


2. 习题

2.1 彩色空间

1. 实现超分辨率评价指标——峰值信噪比 PSNR

“PSNR(input_img1, input_img2)->output”

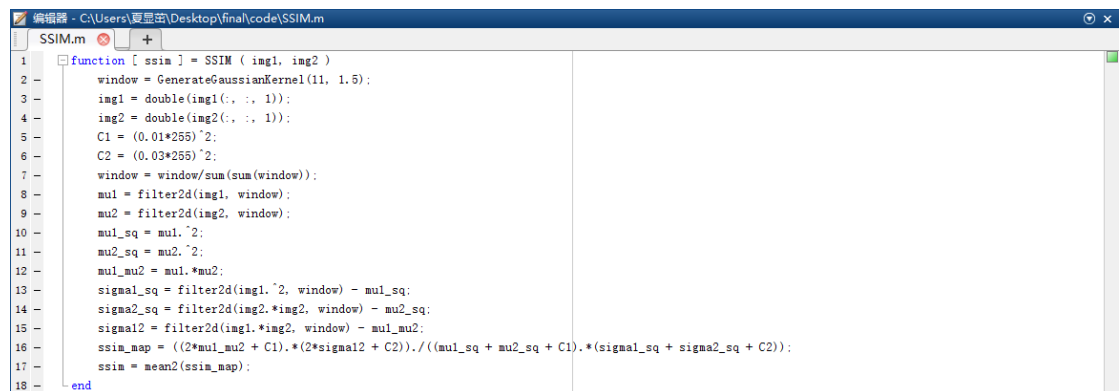


```

1 function [ psnr ] = PSNR( origin, target )
2     origin = double(origin);
3     target = double(target);
4     if ndims(origin) == 3
5         origin_ycber = rgb2ycbcr(origin);
6         target_ycber = rgb2ycbcr(target);
7         origin_y = origin_ycber(:, :, 1);
8         target_y = target_ycber(:, :, 1);
9     else
10        origin_y = origin;
11        target_y = target;
12    end
13    [height, width] = size(origin_y);
14    % MSE = sum(sum((origin(:,:)-target(:,:)).^2))/(height*width);
15    sum = 0.0;
16    for i = 1:height
17        for j = 1:width
18            sum = sum + (origin_y(i,j)-target_y(i,j))^2;
19        end
20    end
21    MSE = sum/(height*width);
22    psnr = 20*log10(255/sqrt(MSE));
23 end
24
  
```

2. 实现超分辨率的另一个评价指标——结构相似性指标

SSIM “SSIM(input_img1, input_img2)->output”



```

1 function [ ssim ] = SSIM ( img1, img2 )
2     window = GenerateGaussianKernel(11, 1.5);
3     img1 = double(img1(:, :, 1));
4     img2 = double(img2(:, :, 1));
5     C1 = (0.01*255)^2;
6     C2 = (0.03*255)^2;
7     window = window/sum(sum(window));
8     mu1 = filter2d(img1, window);
9     mu2 = filter2d(img2, window);
10    mu1_sq = mu1.^2;
11    mu2_sq = mu2.^2;
12    mu1_mu2 = mu1.*mu2;
13    sigma1_sq = filter2d(img1.^2, window) - mu1_sq;
14    sigma2_sq = filter2d(img2.^2, window) - mu2_sq;
15    sigma12 = filter2d(img1.*img2, window) - mu1_mu2;
16    ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq + C1).*(sigma1_sq + sigma2_sq + C2));
17    ssim = mean2(ssim_map);
18 end
  
```

实现思路：

对于 RGB 彩色图像，将其转为 double 类型并取其中的通道一，参考 ssim.m，为了解决简单加窗使映射矩阵出现不良的“分块”效应，使用了 11*11 的对称高斯加权函数，标准差为 1.5. 依照公式计算两张图像的平均值，方差和 X、Y 的协方差。

3. 实现基于双三次插值的图像缩放算法。

“bicubic(input_img, height, width)->output_img”

```
bicubic.m
1 function [retImage] = bicubic( origin, height, width )
2     extendOrigin = extending(origin);
3     [origin_height, origin_width] = size(origin);
4     factorX = origin_height/height;
5     factorY = origin_width/width;
6     retImage = double(zeros(height, width));
7     for x = 1:height
8         for y = 1:width
9             sum = 0.0;
10            targetX = x*factorX;
11            targetY = y*factorY;
12            intX = floor(targetX);
13            intY = floor(targetY);
14            W = getWeight(targetX, targetY);
15            for l = 1:4
16                for m = 1:4
17                    sum = sum + W(1, 1)*W(2, m)*extendOrigin(intX+l, intY+m);
18                end
19            end
20            retImage(x, y) = sum;
21        end
22    end
23    retImage = uint8(retImage);
24 end
```

