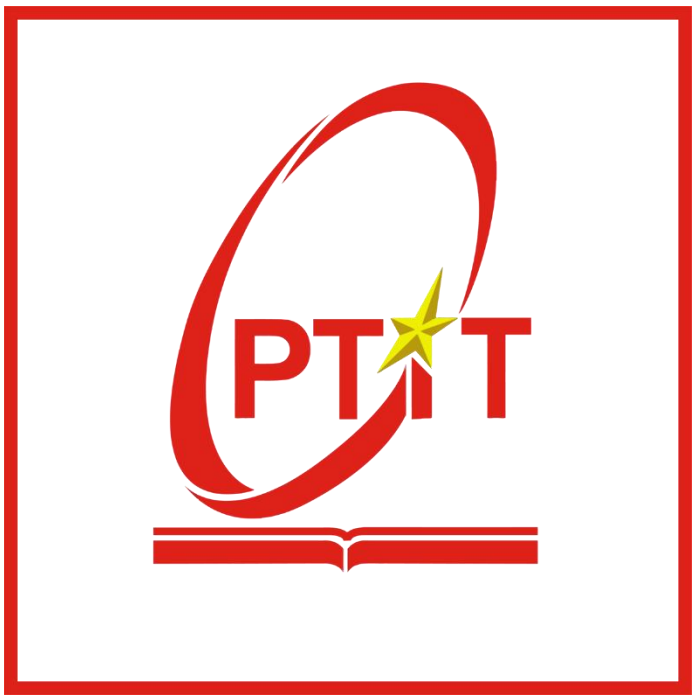
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH



BÁO CÁO CUỐI KỲ

XỬ LÝ ẢNH

Giảng viên hướng dẫn : ThS. HUỲNH TRUNG TRỤ

Sinh viên thực hiện	**Mã sinh viên**	**Lớp**
Nguyễn Phan Nhựt Trường	**N20DCCN082**	**D20CQCNPM01-N**
Bùi Tuấn Anh	**N20DCCN002**	**D20CQCNPM01-N**

Tp. Hồ Chí Minh - 12/2023

Mục lục

Lời nói đầu

Xử lý ảnh là một phân ngành trong xử lý số tín hiệu với tín hiệu xử lý là ảnh. Đây là một phân ngành khoa học mới rất phát triển trong những năm gần đây. Xử lý ảnh gồm 4 lĩnh vực chính: xử lý nâng cao chất lượng ảnh, nhận dạng ảnh, nén ảnh và truy vấn ảnh. Sự phát triển của xử lý ảnh đem lại rất nhiều lợi ích cho cuộc sống của con người.

Ngày nay xử lý ảnh đã được áp dụng rất rộng rãi trong đời sống như: photoshop, nén ảnh, nén video, nhận dạng biển số xe, nhận dạng khuôn mặt, nhận dạng chữ viết, xử lý ảnh thiên văn, ảnh y tế,....

Chính vì thế, ngày hôm nay nhóm chúng em mang đến đây với đề tài "**Xây dựng phần mềm (tool) xử lý ảnh cơ bản**" để vận dụng được những kiến thức đã học trong xử lý hình ảnh thực tế bằng phần mềm này.

Em xin cảm ơn sự giúp đỡ của thầy Huỳnh Trung Trụ, đã giúp em hoàn thành bài báo cáo này. Nhưng do lần đầu tiên lập trình phần mềm xử lý ảnh nên còn nhiều bỡ ngỡ và những thiếu sót. Rất mong nhận được sự góp ý của thầy.

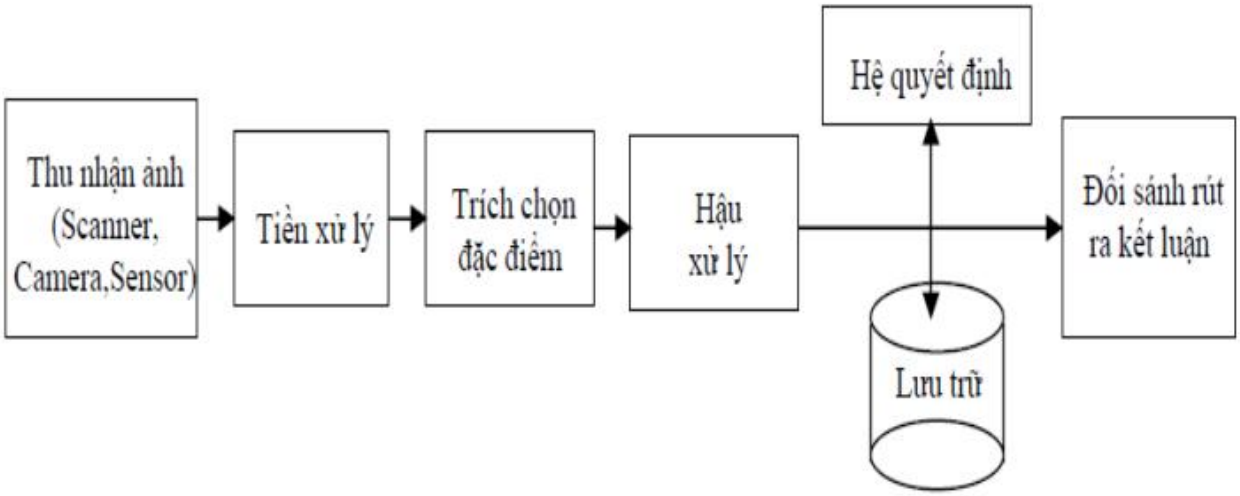
Nhóm sinh viên thực hiện

Phần I, Giới thiệu

1.1, Giới thiệu xử lý ảnh

Quá trình xử lý ảnh là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh "tốt hơn" hoặc một kết luận.

Sơ đồ quá trình xử lý ảnh:

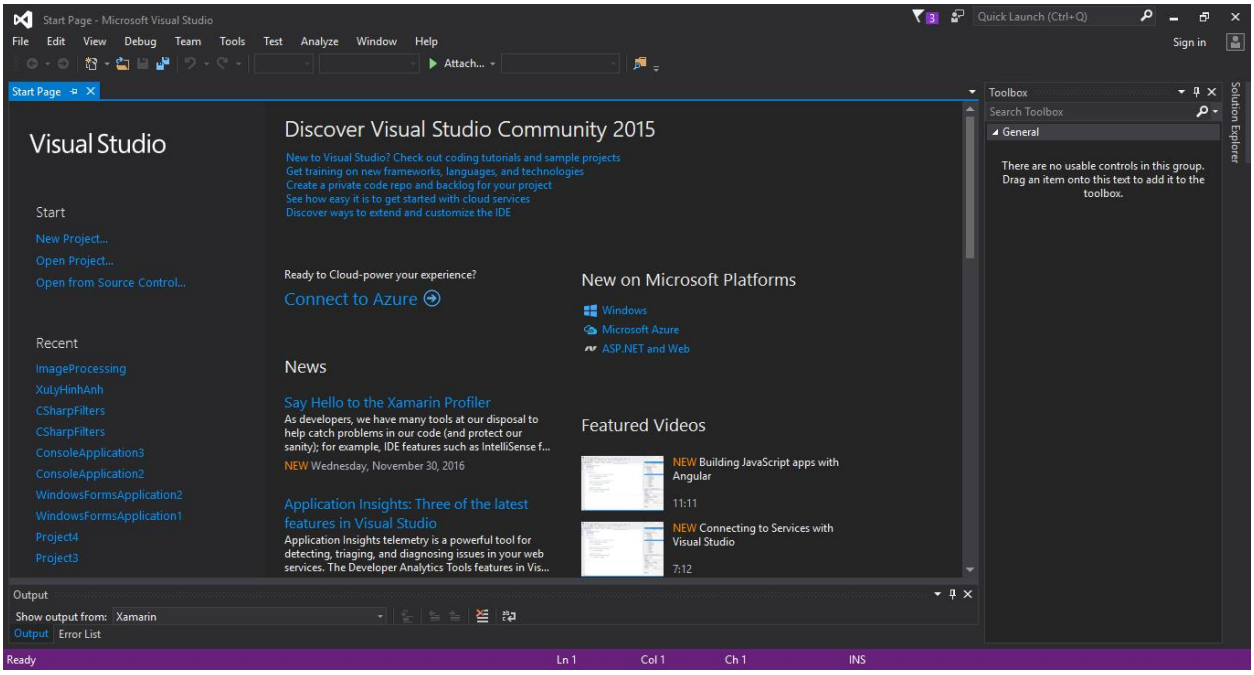


Hình 1.1: Quá trình xử lý ảnh

1.2, Giới thiệu Visual Studio

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C,[4] C++ và C++/CLI (thông qua Visual C++), VB.NET (thông qua Visual Basic.NET), C# (thông qua Visual C#) và F# (như của Visual Studio 2010[5]). Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng rẽ. Nó cũng hỗ trợ XML/XSLT, HTML/XHTML, JavaScript và CSS.



Hình 1.2: Giao diện chính Visual Studio 2015

1.3, Giới thiệu Class Bitmap

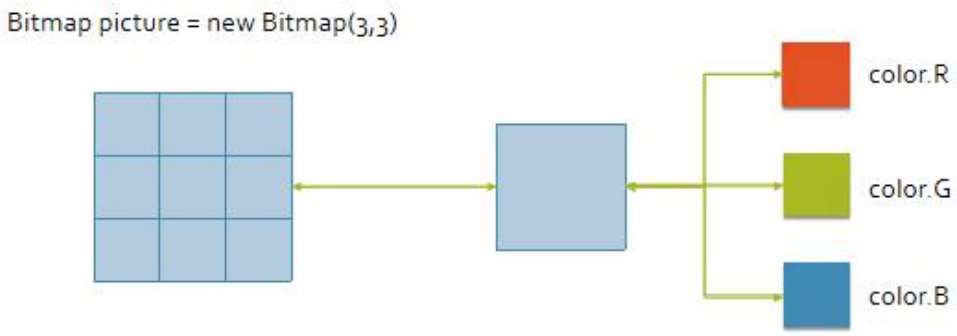
Gói gọn trong một GDI + Bitmap, trong đó bao gồm các dữ liệu pixel cho hình ảnh đồ họa và các thuộc tính của nó. Một Bitmap là một đối tượng được sử dụng để làm việc với hình ảnh được xác định bởi dữ liệu từ các pixel của hình ảnh.

Bao gồm một số thuộc tính xử lý chính:

- Getpixel: Lấy ra một điểm ảnh

- Color: Lấy ra thuộc tính màu của một pixel

- Setpixel: Đặt giá trị cho một điểm ảnh

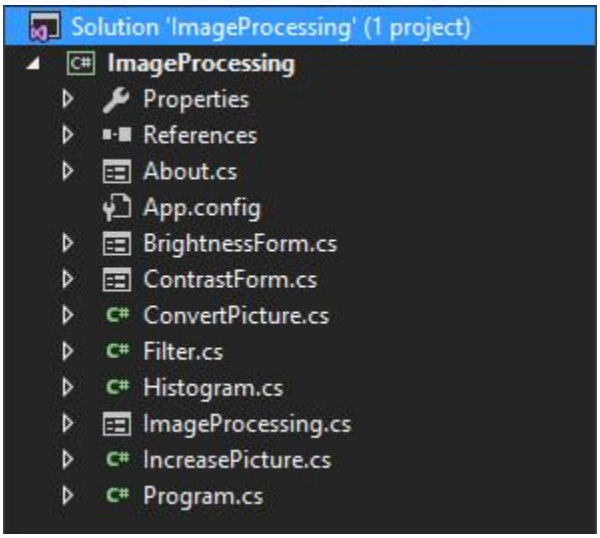



```
Color color = picture.GetPixel(2,1)
picture.SetPixel(2,1,Color.FromArgb(color.R, color.G, color.B))
```

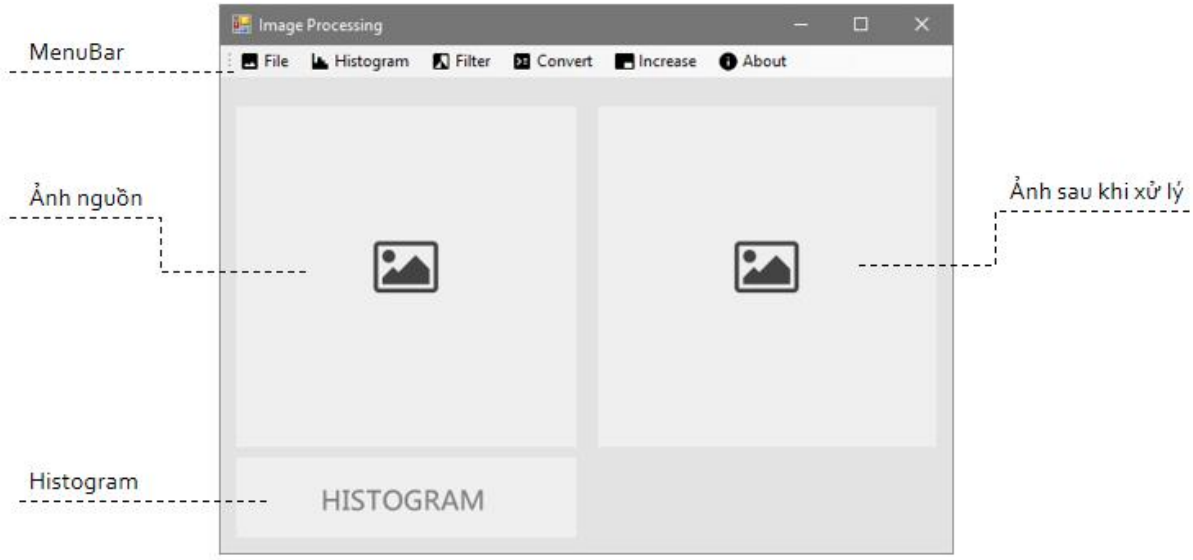
Phần II: Phân tích, thiết kế bố cục và các nhóm chức năng phần mềm

2.1, Phân tích, xây dựng chức năng bố cục phần mềm

Phần mềm được viết bằng ngôn ngữ C#. Bao gồm 3 Form: ImagesProcessing là Form hiển thị chính của phần mềm, các Form còn lại BrightnessForm và ContrastForm là 2 Form phụ. 4 Class chính là ConvertPicture, Filter, Histogram, IncreasePicture.



