Universidade de São Paulo

Trabalho 3

Convolução, Bordas e Aplicação de Filtros

Processamento e Análise de Imagens

Aluno

Thiago Oliveira dos Santos n° 13696220

São Carlos

2024

1 Objetivos

Este trabalho tem como objetivo aplicar os conceitos relacionados a convolução em imagens digitais. Serão exploradas técnicas como aplicação de filtros passa-baixa para aplicação de médias, e aplicação de filtros passa-alta para detecção de bordas.

2 Introdução Teórica

2.1 Convolução

2.1.1 Explicação Matemática

A convolução é uma operação matemática que combina duas funções para produzir uma terceira função. Matematicamente, a convolução de duas funções f e g, geralmente denotadas como $f(x)$ e $g(x)$, é definida como:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x) \cdot g(t - x)\, dx \tag{1}$$

Essa é a forma contínua da convolução, onde t é o parâmetro de deslocamento e x é o parâmetro de integração. Para sinais discretos, como imagens digitais, usamos uma forma discreta da convolução, que é definida como:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n - m] \tag{2}$$

Aqui, m é o índice de posição na imagem e n é o índice de deslocamento.

2.1.2 Aplicação da Convolução em uma Imagem

Para aplicar uma convolução em uma imagem, usamos um filtro, também conhecido como kernel. Um filtro é uma matriz de números que define a operação a ser realizada em cada região da imagem. Para aplicar a convolução (exemplificado na Figura 1):

1. **Deslize o filtro sobre a imagem**: Posicione o filtro sobre a imagem de modo que cada elemento do filtro corresponda a um elemento da imagem.
2. **Computar o valor da convolução**: Para cada posição do filtro na imagem, calcule a soma ponderada dos elementos do filtro multiplicados pelos elementos correspondentes da imagem.
3. **Armazenar o resultado**: Substitua o valor original do pixel na imagem pela soma calculada.

Este processo é repetido para cada pixel na imagem, resultando em uma nova imagem filtrada.

