C#の基本
～様々な画像処理～

今回の目標

- 今回は画像処理ということで、実空間フィルタと空間周波数フィルタのプログラムに関して学習を行う。

- また、高速描画のための自作クラスに関しても紹介する。

実空間フィルタ処理

- まず、実空間フィルタ処理を行うプログラムを作成する。

- 今回は、エッジ検出フィルタ（ラプラシアンフィルタ）の作成を行う。

実空間フィルタ処理とは

- 元となる画像の明るさや色の情報であるピクセル毎の値を、各ピクセルとその周辺のピクセルの値を使って、新たにピクセル値を生成してく画像処理。空間フィルタと呼ばれるフィルタ行列（フィルタマトリクス、重み係数、演算子、オペレータなどともいう）をラスター走査しながら行う。

実空間フィルタ処理とは

- 実画像f(x、y)と空間フィルタh(x、y)の畳み込み演算(コンボリューージョン)を行うことで処理画像g(x、y)が得られる。周波数領域における画像処理は、実画像における畳み込みである。以下の式で表わされる。

$$g(x、y)＝f(x、y)＊h(x、y)$$

ラプラシアンフィルタとは

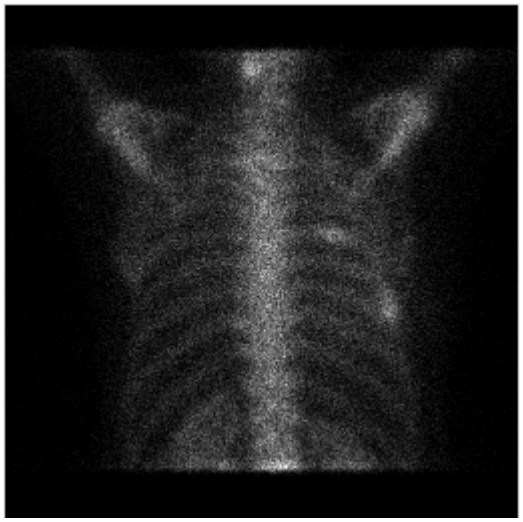
