Praktikum Filter/ Konvolusi

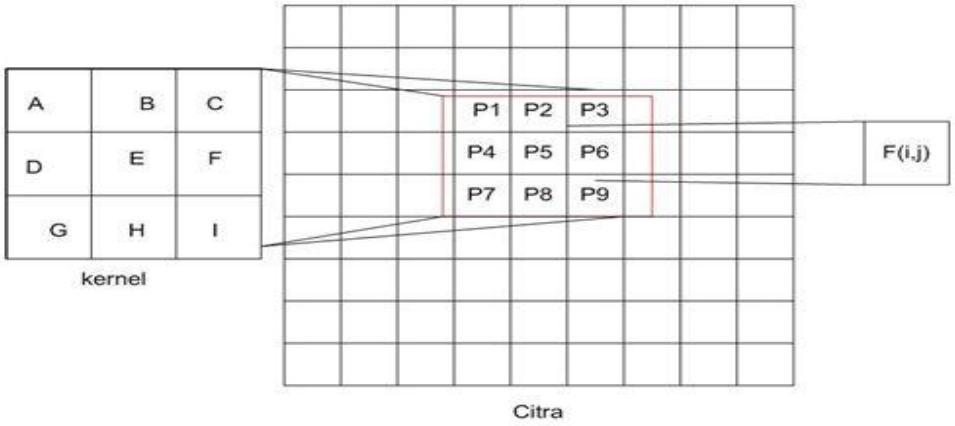
Hero Yudo Martono

Mei 2016

Konvolusi

- Konvolusi diskrit sangat banyak digunakan dalam pengolahan citra untuk :
- Konvolusi = jumlah berbobot dari pixel-pixel di sekeliling pixel sumber
- Bobot ditentukan oleh matrix kecil yang disebut = convolusion mask / kernel filter

Ilustrasi konvolusi



F(i,j)=Ap1+Bp2+Cp3+Dp4+Ep5+Fp6+Gp7+Hp8+Ip9

Contoh: misal citra f(x,y) yang berukuran 5x5 dan sebuah kernel dengan ukuran 3x3, matriks sebagai berikut :

$$F(x,y)=\begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{bmatrix} \qquad g(x,y)=\begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

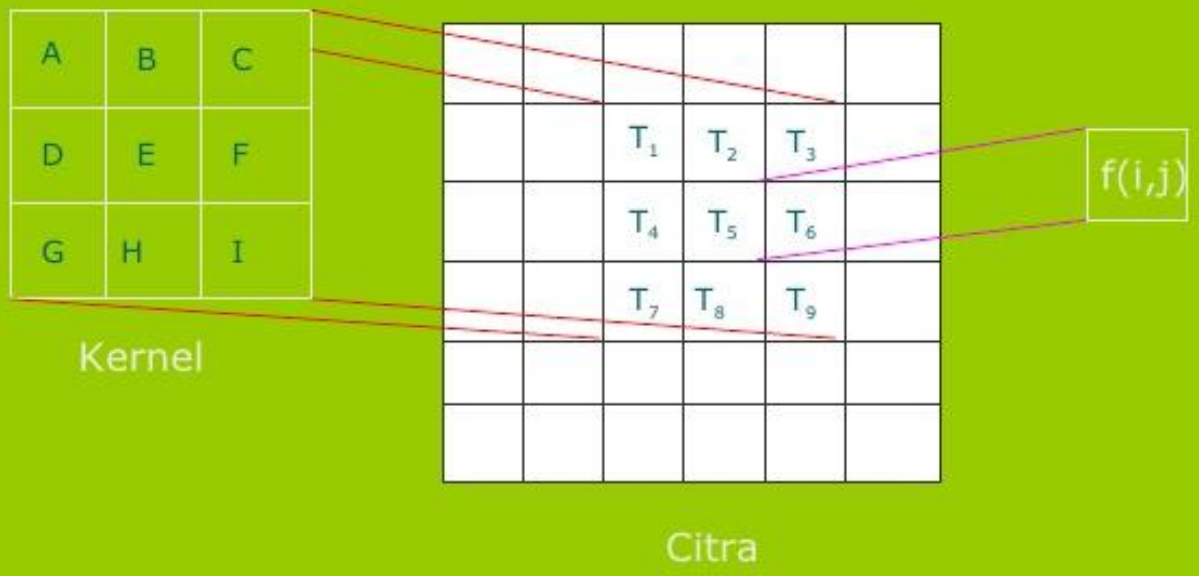
Operasi konvolusi antara citra f(x,y) dengan kernel g(x,y),

F(x,y)*g(x,y)

Konvolusi

Fungsi penapis g(x,y) disebut convolution filter, convolution mask, Convolution kernel, atau template.

Dlm domain diskret kernel konvolusi dinyatakan dlm btk matriks (umumnya 3 x 3, namun ada jg yg berukuran 2x2 atau 2x1 atau 1x2
Ukuran matriks biasanya lebih kecil dr ukuran citra.
Setiap elemen matriks disebut koefisien konvolusi



$$f(i,j) = AT_1 + BT_2 + CT_3 + DT_4 + ET_5 + FT_6 + GT_7 + HT_8 + IT_9$$

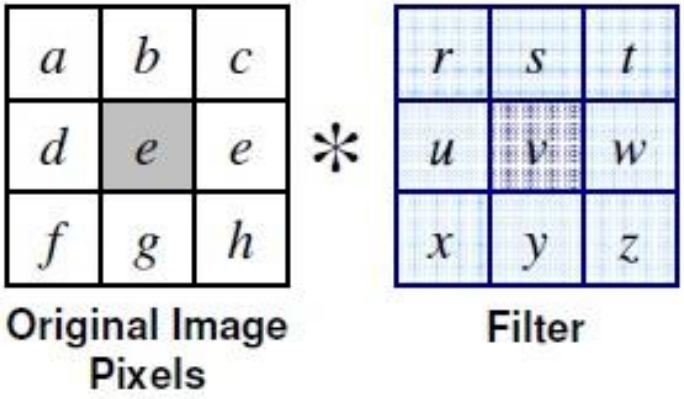
Konvolusi

Konvolusi dari H dan X didefinisikan dengan:

