武漢大學

课 程 设 计 报 告

图像有损压缩 Matlab 仿真及性能测试

姓　　名：陈子昂, 朱鹤然, 卢意帆

学　　号：2021202120085

任课教师：茹国宝

学　　院：电子信息学院

专　　业：信息与通信工程

二〇二一年十一月

Chen Ziang, Zhu Heran, Lu Yifan

说　　明

目　　录

1 图像有损压缩技术的背景

1.1 图像压缩的必要性

图像数据量庞大，给存储和传输带来了许多困难。一张尺寸为 3840x2180 的原始图像，如果每个像素使用 32bit 来表示（RGBA），那么需要的内存为 3840x2180x4 = 33484800 Byte ≈31.9M。相应的，如果拍摄 1 min 30 fps 这样规格的 4k 视频，那么需要的存储空间将会达到 3840x2180x4x30x60 ≈ 56.1G！

数据压缩的目的就是通过去除这些数据冗余来减少表示数据所需的比特数。由于图像数据量的庞大, 在存储、传输、处理时非常困难, 因此图像数据的压缩就显得非常重要。信息时代带来了“信息爆炸”，使数据量大增，因此，无论传输或存储都需要对数据进行有效的压缩。在遥感技术中，各种航天探测器采用压缩编码技术，将获取的巨大信息送回地面。图像压缩是数据压缩技术在数字图像上的应用，它的目的是减少图像数据中的冗余信息从而用更加高效的格式存储和传输数据。

1.2 图像压缩的基本原理

图像数据之所以能被压缩，就是因为图像数据中存在着冗余部分。图像数据的冗余主要表现为 4 种：

- **空间冗余**。一幅图像表面上各像素点之间往往存在着空间连贯性，相邻像素之前也存在相关性，由此产生的空间冗余；
- **时间冗余**。视频的相邻帧往往包含相同的背景和移动物体，相邻帧之间存在相关性，由此产生的时间冗余；
- **频谱冗余**。不同彩色平面或频谱带之间存在相关，由此产生的频谱冗余；
- **视觉冗余**。人类的视觉系统由于受生理特性的限制，对于图像场的注意是非均匀的，人对细微的颜色差异感觉不明显。

2 JPEG 图像有损压缩

2.1 颜色空间转换与色度采样

2.1.1 颜色空间转换

需要将 RGB 颜色空间转化为 YUV 颜色空间，也叫 YCbCr，其中，Y 是亮度 (Luminance)，U 和 V 表示色度 (Chrominance) 和浓度 (Chroma)，UV 分量同时表示色差

做这一步的原因是对于人眼来说，图像中明暗的变化更容易被感知到，而对颜色的变化则没有那么敏感，将两者分开，就可以根据数据的重要程度的做不同的处理

研究表明，红绿蓝三基色所贡献的亮度不同，绿色所贡献亮度最多，蓝色所贡献亮度最少。假定红色贡献为 K_R，蓝色贡献为 K_B，则亮度可以表示为

$$Y = K_R \cdot R + (1 - K_R - K_B) \cdot G + K_B \cdot B \tag{2.1}$$

根据经验值 $K_R = 0.299, K_B = 0.114$，则有

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{2.2}$$

蓝色和红色的色差为

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\ C_b &= -0.1687 \cdot R - 0.3313 \cdot G + 0.5 \cdot B + 128 \\ C_r &= 0.5 \cdot R - 0.4187 \cdot G - 0.0813 \cdot B + 128 \end{aligned} \tag{2.3}$$

