Avaliação - Processamento de Imagens

RODRIGO FORMIGA FARIAS

1. Diferencie os conceitos de amostragem e quantização no processo de digitalização de uma imagem e a consequência desses processos na imagem digital.

Resposta: Em termos de imagem, a amostragem cria a matriz referente à imagem e a quantização define resolução de cor da imagem. Ou seja, a amostragem vai realizar uma diminuição da imagem, com número **reduzido de pixels**, causando um efeito de baixa qualidade ao ampliar a imagem, pois os pixels não são suficientes para definir bem a imagem. Já a quantização da imagem irá gerar versões com número **menores de cores utilizadas entre os pixels**, ou seja, as cores variam menos entre elas, reduzindo a área de transição de cores, reduzindo as bordas e reduzindo a quantidade de bytes utilizados para armazenamento da imagem. **A principal consequência da amostragem é o tamanho que essa imagem poderá ser exibida**, em relação a qualidade de visualização, já **a principal consequência da quantização é a baixa variedade de cores e diferenciação entre os pixels da imagem**.

---

2. No modelo de cor RGB, um tom de cinza é caracterizado por ter valores iguais para R, G e B (por exemplo, R = G = B = 120 corresponde a um tom de cinza). O que caracteriza um tom de cinza no modelo de cor CMYK? Prove.

Resposta: No CMYK, preto é o valor máximo de K (100), ou seja, ausência de qualquer cor. **Para obter um tom de cinza, não pode ter cor ciano (C), magenta (M) ou amarelo (Y), ou seja, devem ser zero.** Nessa situação, o ciano, magenta e amarelo não estão presentes, e o K determina a intensidade do tom de cinza desejado.
Por exemplo, se C = M = Y = 0 e K = 50, obtemos um tom de cinza com 50% de intensidade, enquanto C = M = Y = 0 e K = 20 um tom de cinza com 20% de intensidade. **A variação no valor de K permite controlar o brilho do tom de cinza no modelo CMYK.**

---

3. Que tipo de informação o histograma de uma imagem pode trazer quanto a uma previsão de possível resultado satisfatório ou não da aplicação de um algoritmo de binarização?

Resposta: Em imagens com grande diferenciação do fundo e da parte principal ou grande diferenciação entre objetos presentes (diferença em termos de cores), **é esperado que o histograma apresente uma distribuição da intensidade, variando desde o pixel mais claro até o pixel mais escuro**. É esperado também, para imagens com características de fundo bem distintas, que o histograma apresente picos em região mais clara e picos em região mais escura, sem perder um nível de equilíbrio entre as fases. Em histogramas com concentração de pixels na parte mais clara ou na parte mais escura, indica que o algoritmo de binarização não foi bem sucedido e precisa de ajustes.

---

4. Explique os aspectos computacionais a serem considerados na aplicação de uma convolução de um filtro em uma imagem no domínio espacial.

Resposta: Os aspectos principais na aplicação de uma convolução de um filtro são: **O tamanho do filtro** usado na convolução deve ser pré determinado, sendo utilizado valores para largura e altura, como 3x3 ou 5x5. O tamanho do filtro afetará diretamente a complexidade computacional do processo de convolução, pois filtros maiores implicam em maior tempo de processamento. Outro aspecto importante é: se a **imagem é colorida (RGB com valores diferentes)**, a convolução é realizada individualmente ou pode ocorrer em todos os canais simultaneamente, isso gera um número maior de operações necessárias e a estrutura dos filtros aplicados.

---

5. Suponha que um filtro Box 3x3 (matriz no slide 111 da aula de filtragem) é usado para processar uma imagem. Em seguida, um filtro laplaciano 3x3 (slide 135 da aula de filtragem) é usado na imagem processada pelo filtro Box. Ou seja, temos:

I_Final = Laplaciano(Box(I_Original))

Calcule como deve ser uma máscara única que faria o mesmo que os filtros Laplaciano e Box. Apresente todos os cálculos.

