第2章 图像处理基础

本章是整本书的基础，介绍图像处理的一些基本概念以及最简单的算术和逻辑运算。此外，图像处理经常遇到的直方图等统计量也放入这一章。

2.1 基本概念

2.1.1 图像坐标系

数字图像是以二维数组（矩阵）形式表示的图像，其数字单元为像素。每个像素具有整数行（高）和列（宽）位置坐标，同时每个像素都具有灰度值或颜色值。此外，像素是有大小和形状的，如无特殊说明，本书中的像素都是 1×1 的矩形。为了便于对像素的操作，还需要建立图像坐标系。如图 2.1 所示，我们将图像的左上角定义为原点，横轴为 x 轴，纵轴为 y 轴。这种坐标系与屏幕坐标系是一致的。在编程时使用一些系统函数需要注意，因为 y 轴与笛卡儿坐标系的 y 轴方向不同。有了坐标系，就可以通过坐标来描述像素了，在坐标 (x,y) 处的像素灰度值记为 $f(x,y)$ 或 $I(x,y)$，此外，$f(x,y)$ 和 $I(x,y)$ 也用于表示图像。显然，这样一幅矩形图像可以用矩阵来描述，其中每个元素对应一个像素。实际上，著名软件 MATLAB[2024]和 OpenCV[2012]就是用矩阵来描述图像的。

● 图 2.1　图像坐标系

在进行图像处理时，图像在计算机内存中是按线性方式连续存储的，即从左到右，从上到

下连续存放。对于最常用的 8 位灰度图像来说，坐标(x,y)处的像素在内存中相对于图像数据起始点的地址偏移量为

$$\text{offset}(x,y)=w\times y+x \tag{2.1}$$

式中，w 为图像宽度。但也有例外，设备无关位图（Device Independent Bitmap，DIB）在内存中是上下颠倒存放的，并且要求 4 字节对齐，即每一行的字节数都要取为 4 的整数倍。因此，对于 8 位灰度 DIB，坐标(x,y)处的像素的地址偏移量则为

$$\text{offset}(x,y)=B_l\times(h-y-1)+x \tag{2.2}$$

式中，h 为图像高度；B_l为每一行的字节数，计算公式为

$$B_l=\text{floor}\left(\frac{w\times 8+31}{32}\right)\times 4 \tag{2.3}$$

式中，w 为图像宽度，floor 表示向下取整。对于不能满足 4 字节对齐的 DIB，则在每行最后进行比特填充（以 0 填充），以保证 4 字节对齐。在这种情况下，图像数据在内存中就不是连续存放的。此外，在运行某些硬件加速函数时，比如 Intel 的 SSE2 指令集，需要数据在内存中的起始地址 16 字节对齐，对于非 16 字节对齐的起始地址，则可通过移位来保证起始地址 16 字节对齐，这种情况也会造成数据非连续存放。

