

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL - UFRGS
CIÊNCIA DA COMPUTAÇÃO

*TRABALHO 1: CONCEITOS INICIAIS DE
PROCESSAMENTO DE IMAGENS E APLICAÇÕES*

INF01046 – FUNDAMENTOS DE PROCESSAMENTO DE IMAGENS

Marcelo Basso
00579239

Porto Alegre - RS
2024/01

SUMÁRIO

INTRODUÇÃO	1
SUMÁRIO	2
PARTE I	3
Leitura e Gravação de Arquivos de Imagens	3
PARTE II	4
Espelhamento horizontal e vertical	4
Conversão de imagem colorida para tons de cinza	5
Quantização (de tons) sobre as imagens em tons de cinza	6
Salvamento da imagem em um arquivo JPEG	8
CONCLUSÃO E INTERFACE	9

INTRODUÇÃO

Este trabalho tem por objetivo o estudo de funções básicas para a manipulação de imagens, como leitura e escrita em disco, além da implementação de operações pontuais (*píxel a píxel*) sobre as imagens carregadas, como a transformação dos tons da imagem para escala de cinza (*grayscale*), a quantização dos tons (com a quantidade máxima determinada pelo usuário), e, por fim, a inversão da imagem nos eixos verticais e horizontais.

Para alcançar esse objetivo, implementou-se uma aplicação simples na linguagem C++, utilizando como compilador o g++ na versão 13.2.0, e a biblioteca GTK versão 3 para a manipulação de janelas e construção de interface gráfica.

O trabalho é dividido em duas partes: a primeira diz respeito à implementação de leitura e gravação de imagens, enquanto a segunda expõe o desenvolvimento das funcionalidades incluídas no projeto, assim como imagens que ilustram as implementações.


PARTE I



Leitura e Gravação de Arquivos de Imagens

Para realizar a leitura de imagens, implementou-se a função `open_image()` que, ao ser invocada pelo usuário, abre uma janela para que o mesmo possa selecionar a imagem desejada. A função que realizar o carregamento da imagem em si é a `gtk_image_set_from_file()`, que, dada uma imagem de destino (`GtkImage*`) e o caminho da imagem, a carrega para a variável.

Já para a escrita no disco, criou-se a função `save_image()`, que utiliza o método `gdk_pixbuf_save()` e realiza o salvamento da imagem no disco no formato JPEG.

Nota-se, no entanto, uma diferença nos tamanhos das imagens originais e daquelas salvas após o carregamento. Após uma rápida pesquisa, constatou-se que o formato JPEG é comumente conhecido por armazenar imagens de forma mais compacta, reduzindo seu tamanho. No programa implementado, contudo, a função que realiza o salvamento recebe o parâmetro “*quality*” com valores de 0 a 100, que indica o nível de compressão da imagem salva. Nos testes, utilizou-se o parâmetro com valor 100 e, por isso, as imagens salvas acabaram ficando maiores que as imagens originais.



	cute_dog.jpg	45,0 kB	Today 15:45	☆
	cute_dog_edited.jpg	29,2 kB	Today 19:11	☆
	Gramado_72k.jpg	73,4 kB	14 fev 2001	☆
	Gramado_72k_edited.jpg	90,2 kB	Today 19:00	☆

“cute_dog_edited.jpg” foi salva com o parâmetro quality em 50, enquanto

“Gramado_72k_edited” com quality em 100.

Concluindo, essa etapa foi finalizada com êxito e as imagens podem ser carregadas e salvas em disco após a modificação.

PARTE II

Espelhamento horizontal e vertical

Para realizar o espelhamento horizontal, foi necessário implementar uma função que percorresse a imagem substituindo pares de píxeis. A ideia aqui foi percorrer a imagem linha a linha, do primeiro píxel à esquerda até a metade da imagem, trocando cada píxel com o seu correspondente no extremo direito da imagem.



“Space_187k.jpg” antes e depois do espelhamento horizontal.

