

Universidade Federal do Rio Grande do Sul  
INF01046 – Fundamentos de Processamento de Imagens

## **Software de Processamento de Imagens – Parte III**

Jorge Wichrowski Krieger de Mello – 143283  
jwkmhdr@hotmail.com

Site : [www.inf.ufrgs.br/~jwkmello](http://www.inf.ufrgs.br/~jwkmello)

[www.inf.ufrgs.br/~jwkmello/tudo/trab\\_fpi.html](http://www.inf.ufrgs.br/~jwkmello/tudo/trab_fpi.html)

Prof. Manuel M. Oliveira

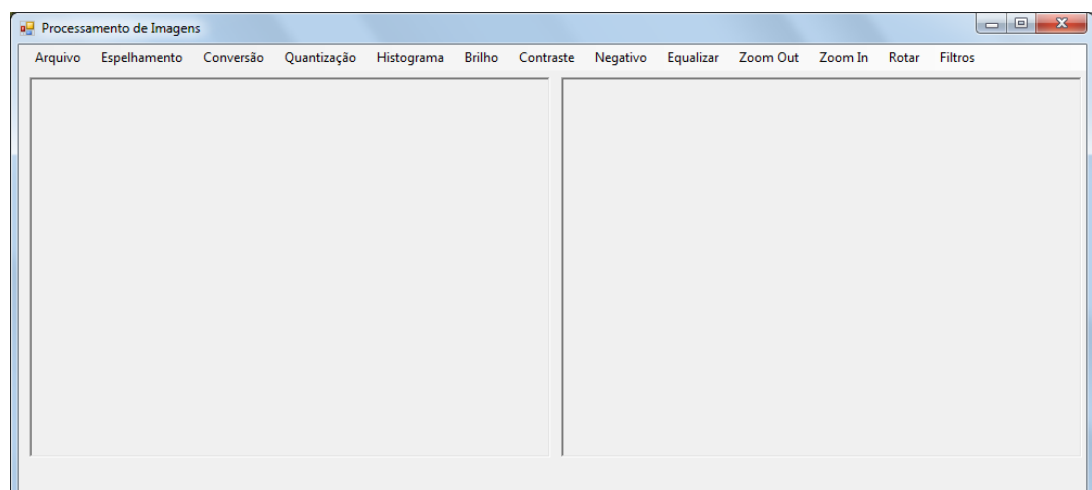
## 1. INTRODUÇÃO

Este trabalho me possibilitou uma ótima experiência de aprendizado, pude trabalhar um pouco com processamento de imagens, pude aprender uma linguagem que não havia tido muito contato C#, entendi um pouco mais sobre formatos das imagens, usei o Visual Studio uma das melhores ferramentas na minha opinião, mas que nem sempre é possível usar nos trabalhos escolares. Também estou tendo a oportunidade de melhorar um pouco meu site pessoal com mais conteúdo.

Neste relatório vou mostrar o desenvolvimento e o resultado dos 5 itens pedidos na definição do trabalho. Mas antes gostaria de explicar seu funcionamento e ressaltar que este programa é possivelmente o mais intuitivo de se usar já feito para esta disciplina. Tudo acontece em uma janela auto dimensionável, com todas as opções visíveis para o usuário. Quando você escolhe uma opção ele abre uma janela para que você selecione uma imagem, após selecionar uma imagem ele mostra ela na janela 1 e o resultado da opção escolhida na janela 2. Você pode salvar esta imagem digitando CTRL + S ou simplesmente Arquivo -> Salvar. Para a quantização existe um campo de texto no próprio menu com o número que você deseja escolher, o valor default é 8. O programa funciona com vários tipos de imagens .jpg, .png, .bmp, entretanto ele salva apenas no formato .jpg. O programa também mantém o valor de Alfa das imagens, ou seja, não modifica sua transparência. Ele irá funcionar em qualquer sistema operacional Windows que tenha o net.framework4.0 instalado.

