

图像视频大作业一 Part I

2014011355 辛杭高

2017 年 5 月 29 日

本部分的主要工作是对 lena.bmp 进行灰度变换和 DCT 变换，并分析几种 DCT 变换的区别，用 PSNR 度量变换前后的影响。

1 Usage

本部分实验全部由 matlab 代码编写完成，直接运行 src 文件夹下的 exp1.m 文件即可，运行时必须保证 lena.bmp 也在 src 文件夹下，运行完毕后会输出一系列参数，并保存一系列图片。

输出参数中，首先会输出 3 种 DCT 变换方法对应的时间和恢复前后的 PSNR 度量值，并且 3 种 DCT 变换恢复后的结果会以图片的格式存储下来，如 recover2DCT.bmp 表示二维 DCT 变换后恢复的结果（即 matlab 中的 dct2 命令）。随后会将原有的参数进行抽取，仅仅保留三种方法中参数的 $\frac{1}{4}$ ， $\frac{1}{16}$ ， $\frac{1}{64}$ ，并利用剩余参数进行恢复。此时会输出在此三种比例下恢复结果的 PSNR 度量值，也会将恢复的结果以图片的形式保留下来，如 'max_1D_reserve_0.015625.bmp' 表示两次 1 维压缩后的参数保留 $\frac{1}{64}$ 恢复得到的结果，其中 max 表示选择 DCT 参数中前 K 大的参数进行保留。

2 灰度化

灰度化的原理为将 RGB 空间转化到 YIQ 空间，并且最终只保留亮度参数 Y 作为最终输出。转化公式为

$$Y = 0.3R + 0.59G + 0.11B \quad (1)$$

灰度化结果为



图 1: 原图



图 2: 灰度图

3 PSNR

PSNR 用来度量两张图的相似性，其计算方式为

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \quad (2)$$

$$MSE = \frac{\sum_{n=1}^{Framesize} (I_n - P_n)^2}{Framesize} \quad (3)$$

4 DCT 变换

4.1 两次一维 DCT 变换

一维 DCT 变换的原理如下，针对给定序列 $f(x)$ ，其 DCT 正变换和逆变换为

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) \cos \frac{(2x+1)u}{2N} \pi \quad (4)$$

$$f(x) = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} C(u) F(u) \cos \frac{(2x+1)u}{2N} \pi \quad (5)$$

在这种形式下，我们先对原图的行作 DCT 变换，再对第一步结果的列作 DCT 变换。



图 3: idct2 恢复两次一维 DCT 变换的结果.

对于一个长度为 N 的序列进行一次 DCT 变换需要 N^2 次运算, 对一个二维图像进行两次一维 DCT 变换, 即需要对 $2N$ 个序列进行 DCT 变换, 因此该方法的时间复杂度为

$$N^2 * 2N = 2N^3 = O(N^3) \quad (6)$$

在 matlab 中, 这个命令实际的操作时间为 0.034277 秒。

使用 idct2 对使用本方法 DCT 后的参数进行恢复, 得到的恢复图为此恢复图像与原图相比的 PSNR 度量结果为

$$PSNR_{1DCT} = 312.8633 \quad (7)$$

4.2 二维 DCT 变换

二维 DCT 变换的原理如下, 针对给定序列 $f(i,j)$, 其 DCT 正变换和逆变换如下



图 4: idct2 恢复二维 DCT 变换的结果.

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos \frac{(2i+1)u}{2N} \pi \cos \frac{(2j+1)v}{2N} \pi \quad (8)$$

$$f(i, j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) C(v) F(u, v) \cos \frac{(2i+1)u}{2N} \pi \cos \frac{(2j+1)v}{2N} \pi \quad (9)$$

由上述公式可以看出, 计算 $F(u, v)$ 中每一个元素需要 N^2 次运算, 整张图有 N^2 个元素, 因此该方法的时间复杂度为

$$N^2 * N^2 = N^4 = O(N^4) \quad (10)$$

在 matlab 中, 这个命令实际的操作时间为 0.026376 秒。

使用 idct2 对使用本方法 DCT 后的参数进行恢复, 得到的恢复图为



图 5: idct2 恢复分块二维 DCT 变换的结果.

该恢复图像与原图相比的 PSNR 度量结果为

$$PSNR_{2DCT} = 312.7920 \quad (11)$$

4.3 分块二维 DCT 变换

分块 DCT 变换的思路是, 首先将原图划分为若干个 $k * k$ 的小块, 对每个小块进行二维 DCT 变换, 将各小块 DCT 的结果拼合成一个原图规模的矩阵, 即为此方法下 DCT 的结果。

在这一方法下, 我们仍有 N^2 个 DCT 元素需要计算, 但是计算每个元素时只需要 k^2 次运算, 因此该方法的时间复杂度为

$$N^2 * k^2 = N^2 k^2 = O(N^2 k^2) \quad (12)$$

在 matlab 中, 这个命令实际的操作时间为 0.554612 秒。

使用 idct2 对使用本方法 DCT 后的参数进行恢复, 得到的恢复图为该恢复图像与原图相比的 PSNR 度量结果为

$$PSNR_{8*82DCT} = 313.8932 \quad (13)$$

度量方法	一维 DCT	二维 DCT	分块二维 DCT
PSNR	312.8633	312.7920	313.8932
时间复杂度	$2N^3$	N^4	N^2k^2
实际耗时 (s)	0.034277	0.026376	0.554612

表 1: 各方法的时间以及性能比较.