或

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.4}$$

2.1.2 色度采样

为了进一步压缩图像数据，JPEG 对色度图像二次采样。YUV 有三种采样方式：

4:4:4 采样：每一个 Y 对应一个 U 和一个 V。

4:2:2 采样：每两个 Y 共用一对 U 和 V。

4:2:0 采样：每四个 Y 共用一对 U 和 V。

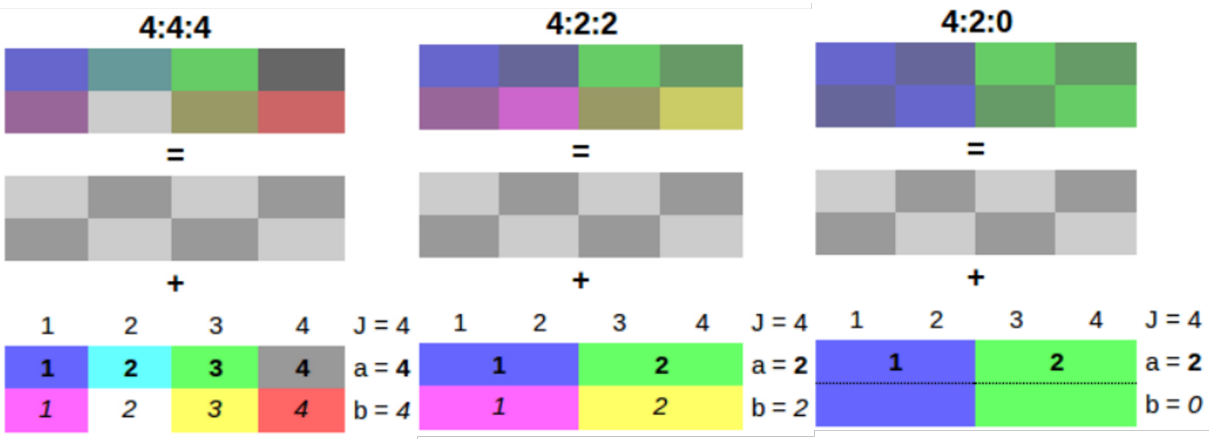

图 2.1　3 种色度采样示意

2.2　图像分块与 DCT 变换

2.2.1　图像分块

2.2.2　DCT 变换

一般的二维 DCT 变换

$$F(u,v)=c(u)c(v)\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}f(i,j)\cos\left(\frac{i+0.5}{M}u\pi\right)\cos\left(\frac{j+0.5}{N}u\pi\right)$$

$$c(u)=\begin{cases}\sqrt{\dfrac{1}{N}}, & u=0\\ \sqrt{\dfrac{2}{N}}, & u\neq 0\end{cases}\qquad u,v=0,1,2,...,7 \tag{2.5}$$

当 $M=N$ 时，DCT 变换可以表示为矩阵相乘的形式，F 的 DCT 变换则是 $T=AFA^T$。变换矩阵 A 为

$$A=\frac{2}{\sqrt{N}}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}}\\ \cos\frac{\pi}{2N} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{(2N-1)\pi}{2N}\\ \cdots & \cdots & & \cdots\\ \cos\frac{(N-1)\pi}{2N} & \cos\frac{3(N-1)\pi}{2N} & \cdots & \cos\frac{(2N-1)(N-1)\pi}{2N}\end{bmatrix} \tag{2.6}$$

当原始图像从 RGB 颜色空间转换到 YCbCr 颜色空间之后，需要对每一个 8×8 的图像块进行二维 DCT 变换

$$F(u,v)=c(u)c(v)\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\left(\frac{i+0.5}{8}u\pi\right)\cos\left(\frac{j+0.5}{8}u\pi\right)$$

$$c(u)=\begin{cases}\sqrt{\dfrac{1}{8}}, & u=0\\ \dfrac{1}{2}, & u\neq 0\end{cases}\qquad u,v=0,1,2,...,7 \tag{2.7}$$

这时候的 DCT 变换矩阵为

$$A=\frac{1}{\sqrt{2}}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}}\\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cdots & \cos\frac{(16-1)\pi}{16}\\ \cdots & \cdots & & \cdots\\ \cos\frac{(N-1)\pi}{16} & \cos\frac{3(N-1)\pi}{16} & \cdots & \cos\frac{(16-1)(N-1)\pi}{16}\end{bmatrix} \tag{2.8}$$

在 Matlab 中可以用 `T = dctmtx(8)` 查看

```
T =
  0.3536  0.3536  0.3536  0.3536  0.3536  0.3536  0.3536  0.3536
  0.4904  0.4157  0.2778  0.0975 -0.0975 -0.2778 -0.4157 -0.4904
  0.4619  0.1913 -0.1913 -0.4619 -0.4619 -0.1913  0.1913  0.4619
  0.4157 -0.0975 -0.4904 -0.2778  0.2778  0.4904  0.0975 -0.4157
  0.3536 -0.3536 -0.3536  0.3536  0.3536 -0.3536 -0.3536  0.3536
  0.2778 -0.4904  0.0975  0.4157 -0.4157 -0.0975  0.4904 -0.2778
  0.1913 -0.4619  0.4619 -0.1913 -0.1913  0.4619 -0.4619  0.1913
  0.0975 -0.2778  0.4157 -0.4904  0.4904 -0.4157  0.2778 -0.0975
```

