Fundamentos de Processamento de Imagens

Trabalho 1 – Parte 2

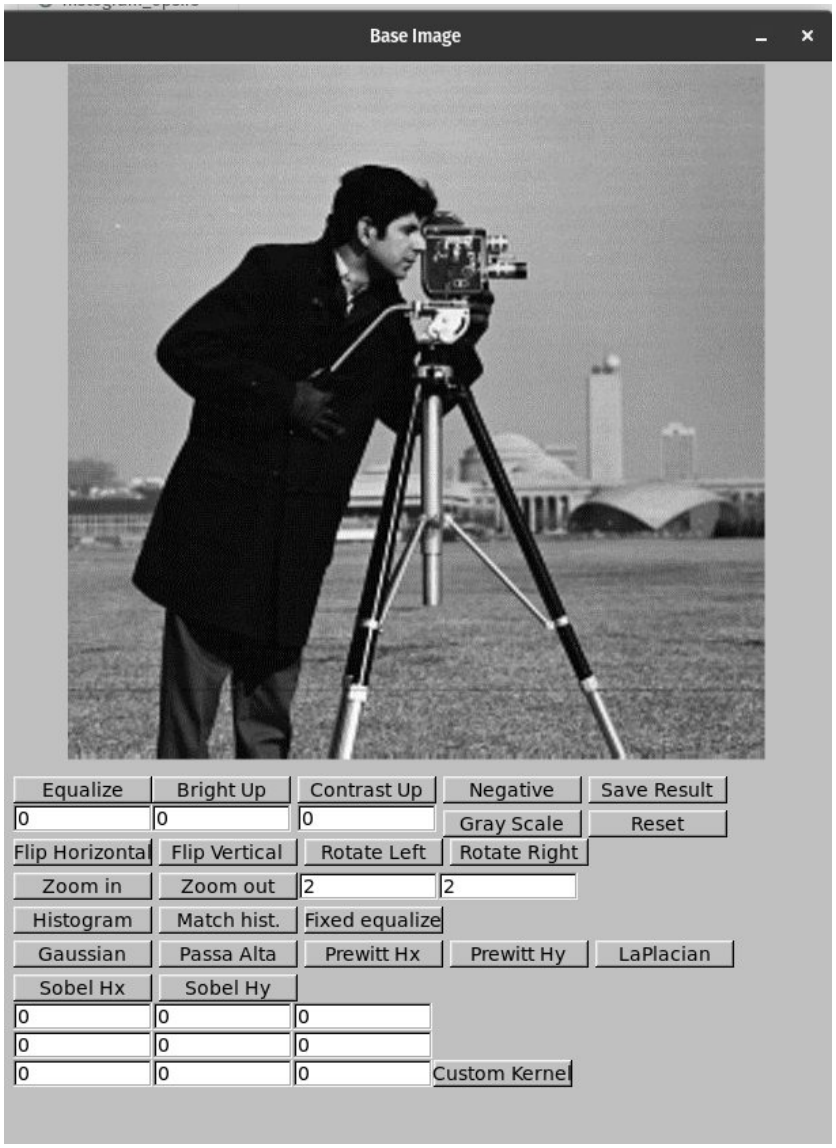
Nome: João Pedro Cosme da Silva / Cartão 00314792

Resumo

Este trabalho tem como objetivo a familiarização com conceitos vistos nas aulas da disciplina de Fundamentos de Processamento de Imagens. Tendo como base a primeira entrega do trabalho, esta etapa apresentou a introdução de conceitos como: operações ponto a ponto, operações de vizinhança e operações baseadas no cálculo de histograma.

1 Parte 1

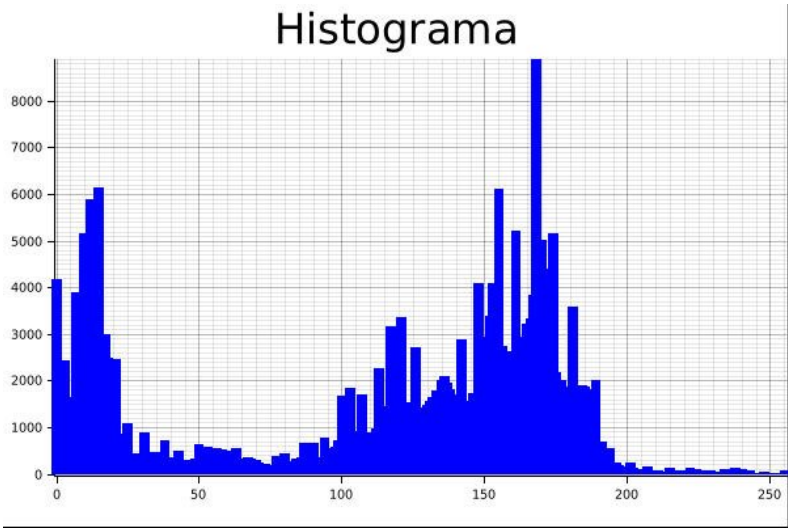
Com a introdução de novas operações, a nova interface do programa é a seguinte:



1.1 Cálculo de Histograma

Tendo como base qualquer imagem, a mesma irá ser convertida em tons de cinza, e em então seu histograma pode ser visto.