$$H \otimes X = \sum_{y} \sum_{x} H(x, y).X(T_x - x, T_y - y)$$

Dimana (x,y) adalah posisi filter
dan (Tx,Ty) adalah titik yang difilter

Konvolusi



$$e_{processed} = v*e + z*a + y*b + x*c + w*d + u*e + t*f + s*g + r*h$$

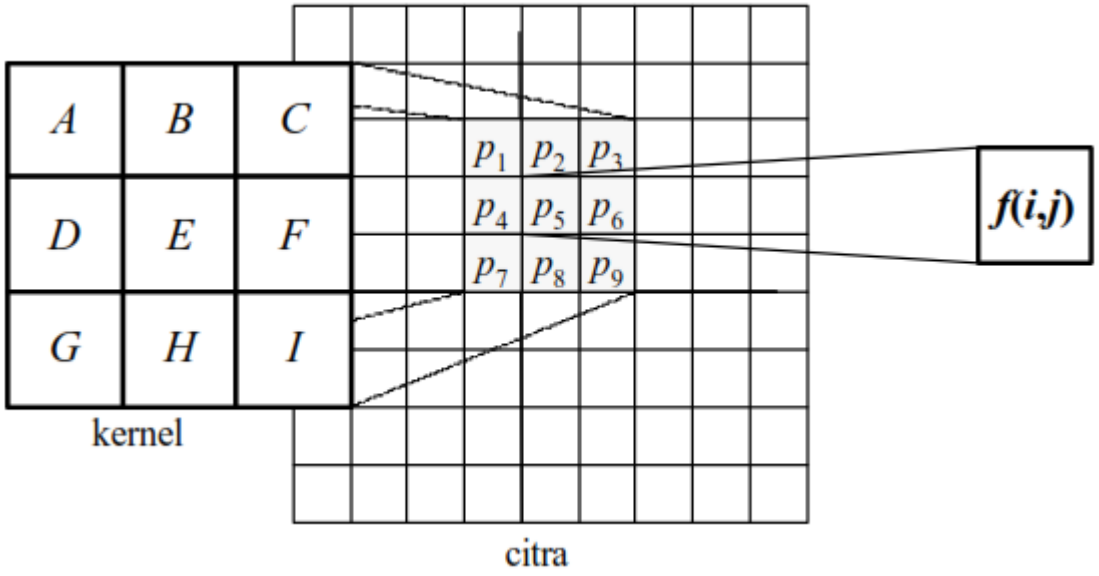
3	4	2	5	1
2	1	6	4	2
3	5	7	1	3
4	2	5	7	1
2	5	1	3	2

(i) Citra Semula

*	*	*	*	*
*	18			

(ii) hasil konvolusi

Konvolusi

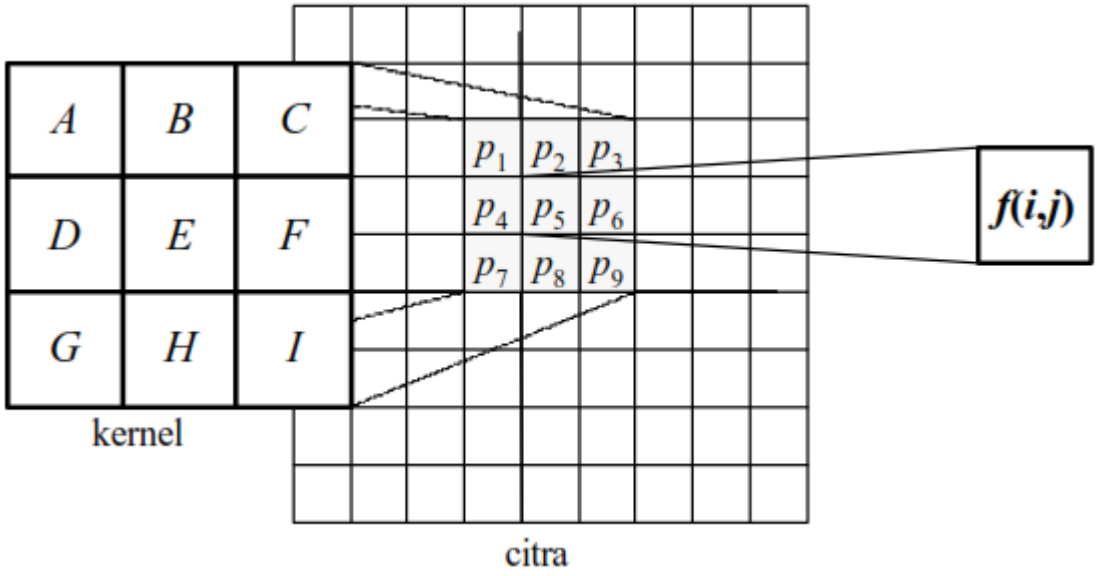


4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

	3	0	2	
	0			

Konvolusi



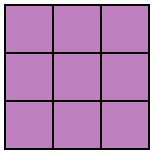
4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
3	5	2	4	4

———

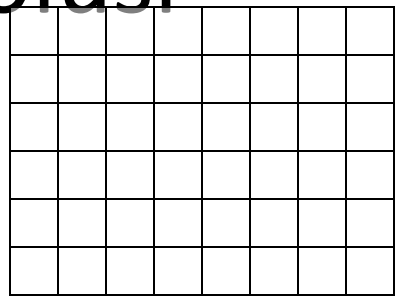
	3	0	2	
	0			

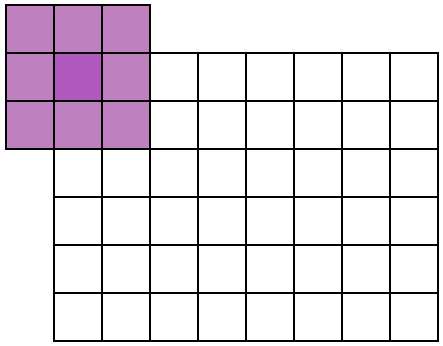
Proses Konvolusi

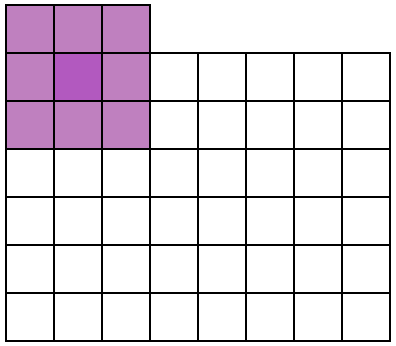
H =

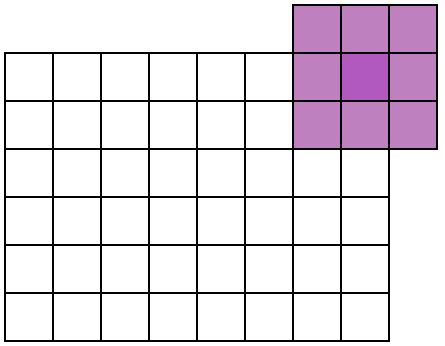


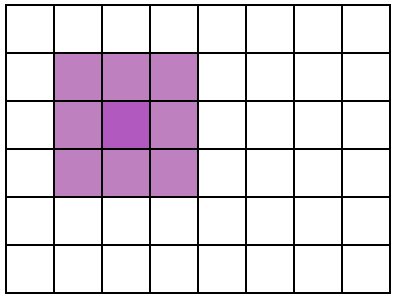
X =

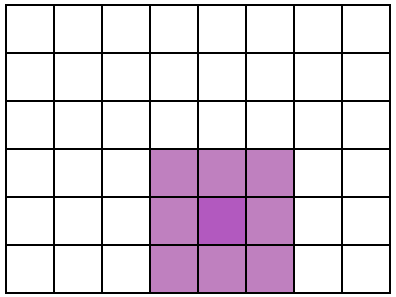


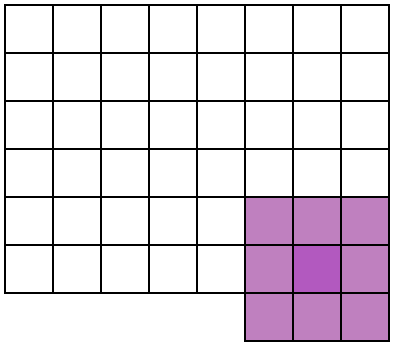












Contoh Konvolusi

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

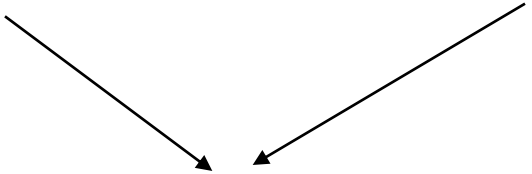
Karena ukuran H adalah 3x3 agar simetri terhadap 0, maka batas perhitungan adalah -1, 0 dan 1 untuk setiap posisi u dan v

