Relatório Trabalho 1 - BoxFilter

Renato Alberto dos Santos[1], **Rodrigo Norio Tsuhako**[2]

[1]Departamento de Informática - Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

ra96565@uem.br, ra95376@uem.br

Resumo. *Os filtros de imagens digitais reduzem os ruídos da imagem de entrada afim de gerar uma imagem de saída mais adequada para uma aplicação específica. O box filter e o downsampling são exemplos de filtros que realizam a redução de dados da imagem. O presente trabalho tem como objetivo aplicar uma redução de escala utilizando o box filter e o downsampling. Os resultados mostram que o box filter gera saídas com maior qualidade, porém o downsampling mostrou ser mais rápido para gerar os resultados.*

Palavras-chave*: processamento digital de imagens, box filter, downsampling.*

1. Introdução

Uma imagem digital pode ser descrita por coordenadas bidimensionais onde seus valores são discretos e finitos. O processamento digital de imagem pode ser caracterizado por qualquer meio de processamento de dados em que os parâmetros de entrada e saída sejam imagens digitais.

Neste contexto, os filtros de imagens são técnicas que aplicam um barramento de dados na imagem digital de entrada, gerando uma imagem digital como saída. Na aplicação de um filtro, nem todos os elementos na imagem de entrada irão compor a imagem de saída. Portanto, é necessário que esses algoritmos consigam filtrar dados, mantendo a essência da imagem de origem.

O presente trabalho busca aplicar uma redução de escala usando os conceitos do box filter e do downsampling em imagens com tons de cinza.

2. Box Filter

O box filter é uma técnica que reduz os ruídos em um domínio espacial, ou seja, manipula os valores de uma área da imagem. Um box filter simples consiste em calcular a média aritmética deste espaço e converter em apenas um valor [Gonzalez and Woods 2010].

O box filter é uma técnica eficiente em questão de filtrar imagens, pois ele reduz a variação de intensidade em um espaço na imagem. A ideia consiste em substituir cada pixels da imagem pela média de um espaço [Sanches et al. 2015].

3. Downsampling

O downsampling é uma operação em que a imagem digital de entrada sofre uma redução dos dados. Neste processo ocorre perdes irreversíveis, ou seja, não é possível voltar a imagem de saída exatamente como ela era antes [Agostini and Bampi 2001].

4. Metodologia

Para a execução do trabalho foi utilizado como base as seguintes especificações.

4.1. Hardware e Software

O trabalho foi desenvolvido e testado em uma máquina com as seguintes especificações:

- Processador Intel Core I5-4210, com frequência de 1,7 GHz;
- 8G de memória RAM;
- HD de 1TB;
- Sistema operacional Windows 10 versão 1909.
- Editor de texto Visual Studio Code
- Linguagem Python 3.8.5

4.2. Condução dos testes

O presente trabalho utilizou como entrada uma imagem em tons de cinza com dimensões 1024×1024.

Figura 1. Imagem de entrada

A figura 1 representa os dados de entrada, na qual seu tamanho é quadrado e em potência de 2.

Além da imagem, outro parâmetro de entrada é a taxa de redução. Afim de simplificar o processo, o tamanho da imagem e a taxa de redução são potências de 2.

Nos testes, primeiramente foi utilizado valor de redução igual a 8 e em seguida redução igual a 16.

5. Resultados

O primeiro teste utilizou valor de redução 8 para o box filter e para o downsampling por fatiamento.

Figura 2. Redução Box Filter - 8

Na figura 2 é possível observar que após aplicar o box filter, a imagem de saída ficou levemente ”embaçada”. E o tempo para executar os cálculos de redução foi de 52,81 segundos.

Figura 3. Redução Downsampling - 8

Enquanto na figura 3, é possível observar que aplicando o downsampling a qualidade da imagem de saída é baixa. Entretanto, a imagem foi gerada instantaneamente (0 segundos).

O segundo teste utilizou valor de redução igual a 16 para o box filter e para o downsampling.

Figura 4. Redução Box Filter - 16

Na figura 4 é possível observar que aumentando o valor de redução no box filter, a qualidade da imagem diminuiu bastante. Porém, a execução dos cálculos de redução foram mais rápidos, demorando apenas 12.037 segundos.