2.1.2 距离测度

参考 12.3.5 节，图像上两个像素点 $p_1(x_1,y_1)$、$p_2(x_2,y_2)$之间的距离 L_p定义为

$$L_p(p_1,p_2)=(|x_1-x_2|^p+|y_1-y_2|^p)^{1/p},\quad p\geqslant 1 \tag{2.4}$$

当 $p=2$ 时，就是经典几何学和日常经验中的欧几里得距离（简称欧氏距离）

$$L_2(p_1,p_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{2.5}$$

欧氏距离的优点是它在事实上是直观且显然的，缺点是平方根的计算费时且其数值不是整数。

当 $p=1$ 时，称为曼哈顿距离或城市街区距离

$$L_1(p_1,p_2)=|x_1-x_2|+|y_1-y_2| \tag{2.6}$$

之所以称为城市街区距离，是因为它类似于具有栅格状街道和封闭房子块的城市里的两个位置的距离，只允许横向和纵向的移动。

当 $p=\infty$ 时，称为棋盘距离

$$L_\infty(p_1,p_2)=\max(|x_1-x_2|,|y_1-y_2|) \tag{2.7}$$

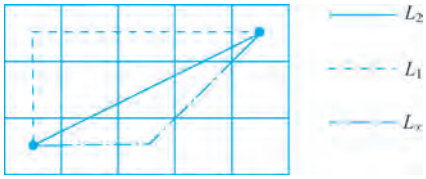
该距离等于国际象棋中国王在棋盘上从一处移动到另一处所需的步数。

图 2.2 给出了这些距离的图示。

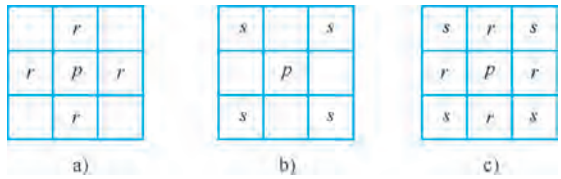
2.1.3 邻域

如图 2.3a 所示，每个方格代表一个像素，像素 p 共有 8 个与之相邻的像素，其中 4 个水平

和垂直的邻近像素（用 r 表示）组成 p 的 4 邻域，记为 $N_4(p)$。4 邻域像素之间的城市街区距离 $L_1=1$；像素 p 的 4 个对角邻近像素（用 s 表示）记为 $N_D(p)$，如图 2.3b 所示；$N_4(p)$ 加上 $N_D(p)$ 合称为 p 的 8 邻域，记为 $N_8(p)$，如图 2.3c 所示。8 邻域像素之间的棋盘距离 $L_\infty=1$



● 图 2.2　距离测度 L_2、L_1、L_∞



● 图 2.3　像素的邻域

a）4 邻域　b）$N_D(p)$　c）8 邻域

2.1.4　连通性

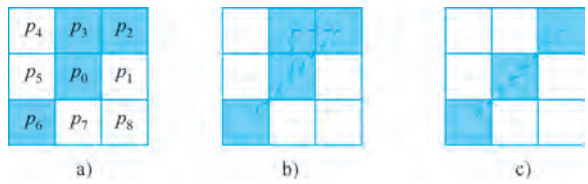
像素间的连通性在确定图像中目标的组成元素时是一个重要的概念，例如第 8 章的区域分析就是以连通性为基础的。虽然连通性适用于灰度图像，但是更多时候是在二值图像下讨论具有相同灰度值的像素之间的连通性，所以这里仅讨论二值图像的连通性。常用的连通种类有 4 连通和 8 连通两种：

1）4 连通：两个像素 p 和 r 的灰度值相同且 r 在 $N_4(p)$ 中，则它们为 4 连通。

2）8 连通：两个像素 p 和 r 的灰度值相同且 r 在 $N_8(p)$ 中，则它们为 8 连通。

如图 2.4 所示，图 2.4a 中的像素分为白色和灰色两种。当考虑灰色像素的 4 连通时，p_3 是 p_0 的 4 连通像素；当考虑 8 连通时，p_3、p_2、p_6 是 p_0 的 8 连通像素。

在图像处理中，经常会遇到提取 8 连通单像素边缘的情况，但实际提取的边缘经常存在多余像素，出现多路连通问题。如图 2.4b 所示，中心像素 p_0 和右上角像素 p_2 之间产生两条连线，分别是 $p_0p_3p_2$ 和 p_0p_2。解决办法就是删除 p_0 和 p_2 共享的 4 连通像素 p_3，如图 2.4c 所示。图 2.4c 这种连通称为 m 连通（或称为混合连通）[章，1999]。



● 图 2.4　图像的连通和区域

a）二值图像　b）多路连通　c）m 连通

2.1.5　连通区域

由一些彼此连通的像素组成的集合，我们称之为区域（region）。区域又被称为连通区域、连通分量或 blob，在本书不同章节可能使用不同的称谓。如图 2.4a 所示，当考虑 4 连通时，灰色像素 $p_0p_2p_3$ 组成一个区域，虽然 p_2 不是 p_0 的 4 邻域，但它是 p_3 的 4 邻域，而 p_3 是 p_0 的 4 邻域；

当考虑 8 连通时，$p_0p_2p_3p_6$组成一个区域，这时将白色像素分成了两个区域：p_4p_5和 $p_1p_7p_8$。但是，在 8 连通下，所有的白色像素又是连通在一起的，因此造成矛盾。为了解决同时处理两种灰度像素时产生的矛盾，必须采用不同的连通准则，即如果一种灰度像素采用 8 连通，另外一种就必须采用 4 连通。

