プログラミング演習　第11回

エッジを検出する　on 2013.01.24

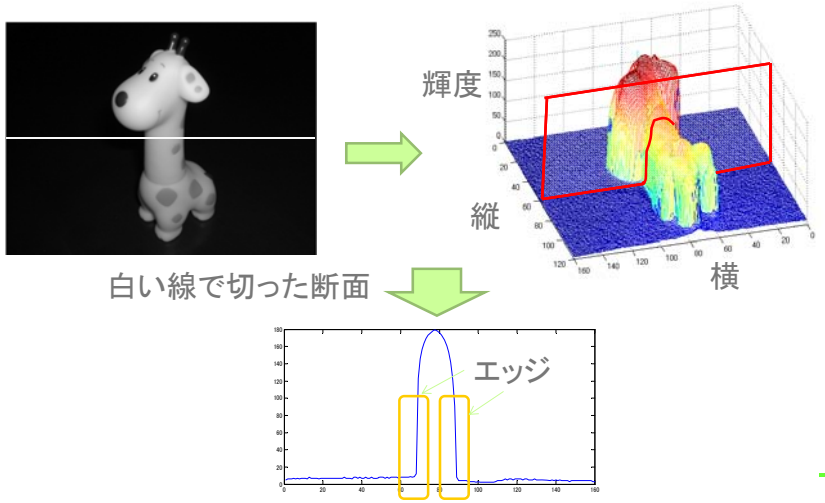
電気通信大学情報理工学部
知能機械工学科
長井隆行

Outline

- 画像の本質
  - 輝度の境目に情報あり！
- 画像の微分と2階微分
- エッジ検出
- 画像をぼかす
- 本日の課題

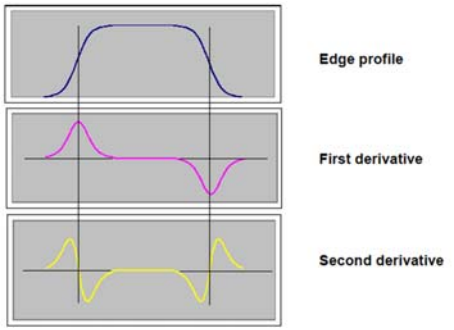
画像の本質

- 画像の情報は境目にあり！



エッジ抽出

- エッジ
  - 輝度が大きく変化しているところ（境界）
- 画像の情報はエッジにあり
  - 人間の視覚系でも特定のエッジの方向に発火するニューロンが見つかっている
- 画像の微分
  - 画像上の全ての点で微分を計算すれば、変化しているところが一目瞭然！
  - 離散（時間）信号なので微分は、**差分**として考えればよい
  - 2階微分がゼロとなる点としても検出できる



画像の微分

- 横方向、縦方向の微分

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y)-f(x,y)}{x+1-x} = f(x+1,y)-f(x,y)$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1)-f(x,y)}{y+1-y} = f(x,y+1)-f(x,y)$$

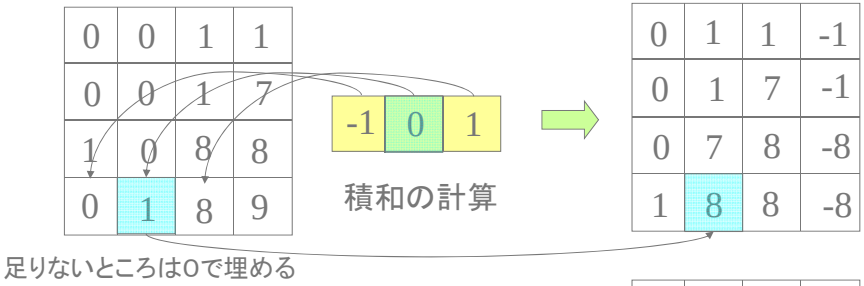
(x, y)における微分とするために

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y)-f(x-1,y)$$

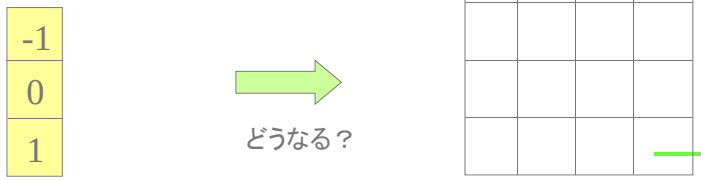
$$\frac{\partial f(x,y)}{\partial x} \approx f(x,y+1)-f(x,y-1)$$

