Universidade Estadual de Maringá
Departamento de Informática
Ciência da Computação
Processamento DIgital de Imagem

Box Filter

Caio Eduardo Kikuti Machado RA:103235
Mateus Felipe Larrosa Furlan RA:102951

Professor: Franklin Cesar Flores

Maringá
2020

Sumário

1 Introdução **2**

2 Metodologia **2**

3 Disposições do Código **2**

4 Resultados e discussão **4**

5 Conclusão **9**

1 Introdução

Os filtros passa-baixa suavizam a imagem atenuando as altas frequência que correspondem às transições abruptas. Esses filtros tendem a minimizar os ruídos da imagem a causar o efeito de borramento da imagem. Nas imagens de maneira geral diz-se que existem alguns componentes de imagem reesposáveis por gerar altas frequências, como por exemplo boras, lados e outras transições abruptas de nível de cinza. A utilização de um filtro de passa baixa causa a perda de detalhes dos componentes de alta frequência de uma imagem. [1]

O algoritmo de Box-blurr ou Box-Filter é um filtro passa baixa linear em que, cada pixel na imagem resultante é igual a média dos seus pixeis vizinhos da imagem original. Por exemplo, para uma matriz 3x3 com proporção 1, pode ser escrita da seguinte maneira:

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

Existem dois problemas na forma de abordar o uso do Box Filter, o primeiro é que para proporções extremamente grandes em imagens pequenas o filtro tem um efeito de deixar a imagem pixelada, tirando a completa nitidez da imagem e não apenas fazendo a suavização dos componentes de alta frequência da imagem. O segundo problema é a complexidade de tempo que o algoritmo implementado de maneira "bruta" demanda, para uma matriz nxm o algoritmo atinge uma complexidade da ordem de $O(n^2 * m^2)$, ou seja no pior caso o algoritmo demanda uma complexidade de tempo da ordem de $O(n^4)$.

O objetivo deste trabalho é a implementação do filtro Box Filter, por meio da linguagem de programação python, utilizando como auxílio as bibliotecas Open CV, numpy e Sci-kit Image. Para a diminuição dos problemas com complexidade citados no parágrafo anterior tentou-se aplicar uma implementação que minimizasse os custos computacionais. Este relatório busca mostrar como o código foi desenvolvido, quais foram os resultados alcançados pelo algoritmo e ainda realizar uma comparação entre o Box filter e um *downsamplig* simples implementado por meio do fatiamento.

2 Metodologia

Conforme explicado na seção anterior para o desenvolvimento deste trabalho foi utilizada a linguagem Python utilizando alguns recursos das bibliotecas, numpy para manipulação de vetores e Open CV e Sci-kit Image para a manipulação da imagem, como leitura e conversão de tipo. Um dos motivos para a utilização de vetores numpy é pela eficiência implementada pela biblioteca. O objeto *array* nessa biblioteca é chamado de *ndarray* e prove funções de suporte e um funcionamento cerca de 50x mais rápido que a lista tradicional implementada por padrão pela linguagem Python. No caso da biblioteca OpenCV a utilização limitou-se a leitura, armazenamento e visualização das imagens em questão. Já a utilização da biblioteca Sci-kit Image restinguiu-se a utilização a uma função para redimensionamento da imagem.

3 Disposições do Código

Para a implementação do código realizou-se a separação em duas partes, primeiramente efetua-se a leitura da imagem por meio da função da biblioteca OpenCV, **imread()** com o parâmetro zero para a leitura da imagem em escala de cinza. Em seguida é chamada a função do Box Filter que recebe como parâmetros a imagem e uma proporção e realiza a "redução" da imagem nos seguintes passos: instanciação dos vetores auxiliares, cálculo das submatrizes que da matriz que representa a imagem, cálculo da média, inserção do pixel que representa a média dos pixeis vizinhos no vetor que representa uma nova linha da imagem reduzida. Vale ressaltar que para o devido processamento da imagem foi necessária a conversão do vetor para o tipo *np.uint8* . Abaixo segue o código da redução da imagem:

```
def boxFilter(imagem, taxa):
    # GUARDAR A COLUNA COM BLURR JA CALCULADO
    blurRow = []
    # blurr image
    blur_image = []
    # tamanho da linha  imagem
    tamLinhas = imagem.shape[0]
    # tamanho da coluna da imagem
    tamColunas = imagem.shape[1]
    # iteradores
    row = 0
    colum = 0

    while row < tamLinhas:
```

```
        while colum < tamColunas:
            media = imagem[row:taxa+row, colum:taxa+colum]
            newPixel = round(media.mean())
            colum = colum + taxa
            blurRow.append(newPixel)
        blur_image.append(blurRow)
        blurRow = []
        row +=taxa
        colum =0

    return np.array(blur_image, dtype=np.uint8)
```