Resposta:

O filtro box 3x3 é assim:

1 1 1
1 1 1
1 1 1

O filtro Laplaciano 3x3 é assim:

0 1 0
1 -4 1
0 1 0

Ao aplicar a união dos filtros:

1 1 1	
1 1 1	
1 1 1*	0 1 0
	1 -4 1
	0 1 0

Exemplo: 0 * 1 + 1 * 1 + 1 * 1 + (-4 * 1) = - 2

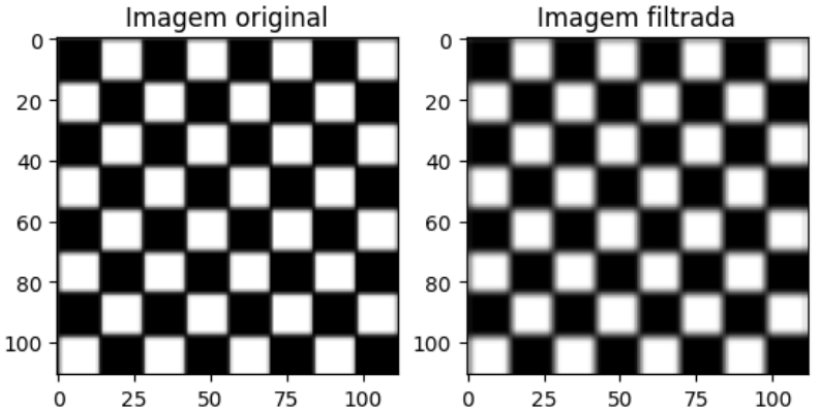
Resultado:

0 1 1 1 0
1 -2 -1 -2 1
1 -1 0 -1 1
1 -2 -1 -2 1
0 1 1 1 0

6. Qual o resultado esperado de uma operação que calcula a diferença absoluta entre uma imagem (em tons de cinza com tons entre 0 e 255) e essa mesma imagem filtrada por um filtro passa-baixa (um filtro Box, por exemplo)? Que tipo de filtro alcançaria esse mesmo resultado? Explique em detalhes. Ou seja, I1 = I – h*I, onde I é a imagem original, h o filtro passa-baixa (como o Box) e * a operação de convolução.

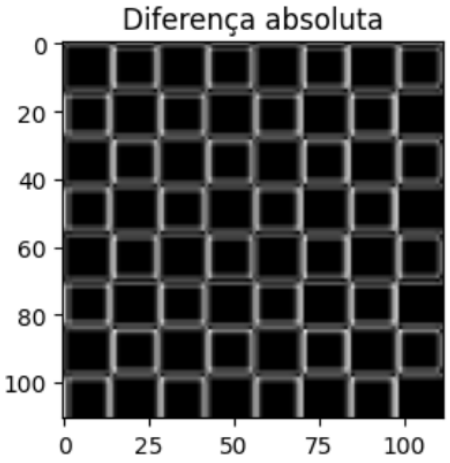
Resposta: Para tanto foi simulado o tipo de resultado para ilustrar:

```
# Aplicar o filtro Box
kernel_size = (3, 3)
f = cv2.boxFilter(image, -1, kernel_size)
# Calcular a diferença absoluta
diff = cv2.absdiff(image, f)
```

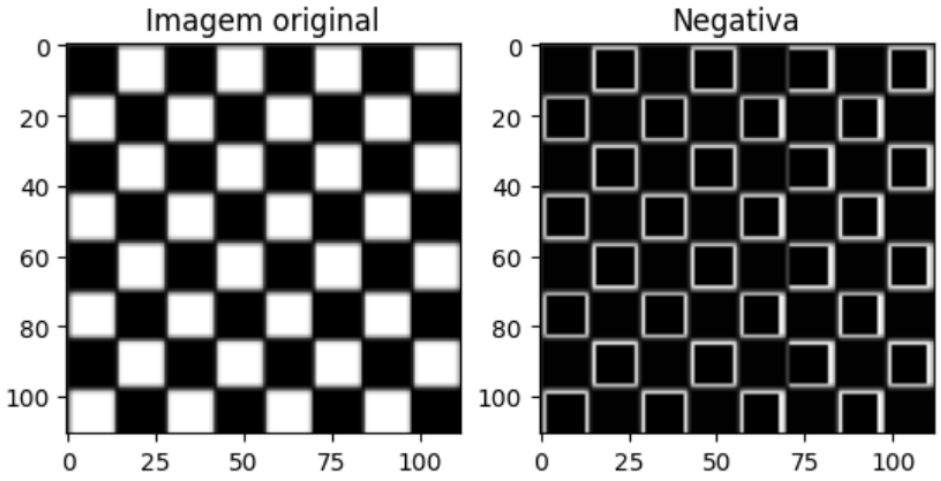


Após a diferença absoluta entre essas imagens, ou seja, quando os pixels são subtraídos, ou seja, pixels com tom 255 (branco) são subtraídos de pixels em escala de cinza (devido ao filtro, que gera pixels próximos do branco e do preto, ou seja, cinza.