Agora o programa também mostra o Histograma, Brilho, Contraste, Negativo e Equaliza a imagem. A interface continua intuitiva no entanto esta pouco prática, mas o objetivo é justamente ver os conceitos de processamento de imagens e não de se vender o software. Então considero que a intuitividade é mais importante que a prática.

Para finalizar este super programa ele foi incrementado com funções de Zoom, IN/OUT, Rotação Horária e Anti-Horária. O programa também recebeu filtros para se aplicar nas imagens.



## 2. Desenvolvimento

### 2.1 Zoom Out:

O usuário deve passar como entrada o tamanho da nova imagem a ser gerada, o programa calcula os fatores, é interessante que a nova imagem tenha suas dimensões divisores das dimensões originais, isto gera fatores inteiros e a redução é perfeita, do contrário a imagem poderá sofrer um pequeno corte, para se evitar esse corte poderia se ter usado uma abordagem matemática considerando fatores racionais, mas por simplicidade de programação foi usado fatores inteiros.

Código Fonte:

```
if (hori <= h)
    {
        if (verti <= v)
            {

                nova_imagem = new Bitmap(hori, verti);

                int fator_x = h / hori;
                int fator_y = v / verti;

                while (i < verti)
                {

                    while (u < hori)
                    {

                        while (i2 < fator_y)
                        {
                            if (acumulado_i + i2 <= v)
                                {
                                    mediaR = mediaR +
imagem.GetPixel(acumulado_u, acumulado_i + i2).R;
                                    mediaG = mediaG +
imagem.GetPixel(acumulado_u, acumulado_i + i2).G;
                                    mediaB = mediaB +
imagem.GetPixel(acumulado_u, acumulado_i + i2).B;
                                    mediaA = mediaA +
imagem.GetPixel(acumulado_u, acumulado_i + i2).A;
                                    a++;
                                }

                            i2 = i2 + 1;
                        }
                    while (u2 < fator_x)
                    {
                        if (acumulado_u + u2 <= h)
                            {
                                mediaR = mediaR +
imagem.GetPixel(acumulado_u + u2, acumulado_i).R;
                                mediaG = mediaG +
imagem.GetPixel(acumulado_u + u2, acumulado_i).G;
```

```

        mediaB = mediaB +
imagem.GetPixel(acumulado_u + u2, acumulado_i).B;
        mediaA = mediaA +
imagem.GetPixel(acumulado_u + u2, acumulado_i).A;
        a++;
    }

    u2 = u2 + 1;
}
mediaA = mediaA / a;
mediaR = mediaR / a;
mediaG = mediaG / a;
mediaB = mediaB / a;

nova_imagem.SetPixel(u, i,
Color.FromArgb(Convert.ToInt32(mediaA), Convert.ToInt32(mediaR),
Convert.ToInt32(mediaG), Convert.ToInt32(mediaB)));
mediaA = 0;
mediaR = 0;
mediaG = 0;
mediaB = 0;
i2 = 0;
u2 = 0;
a = 0;
u = u + 1;
acumulado_u = acumulado_u + fator_x;
}
u = 0;
acumulado_u = 0;

i = i + 1;
acumulado_i = acumulado_i + fator_y;
}

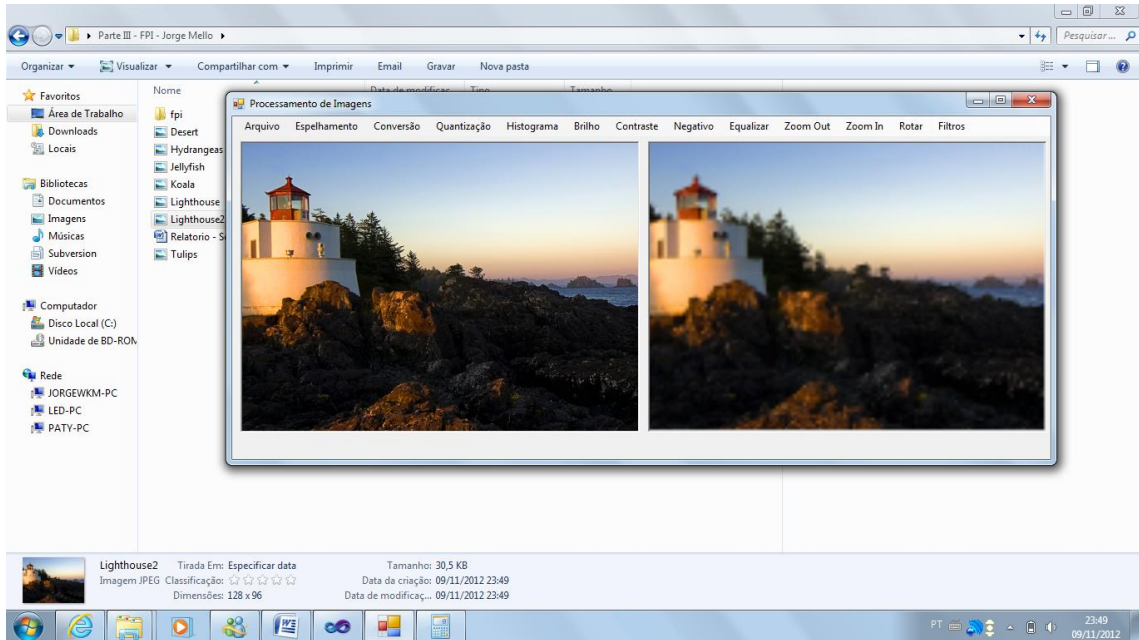
}

else
{
    MessageBox.Show("Tamanho vertical maior que imagem
original");
}
else
{
    MessageBox.Show("Tamanho horizontal maior que imagem
original");
}
}
}

