数字图像处理 重点知识梳理

by 严彬

本课程前半部分介绍低层次图像处理

Chap 01. Introduction (介绍)

Chap 02. Image and MALAB

Chap 03. Image Display (图像显示)

Chap 04. Point Processing (点处理)

Chap 05. Neighborhood Processing (领域处理)

Chap 06. Spatial Transformation (空间变换)

Chap 07. Fourier Transformation (傅立叶变换)

****

Chapter 01 Introduction

1.4 Images and Digital Images

xy 表示---> f (x, y)(空间域上是连续的)

We may assume that in such an image, brightness values can be any real numbers in the range 0.0 (black) to 1.0 (white)

数字图像采用 pixel 表示. x,y 上是离散的像素

1.8 Types of Digital Images

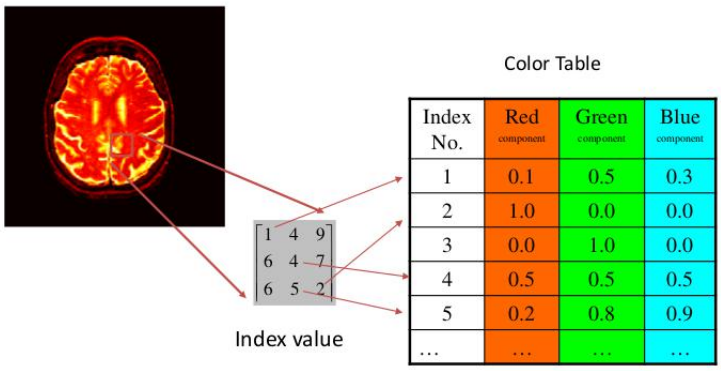
Binary(0 & 1)

Grayscale(0~255)

True color or red-green-blue (RGB)(3 个 0~255)

Indexed(图像的值都是 index.有一张 color table,记录的是 index 与 RGB 值的对应关系)

- **Indexed**



1.9 Image File Sizes

1byte = 8bits , 1kb = 1024b

Chapter 02 Image and MATLAB

2.3 Indexed Color Images

imread 会把图像真实的像素值读取出来,返回一个数组.(图像--->数组)

imshow 将数组显示为图像.(数组--->图像)

如果是索引图像的话,imread 除了返回一个数组之外,还会返回一个 color table

在用 imshow 显示的时候,需要把数组和 color table 一起传进去.

[**em, emap**] = imread('trees.tif');

figure, imshow(**em, emap**)

2.4 Data Types and Conversions

A Uint8 image must be converted to double before any arithmetic is attempted.

在做任何算术运算之前,都应该先将 uint8 转化成 double 类型.不然的话,超出 0~255 范围的值就无法表示了,会产生误差!

将图片转化成 double 类型有 2 种方法:

方法 1: **double**(A) --->将 datatype 变成 double,但是不改变值.

方法 2: **im2double**(A) --->将 datatype 变成 double,同时也把值从[0,255]映射到[0,1] (例如:原来图像中最大值是 252,则新图像中的最大值是 0.9882)

imshow 函数会根据 datatype 来决定输入的值应该在什么范围.

如果 datatype 是 double, 则 MATLAB 会期待它的值在 **0~1** 之间

如果 datatype 是 uint8, 则 MATLAB 会期待它的值在 **0~255** 之间

如果要正常显示一个 double 类型的数组,有两种方法:

方法一: 手动归一化到[0,1]

imshow(A / **max**(A(:)))

方法二: 自动归一化(在 imshow 里加一个[])

imshow(A,**[]**) ---> 会将数组的值对其范围自动归一化展示(show over the full range)

Chapter 03 Image Display

3.2 image 命令

image(C) 会将数组 C 中的数据显示为图像. C 的每个元素指定图像的 1 个像素的颜色.

The default color map is called **jet**, and consists of 64 very bright colors, which is inappropriate for the display of a grayscale image.

