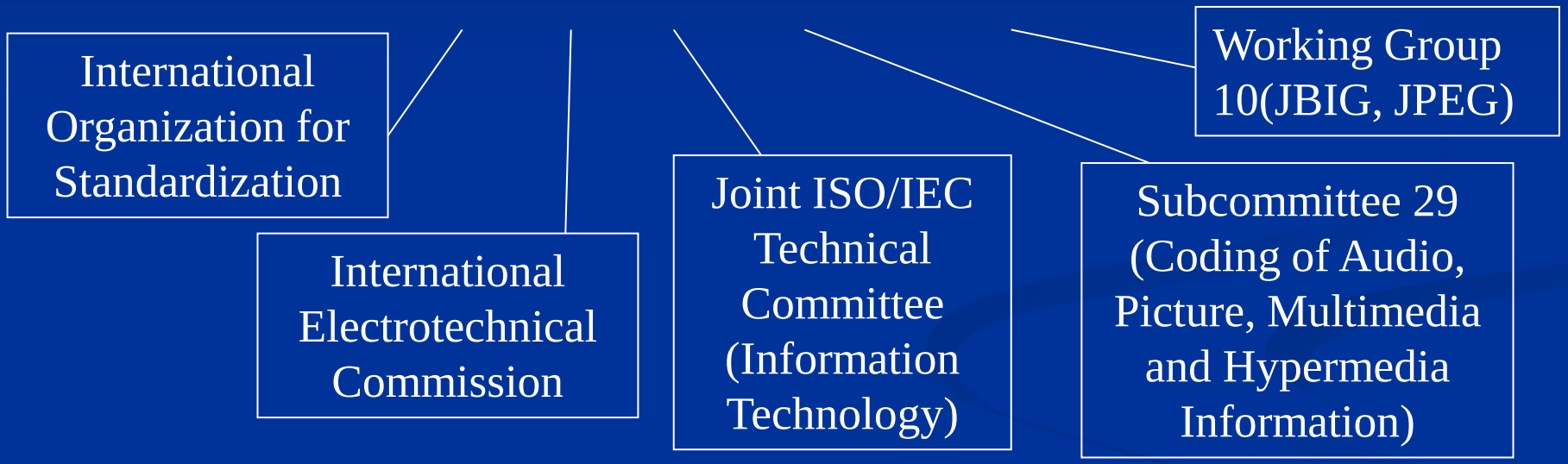
Chp13：变换编码

- JPEG

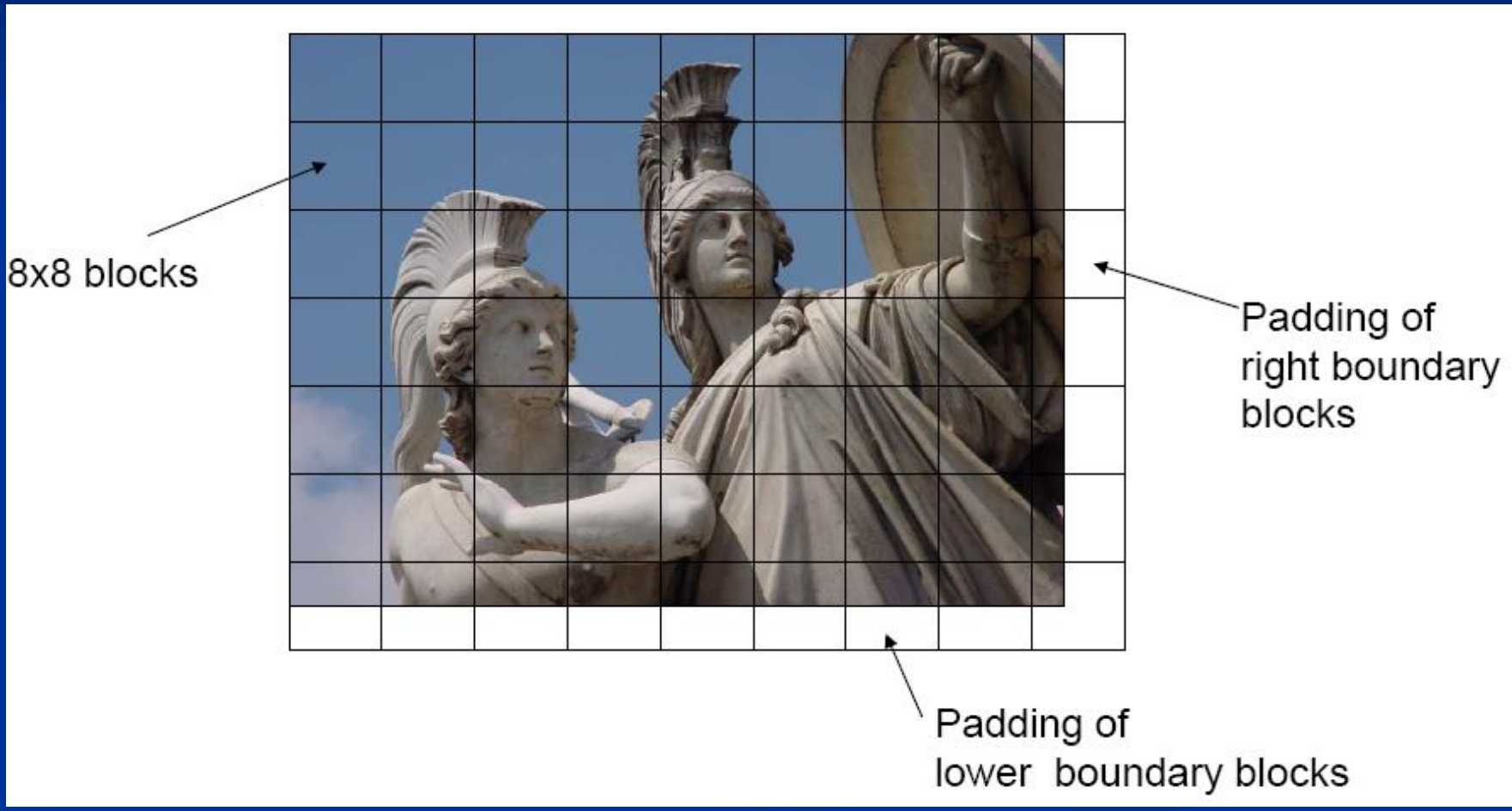
JPEG标准

- JPEG：Joint Photographic Experts Group
- 正式名称：ISO/IEC JTC1/SC29/WG10



- 与CCITT （现为ITU）学习组VIII联合
- 工作开始于1986年
- 于1992年形成国际标准ISO/IEC 10918-1 和CCITT建议 T.81
- 广泛用于图像交换、WWW、数字图像
- Motion-JPEG为数字视频编缉的事实标准

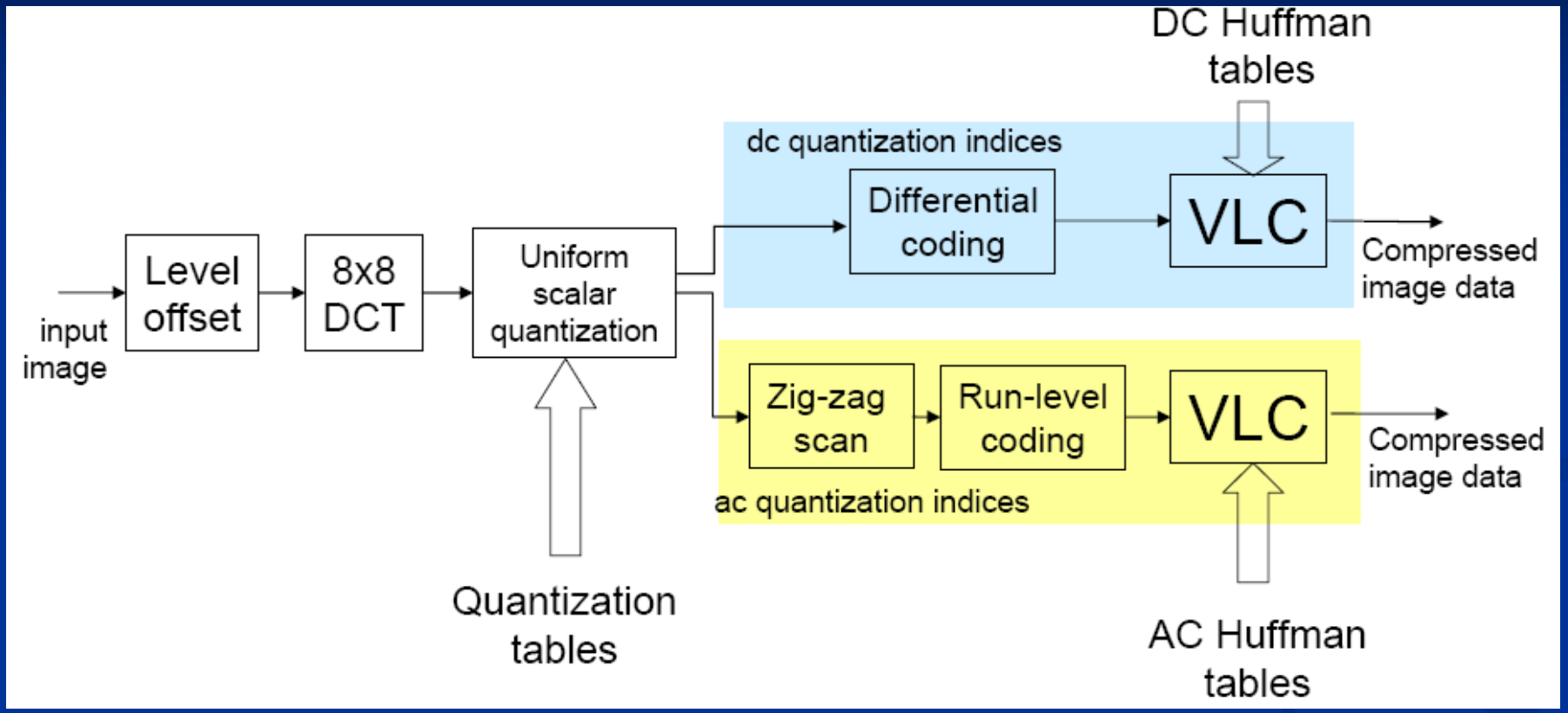
JPEG：8×8的块编码


8x8 blocks
Padding of
right boundary
blocks
Padding of
lower boundary blocks

JPEG标准

- JPEG规定了4种运行模式，以满足不同需要：
  - 基于DPCM的无损编码模式：压缩比可达2:1
  - 基于DCT的有损顺序编码模式：压缩比可达10:1以上
  - 基于DCT的递增编码模式
  - 基于DCT的分层编码模式

