图像压缩编码

一部2小时的高清（1280×720）电影需要多大存储空间？

1280(width)×720(height)×3(channel)×24(fps)×2(hour)×3600(second)÷(2^30)=

445GB

1920×1080？=>1000GB！！！

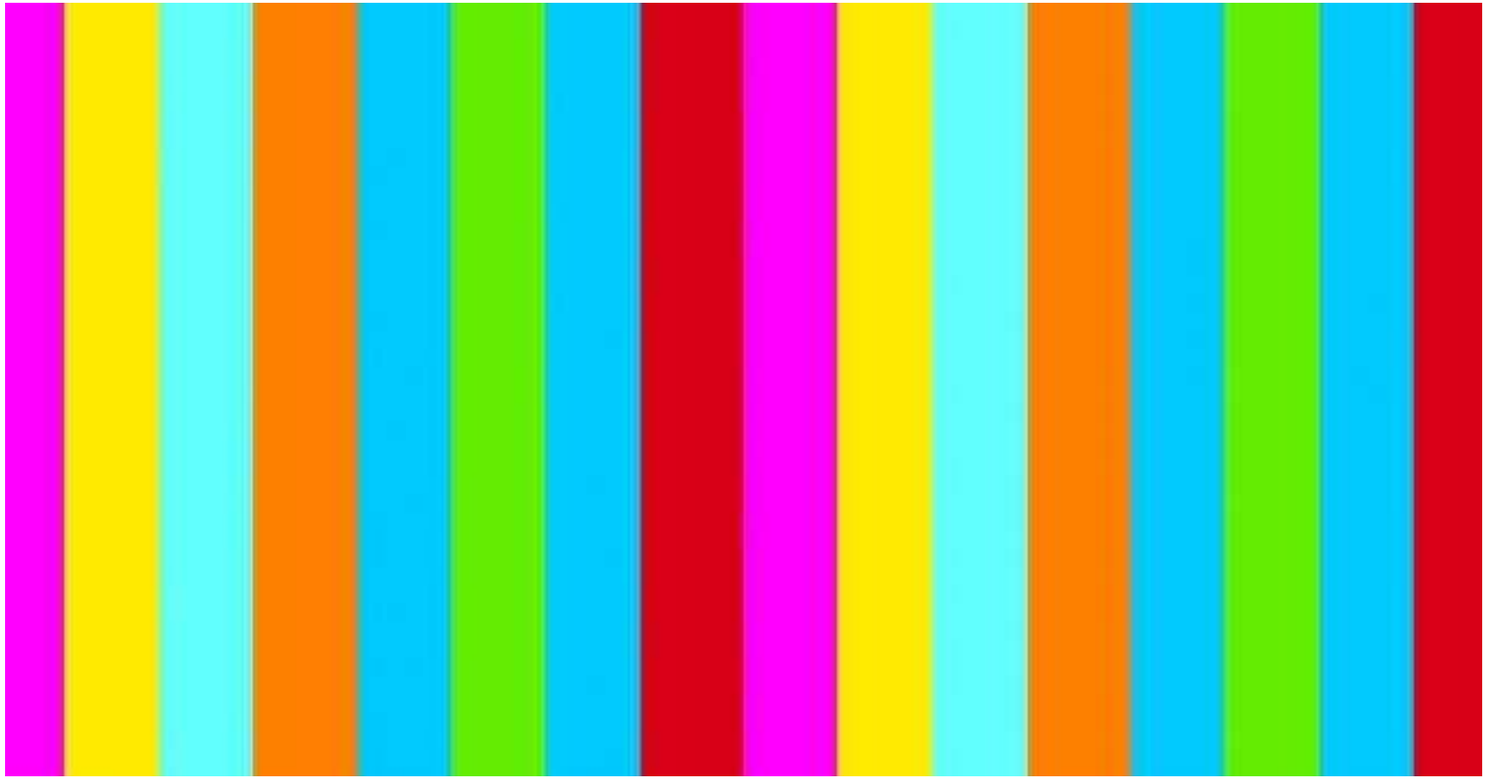
Part 1 数据冗余

对信息序列aaaabbbc使用如下两种编码方式：

①2bit定长编码：a(00)、b(01)、c(10)，则平均码长为2。

②可变长度编码：a(0)、b(1)、c(00)，则平均码长为(4×1+3×1+1×2)/8=1.125。

GLASSSIX



Part 2 常用压缩方法

GLASSSIX

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

对给定正整数m和n，表示为$G_m(n)$的n关于m的Golomb编码是商floor(n/m)的一元编码和n mod m的二进制表示的一个合并。

步骤 1 形成商$\lfloor n/m \rfloor$的一元编码(整数 q 的一元编码定义为 q 个 1 紧跟着一个 0)。

步骤 2 令 $k=\lceil \log_2 m \rceil, c=2^k-m, r=n \mod m$，并计算截短的余数 r'，例如，使其满足

$$r'=\begin{cases} r\text{截短至 } k\text{–1 比特}, & 0 \leqslant r < c \\ r+c \text{ 截短至 } k \text{ 比特}, & \text{其他} \end{cases}$$

步骤 3 连接步骤 1 和步骤 2 的结果。

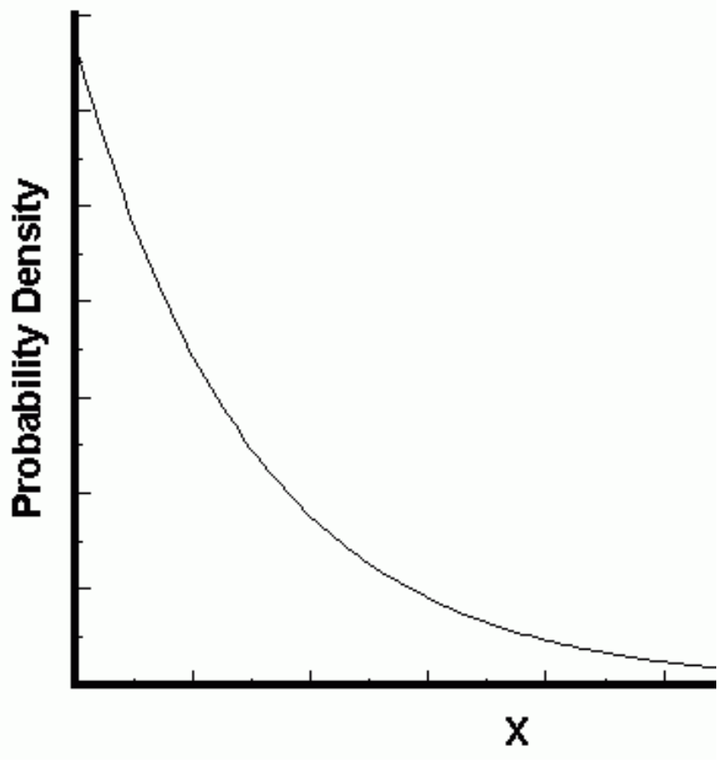
计算$G_4(9)$：step1　floor(9/4)=floor(2.25)=2，商的一元编码为110