Figura 5. Redução Downsampling - 16

O downsampling ilustrado na figura 5, também piorou sua qualidade de imagem, mas manteve o tempo de 0 segundos para gerar a imagem.

6. Conclusão

Em algumas aplicações na área de imagens digitais, é necessário filtrar as imagens afim de reduzir seus ruídos. Os algoritmos de filtros, tem capacidade de reduzir esses dados. O box filter e downsampling são exemplos de filtros simples e eficientes.

Considera-se que o objetivo deste trabalho foi atendido, uma vez que foi aplicado e analisado os conceitos do box filter e do downsamplingem em imagens com tons de cinza.

Com base nos resultados obtidos, é possível observar que a qualidade de redução do box filter é melhor em relação ao downsamplig por fatiamento. Por outro lado, o downsampling é mais rápido e gera a saída de forma instantânea. Além disso, foi observado que o valor da redução influencia no resultado final. Quanto menor o valor da redução, melhor a qualidade da saída e quanto maior a redução, pior a qualidade da saída. No caso do box filter, valores baixos de redução aumentam o tempo de cálculo, e valores altos diminuem o tempo.

Portanto, é possível concluir que em aplicações que a qualidade da imagem importa, o box filter é mais interessante. Por outro lado, aplicações que necessitam de processamento rápido, o downsampling por fatiamento é melhor.

7. Implementação

O código a seguir foi desenvolvido em Python 3.8.5 utilizando as bibliotecas numpy e cv2.

```
import numpy as np
from matplotlib import pyplot as plt
import cv2
import time


#carrega imagem
def carregarimagem():
    imagem = input('Digite o nome da imagem: ')
    img = cv2.imread(imagem)
    matimg = np.array(img)

    return matimg, img

#calcula matriz
def calculamatriz(newimg, auxiliar, matimg, saida):
    for i in range(len(newimg[0])):
        for j in range(len(newimg)):
            auxbool = auxiliar == newimg[i][j]
            res = matimg[auxbool]
            saida[i][j] = np.mean(res)

    saida = np.uint8(saida)
    return saida

#calcula slice
def calculaslice(matimg,redmen):
    saida1 = matimg[::redmen,::redmen]
```

```
    return saida1


#show imagem
def showimagemmatplot(img,saida):
    plt.subplot(121),plt.imshow(img),plt.title('entrada')
    plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(cv2.cvtColor(saida, cv2.COLOR_BGR2RGB))
    plt.xticks([]), plt.yticks([])
    plt.show()



def showimagemcv(img,saida):
    cv2.imshow('entrada',img)
    cv2.imshow('saida', saida)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


##MAIN
matimg, img = carregarimagem()
altura = len(matimg)
largura = len(matimg[0])
redmen = int(input('Digite a taxa de redução (potência de 2): '))
redalt = altura/redmen
redlarg = largura/redmen


b = redlarg*redalt
newimg = np.arange(b)
newimg = newimg.reshape(int(redlarg),int(redalt))
saida = np.arange(b)
saida = newimg.reshape(int(redlarg),int(redalt))
auxiliar = np.repeat((np.repeat(newimg,redmen,axis=0)),redmen,axis=1)

#Box filter
inicio = time.time()
saida = calculamatriz(newimg, auxiliar, matimg, saida)
fim = time.time()
print(fim - inicio)
saidaaux = np.repeat(np.repeat(saida, redmen, 0), redmen, 1)

showimagemcv(img,saidaaux)
#showimagemmatplot(img,saida)
```

```
#Fatiamento
inicio = time.time()
saida1 = calculaslice(matimg,redmen)
fim = time.time()
print(fim - inicio)
saida1aux = np.repeat(np.repeat(saida1, redmen, 0), redmen, 1)

showimagemcv(img,saida1aux)
#showimagemmatplot(img,saida1)
```

Referências

Agostini, L. and Bampi, S. (2001). Arquitetura integrada para conversor de espaço de cores e downsampler para a compressão de imagens jpeg. *VII Workshop IBERCHIP*.

Gonzalez, R. and Woods, R. (2010). *Processamento Digital de Imagens*, volume 3. PEARSON.

Sanches, C. H., Fontoura, P. J., Viera, P. F., and Batista, M. A. (2015). Técnicas de suavização de imagens e eliminação de ruídos. *Anais do EATI - Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação*.