# TUGAS PERCOBAAN 4
# PENGOLAHAN CITRA
# MK401

**Disusun oleh :**

**Ricky Silitonga (4211901034)**

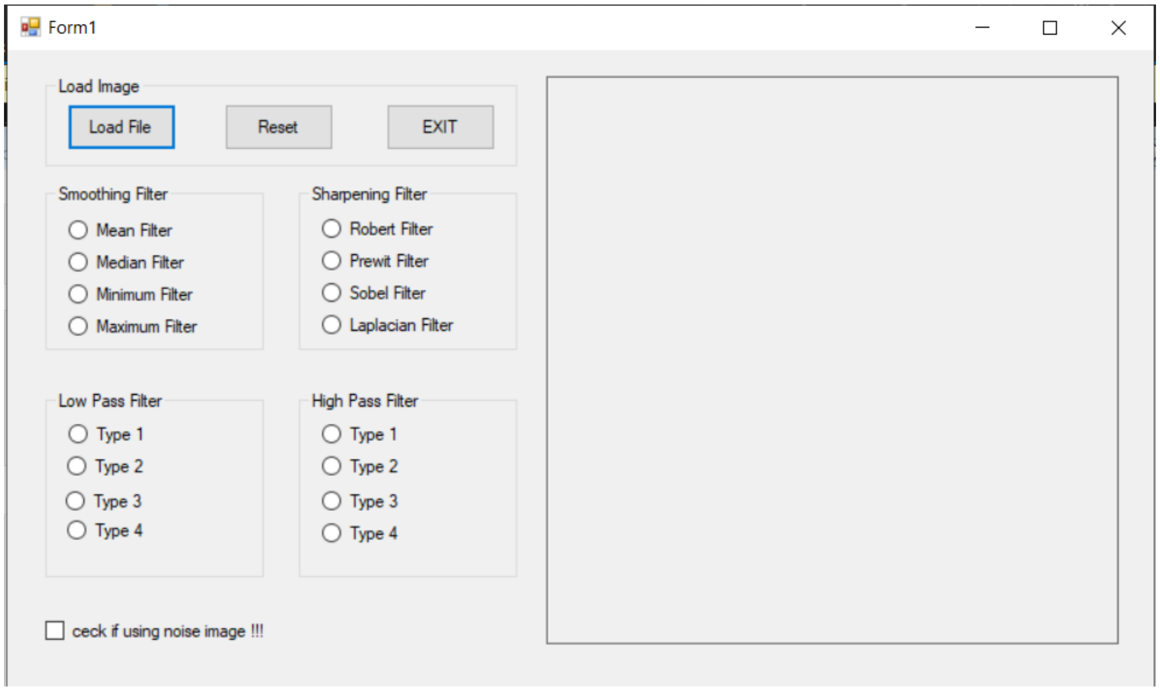
**PROGRAM STUDI TEKNIK MEKATRONIKA**

**JURUSAN TEKNIK ELEKTRO**

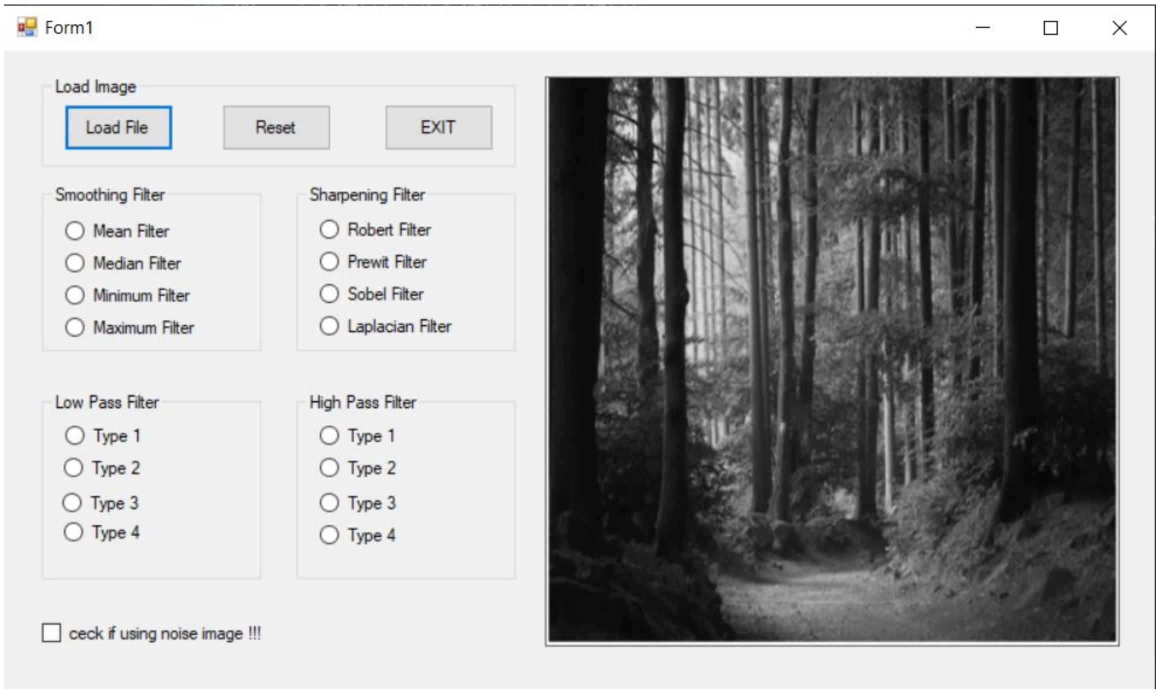
**POLITEKNIK NEGERI BATAM**

**2020**

**Image Filtering**

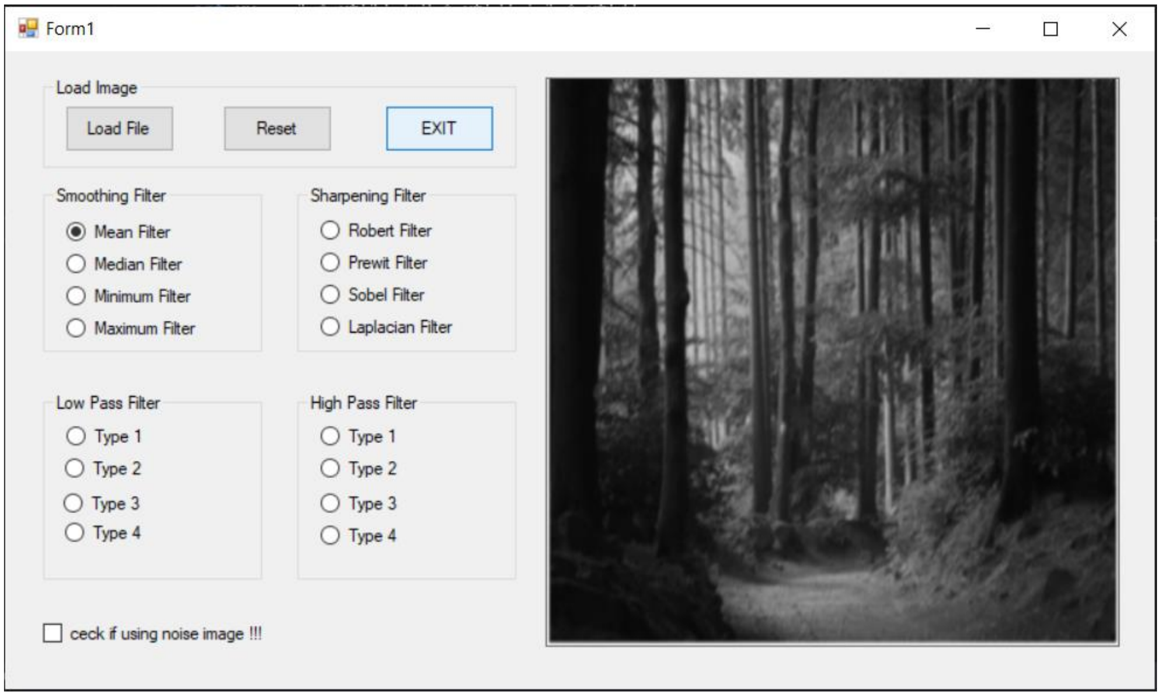


Tampilan Awal

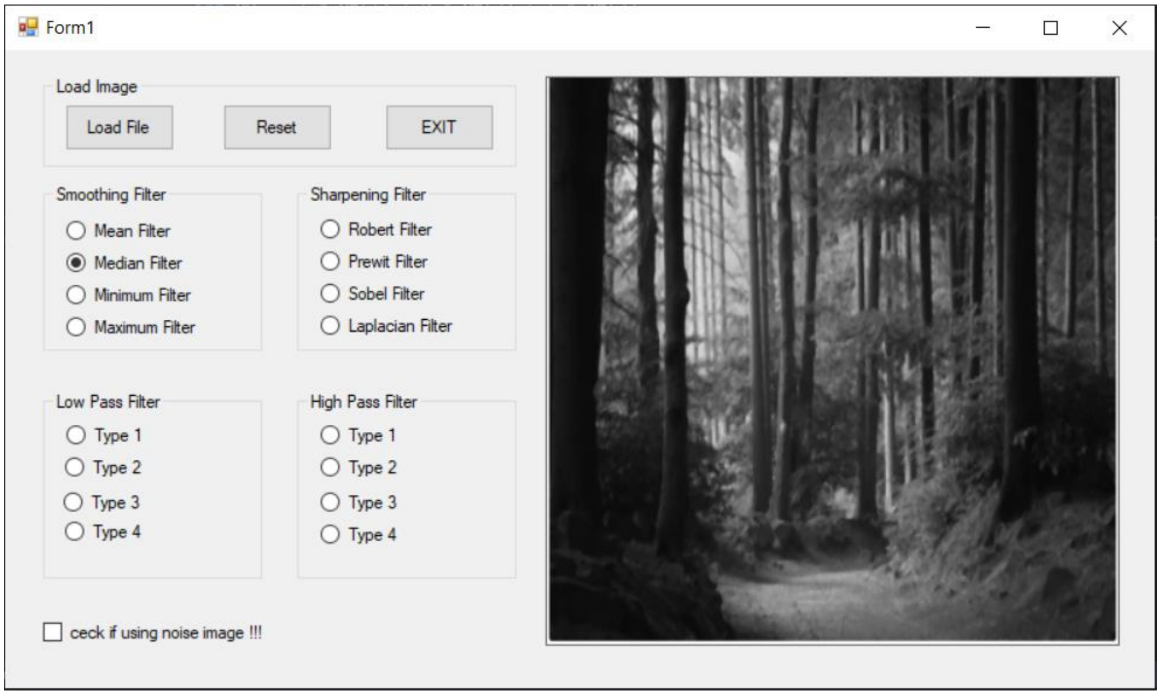


Load Image

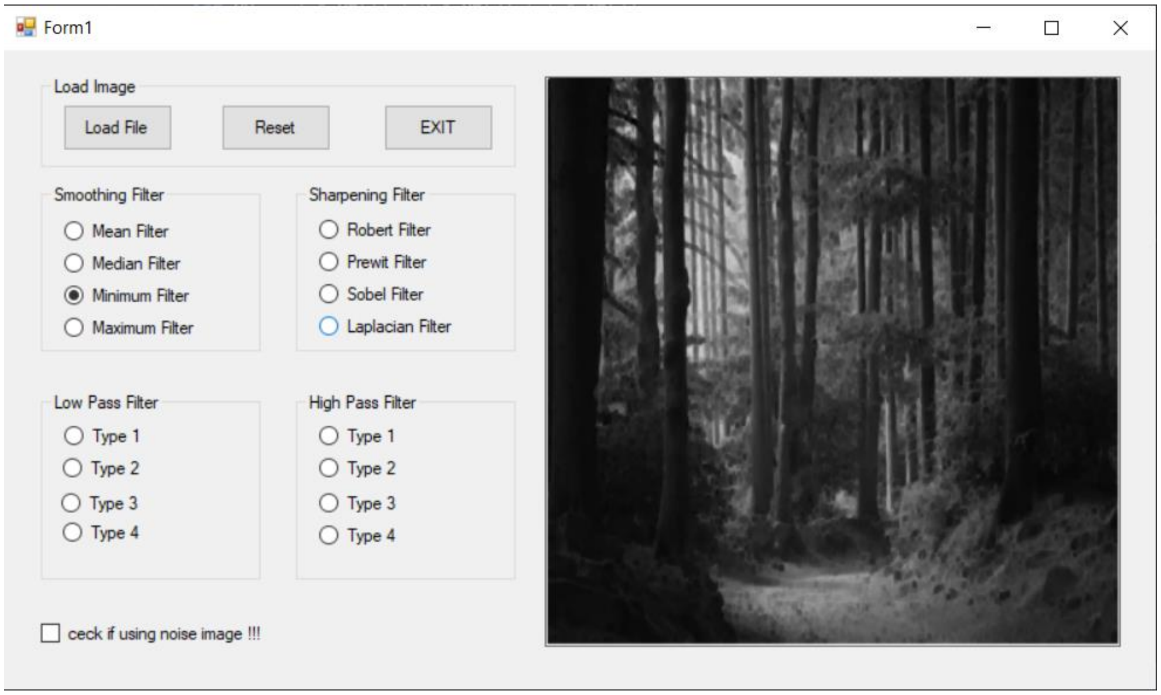