Step2　k=ceil($\log_2 4$)=2，c=2^2-4=0，r=9 mod 4=1，截短后的r'为01

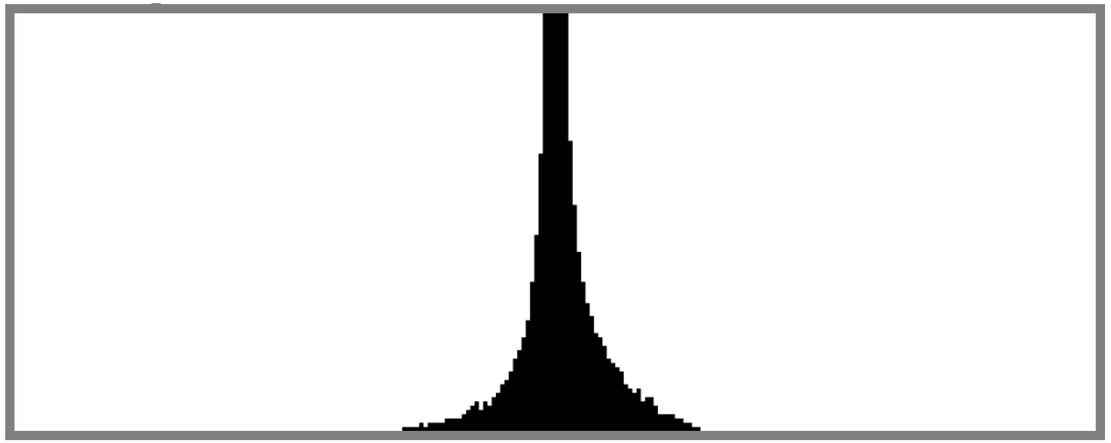
Step3　连接后的$G_4(9)$=11001

GLASSSIX

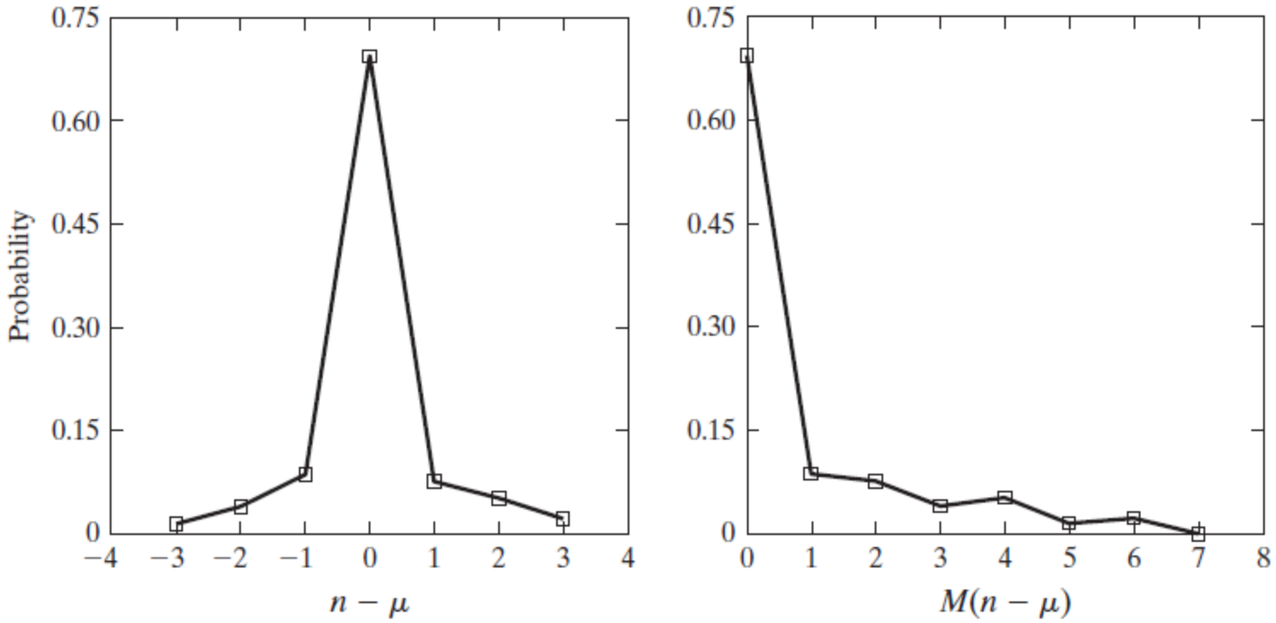
n	$G_1(n)$	$G_2(n)$	$G_4(n)$	$G^0_{\exp}(n)$
0	0	00	000	0
1	10	01	001	100
2	110	100	010	101
3	1110	101	011	11000
4	11110	1100	1000	11001
5	111110	1101	1001	11010
6	1111110	11100	1010	11011
7	11111110	11101	1011	1110000
8	111111110	111100	11000	1110001
9	1111111110	111101	11001	1110010



GLASSSIX



$$M(n) = \begin{cases} 2n & n \geq 0 \\ 2|n| - 1 & n < 0 \end{cases}$$



GLASSSIX

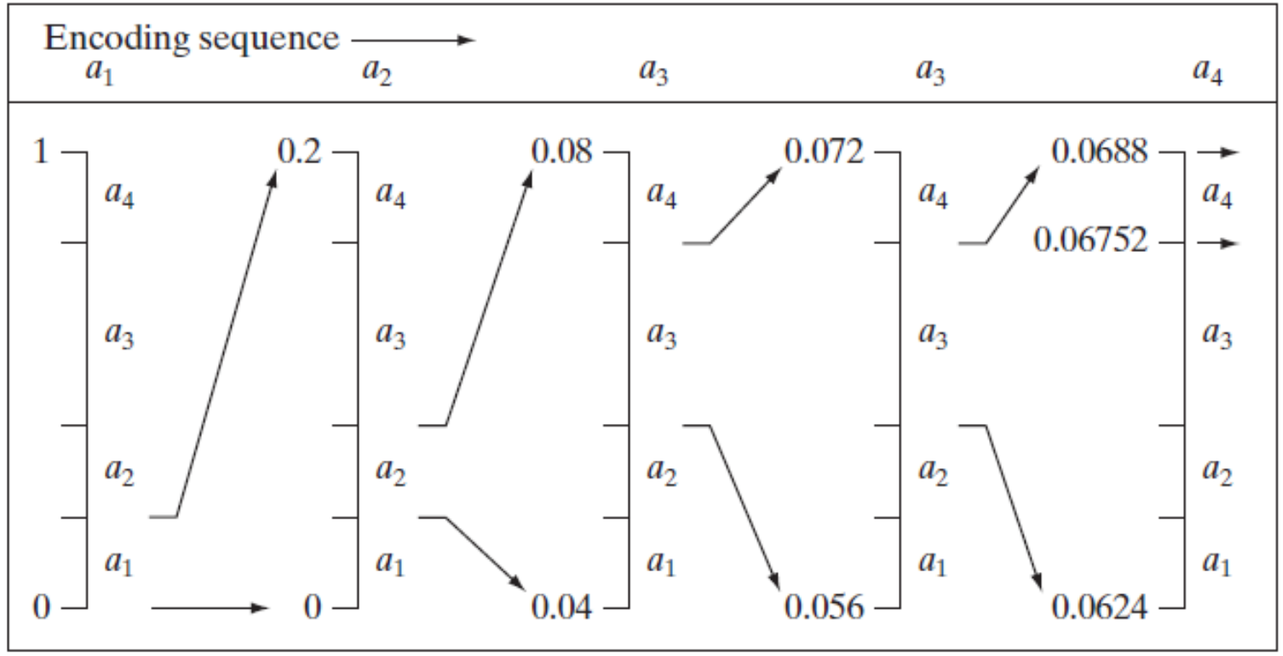
算术编码为整个信源符号序列分配一个单一的算术码字,采用子区间划分的方法，计算公式：

