



# 厦大数字图像处理期末复习下

原创

Karon\_NeverAlone 于 2022-01-08 21:42:31 发布

版权

阅读量3k 收藏 18 点赞数 4

分类专栏: 数字图像处理 文章标签: 算法 计算机视觉 图像处理



数字图像处理 专栏收录该内容

6 订阅 10 篇文章

订阅专栏

## 内容概要

### 10.彩色图像

常见的颜色空间和和应用场合

伪彩色处理有什么用?

灰度 \彩色图像不同模型之间的转换

伪彩色处理的方法 (从灰色图像到彩色图像变换的方法) ——该处有实验

### 11.图像压缩

变换编码、预测编码、统计编码、哈夫曼编码以及其他的基本编码方式

冗余都有哪些? 怎么去除冗余? DCT变换本身并不会去除冗余, 量化才是

### 12.形态学

形态学的基本操作要知道是怎么做的: 腐蚀、膨胀、开、闭

### 13.图像分割

基于颜色的图像分割

边缘检测 算法 (canny)

阈值化方法: 最基本的OTSU、全局阈值、自适应的阈值——该处有实验

阈值化方法的影响因素和改善

时间紧任务重, 加油加油!

---

## 10.彩色图像

> --首先是一些杂七杂八的概念，**可以跳过**

-彩色图像处理 分为：全彩色处理和伪彩色处理

-描述颜色三个基本属性：亮度、色调和饱和度（色调和饱和度合在一起称为色度）

-全彩色处理：三个通道分别处理或者是直接基于一个颜色向量

**色彩切片**（强调一些感兴趣的色彩部分）

**色调和色彩校正**：相片增强和色彩复制

色调矫正：没有修改颜色，只是改了对比度和亮度，用相同的**变换函数**映射所有三种(或四种)颜色分量如果是HSI就直接调整亮度

颜色修正：任何颜色的比例都可以通过**减少相反颜色**的数量来增加

直方图处理：HSI很适合，均匀的拓展颜色强度，颜色本身不变

**彩色图像的平滑和锐化**：和灰色图像的一样，但是要分三个维度

-常见的颜色模型和应用场合

**颜色模型**

**硬件** 导向的：RGB / CMY and CMYK

面向应用程序的：HSI(根人认识颜色的方式最相同)

**RGB**

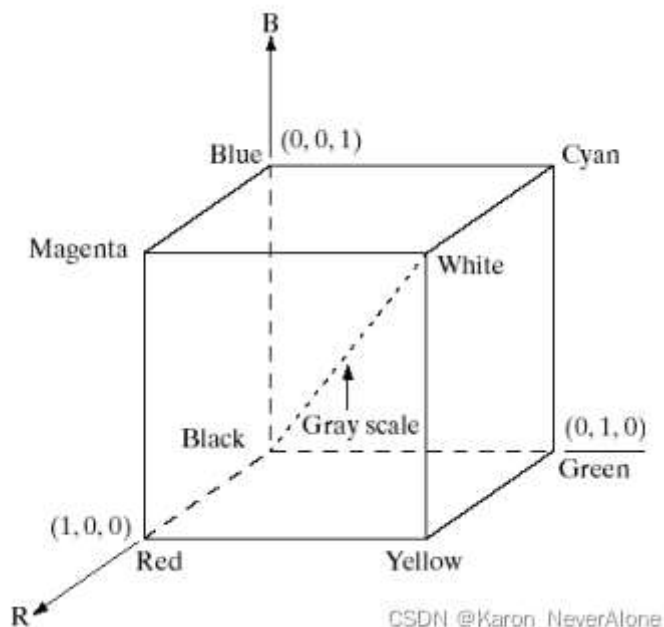
适合显示的时候用，但是中间的颜色处理过程却不一定是RGB

所有的颜色都归一化到1, 1, 1的小立方体里面，**对角线是各种灰色**

像素深度:RGB空间中表示每个像素的位数 现在是8位 (0~255)

