图像压缩

- 1 概述
- 2 JPEG标准
- 3 JPEG2000标准
- 4 JPEG-LS
- 5 JBIG

Ze-Nian Li： c9： 图像压缩标准

c7.7 无损图像压缩

林福宗： c4.6 JPEG c8.6JPEG2000

1、概述

- 1、数字图像的应用需求
  - 扫描仪
  - 数码相机
  - 。。。
- 2、标准的重要性
  - 单独的程序
  - 单独的设备
  - 。。。

图像中存在大量的冗余

- **空间冗余**

基于离散像素采样来表示物体颜色的方式通常没有利用景物表面颜色的这种空间连贯性，从而产生了空间冗余

同一景物表面上各采样点的颜色之间往往存在着空间连贯性

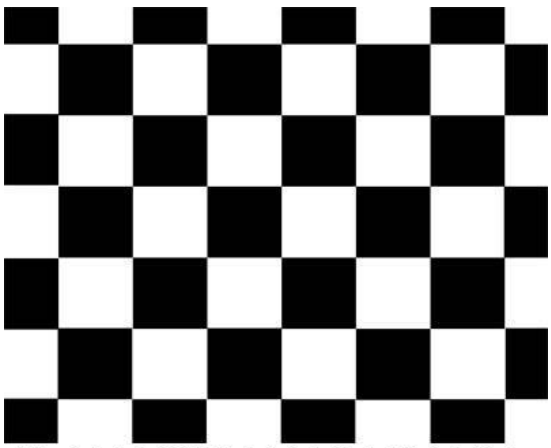
通过改变物体表面颜色的像素存储方式来利用空间连贯性。达到减少数据量的目的

图像中存在大量的冗余

- 结构冗余

在有些图像的纹理区，图像的像素值存在着明显的分布模式,称为结构冗余.

若已知分布模式,可以通过某一过程产生图像。



我们可以看到图像的像素值存在着明显的分布模式，这种规律性的结构有利于减少存储数据量。

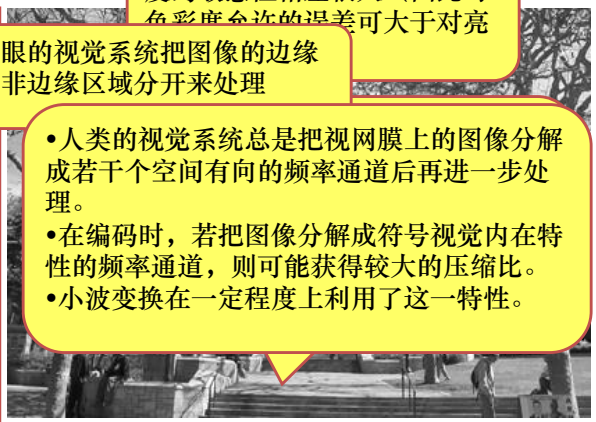
图像中存在大量的冗余

- 视觉冗余

•实际编码比理想编码有更多的数据，称为视觉冗余。
•理想编码：人类的视觉系统对图像场的敏感性是非均匀和非线性的，应把视觉敏感和不敏感的部分区分开来编码。
•实际编码：在记录原始的图像数据时，通常假定视觉系统是线性的和均匀的，对视觉敏感和不敏感的部分同等对待。

视觉系统对图像的亮度和色彩度的敏感性相差很大（因此对色彩度允许的误差可大于对亮

人眼的视觉系统把图像的边缘和非边缘区域分开来处理

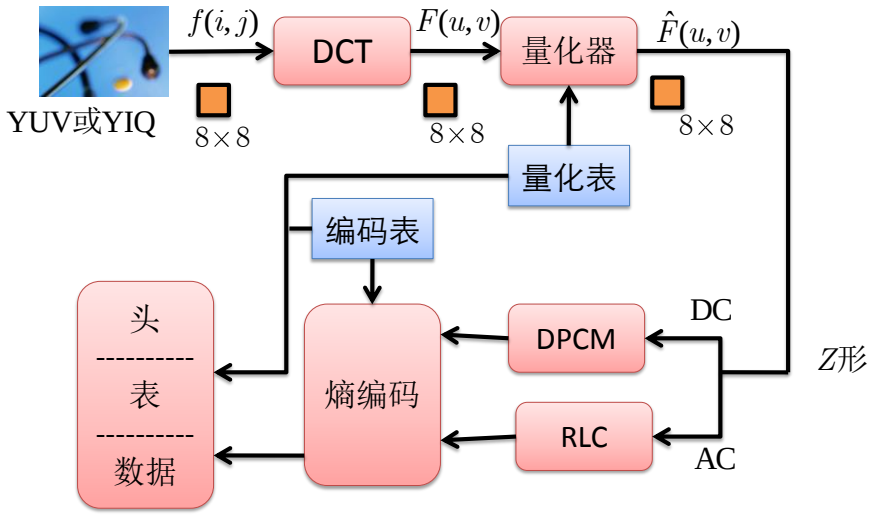


•人类的视觉系统总是把视网膜上的图像分解成若干个空间有向的频率通道后再进一步处理。
•在编码时，若把图像分解成符号视觉内在特性的频率通道，则可能获得较大的压缩比。
•小波变换在一定程度上利用了这一特性。

2 JPEG标准

- JPEG (Joint Photographic Expert Group)
- 制定标准的图像压缩方法是不同厂商生产设备支持互操作性的需要;
- 是第一个连续色调图像（灰度或彩色）的国际标准。
- **JPEG专家组开发的压缩算法，**
  - **以离散余弦变换(Discrete Cosine Transform，DCT)为基础的有损压缩算法**
  - **采用以预测技术为基础的无损压缩算法**
  - **为了在保证图像质量的前提下进一步提高压缩比，又制定了JPEG 2000标准，这个标准中将采用小波变换(wavelet)算法**

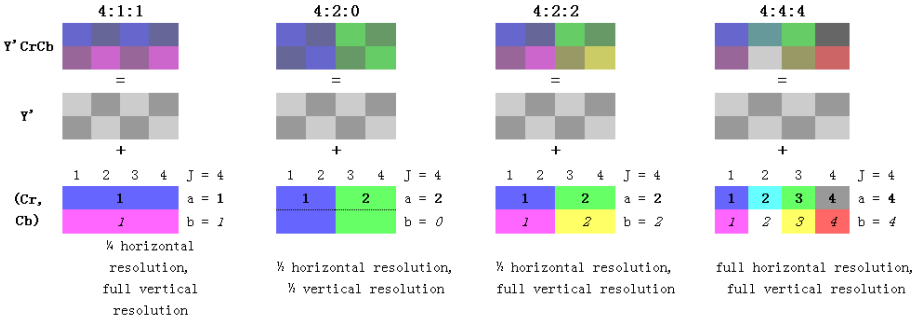
编码器模块图（baseline）

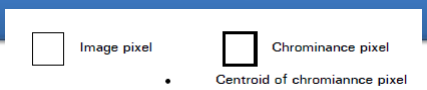


- JPEG可用于彩色和灰度图像。对于彩色图像（如YIQ或YUV），编码器在各自的分量上工作，但使用相同的例程。
- 如果源图像是其他格式的，编码[illegible]换，将其转换为YIQ或YUV。
- 色度图像是二次采样（也称为"色度抽样"（chroma subsampling）的：
  - 4:4:4（无缩减取样）
  - 4:2:2（在水平方向 2 的倍数中取一个）
  - 4:2:0（在水平和垂直方向 2 的倍数中取一个）

