成画像を作成する手順。

1. スライスjにおけるサイノグラムを求める。
   サイノグラムの各スライスの1次元配列は、
   ３２方向の各々の角度から収集されたデータ。



2. サイノグラムの各スライスの1次元配列から、
   収集された各々の角度に傾いた2次元透視画像
   Pθを作成する。



3. Pθを単純に重ね合わせた画像をIとすると
   $I = \int P\theta \, d\theta$ (Simple back projection)。
   I は、回転中心部ほど重ね合せ回数が多くなり、
   中心から距離が遠いほどカウントの低い像になる。

4. つまり、回転中心からの距離ｒに反比例した濃度に補正するフィルタ １／ｒ を
正確な断層像 ｇ に畳み込んだ像が Ｉ である。式で表現すると Ｉ＝ｇ＊(１／ｒ) となる。
Ｉ、ｇ、１／ｒ のフーリエ変換を Ｌ、Ｇ、Ｆ(１／ｒ) と表現すると、畳み込みの定理より
Ｌ＝Ｇ･Ｆ(１／ｒ) となる。

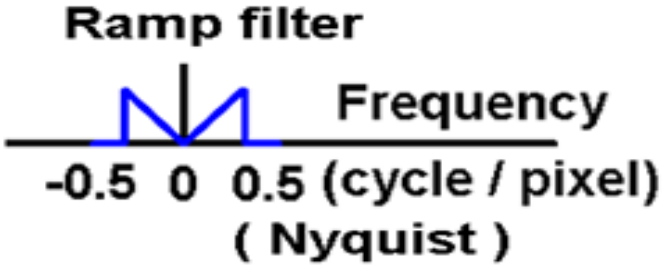
5. ２次元フーリエ変換の公式の極座標表現を用いると、(ｆｒは周波数空間上の原点からの距離)
$F(1/r) = \int\int (1/r) \exp(-j(2\pi r\, fr))\, r\, dr\, d\theta = 1/fr$
これより Ｇ=Ｌ・ｆｒ

6. 以上より、２次元透視画像Ｐθ (＝プロジェクション)を単純に重ね合せた像Ｉのフーリエ変換Ｌに
周波数空間でｆｒを掛けたデータ(＝Ｌ・ fr)を逆フーリエ変換すると、正確な断層像ｇになる。

7. Ｇ=Ｌ・ｆｒ は、畳み込みの定理を用いると、ｇ ＝Ｉ＊ｈ になり、実空間での計算に変換できる。
これに $I = \int P\theta\, d\theta$ を代入すると $g = \int P\theta\, d\theta * h = \int (P\theta * h)\, d\theta = \int \overline{P\theta}\, d\theta$
つまり、Ｐθ に 実空間フィルタｈ(＝ｆｒの逆フーリエ変換)を畳み込めば、
重ね合せると正確な断層像 ｇ になる２次元透視画像 $\overline{P\theta}$ を算出できる。

8.　周波数空間での実際の計算においては、フィルタＨ（＝ｆｒ）は常に正の値であり（絶対値）、
さらにサンプリング定理より、ナイキスト周波数以上の成分を削除する必要があるので、

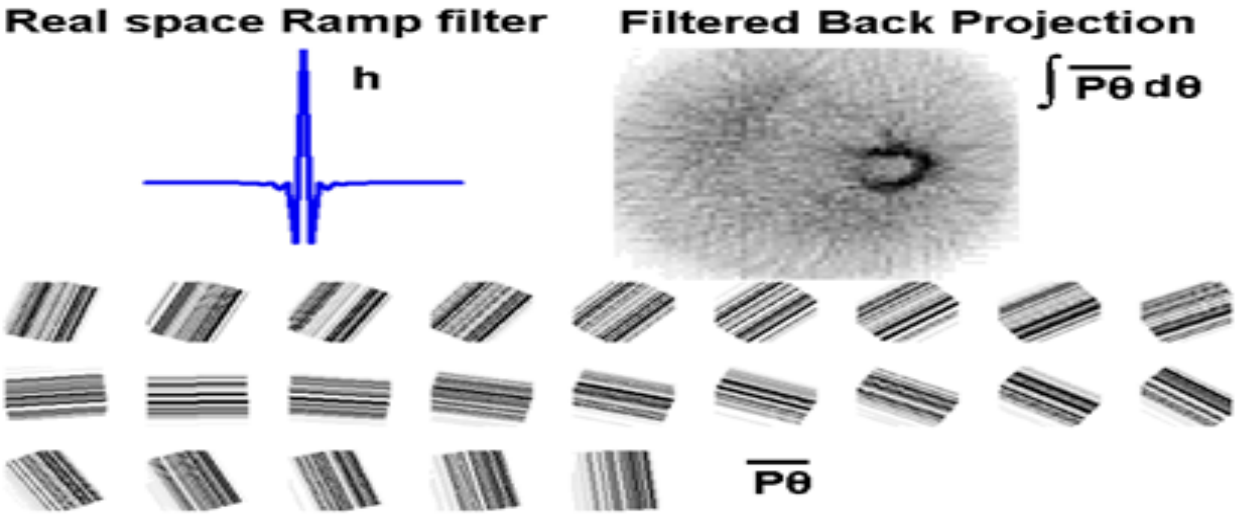
Ｈ＝｜ｆｒ｜（ｆｒがナイキスト周波数未満の場合）
Ｈ＝　０　　（ｆｒがナイキスト周波数以上の場合）
となる。これをＲａｍｐフィルタという。



9.　Ｒａｍｐフィルタを逆フーリエ変換して実空間Ｒａｍｐフィルタｈに
してから、実空間でＰθにｈを畳み込む。

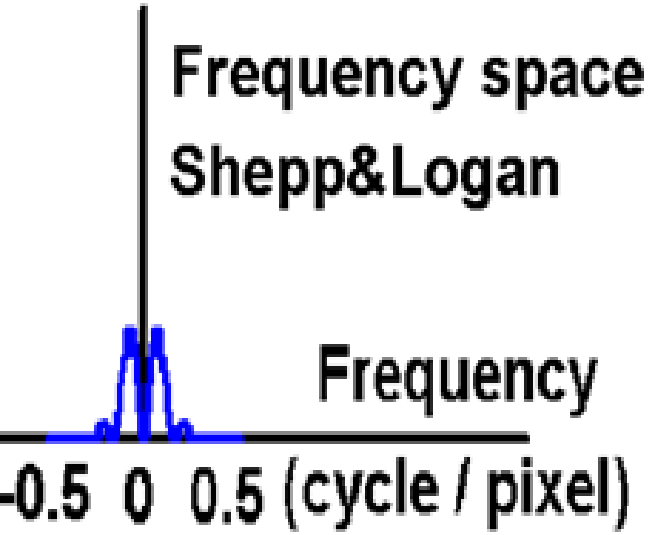
$\overline{P\theta} = P\theta * h$　　（＊は 畳み込み演算）

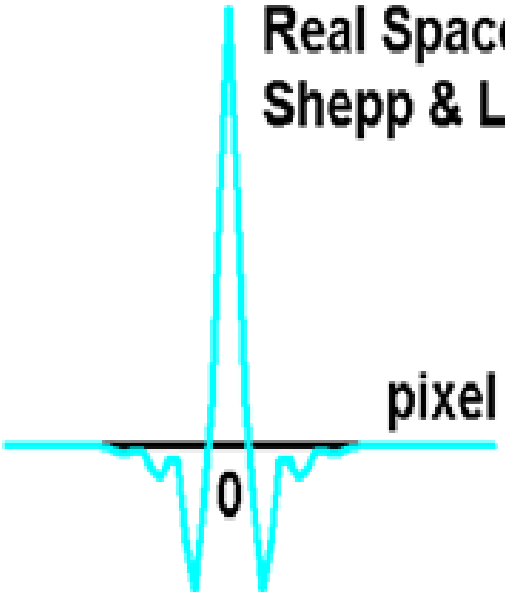
10.　ｈを畳み込んだ２次元透視画像 $\overline{P\theta}$ を単純に重ね合わせた画像　$g = \int \overline{P\theta}\, d\theta$
（Ｆｉｌｔｅｒｅｄ ｂａｃｋ ｐｒｏｊｅｃｔｉｏｎ）は、回転中心部ほど重ね合わせ回数が多くなる誤差が
フィルタｈによって補正され、正しい再構成画像となる。

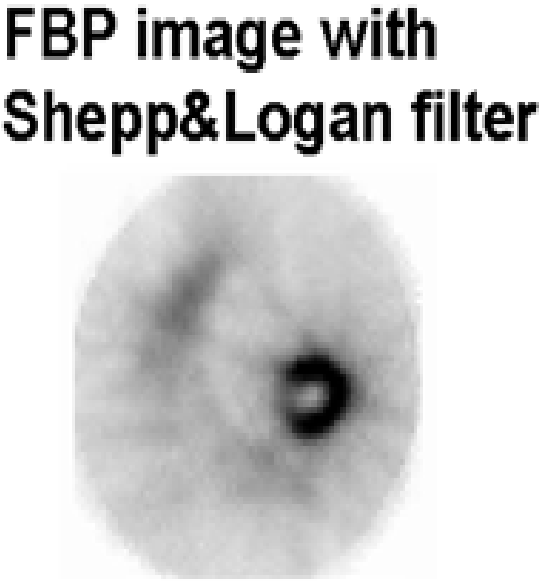


１１．Ｒａｍｐフィルタは理論的には正確な再構成フィルタだが、実際の臨床データに適用すると、
ナイキスト周波数以上の高周波成分を不連続に遮断するために生じる高周波ノイズや
放射状アーチファクトが目立つ場合が多いため、臨床では高周波成分を抑制する工夫を施した
再構成フィルタが用いられている。

