平滑化，微分フィルタ

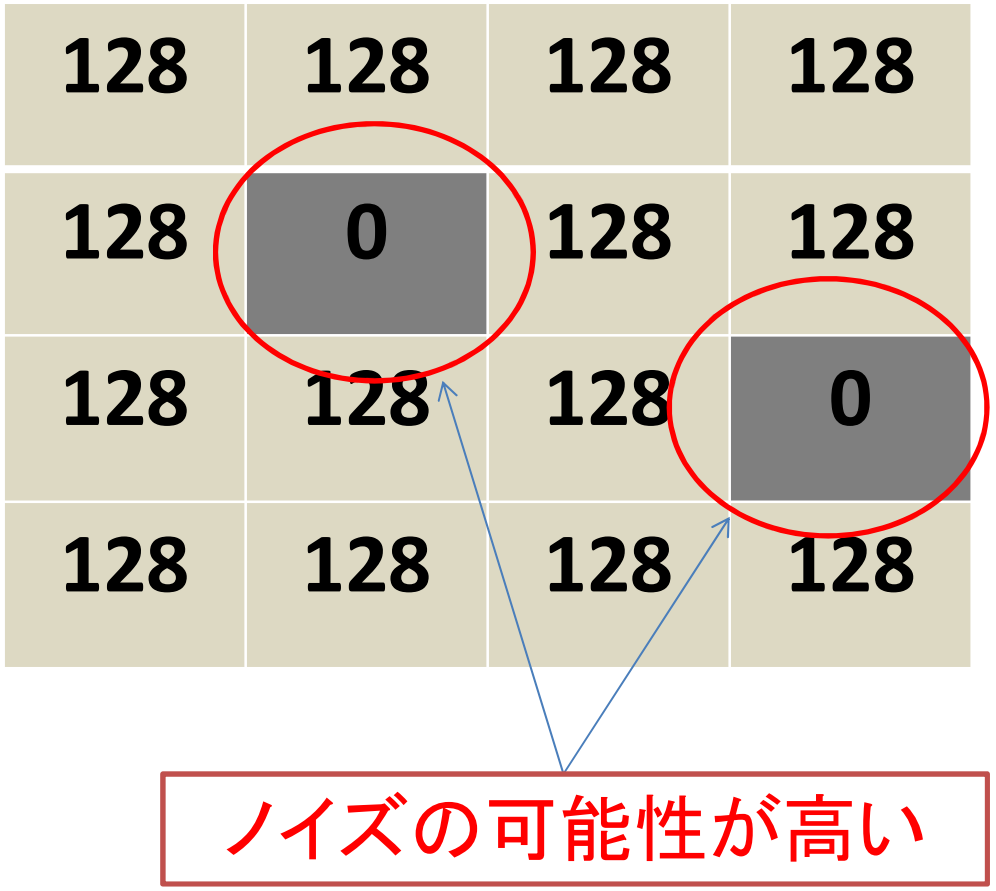
画像に含まれるノイズ

サンプル画像

128	128	128	128
128	0	128	128
128	128	128	0
128	128	128	128

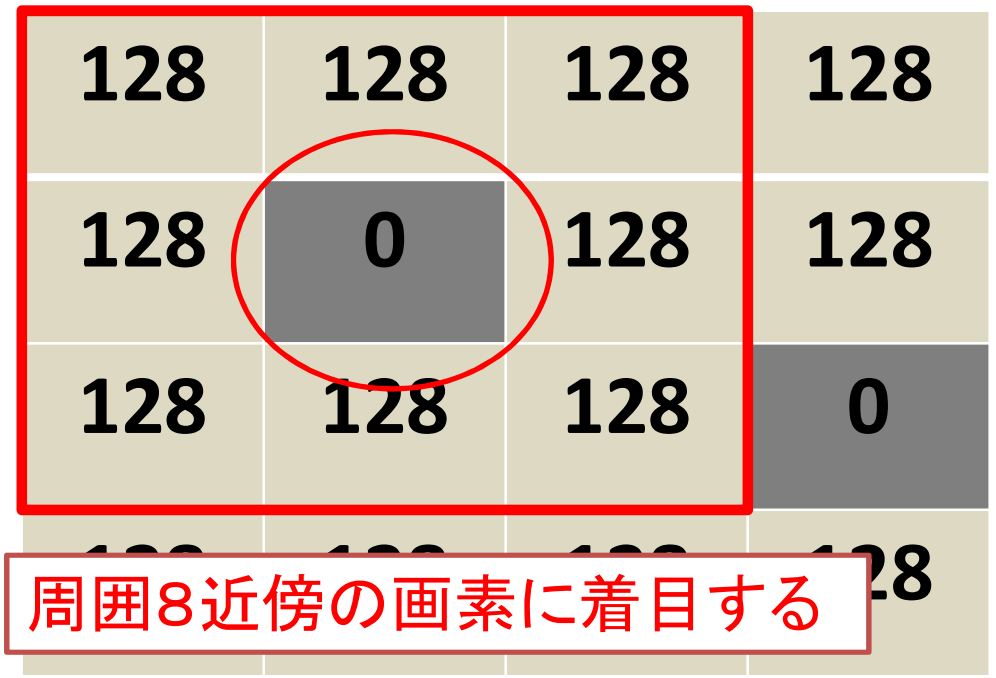
画像に含まれるノイズ

サンプル画像



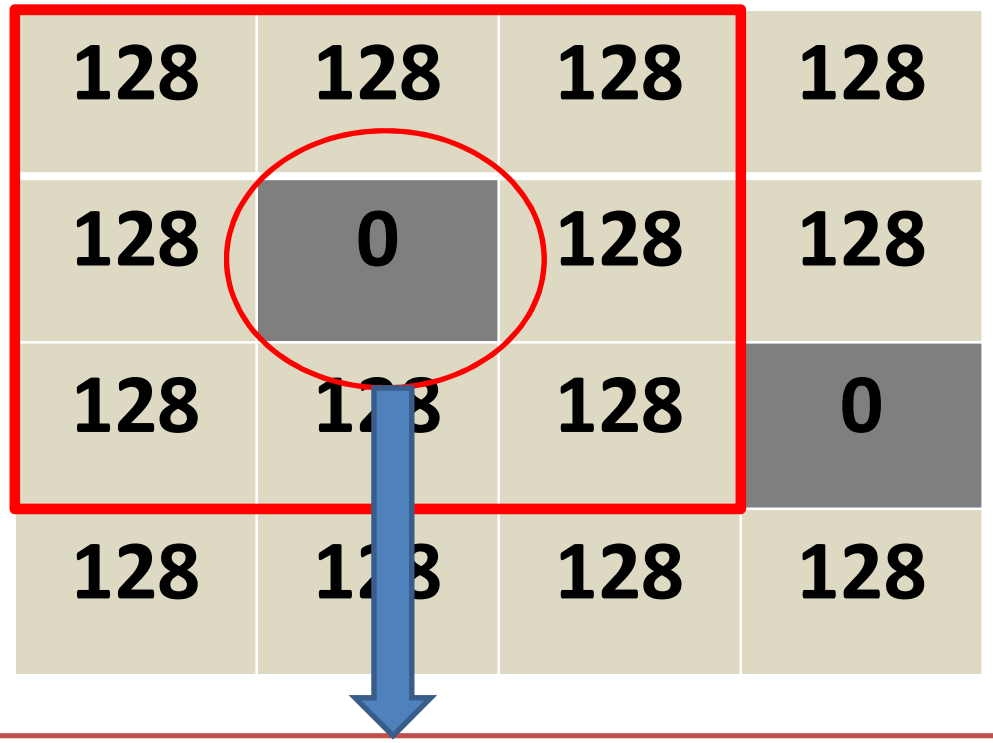