如果要用 image 命令正确显示一张灰度图像.则应该指定对应的 color map.

例如: colormap(gray(128)); colormap(gray(256))等.

可通过 size(unique(A)) 查看 A 中有多少种不同的灰度级.

3.5 Spatial Resolution

分辨率的答案应该是形如 mxn 这种(比如 256x256,64x64)

有效分辨率(effective resolution)等于下采样之后的分辨率!

3.6 Quantization and Dithering

grayslice 命令

grayslice(I,n) 其中 I 是图像数组,n 是阈值个数

将灰度图像转化成使用多级阈值的**索引图像**(Convert grayscale image to indexed image using multilevel thresholding)(在 imshow 的时候就需要加一个 colormap)

例如: imshow(grayslice(I,16), gray(16))就是**用不同数量的灰度级来展示图像**.

Dithering(抖动)(属于减少图像灰度级的一种方式)

将 Dithering 矩阵复制成和原图像一样大(可以用 repmat 来实现)

- Dithering matrix

$$D = \begin{bmatrix} 0 & 128 \\ 192 & 64 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$$

- D or D_2 is repeated until it is as big as the image matrix, when the two are compared
- Suppose $d(i, j)$ is the matrix obtain by replicating the dithering matrix, then an output pixel $p(i, j)$ is defined by

$$p(i,j) = \begin{cases} 1 & \text{if } x(i,j) > d(i,j) \\ 0 & \text{if } x(i,j) \le d(i,j) \end{cases}$$

在压缩到相同的灰度级数的条件下,dithering 的效果要好于 uniform quantilization

Chapter 04 Point Processing

4.3 Histograms

反映每个灰度级在图像中出现的次数的图.(A graph indicating the number of times each gray level occurs in the image)是一种统计特性

4.3.1 Histogram Stretching

采用一个拉伸方程,将原来 5-9 的灰度级映射到新的 2-14 灰度级

拉伸方程可以通过 直线的点斜式方程 得到

$$j = \frac{14-2}{9-5}(i-5)+2$$

如果用 MATLAB 自带的 imadjust 函数来做直方图拉伸的话,还有一个 gamma 参数可以指定.如果 gamma 控制的是映射曲线的凸凹性,类似于幂函数的幂指数的作用.

4.3.2 Histogram Equalization

原理: 对于任意一个(连续)概率分布,如果用其累积分布函数对它进行映射的话,会得到一个均匀分布! 对于离散的概率分布,虽然数学上不成立,但是也可以起到类似的效果.

步骤: 先求出累积分布,再乘以新的量化阶数(本题中是 15)

考试时严格按照下面这个步骤写!!

Gray level i	n_i	Σn_i	$(1/24)\Sigma n_i$	Rounded value
0	15	15	0.63	1
1	0	15	0.63	1
2	0	15	0.63	1
3	0	15	0.63	1
4	0	15	0.63	1
5	0	15	0.63	1
6	0	15	0.63	1
7	0	15	0.63	1
8	0	15	0.63	1
9	70	85	3.65	4
10	110	195	8.13	8
11	45	240	10	10
12	80	320	13.33	13
13	40	360	15	15
14	0	360	15	15
15	0	360	15	15

Chapter 05 Neighborhood Processing

Spatial filtering v.s Spatial convolution

后者的卷积核要翻转 180°(水平和垂直两个方向都翻转)

卷积神经网络里指的卷积其实都是 Spatial filtering

注: 卷积核大小一般都是**奇数**(3x3,5x5,7x7)原因是:在卷积核移动的过程中,我们往往会找一个中心作为参照物.(当然了,这也不是必须的,因为常见的 2x2 Max-Pooling 操作就可以看作是 2x2 卷积核以 2 为步长移动得到的)

