Computação Gráfica

- **Trabalho 4:** Transformação com interpolação em imagens

Bruna Costa Cons

201110341611

Código:

```
function transf(arq, p0)

  pkg load image;
  % 'p0' é matriz de pontos de entrada da imagem
  % 'arq' é a imagem que será lida
  img = imread(arq);
  img = rgb2gray(img);

  % Mapeando os pixels da imagem 'img'
  lin = size(img, 1); % qtd de linhas de 'img'
  col = size(img, 2); % qtd de colunas de 'img'
  % A figura original é exibida:
  figure('Name','Figura Original','NumberTitle','off');
  imshow(img);

  % A seguir, os elementos são retirados um a um da matriz de entrada
  % Os elementos com L na variável são pontos da imagem distorcida
  x0 = p0(1,1); % (linha, coluna)
  x1 = p0(2,1);
  x2 = p0(3,1);
  x3 = p0(4,1);
  y0 = p0(1,2);
  y1 = p0(2,2);
  y2 = p0(3,2);
  y3 = p0(4,2);
  x0L = p0(1,3);
  x1L = p0(2,3);
  x2L = p0(3,3);
  x3L = p0(4,3);
  y0L = p0(1,4);
  y1L = p0(2,4);
  y2L = p0(3,4);
  y3L = p0(4,4);

  % Matriz dos coeficientes (sistema de equações):
  A = [x0 y0 1 0 0 0 (-1)*x0*x0L (-1)*y0*x0L;
      0 0 0 x0 y0 1 (-1)*x0*y0L (-1)*y0*y0L;
      x1 y1 1 0 0 0 (-1)*x1*x1L (-1)*y1*x1L;
      0 0 0 x1 y1 1 (-1)*x1*y1L (-1)*y1*y1L;
      x2 y2 1 0 0 0 (-1)*x2*x2L (-1)*y2*x2L;
      0 0 0 x2 y2 1 (-1)*x2*y2L (-1)*y2*y2L;
      x3 y3 1 0 0 0 (-1)*x3*x3L (-1)*y3*x3L;
      0 0 0 x3 y3 1 (-1)*x3*y3L (-1)*y3*y3L];

  % Matriz dos coeficientes (transformação projetiva):
  L = [x0L; y0L; x1L; y1L; x2L; y2L; x3L; y3L];

  % Matriz de observações:
  x = inv((A.')*A)*((A.')*L);
```

```
  % Queremos obter a matriz de transformação 3x3 a partir dos elementos de 'x' (há apenas 8 elementos em 'x')
  % Para isso, deve-se preencher o último elemento com 1
  % Matriz de Transformação:
 T = [x(1,1) x(2,1) x(3,1);
     x(4,1) x(5,1) x(6,1);
     x(7,1) x(8,1) 1];

  % A imagem transformada é setada com um fundo preto, com a mesma quantidade de linhas e colunas da imagem original
   new_img = zeros(lin,col,"uint8");

 for i = 1:(lin)
   for j = 1:(col)
      % Uma matriz 'M' é criada, representando o vetor do ponto atual, na linha 'i' e coluna 'j'
      M = [i; j; 1];
      % A matriz de transformação 'T' encontrada é aplicada a cada ponto da iteração
      M2 = T*M;
      % Em seguida, é feita a normalização das coordenadas, ao dividir cada elemento do vetor por sua última coordenada (z). A normalização é feita para que possamos utilizar as propriedades de uma transformação afim.
      M3 = M2/M2(3,1);
      % Os valores transformados de x e y são retirados da matriz resultado M3, sendo salvos em Xn e Yn, respectivamente:
      Xn = M3(1,1);
      Yn = M3(2,1);

      % Interpolação Bilinear:
      % u obtém apenas a parte decimal que "sobra" de um pixel inteiro, no eixo x
      u = Xn - floor(Xn);
      % v obtém apenas a parte decimal que "sobra" de um pixel inteiro, no eixo y
      v = Yn - floor(Yn);
      % Xn e Yn são arredondados para o inteiro mais próximo para que a interpolação possa ser feita
      Xn = int32(Xn);
      Yn = int32(Yn);

      % Nos 'ifs' seguintes, é garantido que, se a matriz de transformação inserida pelo usuário fizer ultrapassar o tamanho da imagem escolhida, a imagem transformada irá ser cortada
      if ((Xn <= 0) || (Xn > lin))
        Xn = 1;
      endif

      if ((Yn <= 0) || (Yn > col))
        Yn = 1;
      endif

      % Em seguida, a interpolação é realizada, analisando os 4 pixels em volta do pixel atual da iteração. Porém, quando não houver 4 pixels em volta da imagem, no caso dos cantos da imagem, devemos limitar essa análise, por isso foi usado o 'if' abaixo:
```

