

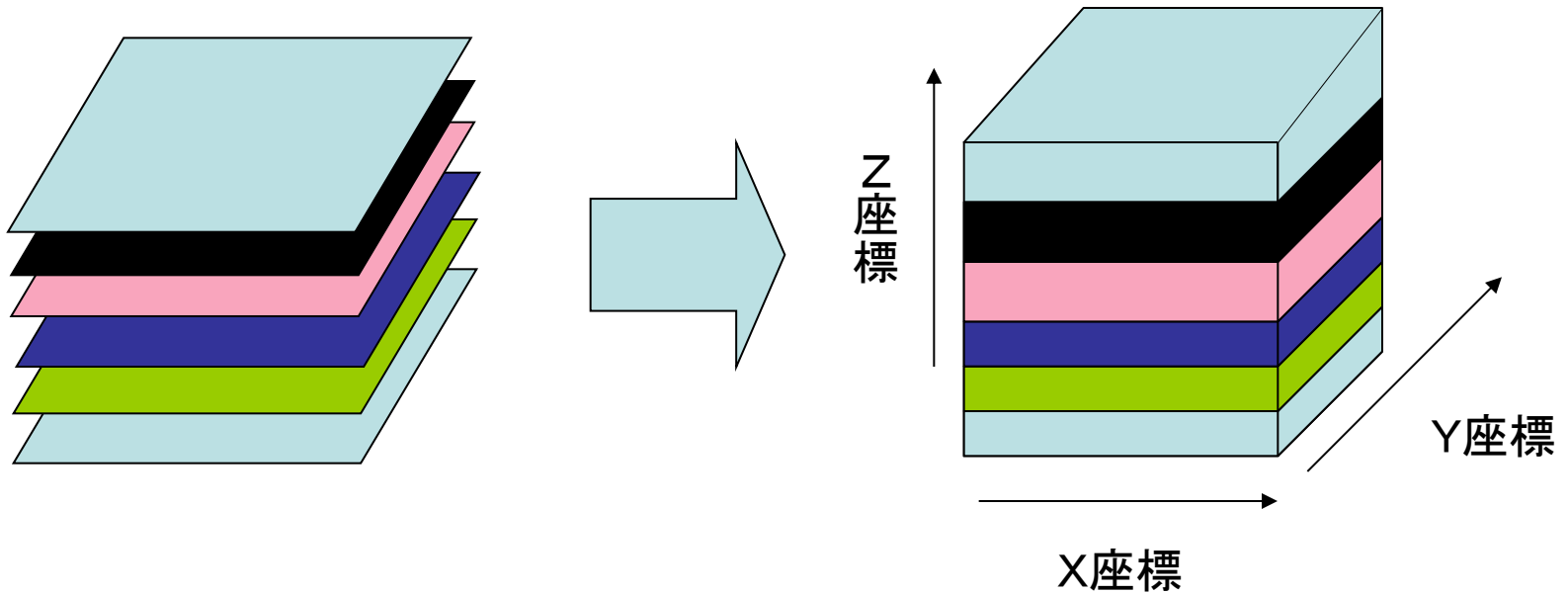
3次元画像処理について

内容

- 3次元画像の扱い
- プログラム中での扱い
- 実際のプログラム(dilation,erosion)

3次元画像の扱い

- CT, MRIのように2次元画像から3次元画像を作るには、2次元画像を何枚も重ねて厚さを持たせることによって3次元画像とする



ファイルへの保存(2次元画像)

(x1,y1) (x2,y1) (x3,y1) ... (x256,y1)

(x1,y2) (x2,y2) (x3,y2) ... (x256,y2)

▪
▪
▪

(x1,y256) (x2,y256) (x3,y256) ... (x256,y256)

ファイルへの保存(3次元画像)

(x1,y1,z1) (x2,y1,z1) (x3,y1,z1) ... (x256,y1,z1)

(x1,y2,z1) (x2,y2,z1) (x3,y2,z1) ... (x256,y2,z1)