•24位RGB颜色立方体（即红绿蓝各8位）

**FIGURE 6.7**  
Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1, 1, 1).



## CMY和CMYK

主要应用在打印机这一类需要颜色堆叠的机器上

K是指在CMY(青色、洋红、黄色)的基础上加上黑色

RGB和CMYK 的区别和联系

RGB是发光色彩模式，就是用在屏幕上的，CMY是依靠反光的色彩模式，用在印刷品

CMY是RGB三原色的三个混色

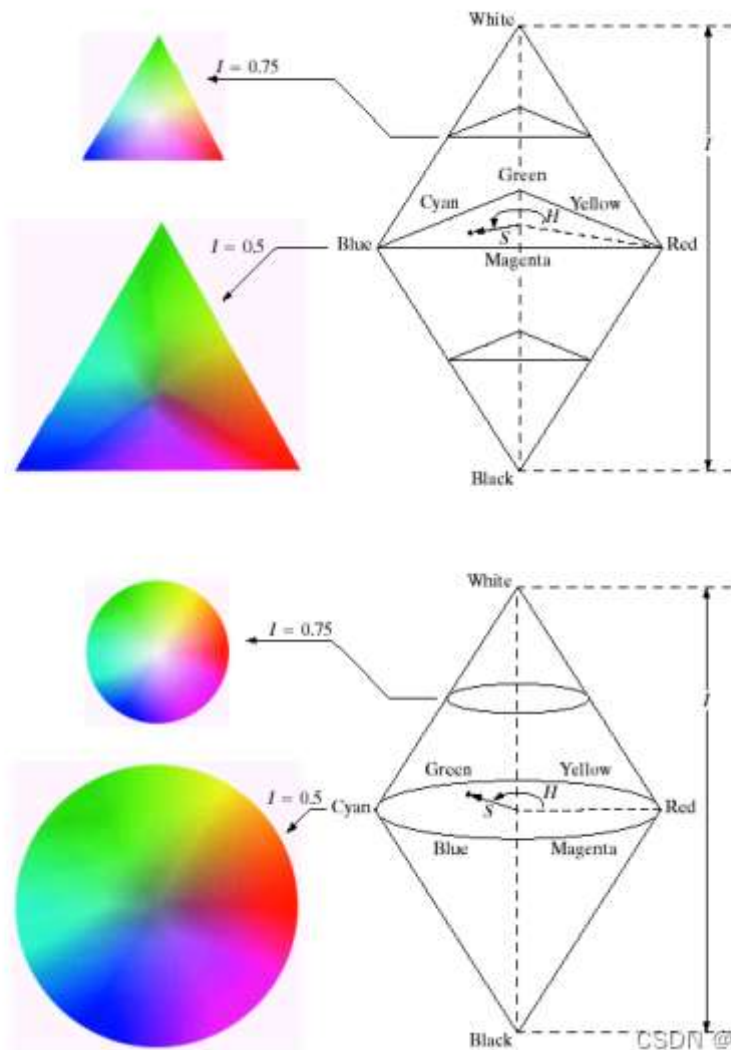
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

CSDN @Karon\_NeverAlone

## HSI(色相，饱和度，强度)

和人对颜色的描述很像，是开发基于颜色描述的图像处理算法的理想工具

三维的空间，不同的角度代表不同的色相，y轴是明度，从内到外是饱和度



CSDN @Karon\_NeverAlone

## -灰度\彩色图像不同模型之间的转换

### -RGB到HSI——该处有实验

表 1 几种算法的转换公式

算法	色调	饱和度	亮度
算法 1 几何推导法	$H = \begin{cases} \theta, & G \geq B \\ 2\pi - \theta, & G < B \end{cases}$ $\text{where } \theta = \cos^{-1} \left( \frac{(R-G) + (R-B)}{2\sqrt{(R-G)^2 + (R-B)(G-B)}} \right)$	$S = 1 - \frac{3 \min(R, G, B)}{R + G + B}$	$I = \frac{R + G + B}{3}$

CSDN @Karon\_NeverAlone

完成上述公式大致有两种思路：

- 一个像素一个像素的处理，对每一个像素的R\G\B值套用上述公式，利用到for循环
- 矩阵处理，直接将R\G\B分为三个矩阵，套用一次上述公式，做矩阵的运算
- 具体看当时的matlab代码

## -HSI到RGB

没讲

## -伪彩色处理有什么用？

定义：根据指定的标准为灰色值分配颜色

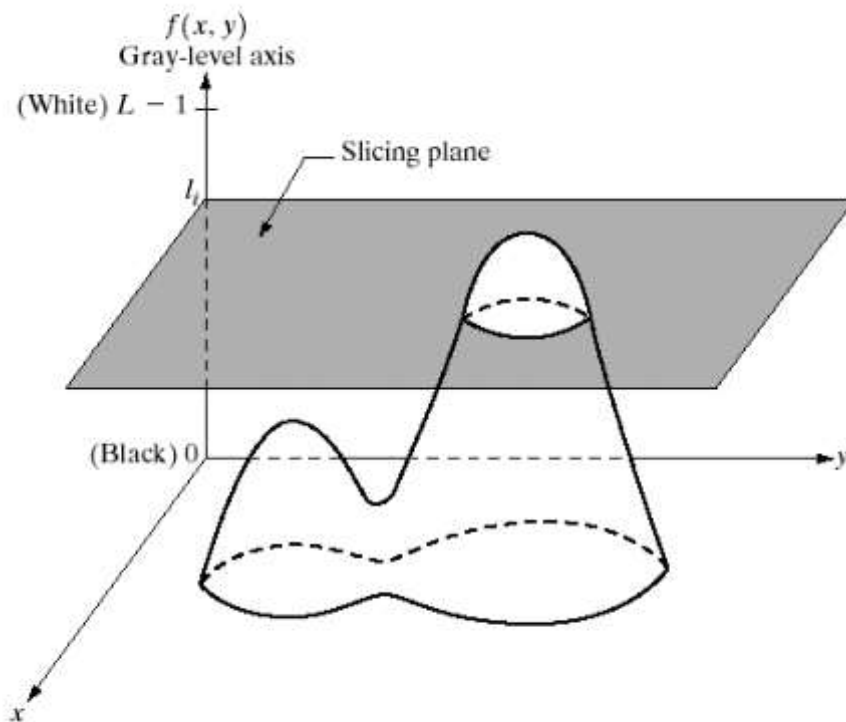
用处：人类可视化;对图像或图像序列中灰度事件的解释，因为人可以识别好多颜色，但只能识别一些些的灰度