Tendo como base a imagem do fotografo, utilizada também nos laboratórios, podemos utilizar o botão “Histograma” e então o seguinte resultado é gerado:



O cálculo é feito utilizando o algoritmo visto em aula, onde um array de tamanho 256 é criado, e sempre que um tom é encontrado a posição do array para aquele tom é incrementada em 1.

1.2 Ajuste do Brilho de Imagem

Podendo ser aplicada em qualquer tipo de imagem, o ajuste do brilho de imagem é feito através da operação linear g(x) = a * f(x) + b, onde a é o *ganho* e b é o *bias*, que é aplicada pixel a pixel.

No caso de ajuste de brilho de uma imagem, o ganho é fixado em **1** e o bias é variado de acordo com a entrada do usuário. Na interface do programa, podemos ver que existe um campo de texto, que permite a entrada de um valor entre [-255,255], permitindo ao usuário fazer o ajuste fino desta operação.

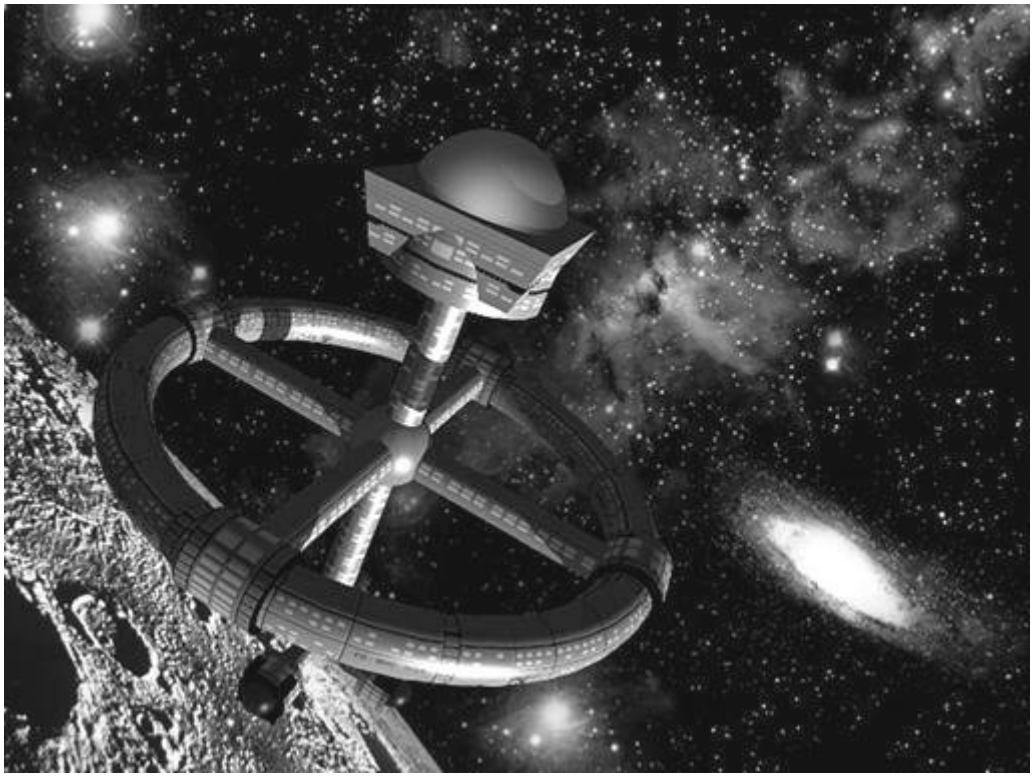
Podemos ver o resultado aplicando o aumento/redução de brilho enquanto imagem colorida:

Adicionando +15:

Reduzindo 50:

E em sua versão em tons de cinza:

Aumentando em 20 o brilho



Reduzindo em 60:

1.3 Ajuste de Contraste

Usando a mesma operação linear vista no item 1.2, desta vez fixamos o bias em zero e o usuário pode inserir um valor qualquer no campo de entrada. Como visto em aula, quando o ganho é menor que 1, o contraste é reduzido e quando é maior que 1 é incrementado. O usuário pode inserir os valores contidos no intervalo (0.0,255.0].