Já para inverter a imagem no eixo vertical, realizou-se a troca de uma linha completa de píxeis pela sua correspondente no outro extremo vertical da imagem. Para isso, utilizou-se a função **memcpy()**.



“Uderwater_53k.jpg” antes e depois do espelhamento vertical.

Conversão de imagem colorida para tons de cinza

Para implementar a transformação para tons de cinza, utilizou-se a fórmula:

$$L = 0.299 * RED + 0.587 * GREEN + 0.114 * BLUE,$$

onde RED, GREEN e BLUE representam os valores dos canais 0, 1 e 2 da imagem. O resultado L é o novo valor atribuído simultaneamente para todos os canais, fazendo com que a imagem se transforme em variações de tons de cinza.

Caso essa transformação seja aplicada a uma imagem que já é uma variação de tons de cinza, nenhuma mudança acontecerá, uma vez que todos os canais terão o mesmo valor:

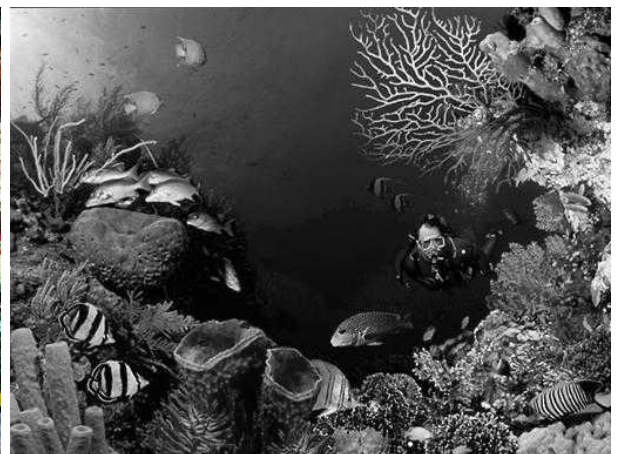
$$L = 0.299 * TONE + 0.587 * TONE + 0.114 * TONE$$

$$L = TONE * (0.299 + 0.587 + 0.114)$$

$$L = TONE * 1$$



“Gramado_72k.jpg” antes e depois da transformação para escala de cinza.



