1 实验内容及目的

1.1 实验目的

完成图片放大后的分辨率重建任务。

1.2 实验内容

1. 完成常用的插值方法 bilinear、bicubic、nearest 的复现
2. 完成基于字典的超分辨重建方法

2 实验相关原理描述

2.1 最邻近插值法(nearest)

定义: 目标各像素点的灰度值代替原图中与其最邻近的像素灰度值

公式:

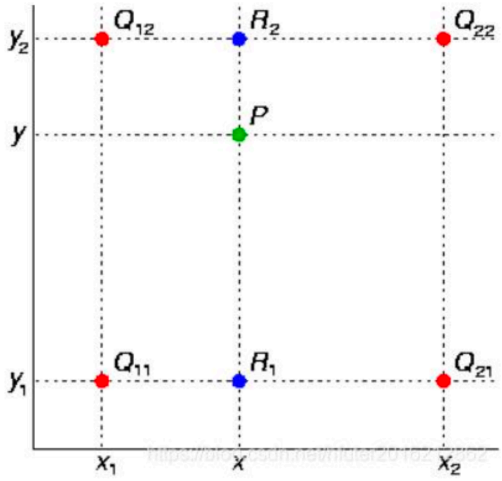
$$\begin{aligned} srcX &= dstX * (srcWidth/dstWidth) \\ srcY &= dstY * (srcHeight/dstHeight) \end{aligned} \tag{1}$$

其中XY均为坐标点

2.2 双线性插值法(bilinear)

定义: 根据点相邻最近的 4 个点的像素值算出该点的像素值

如下图，已知 Q 求 P，先算 R1，R2，再用 R1R2 算 P



$$\begin{aligned} f(R_1) &\approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \\ f(R_2) &\approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \\ f(P) &\approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \end{aligned} \tag{2}$$

利用 1 中方法求插入点于原图的位置，注意当 x，y 取得整数时的特殊情况

2.3 双三次插值法(bicubic)

定义: 利用邻近的 4*4 的点求出权重后插值

a 值一般取值为-0.5 或-1，W(x)中 x 为与 P 点的距离

$$B(X,Y)=\sum_{i=0}^{3}\sum_{i=0}^{3}a_{ij}\times W(i)\times W(j)$$

$$W(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & |x|\le 1\\ a|x|^3-5a|x|^2+8a|x|-4a & 1<|x|<2\\ 0 & \text{otherwise}\end{cases} \tag{3}$$

2.4 基于字典的超分辨率重建

基于字典的超分辨率重建（Dictionary-based Super-Resolution Reconstruction）是一种用于图像或视频超分辨率增强的方法。它的主要思想是通过学习一个高分辨率图像字典，将低分辨率输入图像映射到高分辨率空间中。基于字典的超分辨率重建的一般步骤如下：

1. 数据准备：收集一组具有高分辨率图像和相应低分辨率图像的训练样本。这些样本可以是成对的高分辨率和低分辨率图像，或者只有高分辨率图像。
2. 字典学习：使用训练样本来学习一个字典。字典是一组原子（基本单位），表示高分辨率图像中的局部结构。常用的字典学习方法包括K-SVD、稀疏编码等。通过字典学习，可以捕捉到高分辨率图像的纹理和结构信息。
3. 低分辨率图像表示：将输入的低分辨率图像切分成重叠的块，并表示成稀疏向量。这些稀疏向量是用字典中的原子线性组合表示的，以表示低分辨率图像中的局部结构。
4. 高分辨率重建：通过字典中的原子线性组合，将低分辨率图像块映射到高分辨率空间中。这个过程可以看作是将低分辨率图像的纹理和结构信息与字典中的原子进行匹配，以重建出更高分辨率的图像块。
5. 图像拼合：对重建的高分辨率图像块进行拼合，以生成完整的高分辨率图像。常用的拼合方法包括重叠加权平均（Overlapping Averaging）或者其他插值技术。

