# Homework3

姓名：谢雨汐　　学号：1600012860

## 一、题目要求

模拟 Matlab 的 imresize()写一个你自己的 imresize()函数
至少应实现'nearest'和'bilinear'两种方法
附件的"football"和"kids"可作测试之用

## 二、问题分析与解答

### 1. 读取'jpg'和'tiff'文件

**【详细代码参见文件 test.m】**

这里的'jpg'可以代表 Matlab 所支持的以'uint8'为像素值数据格式的图像文件，所以使用简单的 imread()指令即可得到像素矩阵.

重点是'tiff'文件不属于上述范畴，使用简单的 imread()函数得到的像素矩阵在 imshow()的时候将会呈现黑白图像，所以此处需要：a) 读出 tiff 文件的 map 矩阵，将其转换为三维形式；b) 三维形式中的每一个像素值并非'uint8'格式，故在编写 imresize()时需要将其作为一种特殊情况处理.

```
% read jpg
jpg = imread('football.jpg');
% read tiff
[T, map] = imread('kids.tiff');
tiff = ind2rgb(T, map);
```

### 2. 编写 my_imresize()函数

**【详细代码参见文件 my_imresize.m】**

**1st step**. 计算 resize 后的图像的大小(width * height)，并将所需的各个像素位点的坐标值与原图坐标对齐

之后的 Nearest Neighbor Resampling 和 Bilinear Interpolation 算法均是根据此处的坐标值计算

```
% get the size of input raw image
[width, height, slice] = size(raw_image);
% calculate the size of output image
w = round((width-1) * scale);
h = round((height-1) * scale);
% calculate the location of interplotted node
delta = 1/scale;
W = (1:delta:width)';
H = (1:delta:height)';
% initialize the output image
if (fmp == 'tif')
    resize_image = zeros(w, h, slice);
else
    resize_image = zeros(w, h, slice, 'uint8');
end
```

**2nd step**. Nearest Neighbor Resampling

利用 round()函数找到最近邻点，将其像素值赋给目标像素矩阵即可

```
for k = 1:slice
    for i = 1:w
        x = round(W(i));
        for j = 1:h
            y = round(H(j));
            % use the method of Nearest Neighbor Resampling
            if mode == 'n'
                resize_image(i, j, k) = raw_image(x, y, k);
```

**3rd step**. Bilinear Interpolation

找到与目标点最相邻的四个像素位点，利用 bilinear 插值公式求解

此处要注意特殊情况 — 目标点坐标与原图中重合之处，可直接使用原图像素值.

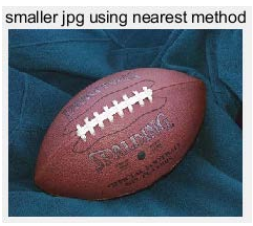
```
% use the method of Bilinear Interpolation
else if mode == 'b'
       m = floor(W(i)); n = floor(H(j));
        M = m + 1; N = n + 1;
        a = W(i) - m; b = H(j) - n;
        if m == W(i)
            M = W(i);
            a = 0.5;
        end
        if n == H(j)
            N = H(j);
            b = 0.5;
        end
        X00 = raw_image(m, n, k);
        X01 = raw_image(m, N, k);
        X10 = raw_image(M, n, k);
        X11 = raw_image(M, N, k);
        resize_image(i, j, k) = (1-a) * (1-b) * X00 + (1-a) * b * X01 + a * (1-b) * X10 + a * b * X11;
```