基本(baseline) JPEG编码器



- Huffman编码：通过简单的查表就可以实现
- Huffman编码可以用自适应二进制算术编码代替（由于专利问题，很少产品支持）
  - 编码效率提高10%，但算法更复杂

颜色空间

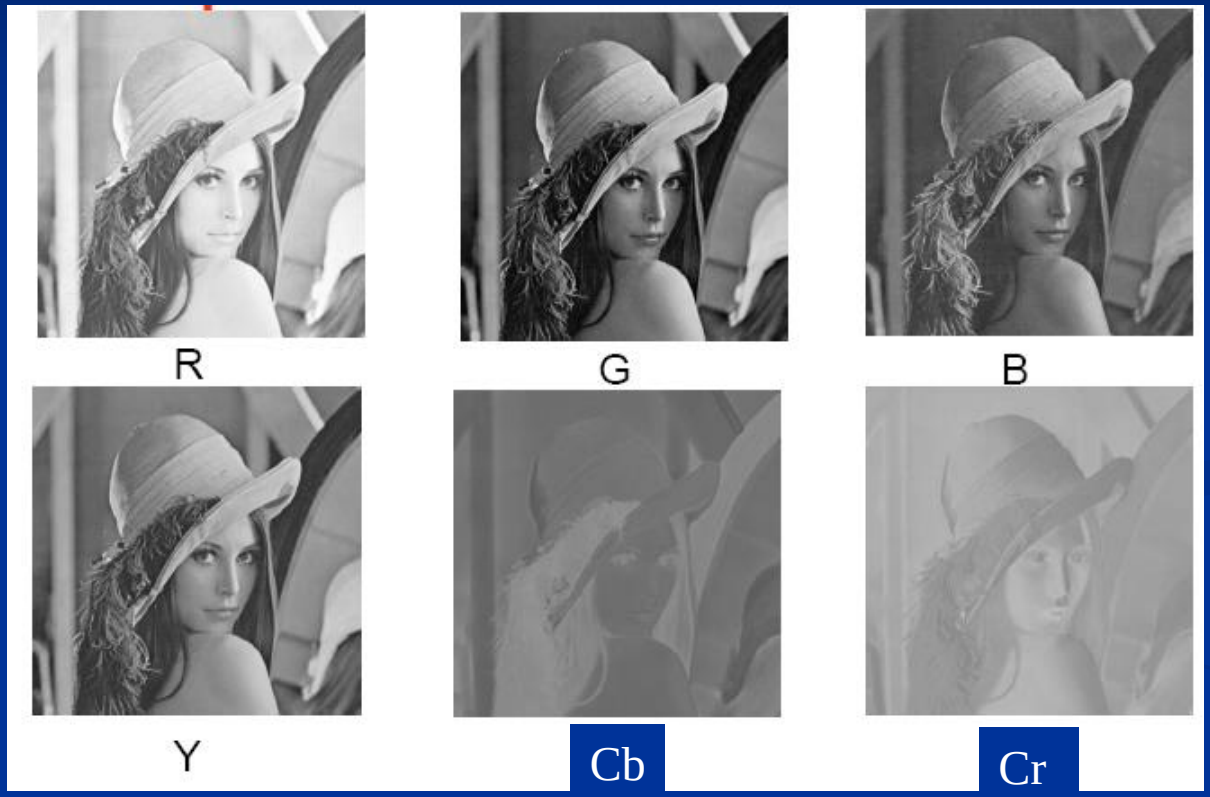
- JPEG标准本身并没有规定具体的颜色空间，只是对各分量分别进行编码
- 实现中通常将高度相关RGB颜色空间转换到相关性较小的YCbCr颜色空间
- RGB→YCbCr（8bit/pixel）

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{pmatrix} \begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

- YCbCr→RGB

$$\begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{pmatrix} \begin{pmatrix} Y \\ Cb-128 \\ Cr-128 \end{pmatrix}$$

颜色空间



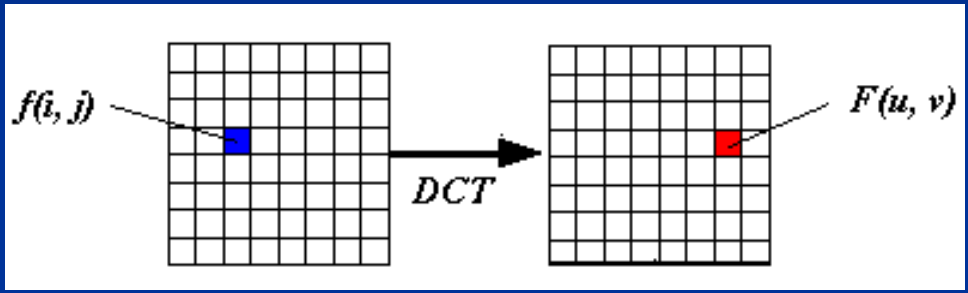
- 图像的主要信息包括在Y通道
  - Cb、Cr更平滑→容易压缩
- 人眼对色度分量不敏感
  - 对色度分量可以进行下采样：如4:2:2, 4:2:0

零偏置（Level Offset）

- 对于灰度级是2^n的像素，通过减去2^{n-1}，将无符号的整数值变成有符号数
  - 对于n=8，即将0~255的值域，通过减去128，转换为值域在128~127]内
- 目的：使像素的绝对值出现3位10进制的概率大大减少

DCT变换

- 对每个单独的彩色图像分量，把整个分量图像分成8×8的图像块，如图所示，并作为两维离散余弦变换DCT的输入



DCT变换：

$$F(u,v)=\frac{1}{4}C(u)C\left(v\left[\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}\right]\right)$$

逆变换：

$$f(i,j)=\frac{1}{4}C(u)C\left(v\left[\sum_{u=0}^{7}\sum_{v=0}^{7}F(u,v)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}\right]\right)$$

其中 $\begin{cases} C(u),C(v)=1/\sqrt{2} & u,v=0 \\ C(u),C(v)=1 & otherwise \end{cases}$

量化

DCT系数

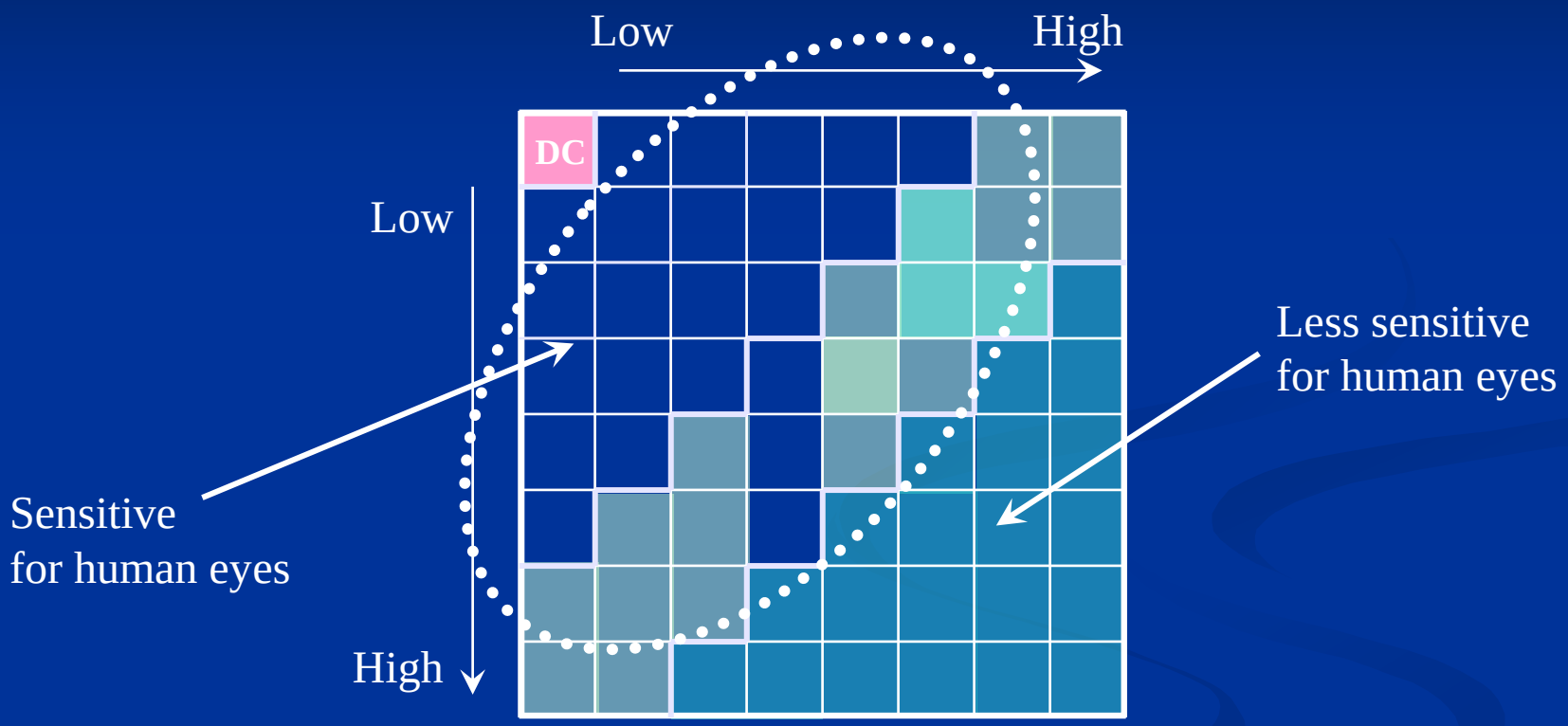
$$l(k)=\left\lfloor \frac{\theta(k)}{Q(k)}+0.5 \right\rfloor,\ k=0,1,....,63$$

量化步长

- 中平型均匀量化器：
- 量化步长是按照系数
  - 所在的位置
  - 颜色分量

  来确定
- 因为人眼对亮度信号比对色差信号更敏感，因此使用了两种量化表：亮度量化值和色差量化值
- 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
  - 如果原始图象中细节丰富，则去掉的数据较多，量化后的系数与量化前差别
  - 反之，细节少的原始图象在压缩时去掉的数据少些