range = high – low; high = low + range * probHigh; low = low + range * probLow.

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

①a_1：range = 1.0 – 0.0 = 1.0;
high = 0.0 + 1.0 * 0.2 = 0.2;
low = 0.0 + 1.0 * 0.0 = 0.0.

②a_2：range = 0.2 – 0.0 = 0.2;
high = 0.0 + 0.2 * 0.4 = 0.08;
low = 0.0 + 0.2 * 0.2 = 0.04.



$a_1\ a_2\ a_3\ a_3\ a_4$ => [0.06752, 0.0688)

GLASSSIX

基于自适应二元算术编码：CABAC(Context-based Adaptive Binary Arithmetic Coding)

① Initial range：[0.0, 1.0) => Initial range：[0, 0XFFFFFFFF)

②

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

=>

Source Symbol	Probability	Initial Subinterval
0	P_0	[probLow * 0XFFFFFFFF，probHigh * 0XFFFFFFFF)
1	1 - P_0	[probLow * 0XFFFFFFFF，probHigh * 0XFFFFFFFF)

③ 自适应更新概率：

$$p^{(t+1)}{}_{LPS} = \begin{cases} \alpha \cdot p^{t}{}_{LPS} & if\ an\ MPS\ occurs \\ \alpha \cdot p^{t}{}_{LPS} + (1-\alpha) & if\ an\ LPS\ occurs \end{cases}$$

注：MPS(Most Probability Symbol)，LPS(Low Probality Symbol)，分别代表出现概率大小的符号；$\alpha = 0.95$。

行程编码

行程编码RLE（Run Length Encoding）算法把数据看成一个存在连续重复数据块的线性序列。采用的压缩策略是用一个字节表示数据块重复的次数，然后存储对应的数据字节本身。

AAAAAABCCCC => 6A1B4C

ABC => 1A1B1C

像素变化次数较多时可能导致数据扩展，因此行程编码较适合二值图像：

000111111000001100011 1 => 3，6，5，2，3，3

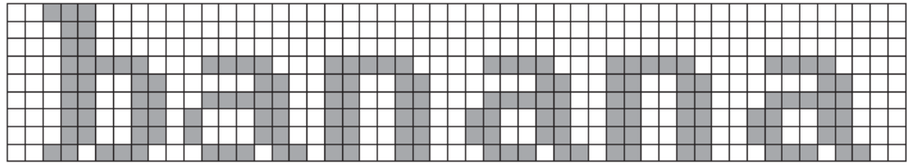
111110000011111100000 => 0，5，5，7，5

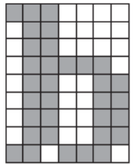
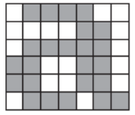
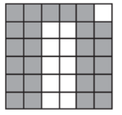
000110001111100000011 => 3，2，3，5，7，2

默认每行起始字符为“0”

GLASSSIX

一幅图像表示为多幅频繁发生的子图像（***符号***）的一个集合。每一个***符号***都存储在符号字典中，且该图像以一个三元组$\{(x_1, y_1, t_1), (x_2, y_2, t_2), \cdots\}$的集合来编码，其中$(x_i, y_i)$表示图像中符号的位置，$t_i$是该符号在字典中的地址。

