RELATÓRIO TRABALHO DE IMPLEMENTAÇÃO 2
Fundamentos de Processamento de Imagens

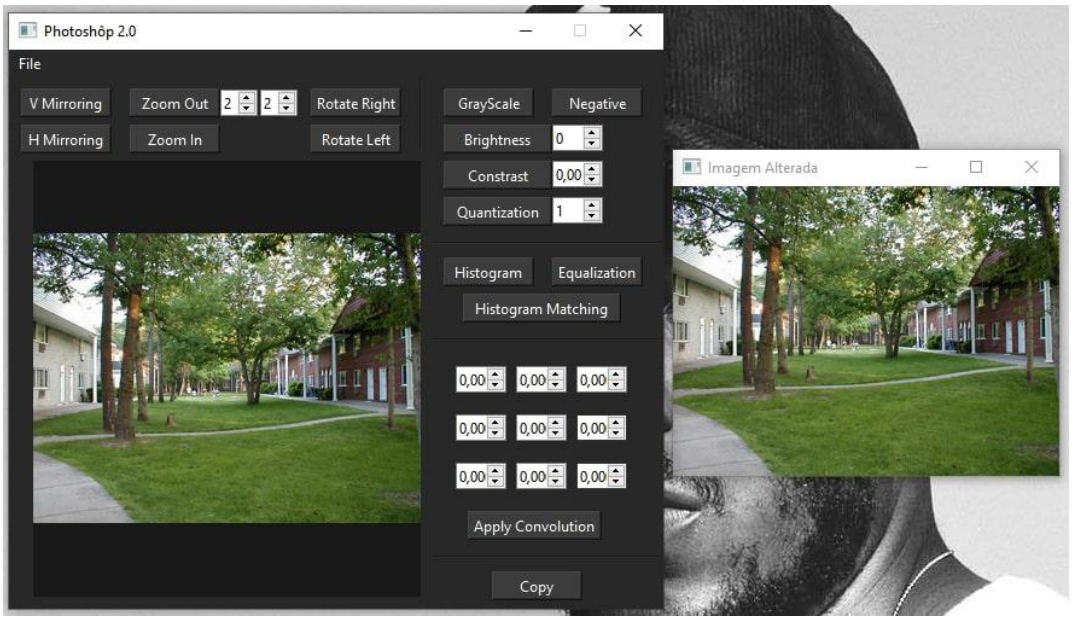
Thomas Wiederkehr
Turma B

INTRODUÇÃO

Este documento apresenta a segunda parte do trabalho prático da cadeira de Fundamentos de Processamento de Imagens. Esta segunda versão do software, apelidada “Photoshôp 2.0”, contém as operações da primeira parte, e mais outras solicitadas na definição do trabalho.

INTERFACE

Para poder aplicar as novas operações, foi necessário realizar atualizações na interface da primeira versão. Esta nova interface mantém um menu para manipulação dos arquivos e um label para a imagem que será modificada. Também mantém 5 botões para as operações antigas e mais 11 botões e 13 inputs para as operações novas (separando áreas relacionadas). Além disso, cria uma nova janela para a imagem alterada (funções como zoom out necessitam desta modificação).



Print da tela do Photoshôp 2.0

PARTE 1

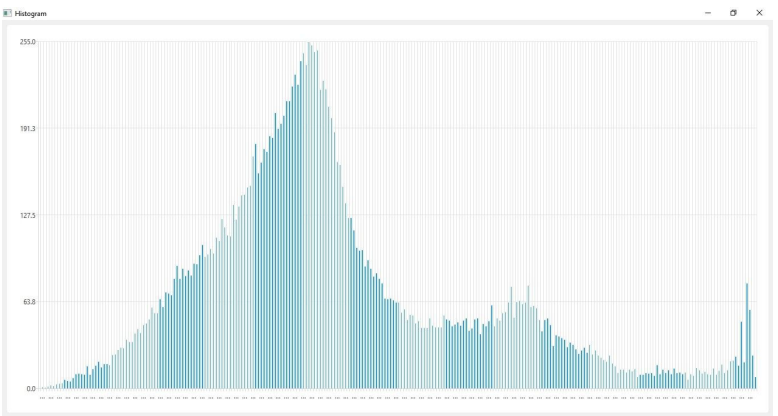
1) Histograma de uma imagem em tons de cinza

