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 **3**

2 Tarefas **3**

2.1 Interpolação em imagens coloridas 3

2.1.1 Arquivo principal 3

2.1.2 Funções auxiliares 4

2.2 Realce de imagens no domínio espacial (da imagem) 6

2.2.1 Arquivo principal 6

2.3 Filtragem espacial 6

2.3.1 Arquivo principal 6

2.4 Dithering 6

2.4.1 Arquivo principal 6

2.5 Operacoes Geométricas 6

2.5.1 Arquivo principal 6

1 Introdução

...

2 Tarefas

2.1 Interpolação em imagens coloridas

2.1.1 Arquivo principal

```
%% Interpolação em imagens coloridas

clc; clear; close all;

addpath(genpath(strcat(pwd, '/mfunctions')));

%% 2. Set resize scale
scale = 1.8;

desktop = com.mathworks.mde.desk.MLDesktop.getInstance;
myGroup = desktop.addGroup('myGroup');
desktop.setGroupDocked('myGroup', 0);
myDim   = java.awt.Dimension(4, 2);   % 4 columns, 2 rows
% 1: Maximized, 2: Tiled, 3: Floating
desktop.setDocumentArrangement('myGroup', 2, myDim)
figH    = gobjects(1, 8);

baseimgs = {'peppers.png', 'onion.png'};
iFig = 1;
for k = 1:2
    figH(iFig) = figure( ...
        'WindowStyle', 'docked', ...
        'Name', 'Original', ...
        'NumberTitle', 'off' ...
    );
    iFig = iFig + 1;
    drawnow;
    pause(0.02);  % Magic, reduces rendering errors

    original = im2double(imread(baseimgs{k}));
    imshow(original, 'InitialMagnification', 'fit');

    fs    = { @nearest_neighbour_resize    , @bilinear_resize      ,
       @bicubic_resize };
    names = {'Nearest Neighbor (0th order)', 'Bilinear (1st order)', '
       Bicubic (3rd order)'};
```

```
    for i = 1:3
        figH(iFig) = figure( ...
            'WindowStyle', 'docked', ...
            'Name', names{i}, ...
            'NumberTitle', 'off' ...
        );
        iFig = iFig + 1;
        drawnow;
        pause(0.02);  % Magic, reduces rendering errors
        imshow(fs{i}(original, scale), 'InitialMagnification', 'fit');
    end
end
```

2.1.2 Funções auxiliares

```
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

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 = 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 = round(X / scale); Y = round(Y / scale);
    X(X < 1) = 1; X(X > cols) = cols;
    Y(Y < 1) = 1; Y(Y > rows) = rows;


    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;
                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 = nearest_neighbour_resize(img, scale)
    [rows, cols, ch] = size(img);
    new_rows = round(rows * scale);
    new_cols = round(cols * scale);
    [Y, X] = meshgrid(1:new_cols, 1:new_rows);
    X = round(X / scale); Y = round(Y / scale);
    X(X < 1) = 1; X(X > rows) = rows;
    Y(Y < 1) = 1; Y(Y > cols) = cols;


    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);
                out(i,j,k) = img(x, y, k);
            end
        end
    end
end
```

2.2 Realce de imagens no domínio espacial (da imagem)

2.2.1 Arquivo principal

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

2.3 Filtragem espacial

2.3.1 Arquivo principal

```
%% Filtragem espacial
```

2.4 Dithering

2.4.1 Arquivo principal

```
%% Dithering
```

2.5 Operacoes Geométricas

2.5.1 Arquivo principal

```
%% Operacoes Geométricas
```