画像の微分（実際の計算方法）

- 縦方向（横方向）のエッジ（微分）

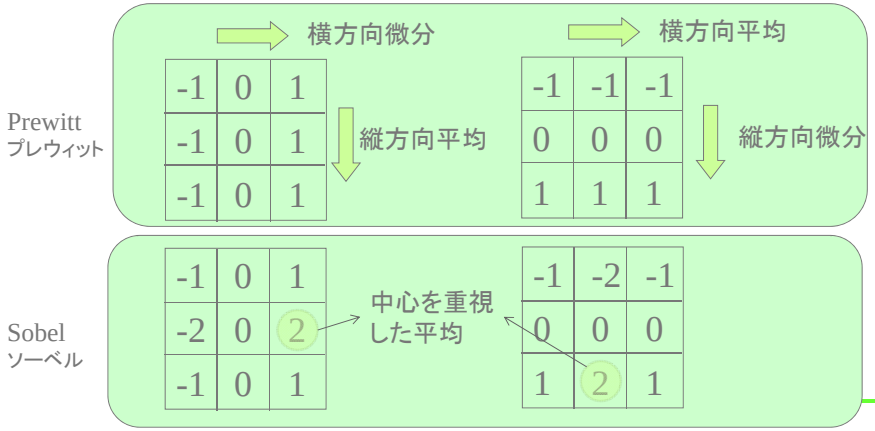


- 横方向（縦方向）のエッジ（微分）

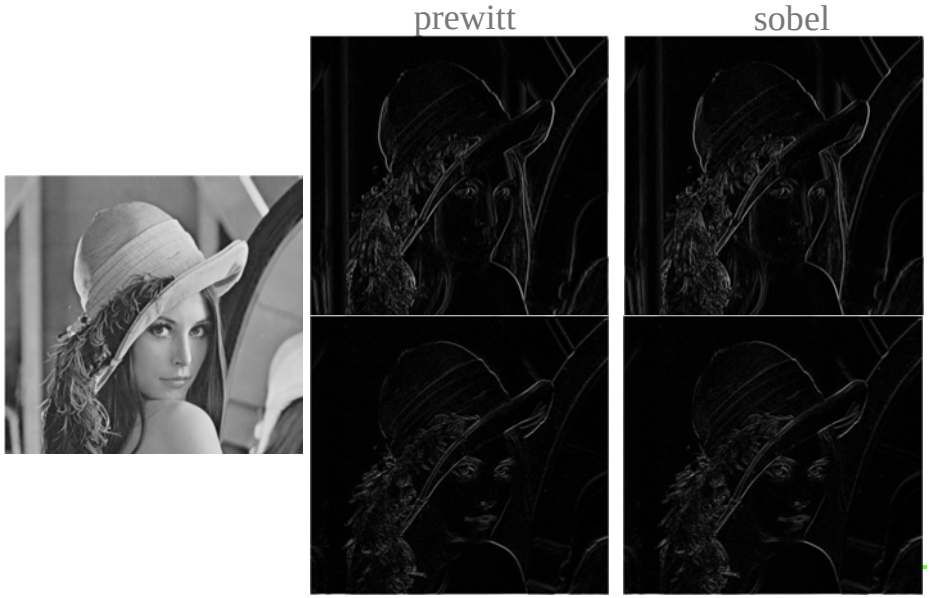


オペレータ

- 実際はノイズ除去のための平滑化と組み合わせる
- 平滑化によって色々なパターンがある
- 3×3がよく用いられる

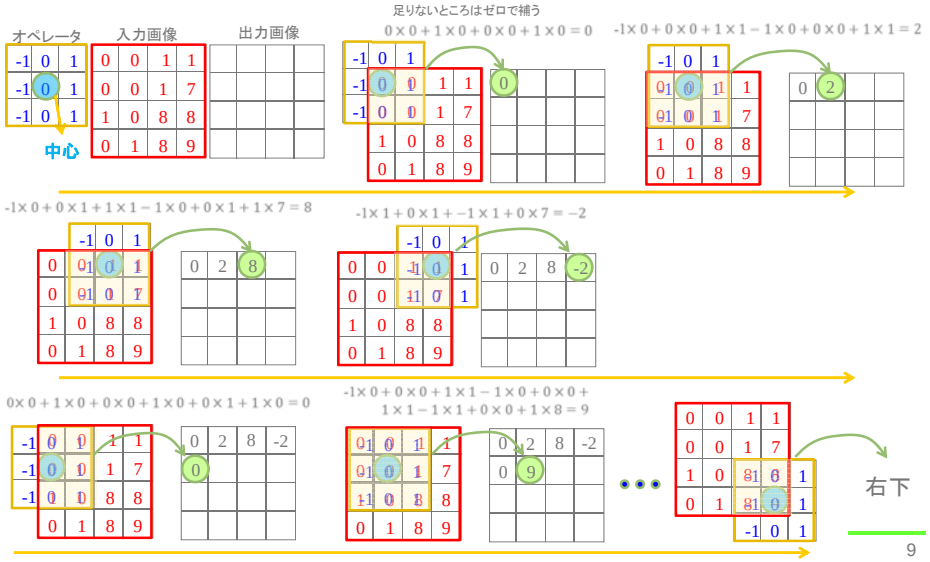


オペレータ適用の例



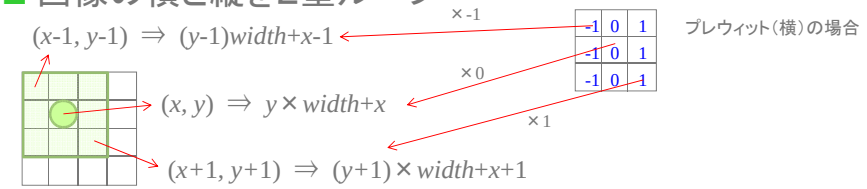
図解　オペレータ適用

- オペレータ適用（フィルタリング・2次元畳み込み）の計算手順を図解



プログラムでの実現

- 画像の横と縦を2重ループ



- （注目画素）を画像の左上から右下まで動かす　⇒　2重ループ

p11-1.c

```
double pixel_val = 0.0;
for(y=1; y<height-1; y++){
        for(x=1; x<width-1; x++){
                pixel_val = -1.0*buffer[(y-1)*width+x-1]+buffer[(y-1)*width+x+1]
                        -1.0*buffer[y*width+x-1]+buffer[y*width+x+1]
                        -1.0*buffer[(y+1)*width+x-1]+buffer[(y+1)*width+x+1];
                out_buffer[y*width+x]  = (fabs(pixel_val) <= 255) ?  (unsigned char) fabs(pixel_val) : 255;
        }
}
```

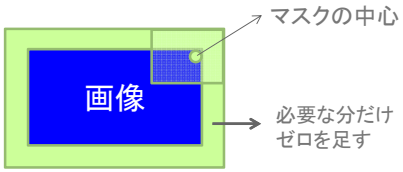
微分値の絶対値を画像で表示するため

条件演算子

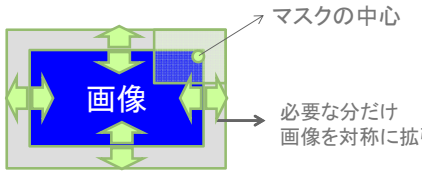
画像の境界問題（補足）

- 図解のスライドではゼロを補って考えましたが・・・