１２．ＳＰＥＣＴで用いられる一般的な再構成フィルタとして、Shepp＆Logan フィルタがある。
周波数空間上で、ナイキスト周波数近傍の高周波成分を連続的に減衰させるように設計されており
再構成画像に生じる高周波ノイズや放射状アーチファクトを抑制する効果をもつ。







Convolution　重畳積分

FBP

Filtered Back Projection

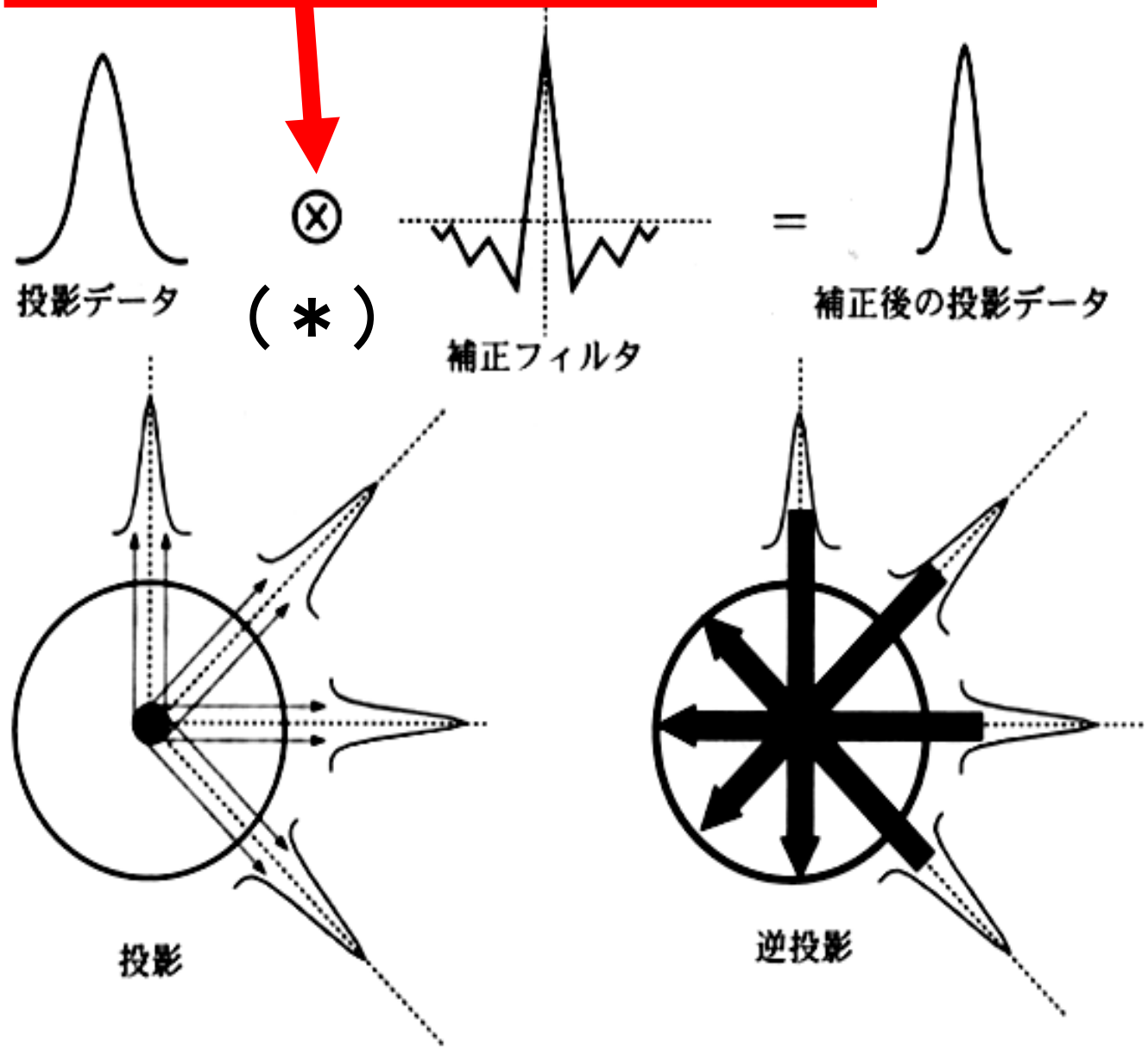
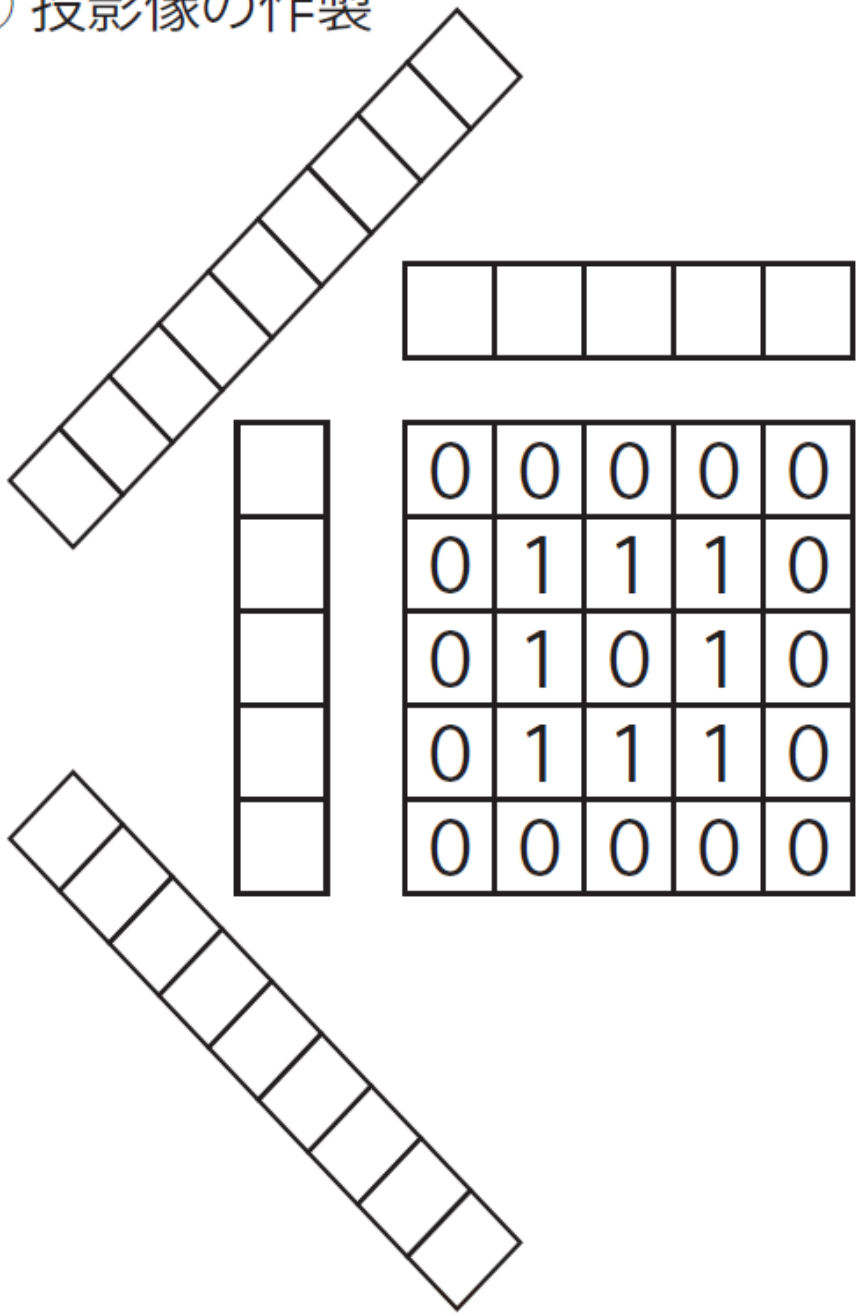