Abaixo, segue o resultado da operação de ajuste de contraste tanto em imagens coloridas:

- Aplicando 0.75:

- Aplicando 1.8:

1.4 Cálculo de Negativo

Usando a operação linear vista nos itens 1.2 e 1.3, fixamos o bias em 255 e o ganho em -1, aplicando a operação em cada pixel de cada canal da imagem para que se possa obter o negativo da imagem:

- Exemplo Colorido

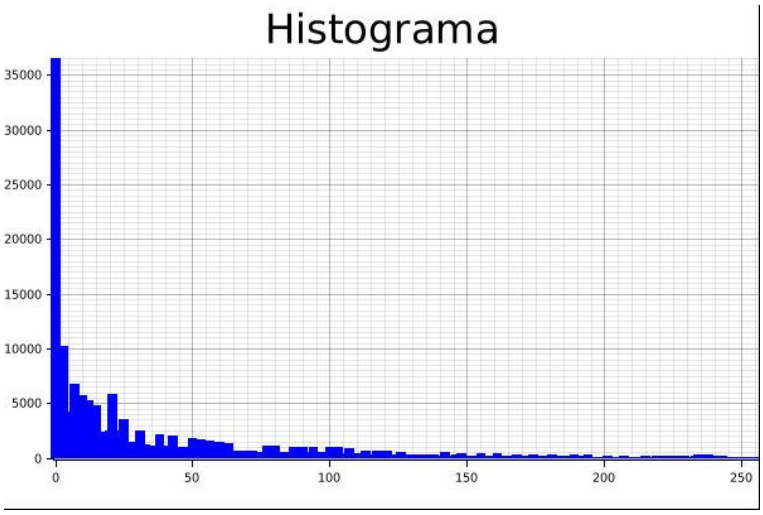
- Exemplo em tons de cinza

1.5 Equalização de Histograma

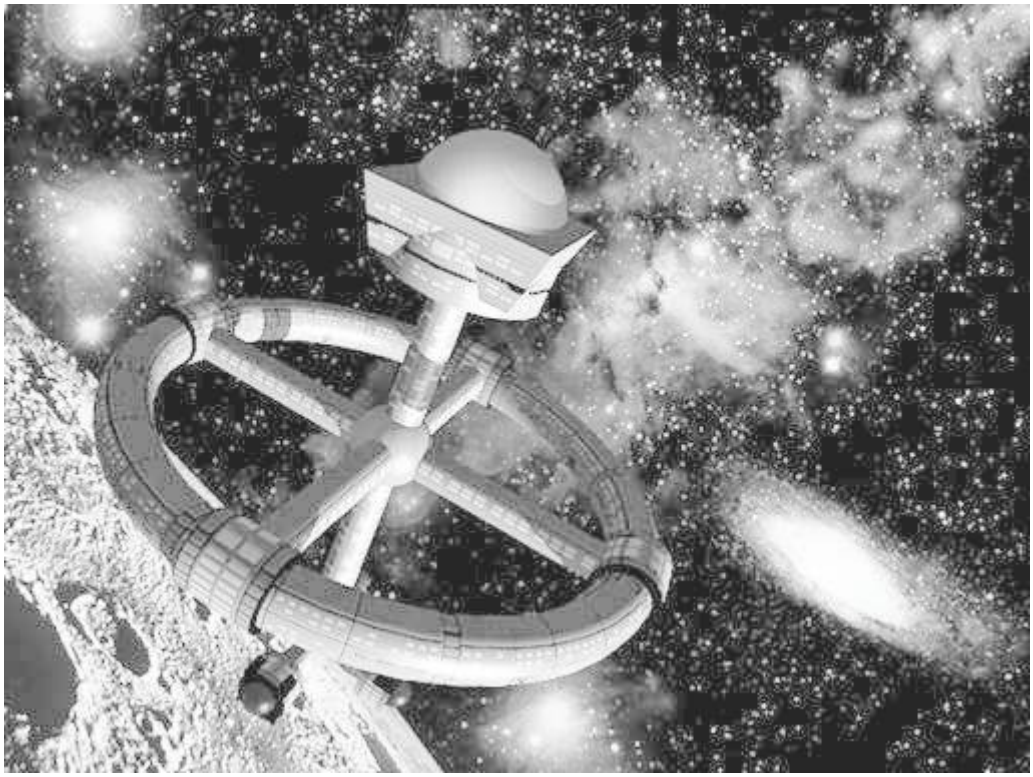
Utilizando a técnica vista em aula, onde primeiro é computado um histograma cumulativo de uma imagem (isto é, acumulando os valores de cada pixel para que se possa verificar onde existe maior concentração de certo tom) e então normalizando esse valor em 255 (valor máximo de um pixel), podemos obter um histograma mais homogêneo e um histograma cumulativo sem grandes inclinações. Esta operação pode ser ativada através do botão “Fixed Equalization”, dado que na parte 1 o botão de Equalização esperaria um valor arbitrário de bins para ajuste.