相关公式：

设高分辨图像块展开成一位数列 $x\in\Re^n$ 可以由一组 K 个基元 (atom) 的过完备字典 $D\in\Re^{n\times K}$ 线性稀疏表示。其中 $\alpha_0\in\Re^K$ 是表征向量

$$x=D\alpha_0 \tag{4}$$

每个图像块减去均值，字典表示图像的纹理，重建阶段将低分辨的均值直接作为高分辨图像的均 值。字典学习问题可以构建为

$$\min_{\alpha}\|FD_l\boldsymbol{\alpha}-Fy\|_2^2+\lambda\|\boldsymbol{\alpha}\|_1 \tag{5}$$

其中 F 是特征提取操作， λ 是拉格朗日乘子。利用高低分辨表征统一的假设，将问题改写为

$$\min_{\alpha}\|\tilde{D}\alpha-\tilde{y}\|_2^2+\lambda\|\alpha\|_1 \tag{6}$$

将问题转化为 QP 问题求解, 上式变为

$$\min_{\alpha}\frac{1}{2}\alpha^TA\alpha+b'\alpha+\lambda|\alpha| \tag{7}$$

其中 $A = DI^{\star} * DI, b = -DI' * Fy$ (此处 DI 为归一化后的)

3 实验过程

3.1 最邻近插值法(nearest)

```
clear
clc

%读入待处理的图像
srcImg = imread('input.bmp');

%读入放大倍数
T = 3;

%最邻近插值法(nearest)
srcWidth = size(srcImg, 2);
srcHeight = size(srcImg, 1);
dstWidth = size(srcImg, 2) * T;
dstHeight = size(srcImg, 1) * T;

%创建一个空的目标图像
dstImg1 = uint8(zeros(dstHeight, dstWidth, size(srcImg, 3)));

%最邻近插值法
%遍历目标图像的每个像素
for y = 1 : dstHeight
    for x = 1 : dstWidth
        %计算目标图像像素对应的原图像像素坐标
        srcX = ceil(x/T);
        srcY = ceil(y/T);
        %将原图像像素的值赋给目标图像像素
        dstImg1(y, x, :) = srcImg(srcY, srcX, :);
    end
end

%显示原图和最邻近插值法的结果
figure;
subplot(1,2,1),imshow(srcImg),title('原图');
subplot(1,2,2),imshow(dstImg1),title('最邻近插值法');
```

这段代码的主要思想是根据放大倍数T创建空的目标图像，然后通过遍历目标图像中的每个像素，计算其在原图像中对应的像素坐标，然后将原图像像素的值赋给目标图像像素。

3.2 双线性插值法(bilinear)

代码的核心思想是在遍历空的目标图像中的每个像素时：

1. 计算原图像中与目标像素最近的四个像素的位置权重
2. 进行检查，对于超出边界的像素，使用最近边界像素进行填充
3. 对于每个目标像素分别计算Q11，Q12，Q21，Q22

4. 分别在x轴方向和y轴方向上进行线性插值

5. 将插值后的像素赋给目标像素

```
clear
clc

% 读入待处理的图像
srcImg = imread('input.bmp');

% 设置放大倍数
T = 3;

srcWidth = size(srcImg, 2);
srcHeight = size(srcImg, 1);
dstWidth = size(srcImg, 2) * T;
dstHeight = size(srcImg, 1) * T;

dstImg2 = uint8(zeros(dstWidth, dstHeight,size(srcImg, 3)));

% 双线性插值
for i = 1:dstHeight % 遍历目标图像的每个像素
    for j = 1:dstWidth
        % 找到原图像中最近的四个像素坐标
        % 得到Q11的坐标点
        x = floor(i / T); % 目标像素在原图像中的行坐标
        y = floor(j / T); % 目标像素在原图像中的列坐标

        % 计算目标像素在最近邻四个像素中的位置权重
        % i/T和j/T为待求像素的位置x
        dx = i / T - x;
        dy = j / T - y;

        % 超出边界的像素，使用最近边界像素进行填充
        if x < 1
            x = 1;
            dx = 0;
        elseif x >= srcHeight
            x = srcHeight - 1;
            dx = 1;
        end

        if y < 1
            y = 1;
            dy = 0;
        elseif y >= srcWidth
            y = srcWidth - 1;
            dy = 1;
        end

        % 双线性插值计算
        A = double(srcImg(x, y,:)); % 最近邻像素Q11
```

```
        B = double(srcImg(x, y+1,:)); % 最近邻像素Q12
        C = double(srcImg(x+1, y,:)); % 最近邻像素Q21
        D = double(srcImg(x+1, y+1,:)); % 最近邻像素Q22

        fR1 = (1-dx)*A + dx*C; % 在x方向上进行线性插值
        fR2 = (1-dx)*B + dx*D; % 在x方向上进行线性插值
        fP = (1-dy)*fR1 + dy*fR2; % 在y方向上进行线性插值
        dstImg2(i, j,:) = fP; % 将插值后的像素值赋给目标像素
    end
end

figure;
subplot(1,2,1),imshow(srcImg),title('原图');
subplot(1,2,2),imshow(dstImg2),title('双线性插值');
```