人眼的对亮度敏感性


Low
High
DC
Low
High
Sensitive
for human eyes
Less sensitive
for human eyes

建议基本量化表

- 基于人的生理感知阈值实验

- Luminance

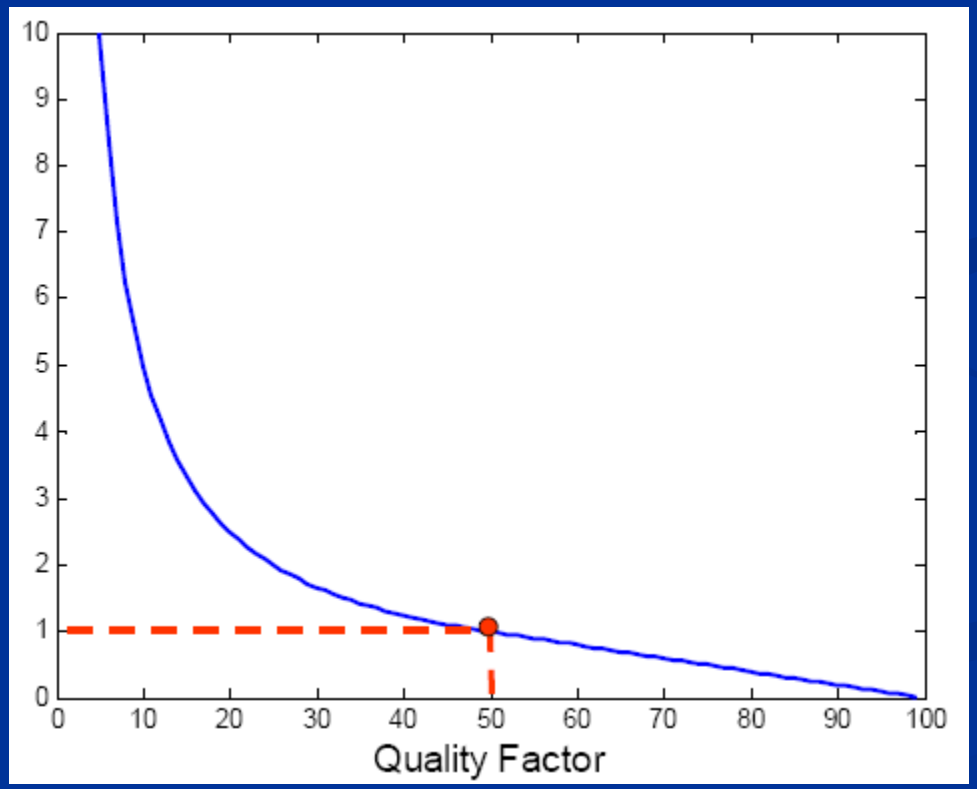
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

- Chrominance, subsampled 2:1

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

量化表缩放

- 真正的量化表=缩放因子×基本量化表
  - 质量因子≤ 50：缩放因子= 50 / 质量;
  - 质量因子> 50：缩放因子 = 2 – 质量 / 50



Quality Factor	Scaling
10	5.0
20	2.5
50	1.0
75	0.5

不同质量因子的图像示例

http://www.cs.sfu.ca/CC/365/mark/material/cgi-bin/whichjpeg.cgi

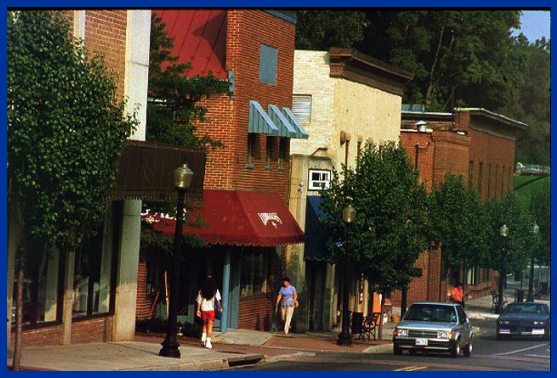
- GIF：258898 bytes

9438 bytes

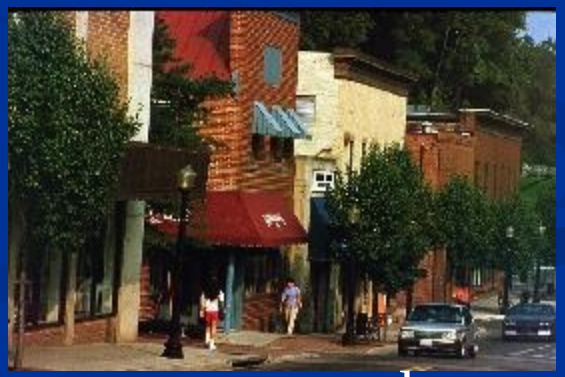
10: 15325 bytes

25: 29360 bytes

50: 46295 bytes

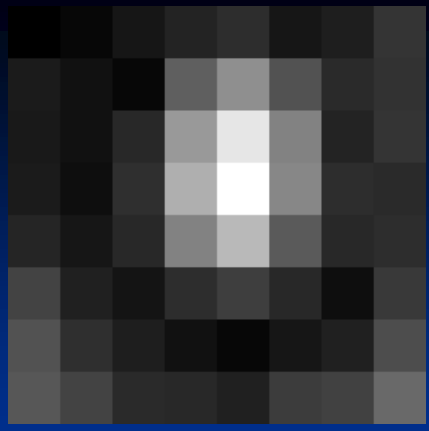


75: 70586 bytes



100: 326321bytes

例：



用8x8的JEPG基线标准，压缩并重构下列子图

52	55	61	66	70	61	64	73
63	59	66	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

0偏置转换后

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-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	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

正向DCT变换（$n = 8$）后变成

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

量化变换后的数组

-26	-3	-6	2	2	-1	0	0
1	-2	-4	1	1	0	0	0
-31	5	-1	-1	0	0	0	0
-41	2	-1	0	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

DC系数的差分编码

- 8×8图像块经过DCT变换之后得到的DC直流系数有两个特点
  - 系数的数值比较大
  - 相邻8×8图像块的DC系数值变化不大：冗余

- 根据这个特点，JPEG算法使用了差分脉冲调制编码(DPCM)技术，对相邻图像块之间量化DC系数的差值DIFF进行编码：

$$DIFF_k = DC_k - DC_{k-1}$$

DC系数的差分编码

- 对DIFF用Huffman编码：分成类别，类似指数Golomb编码
  - 类别ID：一元码编码
  - 类内索引：采用定长码

DC类别	范围	范围大小
0	0	1
1	-1, 1	2
2	-3, -2, 2, 3	4
3	-7, -6, -5, -4, 4, 5, 6, 7	8
4	-15, …, -8, 8, …, 15	16
5	-31, …, -16, 16, …, 31	32
…	…	…
15	[-32767, -16384], [16384, 32767]	32768

DC系数的差分编码

- $DIFF_k = DC_k - DC_{k-1}$

DC Cat.	Prediction Errors	Codeword
0	0	010
1	-1, 1	011x
2	-3, -2, 2, 3	100xx
3	-7, -6, -5, -4, 4, 5, 6, 7	00xxx
4	-15, …, -8, 8, …, 15	101xxxx
5	-31, …, -16, 16, …, 31	110xxxxx
6	-63, …, -32, 32, …, 63	1110xxxxxx
…	…	…

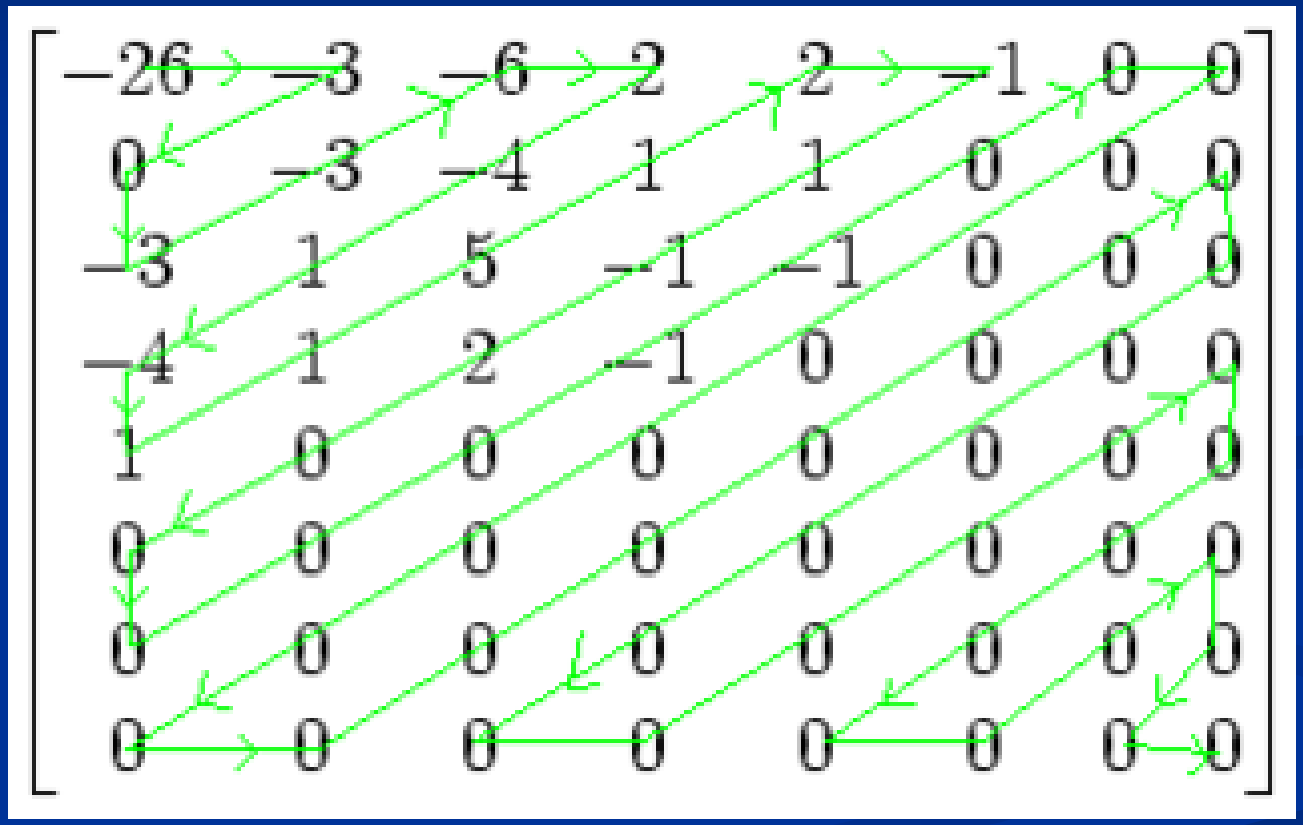
xxx: fixed length index within each category.

