Trabalho 1
Programação de Baixo Nível

Samuel Fries, Vitor Vettoretti
Ciência da Computação - PUCRS
Professor: Edson Moreno

28 de Abril de 2024

Resumo

O relatório a seguir apresenta uma solução para o problema proposto na disciplina de Programação de Baixo Nível no semestre 2024/1, que simula uma espécie de alquimia digital, onde temos uma imagem de origem e uma imagem desejada. O intuito do algoritmo é reordenar todos os pixels da primeira imagem (origem), para produzir a imagem desejada, ou seja, produzir uma nova imagem que se pareça o máximo possível com a segunda. Ao final são mostrados os resultados para diferentes conjuntos de imagens testadas.

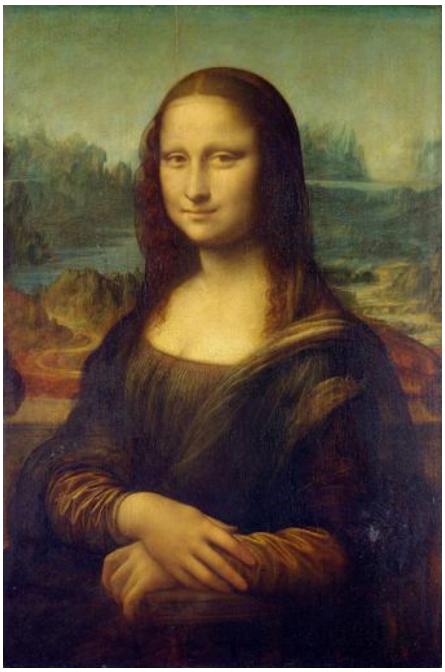
introdução

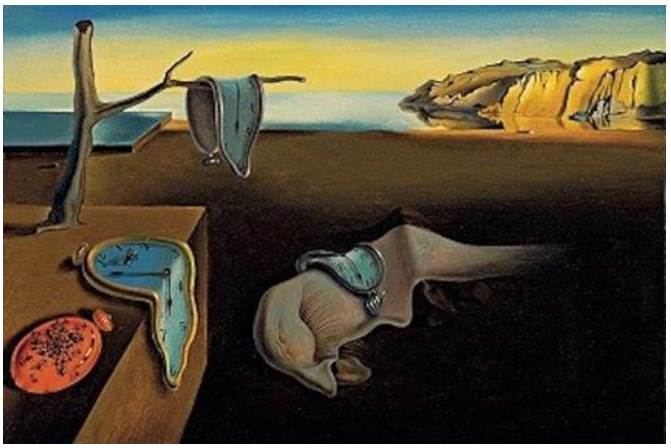
O problema proposto na para o primeiro trabalho da cadeira de Programação de Baixo Nível foi uma simulação . Os alquimistas tinham como metas principais transformar metais básicos em ouro e criar um elixir da vida que conferisse imortalidade. Embora muitos dos seus métodos e conceitos tenham sido refutados, a alquimia teve um papel fundamental na fundação da ciência contemporânea, particularmente na química e na medicina. Atualmente, é considerada mais como um campo de estudo filosófico e espiritual do que científico.

Neste trabalho, inspirado pelos alquimistas de outrora e no antigo anseio pela transmutação de metais, abordaremos uma tarefa bem mais simples: a transmutação de imagens, ou seja, transformar uma imagem em outra. O algoritmo proposto deve abordar uma solução funcional para criar uma imagem desejada utilizando a imagem de origem. Esse processo ocorre reorganizando os pixels da imagem de origem a fim de se aproximar o máximo possível da imagem de saída. Contudo, as seguintes regras têm de serem seguidas:

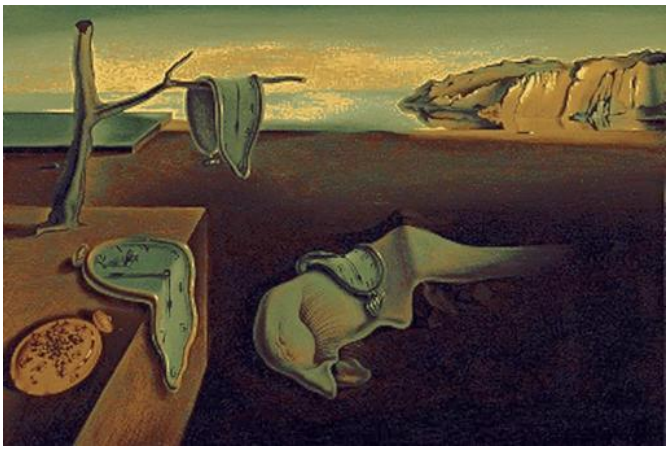
1. Não é necessário que as imagens inseridas tenham o mesmo tamanho, mas devem possuir a mesma área total, isto é, o mesmo número de pixels.
2. A transmutação de imagens deve ocorrer reorganizando os pixels já existentes na imagem, nenhum pixel deve ser alterado.
3. Um pixel deve ser representado por 3 componentes: vermelho (R), verde (G) e azul (B).
4. Uma imagem é representada por uma matriz de pontos (pixels).