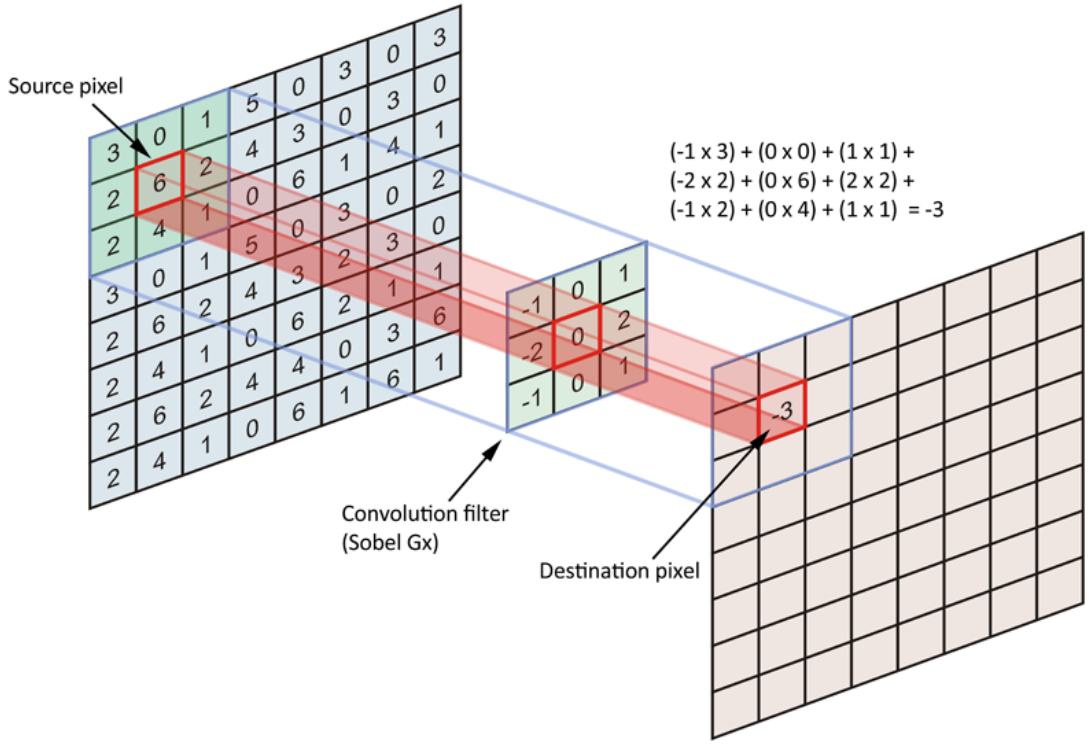

Figura 1: Representação de convolução numa imagem

2.2 Bordas

Note que no processo de convolver uma imagem, tem-se um problema nas bordas da mesma. Uma vez que o centro do filtro é colocado no pixel analisado, deve-se ter um valor das bordas da imagem para que a convolução com o filtro seja completamente aplicada. Com isso em mente, existem certas técnicas para o preenchimento das bordas da imagem, que serão discutidas a seguir. Para isso, será usada como exemplo a matriz apresentada na Equação 10, e será imaginado que o kernel é uma matriz 5x5.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{3}$$

2.2.1 Zero Padding

Essa é a técnica mais simples de todas, e consiste em preencher a borda com zeros:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{2} & \mathbf{3} & 0 & 0 \\ 0 & 0 & \mathbf{4} & \mathbf{5} & \mathbf{6} & 0 & 0 \\ 0 & 0 & \mathbf{7} & \mathbf{8} & \mathbf{9} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

2.2.2 Mirror Padding

Para a aplicação dessa técnica, deve-se imaginar um eixo de simetria nas extremidades da matriz original:

$$\begin{bmatrix} 5 & 4 & 4 & 5 & 6 & 6 & 5 \\ 2 & 1 & 1 & 2 & 3 & 3 & 2 \\ 2 & 1 & \mathbf{1} & \mathbf{2} & \mathbf{3} & 3 & 2 \\ 5 & 4 & \mathbf{4} & \mathbf{5} & \mathbf{6} & 6 & 5 \\ 8 & 7 & \mathbf{7} & \mathbf{8} & \mathbf{9} & 9 & 8 \\ 8 & 7 & 7 & 8 & 9 & 9 & 8 \\ 5 & 4 & 4 & 5 & 6 & 6 & 5 \end{bmatrix} \tag{5}$$

2.2.3 Repetition Padding

Esta é similar ao preenchimento em espelho, mas desta vez imagina-se que o eixo de simetria sejam as próprias linhas e colunas finais da matriz original:

$$\begin{bmatrix} 9 & 8 & 7 & 8 & 9 & 8 & 7 \\ 6 & 5 & 4 & 5 & 6 & 5 & 4 \\ 3 & 2 & \mathbf{1} & \mathbf{2} & \mathbf{3} & 2 & 1 \\ 6 & 5 & \mathbf{4} & \mathbf{5} & \mathbf{6} & 5 & 4 \\ 9 & 8 & \mathbf{7} & \mathbf{8} & \mathbf{9} & 8 & 7 \\ 6 & 5 & 4 & 5 & 6 & 5 & 4 \\ 3 & 2 & 1 & 2 & 3 & 2 & 1 \end{bmatrix} \tag{6}$$

2.2.4 Periodic Padding

Para a aplicação desta técnica, imagina-se que a matriz se repete infinatamente para todos os lados:

$$\begin{bmatrix} 5 & 6 & 4 & 5 & 6 & 4 & 5 \\ 8 & 9 & 7 & 8 & 9 & 7 & 8 \\ 2 & 3 & \mathbf{1} & \mathbf{2} & \mathbf{3} & 1 & 2 \\ 5 & 6 & \mathbf{4} & \mathbf{5} & \mathbf{6} & 4 & 5 \\ 8 & 9 & \mathbf{7} & \mathbf{8} & \mathbf{9} & 7 & 8 \\ 2 & 3 & 1 & 2 & 3 & 1 & 2 \\ 5 & 6 & 4 & 5 & 6 & 4 & 5 \end{bmatrix} \tag{7}$$

2.3 Filtros

2.3.1 Filtro Passa-Baixa

O filtro passa-baixa é um tipo de filtro utilizado em processamento de imagens para suavizar ou borrar a imagem, removendo detalhes de alta frequência. Ele permite apenas a passagem das frequências mais baixas na imagem, resultando em uma redução do contraste e uma suavização das transições de cor.
Na prática, o filtro passa-baixa é aplicado convolvendo a imagem com um kernel que enfatiza a média (filtro de média) dos pixels vizinhos. Isso tem o efeito de suavizar as transições abruptas de cor, resultando em uma imagem mais homogênea e menos ruidosa.