3.3 双三次插值法(bicubic)

代码的核心思想是，对于原图和放大倍数，调用双三次插值函数，先生成空的目标图像，对于目标图像的每个像素，分别进行：

1. 找到原图像中最近的那个像素坐标
2. 进行检查，对于超出边界的像素，使用最近边界像素进行填充
3. 调用权重函数计算权重并进行插值
4. 将插值后的像素赋给目标像素

插值算法以及权重函数均基于**2.3**中的公式。

```
clear
clc

% 读入待处理的图像
srcImg = imread('input.bmp');

% 设置放大倍数
T = 3;

% 双三次插值
dstImg3 = bicubic_interpolation(srcImg, T);

figure;
subplot(1, 2, 1), imshow(srcImg), title('原图');
subplot(1, 2, 2), imshow(dstImg3), title('双三次插值法');

function  res = W(x)
%% 权重函数
a = -0.5; %一般是-1或者-0.5
if abs(x)<=1
    res = (a+2)*abs(x)^3 - (a+3)*abs(x)^2 + 1;
elseif abs(x)>1 && abs(x)<2
    res = a*abs(x)^3 - 5*a*abs(x)^2 + 8*a*abs(x) - 4*a;
```

```
else
    res = 0.;
end
end

function dstImg3 = bicubic_interpolation(srcImg, T)
%% 双三次插值法
[srcHeight, srcWidth,~] = size(srcImg);
% 新图像的大小
dstHeight = round(srcHeight*T);
dstWidth = round(srcWidth*T);

% 创建新图像的矩阵
dstImg3 = uint8(zeros(dstHeight, dstWidth,size(srcImg, 3)));

% 双三次插值法插值
for i = 1:dstHeight
    for j = 1:dstWidth
        % 找到原图像中最近的那个像素坐标
        x = ceil(i / T);
        y = floor(j / T);

        % 超出边界的像素，使用最近边界像素进行填充
        if x < 2
            x = 2;
        elseif x > srcHeight-2
            x = srcHeight-2;
        end

        if y < 2
            y = 2;
        elseif y > srcWidth-2
            y = srcWidth-2;
        end

        BXY = 0;
        % 进行插值
        for m = -1:1:2  % 横坐标
            for n = -1:1:2  %纵坐标
                % 计算权重
                weight_x = W(x+m-i/T);
                weight_y = W(y+n-j/T);
                % 计算插值
                BXY = BXY + double(srcImg(x+m,y+n,:))*weight_x*weight_y;
            end
        end
        dstImg3(i,j,:) = uint8(BXY);
    end
end
end
```

3.4 基于字典的超分辨率重建

基于字典的超分辨率重建的步骤较为复杂，下面我将分步骤进行分析

```
clear
clc

%读入低分辨率图
srcImg = imread('input.bmp');

%设置放大倍数
T = 3;
```

1. 载入字典获取 Dl，Dh，对 Dl 归一化

   ```
   load('D_1024_0.15_5.mat');
   Dl = Dl ./ sqrt(sum(Dl.^2,1));
   ```

2. 获取特征块大小 patch_size(sqrt(size(Dh, 1)))，自定义重叠域 overlap(也可以理解为步长)，超分系数 lambda(这里我们取 0.2)