用途：多光谱图像处理

## -伪彩色处理的方法（从灰色图像到彩色图像变换的方法）——该处有实验

### 强度切片（灰度切片）

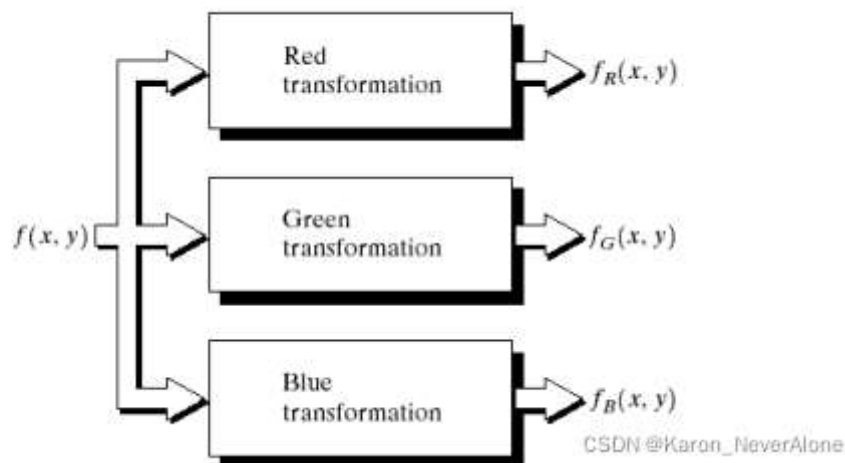
这个应该很好理解，就是①先把灰度分级②给每一级分配一个颜色



6.18 Geometric interpretation of the intensity-slicing technique.

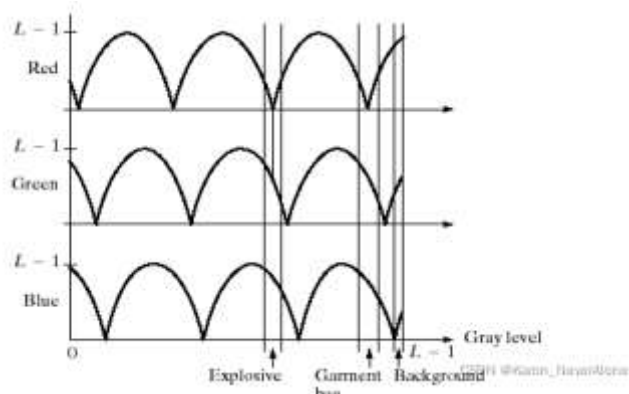
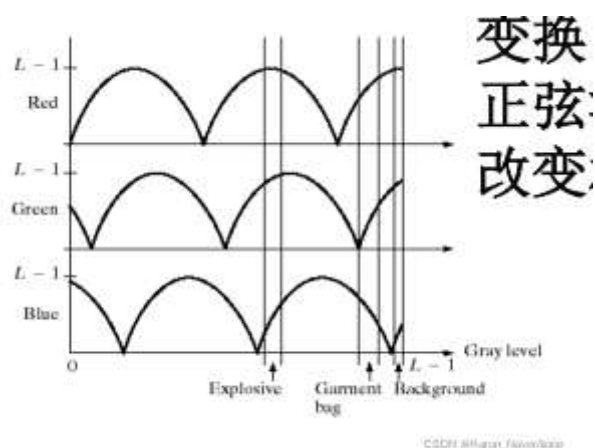
### 灰度到颜色转换(通过一定规则的变换)

对任何输入像素的灰度级进行**三个独立的变换**。然后将这三个结果分别送入彩色电视显示器的红色、绿色和蓝色通道



这个方法主要的问题就是确定“transformation”

