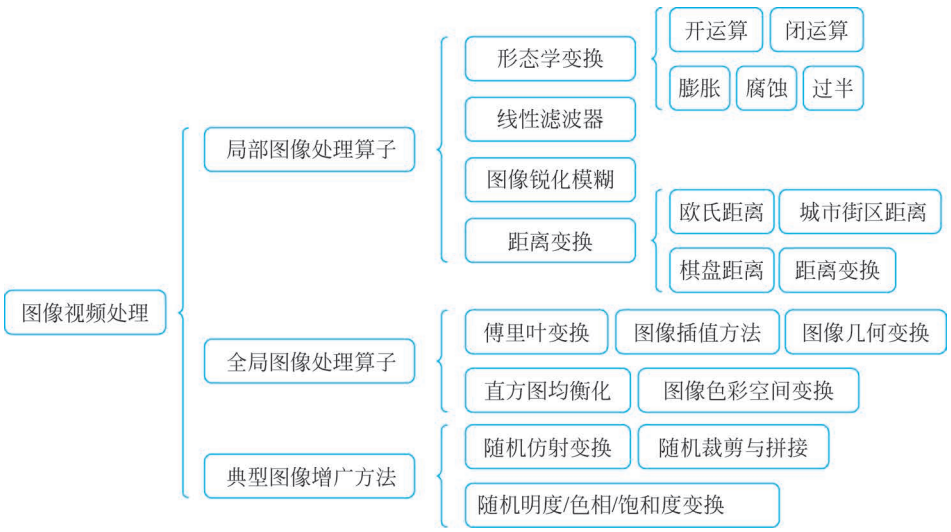
# 第 3 章　图像视频处理




图像和视频几乎遍布人类社会的每一个角落，成为我们生活中不可或缺的信息载体。从社交媒体的视频传播与播放，到海洋勘探、遥感监测等国防与民生工程，图像和视频的处理技术均有其广泛的应用和深远的影响。这些技术背后，是一系列复杂而精妙的算法，它们不仅能够进行精细的数字图像操作，还能深入分析图像内容，从而提取关键信息，优化视觉呈现，并满足多样化的任务需求。

本章将深入剖析图像与视频处理的基础知识、核心技术和应用场景。在 3.1 节中，我们将探讨局部图像处理算子，例如形态学变换、线性滤波器、图像锐化与模糊、距离变换等算子，这些算子专注于图像特定区域，用于去噪、边缘检测和形状识别。在 3.2 节中，我们将转向全局图像处理算子，利用这些算子处理整个图像，实现缩放、旋转和色彩转换等操作。在 3.3 节中，我们将介绍典型图像增广方法，这些方法在深度学习模型训练中至关重要，通过增加数据多样性，提升模型的泛化能力。通过本章的学习，希望读者能够掌握图像与视频处理的基本原理和技术，为深入研究和应用计算机视觉问题打下坚实的基础。

## 3.1　局部图像处理算子

首先了解局部图像处理算子。图像视频处理的一大重要内容在于对图像进行有针对性的操作，以便能够更好地理解和改善图像的特定局部特征。局部图像处理算子是这一领域中的关键工具，它们专注于对图像的局部区域进行操作，从而揭示出有关形状、结

构和边缘等方面的重要信息。局部图像处理算子即为利用给定像素周围像素的值决定此像素的最终输出值，可用于图像变换、图像滤波和图像的锐化与模糊等视觉处理应用。本节将重点介绍形态学变换、线性滤波器、图像模糊与锐化和距离变换四个局部图像处理方法。

### 3.1.1　形态学变换

图像形态学是数学和计算机视觉领域的一个重要分支，它主要研究图像中的形状、结构和空间关系等内容。图像形态学变换作为图像形态学领域的核心技术之一，能够对图像进行形状和结构的操作（Wilson，2000），从而为图像分析、特征提取、目标识别等任务提供了重要的工具和方法。图像形态学变换主要用于对二值图像的处理，属于非线性滤波算子的一种。而二值图像往往出现在对原始图像进行阈值化操作之后，具体公式如下：

$$\theta(f,t)=\begin{cases}1, & f\geqslant t\\ 0, & \text{其他}\end{cases} \tag{3.1}$$

以光学字符识别应用为例，需要先将扫描的灰度图像首先转化为二值图像，然后做后续处理。

形态学操作是一类依赖于特定形状模板的图像处理技术，这些模板被称为结构元素，它们具有特定的形态和尺寸，用于识别图像中感兴趣的区域。这种操作的核心在于将结构元素与图像像素逐一对比，根据对比结果调整像素值，以此改变图像的形态特征。

而结构元素可以是任何形状，既可以是简单的 3×3 方框滤波器也可以是很复杂的圆盘结构，它甚至可以是要在图像中搜寻的一个特殊的形状。令整数 $c=f\odot s$ 表示当扫描整个图像时，每个像素运算结果（处于结构元素内）中 1 的个数，整数 $s$ 表示结构元素的大小（像素的个数）。在二值形态学中，标准的操作包括以下几种。

1. 膨胀

形态学膨胀是一种图像处理技术，它通过使用结构元素来突出图像中的物体轮廓和边缘。该过程涉及将结构元素与图像像素逐一对比，若结构元素内任一像素与图像对应像素对齐，则保留该图像像素；否则，该像素会被消除。通过这种方式，膨胀操作能够有效地强化图像中物体的轮廓和边缘，使其更加清晰可见。公式表达为

$$\text{dilate}(f,s)=\theta(c,1) \tag{3.2}$$

2. 腐蚀

腐蚀操作的原理是将结构元素与图像中的像素进行匹配，如果结构元素中的所有像素都与图像中的像素匹配，则将该像素保留，否则将其删除。腐蚀操作可以用来去除图像中的噪声和细节，使图像更加清晰。公式表达为

$$\text{erode}(f,s)=\theta(c,S) \tag{3.3}$$

3. 过半

过半操作的原理是将结构元素与图像中的像素进行匹配，如果结构元素中的半数像素

与图像中的像素匹配，则将该像素保留，否则将其删除。过半操作可以用来去除图像中的锐利部分，使图像更加平滑。公式表达为

$$\mathrm{maj}(f,s)=\theta\left(c,\frac{S}{2}\right) \tag{3.4}$$

4. 开运算

开运算是一种组合操作。开运算的原理是先进行腐蚀操作，再进行膨胀操作。开运算可以用来去除图像中的噪声和细节，同时保留大的物体和边缘。公式表达为

$$\mathrm{open}(f,s)=\mathrm{dilate}(\mathrm{erode}(f,s),s) \tag{3.5}$$

5. 闭运算

闭运算是另一种组合操作。闭运算的原理是先进行膨胀操作，再进行腐蚀操作。闭运算可以用来填充图像中的小孔和裂缝，同时保留大的物体和边缘。公式表达为

$$\mathrm{close}(f,s)=\mathrm{erode}(\mathrm{dilate}(f,s),s) \tag{3.6}$$

图 3.1 展示了各个形态学变换运算的效果。可见，膨胀使物体扩张（变厚），而腐蚀使其变收缩（变细）。开运算可以消除细小的物体，而闭运算可以填充物体中的小洞，二者皆不会明显改变主要物体的大小与形状。

(a) 原始图像　(b) 膨胀　(c) 腐蚀　(d) 开运算　(e) 闭运算

图 3.1　各形态学变换运算的效果

在实际应用中，当需要用形态学变换处理阈值化图像时，有很多便利的变换运算工具。关于形态学更加详细的介绍，可以参考其他的计算机视觉和图像处理的教材（Wilson，2000），也可以参考有关这个主题的论文和书籍（Yuille，Vincent 和 Geiger，1992）。