- 以下に、ラプラシアンフィルタの式を示す。

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

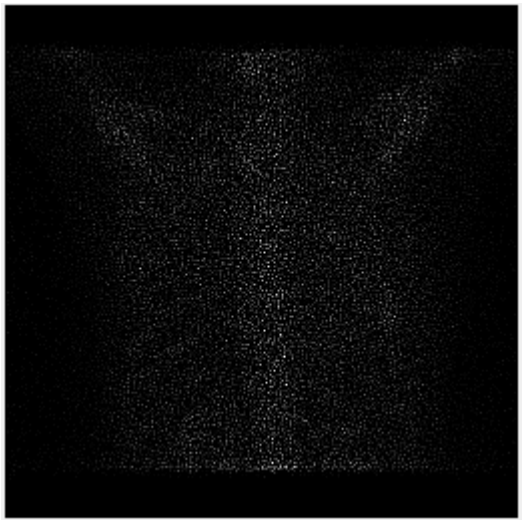
- このフィルタ処理を行うと画像の輪郭を描画する事が出来る。

ラプラシアンフィルタとは

- 以下に、ラプラシアンフィルタの処理結果を示します。
- 画素ごとのエッジが強調されている事が分かる。



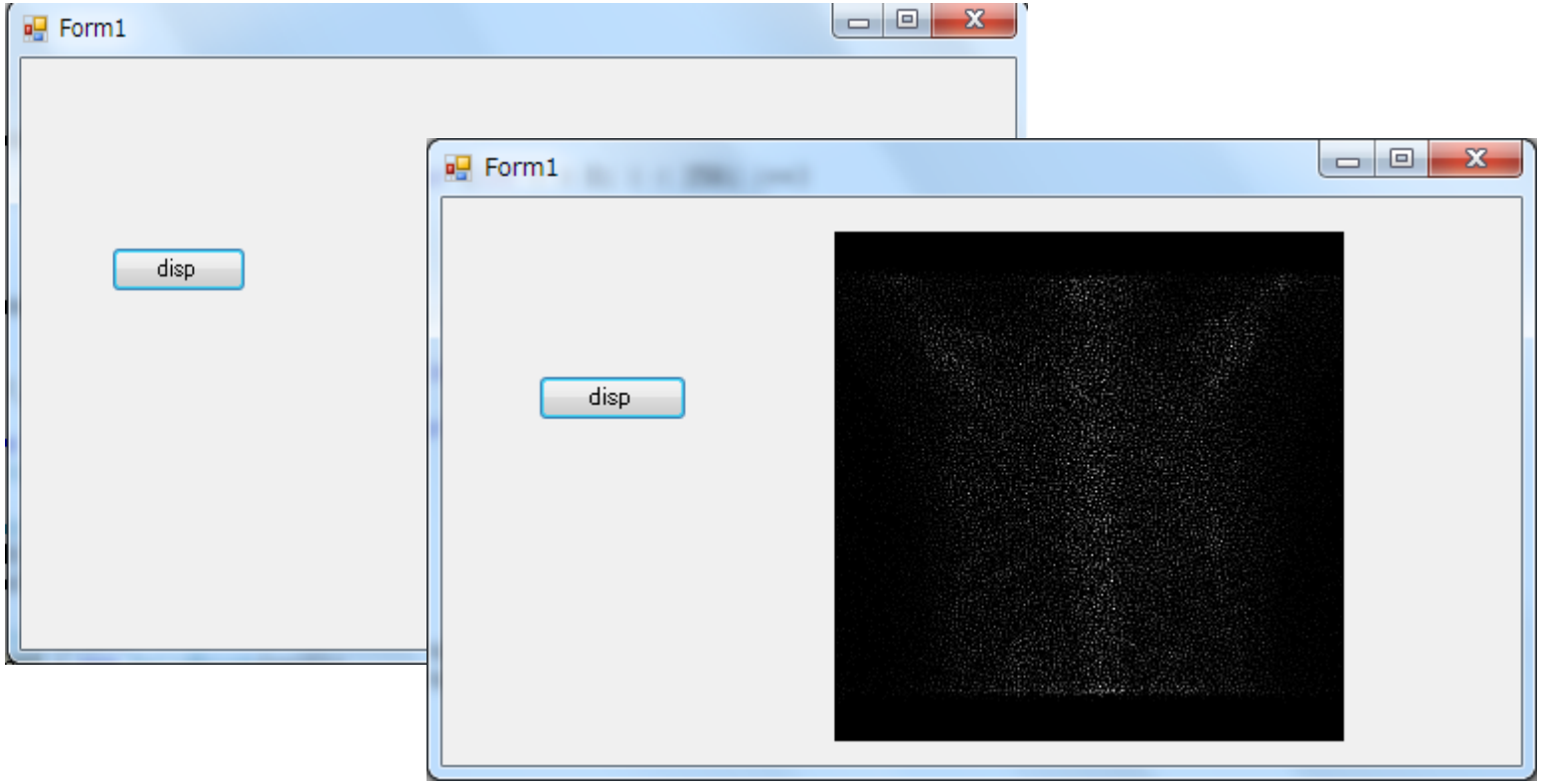
通常画像