- Imagem Base

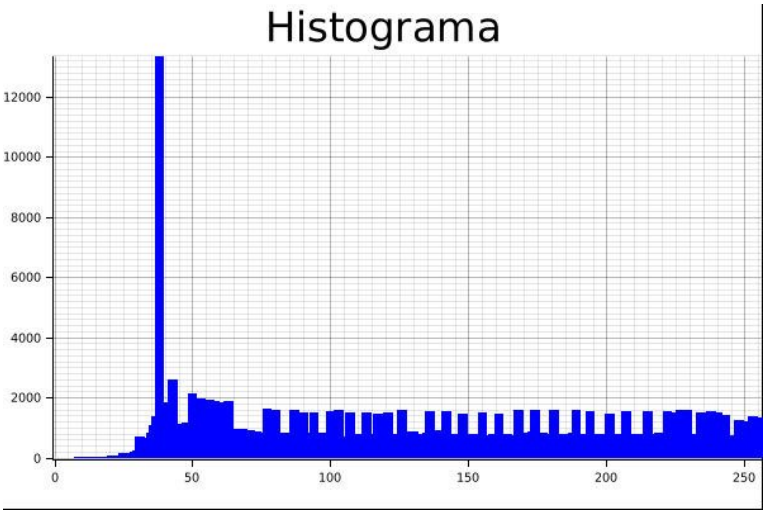
- Histograma



- Imagem Equalizada



- Histograma



Como a imagem base possui uma alta concentração de tons escuros, podemos ver que esta concentração se manteve no resultado, porém, os demais tons se apresentam bem melhor distribuídos do que foi visto anteriormente.

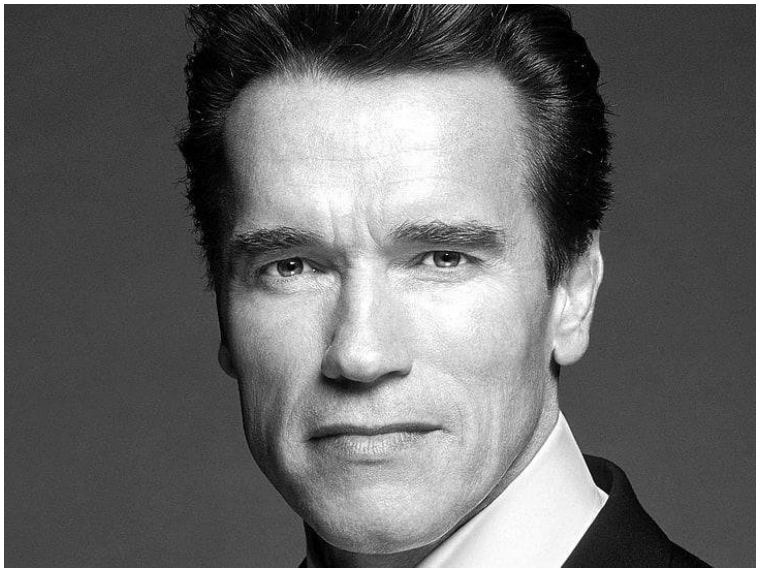
Já para imagens coloridas, tendo como base a seguinte imagem:

Resultado da equalização:

1.6 Histogram Matching

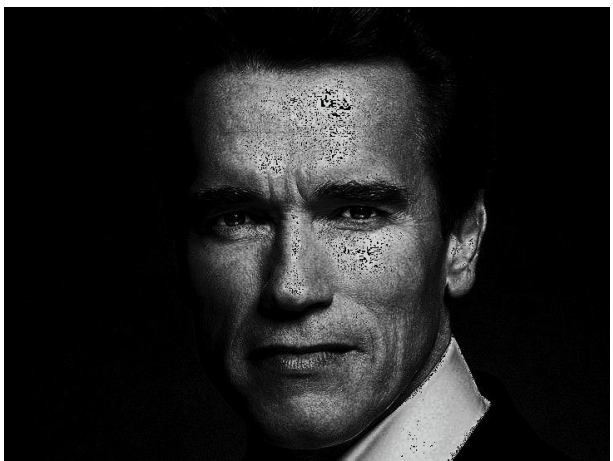
Esta técnica é utilizada para tentar mimicar o efeito/tons de uma imagem em outra. O algoritmo se baseia no cálculo do histograma cumulativo de ambas as imagens, e então realizando um mapeamento de aproximação entre tons: vê-se qual tom na imagem objetivo tem uma concentração similar de pixeis e então mapeia-se o tom base para este tom encontrado. Abaixo,p odemos ver alguns exemplos dessa operação em imagens de luminância.

