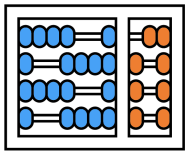

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

MARCOS DA MATA SOUSA (221519)

Relatório - Trabalho 1

CAMPINAS
2022

1. Introdução

O presente relatório apresenta os métodos utilizados para a realização de processamentos básicos em imagens digitais, tais como transformações de intensidade, ajustes de brilho, combinação de imagens e outros, bem como seus respectivos resultados. Todos os processamentos foram realizados através da linguagem Python (3.8). O carregamento e salvamento de todas as imagens foram realizados com auxílio da biblioteca *Imageio* e a biblioteca *Matplotlib* serviu de auxílio para a visualização dos resultados dos processamentos. Além disso, a biblioteca *NumPy* também foi utilizada para fins de transformação das imagens carregadas.

1. Transformação de Intensidade

Dada uma imagem monocromática *city.png*, transformou-se seu espaço de intensidades a fim de obter os resultados especificados nos tópicos a seguir.

- **Negativo de imagem:** Um procedimento simples é realizado com objetivo de obter o negativo de uma imagem. Tendo carregada a imagem numa variável, basta subtrair todos os seus pixels de 255, o nível máximo de intensidade dos pixels de uma imagem monocromática. Após a execução do algoritmo, a imagem se encontra no diretório do código, nomeada por *1.1negativo.png*.

- **Conversão de intervalo de intensidades:** O intervalo de intensidades da imagem foi convertido de [0, 255] para [100, 200] utilizando-se do método *where*, da biblioteca *NumPy*, método que faz um teste condicional e, caso obtenha resultado verdadeiro, atribui um valor *x* a um determinado pixel e um valor *y*, caso contrário. Intensidades menores que 100 foram alteradas para 100 e intensidades maiores que 200 foram convertidas para 200. O resultado pode ser encontrado no diretório após a execução do algoritmo pelo título de *1.1intervalo_100_200.png*

- **Inversão dos valores das linhas pares da imagem:** Este tópico foi resolvido como uma simples operação de fatiamento de matrizes na linguagem Python. Dada uma matriz *d* que representa os níveis de intensidade de cinza de cada pixel, invertemos as linhas pares como se segue:

  ```
  d[0::2, :] = d[0::2, ::-1]
  ```

  Em suma, começando linha 0, a passos de 2 linhas, invertemos (::-1) as linhas pares. A imagem resultante pode ser encontrada pelo título de *1.1inversão_linhas_pares.png* após a execução do algoritmo.

- **Espelhamento da parte superior na parte inferior da imagem:** Esta questão também foi resolvida com a utilização de fatiamento, mas utilizando também uma função de *flip* da biblioteca do *NumPy*, o *flipud*, que gira uma imagem (ou uma porção dela, como neste caso) de cabeça para baixo. O espelhamento da parte superior nada mais é do que a parte de cima refletida de cabeça para baixo na metade inferior da imagem. Dessa forma, o problema foi resolvido facilmente copiando-se a parte superior da imagem na parte inferior (através do fatiamento) e girando metade inferior de cabeça para baixo. Após a execução do algoritmo, é possível encontrar a imagem resultante no diretório, nomeada por *1.1espelhamento_superior.png*.

- **Espelhamento vertical:** Ainda mais simples do que o tópico anterior, o espelhamento vertical pode ser interpretado como uma imagem virada de cabeça para baixo. Assim, foi necessário, novamente, apenas a função *flipud*, que é responsável por realizar esta tarefa. A imagem resultante é intitulada *1.1espelhamento_vertical.png*.

2. Ajuste de Brilho

O primeiro passo para o ajuste de brilho foi a conversão do intervalo de intensidades para [0, 1], tal tarefa pôde ser realizada simplesmente dividindo a matriz pelo escalar 255.

Tendo sido convertido o intervalo, todos os pixels da matriz-imagem são elevados por um escalar $1/\gamma$, com γ arbitrário por meio do método *power*, também da biblioteca *NumPy*.

Finalmente, converte-se a imagem para o intervalo de intensidade original [0, 255] e pode-se observar a imagem com um brilho maior. É possível encontrar as imagens com diferentes γ's (1.5, 2.5 e 3.5) no diretório, nomeadas por *1.2gama_γ.png*, seguidos do valor de gama correspondente. Foi possível observar que as configurações de brilho apresentadas nos resultados possuem um brilho que cresce de maneira diretamente proporcional ao crescimento de γ, conforme o esperado.

3. Planos de Bits

Foram utilizados o métodos *where* e *power* da biblioteca *NumPy*, além do operador *&* para a resolução deste tópico. Para a extração do enésimo plano de bit fez-se necessário o cálculo da base 2 elevada a enésima potência, tarefa que foi cumprida como a utilização do método *power*. Tendo o resultado calculado, utilizou-se este resultado como parâmetro na execução do método where (`np.where(matriz & resultado, 1, 0)`).

O resultado da extração mostrou-se bem sucedido, uma vez que utilizado o operador *&* entre cada valor de intensidade da matriz-imagem e um número 2^n (que em sua representação binária possuirá apenas um dígito igual a 1), apenas os dígitos da enésima posição de cada número em representação binária foi mostrada na matriz resultante. Os

resultados podem ser encontrados nas imagens *1.3plano0.png*, *1.3plano4.png* e *1.3plano7.png*.

4. Mosaico

Considerando que as imagens de entrada possuem dimensões de 512x512, a fim de montar um mosaico de 16 partições, dividiu-se tanto as linhas quanto as colunas por 4. Ou seja, cada partição possui a dimensão de 128x128 e, a partir da utilização de fatiamentos, foi possível embaralhar cada uma das 16 partições como mostrado no exemplo abaixo.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

6	11	13	3
8	16	1	9
12	14	2	7
4	15	10	5

O mosaico resultante pode ser encontrado no diretório, nomeado por *1.4mosaico.png*, e o resultado obtido foi bem sucedido.

5. Combinação de Imagens

A combinação de duas matrizes-imagem A e B pôde ser realizada através da soma das duas matrizes multiplicadas por escalares p_1 e p_2, que representam, respectivamente a porcentagem do total de cada imagem. A soma de $(0.5 * A) + (0.5 * B)$, por exemplo, resultará numa matriz-imagem que apresentará as duas imagens combinadas com a mesma intensidade, 50% cada.

Três configurações de combinação podem ser encontradas no diretório após a execução do algoritmo, com as configurações de p_1 e p_2 iguais a (0.2 e 0.8), (0.5 e 0.5) e (0.8 e 0.2), respectivamente. Os resultados obtidos saíram de acordo com as previsões.

6. Filtragem de Imagens

As convoluções necessárias para a aplicação dos efeitos foram realizadas através do método *convolve2d*, da biblioteca *SciPy*.

Ao aplicar os 11 efeitos na imagem *butterfly.png*, a maioria mostrou resultados semelhantes. Com exceção dos efeitos h_2, h_6 e h_9, que deixaram a imagem com tom

esverdeado e desfocado em diferentes níveis, as imagens resultantes mostraram uma cor azul-esverdeada de mesmo tom em todas suas superfícies. No entanto, as formas dos objetos presentes na imagem original é destacado nas imagens nas quais são aplicados os efeitos, diferindo-se apenas na profundidade com que se destacam as formas das imagens resultantes.

Por fim, somou-se as imagens resultantes dos filtros h_3 e h_4 através da expressão (1) e foi possível observar que o resultado foi idêntico à aplicação do filtro h_{11}.

$$\sqrt{(h_3)^2 + (h_4)^2} \quad (1)$$