根据对图像边缘的处理方式不同,MATLAB 自带的 filter2 函数也提供了 3 种方式:'valid','same','full'.假设原来图像尺寸为 x, 卷积核尺寸为 k, 卷积核移动步长为 1,则 3 种方式处理后的图像的尺寸分别为: **X-k+1**, X, **X+K-1**

valid 要求卷积核在移动过程中始终完全落在图像内部;

full 只要卷积核与图像有重叠就计算

Chapter 06 Image Geometry

6.1 Affine transformation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{v}' = \boldsymbol{H}_{\mathrm{A}}\boldsymbol{v} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{v}$$

图 6.1 仿射变换的形式

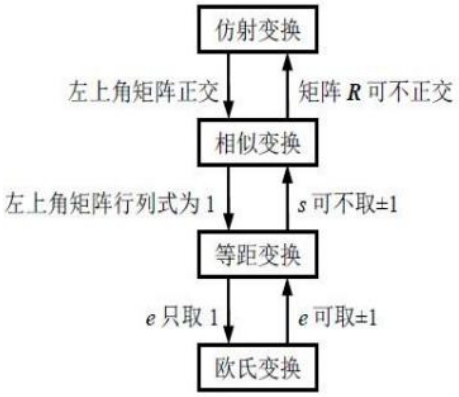
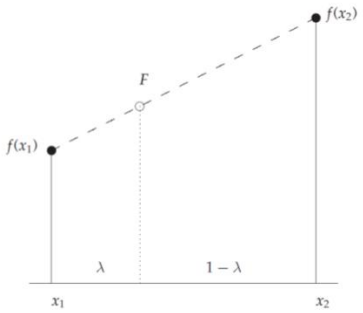


图 6.1.9 四种变换的层次体系

图 6.2 各种层次的仿射变换

6.2 Interpolation of Data

- ***Linear interpolation***



$$\frac{F - f(x_1)}{\lambda} = \frac{f(x_2) - f(x_1)}{1}$$

$$F = \lambda f(x_2) + (1 - \lambda) f(x_1).$$

(Equation 6.1)

是附近两个值的加权平均,"权重"是"对侧距离"之比

Chapter 07 The Fourier Transform
7.3 The 1D DFT

- Definition of the One-Dimensional DFT

$$\mathbf{f} = [f_0, f_1, f_2, \ldots, f_{N-1}] \quad \Rightarrow \quad \mathbf{F} = [F_0, F_1, F_2, \ldots, F_{N-1}],$$

$$F_u = \frac{1}{N}\sum_{x=0}^{N-1} \exp\left[-2\pi i \frac{xu}{N}\right] f_x. \qquad (7.2)$$

- This definition can be expressed as a matrix multiplication

$$F = \mathcal{F} f,$$

where $\mathcal{F}$ is an M × N matrix defined by

$$\mathcal{F}_{m,n} = \frac{1}{N}\exp\left[-2\pi i \frac{mn}{N}\right] \cdot \quad \mathcal{F}_{m,n} = \frac{1}{N}\omega^{mn}$$

Given N, we shall define

$$\omega = \exp\left[\frac{-2\pi i}{N}\right]$$

$$\mathcal{F} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \cdots & \omega^{3(N-1)} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \cdots & \omega^{4(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \omega^{4(N-1)} & \cdots & \omega^{(N-1)^2} \end{bmatrix}$$

MATLAB 中经常用到的与傅里叶变换相关的函数

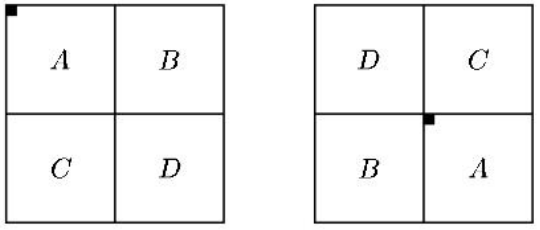
- <*fft*>, which takes the DFT of a vector,
- <*ifft*>, which takes the inverse DFT of a vector,
- <*fft2*>, which takes the DFT of a matrix,
- <*ifft2*>, which takes the inverse DFT of a matrix, and
- <*fftshift*>, which shifts a transform (Fig. 7.7)

我对图像傅里叶变换的理解:

图像的傅里叶变换中的"频率"指的是"空域上变化得快与慢",傅里叶变换的"值"指的是各种成分的强与弱.(快速变化的成分多还是缓慢变化的成分多)

展示傅里叶变换的时候,我们往往会先加一个 **fftshift** 来把频谱移动一下,**使得移动后的频谱中心是低频,外部是高频**.