## 1. Smoothing Filter



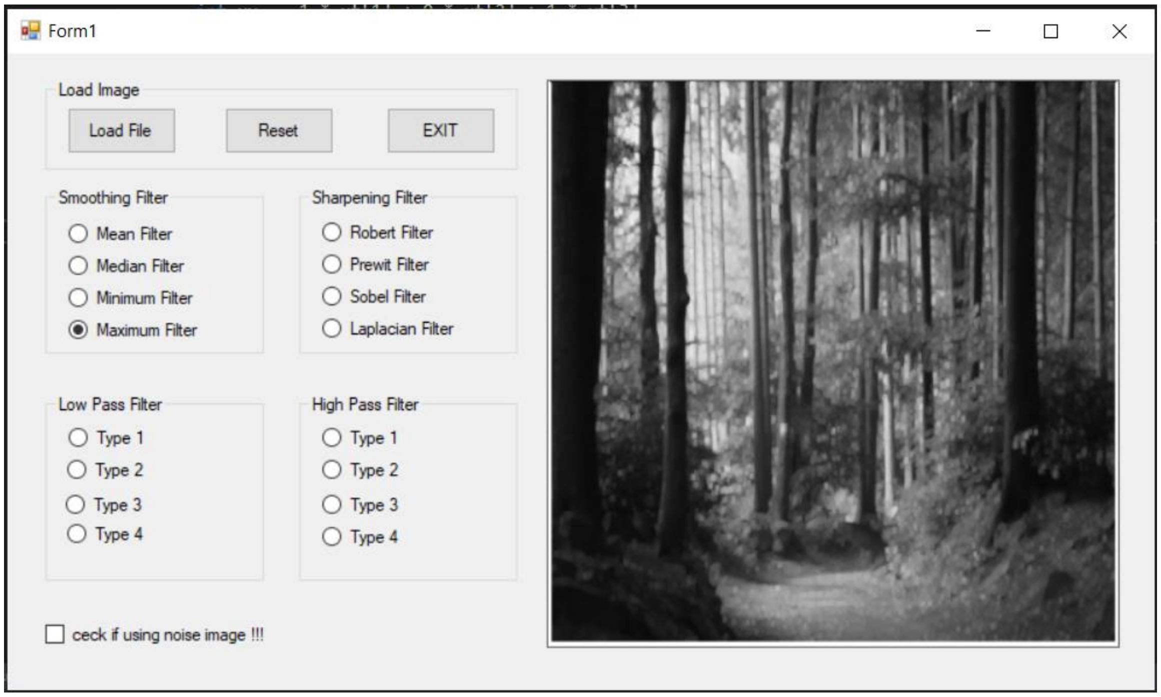
Mean Filter



Median Filter

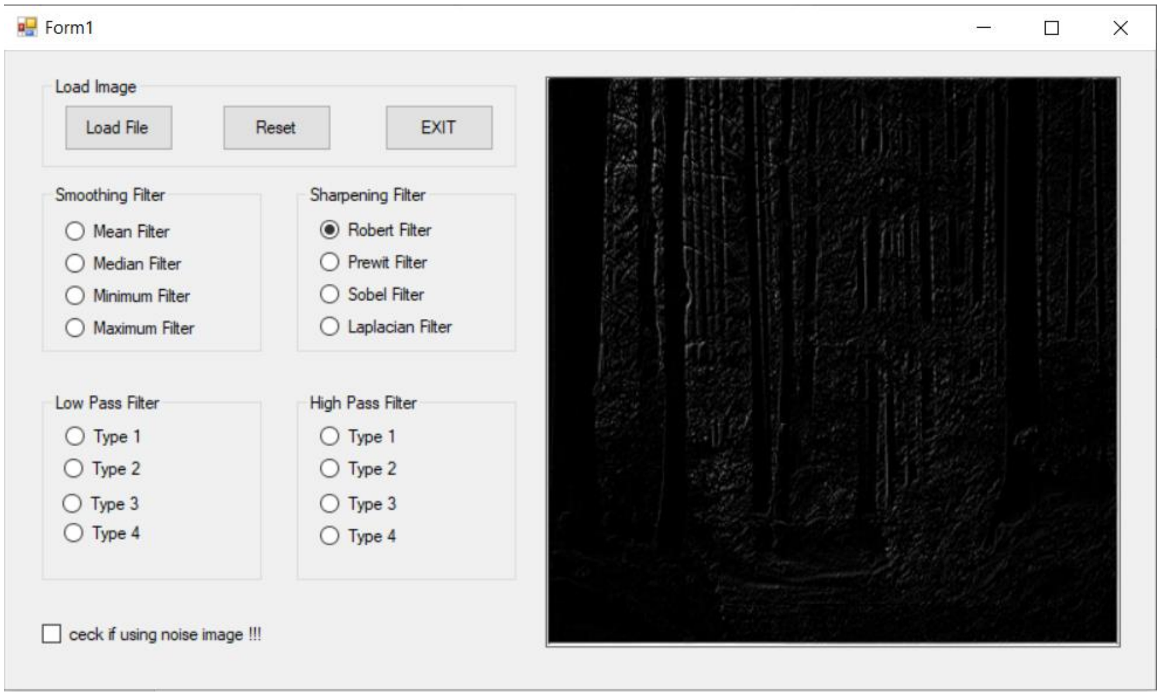


Minimum Filter



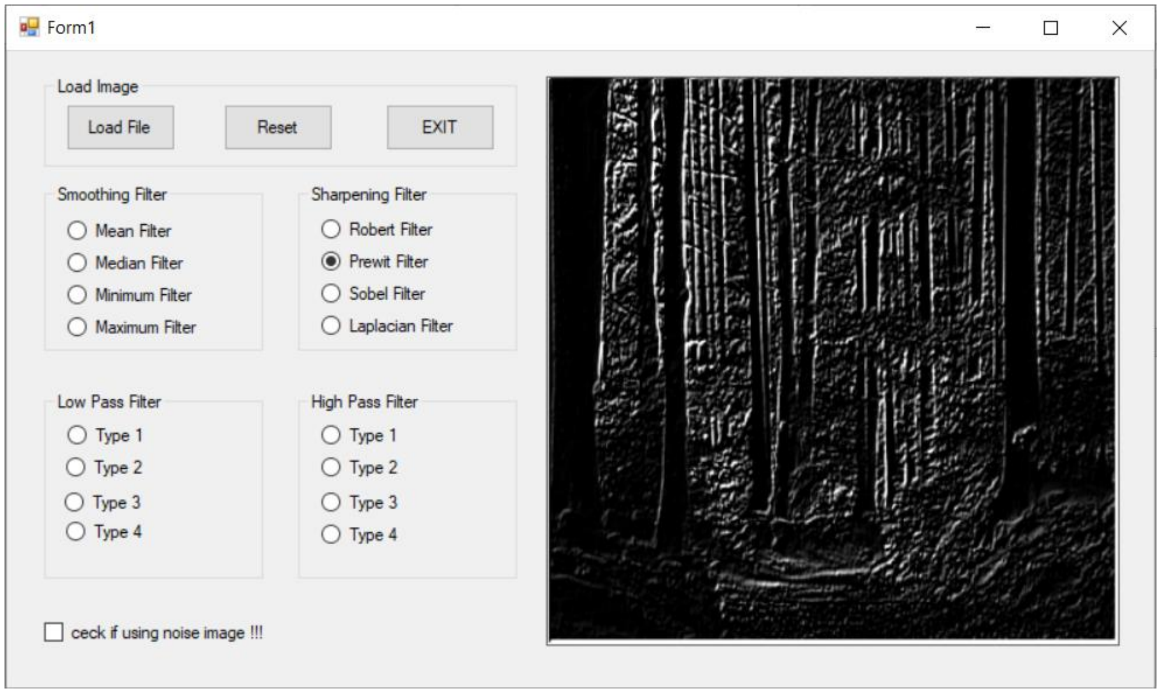
Maximum Filter

## 2. Sharpening Filter



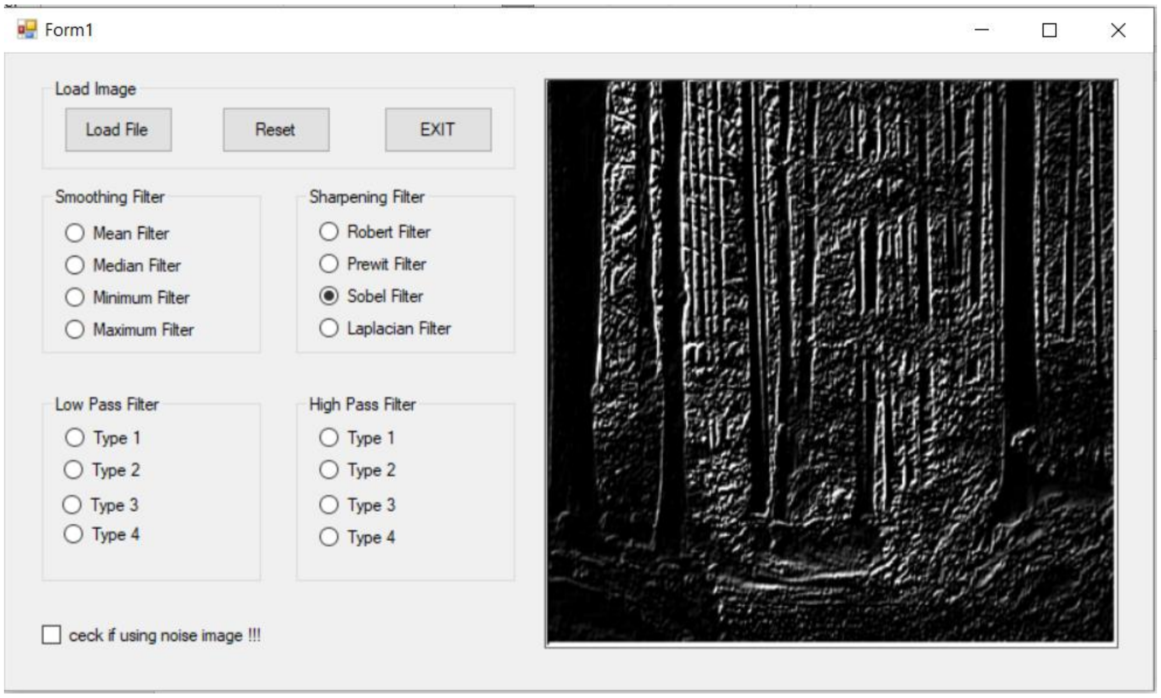
Sharpening Filter: [✓ Robert Filter] Prewit Filter Sobel Filter Laplacian Filter

Robert Filter



Sharpening Filter: Robert Filter [✓ Prewit Filter] Sobel Filter Laplacian Filter