2.1.6 感兴趣区域

在图像处理中，我们可能会对图像的某一个特定区域感兴趣，该区域被称为感兴趣区域（Region Of Interest，ROI）。ROI 可以是矩形，也可以是任意形状。矩形 ROI 通过 4 个顶点确定，而任意形状 ROI 就需要根据掩膜（mask）来确定。掩膜通常是一个与原图像大小相同的二值图像，其中选定区域被标记为 1，其余区域被标记为 0。在对图像进行处理时，根据掩膜像素值来判断原图像上对应点是否属于 ROI。ROI 在图像处理中随处可见，例如，在图像匹配中（见第 10 章），相比整幅图像搜索，在小范围的 ROI 内搜索目标的好处有：提高搜索速度，因为搜索面积变小；提高搜索稳定性，因为在 ROI 内更不容易匹配到错误目标。

与 ROI 相对应的就是不感兴趣（Don't Care）区域，顾名思义，就是该区域的像素不参与计算。不感兴趣区域也是通过掩膜实现的，通常用来对图像匹配中的模板进行设置（见 10.1.3.5 节），使得模板部分区域不参与图像匹配，改进匹配效果。

2.1.7 点运算与邻域运算

在图像处理中，点运算是简单却很重要的一类运算。对于一幅输入图像，经过点运算将产生一幅输出图像，后者的每一个像素点的灰度值仅由相应输入像素点的灰度值决定。这种方法与邻域运算的差别在于，后者每个输出像素的灰度值由相应输入像素的一个邻域内几个或十几个像素的灰度值决定。

若输入图像为 $f(x,y)$，输出图像为 $g(x,y)$，则点运算可表示为

$$g(x,y)=H[f(x,y)] \tag{2.8}$$

点运算可完全由灰度变换函数 $H(\cdot)$确定，后者描述了输入像素的灰度值和与对应的输出像素的灰度值之间的映射关系。例如，一幅图像乘以常数 C，即图像中的每个像素都乘以 C，输出图像的每个像素的灰度值仅与输入图像的对应像素的灰度值及 C 有关。

与点运算对应的则是邻域运算。令 S_{xy}代表图像 f 中以任意一点(x,y)为中心的一个邻域的坐标集。邻域运算在输出图像 g 中的相同坐标处生成一个对应的像素，这个像素的值由输入图像中邻域像素的规定运算和集合 S_{xy}中的坐标确定。例如，假设规定的运算是计算大小为 $m\times n$、中心为(x,y)的矩形邻域中的像素的平均值。这个区域中的像素坐标是集合 S_{xy}的元素。我们用公式将这一平均运算表示为

$$g(x,y)=\frac{1}{m\times n}\sum_{(c,r)\in S_{xy}} f(c,r) \tag{2.9}$$

式中，c 和 r 是像素的列坐标和行坐标，这些坐标属于集合 S_{xy}。

在做邻域运算时，一定要分配两块内存，分别存放输入图像和输出图像，否则会造成处理完的像素又放回输入图像，接着又参与下一像素的处理，造成错误。

2.1.8 线性运算与非线性运算

除了点运算和邻域运算分类外，图像处理算法还有另外一种分类：线性运算和非线性运算。考虑一般算子[⊖] H，该算子对给定的一幅输入图像 $f(x,y)$ 产生一幅输出图像 $g(x,y)$

$$H[f(x,y)]=g(x,y) \tag{2.10}$$

给定两个任意常数 a 和 b，以及两幅任意图像 $f_1(x,y)$ 和 $f_2(x,y)$，若

$$H[af_1(x,y)+bf_2(x,y)]=aH[f_1(x,y)]+bH[f_2(x,y)]=ag_1(x,y)+bg_2(x,y) \tag{2.11}$$

则称 H 是一个线性算子。这个公式表明，两个输入求和的线性运算的输出，与分别对输入进行运算并求和得到的结果相同；另外，输入乘以常数的线性运算的输出，与对输入进行运算并乘以常数的输出相同。按照定义，不满足式（2.11）的运算就称为非线性运算。

很显然，图像乘以常数以及图像之间的加减运算属于线性运算。而中值（median）滤波则为非线性运算（见 4.1.2.4 节）。设有两幅图像

$$\boldsymbol{f}_1=\begin{bmatrix}0 & 2\\ 2 & 3\end{bmatrix} \text{和} \boldsymbol{f}_2=\begin{bmatrix}6 & 5\\ 4 & 7\end{bmatrix}$$