```

## Resultado:

Você deve salvar a imagem e ver seu tamanho reduzido no diretório, pois o software redimensiona ela para caber no "picture box", mas é possível ver como ela perde qualidade em função da diminuição da resolução.



## 2.2 Zoom In 2x2:

Código Fonte:

```
int i = 0;
int u = 0;
int i2 = 0;
int u2 = 0;
int mediaR = 0;
int mediaG = 0;
int mediaB = 0;
int mediaA = 0;
pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
imagem = new
Bitmap(Image.FromFile(openFileDialog1.FileName));
nome = openFileDialog1.FileName;
int h = imagem.Width;
int v = imagem.Height;
nova_imagem = new Bitmap(2 * h, 2 * v);

while (i < v)
{
    while (u < h)
    {
        nova_imagem.SetPixel(u2, i2, imagem.GetPixel(u, i));

        u = u + 1;
        u2 = u2 + 2;
    }
    u = 0;
    u2 = 0;
    i = i + 1;
    i2 = i2 + 2;
}

u = 0;
i = 0;
while (i < 2 * v)
{
    while (u < 2 * h)
    {
        if (u % 2 == 1)
        {
            if (u + 1 >= 2 * h)
            {
                mediaR = (nova_imagem.GetPixel(u - 1, i).R);
                mediaG = (nova_imagem.GetPixel(u - 1, i).G);
                mediaB = (nova_imagem.GetPixel(u - 1, i).B);
                mediaA = (nova_imagem.GetPixel(u - 1, i).A);
            }
            else
            {
                mediaR = (nova_imagem.GetPixel(u - 1, i).R +
nova_imagem.GetPixel(u + 1, i).R) / 2;
                mediaG = (nova_imagem.GetPixel(u - 1, i).G +
nova_imagem.GetPixel(u + 1, i).G) / 2;