$$y(i, j) = \sum_{v=-1}^{1} \sum_{u=-1}^{1} h(u+2, v+2).x(i+u, j+v)$$

Y(2,3) = H(1,1).X(1,2) + H(1,2).X(1,3) + H(1,3).X(1,4) +
H(2,1).X(2,2) + H(2,2).X(2,3) + H(2,3).X(2,4) +
H(3,1).X(3,2) + H(3,2).X(2,3) + H(3,3).X(3,4)
= (1)(0) + (1)(0) + (1)(0) + (1)(1) + (4)(1) + (1)(0) + (1)(1) + (1)(1) + (1)(0)
= 0 + 0 + 0 + 1+ 4 + 0 + 1 + 1 + 0
= 7

Contoh Konvolusi

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



$$Y = H \otimes X = \begin{bmatrix} 6 & 4 & 2 & 1 \\ 8 & 10 & 7 & 2 \\ 8 & 10 & 7 & 2 \\ 6 & 4 & 2 & 1 \end{bmatrix}$$

Filter Kernel

- Filter kernel H adalah suatu matrik yang menyatakan model filter (dalam spacial) yang menjadi operator dalam proses filter pada gambar.
- Bentuk atau komposisi nilai yang ada di dalam filter kernel menunjukkan jenis filter yang digunakan.

$$H = \frac{1}{12}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

LOW PASS FILTER

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 3 \\ -3 & 0 & 1 \end{bmatrix}$$

HIGH PASS FILTER

$$H = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 0.5 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

BAND STOP FILTER

Low Pass Filter

- Low Pass Filter (LPF) adalah suatu bentuk filter yang mengambil frekwensi rendah dan membuang frekwensi tinggi.
- LPF digunakan untuk melakukan proses efek blur dan reduksi noise.
- Ciri-ciri kernel dari LPF adalah semua nilainya positif dan jumlah dari semua nilainya sama dengan satu

$$H(x,y) \geq 0 \quad \text{dan} \quad \sum_{xy} H(x,y) = 1$$

High Pass Filter

- High Pass Filter (HPF) adalah suatu bentuk filter yang mengambil frekwensi tinggi dan membuang frekwensi rendah.
- HPF digunakan untuk melakukan proses deteksi tepi.
- Ciri-ciri kernel dari HPF adalah nilai-nilainya terdiri positif, nol dan negatif, dan jumlah dari semua nilainya sama dengan nol

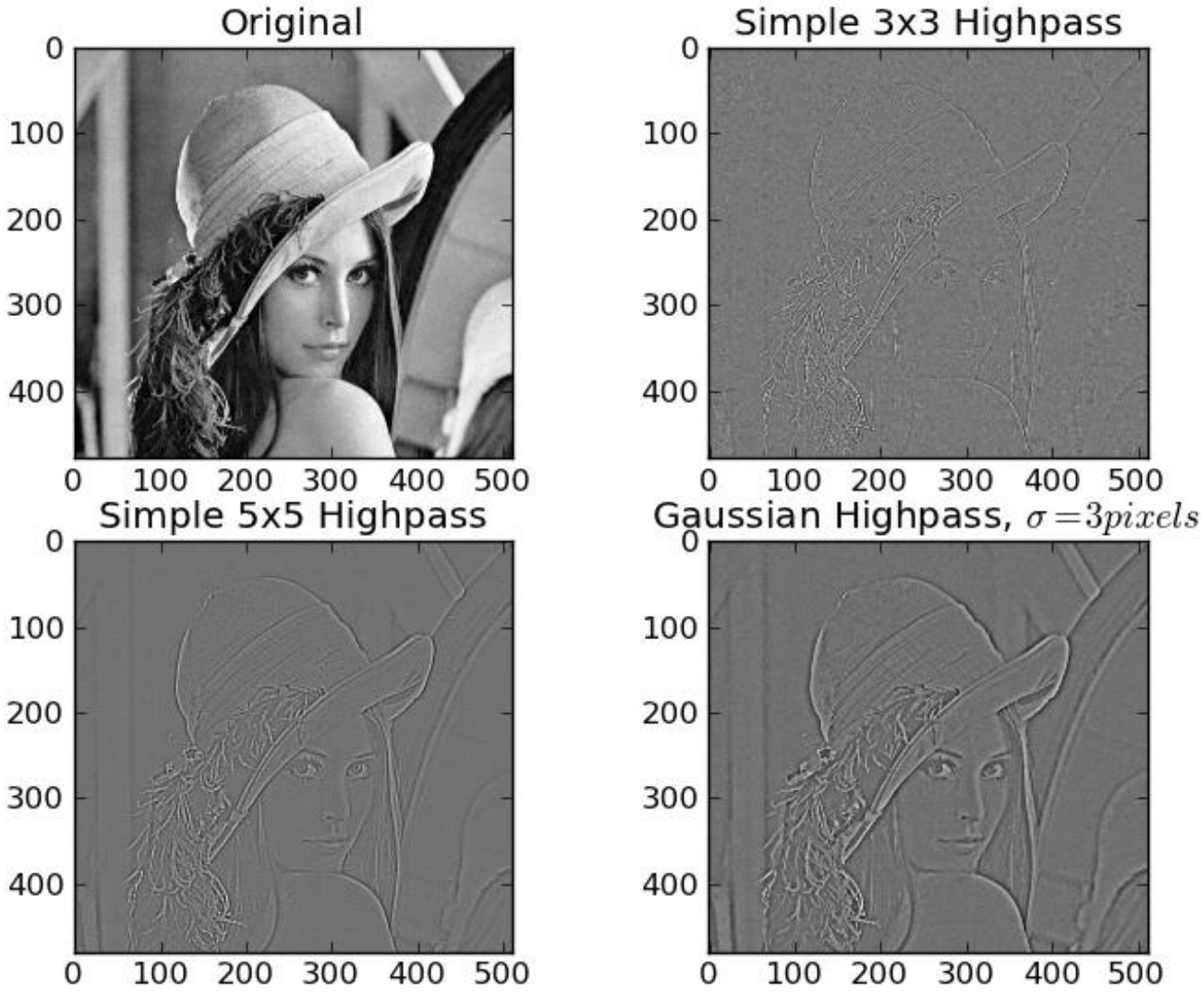
$$\sum_{xy} H(x, y) = 0$$

Band Pass Filter

- Band Pass Filter (BPF) adalah suatu bentuk filter yang mengambil frekwensi tinggi dan rendah dengan batasan tertentu.
- BPF digunakan untuk melakukan proses efek sharpeness.
- Ciri-ciri kernel dari BPF adalah nilai-nilainya terdiri positif, nol dan negatif, dan jumlah dari semua nilainya tidak sama dengan nol

$$\sum_{xy} H(x, y) \neq 0$$

Low/High Pass Filter



Form1

File

View Image | RGB | Flip Horizontal | Kuantisasi | Enhancement | Transformasi | Histogram | Transparan | LBP | Filter | Color Detection