Para exibir o histograma foi utilizado um módulo do QT chamado de QT Charts, que permite a criação e manipulação simples de gráficos. O cálculo do histograma, implementado como um array de 256 posições, ocorre por meio da função *histogramCalc()*, que inicializa todas as posições do histograma com 0 e calcula a ocorrência de cada tom. Quando calculado, o histograma é exibido em um gráfico (x,y) onde o eixo x representa os 256 tons de uma imagem e o eixo y a ocorrência (normalizada) de cada tom. A normalização foi realizada utilizando o valor máximo do histograma, ao invés do número de pixels.

```
//Function to calculate histogram
void Photoshop::histogramCalc(unsigned int histogram[]){
    for (unsigned int i = 0; i < 256; i++){ //set histogram with 0
        histogram[i] = 0;
    }
    for (unsigned int i = 0; i < h; i++){
        for (unsigned int j = 0; j < w; j++){
            histogram[img2.pixelColor(j,i).red()]+=1; //count ocurrences of each tone
        }
    }
}
```

Função que calcula o histograma

Caso a imagem não esteja em tons de cinza, o software transforma ela antes de realizar as operações. Abaixo temos um exemplo de uma imagem e seu histograma:



2) Brilho

O valor escalar do brilho é informado por meio de um *SpinBox* que recebe números entre -255 e 255. Para calcular o brilho, basta somar em cada pixel da imagem, para cada uma cada um dos canais RGB, o valor informado pelo usuário, ajustando nos casos de borda (255 e 0).

Resultado ao somar 100 ao brilho

Resultado ao subtrair 100 do brilho

3) Contraste

Para o contraste foi utilizado um *SpinBox* que recebe valores reais (*double)* entre 0 e 255. O cálculo do contraste, de forma semelhante ao do brilho, consiste em apenas multiplicar os valores de cada pixel pelo valor concebido pelo usuário, ajustando nos casos excepcionais também.

Aplicação de contraste = 2

Aplicação de contraste = 0.5

4) Negativo

O cálculo também é relativamente simples. Basta subtrair de 255 o valor RGB de cada pixel da imagem, resultado na imagem a seguir:

5) Equalização

Para equalizar a imagem reutiliza-se o cálculo de histograma apresentado na primeira operação (convertendo a imagem para escala de cinza para tal). Então calcula-se o histograma cumulativo, somando cada valor do histograma a todos os valores anteriores, por meio de um laço *for*, e normalizando estes valores em outro *for*. Após isso, para cada canal RGB da imagem, igualamos o tom do pixel ao valor *y* associado pelo histograma cumulativo.

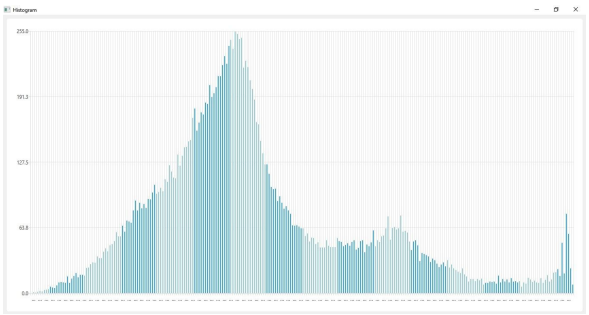
```
histogram_cum[0] = 255 * histogram[0]/(w*h);
for (int i = 1; i < 256; i++) {
    histogram_cum[i] = histogram_cum[i-1] + histogram[i];
}
for (int i = 0; i < 256; i++) {
    histogram_cum[i] = (255*histogram_cum[i])/(w*h);
}
```

Calculo do histograma

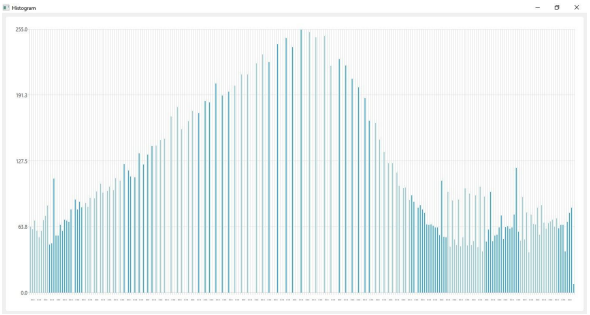
Se a imagem estiver em escala de cinza, são exibidos o histograma da imagem sem equalização e o histograma pós equalização. Em caso contrário, não é exibido nenhum.