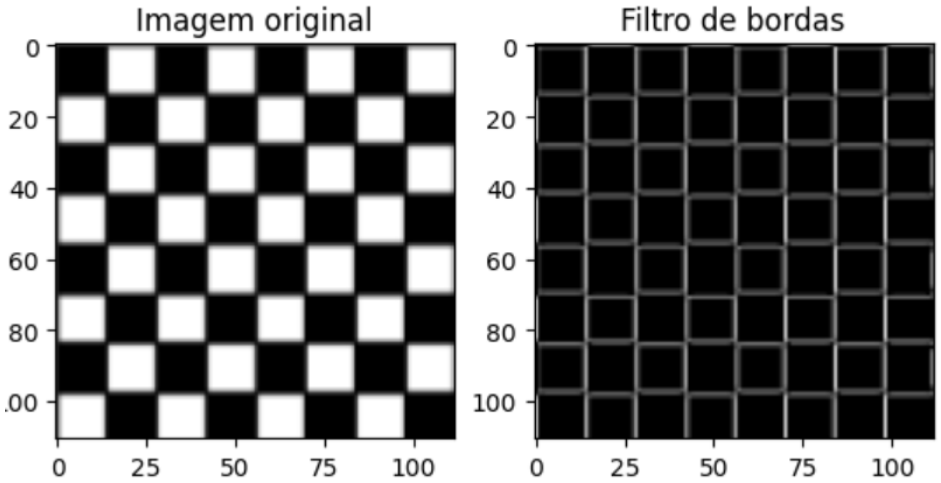
O resultado é:



Um resultado semelhante pode ser obtido através da negativa da imagem, ou seja, “**image - 255”.**

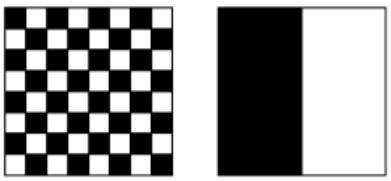


Uma **segunda opção** é o filtro de alta frequência ou filtro de bordas. Esse filtro realça as transições de intensidade na imagem, destacando as bordas. Esse filtro realiza uma operação muito próxima da diferença absoluta.



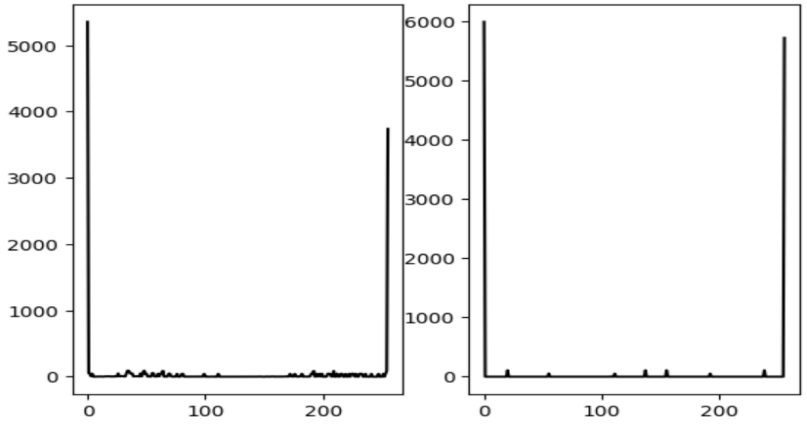
---

7. As imagens abaixo (ambas em preto e branco apenas) são bastante diferentes, mas seus histogramas são absolutamente iguais. Suponha que cada imagem é embaçada com um filtro da média 3x3. Os histogramas das imagens embaçadas ainda serão iguais? Justifique e esboce como devem ser o(s) histograma(s). Obs: As bordas não fazem parte das figuras; são apenas para melhor visualização.

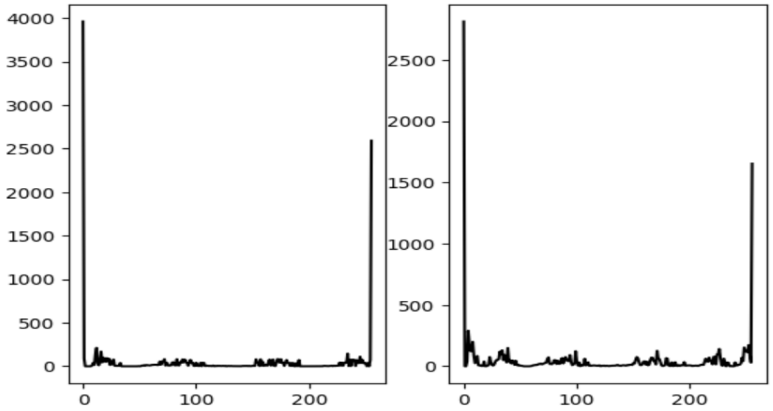


Resposta: Os histogramas não serão idênticos, porém **terão um comportamento semelhante**, ou seja, mesmo após a aplicação do filtro, os histogramas continuaram apresentando 2 picos bem distintos e sem suavização entre eles.