Browse | Sobel | Laplace | Canny | Hough | Gradient

Horizontal | Vertical | Diagonal | Threshold | Hough

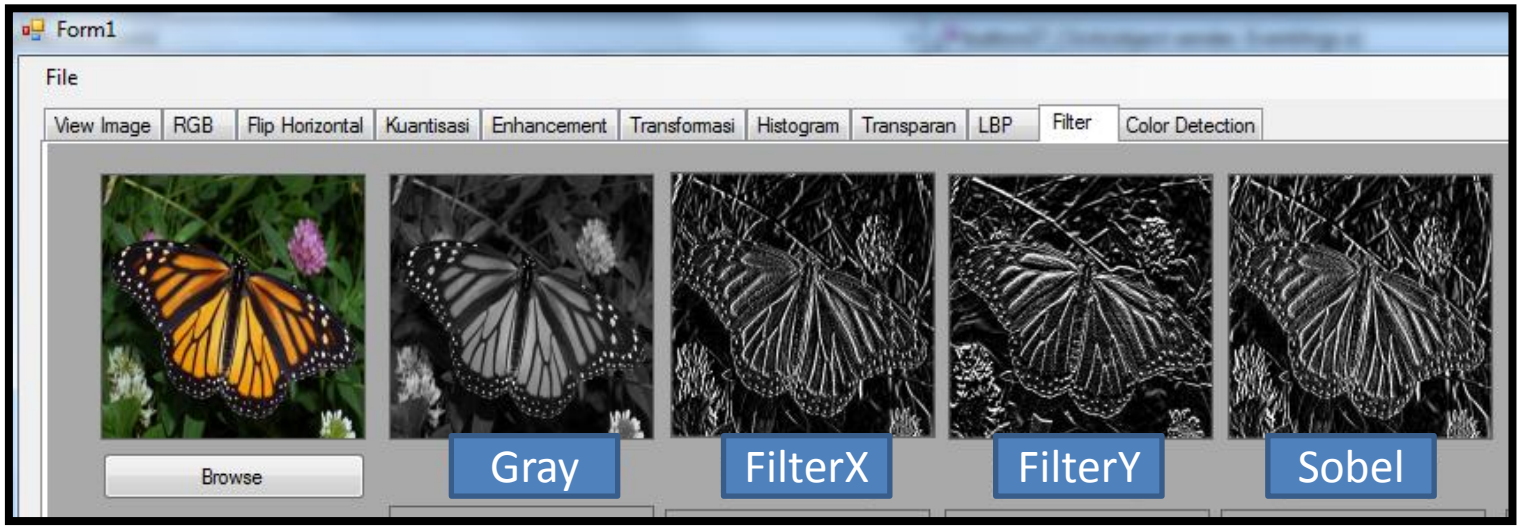
Horizontal | Vertical | Diagonal | Threshold | Hough

Sobel

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} A_{i-1,i-1} & A_{i,i-1} & A_{i+1,i-1} \\ A_{i-1,i} & A_{i,i} & A_{i+1,i} \\ A_{i-1,i+1} & A_{i,i+1} & A_{i+1,i+1} \end{bmatrix}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} A_{i-1,i-1} & A_{i,i-1} & A_{i+1,i-1} \\ A_{i-1,i} & A_{i,i} & A_{i+1,i} \\ A_{i-1,i+1} & A_{i,i+1} & A_{i+1,i+1} \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}$$



Form1

File

View Image | RGB | Flip Horizontal | Kuantisasi | Enhancement | Transformasi | Histogram | Transparan | LBP | Filter | Color Detection

Browse | Sobel | Laplace | Canny | Hough | Gradient

Horizontal | Vertical | Diagonal | Threshold | Hough

Horizontal | Vertical | Diagonal | Threshold | Hough

ImageViewer1.Form1

```
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;



namespace ImageViewer1
{

    public partial class Form1 : Form...
}
```

ImageViewer1.Class1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

//tambahan
using System.Drawing; //untuk menggambar



namespace ImageViewer1
{
    class Class1...
}
```

```
namespace ImageViewer1
{
    class Class1
    {

        public int[,] matrix = new int[3, 3];
        public int[,] sobelX = new int[3, 3];
        public int[,] sobelY = new int[3, 3];
        public int[,] laplace1 = new int[3, 3];
        public int[,] lowPass = new int[3, 3];
        public int[,] highPass = new int[3, 3];
        public double[,] bandStop = new double[3, 3];
        public double[,] canny = new double[5, 5];
        public void testing()...
        public void Inisialisasi()...
        public double[,] getGaussian() {return canny;}
        public int[,] getSobel(int i)...
        public double[,] getFilter(int i)...
        public int getValueKonvolusi(int[,] filter, Bitmap bmp1, int x, int y)...
        public int getValueKonvolusi1(int[,] filter, Bitmap bmp1, int x, int y)...
        public int getValueKonvolusi2(double[,] filter, Bitmap bmp1, int x, int y)...
        public double getValueKonvolusi3(double[,] filter, Bitmap bmp1, int x, int y)...
        public int getMagnitude(Bitmap bmp2a, Bitmap bmp2b, int x, int y)...
        public int[,] getMatrix(Bitmap bmp2, int[,] matrix, int x, int y, int i)...
```

