Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Processamento Digital de Imagens

Relatório do Trabalho 2 - Filtragem High-Boost

Professor: Franklin Cesar Flores

Vítor Hugo Santos de Camargo RA: 116426

1 Introdução

A visão é um dos sentidos mais dominantes nos seres humanos, e imagens têm sido uma forma fundamental de comunicação e expressão ao longo da história. Na era digital atual, com o advento das câmeras digitais e dos dispositivos móveis, nunca antes produzimos e consumimos tantas imagens. Contudo, nem todas as imagens capturadas são perfeitas em termos de nitidez e detalhes. Fatores como movimento da câmera, configurações de foco incorretas ou condições de iluminação adversas podem resultar em imagens que não são tão nítidas quanto desejado.

A filtragem é uma técnica essencial no processamento digital de imagens, permitindo aprimorar, modificar ou extrair informações específicas de uma imagem. Dentre várias técnicas de filtragem, a filtragem High-Boost destaca-se como uma abordagem que visa realçar ou enfatizar detalhes em uma imagem, como bordas ou texturas, que podem não ser claramente visíveis na imagem original.

O método de filtragem High-Boost, em essência, amplifica características de alta frequência (detalhes) de uma imagem. Ele faz isso subtraindo uma versão suavizada da imagem da original para obter uma "máscara"de realce. Esta máscara captura os detalhes que queremos realçar. Em seguida, adicionamos esta máscara amplificada de volta à imagem original, produzindo uma imagem com detalhes realçados.

Neste trabalho, apresentamos uma implementação da técnica de filtragem High-Boost utilizando as bibliotecas OpenCV e Numpy da linguagem de programação Python. Descrevemos o processo passo a passo, ilustrando como essa técnica pode ser utilizada para melhorar a nitidez e realçar detalhes em imagens digitais.

2 Desenvolvimento

A técnica de filtragem High-Boost é uma abordagem para realçar características de alta frequência em imagens, como bordas e texturas. Essencialmente, esta técnica busca aprimorar os detalhes de uma imagem ao adicionar um valor amplificado dos detalhes da imagem original. A técnica combina o uso de filtragem passa-baixa para suavizar a imagem e posteriormente subtrair essa imagem suavizada da original, gerando assim a máscara que captura os detalhes.

A implementação principal dessa técnica é realizada na função high_boost _filtering. Esta função recebe o caminho da imagem e outros parâmetros para definir o grau de realce e o tamanho da máscara de suavização. Os passos seguintes detalham a operação:

1. **Carregamento da Imagem:** A imagem é carregada usando a função `imread` do OpenCV. Ela é carregada em tons de cinza para simplificar a filtragem.

2. **Suavização:** Uma máscara de média 3x3 é criada para suavizar a imagem original. Esta máscara essencialmente toma a média dos valores dos pixels em uma janela, o que tem o efeito de suavizar a imagem.

3. **Geração da Máscara:** A imagem suavizada é subtraída da imagem original para gerar a máscara de realce.

4. **Filtragem High-Boost:** A máscara é então adicionada de volta à imagem original. O peso da máscara durante essa adição é controlado por um parâmetro k. Quando k é igual a 1, a técnica é simplesmente chamada de realce de unsharp. Quando k é maior que 1, estamos amplificando os detalhes mais do que o normal, daí o nome High-Boost.

Vamos analisar um trecho do código:

```
1 blur_mask = np.ones(blur_mask_size, dtype=np.float32) / (blur_mask_size
      [0] * blur_mask_size[1])
2 img_blurred = cv2.filter2D(img, -1, blur_mask)
3 mask = cv2.subtract(img, img_blurred)
4 img_highboost = cv2.addWeighted(img, 1, mask, k, 0)
```

Neste trecho, a variável `blur_mask` define uma máscara (ou kernel) de suavização. Essa máscara é uma matriz de tamanho `blur_mask_size` (3x3) preenchida com valores uniformes, onde cada valor é o inverso da área total da máscara. Esta é uma técnica comum para criar uma máscara de média, usada para realizar uma filtragem de média (ou suavização) na imagem. A filtragem de média é um tipo de filtragem passa-baixa que tem o efeito de reduzir o ruído e suavizar a imagem.

`img_blurred` é a versão suavizada da imagem original `img`, obtida usando a função `filter2D` do OpenCV com a máscara `blur_mask`, o valor '-1' especifica que a imagem resultante deve ter a mesma profundidade de bits da imagem original.

A função `subtract` do OpenCV é utilizada para gerar a máscara `mask`, que captura os detalhes da imagem subtraindo a imagem borrada da original.