Para fins de comparação do algoritmo do Box Filter implementado, utilizou-se o recurso do fatiamento para a realização de um *downsampling* simples da imagem de entrada a fim de que se compare os resultados obtidos por pelo algoritmo implementado e pela técnica de fatiamento. Para realização do fatiamento simples basta passar a proporção na qual deseja-se reduzir a imagem:

```
downsampling = img1[::int(taxa),::int(taxa)]
```

Uma vez realizada a redução de ambas imagens é realizado o redimensionamento da imagem para o tamanho original por meio da função **resize()** implementada na biblioteca Sci-kit Image e posterior armazenamento das imagens suavizadas nos seus respectivos formatos, conforme o código abaixo.

```
def main():
 if (len(sys.argv)!=3):
     print("Error, Try like this:
     python boxfilter.py <nome_do_arquivo_de_imagem.png>
     <TAXA DE REDUÇÃO>")
     sys.exit()

 path ='./'
 # Abrindo a imagem
 nomeImagem = str(sys.argv[1])
 taxa = int(sys.argv[2])
 img1 = cv2.imread(nomeImagem, 0)
 #Downsampling simples
 downsampling = img1[::int(taxa),::int(taxa)]
 # box filter
 filtrado = boxFilter(img1, taxa)

 #Voltando ao tamanho normal BOX FILTER
 upsamplingBOX = resize(filtrado, (img1.shape[0], img1.shape[1]))
 upsamplingBOX = cv2.convertScaleAbs(upsamplingBOX, alpha=(255.0))
 cv2.imwrite(os.path.join(path ,'BOXFILTERED'+str(taxa)+nomeImagem), upsamplingBOX)

 #Voltando ao tamanho normal downsampling
 upsamplingDOWN = resize(downsampling, (img1.shape[0], img1.shape[1]))
 upsamplingDOWN = cv2.convertScaleAbs(upsamplingDOWN, alpha=(255.0))
 cv2.imwrite(os.path.join(path ,'DOWNSAMPLED'+str(taxa)+nomeImagem), upsamplingDOWN)
```

Para a análise na próxima seção foram utilizadas quatro imagens para a realização dos testes, o nome das imagens e suas respectivas proporções estão contidas na tabela a seguir:

Nome Imagem	Height	Width
ronaldinho	256	256
cat	512	512
cr7500x500	500	500
paisagem	1024	1024

Tabela 1: Dados das imagens de entradas

Para a efetiva comparação dos dois processos de filtragem contidos neste trabalho é necessária a definição das métricas que serão analisadas. Se tratando da suavização de imagens é interessante que se faça a análise do *aliasing* das imagens.

Segundo [1] o *aliasing* é provocado pela subamostragem de uma função, em outras palavras o *aliasing* é um processo no qual componentes de alta frequência de uma função se ”mascaram“ como frequências mais baixas na função amostrada. Isso está de acordo com a utilização comum do termo alias, que significa falsa identidade. Numa abordagem mais prática, a comparação das imagens se deu por meio da análise das bordas, ou seja o quão suavizada encontram-se as bordas da imagens após a aplicação dos filtros.

4 Resultados e discussão

Como primeira amostra de imagem tem-se a foto de uma pessoa, mais especificamente o conhecido jogador Ronaldinho Gaúcho,com o tamanho 256x256, reduzido por uma proporção 2. A imagem original e colorida corresponde a figura 1. E as figuras 2a e 2b correspondem as figuras após a aplicação do filtro.

Figura 1: Amostra 256x256 original

(a) Imagem Ronaldinho com Box filter.

(b) Imagem Ronaldinho com Downsamplig simples.

Figura 2: Amostra 256x256 após aplicação dos filtros

Como pode-se observar pelas imagens acima é possível identificar que o trabalho de *antialising* realizado pelo Box Filter implementado fica visualmente mais suavizado, pode-se notar que ambas as imagens tiveram um efeito de borramento no entanto é possível perceber um certo serrilhamento nas bordas da imagem 2b.

Para a segunda análise utilizou-se uma imagem 500x500, ou seja um tamanho de altura e largura que não é uma potencia de dois. A Figura 3 corresponde a figura original e as Figura 4a e 4b correspondem as imagens após a aplicação dos filtros. Novamente para a redução da imagem foi utilizada uma proporção de tamanho 2.