                mediaB = (nova_imagem.GetPixel(u - 1, i).B +
nova_imagem.GetPixel(u + 1, i).B) / 2;
```

```

        mediaA = (nova_imagem.GetPixel(u - 1, i).A +
nova_imagem.GetPixel(u + 1, i).A) / 2;
    }
    nova_imagem.SetPixel(u, i, Color.FromArgb(mediaA,
mediaR, mediaG, mediaB));
    }
    u = u + 1;
}
u = 0;
i = i + 2;
}

u = 0;
i = 0;
while (u < 2 * h)
{
    while (i < 2 * v)
    {
        if (i % 2 == 1)
        {
            if (i + 1 >= 2 * v)
            {
                mediaR = (nova_imagem.GetPixel(u, i - 1).R);
                mediaG = (nova_imagem.GetPixel(u, i - 1).G);
                mediaB = (nova_imagem.GetPixel(u, i - 1).B);
                mediaA = (nova_imagem.GetPixel(u, i - 1).A);
            }
            else
            {
                mediaR = (nova_imagem.GetPixel(u, i - 1).R +
nova_imagem.GetPixel(u, i + 1).R) / 2;
                mediaG = (nova_imagem.GetPixel(u, i - 1).G +
nova_imagem.GetPixel(u, i + 1).G) / 2;
                mediaB = (nova_imagem.GetPixel(u, i - 1).B +
nova_imagem.GetPixel(u, i + 1).B) / 2;
                mediaA = (nova_imagem.GetPixel(u, i - 1).A +
nova_imagem.GetPixel(u, i + 1).A) / 2;
            }
            nova_imagem.SetPixel(u, i, Color.FromArgb(mediaA,
mediaR, mediaG, mediaB));
        }
        i = i + 1;
    }
    i = 0;
    u = u + 1;
}

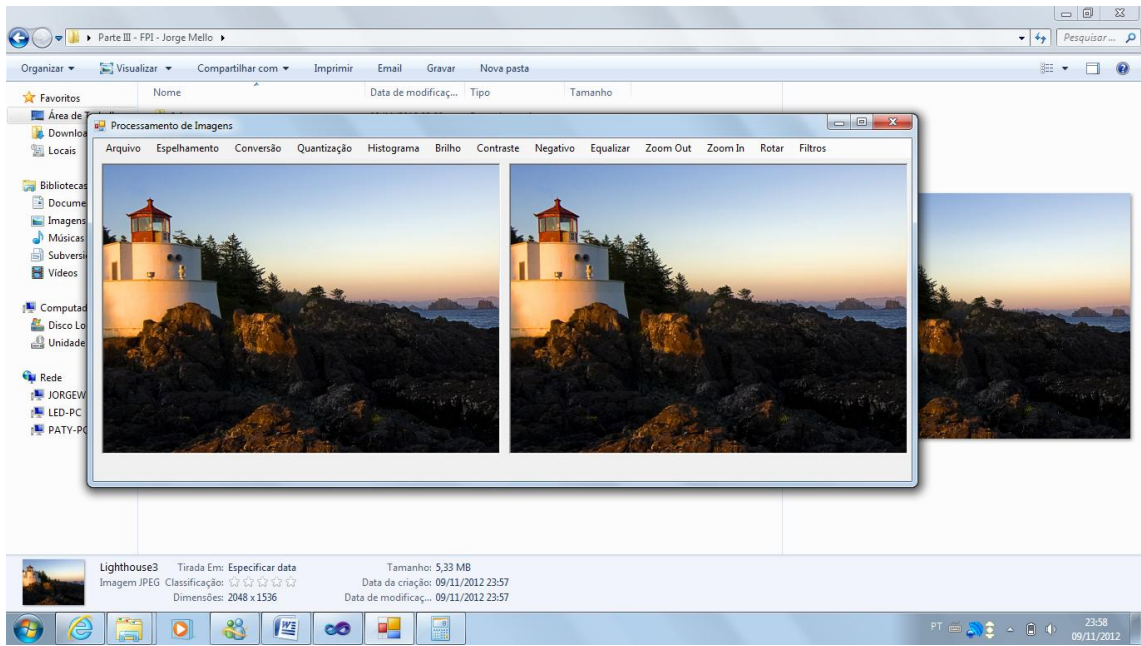
pictureBox2.Image = nova_imagem;
MessageBox.Show("Salve a imagem para ver o seu tamanho no
diretório!");

}

}
catch { }