## 3.1.2　线性滤波器

在介绍线性滤波器之前，先理解几个空间滤波图像处理的基本概念。“滤波”一词其实借用于频域处理，我们将在下一节详细讲解频率域处理的相关知识。“滤波”是一种图像处理技术，它涉及选择性地保留或抑制特定的频率成分。与基于频率的方法不同，空间滤波器（也被称作空间掩膜、核、模板或窗口）可以直接应用于图像，实现多种图像处理效果。

空间滤波器由一个局部区域（通常是一个小矩形区域）和对该区域中的图像像素执行的既定操作构成。通过这一过程，生成一个新的像素，其位置与局部区域的中心对齐，而其值则是滤波操作的结果。当滤波器的中心遍历输入图像的每个像素时，就形成了处理后的图像。如果对图像像素执行的是线性操作，那么该滤波器被称为线性空间滤波器；如果不是线性操作，则称为非线性空间滤波器。本节将主要讨论线性滤波器。

图 3.2 所示为使用 3×3 邻域的线性空间滤波的机理。对于图像中的任意一点$(x,y)$，滤波器的响应 $g(x,y)$是滤波器系数与由该滤波器包围的图像像素的乘积之和：

$$g(x,y)=w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y)+\cdots+w(0,0)f(x,y)+\cdots+w(1,1)f(x+1,y+1) \tag{3.7}$$

很明显，滤波器的中心系数 $w(0,0)$对准位置$(x,y)$的像素。

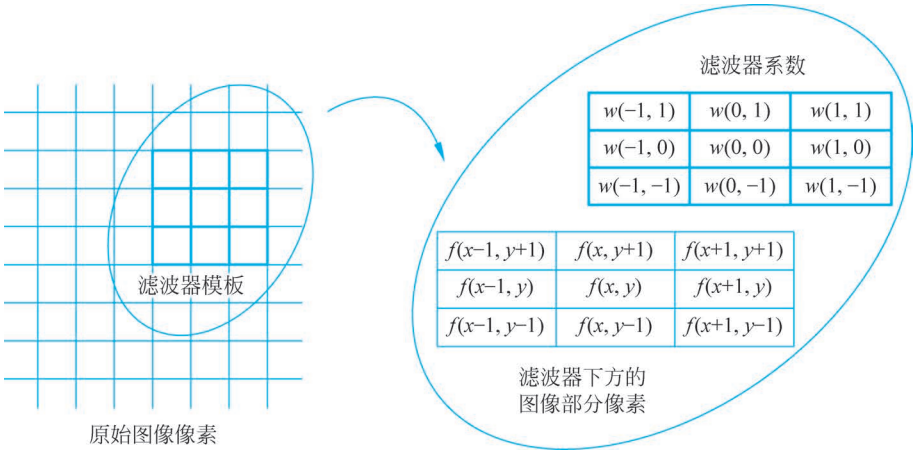



**图 3.2　使用大小为 3×3 的滤波器模板的线性空间滤波的机理**

一般地，对于一个大小为 $m\times n$ 的模板，假设 $m=2a+1$ 且 $n=2b+1$，其中 $a$，$b$ 为正整数。在后续的讨论中，我们主要关注奇数尺寸的滤波器，其最小尺寸为 3×3。一般来说，使用大小为 $m\times n$ 的滤波器对大小为 $M\times N$ 的图像进行线性空间滤波，可由下式表示：

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t) \tag{3.8}$$

式中，$x$ 和 $y$ 是可变的，以便 $w$ 中的每个像素可访问 $f$ 中的每个像素。

在执行线性空间滤波时，必须清楚地理解两个相近的概念：相关与卷积。**相关**是滤波器模板移过图像并计算每个位置乘积之和的处理。**卷积**的机理相似，但滤波器首先要旋转 180°。以二维情形为例，相关和卷积操作机理见图 3.3。

对于大小为 $m\times n$ 的滤波器，在图像的顶部和底部至少填充 $m-1$ 行 0，在左侧和右侧填充 $n-1$ 列 0，这是为了使 $w$ 中的每一个像素都可访问 $f$ 图像中的每一个像素。在图示情况下，$m$ 和 $n$ 都等于 3，如图 3.3(a)所示，因此用两行 0 填充图像的顶部和底部，用两列 0 填充图像的左侧和右侧，如图 3.3(b)所示。图 3.3(c)所示显示了执行相关操作的滤波器模板的初始位置，相关的第一个值是如图 3.3(c)所示初始位置的 $f$ 和 $w$ 的乘积之和(乘积之和为 0)，位移 $x$ 为 0。为了得到相关的第二个值，把 $w$ 向右移动一个像素位置(位移 $x=1$)并计算乘积之和(结果为 0)。当位移 $x=6$ 时，发现 $w$ 不能够再向右移动，此时把 $w$ 向下移动一行并回到最左端，而后再向右移动计算乘积之和，并用乘积之和不断替代滤波器中心位置对应的数值。由此可得到所有相关操作的结果，如图 3.3(d)所示。图 3.3(e)显示了裁剪填充边缘后的相应结果。对于卷积，需要预先旋转模板 180°，然后使用刚才描述的方法对图像作滑动乘积求和操作。图 3.3(g)显示了所有卷积操作的结果，图 3.3(f)显示了裁剪

后的卷积结果。

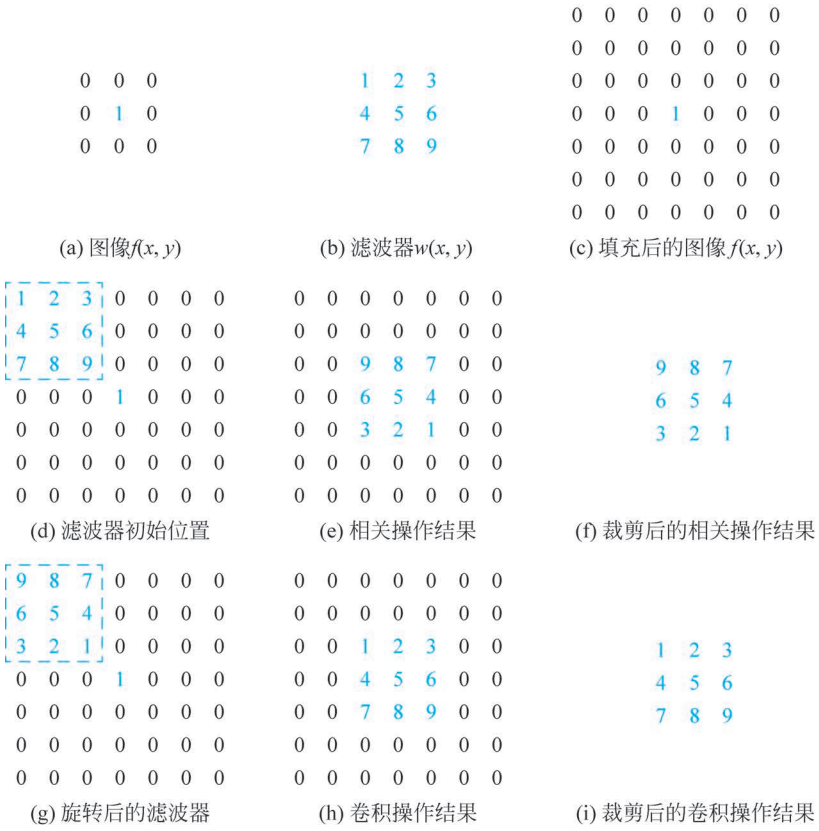



**图 3.3　相关和卷积操作机理示图**

以公式形式总结一下前面的讨论。一个大小为 $m \times n$ 的滤波器 $w(x,y)$ 与一幅图像 $f(x,y)$ 进行相关操作，可表示为 $w(x,y) \circ f(x,y)$，其计算公式可表达如下（如前所述）：

$$w(x,y) \circ f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{3.9}$$

将式(3.9)对所有位移变量 $x$ 和 $y$ 求值，以便 $w$ 的所有元素访问 $f$ 的每一个像素，其中假设 $f$ 已被适当地填充。根据前文描述，为表示方便，假设 $m$ 和 $n$ 是奇数，同时定义 $a=(m-1)/2$，$b=(n-1)/2$。类似地，$w(x,y)$ 和 $f(x,y)$ 的卷积表示为 $w(x,y) * f(x,y)$，计算公式为