正弦波，改变相位和频率（一毛一样的话就还是灰色的）



不过还有比较简单的方法，就是直接自己定一个规则，比如R分量在128-240之间的赋值为128否则为0之类的

## 11.图像压缩

-压缩比&相对数据冗余（ $n_1$ 是压缩前占用的存储， $n_2$ 是压缩后）

$$C_R = \frac{n_1}{n_2}$$

$$R_D = 1 - \frac{1}{C_R}$$

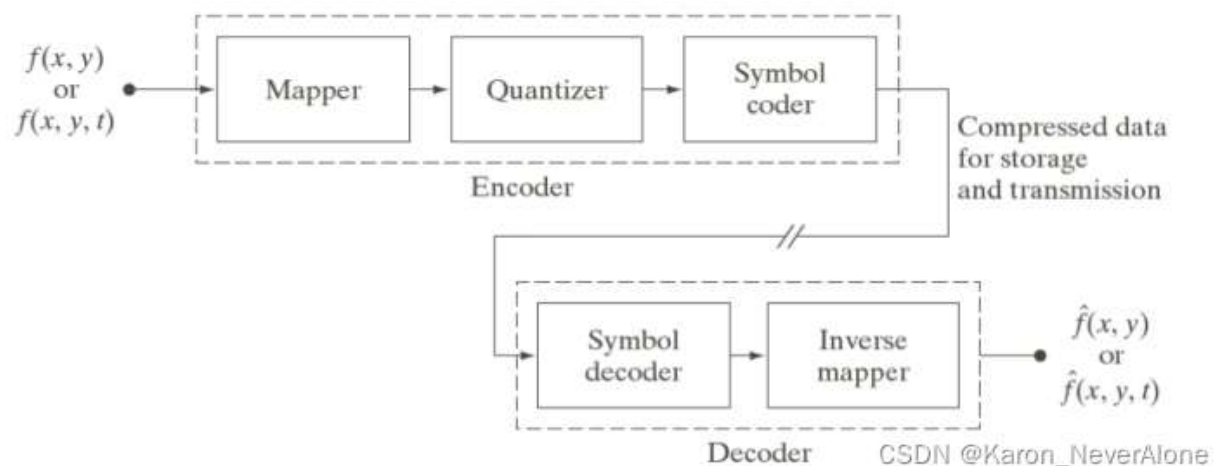
## -冗余都有哪些？

- 空间和时间冗余：相邻位置和相邻帧的相同位置灰度都差不多
- 编码冗余：表示某个量的信息不需要那么多编码空间
- 感知冗余：人们对色度不敏感但是对亮度敏感

## -图像压缩模型：怎么去除冗余？编码不会去除冗余，量化才会

①**Mapper转换：去除时空冗余**为了减少图像中像素间的冗余，通常用于人类观看和解释的二维像素阵列必须**转换**成一种更有效(但通常是“非视觉”)的格式

②**Quantizer量化：减少心理视觉上的冗余**（无损压缩不做量化哦）



③**coder编码：解决了编码冗余**

## -平均编码长度（变长、词典、位平面）

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

某个灰度的占比 (pointing to  $p_r(r_k)$ )  
某个灰度编码长度 (pointing to  $l(r_k)$ )

$$p_r(r_k) = \frac{n_k}{n} \quad k = 0, 1, 2, \dots, L-1$$

CSDN @Karon\_NeverAlone

## -常见压缩编码方式



----无损压缩编码

**变长编码：**不用固定的长度表示像素，尽可能短的码字分配给最可能的灰度级，当原图像灰度特别平均的 时候，这个编码的区别就不是很大了

①**哈夫曼编码：**其实就是构建哈夫曼树呀，需要会计算平均编码长度！！

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				

CSDN @Karon\_NeverAlone

Original source			Source reduction			
Sym.	Prob.	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				

CSDN @Karon\_NeverAlone

②**其他改进版霍夫曼：**截断 霍夫曼编码

截断哈夫曼

Source symbol	Probability	Binary Code	Huffman	Truncated Huffman	B <sub>2</sub> -Code	Binary Shift	Huffman Shift
<b>Block 1</b>							
$a_1$	0.2	00000	10	11	C00	000	10
$a_2$	0.1	00001	110	011	C01	001	11
$a_3$	0.1	00010	111	0000	C10	010	110
$a_4$	0.06	00011	0101	0101	C11	011	100
$a_5$	0.05	00100	00000	00010	C00C00	100	101
$a_6$	0.05	00101	00001	00011	C00C01	101	1110
$a_7$	0.05	00110	00010	00100	C00C10	110	1111
<b>Block 2</b>							
$a_8$	0.04	00111	00011	00101	C00C11	111 000	00 10
$a_9$	0.04	01000	00110	00110	C01C00	111 001	00 11
$a_{10}$	0.04	01001	00111	00111	C01C01	111 010	00 110
$a_{11}$	0.04	01010	00100	01000	C01C10	111 011	00 100
$a_{12}$	0.03	01011	01001	01001	C01C11	111 100	00 101
$a_{13}$	0.03	01100	01110	100000	C10C00	111 101	00 1110
$a_{14}$	0.03	01101	01111	100001	C10C01	111 110	00 1111
<b>Block 3</b>							
$a_{15}$	0.03	01110	01100	100010	C10C10	111 111 000	00 00 10
$a_{16}$	0.02	01111	010000	100011	C10C11	111 111 001	00 00 11
$a_{17}$	0.02	10000	010001	100100	C11C00	111 111 010	00 00 110
$a_{18}$	0.02	10001	001010	100101	C11C01	111 111 011	00 00 100
$a_{19}$	0.02	10010	001011	100110	C11C10	111 111 100	00 00 101
$a_{20}$	0.02	10011	011010	100111	C11C11	111 111 101	00 00 1110
$a_{21}$	0.01	10100	011011	101000	C00C00C00	111 111 110	00 00 1111
<b>Entropy 4.0</b>							
<b>Average length</b>		5.0	4.05	4.24	4.65	4.59	4.13

**TABLE 8.5**  
Variable-length codes.

这两仅作了解

$\Psi = 12$   
for truncated  
Huffman

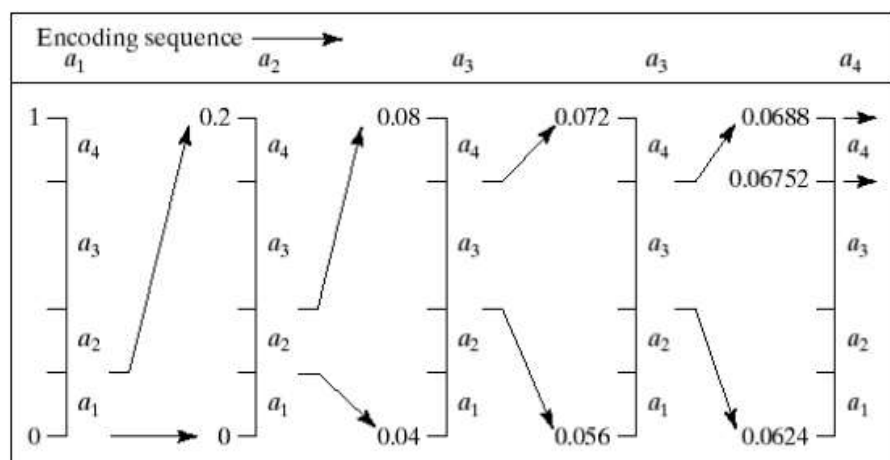
截断系数是12  
出现最频繁的  
前12个霍夫曼  
后面的正常编码

CSDN @Karon\_NeverAlone

③ 算数编码：掌握如何编码，如何解码

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)