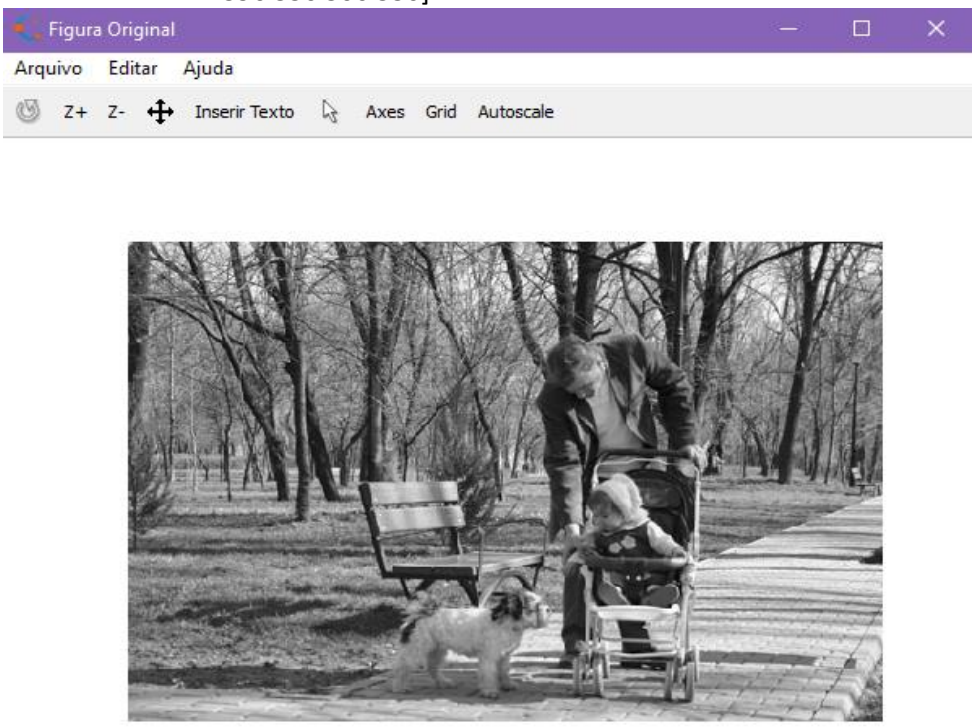
```
      if ((i < lin) && (j < col))
         new_img(Xn,Yn) = img(i,j)*(1-u)*(1-v) + img(i+1,j)*u*(1-v) + img(i,j+1)*(1-u)*v +
img(i+1,j+1)*u*v;
      else
      % No caso dos cantos da imagem, apenas parte da fórmula da interpolação foi usada, ao
analisar apenas um pixel
         new_img(Xn,Yn) = img(i,j)*(1-u)*(1-v);
      endif

   endfor
  endfor

  figure('Name','Figura Transformada','NumberTitle','off');
  imshow(new_img);

endfunction
```

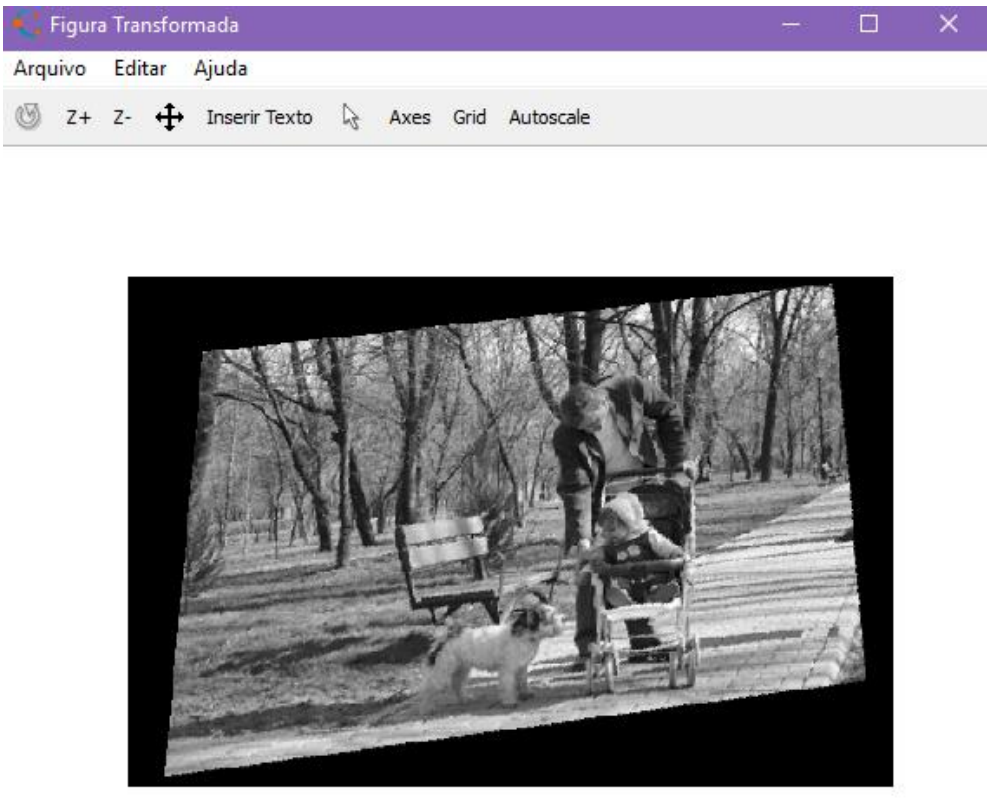
Exemplos de execução:

- **transf("dog.png", [1 1 50 50; 1 550 1 500; 590 1 590 1; 590 550 500 550]);**

Sendo:

arq = "dog.png" e p0 = [1 1 50 50;
1 550 1 500;
590 1 590 1;
590 550 500 550].





- **transf("gatos.png", [1 1 50 50; 1 350 1 300; 390 1 390 1; 390 350 300 350]);**

Sendo:

arq = "gatos.png" e p0 = [1 1 50 50;
1 350 1 300;
390 1 390 1;
390 350 300 350].