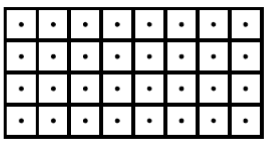
人的亮度敏感性
高于彩色敏感性

色度采样模式

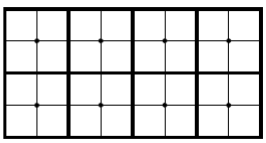




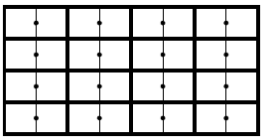
4:4:4
H: 1/1
V: 1/1
T: 1/1



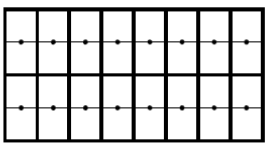
4:2:0 ①
H: 1/2
V: 1/2
T: 1/4



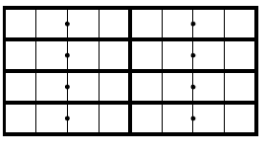
4:2:2
H: 1/2
V: 1/1
T: 1/2



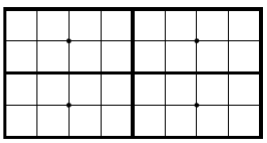
4:4:0
H: 1/1
V: 1/2
T: 1/2

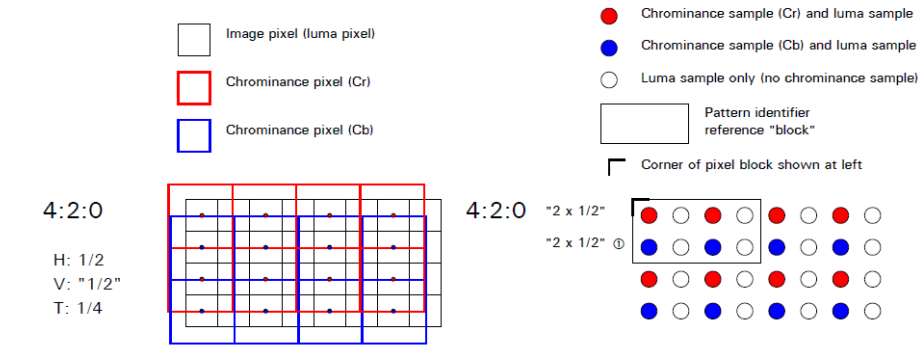


4:1:1
H: 1/4
V: 1/1
T: 1/4

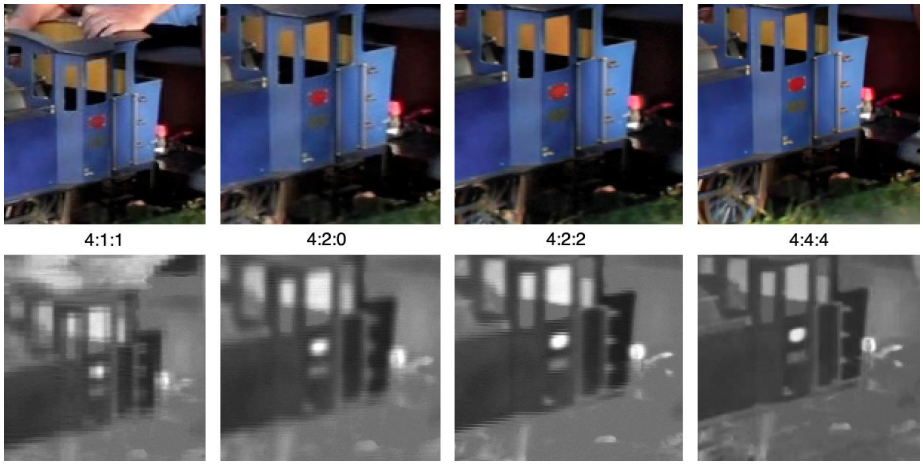


4:1:0
H: 1/4
V: 1/2
T: 1/8

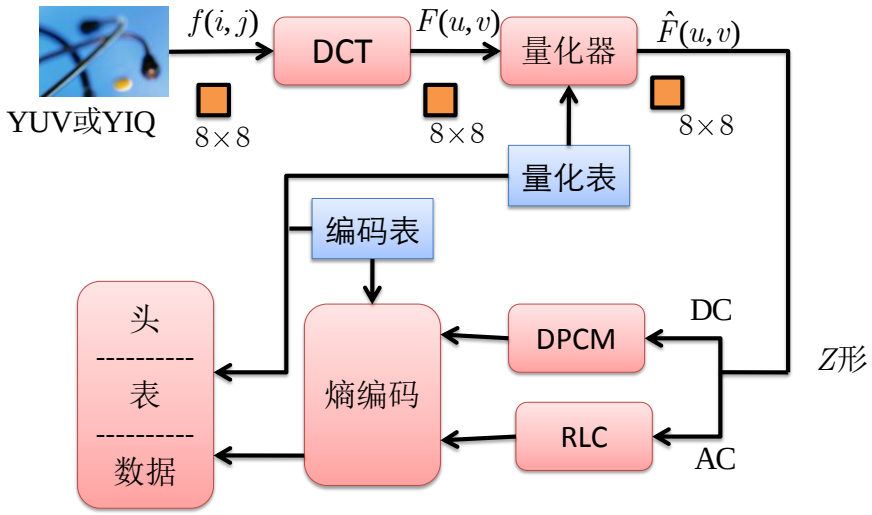




DV-PAL subsampling pattern



编码器模块图（baseline）



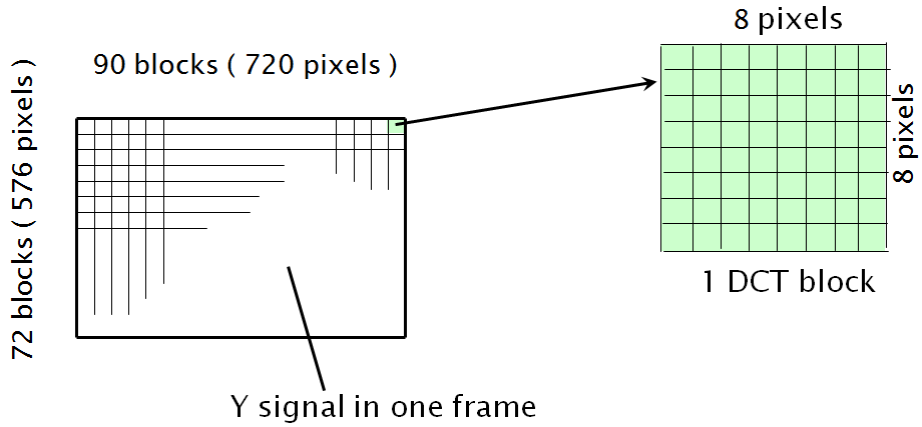
① 图像分块

根据实践证明子图像尺寸取4×4、8×8、16×16适合作图像的压缩，这是因为：

① 如果子图像尺寸取得太小，虽然计算速度快，实现简单，但压缩能力有一定的限制。

② 如果子图像尺寸取得太大，虽然去相关效果变好，因为象DFT、DCT等正弦型变换均具有渐近最佳性，但也渐趋饱和。若尺寸太大，由于图像本身的相关性很小，反而使其压缩效果不显示，而且增加了计算的复杂性。

②图像块的DCT变换



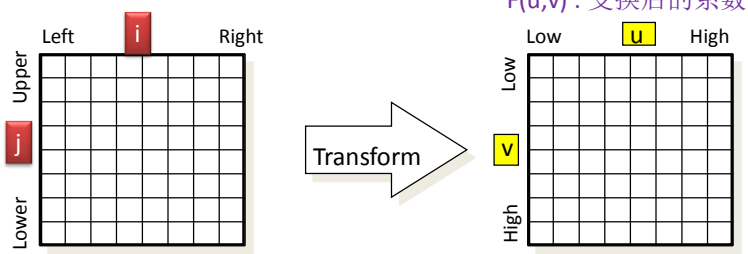
DCT(Discrete Cosine Transform)

$$F(h,v)=C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}$$

$$C(\xi)=\begin{cases}\frac{1}{\sqrt{2}} & (\xi=0)\\ 1 & (\xi>0)\end{cases}$$

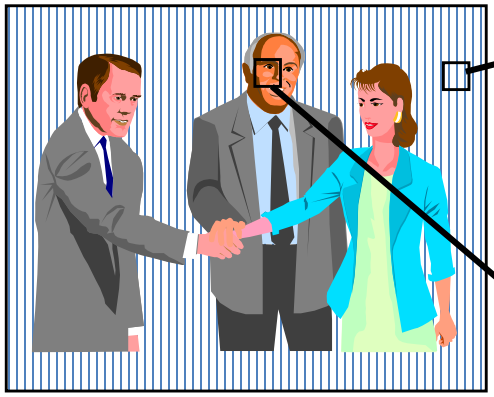
f(i,j) : 变换前像素数据

F(u,v) : 变换后的系数



时域数据　　频域数据

DCT变换 例



DCT

521	100	-15	8	-4	-1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

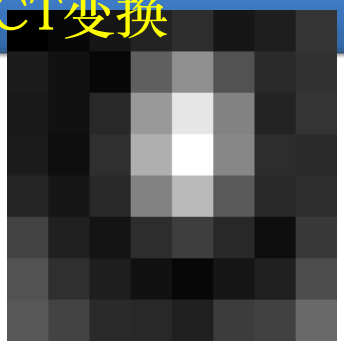