フィルタ処理後

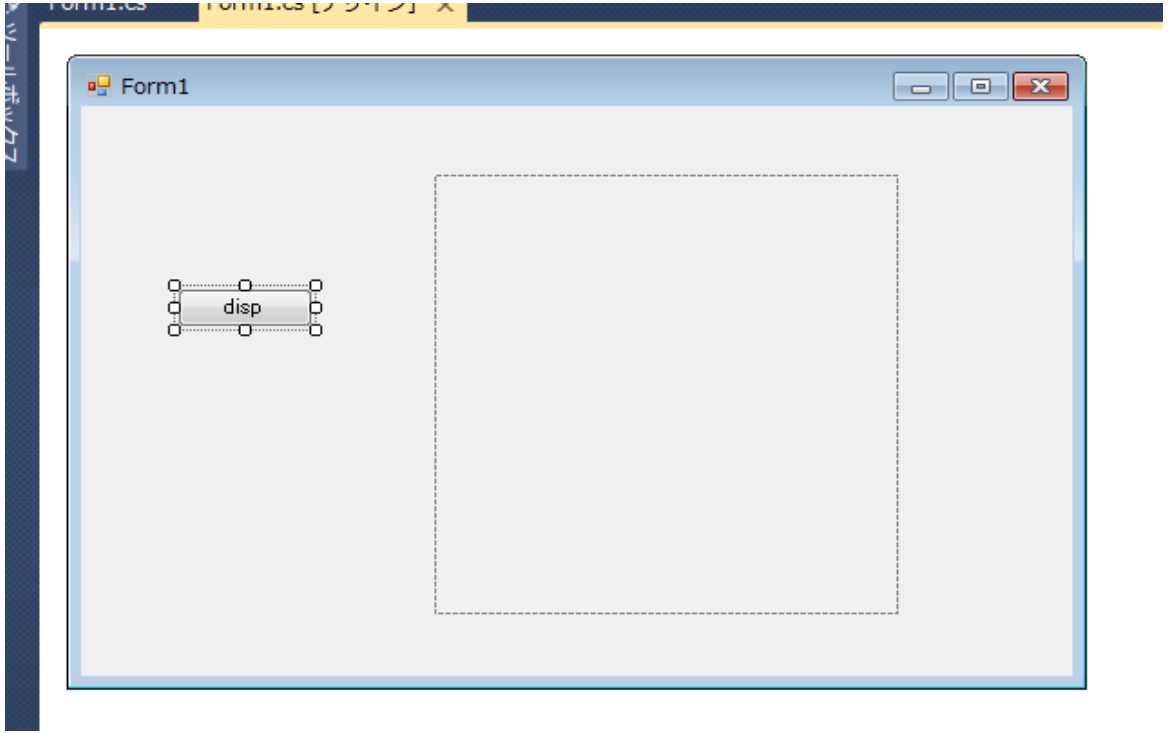
今回作るプログラム

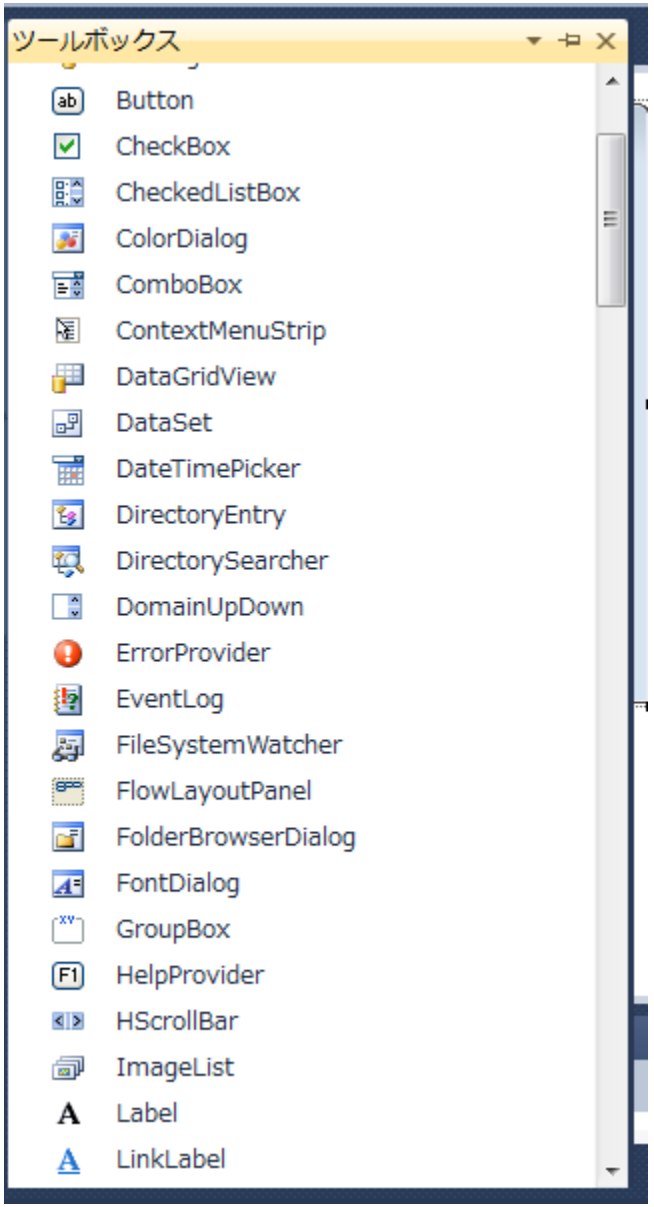
- 入力画像（bone1～6）をフィルタ処理をして出力するプログラムを作る。



フォーム作り

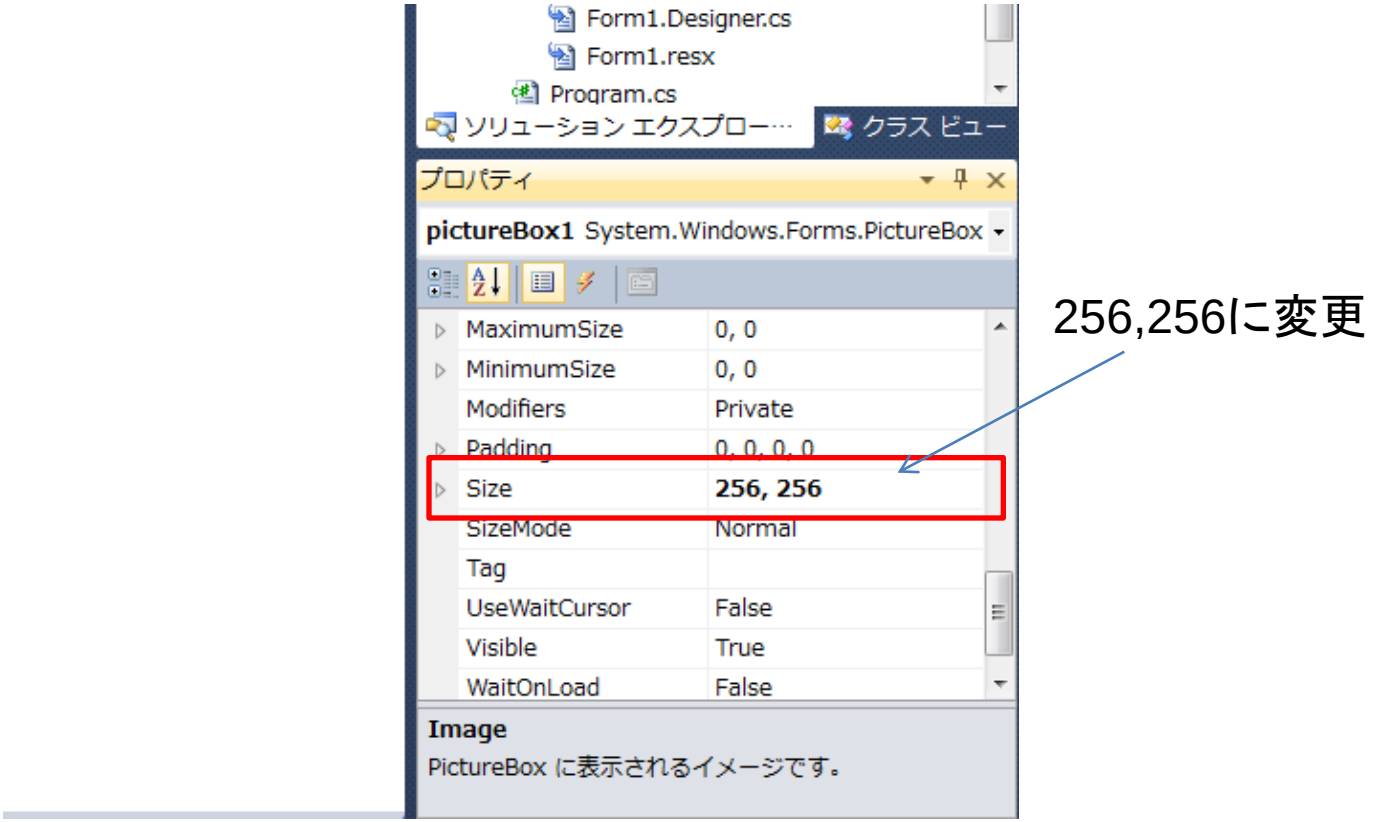
- まず、label、pictureBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