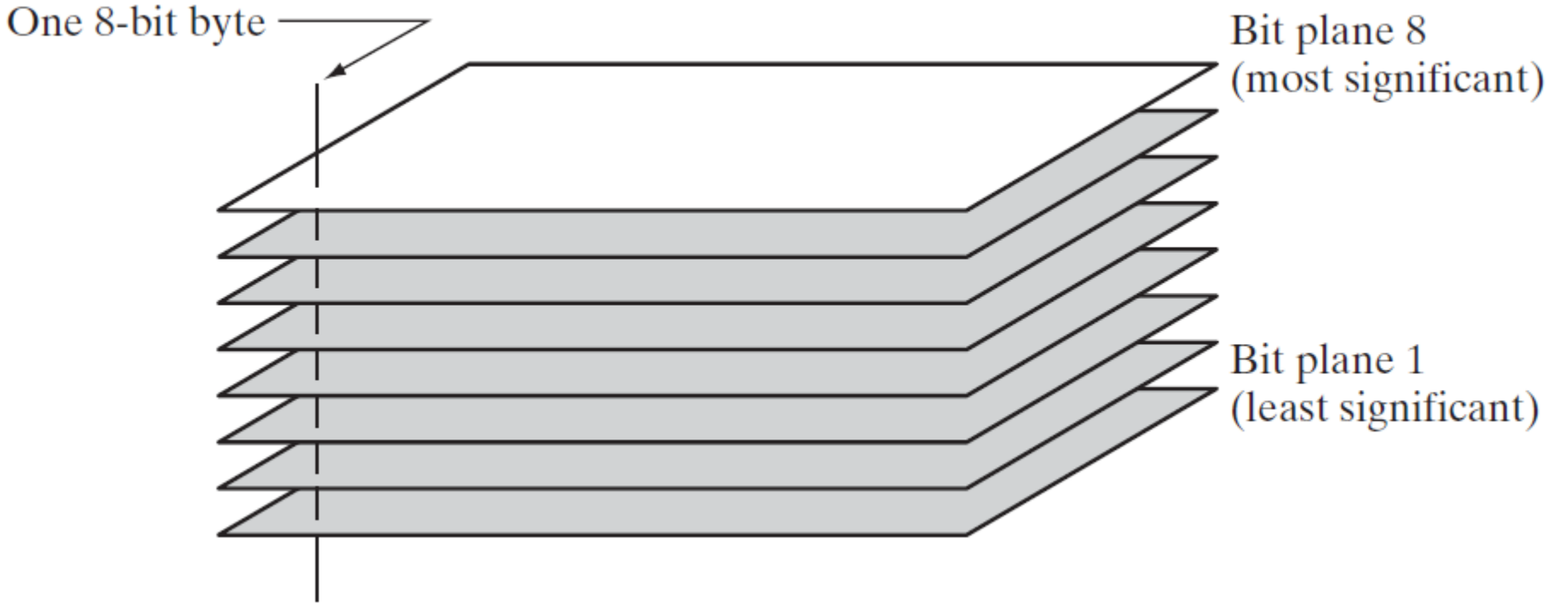


Token	Symbol
0	
1	
2	

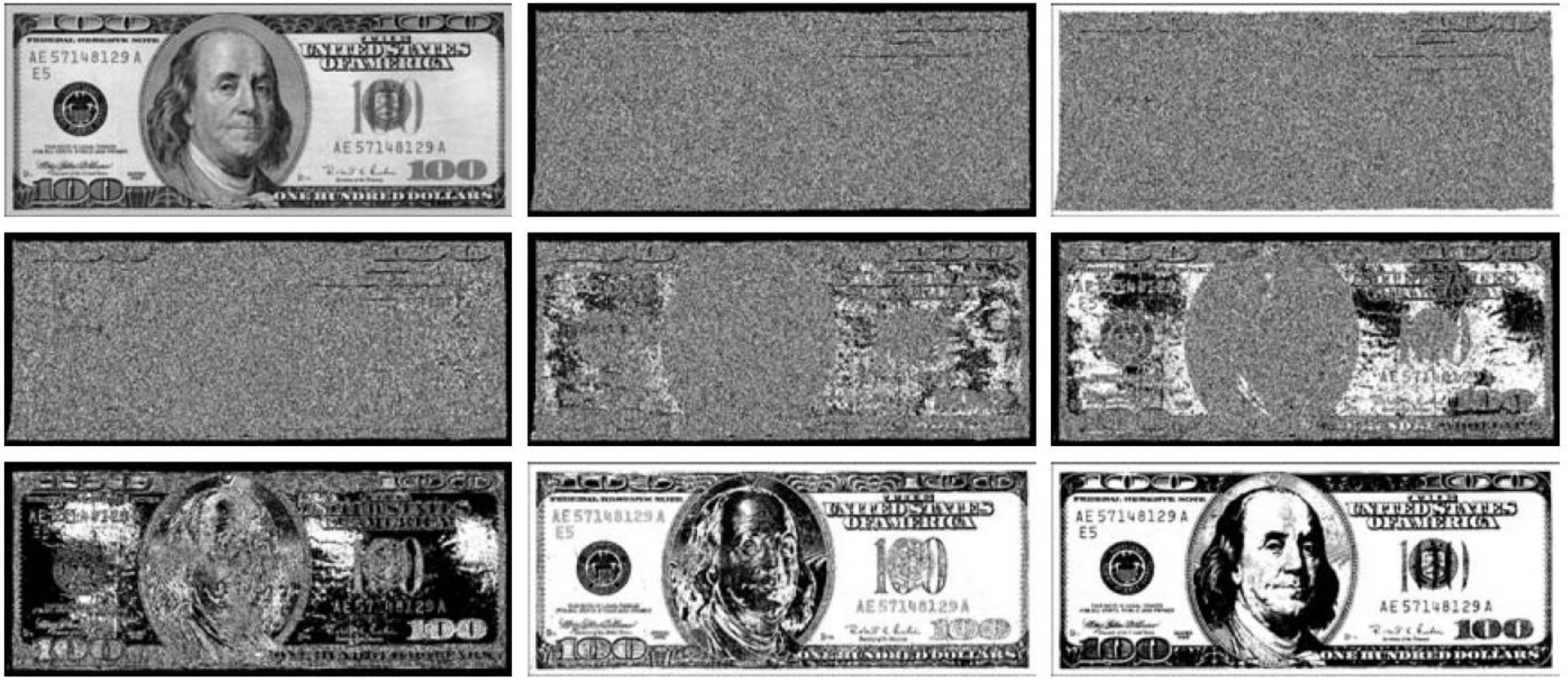
Triplet
(0, 2, 0) (3, 10, 1) (3, 18, 2) (3, 26, 1) (3, 34, 2) (3, 42, 1)

将一幅多级（彩色）图像分解为一系列二值图像，然后使用某种二值压缩方法（行程编码）来压缩每幅二值图像。

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0$$



0阶平面对灰度值贡献最大为2^0=1, 1阶平面对灰度值贡献最大为2^1= 2, 2阶平面对灰度值贡献最大为2^2= 4, 直接舍弃这3个平面可减少3/8=37.5%的数据量。



块变换编码将图像分成大小相等（如8×8）且不重叠的小块，然后用可逆线性变换把每个小块映射为变换系数的集合。变换后的大多数系数都有较小的值，因此可以参照降维原理（主成分分析 PCA、奇异值分解SVD ），取一定比例的最大幅值，在保证图像质量的同时减少数据量。

离散变换：$$T(u,v)=\sum_{x=0}^{n-1}\sum_{y=0}^{n-1}g(x,y)r(x,y,u,v)$$

离散逆变换：$$g(x,y)=\sum_{u=0}^{n-1}\sum_{v=0}^{n-1}T(u,v)s(x,y,u,v)$$

r(x,y,u,v)和s(x,y,u,v)分别称为正变换核和逆变换核

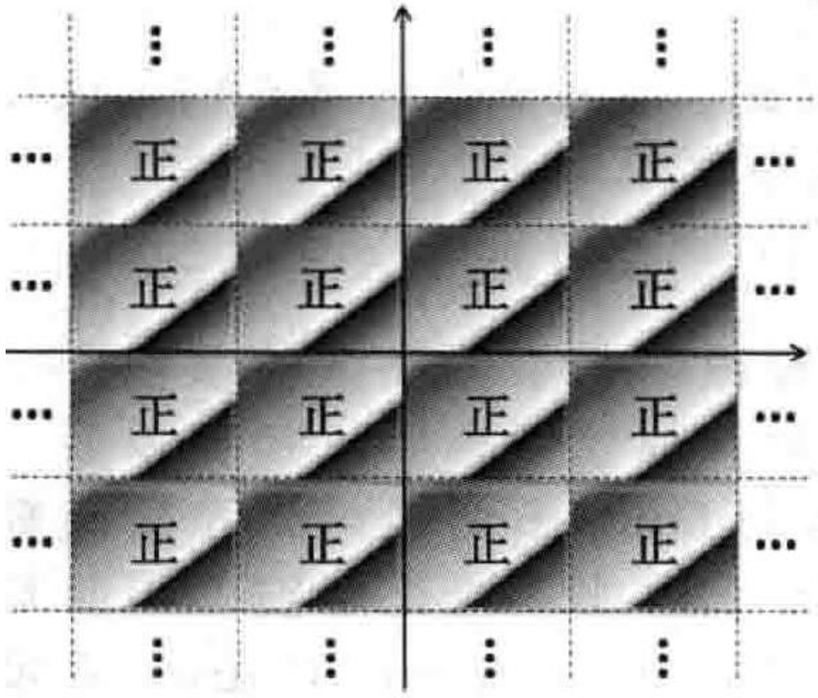
根据所使用变换核的不同，主要有3种变换形式：

①Walsh-Hadamard变换(WHT)：$$r(x, y, u, v) = s(x, y, u, v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1} \lfloor b_i(x)p_i(u)+b_i(y)p_i(v) \rfloor}$$