Para parâmetros de comparação serão apresentadas duas imagens de exemplo. A primeira imagem representa a imagem de origem, e a segunda representa a imagem desejada:





A imagem de saída gerada foi a seguinte:



Solução

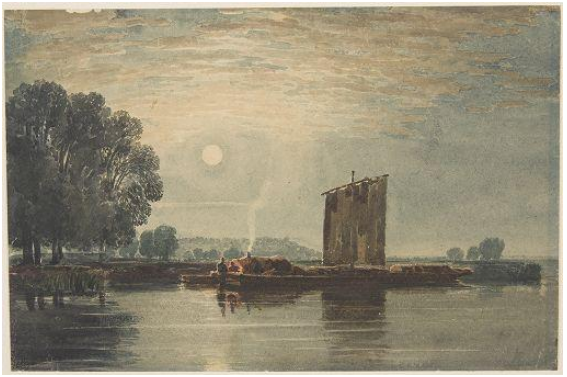
Após analisar o problema desenvolvemos um algoritmo capaz de gerar uma imagem de saída de acordo com os pixels da imagem de origem. Para isso, o algoritmo realiza comparações entre os valores definidos como vermelho (R), verde (G) e azul (B), os quais integram o valor total adotado por um pixel. A comparação ocorre entre a imagem de origem e a imagem desejada, onde para cada pixel da imagem de origem é realizada uma comparação para cada pixel da imagem desejada, assim podendo chegar ao pixel que mais se assemelha entre as duas imagens. O seguinte código mostra a implementação adotada para melhor entendimento:

```
for (int i = 0; i < tam; i++) {
    int min_dist = 10000;
    RGBpixel closest_pixel;

    for (int j = 0; j < tam; j++) {
      double dist = sqrt(pow(pic[ORIGEM].pixels[j].r - pic[DESEJ].pixels[i].r, 2) +
                         pow(pic[ORIGEM].pixels[j].g - pic[DESEJ].pixels[i].g, 2) +
                         pow(pic[ORIGEM].pixels[j].b - pic[DESEJ].pixels[i].b, 2));

      if (dist < min_dist) {
          min_dist = dist;
          closest_pixel = pic[ORIGEM].pixels[j];
      }
    }

    pic[SAIDA].pixels[i] = closest_pixel;
    }
```

É possível observar dois laços no código, o primeiro para percorrer a imagem desejada e o segundo para percorrer a imagem de origem. Isso ocorre pois precisamos saber qual é o pixel da imagem de origem de valor mais próximo ao respectivo pixel da imagem desejada. Para descobrir qual pixel tem o valor mais aproximado, é implementado uma função matemática que calcula a distância relativa entre os pixels denominada *dist*. Depois de tirar a distância de todos os pixels da imagem de origem, o pixel mais próximo é atribuído ao pixel da imagem de saída. O programa faz essa comparação para todos pixel da imagem desejada e ao final obtemos uma imagem de saída que corresponde ao esperado.

Resultados

Para os testes foram utilizadas duas imagens de entrada mostradas a seguir que correspondem à imagem original e imagem desejada respectivamente.



Ao final da execução do programa obteve-se a seguinte imagem de resultado:

Conclusão

Para implementar o algoritmo utilizamos uma abordagem que acreditamos não ser muito desejada, pois é computacionalmente intensiva pois realiza diversas comparações. Essas comparações ocorrem uma quantidade de vezes relativa ao número de pixels da imagem original para cada pixel da imagem desejada, ou seja, para cada pixel contido no vetor da imagem desejada, o vetor de pixels da imagem original é percorrido por inteiro. Devido a essa abordagem do algoritmo, acaba sendo uma implementação muito pouco eficiente, com termos mais técnicos, o algoritmo tem uma complexidade de $O(n^2)$.

Acreditamos ser possível abordar esse problema de uma forma um pouco diferente para obter resultados mais eficientes. Uma possível implementação que chegamos a considerar foi a duração de uma imagem completamente embaralhada, e na sequência, a realização de diversas comparações para melhorar a imagem aos poucos. Para cada iteração são feitas algumas alterações e avaliadas se as alterações foram boas, se forem boas, são mantidas, caso contrário são descartadas.