⋮

⋮

⋮

(x1,y256,z1) (x2,y256,z1) (x3,y256,z1) ... (x256,y256,z1)

(x1,y1,z2) (x2,y1,z2) (x3,y1,z2) ... (x256,y1,z2)

(x1,y2,z2) (x2,y2,z2) (x3,y2,z2) ... (x256,y2,z2)

⋮

⋮

⋮

(x1,y256,z2) (x2,y256,z2) (x3,y256,z2) ... (x256,y256,z2)

⋮

⋮

⋮

(x1,y1,z256) (x2,y1,z256) (x3,y1,z256) ... (x256,y1,z256)

(x1,y2,z256) (x2,y2,z256) (x3,y2,z256) ... (x256,y2,z256)

⋮

⋮

⋮

(x1,y256,z256) (x2,y256,z256) (x3,y256,z256) ... (x256,y256,z256)

プログラム中での扱い

- 画像データは配列を用いて扱う
- 2次元画像
 - Image[Y_SIZE][X_SIZE]のように2次元配列にする
 - Image[34][65]には(65,34)の値が入っている
- 3次元画像
 - Image[Z_SIZE][Y_SIZE][X_SIZE]のように3次元配列にする
 - Image[44][67][32]には(32,67,44)の値が入っている

プログラム中での扱い

- カラー画像の場合
- `Image[3][Z_SIZE][Y_SIZE][X_SIZE]`のように4次元配列にする
 - `Image[0][z][y][x]`にはRの値
 - `Image[1][z][y][x]`にはGの値
 - `Image[2][z][y][x]`にはBの値

Dilationのプログラム

```
void dilation_3D(unsigned char image_in[Z_SIZE][Y_SIZE][X_SIZE],
                unsigned char image_out[Z_SIZE][Y_SIZE][X_SIZE],)
{
    int h, i, j;
    int hp, hm, ip, im, jp, jm;

    for (h = 0; h < Z_SIZE; h++){
        for (i = 0; i < Y_SIZE; i++){
            for (j = 0; j < X_SIZE; j++){
                hm = h - 1;
                hp = h + 1;
                im = i - 1;
                ip = i + 1;
                jm = j - 1;
                jp = j + 1;
                if (hm < 0) hm = 0;
                if (hp > Z_SIZE - 1) hp = Z_SIZE - 1;
                if (im < 0) im = 0;
                if (ip > Y_SIZE - 1) ip = Y_SIZE - 1;
                if (jm < 0) jm = 0;
                if (jp > X_SIZE - 1) jp = X_SIZE - 1;
```

対象画素の前後
の座標を計算


```
if (image_in[h][i][j] == 0 &&  
    image_in[h][i][jp] == 0 &&  
    image_in[h][i][jm] == 0 &&  
    image_in[h][ip][j] == 0 &&  
    image_in[h][ip][jp] == 0 &&  
    image_in[h][ip][jm] == 0 &&  
    image_in[h][im][j] == 0 &&  
    image_in[h][im][jp] == 0 &&  
    image_in[h][im][jm] == 0 &&  
    image_in[hp][i][j] == 0 &&  
    image_in[hp][i][jp] == 0 &&  
    image_in[hp][i][jm] == 0 &&  
    image_in[hp][ip][j] == 0 &&  
    image_in[hp][ip][jp] == 0 &&  
    image_in[hp][ip][jm] == 0 &&  
    image_in[hp][im][j] == 0 &&  
    image_in[hp][im][jp] == 0 &&  
    image_in[hp][im][jm] == 0 &&  
    image_in[hm][i][j] == 0 &&  
    image_in[hm][i][jp] == 0 &&  
    image_in[hm][i][jm] == 0 &&  
    image_in[hm][ip][j] == 0 &&  
    image_in[hm][ip][jp] == 0 &&  
    image_in[hm][ip][jm] == 0 &&  
    image_in[hm][im][j] == 0 &&  
    image_in[hm][im][jp] == 0 &&  
    image_in[hm][im][jm] == 0){  
    image_out[h][i][j] = 0;
```