Equalização em Imagem cinza

Equalização em imagem colorida



Histograma antes da equalização



Histograma pós equalização

6) Matching de Histogramas

Para realizar o matching de histogramas de 2 imagens, o usuário deve fornecer a imagem destino por meio de uma janela de carregamento (como a da opção *Open File*). Ao abrir a imagem, as duas são convertidas para luminância, se ainda não forem, e calcula-se os histogramas e histogramas cumulativos de ambas, por meio das mesmas funções utilizadas na equalização. Então, a imagem original é percorrida e, para cada tom, é encontrado um tom da imagem destino com o número de ocorrências mais próximo (utilizando o módulo da diferença das ocorrências), que substitui o tom original.

```
for (unsigned int i = 0; i < h; i++){
    for (unsigned int j = 0; j < w; j++){
        int min_x = 0;
        int dif = fabs(histogram_cum[img2.pixelColor(j,i).red()] - histogram_cum2[min_x]);
        for (unsigned int x = 1; x < 256; x++){ //finds target cumulative histogram closest value
            if (fabs(histogram_cum[img2.pixelColor(j,i).red()] - histogram_cum2[x]) < dif){
                dif = fabs(histogram_cum[img2.pixelColor(j,i).red()] - histogram_cum2[x]);
                min_x = x;
            }
        }
        QColor newcolor;
        newcolor.setRed(min_x); //sets each RGB tone to histogram_cum's tone associated
        newcolor.setBlue(min_x);
        newcolor.setGreen(min_x);
        img2.setPixelColor(j,i,newcolor);
    }
}
```

Cálculo dos tons destino

Imagem Original

Imagem Destino

Resultado do Matching

PARTE 2

7) Zoom Out

Foram adicionadas 2 *SpinBox* do tipo int para que o usuário possa informar os fatores de redução *sx* e *sy*. Ambas possuem um valor mínimo de 1 e valor máximo de 1 bilhão e 600 milhões. Porém, o programa apenas realiza a operação se os fatores não forem maiores do que o tamanho da imagem correspondente.

Ao clicar no botão, é criada uma nova imagem com resolução de *altura/sx* e *largura/sy*, sendo *altura* e *largura* os tamanhos da imagem original. Então, a imagem original é percorrida (por meio de um laço duplo *for*) calculando a média dos valores RGB dos pixels que se encontram dentro da matriz sx/sy e substituindo no pixel correspondente da nova imagem. Este cálculo é realizado por meio de um outro laço *for* que percorre a matriz sx/sy, somando os valores dos pixels e contando o número de pixels percorridos para os casos em que a matriz “atravessar” a imagem.