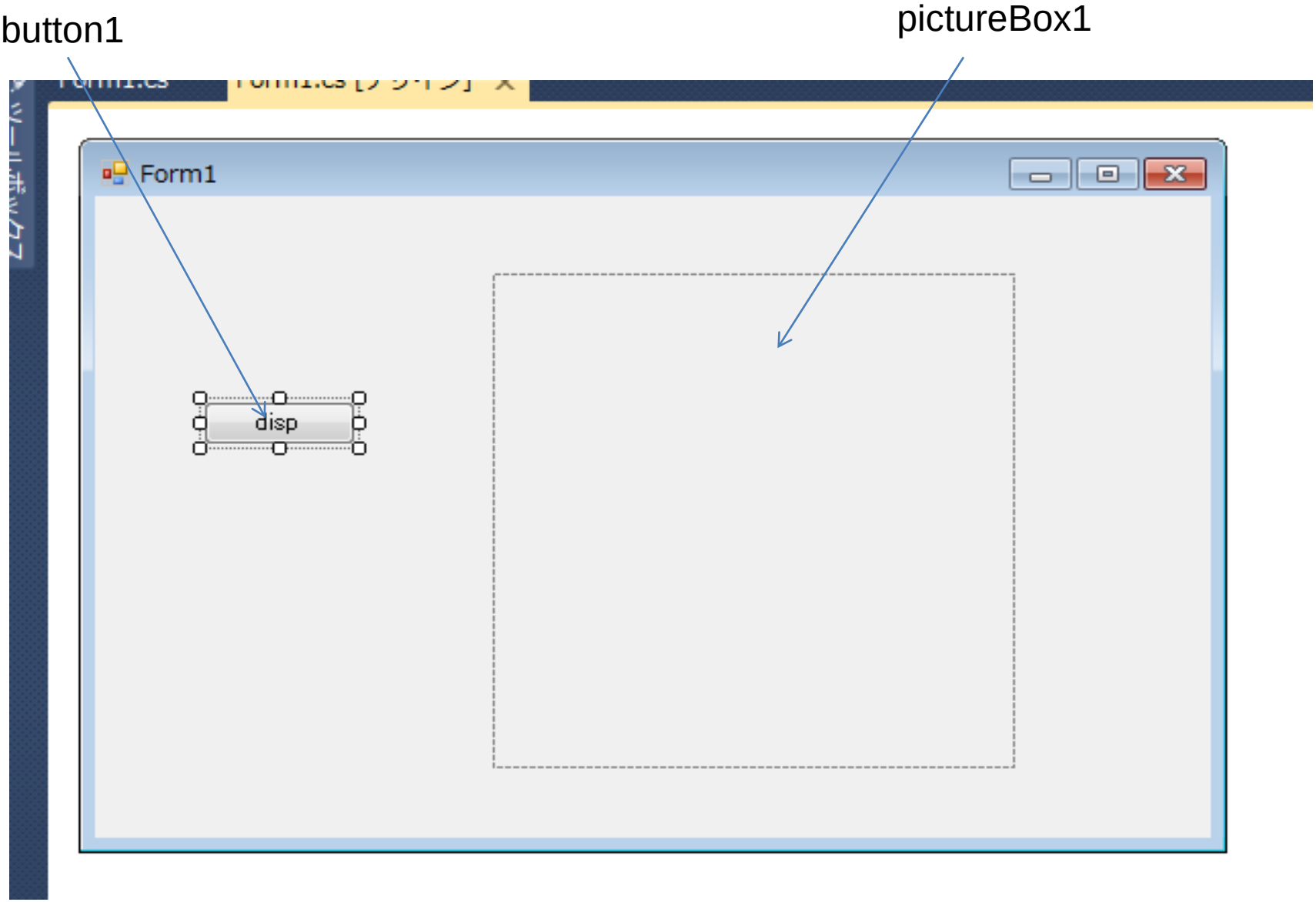


フォーム作り

- pictureBoxを右クリック→プロパティより”Size”の値を”256,256”に変更する。



フォーム作り



コード記述

- button1をダブルクリックし、

Private void button1_Click(object sender, EventArgs e)
を表示させる。

- コード全体に関してはprogram5aの

Private void button1_Click(object sender, EventArgs e)
の中を参照する。

コード解説（要点解説）

```
int[,] img0 = new int[256, 256];
int[,] img = new int[256, 256];
```

- 今回は2つの配列を宣言している。
- img0には読み込んだデータの値が入り、imgnにはフィルタ処理後の配列が入る。

コード解説

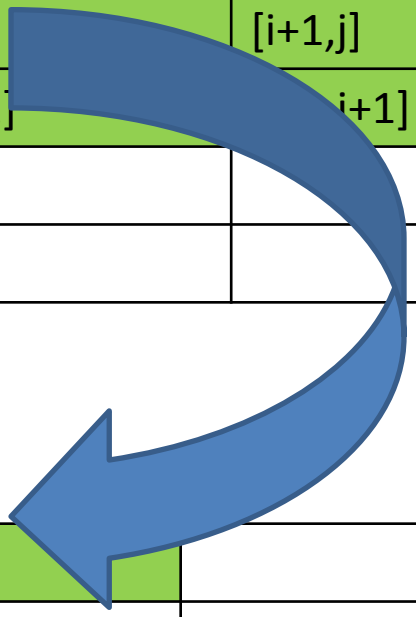
```
for (int j = 1; j < 255; j++)
        {
            for (int i = 1; i < 255; i++)
            {
    img[i, j] = img0[i - 1, j - 1] * (-1) + img0[i, j - 1] * (-1) + img0[i + 1, j - 1] * (-1) +
                img0[i - 1, j] * (-1) + img0[i, j] * 8 + img0[i + 1, j] * (-1) +
                img0[i - 1, j + 1] * (-1) +      img0[i, j + 1] * (-1) + img0[i + 1, j + 1] * (-1);
            }
        }
```