并假设 $a=1$ 和 $b=2$。将上面图像代入式（2.11）的等号左边

$$\text{median}\left\{(1)\begin{bmatrix}0 & 2\\ 2 & 3\end{bmatrix}+(2)\begin{bmatrix}6 & 5\\ 4 & 7\end{bmatrix}\right\}=\text{median}\left\{\begin{bmatrix}12 & 12\\ 10 & 17\end{bmatrix}\right\}=12$$

由于数量是偶数，中值滤波取中间两个数的均值。将图像代入式（2.11）的等号右边

$$(1)\,\text{median}\begin{bmatrix}0 & 2\\ 2 & 3\end{bmatrix}+(2)\,\text{median}\begin{bmatrix}6 & 5\\ 4 & 7\end{bmatrix}=2+(2)\times 5.5=13$$

此时，式（2.11）的等号左边和右边不相等，证得中值滤波是非线性的。

2.1.9 边缘扩展

在 2.1.7 节我们讲到邻域运算的输出像素的灰度值由输入像素邻域内几个或十几个像素的灰度值决定。如图 2.5 左图所示，这是一幅 8×8 的图像，每个方格代表一个像素。假设邻域为图中的 3×3 的彩色窗口，当处理边缘像素时，会发现窗口有部分区域没有覆盖到像素，无法进行邻域计算，导致输出图像边缘像素没有值。对于 3×3 邻域来说，影响的边缘宽度只有一个像素，但是对于 25×25 邻域来说，无法处理的边缘宽度就是 12 个像素。在实际图像处理中，如果不进行

⊖ 算子（operator）的定义相当宽泛，对图像的任何操作都可以认为是一个算子，例如，对图像的加减乘除操作可以认为是算子，但是算子更多还是用于对滤波器的称呼，如 Sobel 算子、Canny 算子等。

边缘扩展而强行进行邻域计算，会造成数组越界等不可预料的结果。因此在使用大尺寸邻域时，必须对图像边缘进行扩展。常用的扩展部分的像素填充方式有如下几种[OpenCV，2012]。

1）边缘像素复制扩展：aaa|abcdefgh|hhh。

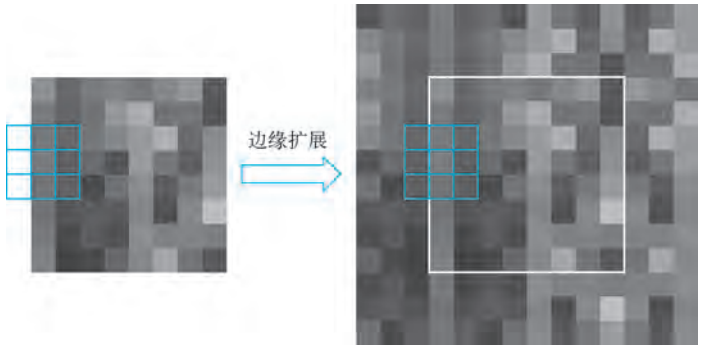
2）边缘像素镜像扩展：cba|abcdefgh|hgf。

3）边缘像素镜像扩展 1：dcb|abcdefgh|gfe。

4）常量扩展：iii|abcdefgh|iii。

其中，“abcdefgh”为原图像数据，“|”为边界，其外侧的数据为填充值。第 1）、2）、3）种扩展方式在实际中使用比较多。

图 2.5 右图给出了第 3）种扩展方式的示例，每个边缘向外扩展了 3 个像素。如图所示，经过边缘扩展，已经可以处理边缘像素了，在图示的情况下，最大支持 7×7 的邻域。在本书后面的讨论中，都假设根据需要对图像边缘进行了扩展。



• 图 2.5　边缘扩展

2.2 彩色图像

2.2.1 彩色基础

根据人眼结构，所有颜色都可看作是三个基本颜色——红（Red）、绿（Green）和蓝（Blue）的不同组合。为了建立标准，国际照度委员会（CIE）早在 1931 年就规定三种基本色的波长为“R：700 nm、G：546.1 nm、B：435.8 nm”。利用三基色叠加可产生光的三补色（二次色）：青色（Cyan，绿加蓝）、品红色（Magenta，红加蓝）、黄（Yellow，红加绿）。按一定比例混合三基色或将一个补色与相对的基色混合就可以产生白色。