示意图如下:



An FFT　　After shifting

Figure 4.6: Shifting a DFT

展示经过 fftshift 后的频谱时,常常有 2 种方法(教材 fftshow 函数)
一种是**直接归一化.**即|F(u,v)|/max(|F(u,v)|),这种方法的缺点是当各种频率成分的值(强弱)相差较大的时候(比如一个极值特别大,其他几个极值和它相比特别小,比如 10:1),这个时候其他极值就会很不明显.
另一种是取 **log(1+|F(u,v)|)**之后**再归一化.**这种方法克服了上面的方法的缺点,明显缩小了极值之间的差异,(10:1 就变成了大约 3:1),会让次极大值变得更加明显.
理想低通滤波器的问题
引入了 ringing(出现了很多波动)

改进方法:
Butterworth 滤波器, Gaussian 滤波器
从滤波效果来看:
Gaussian Filtering > Butterworth Filtering > Ideal Filtering
本书后半部分介绍高层次图像处理
Chap 08. Image Restoration (图像恢复)
Chap 09. Image Segmentation (图像分割)
Chap 10. Mathematical Morphology (数学形态学)
Chap 11. Image Topology (图像的拓扑结构)
Chap 12. Shapes and Boundaries (形状和边界)
Chap 13. Color Processing (彩色处理)
Chap 14. Image Coding and Compression (图像编码及压缩)

Chapter 08 Image Restoration

综述:

Salt & Pepper noise, Also called impulse noise, shot noise, or **binary noise**
Gaussian noise is an idealized form of **white noise**
Speckle noise can be modeled by random values **multiplied** by pixel values

Salt & Pepper noise
Gaussian noise
Speckle noise
} **Spatial filtering**

These noise can be modeled as local degradations

Periodic noise ➔ Frequency domain filtering
Periodic noise is a global effect

Noise

- Salt & Pepper noise
  - Average Filtering(效果一般)
  - **Median Filtering**(中值滤波滤除椒盐噪声效果特别好)
  - Rank-Order Filtering(中值滤波属于 rank-order 滤波的特例)
  - **Outlier Filtering**(不像中值滤波需要排序,只需要求均值,速度会更快,效果也不错)
- Gaussian Noise
  - Image Averaging(对多张图片求平均)
  - Average Filtering(可以抑制噪声,但是会让图像变模糊)
  - **Adaptive Filtering**(对高斯噪声抑制效果好,虽然会让图片变模糊,但是边缘仍然可以保留下来,这一点要好于直接使用低通滤波器)
- Speckle Noise
- Periodic Noise
  - Band Reject Filtering(带阻滤波器,圆圈)
  - Notch Filtering(陷波滤波器,两组平行线)

• Blurring(了解)
  •Inverse Filtering
  •Wiener Filtering

Outlier Method(异常值方法)步骤

- Choose a threshold value D
- For a given pixel, compare its value p with the mean m of the values of its eight neighbors
- If $|p-m|>D$, then classify the pixel as noisy, otherwise not
- If the pixel is noisy, replace its value with m; otherwise leave its value unchanged

Chap 09 Image Segmentation

9.6 Edge Detection

经典一阶边缘检测算法采用空域滤波方法,最有名的算法有以下几种

Prewitt filters, Roberts cross-gradient filters, Sobel filters

它们的卷积核依次如下图所示,

$$P_x=\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}\quad P_y=\begin{bmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{bmatrix},\quad \begin{bmatrix}1&0&0\\0&-1&0\\0&0&0\end{bmatrix}\text{ and }\begin{bmatrix}0&1&0\\-1&0&0\\0&0&0\end{bmatrix},\quad \begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}\text{ and }\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$$