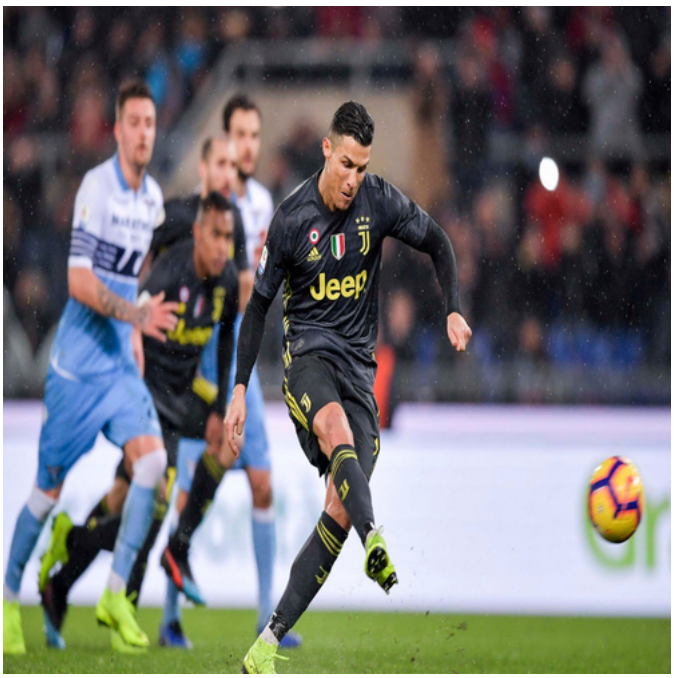

Figura 3: Amostra 500x500 original

(a) Imagem cobrança de pênalti com Box filter.

(b) Imagem cobrança de pênalti com Downsamplig simples.

Figura 4: Amostra 500x500 após aplicação dos filtros

Como pode-se ver novamente o Box Filter teve um desempenho maior, porém é notável que para uma imagem maior as diferenças ficam menos perceptíveis. Ambas imagens tiveram o efeito de borramento, porém para imagem com *down-samplig* simples as bordas do uniforme do jogador evidenciam um serrilhamento.

A Imagem 5, originalmente tem tamanho 512x512 foi reduzida na proporção 4 gerando as imagens 6a e 6b e contempla a imagem de um gato. Vale ressaltar que novamente a o tamanho da imagem testado é uma potência de 2.

Figura 5: Amostra 500x500 original

(a) Imagem gato com Box filter.

(b) Imagem gato com Downsamplig simples.

Figura 6: Amostra 512x512 após aplicação dos filtros

Para essa imagem o efeito de borramento foi semelhante nas imagens, entretanto novamente o Box Filter teve um desempenho melhor se tratando da suavização das bordas, na imagem com *downsamplig* simples é possível perceber, principalmente no lado esquerdo da face do gato que há alguns indícios de serrilhamento na imagem. Um ponto que fica em destaque nessa imagem é o de que ao realizar a aplicação dos filtros perde-se alguns detalhes na imagem. Isso fica evidenciado pelo bigode do felino que praticamente some nas imagens filtradas, principalmente na que faz aplicação do Box Filter.

A última imagem que foi analisada contempla uma paisagem e tem tamanho de 1024x1024, essa imagem diferentemente das demais foi reduzida com uma proporção 8 gerando as imagens 8 e 9.

Figura 7: Amostra 1024x1024 original

Figura 8: Paisagem filtrada com Box Filter na proporção 8x8

Figura 9: Paisagem filtrada com Downsamplig simples na proporção 8x8

Na última figura a diferença entre a aplicação do Box Filter e do *downsamplig* simples fica mais notável. O efeito de borramento foi conseguido em ambas imagens, entretanto observando a água, o céu e os barcos é possível visualizar com facilidade que o *downsamplig* simples gerou uma maior serrilhamento das bordas, quando comparado ao Box Filter. Ao analisar-se, principalmente as nuvens no Box Filter fica evidente o borramento em relação a imagem original, outro aspecto que chama a atenção é o de que os barcos perdem um grande nível de detalhamento, uma vez que a taxa de diminuição sofreu um aumento.

5 Conclusão

Neste trabalho fora abordado a implementação de um filtro passa-baixa conhecido como Box Filter, caracterizado pela suavização das bordas e outros componentes de imagem de alta frequência. Foi realizada também a comparação do filtro implementado com o *Downsamplig* simples realizado por meio do fatiamento de lista implementado por padrão na linguagem Python.

Ao analisar-se as imagens geradas pelos filtros pôde-se notar que o efeito de borramento fora conseguido em ambos

algoritmos. Outro ponto que merce destaque é o aumento da perda de detalhes a medida que a taxa de redução aumenta, essa perda era esperada para ambas filtragens, e ficou mais evidenciada nas imagens 8 e 9.As imagens geradas pelo filtro Box Filter, de maneira geral, conseguiram um melhor resultado, ou seja a suavização dos componentes de alta frequência ficaram mais evidentes na imagem de saída. Conforme citado para o *downsampling* simples também conseguiu-se também um efeito semelhante ao Box Filter, entretanto ficou nítido que em todas as imagens utilizadas como caso de teste, o *downsampling* simples não conseguiu um bom efeito de suavização das bordas gerando um serrilhamento nas bordas das 4 figuras, tornando a imagem de certa forma mais ”pixelada“.

Por fim pode-se concluir que os objetivos deste trabalho foram contemplados, uma vez que foi possível a implementação e comparação do filtro Box Filter com *downsampling* simples gerado por meio do fatiamento. Foi possível também a ampliação dos conhecimentos a cerca dos filtros passa baixa.

Referências

[1] R. C. Gonzalez and R. E. Woods, *Processamento de imagens digitais*. Editora Blucher, 2000.