以上讲的是光的三基色，除此之外还有颜料的三基色。颜料的基色是指吸收一种光基色并让其他两种光基色反射的颜色，所以颜料的三基色正是光的三补色，而颜料的三补色正是光的

三基色。如果以一定的比例混合颜料的三基色或者将一个补色与相对的基色混合就可以得到黑色。

区分不同颜色的特性通常是亮度、色调和饱和度。颜色中掺入白色越多就越明亮，掺入黑色越多亮度就越小。色调是与混合光谱中主要波长相联系的，是被观察者感知的主导色。饱和度与一定色调的纯度有关，纯光谱色是完全饱和的，随着白光的加入饱和度逐渐减小。

2.2.2 颜色模型

为了正确地使用颜色，需要建立颜色模型。上面提到，一种颜色可用三个基本量来描述，所以建立颜色模型就是建立一个 3D 坐标系，其中每个空间点都代表某一种颜色。

目前常用的颜色模型可分为两类，一类面向诸如彩色显示器或者打印机之类的硬件设备；另一类面向以彩色处理为目的的应用，如为动画创建的彩图。针对彩色显示器和彩色摄像机开发的 RGB（红色、绿色、蓝色）模型；针对彩色打印机开发 CMY（青色、品红色、黄色）模型和 CMYK（青色、品红色、黄色、黑色）模型。而面向彩色处理最常用的模型是 HSV 模型（色调、饱和度、亮度）。在图像处理中，HSV 模型有一优点，即能够独立出图像中的颜色和灰度信息，消除两者在 RGB 模型中的关联，有利于提取有用的信息。此外，还有一种与 HSV 类似的 HSI 模型。

限于篇幅，本书仅介绍 RGB 模型和 HSV 模型，对其他模型感兴趣的读者可以参考[Gonzalez，2020][OpenCV，2012]。

2.2.2.1 RGB 模型

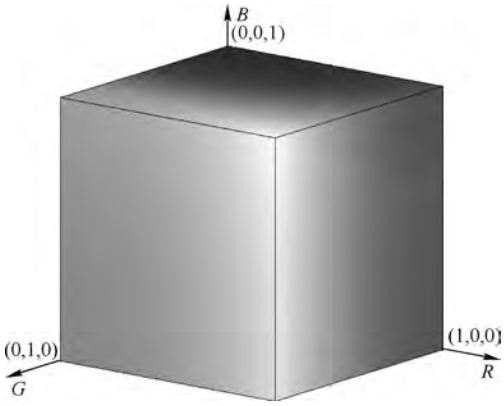
RGB 是常用的一种彩色信息表达方式，它使用红、绿、蓝三基色的亮度来定量表示颜色。该模型也称为加色混色模型，是以 RGB 三色光互相叠加来实现混色的方法，因而适合于显示器等发光体的显示。

RGB 可以看作三维直角坐标颜色系统中的一个单位正方体，3 个轴分别为 R、G、B，如图 2.6 所示（扫码查看彩图）。任何一种颜色在 RGB 颜色空间中都可以用三维空间中的一个点来表示，其中 RGB 的原色位于 3 个角上；三补色（青色、品红色和黄色）位于另外 3 个角上；原点对应黑色；当三种基色都达到最高亮度时，就表现为白色，位于离原点最远的角上；在连接黑色与白色的对角线上，是亮度相同的三基色混合而成的灰色，该线称为灰色线。为了方便，我们将立方体归一化为单位立方体，这样所有的 RGB 值都在区间[0，1]中。

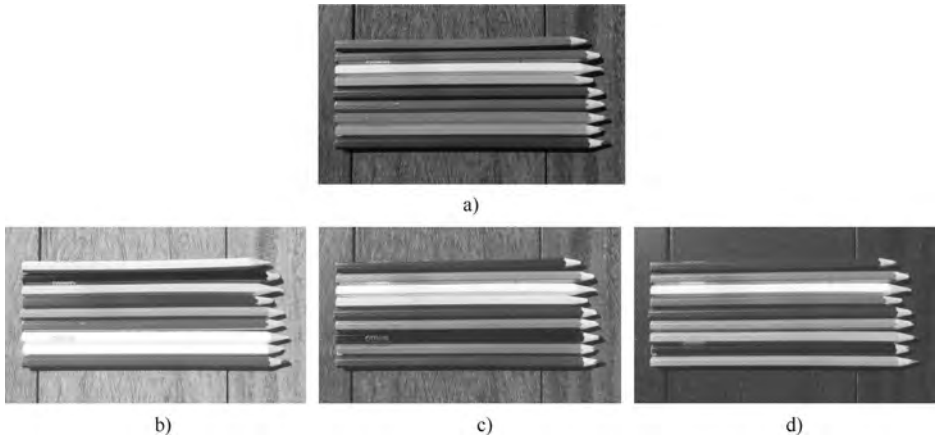
RGB 模型中的各分量又称为通道，如红色通道、绿色通道等。在图像处理中，常用的是 24 位 RGB 图像，即 3 个字节代表一个像素，每个通道占用 8 位（一个字节），灰度等级为 256。各个通道的排列方式有：*RR…GG…BB…*、*RGBRGB…*、*BB…GG…RR…*、*BGRBGR…*。