对图像进行 8×8 分块后，对每一个矩阵块 A 都进行 DCT 变换 TAT^T

2.3 量化

2.4 熵编码

2.4.1 霍夫曼编码

霍夫曼编码是一种用于无损数据压缩的熵编码（权编码）算法，其使用变长编码表对数据进行编码。

变长编码表是通过评估符号出现概率得到的，出现概率高的符合使用较短的编码，反之出现概率低的则使用较长的编码，使编码之后的字符串的平均长度降低，从而达到无损压缩数据的目的。

霍夫曼编码主要分为两步：

根据每个字符出现的概率构建一颗最优二叉树；

再根据二叉树对字符进行编码。

3 图像有损压缩实验结果

3.1 实验结果的定性分析

3.2 实验结果的定量分析

3.2.1 性能指标

3.2.1.1 MSE

均方误差 (MSE) 定义为原图各像素 $I(i,j)$ 与压缩后图像各像素 $K(i,j)$ 差的平方和（式3.1）

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \tag{3.1}$$

3.2.1.2 PSNR

峰值信噪 (PSNR) 比定义为

$$PSNR = 20\lg(\frac{Max_1}{\sqrt{MSE}}) \tag{3.2}$$

其中 Max_1 是表示图像点颜色的最大数值，MSE 为均方误差。

3.2.1.3 SSIM

样本 (x,y) 的结构相似度为

$$SSIM(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2+\mu_y^2+C_1}\cdot\frac{2\delta_{xy}+C_2}{\delta_x^2+\delta_y^2+C_2} \tag{3.3}$$

3.2.1.4 压缩比

压缩比定义为原图片比特数与压缩后图片比特数之比。

3.2.2 定量结果

表 3.1 MSE

压缩质量	0.8	0.5	0.3	0.2	0.1
MSE	4.92	5.40	5.73	5.90	6.00

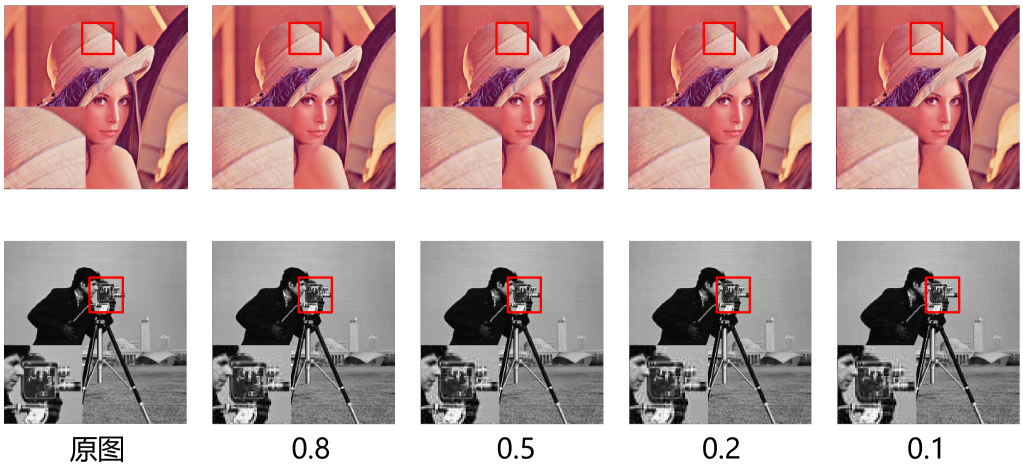

图 3.1　实验结果的定性分析

表 3.2　PSNR

压缩质量	0.8	0.5	0.3	0.2	0.1
PSNR	41.2	40.8	40.5	40.4	40.3

表 3.3　SSIM

压缩质量	0.8	0.5	0.3	0.2	0.1
PSNR	0.98	0.98	0.98	0.98	0.98

表 3.4　压缩比

压缩质量	0.8	0.5	0.3	0.2	0.1
游程压缩比	7.99	9.88	10.9	11.4	11.8
总压缩比	13.0	16.2	18.2	19.2	20.0
Matlab 压缩比	17.8	32.3	44.5	56.6	82.2