画像フィルタ

サンプル画像



画像フィルタ

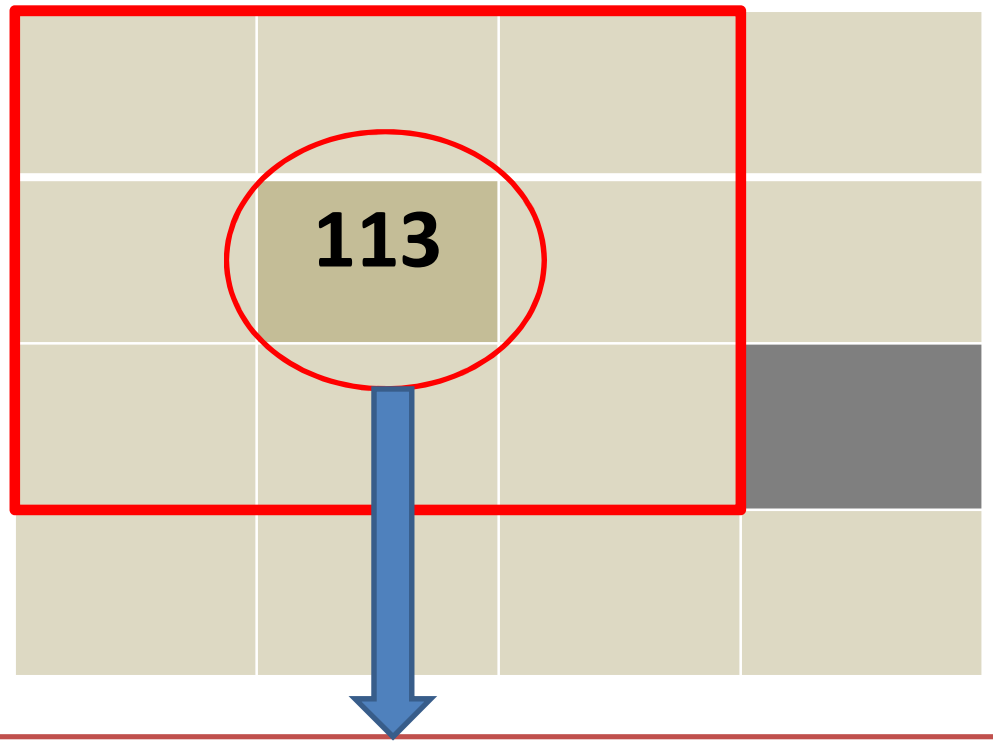
サンプル画像



$$\sum_{y=1}^{3}\sum_{x=1}^{3} f(x,y)/9 = (\text{int})113.8 = 113$$

画像フィルタ

サンプル画像



$\frac{1}{9}\sum_{y=1}^{3}\sum_{x=1}^{3} f(x,y)$ =(int)113.8=113

画像フィルタ

サンプル画像