対象画素の周囲が
すべて0なら0

```
}  
else{  
    image_out[h][i][j] = 1;
```

対象画素の周囲に1が
あれば1

```
}
```

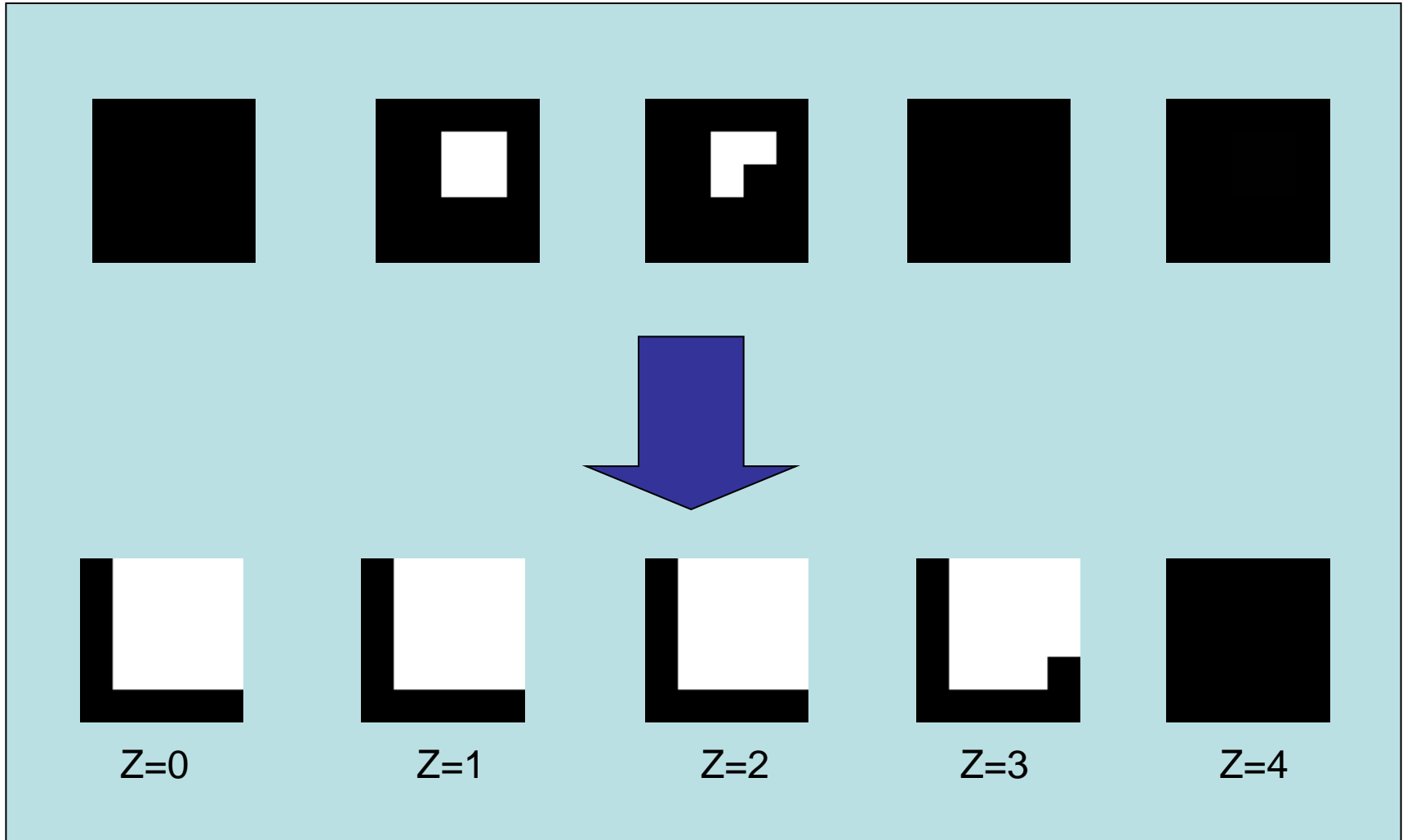
```
}
```

```
}
```

```
}
```

```
}
```