図 4・31　投影データ収集とフィルタ補正逆投影

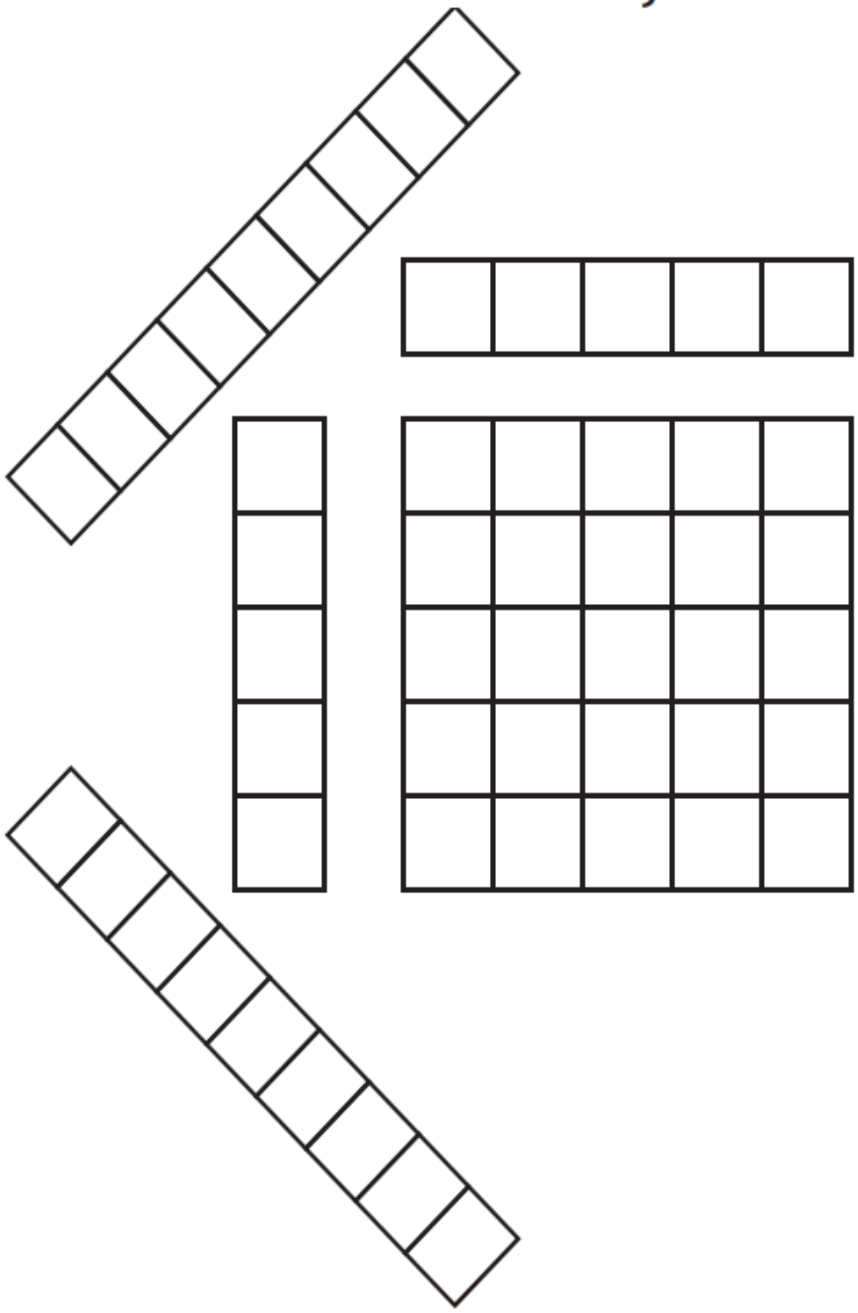
雑音（高周波成分）を除去するフィルタで高周波成分は除去できるが，同時に周辺部やエッジにボケを生ずる．

