

Nome: Eric Peracchi Pisoni

Matrícula: 318500

Introdução

Neste trabalho foi implementado um programa que faz a leitura de uma imagem JPEG e oferece ao usuário diferentes opções de operações que podem ser executadas nesta imagem. A imagem resultante é salva em um novo arquivo JPEG.

O código foi escrito na linguagem C++ usando Visual Studio 2022 no sistema operacional Windows 11.

Parte I: Leitura e gravação de imagens

Na primeira etapa, o programa apenas faz a leitura da imagem, exibe na tela e salva como uma nova imagem. Para isso, foram usadas as bibliotecas SDL para gerenciamento de janelas e SDL_image para leitura dos arquivos JPEG. O programa executou como esperando, criando duas janelas lado a lado, com os nomes "Original Image" e "Modified Image", onde foram exibidas a imagem original e uma cópia dela.

O desafio nessa etapa estava em descobrir como usar corretamente as bibliotecas escolhidas. Com algumas pesquisas e tutoriais, em uma hora foi encontrada uma solução adequada para realizar a atividade proposta.





Imagens exibidas lado a lado

Percebeu-se que, apesar de ser uma cópia da imagem original, os novos arquivos JPEG possuem um tamanho de arquivo maior. No melhor caso, houve um aumento de 34 KB; enquanto no pior caso, o aumento foi de 195 KB (quase cinco vezes o tamanho do arquivo original!). Visualmente não há diferenças entre as imagens.

Isso acontece por conta da maneira com que é feita a compressão nas imagens JPEG. No programa, foi definido que as imagens seriam salvas com qualidade 100 (a máxima possível), portanto o algoritmo de

compressão dará preferência à qualidade da imagem, em troca do aumento do tamanho do arquivo resultante.

Name	Type	Size
 Underwater_53k.jpg	JPG File	53 KB
 Underwater_copy.jpg	JPG File	248 KB

Comparação do tamanho dos arquivos

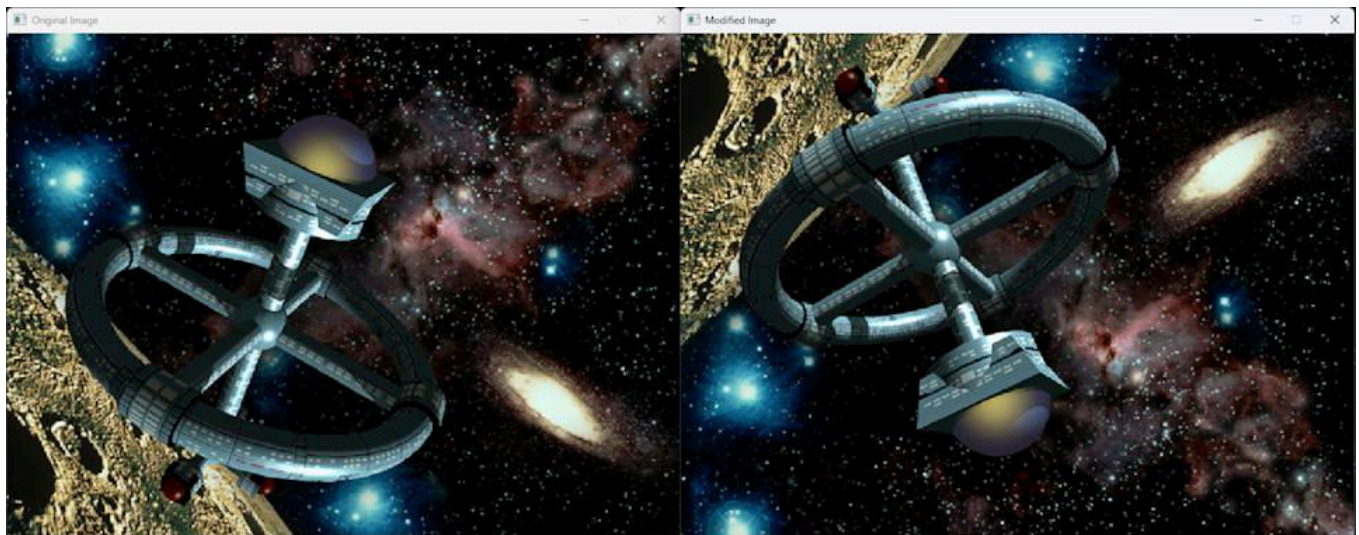
Parte II: Operações sobre imagens

a) Espelhamento horizontal e vertical

Como a linguagem escolhida foi C++, é altamente recomendável utilizar outras funções da biblioteca padrão do C++ para realizar a cópia de vetores, como `std::copy()`. Porém, para esta implementação, foi usada a função `memcpy()` para esse fim.

Na operação de espelhamento vertical, copia-se cada linha de pixels da imagem original, de cima para baixo, para as linhas da nova imagem, de baixo para cima. No espelhamento horizontal, não é possível copiar grandes trechos do vetor de pixels, já que não são feitos acessos contíguos à memória. Então, para cada linha, copia-se cada pixel da imagem original, da esquerda para a direita, para o pixel correspondente na nova imagem, da direita para a esquerda.