- 前のプログラムでは端を処理しないことで対処しました
  - 単純に面倒だから
- 境界をちゃんと処理するには
  - ゼロを埋める
  - 対称拡張

⇒ if文で一生懸命場合分け
端を膨らませたバッファを用意する



ゼロ埋め



対称拡張法

2階微分

- 微分の微分（差分の差分）

$$\nabla^2 = \{f(x+1,y) - f(x,y)\} - \{f(x,y) - f(x-1,y)\}$$
$$+ \{f(x,y+1) - f(x,y)\} - \{f(x,y) - f(x,y-1)\}$$
$$= f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

laplacian
4近傍

0	1	0
1	-4	1
0	1	0

上下左右のみ考慮する

laplacian
8近傍

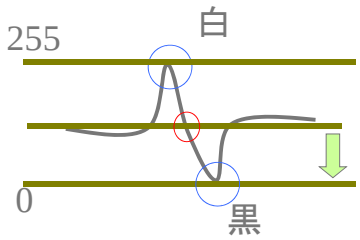
1	1	1
1	-8	1
1	1	1

ななめも考慮する

2階微分の例

- 4近傍のラプラシアン

白と黒にはさまれた灰色が
2階微分がゼロの点に相当する
（ゼロクロス）



注）表示を見やすくするために最小値を（負の値）
引いて、そのときの最大値で正規化している

エッジ検出

- 最終的には、その画素がエッジなのかエッジでないかを判定する必要がある
  ⇒様々な手法が提案されている
- オペレータの結果を閾値処理する（最も簡単）

$$p(x,y)=\sqrt{prewitt\_x(f(x,y))^2+prewitt\_y(f(x,y))^2}$$

$$p(x,y)=\begin{cases}\geq \text{閾値：}\ 255\ (\text{エッジ})\\ <\text{閾値：}\ 0\ (\text{エッジではない})\end{cases}$$

- ラプラシアンの結果のゼロクロスを抽出する
  - ゼロクロスは符号の変化を捉えれば簡単に見つかる
  - ノイズのために細かなエッジが大量に見つかってしまう
  - 対象としている点の周りの値が大きく変化しているかどうかをチェックする（閾値）
  - ラプラシアンの結果をガウスフィルタでぼかす（ノイズを除去する）
    ⇒LOGフィルタ（Laplacian Of Gaussianフィルタ）