Prewit Filter


Form1

Load Image

Load File | Reset | EXIT

Smoothing Filter
- Mean Filter
- Median Filter
- Minimum Filter
- Maximum Filter

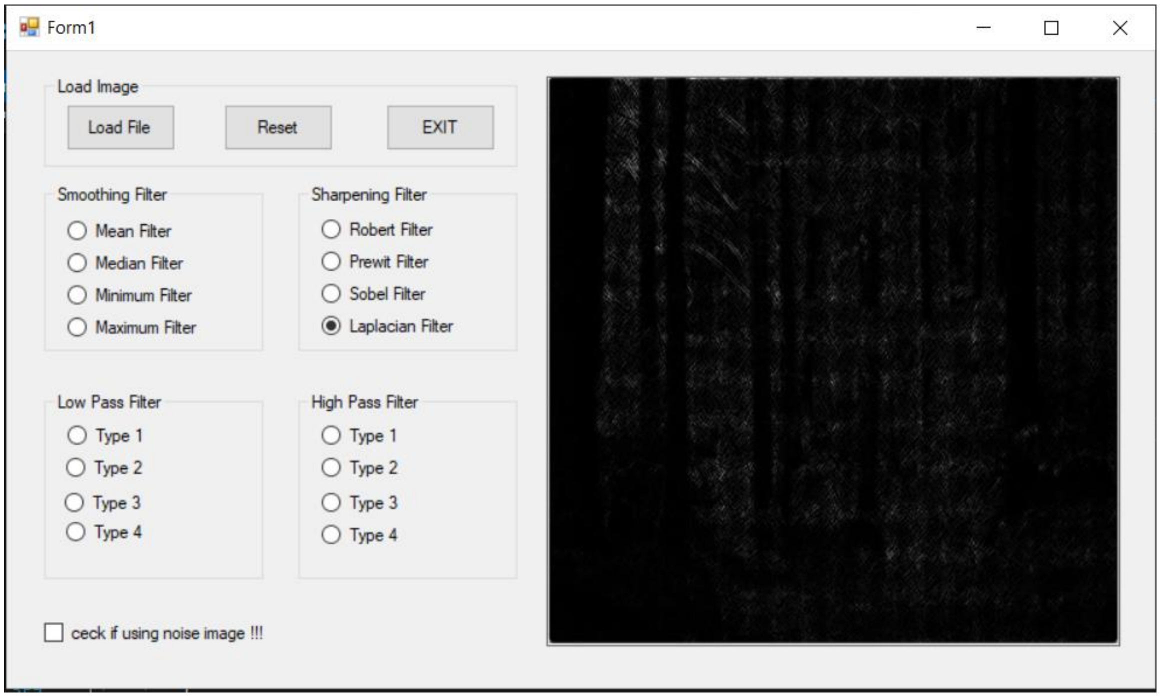
Sharpening Filter
- Robert Filter
- Prewit Filter
- [✓ Sobel Filter]
- Laplacian Filter

Low Pass Filter
- Type 1
- Type 2
- Type 3
- Type 4

High Pass Filter
- Type 1
- Type 2
- Type 3
- Type 4

☐ ceck if using noise image !!!