```

## Resultado:



### 2.3 Rotação Anti Horária e Horária:

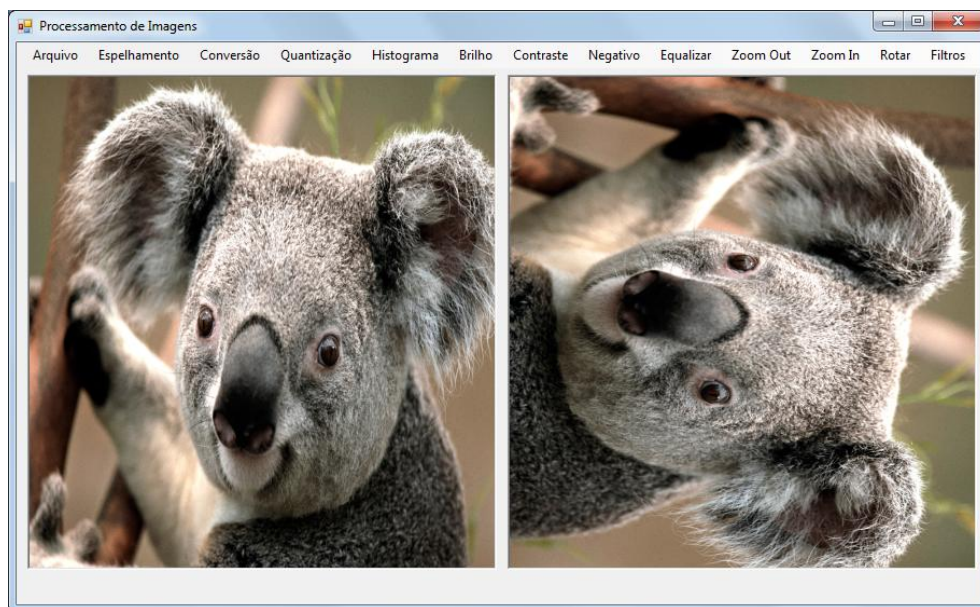
É feita a rotação onde os pixels são deslocados para posição oposta ao qual estão em sua linha ou coluna. Para aplicar mais de uma rotação salve a imagem e execute de novo.

Código Fonte:

```
//Anti-horário
while (i < v)
{
    while (u < h)
    {
        nova_imagem.SetPixel(i, h-u-1, imagem.GetPixel(u,
i));
        u = u + 1;
    }
    u = 0;
    i = i + 1;
}

//Horário
while (u < h)
{
    while (i < v)
    {
        nova_imagem.SetPixel(v-i-1, u, imagem.GetPixel(u,
i));
        i = i + 1;
    }
    i = 0;
    u = u + 1;
}
```

Resultado:



## 2.4 Filtros

Os filtros são aplicados segundo o código fonte abaixo, a única mudança são os valores da convolação e o somar 127 para os filtros de relevo. As linhas e colunas das bordas são desprezadas na aplicação do filtro e seu valor permanece o mesmo.