Hình 2.1: Cấu trúc các Form và Class



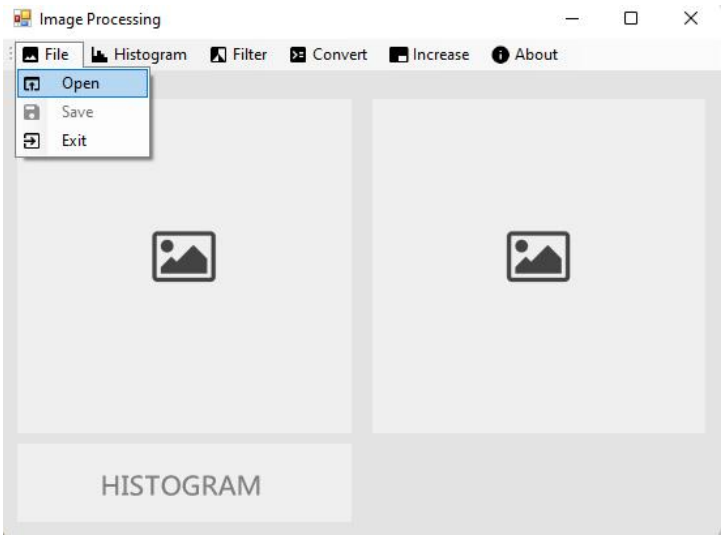
Hình 2.2: Bố cục của phần mềm

2.2, Chức năng mở hình

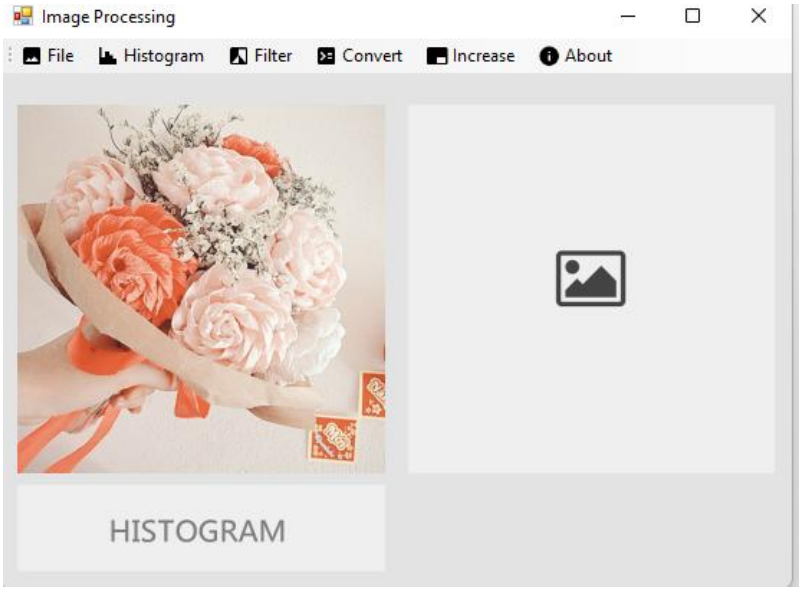
Chức năng này sẽ mở ra một cửa sổ cho phép chọn một hình ảnh và dùng nó làm ảnh nguồn để chỉnh sửa.

Vị trí: Thuộc vào Form ImagesProcessing (Form chính).

Sự kiện: Click vào nút Open.




```
private void openImages_Click(object sender, EventArgs e)
{
    OpenFileDialog ofile = new OpenFileDialog();
    if (DialogResult.OK == ofile.ShowDialog())
    {
        this.PictureSource.Image = new Bitmap(ofile.FileName);
    }
}
```



2.3, Nhóm chức năng chuyển ảnh (Convert)

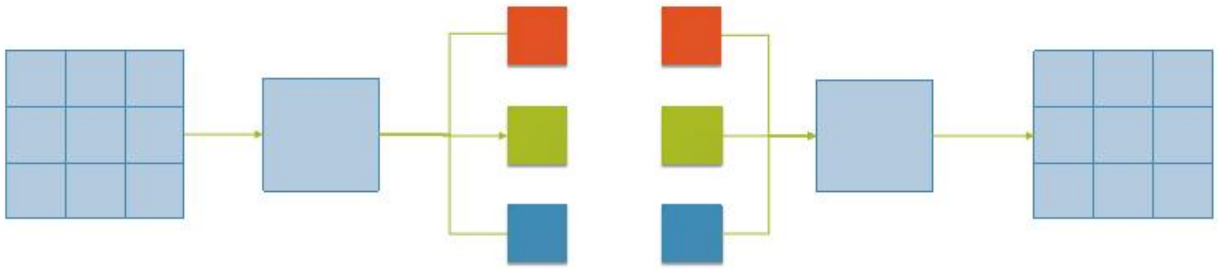
Bao gồm chuyển thành ảnh xám và chuyển thành âm bản.

2.3.1, Chuyển thành ảnh xám

Chức năng này cho phép chuyển ảnh nguồn thành ảnh xám.

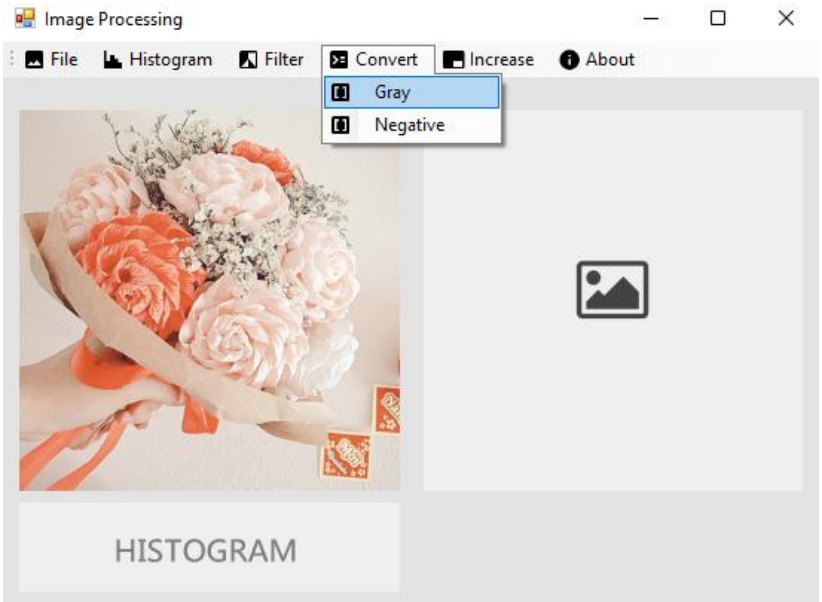
Công thức: 0.287 x red + 0.599 x green + 0.114 x blue

Ý tưởng: Sử dụng vòng lặp để đọc từng điểm ảnh. Sau đó sử dụng công thức chuyển ảnh màu thành ảnh xám để tính giá trị mới. Cuối cùng đặt lại giá trị mới cho điểm ảnh vừa đọc.



Vị trí: Thuộc nhóm chức năng Convert trong thanh Menubar của Form ImagesProcessing

Sự kiện: Click vào nút Gray trong Convert.