Sobel Filter


Form1

Load Image

Load File | Reset | EXIT

Smoothing Filter
- Mean Filter
- Median Filter
- Minimum Filter
- Maximum Filter

Sharpening Filter
- Robert Filter
- Prewit Filter
- Sobel Filter
- [✓ Laplacian Filter]

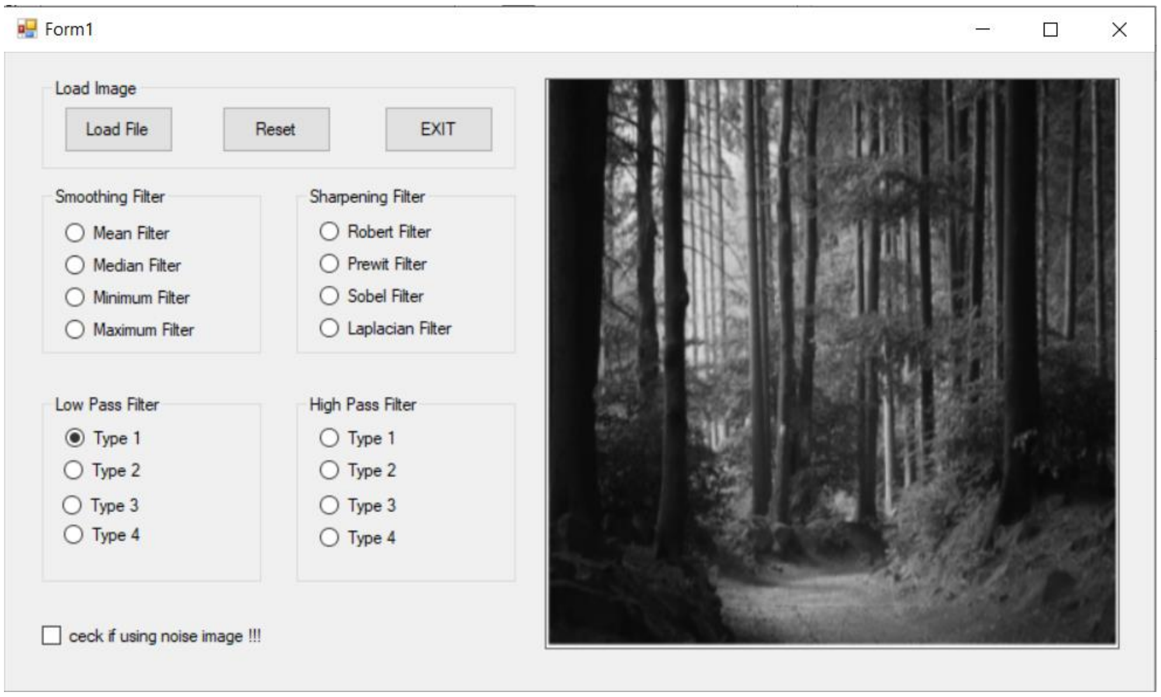
Low Pass Filter
- Type 1
- Type 2
- Type 3
- Type 4

High Pass Filter
- Type 1
- Type 2
- Type 3
- Type 4

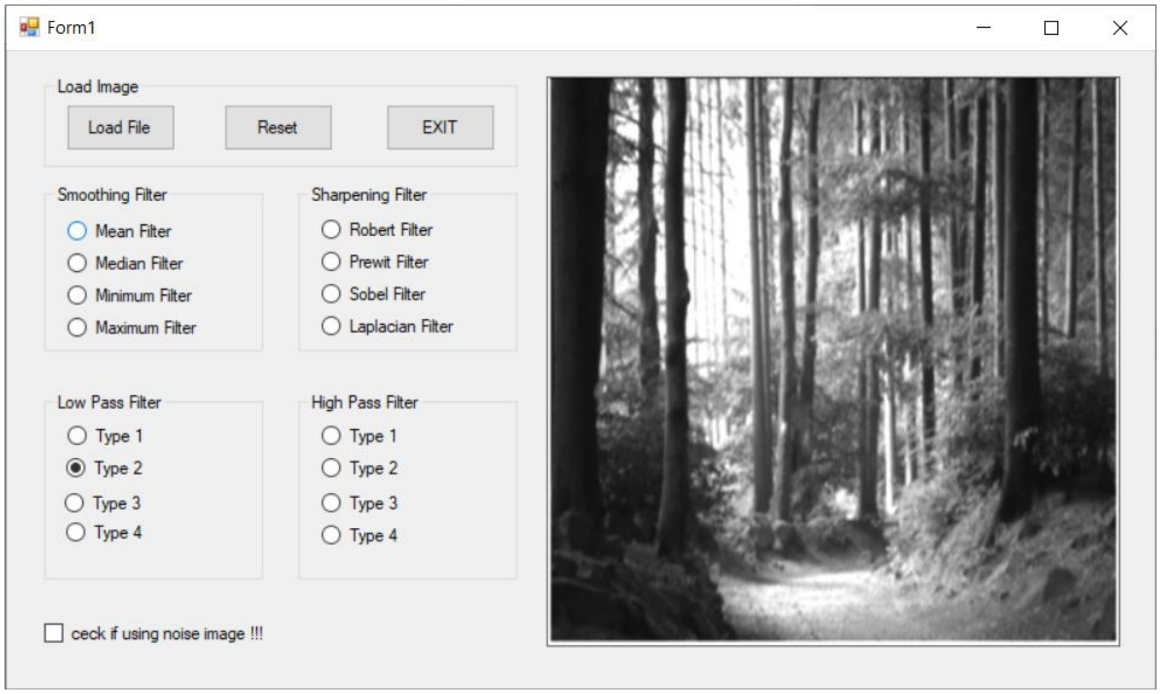
☐ ceck if using noise image !!!