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \tag{3.10}$$

式中，等式右侧的减号表示对图像 $f$ 进行翻转操作（即旋转 180°）。实际上，为简化运算过程，翻转和移位 $w$ 而不是 $f$，结果是一样的。与相关一样，将式(3.10)也对所有位移变量 $x$ 和 $y$ 求值。因此，$w$ 的每个元素访问 $f$ 中的每一个像素，同样也假设 $f$ 已被适当地填充了。

使用相关或卷积来执行线性空间滤波是优先选择的方法，原因在于二者能够通过简单的旋转与平移执行空间滤波功能。实际上，模板与图像的滤波计算通常用刚刚讨论的滑动乘积求和处理，而不必过度区分相关与卷积之间的差异。另外，在其他图像处理文献中遇到的卷积滤波器、卷积模板或卷积核这些术语，也无须过度在意，可将其理解为线性空间滤波器即可。对于实现某种特定的图像处理效果，往往可以对应设计滤波器模板的权重数值，在后续章节中，关于图像的锐化与模糊，皆是通过设计滤波器模板权重来实现的。

### 3.1.3　图像锐化与模糊

本节主要关注的是运用线性空间滤波器来实现对图像的锐化与模糊处理，锐化与模糊的区别仅仅是滤波器的权重模式不同，前者需要微分滤波器，后者需要平滑滤波器。平滑滤波器相对于微分滤波器更为简单直接。因而本节先介绍平滑滤波器，后介绍微分滤波器。

平滑滤波器主要用于模糊处理和降低噪声。模糊处理常用于预处理任务中，例如在目标提取之前去除图像中的一些琐碎细节，以及桥接直线或曲线的缝隙。平滑滤波器也可以用于降低噪声，使图像更加真实高质。

平滑型线性空间滤波器的输出基于滤波器模板覆盖区域内像素值的算术平均，有时这类滤波器也被称作均值滤波器。其核心思想是直接明了的，即用滤波器模板所覆盖区域的像素平均亮度值来替换图像中对应像素的值，这样的处理有助于减少图像亮度的剧烈变化。因此，这种滤波器常用于降低图像中的噪声。此外，通过消除与滤波器模板尺寸相比较小的像素区域中的无关细节，这种处理还能实现图像的模糊效果。

图 3.4 显示了两个 3×3 平滑滤波器模板。第一个滤波器产生模板邻域内像素平均值，可以注意到在单位滤波器处理之后，整个图像除以 9。实际上，一个 $m \times n$ 模板应有等于 $1/mn$ 的归一化常数。这种所有系数都相等的空间均值滤波器有时也被称为盒状滤波器。

图 3.4 所示的第二个模板其实更为重要一些。该模板产生所谓的加权平均图像处理效果，这一术语是指用不同的系数乘以像素，即一些像素的重要性（权重）比另一些像素的重要性更大。在图 3.4 所示的第二个模板中，处于该模板中心位置的像素所乘的值比其他任何像素所乘的值都要大，因此，在均值计算中为该像素提供更大的重要性。其他像素如同是模板中心距离的函数那样赋以成反比的权重。由于对角项离中心比离正交方向相邻的像素（参数为 2）更远，所以它的权重比与中心直接相邻的像素更小。赋予中心点最高权重，然后随着距中心点距离的增加而减小系数值，这种加权策略的目的是在平滑处理中试图降低模糊程度，也可以选择其他权重来达到相同的目的。

$\frac{1}{9}\times$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$\frac{1}{16}\times$

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

*a b*

图 3.4　平滑滤波器模板

一般地，一幅 $M \times N$ 的图像经过一个大小为 $m \times n$（$m$ 和 $n$ 是奇数）的加权均值滤波器滤波的过程可由下式给出：

$$g(x,y)=\frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)} \tag{3.11}$$

上式中的参数定义可见前文。可以这样理解这些参数，即一幅完全滤波的图像是通过对 $x=0,1,2,\cdots,M-1$ 和 $y=0,1,2,\cdots,N-1$ 依次执行上式得到的。式(3.11)中的分母简单地表示为模板的各系数之和，它是一个仅需计算一次的常数。

接下来将介绍图像锐化处理及其所需的微分滤波器。首先，锐化处理的主要目的在于突出图像灰度的过渡部分。图像锐化的用途多种多样，应用范围从电子印刷、医学成像到工业检测、军事系统的制导等。在之前讨论的内容中，图像模糊可以通过在空间域内对像素邻域进行平均处理来实现。均值处理与积分过程相似，因此逻辑上可以推断出，锐化处理可以通过空间微分来实现。实际上，本节将探讨如何通过数字微分来定义和实现锐化算子的不同方法。基本上，微分算子的响应强度与图像在该算子作用点的突变程度成正比，这意味着图像微分可以增强边缘和其他突变区域(例如噪声)，同时减少灰度变化平缓区域的影响。

以下将分别详细讨论基于一阶和二阶微分的锐化滤波器。

对于函数 $f(x)$，其一阶微分的基本定义是差值：

$$\frac{\partial f}{\partial x}=f(x+1)-f(x) \tag{3.12}$$

而二阶微分可定义为如下差分：

$$\frac{\partial^2 f}{\partial x^2}=f(x+1)+f(x-1)-2f(x) \tag{3.13}$$

可以证明，最简单的各向同性微分算子是拉普拉斯算子。一个二维图像函数 $f(x,y)$ 的拉普拉斯算子定义为

$$\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2} \tag{3.14}$$

为了以离散形式描述这一公式以便构建数字滤波器，在 $x$ 方向上有

$$\frac{\partial^2 f}{\partial x^2}=f(x+1,y)+f(x-1,y)-2f(x,y) \tag{3.15}$$

类似地，在 $y$ 方向上有

$$\frac{\partial^2 f}{\partial y^2}=f(x,y+1)+f(x,y-1)-2f(x,y) \tag{3.16}$$

所以，遵循这三个公式，两个变量的离散拉普拉斯算子是

$$\nabla^2 f(x,y)=f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y) \tag{3.17}$$

式(3.17)可以如图 3.5 所示的微分滤波器模板来实现，该图给出了以 90°为增量进行旋转的一个各向同性结果，实现机理与线性平滑滤波器一样。

拉普拉斯算子作为一种微分算子，其特点是突出图像中灰度的急剧变化，而对灰度渐变区域的影响较小。这导致它能够将浅灰色的边缘和突变点在暗色背景中凸显出来。因

此，为了恢复背景特征的同时保留拉普拉斯锐化的效果，一个简单的方法是将原始图像与拉普拉斯处理后的图像进行叠加，以获得最终的锐化效果。所以使用拉普拉斯算子对图像增强的基本方法可表示为下式：

$$g(x,y)=f(x,y)+c[\nabla^2 f(x,y)] \tag{3.18}$$

图 3.6 显示了对月球图像锐化处理的效果，左侧为月球北极稍微模糊的图像，右侧为对其用图 3.5 所示微分滤波器模板微分锐化后得到的图像，可见突出边缘突变处的效果明显。

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

图 3.5　微分滤波器模板

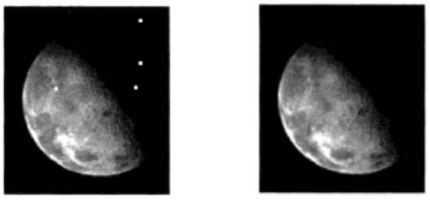


图 3.6　月球图像锐化结果

## 3.1.4　距离变换

距离变换，也称为距离函数或者斜切算法，是一种基本的图像处理方法，可以用于二值图像的配准(Huttenlocher，1993)、图像拼接和图像混合的羽化以及邻近点配准。其本质上是距离概念的一个应用方式，因而需要先对距离进行定义。