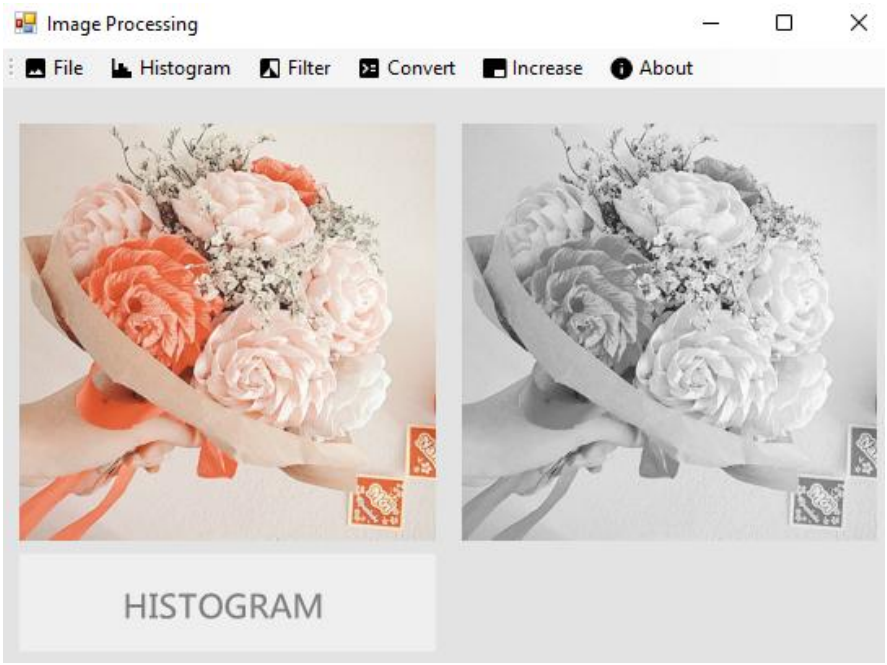

```
private void convertToGray_Click(object sender, EventArgs e)
{
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    ConvertPicture.convertToGray(pic);
```

```
    this.PictureResult.Image = pic;
}
```

Class ConvertPicture chứa hàm chuyển thành ảnh xám:

```
class ConvertPicture
{
    public static bool convertToGray(Bitmap picture)
    {
        for (int i = 0; i < picture.Width; i++)
            for(int j = 0; j<picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                int red = color.R;
                int green = color.G;
                int blue = color.B;
                int gray = (byte)(.287 * red + .599 * green + .114 * blue);
                picture.SetPixel(i, j, Color.FromArgb(gray, gray, gray));
            }
        return true;
    }

}
```

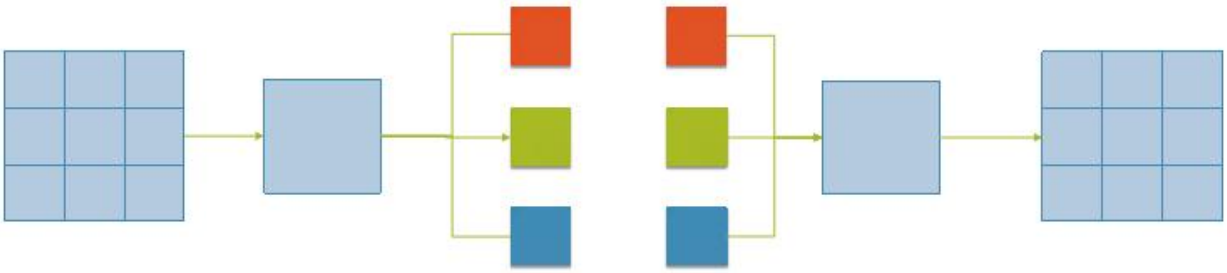


2.3.2, Chuyển thành âm bản

Chức năng này cho phép chuyển ảnh nguồn thành âm bản

Công thức: 255 – mức sáng

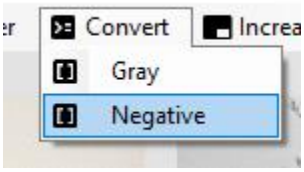
Ý tưởng: Sử dụng vòng lặp để đọc từng điểm ảnh trong hình. Với mỗi điểm ảnh, mỗi giá trị màu bằng 255 trừ đi giá trị hiện tại. Cuối cùng đặt lại giá trị mới cho điểm ảnh vừa đọc.



255-Color

Vị trí: Thuộc nhóm chức năng Convert trong thanh Menubar của Form ImagesProcessing

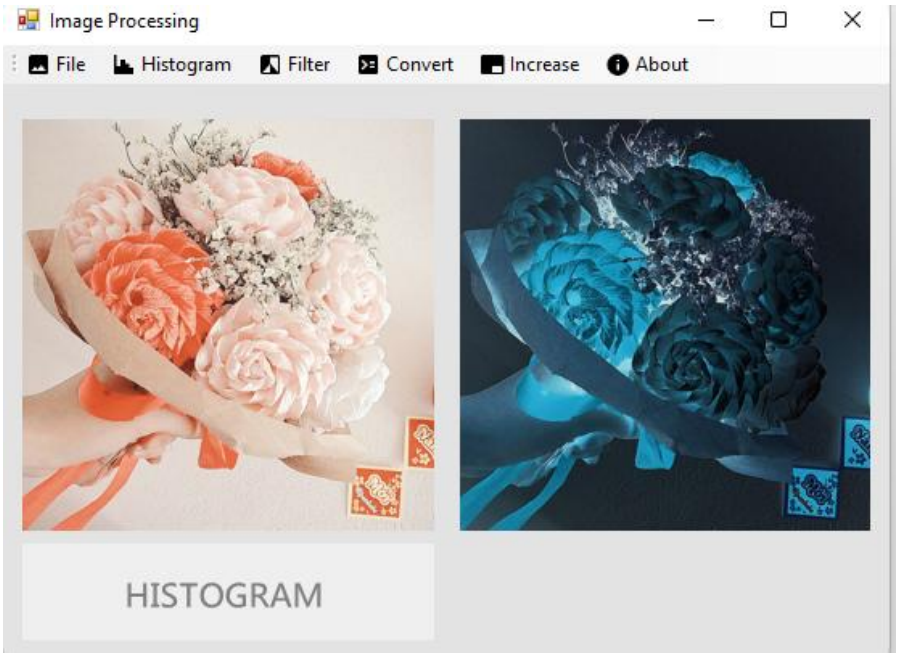
Sự kiện: Click vào nút Negative trong Convert




```
private void convertToNegative_Click(object sender, EventArgs e)
{
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    ConvertPicture.convertToNegative(pic);
    this.PictureResult.Image = pic;
}
```

Class ConvertPicture chứa hàm chuyển thành âm bản:

```
class ConvertPicture
{
    public static bool convertToNegative(Bitmap picture)
    {
        for (int i = 0; i < picture.Width; i++)
            for (int j = 0; j < picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                int red = 255 - color.R;
                int green = 255 - color.G;
                int blue = 255 - color.B;
                picture.SetPixel(i, j, Color.FromArgb(red, green, blue));
            }
        return true;
    }
}
```



2.4, Nhóm chức năng Histogram

Bao gồm chức năng hiển thị Histogram và cân bằng tự động Histogram

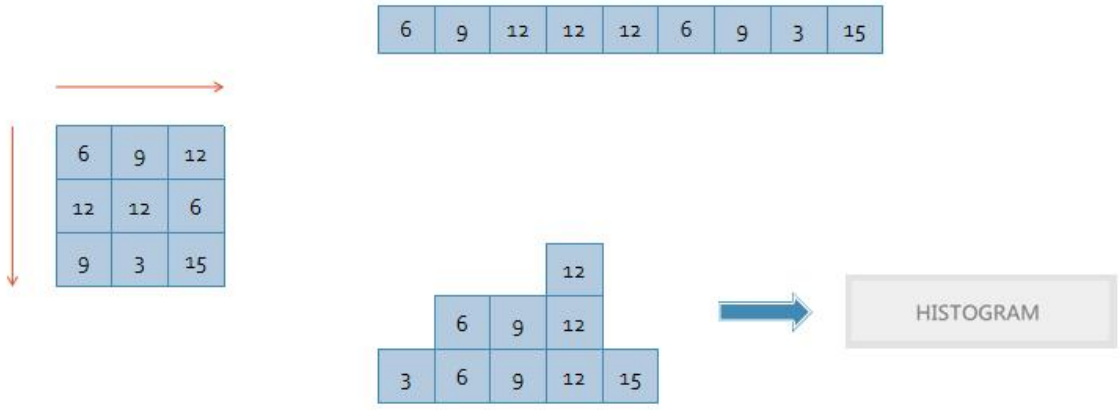
2.4.1, Chức năng hiển thị Histogram

Biểu đồ Histogram là biểu đồ tần suất mức xám g của ảnh I là số điểm ảnh có giá trị g của ảnh I.