Laplacian Filter

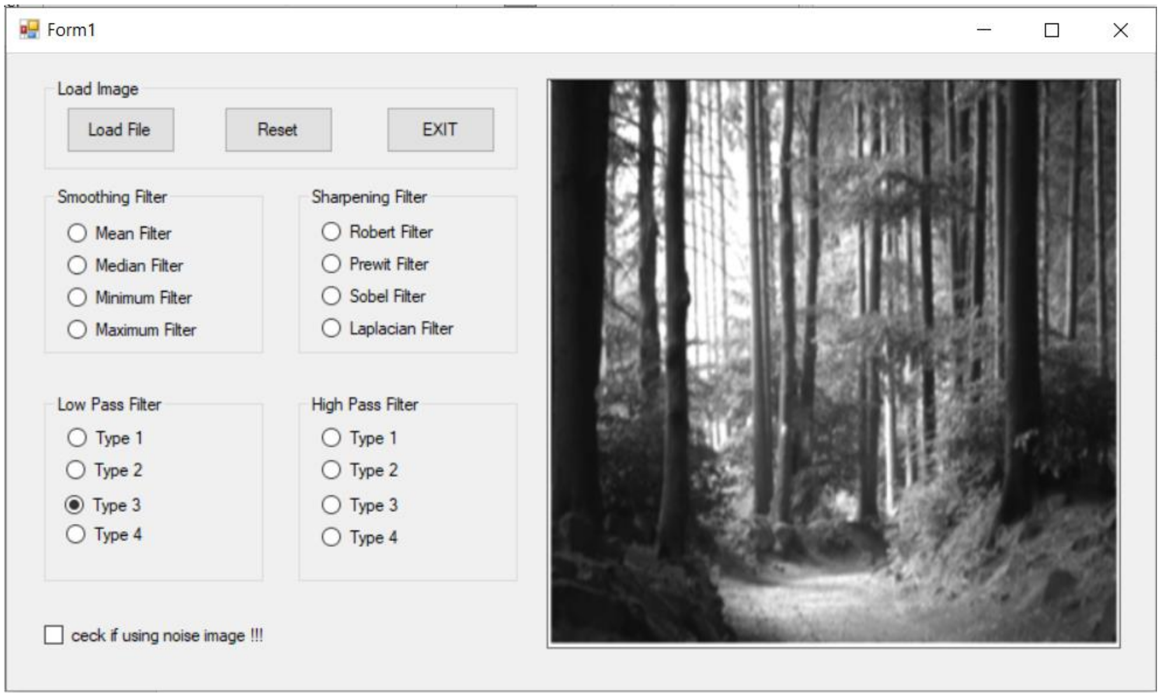
## 3. Low Pass Filter



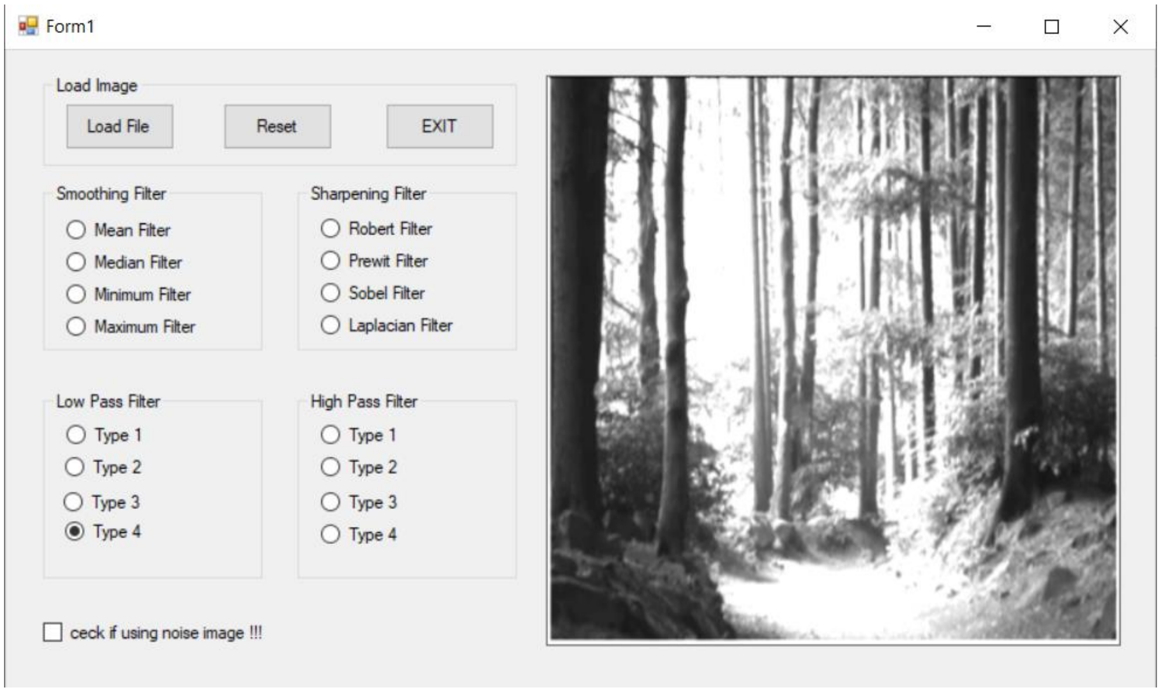
Low Pass Filter Type 1



Low Pass Filter Type 2

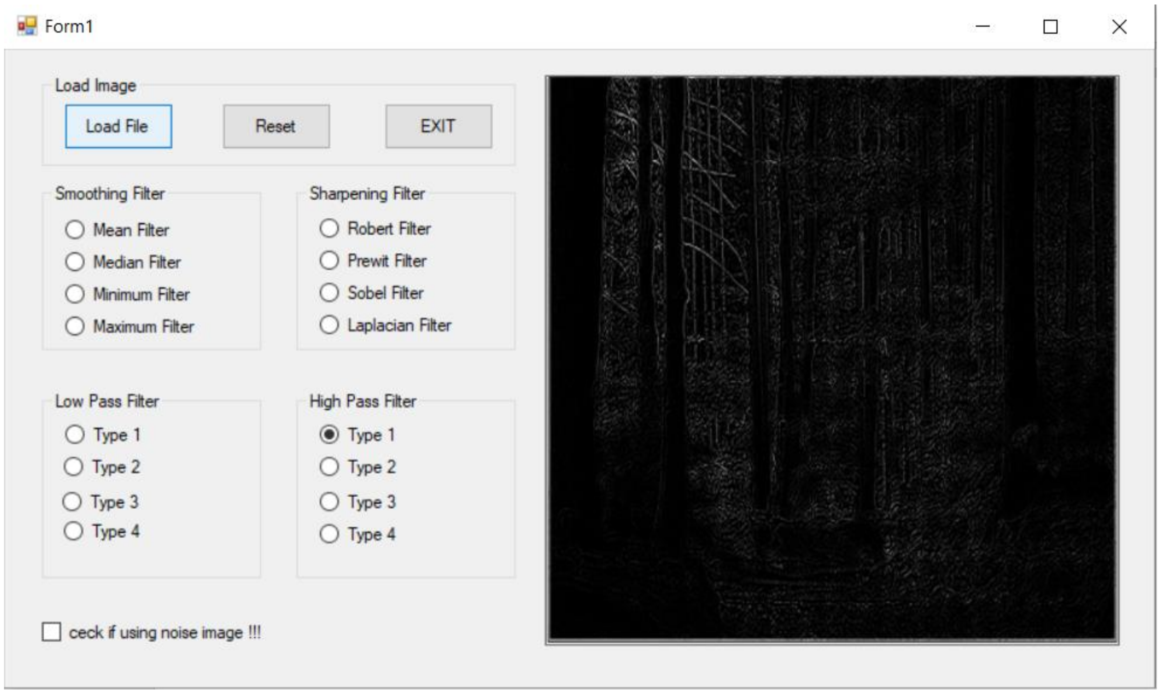


Low Pass Filter Type 3



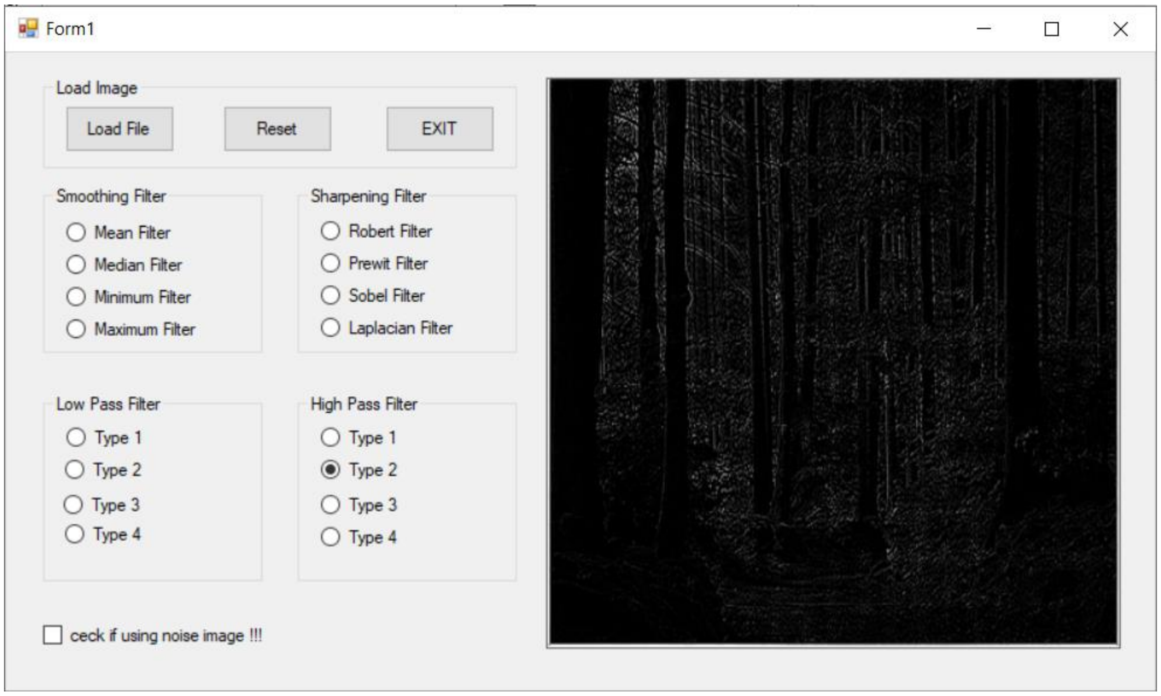
Low Pass Filter Type 3

## 4. High Pass Filter



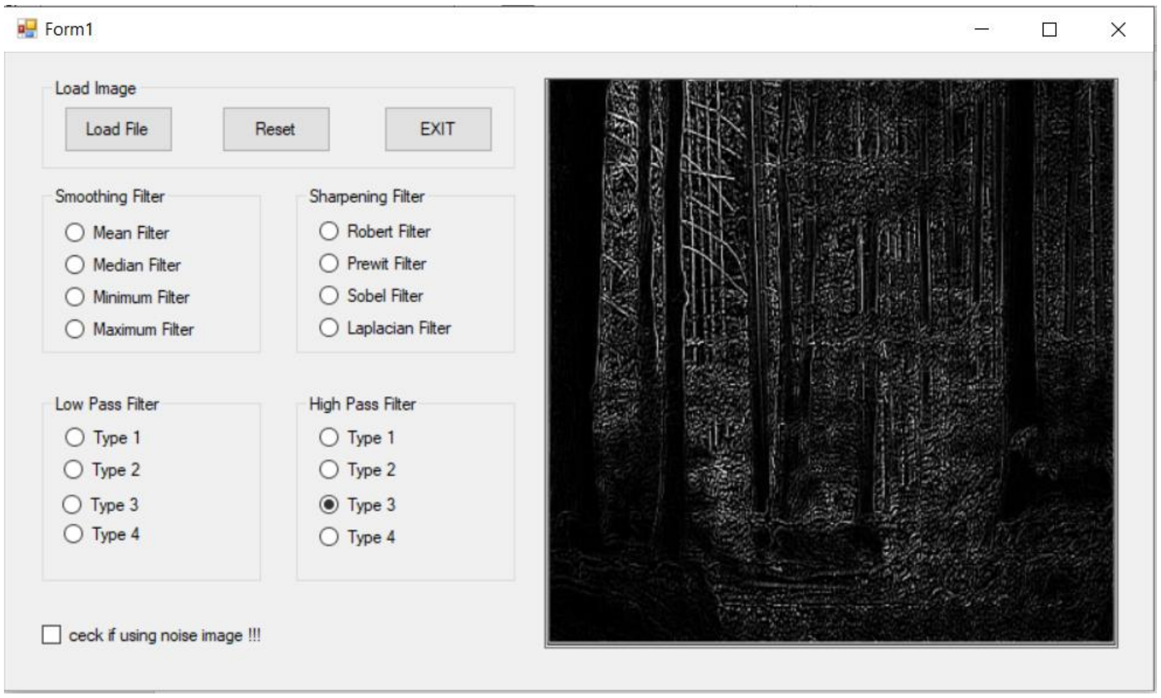
High Pass Filter: [✓ Type 1]