附录 A　Matlab 代码

A.1　DCT 分解

```
function out = func_dct(image)
    T = dctmtx(8); % 8*8DCT
    % dctmtx(N)可以生成N*N的DCT变换矩阵。
    % 对图像进行二维DCT变换有两种方法:
    %   1 直接使用dct2()函数。
    %   2 用dctmtx()获取DCT变换矩阵, 再T×A×T'。反变换T'×A×T
    func=@(block) T*block.data*T'
    for i=1:3
        out(:,:,i) = blockproc(image(:,:,i), [8 8], 'P1*x*P2',
           T, T');
        % blkproc函数为分块操作函数, 即对所有8*8的块执行DCT。
        % 对于不能整除的情况, 会自动用0填充右边缘和下边缘,
        % 在计算结束后自动删除它们
    end
end
```

```
function out = func_idct( im_dct)
    T = dctmtx(8);
    for i=1:3
        out(:,:,i) = blkproc(im_dct(:,:,i), [8 8], 'P1*x*P2',
           T', T);
    end
end
```

```
function  out = func_quantization(im_dct,quality_scale)
    % 使用JPEG2000推荐的标准亮度量化表和标准色差量化表进行量化
    %
        这里的quality_scale作为一个权重, 用来控制量化表中的参数, 输入范围是0到1
    %
        标准量化表中的数值越大->round(DCT结果/量化值)得到的值越小->更多的值接近
    % 实际量化表=标准量化表*(1-quality_scale+0.5)
        就可以控制实际用的量化表等于标准量化表的0.5到1.5倍
    Luminance_Quantization_Table = [
    16,  11,  10,  16,  24,  40,  51,  61;
  12,  12,  14,  19,  26,  58,  60,  55;
  14,  13,  16,  24,  40,  57,  69,  56;
  14,  17,  22,  29,  51,  87,  80,  62;
```

```
  18,  22,  37,  56,  68, 109, 103,  77;
  24,  35,  55,  64,  81, 104, 113,  92;
  49,  64,  78,  87, 103, 121, 120, 101;
  72,  92,  95,  98, 112, 100, 103,  99];
    Chrominance_Quantization_Table = [
  17,  18,  24,  47,  99,  99,  99,  99;
  18,  21,  26,  66,  99,  99,  99,  99;
  24,  26,  56,  99,  99,  99,  99,  99;
  47,  66,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99];

    Luminance_Quantization_Table = Luminance_Quantization_Table
       * (1 - quality_scale + 0.5);
    Chrominance_Quantization_Table =
       Chrominance_Quantization_Table * (1 - quality_scale +
       0.5);
    out(:,:,1) = blkproc(im_dct(:,:,1),[8 8], 'round(x./P1)'
       ,Luminance_Quantization_Table);
    out(:,:,2) = blkproc(im_dct(:,:,2),[8 8], 'round(x./P1)'
       ,Chrominance_Quantization_Table);
    out(:,:,3) = blkproc(im_dct(:,:,3),[8 8], 'round(x./P1)'
       ,Chrominance_Quantization_Table);
end
```

```
function out = func_iquantization(quantified, quality_scale)
    Luminance_Quantization_Table = [
    16,  11,  10,  16,  24,  40,  51,  61;
  12,  12,  14,  19,  26,  58,  60,  55;
  14,  13,  16,  24,  40,  57,  69,  56;
  14,  17,  22,  29,  51,  87,  80,  62;
  18,  22,  37,  56,  68, 109, 103,  77;
  24,  35,  55,  64,  81, 104, 113,  92;
  49,  64,  78,  87, 103, 121, 120, 101;
  72,  92,  95,  98, 112, 100, 103,  99];
    Chrominance_Quantization_Table = [
  17,  18,  24,  47,  99,  99,  99,  99;
  18,  21,  26,  66,  99,  99,  99,  99;
  24,  26,  56,  99,  99,  99,  99,  99;
  47,  66,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99;
```

```
  99,  99,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99;
  99,  99,  99,  99,  99,  99,  99,  99];

    Luminance_Quantization_Table = Luminance_Quantization_Table
       * (1 - quality_scale + 0.5);
    Chrominance_Quantization_Table =
       Chrominance_Quantization_Table * (1 - quality_scale +
       0.5);
    out(:,:,1) = blkproc(quantified(:,:,1),[8 8], 'x.*P1'
       ,Luminance_Quantization_Table);
    out(:,:,2) = blkproc(quantified(:,:,2),[8 8], 'x.*P1'
       ,Chrominance_Quantization_Table);
    out(:,:,3) = blkproc(quantified(:,:,3),[8 8], 'x.*P1'
       ,Chrominance_Quantization_Table);
end
```