Código Fonte:

```
while (i < v)
{
    while (u < h)
    {
        if (u == 0 || i == 0 || u == h-1 || i == v-1)
        {
            nova_imagem.SetPixel(u, i, imagem.GetPixel(u,
i));
        }
        else
        {
            colorR = imagem.GetPixel(u - 1, i - 1).R * 0.0625
+
            imagem.GetPixel(u, i - 1).R * 0.125 +
            imagem.GetPixel(u + 1, i - 1).R * 0.0625
+
            imagem.GetPixel(u - 1, i).R * 0.125 +
            imagem.GetPixel(u, i).R * 0.25 +
            imagem.GetPixel(u + 1, i).R * 0.125 +
            imagem.GetPixel(u - 1, i + 1).R * 0.0625
+
            imagem.GetPixel(u, i + 1).R * 0.125 +
            imagem.GetPixel(u + 1, i + 1).R *
0.0625;

            colorG = imagem.GetPixel(u - 1, i - 1).G *
0.0625 +
            imagem.GetPixel(u, i - 1).G * 0.125 +
            imagem.GetPixel(u + 1, i - 1).G * 0.0625
+
            imagem.GetPixel(u - 1, i).G * 0.125 +
            imagem.GetPixel(u, i).G * 0.25 +
            imagem.GetPixel(u + 1, i).G * 0.125 +
            imagem.GetPixel(u - 1, i + 1).G * 0.0625
+
            imagem.GetPixel(u, i + 1).G * 0.125 +
            imagem.GetPixel(u + 1, i + 1).G *
0.0625;

            colorB = imagem.GetPixel(u - 1, i - 1).B *
0.0625 +
            imagem.GetPixel(u, i - 1).B * 0.125 +
            imagem.GetPixel(u + 1, i - 1).B * 0.0625
+