```
 public void Inisialisasi()
{
    matrix[0, 0] = 0; matrix[0, 1] = 0; matrix[0, 2] = 0;
    matrix[1, 0] = 0; matrix[1, 1] = 0; matrix[1, 2] = 0;
    matrix[2, 0] = 0; matrix[2, 1] = 0; matrix[2, 2] = 0;

    sobelX[0, 0] = -3;  sobelX[0, 1] = 0; sobelX[0, 2] = 3;
    sobelX[1, 0] = -10; sobelX[1, 1] = 0; sobelX[1, 2] = 10;
    sobelX[2, 0] = -3;  sobelX[2, 1] = 0; sobelX[2, 2] = 3;

    sobelY[0, 0] = -3; sobelY[0, 1] = -10; sobelY[0, 2] = -3;
    sobelY[1, 0] = 0; sobelY[1, 1] = 0; sobelY[1, 2] = 0;
    sobelY[2, 0] = 3; sobelY[2, 1] = 10; sobelY[2, 2] = 3;

    laplace1[0, 0] =-1;  laplace1[0, 1] =-1;  laplace1[0, 2] =-1;
    laplace1[1, 0] =-1;  laplace1[1, 1] =8;   laplace1[1, 2] =-1;
    laplace1[2, 0] =-1;  laplace1[2, 1] =-1;  laplace1[2, 2] =-1;

    canny[0, 0] = 2.0f; canny[1, 0] = 4.0f; canny[2, 0] = 5.0f; canny[3, 0] = 4.0f; canny[4, 0] = 2.0f;
    canny[0, 1] = 4.0f; canny[1, 1] = 9.0f; canny[2, 1] = 12.0f; canny[3, 1] = 9.0f; canny[4, 1] = 4.0f;
    canny[0, 2] = 5.0f; canny[1, 2] = 12.0f; canny[2, 2] = 15.0f; canny[3, 2] = 12.0f; canny[4, 2] = 5.0f;
    canny[0, 3] = 4.0f; canny[1, 3] = 9.0f; canny[2, 3] = 12.0f; canny[3, 3] = 9.0f; canny[4, 3] = 4.0f;
    canny[0, 4] = 2.0f; canny[1, 4] = 4.0f; canny[2, 4] = 5.0f; canny[3, 4] = 4.0f; canny[4, 4] = 2.0f;
}
```

```
public int[,] matrix = new int[3, 3];
public int[,] sobelX = new int[3, 3];
public int[,] sobelY = new int[3, 3];
public int[,] laplace1 = new int[3, 3];
public double[,] canny = new double[5, 5];
```

```
public int[,] getSobel(int i)
    {
        int[,] kernelFilter = new int[3, 3];
        switch (i)
        {
            case 0: kernelFilter = sobelX; break;
            case 1: kernelFilter = sobelY; break;
            case 2: kernelFilter = laplace1; break;
            case 3: kernelFilter = lowPass; break;
            case 4: kernelFilter = highPass; break;
        }
        return kernelFilter;
    }
```

```
int[,] filterX = new int[3, 3];
int[,] filterY = new int[3, 3];
Class1 action = new Class1();
action.Inisialisasi();
filterX = action.getSobel(0);
filterY = action.getSobel(1);
```

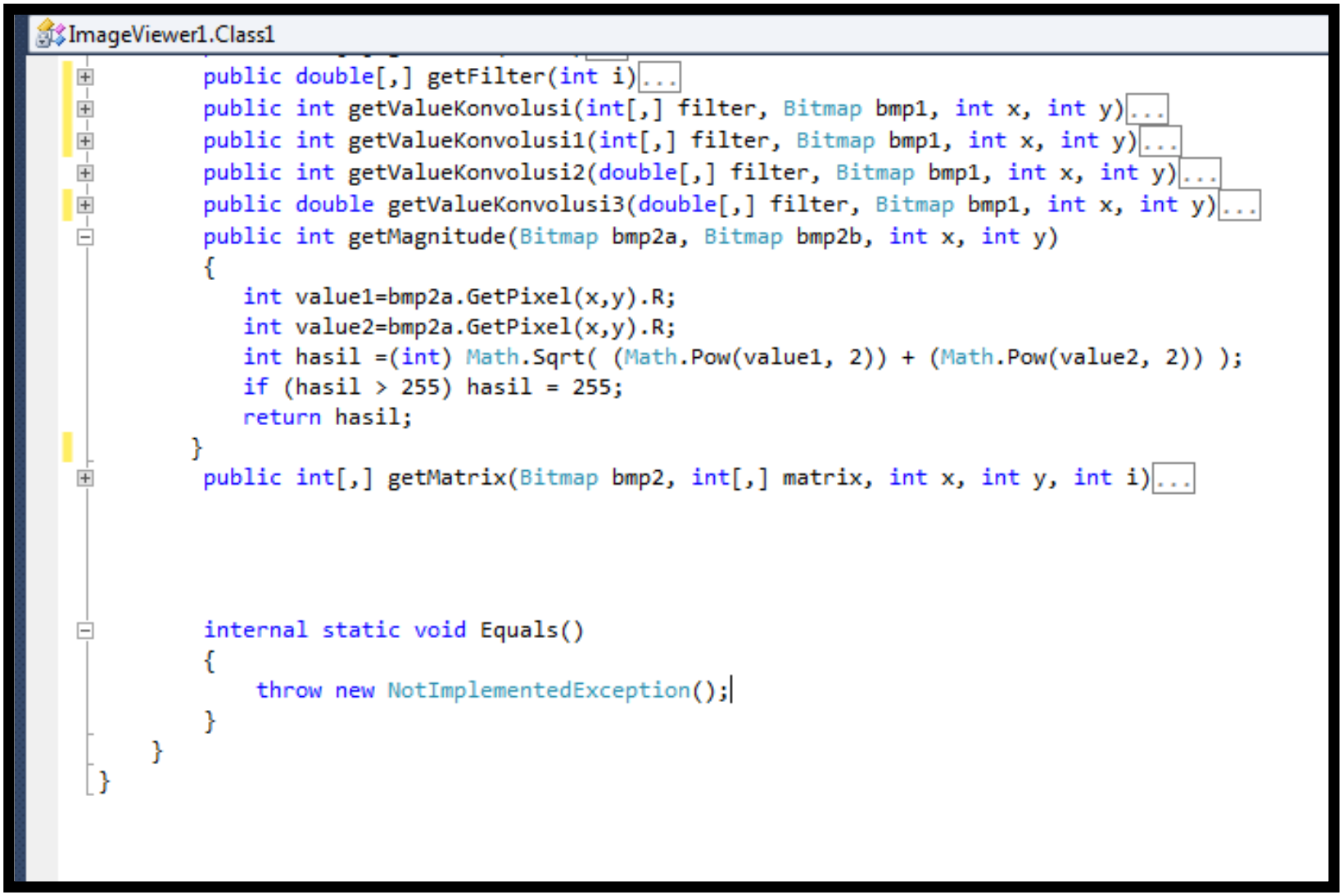
```
//convert to Gray
        for (int y = 0; y < bmp1.Height; y++){
           for (int x = 0; x < bmp1.Width; x++) {
              pixelColor1 = bmp.GetPixel(x, y);
              int red = pixelColor1.R;
              int green = pixelColor1.G;
              int blue = pixelColor1.B;
              int rata = (int)(red + green + blue) / 3;
              bmp1.SetPixel(x, y, Color.FromArgb(rata, rata, rata));
           }
        }
```

```
//konvolusi filer_X
        bmp2 = bmp1;
        for (int y = 1; y <bmp1a.Height-1; y++)
        {
           for (int x = 1; x < bmp1a.Width-1; x++)
           {
              int hasil=action.getValueKonvolusi(filterX ,bmp2,x,y);
              bmp1a.SetPixel(x, y, Color.FromArgb(hasil,hasil, hasil));
           }
        }
```

