〈第７章〉
トモシンセシスと画像再構成

7

〔第１節〕　トモシンセシスの投影データ

トモシンセシスの計測は，第２章で述べたとおり，X 線源と２次元検出器を反対方向に動かしてデータを取得する．その様子を**図 7-1** に示す．計測の方法は，コーンビームの投影と同じであるが，X 線源と検出器が直線状に動くところが異なっている．投影のプログラムを作成するときは，コーンビーム投影の作成と同じように，被写体の中央に仮想の検出器を考える（**図 7-2**）．仮想検出器は，**図 7-3** に示すように動かす必要がなく，X 線源のみ横方向にスライドさせればよい．

プログラム P7-01tomo3d_para_sampling.c

このプログラムは，３次元被写体に対し，平行ビームでトモシンセシスの投影を作成するものである．投影の角度は，線源が横方向にスライドすることを仮定して設定する．投影は，被写体の３次元画像に対して，投影線上をサンプリングしながら値を合計することで算出する．基本的には，プログラム「P4-08proj3d_sampling.c」をベースに作成している．

投影角度を算出するために，原点から仮定した面線源までの最小距離とその面線源が横方向に移動する距離を入力する．距離の単位はピクセル（画素）とする．面線源は，投影の数だけ等間隔に移動し，最終的に移動量が入力した距離になるようにする．

【コード】P7-01tomo3d_para_sampling.c　（main 関数）

```
printf(" 原点から面線源までの最小距離（ピクセル）: ");
scanf("%lf", &g_d);

printf(" 面線源が横方向に移動する距離（ピクセル）: ");
scanf("%lf", &g_sd);
```

図 7-4 に示すように投影線の単位方向ベクトル（a_x, a_y, a_z）は，面線源の x 方向の移動量 dx と面線源までの距離 D を使って表すと，

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{-1}{\sqrt{dx^2 + D^2}} \begin{pmatrix} dx \\ D \\ 0 \end{pmatrix} \tag{7-1}$$

となる．ここで $\sqrt{dx^2 + D^2}$ は，線源の移動量の位置から仮想検出器の原点までの距離である．三平方の定理から求められる．

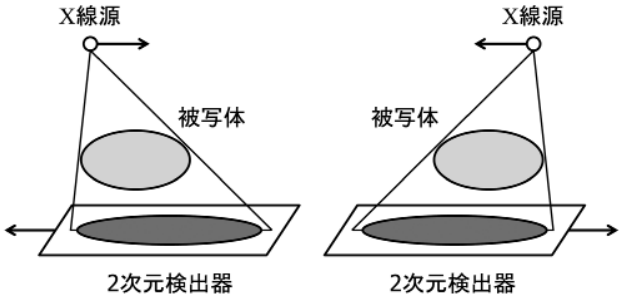

図 7-1　トモシンセシスの X 線源と 2 次元検出器の動き
X 線源と 2 次元検出器は逆方向に動く.

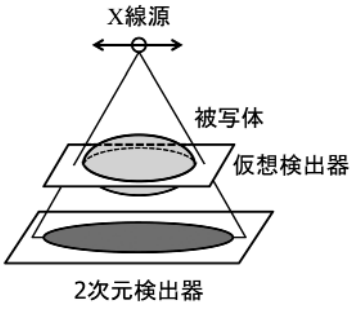

図 7-2　2 次元検出器と被写体を横切る仮想検出器

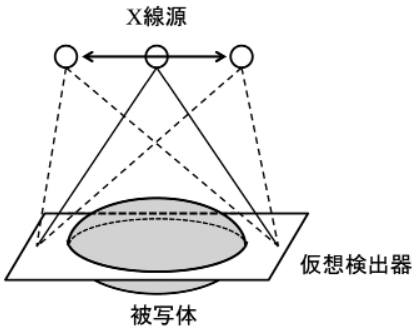

図 7-3　X 線源と仮想検出器の動き
仮想検出器は動かさずに，X 線源のみを横方向にスライドする.