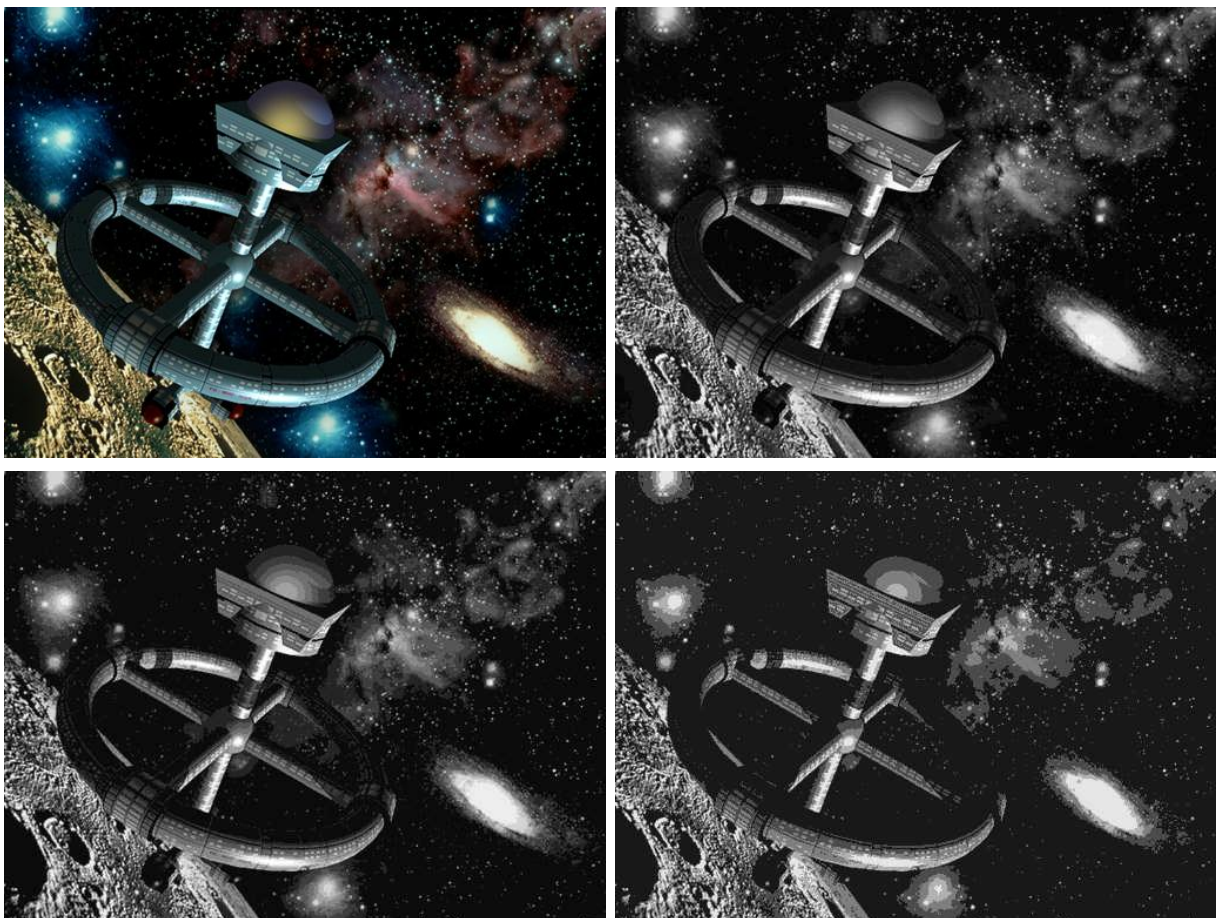
“Underwater_53k.jpg” antes e depois da transformação para escala de cinza.

Quantização (de tons) sobre as imagens em tons de cinza

Para implementar a quantização de tons dado um valor inteiro fornecido pelo usuário, primeiro realizou-se a conversão da imagem para escala de cinza, já discutida no tópico anterior. Após esse passo, utilizou-se a informação do intervalo de valores da imagem e, a partir dele, calculou-se o tamanho de cada subseção dentro do intervalo de tons da imagem (`bin_size`), dado o número de tons finais desejados. Considerando que todos os canais da imagem possuem o mesmo valor (por estar em escala de cinza), utilizou-se o valor do canal 0 para realizar o seguinte cálculo:

```
L = (int) (pixel[0] / bin_size) * bin_size + (bin_size / 2);
```

Onde L é o novo valor de todos os canais do píxel processado. A soma da metade do valor no final da fórmula visa situar o valor no ponto médio da subseção, fazendo com que a distribuição seja mais uniforme no intervalo definido.



“Space_187k.jpg” original e quantizado com 20, 10 e 5 tons, respectivamente.



“cute_dog.jpg” original e quantizado com 20, 10 e 5 tons, respectivamente.