DCT

922	-150	-25	-5	-6	-12	-6	-1
-208	-10	6	2	4	1	3	3
-29	31	2	-3	2	-1	-2	0
-19	1	-3	-6	3	3	3	2
0	-5	-4	2	1	2	0	0
0	-4	6	5	1	4	2	1
-1	1	-2	4	-1	1	4	3
0	1	0	2	3	0	-3	-1

简单程序

Y分量去均值的DCT变换

如果有一个如这样的的8×8的8-位元（0~255）子区域：



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

平移128，使其范围变为-128~127，得到结果为

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

缩小数据的取值范围，以缩小处理规模

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

接着使用DCT离散余弦变换，和舍位取最接近的整数，得到结果为

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

平移操作后，AC系数不变，只改变了DC系数

③ DCT系数的量化

- DCT本身并不能进行码率压缩，因为64个样值仍然得到64个系数。样值为8bit，得到的直流分量为原来256的64/8=8倍，即0~2047。交流分量的范围是-1023~1023。
- DCT变换后直流分量一般最大，低频分量又较大于高频分量，所以系数值通常从左上角向右下角迅速减小。
- 压缩是从量化开始的。根据人眼的视觉特性（对低频敏感，对高频不太敏感）对低频分量采取较细的量化，对高频分量采取较粗的量化。压缩比通过改变量化矩阵来实现。图像内容经DCT变换和Q量化后，高频分量会多数变为0而被舍弃。如果原始图像中细节丰富，则去掉的数据较多。量化后的系数与量化前差别大。反之，细节少的原始图像在压缩时去掉的数据少些。

DCT系数的物理意义?

- DCT系数值表示 8×8块中二维空间频率的幅值和相位
- F(0,0) 称为DC系数，表示块的平均能量
- 其它系数称为AC系数，对于绝大多数图像而言，高频位置的系数接近于0
- 去数据相关，即去除冗余
- 能量集中于系数块的左上角，因此可以丢弃大多数系数从而达到数据压缩

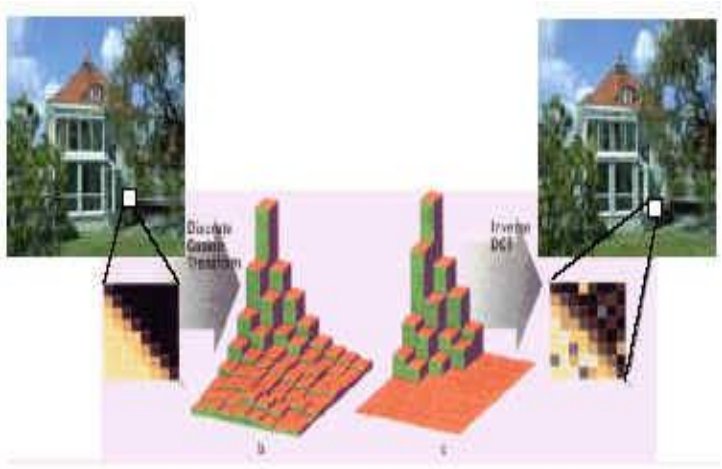
量 化

$$F'(u,v) = Round(\frac{F(u,v)}{Q(u,v)})$$

F(u,v): DCT系数
F^(u,v): 量化后的DCT系数
Q(u,v): 系数对应的量化阶
Round():取整函数

- **为什么要进行量化? ——压缩的源泉**
- **由于不同位置的DCT系数对应不同的空间频率，与每个DCT系数对应的量化值并不相同，高频系数的量化值更大**
- **量化使得绝大多数系数为0，因此得到更好的压缩性能，但也是压缩系统“损失”的来源**