2.3.2 Filtro Passa-Alta

Por outro lado, o filtro passa-alta é usado para realçar as características de alta frequência em uma imagem, destacando bordas e detalhes finos. Ele faz isso permitindo apenas a passagem das frequências mais altas na imagem, suprimindo as informações de baixa frequência.
Assim como o filtro passa-baixa, o filtro passa-alta é aplicado convolvendo a imagem com um kernel específico, geralmente levando em consideração o Laplaciano. Esse kernel realça as diferenças entre os pixels vizinhos, destacando as transições de cor e realçando as características de alta frequência na imagem.

3 Metodologia e Resultados

3.1 Materiais anteriores

Antes de abordar os métodos utilizados para resolver os problemas propostos neste trabalho, é importante ressaltar que estou continuando o desenvolvimento da biblioteca própria de manipulação de imagens apresentada no trabalho anterior. Neste trabalho, foquei em aprimorar essa biblioteca, adicionando novos métodos e funcionalidades para a resolução dos problemas propostos.
Entre as adições feitas, destacam-se:

- **Novos Métodos Adicionados:** Foram adicionados o método *convolution* à classe *Image*. Esse método proporciona a possibilidade de aplicar filtros às imagens.
- **Novas Funções Adicionadas:** Introduzi a função *pgm_from_matrix*, que permite criar uma imagem .pgm a partir de uma matriz de valores. Implementei também a função *kernel_gaussiano*, que cria um kernel para a aplicação do filtro gaussiano.
- **Novos Métodos Estáticos:** Implementei os principais tipos de criação de bordas para uma imagem: *_zero_padding*, *_mirror_padding*, *_rep_padding* e *_periodic_padding*.

Essas adições fortaleceram a funcionalidade da minha biblioteca, tornando-a mais útil para lidar com a aplicação de filtros no processamento de imagens. Com isso, foi possível resolver todos os problemas propostos. Segue em anexo a nova versão da biblioteca:

3.2 Criação de uma imagem a partir de uma matriz

Para as seguintes seções, vide o arquivo do trabalho:
Nesse trabalho foram usadas como objeto de estudos as Imagens 2 e 3.
Para a criação da imagem pedida, desenvolvi uma função que converte uma matriz de valores (que vão de 0 a 255) em uma imagem .pgm, nomeada como *pgm_from_matrix*. Para criar a matriz requisitada, usei um simples algoritmo:

Listing 1: Criação da matriz-imagem

```
matriz = [] #faz a imagem com os 4 quadrados
linha = [] #cria a linha padrao
for pixel in [0, 255]: #itera duas vezes
```

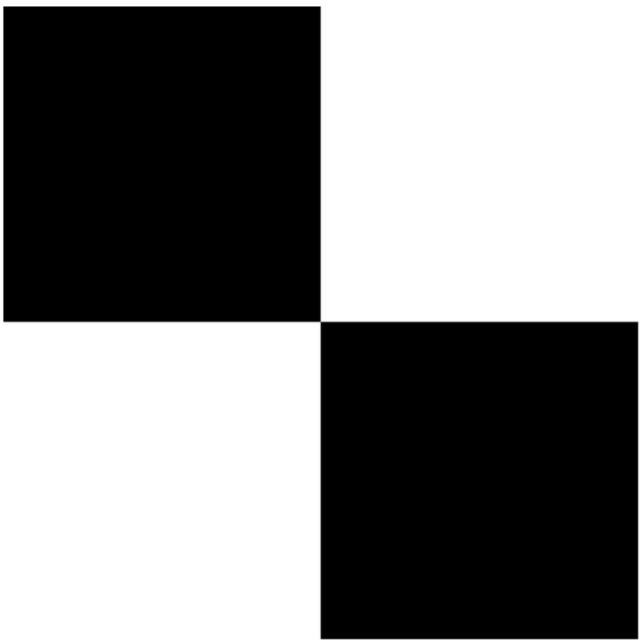

Figura 2: Quadrados

Figura 3: Lena

```
    for j in range(256):
        linha.append(pixel)
for iterador in [1, -1]: #itera duas vezes
    for j in range(256):
        matriz.append(linha[::iterador]) #iterador inverte a ordem

quadrados = img.pgm_from_matrix("caminho_arquivo", matriz)
img.print_grayscale_image(quadrados)
```

O resultado disso pode ser visualizado na Imagem 2.