Finalmente, a função `addWeighted` combina a imagem original e a máscara com o fator de realce k para produzir a imagem final realçada. O peso de '1' atribuído à imagem original garante que sua essência seja preservada, enquanto os detalhes realçados pela máscara são progressivamente incorporados com base no valor de k.

O uso dessa técnica pode ser extremamente útil em situações em que uma imagem contém detalhes que não são claramente visíveis na versão original. Por exemplo, em imagens médicas, onde a clareza de certos detalhes pode ser vital, ou em aplicações de visão computacional onde detalhes de borda são essenciais para a detecção de objetos.

3 Resultados

Nesta seção, apresentamos os resultados obtidos ao aplicar a filtragem High-Boost em algumas imagens. Foram testados diferentes valores de k para avaliar o impacto deste parâmetro na nitidez da imagem. As imagens testadas foram obtidas do livro do Processamento de Imagens Digitais, de Gonzalez e Woods. [1]

Figura 1: Imagem 3.38 original

Figura 2: Imagem 3.38 após a filtragem high-boost com k = 2,0

Figura 3: Imagem 3.38 após a filtragem high-boost com k = 4,5

Figura 4: Imagem 3.38 após a filtragem high-boost com k = 10,0

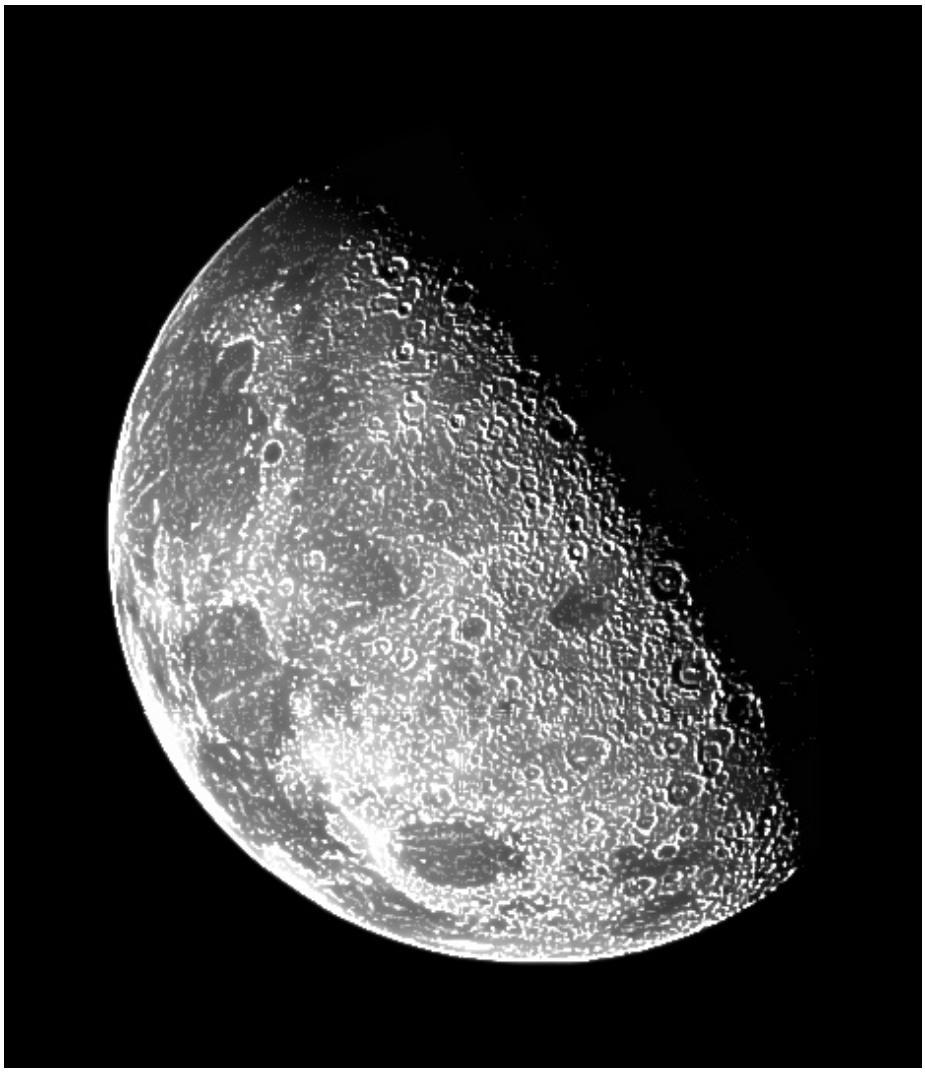

Figura 5: Imagem 3.38 após a filtragem high-boost com k = 30,0

Na imagem 3.38, que retrata a Lua, é evidente um aumento de nitidez conforme o valor de k cresce. No entanto, há um limite para essa melhoria. A partir de k = 10, os detalhes são amplificados a um ponto que começa a gerar distorções na imagem. Essas distorções não apenas comprometem a qualidade visual, mas também têm o potencial de introduzir informações que podem ser interpretadas de maneira errônea ou enganosa.