量化的目的是减小非“0”系数的幅度以及增加“0”值系数的数目。量化是图像质量下降的最主要原因。



量化 矩阵（量化步长）

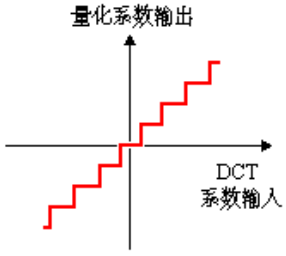
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

JPEG 亮度量化表

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

JPEG 色度量化表

从心理学的研究结果中得出来，能够得到最大的压缩率，同时使JPEG图片的感知损失最小。



量化示例

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \div \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \rightarrow \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{round}\left(\frac{-415}{16}\right) = \text{round}\,(-25.9375) = -26$$

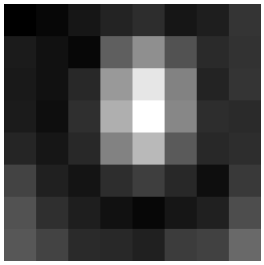
解码—编码的逆过程

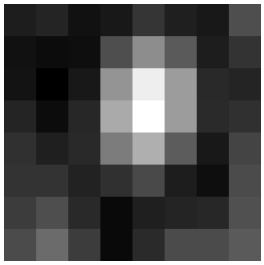
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{反量化}} \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{反DCT}}$$

$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix} \xleftarrow{\text{加128}} \begin{bmatrix} -68 & -65 & -73 & -70 & -58 & -67 & -70 & -48 \\ -70 & -72 & -72 & -45 & -20 & -40 & -65 & -57 \\ -68 & -76 & -66 & -15 & 22 & -12 & -58 & -61 \\ -62 & -72 & -60 & -6 & 28 & -12 & -59 & -56 \\ -59 & -66 & -63 & -28 & -8 & -42 & -69 & -52 \\ -60 & -60 & -67 & -60 & -50 & -68 & -75 & -50 \\ -54 & -46 & -61 & -74 & -65 & -64 & -63 & -45 \\ -45 & -32 & -51 & -72 & -58 & -45 & -45 & -39 \end{bmatrix}$$

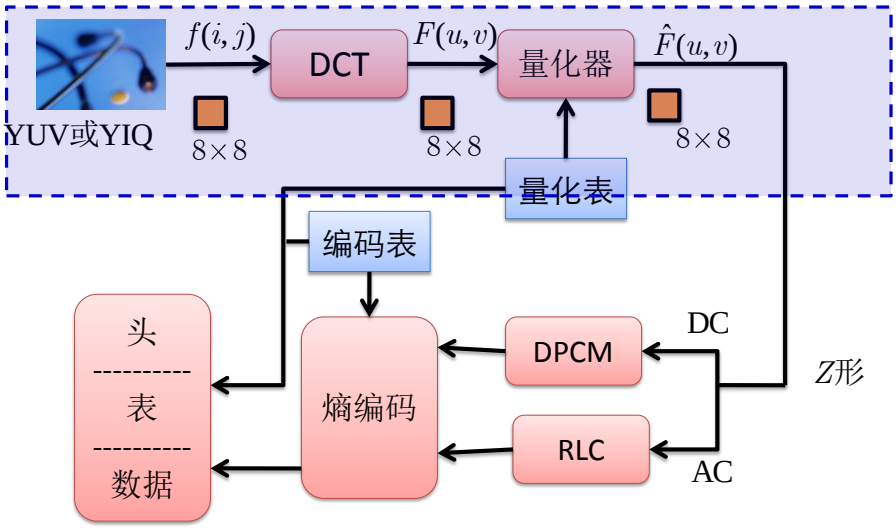
原图与解压缩图的误差

$$\begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$





编码器模块图（baseline）



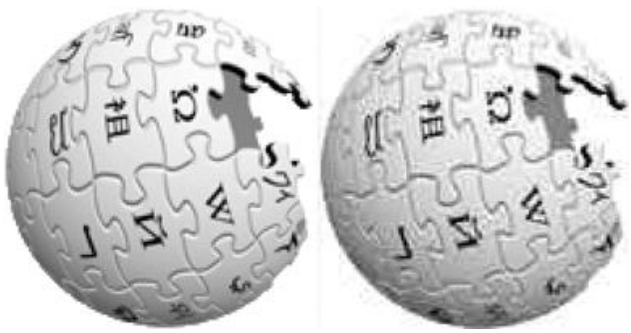
质量因子（Quality factor）

- 原图

ps中品质为1的图

压缩率与人为现象（artifact）

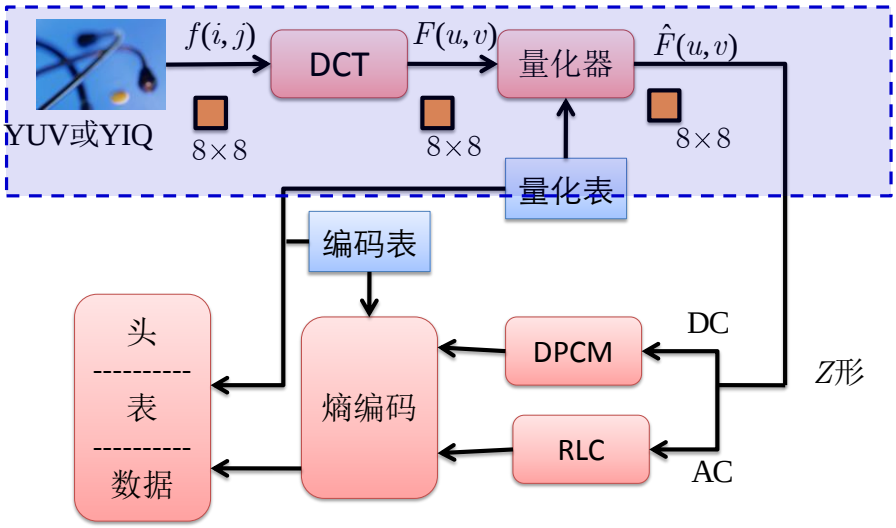
- 在量化阶段时，依照除数的不同，会使结果的压缩比率可能有很多变化。25:1通常可得到无法使用肉眼分辨与原图差异的影像。100:1压缩通常是可行的，但与原图相较，会看出明显的人为现象。压缩的适当等级是依据要压缩哪一种影像而定。



平滑图像块和纹理图像块的量化

- p179图9-2和p180图9-3
- 图9-2：除了DC和前几个AC，大部分DCT系数的幅值都很小。
  - 误差：较小
- 图9-3：AC同样具有较大幅值。
  - 误差：较大
- 如果图像变化较为剧烈，JPEG会引起较大损失。