3.3 Estudo da escolha das bordas

Nesta seção, estuda-se como a escolha da borda afeta a imagem que será convolvida. Para isso, aplica-se um filtro de média na Imagem 2, com escolhas de preenchimento de bordas diferentes.
Utilizando o Zero Padding, o resultado está na Imagem 4.

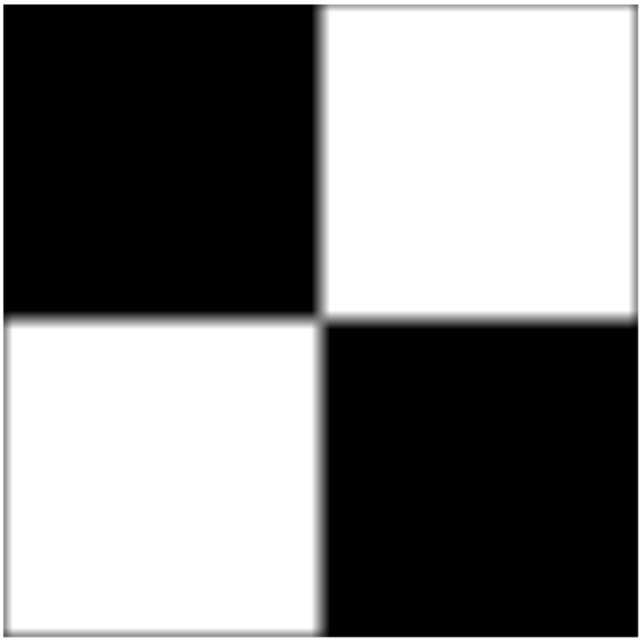

Figura 4: Quadrados com Zero Padding

Note que como o preenchimento foi feito com zeros (a intensidade mais baixa), aparece uma linha preta em volta dos quadrados brancos. Isso porque a média leva em consideração a borda com zeros, e diminui a intensidade daqueles pixels com a intensidade alta.
Utilizando o Mirror Padding, o resultado está na Imagem 5.

Note que com a utilização dessa borda, não aparecem "intrusos"nos limites da imagem, uma vez que esse tipo de borda apenas reflete os pixels numa deter-

Figura 5: Quadrados com Mirror Padding

minada direção.
Utilizando o Repetition Padding, o resultado está na Imagem 6.

Note que esta imagem é idêntica à Imagem 5. Isso se dá pois esses métodos de preenchimento de bordas são muito semelhantes, podendo, neste caso, diferir somente nas linhas mais na extremidade da figura. Porém, para este caso em específico, pode-se considerar as imagens como idênticas.
Utilizando o Periodic Padding, o resultado está na Imagem 7.

Note como nesta imagem, aparece também uma faixa branca em volta dos quadrados pretos. Como essa escolha de bordas, a grosso modo, "colocaria os quadrados de baixo em cima da imagem", isso faz com que o quadrado preto faça uma média com quadrados brancos em volta dele, e que o quadrado branco faça uma média junto com quadrados pretos. Isso faz com que as bordas da imagem resultante sofram uma grande interferência do Periodic Padding.
Dito isso, pode-se ver que para a aplicação de uma convolução numa imagem, deve-se escolher cautelosamente a técnica de preenchimento de bordas, pois esta é um fator determinante sobre como vão ficar os pixels mais às extremidades da imagem resultante.

3.4 Aplicação de filtro passa-baixa

Como já discutido, um filtro passa-baixa basicamente suaviza as transições da imagem, e "corta"as maiores intensidades da imagem. Para isso, o kernel a ser usado é um kernel de média; ou seja, a convolução resultante será basicamente uma média com os vizinhos de um pixel. Com isso em mente, foram aplicados filtros com um kernel do tipo box, e com um kernel do tipo gaussiano, que serão explicados logo mais.