**TABLE 8.6**  
Arithmetic coding  
example.



**FIGURE 8.13**  
Arithmetic coding  
procedure.

How to encode  
 $a_3 a_3 a_1 a_2 a_4$

CSDN @Karon\_NeverAlone

词典编码 LZW

思想就是用较短的代码来表示较长的字符串来实现压缩，具体怎么做的可以看看例子

39 39 126 126  
 39 39 126 126  
 39 39 126 126  
 39 39 126 126

Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

CSDN @Karon\_NeverAlone

39 39 126 126  
 39 39 126 126  
 39 39 126 126  
 39 39 126 126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

CSDN @Karon\_NeverAlone

## 位平面编码

思想将多层(单色或彩色)图像分解为一系列**二值图像**，然后用二值图的压缩编码处理(块编码和游程编码和预测编码)，比如一个0-255的图像可以表示为，这样每一位上分开，比如127表示为01111111

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \dots + a_12^1 + a_02^0$$

CSDN @Karon\_NeverAlone

但是这样还不能直接用，要变成**灰码**才可以，127变成10000000，高价位平面远比低价位平面复杂

$$g_{m-1} = a_{m-1}$$

$$g_i = a_i \oplus a_{i+1} \quad 0 \leq i \leq m-2$$

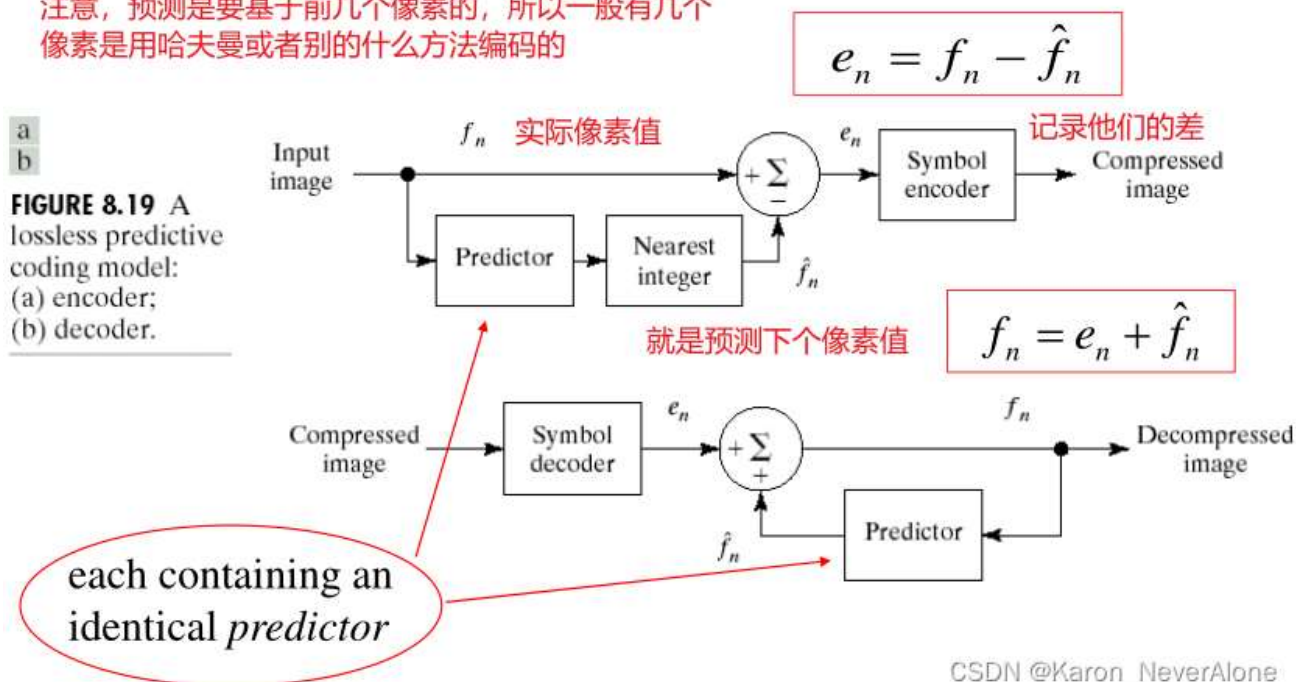
CSDN @Karon\_NeverAlone

## ①块编码

## ②游程编码

## ③预测编码（这个有讲的比较细节）

注意，预测是要基于前几个像素的，所以一般有几个像素是用哈夫曼或者别的什么方法编码的



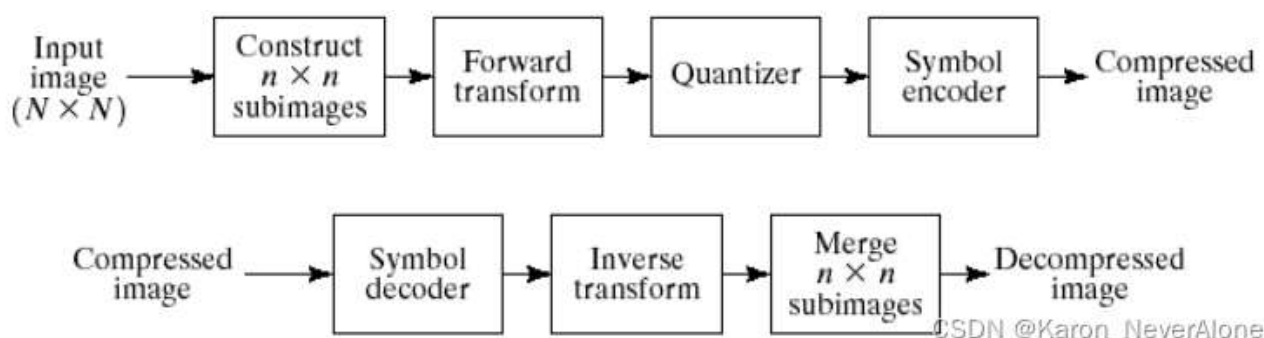
## ----有损压缩编码

有损的预测编码：对误差 $e_n$ 进行了量化

## DPCM差分脉冲编码

## 变换编码 (DCT)

变换可以去除冗余和相关性，可以使得图像的能量更集中





# Image compression using DCT

- Quantize
  - More coarsely for high frequencies (which also tend to have smaller values)
  - Many quantized high frequency values will be zero
- Encode
  - Can decode with inverse dct

Filter responses

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Quantized values

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

Quantization table

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