Nesta etapa, foi exercitada mais a aritmética de ponteiros, já que não foram realizadas operações nos valores individuais dos pixels, apenas foram movimentados blocos ou elementos da imagem.



Demonstração do espelhamento vertical

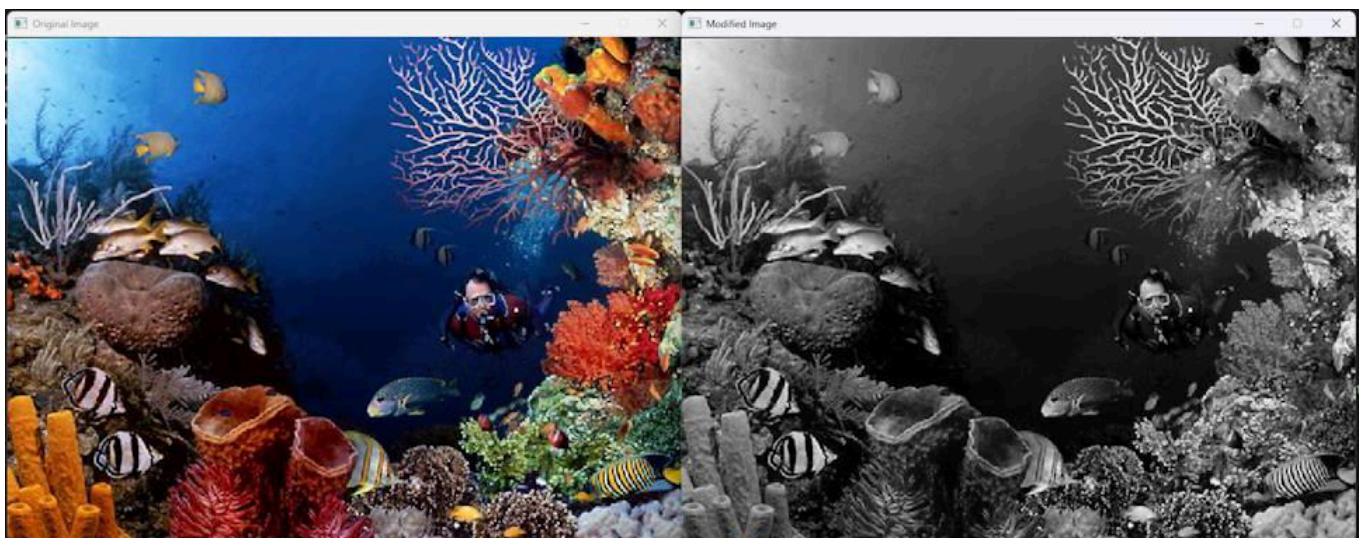


Demonstração do espelhamento horizontal

b) Conversão para tons de cinza

Para esta etapa, é necessário iterar sobre cada pixel da imagem, lendo o valor de cada um dos canais (R, G e B) e aplicando a fórmula $L = 0.229 * R + 0.587 * G + 0.114 * B$, que retorna um valor de luminância L , que então é copiado para os três canais. Desse modo, os três canais da imagem têm o mesmo valor, fazendo com que a imagem fique em tons de cinza.

A principal dificuldade nesta etapa estava em entender como manipular individualmente os canais de cada pixel, dado que um pixel era armazenado como um inteiro de 24 bits, sendo necessário aplicar uma máscara para decodificar cada valor de R, G e B. Após realizar as operações em cada valor, a máscara era revertida para obter o inteiro de 24 bits novamente.



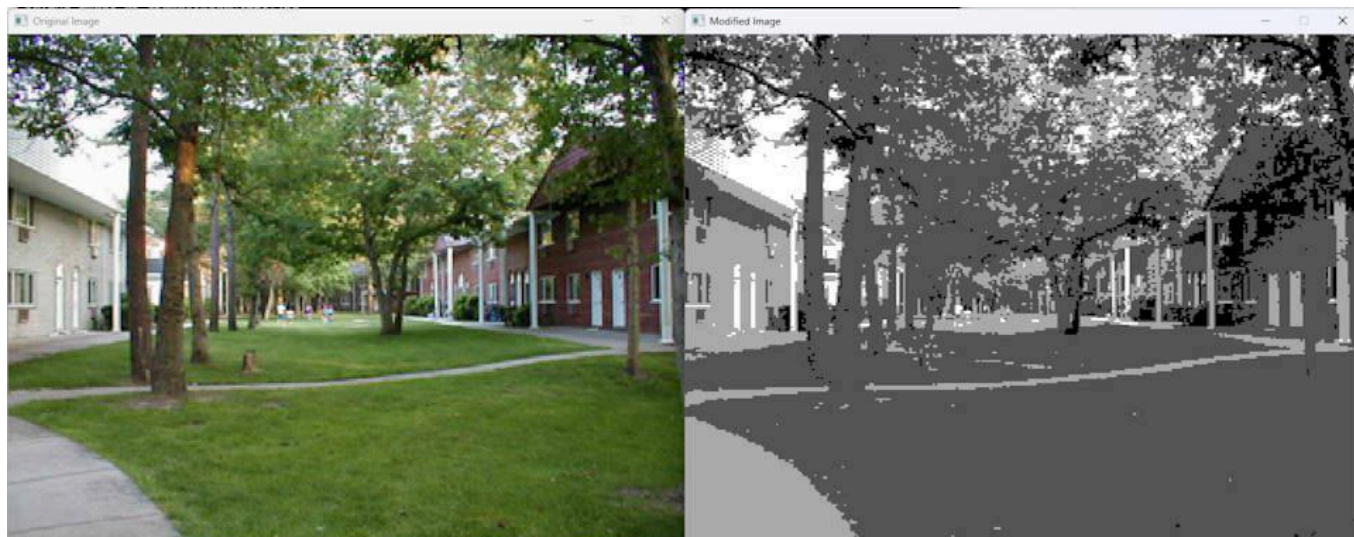
Demonstração da conversão em tons de cinza