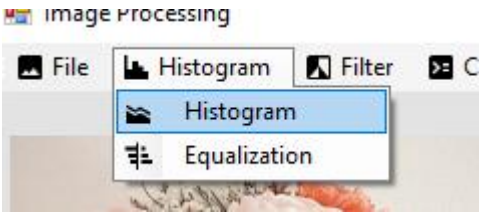
Chức năng này sẽ hiển thị ra biểu đồ Histogram.

Vị trí: Thuộc nhóm chức năng Histogram trong thanh Menubar của Form ImagesProcessing

Ý tưởng: Để vẽ được biểu đồ Histogram, đầu tiên cần phải tìm ra được tất cả các mức xám có trong ảnh và đếm số lượng các mức xám đó. Sau đó sử dụng Bitmap để tạo một ảnh mới chứa Histogram. Hiển thị ra ngoài phần mềm.



Sự kiện: Click vào nút Histogram trong nhóm Histogram




```
private void showHistogram_Click(object sender, EventArgs e)
{

    Bitmap pic = new Bitmap(this.PictureSource.Image);
    ConvertPicture.convertToNegative(pic);
    Bitmap his = new Bitmap(256, 60);
    Histogram.showHistogram(pic, his);
    this.PictureHistogram.Image = his;
}
```

Class Histogram xử lý hàm hiển thị Histogram (showHistogram):

```
class Histogram
{
    public static bool showHistogram(Bitmap picture, Bitmap histogram)
    {
        // Biến lưu tất cả giá trị điểm ảnh ban đầu
        int[] pixelValue = new int[picture.Width * picture.Height];
        // Biến đếm
        int count = 0;
        // Vòng lặp lấy ra giá trị của tất cả điểm ảnh ban đầu
        for(int i = 0; i<picture.Width; i++)
            for(int j = 0; j<picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                pixelValue[count] = color.R;
                count++;
            }

        // Biến lưu số lượng các mức xám từ 0-255
        int[] grayValue = new int[256];
        // Reset biến đếm
        count = 0;
        // Vòng lặp đếm số lượng các mức xám
        for (int i = 0; i <= 255; i++)
        {
            for(int j=0; j<picture.Height*picture.Width; j++)
                if (i == pixelValue[j])
                    count++;
            grayValue[i] = count;
            count = 0;
        }

        // Tìm ra điểm sáng max và chia tỉ lệ để hiển thị Histogram
        int max = grayValue[0];
        for (int i = 1; i <= 255; i++)
            if (grayValue[i] > max)
                max = grayValue[i];
```

```
            // Chia tỉ lệ
            float ratio = histogram.Height / (float)max;
            // Điều chỉnh lại tỉ lệ hiển thị
            for(int i = 0; i<=255; i++)
                grayValue[i] = Convert.ToInt32(grayValue[i] * ratio);

            // Hiển thị Histogram
            // Quét màu đen
            for(int i=0; i<histogram.Width; i++)
                for(int j=0; j<histogram.Height; j++)
                    histogram.SetPixel(i, j, Color.FromArgb(240, 240, 240));
            // Quét điểm ảnh để hiển thị
            for (int i = 0; i < histogram.Width; i++)
                for (int j = 0; j < histogram.Height; j++)
                {
                    if (j >= (histogram.Height - grayValue[i]) && j < histogram.Height)
                        histogram.SetPixel(i, j, Color.FromArgb(0, 0, 0));
                }

            return true;
        }
    }
```

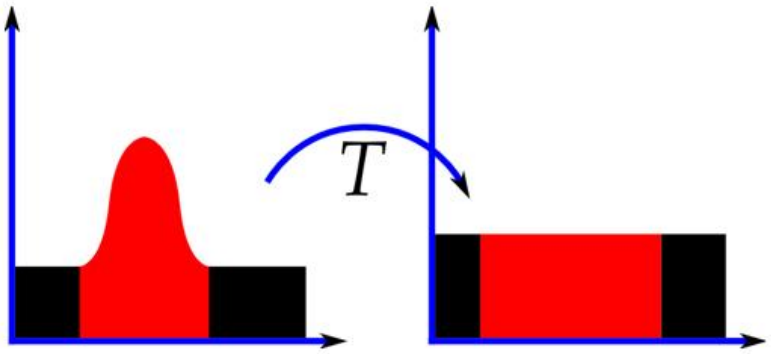
2.4.2, Chức năng cân bằng Histogram tự động

Vị trí: Thuộc nhóm chức năng Histogram trong thanh Menubar của Form ImagesProcessing

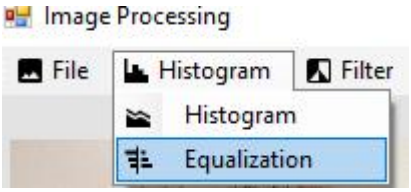
Chức năng này sẽ cân bằng Histogram tự động.

Ý tưởng: Dựa vào thuật toán cân bằng Histogram.

- Cân bằng Histogram là dàn đều các điểm sáng tập trung cho các mức xám khác



Sự kiện: Click vào nút Equalization trong nhóm Histogram




```
private void equalHistogram_Click(object sender, EventArgs e)
{
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    ConvertPicture.convertToGray(pic);
    Bitmap his = new Bitmap(256, 60);
    Histogram.equalHistogram(pic, his);
    Histogram.showHistogram(pic, his);
    this.PictureResult.Image = pic;
    this.PictureHistogram.Image = his;
}
```

Class Histogram cân bằng Histogram tự động

```
class Histogram
{
    public static bool equalHistogram(Bitmap picture, Bitmap histogram)
    {
        // Biến lưu tất cả giá trị điểm ảnh ban đầu
        int[] pixelValue = new int[picture.Width * picture.Height];
        // Biến đếm
        int count = 0;
        // Vòng lặp lấy ra giá trị của tất cả điểm ảnh ban đầu
        for (int i = 0; i < picture.Width; i++)
            for (int j = 0; j < picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                pixelValue[count] = color.R;
                count++;
            }

        // Biến lưu số lượng các mức xám từ 0-255
        int[] grayValue = new int[256];
        // Reset biến đếm
```