Figura 6: Quadrados com Repetition Padding

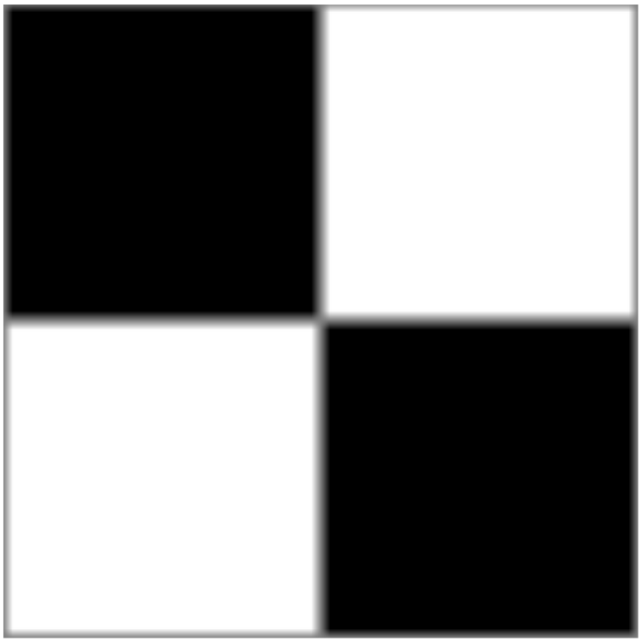

Figura 7: Quadrados com Periodic Padding

3.4.1 Filtro Box

Um filtro box, leva em consideração a intensidade dos pixels vizinhos numa relação de 1:1:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{8}$$

Então, basicamente, quando o filtro de média for aplicado, a intensidade do pixel analisado terá o mesmo peso que o de seus vizinhos. Na Figura 8, tem-se o resultado de se aplicar um filtro de média tipo box na Figura 3 com diferentes tamanhos de kernel $k \times k$.

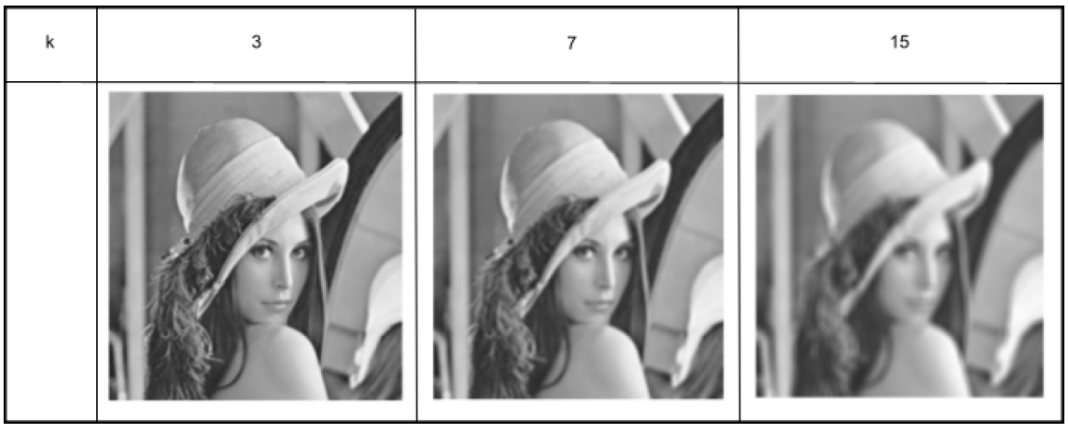

Figura 8: Filtro de Média tipo Box

Note que a figura resultante é uma versão mais embaçada da figura original; e que, quanto maior o tamanho do kernel, mais borrada a figura fica. Isso pode ser explicado dado que quanto maior o kernel, mais vizinhos do pixel serão considerados na convolução, o que interfere no resultado final da média.