图 2.7 给出了 RGB 图像各分量的示例（扫码查看彩图）。图 2.7a 是彩铅原图，图 2.7b～图 2.7d 为红、绿、蓝各分量，在这里我们暂不考虑最接近白色的第 3 根彩铅。从图 2.7b 中可

以看出，接近红色的第 1、7、8 根彩铅以及背景亮度更高，图 2.7c 中的接近绿色的第 2、4 根彩铅亮度更高，而图 2.7d 中的含蓝色成分的第 7、9 根彩铅亮度更高。显然，在 RGB 颜色空间中可以实现对彩色图像的分割，具体示例见 7.5.2 节。



● 图 2.6　RGB 彩色立方体



● 图 2.7　RGB 各分量

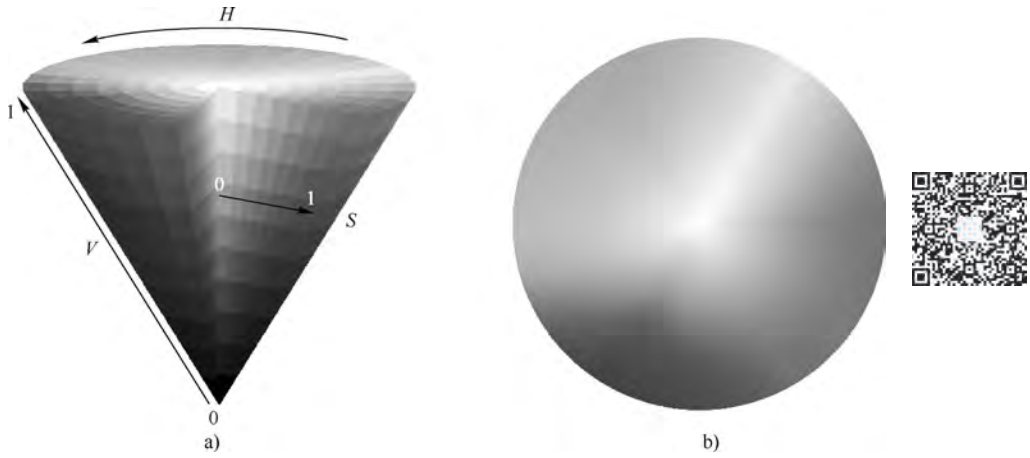
a）原图　b）红色分量　c）绿色分量　d）蓝色分量

2.2.2.2　HSV 模型

在彩色图像处理中，除了 RGB 模型外，最常用的模型就是 HSV 模型。HSV 模型用 Hue、Saturation、Value 三个参数描述颜色特性。其中 H 定义颜色的波长，称为色调；S 表示颜色的深浅程度，称为饱和度；V 表示明度或亮度。HSV 颜色模型反映了人的视觉对色彩的感觉。HSV 颜色模型常用一个倒圆锥体来描述，如图 2.8a 所示（扫码查看彩图）。圆锥体的横截面可以看作是一个极坐标系，H 用极坐标的极角表示，取值区间[0°,360°]；S 用极坐标的极轴长

度表示，取值区间[0,1]；V 用锥体中轴的高度表示，取值区间[0,1]。在 HSV 颜色模型中，色调 H 和饱和度 S 包含了颜色信息，而亮度 V 则与颜色信息无关。色调 H 反映了颜色最接近哪种光谱波长，即光的不同颜色，如红、蓝、绿等。通常假定 0°表示的颜色为红色，120°的为绿色，240°的为蓝色。0°~360°的色相覆盖了所有可见光谱的彩色，如图 2.8b 所示。饱和度 S 表征颜色的深浅程度，饱和度越高，颜色越深，如深红、深绿。由图 2.8b 可以看出，在圆的边界上的颜色饱和度最高，其饱和度值为 1；在中心的则是中性（灰色）色调，其饱和度为 0。亮度 V 是指光波作用于物体所发生的效应，其大小由物体反射系数来决定。反射系数越大，物体的亮度越大，反之越小。沿着图 2.8a 轴线自下而上亮度逐渐增大，由底部的黑渐变成顶部的白。圆锥顶部的圆周上的颜色具有最高亮度和最大饱和度。