例：DC=8，上一DC=5，则DIFF=8-5=3
类别ID=2，类内索引=3，则码流=10011

AC系数的Z字扫描

- 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 (End of Block)即可。

Zig-zag 扫描



26 −3 0 −3 −2 −6 2 −4 1 −4 1 1 5 1 2 −1 1 −1 2 0 0 0 0 0 −1 −1 EOB

AC系数的游程编码

- 在JPEG和MPEG编码中规定为：(run, level)
  - 表示连续run个0，后面跟值为level的系数
  - 如： −3 0 −3 −2 −6 2 −4 1 −4 1 1 5 1 2 −1 1 −1 2 0 0 0 0 0 −1 −1 EOB
  - 表示为(0,-3) ; (1,-3) ; (0,-2) ; (0,-6) ; (0,2) ; (0,-4) ; (0,1) ; (0,-4) ; (0,1) (0,1) ; (0,5) ; (0,1); (0,2) ; (0,-1) ; (0,1) ; (0,-1) ; (0,2) ; (5,-1) ; (0,-1) ; EOB

- 编码：
  - Run: 最多15个，用4位表示Z
  - Level：类似DC
    - 分成16个类别，用4位表示表示类别号C
    - 类内索引
  - 对(Z, C)联合用Huffman编码
  - 对类内索引用定长码编码

AC系数的Huffman编码

- Z：0的游程；　C：类别

Z/C	Codeword	Z/C	Codeword	…	Z/C	Codeword
0/0 (EOB)	1010			…	F/0 (ZRL)	1111111001
0/1	00	1/1	1100	…	F/1	1111111111110101
0/2	01	1/2	11011	…	F/2	1111111111110110
0/3	100	1/3	1111001	…	F/3	1111111111110111
0/4	1011	1/4	11111001 0	…	F/4	1111111111111000
0/5	11010	1/5	11111110110	…	F/5	1111111111111001
⋮	⋮	⋮	⋮		⋮	

- ZRL：表示16个0；当0的个数大于15时，分成多次
  - 如20个0，紧跟-1：ZRL; (4,-1)
- (run, level) 序列：(0,-3) ; (1,-3) ; ...
  - Z/C序列：0/2, 1/2, …
  - -3是第2类的第1个值，(0, -3): 01 00
  - -3是第2类的第1个值，(1, -3): 11011 00

重构

- 与编码相反
  - 解码Huffman数据
  - 解码DC差值
  - 重构量化后的系数
  - DCT逆变换
  - 丢弃填充的行/列
  - 反0偏置
  - 对丢失的CbCr分量差值（下采样的逆过程）
  - YCbCr → RGB

重构

- 量化后的系数（已通过DC差重构DC系数）

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- 乘以量化表，得到

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- IDCT:

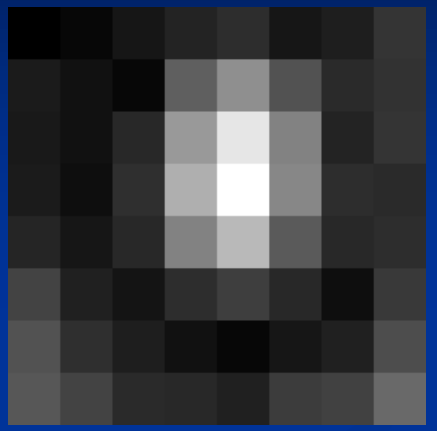
$$\begin{bmatrix} -68 & -65 & -73 & -70 & -58 & -67 & -70 & -48 \\ -70 & -72 & -72 & -45 & -20 & -40 & -65 & -57 \\ -68 & -76 & -66 & -15 & 22 & -12 & -58 & -61 \\ -62 & -72 & -60 & -6 & 28 & -12 & -59 & -56 \\ -59 & -66 & -63 & -28 & -8 & -42 & -69 & -52 \\ -60 & -60 & -67 & -60 & -50 & -68 & -75 & -50 \\ -54 & -46 & -61 & -74 & -65 & -64 & -63 & -45 \\ -45 & -32 & -51 & -72 & -58 & -45 & -45 & -39 \end{bmatrix}$$

- +128:

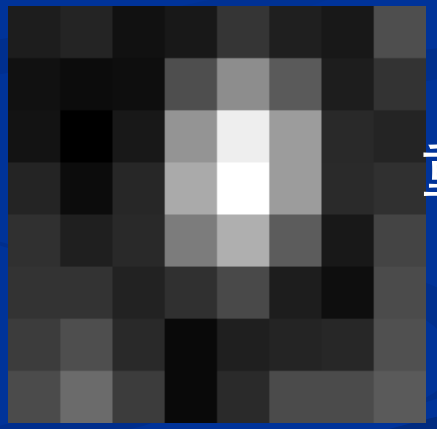
$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$