**3. 测试函数**

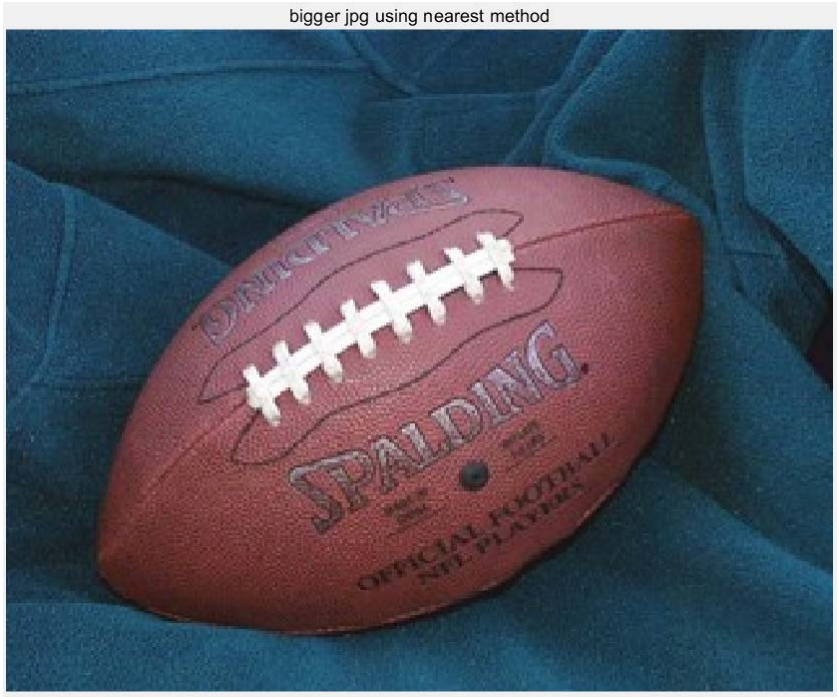
**【具体代码参见文件 test.m】**

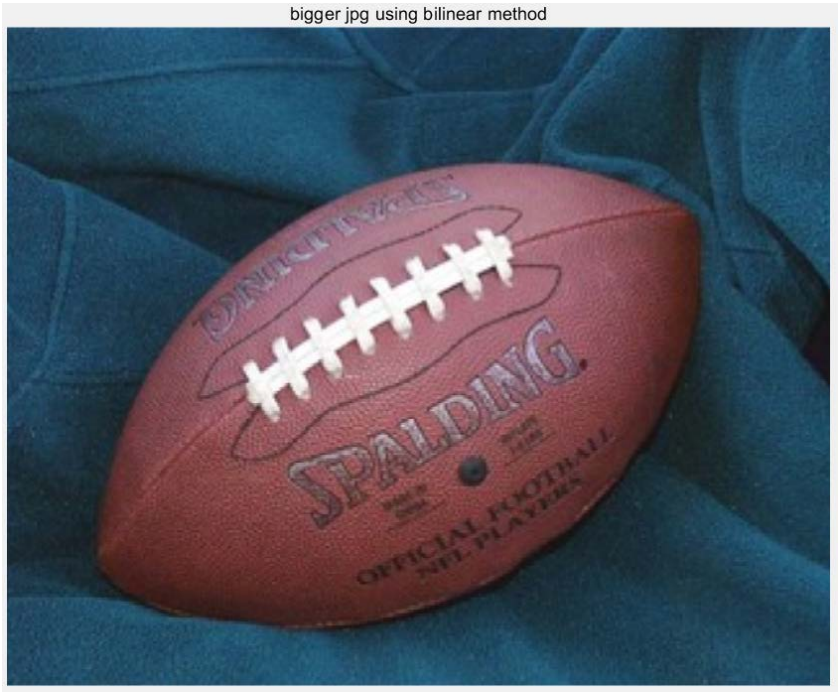
分别对'jpg'和'tiff'两种文件格式的两种插值方法 nearest / bilinear 均作了放大和缩小的测试，其中，'jpg'文件的测试成功可代表其他 Matlab 所支持的'uint8'格式文件(如'ppm'等)的可行性，结果如下：

• 'football.jpg'缩小为原图的 0.7 倍




• ’football.jpg’放大为原图的 2.6 倍




• ’kids.tiff’ 缩小为原图的 0.4 倍


• ’kids.tiff’ 放大为原图的 1.3 倍