5 DCT 变换的时间复杂度和 PSNR

从表格 1 中可以看出, 三种方法在 PSNR 度量下的效果几乎一致。实际上二维 DCT 应该最能表达整张图象在二维频域下的信息, 但是其他两种方法也能在一定程度上表明该图象的频域特征, 另一方面由于我们采用了全部的 DCT 信息进行恢复, 在这种情况下对三种方法的结果要求不是很高, 在下一节中, 仅仅保留部分 DCT 参数来恢复图像, 这就为 DCT 参数的质量提出了更高的要求。

另一个值得关注的问题是, 算法的时间复杂度与实际运行时间并不一致。导致这一原因的是 matlab 的一些特性, 在计算分块二维 DCT 时, 由于我们需要手动遍历各个区域块, 而 matlab 对循环操作非常不友好, 会导致循环的速度很慢, 也就造成了二维 DCT 耗时最长的特点。二维 DCT 快于一维 DCT 主要是因为 matlab 对于矩阵的运算更加亲近, 会快于对矩阵中向量的操作, 而且在 matlab 里从一个矩阵中提取向量的效率也不够可观, 所以造成了现在我们看到的实际运行时间。

从理论层面上来说, 二维 DCT 耗时最久, 一般来说使用分块二维 DCT 更为合理, 可以根据自己的需要调节 k 的大小, 显然当 $k = N$ 时, 分块二维 DCT 会退化为二维 DCT。

6 保留部分参数

出于节省空间的目的, 我们尝试仅仅保留 3 种方法 DCT 参数的 $\frac{1}{4}$, $\frac{1}{16}$, $\frac{1}{64}$ 来恢复原图像, 比较不同方法下的结果, 并分析原因。

首先, 我们面临的第一个问题是如何选取要保留的 DCT 参数, 我提供了两种实现, 分别对应于 src 文件夹下的 choose_first_coe.m 和 choose_max_coe.m。一种方法是基于人眼的特征来考虑, 人眼对于高频分量不够敏感, 对于低频分量更加敏感, 因此我们保留一张图片 DCT 参数的左上角。同时, 我们若





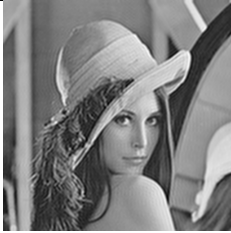




保留比例	一维 DCT	二维 DCT	分块二维 DCT
$\frac{1}{4}$			
$\frac{1}{16}$			
$\frac{1}{64}$			

表 2: 保留左上角参数恢复后图片比较.

将图片的 DCT 分量打印出来, 可以发现图片的左上角以外的区域 DCT 分量几乎为 0, 这是保留左上角 DCT 分量的另一个依据。另一种方法是保留 DCT 分量中较大的那些, 对 DCT 分量的绝对值从大到小进行排序, 保留排序后结果最大的前 x 个分量。

如表 2 和表 3 所示, 一维 DCT 的效果与二维 DCT 的效果在保留左上角参数时基本相同, 这两个方法的结果优于 $8*8$ 分块的 DCT 方法, 从图象上来看, 当删除元素的比例增加时 $8*8$ 分块的方法开始出现锯齿, 而前两个方法并没有出现锯齿现象。另一方面 $8*8$ 分块的 DCT 方法对应的 PSNR 也更低一些。

另一方面, 我们对比选取最大元素保留的结果

对比表 3 和表 5 的结果, 保留前 K 大参数时, 可以看出一维 DCT 和二维 DCT 的效果基本相同, 分块二维 DCT 在仅仅保留 $\frac{1}{64}$ 比例的参数时

保留比例	一维 DCT	二维 DCT	分块二维 DCT
$\frac{1}{4}$	36.2345	36.2345	34.8839
$\frac{1}{16}$	29.9222	29.9222	28.2162
$\frac{1}{64}$	25.8642	25.8642	23.6717

表 3: 保留左上角参数恢复后的 PSNR 指标.





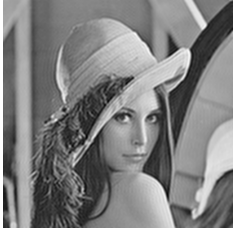

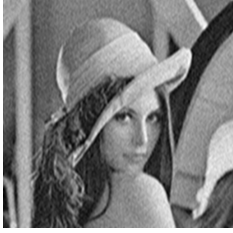
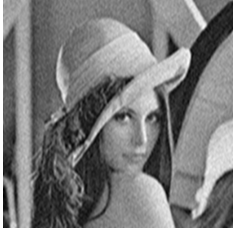

保留比例	一维 DCT	二维 DCT	分块二维 DCT
$\frac{1}{4}$			
$\frac{1}{16}$			
$\frac{1}{64}$			

表 4: 保留前 k 大参数恢复后图片比较.

保留比例	一维 DCT	二维 DCT	分块二维 DCT
$\frac{1}{4}$	40.0671	40.0671	40.3899
$\frac{1}{16}$	32.4749	32.4749	30.4301
$\frac{1}{64}$	27.7887	27.7887	23.6717

表 5: 保留前 k 大参数恢复后的 PSNR 指标.

会出现锯齿现象，而且 PSNR 指标会略微低一些。

但是在保留 $\frac{1}{16}$ 比例的参数时，一维 DCT 和分块二维 DCT 的效果相比于仅仅保留左上角要好一些，这是因为左上角的元素不一定是关键元素，也可能有 0 元素的干扰。保留前 k 大参数相较于保留左上角元素在 PSNR 的度量下效果会更好一些，但是保留前 k 大参数需要对元素进行排序，增大了时间复杂度，在实际应用中应该根据实际需求进行选择。