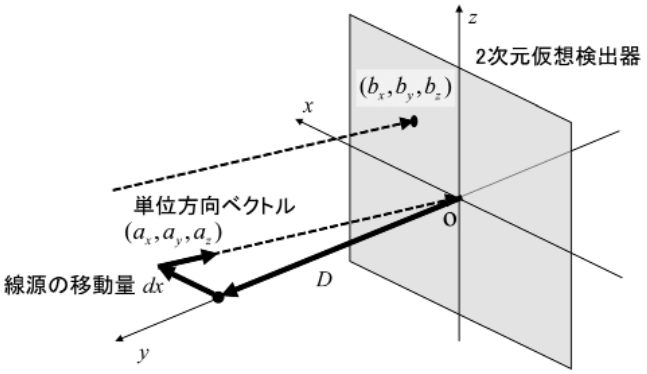

図 7-4　平行ビームのトモシンセシス投影の幾何学配置

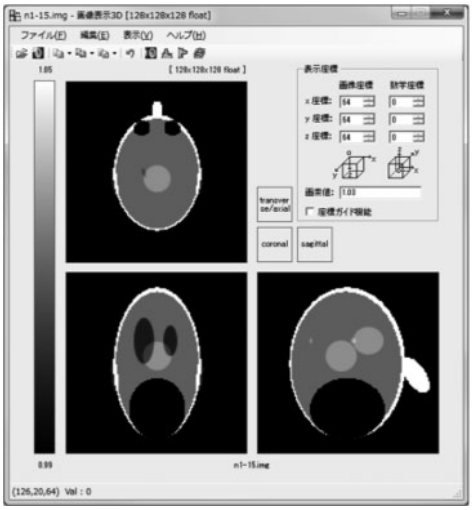

図 7-5　プログラム「P1-15shepp3d_divide.c」で作成した 128×128×128 画素の 3 次元 Shepp-Logan ファントム

【コード】P7-01tomo3d_para_sampling.c　（tomo_3D_para_sampling 関数）

```
dx = (i-PN/2)*g_sd/PN; // 面線源の x 方向の移動距離

// 平行ビームの単位方向ベクトル
ax = -dx/sqrt(dx*dx+g_d*g_d);
ay = -g_d/sqrt(dx*dx+g_d*g_d);
az = 0;
```

直線上の 1 点（b_x, b_y, b_z）は，2 次元検出器上の点を指定する．

【コード】P7-01tomo3d_para_sampling.c　（tomo_3D_para_sampling 関数）

```
// 投影位置の固定座標
bx = x;
by = 0;
bz = z;
```

その他のコードは，プログラム「P4-08proj3d_sampling.c」の proj_3D_sampling 関数と同じである．

【実行例 7-01】

P1-15shepp3d_divide.c で作成した 128×128×128 画素の 3 次元 Shepp-Logan ファントムを読み出し，平行ビームのトモシンセシス投影を作成する．

① P1-15shepp3d_divide.c で「n1-15.img」を作成（**図 7-5**）．

② P7-01tomo3d_para_sampling を実行し，**図 7-6** に示すように入力．

③ 画像「n7-01.prj」の表示（**図 7-7**）．

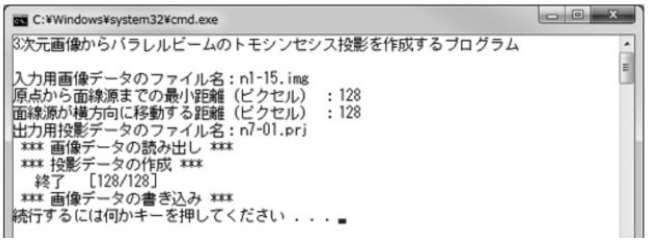

図 7-6　プログラム「P7-01tomo3d_para_sampling.c」の実行例

図 7-7　平行ビームのトモシンセシス投影

プログラム P7-04tomo3d_cone_sampling.c

このプログラムは，3 次元被写体に対し，コーンビームでトモシンセシスの投影を作成するものである．投影は，点線源が横方向にスライドすることを仮定して設定する．基本的には，プログラム「P4-12proj3d_cone_sampling.c」をベースに作成している．線源位置（s_x, s_y, s_z）と直線の単位方向ベクトル（a_x, a_y, a_z）と直線上の 1 点の座標（b_x, b_y, b_z）を**図 7-8** に示す．線源位置（s_x, s_y, s_z）は，