CSDN @Karon\_NeverAlone

-JPEG的编码方式

## JPEG Compression Summary

1. Convert image to YCrCb
2. Subsample color by factor of 2 对CbCr部分下采样保留一半
  - People have bad resolution for color
3. Split into blocks (8x8, typically), subtract 128
4. For each block
  - a. Compute DCT coefficients
  - b. Coarsely quantize
    - Many high frequency components will become zero
  - c. Encode (e.g., with Huffman coding)

CSDN @Karon\_NeverAlone

## -保真度准则

——客观逼真度标准（算出来的）——实验涉及

– root-mean-square (rms) error 均方根误差

$$e_{rms} = \left[ \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[ \hat{f}(x, y) - f(x, y) \right]^2 \right]^{\frac{1}{2}}$$

– mean-square signal-to-noise ratio 均方信噪比

$$SNR_{ms} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2 / \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[ \hat{f}(x, y) - f(x, y) \right]^2$$

– rms value of the signal-to-noise ratio 信噪比的rms值

$$SNR_{rms} = \sqrt{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2 / \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[ \hat{f}(x, y) - f(x, y) \right]^2}$$

CSDN @Karon\_NeverAlone

——主观逼真度标准：就是凭感觉，你觉得好就是好

## 12.形态学

**腐蚀**：结构元可以被完全包含

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

**膨胀**：结构元和主体有交集即可

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\}$$

**开（先腐蚀后膨胀）**：去掉结构元在主体中怎么移动都覆盖不到的部分

**闭（先膨胀后腐蚀）**

## 13.图像分割

-基于颜色的图像分割

基于HSI：一般是分H（色相）

基于RGB：首先先算出来感兴趣颜色的Average颜色，记作向量a，然后对相似性进行衡量，衡量的方法有

欧氏距离

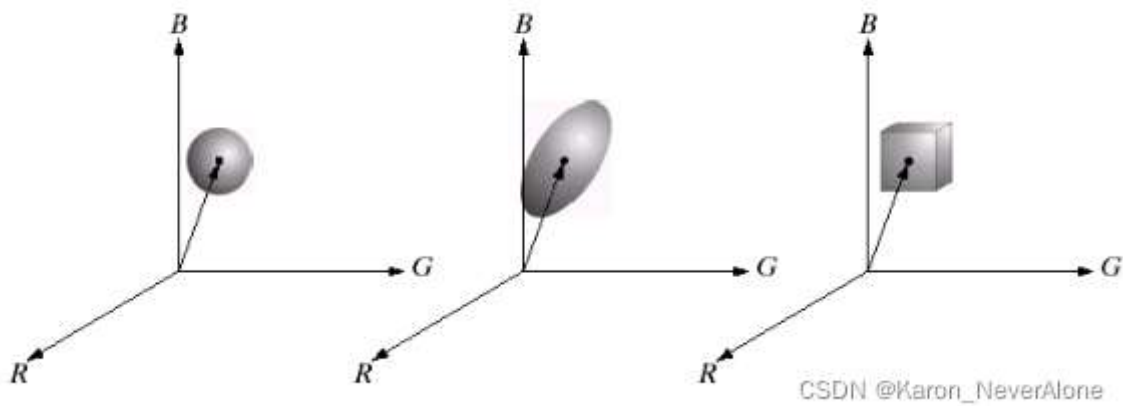
$$D(\mathbf{z}, \mathbf{a}) = \|\mathbf{z} - \mathbf{a}\|$$