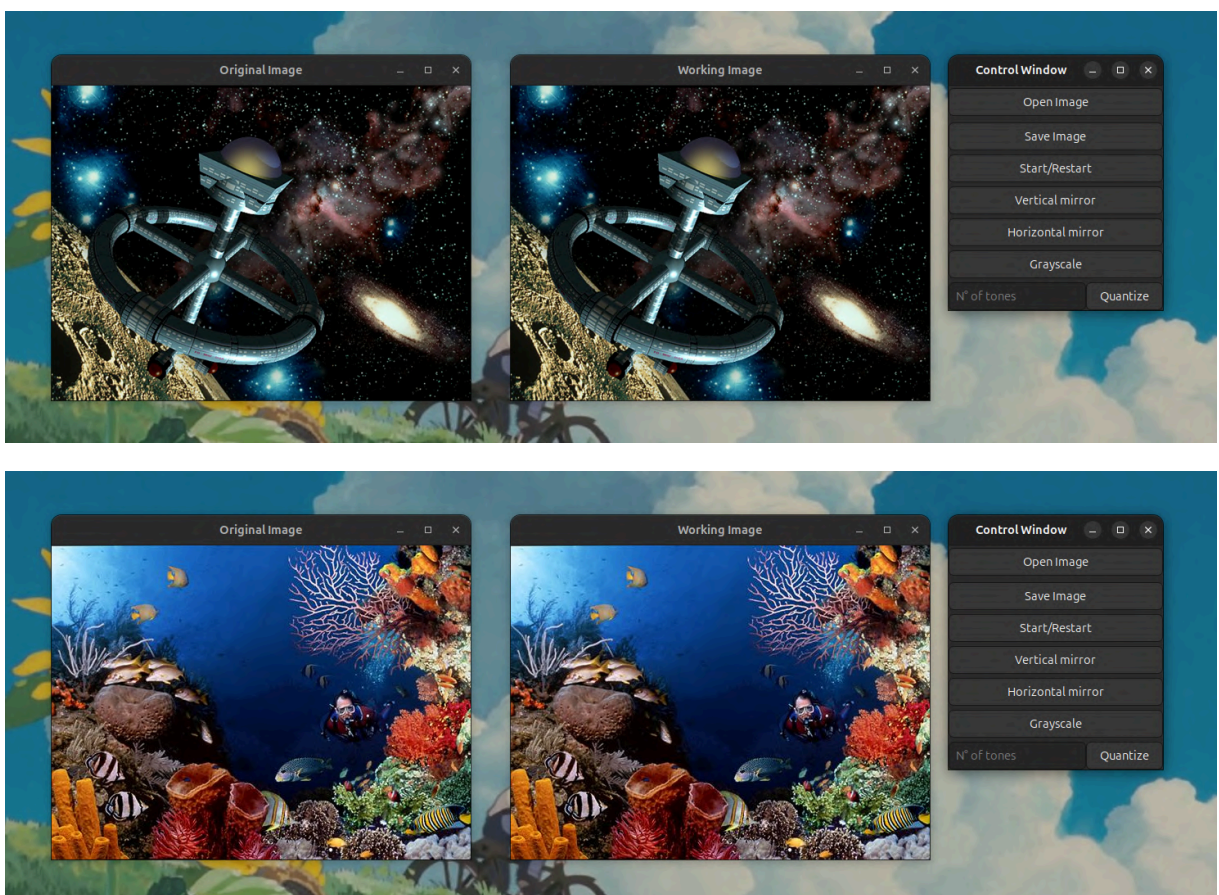
"Underwater_53k.jpg" original e quantizado com 20, 10 e 5 tons, respectivamente.

Salvamento da imagem em um arquivo JPEG

Por padrão, a implementação realizada realiza o salvamento de todas as imagens em formato JPEG, independente do formato de entrada. Poder-se-ia implementar a aplicação para salvar a imagem no mesmo formato de entrada, mas isso não ocorreu justamente para tornar a comparação entre tamanhos de saída possível para diferentes formatos de entrada.

CONCLUSÃO E INTERFACE

Todas as etapas do trabalho foram concluídas com êxito. Contudo, houve dificuldades no caminho, principalmente com a manipulação inicial dos píxeis e das funções disponibilizadas pela biblioteca escolhida. Isso se deve ao fato de esse ser o primeiro contato com a biblioteca escolhida e, portanto, a curva de aprendizado se tornou maior. Todavia, a manipulação das imagens, segundo os requisitos do trabalho, não se mostrou uma tarefa árdua.



Imagens da interface desenvolvida

A implementação mais divertida foi a função de quantização, e, ao mesmo tempo, foi a que gerou mais divagações e conclusões inesperadas, uma vez que é contraintuitivo imaginar que se possa representar imagens com uma quantidade tão pequena de tons sem perder a definição e qualidade.