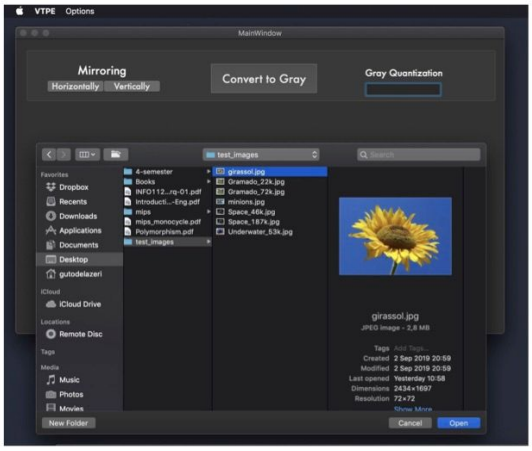
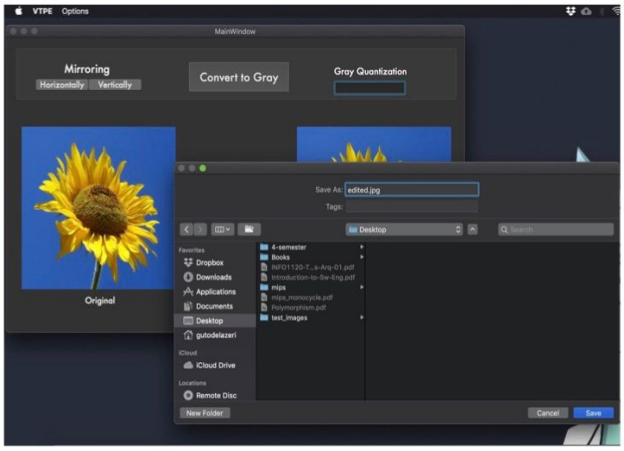
1. Leitura e Gravação de Arquivos de Imagens

A linguagem utilizada para implementar o programa foi C++ juntamente com Qt para manipular janelas e OpenCV para manipular as imagens a nível de pixel. Também foi utilizada a IDE Qt Creator visando facilitar o desenvolvimento da interface gráfica da aplicação. Podemos carregar uma nova imagem clicando no menu Options e em Open Image. Escolhemos então uma imagem e clicamos em Open.

Para salvarmos uma imagem precisamos primeiro colocá-la na área Edited (clicar na seta localizada no meio da janela) e depois clicar no menu Options e em Save Image. Na escolha do nome da imagem resultante é importante especificar a extensão (deve ser a mesma da imagem de entrada).



Carregando Imagem

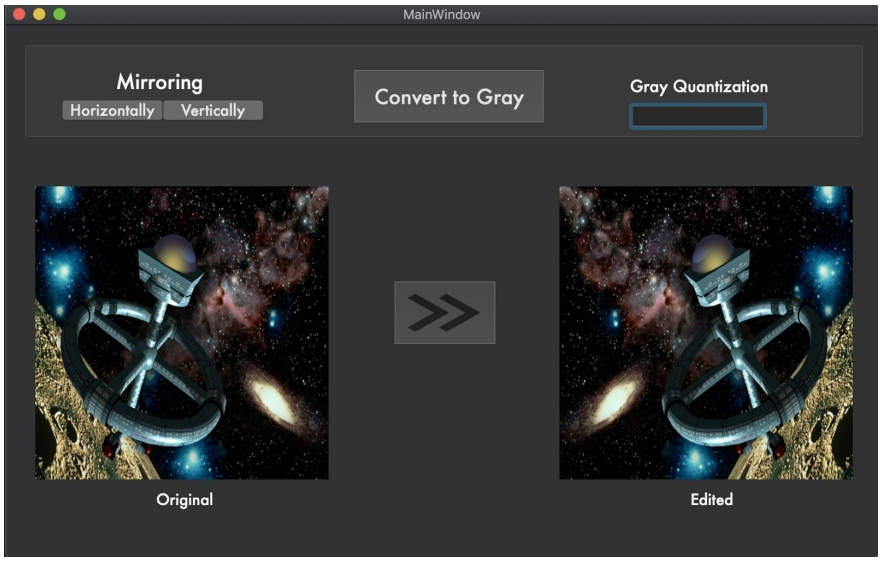


Salvando Imagem

É interessante notar que a imagem usada como exemplo, após ser salva, teve uma diminuição de cerca de 42% do seu tamanho original em disco. Uma das explicações que encontrei para tal fato diz que toda vez que salvamos uma imagem no formato JPEG os algoritmos de compressão operam sobre a imagem, mesmo se ela já está no formato JPEG. Inevitavelmente o tamanho da imagem em bytes diminui, mas ocorre perda de informação.