$$= [(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})]^{\frac{1}{2}}$$

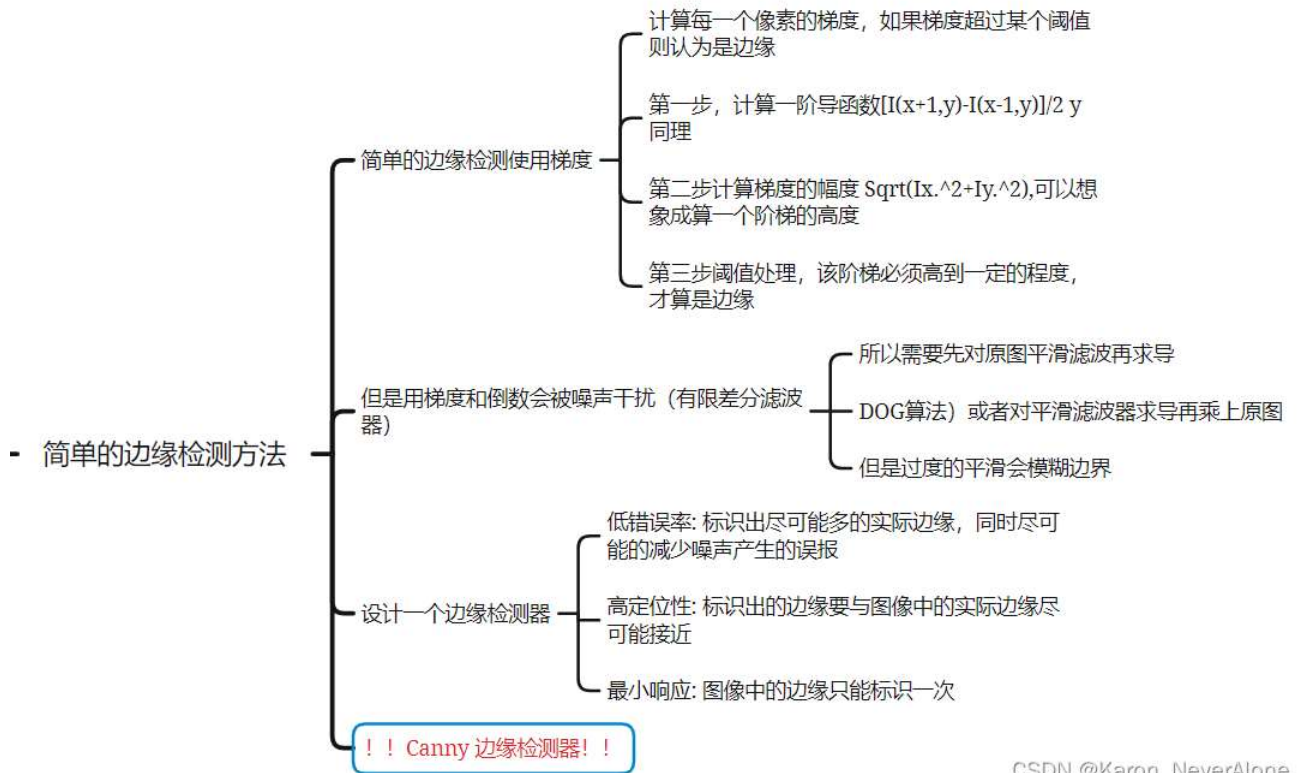
或者椭圆形的（C是样本的协方差矩阵）

$$D(\mathbf{z}, \mathbf{a}) = [(\mathbf{z} - \mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{a})]^{\frac{1}{2}}$$

或者正方形的



-边缘检测



CSDN @Karon\_NeverAlone

$$\text{Gradient Vector: } \nabla I = \left[ \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right]^T \quad \text{一阶方差}$$