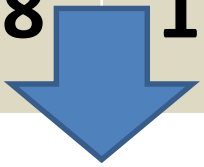
128	128	128	128
128	0	128	128
128	128	128	0
128	128	128	128

4つの画素全てに対して計算する

画像フィルタ

サンプル画像

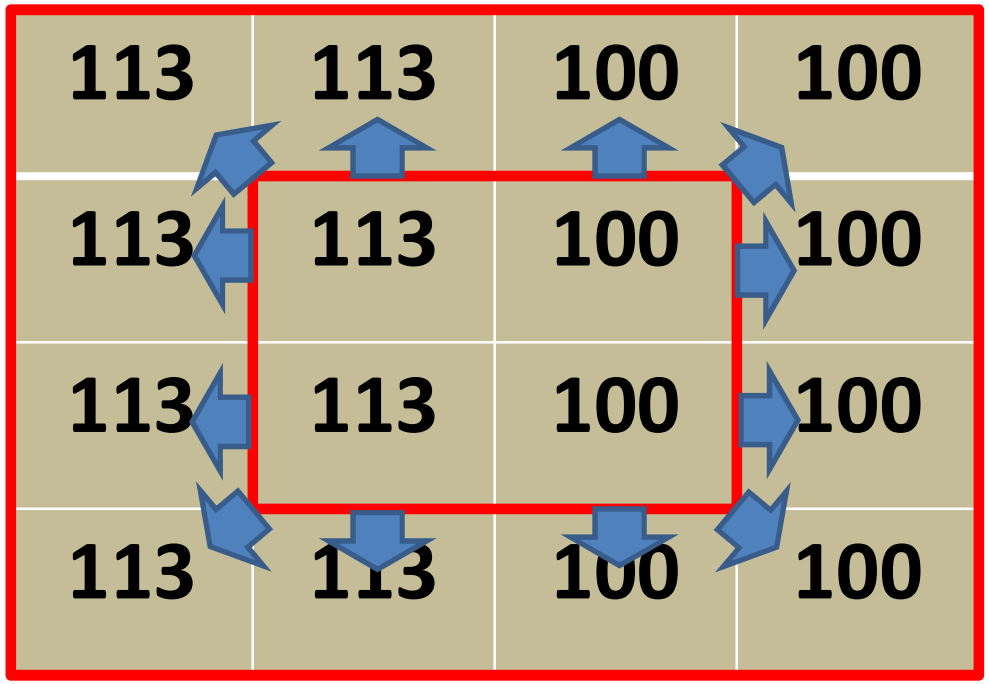
128	128	128	128
128	113	100	128
128	113	100	0
128	128	128	128