距离概念在日常生活中并不陌生，而在数学上，一般性地，可将满足以下三个条件的函数 $D$ 称作距离。

(1) 同一性：$D(p,q)\geqslant 0$，当且仅当 $p=q$，$D(p,q)=0$。

(2) 对称性：$D(p,q)=D(q,p)$。

(3) 三角不等式：$D(p,r)\leqslant D(p,q)+D(q,r)$。

而对于数字图像，有多种距离函数计算方式，包括欧氏距离、城市街区距离(曼哈顿距离)和棋盘距离等。以下以两坐标点 $a=(i,j)$ 和 $b=(k,l)$ 的距离为例，来说明各种经典距离函数的计算方式。

1. 欧氏距离

即通常所说的距离，可定义为 $D_{\mathrm{E}}(a,b)=\sqrt{(i-k)^2+(j-l)^2}$。欧氏距离在事实上比较直观，但是在实际使用的过程中，计算平方根往往比较消耗计算资源。

2. 城市街区距离

城市街区距离定义为在只允许横向和纵向运动的情况下，从起点到终点的移动步数。可用公式表达为 $D_4(a,b)=|i-k|+|j-l|$。其中符号 $D_4$ 中的 4 表示在这种定义下，像素点皆为 4 邻接的，即每个点只与它的上、下、左、右相邻的 4 个点之间的距离为 1。

3. 棋盘距离

在城市街区距离的定义基础上，如果允许横向、纵向和沿对角线方向移动，则可以定义棋盘距离，公式表达为 $D_8(a,b)=\max\{|i-k|,|j-l|\}$。类似地，符号 $D_8$ 中的 8 表示在

这种定义下，像素点皆为 8 邻接的，即每个点只与它的上、下、左、右和四个对角线方向相邻的 8 个点之间的距离为 1。

显然，以上几种经典的距离函数计算方式均满足距离的定义条件。

在理解距离概念和典型距离函数计算方式的基础上，可以进一步学习距离变换，它描述的是在二值图像中像素点与像素点为 0 的区域块之间的距离。因而像素点为 0 的区域块内像素点值为 0，邻近区域块的值不为 0 的像素点有较小的值，离区域块越远则值越大。图 3.7 所示为对二值图像的距离变换结果示例。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(a) 输入图像

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 1 | 0 | 0 |
| 0 | 0 | 1 | 2 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(b) $D_4$距离下距离变换结果

**图 3.7　对二值图像的距离变换结果**

| | |
|---|---|
| $q$ | $q$ |
| $q$ | $P$ |
| $q$ | |

掩膜1

| | | |
|---|---|---|
| | | $q$ |
| $P$ | | $q$ |
| $q$ | | $q$ |

掩膜2

**图 3.8　距离变换算法掩膜**

距离变换计算的具体实现可使用两遍扫描计算的光栅算法(Fabbri,2008)，可以快速预计算得到图像的距离变换结果。扫描计算过程的核心是利用两个小的局部掩膜遍历图像。第一遍利用图 3.8 所示掩膜 1，左上角开始，从左往右，从上往下。第二遍利用图 3.8 所示掩膜 2，右下角开始，从右往左，从下往上。

若按照城市街区距离函数对大小为 $M\times N$ 的图像作距离变换，具体算法过程可如下所示。

(1) 建立一个大小为 $M\times N$ 的数组 $F$，作如下的初始化：将值为 0 的区域块中的元素设置为 0，其余元素设置为无穷。

(2) 利用掩膜 1，左上角开始，从左往右，从上往下遍历数组，将掩膜中 $P$ 点对应的元素的值作如下更新：

$$F(P)=\min_{q\in \text{mask1}}\{F(P),D(P,q)+F(q)\} \tag{3.19}$$

(3) 利用掩膜 2，右下角开始，从右往左，从下往上遍历数组，将掩膜中 $P$ 点对应的元素的值作如下更新：

$$F(P)=\min_{q\in \text{mask2}}\{F(P),D(P,q)+F(q)\} \tag{3.20}$$

最终得到的更新后的数组即为距离变换的结果。

这个算法过程在图像边界需要做出调整，因为在边界处，掩膜不能全部覆盖图像，这时可以将掩膜中没有对应元素的位置的值当作 0 来处理。

而对于欧氏距离变换的有效计算则较为复杂。此时，仅仅在两遍扫描时保持与边界距离的最小标量值是不够的，而是应该使用正方形距离(斜边)规则，保留和比较由到边界的距离 $x$ 和 $y$ 坐标组成的向量值，搜索区域亦需要扩大以得到合理的结果。此处不再介绍算

法的更多细节。

符号距离变换是基本距离变换的一个有用的扩展，它计算了所有像素到边界像素的距离(Lavallee，1995)。生成它的最简单方法是分别计算原始二值图像和它的补图的距离变换，然后在结合之前将它们中的一个求负。因为这种距离场一般是平滑的，所以可以使用定义在四叉树或八叉树数据结构上的样条来紧凑地存储它们(Frisken，2000)。这种预先计算的符号距离变换在二维曲线和三维表面的有效配准和合并中非常有用(Curless，1996)，尤其是如果存储和插值矢量型的距离变换，即从每个像素或体素到最邻近的边界或表面元素的指针。符号距离场也是水平集演化的必要组成部分，此时它们称为“特征函数”。

## 3.2　全局图像处理算子

在上一节的学习中，我们对局部图像处理算子有了初步的认识。现在，让我们深入探讨全局图像处理算子，它们是专门设计来对整个图像进行统一处理的工具，以实现图像的缩放、旋转、色彩转换等，从而提升图像的整体视觉效果。这些算子不关注图像的局部细节，而是对整个图像执行一致的操作。

首先介绍傅里叶变换，这是一种强大的技术，可以将图像从空间域转换到频率域。在图像去噪、特征提取等领域，傅里叶变换发挥着至关重要的作用。它能够将图像中的高频信息(如边缘和纹理)与低频信息(如背景和整体色彩)区分，为图像的深入分析和处理提供了可能。紧接着探讨图像插值方法，这是一套在图像放大或缩小时保持图像质量的技术，包括最近邻插值、双线性插值和样条插值在内的多种插值方法，它们在解决图像分辨率问题时扮演着核心角色。随后介绍图像几何变换，它包括图像的平移、旋转和缩放等操作。这些变换在图像配准、目标跟踪等领域有着广泛的应用，能够模拟图像在现实世界中可能发生的各种变化。然后讨论图像色彩空间变换，这是将图像从一个色彩空间转换到另一个色彩空间的过程。不同的色彩空间各有其特点和优势，利用色彩空间的转换能够对图像的颜色进行调整，以满足不同的视觉需求。最后介绍直方图均衡化，这是一种通过重新分配图像的灰度级来提高图像对比度的方法。直方图均衡化可以使图像的直方图更加均衡，从而显著提升图像的视觉效果。

### 3.2.1　傅里叶变换

傅里叶变换的历史可以追溯到 18 世纪，当时法国数学家约瑟夫·傅里叶(Joseph Fourier)在他的著作《热的解析理论》中提出了这一概念。傅里叶观察到，任何周期函数都可以表示为不同频率的正弦和余弦函数的和。这个观察结果后来被称为傅里叶级数。傅里叶的工作在当时并没有受到广泛的认可，直到 19 世纪初，数学家和物理学家才开始认识到傅里叶工作的重要性。

19 世纪，傅里叶变换的概念被进一步发展和完善。例如，德国数学家格奥尔格·弗里德里希·伯恩哈德·黎曼(Georg Friedrich Bernhard Riemann)提出了黎曼积分，这使得傅里叶变换的定义更加精确。同时，英国数学家詹姆斯·克拉克·麦克斯韦(James Clerk Maxwell)将傅里叶变换应用于电磁学，这标志着傅里叶变换在物理学中的应用的开始。