```
public int getValueKonvolusi(int[,] filter, Bitmap bmp1, int x, int y)
      { int value=0;
       value +=filter[0,0] * bmp1.GetPixel(x-1,y-1).R;
       value +=filter[0,1] * bmp1.GetPixel(x,y-1).R;
       value +=filter[0,2] * bmp1.GetPixel(x+1,y-1).R;

       value += filter[1, 0] * bmp1.GetPixel(x-1,y).R;
       value += filter[1, 1] * bmp1.GetPixel(x, y).R;
       value += filter[1, 2] * bmp1.GetPixel(x+1,y).R;

       value += filter[2, 0] * bmp1.GetPixel(x-1, y+1).R;
       value += filter[2, 1] * bmp1.GetPixel(x,y+1).R;
       value += filter[2, 2] * bmp1.GetPixel(x+1,y+1).R;
       if (value < 0) value = 0;
       if (value > 255) value = 255;

        return value;
      }
```

```
        //konvolusi filer_Y
bmp3 = bmp1;
for (int y = 1; y < bmp1b.Height - 1; y++)
{
   for (int x = 1; x < bmp1b.Width - 1; x++)
   {
      int hasil = action.getValueKonvolusi(filterY, bmp3, x, y);
      bmp1b.SetPixel(x, y, Color.FromArgb(hasil, hasil, hasil));
   }
}
```

```
//Sobel
        bmp2a = bmp1a;
        bmp2b = bmp1b;
        for (int y = 1; y < bmp4.Height - 1; y++)
        {
           for (int x = 1; x < bmp4.Width - 1; x++)
           {
              int hasil = action.getMagnitude(bmp2a, bmp2b, x, y);
              bmp4.SetPixel(x, y, Color.FromArgb(hasil, hasil, hasil));
           }
        }
```


ImageViewer1.Class1
public double[,] getFilter(int i)...
public int getValueKonvolusi(int[,] filter, Bitmap bmp1, int x, int y)...
public int getValueKonvolusi1(int[,] filter, Bitmap bmp1, int x, int y)...
public int getValueKonvolusi2(double[,] filter, Bitmap bmp1, int x, int y)...
public double getValueKonvolusi3(double[,] filter, Bitmap bmp1, int x, int y)...
public int getMagnitude(Bitmap bmp2a, Bitmap bmp2b, int x, int y)
{
int value1=bmp2a.GetPixel(x,y).R;
int value2=bmp2a.GetPixel(x,y).R;
int hasil =(int) Math.Sqrt((Math.Pow(value1, 2)) + (Math.Pow(value2, 2)));
if (hasil > 255) hasil = 255;
return hasil;
}
public int[,] getMatrix(Bitmap bmp2, int[,] matrix, int x, int y, int i)...
internal static void Equals()
{
throw new NotImplementedException();
}
}
}

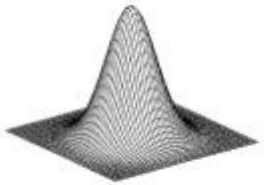
Laplacian

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

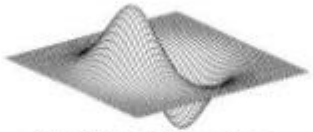
1	-2	1
-2	4	-2
1	-2	1

2D edge detection filters



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

0	–1	0
–1	4	–1
0	–1	0

–1	–1	–1
–1	8	–1
–1	–1	–1

Form1

File

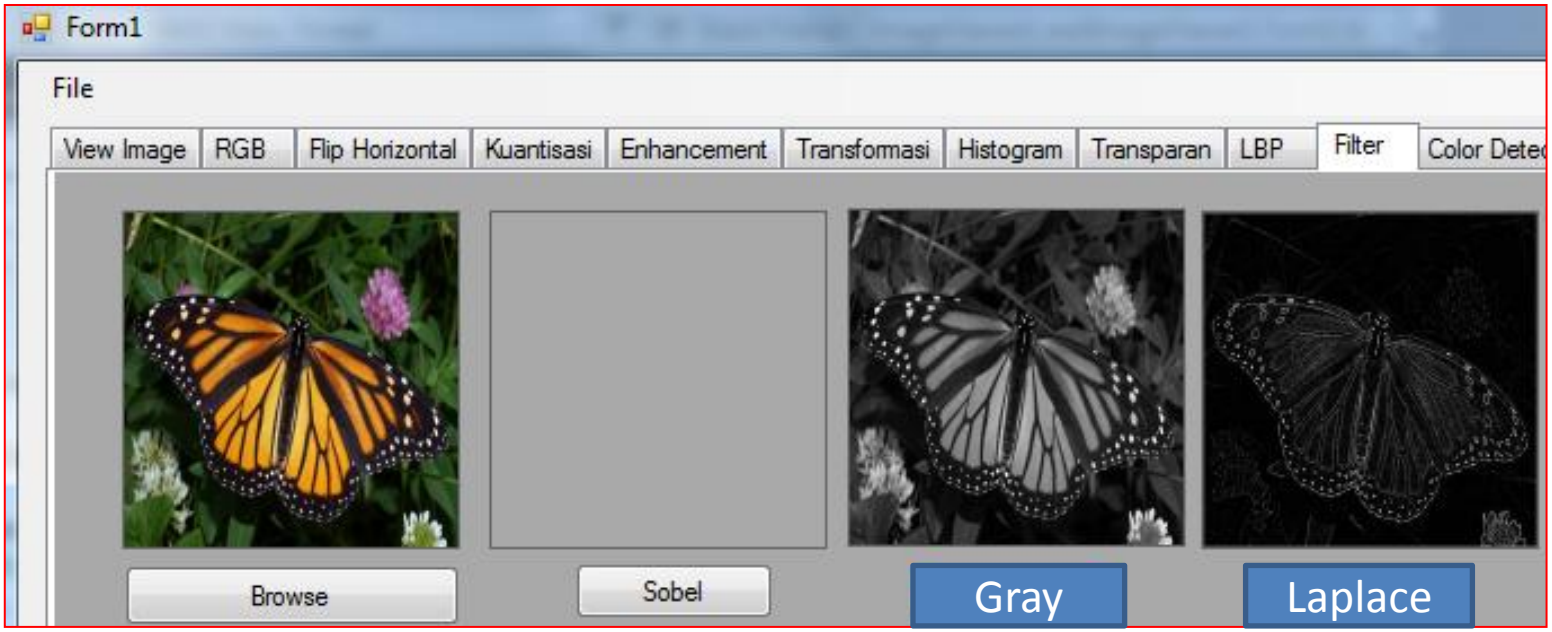
View Image | RGB | Flip Horizontal | Kuantisasi | Enhancement | Transformasi | Histogram | Transparan | LBP | Filter | Color Detection

Browse | Sobel | Laplace | Gradient

Horizontal | Vertical | Diagonal | Threshold | Hough

Horizontal | Vertical | Diagonal | Threshold | Hough

Laplacian


Form1
File
View Image
RGB
Flip Horizontal
Kuantisasi
Enhancement
Transformasi
Histogram
Transparan
LBP
Filter
Color Dete
Browse
Sobel
Gray
Laplace

