Processamento de imagens

Enzo Borges Segala Jonathan Gabriel Nunes Mendes
Matheus Machado Cezar Marcos Henrique Volpato de Moraes
Miguel Predebon Abichequer Nicolas Rosenthal Dal Corso
Rafael Silveira Bandeira

11/11/2025

Sumário

1 Introdução

...

2 Tarefas

2.1 Interpolação em imagens coloridas

```
%% Interpolação em imagens coloridas

%% 100% chatGPT --- Está aqui apenas para testar os estilos. Não funciona
    no MATLAB da UFRGS

clc; clear; close all;

%% 1. Read two colored images
img1 = im2double(imread('peppers.png'));
img2 = im2double(imread('onion.png'));

%% 2. Set resize scale
scale = 1.8;

%% 3. Get new dimensions
[rows, cols, ch] = size(img1);
new_rows = round(rows * scale);
new_cols = round(cols * scale);

%% 4. Manual interpolation methods

% === NEAREST NEIGHBOR ===
nearestInterp = @(img, r, c, s) ...
    arrayfun(@(x,y) img(round(x), round(y), :), ...
    linspace(1, r, new_rows)', linspace(1, c, new_cols));

% (Note: 'well use helper functions for clarity)
img1_nn = nearest_neighbor_resize(img1, scale);
img2_nn = nearest_neighbor_resize(img2, scale);

% === BILINEAR ===
img1_bilinear = bilinear_resize(img1, scale);
img2_bilinear = bilinear_resize(img2, scale);

% === BICUBIC ===
img1_bicubic = bicubic_resize(img1, scale);
img2_bicubic = bicubic_resize(img2, scale);
```

```
%% 5. Display results
methods = {'Nearest Neighbor (0th order)', ...
           'Bilinear (1st order)', ...
           'Bicubic (3rd order)'};
imgs1 = {img1_nn, img1_bilinear, img1_bicubic};
imgs2 = {img2_nn, img2_bilinear, img2_bicubic};

figure('Name','Interpolation Comparison','Position',[100,100,1200,700]);
for i = 1:3
    subplot(3,2,2*i-1);
    imshow(imgs1{i});
    title(['Image 1 - ' methods{i}]);
    subplot(3,2,2*i);
    imshow(imgs2{i});
    title(['Image 2 - ' methods{i}]);
end


%% ==== Helper functions ====

function out = nearest_neighbor_resize(img, scale)
    [rows, cols, ch] = size(img);
    new_rows = round(rows * scale);
    new_cols = round(cols * scale);
    [X, Y] = meshgrid(1:new_cols, 1:new_rows);
    Xq = round(X / scale);
    Yq = round(Y / scale);
    Xq(Xq < 1) = 1; Xq(Xq > cols) = cols;
    Yq(Yq < 1) = 1; Yq(Yq > rows) = rows;
    out = zeros(new_rows, new_cols, ch);
    for k = 1:ch
        for i = 1:new_rows
            for j = 1:new_cols
                out(i,j,k) = img(Yq(i,j), Xq(i,j), k);
            end
        end
    end
end


function out = bilinear_resize(img, scale)
    [rows, cols, ch] = size(img);
    new_rows = round(rows * scale);
    new_cols = round(cols * scale);
    [X, Y] = meshgrid(1:new_cols, 1:new_rows);
    X = X / scale; Y = Y / scale;
```

```
    out = zeros(new_rows, new_cols, ch);
    for k = 1:ch
        for i = 1:new_rows
            for j = 1:new_cols
                x = X(i,j); y = Y(i,j);
                x1 = floor(x); y1 = floor(y);
                x2 = min(x1 + 1, cols); y2 = min(y1 + 1, rows);
                a = x - x1; b = y - y1;
                if x1 < 1 || y1 < 1, continue; end
                Q11 = img(y1, x1, k);
                Q12 = img(y2, x1, k);
                Q21 = img(y1, x2, k);
                Q22 = img(y2, x2, k);
                out(i,j,k) = Q11*(1-a)*(1-b) + Q21*a*(1-b) + Q12*(1-a)*b
                    + Q22*a*b;
            end
        end
    end
end


function out = bicubic_resize(img, scale)
    [rows, cols, ch] = size(img);
    new_rows = round(rows * scale);
    new_cols = round(cols * scale);
    out = zeros(new_rows, new_cols, ch);
    for k = 1:ch
        out(:,:,k) = bicubic_channel(img(:,:,k), scale);
    end
end


function out = bicubic_channel(channel, scale)
    [rows, cols] = size(channel);
    new_rows = round(rows * scale);
    new_cols = round(cols * scale);
    out = zeros(new_rows, new_cols);
    [X, Y] = meshgrid(1:new_cols, 1:new_rows);
    X = X / scale; Y = Y / scale;

    for i = 1:new_rows
        for j = 1:new_cols
            x = X(i,j); y = Y(i,j);
            x1 = floor(x); y1 = floor(y);
            dx = x - x1; dy = y - y1;
            patch = zeros(4,4);
```

```
            for m = -1:2
                for n = -1:2
                    xm = min(max(x1 + m, 1), cols);
                    yn = min(max(y1 + n, 1), rows);
                    patch(n+2,m+2) = channel(yn, xm);
                end
            end
            out(i,j) = bicubic_interpolate(patch, dx, dy);
        end
    end
end


function val = bicubic_interpolate(patch, dx, dy)
    % Cubic kernel
    function w = cubic(t)
        a = -0.5; % Catmull-Rom
        abs_t = abs(t);
        if abs_t <= 1
            w = (a+2)*abs_t^3 - (a+3)*abs_t^2 + 1;
        elseif abs_t < 2
            w = a*abs_t^3 - 5*a*abs_t^2 + 8*a*abs_t - 4*a;
        else
            w = 0;
        end
    end

    wx = zeros(1,4); wy = zeros(1,4);
    for i = 1:4, wx(i) = cubic(dx + 1 - i); end
    for j = 1:4, wy(j) = cubic(dy + 1 - j); end
    val = wy * patch * wx';
end
```

2.2 Realce de imagens no domínio espacial (da imagem)

```
%% Realce de imagens no domínio espacial (da imagem)
```

2.3 Filtragem espacial

```
%% Filtragem espacial
```

2.4 Dithering

```
%% Dithering
```

2.5 Operacoes Geométricas

```
%% Operacoes Geométricas
```