High Pass Filter Type 1



High Pass Filter: [✓ Type 2]

High Pass Filter Type 2



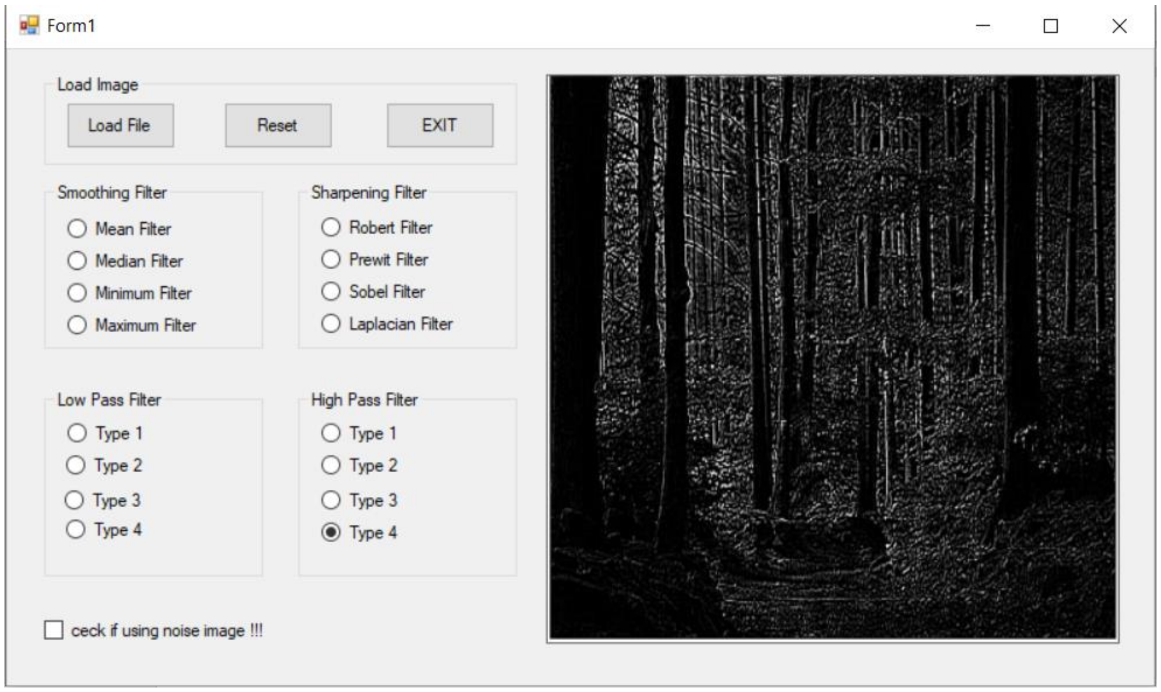
Smoothing Filter: Mean Filter, Median Filter, Minimum Filter, Maximum Filter

Sharpening Filter: Robert Filter, Prewit Filter, Sobel Filter, Laplacian Filter

Low Pass Filter: Type 1, Type 2, Type 3, Type 4

High Pass Filter: Type 1, Type 2, [✓ Type 3], Type 4

High Pass Filter Type 3



Smoothing Filter: Mean Filter, Median Filter, Minimum Filter, Maximum Filter

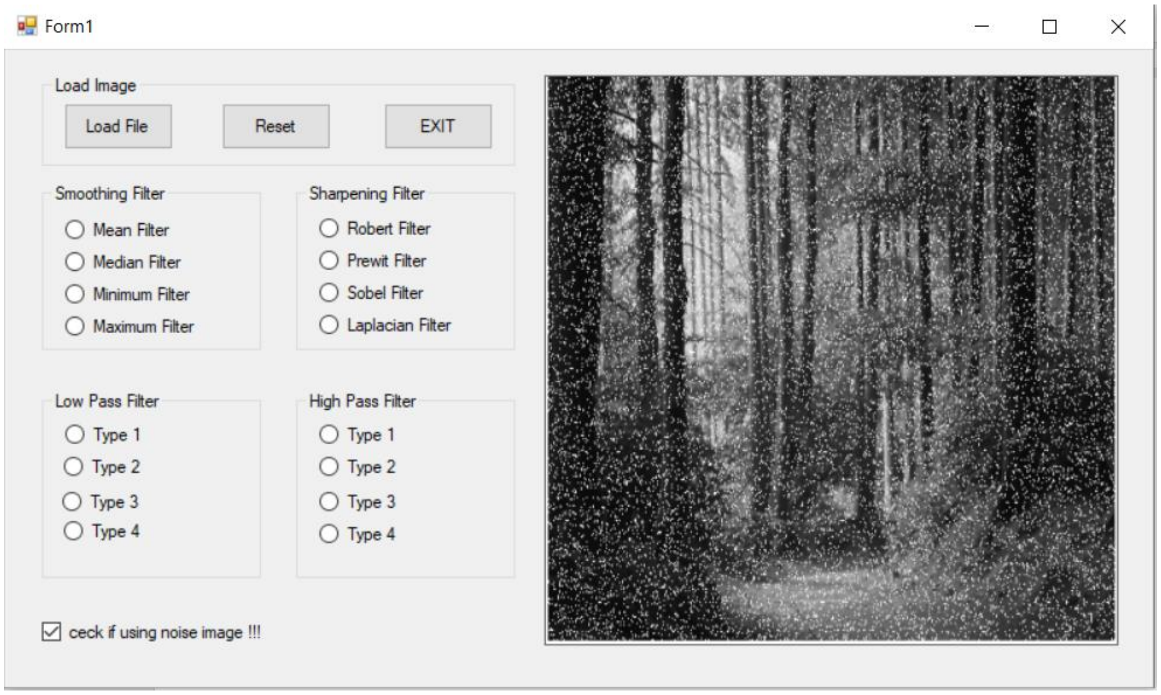
Sharpening Filter: Robert Filter, Prewit Filter, Sobel Filter, Laplacian Filter

Low Pass Filter: Type 1, Type 2, Type 3, Type 4

High Pass Filter: Type 1, Type 2, Type 3, [✓ Type 4]

High Pass Filter Type 4

## 5. Check Box



Checkbox True

**Sourcecode**

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

namespace Percobaan4_4211901034
{
    public partial class Form1 : Form
    {
        // global variable
        Bitmap sourceImage; // rgb image
        Bitmap grayImage; // gray image without noise
        Bitmap noiseImage; // gray image with noise
```

```
int filterSmoothingType;
int filterSharpeningType;
int lowPassType;
int highPassType;

public Form1()
{
   InitializeComponent();
}

private void groupBox5_Enter(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
   if(openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      // loading source image
      sourceImage = (Bitmap)Bitmap.FromFile(openFileDialog1.FileName);

      // mengkonversi ke gray image
      grayImage = grayImaging(sourceImage);

      // menambahkan noise ke gray image
      noiseImage = noiseImaging(grayImage);

      // original image ditampilkan dalam bentuk gray image
      pictureBox1.Image = grayImage;

      // reset
      resetRadioButtonSmoothing();
      resetRadioButtonSharpening();
      resetRadioButtonLowPass();
      resetRadioButtonHighPass();
      resetCeckBox();
   }
}

private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
{
}

// function
private Bitmap grayImaging(Bitmap image)
{
    Bitmap tempImage = new Bitmap(image);
    // grayscale convertion
    for(int x = 0; x < sourceImage.Width; x++)
        for(int y = 0; y < image.Height; y++)
        {
            Color w = image.GetPixel(x, y);
            int r = w.R; int g = w.G; int b = w.B;
            int xg = (int)((r + g + b) / 3);
            Color wb = Color.FromArgb(xg, xg, xg);

            tempImage.SetPixel(x, y, wb);
        }
    return tempImage;
}
private Bitmap noiseImaging(Bitmap image)
{
    noiseImage = new Bitmap(grayImage);
    int noiseProb = 10;
    Random r = new Random();
    for (int x = 0; x < grayImage.Width; x++)
        for (int y = 0; y < grayImage.Height; y++)
        {
            Color w = image.GetPixel(x, y);
            int xg = w.R;
            int xb = xg;
            //generate random number (0-100)
            int nr = r.Next(0, 100);
            //generationg 20% gaussian noise
            if (nr < noiseProb) xb = 255;
            Color wb = Color.FromArgb(xb, xb, xb);
            noiseImage.SetPixel(x, y, wb);
        }
    return noiseImage;
}
// smooting filter
private Bitmap smoothingfilter(int filterType)
{
```