Dilationの結果



Erosionのプログラム

```
void erosion_3D(unsigned char image_in[Z_SIZE][Y_SIZE][X_SIZE],
               unsigned char image_out[Z_SIZE][Y_SIZE][X_SIZE],)
{
    int h, i, j;
    int hp, hm, ip, im, jp, jm;

    for (h = 0; h < Z_SIZE; h++){
        for (i = 0; i < Y_SIZE; i++){
            for (j = 0; j < X_SIZE; j++){
                hm = h - 1;
                hp = h + 1;
                im = i - 1;
                ip = i + 1;
                jm = j - 1;
                jp = j + 1;
                if (hm < 0) hm = 0;
                if (hp > Z_SIZE - 1) hp = Z_SIZE - 1;
                if (im < 0) im = 0;
                if (ip > Y_SIZE - 1) ip = Y_SIZE - 1;
                if (jm < 0) jm = 0;
                if (jp > X_SIZE - 1) jp = X_SIZE - 1;
```

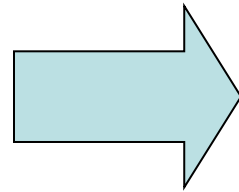
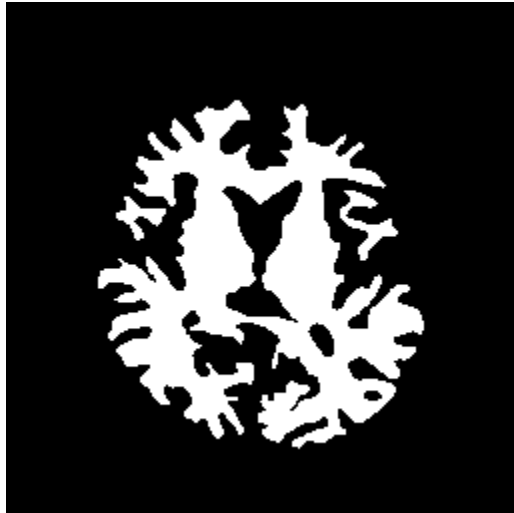
対象画素の前後
の座標を計算

```
if (image_in[h][i][j] == 1 &&
    image_in[h][i][jp] == 1 &&
    image_in[h][i][jm] == 1 &&
    image_in[h][ip][j] == 1 &&
    image_in[h][ip][jp] == 1 &&
    image_in[h][ip][jm] == 1 &&
    image_in[h][im][j] == 1 &&
    image_in[h][im][jp] == 1 &&
    image_in[h][im][jm] == 1 &&
    image_in[hp][i][j] == 1 &&
    image_in[hp][i][jp] == 1 &&
    image_in[hp][i][jm] == 1 &&
    image_in[hp][ip][j] == 1 &&
    image_in[hp][ip][jp] == 1 &&
    image_in[hp][ip][jm] == 1 &&
    image_in[hp][im][j] == 1 &&
    image_in[hp][im][jp] == 1 &&
    image_in[hp][im][jm] == 1 &&
    image_in[hm][i][j] == 1 &&
    image_in[hm][i][jp] == 1 &&
    image_in[hm][i][jm] == 1 &&
    image_in[hm][ip][j] == 1 &&
    image_in[hm][ip][jp] == 1 &&
    image_in[hm][ip][jm] == 1 &&
    image_in[hm][im][j] == 1 &&
    image_in[hm][im][jp] == 1 &&
    image_in[hm][im][jm] == 1){
    image_out[h][i][j] = 1;
}
else{
    image_out[h][i][j] = 0;
}
}
```

対象画素の周囲が
すべて1なら1

対象画素の周囲に0が
あれば0

Erosionの結果



Erosionを3回実行