Figura 6: Imagem 3.40 Original

Figura 7: Imagem 3.40 após a filtragem high-boost com k = 2,0

Figura 8: Imagem 3.40 após a filtragem high-boost com k = 4,5

Figura 9: Imagem 3.40 após a filtragem high-boost com k = 10,0

Figura 10: Imagem 3.40 após a filtragem high-boost com k = 30,0

Na imagem 3.40 (texto), observa-se um aprimoramento contínuo da nitidez à medida que o valor de k aumenta. Notavelmente, mesmo com um valor tão elevado quanto k=30, a imagem ainda retém sua utilidade, não apresentando distorções tão pronunciadas quanto na imagem da Lua. Este comportamento sugere que, dependendo das características e conteúdos específicos de uma imagem, diferentes valores de k podem ser ideais para realçar detalhes sem introduzir artefatos indesejáveis. No entanto, um valor ideal parece situar-se entre 4,5 e 10, tornando-se mais evidente ao ampliar as imagens.

4 Código

```
1
2  import numpy as np
3  import cv2
4  import matplotlib.pyplot as plt
5
6  def high_boost_filtering(img_path, k=4.5, blur_mask_size=(3, 3)):
7      # Carregar a imagem
8      #img = f_x_y
9      img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
10
11     # mscara de mdia
12     blur_mask = np.ones(blur_mask_size, dtype=np.float32) / (
       blur_mask_size[0] * blur_mask_size[1])
13
14     #1. Borrar a imagem original
15     #img_blurred = f^_ (f_borrada)
16     img_blurred = cv2.filter2D(img, -1, blur_mask)
17
18     #2.  Subtrai a imagem borrada da original (a diferena resultante
       chamada de mscara)
```

```
    #mask = g_mascara(x,y)
    mask = cv2.subtract(img, img_blurred)

    # 3. Adicionar a mscara  imagem original
    #img_highboost = g_x_y
    img_highboost = cv2.addWeighted(img, 1, mask, k, 0)

    return img, img_blurred, mask, img_highboost

def show_individual_image(img, title):
    plt.figure(figsize=(6,6))
    plt.imshow(img, cmap='gray')
    plt.title(title)
    plt.axis('off')
    plt.tight_layout(pad=0)
    plt.subplots_adjust(left=0, right=1, top=0.95, bottom=0)

def show_comparison(img_path, k=4.5):
    original_img, blurred_img, mask, highboost_img = high_boost_filtering
    (img_path, k)

    show_individual_image(original_img, "Original Image")
    show_individual_image(blurred_img, "Blurred Image")
    show_individual_image(mask, "Mask")
    show_individual_image(highboost_img, "Highboost Image")

    plt.show()

def selecionar_imagem():
    print("Selecione uma das imagens:")
    print("1 - 3.38.tif")
    print("2 - 3.40.tif")
    print("3 - Inserir seu prprio caminho de imagem")

    opcao = input("Digite o nmero correspondente  imagem desejada (1-3): 
    ")

    if opcao == "1":
        return "3.38.tif"
    elif opcao == "2":
        return "3.40.tif"
    elif opcao == "3":
        return input("Digite o caminho completo da sua imagem: ")
    else:
        print("Opo invlida. Por favor, escolha novamente.")
        return selecionar_imagem()

def main():
    img_path = selecionar_imagem()
    k = float(input("Digite o valor de k: "))
```

```
67      show_comparison(img_path, k)
68      print(f"Imagens␣processadas␣e␣salvas␣na␣pasta␣'resultados'.")
69
70  if __name__ == "__main__":
71      main()
```

Referências

[1] Rafael C. Gonzalez and Richard E. Woods. *Processamento de Imagens Digitais*. Pearson Education do Brasil, So Paulo, 3 edition, 2009. Traduzido de: Digital image processing, 3rd ed. Pearson Education, Inc., 2008.

[2] Image Processing Place. Image processing learning resources. Acessado em: 19/10/2023. URL: `https://www.imageprocessingplace.com/root_files_V3/image_databases.htm`.