编码器模块图（baseline）



④熵编码的准备

- AC系数
  - Zigzag扫描
  - 游程编码
- DC系数
  - DPCM编码

AC系数的Zigzag扫描

- **由于经DCT变换后，系数大多数集中在左上角，即低频分量区，因此采用Z字形扫描，按频率的高低顺序读出。**
- **可以出现很多连零的机会。可以使用游程编码。**
- **尤其在最后，如果都是零，给出EOB即可。**

Zig-zag 扫描

231	-74	-12	-1	-1	-2	-1	0
-102	-5	1	0	1	0	0	0
-13	7	0	0	0	0	0	0
-4	0	0	-1	0	0	0	0
0	-1	-1	0	0	0	0	0
0	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

EOB (End Of Block)

EOB is transmitted instead of zeros

231	-74	-12	-1	-1	-2	-1	0
-102	-5	1	0	1	0	0	0
-13	7	0	0	0	0	0	0
-4	0	0	-1	0	0	0	0
0	-1	-1	0	0	0	0	0
0	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

EOB

AC系数的游程编码 RLC

- **Run-Length Coding，又称“运行长度编码”或“行程编码”，是一种统计编码，属于无损压缩编码。对具有特定数据量值特性的数据序列采用的一种简化记述方法**
- **在JPEG和MPEG编码中规定为：**

  (n,m)

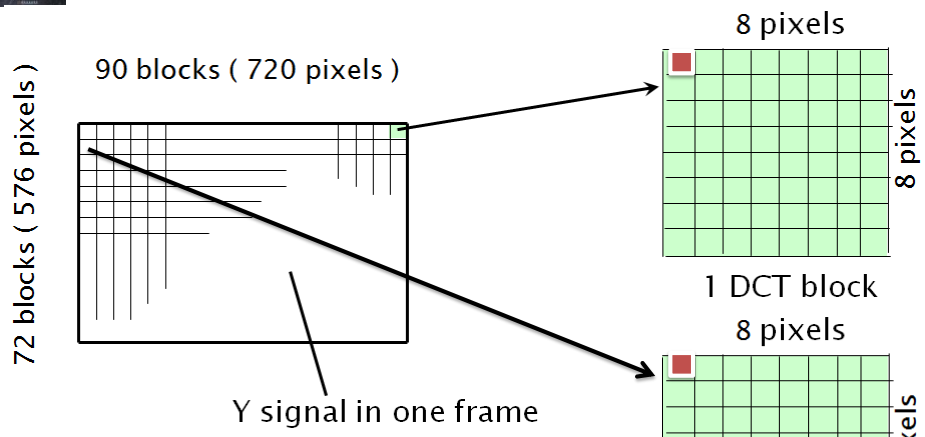
  其中m为某个非零系数值，n为该m值前0的个数。

例如：Zigzag扫描结果如下：

0，2，0，0，3，0，-4，0，0，0，-6，0，0，5，7

则表示为（1，2），（2，3），（1，-4），（3，-6），（2，5），（0，7）

DC系数的DPCM编码



- DC系数表示每个图像块的平均亮度，所以不同的块DC系数会很大，并且不同。
- 在邻近的块之间，DC系数不会产生特别大的变化。

DC系数DPCM 举例

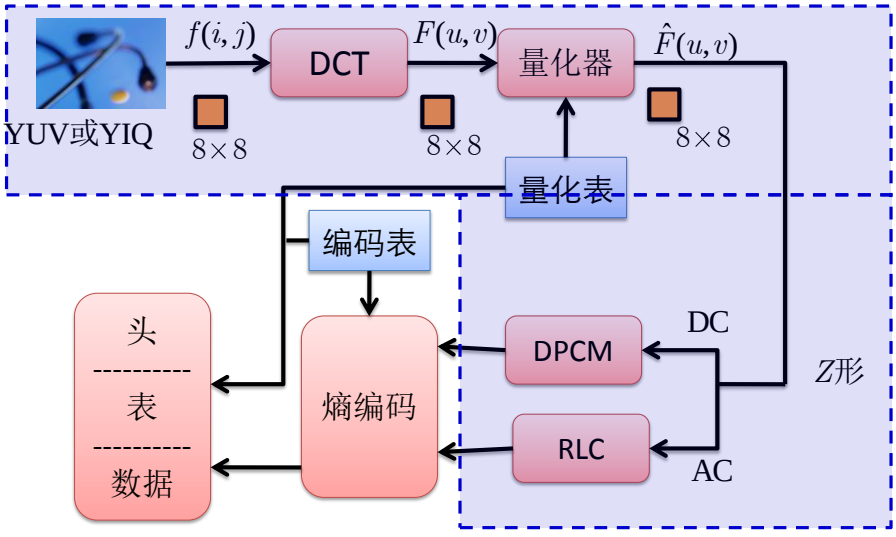
- 例如：前5个块的DC系数：150,155,149,152,144
- 假设d_0=DC0，第i个块的预测器为

$$d_i = DC_i - DC_{i-1}(i > 0)$$

- DPCM编码后：150，5，-6，3，-8

- AC系数的游程编码需要对每一个图像块单独进行，而JPEG的DC系数的DPCM编码是对整个图像实施一次即可。即AC系数编码是分块内完成，而DC系数编码是在分块之间。

编码器模块图（baseline）



⑤ 熵编码（baseline）

- **使用熵编码还可以对DPCM编码后的直流DC系数和RLC编码后的交流AC系数作进一步的压缩。**

A. DC系数的huffman编码

- **S1：构造中间码**

  表达方法（size，amplitude）即（位数，差分DC系数）

- **在JPEG中，只对size进行huffman编码；amplitude值变化范围较大，不进行huffman编码**

如： 150，5，-6，3，-8

（size，amplitude）

分别用符号1和符号2表示。
符号1：位长及其码字
符号2：具体差值（幅值）对应的位长和码字。

符号2的表示：
表示DPCM的值需要超过8位，而且值有正也有负。
正数直接用二进制编码表示，负数则用编码取反表示。
例如
5 -> 101
-8 ->0111

位长（size)	DC或AC系数值
0	0
1	-1，1
2	-3,-2,2,3
3	-7,-6,-5-4,4,5,6,7
4	-15…-8,8…15
5	-31…-16,16…31
……	……
10	-1023..-512,512..1023

• DC系数差值幅度范围、分类与huffman编码表

符号1
的表示

DC 系数差值幅度范围	位长	编码
0	0	00
-1，1	1	010
-3，-2，2，3	2	011
-7,…,-4,4,…,7	3	100
-15,…,-8,8,…,15	4	101
-31,…,-16,16,…,31	5	110
-63,…,-32,32,…,63	6	1110
…	7	11110
…	8	111110
…	9	1111110
…	10	11111110
-2047,…,-1024;1024…2047	11	111111110