3.4.2 Filtro Gaussiano

Um filtro gaussiano, leva em consideração as intensidades dos pixels vizinhos de acordo com uma distribuição Gaussiana (atribui um peso maior para o pixel central, e decresce os pesos de maneira Gaussiana para os vizinhos). Defino aqui então, o kernel gaussiano w:

$$w(s,t) = \frac{1}{2\pi\sigma^2} e^{-\frac{s^2+t^2}{2\sigma^2}} \tag{9}$$

Onde s indica a distância em linhas do centro, e t indica a distância em colunas do centro. Com isso em mente, eu implementei a função *kernel_gaussiano*, que dado um desvio padrão σ, e uma ordem k para a matriz, retorna o kernel gaussiano:

Listing 2: Kernel Gaussiano

```
def kernel_gaussiano(desvio_padrao, k):
    aux = int((k - 1) / 2) #tamanho do kernel
    K = 1 / (2 * pi * desvio_padrao ** 2) #constante K
```

```
    matriz_gauss = [] #kernel gaussiano
    for s in range(-aux, aux + 1): #linha
        linha = []
        for t in range(-aux, aux + 1): #coluna
            valor = K *      exp(-1 * (s ** 2 + t ** 2) /
                (2 * desvio_padrao ** 2)) #valor a ser
                colocado no kernel
            linha.append(valor)
        matriz_gauss.append(linha)

    return matriz_gauss
```

Assim, usando o método *convolution* da classe Image, posso aplicar um filtro gaussiano na imagem. Os resultados da aplicação deste filtro estão na Figura 9, onde k indica a ordem do kernel, e σ indica o desvio padrão usado.

