图像超分辨率 project 报告

15331354 杨加佳

2.1

文档要求里有给出公式，这一题函数的实现照公式打即可实现，不多做赘述。

$$MSE(X,Y) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[X(i,j) - Y(i,j)]^2$$

$$PSNR(X,Y) = 20 \cdot \log_{10}(\frac{MAX_I}{\sqrt{MSE(X,Y)}})$$

需要注意的是对于彩色图像用 Y 通道进行计算。

2.2

这一题若按所给公式，然后用一般数学方法算方差以及协方差的话，再套公式得到的 SSIM 值会很高（都在 0.8 以上）。查阅文档给出的参考网页，发现需要自己生成一个高斯矩阵，用高斯矩阵来进行方差以及协方差的运算，这样得到的 SSIM 值就比较符合要求了。

高斯矩阵的生成代码如下：

```
w = zeros(11, 11);

center = (11 - 1) / 2;
for i = 1 : 11
    for j = 1 : 11
        w(i,j) = exp(-((i - center)^2 + (j - center)^2) /(2 * 1.5^2)) / (2 * pi * 1.5^2);
    end
end
w = w/sum(sum(w));
```

2.3

双三次插值算法的实现也比较简单，文档已给出公式，直接使用即可。

我在这里用了矩阵的乘法来让对应关系看起来更明了。

```
A = [W(1 + u) W(u) W(1 - u) W(2 - u)];
C = [W(1 + v);W(v);W(1 - v);W(2 - v)];
B = [f(si - 1, sj - 1) f(si - 1, sj) f(si - 1, sj + 1) f(si - 1, sj + 2);
     f(si, sj - 1) f(si, sj) f(si, sj + 1) f(si, sj + 2);
     f(si + 1, sj - 1) f(si + 1, sj) f(si + 1, sj + 1) f(si + 1, sj + 2);
     f(si + 2, sj - 1) f(si + 2, sj) f(si + 2, sj + 1) f(si + 2, sj + 2);];

output_img(i,j) = (A*B*C);
```

2.4

运行 main（）函数，得到各个图对应的 PSNR 以及 SSIM 值如下：

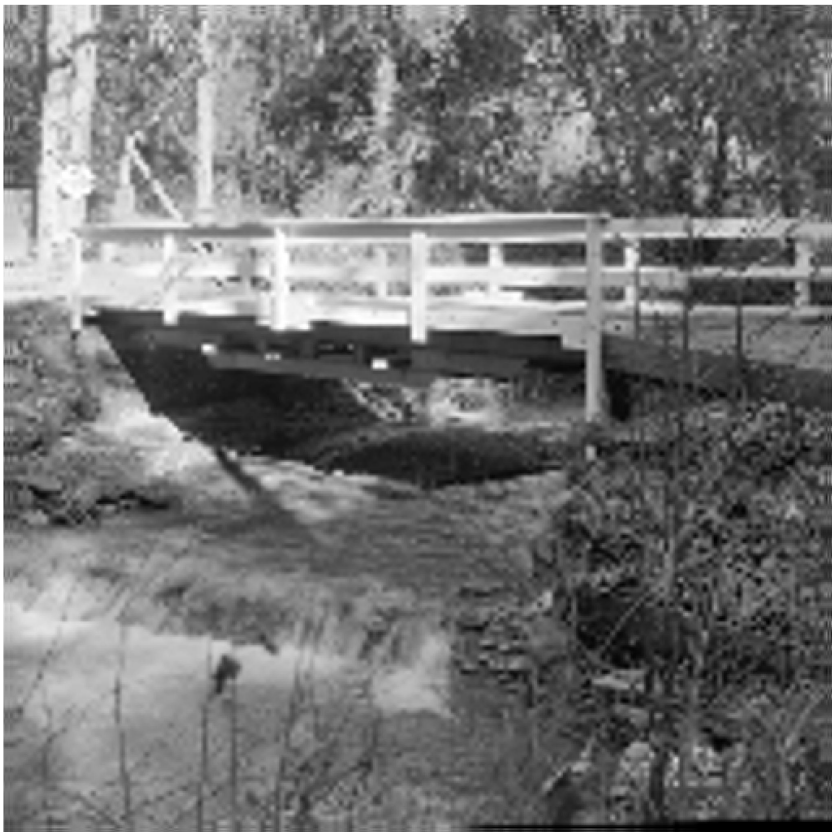
图片	PSNR	SSIM	论文 PSNR	论文 SSIM
baboon	21.4279	0.5382	23.21	
barbara	24.4006	0.7540	26.25	
bridge	23.0397	0.6548	24.40	
coastguard	23.0119	0.6088	26.55	
comic	21.8729	0.7157	23.12	
face	31.1957	0.7751	32.82	
flowers	26.5945	0.8104	27.23	
foreman	22.9440	0.8948	31.18	
lenna	31.1045	0.8551	31.68	
man	25.8659	0.7483	27.01	
monarch	28.9624	0.9203	29.43	
pepper	29.0580	0.8616	32.39	
ppt3	23.0983	0.8922	23.71	
zebra	26.3828	0.8065	26.63	
均值	25.6339	0.7740	27.54	0.7736