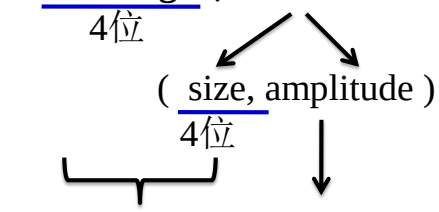
符号2： 正负值振幅码表

位 长 (Size)	振 幅 (Amplitude)	码 字 (Code word) (binary)
0	0	–
1	−1,1	0,1
2	−3,−2,2,3	00,01,10,11
3	−7,…,−4,4,…,7	000,…,011,100,…,111
4	−15,…,−8,8,…,15	0000,…,0111,1000,…,1111
⋮	⋮	⋮
16	32768	–

例：5 符号1—— 100
符号2——101 结果：100101
-8 符号1——101
符号2——0111 结果：1010111

B. AC系数的huffman编码

- （**Runlength，Value**）



- 先用零游程编码，再根据（n，m）转换成符号1和符号2。查找相应的两张表完成**二进制编码**。
- 在JPEG中，对AC系数的Symbol 1进行huffman编码；
- Symbol 2值变化范围较大，不进行huffman编码。

Runlength可能超过15。每16个连续的零用一个用“1111/0000”表示。
EOB用“0000/0000”

AC的幅值与对应的位长size

求得位长

实际的AC系数值，查出AC幅值所对应的二进制码字。**-2——10** .位长为2，符号1则为

位　长 (Size)	振　幅 (Amplitude)	码　字 (Code word) (binary)
0	0	—
1	−1,1	0,1
2	−3,−2,2,3	00,01,10,11
3	−7,⋯,−4,4,⋯,7	000,⋯,011,100,⋯,111
4	−15,⋯,−8,8,⋯,15	0000,⋯,0111,1000,⋯,1111
⋮	⋮	⋮
16	32768	

正数直接用二进制编码表示，负数则用编码取反表示。

AC系数（NNNN/SSSS）对应的huffman 编码

n/s	Huffman 编码
0/0（EOB)	1010
0/1	00
0/2	01
0/3	100
0/4	1011
0/5	11010
0/6	1111000
0/7	11111000
0/8	1111110110
0/9	11111111 10000010
0/A	11111111 10000010

1/1	1100
1/2	11011
1/3	1111001
1/4	11111011 0
1/5	11111110 110
1/6	11111111 10000100
1/7	11111111 10000101
1/8	11111111 10000110
1/9	11111111 10000111
1/A	11111111 10001000
2/1	11100
2/2	11111001
2/3	11111101 11
2/4	11111111 0100
…	
F/0	11111111 001
F/1	11111111 11110101
…	
F/9	11111111 11111101
F/A	11111111 11111110

举例

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

规格化量化系数

DC系数的编码

1：DC系数15，假如其前一个样本的DC系数13，则差值为2，查表得位长为2.

2：用位长查亮度或色度DC系数Huffman表，得该分组的编码为011。（这是符号1的编码）

3：在分组的编码为011的后面附加上DC系数2的码（10），得DC系数的输出编码为011 10。

AC系数的编码

- **1：对AC系数进行“Z”字排序，得：0-2-1-1-100-1000…000**
  （1,-2）（0,-1）（0,-1）（0,-1）（2,-1）EOB
- **2:第一个非零值为-2，查表得分组大小为2，其前面的零的个数为1，即行程长度为1，于是中间码的前半部分为1/2；而后半部分为-2的查表为01。**
- **3：再由中间码的前半部分为1/2查JPEG提供的亮度或色度AC系数HUFFMAN表，得此系数的编码为11011。（这是符号1的编码）**
- **所以（1,-2）的编码结果为1101101**

综上所述，我们可用中间码的形式表示前面所举8*8样本子块：

- (2)(2), (1/2)(-2), (0/1)(-1), (0/1)(-1), (0/1)(-1), (2/1)(-1), (0/0)
- 利用JPEG提供的HUFFMAN表，可得到最后的熵编码的输出序列为：
- 01110 1101101 0000 0000 0111000 1010
- 可见，经过处理后，表示8*8样本只需要31bit。压缩后，每样本不到0.5个bit.

注意：压缩比

压缩比=编码前的平均码长/编码后的平均码长

在熵编码后，比特数大大降低。如输出31比特。

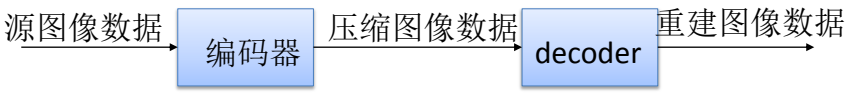
1.该系数块每样值比特数为31/64≈0.484bpp（比特/样值），原始为8bpp。

压缩比为 8/0.484 ≈ 16.5

2.原始的8×8像块中总比特数为（8×8）×8=512比特。压缩后，一个像块对应输出为31比特。

压缩比为 512/31 ≈ 16.5

⑥ JPEG模式



图像压缩系统

JPEG 支持下述编码模式

- 基于DCT的顺序编码模式
- 基于DCT的累进编码模式
- 分层模式
- 顺序无损编码模式

JPEG 熵编码器支持:

- 霍夫曼编码
- 算术编码（产品很少支持）

A. 顺序模式

- 顺序：对灰度图或彩色图像分量进行从左到右、从上到下的扫描并编码。
- 前面的JPEG Baseline 系统—**顺序模式**
  - 基于DCT的顺序编码
  - 熵编码使用霍夫曼编码
  - 8比特数据精度

Baseline基线：软件文档或源码(或其它产出物)的一个稳定版本，它是进一步开发的基础。

B. JPEG 累进模型

- 为什么要采用累进模型?
  - 图像传输过程中，可快速的由粗到细。
  - 首先快速传送低质量的图像，接着传送高质量的图像
- 第一步: 对图像进行粗略但可接受的编码
- 第二步:对图像逐次细化直到得到最后的图像

累进模型的两种实现方法

- 频谱选择 – 一次扫描中，只对某些空间频带段的系数编码传送，然后以累进的方式对其它频带段进行编码传送。直至将全部系数传送完毕
- 逐步逼近 – 对DCT系数按位平面（bit plane）由高至低分成若干段，然后依次对各段进行压缩编码，现对最高有效位进行编码传送，然后再对剩余的位平面分别编码传送。

两种方法的比较

频谱选择

- 利用DCT系数的频谱特性，高等级的AC分量仅提供细节信息
- 第*1*次扫描：
  - 对DC和前几个AC分量编码，例如AC1、AC2
- 第*2*次扫描：
  - 对更多的AC分量编码：例如AC3、AC4、AC5
- ……
- 第*k*次扫描：
  - 对最后几个AC分量编码，例如AC61、AC62、AC63。