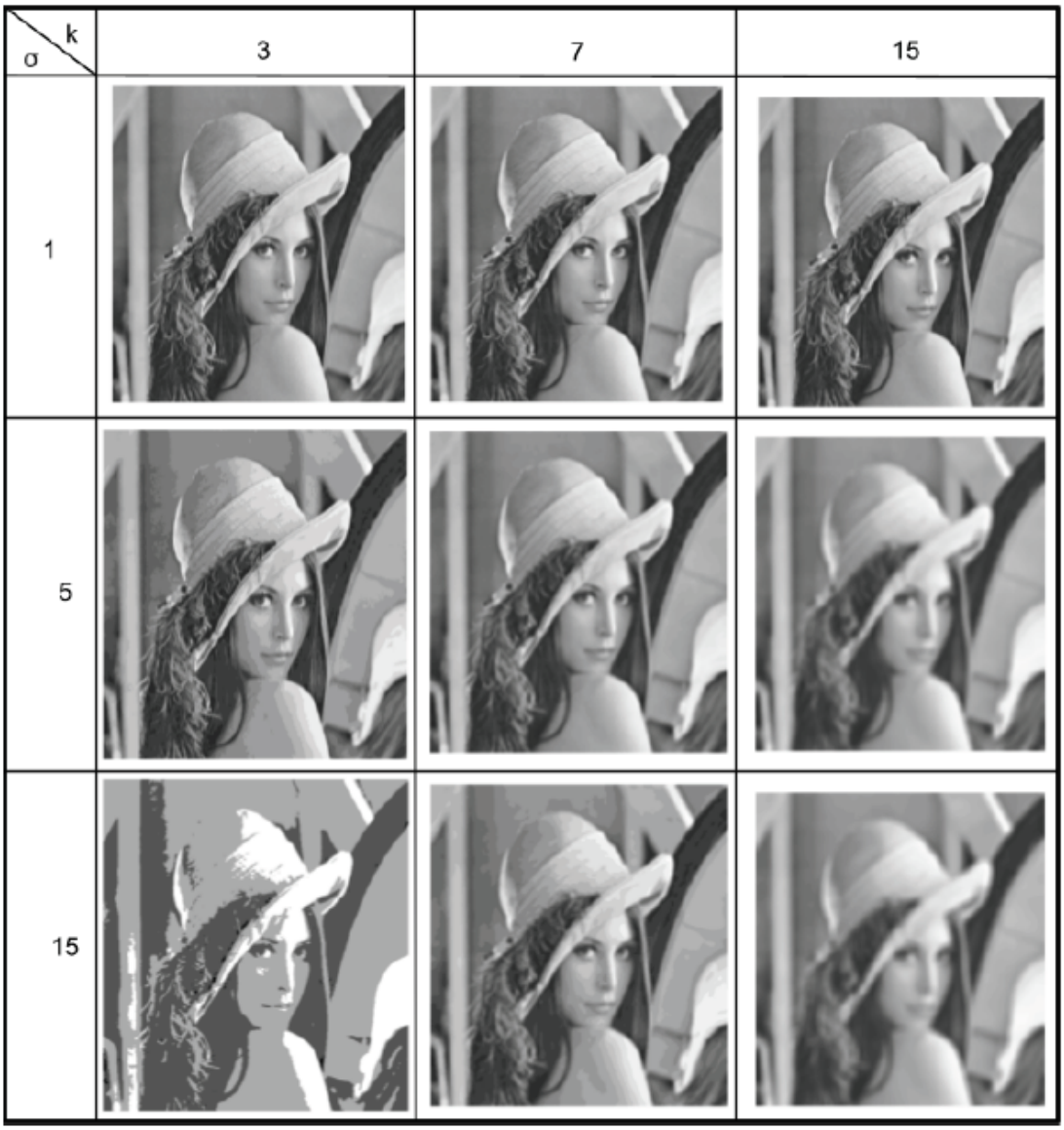

Figura 9: Filtro de Média tipo Gaussiano

O desvio padrão é uma medida que expressa o grau de dispersão de um conjunto de dados. Ou seja, indica o quanto um conjunto de dados é uniforme.

Quanto mais próximo de 0 for o desvio padrão, mais homogêneo são os dados. Note que na Figura 9, quanto maior a ordem k, de acordo com o que foi dito a respeito da Figura 8, mais embaçada fica a figura. Porém, note também como a escolha do desvio padrão influencia na figura resultante: quanto maior for, mais "segmentada"fica a figura. Uma vez que um filtro de média tenta homogeneizar a distribuição de intensidades, isso ocorre pois quanto maior for o desvio padrão, mais o filtro atuará nas intensidades do pixel para atingir seu objetivo.

3.5 Detecção de bordas

Para que se detectem as bordas de uma imagem, aplica-se um filtro passa-alta; ou seja, um filtro cujo kernel mantenha as altas intensidades dos pixels, e "corte"as baixas. Assim, usei uma matriz de convolução na forma de uma filtragem passa-alta laplaciano normalizado:

$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{10}$$

Quando este kernel é aplicado a uma imagem, a imagem resultante é composta basicamente somente pelas bordas da imagem orignal, como pode-se ver na Figura 10.

Figura 10: Detecção de Bordas Normalizada

Caso se queira realçar as bordas da imagem original, pode-se "somar"o kernel utilizado com um kernel identidade, definido por:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{11}$$

O resultado da imagem realçada está na Figura 11

Figura 11: Realce de Bordas Normalizada

Caso o kernel não seja normalizado, o efeito produzido está nas Figuras 12 e 13.

Note que, caso o kernel não seja normalizado, é inserido muito ruído na imagem. Portanto, para maior eficiência, usa-se o filtro normalizado.

4 Conclusão

Em conclusão, este trabalho explorou os fundamentos e aplicações da convolução em imagens digitais. Iniciamos com uma explicação detalhada da convolução, tanto na forma contínua quanto discreta, destacando sua importância no processamento de imagens. Em seguida, discutimos a aplicação prática da convolução em imagens, demonstrando como os filtros podem ser usados para realizar operações como suavização e detecção de bordas.

Além disso, investigamos os efeitos das diferentes técnicas de preenchimento de bordas na aplicação da convolução, destacando como essa escolha pode impactar significativamente o resultado final. Observamos que a escolha da técnica de preenchimento de bordas deve ser feita com cuidado, levando em consideração as características específicas da imagem e o resultado desejado.

Em seguida, exploramos a aplicação de filtros passa-baixa, como os filtros de média e gaussiano, para suavizar imagens e remover detalhes de alta frequência.

Figura 12: Quadrados

Figura 13: Lena

Demonstramos também como o tamanho do kernel e o desvio padrão afetam a intensidade do efeito de suavização.
Por fim, examinamos a detecção de bordas usando filtros passa-alta, destacando como os kernels laplacianos podem ser usados para realçar as transições de intensidade na imagem. Observamos a importância da normalização do kernel para evitar a introdução de ruído indesejado na imagem resultante.