结果分析：可见自己实现的双三次插值的 PSNR 值与论文还是有一点差距的，但 SSIM 值与论文提供的均值基本接近。

部分双三次插值效果如下：

原图

下采样图



上采样图

原图

下采样图

上采样图

2.5

实现的整体思路

训练阶段：

1. 训练聚类中心

将训练集的 HR 图转换成训练用的 LR 图，对 LR 图切块并提取特征，对得到的 LR 块特征进行聚类操作得到每一个类的中心；

2. 训练系数矩阵

通过先对多组对应的 HR 块的特征和 LR 块特征分类，然后对于每一类的 HR 块特征和 LR 块特征，求出他们之间的系数矩阵。

还原阶段：

对一个待还原的 LR 图，先高斯模糊然后切块，对每一个块，通过训练得到的中心判断其所属的类，然后找到该类所对应的训练得到的系数矩阵，将 LR 块映射为对应的 HR 块，然后将得到的 HR 块拼接起来得到超分辨率图像。

实验过程

1）需要实现的函数

1. HR 图到 LR 图的转换

依据论文的算法，需要先对 HR 图高斯模糊然后下采样；

2. 切块

依据论文的算法，需要对图片进行切块、去角这两个操作；

3. 提取特征

提取训练集低分辨率图的特征用于做聚类；

对两张对应的低分辨率和高分辨率图提取特征训练系数矩阵；

对一张新的低分辨率图片做超分辨率操作时，也需要提取其特征，用作映射操作的输入；

4. 聚类操作

将提取到的低分辨率图的特征进行归类；

5. 训练系数矩阵

按论文给出的思路，通过多个对应的低分辨率图片的特征与高分辨率图片的特征，按照聚类得到的分类，求得每个类对应的系数矩阵；

6. 得到超分辨率图片

通过训练的对应关系，小的低分辨率块经过系数矩阵的映射得到预测的大的块，并将其拼接起来得到超分辨率图片。

2）实现过程

1. HR 图到 LR 图的转换（函数 lr = HR_To_LR（hr, sigma））

输入为 HR 图，以及高斯矩阵的 sigma 参数（缩放比默认为 3），输出为用于训练的 LR 图。

先对 HR 图进行裁剪，使其长宽都能被 3 整除，然后用 sigma 生成高斯矩阵，对裁剪后的 HR 图进行模糊，这里的滤波操作中，对边界的扩展为复制边界，且滤波结果图大小与原 HR 图大小一致。

```
maps = padarray(f, [bias, bias], 'replicate', 'both');
```