```
Bitmap filteredImage = new Bitmap(noiseImage);
int[] xt = new int[10];
int xb = 0;
for (int x = 1; x < noiseImage.Width - 1; x++)
    for (int y = 1; y < noiseImage.Height - 1; y++)
    {
        Color w1 = noiseImage.GetPixel(x - 1, y - 1);
        Color w2 = noiseImage.GetPixel(x - 1, y);
        Color w3 = noiseImage.GetPixel(x - 1, y + 1);
        Color w4 = noiseImage.GetPixel(x, y - 1);
        Color w5 = noiseImage.GetPixel(x, y);
        Color w6 = noiseImage.GetPixel(x, y + 1);
        Color w7 = noiseImage.GetPixel(x + 1, y - 1);
        Color w8 = noiseImage.GetPixel(x + 1, y);
        Color w9 = noiseImage.GetPixel(x + 1, y + 1);

        xt[1] = w1.R; xt[2] = w2.R; xt[3] = w3.R;
        xt[4] = w4.R; xt[5] = w5.R; xt[6] = w6.R;
        xt[7] = w7.R; xt[8] = w8.R; xt[9] = w9.R;
        if (filterType == 1) //mean filter
        {
            xb = 0;
            for(int i = 1; i < 9; i++) {
                xb += xt[i];
            }
            xb = xb / 9;
        }
        else if (filterType == 2) //median filter
        {
            //looking for median
            for (int i = 1; i < 9; i++)
                for (int j = 1; j < 9; j++)
                {
                    if (xt[j] > xt[j + 1])
                    {
                        int a = xt[j];
                        xt[j] = xt[j + 1];
                        xt[j + 1] = a;
                    }
                }
            //the median
            xb = xt[5];
        }
```

```
            else if (filterType == 3) //minimum filter
            {
                int xMinimum = xt[1];//initialization
                                     //looking for minimum
                for (int i = 2; i < 10; i++)
                {
                    if (xt[i] < xMinimum)
                    {
                        xMinimum = xt[i];
                    }
                }
                xb = xMinimum;
            }
            else if (filterType == 4) //maximum filter
            {
                // max
                int xMax = xt[1];//initialization
                                     //looking for minimum
                for (int i = 2; i < 10; i++)
                {
                    if (xt[i] > xMax)
                    {
                        xMax = xt[i];
                    }
                }
                xb = xMax;
            }
            Color wb = Color.FromArgb(xb, xb, xb);
            filteredImage.SetPixel(x, y, wb);
        }
    return filteredImage;
}

private Bitmap lowPassFilter(int lowPassType)
{
    Bitmap filteredImage = new Bitmap(noiseImage);
    int[] xt = new int[10];
    int xb = 0;
    for (int x = 1; x < noiseImage.Width - 1; x++)
        for (int y = 1; y < noiseImage.Height - 1; y++)
        {
            Color w1 = noiseImage.GetPixel(x - 1, y - 1);
            Color w2 = noiseImage.GetPixel(x - 1, y);
Color w3 = noiseImage.GetPixel(x - 1, y + 1);
Color w4 = noiseImage.GetPixel(x, y - 1);
Color w5 = noiseImage.GetPixel(x, y);
Color w6 = noiseImage.GetPixel(x, y + 1);
Color w7 = noiseImage.GetPixel(x + 1, y - 1);
Color w8 = noiseImage.GetPixel(x + 1, y);
Color w9 = noiseImage.GetPixel(x + 1, y + 1);

xt[1] = w1.R; xt[2] = w2.R; xt[3] = w3.R;
xt[4] = w4.R; xt[5] = w5.R; xt[6] = w6.R;
xt[7] = w7.R; xt[8] = w8.R; xt[9] = w9.R;
// low pass filter type 1
// 0 1 0
// 1/6 * 1 2 1
// 0 1 0
//
// low pass filter type 2
// 1 1 1
// 1/10 * 1 2 1
// 1 1 1
//
// low pass filter type 3
// 1 1 1
// 1/9 * 1 1 1
// 1 1 1
//
// low pass filter type 4
// 1 2 1
// 1/16 * 2 4 2
// 1 2 1
//
//calculation of low pass filter
if (lowPassType == 1)
{
   xb = (int)(0 * xt[1] + 1 * xt[2] + 0 * xt[3] +
   1 * xt[4] + 2 * xt[5] + 1 * xt[6] +
   0 * xt[7] + 1 * xt[8] + 0 * xt[9]) / 6;
   if (xb < 0) xb = 0;
   if (xb > 255) xb = 255;
}
else if (lowPassType == 2)
{
   xb = (int)(1 * xt[1] + 1 * xt[2] + 1 * xt[3] +
                1 * xt[4] + 2 * xt[5] + 1 * xt[6] +
                1 * xt[7] + 1 * xt[8] + 1 * xt[9]) / 6;
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (lowPassType == 3)
            {
                xb = (int)(1 * xt[1] + 1 * xt[2] + 1 * xt[3] +
                1 * xt[4] + 1 * xt[5] + 1 * xt[6] +
                1 * xt[7] + 1 * xt[8] + 1 * xt[9]) / 6;
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (lowPassType == 4)
            {
                xb = (int)(1 * xt[1] + 2 * xt[2] + 1 * xt[3] +
                2 * xt[4] + 4 * xt[5] + 2 * xt[6] +
                1 * xt[7] + 2 * xt[8] + 1 * xt[9]) / 6;
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            Color wb = Color.FromArgb(xb, xb, xb);
            filteredImage.SetPixel(x, y, wb);
        }
    return filteredImage;
}
private Bitmap sharpeningFilter(int filterType)
{
    noiseImage = grayImage;
    Bitmap filteredImage = new Bitmap(noiseImage);
    int[] xt = new int[10];
    int xb = 0;
    for (int x = 1; x < noiseImage.Width - 1; x++)
        for (int y = 1; y < noiseImage.Height - 1; y++)
        {
            Color w1 = noiseImage.GetPixel(x - 1, y - 1);
            Color w2 = noiseImage.GetPixel(x - 1, y);
            Color w3 = noiseImage.GetPixel(x - 1, y + 1);
            Color w4 = noiseImage.GetPixel(x, y - 1);
            Color w5 = noiseImage.GetPixel(x, y);
            Color w6 = noiseImage.GetPixel(x, y + 1);
            Color w7 = noiseImage.GetPixel(x + 1, y - 1);
            Color w8 = noiseImage.GetPixel(x + 1, y);
Color w9 = noiseImage.GetPixel(x + 1, y + 1);
xt[1] = w1.R; xt[2] = w2.R; xt[3] = w3.R;
xt[4] = w4.R; xt[5] = w5.R; xt[6] = w6.R;
xt[7] = w7.R; xt[8] = w8.R; xt[9] = w9.R;
// Robert filter
// -1 1
// 1 -1
//
// Prewit vertical filter
// -1 0 1
// -1 0 1
// -1 0 1
//
// Prewit horizontal filter
// -1 -1 -1
// 0 0 0
// 1 1 1
//
// Sobel horizontal filter
// -1 -2 -1
// 0 0 0
// 1 2 1
//
// Sobel vertical filter
// -1 0 1
// -2 0 2
// -1 0 1
//
// Laplacian filter
// 1 -2 1
// -2 4 -2
// 1 -2 1
//
if (filterType == 1) //Robert filter
{
   //calculation of mean
   xb = xt[5] - xt[2] + xt[5] - xt[4];
   if (xb < 0) xb = 0;
   if (xb > 255) xb = 255;
}
else if (filterType == 2) //Prewitt filter
{
   int xh = -1 * xt[1] - 1 * xt[2] - 1 * xt[3] +
                0 * xt[4] + 0 * xt[5] + 0 * xt[6] +
                1 * xt[7] + 1 * xt[8] + 1 * xt[9];
                int xv = -1 * xt[1] + 0 * xt[2] + 1 * xt[3] -
                1 * xt[4] + 0 * xt[5] + 1 * xt[6] -
                1 * xt[7] + 0 * xt[8] + 1 * xt[9];
                xb = xh + xv;
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (filterType == 3) //Sobel filter
            {
                int xh = -1 * xt[1] - 2 * xt[2] - 1 * xt[3] +
                0 * xt[4] + 0 * xt[5] + 0 * xt[6] +
                1 * xt[7] + 2 * xt[8] + 1 * xt[9];
                int xv = -1 * xt[1] + 0 * xt[2] + 1 * xt[3] -
                2 * xt[4] + 0 * xt[5] + 2 * xt[6] -
                1 * xt[7] + 0 * xt[8] + 1 * xt[9];
                xb = xh + xv;
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (filterType == 4) //Laplacian filter
            {
                xb = (int)(1 * xt[1] - 2 * xt[2] + 1 * xt[3] +
                -2 * xt[4] + 4 * xt[5] - 2 * xt[6] +
                1 * xt[7] - 2 * xt[8] + 1 * xt[9]);
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            Color wb = Color.FromArgb(xb, xb, xb);
            filteredImage.SetPixel(x, y, wb);
        }
    return filteredImage;
}
private Bitmap highPassFilter(int highPassType)
{
    noiseImage = grayImage;
    Bitmap filteredImage = new Bitmap(noiseImage);
    int[] xt = new int[10];
    int xb = 0;
    for (int x = 1; x < noiseImage.Width - 1; x++)
        for (int y = 1; y < noiseImage.Height - 1; y++)
        {
```