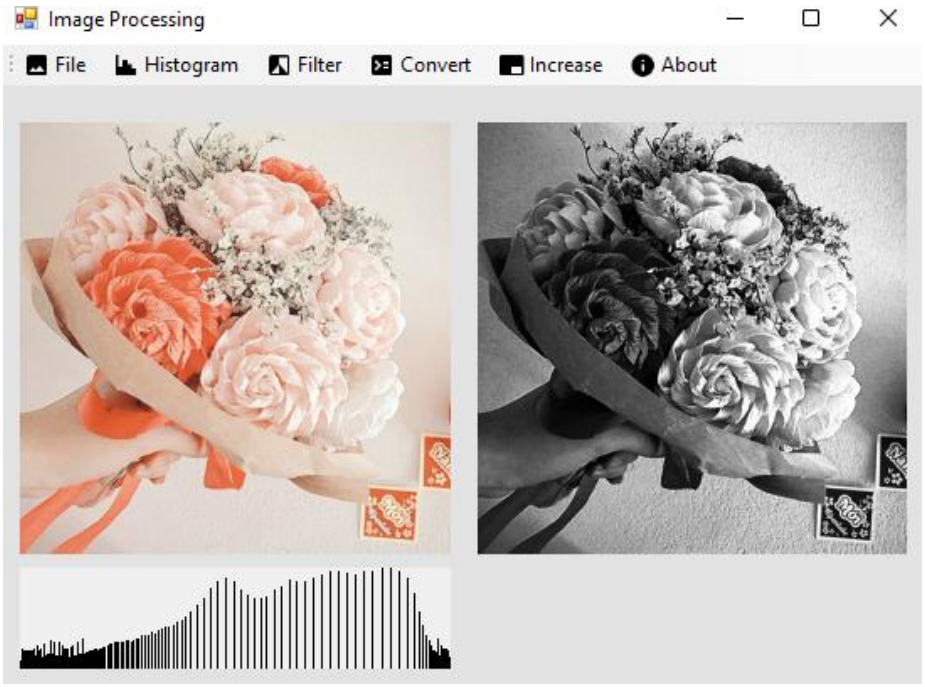
```
        count = 0;
        // Vòng lặp đếm số lượng các mức xám
        for (int i = 0; i <= 255; i++)
        {
            for (int j = 0; j < picture.Height * picture.Width; j++)
                if (i == pixelValue[j])
                    count++;
            grayValue[i] = count;
            count = 0;
        }

        // Tìm ra độ sáng thấp nhất tổng giá trị sáng
        int min = grayValue[0];
        int allValue = 0;
        for (int i = 0; i <= 255; i++)
        {
            allValue += grayValue[i];
            if (grayValue[i] < min)
                min = grayValue[i];
        }


        // Chuyển hết mức xám về giá trị mới
        // Biến cộng dồn
        int increment = 0;
        for(int i=0; i<=255; i++)
        {
            //System.Console.WriteLine(allValue);
            increment += grayValue[i];
            double num = (double)(increment - min) / (allValue-1);
            //System.Console.WriteLine(num);
            grayValue[i] = (int)Math.Round(num*255);
            //System.Console.WriteLine(grayValue[i]);
        }

        // Đặt lại ảnh xám
        for (int i = 0; i < picture.Width; i++)
            for (int j = 0; j < picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                int value = color.R;
                picture.SetPixel(i, j, Color.FromArgb(grayValue[value], grayValue[value],
grayValue[value]));
            }

    return true;
    }
}
```



2.5, Nhóm chức năng hiệu chỉnh (Increase)

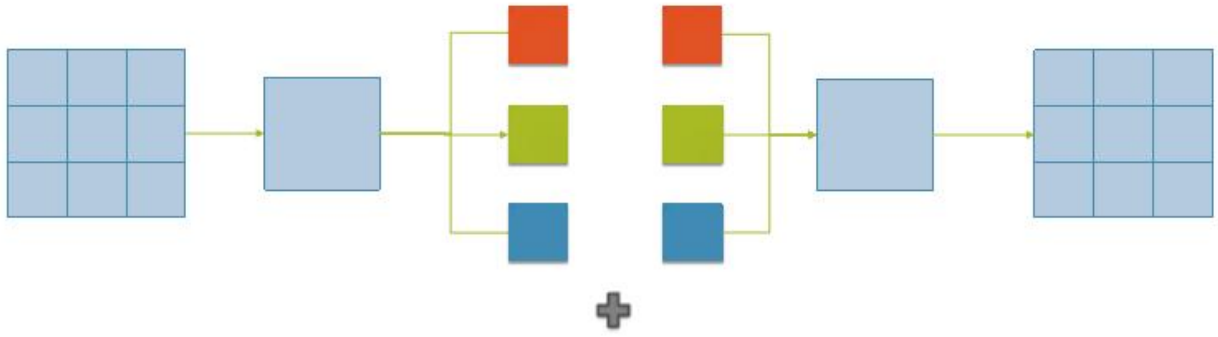
Bao gồm thay đổi độ sáng và thay đổi độ tương phản.

2.5.1, Thay đổi độ sáng

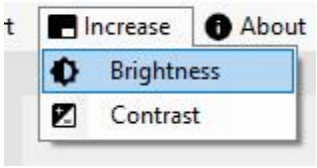
Chức năng này cho phép thay đổi độ sáng theo giá trị đã nhập vào ô input.

Vị trí: Thuộc nhóm chức năng Increase trong thanh Menubar của Form ImagesProcessing

Ý tưởng: Hiện ra một Form Brightness cho phép nhập vào giá trị mức sáng cần tăng. Xử lý lệnh này bằng các cộng giá trị vừa nhập vào từng điểm ảnh. Nếu giá trị sáng quá 255 thì giá trị sáng max đó bằng 255.

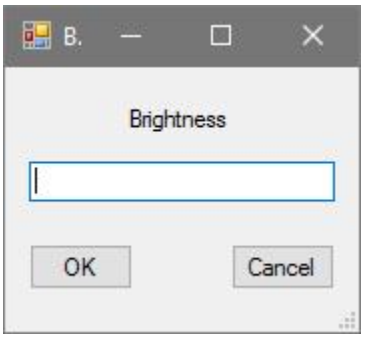


Sự kiện: Click vào nút Brightness trong Increase, nhập giá trị vào ô Brightness trong Form mới và nhấn OK.




```
private void changeBrightness_Click(object sender, EventArgs e)
{
    brightnessForm bnForm = new brightnessForm();
    bnForm.ShowDialog();
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    IncreasePicture.increaseBrightness(pic, bnForm.getBrightness());
    this.PictureResult.Image = pic;
}
```

Form Brightness chứa ô nhập giá trị Brightness và nút OK



Sự kiện Click vào OK trong Form Brightness:

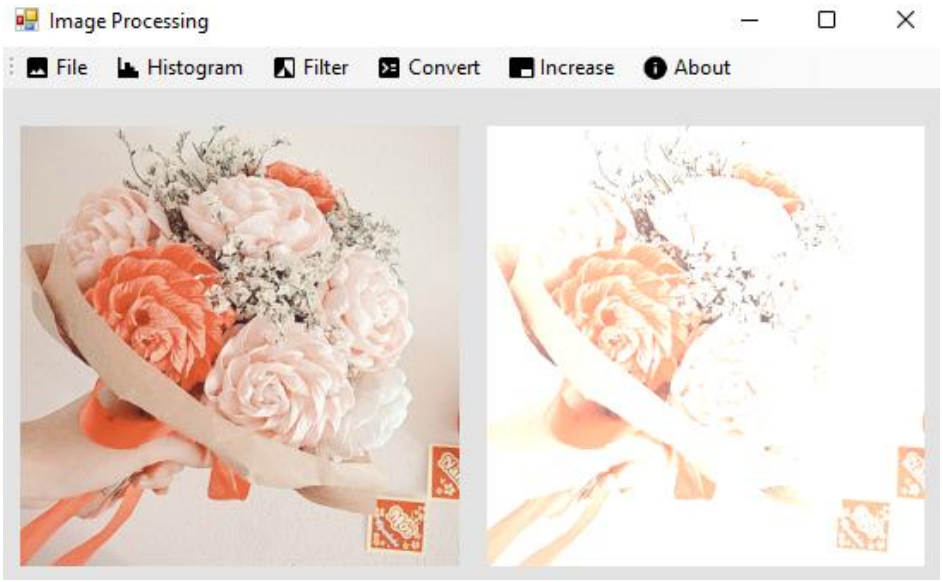
```
private void ok_button_Click(object sender, EventArgs e)
{
    brightnessValue = String.IsNullOrEmpty(brightness_textBox.Text) ? 0 :
Convert.ToInt32(brightness_textBox.Text);
    this.Close();
}

public int getBrightness()
{
    return brightnessValue;
}
```