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = \begin{pmatrix} dx \\ D \\ 0 \end{pmatrix} \tag{7-2}$$

となる．

【コード】P7-04tomo3d_cone_sampling.c　（tomo_3D_cone_sampling 関数）

```
dx = (i-PN/2)*g_sd/PN; // 線源の x 方向の位置


// 線源の固定座標
sx = dx;
sy = g_d;
sz = 0;
```

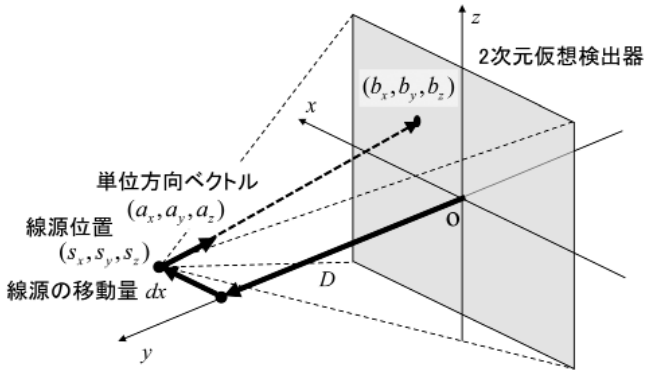

図 7-8　コーンビームのトモシンセシス投影の幾何学配置

直線上の 1 点（b_x, b_y, b_z）は，2 次元検出器上の点を指定する．

【コード】P7-04tomo3d_cone_sampling.c　（tomo_3D_cone_sampling 関数）

```
// 投影位置の固定座標
bx = x;
by = 0;
bz = z;
```

直線の単位方向ベクトル（a_x, a_y, a_z）は，線源の固定座標（s_x, s_y, s_z）と直線上の 1 点（b_x, b_y, b_z）から

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{1}{\sqrt{D^2 + x^2 + z^2}} \begin{pmatrix} b_x - s_x \\ b_y - s_y \\ b_z - s_z \end{pmatrix} \tag{7-3}$$

となる．ここで $\sqrt{D^2 + x^2 + z^2}$ は，線源の固定座標（s_x, s_y, s_z）と直線上の 1 点（b_x, b_y, b_z）の距離である．これも，三平方の定理から求められる．

【コード】P7-04tomo3d_cone_sampling.c　（tomo_3D_cone_sampling 関数）

```
// 線源から投影位置までの単位方向ベクトル
ax = (bx-sx)/sqrt(g_d*g_d+x*x+z*z);
ay = (by-sy)/sqrt(g_d*g_d+x*x+z*z);
az = (bz-sz)/sqrt(g_d*g_d+x*x+z*z);
```

その他のコードは，プログラム「P4-12proj3d_cone_sampling.c」の proj_3D_cone_sampling 関数と同じである．

【実行例 7-04】

P1-15shepp3d_divide.c で作成した 128×128×128 画素の 3 次元 Shepp-Logan ファントムを読み出し，コーンビームのトモシンセシス投影を作成する．

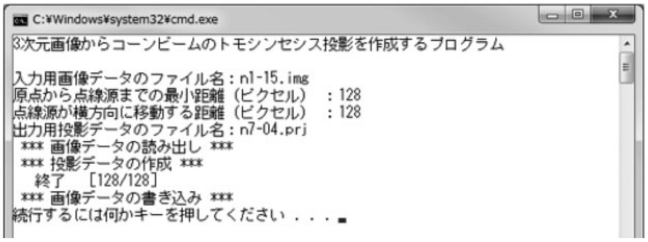

図 7-9　プログラム「P7-04tomo3d_cone_sampling.c」の実行例

図 7-10　コーンビームのトモシンセシス投影

① P1-15shepp3d_divide.c で「n1-15.img」を作成（**図 7-5**）.
② P7-04tomo3d_cone_sampling を実行し，**図 7-9** に示すように入力.
③ 画像「n7-04.prj」の表示（**図 7-10**）.