20 世纪，随着电子计算机的发展，傅里叶变换的计算变得更加容易，这使得傅里叶变换

在许多领域得到了广泛的应用，包括信号处理、图像处理、量子力学和统计学等。特别是在信号处理和图像处理领域，傅里叶变换提供了一种有效的工具，可以将信号或图像从时间域或空间域转换到频率域，从而实现对信号或图像的有效处理。

傅里叶分析可以用于分析不同滤波器的频率特征。想象一下，你正在听一首歌曲，这首歌包含了各种不同的声音，如吉他、鼓声和人声等。傅里叶变换就像是一个魔术师，它可以告诉你这首歌中包含了多少种不同频率的声音，每种声音的强度是多少。本节将阐述如何通过傅里叶分析认识这些特征。有关傅里叶变换的全面介绍可以参考 Bracewell(1989)的论文。

傅里叶变换是一种线性积分变换，它可以将一个复杂的信号或函数分解成一系列简单的正弦波和余弦波。这种分解可以帮助我们更好地理解和分析信号。其中，正变换公式为

$$F(\omega)=\int_{-\infty}^{\infty} f(t)\mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t \tag{3.21}$$

式中，j 是虚数单位，满足 $\mathrm{j}^2=-1$；$\omega$ 是角频率；$t$ 是时间。$F(\omega)$表示函数 $f(t)$在频率 $\omega$ 处的幅度。傅里叶变换的逆变换是将傅里叶变换的结果转换回原始函数。逆变换的公式为

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\,\mathrm{d}\omega \tag{3.22}$$

式中，$f(t)$是原始函数；$F(\omega)$是傅里叶变换的结果；$t$ 是时间。

离散傅里叶变换(discrete Fourier transform，DFT)是傅里叶变换的一种特殊形式，用于处理离散数据，如计算机中的数字图片或音频文件。DFT 将一组离散的时间序列数据转换为一组离散的频率域数据。DFT 的定义如下：

$$X(k)=\sum_{n=0}^{N-1} x(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{N}} \tag{3.23}$$

式中，$x(n)$是原始的时间序列数据；$X(k)$是频率域数据；$N$ 是数据的点数；$k$ 是频率索引。DFT 的一个重要性质是其对称性，即 $X(k)=X(N-1-k)$。利用这个性质可以只计算一半的 DFT 值，利用对称性得到另一半的值，从而大大降低计算量。

快速傅里叶变换(fast Fourier transform，FFT)是一种高效计算 DFT 的方法，由库利和图基于 1965 年提出，广泛应用于信号处理、图像处理和通信等领域。考虑到 FFT 相对较为复杂，此处不再阐述算法的细节，推荐感兴趣的读者参考 Bracewell(1989)的论文。想象一下，你需要将一本厚厚的书一页一页地翻完，这是一件非常耗时的事情。但是，如果你知道书的目录，你就可以直接跳到你想要的那一页，这样就会快很多。FFT 就像是这本书的目录，它可以让你更快地找到你想要的信息。其基本思想是将 DFT 分解为多个较小的 DFT，然后递归地计算这些较小的 DFT(通常被称为蝶形运算)，它使原始的复杂度 $O(n^2)$降低到 $O(n\log n)$。

傅里叶变换具有许多重要的性质，常见性质见表 3.1。几项性质的详细描述如下。

**表 3.1　傅里叶变换的常见性质**

| 性　　质 | 时　　域 | 频　　域 |
|---|---|---|
| 线性性 | $f_1(x)+f_2(x)$ | $F_1(\omega)+F_2(\omega)$ |
| 位移性 | $f(x-x_0)$ | $F(\omega)\mathrm{e}^{-\mathrm{j}\omega_0}$ |

续表

| 性　　质 | 时　　域 | 频　　域 |
| --- | --- | --- |
| 时域反向 | $f(-x)$ | $F^*(\omega)$ |
| 时域卷积 | $f(x)*h(x)$ | $F(\omega)H(\omega)$ |
| 时域相关 | $f(x)\otimes h(x)$ | $F(\omega)H^*(\omega)$ |
| 时域乘法 | $f(x)h(x)$ | $F(\omega)*H(\omega)$ |
| 时域微分 | $f'(x)$ | $\mathrm{j}\omega F(\omega)$ |
| 尺度变换 | $f(ax)$ | $F\left(\frac{\omega}{a}\right)\Big/a$ |
| 帕塞瓦尔定理 | $\Sigma_x[f(x)]^2=\Sigma_\omega[F(\omega)]^2$ | |

1. 线性性

傅里叶变换保持了线性，如果一个函数是另外两个函数的线性组合，那么它的傅里叶变换也是这两个函数傅里叶变换的线性组合。

2. 位移性

如果将时域的函数向右平移了 $t_0$，那么在频域上，相应的函数也会有一个相位的变化。

3. 时域卷积定理

傅里叶变换的时域卷积定理意味着，如果有两个函数的傅里叶变换，那么这两个函数在时域上的卷积可以通过这两个函数的傅里叶变换相乘得到。

4. 频域卷积定理

类似时域卷积定理，如果有两个函数的傅里叶变换，那么这两个函数在频域上的卷积可以通过这两个函数的傅里叶变换相乘得到。

5. 时域微分

傅里叶变换的时域微分性质提供了一种简单快捷的计算函数微分的方法。一个函数导数的傅里叶变换可以通过该函数傅里叶变换乘以 $\mathrm{j}\omega$ 得到。

6. 帕塞瓦尔定理

傅里叶变换的帕塞瓦尔定理意味着，如果有一个函数的傅里叶变换，那么这个函数在时域上的能量可以通过这个函数的傅里叶变换的平方积分得到。

傅里叶变换的主要性质在信号处理、图像处理、通信等领域有广泛的应用。这些性质使得傅里叶变换成为一种强大的工具，可以用于分析和处理各种复杂的问题。

为了方便在学习和科研中对常见函数进行傅里叶变换，重点关注一些常用函数的傅里叶变换对，如表 3.2 所示。

表 3.2　常见的傅里叶变换对

| 时　　域 | 频　　域 |
| --- | --- |
| 1 | $\sqrt{2\pi}\cdot\delta(\omega)$ |
| $\delta(t)$ | $\frac{1}{\sqrt{2\pi}}$ |
| $e^{jat}$ | $\sqrt{2\pi}\cdot\delta(\omega-a)$ |
| $\cos(at)$ | $\sqrt{2\pi}\frac{\delta(\omega-a)+\delta(\omega+a)}{2}$ |
| $\sin(at)$ | $\sqrt{2\pi}\frac{\delta(\omega-a)-\delta(\omega+a)}{2}$ |
| $t^n$ | $j^n\ \sqrt{2\pi}\delta^{(n)}(\omega)$ |
| $\frac{1}{t}$ | $-j\sqrt{\frac{\pi}{2}}\,\mathrm{sgn}(\omega)$ |
| $\frac{1}{t^n}$ | $-j\sqrt{\frac{\pi}{2}}\frac{(-j\omega)^{n-1}}{(n-1)!}\mathrm{sgn}(\omega)$ |
| $\mathrm{sgn}(t)$ | $\sqrt{\frac{2}{\pi}}\frac{1}{j\omega}$ |
| $u(t)$ | $\sqrt{\frac{\pi}{2}}\left(\frac{1}{j\pi\omega}+\delta(\omega)\right)$ |

对于二维函数，可以将一维信号及其变换中的公式和观点直接扩展到二维函数中。如果 $f(x,y)$是一个定义在 $R^2$ 上的适当光滑的函数，那么它的二维傅里叶变换 $F(u,v)$定义为

$$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)e^{-j(ux+vy)}\,dx\,dy \tag{3.24}$$