下采样操作使用 2.3 中自己实现的双三次插值算法。

```
lr = bicubic(temp_img, lr_height, lr_width);
```

2. 切块（patch = Cutting（src, patch_size））

论文给出的例子是缩放比为 4 的时候的，当缩放比变为 3 时要做出如下的改变：

对于低分辨率的图，每隔一个像素切一次块，每个块大小为 7×7，对于对应的高分辨率的图，每隔三个像素切一次块，每个块大小为 21×21，这样便能保证其对应关系，即对于同一张图，切出来低分辨率的块和高分辨率的块的数量是一样的（切块前，需保证其宽、高能被 3 整除）。

src 为输入的图片，patch_size 为切成的块的大小，通过这个值来判断输入的是 HR 图还是 LR 图。对切得的块，转换成行向量存到返回值 patch 里。

```
temp_patch = reshape(temp_patch, 1, patch_size * patch_size);
patch(flag, :) = temp_patch;
```

即返回的 patch 每一行为一个块。

3. 提取特征（feature = Get_Feature（patch））

因为切块时没包含去角的操作，所以对于 LR 块，提取特征时需要去掉 4 个角，即只提取 45 个像素点的特征。

```
lr_cut = [2:6 8:42 44:48];
```

LR 块每个像素点的特征为该像素点减去当前小块 45 个像素点的均值。

对于 HR 块，则只提取中心 9×9 区域的特征。特征为当前像素点减去对应的低分辨率块的 45 个像素点的均值；

输入 patch 为保存 HR 块或 LR 块的矩阵，每一行为一个小块，通过矩阵列数判断是 HR 块还是 LR 块。返回的 feature 矩阵每一行为一个块的 feature。

4. 聚类操作（Clusting（feature））

用 Matlab 自带的 kmeans 函数进行聚类操作即可，返回一个记录每个行向量所属类的 id 的数组以及一个记录每个类中心位置的数组。feature 为训练用的 LR 块的特征。

实验所用的为顺序提取的 20 万样本做 512 类聚类的操作。

为节省测试的时间，不必每测试一张图片就进行一次聚类操作，所以将得到的每个类中心位置的数组保存为 center.mat 文件留作后续使用。

5. 训练系数矩阵（Get_Coef_Matrix（））

用 200 万组顺序获得的对应的 LR 块和 HR 块，首先提取 LR 块和 HR 块的特征，然后根据 LR 块的特征以及第四步得到的每个类的中心位置，对每个 LR 块及其对应的 HR 块进行分类。

```
for i = 1 : 2000000
    lr_patch = lr_features(i, :);
    temp = repmat(lr_patch, 512, 1);
    diff = temp - C;
    diff = diff.^2;
    distance = sum(diff, 2);
    [~, id] = min(distance);
    idx(i, 1) = id;
end
```

然后对于每一类，将该类的 LR 块拼成一个矩阵以及将对应的 HR 块拼成另一个矩阵，两个矩阵相除，求得该类的系数矩阵。

```
for i = 1  : 512
    Coef(:, :, i) = hr_features(idx == i, :)'/lr_features(idx == i, :)';
end
```

为了节省测试时间，不必每测试一张图片都进行系数矩阵的训练，训练一次后便将系数矩阵保存为文件 coef.mat 留作后续使用。

6. 得到超分辨率图片（hr = Generate_HR（ lr））

对于一个输入的已模糊的低分辨率图片 LR，先对其切块、提取特征，然后根据特征，判断该块所属类的 id，然后用特征矩阵乘以该类对应的系数矩阵，得到对应的超分辨率块，最后将这些超分辨率块有重叠地“铺”在一起，因为得到的超分辨率块是有重叠的，所以用一个矩阵来记录每个像素叠加的次数，最后除以这个图片便可得到超分辨率图片 HR。

```
hr(rh : rh1, ch : ch1) = hr(rh : rh1, ch : ch1) + hr_patch;
count(rh : rh1, ch : ch1) = count(rh : rh1, ch : ch1) + 1;

hr = hr ./ count;
```