実空間 Filtered Back Projection

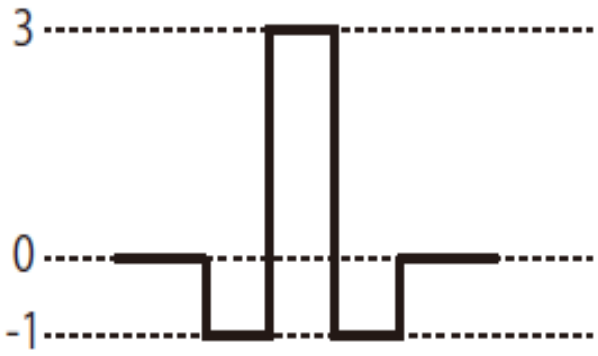
① 投影像の作製

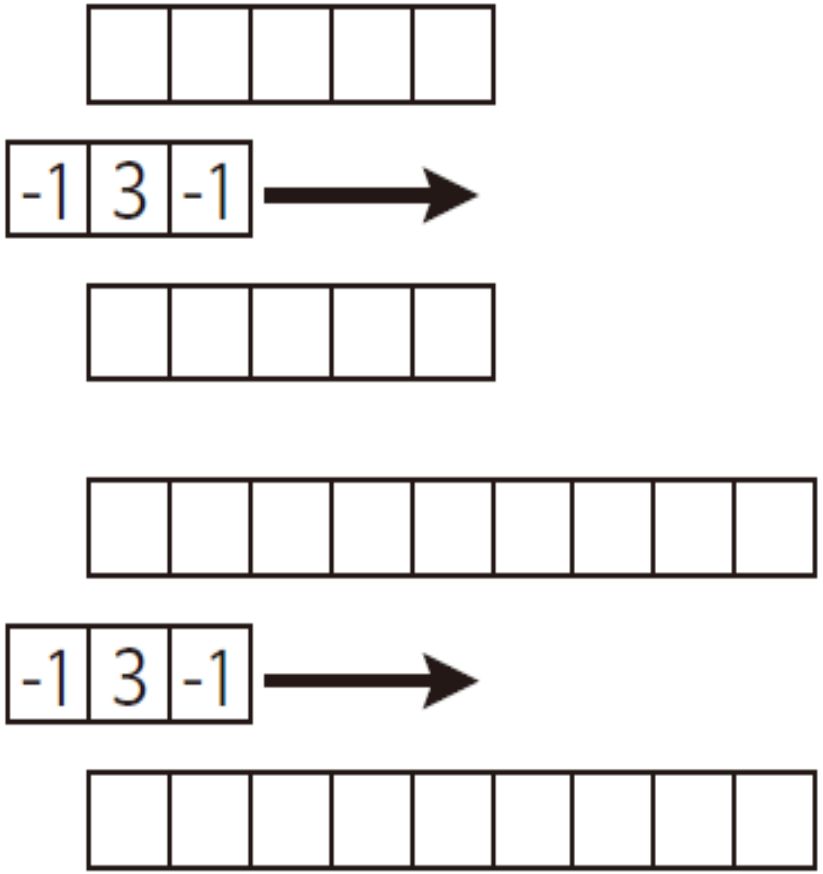


② そのままBack Projection

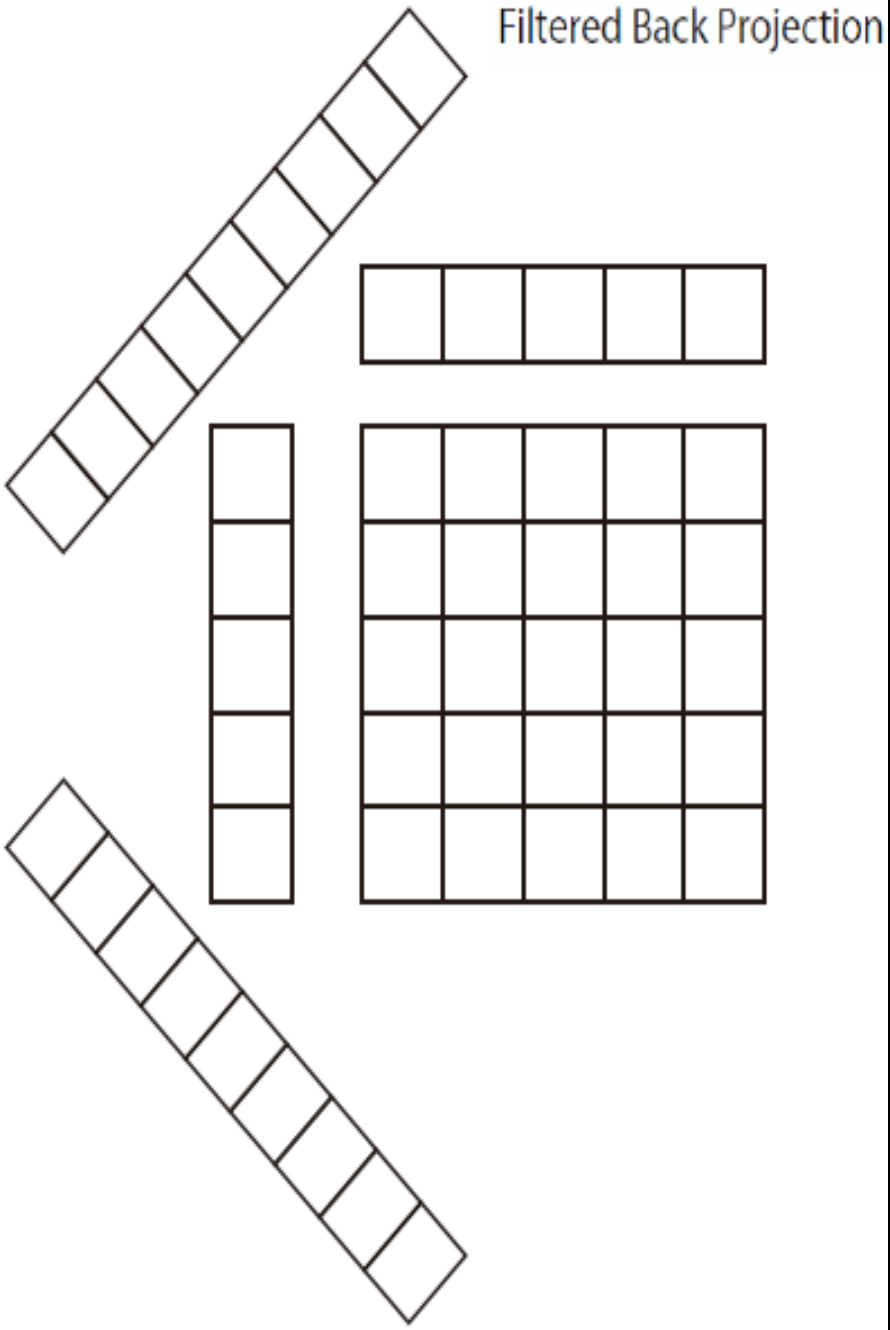


③ フィルター処理

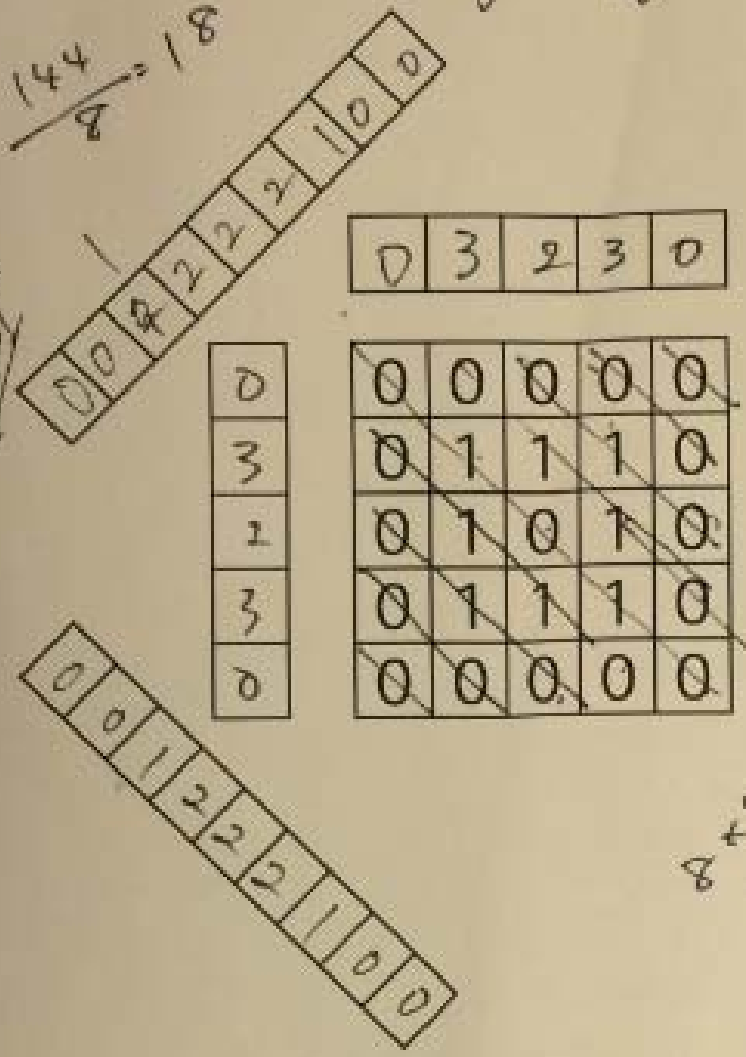




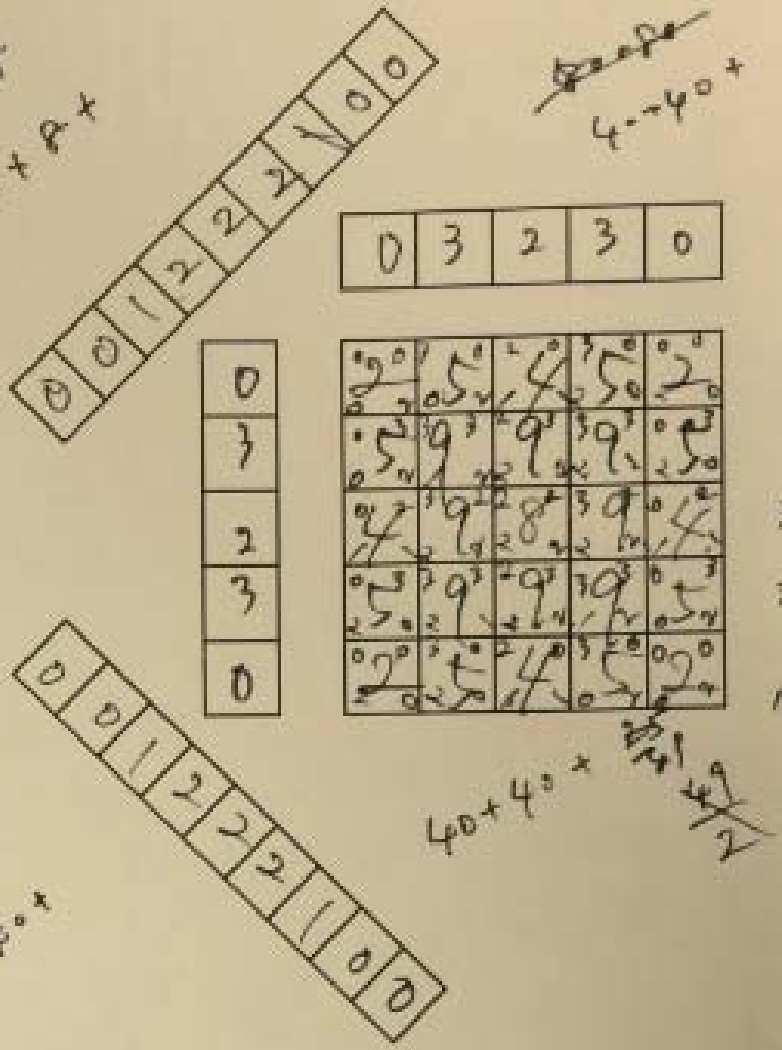
④ フィルター処理後の投影像を畳み込み積分



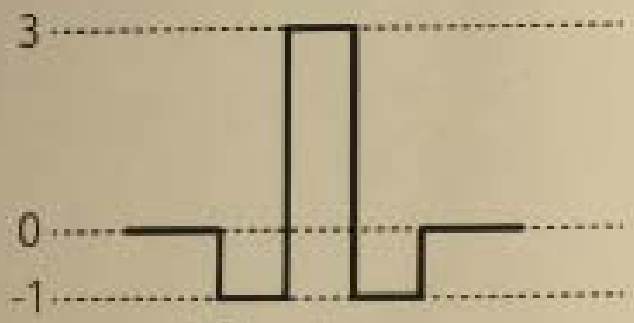