If Px and Py are the gray values produced by Px and Py to an image, then the magnitude of the gradient is obtained by

$\sqrt{p_x^2+p_y^2}$ ----> 将两个方向上的梯度信息融合在一起

9.8 Second Derivatives

Second derivative filters are **very sensitive to noise**

The Laplacian gives double edges

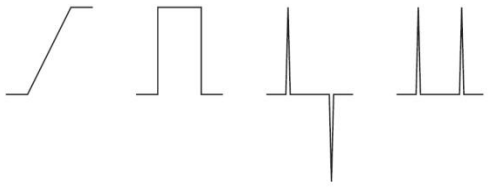


FIGURE 9.21 *Second derivatives of an edge function.*

Of the three filters, the **Sobel filters** are probably the best; they provide good edges, and they perform **reasonably well in the presence of noise**.

下面这张总结是重点:

- **Edge Detection**
  - **Prewitt Filter**
  - **Roberts Filter (Cross-gradient)**
  - **Sobel Filter (Robustness for Noise)**
  - **Laplacian + Zero Crossing (Weak for Noise)**
  - **LoG smoothing +zeros Crossing (Robustness for Noise)**
  - **Canny Detector**
    - ✓ **Non-maximum suppression**
    - ✓ **Hysteresis Thresholding**

The Process of Canny edge detection algorithm can be broken down to 5 different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

解释:

1. 这里说的 Gaussian filter 并不是 Fourier Transform 里讲的频域滤波器.而是一种 Spatial Domain 滤波器.(常用的是 5x5 的)
2. 用边缘检测算子(例如 prewitt,robert,sobel)得到水平梯度 Gx,垂直梯度 Gy.并用

arctan 函数计算出和坐标轴的夹角,并对其量化(例如:选择 4 个量化角度 0°,45°,90°,135°)

3. 非极大值抑制.给定一个像素,假设它的梯度方向是南北方向,则其对应的边缘方向应该是东西方向,这时应该将在南北方向上与该像素相邻的像素的梯度值与该像素的梯度值相比较,如果该像素的梯度值最大,则将其标记为 edge,否则就被抑制掉(不是 edge).

4. 使用双重阈值(TH,TL).高于 TH 的被标记为"strong edge",介于 TH 和 TL 之间的被标记为"weak edge",低于 TL 的被抑制掉(一定不是 edge).

5. 最后再特别处理一下 weak edge.把被标记为 weak edge 的像素与它的 8 连接邻居像素作比较.只要邻居中有 strong edge,那么这个 weak edge 就应该被保留.如果邻居中没有 strong edge,则该像素应该被舍弃.

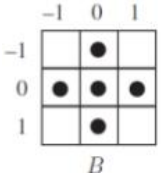
Chapter 10 Mathematical Morphology

10.3.1 **Dilation** 含义: 在 A 上的每个位置,都用 B 走一遍,把得到的所有区域取并集.

10.3.2 **Erosion** 含义: A 中所有能把 B 完全包围住的位置所构成的集合.

内部边界, 外部边界, 形态学梯度

假设 B 是一个小型的简单结构(比如下图所示的这种)



(i)	$A-(A\ominus B)$	internal boundary	***内部边界***
(ii)	$(A\oplus B)-A$	external boundary	***外部边界***
(iii)	$(A\oplus B)-(A\ominus B)$	morphological gradient	***形态学梯度***

从公式中可以容易看出: **形态学梯度 = 内部边界 + 外部边界**

10.4.1 Opening

含义: A 先对 B 求腐蚀,之后再对 B 求膨胀! 可以想象,**结果会比原图像少一些东西**

效果(作用): 可以滤除掉原来图片中的一些多余的细小瑕疵(因为这些微小的结构求了腐蚀之后就变成空集了,再做膨胀也不会把它还原回来))用 ppt 里的话来说就是"Opening tends to **smooth** an image, to **break narrow joins, and to remove thin protrusions**"

10.4.2 Closing

含义: A 先对 B 求膨胀,再对 B 求腐蚀. 可以想象,**结果会比原图像多一些东西**.

