UNIVERSIDADE FEDERAL DE SÃO CARLOS

DEPARTAMENTO DE COMPUTAÇÃO

EQUALIZAÇÃO LOCAL DE HISTOGRAMA

Discentes:

Caio L. R. S. Ueno 743516,

Gabriel C. P. Mendes 743535,

João A. Leite 743551 e
Vinicius H. S. Carvalho 743602

Docente:

Dr. Cesar Henrique Comin

São Carlos / SP

22 de setembro de 2020

1 Motivação do Uso do Método

Histogramas são representações gráficas de dados divididos em classes, esse é feito com um gráfico de barras demonstrando a distribuição de frequências. No contexto de Processamento Digital de Imagens, histogramas contêm informações sobre os níveis de intensidade presentes na imagem. Assume-se que as intensidades estão contidas no intervalo [0, L-1], sendo L o número de níveis de intensidade que podem ser representados com a precisão escolhida (profundidade de bits da imagem). Dessa forma, histograma é a informação sobre quantos pixels na imagem possuem o valor de intensidade $k, \forall k \in [0, L-1]$.

Embora haja perda informações como a posição de cada pixel, é possível definir diversas técnicas de processamento de imagem de forma intuitiva; dentre elas, a Equalização de Histograma, essa trata-se de uma metodologia para tornar um histograma mais "uniforme", ou seja, os níveis de intensidade tornam-se presentes em quantidade próxima. O método aumenta o contraste e, em alguns casos, melhora a aparência da imagem; além disso, tem um ótimo funcionamento para visualizar regiões de imagem com baixo contraste. Como vantagem o método não possui parâmetros para ajustar, porém, como desvantagem, pode não ser satisfatório para melhorar a aparência da imagem.

2 Explicação do Método Implementado

O método consiste em criar uma tabela em que para cada nível de intensidade de 0 a L-1, chamado de n_k, contabilizar a quantidade de pixels dessa intensidade, para o caso de implementação, é usado um vetor, uma vez que os níveis de intensidades coincidem com os índices do vetor. Após, para cada valor de intensidade k no intervalo [0, L-1], é aplicado:

$$S_k = T(k) = \frac{(L-1)}{M * N} \sum_{j=0}^{k} n_j \tag{2.1}$$

Com M o número de linhas da imagem, N o número de colunas da imagem, ou seja, $M * N$ é o número de pixels da imagem. O valor s_k deve ser arredondado e adicionado a uma nova tabela, chamada de *Lookup Table*, em que os valores de s_k são relacionados aos valores de intensidade k, $\forall k \in [0, L-1]$. Essa tabela é utilizada para a substituição

do valor da intensidade de cada pixel da imagem original, gerando uma imagem cujo histograma é mais uniforme comparado ao anterior.

3 Explicação do Código

O código foi separado nos seguintes módulos:

3.1 Função: main

É definido pelo usuário a imagem, o número de linhas e colunas em que essa será dividida. Caso a imagem seja colorida, antes, essa é convertida para escalas de cinza para, posteriormente, chamar os outros módulos e o resultado ser exibido.

3.2 Função: image_split

É responsável por dividr a imagem em outras pequenas, de acordo com o número de linhas e colunas definido pelo usuário e retorna todas em uma lista.

3.3 Função: equalization

```
def equalize(m: np.ndarray, size=256) -> np.ndarray:
    caixas = range(0, size+1)
    hist, _ = np.histogram(m, caixas)

    sum_hist = sum(hist)
    const = (size-1)/sum_hist
    saida = np.zeros(size+1)
    for i in range(size+1):
        soma = sum(hist[:i+1])
        saida[i] = soma*const

    m_eq = np.zeros(m.shape)
    n_lin, n\_col = m.shape
```

```
    for i in range(n_lin):
        for j in range(n\_col):
            m_eq[i, j] = saida[int(m[i, j])]

    return m_eq
```

É a função que faz toda a equalização de cada fragmento da imagem. Inicialmente, recebe o fragmento da imagem e o tamanho máximo de intensidade, caso não especificado, é definido como 256. Com o histograma gerado, a constante $\frac{(L-1)}{MN}$ é definida pelo número de niveis de intensidade dividido pela quantidade de pixels e a *Lookup Table* é criada como saída. Dentro do *loop*, é feito o somatório da quantidade de pixels até a intensidade i e esse valor é multiplicado pela constante, e inserido na posição i da *Lookup Table*. Por último, em um *loop* duplo é feita a conversão da imagem para a equalizada passando por cada pixel e definindo seu novo valor de acordo com a *Lookup Table*. O retorno é o fragmento equalizado.