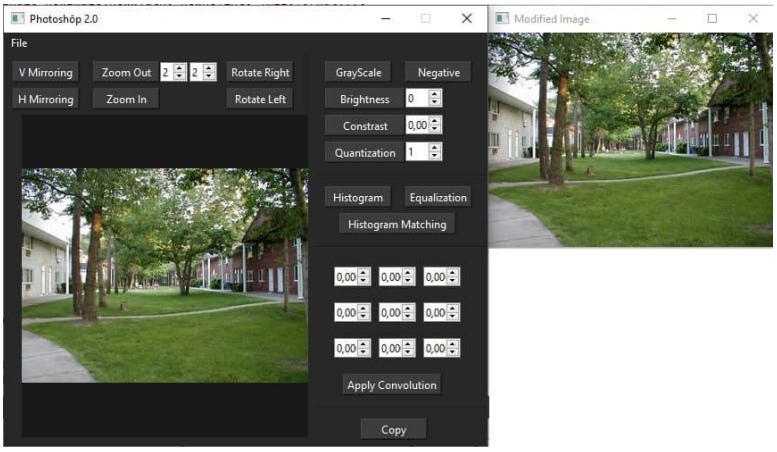
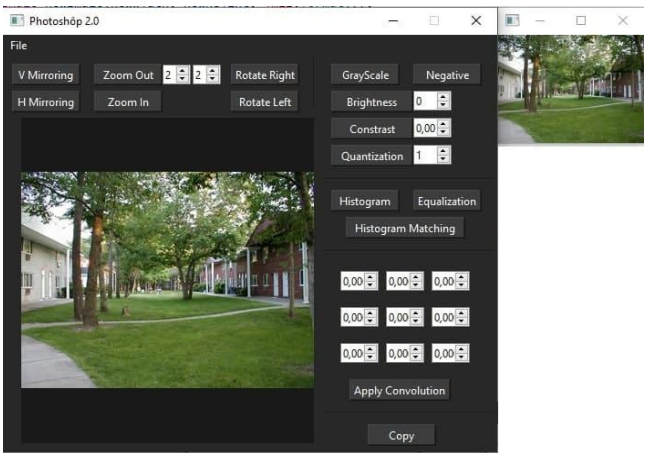
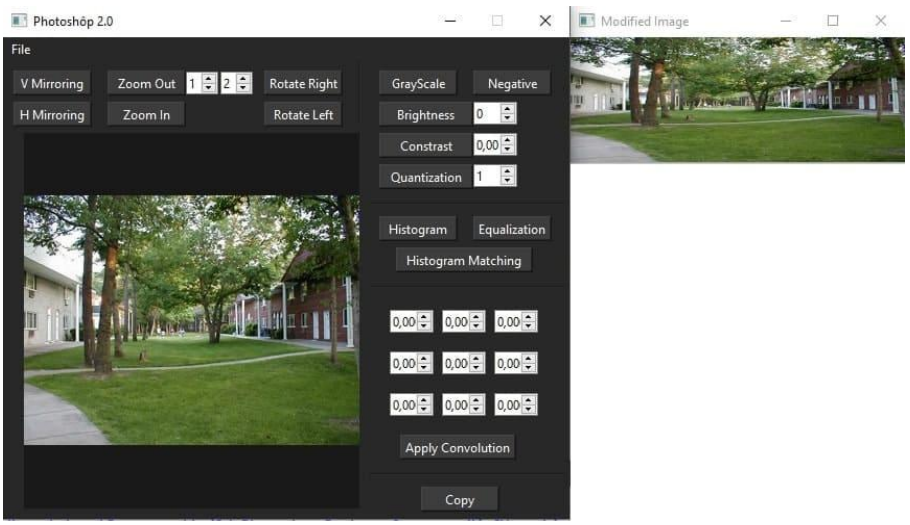