效果: 可以补全原图中的一些微小的裂痕,缝隙.Closing also tends to smooth an image, but it **fuses narrow breaks and thin gulfs and eliminates small holes**

这两种操作的特点:

只能做一次,原因是做更多次的效果不会继续发生变化.用公式表示就是:(这个公式我还没想好怎么证.....s)

$$(A\circ B)\circ B = A\circ B$$

$$(A\bullet B)\bullet B = A\bullet B$$

消除二值图像中的噪声,可以采用"**先开后闭**"的方式!

10.6.2 Skeletonization

从 A 开始,用**当前形状**与"**当前形状的开**"***做差***

接下来依次对当前形状求**腐蚀**,作为下一时刻的当前形状

迭代下去,直到无法继续计算,把前面每一时刻计算出的"差值"累加起来,即为骨化的结果.

Chapter 11 Image Topology

11.3 Paths and Components

这一块课件讲得不清楚,我上 wikipedia 又查了查

Definition of a **4-connected component** :

A set of black pixels, P, is a 4-connected component if for every pair of pixels pi and pj in P, there exists a sequence of pixels pi, ..., pj such that:

a) all pixels in the sequence are in the set P i.e. are black, and

b) every 2 pixels that are adjacent in the sequence are **4-neighbors**

Definition of an **8-connected component:**

A set of black pixels, P, is an 8-connected component (or simply a connected component) if for every pair of pixels pi and pj in P, there exists a sequence of pixels pi, ..., pj such that:

a) all pixels in the sequence are in the set P i.e. are black, and

b) every 2 pixels that are adjacent in the sequence are **8-neighbors**

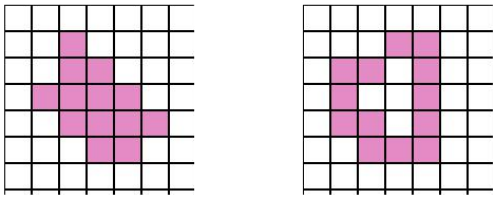


图 1: 典型的四连接分量

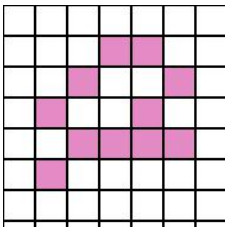


图 2: 典型的 8 连接分量

11.7 Distances and Metrics

已知从一个点往周围若干方向走所需要付出的距离(代价)

方法一:

通过**迭代**的方式,计算图上每一个点对应的**最短距离**.(一次只解决一部分点的最短距离就可以了,迭代几轮就可以知道图中每个点的最短距离)

方法二:(只需要进行 2 轮 pass(方向正好相反,每次只考虑一半的 mask),速度更快)

例如:

第一轮从左向右,从上往下,每次只考虑 top-left 方向的点,计算与它们之间的最短距离

第二轮从右向左,从下至上,每次只考虑 bottom-right 方向的点,计算与它们的最短距离.

下面给出了用 MATLAB 实现的思路:

✓ **IMPLEMENTATION IN MATLAB**

1. Using the size of the mask, pad the image with an appropriate number of 0s
2. Change each 0 to infinity, and each 1 to 0
3. Create forward and backward masks
4. Perform a forward pass. Replace each label with the **minimum** of its neighborhood plus the forward mask
5. Perform a backward pass. Replace each label with the **minimum** of its neighborhood plus the backward mask

11.8 Skeletonization

11.8.3 The Zhang-Suen Skeletonization Algorithm

通过反复迭代,每经过一次迭代,就删除一部分像素,删到最后所有剩下的像素都是 undeleteble 了,就结束了.得到的结果就是 skeleton.