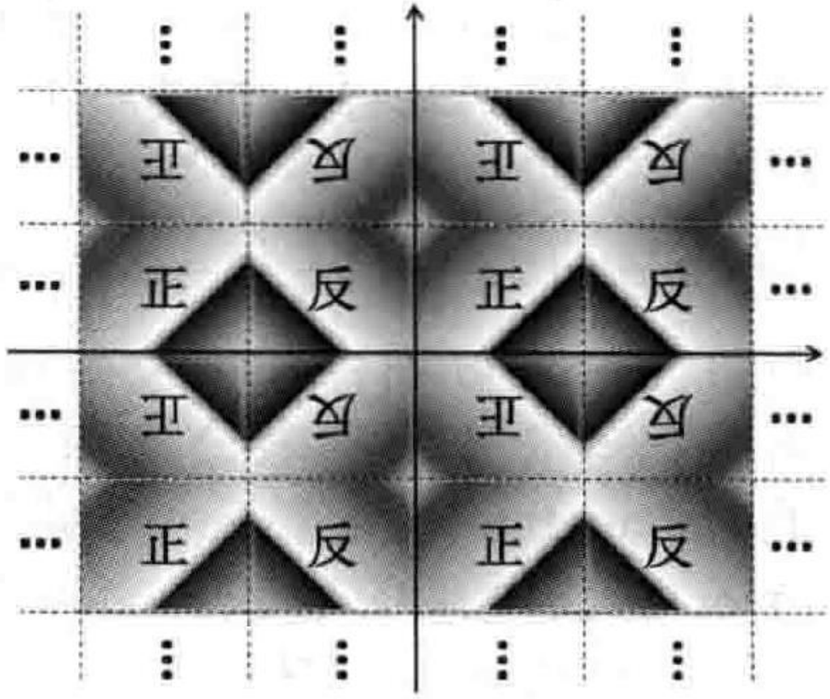
②离散傅里叶变换(DFT)：$$r(x, y, u, v) = e^{-j2\pi(ux+vy)/n}$$

$$s(x, y, u, v) = \frac{1}{n^2}e^{j2\pi(ux+vy)/n}$$

③离散余弦变换(DCT)：$$r(x, y, u, v) = s(x, y, u, v) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$



(a) DFT所对应的周期延拓



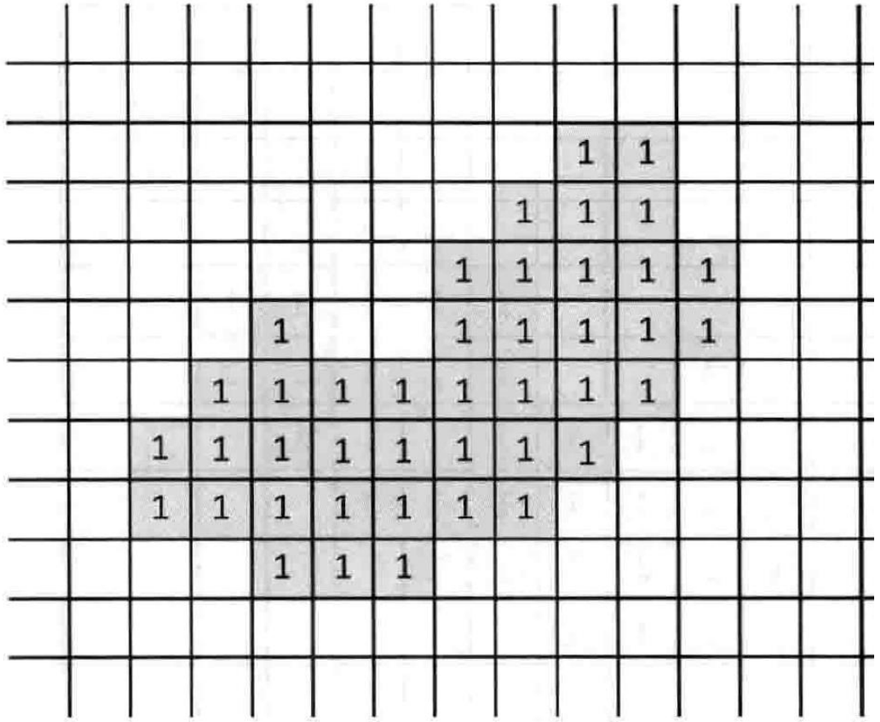
(b) DCT所对应的周期延拓

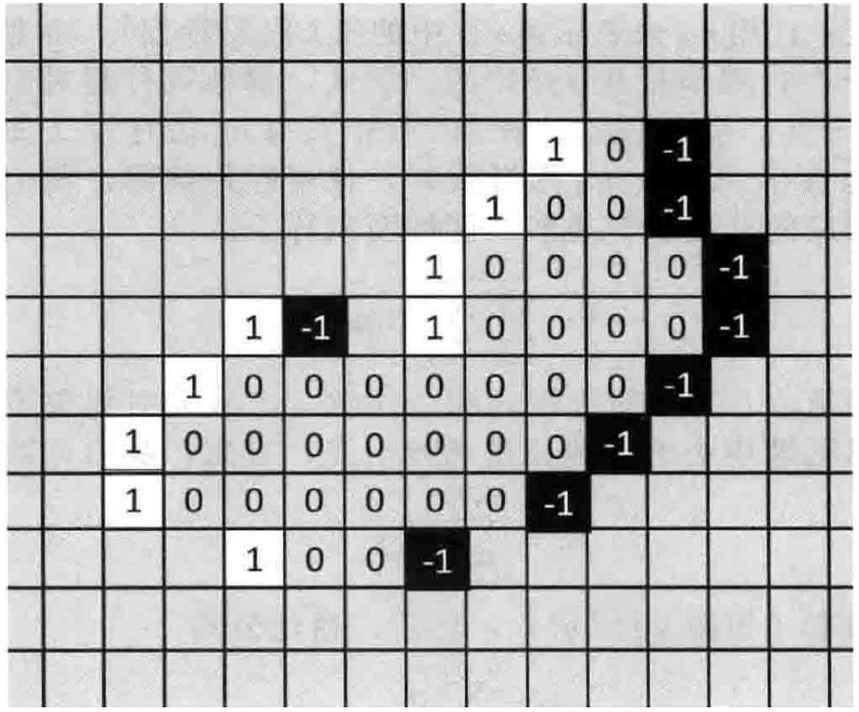
$$e^{j\theta} = \cos\theta + j\sin\theta$$

$$\int_{-\infty}^{+\infty}(acosx + bsinx)d_x = \int_{-\infty}^{+\infty}(acosx)d_x$$