3.4 Função: image_merge

Módulo que receberá todos os fragmentos, juntá-los em uma única imagem e retorná-la.

4 Resultados

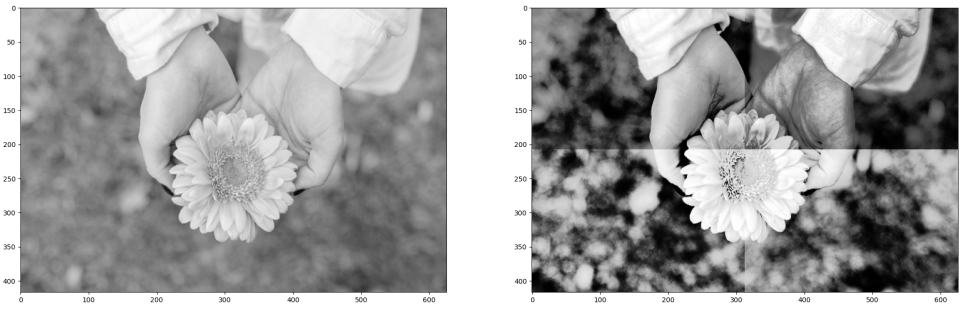
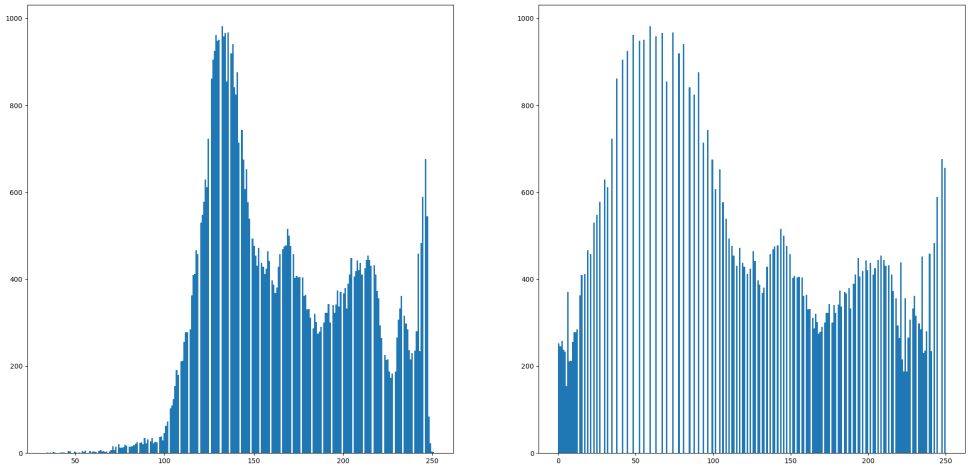
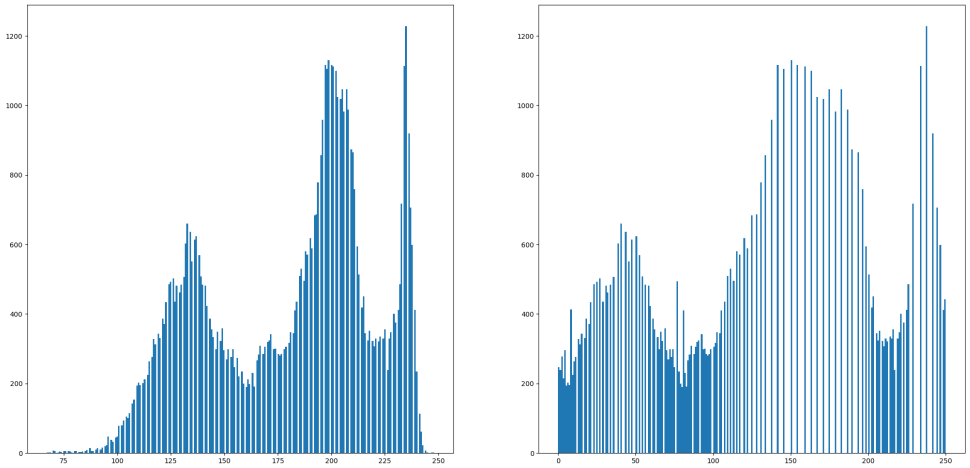