Imagem Original



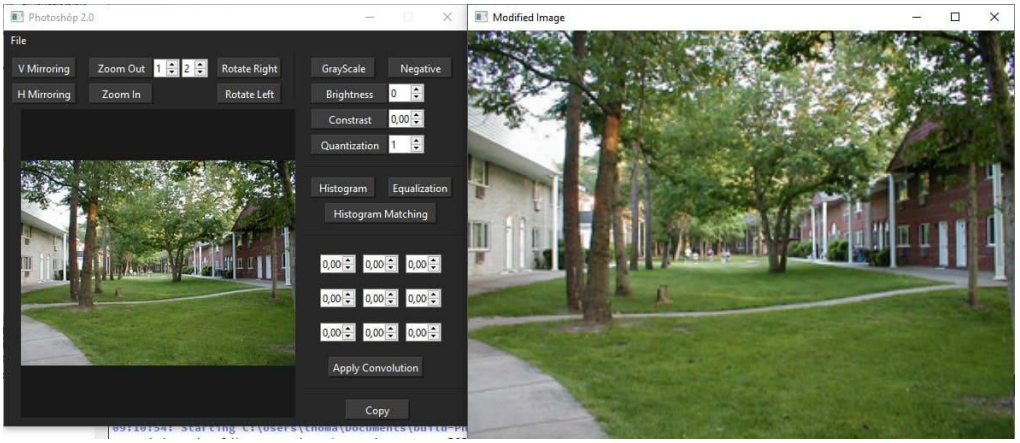
Zoom Out de 2 x 2



Zoom Out de 2 x 1

8) Zoom In

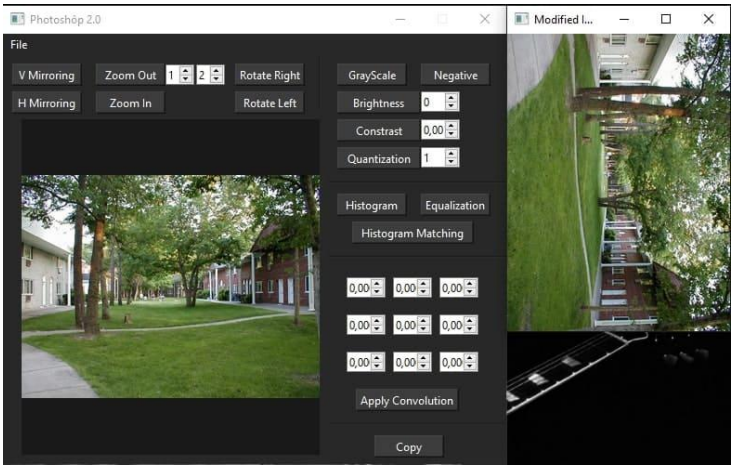
No caso do Zoom In, nenhum outro valor é necessário. Apenas cria-se uma nova imagem com tamanhos *altura*2-1* e *largura*2-1*. Então, percorre-se a imagem original, fixando cada pixel (x,y) no pixel (x*2, y*2) da nova imagem. Na mesma iteração, calcula-se a média dos valores do pixel atual com os valores dos pixels anteriores (x-1,y) e (x,y-1), se existirem, posicionando-os nas posições (x*2-1, y*2) e (x*2, y*2-1) da nova imagem, respectivamente. Após isso, também é calculado o valor do pixel (x*2-1, y*2-1), por meio da media dos pixels (x*2-2, y*2-1) e (x*2, y*2-1), já calculados.



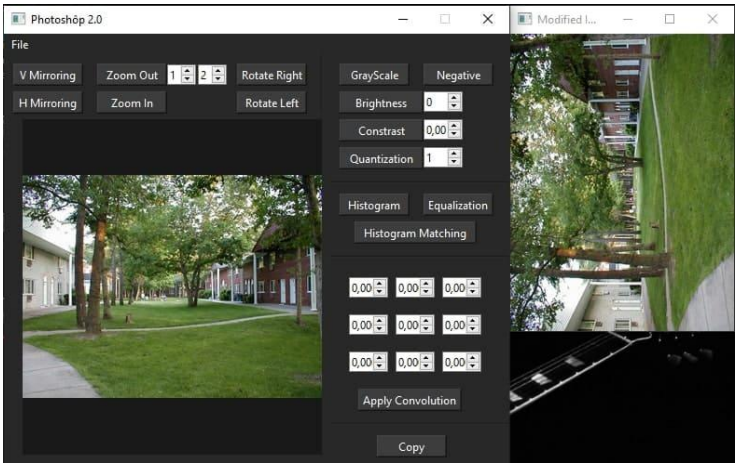
Zoom In

9) Rotação a Direita e a Esquerda

A aplicação das rotações é relativamente mais simples. Basta criar uma nova imagem com resolução *largura x altura*, sendo *altura x largura* a resolução da imagem original. Então, os pixels (x,y) da imagem original são inseridos nas posições (altura-1-y,x), para a rotação a direita, ou nas posições (y, largura-1-x), para a rotação a esquerda.