- 重构误差：

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



原图

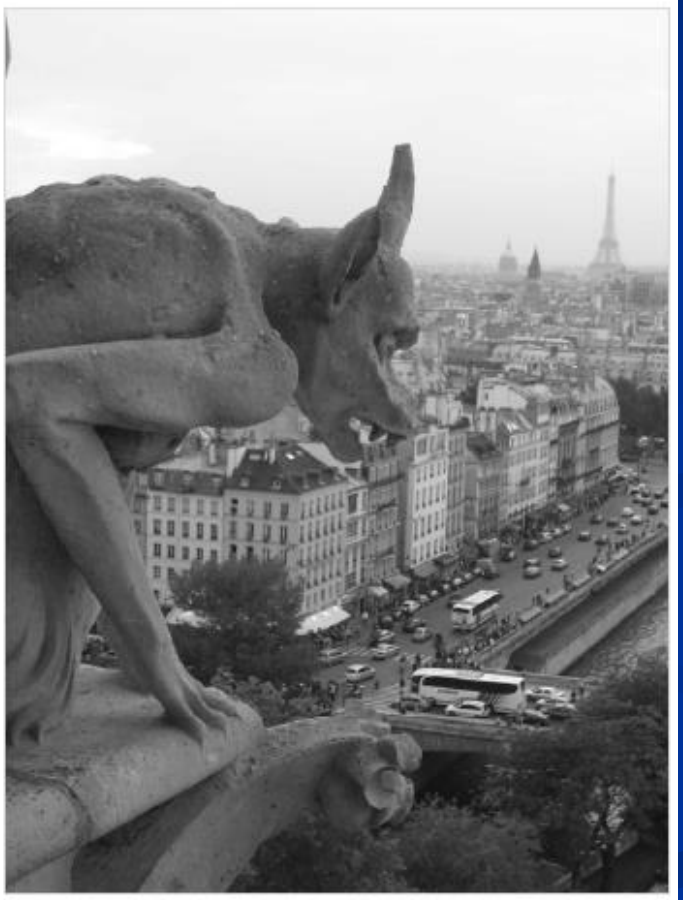


重构图

- 每个像素大约为5的平均绝对误差
  - 误差在左下角比较明显

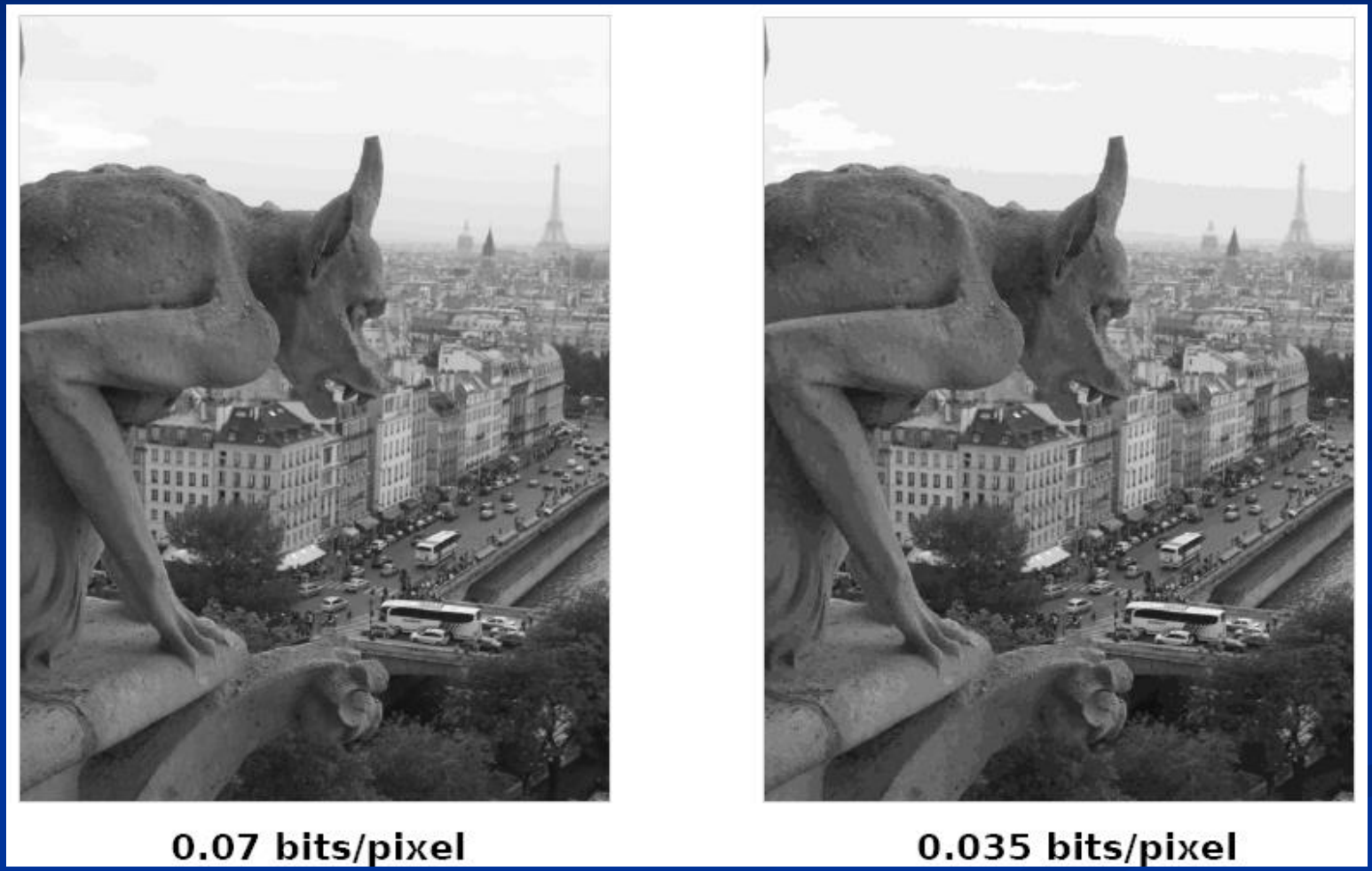
示例图像

0.56 bits/pixel



0.14 bits/pixel

示例图像



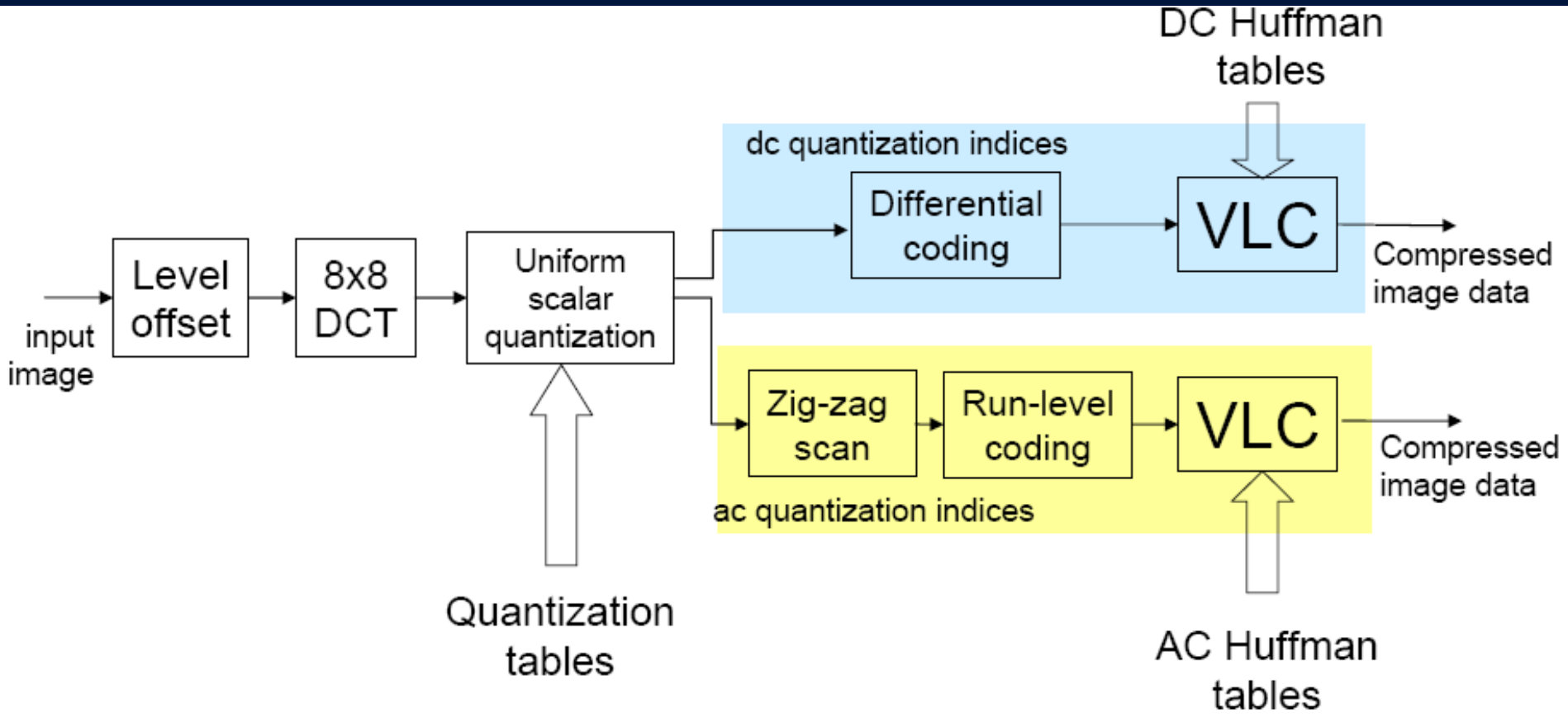
0.07 bits/pixel

0.035 bits/pixel

JPEG文件格式http://en.wikipedia.org/wiki/JPEG

- JPEG Interchange Format' (JIF)
  - specified in Annex B of the standard
  - 考虑所有的模式，过于复杂，很少应用
- JPEG File Interchange Format (JFIF)
  - 是JIF的缩减版本
  - 由C-Cube Microsystems开发
- Exchangeable image file format(EXIF)
  - 由Japan Electronic Industries Development Association开发
  - 保存了丰富的metadata

JPEG Baseline 系统分析



- 两张量化表 – 亮度和色度
- 一般四张Huffman码表
  - 亮度DC，亮度AC，色度DC，色度AC

如何保存

JPEG标准中的语法结构

- 定义了一系列标记（Markers）
  - 均以0xFF开始，后跟1字节的标记标识符和2字节的标记长度以及该标记所对应的payload
  - 标记长度部分高位在前，低位在后，不包含该标记的头两个字节
  - 熵编码部分的数据在0xFF后由编码器插入0x00，解码器解码时跳过此字节不予处理
  - SOI(Start Of Image)和EOI(End Of Image)标记没有payload

Common JPEG markers

Short name	Bytes	Payload	Name	Comments
SOI	0xFFD8	*none*	Start Of Image	
SOF0	0xFFC0	*variable size*	Start Of Frame (Baseline DCT)	Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and subsampling (e.g., 4:2:0).
SOF2	0xFFC2	*variable size*	Start Of Frame (Progressive DCT)	Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, subsampling (e.g., 4:2:0).
DHT	0xFFC4	*variable size*	Define Huffman Table (s)	Specifies one or more huffman tables.
DQT	0xFFDB	*variable size*	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFFDD	2 bytes	Define Restart Interval	Specifies the interval between RST*n* markers, in macroblocks.
SOS	0xFFDA	*variable size*	Start Of Scan	Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Pr JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is in followed by entropy-coded data.
RST*n*	0xFFD*n*	*variable size*	Restart	Inserted every *r* macroblocks, where *r* is the restart interval set by a DRI marker. Not used if there was no DRI low 4 bits of the marker code, cycles from 0 to 7.
APP*n*	0xFFE*n*	*variable size*	Application-specific	For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely
COM	0xFFFE	*variable size*	Comment	Contains a text comment.
EOI	0xFFD9	*none*	End Of Image	

JPEG文件的解析

- 由若干个必不可少的标记顺序连接构成整个文件
- 一定以0xFFD8开始，即表示图像开始SOI (Start of Image)标记
- 一定以0xFFD9结束，表示图像结束EOI(End of Image)标记

二、APP0 Segment（0xFFE0）		
2字节	APP0块的长度	
5字节	“JFIF”+”0”	
1字节	<Major version>	
1字节	<Minor version>	
1字节	<Units for the X and Y densities>	Units=0无单位 Units=1 DPI Units=2 点/厘米
2字节	<X density>	水平方向像素密度
2字节	<Y density>	垂直方向像素密度
1字节	<X thumbnail>	缩略图水平像素数目
1字节	<Y thumbnail>	缩略图垂直像素数目
3n	<Thumbnail RGB bitmap>	缩略RGB位图，n为像素数

三、APPn标记，数值0xFF E1~0xFFEF

- N=1~15,数值对应0xE1~0xEF
- APPn长度（Length）
- 应用细节信息（Application specific information）
- EXIF文件格式中多写入0xFF E1

四、量化表DQT，数值0xDB

- 一般为两个量化表，即亮度和色度各一张
- 以0xFFDB开始
  - 量化表长度，一般为00 43（或00 84）
  - 量化表信息（1字节）
    - Bit 0~3　QT号（只能取值为0~3，否则错误）
    - Bit 4~7　QT精度（0为8比特，否则表示16比特）
  - 量化表的实际数据
    - 量化表中的数据按照Z字形保存量化表内8x8的数据

五、帧图像开始SOF0,数值0xFF C0

SOF长度(2字节)	0x00 11
精度(1字节)	每个颜色分量每个像素的位数（通常为8）
图像高度(2字节)	以像素数表示图像的高度
图像宽度(2字节)	以像素数表示图像的宽度
颜色分量数(1字节)	通常为3
对每个颜色分量：	颜色索引ID(1字节,从01,02,03)
	Sample factor(1字节， 高四位水平因子，低四位垂直因子)
	量化表号(1字节)

六、一个或多个Huffman表，数值0xFF C4

- Huffman表的长度
- 类型（AC或DC）
- 索引（Index）
- 位表（bit table）
- 值表（value table）

Huffman表存储方式举例说明

- **FF C4 00 1D 00 00 03 01 01 01 01 01 01 01 00 00 00 00 00 00 00 04 05 06 03 02 01 00 09 07 08**
- **红色部分** 为哈夫曼表ID和表类型，其值0x00表示此部分数据描述的是DC直流0号表。
- **灰色部分**（16个字节）为不同位数的码字的数量。这16个数值实际意义为：没有1位的哈夫曼码字；2位的码字有3个；3位-9位的码字各有1个；没有9位或以上的码字。
- **绿色部分**（10个字节）为编码内容。由蓝色部分数据知道，此哈夫曼树有0+3+1+1+1+1+1+1+1=10个叶子结点，即本字段应该有10个字节。这段数据表示10个叶子结点按从小到大排列，其权值依次为**04**、**05**、**06**、**03**、**02**、**01**、**00**、**09**、**07**、**08**（16进制）
- 见JPEG和Trace文件。

Huffman表存储方式举例说明

- FF C4 00 3E 10 00 01 02 05 03 03 03 02 05 03 04 02 02 02 01 05 00 01 03 02 04 05 11 21 22 31 61 06 12 A1 32 41 62 13 51 23 42 71 81 91 15 52 63 07 14 33 53 16 43 08 B1 34 C1 24 D1 09 72 F0 A2
- 红色部分（1字节）为哈夫曼表ID和表类型，其值0x10表示此部分数据描述的是AC交流0号表。
- 灰色部分（16字节）为不同位数的码字的数量。这16个数值实际意义为：没有1位的哈夫曼码字……。
- 绿色部分（3E-16-2-1=43字节）为编码内容。由蓝色部分数据知道，此哈夫曼树有（绿色数据相加）=43个叶子结点，即本字段应该有43个字节。这段数据表示43个叶子结点按从小到大排列，其权值依次为（16进制）
- 看具体JPEG和Trace文件。

建立huffman树/表

- 在读出哈夫曼表的数据后，就要建立哈夫曼树。具体方法为：

1）第一个码字必定为0。
如果第一个码字位数为1，则码字为0；
如果第一个码字位数为2，则码字为00；
如此类推。

2）从第二个码字开始，
如果它和它前面的码字位数相同，则当前码字为它前面的码字加1；
如果它的位数比它前面的码字位数大，则当前码字是前面的码字加1后再在后边添若干个0，直至满足位数长度为止。

建立huffman树/表

- 利用第一个例子：
- FF C4 00 1D 00 00 03
  01 01 01 01 01 01 01
  00 00 00 00 00 00 04
  05 06 03 02 01 00 09
  07 08

码字： 加一（补0）

第一个码字必定为0，当前码长有两位

01表示只有一个码长只有1个码字

由00 03得到前三个码字码长都为2

权值顺序写入

序号	码长	码字	权值
1	2	00	04
2	2	01	05
3	2	10	06
4	3	110	03
5	4	1110	02
6	5	11110	01
7	6	111110	00
8	7	1111110	09
9	8	11111110	07
10	9	111111110	08

建立huffman树/表（直流）

- 刚才建立的是DC(0)huffman表，其权值就是解码时再需要读入的bit位数。这个再次读入的位数通过查表得到真正的码值。

  例如：0110101011

根据刚建立的huffman表分解： 01,10101011.