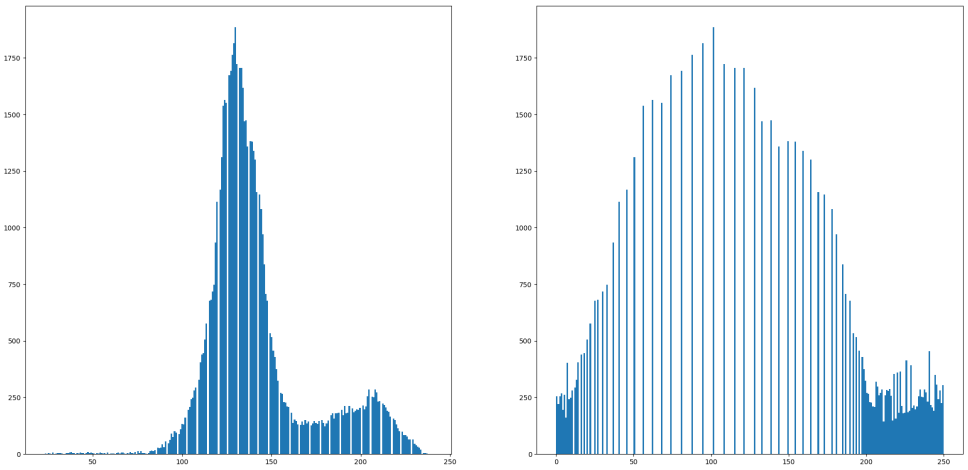
Foto antes/depois da equalização



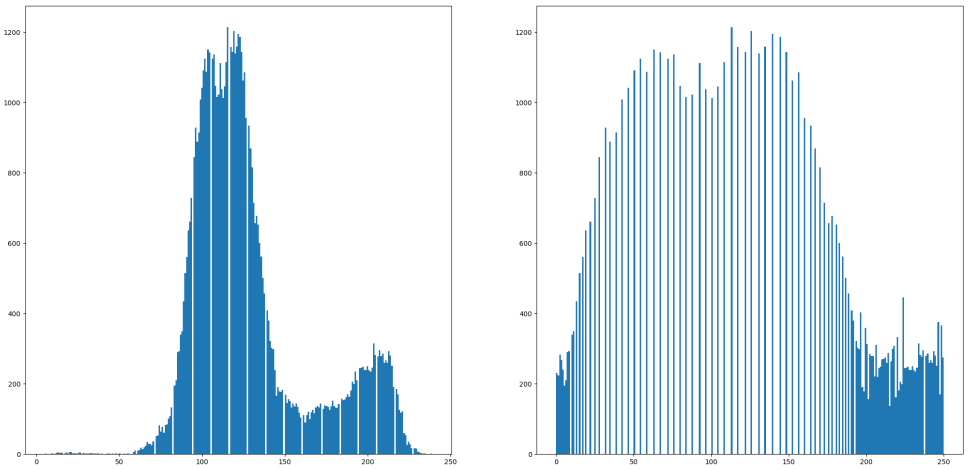
Histograma antes/depois da equalização na região 1



Histograma antes/depois da equalização na região 2



Histograma antes/depois da equalização na região 3



Histograma antes/depois da equalização na região 4