GLASSSIX

通过消除紧邻像素在空间和时间上的冗余来实现，仅对每个像素中的新信息进行提取和编码。

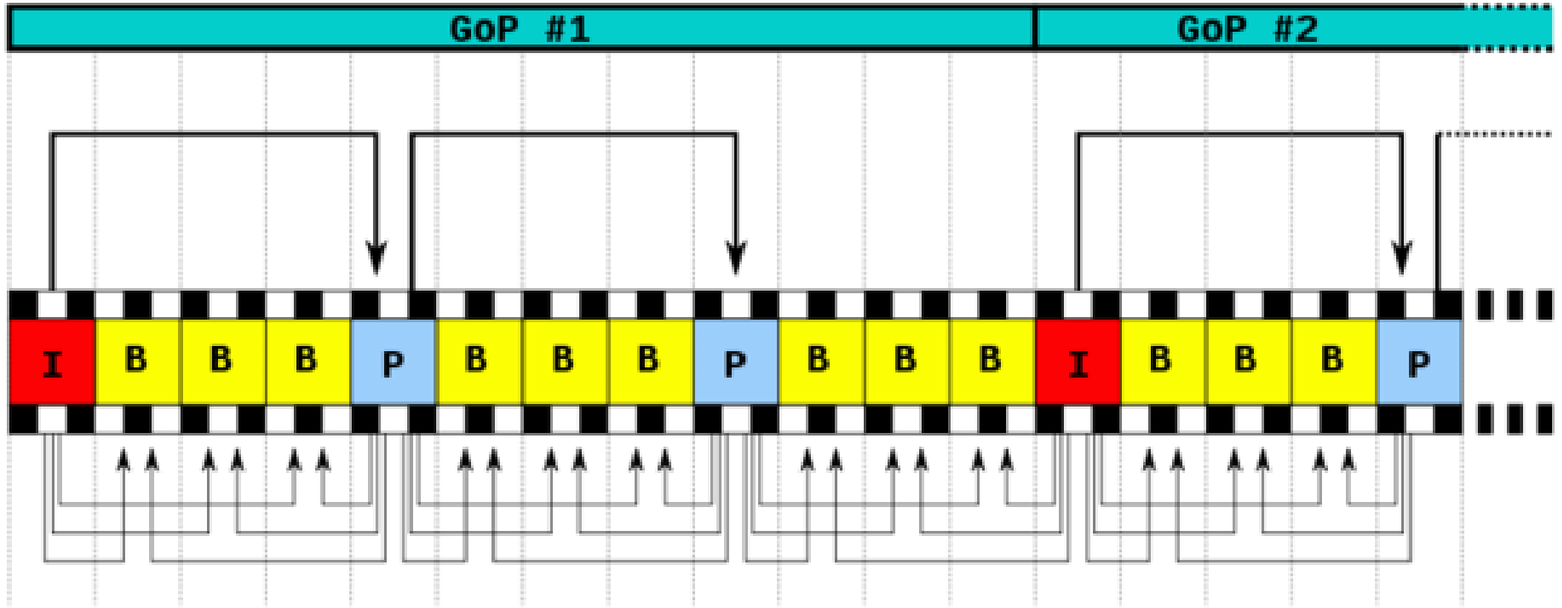


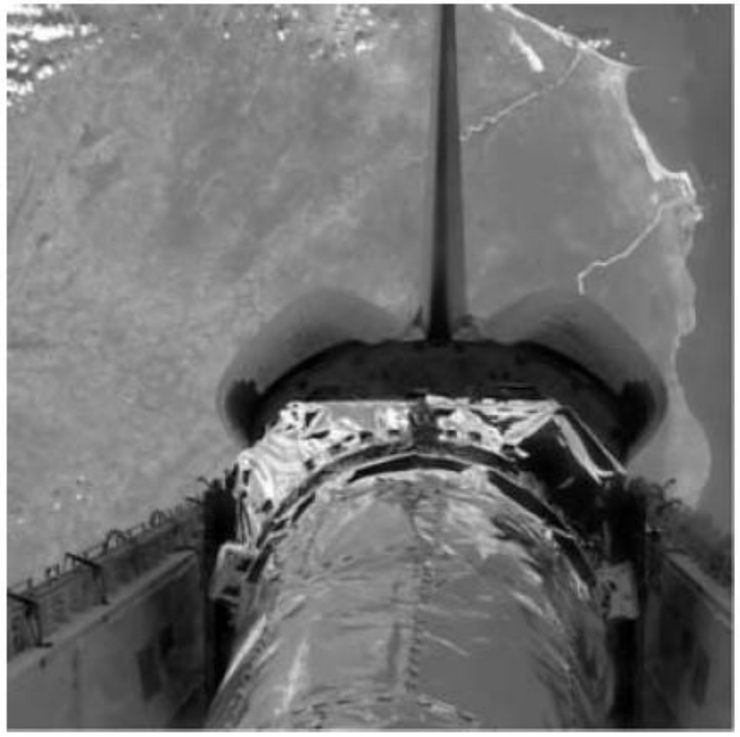


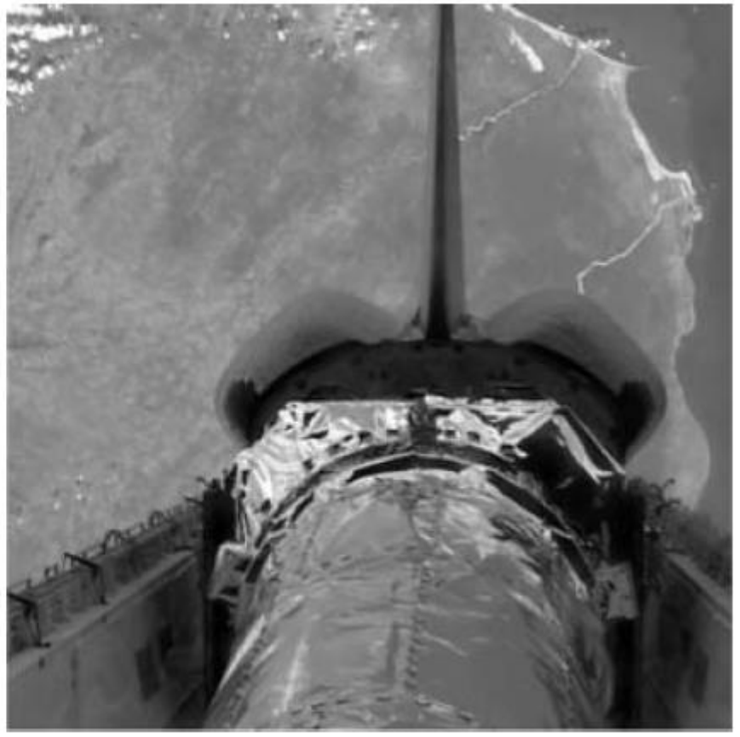
差分编码：$e(x,y) = f(x,y) - f(x,y-1)$

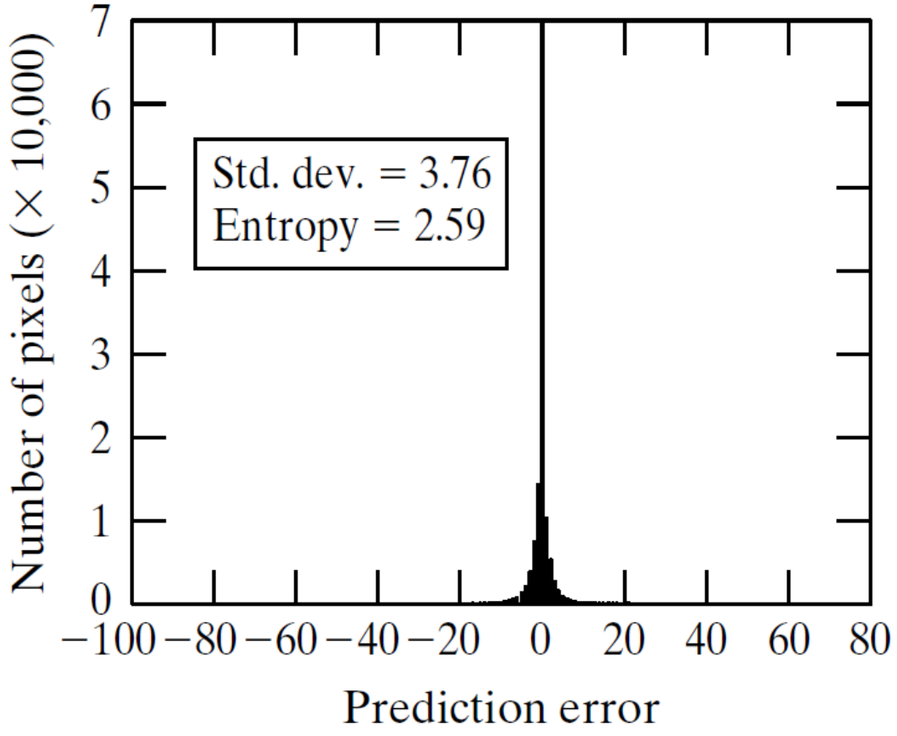
•I帧：关键帧，完整编码的帧，可独立重建图像。
•P帧：前向参考帧，只包含与前一帧差异部分编码的帧，需结合前一帧才能重建图像。
•B帧：双向参考帧，既参考前面的帧，又参考后面的帧，需结合前后帧才能重建图像。