码字01 的权值为05.则再读取5位。 01,10101，011

这5位10101进行译码为：21. 表示直流系数为21.

注意：直流系数是差分编码的

直流系数的差分编码

- **直流系数的差分编码**
- 把所有的颜色分量单元按颜色分量（**Y**、**Cr**、**Cb**）分类。每一种颜色分量内，相邻的两个颜色分量单元的直流变量是以差分来编码的。也就是说，通过步骤**3**解码出来的直流变量数值只是当前颜色分量单元的实际直流变量减去前一个颜色分量单元的实际直流变量。也就是说，当前直流变量要通过前一个颜色分量单元的实际（非解码）直流分量来校正：
- **DCn=DCn-1+Diff**
- 其中**Diff**为差分校正变量，也就是直接解码出来的直流系数。但如果当前颜色分量单元是第一个单元，则解码出来的直流数值就是真正的直流变量。

建立huffman树/表（交流）

- 对于交流系数，用交流哈夫曼树/表查得该码字对应的权值。权值的高4位表示当前数值前面有多少个连续的零，低4位表示该交流分量数值的二进制位数，也就是接下来需要读入的位数。
- 例如：权值为0X31.可表示为（3，1）。表明此交流系数前有3个0，而此交流系数的具体值还需要再读入1个bit的码字，才能得到。
- 具体见Trace文件。

解码一个JPEG文件

- Start of Image SOI
  - Set image started to true.
- An APP0 Marker
  - Get the APP0 length.
  - Check if the Identifier is JFIF
  - Skip over any extensions.
  - Get the Version (This will be ignored)
  - Get the Units for X & Y densities (This will be ignored)
  - Get the X density (This will be ignored)
  - Get the Y density (This will be ignored)
  - Get the Thumbnail horizontal pixels
  - Get the Thumbnail vertical pixels
  - Skip over Thumbnail RGB bitmap, if there is one.
  - Skip to the end of this section if we are not already there.

解码一个JPEG文件

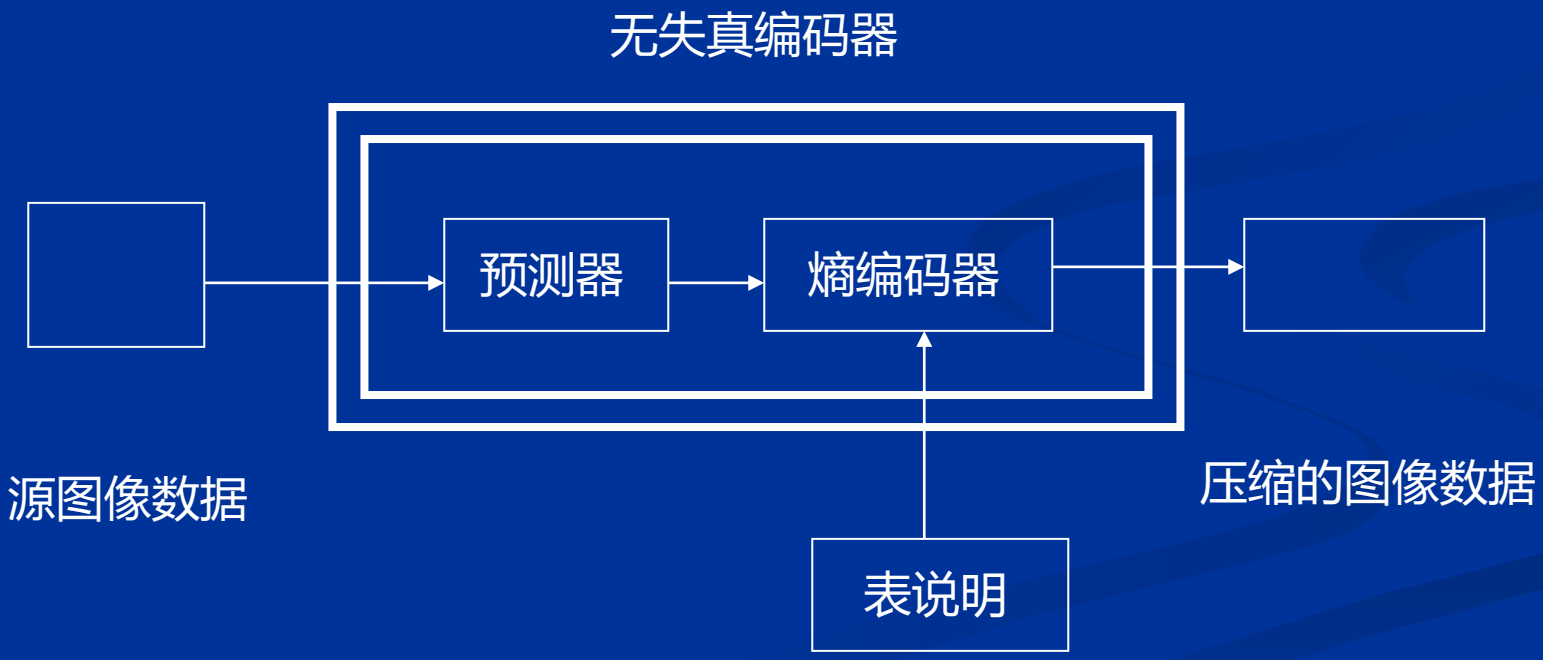
- Quantization table DQT
  - Get the Quantization table length.
  - Get the Quantization table number.
    - This will be used to determine which component will use this table. See Start of Frame.
  - Get the 64 entries in the Quantization table.
- Start of Frame SOF0
  - Get the Start of Frame length
  - Check that the Precision is 8 Bits per pixel per color component.
  - Get the Image height
  - Get the Image width
  - Check that the Number of color components is 3.
  - For each component
    - Get An ID (This will be ignored)
    - Get A vertical sample factor (Low Nibble)
    - Get A horizontal sample factor (High Nibble)
    - Get A quantization table# See Quantization table above.

解码一个JPEG文件

- Huffman table DHT
  - Get the Huffman table length.
  - Get the Type, AC or DC (1 -> AC)
    - Setup table pointer to table of correct type , AC or DC.
  - Get the Index.
    - This will be used to determine which component will use this table. See Start of Scan.
  - Get the Bits table & add up the entries to determine how many entries are in the value table.
  - Get the Value table.

基于DPCM的无损编码模式(旧的JPEG标准)