周囲8近傍平滑化アルゴリズム

外周部を補間

サンプル画像



平滑化フィルタ

- 以下のようなフィルタ係数を定義して利用する

```
帯域変数
int Cij[3][3]=
      {{1,1,1},{1,1,1},{1,1,1}};
double K = 1.0/9.0;
```

Cij*K　がフィルタ係数になるようにする

演習1

- 平滑化フィルタを実装せよ
- main関数の構造は以下の通りとすること

```
load_image(0,””);
filtering(0,1);  //平滑化フィルタ
                 //画像0を平滑化し，
                 //画像1を作成
save_image(1,””);
```

畳み込み積分

```
int calc(int n, int x, int y)
{
	int I,j,value=0;
	for(j=0;j<3;j++){
		for(i=0;i<3;i++){
			value += Cij[i][j]*image[n][x+i-1][y+j-1];
		}
	}
	return (int)(K * value);
}
```

外周部の補間

```
for(y=0;y<height[n2];y++){
        for(x=0;x<width[n2];x++){
                xsft = 0;ysft = 0;

                if(x == 0) xsft = 1;
                else if(x == width[n2]-1) xsft=-1;

                if(y == 0) ysft = 1;
                else if(y == width[n2]-1) ysft=-1;

                if(xsft != 0 || ysft != 0)
                        image[n2][x][y] = image[n2][x+xsft][y+ysft];
        }
}
```

外周である場合

微分フィルタ

- フィルタ係数を変更すると様々なフィルタになる

```
帯域変数
int Cij[3][3]=
      {{1,1,1},{1,1,1},{1,1,1}};
double K = 1.0/9.0;
```

エッジ強調（微分）処理

一次微分フィルタ（横方向）

-1	0	1
-1	0	1
-1	0	1

(K = 1)

一次微分フィルタ（縦方向）

-1	-1	-1
0	0	0
1	1	1

(K = 1)