Class IncreasePicture chứa hàm thay đổi độ sáng:

```
class IncreasePicture
{
    public static bool increaseBrightness(Bitmap picture, int value)
    {
        for (int i = 0; i < picture.Width; i++)
            for (int j = 0; j < picture.Height; j++)
            {
                Color color = picture.GetPixel(i, j);
                int red = color.R + value < 255 ? color.R + value : 255;
                int green = color.G + value < 255 ? color.G + value : 255;
                int blue = color.B + value < 255 ? color.B + value : 255;
                picture.SetPixel(i, j, Color.FromArgb(red, green, blue));
```

```
            }
        return true;
    }
}
```



2.5.2, Thay đổi độ tương phản

Chức năng này cho phép thay đổi độ tương phản theo giá trị nhập vào

Vị trí: Thuộc nhóm chức năng Increase trong thanh Menubar của Form ImagesProcessing

Ý tưởng: Hiện ra một Form Brightness cho phép nhập vào giá trị tương phản mới.

Giá trị ở đây nằm từ -100 đến 100

Sử dụng công thức:

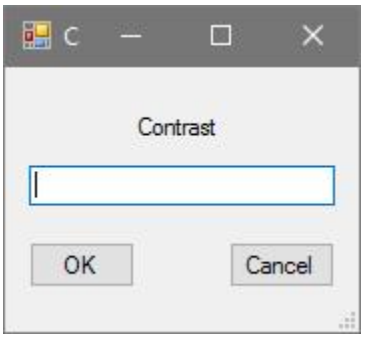
```
T = colorValue / 255;
T -= 0.5;
T *= contrastValue;
T += 0.5;
T *= 255;
if (T > 255)
    colorValue = 255;
else if (T < 0)
    colorValue = 0;
```

Tính lần lượt với colorValue là các giá trị màu R, G, B và cuối cùng SetPixel lại điểm ảnh

Sự kiện: Click vào nút Contrast trong Incease, nhập giá trị vào ô Contrast và chọn OK


```
private void contrastToolStripMenuItem_Click(object sender, EventArgs e)
{
    ContrastForm ctForm = new ContrastForm();
    ctForm.ShowDialog();
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    IncreasePicture.increaseContrast(pic, ctForm.getContrast());
    this.PictureResult.Image = pic;
}
```

Form Contrast chứa ô nhập giá trị Contrast và nút OK



Sự kiện Click vào OK trong Form Contrast:

```
private void ok_button_Click(object sender, EventArgs e)
{
    contrastValue = String.IsNullOrEmpty(contrast_textBox.Text) ? 0 :
Convert.ToDouble(contrast_textBox.Text);
    this.Close();
}

public double getContrast()
{
    return contrastValue;
}
```

Class IncreasePicture chứa hàm thay đổi độ tương phản

```
class IncreasePicture
{
    public static bool increaseContrast(Bitmap picture, double value)
    {
        // Khai báo các biến
        int R = 0, G = 0, B = 0;
        double T;
        Color color;

        value = (100.0 + value) / 100.0;
        value *= value;

        // Vòng lặp chuyển điểm ảnh
```

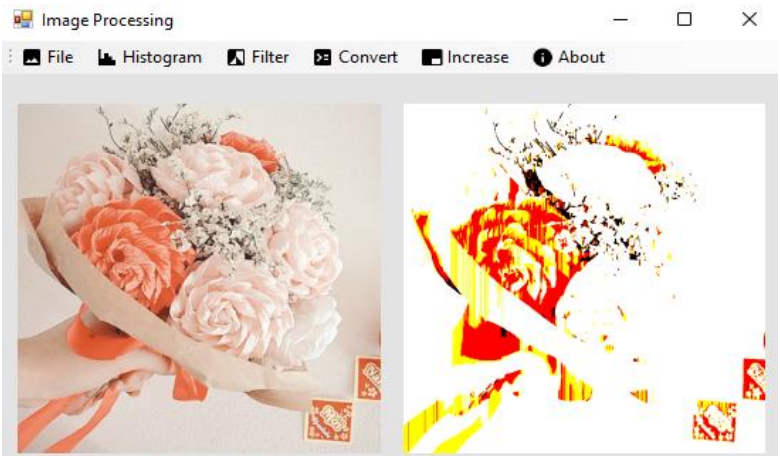
```
        for (int i = 0; i < picture.Height; i++)
        {
            for (int j = 0; j < picture.Width; j++)
            {
                color = picture.GetPixel(i, j);

                T = color.R / 255.0;
                T -= 0.5;
                T *= value;
                T += 0.5;
                T *= 255;
                if (T > 255)
                    R = 255;
                else if (T < 0)
                    R = 0;

                T = color.G / 255.0;
                T -= 0.5;
                T *= value;
                T += 0.5;
                T *= 255;
                if (T > 255)
                    G = 255;
                else if (T < 0)
                    G = 0;

                T = color.B / 255.0;
                T -= 0.5;
                T *= value;
                T += 0.5;
                T *= 255;
                if (T > 255)
                    B = 255;
                else if (T < 0)
                    B = 0;

                picture.SetPixel(i, j, Color.FromArgb(R, G, B));
            }
        }
        return true;
    }
}
```



2.6, Nhóm chức năng phát hiện biên (Filter)

2.6.1, Phát hiện biên bằng Sobel

Thuật toán phát hiện biên Sobel là làm nổi bật đường biên. Được xác định bằng tổng tích chập của 2 ma trận chập $\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$ và $\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$

Tính tích chập với từng ma trận chập sau đó cộng hai giá trị tuyệt đối của từng điểm ảnh lại với nhau, nếu giá trị mới > 255 thì giá trị đó = 255

Vị trí: Thuộc nhóm chức năng Filter trong thanh Menubar của Form ImagesProcessing

Sự kiện: Click vào EDGE trong nhóm Filter


```
private void toEDGE_Click(object sender, EventArgs e)
{
    Bitmap pic = new Bitmap(this.PictureSource.Image);
    ConvertPicture.convertToGray(pic);
    Filter.toEDGE(pic);
    this.PictureResult.Image = pic;
}
```