① 投影像の作製

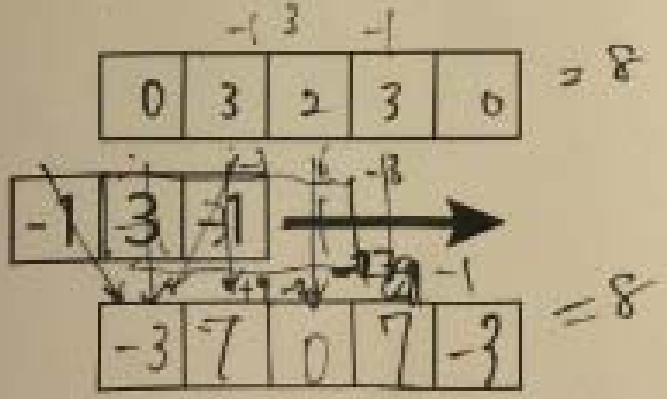


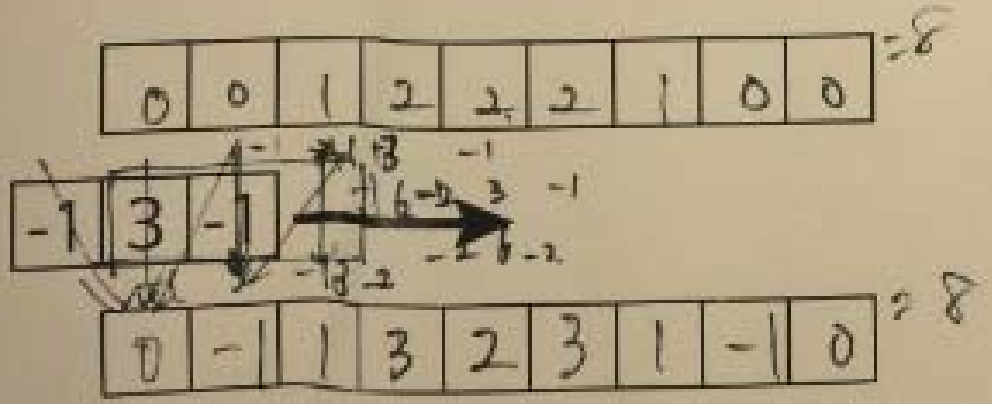
② そのまま畳み込み積分



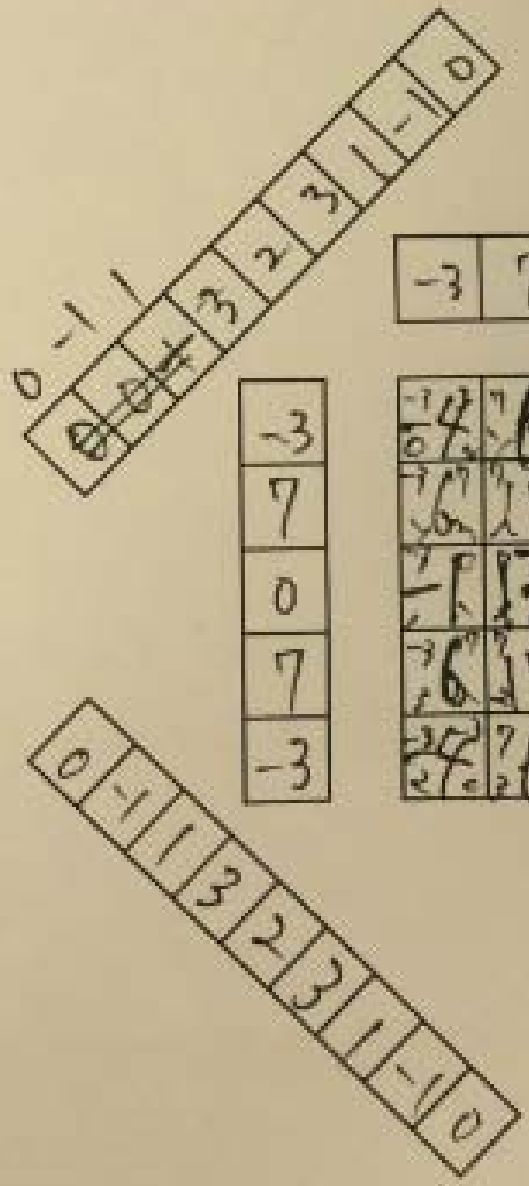
③ フィルター処理







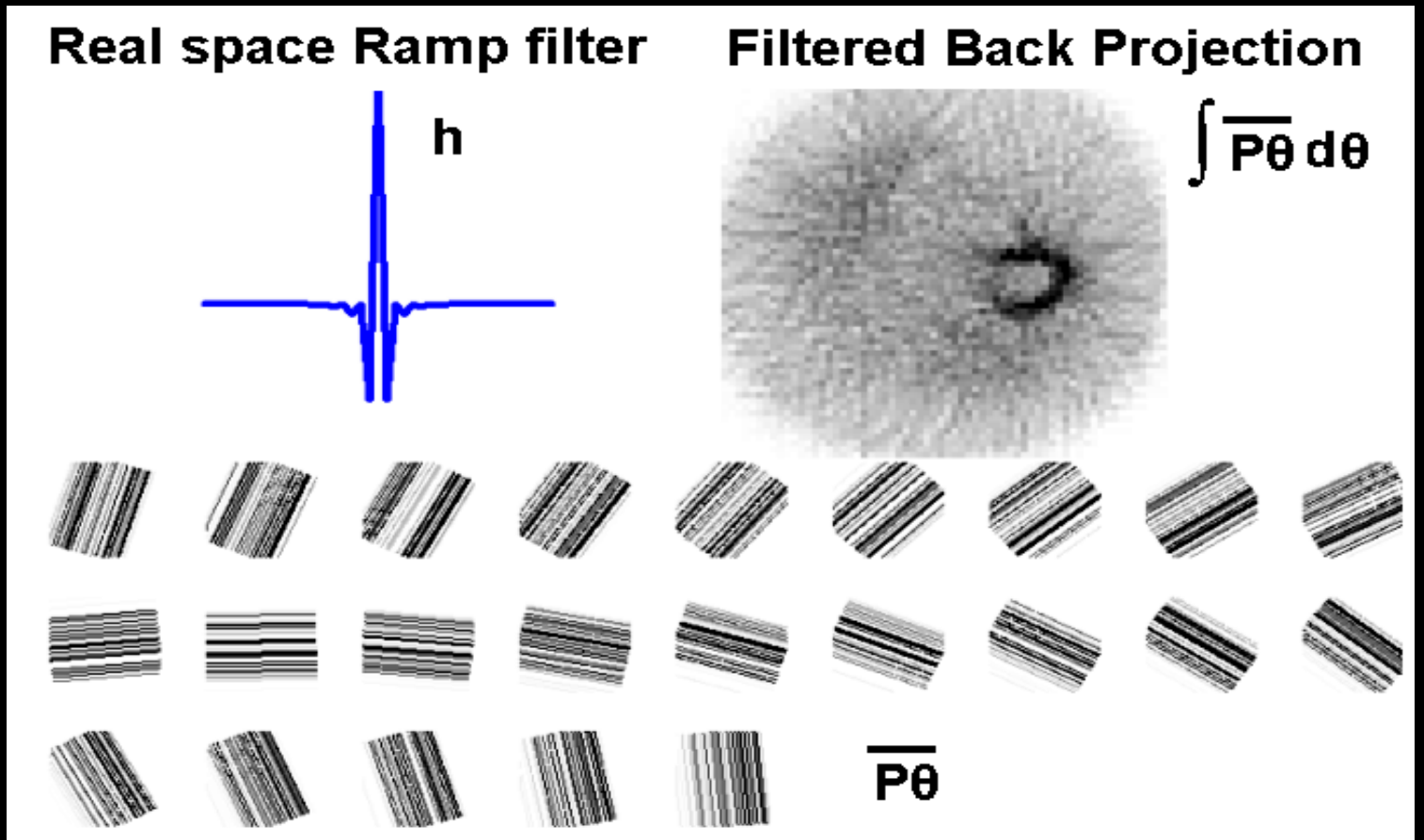
④ フィルター処理後の投影像を畳み込み積分



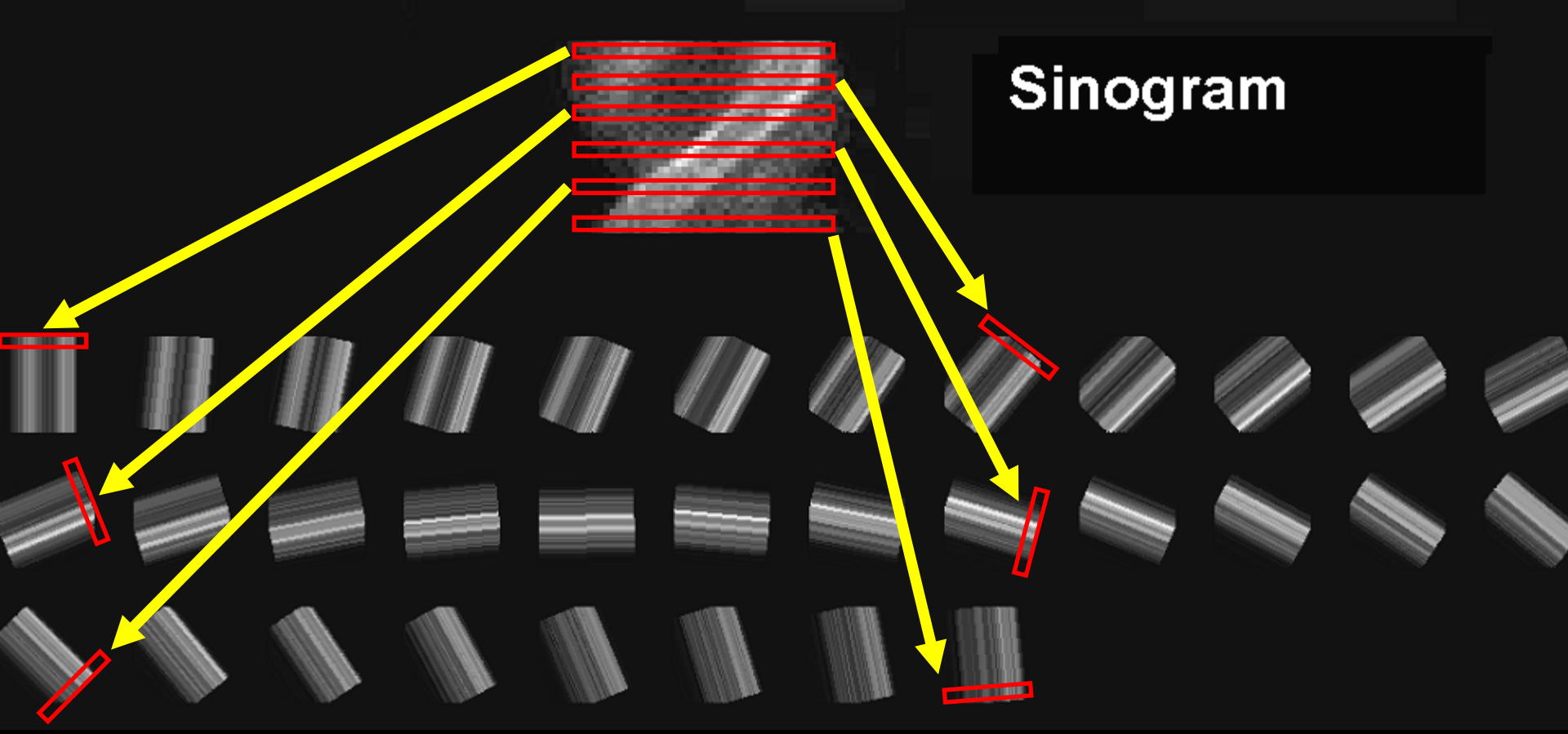
hを畳み込んだ2次元透視画像 $\overline{P\theta}$ を単純に重ね合わせた画像 g は、