Base:



Alvo:

Resultado:



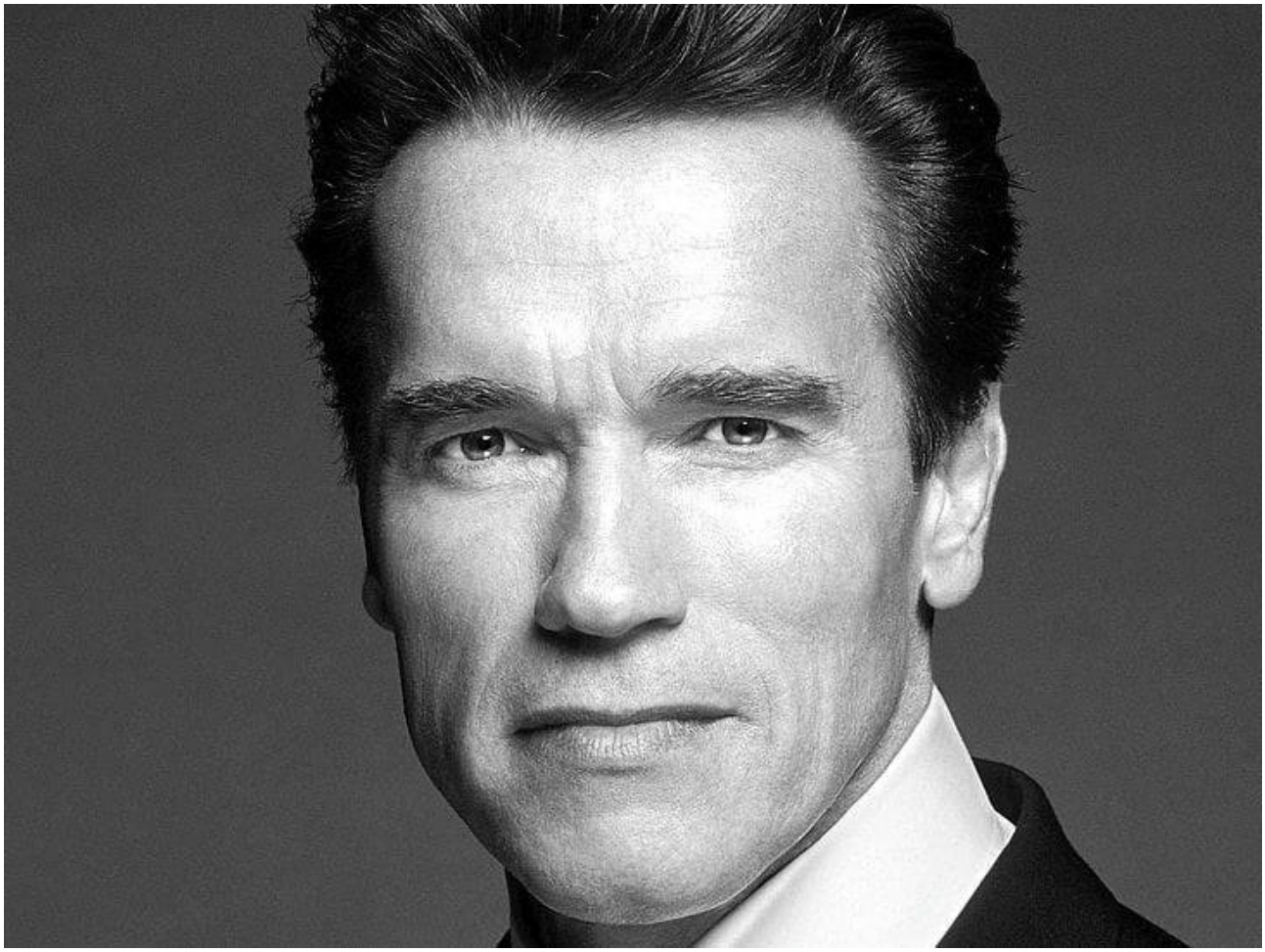
Podemos ver que o efeito “foto no escuro” foi replicado, mas diversos ruidos surgem devido a alta diferença entre os histogramas das duas imagens e a alta concentração de tons mais escuros na imagem alvo.