```
//convert to Gray
    for (int y = 0; y < bmp1.Height; y++)
    {
        for (int x = 0; x < bmp1.Width; x++)
        {
            pixelColor1 = bmp1.GetPixel(x, y);
            int red = pixelColor1.R;
            int green = pixelColor1.G;
            int blue = pixelColor1.B;
            int rata = (int)(red + green + blue) / 3;
            bmp1.SetPixel(x, y, Color.FromArgb(rata, rata, rata));
        }
    }
```

```
bmp2 = bmp1;
    for (int y = 1; y < bmp2.Height - 1; y++)
    {
        for (int x = 1; x < bmp2.Width - 1; x++)
        {
            int hasil = action.getValueKonvolusi(filterX, bmp2, x, y);
            //int hasil = 100;
            bmp1a.SetPixel(x, y, Color.FromArgb(hasil, hasil, hasil));
        }
    }
```

Canny

Process of Canny edge detection algorithm [edit]

The Process of Canny edge detection algorithm can be broken down to 5 different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

https://en.wikipedia.org/wiki/Canny_edge_detector

Canny

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

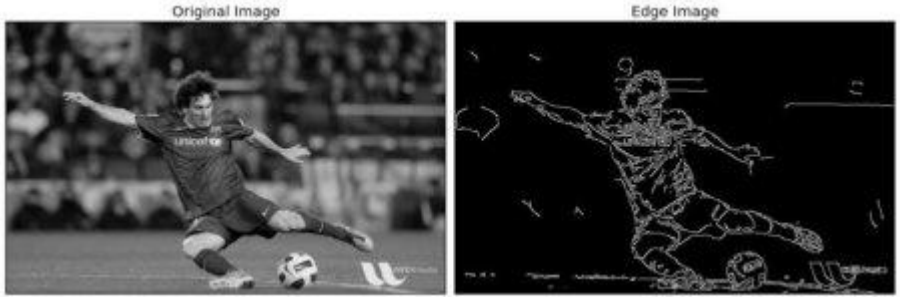
$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Canny

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Canny



Filter Kernel