- ここで、ラプラシアンフィルタ処理を行っている。
- 周囲3×3の画素値からimgの1画素を作っていることが分かる。

フィルタ処理のイメージ

img0

[i-1,j-1]	[i,j-1]	[I,j+1]		
[i-1,j]	[i,j]	[i+1,j]		
[i-1,j+1]	[i,j+1]	i+1]		



img

[i,j]		

- このように周辺3×3画素から一つの値を作っている

コード解説

- 以外のコードに関してはLecture4スライドにて解説済みである。

- もし、不明な点がある場合はそちらを確認する。

空間周波数フィルタ処理

- 実空間フィルタ処理に対して、空間周波数フィルタ処理は周波数空間に対してフィルタ処理を行う。

- 実空間画像をフーリエ変換する事により、空間周波数に変換する事が出来る。

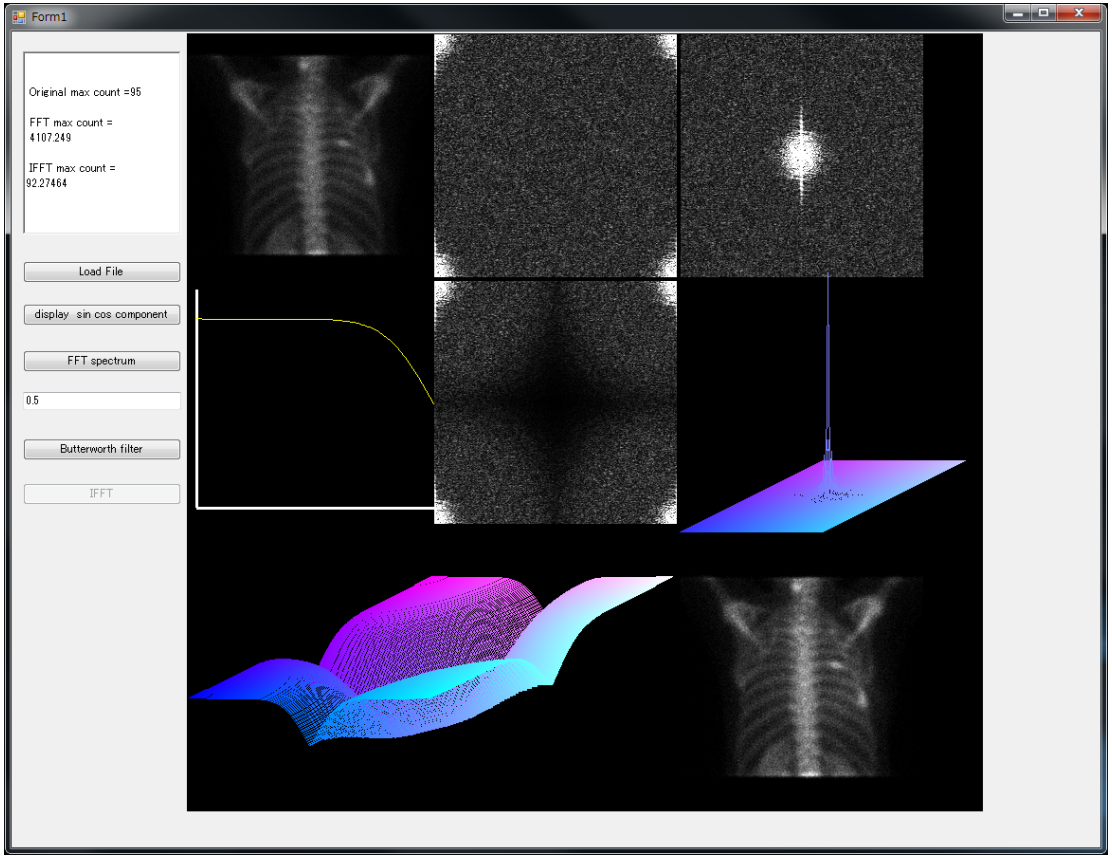
フーリエ変換

- 実空間画像を空間周波数に変換するための処理。数学的な詳しい解説は教科書等を参照してください。

- ここでは、技術評論社「NUMERICAL　RECIPES IN C～C言語による数値計算のレシピ～」における高速フーリエ変換（FFT）のアルゴリズムをそのまま使用します。