这个模型有两个特点：V 分量与图像的颜色信息无关；H 和 S 分量与人感受颜色的方式是紧密相连的，非常直观地表达颜色的色调和鲜艳程度，方便进行颜色对比。这些特点使得 HSV 模型非常适合于借助人的视觉系统来感知彩色特性的图像处理算法，比如分割指定颜色的目标，具体示例见 7.5.1 节。



● 图 2.8　HSV 模型

a）倒圆锥体 HSV 模型　b）图 a 剖面

HSV 色彩空间颜色的划分一般为：330°≤H<30°（红色）、30°≤H<90°（黄色）、90°≤H<150°（绿色）、150°≤H<210°（青色）、210°≤H<270°（蓝色）、270°≤H<330°（品红色）。

HSV 图像一般用于图像处理，不能直接显示。

2.2.3　颜色空间变换

2.2.3.1　RGB 到灰度

在图像处理中，经常需要将 RGB 彩色图像转换成灰度图像后再进行处理。转换方法有多种：

1. 平均法

平均法是最简单的方法，即将同一个像素位置 3 个通道 R、G、B 的值进行平均

$$I=\frac{R+G+B}{3} \tag{2.12}$$

2. 最大最小平均法

取同一个像素位置的 R、G、B 中亮度最大的和最小的进行平均

$$I=\frac{\max(R,G,B)+\min(R,G,B)}{2} \tag{2.13}$$

3. 加权平均法

$$I=0.299R+0.587G+0.114B \tag{2.14}$$

上式是最常用的公式，0.299、0.587、0.114 是根据人的亮度感知系统调节出来的加权系数，是广泛使用的标准化参数。

不同的转换公式造成转换后的灰度图像有一定的差异，相对于加权平均法，最大最小平均法转换的图像边缘亮度噪声少、平滑，但对比度稍差。

图 2.9 给出了 RGB 图像转灰度图像的示例。图 2.9a～c 分别为用平均法、最大最小平均法、加权平均法将图 2.7a 转换为灰度图像。可以看出，图 2.9c 的对比度最大，视觉效果最好。

● 图 2.9　RGB 转灰度

a）平均法　b）最大最小平均法　c）加权平均法

2.2.3.2　RGB 到 HSV

首先，将 R、G、B 的值归一化到区间[0,1]，得到分量的最大值和最小值：

$$Min=\min(R,G,B) \tag{2.15}$$

$$Max=\max(R,G,B) \tag{2.16}$$

HSV 各分量为

$$V=Max \tag{2.17}$$

$$S=\begin{cases}0, & Max=0\\ (Max-Min)/Max, & Max\neq 0\end{cases} \tag{2.18}$$

$$H=\begin{cases}0, & Max=Min\\ 60\times(G-B)/(Max-Min), & Max=R\\ 120+60\times(B-R)/(Max-Min), & Max=G\\ 240+60\times(R-G)/(Max-Min), & Max=B\end{cases} \tag{2.19}$$

如果计算得到的 H 值小于 0，将该值再加上 360，得到最终的 H 值：

$$H=H+360 \tag{2.20}$$

HSV 各分量取值区间：$H\in[0,360]$，$S\in[0,1]$，$V\in[0,1]$。

2.2.3.3 HSV 到 RGB

设 HSV 的 H、S、V 值的区间分别为[0,360]、[0,1]、[0,1]，转 RGB 的计算公式如下

$$H_i=\text{integer}(H/60) \tag{2.21}$$

$$H_f=\text{fraction}(H/60) \tag{2.22}$$

$$p=V(1-S) \tag{2.23}$$

$$q=V(1-H_fS) \tag{2.24}$$

$$t=V(1-(1-H_f)S) \tag{2.25}$$

$$(R,G,B)=\begin{cases}(V,t,p), & H_i=0\\ (q,V,p), & H_i=1\\ (p,V,t), & H_i=2\\ (p,q,V), & H_i=3\\ (t,p,V), & H_i=4\\ (V,p,q), & H_i=5\end{cases} \tag{2.26}$$