$g = \int \overline{P\theta}\, d\theta$　　（ Filtered back projection ）

Filtered back projection は、
回転中心部ほど重ね合わせ回数が多くなる誤差が
フィルタhによって補正され、正しい再構成画像となる。



```
//-------------- Disp Sinogram ----------------------------
for(frame=0; frame<MAXFRAME; frame++){ for(i=0;i<64;i++){

        Sino[ i ][ frame ] = Proj[ i ][ RJ ][ frame ] ;
}}

for(j=0;j<64;j++){ for(i=0;i<64;i++){

        c = Sino[i][j] * 800/max;  if(c>255)c=255;

        for(ii=0;ii<=3;ii++){ for(jj=0;jj<=3;jj++){

            SetColor(RGB(c,c,c)); SetPixel(4*i+ii+260,4*j+jj+140);
        }}
}}
Flush( );   // 画像描画の後に Flush( ); を記述すると描画がスムーズになる。

SetColor(WHITE); FontSize(30);
DrawString(300, 300, "Sinogram" );
```

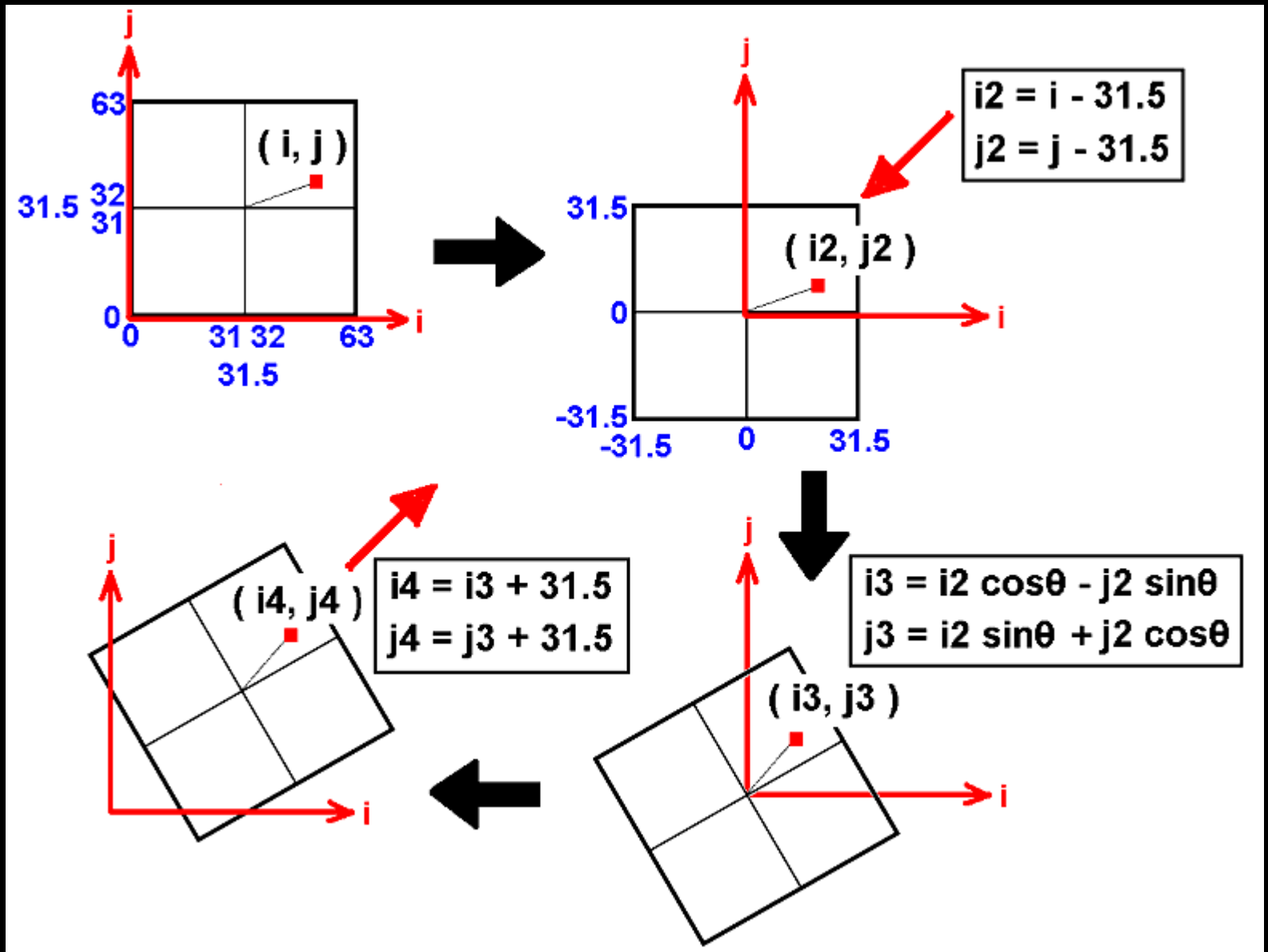


サイノグラムの各スライスの１次元配列は、
３２方向の各々の角度から収集されたデータ。

このデータから、収集された各々の角度に傾いた
２次元透視画像（ Ｐθ ）を作成する。

```
//------------ Generate Pth[ ][ ][ ] (P θ ) data ----------------

PAI = 3.1415926/180.0;        dT = 180.0/(double)MAXFRAME;

for(T=0;T<MAXFRAME;T++){ for(j=0;j<64;j++){ for(i=0;i<64;i++){
   Pth[i][j][T] = -1;
}}}

for(T=0; T<MAXFRAME;T++){

   for(j=0;j<64;j++){for(i=0;i<64;i++){ Simple_Sino[i][j] = Sino[i][T]; }}

   rot = (double)T * dT * PAI ;

//  画像の回転
   for(j=0;j<64;j++){for(i=0;i<64;i++){  I = (double)i ;  J = (double)j ;

     ir = (int)( ( I-31.5 )*cos(rot) - ( J-31.5 )*sin(rot) + 31.5 );
     jr = (int)( ( I-31.5 )*sin(rot) + ( J-31.5 )*cos(rot) + 31.5 );

     if(ir>=0 && ir<64 && jr>=0 && jr<64)Pth[ir][jr][T]=Simple_Sino[i][j];
   }}
}
```

画像の回転は、 画像の中心を原点に平行移動してから回転移動し、再び元の座標に平行移動。

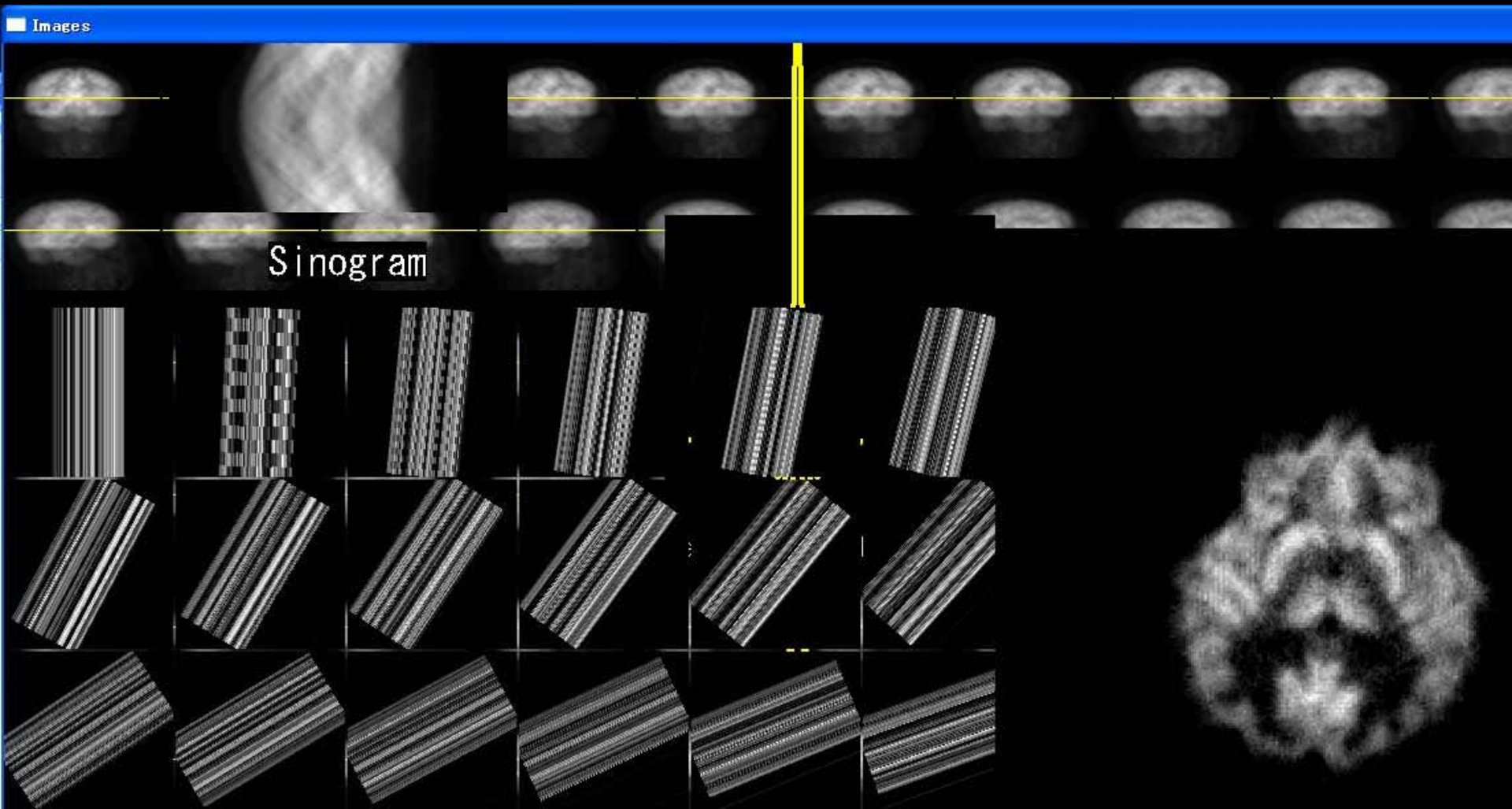


回転後の画像には、画素値が入力されない画素が少数だが発生する。（画素の座標は整数のため）
そのため、配列 Pth[ ][ ] には初めに −1を代入して回転後にも −1であれば画素値が入力されなかったと判断し、その画素の前後の平均値を代入している。

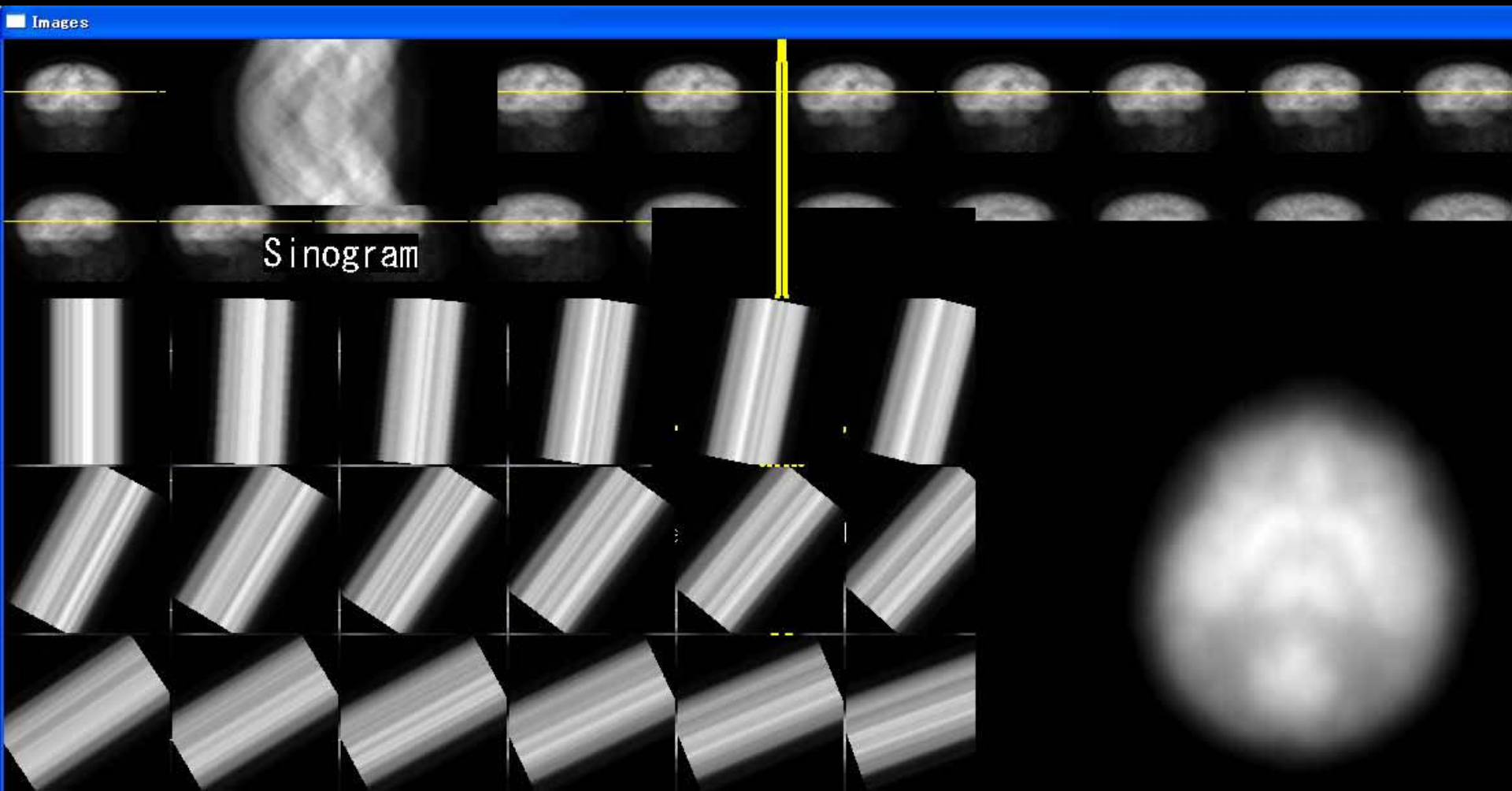
```
for(T=0; T<MAXFRAME;T++){

   for(j=1;j<63;j++){for(i=1;i<63;i++){

     if(Pth[i][j][T]==-1 && Pth[i-1][j][T]!=-1 && Pth[i+1][j][T]!=-1)
     Pth[i][j][T] = (Pth[i-1][j][T]+Pth[i+1][j][T])/2;

     if(Pth[i][j][T]==-1 && Pth[i][j-1][T]!=-1 && Pth[i][j+1][T]!=-1)
     Pth[i][j][T] = (Pth[i][j-1][T]+Pth[i][j+1][T])/2;
  }}

  for(j=0;j<64;j++){for(i=0;i<64;i++){ if(Pth[i][j][T]<0)Pth[i][j][T]=0; }}
}
```

プログラム PETFBP.c

PET画像を Simple BackProjection、または FBP
（ Filtered BackProjection ）で再構成する。

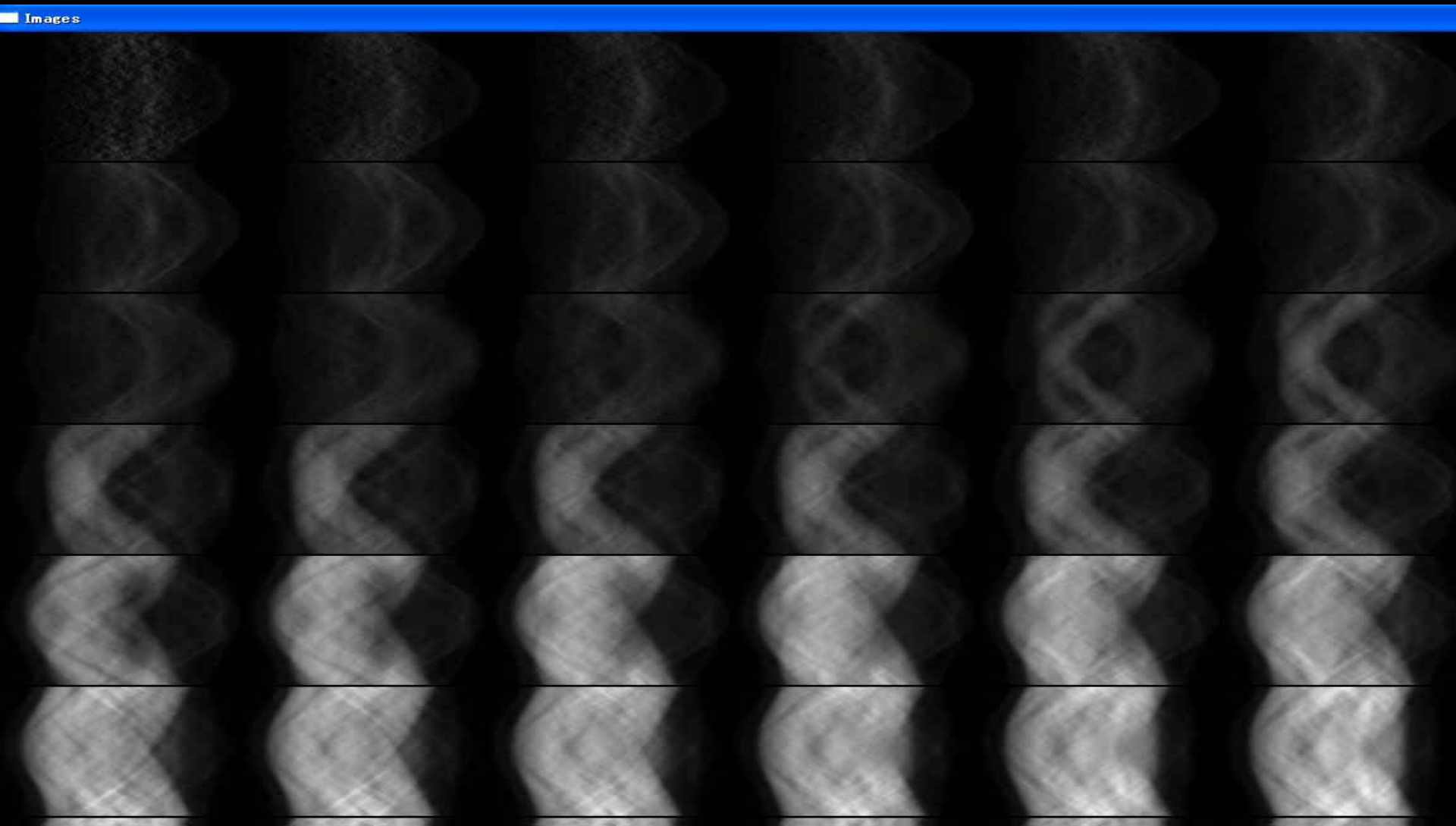


PETFBP.c を実行して、Simple BackProjection と
FBP の 再構成画像の違いを観察し、
Ramp等のフィルタ関数の必要性を理解してください。