Quando a operação é aplicada em uma imagem que já está em tons de cinza, a imagem resultante é idêntica à original.

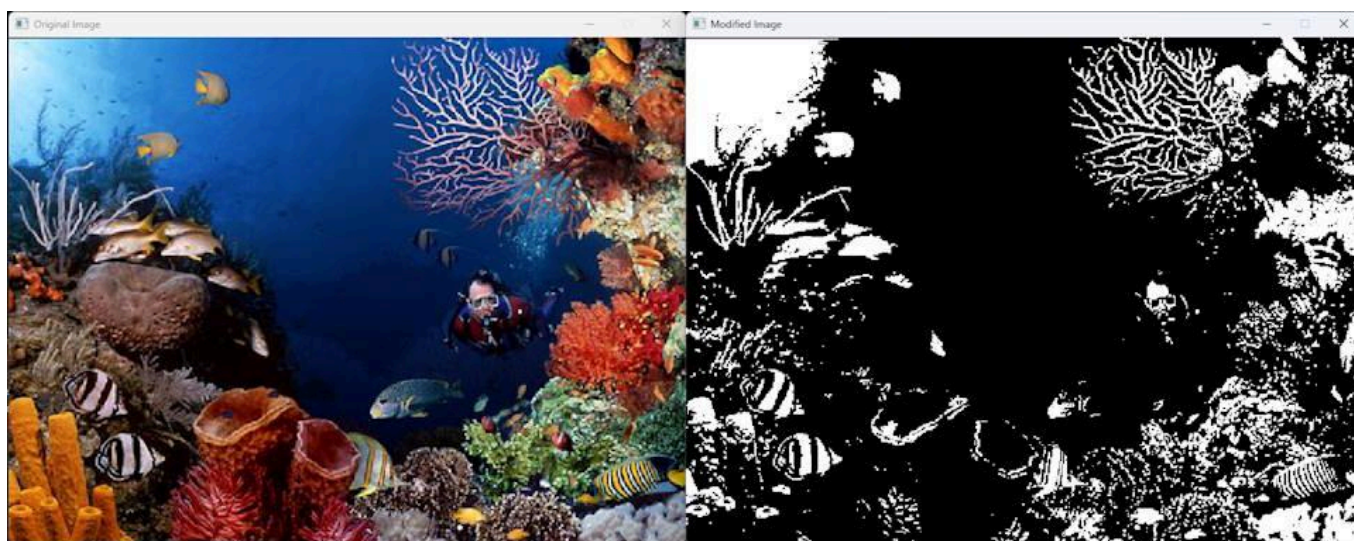
c) Quantização

Na quantização, o programa pede ao usuário para digitar a quantidade de tons de cinza que ele deseja usar para quantizar a imagem.

A imagem original é convertida para tons de cinza usando o mesmo algoritmo do item b), e então é aplicado o algoritmo de quantização, calculado com base no número de tons de cinza que o usuário digitou. Os valores de luminância são arredondados para o tom de cinza mais próximo disponível.



Demonstração da quantização com 4 tons



Demonstração da quantização com 2 tons

d) Salvar imagem

A operação de salvar uma imagem já tinha sido implementada na primeira etapa, portanto esta etapa já estava encaminhada e não houve muita dificuldade para executá-la.

```
Enter the name of the original JPEG (with extension): Space_187k.jpg
What operation do you want to perform?
(1) Mirror horizontally (2) Mirror vertically (3) Grayscale (4) Quantize
1
Enter a name for the new JPEG (type N if you don't want to save): Space_vert.jpg
Saving image as Space_vert.jpg
Program finished. Close this terminal to close the program.
|
```

Salvamento da imagem modificada

Hindsight

Várias melhorias podem ser feitas no código. A principal delas sendo a separação das operações de inicialização e renderização das janelas em funções externas.

Como mencionado antes, utilizar funções da *Standard Template Library* é fortemente recomendado.

Também pode-se aproveitar melhor a orientação a objetos proporcionada pelo C++.

Infelizmente, uma interface gráfica intuitiva não foi implementada. O programa roda numa janela do terminal. Nas próximas etapas, espero conseguir usar algum dos *toolkits* mencionados no enunciado para criar uma GUI e melhorar a apresentação do programa.