空間周波数フィルタ処理

- 今回は空間周波数画像に対し、バターワースフィルタを掛けるプログラムを作成する。



フォーム作り＆コード全文

- フォームにおけるコントロールの配置法、プログラムコードに関してはButterworth.zipを参照してください。
- コードが長いため抜粋して解説していく。
- 各ボタンの動作に対応するコード（button1_Clickの様な部分）は各ボタンをだブルクリックする事により作ることができる。

コード解説　抜粋

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Imaging;
```

- 赤字の名前空間に関しては自分で追加してください。

コード解説　BitmapPlus名前空間

- これはBitmapクラスにおけるSetPixelメソッドを高速化させるための物である。

- System.Drawing.Imaging名前空間はこの名前空間を使用するために追加する。

- コード下部に記載されてる。

BitmapPlusクラス　使い方

```
BitmapPlus.BitmapPlus　bmpP
                = new BitmapPlus.BitmapPlus(bmp)
```

- 通常のBitmapクラスであるbmpから
  BitmapPlusクラスであるbmpPを作成する。

BitmapPlusクラス　使い方

bmpP.BeginAccess();

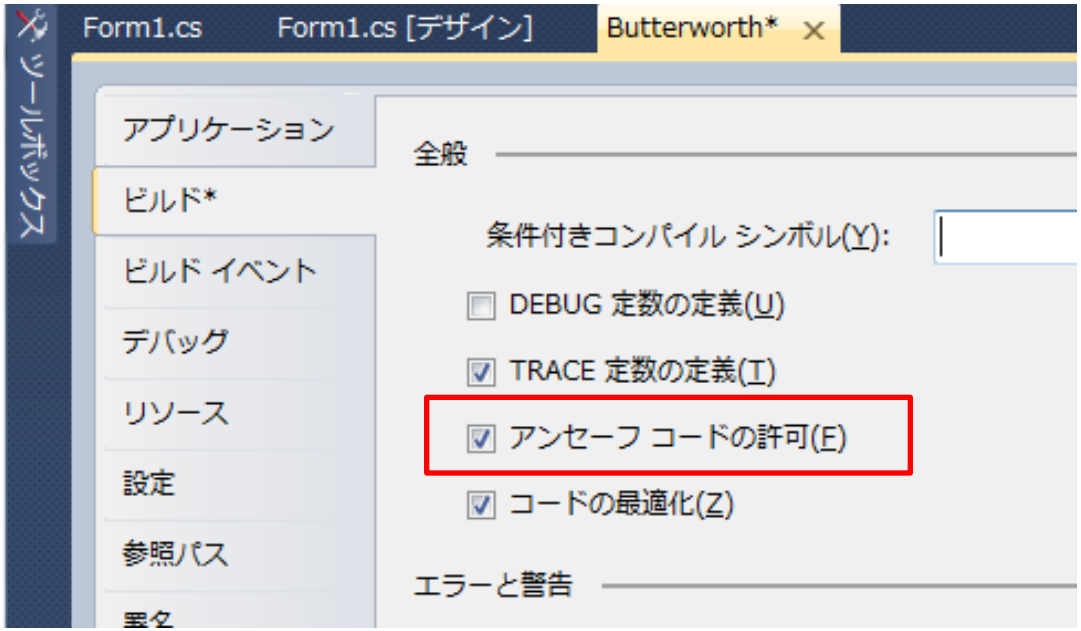
- BeginAccessメソッドにより処理の高速化を開始する。この間にSetPixelメソッドにより描画を行う。

bmpP.EndAccess();

- EndAccessメソッドを行わないと画像を表示させる事が出来ない。pictureBoxに表示させる前に行う。

BitmpapPlus名前空間　使い方

- BitmpapPlus名前空間を使用する際には、必ず、プロジェクト→Butterworthのプロパティの左側タブのビルドを選択。そこでアンセーフコードの許可にチェックを入れる。



コード解説　変数宣言

- ここで変数宣言をすると、すべてのボタン共通で使える変数になる。（例えば、ボタン1で読み込んだデータをボタン2以降でも使うことができる。）

- このように宣言する場所により変数の使える範囲（スコープ）を変えることができる。

コード解説　FFT