逐步逼近

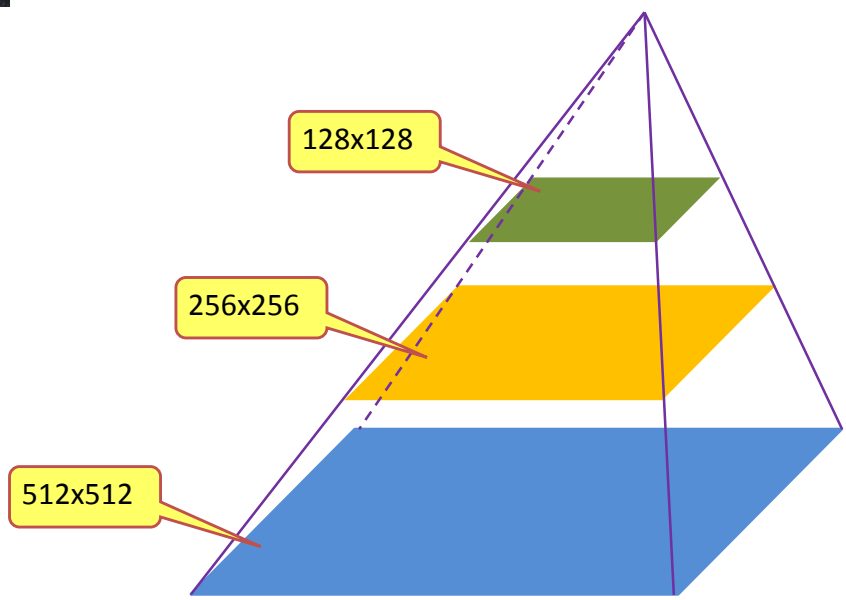
- 将所有DCT系数同时编码，最重要位MSB最先编码
- 第*1*次扫描：
  - 对前几个MSB编码，例如Bits7、6、5
- 第*2*次扫描：
  - 对重要程度稍有降低的比特编码：例如Bits4、3
- ……
- 第*k*次扫描：
  - 对最不重要的位LSB编码，例如Bit0。

65

C. JPEG 分层模型

- 对处于不同分辨率层次中的图像进行编码。低分辨率图像是通过低通滤波器压缩后的图像，更高分辨率的图像提供更多的细节信息（和低分辨率图像提供的不同）
- 和累进JPEG类似，分层JPEG可以通过多次扫描，渐进改善图像质量来传送（**金字塔型**编码）
- 步骤:
  - 将原图像经过下采样和滤波，得到降低分辨率的图像
  - 使用前述编码模式对降低分辨率后的图像进行编码
  - 将经过上采样后的图像作为原图像的预测图像，对差值进行编码
  - 不断重复，直到对原图像编码完毕

金字塔模型



分层模型举例 三级JPEG编码器

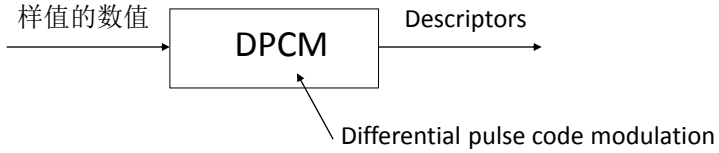
1. 降低图像分辨率。在输入图像f（如512x512）的每一维上除以2得到f_2（256x256）。重复这一步骤得到f_4（128x128）
2. 压缩低分辨率图像f_4。对f_4采用其他的JPEG方法（如顺序、累进）编码，得到F_4。
3. 压缩差值图像d_2
   a. 对F_4解码得到 $\hat{f}_4$。用某种插补方法扩展 $\hat{f}_4$，得到f_2同一分辨率的图像 $E(\hat{f}_4)$
   b. 对差值 $d_2 = f_2 - E(\hat{f}_4)$用某种JPEG方法编码得到D_2
4. 压缩差值图像d1
   a. 对D_2解码得到$\hat{d}_2$；把它加到 $E(\hat{f}_4)$上得到 $\hat{f}_2 = E(\hat{f}_4) + \hat{d}_2$
   b. 对差值$d_1 = f - E(\hat{f}_2)$用某种JPEG方法编码得到D_1

分层模型举例 三级JPEG解码器

1. 对低分辨率图像F4解压缩。使用和编码器相同的JPEG方式解压F4得到 $\tilde{f}_4$
2. 还原中间分辨率图像 $\tilde{f}_2 = E(\tilde{f}_4) + \tilde{d}_2$
3. 还原原始分辨率图像 $\tilde{f} = E(\tilde{f}_2) + \tilde{d}_1$

D. JPEG 无损模型

采用一种简单的差分编码方法，不涉及任何的变换编码.



由于物理世界的连续性，变化缓慢的成像性质会使得图像空间上的相邻像素具有相似亮度值的几率变大。给定一副图像I（x，y），使用简单的差分操作，可以定义一个差分图像：

$$d(x,y)=I(x,y)-I(x-1,y)$$

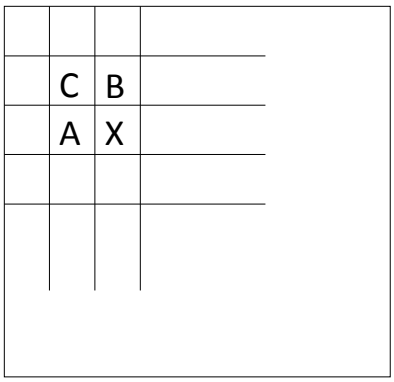
或者

$$d(x,y)=4I(x,y)-I(x-1,y)-I(x+1,y)\\-I(x,y-1)-I(x,y+1)$$

JPEG 无损模型

s1：形成差分预测

Predictors for lossless coding	
selection value	prediction strategy
0	no prediction
1	A
2	B
3	C
4	A+B-C
5	A+(B-C)/2
6	B+(A-C)/2
7	(A+B)/2