需要注意的是，若 LR 到 HR 按 45 到 81 的映射，得到的图片是缺了一圈边界的，所以在切块前，对 LR 图进行 2 个像素的复制边界扩展，这样映射出来的 HR 块就会相应变大一圈，解决了边界的问题。

```
lr = padarray(lr, [2,2], 'replicate', 'both');
```

实验结果与分析

部分结果如下：

原 HR 图（未模糊）

对模糊后 LR 图做超分辨率结果

原 HR 图（无模糊）

对模糊后 LR 图做超分辨率结果

实验过程中，训练聚类中心所用时间为 587.020581 秒，训练系数矩阵所用时间为 657.963764 秒。

下表为按论文超分辨率算法得到的结果与双三次插值法得到的结果及论文中 PSNR 与 SSIM 值的比较。其中每张图的处理时间为直接运用训练得到的聚类中心以及系数矩阵进行超分辨率操作的时间。

图片	双三次插值算法 PSNR	论文 PSNR	超分辨率算法 PSNR	双三次插值算法 SSIM	超分辨率算法 SSIM	超分辨率算法时间（秒）
baboon	21.4279	23.21	23.8745	0.5382	0.6217	14.975284
barbara	24.4006	26.25	26.7352	0.7540	0.7833	24.869068
bridge	23.0397	24.40	25.8531	0.6548	0.7401	8.091725
coastguard	23.0119	26.55	26.9544	0.6088	0.6625	7.612900
comic	21.8729	23.12	24.8496	0.7157	0.7879	6.841352
face	31.1957	32.82	33.5023	0.7751	0.8212	5.440429
flowers	26.5945	27.23	29.0763	0.8104	0.8575	12.602729
foreman	22.9440	31.18	29.7569	0.8948	0.9239	7.093956
lenna	31.1045	31.68	34.0102	0.8551	0.9053	16.394428
man	25.8659	27.01	28.9449	0.7483	0.8249	16.657335
monarch	28.9624	29.43	31.2787	0.9203	0.9384	24.815694
pepper	29.0580	32.39	34.7038	0.8616	0.9182	16.314559
ppt3	23.0983	23.71	25.9437	0.8922	0.9222	21.029129
zebra	26.3828	26.63	29.0129	0.8065	0.8565	14.259912
均值	25.6339	27.54	28.8926	0.7740	0.8260	14.071321

结果分析：可见用超分辨率算法得到的图片的 PSNR 值与 SSIM 较双三次插值算法都有不小的提升，且已经超过了论文中的 PSNR 值，说明对于论文中的超分辨率算法的实现是基本正确且达到预期的。

缺点及改进方案

由于本次实验的时间有限，且过程中出现了不少理解上的偏差，所以代码还有许多不足的地方：

1）由于开始时对于论文方法的理解偏差，在代码结构组织方面可能有一些不够科学。开始时以为求 HR 的 feature 的时候减的是 HR 块的均值，所以在分配代码结构的时候，提取特征和切块，对 LR 和 HR 是分开的，后面发现是减对应 LR 块的均值后，只能在训练矩阵的时候添加减均值的代码，而在调用 Get_Feature（）对 HR 块进行处理时，只是做一个简单的取中心块操作。这里提出来也是做一个说明；

2）在选取训练用的块时，为了减少花费的时间，用的是顺序的方法，导致缺乏随机性，相近的块的数量比较多，导致训练矩阵的时候出现了很多秩不足的情况。

如果能在选块的时候做到随机，减少相似块的数量，相信会对结果有一定的提升；

3）由于电脑配置的限制，内存不允许进行太大量的运算，所以实验中的样本数量参数基本已经是当前配置下效果最好的极限数量了。如果能够加大样本数量，得到的结果肯定会更好；

4）在做 PSNR 和 SSIM 比较时，没有裁剪边界，若裁剪边界，结果也会有一定提升。因为没裁剪就已经超过了双三次插值的效果，所以也就没裁了。

5）在超分辨率算法里的上下采样都用的自己实现的双三次，如果用 Matlab 自带的双三次的话效果会好一些。