Canny エッジ検出器（補足）

- 最もよく使われる手法
- 性能がよい
- 画像をぼかす（ガウシアンでノイズを除去）
- 微分の計算（大きさと方向）

$$|G|=|G_x|+|G_y| \qquad \theta=\tan^{-1}\frac{G_x}{G_y}$$

- エッジの細線化（エッジの方向を利用）
- 2つの閾値で値の大きい（信頼性の高いエッジ）と値の小さい（信頼性の低いエッジ）を選ぶ
- 信頼性の高いエッジをもとに、それに接続した信頼性の低いエッジを選んで行く

画像をぼかす

- 画像をぼかすためには
  - エッジのような変化を小さくする
  - 近傍画素を平均化すればよい

⇒平滑化（平らにするということ）
⇒積分と考えてもよい

平均化フィルタ
（移動平均）

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

移動平均よりもなめらかになる
ガウス関数をベースに作られる

3×3ガウシアン
フィルタ

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

オペレータのサイズはN×Nにできる

フィルタリング（補足）

- オペレータの適用による処理は一般化できる
  - ⇒フィルタリング（時間領域での畳み込み）
  - ⇒周波数領域では
    - 微分はハイパスフィルタ
    - 平滑化はローパスフィルタ
- 用いるオペレータ（フィルタ）の係数を変えれば同じプログラムが使える

フィルタリングプログラム

```
int x,y,i,k;
double mask[3][3]={{1.0/9,  1.0/9,  1.0/9},
                   {1.0/9,  1.0/9,  1.0/9},
                   {1.0/9,  1.0/9,  1.0/9}};
double pixel_val;

for(y=1; y<height-1; y++){
   for(x=1; x<width-1; x++){
      /*ここで3×3のフィルタリング（オペレータを適用）*/
      pixel_val  = 0.0;
      for(i=-1;i<2;i++){
         for(k=-1;k<2;k++){
            pixel_val += mask[i+1][k+1]*buffer[(y+i)*width+x+k];
         }
      }
      /*オーバーフローの処理*/
      out_buffer[y*width+x] = (unsigned char) pixel_val;
      if(pixel_val>255) out_buffer[y*width+x] = 255;
      if(pixel_val<0) out_buffer[y*width+x] = 0;
   }
}
```

3×3のオペレータ

p11-2.c

本日の演習

―準備―
- まずは、p11-1.c（画像を横方向に微分するプログラム）をダウンロードする
- p11-1.cを実行して画像の縦方向の線が検出されていることを確かめる
  - 画像はgray_girl.bmpを使う（自分で用意してもよいがサイズに注意）

―課題―　（ここから先をメールで送る）
- 画像のエッジを検出するプログラムを作る
  - p11-3.cを完成させる
  - 手法は、横方向のプレウィットと縦方向のプレウィットの出力の二乗和の平方根を閾値処理する　⇒　p.14の式
  - 閾値(threshold)を変えると結果の画像はどのように変化するかを観察する
- ソースコードと結果の画像（どの閾値でもよい）をメールで送る
- 送る画像のサイズは小さいものを送る（こちらで用意した画像と同じ程度）
- 送る際には注意事項をよく確認すること
  http://apple.ee.uec.ac.jp/COMPROG　⇒　諸注意
- 今日の講義の感想・質問をメール本文に書いてください
- よく分かった、ここが分からない、など

課題のヒント

```
/*****************画像処理をここで行う*********************/
  for(y=1; y<height-1; y++){
    for(x=1; x<width-1; x++){

      /*ここで横方向の微分を計算してpixel_val1に代入*/
      pixel_val1 = 0.0;
      for(i=-1;i<2;i++){
        for(k=-1;k<2;k++){
          pixel_val1 += mask1[i+1][k+1]*Rbuffer[(y+i)*width+x+k];
        }
      }

      /*ここで縦方向の微分を計算してpixel_val2に代入*/
      pixel_val2=0.0;
      /*************************************************************
      [redacted]
      *************************************************************/


      /*微分値の2乗和の平方根が閾値(threshold)より大きければ255そうでなければ0を代入*/
      if(  [redacted]  )
      {
        out_buffer[y*width+x] = 255;
      }else{
        out_buffer[y*width+x] = 0;
      }
    }
  }
/*******************画像処理ここまで*****************************/
```

ここを参考にする

$$prewitt\_x(f(x,y)) \Rightarrow \text{pixel\_val1}$$

$$prewitt\_y(f(x,y)) \Rightarrow \text{pixel\_val2}$$