```
Color w1 = noiseImage.GetPixel(x - 1, y - 1);
Color w2 = noiseImage.GetPixel(x - 1, y);
Color w3 = noiseImage.GetPixel(x - 1, y + 1);
Color w4 = noiseImage.GetPixel(x, y - 1);
Color w5 = noiseImage.GetPixel(x, y);
Color w6 = noiseImage.GetPixel(x, y + 1);
Color w7 = noiseImage.GetPixel(x + 1, y - 1);
Color w8 = noiseImage.GetPixel(x + 1, y);
Color w9 = noiseImage.GetPixel(x + 1, y + 1);
xt[1] = w1.R; xt[2] = w2.R; xt[3] = w3.R;
xt[4] = w4.R; xt[5] = w5.R; xt[6] = w6.R;
xt[7] = w7.R; xt[8] = w8.R; xt[9] = w9.R;
// high pass filter type 1
// 0 1 0
// 1 -4 1
// 0 1 0
//
// high pass filter type 2
// 0 -1 0
// -1 4 -1
// 0 -1 0
//
// high pass filter type 3
// 1 1 1
// 1 -8 1
// 1 1 1
//
// high pass filter type 4
// -1 -1 -1
// -1 8 -1
// -1 -1 -1
//
//calculation of low pass filter
if (highPassType == 1)
{
   xb = (int)(0 * xt[1] + 1 * xt[2] + 0 * xt[3] +
   1 * xt[4] - 4 * xt[5] + 1 * xt[6] +
   0 * xt[7] + 1 * xt[8] + 0 * xt[9]);
   if (xb < 0) xb = 0;
   if (xb > 255) xb = 255;
}
else if (highPassType == 2)
{
```

```
                xb = (int)(0 * xt[1] - 1 * xt[2] + 0 * xt[3] -
                1 * xt[4] + 4 * xt[5] - 1 * xt[6] +
                0 * xt[7] - 1 * xt[8] + 0 * xt[9]);
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (highPassType == 3)
            {
                xb = (int)(1 * xt[1] + 1 * xt[2] + 1 * xt[3] +
                1 * xt[4] - 8 * xt[5] + 1 * xt[6] +
                1 * xt[7] + 1 * xt[8] + 1 * xt[9]);
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            else if (highPassType == 4)
            {
                xb = (int)(-1 * xt[1] - 1 * xt[2] - 1 * xt[3] -
                1 * xt[4] + 8 * xt[5] - 1 * xt[6] -
                1 * xt[7] - 1 * xt[8] - 1 * xt[9]);
                if (xb < 0) xb = 0;
                if (xb > 255) xb = 255;
            }
            Color wb = Color.FromArgb(xb, xb, xb);
            filteredImage.SetPixel(x, y, wb);
        }
    return filteredImage;
}
// reset
private void resetRadioButtonSmoothing()
{
    radioButton1.Checked = false;
    radioButton2.Checked = false;
    radioButton3.Checked = false;
    radioButton4.Checked = false;
}
private void resetRadioButtonSharpening()
{
    radioButton5.Checked = false;
    radioButton6.Checked = false;
    radioButton7.Checked = false;
    radioButton8.Checked = false;
}
private void resetRadioButtonLowPass()
```

```
{
   radioButton9.Checked = false;
   radioButton10.Checked = false;
   radioButton11.Checked = false;
   radioButton12.Checked = false;
}
private void resetRadioButtonHighPass()
{
   radioButton13.Checked = false;
   radioButton14.Checked = false;
   radioButton15.Checked = false;
   radioButton16.Checked = false;
}
private void resetCeckBox()
{
   checkBox1.Checked = false;
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
   // median filter
   if (radioButton2.Checked == false) return;
   //resetting radio button
   resetRadioButtonSharpening();
   resetRadioButtonLowPass();
   resetRadioButtonHighPass();
   if (noiseImage == null) return;
   Bitmap tempImage = new Bitmap(noiseImage);
   filterSmoothingType = 2;
   tempImage = smoothingfilter(filterSmoothingType);
   pictureBox1.Image = tempImage;
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
   // minimum filter
   if (radioButton3.Checked == false) return;
   //resetting radio button
   resetRadioButtonSharpening();
   resetRadioButtonLowPass();
   resetRadioButtonHighPass();
   if (noiseImage == null) return;
   Bitmap tempImage = new Bitmap(noiseImage);
```

```
    filterSmoothingType = 3;
    tempImage = smoothingfilter(filterSmoothingType);
    pictureBox1.Image = tempImage;
}
private void radioButton5_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton5.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    Bitmap tempImage = new Bitmap(noiseImage);
    filterSharpeningType = 1;
    tempImage = sharpeningFilter(filterSharpeningType);
    pictureBox1.Image = tempImage;
}

private void radioButton9_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton9.Checked == false) return;
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    resetRadioButtonSmoothing();
    resetRadioButtonSharpening();
    resetRadioButtonHighPass();
    lowPassType = 1;
    tempImage = lowPassFilter(lowPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton6_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton6.Checked == false) return;
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    filterSharpeningType = 2;
    tempImage = sharpeningFilter(filterSharpeningType);
    pictureBox1.Image = tempImage;
```

```
}

private void radioButton13_CheckedChanged(object sender, EventArgs e)
{
   if (radioButton13.Checked == false) return;
   if (noiseImage == null) return;
   resetCeckBox();
   resetRadioButtonSmoothing();
   resetRadioButtonLowPass();
   resetRadioButtonSharpening();
   Bitmap tempImage = new Bitmap(noiseImage);
   highPassType = 1;
   tempImage = highPassFilter(highPassType);
   pictureBox1.Image = tempImage;
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
   //resetting radio button
   resetRadioButtonSmoothing();
   resetRadioButtonSharpening();
   if (checkBox1.Checked == true)
   {
      Bitmap tempImage = noiseImaging(grayImage);
      noiseImage = tempImage;
      //menampilkan noise image
      pictureBox1.Image = noiseImage;
   }
   else
   {
      Bitmap tempImage = grayImaging(sourceImage);
      grayImage = tempImage;
      noiseImage = grayImage;
      //menampilakan gray image
      pictureBox1.Image = grayImage;
   }
}

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
   // mean filter
   if (radioButton1.Checked == false) return;
   //resetting radio button
```

```
    resetRadioButtonSharpening();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    filterSmoothingType = 1;
    tempImage = smoothingfilter(filterSmoothingType);
    pictureBox1.Image = tempImage;
}

private void radioButton4_CheckedChanged(object sender, EventArgs e)
{
    // maximum filter
    if (radioButton4.Checked == false) return;
    //resetting radio button
    resetRadioButtonSharpening();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    filterSmoothingType = 4;
    tempImage = smoothingfilter(filterSmoothingType);
    pictureBox1.Image = tempImage;
}

private void radioButton10_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton10.Checked == false) return;
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    resetRadioButtonSmoothing();
    resetRadioButtonSharpening();
    resetRadioButtonHighPass();
    lowPassType = 2;
    tempImage = lowPassFilter(lowPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton11_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton11.Checked == false) return;
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    resetRadioButtonSmoothing();
    resetRadioButtonSharpening();
    resetRadioButtonHighPass();
    lowPassType = 3;
    tempImage = lowPassFilter(lowPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton12_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton12.Checked == false) return;
    if (noiseImage == null) return;
    Bitmap tempImage = new Bitmap(noiseImage);
    resetRadioButtonSmoothing();
    resetRadioButtonSharpening();
    resetRadioButtonHighPass();
    lowPassType = 4;
    tempImage = lowPassFilter(lowPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton14_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton14.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonSharpening();
    Bitmap tempImage = new Bitmap(noiseImage);
    highPassType = 2;
    tempImage = highPassFilter(highPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton15_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton15.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonSharpening();
    Bitmap tempImage = new Bitmap(noiseImage);
    highPassType = 3;
    tempImage = highPassFilter(highPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton16_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton16.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonSharpening();
    Bitmap tempImage = new Bitmap(noiseImage);
    highPassType = 4;
    tempImage = highPassFilter(highPassType);
    pictureBox1.Image = tempImage;
}

private void radioButton7_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton7.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    Bitmap tempImage = new Bitmap(noiseImage);
    filterSharpeningType = 3;
    tempImage = sharpeningFilter(filterSharpeningType);
    pictureBox1.Image = tempImage;
}

private void radioButton8_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton8.Checked == false) return;
    if (noiseImage == null) return;
    resetCeckBox();
    resetRadioButtonSmoothing();
    resetRadioButtonLowPass();
    resetRadioButtonHighPass();
    Bitmap tempImage = new Bitmap(noiseImage);
```

```
            filterSharpeningType = 4;
            tempImage = sharpeningFilter(filterSharpeningType);
            pictureBox1.Image = tempImage;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // reset
            resetRadioButtonSmoothing();
            resetRadioButtonSharpening();
            resetRadioButtonLowPass();
            resetRadioButtonHighPass();
            resetCeckBox();
        }
    }
}
```