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Magnitude: 梯度

$$\theta = \text{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

Orientation 边缘的方向

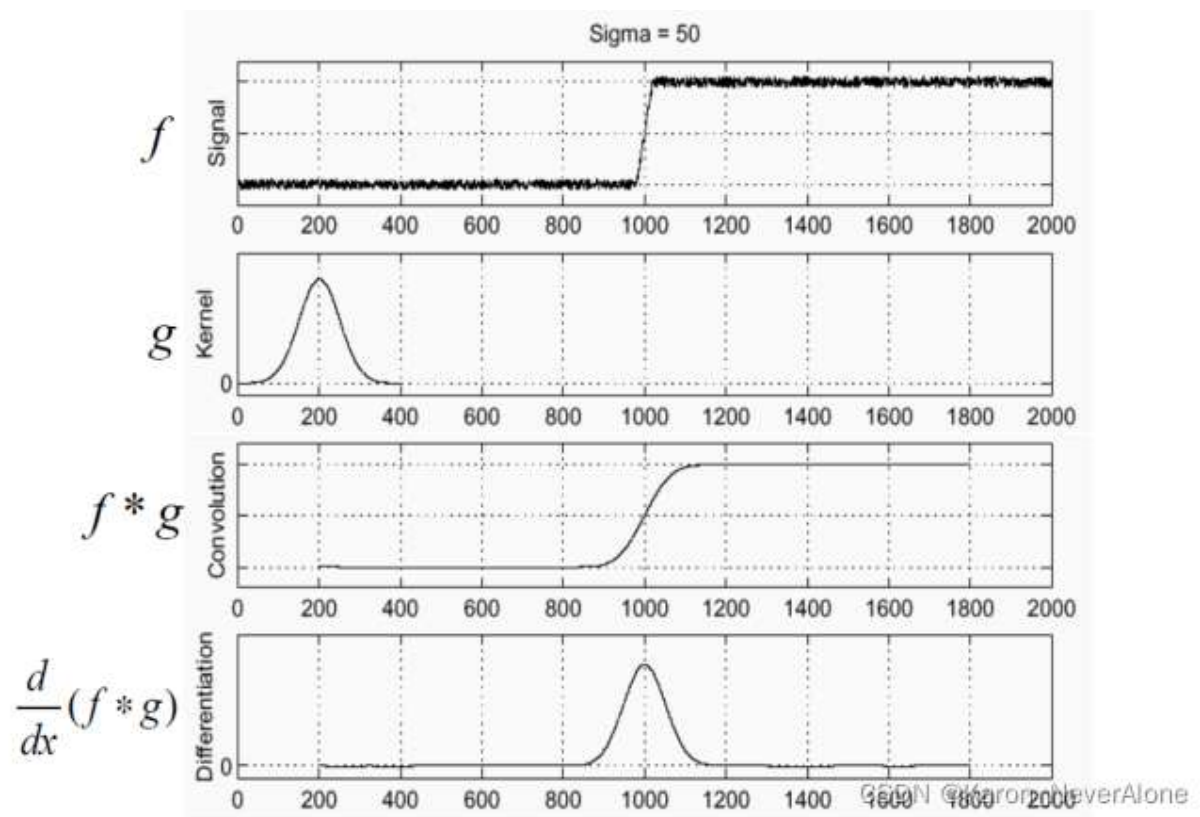
CSDN @Karon\_NeverAlone

## -霍夫变换

极坐标里的一个点 $\rho = x\cos\theta + y\sin\theta$ ，是霍夫空间的一个正弦曲线

## -DOG算法:先对滤波器求导再检测边缘





-边缘检测器 (canny)

## Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
  - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

CSDN @Karon\_NeverAlone

①高斯滤波 (不能太大, 不然边缘就会很模糊)

②使用Sobel算子计算像素梯度

$$G_x = S_x * I = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

$$G_y = S_y * I = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

CSDN @Yaron\_NeverAlone

### ③非极大值像素梯度抑制（这是理解难点）

沿着边缘方向，检测前后是否有比该点梯度的更大的点（用线性插值），如果该点最大，那就视他为边缘点，否则舍弃

### ④阈值滞后处理&孤立弱边缘抑制

设置一个最大阈值一个最小阈值，小于最小阈值的，直接舍弃，大于最大阈值的保留，中间的属于不知道是噪声还是边缘的，这时候就用大于最大阈值的部分做联通扩张，联通的部分视为边缘部分

篇幅限制不能赘述，可以看这个博主的

### -阈值化方法：

**全局阈值——该处有实验**

## 实验思想

需要有能对每幅图像自动估计阈值的算法，下面的迭代算法可用于这一目的：

1.为全局阈值  $T$  选择一个初始估计值。我准备直接选择图像的平均

灰度作为初始的  $T$ 。

2.使用用初始的  $T$  分割该图像。这将产生两组像素：

$G_1$  由灰度值大于  $T$  的所有像素组成， $G_2$  由所有小于等于  $T$  的像素组成。

3.对  $G_1$  和  $G_2$  的像素分别计算平均灰度值 (均值)  $m_1$  和  $m_2$ 。

平均灰度值怎么算？

4.计算一个新的阈值:  $T = (m_1 + m_2)/2$ 。

5.重复步骤 2 到步骤 4，直到连续迭代中的  $T$  值间的差小于个预定义参数  $\Delta T$  为止。书里面 $\Delta T$  取的是 0。

CSDN @Karon\_NeverAlone

### 最基本的OTSU

Otsu 算法计算步骤如下:

1. 计算输入图像的归一化直方图。使用  $p_i, i=0,1,2,..,L-1$  表示该直方图的各个分量。

2. 对于  $k=0, 1,2,..,L-1$ , 计算累积和  $P1(k)$  和  $P2(k)$ , 计算公式为:

$$P1(k) = \sum_{i=0}^k p_i = 1 - P2(k)$$

3. 对于  $k=0,1,2,...,L-1$ , 计算累积均值  $m1(k)$  和  $m2(k)$ 。计算公式为:

$$m1(k) = \frac{1}{P1(k)} * \sum_{i=0}^k (i * p_i) \quad m2(k) = \frac{1}{P2(k)} * \sum_{i=k+1}^{L-1} (i * p_i)$$

4. 计算全局灰度均值  $mg$ 。计算公式为:

$$mg = \sum_{i=0}^{L-1} (i * p_i)$$

5. 对于  $k=0,1,2,...,L-1$ , 计算类间方差  $b^2(k)$ , 公式为

$$b^2(k) = P1(k) * (m1(k) - mg)^2 + P2(k) * (m2(k) - mg)^2$$

6. 得到 Otsu 阈值  $k$ , 即使得  $b^2(k)$  最大的  $k$  值。如果最大值不唯一,

用相应检测到的各个值  $k$  的平均得到  $k^*$ 。

CSDN @Karon\_NeverAlone

## -阈值化方法的影响因素和改善

-影响因素: 噪声, 光照不均匀

-平滑滤波, 降噪之后再阈值化

-使用边缘来改善全局阈值

只考虑那些位于或靠近物体和背景之间的边缘的像素

-局部自适应的阈值


这个就是解决光照不均的问题, 把图像分块, 然后每一块都单独的阈值化

[关于我们](#)

[招贤纳士](#)

[商务合作](#)

[寻求报道](#)

 400-660-0108

 [kefu@csdn.net](mailto:kefu@csdn.net)

 在线客服

工作时间 8:30-22:00

[公安备案号11010502030143](#) [京ICP备19004658号](#) [京网文〔2020〕1039-165号](#) [经营性网站备案信息](#)

[北京互联网违法和不良信息举报中心](#) [家长监护](#) [网络110报警服务](#) [中国互联网举报中心](#) [Chrome商店下载](#) [账号管理规范](#)

[版权与免责声明](#) [版权申诉](#) [出版物许可证](#) [营业执照](#) ©1999-2024北京创新乐知网络技术有限公司