Antes do filtro:



Depois do filtro:



Ou seja, apesar da imagem metade branco, metade preto apresentar uma maior quantidade de cinza, ou seja, valores de transição, os histogramas são semelhantes, ainda após aplicação do filtro.

---

8. Um filtro com muitas aplicações é o de Diferença de Gaussianas. Sejam G1(1) e G2(2) duas funções Gaussianas diferentes, o filtro DoG (Difference of Gaussians) é dado por:

DoG = G1(1) - G2(2)

O filtro Gaussiano sozinho tem a função de um filtro passa baixa e, como tal, embaça uma imagem. Nessa combinação de filtros Gaussianos diferentes, como você imagina que será a aplicação do filtro DoG em uma imagem?

---

9. Explique porque as operações de erosão e dilatação (com um mesmo elemento estruturante) não são necessariamente operações inversíveis.

Resposta: A erosão e a dilatação podem resultar na perda de informações da imagem original. Porém são operações opostas, ocorrendo, durante a erosão, redução e, consequentemente, perda de detalhes, o oposto ocorre na dilatação, onde os objetos são

expandidos, resultando em uma perda de informações sobre a forma original dos objetos. Contudo, essas operações, se aplicadas em sequência, poderão gerar uma imagem diferente da original, ou seja, é improvável que a imagem original possa ser recuperada com precisão.

As operações de erosão e dilatação são feitas a partir do elemento estruturante (matriz semelhante ao filtro), que define a vizinhança considerada durante as operações. Se o elemento estruturante não for simétrico ou não preservar informações suficientes sobre a forma original dos objetos, a inversão das operações não será possível, pois a informação perdida não poderá ser recuperada.

Atividade Prática

Considere a imagem Flor_Joaninha.jpg. Essa imagem apresenta um efeito chamado de Low-Depth of Field (baixo campo de profundidade), onde um objeto em foco aparece em primeiro plano com o fundo da imagem desfocado. Tente segmentar essa imagem automaticamente de forma a preservar apenas os objetos em primeiro plano.

Resposta: Para essa atividade, **foi anexado um script em python,** que rodei no google colab, com caminho do arquivo especificado para meu workspace.
Para essa atividade, foi necessário realizar uma segmentação da seguinte imagem:

Onde existem alguns objetos em primeiro plano, como a flor e a joaninha, e outros elementos em segundo plano, sendo o objetivo principal segmentar tais objetos, destacando apenas aqueles em primeiro plano.

Para tanto, em primeiro lugar foi carregada a imagem e, então, convertida para escala de cinza, facilitando assim o processo de segmentação das bordas.

Método “imread” para ler a imagem de um caminho determinado.

```
image = cv2.imread(path)
```

Método de converter cor para realizar um filtro na imagem, transformando de BGR para estaca de cinza.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Resultando em uma imagem em escala de cinza, como apresentado:

Em seguida foi utilizado o método de linearização Canny para realizar uma detecção de bordas, resultando na imagem seguinte:

Aplicar o método de Canny para segmentar a imagem. Os valores 50 e 150 foram escolhidos através de testes. 50 e 150 são os valores utilizados no algoritmo para decidir o que é borda ou não. Qualquer diferença de intensidade superior a 100 é uma borda forte, já qualquer valor abaixo de 50 é descartado.

linhas = cv2.Canny(gray, 50, 150)

Em seguida, são utilizados métodos de criação de um kernel, ou seja, uma matriz para realização de filtro de fechamento, buscando conectar as linhas dos contornos e aproximar de uma forma conhecida, ou seja, a forma dos elementos em primeiro plano.

Método para criação de um elemento estruturante, de tamanho 7 por 7. Podendo ser construído através de listas, mas preferi utilizar um método disponível. Os valores do tamanho do elemento foi testado, pois valores muito pequenos resultaram em baixo fechamento e valores altos geraram máscara que englobava mas que o objeto de interesse.