2. Leitura, Exibição e Operações

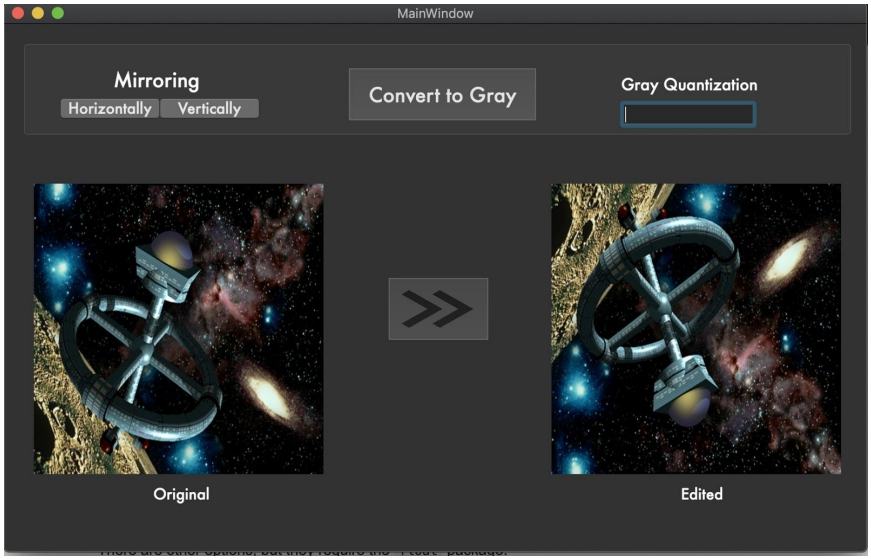
a. Espelhamento Horizontal



O algoritmo de espelhamento horizontal é relativamente simples e consiste em copiar e colar em ordem reversa as colunas da imagem original em uma outra imagem.

b. Espelhamento Vertical

O algoritmo de espelhamento vertical segue a mesma lógica do algoritmo de espelhamento horizontal. No entanto, o processo agora trabalha com as linhas da imagem.

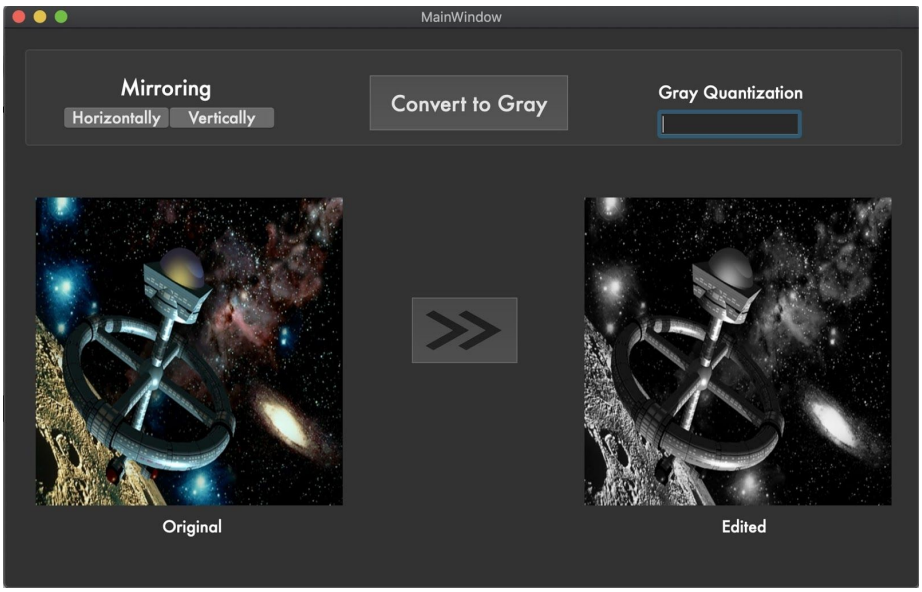


c. Conversão para tons de cinza

O algoritmo de conversão para tons de cinza consiste em criar uma nova imagem com as mesmas dimensões da imagem original, mas com apenas um canal de cor. A luminância de cada pixel da nova imagem é definida usando a fórmula vista em aula. Um detalhe importante que deve ser lembrado é a forma como o OpenCV codifica as imagens em memória, usando o padrão BGR. Dessa forma, devemos trocar a ordem dos coeficientes multiplicados a cada canal RGB para manter a coerência. No final do processo, convertemos a imagem para 3 canais novamente.

Um fato interessante é que aplicar o algoritmo de conversão em uma imagem que já está convertida não surte efeito algum, pois

$$L_{n+1} = L_n * 0.299 + L_n * 0.587 + L_n * 0.114 = L_n * (0.299 + 0.587 + 0.114) = L_n$$



d. Quantização

O algoritmo de quantização tenta quantizar a imagem da maneira mais uniforme possível. O algoritmo armazena em um vetor de 256 palavras o tom no qual o índice i é mapeado. Por exemplo, se vetor[50] = 30, então o tom número 50 foi mapeado para o tom de número 30. O cálculo do mapeamento é feito através de um laço for. Uma característica importante deste algoritmo está no fato de que quando o número de tons máximos utilizados é alcançado, o último tom usado é mapeado para os demais elementos do domínio, prejudicando a uniformidade do mapeamento.

200 tons

50 tons

3 tons

2 tons