式中，$u$ 和 $v$ 是频域中的变量，分别对应于空域中的 $x$ 和 $y$。这个公式表示的是二维傅里叶正变换，也就是将空域中的函数 $f(x,y)$转换到频域中的函数 $F(u,v)$。二维傅里叶逆变换的公式可以表示为

$$f(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(u,v)e^{j(ux+vy)}\,du\,dv \tag{3.25}$$

在实际应用中，由于计算机处理的是数字信号，因此通常使用的是离散形式的二维傅里叶变换。离散形式的二维傅里叶变换公式为

$$F(u,v)=\frac{1}{MN}\sum_{x=1}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi\frac{ux+vy}{MN}} \tag{3.26}$$

式中，$M$ 和 $N$ 分别是图像的宽度和高度。

### 3.2.2　图像插值方法

图像插值是一种在图像处理中常用的技术，主要用于在保持图像质量的同时，改变图像的大小或者分辨率。要将一幅图像插值到较高分辨率，需要选择一些插值核函数来卷积图像：

$$g(i,j)=\sum_{k,l}f(k,l)h(i-\tau k,j-\tau l) \tag{3.27}$$

式中，$\tau$ 表示上采样率。

图像插值的方法有很多种，根据选择的插值函数类型，常见的有最近邻插值、双线性插值、样条插值等。

1. 最近邻插值

最近邻插值是最简单的插值方法，它直接取最接近的像素值作为插值结果。对于图像中的一个点$(x,y)$，其最近的像素点的坐标为$(\tilde{x},\tilde{y})$，那么该点的像素值就是对应最近像素点处的像素值。这种方法计算速度快，但是插值结果可能会有明显的锯齿现象，如图 3.9 所示。

2. 双线性插值

双线性插值是一种基于像素的插值方法，它首先在水平方向和垂直方向上进行线性插值，然后再将两个方向的插值结果进行线性插值。对于图像中的一个点$(x,y)$，它最近的四个像素点的坐标分别为

$$Q_{11}:(x_1,y_1);\ Q_{12}:(x_1,y_2);\ Q_{21}:(x_2,y_1);\ Q_{22}:(x_2,y_2) \tag{3.28}$$

首先沿着 $x$ 轴方向进行线性插值：

$$f(x,y_1)=\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{x-x_1}{x_2-x_1}f(Q_{21}) \tag{3.29}$$

$$f(x,y_2)=\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{x-x_1}{x_2-x_1}f(Q_{22}) \tag{3.30}$$

然后再沿着 $y$ 轴方向进行线性插值：

$$f(x,y)=\frac{y_2-y}{y_2-y_1}f(x,y_1)+\frac{y-y_1}{y_2-y_1}f(x,y_2) \tag{3.31}$$

这种方法的计算速度较快，插值效果也比最近邻插值好，但是可能会出现轻微的模糊现象，如图 3.10 所示。

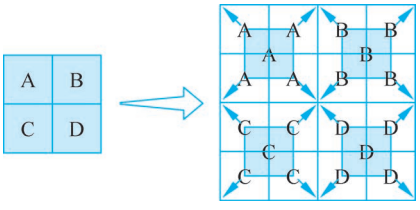



图 3.9　最近邻插值示意图

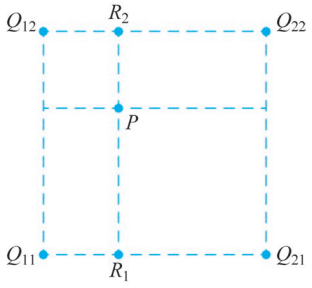



图 3.10　双线性插值示意图

3. 样条插值

样条插值是使用一种名为样条的特殊分段多项式进行插值的形式。由于样条插值可以使用低阶多项式样条实现较小的插值误差，这样就避免了使用高阶多项式所出现的龙格现象，所以样条插值应用较广泛。这种方法的计算较为复杂，但是插值效果非常好，可以有

效地减少模糊现象和锯齿现象，本书不对其细节进行详细展开，推荐读者参考教材（李庆扬，2001）。

在实际的图像处理中，需要根据具体的应用场景和需求来选择合适的插值方法。例如，如果需要在短时间内处理大量的图像，那么可能会选择计算速度快的最近邻插值或双线性插值；而如果对图像的质量要求很高，那么可能会选择插值效果好的双三次插值或样条插值。总体来说，图像插值是一种非常重要的图像处理技术，它在图像的大小调整、分辨率改变、图像缩放等方面都有着广泛的应用。随着计算机技术的发展，图像插值算法也在不断发展和改进，以提供更高质量的插值结果。

### 3.2.3 图像几何变换

图像几何变换是指在图像上进行的某种几何变形的操作，包括平移、旋转、缩放、仿射变换和透视变换等。这些变换通常可以用一个矩阵和一个偏移量来表示，其中，矩阵描述了变换的性质，偏移量描述了变换的位置。这种表示方法通过矩阵运算来快速地实现各种几何变换，同时也便于用户理解和控制变换的效果。

1. 图像平移

图像平移是指将图像沿着一定的方向移动一定的距离。设原图像为 $f(x,y)$，平移后的图像为 $g(x,y)$，平移向量为 $(t_x,t_y)$，则平移变换可以表示为

$$g(x,y)=f(x-t_x,y-t_y) \tag{3.32}$$

式(3.32)表明，对于原图像中的任意一点 $(x,y)$，它在平移后的图像中的对应点为 $(x-t_x,y-t_y)$。因此，平移变换实际上是将图像中的所有点都按照同一方向移动相同的距离。

2. 图像旋转

图像旋转是指将图像绕着一个点旋转一定的角度。设原图像为 $f(x,y)$，旋转后的图像为 $g(x,y)$，旋转中心为 $(c_x,c_y)$，旋转角度为 $\theta$，则旋转变换可以表示为

$$\begin{aligned} g(x,y)&=f(x',y') \\ x'&=(x-c_x)\cos\theta-(y-c_y)\sin\theta \\ y'&=(x-c_x)\sin\theta+(y-c_y)\cos\theta \end{aligned} \tag{3.33}$$

3. 图像缩放

图像缩放是指改变图像的大小，可以是放大也可以是缩小。设原图像为 $f(x,y)$，缩放后的图像为 $g(x,y)$，缩放因子为 $s$，则缩放变换可以表示为

$$g(x,y)=f(sx,sy) \tag{3.34}$$

4. 图像仿射变换

图像仿射变换是一种更复杂的变换，它可以同时进行平移、旋转和缩放。设原图像为 $f(x,y)$，仿射变换后的图像为 $g(x,y)$，变换矩阵为 $\boldsymbol{A}$，偏移量为 $t$，则仿射变换可以表示为

$$g(x,y)=f(\boldsymbol{A}[xy]+t) \tag{3.35}$$

5. 图像透视变换

图像透视变换是一种模拟人眼观察物体的变换，它可以改变图像的深度感。设变换矩阵为 $\boldsymbol{P}$，则透视变换可以表示为

$$g(x,y)=f(\boldsymbol{P}[xy]) \tag{3.36}$$

图像几何变换在图像处理中有广泛的应用，如图像配准、图像拼接、物体检测等。在图像配准中，需要找到两幅图像之间的几何变换关系，以便将一幅图像的信息映射到另一幅图像上。这可以通过比较两幅图像中的特征点来完成，然后利用这些特征点计算出变换矩阵和偏移量，一旦得到了这些参数，就可以通过上述公式对图像进行变换，从而实现图像配准。常见的傅里叶变换对见表 3.3。

表 3.3　常见的傅里叶变换对

| 变　　换 | 矩　　阵 | 示　意　图 |
| --- | --- | --- |
| 图像平移 | $[I\|t]_{2\times3}$ |  |
| 图像旋转 | $[R\|1]_{2\times3}$ |  |
| 图像缩放 | $[sI\|1]_{2\times3}$ |  |
| 图像仿射变换 | $[A]_{2\times3}$ |  |
| 图像透视变换 | $[\widetilde{H}]_{2\times3}$ |  |