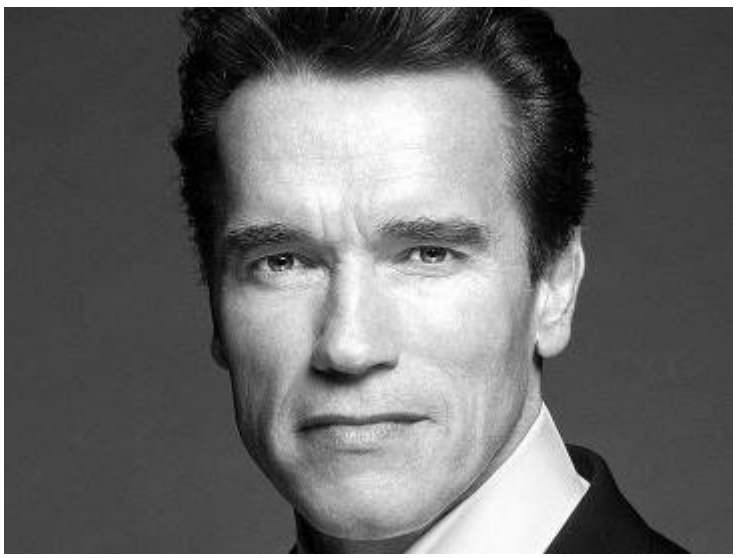
2 Parte 2

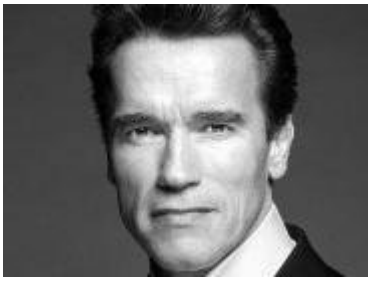
2.1 Zoom Out

Utilizando um operação de vizinhança onde, a partir de um sx/sy entrado pelo usuário, cria-se uma matriz de dimensões *sx* x *sy* e percorremos a imagem com este “filtro”, onde computamos a média dos valores dos pixeis dentro desta matriz, cria-se uma imagem reduzida pelos fatores entrados. Esta é uma operação de vizinhança, similar a outras que veremos a seguir, onde cada operação é aplicada sobre uma série de pixeis vizinhos para computar um resultado.

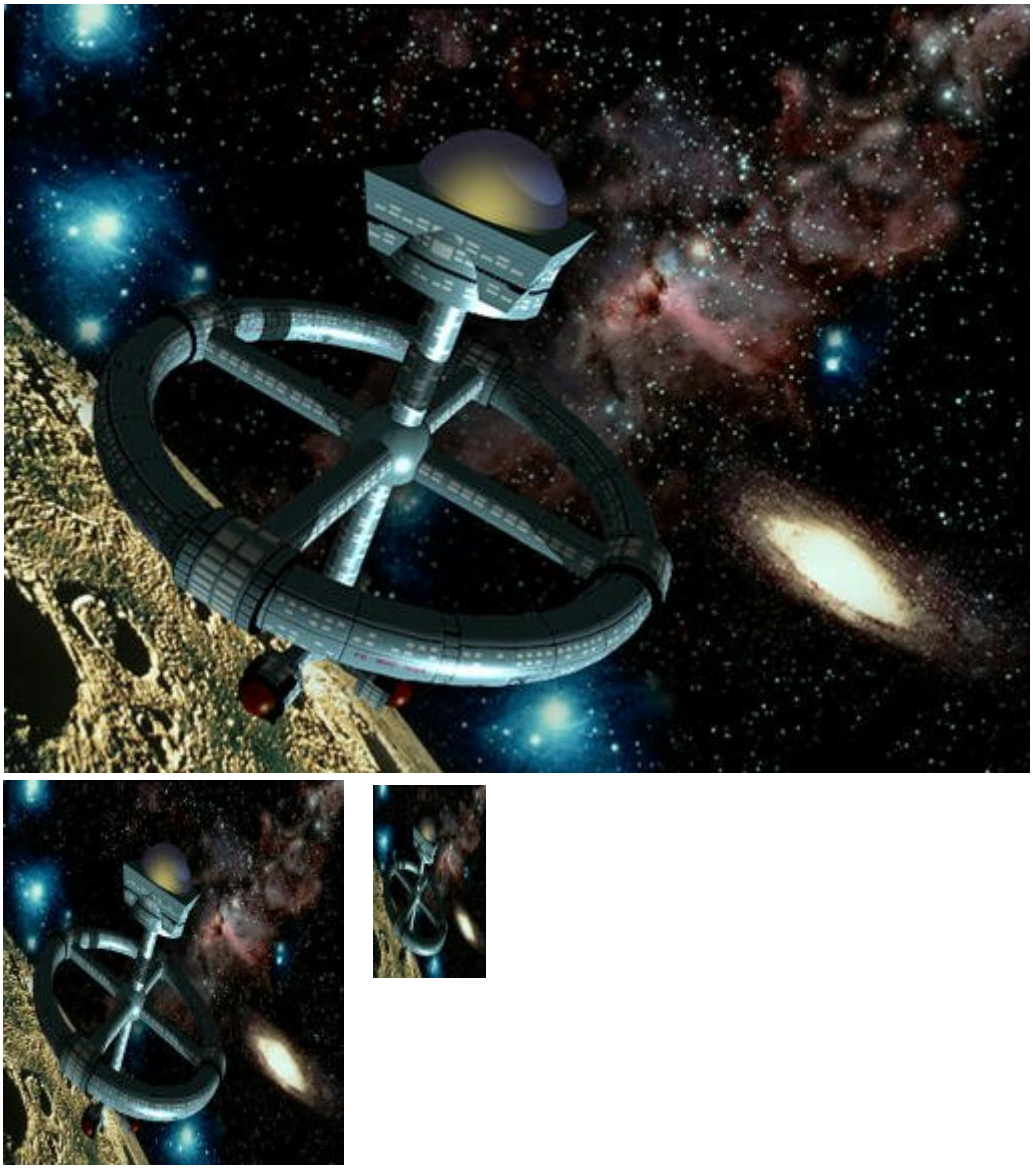
Resultado de zoom out de imagem cinza em 2x2(Imagem original, um zoom out, dois zoom outs):