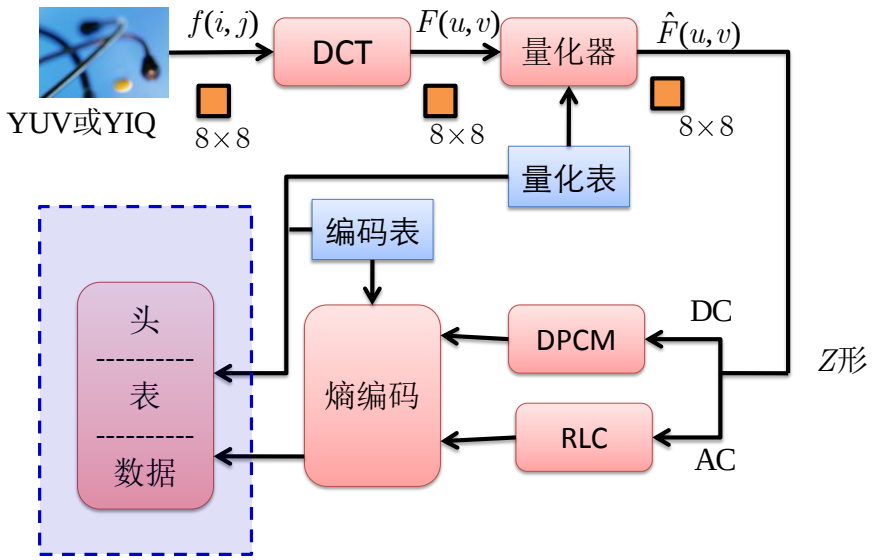


s2：将预测值与位置X上的实际值比较，并使用某种无损压缩算法对差值进行编码

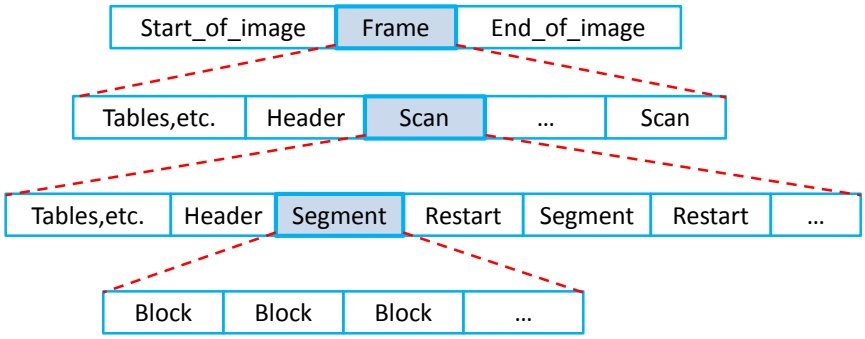
JPEG 无损模型的特点

- 压缩率较低（1.0~3.0），不适用于大多数的媒体应用。
- 对特殊图像的处理采用该模式：如医学图像。

JPEG baseline框图



⑦JPEG位流概述(bit stream)



- 帧Frame就是一个图片，扫描Scan就是对全部像素（比如亮度分量）的一次遍历，段Segment就是一组块，块Block由8x8的像素组成。

3 JPEG2000介绍

- 毫无疑问，JPEG标准是迄今为止最为成功和通用的图像格式。它在相对出色的压缩率下仍有很好的输出质量。
- JPEG2000标准不仅在**压缩率-失真**间进行了更好的权衡，改善了图像质量，而且新增了JPEG标准中所缺乏的一些功能。特别是：
  - 低位率压缩：在中高位率情况下，JPEG标准非常出色。但在位率低于0.25bpp时图像失真变得无法接受。
  - 感兴趣区域编码（ROI）：允许ROI区域的编码质量优于图像的其他部分
  - 大图像：能处理分辨率超过64Kx64K的图像而不会产生失真，上限达2^{32}-1
  - 无损和有损压缩 、噪声环境中的传输 、渐进传输、计算机生成的影像

JPEG2000编码模式

- 基于DCT
  - 为了能够向后兼容JPEG标准并实现基本JPEG
- 基于小波变换
  - 所有新的功能和改进性能都是在基于小波变换模式下获得的

感兴趣区域编码举例

0.4bpp

0.5bpp

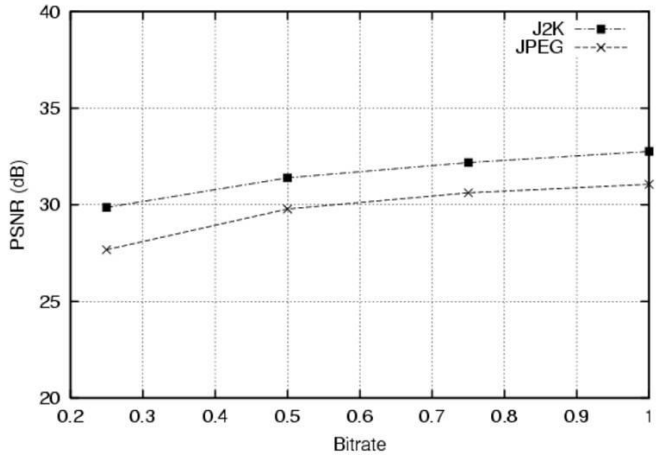
0.6bpp

0.7bpp

JPEG和JPEG2000性能比较

78

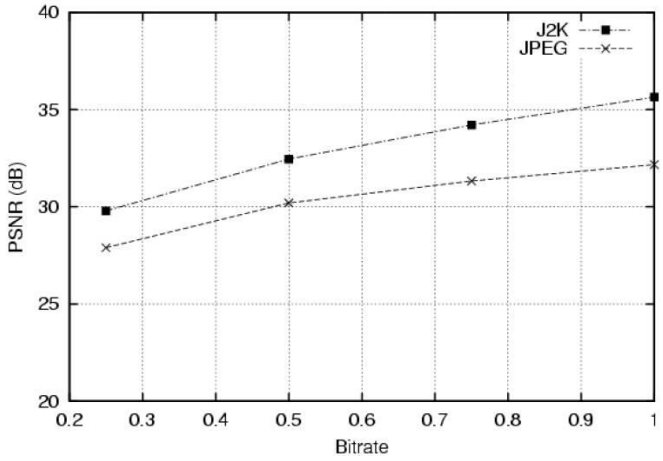
（a）自然图像



JPEG和JPEG2000性能比较

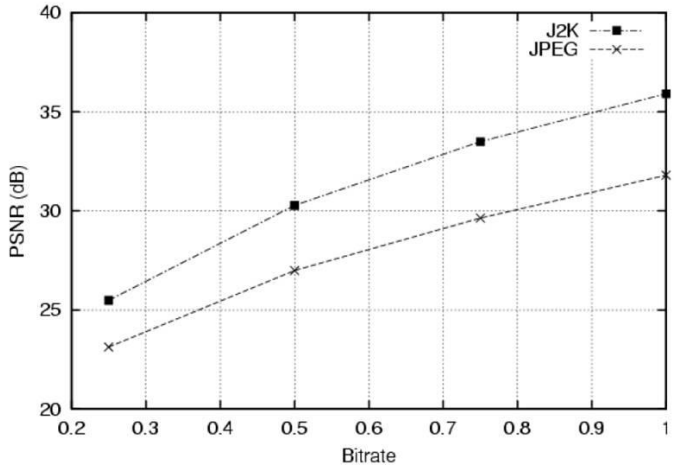
79

（b）计算机生成的图像



JPEG和JPEG2000性能比较

（c）医学图像



JPEG vs JPEG2000

JPEG　　JPEG2000

0.75bpp

JPEG vs JPEG2000

JPEG JPEG2000

0.25bpp