## 3.2.4　图像色彩空间变换

在计算机图形学和图像处理中，色彩空间是用来表示颜色的一种方式。不同的色彩空间有不同的特点和应用领域。不同色彩空间的比较见图 3.11。

1. RGB 色彩空间

RGB 色彩空间是最常用的色彩空间，它基于人眼对红、绿、蓝三种颜色的感知。在 RGB 色彩空间中，每一种颜色都可以通过红、绿、蓝三个分量的组合来表示。RGB 色彩空间主要用于显示器和其他自发光设备。

2. CMYK 色彩空间

CMYK 色彩空间常用于印刷行业，它基于青色、洋红色、黄色和黑色的墨水。在 CMYK 色彩空间中，每一种颜色都可以通过青色、洋红色、黄色和黑色四个分量的组合来表示。CMYK 色彩空间主要用于印刷和其他反射光设备。

3. HSV 色彩空间

HSV 色彩空间是基于人类视觉感知的色彩空间，它由色相、饱和度和亮度三个分量组

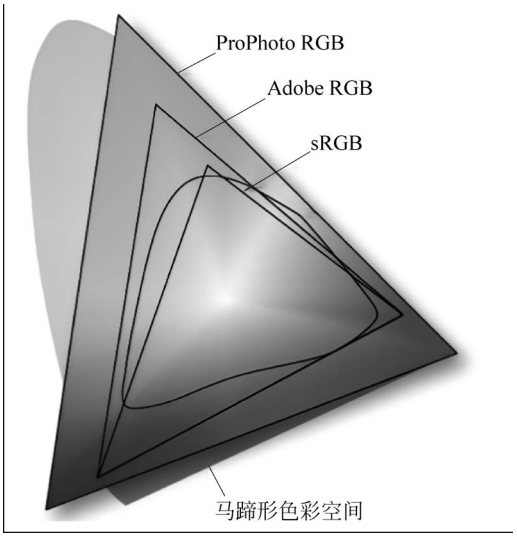



图 3.11　不同色彩空间的比较

成。色相表示颜色的种类，饱和度表示颜色的纯度，亮度表示颜色的明暗程度。HSV 色彩空间常用于图像编辑和其他需要直观调整颜色的地方。

4. YUV 色彩空间

YUV 色彩空间常用于视频压缩和编码，它分离了亮度信号和色度信号。在 YUV 色彩空间中，Y 表示亮度，U 和 V 表示色度。YUV 色彩空间可以充分利用人眼对亮度的敏感度高于对色度的特性，实现有效的压缩。

色彩空间转换是将一种色彩空间的值转换为另一种色彩空间的值的过程。这个过程可以通过一组线性或非线性的方程来实现。本节主要讲解常见的色彩空间 RGB 和 HSV 之间的转换。

HSV 空间与 RGB 空间的关系见图 3.12。

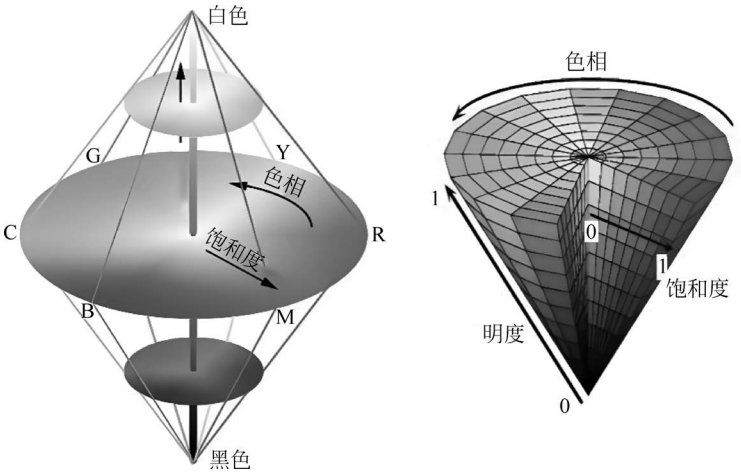



图 3.12　HSV 空间与 RGB 空间的关系

RGB 到 HSV 的转换首先需要确定颜色的最大值($V$),然后计算饱和度($S$),最后确定色相($H$)。最大值($V$)是 RGB 三个值中的最大值,它代表了颜色的明度。饱和度($S$)是最大值和最小值之差与最大值的比值。如果最大值为 0,那么饱和度也为 0,因为在这种情况下,颜色是灰色的,没有饱和度。色相($H$)的计算稍微复杂一些。首先,需要确定 RGB 三个值中最大值和最小值的位置。然后,可以根据这两个值的位置和它们的差值来计算色相。具体转换流程如下:

$$R'=\frac{R}{255},\quad G'=\frac{G}{255},\quad B'=\frac{B}{255}$$

$$C_{\max}=\max(R',G',B')$$

$$C_{\min}=\min(R',G',B')$$

$$\Delta=C_{\max}-C_{\min}$$

$$H=\begin{cases}0, & \Delta=0\\ 60^\circ\times\left(\dfrac{G'-B'}{\Delta}+0\right), & C_{\max}=R'\\ 60^\circ\times\left(\dfrac{B'-R'}{\Delta}+2\right), & C_{\max}=G'\\ 60^\circ\times\left(\dfrac{R'-G'}{\Delta}+4\right), & C_{\max}=B'\end{cases}\tag{3.37}$$

$$S=\begin{cases}0, & C_{\max}=0\\ \dfrac{\Delta}{C_{\max}}, & C_{\max}\neq 0\end{cases}\tag{3.38}$$

$$V=C_{\max}\tag{3.39}$$

式中,$R$、$G$、$B$ 分别表示红、绿、蓝的值;$H$、$S$、$V$ 分别表示 HSV 色彩空间中的色相、饱和度和明度。假设有 RGB 值为(255,0,0)的红色,可以按照上述公式进行转换:

$$H=60^\circ\times\left(\frac{(0-0)}{255-0}+0\right)=0^\circ\tag{3.40}$$

$$V=\max(1,0,0)=1\tag{3.41}$$

$$S=\frac{1-0}{1-0}=1\tag{3.42}$$

所以,(255,0,0)的红色在 HSV 色彩空间中表示为(0,1,1)。

色彩空间的转换在许多领域都有应用,如图像编辑、图像压缩、打印输出等。

1) 图像编辑

在图像编辑软件中,经常需要将图像从一种色彩空间转换到另一种色彩空间,以便进行颜色选取、调整等操作。例如,用户可能需要将图像从 RGB 色彩空间转换到 HSV 色彩空间,以便直观地调整颜色的饱和度和亮度。

2) 图像压缩

在图像和视频压缩中,经常需要将图像从 RGB 色彩空间转换到 YUV 色彩空间,以利用人眼对亮度的敏感度高于对色度的特性,实现有效的压缩。这种转换可以减少图像的数据量,从而降低存储和传输的成本。

3）打印输出

在打印输出时，需要将图像从 RGB 色彩空间转换到 CMYK 色彩空间，以便印刷设备识别和处理。这种转换可以确保图像的颜色在打印出来后与屏幕上显示的一致。

### 3.2.5 直方图均衡化

直方图均衡化是图像处理领域常用的一种技术，用于提升图像显示质量。它通过调节图像亮度和增益参数来自动找到最佳视觉效果。具体而言，直方图均衡化包含两种主要方式：一是将图像最亮和最暗的像素值分别映射至纯白和纯黑；二是找出图像像素值的均值作为中等灰度值，并拓展其范围以充分使用可显示的动态范围。