式中，integer 表示取整数部分，fraction 表示取小数部分。变换后 RGB 值的区间为[0,1]。

2.3 算术和逻辑运算

2.3.1 算术运算

两幅图像 $f(x,y)$ 和 $g(x,y)$ 之间的算术运算表示为

$$\begin{cases}s(x,y)=f(x,y)+g(x,y), & d(x,y)=f(x,y)-g(x,y),\\ p(x,y)=f(x,y)\times g(x,y), & v(x,y)=f(x,y)\div g(x,y)\end{cases} \tag{2.27}$$

这些运算都是点运算，即这些运算都是 f 和 g 中对应像素对之间的加减乘除。与一般的算术运算不同之处有：两幅图像需要相同的尺寸和数据类型；防止溢出。比如对于 8 位图像来说，灰度区间为[0, 255]，因此在计算时必须考虑像素灰度值超出该区间时的取值，否则会产

生不可预期的结果。

除了两幅图像之间的算术运算外，还有一幅图像 $f(x,y)$ 与常数 C 的算术运算：

$$\begin{cases} s(x,y)=f(x,y)+C, & d(x,y)=f(x,y)-C \\ p(x,y)=f(x,y)\times C, & v(x,y)=f(x,y)\div C \end{cases} \tag{2.28}$$

这些运算都是图像 $f(x,y)$ 的每个像素与 C 之间的算术运算。同样，这些运算也需要考虑溢出的问题。除了式（2.28）的图像在运算符之前，C 在运算符之后的运算之外，还有 C 在运算符之前，图像在运算符之后的算术运算。

除了上述这些算术运算类型外，还有一种运算也归到算术运算中，那就是取绝对值运算。在进行算术运算过程中往往会产生负值，但是由于数字图像不支持负值，这时就需要对结果取绝对值。比如比较两幅图像的差异，就是两幅图像先相减再求绝对值。

算术运算虽然简单，但在图像处理中的应用却十分广泛。其中加法运算的一个重要应用就是降低图像噪声。在相机采集到的图像中，往往会存在一定的噪声。对于一幅相机采集到的图像 $g(x,y)$，可以表示为无噪声图像 $f(x,y)$ 和加性噪声 $n(x,y)$ 的组成（见 4.3 节），即

$$g(x,y)=f(x,y)+n(x,y) \tag{2.29}$$

并假设加性噪声是均值为零的高斯分布。在同一个场景拍摄 k 幅图像并取平均，平均图像的噪声 $\bar{n}(x,y)$ 的方差为单幅图像的 $1/k$[浙，1979]，即

$$\sigma^2_{\bar{n}(x,y)}=\frac{1}{k}\sigma^2_{n(x,y)} \tag{2.30}$$

从式（2.30）中我们不难发现，通过增大 k 值，即增加平均图像的数量，可减少噪声。但同时我们也发现：$\sigma_{\bar{n}}\propto 1/\sqrt{k}$，$\partial\sigma_{\bar{n}}/\partial k\propto -1/(2\sqrt{k^3})$，随着 k 值的增大，$\sigma_{\bar{n}}$ 的变化越来越小，即当图像达到一定数量时，图像平均法去噪的效果越来越差。下面通过一个例子来说明这一现象。

例 2.1　使用多图像平均法降低噪声示例一

如图 2.10 所示，图 2.10a 是一幅 8 位的灰度图像，图 2.10b 为加入了高斯噪声的图像（见 4.3.3 节）。图 2.10c 和图 2.10d 分别显示了对 10 幅、100 幅添加了噪声的图像取平均的结果。从图中可知，取 10 幅图像的平均就得到了明显的视觉改善，进一步增加图像数量，虽然图像质量得到进一步改善，但改善效果不明显。

接下来举一个多图像平均法用于实际应用的案例。

例 2.2　使用多图像平均法降低噪声示例二

如图 2.11 所示，图 2.11a 和图 2.11b 是用 X 光拍摄的同一产品内部的 10 幅图像中的两幅，需要识别其中的点阵 DM（Data Matrix）码。图像噪声信号明显，点阵模糊不清。通过对 10 幅图像平均，图像质量得到明显改善，如图 2.11c 所示。后续可通过动态阈值分割实现二维码的读取（见例 7.1）。