Pontifícia Universidade Católica de Goiás
Escola de Ciências Exatas e da Computação
Curso de Engenharia de Computação

LUCAS MACEDO DA SILVA

LISTA 3 – FILTRAGEM ESPACIAL

Goiânia
2020

Considerações iniciais

Para o desenvolvimento do presente trabalho foi utilizada a seguinte imagem.

Figura 1 – Imagem utilizada

Fonte: Google imagens (2020)

Ela é uma imagem de 450 x 600 pixels e foi utilizada em todas as questões.

Questão 1

Conceitue correlação e convolução.

Convolução

Segundo Gonzalez e Woods (2010), a convolução é um processo em que se move uma máscara (kernel) por uma imagem e calcula-se a soma dos produtos em cada posição, esse

valor é atribuído ao valor do pixel na imagem e o processo é repetido por todos os pixels da imagem. A convolução, então pode ser usada como uma forma de modificar a imagem original e gerar uma nova imagem.

A operação de convolução é definida como:

$$w(x,y)\circledast f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

Onde,

$w(x,y)$ é a máscara, também conhecida como filtro, que percorre a imagem;

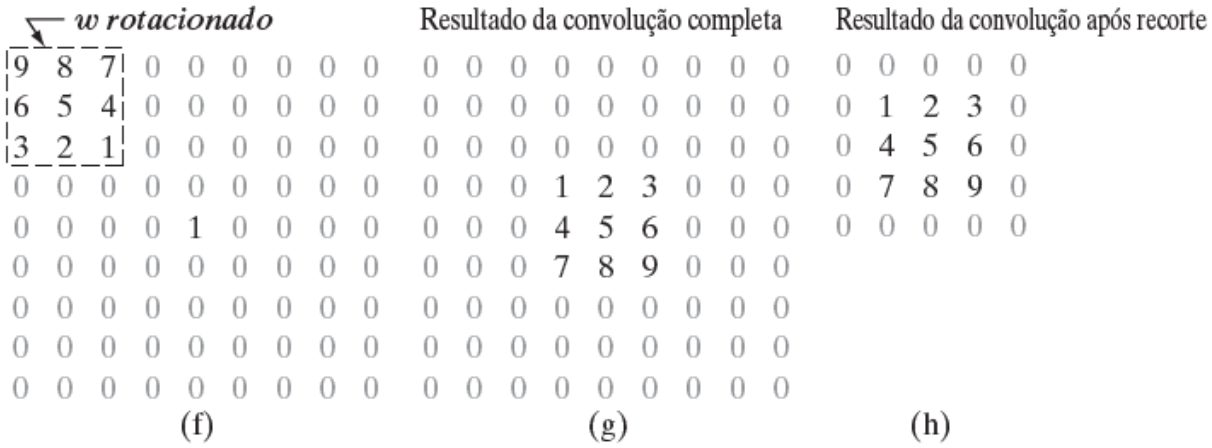
$f(x,y)$ é a imagem;

$a = \frac{m-1}{2}$, onde m corresponde a dimensão do filtro, e é um número inteiro e ímpar;

$b = \frac{n-1}{2}$, onde n corresponde a dimensão do filtro, é um número inteiro e ímpar;

Na operação de convolução costuma-se rotacionar o kernel em 180º. A Figura 2, a seguir, mostra a operação de convolução entre uma imagem e o kernel w. A imagem é uma função de impulso unitário, possui apenas um 1 no centro e os demais valores são 0.

Figura 2 – Exemplo de aplicação da função de convolução



Fonte: Adaptado de Gonzalez e Woods (2010)

Para realizar a operação de convolução, primeiro a imagem foi preenchida com zeros, para realização correta da operação (Figura 2f), a máscara foi rotacionada em 180º e foi movida (Figura 2f) pela imagem, for fim o resultado foi obtido (Figura 2g) e então os zeros colocados a mais foram retirados (Figura 2h).

Uma propriedade importante de uma convolução é que realizar a convolução de uma função com impulso unitário gera uma cópia da função na posição do impulso (na posição que se encontra o valor 1). Conforme, pode ser visto na Figura 2h.

Correlação

Segundo Gonzalez e Woods (2010), a correlação é um processo em que se move uma máscara (kernel) por uma imagem e calculando a soma dos produtos em cada posição. A correlação diferentemente da convolução que rotaciona a máscara, mantém a máscara sem rotacionar.

Ela é uma operação em que o filtro é deslocado, dessa forma o primeiro valor da correlação corresponde ao deslocamento zero do filtro, o segundo corresponde ao segundo e assim por diante. Além disso, realizar a operação de correlação entre uma função e um filtro que contém apenas zeros e um único número um resulta na cópia da função rotacionada em 180º, conforme pode ser visto na Figura 3e.

A operação de correlação é definida como:

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Onde,

$w(x,y)$ é a máscara, também conhecida como filtro, que percorre a imagem;

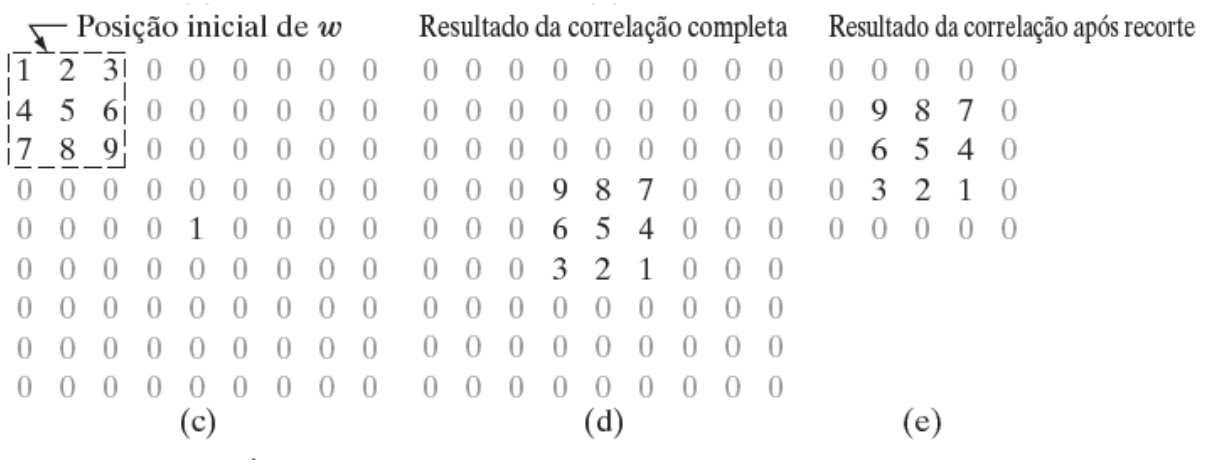
$f(x,y)$ é a imagem;

$a = \frac{m-1}{2}$, onde m é um número inteiro e ímpar;

$b = \frac{n-1}{2}$, onde m é um número inteiro e ímpar;

A Figura 3 a seguir mostra a operação de convolução entre uma imagem e o kernel w. A imagem é uma função de impulso unitário, possui apenas um 1 no centro e os demais valores são 0.

Figura 3 – Exemplo de aplicação da operação de correlação



Fonte: Adaptado de Gonzalez e Woods (2010)

Para realizar a operação de convolução, primeiro a imagem foi preenchida com zeros, para realização correta da operação (Figura 3c), a máscara foi movida (Figura 3c) pela imagem, for fim o resultado foi obtido (Figura 3d) e então os zeros colocados a mais foram retirados (Figura 3e).

Por se tratar de uma função impulso o resultado obtido foi a máscara rotacionada em 180 º, como era de se esperar.

Implementação da convolução e correlação

As operações de convolução e correlação foram implementadas em Matlab, conforme o código a seguir. Foi utilizada o mesmo código para realizar tanto a operação de correlação quanto a operação de convolução. Segundo Gonzalez e Woods (2010), o funcionamento da convolução é o mesmo que o da correlação exceto pelo fato de que a máscara é rotacionada em 180º na operação de convolução.

Para diferenciar entre as duas operações, a função desenvolvida, denominada de corr_conv, recebe como parâmetro uma *string*, que define o tipo de operação a ser realizada, se o valor de tipo for 'conv' a máscara é rotacionada com a função rot90 do Matlab, caso contrário a máscara não é rotacionada e a operação de correlação é realizada. Os demais parâmetros da função são img e w se referem a imagem a ser aplicada a operação e a máscara, respectivamente.

A primeira etapa do processamento consiste em criar uma imagem em que as m-1 e n-1 primeiras linhas são compostas de zeros, m e n se referem as dimensões da máscara, as demais posições são preenchidas pelos pixels correspondentes da imagem original. A função preenche_zeros realiza essa operação.

No próximo passo a imagem de saída é criada, ela tem um tamanho correspondente a a + m -1 e b + n – 1, a e b se referem as dimensões da imagem de entrada. Por fim, a cada pixel da imagem de entrada um novo valor é calculado utilizando o mesmo processo descrito na Figura 2. Isto é, para realizar a operação, a função cálculo recebe como parâmetros a imagem (parâmetro f) e a máscara (parâmetro w), ambos são matrizes do mesmo tamanho, do tamanho da máscara. Com isso, as duas matrizes são multiplicadas resultando no novo valor para o pixel.

Código 1 – Função para cálculo da convolução e correlação

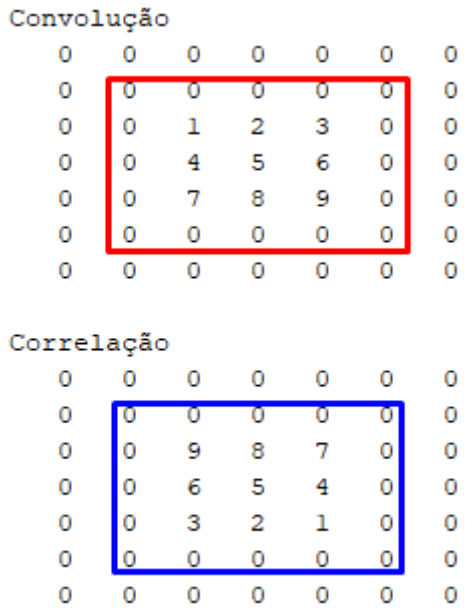
```
function imagem = corr_conv(img, w, tipo)
   % img = imagem de entrada
   % w = mascara
   % tipo = operacao a ser realizada 'corr' = correlação, 'conv' = convolução

   % Rotaciona w em 180º
   if (tipo == 'conv')
      w = rot90(w, 2);
   end
   [a b k] = size(img);
   [m n] = size(w);
   img_zeros = preenche_zeros(img, m-1, n-1);
   imagem = zeros(a + m -1, b + n - 1, k, 'uint8');
   for i = 1 : a + m - 1
      for j = 1: b + n - 1
         imagem(i, j, :) = calculo(img_zeros(i: i + m - 1, j:j + n - 1, :), w);
      end
   end
end
```

```
function mat = preenche_zeros(f, m, n)
   [x y k] = size(f);
   mat = zeros(x + 2 * m, y + 2 * n, k);

   for i = 1: x
      for j = 1: y
         mat(i + m , j + n, :) = f(i, j, :);
      end
   end
end

function pixel = calculo (f, w)
   [a b k] = size(f);
   pixel = 0;
   for i = 1: a
      for j = 1: b
         pixel = pixel + f(i, j, :) * w(i, j);
      end
   end
end
```

Fonte: Autoria Própria (2020)

A fim de testar a veracidade do código foi utilizada a matriz e a máscara das Figuras 2 e 3. O resultado obtido está descrito na Figura 4 a seguir. O resultado obtido foi o mesmo que o das Figuras 2 e 3, porém no presente trabalho o resultado contém zeros envolta da matriz original, isso ocorre devido ao preenchimento dos zeros para facilitar os cálculos. Os zeros poderiam ser removidos, porém foi optado por deixá-los. Os quadrados em vermelho são as matrizes resultantes sem os zeros. O primeiro quadrado vermelho refere-se à operação de convolução e o quadrado azul se refere a operação de correlação.

Figura 4 – Teste com a matriz e máscara das Figuras 2 e 3



Fonte: Autoria Própria (2020)

A fim de validar a operação foi realizada a operação utilizando a função conv2 do Matlab. O resultado obtido foi o mesmo que o obtido na função implementada mostrando assim a veracidade.

Figura 5 – Teste com a matriz e máscara das Figuras 2 e 3 utilizando a função do Matlab

```
>> conv2(img, mask)

ans =

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     1     2     3     0     0
     0     0     4     5     6     0     0
     0     0     7     8     9     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

Fonte: Autoria Própria (2020)

Gonzalez e Woods (2010) definem a operação de filtragem espacial como sendo a operação de deslocar um filtro (matriz) de dimensão mxn sobre uma matriz de dimensão MxN. Essa operação pode ser descrita pela seguinte expressão:

$$f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Percebe-se que a expressão direita é a mesma utilizada pela operação de correlação. Sendo assim, para a resolução da maioria das questões, foi utilizado o Código 1 desenvolvido. A operação utilizada foi de correlação pois sua expressão é a mesma que a descrita acima. Em todas as questões também foi utilizada a função imfilter do Matlab para verificação da operação. Em questões onde a máscara era muito grande, não foi utilizado o Código 1 “corr_conv” pois ele é lento. Isso ocorre já que ele não foi otimizado.

Questão 2

Conceitue a filtragem espacial *fuzzy* baseada em regras.

O termo *fuzzy* refere-se a alguma informação vaga/imprecisa (GONZALEZ; WOODS 2010). A filtragem espacial *fuzzy* baseia-se na teoria no uso dessas informações imprecisas para realizar a filtragem.

Um conjunto é uma coleção e objetos, e a teria de conjuntos diz respeito as ferramentas que lidam com operações entre conjuntos. Em um conjunto normal, um objeto pode ou não estar “dentro” daquele conjunto, ou seja, o objeto está ou não. Enquanto, em um conjunto *fuzzy* o objeto é pertinente em algum grau ao conjunto. Ou seja, no lugar de estar ou não, o objeto possui algum grau de pertinência ao conjunto. Essa ideia permite expressar as ideias imprecisas (GONZALEZ; WOODS 2010).

Para determinar o grau de pertinência existe uma função que descreve a pertinência do objeto no conjunto, tal função é denominada função de pertinência. Outro conceito importante

são as regras *fuzzy* que permitem expressar o conhecimento especifico do problema em na forma de SE-ENTÃO (IF-THEN) (GONZALEZ; WOODS 2010). Por conhecimento especifico compreende-se a forma de se expressar a informação imprecisa.

Para realizar a operação de filtragem espacial, segundo Gonzalez e Woods (2010) a metodologia é definir as propriedades da vizinhança que "capturem" a essência do que o filtro deve detectar. Ou seja, primeiro são definidas as regras *fuzzy* que capturam a essência do filtro (GONZALEZ; WOODS 2010).

Por exemplo pode-se utilizar os conjuntos *fuzzy* para realizar a extração de fronteira. O conhecimento especifico do problema são a mudança da cor que implica na detecção da fronteira. Ele pode ser expresso a partir da seguinte regra: Se um pixel pertencer a uma região uniforme, faça com que ele seja branco; senão, faça com que ele seja preto, onde preto e branco são conjuntos *fuzzy* (GONZALEZ; WOODS 2010). Essa regra pode ser descrita em regras *fuzzy* e então aplicadas na imagem para determinar as fronteiras.

Questão 3

Gere uma máscara para um filtro espacial usando o seguinte código:

Tam = 3;

Mascara = 1/(Tam*Tam) * ones(Tam,Tam);

Realize a filtragem espacial com a máscara acima e exiba a imagem de saída.

A máscara foi gerada e a imagem foi então filtrada a partir utilizando a função corr_conv desenvolvida para aplicar o filtro na imagem.

O seguinte trecho de código lê a imagem do disco e aplica a operação de filtragem com a máscara especificada.

Código 2 – Trecho de código para a resolução da questão 3

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 3\Imagens\img.jpg');
figure();
imshow(img);
title('Imagem Original');
Tam = 3;
Mascara = 1/(Tam*Tam) * ones(Tam,Tam);
img_filter = corr_conv(img, Mascara, 'corr');
figure();
imshow(img_filter);
title('Imagem Filtrada');
```

Fonte: Autoria Própria (2020)

A figura a seguir é a imagem de saída após aplicar a operação com a máscara. Foi utilizada a função corr_conv presente no Código 1.

Figura 6 – Imagem após aplicar a máscara da questão 3

Fonte: Autoria Própria (2020)

Em relação a imagem original a imagem filtrada está menos “desfocada”.

Questão 4

Altere o tamanho (Tam) para 10 e 100, use a máscara recém-gerada para filtrar a imagem original.

O tamanho da máscara foi alterado gerando uma nova máscara de tamanho 10 e 100 respectivamente. O seguinte trecho de código foi utilizado para ler a imagem do disco e aplicar as operações.

Código 3 – Trecho de código para a resolução da questão 4

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 3\Imagens\img.jpg');


figure();
imshow(img);
title('Imagem Original');

Tam = 10;
Mascara = 1/(Tam*Tam) * ones(Tam,Tam);

img_filter = corr_conv(img, Mascara, 'corr');
figure();
imshow(img_filter);
title(strcat('Imagem filtrada com Tam = ', num2str(Tam)));

img_filter1 = imfilter(img, Mascara);
figure();
imshow(img_filter1);
title('Imagem filtrada com imfilter');
```

Fonte: Autoria Própria (2020)

As imagens a seguir são o resultado ao aplicar a máscara na imagem original. Para o tamanho de máscara igual a 100, foi utilizada apenas a função imfilter do Matlab, pois devido a função, corr_conv, de cálculo aqui desenvolvida não estar otimizada e o tempo de resposta pode ser lento dependendo do tamanho da imagem e máscara.

Tamanho da máscara igual a 10

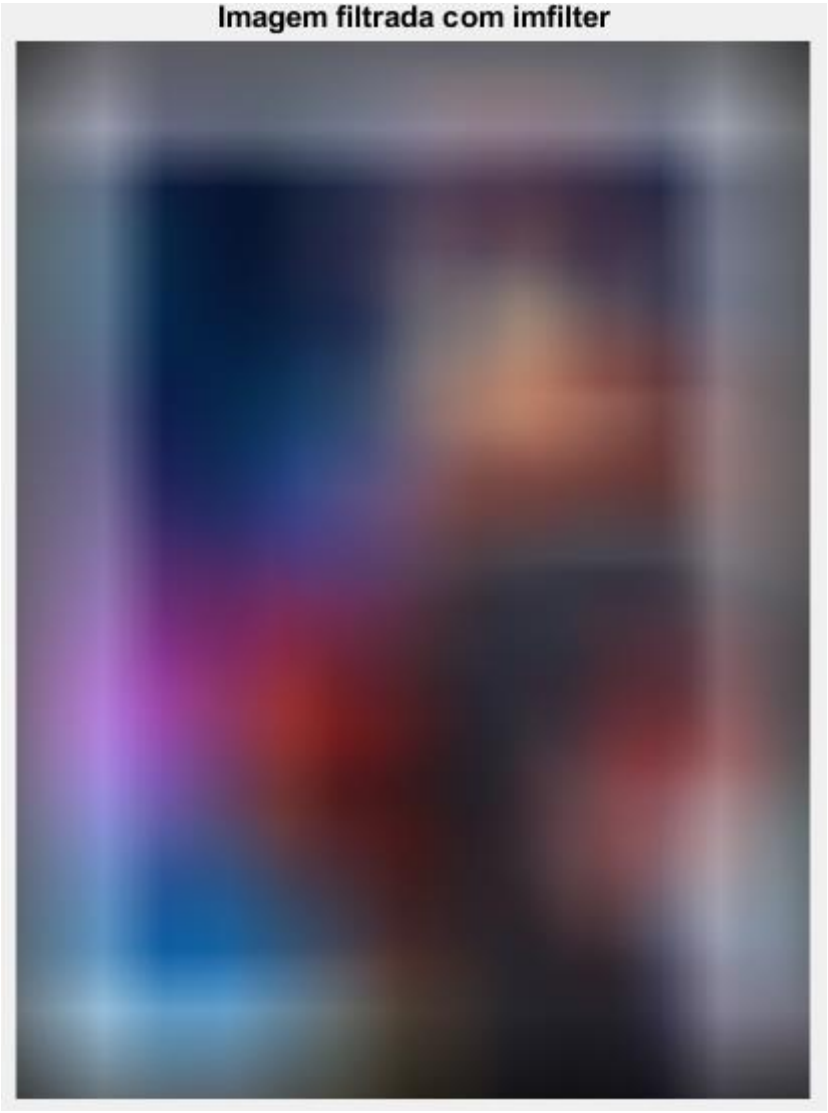
Figura 7 – Imagem filtrada com máscara de tamanho 10 utilizando a função corr_conv

Fonte: Autoria Própria (2020)

Figura 8 – Imagem filtrada com máscara de tamanho 10 utilizando a função imfilter do Matlab

Fonte: Autoria Própria (2020)

Figura 9 – Imagem filtrada com máscara de tamanho 100 utilizando a função imfilter do Matlab



Fonte: Autoria Própria (2020)

A partir das imagens geradas percebe-se que para a máscara ade 100 a imagem de saída está mais “embaçada”. Nela é mais difícil de ver o personagem que na imagem utilizando a máscara de tamanho igual a 10.

Questão 5

Realize a filtragem espacial com as máscaras (Laplace e Sobel) abaixo:

A filtragem espacial com as máscaras de Laplace e Sobel são um tipo de filtragem denominada filtragem de aguçamento. Os filtros de aguçamento têm por objetivo salientar transições de intensidade para o aumento da nitidez de uma imagem (GONZALEZ; WOODS 2010). As máscaras de Sobel e Laplace são baseadas em diferenciação (derivada) e isso permite a realce das bordas da imagem (GONZALEZ; WOODS 2010).

A filtragem das imagens foi realizada utilizando o seguinte trecho de código que lê a imagem do disco e então aplica a operação. Para cada máscara foi utilizada a função “corr_conv” desenvolvida e a função imfilter do Matlab para fins de comparação e validação dos resultados obtidos.

Código 4 – Trecho de código para a resolução da questão 5

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 3\Imagens\img.jpg');


laplace = [1 1 1; 1 -8 1; 1 1 1];
sobel = [-1 -2 -1; 0 0 0; 1 2 1];


img_laplace_imf = imfilter(img, laplace);
img_laplace_fun = corr_conv(img, laplace, 'corr');

img_sobel_imf = imfilter(img, sobel);
img_sobel_fun = corr_conv(img, sobel, 'corr');


figure();
imshow(img_laplace_imf);
title('Laplace: Imagem filtrada imfilter');
figure();
imshow(img_laplace_fun);
title('Laplace: Imagem filtrada função');

figure();
imshow(img_sobel_imf);
title('Sobel: Imagem filtrada imfilter');
figure();
imshow(img_sobel_fun);
title('Sobel: Imagem filtrada função');
```

Fonte: Autoria Própria (2020)

Máscara de Sobel

A máscara de Sobel utilizada para a filtragem foi a seguinte:

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

O resultado obtido foi o seguinte.

Figura 10 – Imagem filtrada com máscara de Sobel utilizando a função corr_conv

Fonte: Autoria Própria (2020)

Figura 11 – Imagem filtrada com máscara de Sobel utilizando a função imfilter do Matlab

Fonte: Autoria Própria (2020)

Máscara de Laplace

A máscara de Laplace utilizada para a filtragem foi a seguinte:

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

Os resultados obtidos estão descritos nas figuras a seguir.

Figura 12 – Imagem filtrada com máscara de Laplace utilizando a função corr_conv

Fonte: Autoria Própria (2020)

Figura 13 – Imagem filtrada com máscara de Laplace utilizando a função imfilter do Matlab

Fonte: Autoria Própria (2020)

Os resultados obtidos entre a função do Matlab e a função corr_conv são bastante similares. Além disso, é possível perceber que as bordas das imagens estão destacadas, o que é o objetivo das máscaras de Sobel e Laplapce.

Questão 6

Gere outra máscara de filtro usando o seguinte código:

Mascara2 = [0 0 0 0 0; 1 1 1 1 1; 0 0 0 0 0; -1 -1 -1 -1 -1; 0 0 0 0 0];

angulo = 0;

filtro = imrotate(mascara2, angulo, ‘crop’);

Após executar o trecho de código no software Matlab, obtém-se a seguinte matriz.

Figura 14 – Matriz de saída após executar o trecho de código no software Matlab

```
filtro =

     0     0     0     0     0
     1     1     1     1     1
     0     0     0     0     0
    -1    -1    -1    -1    -1
     0     0     0     0     0
```

Fonte: Autoria Própria (2020)

Essa matriz foi então utilizada como a máscara para realizar as operações de filtragem.

O seguinte trecho de código lê a imagem do disco e aplica a filtragem usando a função "corr_conv" e a função imfilter do Matlab. A função imfilter do Matlab foi utilizada para fins de comparação.

Código 5– Trecho de código para a resolução da questão 6

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 3\Imagens\img.jpg');

mascara2 = [0 0 0 0 0; 1 1 1 1 1; 0 0 0 0 0; -1 -1 -1 -1 -1; 0 0 0 0 0];
angulo = 0;
filtro = imrotate(mascara2, angulo, 'crop');

img_imfilter = imfilter(img, filtro);
img_fun = corr_conv(img, filtro, 'corr');

figure();
imshow(img_imfilter);
title('Imagem filtrada com imfilter');

figure();
imshow(img_fun);
title('Imagem filtrada com a função desenvolvida');
```

Fonte: Autoria Própria (2020)

Figura 15 – Imagem filtrada com a função corr_conv com máscara da questão 6

Fonte: Autoria Própria (2020)

Figura 16 – Imagem filtrada com a função imfilter do Matlab com máscara da questão 6

Fonte: Autoria Própria (2020)

Ao analisar as imagens é possível verificar que as bordas horizontais foram intensificadas. O fundo da imagem original, Figura 1, contém alguns pontos brancos, depois de realizar a operação de filtragem esses pontos ficaram na horizontal. Isso ocorre, pois, o filtro contém linhas com 0 e 1 alternadas.

Questão 7

Altere o ângulo para 90 e 196, exiba o resultado e explique a diferença entre diferentes saídas.

Para resolução deste problema foi utilizado o Código 5– Trecho de código para a resolução da questão 6, com a mudança do valor do ângulo.

As seguintes imagens são o resultado da aplicação da mudança do ângulo e depois a filtragem.

Ângulo igual a 90 º

Figura 17 – Imagem filtrada com a função corr_conv com máscara da questão 6 com ângulo de 90º

Fonte: Autoria Própria (2020)

Figura 18 – Imagem filtrada com a função imfilter do Matlab com máscara da questão 6 com ângulo de 90º

Fonte: Autoria Própria (2020)

Ângulo igual a 196 º

Figura 19 – Imagem filtrada com a função corr_conv com máscara da questão 6 com ângulo de 196º

Fonte: Autoria Própria (2020)

Figura 20 – Imagem filtrada com a função imfilter do Matlab com máscara da questão 6 com ângulo de 196º

Fonte: Autoria Própria (2020)

Explicação

A diferença entre as imagens está na forma em que as bordas da imagem foram destacadas. Para o ângulo de 90º as bordas ficaram distorcidas quase horizontalmente, o fundo por exemplo, na Figura 1 existem círculos brancos, já na Figura 17 que a máscara foi rotacionada em 90º os mesmos círculos brancos ficarão alongados e na vertical, já na figura 19 em que a máscara foi rotacionada em 196º eles ficaram quase na horizontal.

Além disso, os traços detectados nas duas imagens foram diferentes. Por exemplo, com a rotação de 196º os traços da roupa ficaram mais bem destacados, porém ficaram tremidos.

Na comparação das duas imagens (Figura 17 e 19) com a imagem gerada com rotação igual a 0º (Figura 15), percebe-se que para rotação em 0º os círculos no fundo branco ficaram na horizontal.

Desta forma, baseado nas três imagens retro mencionadas, empiricamente pode-se verificar que a saída tendeu a ter as bordas destacadas no ângulo em que a máscara foi

rotacionada. Para o ângulo de 0º a máscara não estava rotacionada e as bordas horizontais foram destacadas, para um ângulo de 90º a máscara foi rotacionada, conforme a Figura 21 a seguir, e as bordas verticais foram destacadas, já para 196º a máscara foi rotacionada e os uns ficaram mais espalhados pela matriz, Figura 22, assim, as bordas quase horizontais foram destacadas.

Figura 21 – Máscara rotacionada em 90º

```
angulo =

    90


filtro =

     0     1     0    -1     0
     0     1     0    -1     0
     0     1     0    -1     0
     0     1     0    -1     0
     0     1     0    -1     0
```

Fonte: Autoria Própria (2020)

Figura 22 – Máscara rotacionada em 196º

```
angulo =

   196


filtro =

     0     0     0     0    -1
     0    -1    -1    -1     0
    -1     0     0     0     1
     0     1     1     1     0
     1     0     0     0     0
```

Fonte: Autoria Própria (2020)

Questão 8

Utilize a máscara da questão 4, Tam igual a 10, na função imfilter, com os padding symmetric, replicate, circular. Explique a diferença de cada um deles.

Para a resolução desta questão foi utilizada uma versão reduzida e cortada da imagem original, a nova imagem possui dimensões de 200 x 278 pixels. Para ser possível verificar a alteração dos pixels.

Figura 23 – Imagem usada para a resolução da questão 8

Fonte: Autoria Própria (2020)

O seguinte trecho de código lê a imagem do disco e mostra os resultados obtidos.

Código 6 – Trecho de código para a resolução da questão 6

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 3\Imagens\img2.jpg');
Tam = 10;
w = 1/(Tam*Tam) * ones(Tam,Tam);

figure();
img_s = imfilter(img, w, 'symmetric');
imshow(img_s);
title('Imagem com padding symmetric');

figure();
img_r = imfilter(img, w, 'replicate');
imshow(img_r);
title('Imagem com padding replicate');

figure();
img_c = imfilter(img, w, 'circular');
imshow(img_c);
title('Imagem com padding circular');
```

Fonte: Autoria Própria (2020)

Figura 24 – Imagem filtrada com imfilter com padding symmetric

Fonte: Autoria Própria (2020)

Figura 25 – Imagem filtrada com imfilter com padding replicate

Fonte: Autoria Própria (2020)

Figura 26 – Imagem filtrada com imfilter com padding circular

Fonte: Autoria Própria (2020)

Nos parágrafos subsequentes a palavra borda refere-se aos “lados” da imagem, isto é, as partes superiores e inferiores da mesma.

Segundo Mathworks (2020), o *paddings* utilizados possuem as seguintes características:

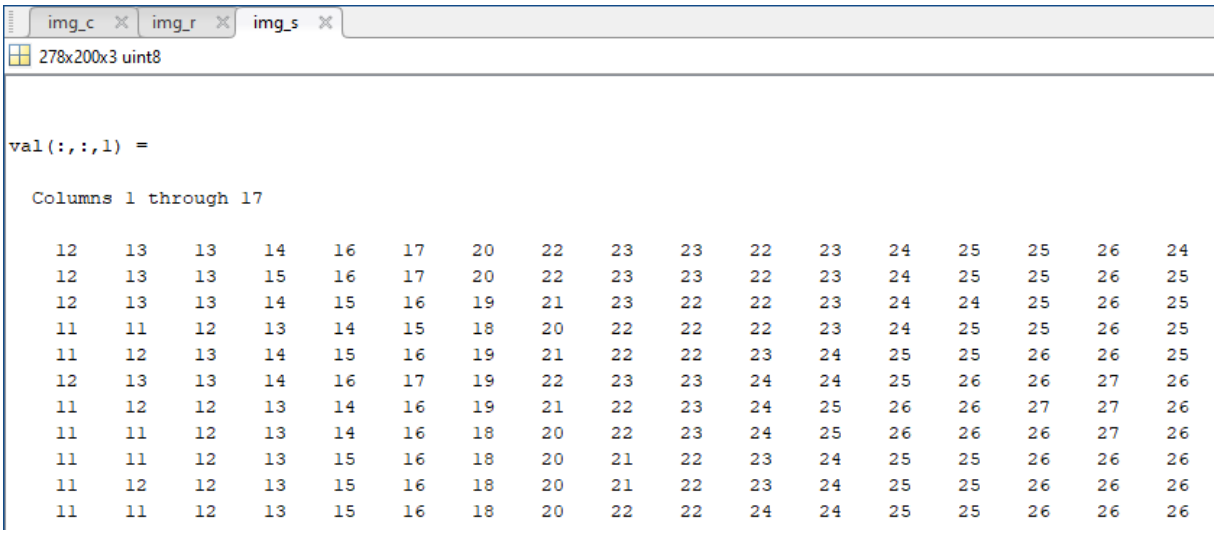
- Circular: O *padding* aplicado na imagem é realizado com a repetição circular dos elementos dentro da dimensão;
- *Replicate*: O *padding replicate* é feito repetindo os elementos da borda da imagem;
- *Symmetric*: O *padding symmetric* os pixels que compõem os *padding* são reflexos da matriz ao longo da borda.

A partir das Figuras 24, 25 e 26 não foi possível verificar alterações visualmente chamativas nas imagens. Na Figura 25, é possível perceber uma pequena linha preta na parte superior que corresponde a parte debaixo da image, já que o *padding* circular tende a tratar a imagem circularmente e os pixels são repetidos. Já nas demais imagens não foi possível verificar nenhuma diferença visualmente.

O fato por trás da semelhança entre as imagens é que os *padding* tendam pegar as bordas da imagem, e como as bordas da imagem são brancas não existe muita diferença. Porém ao verificar os valores dos armazenados nas imagens, percebe-se que eles estão muito

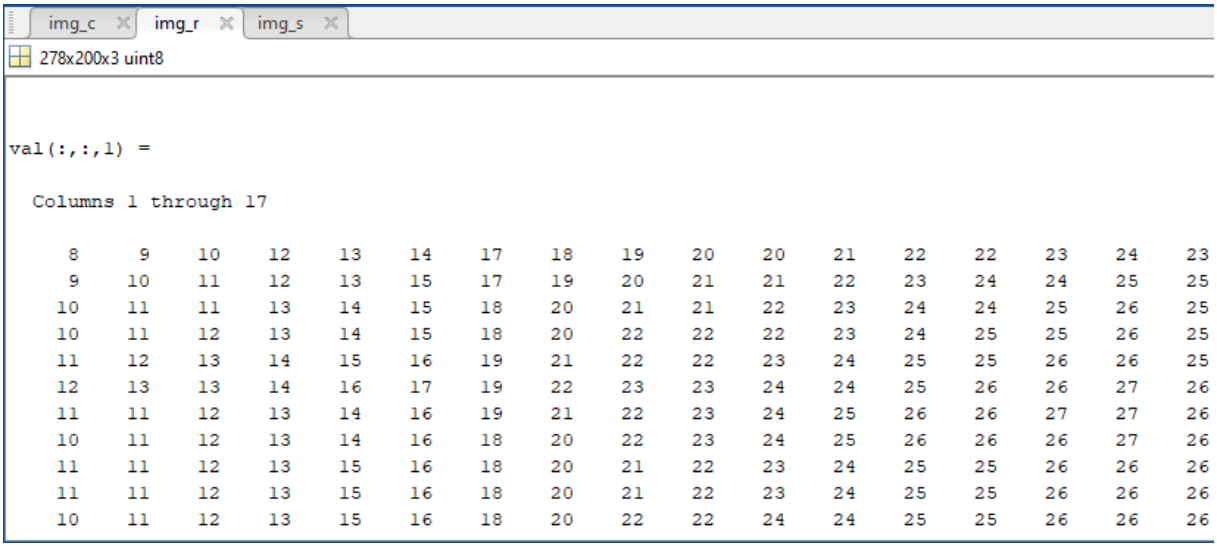
próximos, o que pode causar a impressão das imagens serem as mesmas. As figuras a seguir referem-se a esses pixels.

Figura 27 – Parte dos pixels presentes na imagem filtrada com o *padding symmetric*



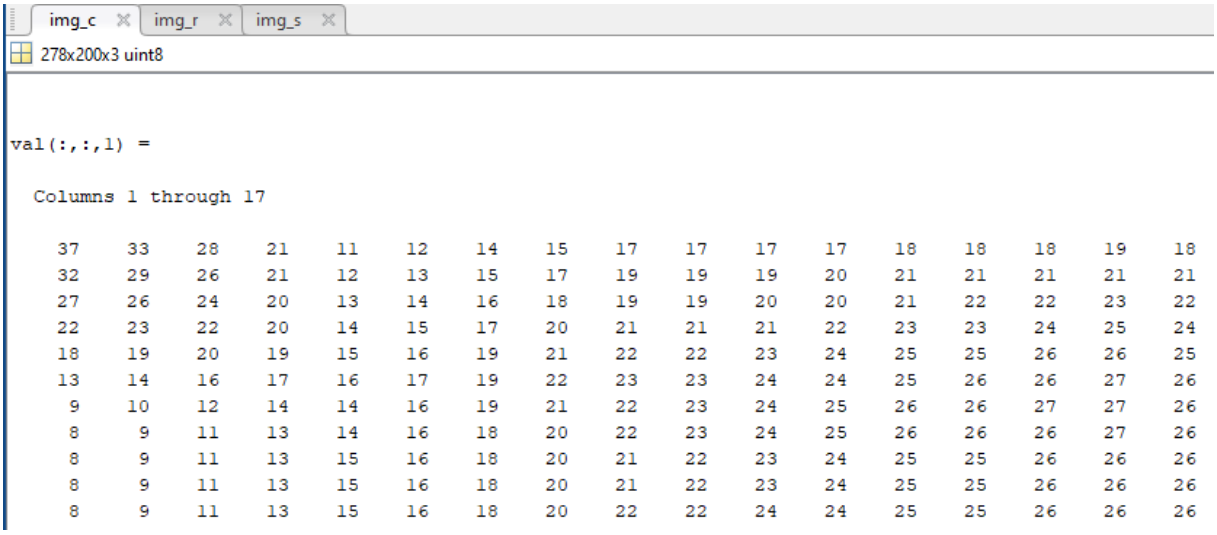
Fonte: Autoria Própria (2020)

Figura 28 – Parte dos pixels presentes na imagem filtrada com o *padding replicate*



Fonte: Autoria Própria (2020)

Figura 29 – Parte dos pixels presentes na imagem filtrada com o *padding circular*



Fonte: Autoria Própria (2020)

É possível ver que os valores dos pixels estão bem próximos, porém são diferentes, o que causa a impressão de que as imagens são iguais. Assim sendo, dependendo do *padding* aplicado a borda da imagem pode mudar e consequentemente o resultado da imagem também pode mudar, pois no meio dela pode haver algum valor de pixel fora do padrão que pode ser utilizado para compor o *padding* e assim distorcer o resultado.

Portanto, é importante verificar o tipo do *padding* antes de aplica-lo na imagem pois ele pode impactar no resultado. No presente trabalho, devido as bordas da imagens serem muito parecidas os valores dos pixels variaram pouco de uma imagem para outra gerando a ideia de que elas eram iguais.

Referências Bibliográficas

GONZALEZ, Rafael C.; WOOD, Richard E.. **Processamento Digital de Imagens**. 3. ed. São Paulo: Pearson Pratice Hall, 2010.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999. ISBN 8574520098.

MATHWORKS. **Padarray**. Disponível em: https://www.mathworks.com/help/images/ref/padarray.html. Acesso em: 29 mar. 2020.