Trabalho 2 de Introdução ao Processamento de Imagem Digital

Gabriel de Alcantara Bomfim Silveira - RA197244

Introdução

O objetivo deste trabalho é de reduzir a quantidade de cores em uma imagem utilizando técnicas de pontilhado com difusão de erro.

O arquivo **GABRIEL_SILVEIRA_197244.zip**, entregue por meio do Google Classroom, possui todos os arquivos citados no relatório.

A solução é implementada em *Python3* por meio do *script* **projeto2.py**. Para executar o *script* é necessário utilizar a linha de comando passando a imagem que se deseja reduzir a quantidade de cores como parâmetro. Por exemplo:

```
$python3 projeto2.py ./images/baboon.png
```

O script vai aplicar a técnica de pontilhado com todas as distribuições de erro e salvar os resultados na pasta **outputs** de acordo com os nomes definidos no enunciado. As distribuições de erro aplicadas são: Floyd-Steinberg, Stevenson-Arce, Burkes, Sierra, Stucki e Jarvis-Judice-Ninke. Ao fim deste relatório, apresentarei os resultados obtidos.

Observação 1: é importante que exista uma pasta ***outputs*** no mesmo local do *script,* para que as saídas sejam geradas automaticamente sem problemas.

Observação 2: para que a execução do *script* ocorra corretamente é preciso instalar os pacotes *numpy* e *cv2* (*OpenCV).*

Execução

Para gerar imagens coloridas a partir da técnica de pontilhado, o primeiro passo foi, para cada *pixel* na imagem, encontrar a cor bruta que mais se aproxima de sua cor original. Desta forma, para cada canal de cor de cada *pixel*, foi executada a seguinte função

```
def find_closest_color(color):
    return round(color/255) * 255
```

Os valores de cada canal retornados pela função serão agora os novos valores para os canais do *pixel*, gerando uma nova cor. Desta forma, um pixel composto pelos canais vermelho, verde e azul, ao final da aplicação da técnica, só poderão ter apenas as cores vermelho, verde, azul, magenta, amarelo, ciano, branco e preto.

Para aplicar a técnica de pontilhado corretamente, também é preciso calcular e distribuir o erro. Neste caso, o erro é a diferença entre a cor original de um *pixel* e a nova cor obtida para este *pixel*.

```
error = output[k][line][column] - newColor
```

No código, a variável "*error"* armazena uma lista de três posições. Estas três posições têm valores que representam os erros de cada canal.

Para cada distribuição de erro definida na proposta do projeto, foi gerado um *numpy array*. A célula destacada em amarelo de cada tabela é o *pivot*, ou seja, a posição que corresponde à posição do *pixel* que está sendo analisado no momento.

Floyd-Steinberg

0	0	7/16
3/16	5/16	1/16

Stevenson-Arce

0	0	0	0	0	32/200	0
12/200	0	26/200	0	30/200	0	16/200
0	12/200	0	26/200	0	12/200	0
5/200	0	12/200	0	12/200	0	5/200

Burkes

0	0	0	8/32	4/32
2/32	4/32	8/32	4/32	2/32

Sierra

0	0	0	5/32	3/32
2/32	4/32	5/32	4/32	2/32
0	2/32	3/32	2/32	0

Stucki

0	0	0	8/42	4/42
2/42	4/42	8/42	4/42	2/42
1/42	2/42	4/42	2/42	1/42

Jarvis-Judice-Ninke

0	0	0	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

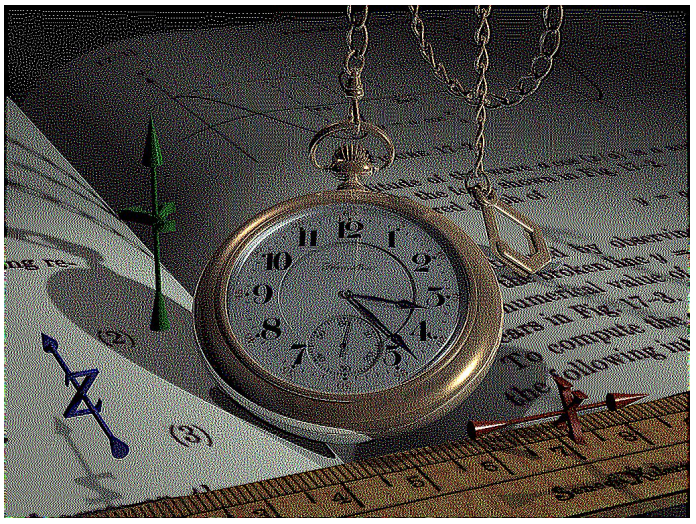
Para finalizar, o erro de cada canal é multiplicado pela matriz que representa a distribuição escolhida, gerando outras três matrizes, uma para cada canal. Cada canal de cor dos *pixels* vizinhos ao *pixel* analisado no momento, tem seu valor somado com um dado erro distribuído. Estes erros distribuídos são escolhidos comparando a posição do *pixel* vizinho com relação ao *pixel* analisado atualmente, com a posição do *pivot* em relação à uma posição na matriz de distribuição.

Resultados e discussão

Nesta seção apresento os resultados gerados após a aplicação da técnica de pontilhado com diferentes distribuições de erro. Para este relatório, escolhi mostrar apenas a imagem do relógio **(watch.png)**, que se encontra na pasta **images**. Os demais resultados podem ser encontrados na pasta **resultados salvos**.

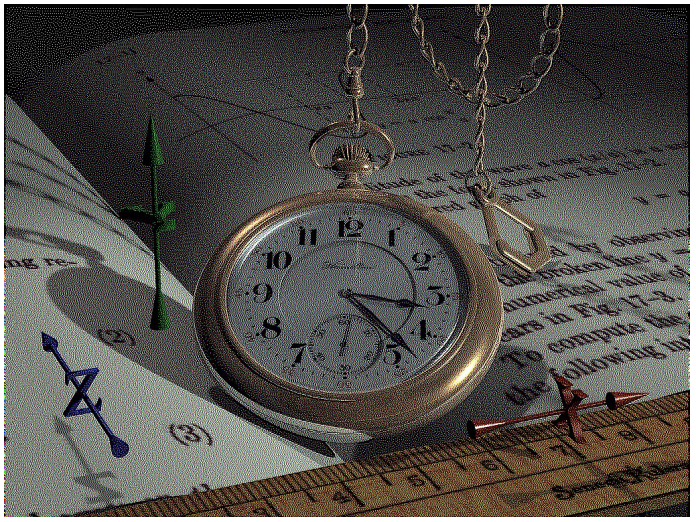
watch.png - imagem original

watch.png após aplicação de Floyd-Steinberg



watch.png após aplicação de Stevenson-Arce

watch.png após aplicação de Burkes



watch.png após aplicação de Sierra

watch.png após aplicação de Stucki

watch.png após aplicação de Jarvis-Judice-Ninke

Conclusão

A técnica de pontilhado é útil para realizar a compressão de uma imagem, mantendo seu conteúdo. As diferentes distribuições de erro geraram resultados parecidos, tendo diferenças que podem ser enxergadas ao se aproximar as imagens. Dependendo do tamanho da matriz de distribuição de erro, a qualidade do resultado nas bordas da imagem também é afetada. Em geral, a técnica nos permite obter uma imagem que tem um conteúdo discernível, porém que ocupa menos espaço no disco. No caso da distribuição *Floyd-Steinberg,* por exemplo, saímos de uma imagem de 700 KB para uma imagem de 492 KB, uma taxa de compressão de 30%.