- 采用三邻域一维/二维预测编码和熵编码
  - 提供8种可供用户选择的预测方案- 第一种是不预测
  - 3种是一维预测，4种是二维预测

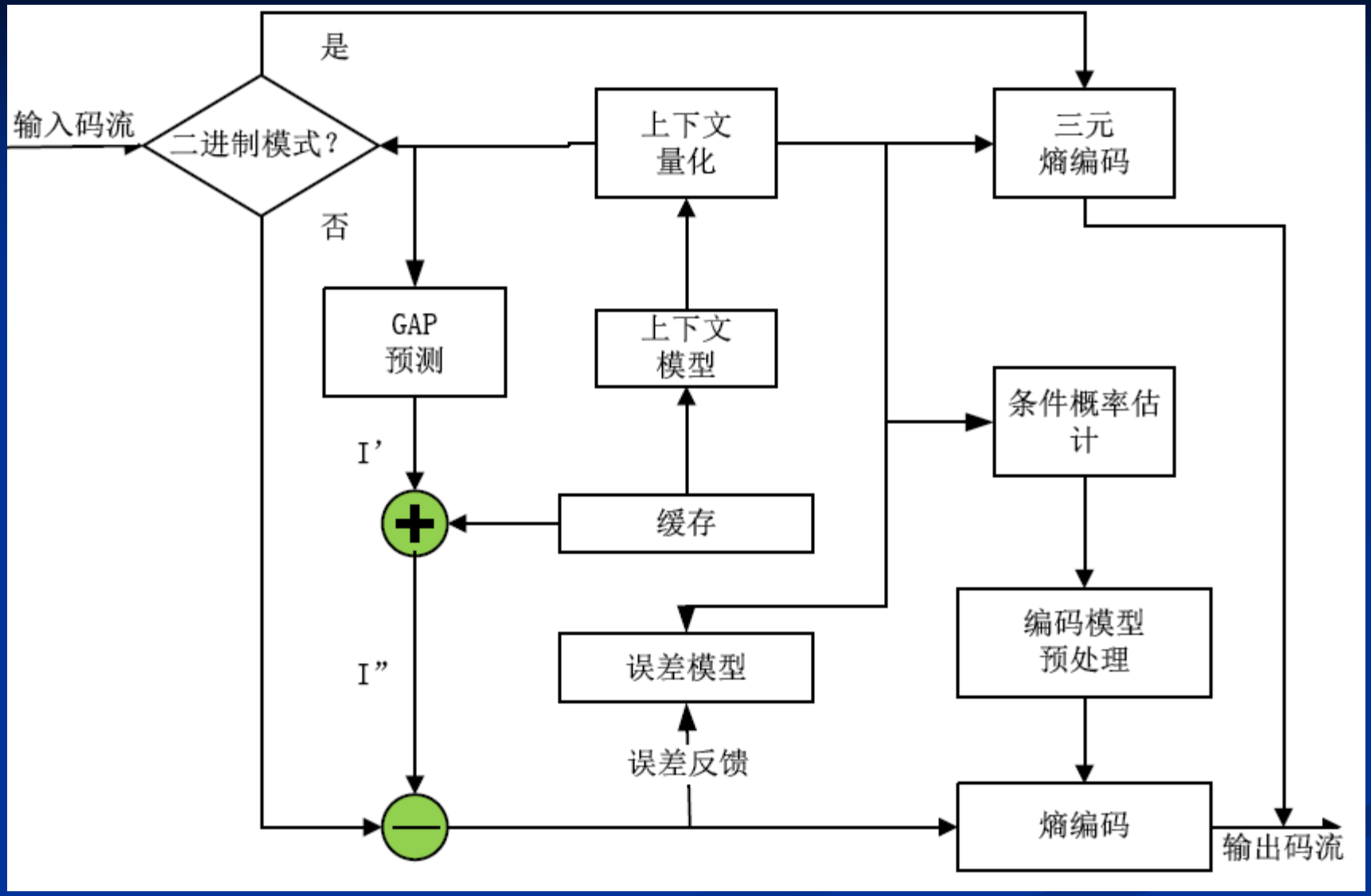


DPCM预测编码框图

CALIC

- 观察
  - 一个给定像素的取值通常与其某个相邻像素很接近
  - 究竟与哪个像素更为接近，取决于图像的局部结构
  - 预测值与待编码像素的接近程度，取决于周围的纹理结构
- 结论
  - 算法必须对编码像素的环境做出判断
  - 编码器和解码器都可以获得做出这一判断所需要的信息
  - 给定上下文，某些符号的发生概率要远高于(或远低于)没有上下文的情况。即给定上下文时的概率分布会更不均匀—>条件熵更小—>压缩效率更高

CALIC



两种工作模式：二进制模式，连续色调模式

CALIC

初始预测值

		NN	NNE
	NW	N	NE
WW	W	X	

- 判断X的附近是否可能存在某种边界

$$d_h = |W - WW| + |N - NW| + |NE - N|$$

水平梯度

$$d_v = |W - NW| + |N - NN| + |NE - NNE|$$

垂直梯度

  - 如果d_h>>dv，则存在很大的水平变化量；最好选择N作预测
  - 如果$dv >> d_h$，则存在很大的垂直变化量；最好选择W作预测
  - 如果这些差值都比较适中或者更小，则预测值可以取这些相邻像素的加权平均值

CALIC

		NN	NNE
	NW	N	NE
WW	W	X	

预测参数和门限是实验性选择。通常对一幅或一类图像通过使 $E|I-\hat{I}|$ 最小，来最优化参数和门限。

对8比特灰度图像

$$
\begin{aligned}
&d_v - d_h > 80, && \text{锋锐水平} \quad \hat{I} = I_w \\
&d_v - d_h < -80, && \text{锋锐垂直} \quad \hat{I} = I_n \\
&else \quad \hat{I} = \frac{(I_w + I_n)}{2} + \frac{I_{ne} - I_{nw}}{4} \\
&\begin{cases}
if \quad d_v - d_h > 32, & \text{水平} \quad \hat{I} = (\hat{I} + I_w)/2 \\
elseif \quad d_v - d_h > 8, & \text{微弱水平} \quad \hat{I} = (3\hat{I} + I_w)/4 \\
elseif \quad d_v - d_h < -32, & \text{垂直} \quad \hat{I} = (\hat{I} + I_w)/2 \\
elseif \quad d_v - d_h < -8, & \text{微弱垂直} \quad \hat{I} = (3\hat{I} + I_n)/4
\end{cases}
\end{aligned}
$$

编码上下文选择和量化

- Error energy（误差能量）估计

$$C_1 = ad_h + bd_v + c|e_w|$$

  - 引入e_w是因为大的误差趋于连续发生
- C_1用于计算条件概率
- $P(e|C_1)$用于之后的熵编码
  - 以C_1的分布为条件，可以根据不同的值把预测误差区分为若干类。这样，在熵编码中使用了条件概率$P(e|C_1)$比使用$P(e)$更增加编码效率。
  - 把C_1进行分类，分的标准是使误差的条件熵最小。

编码上下文选择和量化

- Error energy估计

$$C_1 = ad_h + bd_v + c|e_w|$$

- 最佳系数a、b、c的计算
  - 在对测试图像进行训练后，通过选择令

$$F = \sum_{S}(|e| - c_1)^2 = \sum_{S}(|e| - ad_k - bd_v - c|e_w|)^2$$

  - 最小，来求出最佳系数。**S**代表多幅图像像素的集合

$$\frac{dF}{da} = -2d_k \sum_{S}(|e| - ad_k - bd_v - c|e_w|) = 0$$

$$\frac{dF}{db} = -2d_v \sum_{S}(|e| - ad_k - bd_v - c|e_w|) = 0$$

$$\frac{dF}{dc} = -2|e_w| \sum_{S}(|e| - ad_k - bd_v - c|e_w|) = 0$$

编码上下文选择和量化

- Error energy估计

$$C_1 = ad_h + bd_v + c|e_w|$$

- 最佳系数a、b、c的计算

$$\begin{bmatrix} \sum d_k^2 & \sum d_k d_v & \sum d_k |e_w| \\ \sum d_v d_k & \sum d_v^2 & \sum d_v |e_w| \\ \sum |e_w| d_k & \sum |e_w| d_v & \sum (e_w)^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum d_k |e| \\ \sum d_v |e| \\ \sum |e_w| |e| \end{bmatrix}$$

a=1　b=1　　c=2

编码上下文选择和量化

- Q: 量化器 $C_1 = ad_h + bd_v + c|e_w|$

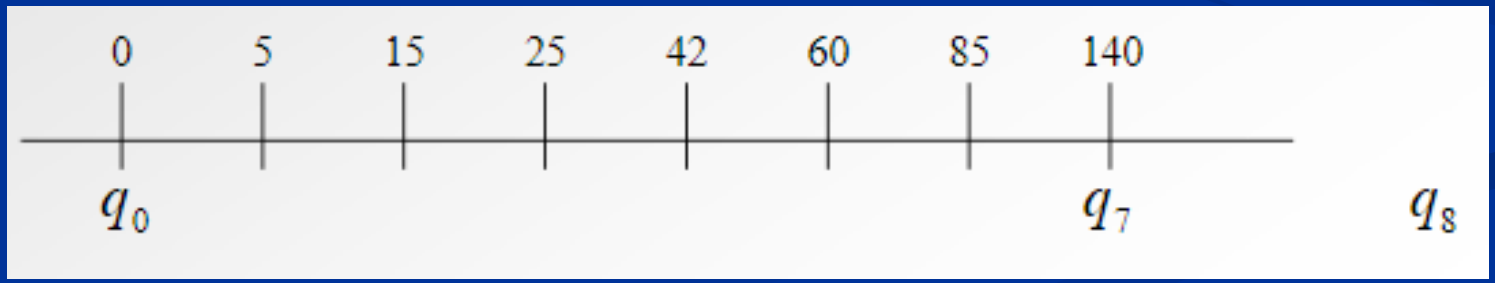
由于过多的条件概率参数)需要估计而图像样本却有限，因此容易导致“上下文稀释”。所以采用上下文量化对相似的上下文进行合并，以达到降低上下文的数量

- 将C_1划分$\{0, 1, 2, \ldots, 7\}$

$$0 \le q_0 < q_1 < \ldots q_{L-1} < q_L = \infty, \text{ where L=8}$$

- 如何划分?
  - 分的标准是使误差的条件熵最小。

$$-\sum_e p(e) log p(e|q_d \le C_1 \le q_{d+1})$$



结构上下文

结构上下文B用来探索纹理结构，它暗示了误差e的形式。

		NN	NNE
	NW	N	NE
WW	W	X	

[N，W，NW，NE，NN，WW，2N-NW，2W-WW

$$C = \{x_0, x_1, \ldots, x_6, x_7\}$$
$$= \{I_n, I_w, I_{nw}, I_{ne}, I_{nn}, I_{ww}, 2I_n - I_{nn}, 2I_w - I_{ww}\}$$
$$b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7$$

$$\beta = b_7, b_6, \ldots, b_1, b_0$$

$$b_k = \begin{cases} 0 & \text{if } x_k \geq \hat{I} \\ 1 & \text{if } x_k < \hat{I} \end{cases} \qquad 0 \leq k < K = 8$$

结构上下文

		NN	NNE
	NW	N	NE
WW	W	X	

$[N，W，NW，NE，NN，WW，2N\text{-}NW，2W\text{-}WW$

- 在模型上下文中，x_i表示的并不一定必须是I周围的像素，也可能是周围像素的组合。就像b_6，b_7一样，分别表示的是被预测值I_1和周围像素在垂直和水平方向上是否形成一个凸面或凹面。这些凸面或凹面的表示是非常有用的。

结构上下文

$$C = \{x_0, x_1, \ldots, x_6, x_7\}$$
$$= \{I_n, I_w, I_{nw}, I_{ne}, I_{nn}, I_{ww}, 2I_n - I_{nn}, 2I_w - I_{ww}\}$$
$$b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7$$

b_7, b_5, b_1 are not independent

- When $I_n < \hat{I} < I_{nn}$ 或 $I_{nn} < \hat{I} < I_n$ 時
  $2I_n - I_{nn}$ is fixed

b_6, b_4, b_0 are not independent

- When $I_w < \hat{I} < I_{ww}$ 或 $I_{ww} < \hat{I} < I_w$ 時
  $2I_w - I_{ww}$ is fixed