图像序列GOP(Group Of Pictures)：一个序列就是一段内容差异不太大的图像编码后生成的一串数据流，两个I帧之间是一个图像序列。



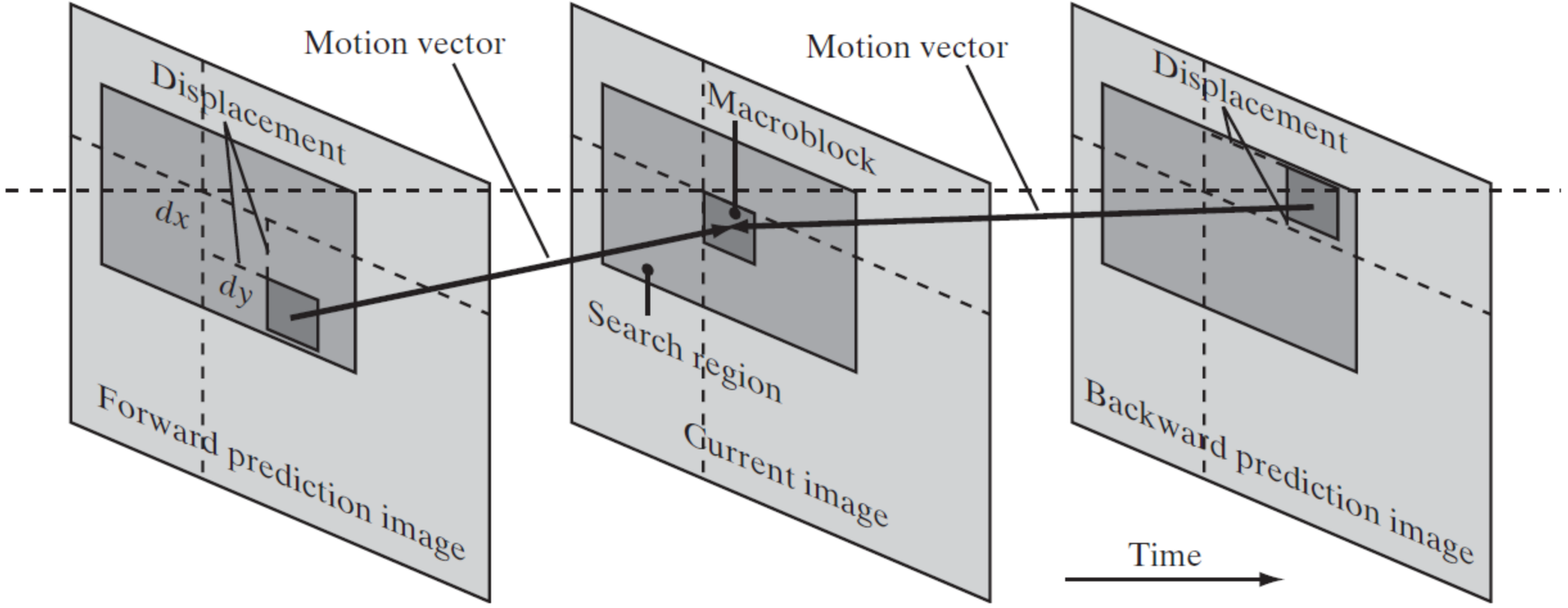




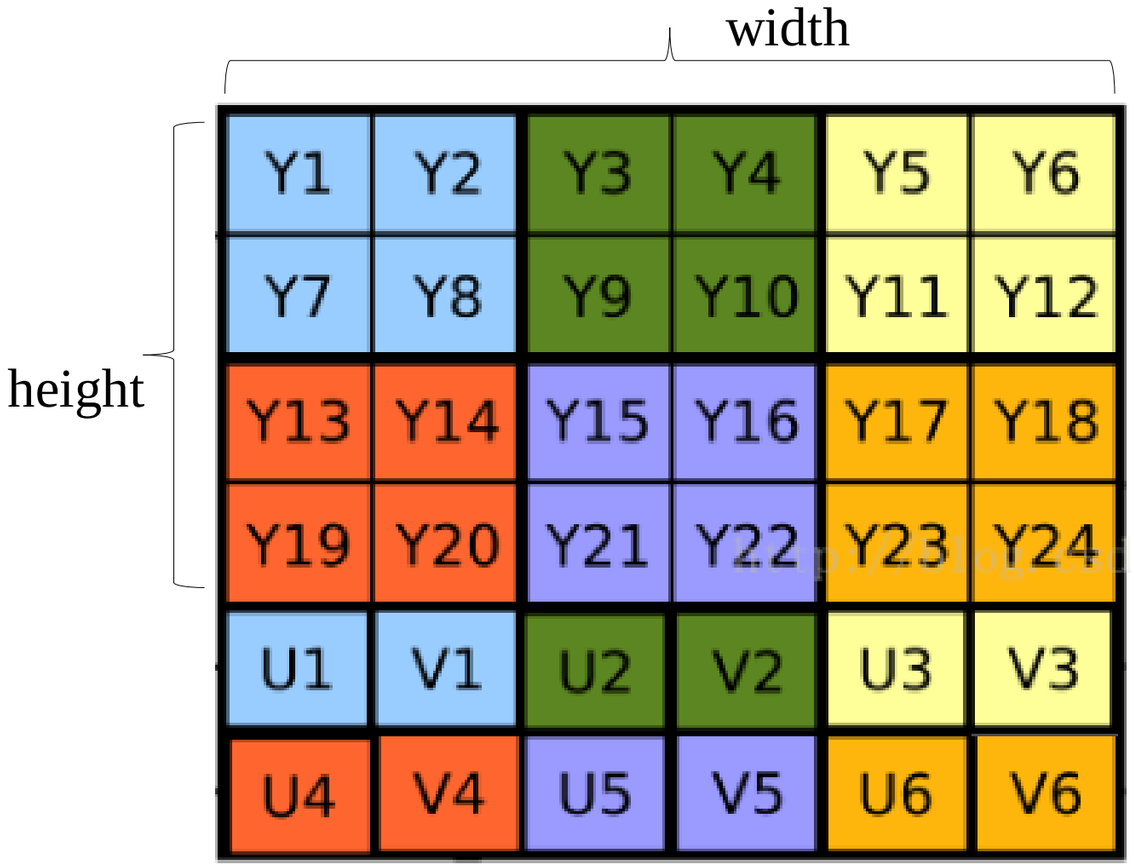


预测残差：$e(x,y,t) = f(x,y,t) - f(x,y,t-1)$

运动补偿：在预测和差分处理期间，跟踪目标运动并对其进行补偿。



GLASSSIX



$Pixel_1$

$R_1 = Y_1 * 1.1644 + V_1 * 1.596;$

$G_1 = Y_1 * 1.1644 - U_1 * 0.3918 - V_1 * 0.813;$

$B_1 = Y_1 * 1.1644 + U_1 * 2.0172;$

$Pixel_2$

$R_2 = Y_2 * 1.1644 + V_1 * 1.596;$

$G_2 = Y_2 * 1.1644 - U_1 * 0.3918 - V_1 * 0.813;$

$B_2 = Y_2 * 1.1644 + U_1 * 2.0172;$

$Pixel_7$