            imagem.GetPixel(u - 1, i).B * 0.125 +
            imagem.GetPixel(u, i).B * 0.25 +
            imagem.GetPixel(u + 1, i).B * 0.125 +
            imagem.GetPixel(u - 1, i + 1).B * 0.0625
+
            imagem.GetPixel(u, i + 1).B * 0.125 +
            imagem.GetPixel(u + 1, i + 1).B * 0.0625;

            if (colorB > 255)
```

```
        colorB = 255;
    else if(colorB < 0)
        colorB = 0;

    if (colorR > 255)
        colorR = 255;
    else if (colorR < 0)
        colorR = 0;

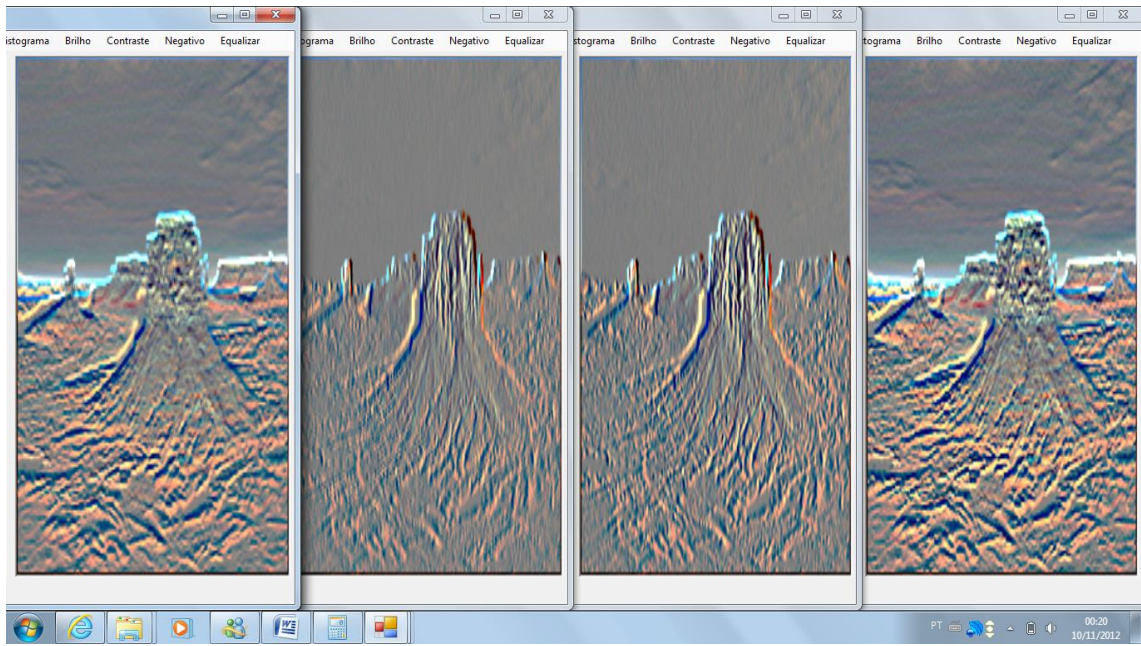
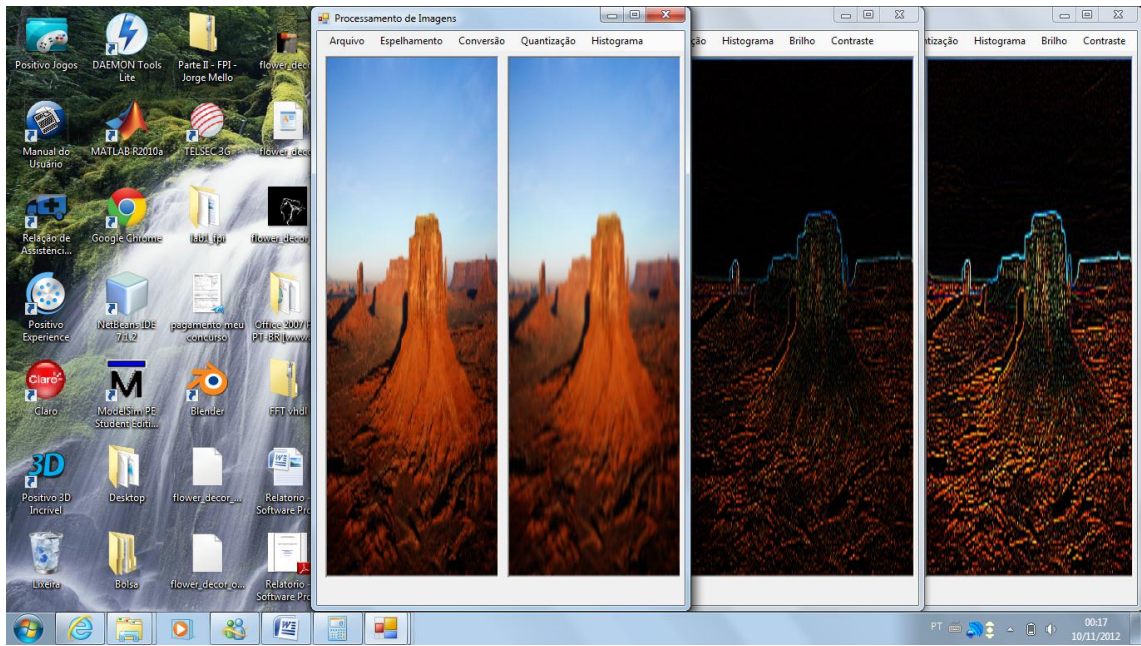
    if (colorG > 255)
        colorG = 255;
    else if (colorG < 0)
        colorG = 0;

        nova_imagem.SetPixel(u, i,
    Color.FromArgb(imagem.GetPixel(u, i).A, Convert.ToInt32(colorR),
    Convert.ToInt32(colorG), Convert.ToInt32(colorB)));
    }

    u = u + 1;

}
u = 0;
i = i + 1;
}
```

# Resultados:



### 3. Conclusão

Pude trabalhar vários conceitos, fazer um programa divertido, usar aquilo que eu julgo o melhor para trabalhar com software. Enfim usei muitos conhecimentos que aprendi durante a faculdade, foi realmente muito bom este trabalho e prazeroso.

Site : [www.inf.ufrgs.br/~jwkmello](http://www.inf.ufrgs.br/~jwkmello)

[www.inf.ufrgs.br/~jwkmello/tudo/trab\\_fpi.html](http://www.inf.ufrgs.br/~jwkmello/tudo/trab_fpi.html)