- Filter kernel H adalah suatu matrik yang menyatakan model filter (dalam spacial) yang menjadi operator dalam proses filter pada gambar.
- Bentuk atau komposisi nilai yang ada di dalam filter kernel menunjukkan jenis filter yang digunakan.

$$H = \frac{1}{12}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

LOW PASS FILTER

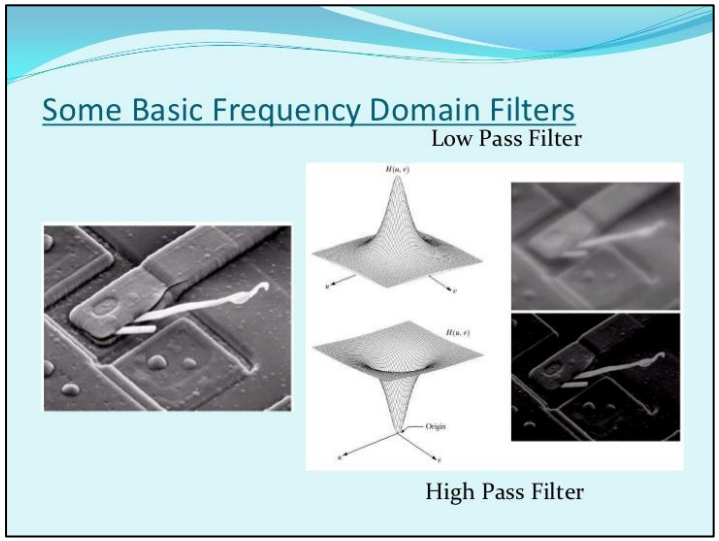
$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 3 \\ -3 & 0 & 1 \end{bmatrix}$$

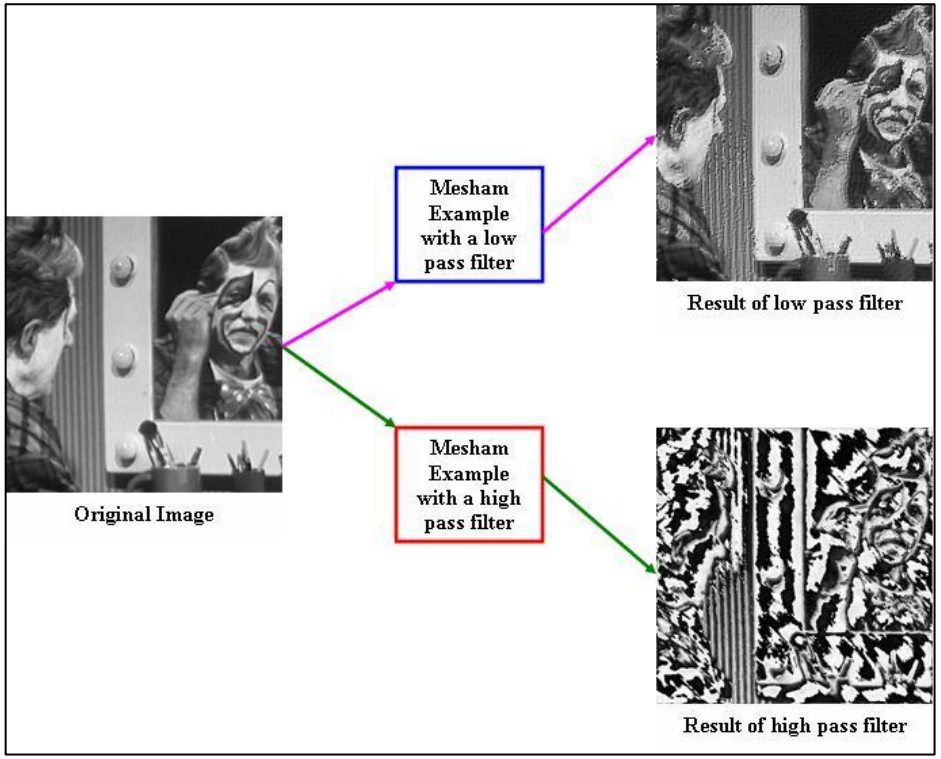
HIGH PASS FILTER

$$H = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 0.5 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

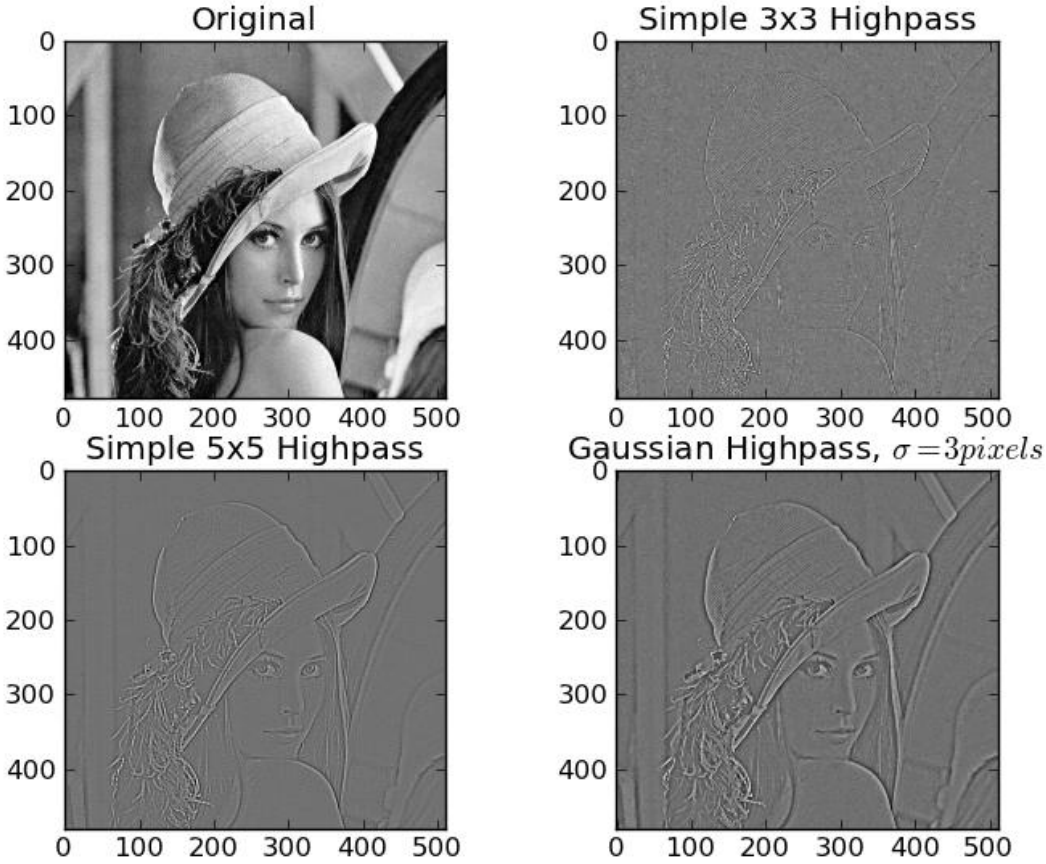
BAND STOP FILTER

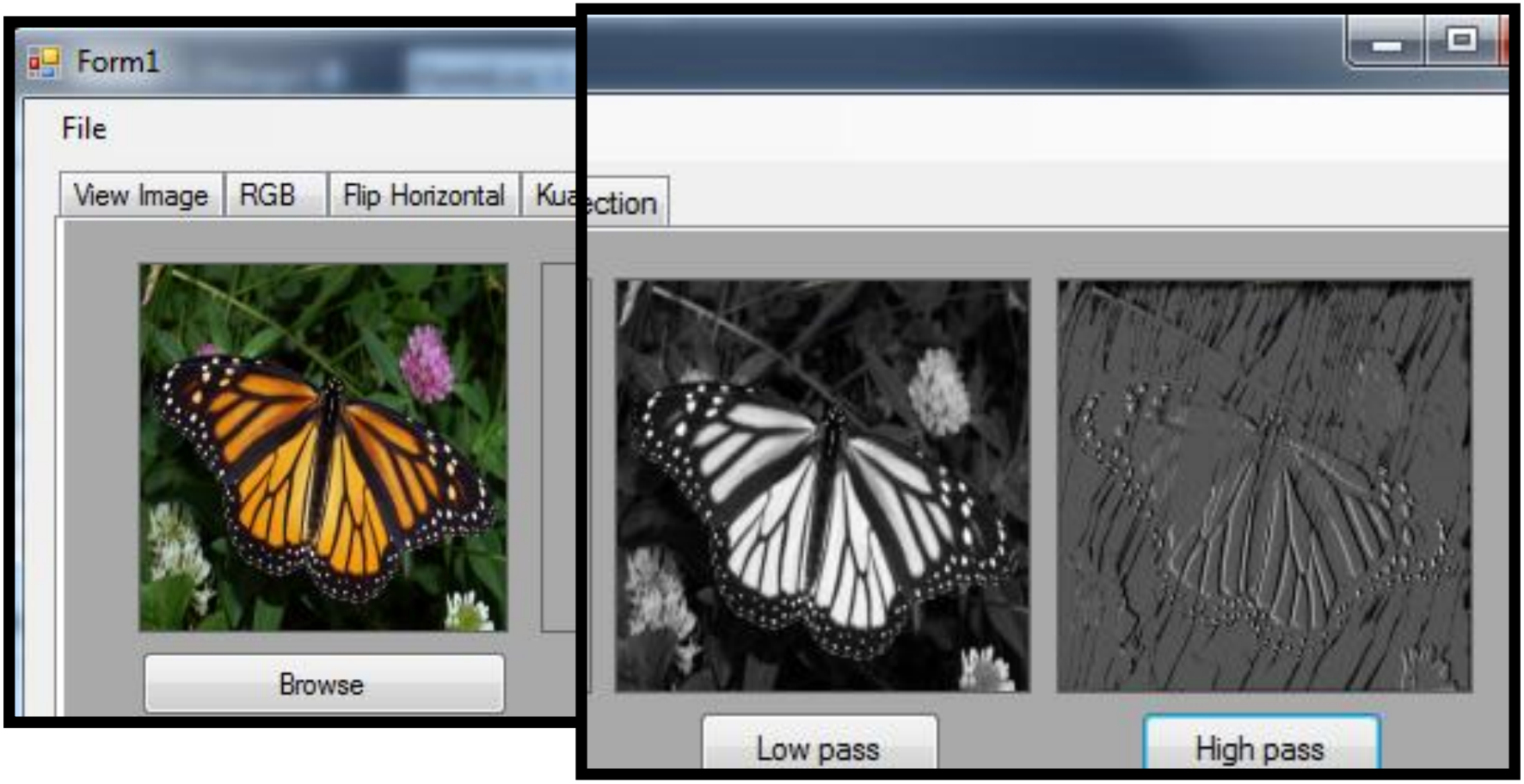
Low/High Pass Filter





Low/High Pass Filter




Form1
File
View Image
RGB
Flip Horizontal
ction
Browse
Low pass
High pass

Form1

File

View Image | RGB | Flip Horizontal | Kuantisasi | Enhancement | Transformasi | Histogram | Transparan | LBP | Filter | Color Detection

Browse

GO

Filter Rata-rata

- Filter Rata-rata
- Filter Gaussian
- Filter Median
- Noise Gaussian
- Noise Speckle
- Noise Salt & Pepper
- Deteksi Tepi Robert
- Deteksi Tepi Prewitt
- Sharpness

Sobel | Laplace | Canny | Low pass | High pass

Horizontal | Vertical | Fourier | Wavelett | Band stop

Noise