Resultado de zoom out de imagem colorida em 3x2:



2.2 Zoom In

Diferente de operação de zoom in, onde reduz-se o espaço de imagem e, consequentemente, temos menos pixeis e devemos computar suas médias, a operação de zoom in aumenta o espaço de imagem e devemos preencher os pixeis “virtuais” que não existiam na imagem base. Para isso, interpola-se os vizinhos de cada pixel para que seu novo valor seja uma média entre eles.

Resultado do zoom in em imagem tons de cinza:

Resultado do zoom out em imagens coloridas:

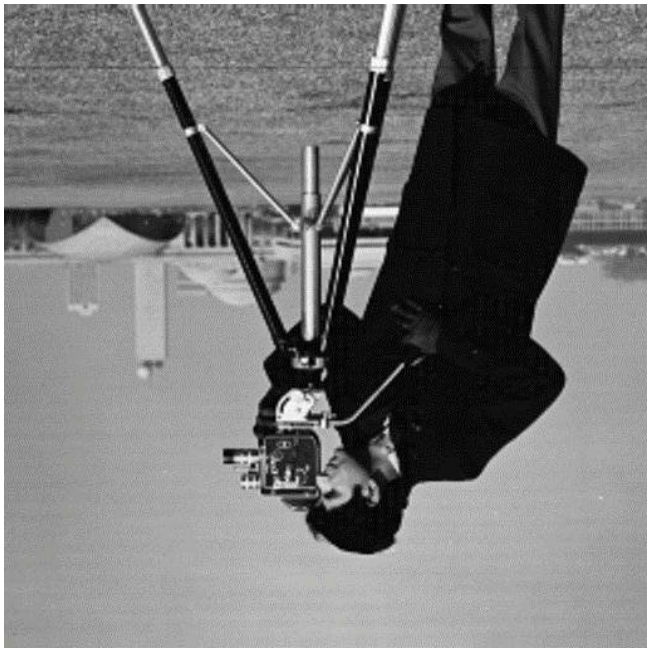
2.3 Rotação de Imagem

Similar ao que foi feito na Parte 1 deste trabalho, esta é uma operação pixel a pixel onde deslocar todos os pixeis de uma imagem para que possamos ter um efeito de rotação. A nova posição de cada pixel é dado pelo seguinte cálculo

- Para a esquerda, x2=y1 e y2=altura_da_imagem-x-1

- Para a direita, x2=largura_da_imagem-y-1 e y2=x

Resultado de rotações:



2.4 Aplicação de Convolução

A convolução é uma operação onde se um filtro é aplicado em um sinal. No dominio do processamento de imagens, isto é realizado através de uma operação de vizinhança, onde o filtro é representado por uma matriz que é “deslocada” através da imagem para que se possa computar a convolução de um pixel junto de seus pixeis vizinhos. O novo valor de um pixel, é dado através da soma da operação de cada pixel multiplicado de seu *peso*, que é definido no filtro (que também é chamado de kernel).

2.4.1 Filtro Gaussiano

Efeito “borrado”, resultado após 4 aplicações:

2.4.2 Filtro LaPlaciano

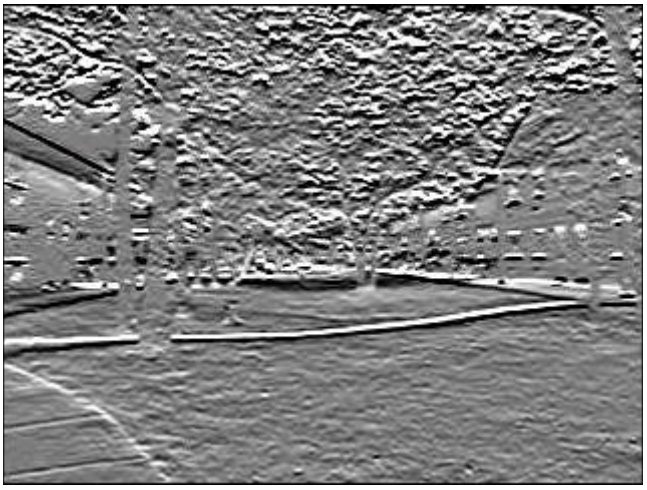
Detecção de Arestas:

2.4.3 Filtro Passa Alta

Detecção de arestas mais sensivel que o filtro LaPlaciano.