实现思路：

先计算放缩比，通过放缩比依次寻找插值点的 16 个邻近点（邻居）（x,y 为插值点向下取整的坐标，选取的邻域为[x:x+3, y:y+3]），计算基于 x,y 两个维度上的距离的权重，计算最终的插值。

4. 用基于双三次插值的图像超分辨率算法进行实验

- 1) 对 Set14 的每一张图像 I_{HR} ，使用双三次插值，把原图缩小至原来尺寸的 1/3 得到图像 I_{LR} 。

详见 “./temp”

- 2) 再次将 I_{LR} 上采样至 I_{LR} 三倍尺寸的图像 I_{BI} 。

详见 “./Result”

- 3) 计算 $PSNR(I_{HR}, I_{BI})$ 和 $SSIM(I_{HR}, I_{BI})$,

详见表 1

Set14 images	双三次插值法		基本任务超分辨率算法		
	PSNR	SSIM	PSNR	SSIM	Time(s)
baboon	21.586711	0.463441	18.051800	0.248449	9.073947
barbara	24.723661	0.718141	20.531612	0.511085	15.405513
bridge	23.355589	0.649869	17.256735	0.221943	11.355301
coastguard	24.926268	0.589276	21.219815	0.373572	4.468837
comic	22.584740	0.708283	16.991767	0.342813	3.670554
face	31.632029	0.655763	26.922044	0.521064	2.957306
flowers	26.744609	0.787436	19.111679	0.418685	6.547487
foreman	27.478892	0.874750	23.60584	0.643542	3.748863
lenna	31.383253	0.830698	21.920823	0.521103	9.125845
man	26.223038	0.723813	19.627069	0.330622	9.933277
monarch	29.122786	0.915715	21.589566	0.694704	9.781736
pepper	29.459945	0.740192	21.905474	0.465524	9.030580
ppt3	22.993149	0.879396	17.012365	0.595418	8.804964
zebra	26.590499	0.797895	18.314880	0.407434	9.471596
average	26.343226	0.738191	20.290105	0.449711	8.098272

表 1: 不同算法在 Set14 测试集上的 PSNR, SSIM 以及运行时间

5. 阅读并实现论文 1（ICCV2013）的超分辨率算法。对步骤 4 中得到的 I_{LR} ，通过超分辨率算法得到图像 I_{SR} ，并计算出 PSNR 和 SSIM，同时将每张图片进行超分辨率运算的时间填写在表格 1 中。

实现思路：

根据论文思路，大体思路和流程如下。

1. 对训练集中的每张图像，先做一个大小为 11×11 ，标准差为 1.6 的高斯模糊，并且缩小为原来的 $1/3$ 得到 LR 图片集。分割 LR 图片集中的每张图像，每张图像都能被分割成若干个 patch（1 个 7×7 ）的矩形块（其中平滑的矩形块被忽略）并同时记录下所有 patch 的中心位置 `table_position_center`，对应的 feature（45 个 pixel）以及该 patch 的均值 `patch_mean`。
2. 随机选择若干个（经过比较，最终选择了 30W）用来训练的 feature，并将它们整理到一起并记录以供步骤 3 使用。
3. 使用步骤 2 中的 feature 集合，运行 kmeans，聚类数量为 2048 个聚类中心并记录。
4. 根据步骤 3 得到的 2048 个聚类中心，将步骤 1 中的所有 feature 进行标记（将与聚类中心欧氏距离最小的聚类中心编号作为标记值）并记录。
5. 通过步骤 1 中所记录的 patch 中心位置，每个位置对应的 feature，`patch_mean` 以及步骤 4 中所记录的每个 feature 的标记，对每个聚类中心，训练从 `[feature - patch_mean; 1]` 到 `[feature_hr]` 的映射关系（一个 81×46 的矩阵，因为缩放比 `sf` 为 3，所以 `featurelength_hr = (3 \times 3)^2 = 81`）。
6. 根据步骤 5 的映射矩阵，得到 HR 图像。

运行在 Windows10 上，matlabR2016a 版本上