		NN	NNE
	NW	N	NE
WW	W	X	

- b0、b4和b6以及b1、b5、b7之间不是独立的。他们之间只能产生2^3-2=6种结合形式。这样，B代表的仅仅是6*6*4=144种结合形式

综合上下文的形成

- Error energy(误差能量)被划分为4个区间

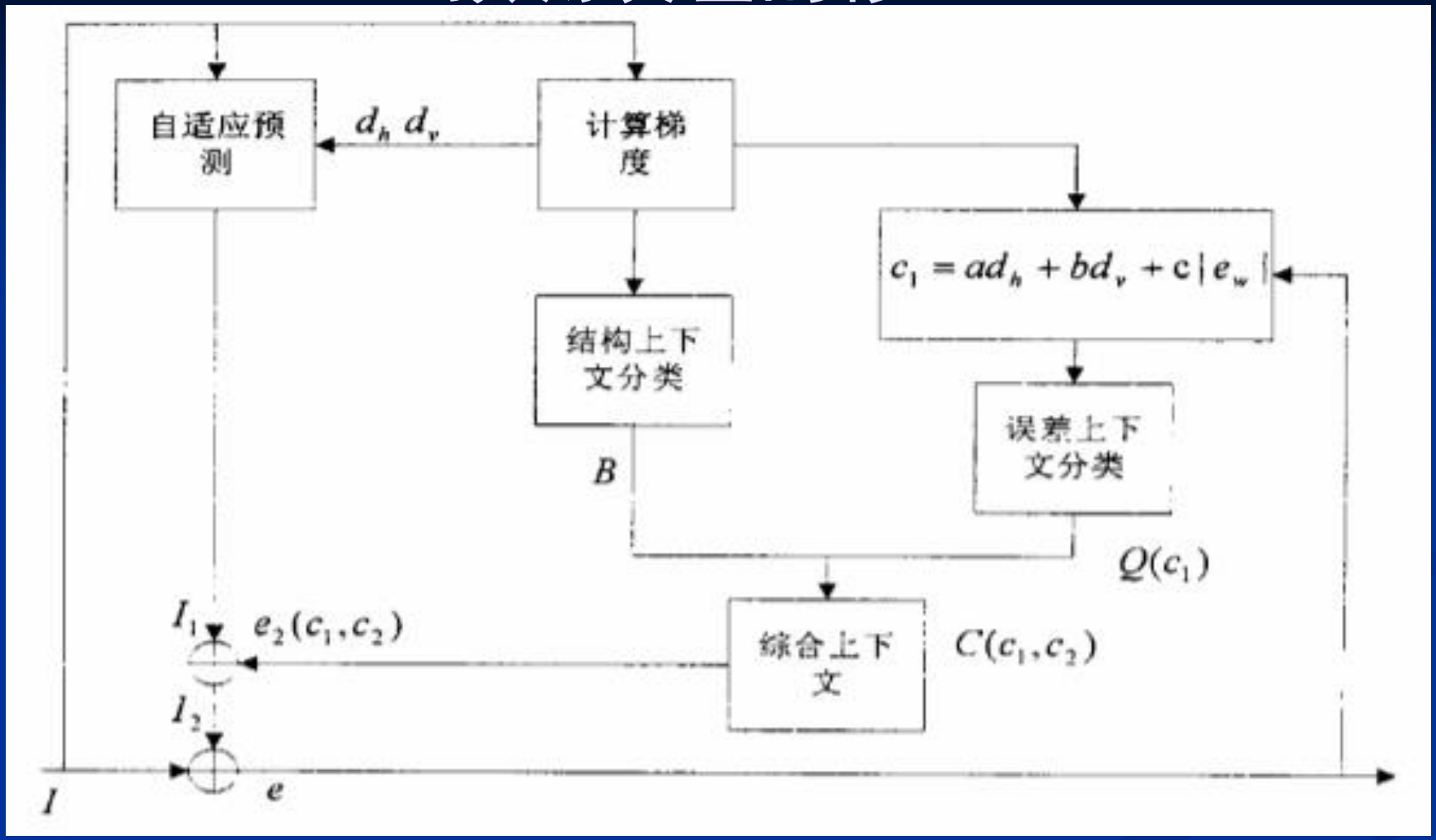
$$0 \le \left|\frac{Q(C_1)}{2}\right| < \frac{L}{2} = 4$$

- 综合上下文共有：144*4=576个
- 对于每个综合的上下文，我们使用上下文样本的均值e_1来估计上下文的条件期望值$E(e|C(c_1,c_2))$。

$$e_2(c_1,c_2) = S(c_1,c_2)/N(c_1,c_2)$$

$N(C_1,C_2)$是对于该上下文出现过的像素数目
$S(C_1,C_2)$是对于该上下文出现过的所有误差值的累加和

预测误差的修正



- 通过一个反馈回路来纠正预测的偏差 $I_2 = I_1 + e_2(c_1, c_2)$
- 通过误差的上下文模型和误差反馈形成了一个两个阶段的自适应预测方案

熵编码

- 该压缩方法把预测和上下文模型与熵编码隔离开，所以从原则上讲，任何熵编码，不管是huffman还是算术编码，不管是静态的还是自适应的，都可以很容易地兼容
- 在图像的自适应预测和上下文补偿时，图像的误差有可能出现小数，进行熵编码前有必要将小数变成整数
- 应充分考虑基于上下文的条件熵

熵编码

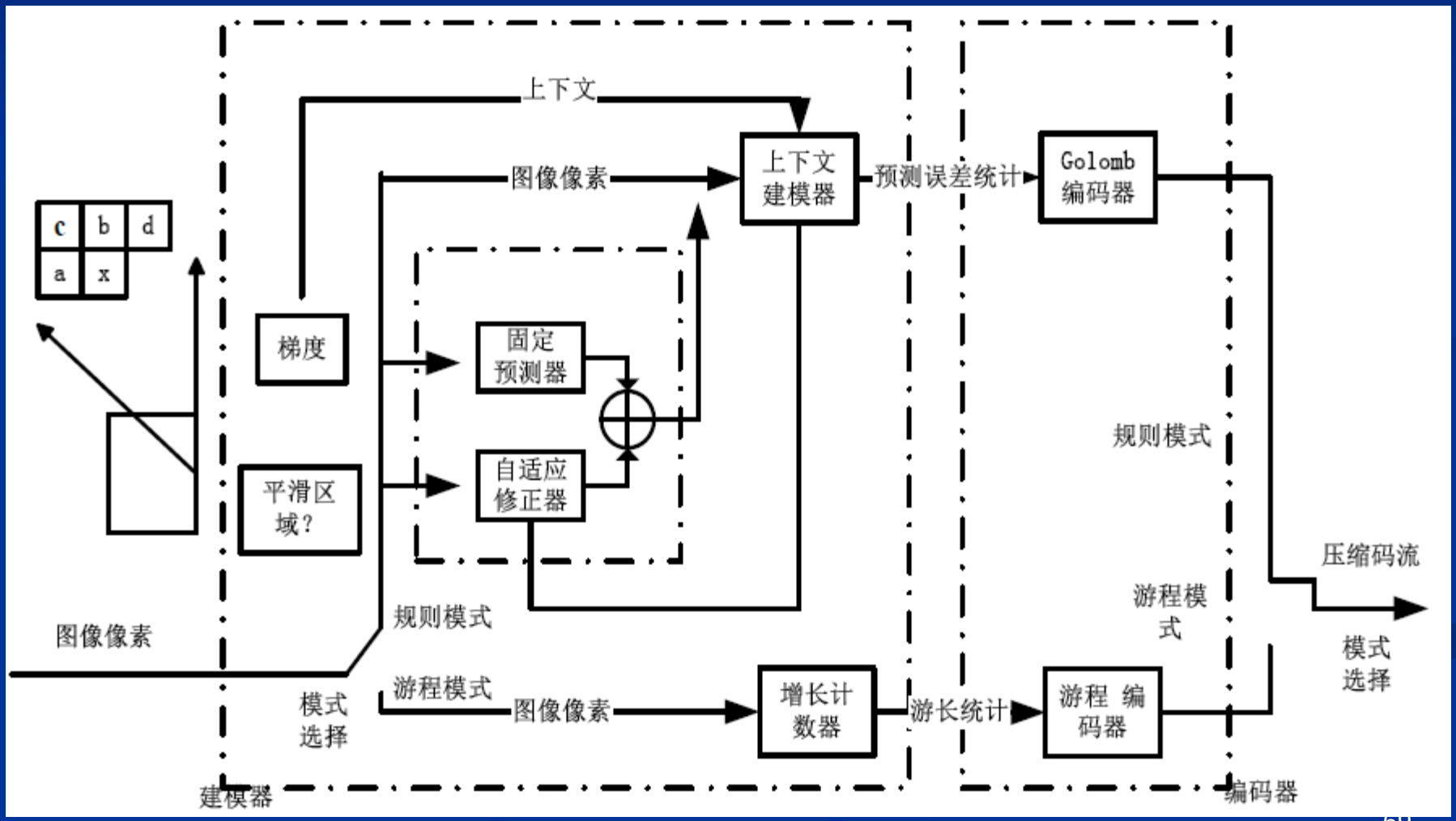
- 误差符号的正负导致的处理
  - 在误差上下文相同而结构上下文不同的两个综合上下文$c(d,t_1)$和$c(d,t_2)$中，他们的均值$e_2(d,t_1)$和$e2(d,t_2)$有可能具有相反的符号，反映的是不同的条件概率$p(e|c(d,t_1))$和$p(e|c(d,t_2))$
- 在对e进行编码之前，先检查$e_2(d,t)$是否小于零，如果小于0，就将e进行反号。如果大于零，就不变。因为解码器也知道c(d,t)和e2(d,t)，可以将符号反转过来。因此它是一个可逆过程。

CALIC Conclusion

- Feature of CALIC
  - Adaptive(Gradient-adjusted)
  - Context-based(Refinement)
  - Lossless
  - Raster-scan(Single-pass)
- Main idea
  - Error feedback
  - Learning from its mistakes under a given context in the past（鉴往知来）

JPEG-LS Lossless and near-lossless compressiion of continuous-tone still images

- 中值自适应预测作为初始预测值
- 用特定上下文中预测误差的平均值改进该初始预测值



基于DCT的递增编码模式

- 此模式与顺序模式编码步骤基本一致
- 不同之处在于：递增模式每个图像分量的编码要经过多次扫描才完成
  - 第一次扫描只进行一次粗糙的压缩，然后根据此数据先重建一幅质量低的图像
  - 以后的扫描再作较细的扫描，使重建图像质量不断提高，直到满意为止
- 递增模式分为两种：
  - 按频段累进
  - 按位累进

基于DCT的分层编码模式

- 1、降低原始图像的空间分辨率
- 2、对已经降低分辨率的图像按照顺序编码模式进行压缩并存储或传输
- 3、对低分辨率图像进行解码，然后用插值法提高图像的分辨率
- 4、将分辨率已经升高的图像作为原图像的预测值，并把它与原图像的差值进行基于DCT的编码
- 5、重复步骤3、4直到图像达到完整的分辨率

总结

- JPEG：
  - 各种基本算法的精妙组合：DCT + DPCM + Huffman
  - 其他标准也类似

References

- G. K. Wallace, “The JPEG still picture compression standard,”IEEE Trans. Consumer Electronics, vol. 38, no. 1, pp. xviii-xxxiv, Feb. 1992.
- TU-T Rec. T.81
  - http://www.itu.int/ITU-T/studygroups/com16/jpeg1x/index.html
- Wiki:
  - http://en.wikipedia.org/wiki/Jpeg