```
public void FFT2D(float[] x, float[] y, int data_n, int
    invert)
        {
                        ～

        }
```

- このように外部に自分のメソッドを作成することもできる。
- このアルゴリズムに関しては使い方だけ分かればよい。

フーリエ変換の使い方

- もし、invert＝1ならば、xの配列に原画像のデータを1次元配列で代入する。

- data_nには縦横のピクセル数を記入する

- すると、画像の実数成分はxに、虚数成分はyに入る様な関数になっている。

逆フーリエ変換の使い方

- invert＝-1ならば、x、yの配列にxに画像の実数成分、yに画像の虚数成分を入力
- data_nには縦横のピクセル数を記入する。
- すると、実画像はxの1次元配列に入る。

例外処理　try,catch

- try,catchは例外処理に使われる。

- この文はもしtry{～}の処理中にエラーが起こった場合にはcatchの処理を行いなさいという意味になる。

コード解説　button1_click

```
button2.Enabled = false;
button3.Enabled = false;
button4.Enabled = false;
button5.Enabled = false;
```

- 順番にボタンを押させるためにボタン1を押した時点ではボタン2～5は押せない状態にして置く。
- ボタン1の最後でボタン2を使える状態にしている

コード解説　button2_click

```
for (j = 0; j <= 255; j++)
        {
           for (i = 1; i <= 256; i++)
           {

              x[i + 256 * j] = (float)img[i, j + 1]; y[i + 256 * j] = 0;
           }
        }
```

- 2次元配列であるimgを1次元配列xに代入する。（yの方はすべて0にして置く）
- FFTが1次元配列の入力しか受け付けないためこのような処理が必要なる。

コード解説　button2_click

```
FFT2D(x, y, 256, 1);
```

- 今作ったx、yを用いてフーリエ変換を行う。
- この関数を用いるとx、yには実数成分と今日数成分が入る。

コード解説　button2_click

```
for (j = 0; j <= 255; j++)
        {
           for (i = 1; i <= 256; i++)
           {

  imgFFT[i, j + 1] = (float)Math.Sqrt(x[i + 256 * j] * x[i + 256 * j] + y[i + 256 * j] * y[i + 256 * j]);

           }
        }
```

- imgFFTに実数成分と虚数成分からパーワースペクトルの情報を代入する。

コード解説　button2_click

```
for (j = 1; j <= 128; j++)
        {
           for (i = 1; i <= 128; i++)
           {

              g[i, j] = x[(129 - i) + (129 - j) * 256];
              g[i + 128, j] = x[(257 - i) + (129 - j) * 256];
              g[i, j + 128] = x[(129 - i) + (257 - j) * 256];
              g[i + 128, j + 128] = x[(257 - i) + (257 - j) * 256];
           }
        }
```

- フーリエ変換の原点が四隅にあるため原点を中心に直している。

コード解説　button4_click

```
cutoff = double.Parse(textBox1.Text);
```

・入力された遮断周波数（0.5～0.01）をdouble型の変数に変換している。

コード解説　button4_click

```
for (i = 0; i <= 200; i++)
        {

           Bu[i] = 1.0 / (1.0 + Math.Pow(((double)i / (cutoff * 256.0)), 9.0));
        }
```

- cuttoff周波数からバターワースフィルタを作成する。

コード解説　button4_click

```
for (j = 1; j <= 256; j++)
            {
               for (i = 1; i <= 256; i++)
               {

                  Bx[i + 256 * (j - 1)] = x[i + 256 * (j - 1)] * B2[i, j];

                  By[i + 256 * (j - 1)] = y[i + 256 * (j - 1)] * B2[i, j];
               }
            }
```

- 作成したバターワースフィルタを用いてフィルタ処理を行う。

コード解説　button5_click

FFT2D(Bx, By, 256, -1);

- Bx、Byにはそれぞれバターワースフィルタ処理後のBx、By周波数成分の配列入る。
- Bxには処理後の画像出力が行われる。

まとめ

- 今回は実空間フィルタと空間周波数フィルタによる画像処理の紹介を行った。

- ややコードが難解に感じるかもしれないが、ここまでのスライドやMicroSoftのサイトを参考にし理解を深めて下さい。