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7, 7))

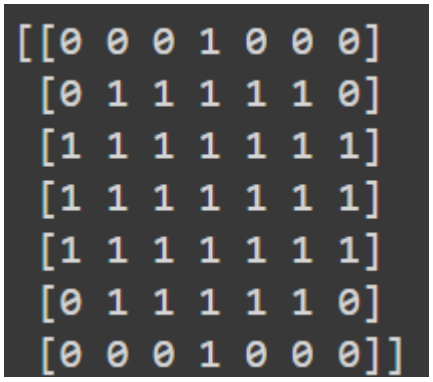
Opção 2
element = np.array([[1,1,1,1,1],

```
#               [1,1,1,1,1],
#               [1,1,1,1,1],
#               [1,1,1,1,1],
#               [1,1,1,1,1]],dtype=np.uint8)
```

Utilizamos esse elemento para realizar uma operação de fechamento morfológico aplicado na imagem dos contornos. O fechamento morfológico é uma operação que consiste em realizar a **erosão** seguida da **dilatação** da imagem. A **erosão** é uma operação que desgasta as bordas dos objetos e remove pequenos detalhes, enquanto a **dilatação** expande as áreas brancas e preenche lacunas nas bordas dos objetos. Resultando na imagem seguinte (da direita).

Método para realizar um fechamento morfológico, com o elemento criado na linha anterior.

```
fechar = cv2.morphologyEx(linhas , cv2.MORPH_CLOSE, kernel)
```



Com os contornos ligados entre si, conseguimos visualizar os limites da imagem, portanto, é possível aplicar métodos de detecção de contornos externos para isolar totalmente os objetos de primeiro plano.

Método para encontrar contornos com argumento “RETR_EXTERNAL” com objetivo de retornar apenas os contornos externos da imagem, ou seja, desconsiderar contornos internos. O parâmetro “cv2.CHAIN_APPROX_SIMPLE” foi utilizado apenas para economizar espaço de memória e armazenar apenas os pontos extremos dos contornos.

```
contours, _ = cv2.findContours(fechar,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

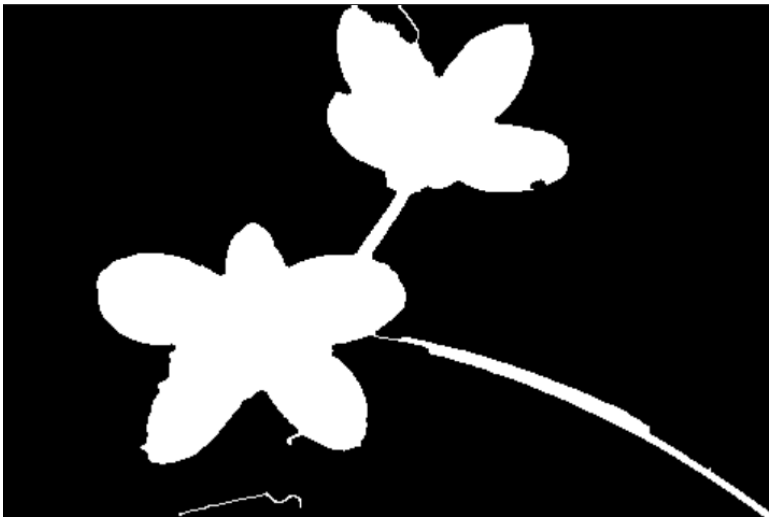
Após retornar os contornos externos, foi aplicado um método para criação da máscara, utilizando os contornos gerados.

“np.zeros_like(image)” cria uma matriz de zeros com as mesmas dimensões da imagem original. Essa matriz será usada como base para a criação da máscara. Também é passado

os contornos obtidos. O parâmetro “-1” indica que todos os contornos presentes na lista serão desenhados, a array “(255,255,255)” indica que serão considerados os contornos brancos, e o “thickness=cv2.FILLED” irá realizar um preenchimento da máscara.

mask = cv2.drawContours(np.zeros_like(image), contours, -1, (255, 255, 255), thickness=cv2.FILLED)

Resultado da aplicação da máscara:



Por fim, é aplicado a máscara sobre a imagem original, deixando “exposto” apenas os objetos em primeiro plano.

combina os pixels da imagem de entrada com os pixels correspondentes da máscara, usando a operação AND bit a bit.

resultado = cv2.bitwise_and(image, mask)

RESULTADO FINAL: