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

LUCAS MACEDO DA SILVA

LISTA 1 – TRANSFORMAÇÕES GEOMÉTRICAS

Goiânia
2020

Considerações iniciais

Em todos os exercícios foi utilizada a Figura 1 a seguir. Ela é uma imagem de 500 pixels por 500 pixels.

Figura 1 – Imagem utilizada para a aplicação das transformações

Fonte: Autoria Própria (2020)

Vale ressaltar que essa imagem não foi utilizada na questão 5 que pedia para aumentar a escala. Devido ao fato de o tamanho da imagem dificultar a visualização da operação, foi utilizada uma versão de 200 pixels por 200 pixels da imagem.

O trecho de código a seguir foi utilizado para ler a imagem do disco para uso nas funções que implem as operações.

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 1\Imagens\img.jpg');

I = operacao (img);

imshow(I);
title(‘Imagem da Operação X’);
```

Nota: Ele foi utilizado em todas as questões. “oparacao” se refere a função que implementa a operação da questão. E “Imagem da Operação X” é o título da imagem.

Questão 1

Aplique a operação de espelhar a imagem horizontalmente e verticalmente.

Segundo Marques Filho e Vieira Neto (1999) a transformação de espelhamento ou *flip* combina múltiplas rotações em ângulos de 90º com o cálculo de matriz transposta. O espelhamento horizontal consiste na rotação de 90º no sentido anti-horário e o espelhamento vertical consiste na rotação em 90º no sentido horário.

Desta forma, a operação de espelhamento é semelhante ao ver o objeto através de um espelho, as direções aparecem opostas no espelho.

Para realizar a operação de espelhamento horizontal, as coordenadas dos pixels em cada imagem foram trocadas, isto é, o eixo x na imagem espelhada transformou-se no eixo y e o eixo y no eixo x. A função abaixo realiza a operação de espelhamento horizontal se encontra definida a seguir.

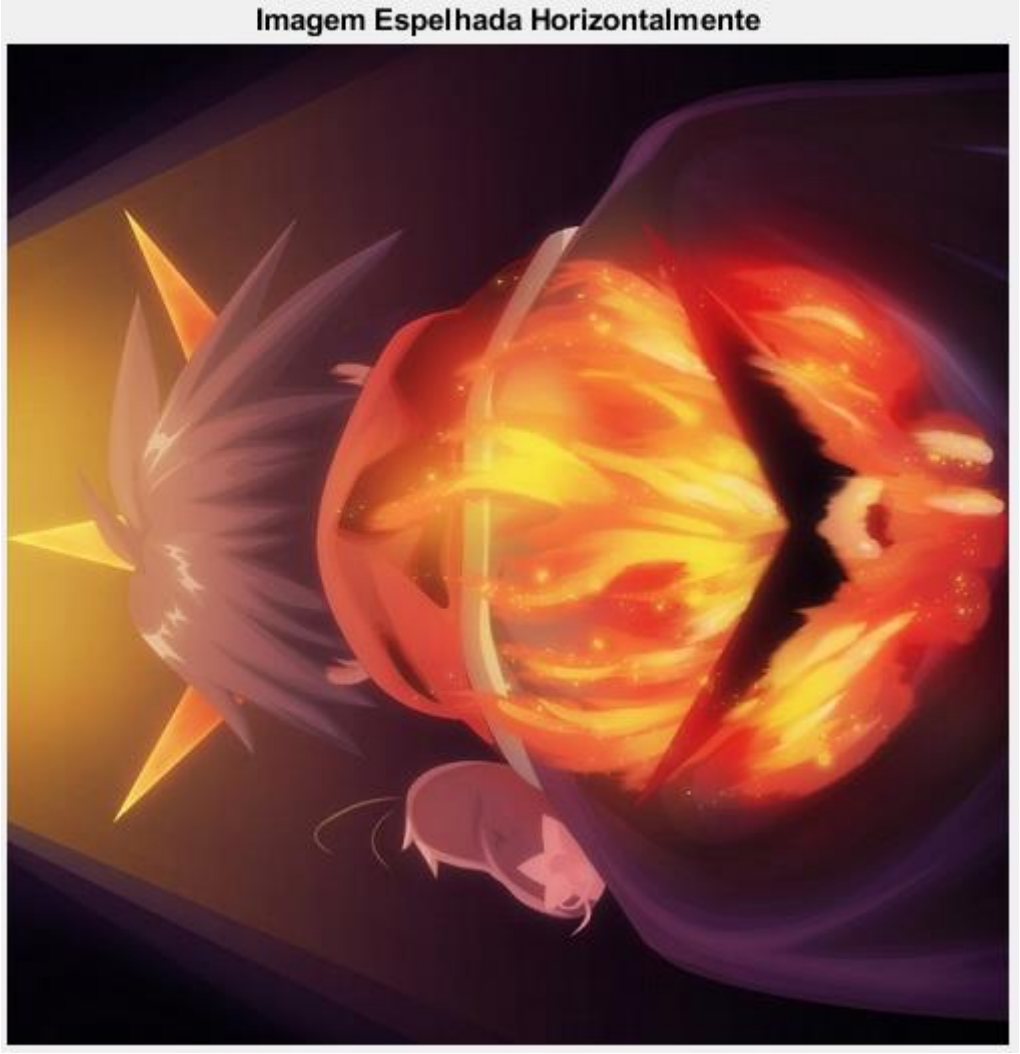
```
function imagem = espelhar_horiontal(img)
   [i j k] = size(img);
   imagem = zeros(j, i, k, 'uint8');

   for x = 1: i
      for y = 1: j
         imagem(y, x, :) = img(x, y, :);
      end
   end

end
```

Como resultado da transformação, foi obtida a seguinte imagem.

Figura 2 – Imagem espelhada horizontalmente



Fonte: Autoria Própria (2020)

Para realizar a operação de espelhamento vertical, a imagem foi “virada de cabeça para baixo”, isto é, os pixels que se encontravam na última linha, foram para a primeira linha, e os que estavam na última linha, foram para a primeira. Em resumo, a operação se baseou na troca de lugares dos pixels.

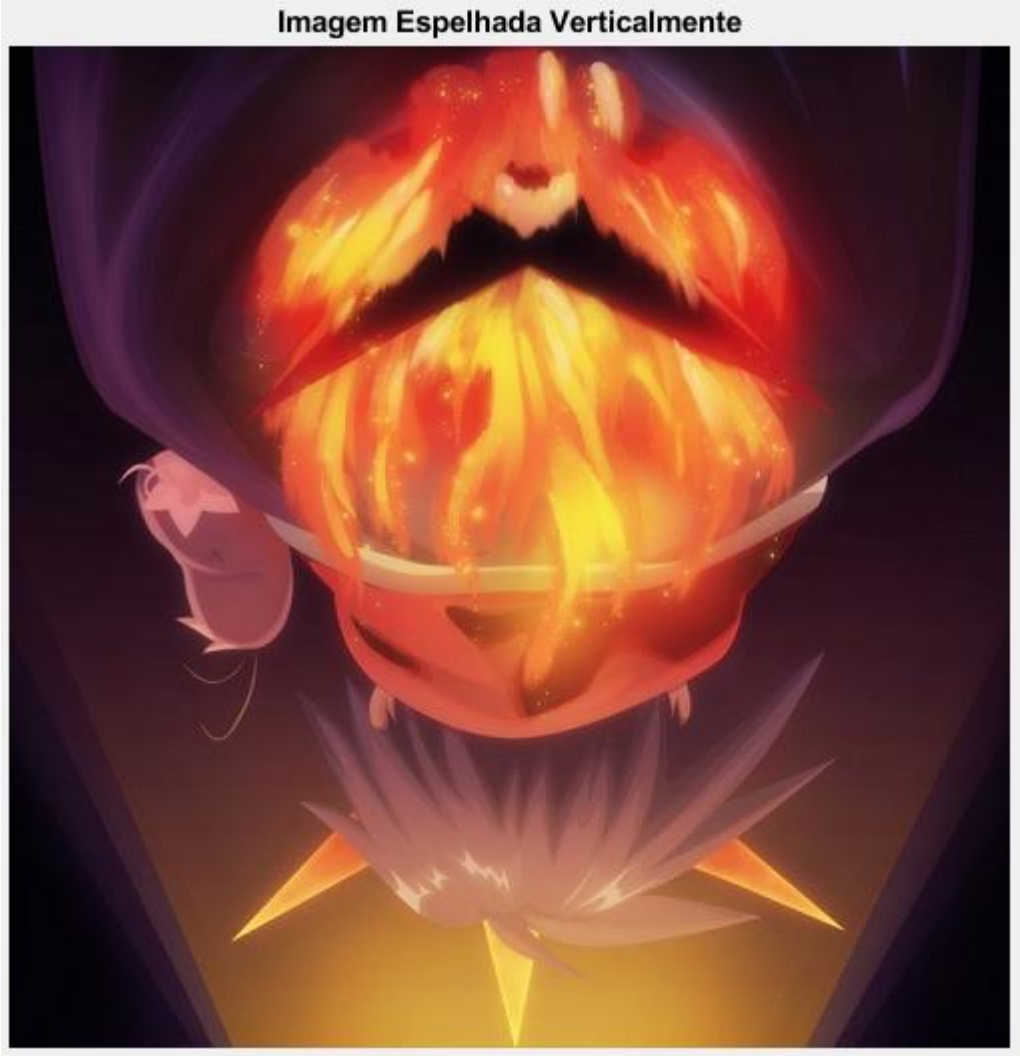
Para realizar a operação foi utilizado o seguinte trecho de código.

```
function imagem = espelhar_vertical(img)
   [i j k] = size(img);
   imagem = zeros(j, i, k, 'uint8');

   for x = 1: i
      for y = 1: j
         imagem(i - x + 1, j - y + 1, :) = img(x, y, :);
      end
   end
end
```

Como resultado, obteve-se a seguinte figura.

Figura 3 – Imagem espelhada horizontalmente



Fonte: Autoria Própria (2020)

Questão 2

Aplique a operação de deformação, a sua escolha.

A operação de deformação escolhida foi o cisalhamento vertical. A operação de cisalhamento desloca cada ponto da imagem em uma direção fixa, desta forma, Gonzalez e Woods (2010) trazem as equações para realizar o cisalhamento vertical e horizontal:

$$Cisalhamento\ vertical: \begin{cases} x = v + sv * w \\ y = w \end{cases}$$

$$Cisalhamento\ Horizontal: \begin{cases} x = v \\ y = sh * v + w \end{cases}$$

Onde x e y referem-se às novas coordenadas do pixel, w e v se referem as coordenadas originais do pixel. Sv se refere ao deslocamento em x e sh se refere ao deslocamento em y. Desta forma, ao calcular as novas para o pixel, obtém-se uma imagem deformada.

A imagem foi deformada com o cisalhamento vertical, com sh (sy no código) igual a 0.5. Para isso foi utilizado o seguinte trecho de código,

```
function imagem = cisalhamento(img, sx, sy)
   % sx e sy, definem o cisalhamento horizontal ou vertical
   % se sx então o cisalhamento é na vertical, se sy = 0 então o
   % cisalhamneto é na horizontal
   % Economizando assim, linhas de código

   [i j k] = size(img);
   imagem = zeros(j, i, k, 'uint8');

   for x = 1: i
      for y = 1: j
         x1 = round(x + sx * y); % round caso o coeficiente seja um numero real
         y1 = round(y + sy * x);
         imagem(x1, y1, :) = img(x, y, :);
      end
   end

end
```

Após a execução do código foi obtida a seguinte imagem deformada.

Figura 4 – Deformada verticalmente

Fonte: Autoria Própria (2020)

Questão 3

Aplique a operação de cortar a imagem.

Para cortar a imagem foi considerado que seria aplicado um corte de x pixels por y pixels, assim, foi escolhida uma submatriz de dimensões x por y, como a imagem cortada. O trecho de código foi utilizado para cortar a imagem em 100 x 500 pixels.

```
function imagem = cortar(img, cx, cy)
% cx quantos pixels serão cortados em x
% cy quantos pixels serão cortados em y
   [i j k] = size(img);
   imagem = zeros(cx, cy, k, 'uint8');

   for x = 1: cx
      for y = 1: cy
         imagem(x, y, :) = img(x, y, :);
      end
   end
end
```

Como, pode-se perceber, a imagem original foi cortada resultado em uma imagem de 100 x 500 pixels de dimensão. A figura a seguir foi obtida ao executar esse trecho de código.

Figura 5 – Imagem cortada em 100 x 500 pixels

Fonte: Autoria Própria (2020)

Percebe-se que a parte superior da imagem original foi cortada.

Questão 4

Aplique a operação de translação em uma parte da imagem.

Para Marques Filho e Vieira Neto (1999) a translação consiste no deslocamento linear de cada pixel da imagem na horizontal ou vertical para uma nova posição. Isto é dado um dx ou dy que indica o deslocamento é possível determinar a posição do objeto calculando a nova coordenada com base nesses deslocamentos.

Gonzalez e Woods (2010) trazem as equações para realizar a transformação de translação.

$$\begin{cases} x = v + dx \\ y = w + dy \end{cases}$$

Onde x e y são as novas coordenadas do pixel, w e v são as coordenadas antigas dos pixels, dx e dy são os deslocamentos na horizontal e verticalmente, respectivamente. Baseando-se nessas equações foi desenvolvido o seguinte trecho de código que realiza a translação de uma parte da imagem. Foram transladados 150 x 150 pixels.

```
function imagem = translacao(img, dx, dy)
% dx quantos pixels serão transladados em x
% dy quantos pixels serão transladados em y
   [i j k] = size(img);
   imagem = img;

   for x = 1: dx
      for y = 1: dy
         imagem(x, y, :) = 0;
         imagem(x, j - y, :) = img(x, y, :);
      end
   end

end
```

A parte transladada é deslocada para o lado oposto da imagem e sua posição antiga é preenchida com zeros, resultando assim na imagem a seguir.

Figura 6 – Imagem com parte transladada

Fonte: Autoria Própria (2020)

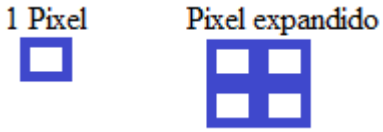
Questão 5

Aplique a operação de aumento de escala da imagem.

O processo de aumento de escala, consiste em ampliar a imagem, ampliar ou reduzir a imagem para melhorar a visualização (MARQUES FILHO; VIEIRA NETO, 1999). Para realizar a operação de aumento de escala pedido, foi utilizado o método sugerido por Marques Filho e Vieira Neto (1999), em que se duplica os valores dos pixels nas direções X ou Y ou em ambas.

Desta forma, para a resolução do exercício os valores dos pixels foram duplicados nas duas direções, a Figura a seguir ilustra essa situação.

Figura 7 – Processo de aplicação de aumento de escala



Fonte: Autoria Própria (2020)

O pixel a esquerda foi expandido em nas duas direções, resultando em 4 pixels, aumentando assim a escala da imagem. Esse processo foi aplicado em todos os pixels da imagem. Para tanto, foi utilizado o seguinte trecho de código.

```
function imagem = aumento_escala(img, ex, ey)
% ex escala em x
% ey escala em y
   [i j k] = size(img);
   imagem = zeros(ex*i, ey*j, k, 'uint8');

   xi = 1;
   yi = 1;

   for x = 1: i
      for y = 1: j
         k = 1;
         while k <= ex
            l = 1;
            while l <= ey
               imagem(xi + k, yi + l, :) = img(x, y, :);
               l = l + 1;
            end
            k = k + 1;
            yi = yi + 1;
         end
      end
      xi = xi + ex;
      yi = 1;
   end
end
```

Vale ressaltar que foi utilizada uma imagem de 200 x 200 pixels para ser possível ver de forma mais fácil a mudança na escala sofrida por ela. A imagem utilizada para esse experimento é apresentada a seguir. Foi usado como escala 2 para x e 2 para y.

Figura 8 – Imagem de 200 x 200 pixels usada

Fonte: Autoria Própria (2020)

Para verificar se o algoritmo estava funcionando corretamente foi empregada também a função imresize do matlab para redimensionar a imagem. O resultado obtido se encontra na Figura a seguir.

Figura 9 – À esquerda imagem obtida com a aplicação da função do imresize e à direita a imagem obtida com o código desenvolvido

Fonte: Autoria Própria (2020)

A imagem a esquerda foi obtida com a função do matlab e a imagem a direita foi obtida depois da execução do código escrito pelo autor. Como pode-se verificar ambos os resultados foram bastante próximos, ou até mesmo idênticos. Isso prova que o código funciona corretamente.

Questão 6

Crie um método que reduza a escala da imagem pela metade e outro que rotacione a imagem original em 180°. (No exemplo está com o fundo preto do tamanho

da imagem original e a resultante dentro apenas para mostrar a redução não será necessário o fundo preto no relatório)

Parte 1

Para reduzir a escala da imagem foi utilizado o processo inverso do aplicado na questão anterior. Ao invés de expandir os pixels eles são reduzidos, isto é, os conjuntos de pixels são reduzidos a um único pixel. Ao aplicar essa operação a escala da imagem é reduzida.

Para tanto, foi desenvolvido o seguinte trecho de código, observe que os laços não incrementam com passo 1 e sim com passo equivalente a escala. Mais uma vez foram mostradas duas imagens uma com a função imresize do Matlab e outra com o código desenvolvido a fim de verificar a resolução do problema.

```
function imagem = diminui_escala(img, e)
   [i j k] = size(img);
   imagem = zeros(round(i/e), round(j/e), k, 'uint8');

   xi = 1;
   yi = 1;

   for x  = 1: e: i
      yi = 1;
      for y  = 1: e: j
         imagem(xi, yi, :) = img(x, y, :);
         yi = yi + 1;
      end
      xi = xi + 1;
   end

end
```

A Figura a seguir traz a imagem reduzida usando a função do imresize (imagem a esquerda) e a imagem reduzida usando o trecho desenvolvido pelo autor. Como pode-se perceber os resultados obtidos foram bastante semelhantes.

Figura 10 – À esquerda imagem obtida com a aplicação da função do imresize e à direita a imagem obtida com o código desenvolvido

Fonte: Autoria Própria (2020)

Parte 2

A operação de rotação de uma imagem consiste em rotacionar a imagem em um ângulo arbitrário (MARQUES FILHO; VIEIRA NETO, 1999). Ela é uma das operações mais exigentes em termos de preservação das características de linhas retas (GONZALEZ; WOODS, 2010).

Segundo Gonzalez e Woods (2010) a operação de rotação pode ser realizada pelas seguintes equações.

$$x = v * \cos\alpha - w * sen\,\alpha$$

$$y = v * sen\,\alpha + w * \cos\alpha$$

Onde x e y são as novas coordenadas do pixel, v e w são as coordenadas antigas do pixel e α é o ângulo que se deseja rotacionar a imagem. Sendo assim, a base dessas equações foi aplicada para realizar a operação de rotação em 180 º.

Porém apenas aplicar as equações para gerar as novas coordenadas do pixel podem gerar alguns problemas como valores negativos ou valores reais. Que não existem no sistema em que a imagem está representada. Desta forma, o trecho de código aqui desenvolvido foi inspirado em https://stackoverflow.com/questions/19684617/image-rotation-by-matlab-without-using-imrotate.

No código primeiro calcula-se as dimensões da matriz para que a imagem rotacionada caiba dentro dela.

Depois cria-se a matriz que conterá a imagem rotacionada. Para que a imagem não fique com as pontas para fora, ou seja, a imagem rotacionada fique completamente da matriz, foram calculadas as coordenadas do centro da matriz. Por meio disso, a partir do centro tem-se uma estrutura parecida com a estrutura do plano cartesiano, essa abstração permite a manipulação mais fácil da operação.

Após isso, os laços realizam o cálculo das novas coordenadas e então são atribuídos os valores aos locais corretos. Para evitar inconsistências a linha que contém o if, verifica se os valores da coordenada estão dentro da matriz.

O código que realiza a rotação da imagem se encontra definido a seguir.

```
% Inspirado em: https://stackoverflow.com/questions/19684617/image-rotation-by-matlab-without-using-imrotate
function imagem = rotacao(img, teta)
   [i j k] = size(img);
   imagem = zeros(j, i, k, 'uint8');

   xf = ceil(i * abs(cosd(teta)) + j * abs(sind(teta)));
   yf = ceil(i * abs(sind(teta)) + j * abs(cosd(teta)));


   imagem = zeros(xf, yf, k, 'uint8');

   l = ceil(i / 2);
   h = ceil(j / 2);

   [a b c] = size(imagem);
```

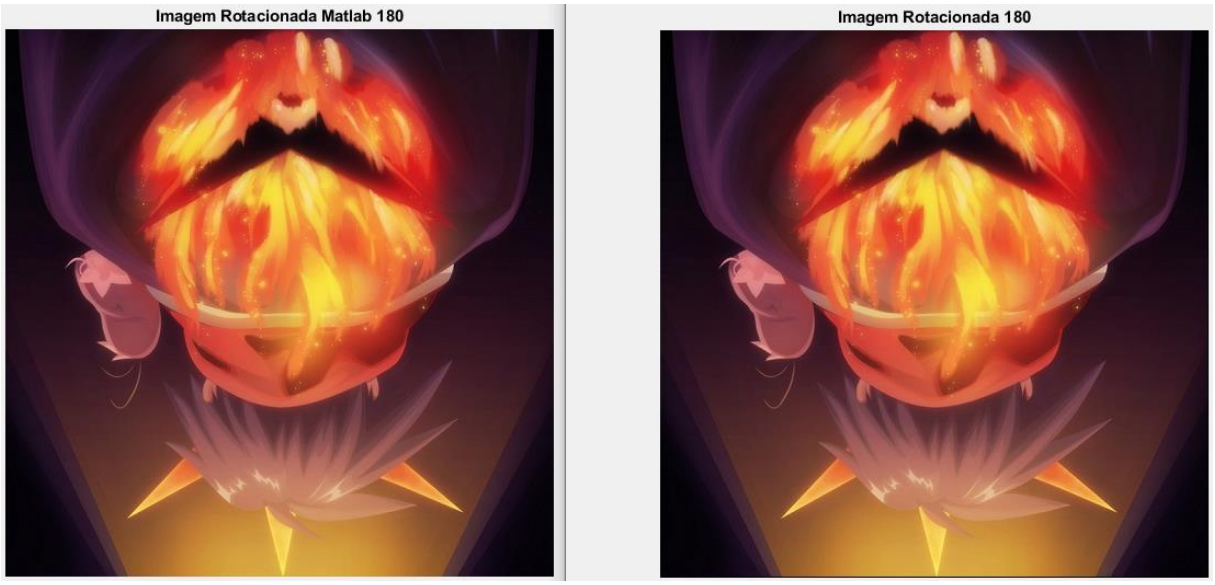
```
    meio_rot_x = ceil(a / 2);
    meio_rot_y = ceil(b / 2);

    for x  = 1: a
      for y  = 1: b
         x1 = round((x - meio_rot_x) * cosd(teta) + (y - meio_rot_y) * sind(teta)) + l;
         y1 = round(-(x - meio_rot_x) * sind(teta) + (y - meio_rot_y) * cosd(teta)) + h;
         if (x1 >= 1 && y1 >= 1 && x1 <= i && y1 <= j)
             imagem(x, y, :) = img(x1, y1, :);
         end
      end
    end

end
```

Mais uma vez para base de comparação, foi utilizada a função imrotate do Matlab para conferir o resultado obtido em relação ao resultado esperado. A Figura a seguir foi obtida após executar o código.

Figura 11 – À esquerda imagem obtida com a aplicação da função do imrotate e à direita a imagem obtida com o código desenvolvido



Fonte: Autoria Própria (2020)

Questão 7

Conceitue *morphing*.

Morphing (metamorfose) é uma transformação que leva uma imagem a outra, alterando tanto a sua forma quanto as suas intensidades (SILVA, 1994). Em outras palavras a *Morphing* descreve a combinação de distorção generalizada da imagem com dissolução cruzada entre os elementos da imagem, entende se por distorção a transformação geométrica que mapeia uma imagem de origem em uma imagem de destino (PENKLER, 2020).

Ela também é uma técnica bastante empregada em filmes ou animações para tornar uma imagem em outra (WIKIPÉDIA, 2020).

A ideia trás da técnica é que as imagens iniciais da sequência sejam muito parecidas com a imagem de origem e as imagens posteriores sejam muito parecidas com a imagem de saída (PENKLER, 2020).

Questão 8

Utilize a imagem resultante da questão 2 em seu método de rotação, rotacione em apenas 90º.

Para a rotação da imagem em 90 º foi utilizado o mesmo código da parte 2 da questão anterior, exceto pelo fato que a o ângulo utilizado foi diferente. Dessa forma, o seguinte código pode ser usado para rotacionar a imagem obtida usando a função da questão 2.

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 1\Imagens\img1.jpg');

Ic = cisalhamento(img, 0, 0.5);
I = rotacao(Ic, 90);

imshow(img);
title('imagem original');
figure();
imshow(I);
title('Imagem Rotacionada e Deformada');

figure();
imshow(imrotate(Ic, 90));
title('Imagem Rotacionada e Deformada Matlab');
```

Assim como na questão 7, foram mostradas duas imagens, uma com a saída do código e outra com a saída após aplicar a função imrotate do Matlab. Como pode-se ver, ambas os resultados foram bastante parecidos.

Figura 12 – À esquerda imagem obtida com a aplicação da função do imrotate e à direita a imagem obtida com o código desenvolvido

Fonte: Autoria Própria (2020)

Apêndice A

Código completo

```
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 1\Imagens\img.jpg');
[i j k] = size(img);

% ---------------------- Questao 1 ---------------------------------------
% Horizontal
img_espelhada = zeros(i, j, k, 'uint8');

for x = 1: i
   for y = 1: j
      img_espelhada(x, y, :) = img(y, x, :);
   end
end

figure();
imshow(img_espelhada);
title('Imagem Espelhada Horizontalmente');

% Vetical
img_espelhada = zeros(i, j, k, 'uint8');

for x = 1: i
   for y = 1: j
      img_espelhada(x, y, :) = img(i - x + 1, j - y + 1, :);
   end
end

figure();
imshow(img_espelhada);
title('Imagem Espelhada Verticalmente');
```

```
% ---------------------- Questao 1 ---------------------------------------

% ---------------------- Questao 2 ---------------------------------------
sx = 0;
sy = 0.5;

img_cisalhada = zeros(i, j, k, 'uint8');

for x = 1: i
   for y = 1: j
      x1 = round(x + sx * y); % round caso o coeficiente seja um numero real
      y1 = round(y + sy * x);
      img_cisalhada(x1, y1, :) = img(x, y, :);
   end
end

figure();
imshow(img_cisalhada);
title('Imagem com deformação');
% ---------------------- Questao 2 ---------------------------------------

% ---------------------- Questao 3 ---------------------------------------
cx = 100;
cy = 500;
img_cortada = zeros(cx, cy, k, 'uint8');

for x = 1: cx
   for y = 1: cy
      img_cortada(x, y, :) = img(x, y, :);
   end
end
```

```
figure();
imshow(img_cortada);
title('Imagem Cortada');
% ---------------------- Questao 3 ---------------------------------------

% ---------------------- Questao 4 ---------------------------------------
cx = 150;
cy = 150;
img_translacao = img;

for x = 1: cx
   for y = 1: cy
      img_translacao(x, y, :) = 0;
      img_translacao(x, j - y, :) = img(x, y, :);
   end
end

figure();
imshow(img_translacao);
title('Imagem com translação');
% ---------------------- Questao 4 ---------------------------------------

% ---------------------- Questao 5 ---------------------------------------
e = 2;
img_escala = zeros(e*i, e*j, k, 'uint8');

xi = 1;
yi = 1;

for x = 1: i
  for y = 1: j
     k = 1;
```

```
      while k <= e
         l = 1;
         while l <= e
             img_escala(xi + k, yi + l, :) = img(x, y, :);
             l = l + 1;
         end
         k = k + 1;
         yi = yi + 1;
      end
   end
   xi = xi + e;
   yi = 1;
end

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

figure();
imshow(img_escala);
title('Imagem com escala');

figure();
imshow(imresize(img, e));
title('Imagem com escala Matlab');
% ---------------------- Questao 5 ---------------------------------------

% ---------------------- Questao 6 ---------------------------------------
% Parte 1
e = 2;
img_metade = zeros(round(i/e), round(j/e), k, 'uint8');
```

```
xi = 1;
yi = 1;

for x  = 1: e: i
   yi = 1;
  for y  = 1: e: j
     img_metade(xi, yi, :) = img(x, y, :);
     yi = yi + 1;
  end
  xi = xi + 1;
end

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

figure();
imshow(img_metade);
title('Imagem com escala reduzida');

figure();
imshow(imresize(img, .5));
title('Imagem com escala reduzida Matlab');

% Parte 2
% Inspirado em: https://stackoverflow.com/questions/19684617/image-rotation-by-matlab-without-using-imrotate
teta = 180;
xf = ceil(i * abs(cosd(teta)) + j * abs(sind(teta)));
yf = ceil(i * abs(sind(teta)) + j * abs(cosd(teta)));
```

```
img_rot = zeros(xf, yf, k, 'uint8');

l = ceil(i / 2);
h = ceil(j / 2);

[a b c] = size(img_rot);
meio_rot_x = ceil(a / 2);
meio_rot_y = ceil(b / 2);

for x  = 1: a
  for y  = 1: b
     x1 = round((x - meio_rot_x) * cosd(teta) + (y - meio_rot_y) * sind(teta)) + l;
     y1 = round(-(x - meio_rot_x) * sind(teta) + (y - meio_rot_y) * cosd(teta)) + h;
     if (x1 >= 1 && y1 >= 1 && x1 <= i && y1 <= j)
         img_rot(x, y, :) = img(x1, y1, :);
     end
  end
end

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

figure();
imshow(img_rot);
title('Imagem Rotacionada 180');

figure();
imshow(imrotate(img, teta));
title('Imagem Rotacionada Matlab 180');
% ---------------------- Questao 6 ---------------------------------------
```

```
% ---------------------- Questao 8 ---------------------------------------
img = imread('D:\Estudos\Eng. de Computação\Processamento Digital de Imagens\Listas\Lista 1\Imagens\img2.png');
[i j k] = size(img);
teta = 90;
xf = ceil(i * abs(cosd(teta)) + j * abs(sind(teta)));
yf = ceil(i * abs(sind(teta)) + j * abs(cosd(teta)));



img_rot = zeros(xf, yf, k, 'uint8');


l = ceil(i / 2);
h = ceil(j / 2);

[a b c] = size(img_rot);
meio_rot_x = ceil(a / 2);
meio_rot_y = ceil(b / 2);

for x  = 1: a
  for y  = 1: b
     x1 = round((x - meio_rot_x) * cosd(teta) + (y - meio_rot_y) * sind(teta)) + l;
     y1 = round(-(x - meio_rot_x) * sind(teta) + (y - meio_rot_y) * cosd(teta)) + h;
     if (x1 >= 1 && y1 >= 1 && x1 <= i && y1 <= j)
         img_rot(x, y, :) = img(x1, y1, :);
     end
  end
end

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

```
figure();
imshow(img_rot);
title('Imagem Rotacionada 90');

figure();
imshow(imrotate(img, teta));
title('Imagem Rotacionada Matlab 90');
% ---------------------- Questao 8 ---------------------------------------
```

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.

PENKLER, Thomas. **Morphing**. Disponível em: https://old.cescg.org/CESCG97/penkler/morphing.htm. Acesso em: 11 mar. 2020.

SILVA, Bruno Costa da**. Deformação e Metamorfose de Imagens Digitais**. 1994. 65 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1994.

WIKIPÉDIA. **Morphing**. Disponível em: https://en.wikipedia.org/wiki/Morphing. Acesso em: 11 mar. 2020.