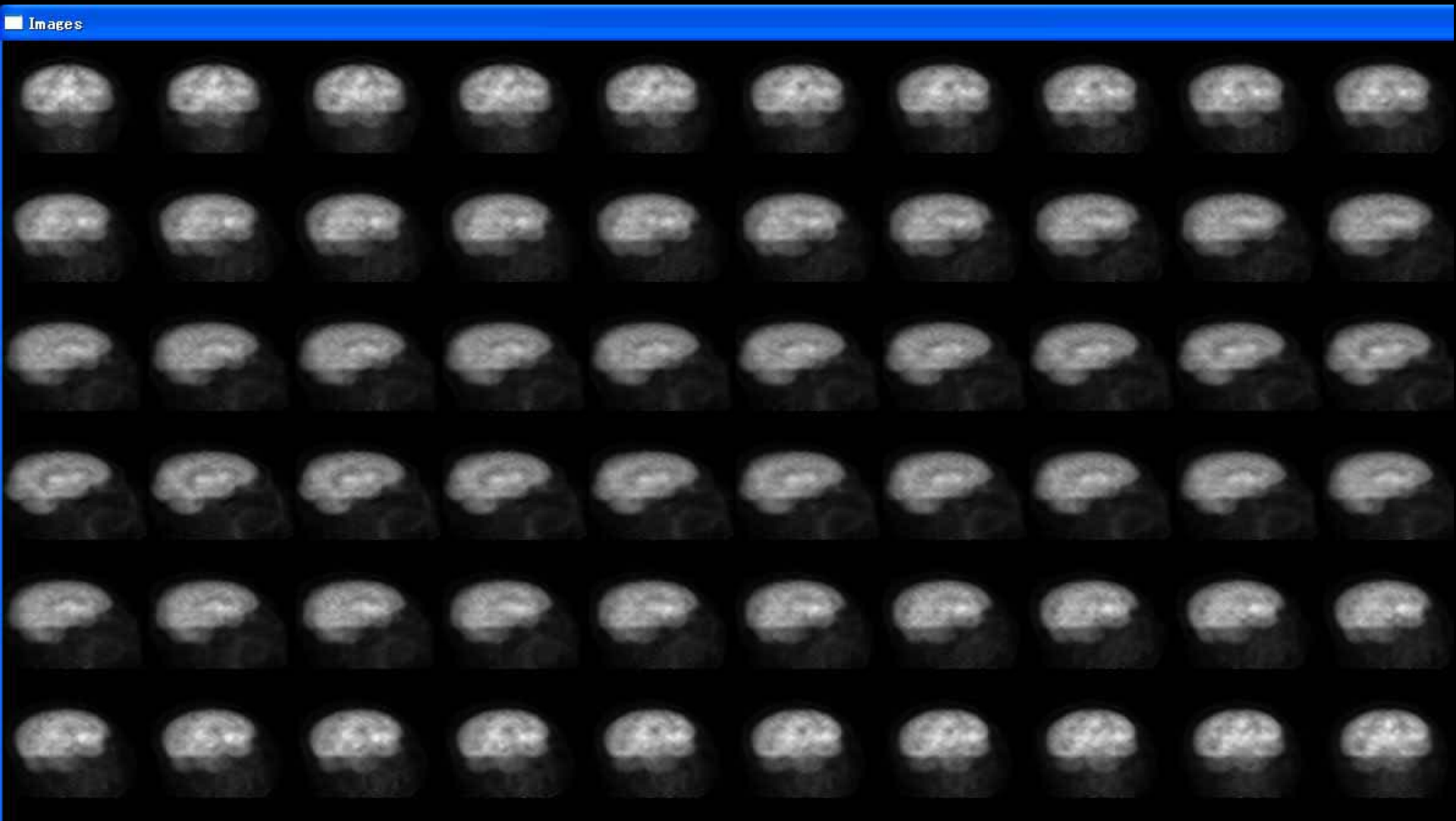


PETsinpgram ファイル

PETの収集データは各スライスのサイノグラムが並んでいる 3次元データ。(CTと同じ)。



各スライスのサイノグラムが並んでいる3次元データを並べ替えれば、SPECTのプロジェクション像と同じ並びになる。 Sinogram[i][θ][j] = Projection [i][j][θ]

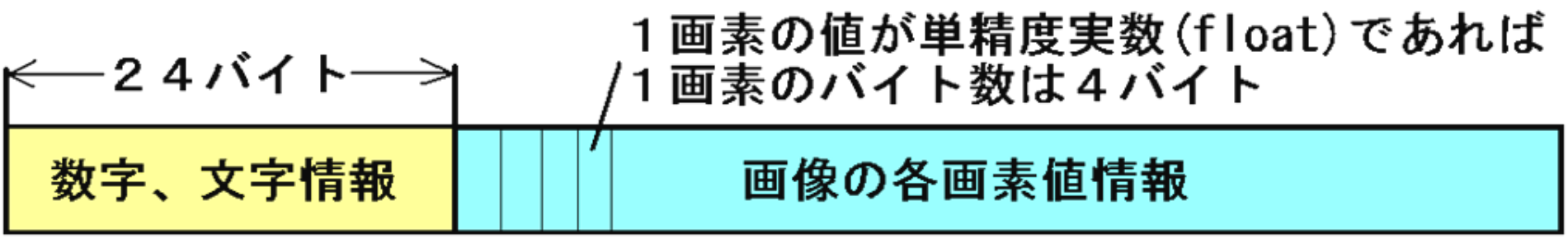


```
//-------------- GET sinogram ---------------------

START:   printf("¥n¥n Load PET Singram data ") ;

GetFileName( f, 0) ;  fp = fopen( f, "rb" ) ;

    OFFSET = 24;   fseek( fp, OFFSET, SEEK_SET ) ;

    for(slice=0;slice<MAXSLICE;slice++){  Event( ) ;

            for(j=0; j<128; j++){ for(i=0; i<256; i++){

            fread( &data, sizeof(float), 1, fp ) ;

            y[ i ][ j ][ slice ] = data ;

    }}} fclose(fp);
```

データファイルのオフセット（ OFFSET ）

DICOMなどの医用画像データファイルは、
画像の画素数や患者名などの数字や文字情報と
画素値情報がつながって入っている。

用意したPETsinogramファイルは、
先頭24バイトには数字や文字情報が入っている。
画像を取り出したい場合は、24バイト読み飛ばす
必要がある。これを OFFSET という。



OFFSET = 24;　fseek(fp, OFFSET, SEEK_SET) ;

OFFSET を 入れる変数は巨大整数を扱える必要があり、倍精度整数で宣言する(long 型)。

fseek 関数は、何バイト目からファイルの読み書きを開始するかを指定する。

fseek(読むファイルのポインタ、OFFSET、SEEK_SET) ;

fread(&data, sizeof(float), 1, fp) ;

PETsinogramファイルの画素は、単精度実数(float型)で書き込まれている。fread関数は、各種の型で読み出せる。

fread(読んだデータを保存する場所(変数のアドレス)、
　　読み込むデータの型 を sizeof(型) と記述、
　　読み込むデータの数　(ここでは1個づつ読む)、
　　読むファイルのポインタ
　)

fseek や fread関数 などの標準C言語関数の文法を調べるオプションが、Visual C++ に装備されている。

たとえば freadの文法を調べたい場合は、fread の5文字のどこかにマウスカーソルをおいて左クリックしてから**ファンクションキー1（F1）**を押すと、Microsoft の 文法解説サイトの fread の説明ページが開く。

Event 関数、Flush 関数（大変重要な 裏関数）

ここでは3次元画像データを読むため for文が3重にループしている。長時間このプログラム実行にWindows OS が占領される可能性がある。 その間はマウスやキーボードなどを操作しても Windows に無視される（ソフトが固まった状態）。この不都合を避けるため、多重ループの中に、時々マウスやキーボードなどの操作状態（ハンドル）を Windows に渡す関数 Event( ) を入れる。 引数は無いので括弧内は空白。

類似した関数として Flush( ) がある。 描画に時間のかかる画像を描くときに、描画が固まったような状態になる場合に SetPixel 等の描画関数を書いた後に入れる。

```
for(slice=0;slice<MAXSLICE;slice++){  Event( ) ;
        for(j=0; j<128; j++){ for(i=0; i<256; i++){
        fread( &data, sizeof(float), 1, fp ) ;
        y[ i ][ j ][ slice ] = data ;
}}} fclose(fp);
```

画像処理プログラム作成に興味を持った人、
重要性に気付いた人、さらに学習したい人は

後期に選択講義　医用画像処理演習　月曜2講目
を受講して下さい。

卒業研究に5名程度、画像処理プログラム作成技術
の習得を目的とした研究テーマを用意するので
是非志願してください。

保健学科大学院には画像処理を研究する講義
機能画像解析学特論が用意されています。

卒業後、画像処理プログラムを学習し直したいと
思ったら、いつでもこの講義ホームページを参照
してください。質問も随時受付けます。