**数字图像处理第六次实验报告**

PB16050141

高松群

霍夫曼编码：

1. 原理：

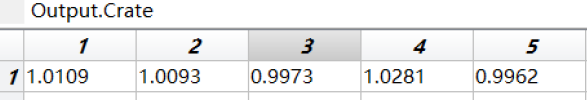
在[计算机](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA" \o "计算机)[数据处理](https://zh.wikipedia.org/w/index.php?title=%E8%B3%87%E6%96%99%E8%99%95%E7%90%86&action=edit&redlink=1)中，霍夫曼编码使用[变长编码表](https://zh.wikipedia.org/w/index.php?title=%E8%AE%8A%E9%95%B7%E7%B7%A8%E7%A2%BC%E8%A1%A8&action=edit&redlink=1)对源符号（如文件中的一个字母）进行编码，其中[变长编码表](https://zh.wikipedia.org/w/index.php?title=%E8%AE%8A%E9%95%B7%E7%B7%A8%E7%A2%BC%E8%A1%A8&action=edit&redlink=1)是通过一种评估来源符号出现几率的方法得到的，出现几率高的字母使用较短的编码，反之出现几率低的则使用较长的编码，这便使编码之后的字符串的平均长度、[期望值](https://zh.wikipedia.org/wiki/%E6%9C%9F%E6%9C%9B%E5%80%BC)降低，从而达到[无损压缩](https://zh.wikipedia.org/wiki/%E6%97%A0%E6%8D%9F%E5%8E%8B%E7%BC%A9)数据的目的。

1. 霍夫曼编码：
   1. 首先统计100张测试图像的灰度图，得到平均的灰度概率分布
   2. 记录每一个灰度对应的编码
   3. 当前概率最小的两个索引在其前面添加 0 和 1
   4. 将该两个索引对应的灰度集合合并，并将两个索引指向的两个概率相加，变成新的概率，参与到下一轮的运算中
   5. 将两个旧的概率所对应的灰度删去
   6. 重复步骤3-7，直到剩下一个概率1
2. 实验结果
   1. 100张图片进行训练，得到灰度图的霍夫曼编码结果

可以看出：ave平均码长为7.9619 < 8，达到了压缩效果。

之所以非常靠近8的原因是，100张训练图片太过掺杂，导致最后灰度概率大约为均匀分布，在出现概率最大的字符进行编码时，并没有实现压缩特别多的空间。

* 1. 压缩比：



如3与5的压缩比略小于1，而其余的压缩比则大于1，这是由于在该特定图片中，灰度概率分布并不很好的契合平均概率分布。利用其它图片的概率作为基准去压缩其他图片不一定是一个非常好的方法，个人认为应该将图片进行预处理分类，然后再对一类进行特定编码，这样的压缩效率会更高。