Step N Flag a foreground pixel p = 1 to be deletable if

(翻译一下上面这一行: **在第 N 次迭代中,如果满足下面这 3 个条件,就把该前景像素标记为"可删除的"**)

1. **2 ≤ B(p) ≤ 6**,
2. **X(p) = 1**,
3. If N is odd, then

   $p2 \cdot p4 \cdot p6 = 0$

   $p4 \cdot p6 \cdot p8 = 0$

   If N is even, then

   $p2 \cdot p4 \cdot p8 = 0$

   $p2 \cdot p6 \cdot p8 = 0$

我们通过检查以该像素为中心的 3x3 邻域,来判断该像素是否"可删除"

B(P)是去掉该像素后,邻域内 foreground 像素的个数

X(P)叫做 Crossing number.把该像素的 8 邻居按照顺时针方向展开,其中'01'这个组合出现的次数

Chapter 12 Shapes and Boundaries

12.2 Chain Codes and Shape Numbers

The idea of a **chain code** is quite straightforward:

- We walk around the boundary of an object, taking note of the direction we take
- The resulting list of directions is the chain code

Suppose we walk along the boundary in a **counter-clockwise** direction starting at the leftmost point in the top row and list the directions as we go

12.2.1 **Normalization** of chain codes

There are two **problems** with the definition of the chain code as given in previous sections:

The chain code is dependent on the starting pixel

The chain code is dependent on the orientation of the object.

起始点问题:

链码的循环移位还是链码.这时,我们会把链码看做一个整数,我们选最小的整数作为结果.

物体方向问题:

如果物体发生旋转,那么按照原来链码的编码规则,其结果也会发生改变.但是物体的几何形状(边与边之间的夹角并没有发生改变:))这时我们想到用**前后两个码的差分**来作为编码结果!

Chap 13. Color Processing

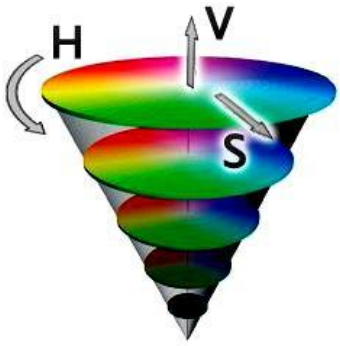
13.2 Color Models

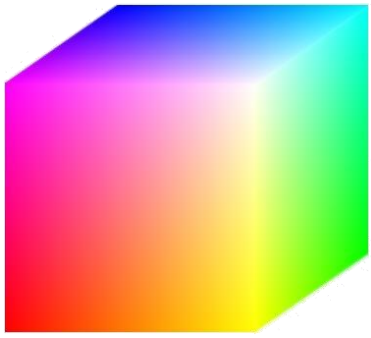
13.2.2 HSV Model

Hue(色彩): The “true color” attribute (red, green, blue, orange, yellow, and so on)

Saturation(饱和度): The amount by which the color has been diluted with white. The more white in the color, the lower the saturation

Value(明暗度): The degree of brightness. A well-lit color has high intensity; a dark color has low intensity





HSV 与 RGB 之间的转换关系(在我看来,其实就是”RGB 采用直角坐标系,HSV 采用柱坐标系”)

本质: **同一种颜色**,采用不同的表达方式,值也不同.个人感觉不用背公式,考试应该会给.

13.2.4 YIQ Model(采用 NTSC 标准)

In this scheme **Y** is the luminance (大致等价于”**亮度**”)

I and **Q** carry the color information.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$ ***RGB → YIQ* (rgb2ntsc)**

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$ ***YIQ → RGB* (ntsc2rgb)**

RGB 与 NTSC(National Television system committee)转换的指令

rgb2ntsc, **ntsc2rgb**

重点,重点,重点!!!

把 13.4 的所有代码都好好看!!!考试很可能会让我们编写类似的代码