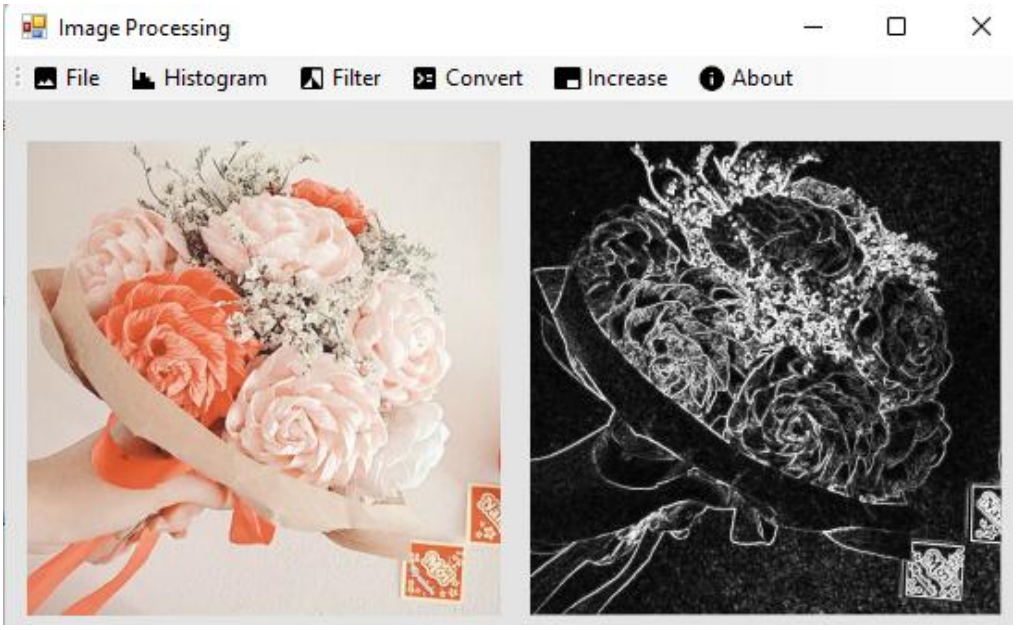
Class Filter chứa hàm xử lý:

```
class Filter
{

    public static bool toEDGE(Bitmap picture)
    {
        int[] matrix = new int[9];
        int[,] pixelPicture = new int[picture.Width - 2, picture.Height - 2];
        int[,] pixelPicture1 = new int[picture.Width-2,picture.Height-2];
        int[,] pixelPicture2 = new int[picture.Width - 2, picture.Height - 2];
        int pixel;
        int count = 0;
        for(int i =0; i<picture.Width-2; i++)
            for(int j=0; j<picture.Height-2; j++)
            {
                count = 0;
                for (int n = i; n <= i + 2; n++)
                    for (int m = j; m <= j + 2; m++)
                    {
                        Color color = picture.GetPixel(n, m);
                        matrix[count] = color.R;
                        count++;
                    }
                pixel = matrix[0] * -1 + matrix[1] * -2 + matrix[3] * -1 + matrix[6] * 1
+ matrix[7] * 2 + matrix[8] * 1;
                pixelPicture1[i, j] = pixel;
```

```
            }
        for (int i = 0; i < picture.Width - 2; i++)
            for (int j = 0; j < picture.Height - 2; j++)
            {
                count = 0;
                for (int n = i; n <= i + 2; n++)
                    for (int m = j; m <= j + 2; m++)
                    {
                        Color color = picture.GetPixel(n, m);
                        matrix[count] = color.R;
                        count++;
                    }
                pixel = matrix[0] * -1 + matrix[2] * 1 + matrix[3] * -1 + matrix[5] * 1 +
matrix[6] * -1 + matrix[8] * 1;
                pixelPicture2[i, j] = pixel;

            }
        for (int i = 0; i < picture.Width - 2; i++)
            for (int j = 0; j < picture.Height - 2; j++)
            {
                pixelPicture[i, j] = Math.Abs(pixelPicture1[i, j]) +
Math.Abs(pixelPicture2[i, j]);
            }
        for (int i = 0; i < picture.Width - 2; i++)
            for (int j = 0; j < picture.Height - 2; j++)
            {
                Color color = picture.GetPixel(i, j);
                if (pixelPicture[i, j] > 255)
                    picture.SetPixel(i, j, Color.FromArgb(255, 255, 255));
                else
                    picture.SetPixel(i, j, Color.FromArgb(pixelPicture[i, j],
pixelPicture[i, j], pixelPicture[i, j]));
            }
        return true;
    }
}
```



2.7, Chức năng lưu hình

Cho phép lưu lại bức hình đã chỉnh sửa

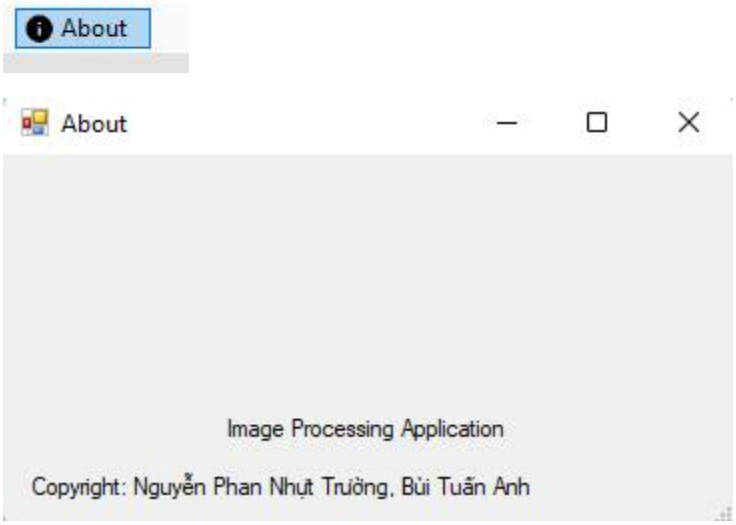
Sự kiện: Click vào nút Save.


```
private void saveImages_Click(object sender, EventArgs e)
{
    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "JPEG files(*.jpeg)|*.jpeg";
    if(DialogResult.OK == save.ShowDialog())
    {
        this.PictureResult.Image.Save(save.FileName);
    }
}
```

2.8, Về phần mềm

Hiển thị thông tin phần mềm.

Sự kiện: Click vào nút About trên thanh Menu.



2.9, Hiệu chỉnh phần mềm

Khi chưa chọn một hình ảnh nào. Các chức năng trong các nhóm chức năng của phần mềm sẽ không cho phép chọn.

Vô hiệu hóa các nút bấm đó ngay sau khi sự kiện FormMain_Load() sảy ra. Đây là sự kiện đầu tiên khi chạy phần mềm.

```
private void FormMain_Load(object sender, EventArgs e)
{
    savePicture.Enabled = false;
    showHistogram.Enabled = false;
    equalHistogram.Enabled = false;
    convertToGray.Enabled = false;
    convertToNegative.Enabled = false;
    changeBrightness.Enabled = false;
    contrastToolStripMenuItem.Enabled = false;
    toEDGE.Enabled = false;
}
```

Sau đó bật lại bằng cách nếu đã chọn hình:

```
private void openImages_Click(object sender, EventArgs e)
{
    OpenFileDialog ofile = new OpenFileDialog();
    if (DialogResult.OK == ofile.ShowDialog())
    {
        this.PictureSource.Image = new Bitmap(ofile.FileName);
        savePicture.Enabled = true;
        showHistogram.Enabled = true;
        equalHistogram.Enabled = true;
        convertToGray.Enabled = true;
        convertToNegative.Enabled = true;
        changeBrightness.Enabled = true;
        contrastToolStripMenuItem.Enabled = true;
        toEDGE.Enabled = true;
    }
}
```

Sự kiện `openImages_Click()` là sự kiện mở hình ảnh.