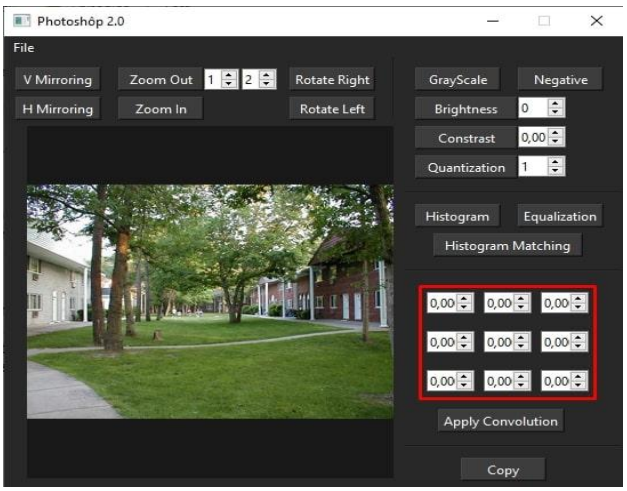
Rotação a direita



Rotação a esquerda

10) Convolução

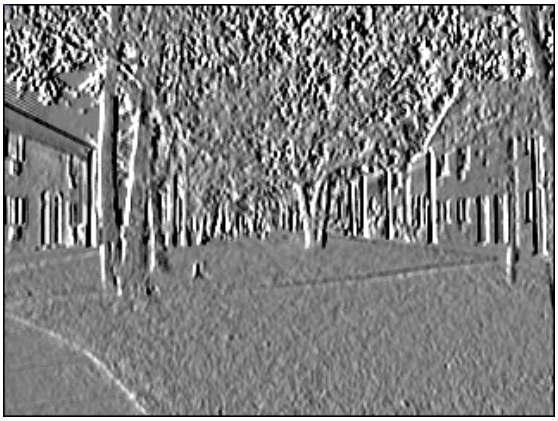
Para a convolução, foi criada uma matriz de *inputs* de valores fracionários, onde o usuário fornece o filtro que deseja aplicar a imagem. Estes valores vão de -255 a 255 e possuem até 2 casas decimais.



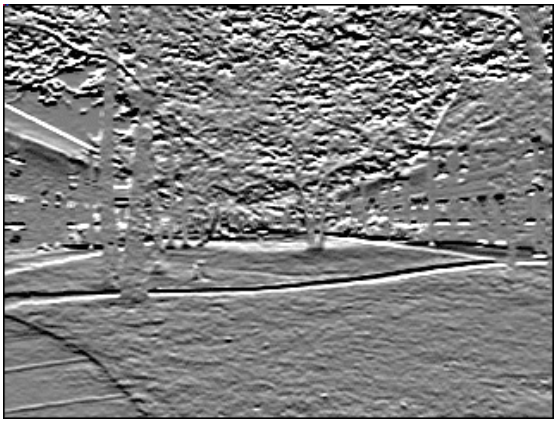
Ao pressionar o botão da convolução, o valor de cada *QDoubleSpinBox* é passado para a matriz 3x3 chamada de *kernel*. Então, percorre-se a imagem, sem passar pelas bordas, e multiplica-se os valores RGB do pixel (x,y) e dos 8 pixels em sua volta com os valores do *kernel* (rotacionado em 180º) correspondentes. Estes valores são somados, e os pixels da nova imagem são substituídos por essa soma, ajustando nos casos em que o resultado é maior que 255 ou menor que 0. Os filtros especiais são detectados pelo programa, mas devem ser passados pelo usuário.

Aplicação do filtro Gaussiano

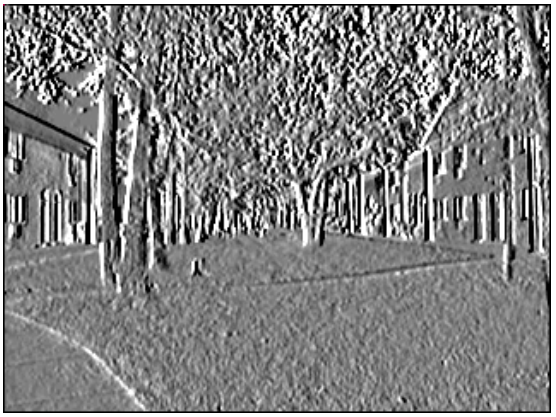
Aplicação do filtro Laplaciano



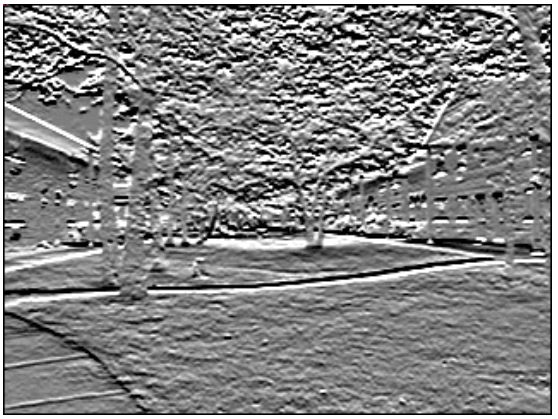
Aplicação do Prewitt Horizontal



Aplicação do Prewitt Vertical



Aplicação do Sobel Horizontal



Aplicação do Sobel Vertical