Alquimia Digital

Nome:Leonardo T. Rubert

Matrícula: 19180465-7

Apresentação: https://youtu.be/vCQ_j8HCTkI

- **1. Introdução: apresentação do problema**

  A definição do problema pede com que eu faça uma análise de todos os pixels de uma imagem, e consiga montar outra imagem sem perder nenhum destes pixels que foram usados na primeira imagem, ou seja, alguma forma de aproximação

- **2. Desenvolvimento: conjunto de passos a serem realizados e validação de cada passo, descrição do(s) algoritmo(s)]**

  Para o desenvolvimento o primeiro passo foi descobrir como percorrer todos os pixels em todas as posições que eles ocupam dentro da própria imagem

  Depois de ter noção de como percorrer a imagem, eu parti para um método de definir qual o pixel mais próximo, eu testei alguns métodos conhecidos para distância entre valores, como Distância de Manhattan e Distância Euclidiana, no fim fiquei com a Euclidiana por ser mais precisa do que Manhattan.

Dentro desse método de comparação de distância, eu perguntava se o valor resultante do pixel analisado dentro da imagem desejada com a imagem origem tinha um valor menor ou igual do que a tolerância que eu defini como constante

Feito isso, parti para outro loop percorrendo a imagem original, dentro do loop que percorria a imagem desejada, para que eu pudesse comparar todos os pixels até o fim da imagem, dizendo se eram vantajosos ou não com o método da distância euclidiana

Encontrei um problema durante esse caminho, porque eu não tinha como saber se algum dos pixels que eu já tinha substituído, já tinha sido utilizado, conforme a definição do enunciado, todos os pixels dentro da imagem origem precisam ser utilizados uma vez e só uma vez, então minha imagem saida com input de Monalisa e Persistence of memories me retornava uma versão bem esverdeada da Persistence of memories.

O próximo passo seria encontrar uma forma de validar se cada pixel em cada posição já tinha sido usado ou não, minha abordagem foi de fazer um vetor de booleanos cujo tamanho é o mesmo tamanho da imagem original, o que isso me garante? em primeira vista nada, mas durante todas as trocas positivas durante o loop onde chamamos o método de distância euclidiana, setamos a posição que existe no vetor de imagem original como verdadeira dentro do vetor que contem todas as posições com valor booleano, assim podemos se podemos usar o pixel ou não.

Mas como nada na vida é fácil, após realizar esse último passo, eu imaginava que o trabalho estaria pronto, porém, eu adicionei o método validar() no fim de cada execução, para facilitar meu

processo, e descobri que meus pixels da imagem origem não estavam todos presentes na imagem final, apesar de ela ser composta unicamente de todos os pixels da imagem original, então bati cabeça por mais uns 2 dias, tentando encontrar formas de garantir que todos os pixels da imagem original estariam na imagem saida, até que re-lendo o enunciado pela quinquagésima vez, eu percebi que estava escrito “copia-se a imagem de origem para a imagem resultado”, então eu tive a ideia de copiar tudo em pic[origem].pixels para dentro de pic[saida].pixels, e logo após fazer isso eu rodei, e de fato o algoritmo já estava funcionando, mas não tinha como garantir que todos os pixels iriam para a imagem final, pois estava sempre filtrando alguns para fora

no fim foi apenas questão de acertar um valor de tolerância para a distância euclidiana manter sentido da imagem e também que os pixels sejam certos

- **3. Conclusão: principais dificuldades encontradas e possíveis melhorias**

A minha maior dificuldade não foi exatamente em montar o método para encontrar as cores de aproximação, ou percorrer as listas, ou até mesmo percorrer as listas enquanto percorria as outras listas, a minha maior dificuldade foi entender que eu podia simplesmente copiar os pixels de uma para outra, e também analisar o código proposto, não porque estava ilegível ou coisas do gênero, mais porque era muito grande, e a maioria das coisas era setup para fazer a visualização e leitura de imagens funcionar, eu gostaria de sugerir que fossem feitos métodos em arquivos separados, para deixar essa questão de load e visualização bem alto nivel para as pessoas ficarem menos confusas.

Também outra dificuldade encontrada se deu com rodar o próprio código, tive menos tempo para desenvolver o trabalho porque não estava conseguindo rodar, talvez seja legal sinalizar que precisa ser seguido o tutorial de instalação do MinGW sugerido no moodle, mas como eu fui um caso a parte talvez não seja algo tão comum.