   ```
   patch_size = sqrt(size(Dh, 1));
   overlap = 3;
   lambda = 0.2;
   ```

3. 利用插值法，把低分辨率图变大(与目标高分辨率图大小一致)，这里使用 imresize ，超分操作只对 Y 域操作，所以这里将低分辨率图从 RGB 域转换到 YCbCr 域。

   ```
   im_l = imresize(srcImg, T);
   im_l_ycbcr = rgb2ycbcr(im_l);
   im_l_y = im_l_ycbcr(:,:,1);
   im_l_cb = im_l_ycbcr(:,:,2);
   im_l_cr = im_l_ycbcr(:,:,3);
   ```

   `rgb2ycbcr()` : 将彩色图由 rgb 转换为 Y Cb Cr 彩色域

4. 提取 resize 后的低分辨率图特征，一共有四层特征

   ```
   % 第一层和第二层采用一阶导算子
   h1 = [-1, 0, 1];
   h2 = [-1; 0; 1];
   img_c1 = conv2(im_l_y, h1,'full');
   img_c2 = conv2(im_l_y, h2,'full');
   % 第三层和第四层采用二阶导算子
   h3 = [1, 0, -2, 0, 1];
   h4 = [1; 0; -2; 0; 1];
   img_c3 = conv2(im_l_y, h3,'full');
   img_c4 = conv2(im_l_y, h4,'full');
   ```

5. 对每个特征块求最优高分辨率块：

   （1）计算 resize 后低分辨率图像块(5*5)均值 m_patch

   （2）找到对应位置的特征(554)向量，展开为一维向量(100*1)，并且归一化，得到 Fy

（3）利用 Dl，Fy，求得 A，b，代入函数求得该块的最优稀疏系数 a

（4）生成高分辨率图块 x = Dh * a，并将 x+m 加入高分辨率图像 im_h_y，并且用flag数组记录像素块相加的次数

```
[m, n] = size(im_l_y);
im_h_y = zeros([m, n]);
% 计算像素块加的次数
flag = zeros([m, n]);

for i = 1:overlap:(m - patch_size)
    for j = 1:overlap:(n - patch_size)
        % 计算图像块的均值 m
        idx_i = i: i + patch_size -1;
        idx_j = j: j + patch_size -1;
        patch = im_l_y(idx_i, idx_j);
        m_patch = mean(patch(:));

        % 找到对应位置的特征(5*5*4)向量，展开为一维向量(100*1)，并且归一化，得到 Fy
        sub_img_c1 = img_c1(idx_i, idx_j);
        sub_img_c2 = img_c2(idx_i, idx_j);
        sub_img_c3 = img_c3(idx_i, idx_j);
        sub_img_c4 = img_c4(idx_i, idx_j);
        Fy = [sub_img_c1(:); sub_img_c2(:); sub_img_c3(:); sub_img_c4(:)];
        Fy = Fy./norm(Fy);

        % 利用 Dl，Fy，求得 A，b，代入函数求得该块的最优稀疏系数 a
        A = Dl'*Dl;
        b = -Dl'*Fy;
        a = L1QP(lambda, A, b);
        x = Dh*a;
        im_h_y(idx_i, idx_j) = im_h_y(idx_i, idx_j)+ reshape(x+m_patch, [5 5]);
        flag(idx_i,idx_j) = flag(idx_i,idx_j) + 1;
    end
end
```

6. 解决目标图像的过亮问题

```
im_h_y = uint8(im_h_y./flag);
```

7. 生成重建后的图像

```
% 超分重建只对Y域操作，所以Cb和Cr域直接取resize后的值
im_h_cb = im_l_cb;
im_h_cr = im_l_cr;

im_h_ycbcr = cat(3, im_h_y,im_h_cb,im_l_cr);
im_h = ycbcr2rgb(uint8(im_h_ycbcr));

figure;
subplot(1,2,1),imshow(srcImg);
subplot(1,2,2),imshow(im_h);
```