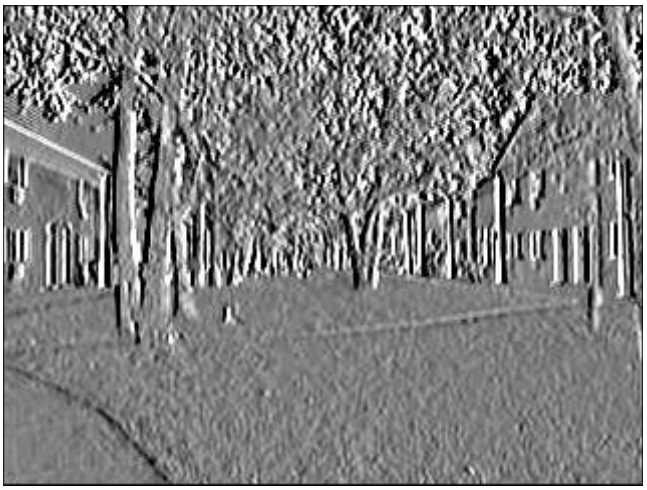
2.4.4 Filtro Prewitt Horizontal

Alto relevo com foco nas arestas horizontais.



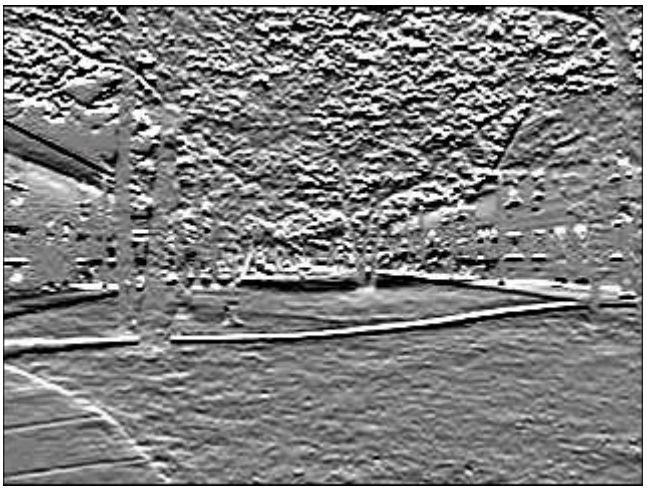
2.4.5 Filtro Prewitt Vertical

Alto relevo com foco nas arestas verticais.



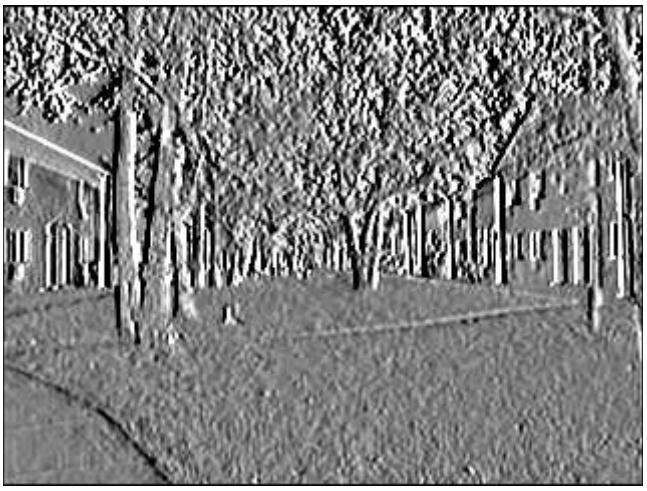
2.4.6 Filtro Sobel Vertical

Alto relevo, mas com foco nas arestas horizontais. Os filtros sobel se mostraram mais sensiveis a ruidos.



2.4.7 Filtro Sobel Horizontal

Alto relevo, mas com foco nas arestas verticais.



2.4.8 Kernel Arbitrário

Ainda, o programa nos permite entrar kernels arbitrários, um exemplo é o kernel abaixo:

-1	-1	-1
-1	9	-1
-1	-1	-1

Que nos retorna:

3 Conclusão

A execução deste trabalho me deu a oportunidade de ver em mais baixo nível como implementar operações como a convolução (que já haviamos utilizados em um nivel mais alto no laboratório 1), e a enfrentar desafios com a aquisição de pixels em uma linguagem que trata a imagem como um objeto e não como uma matriz 3D, o que fez com que a implementação da passagem de um filtro de convolução se tornasse mais trabalhosa que somente realizar o *fetch* de valores em uma matriz.

Outro ponto importante foi a da implementação de operações que não foram vistas em aula (como zoom in e zoom out), mas que se mostraram objetivas de serem implementadas e incrementaram meu entendimento sobre essas ferramentas que estamos tão acostumados a usar todos os dias.