$R_7 = Y_7 * 1.1644 + V_1 * 1.596;$

$G_7 = Y_7 * 1.1644 - U_1 * 0.3918 - V_1 * 0.813;$

$B_7 = Y_7 * 1.1644 + U_1 * 2.0172;$

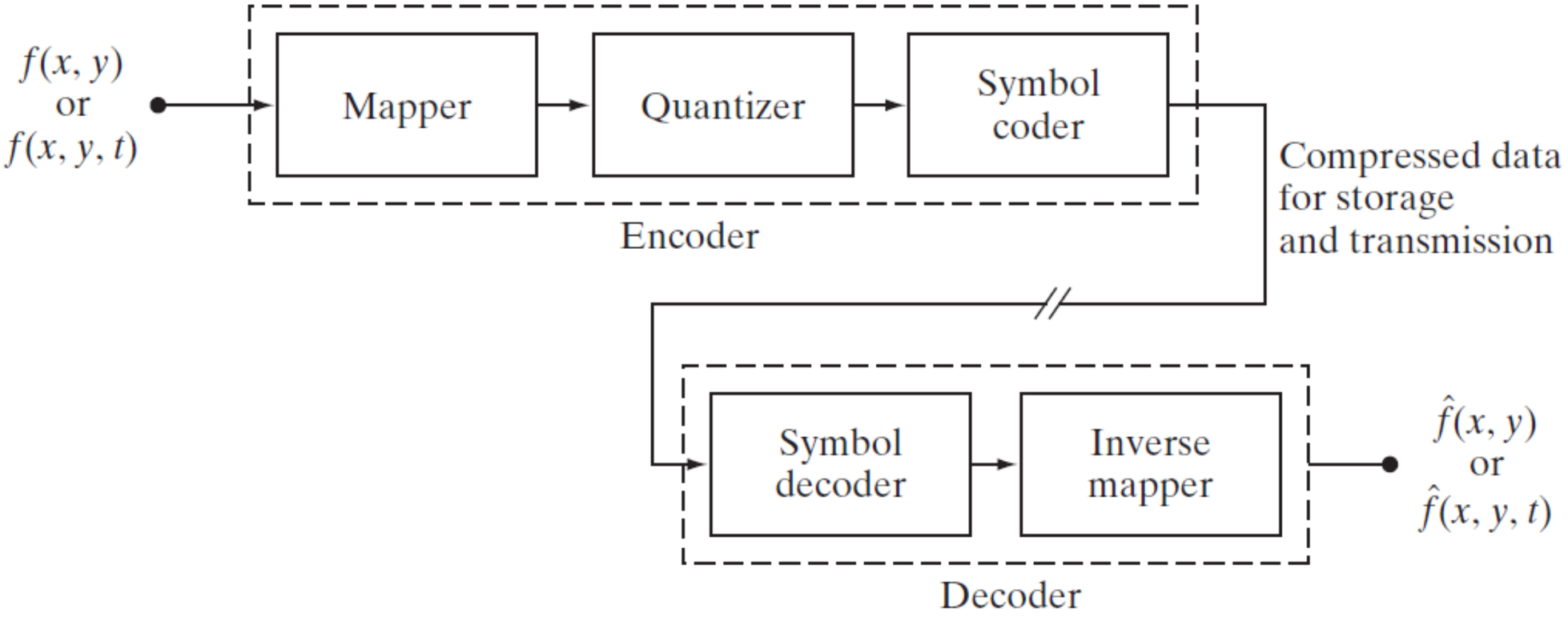
$Pixel_8$

$R_8 = Y_8 * 1.1644 + V_1 * 1.596;$

$G_8 = Y_8 * 1.1644 - U_1 * 0.3918 - V_1 * 0.813;$

$B_8 = Y_8 * 1.1644 + U_1 * 2.0172;$

数据量：height*width+1/4*height*width +1/4*height*width=3/2*height*width，减少50%


$f(x, y)$
or
$f(x, y, t)$
Mapper
Quantizer
Symbol coder
Encoder
Compressed data for storage and transmission
Symbol decoder
Inverse mapper
Decoder
$\hat{f}(x, y)$
or
$\hat{f}(x, y, t)$