エッジ強調（微分）処理

一次微分フィルタ（横方向）

-1	0	1
-1	0	1
-1	0	1

0	128	128	128
0	128	128	128
0	128	128	128
0	128	128	128

フィルタリングを実行

エッジ強調（微分）処理

一次微分フィルタ（横方向）

-1	0	1
-1	0	1
-1	0	1

	384		

エッジ強調（微分）処理

一次微分フィルタ（横方向）

-1	0	1
-1	0	1
-1	0	1

0	128	128	128
0	128	128	128
0	128	128	128
0	128	128	128

エッジ強調（微分）処理

一次微分フィルタ（横方向）

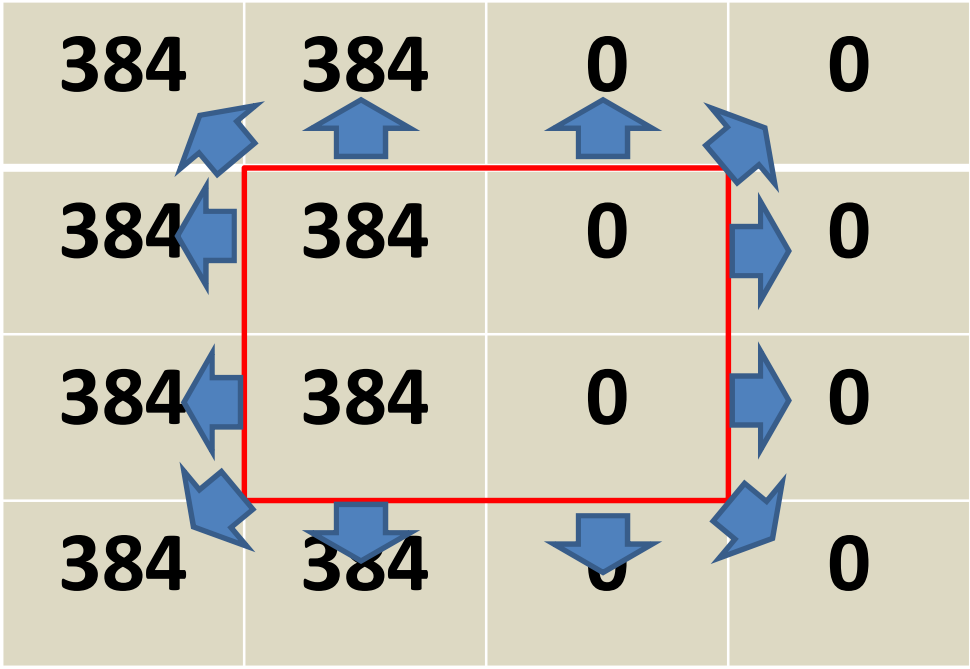
-1	0	1
-1	0	1
-1	0	1

	384	0	
	384	0	

エッジ強調（微分）処理

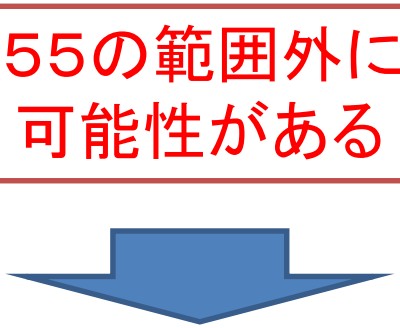
一次微分フィルタ（横方向）

-1	0	1
-1	0	1
-1	0	1



エッジ強調（微分）処理

0~255の範囲外になる
可能性がある



最小値~最大値が0~255
になるように変換する

384	384	0	0
384	384	0	0
384	384	0	0
384	384	0	0

エッジが強調される

フィルタリング後の画素の最大･最小値

```
int value, min = 255, max=0;
width[n2]=width[n1]; height[n2]=height[n1];

for(y=1;y<height[n1]-1;y++){
        for(x=1;x<width[n1]-1;x++){
                value = calc(n1,x,y);//前記
                if(value < min) min = value;
                if(value > max) max = value;
        }
}
```

画素値の調整

```
int value, min = 255, max=0;


for(y=1;y<height[n1]-1;y++){
        for(x=1;x<width[n1]-1;x++){
                value = calc(n1,x,y);    //前記
                image[n2][x][y] = (int)((double)(value-min)/(max-
min)*255.0);   //0～255に正規化
        }
}
//前述の外周の補間
```

ラプラシアンフィルタ
（二次微分フィルタ）

- ラプラシアンフィルタ $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\right) = \frac{\partial}{\partial x}\left(f(x,y) - f(x-1,y)\right)$$
$$= \left(f(x,y) - f(x-1,y)\right) - \left(f(x-1,y) - f(x-2,y)\right)$$
$$= f(x-2,y) - 2f(x-1,y) + f(x,y)$$

$x-1$をxと置き換えると

$$f(x-1,y) - 2f(x,y) + f(x+1,y)$$

ラプラシアンフィルタ
（二次微分フィルタ）

- ラプラシアンフィルタ $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

$$\Delta f(x,y) = f(x-1,y) + f(x,y-1) - 4f(x,y) + f(x+1,y) + f(x,y+1)$$

ラプラシアンフィルタ
（二次微分フィルタ）

0	0	0
1	-2	1
0	0	0

+

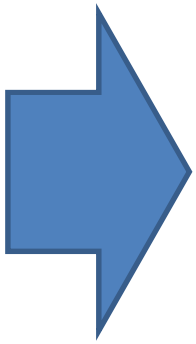
0	1	0
0	-2	0
0	1	0

+

1	0	0
0	-2	0
0	0	1

+

0	0	1
0	-2	0
1	0	0



1	1	1
1	-8	1
1	1	1

フィルタ係数を変更して
プログラムを実行する

演習２，３

- X方向のエッジ強調フィルタ処理を行い，処理前後の画像の比較を行え

- ラプラシアンフィルタ処理を行い，処理前後の画像の比較を行え