〔第 2 節〕　トモシンセシスの画像再構成

トモシンセシスの原理に基づく再構成法は，線源位置を移動して取得した投影データを，ずらして足し合わせることである．その場合，コーンビームのトモシンセシス投影では，深さ方向に拡大率が変わるので，深さ方向に大きさの異なる再構成画像が作られる．投影線を考慮して逆投影を行う方法では，深さ方向に大きさを合わせて再構成できる．また，画像再構成フィルタを利用すれば，ある程度シャープに再構成される．

プログラム P7-07tomo3d_recon_shift.c

このプログラムは，トモシンセシス投影から投影データをシフトして加えることで画像再構成するシフト法のプログラムである．**図 7-11** に示すように，y 軸上の y の位置にある焦点面に対して，線源の移動量 dx に対する仮想検出器における移動量を ex とすると，三角形の比の関係より

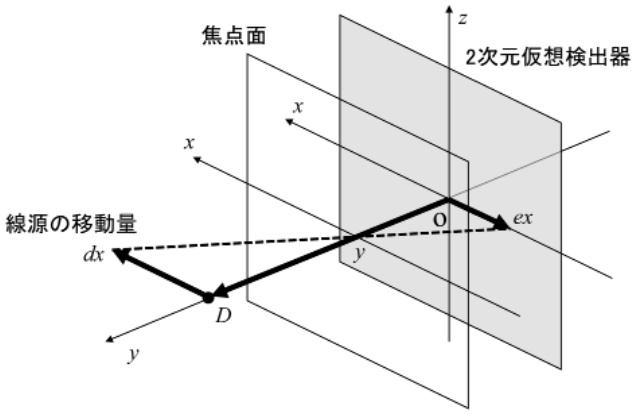

図 7-11　トモシンセシス投影とシフト再構成法のシフト量の関係

$$ex = \frac{y}{D-y}dx \tag{7-4}$$

と表される．

【コード】P7-07tomo3d_recon_shift.c　（tomo_3D_recon_shift 関数）

```
// 仮想検出器における x 方向の移動量
ex = y/(g_d-y)*dx
```

移動量 *ex* だけ *x* 方向にシフトしているので，その分を差し引いて投影全体を移動させる．移動については 1 次元の線形補間で移動位置の値を算出する．

【コード】P7-07tomo3d_recon_shift.c　（tomo_3D_recon_shift 関数）

```
x = k-ex;  // 移動量を補正した x 方向の位置
```

移動量 *ex* を補正した投影を全て足し合わせることで，焦点面に焦点を合わせた画像が再構成される．

【コード】P7-07tomo3d_recon_shift.c　（tomo_3D_recon_shift 関数）

```
// 移動量を補正した投影を足し合わせる
for(i = 0 ; i < NZ ; i++)
        for(k = 0 ; k < NX ; k++)
                for(m = 0 ; m < PN ; m++)
                        g_img[i][j][k] += g_sft[m][i][k];
```

【実行例 7-07-1】

P7-01tomo3d_para_sampling.c で作成した 128×128×128 画素の平行ビームのトモシンセシス投影を読み出し，シフト法で再構成した画像を作成する．

① P7-01tomo3d_para_sampling.c で「n7-01.prj」を作成（**図 7-7**）．
② P7-07tomo3d_recon_shift を実行し，**図 7-12** に示すように入力．
③ 画像「n7-07-1.img」の表示（**図 7-13**）．