为了深入理解这些方法的成效，可以通过绘制各颜色通道及亮度值的直方图来形象化表示图像的亮度值集合。直方图作为一种统计工具，能直观展现数据分布。在图像处理中，直方图主要用于揭示像素值的分布情况，从而提供诸如最大值、最小值和平均亮度值等有用信息。然而，某些图像的直方图会存在问题，比如黑色和白色像素过多，而中间范围的像素较少。为解决这一问题，可以尝试将较暗的像素调亮，同时把较亮的像素调暗，以充分利用整个可用动态范围。实现这一目标的关键在于找到一个合适的映射函数。

直方图均衡化为此问题提供了有效解决方案。它的核心思想是找到一个映射函数 $f(I)$，使映射后的直方图尽可能均匀。这类似于从概率密度分布函数生成随机样本的过程，首先需要通过 $h(I)$ 的分布计算原图像素值的累积分布函数 $c(I)$ 如下：

$$c(I)=\frac{1}{N}\sum_{i=0}^{I}h(i)=c(I-1)+\frac{1}{N}h(I) \tag{3.43}$$

式中，$N$ 是图像中像素的个数。实际操作中，当处理 8 位像素值时，像素值的取值范围通常是 0～255。经过直方图均衡化后，可以看到转换后的直方图变得更均匀。虽然这种均匀感可能让人感觉缺乏对比度，但实际上并非如此。为了解决这个问题，可以采用局部补偿方法来改善结果，例如使用映射函数：

$$f(I)=\alpha c(I)+(1-\alpha)I \tag{3.44}$$

这个函数是累积分布函数和等变换的线性组合。直方图均衡化效果见图 3.13。

图 3.13 直方图均衡化效果

另外，对于某些图像，局部自适应直方图均衡化（local area histogram equalization，LAHE）可能效果更佳。该方法将图像分成多个子区域，并对每个子区域单独进行直方图均衡化。这样能更好地适应图像中不同区域的亮度变化。然而，这可能导致图像中出现人工区块效应，即块边界处亮度不连续。为解决这一问题，可以使用移动窗口方法，对以每个像素为中心的 $M\times M$ 大小的块重新计算直方图。尽管这种方法计算量大（每个像素需进行 $M^2$ 次运算），但能有效消除区块效应。

最后，一种更有效的改进方法是先对图像进行分块直方图均衡化，然后对各块间的转换函数进行平滑插值。这种方法称为“自适应直方图均衡化”（adaptive histogram equalization，

AHE)。对比度罗限 AHE(contrast limited AHE,CLAHE)在此基础上进一步限制对比度(限制增益),通过在各块的角点设置查找表,可以更高效地计算每个像素的结果。

## 3.3　典型图像增广方法

在机器学习和深度学习的研究领域,训练数据的规模与模型的函数空间大小构成了一对紧密相连的概念。数据规模需要和模型空间匹配,过少的数据往往会导致严重的模型过拟合问题。为了将人类的认知知识融入深度学习模型之中,研究者们设计了多种知识编码策略。图像增广技术便是其中一种,它通过在训练图像上应用各种变换来生成新的样本,有效扩展了训练数据集,有助于防止模型的过拟合现象。以含有“小猫”的图片为例,无论图片如何旋转,人类都能准确识别出其内容。因此,通过随机旋转图片来创建不同角度的副本,以此来扩充训练集,我们期望深度学习模型能够在这些扩充的数据中学习到与人类视觉认知相一致的模式。

本节将专注于探讨图像增广的各种方法,包括随机仿射变换、随机明度/色相/饱和度变换、随机裁剪与拼接等。随机仿射变换模拟了相机拍摄时角度和距离的变化,通过对图像进行随机缩放、旋转和平移,生成新的训练样本,从而增强模型的鲁棒性。随机明度/色相/饱和度变换则模拟了光照条件的变化,通过调整图像的明度来生成新的样本,提升模型对不同光照环境的适应力。而随机裁剪与拼接则通过模拟图像的局部特征,通过对图像进行裁剪和拼接生成新的样本,以提高模型对图像细节的识别能力。

每种图像增广技术都有其独特的优势,可以根据具体的应用场景和需求来选择最合适的方法。接下来,将深入讨论这些图像增广技术的原理及其实现方式,以帮助读者更全面地理解并应用这些技术。

### 3.3.1　随机仿射变换

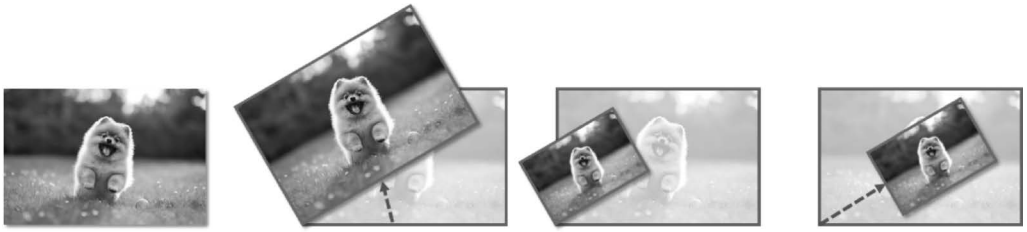
针对可学习的计算机视觉模型(例如线性分类器、神经网络等),需通过融入人工知识的方法来扩展其训练数据集,并引入人类对自然图像中不变性的理解。早期引入的一种图像增强技术是随机仿射变换。随机仿射变换的思想最早可以追溯到 20 世纪 60 年代,当时的研究者发现,人类的视觉系统能够适应图像的各种变化,如旋转、缩放和平移等,而不会影响到对图像的理解。这一发现启发了后来的研究者,他们开始尝试将这些不变性引入计算机视觉系统中,以提高模型的性能。通过模拟图像的各种变化,使得模型能够在训练过程中学习到这些不变性,从而提高它在实际应用中的性能。随机仿射变换是一种图像增强技术,主要用于模仿相机拍摄角度和距离的变化,该技术包含随机缩放、随机旋转和随机平移等多种操作,如图 3.14 所示。

**随机缩放**指的是随机调整图像尺寸。实现方式是在原有图像宽度和高度的基础上,随机选取一个比例因子进行放大或缩小。这个比例因子可以表示为

$$s = 1 + \mathrm{rand}(-\tau, \tau) \tag{3.45}$$

式中,$\mathrm{rand}(-\tau,\tau)$表示生成一个在$[-\tau,\tau]$内的随机数。这种方法可以有效地模拟图像在不同距离下的缩放效果,使得模型能够学习图像在不同尺度下的特征。

**随机旋转**则是随机改变图像的角度。具体做法是随机选择一个角度,以图像中心为轴



(a) 原图　(b) 沿原点旋转　(c) 缩放图片　(d) 平移图片

图 3.14　随机仿射变换效果

心进行旋转。这个角度可以表示为

$$\theta = \mathrm{rand}(-\rho,\rho) \tag{3.46}$$

式中，$\mathrm{rand}(-\rho,\rho)$表示生成一个在$[-\rho,\rho]$范围内的随机数。这种方法可以有效地模拟图像在不同角度下的旋转效果，使得模型能够学习图像在不同方向下的特征。

**随机平移**则涉及随机调整图像的位置。通过随机选择水平方向和垂直方向上的偏移量，移动图像中的像素点。这个偏移量可以表示为

$$(\mathrm{d}x,\mathrm{d}y) = (\mathrm{rand}(-t,t),\mathrm{rand}(-t,t)) \tag{3.47}$$

其中，这种方法可以有效地模拟图像在不同位置下的平移效果，使得模型能够学习图像在不同位置下的特征。

随机仿射变换广泛应用于图像分类和目标检测等领域。在图像分类任务中，随机仿射变换能增加图像种类，避免模型过拟合，从而提升模型的泛化能力。而在目标检测任务中，随机仿射变换能够模拟目标物体的各种姿态和位置变化，有助于提高模型的检测准确度。

随机仿射变换效果的评估可通过对比使用与未使用该技术的模型性能来进行。例如，可以计算模型在验证集上的准确率，或者使用交叉验证等方法进行评估。此外，还可以通过观察模型在测试集上的表现，以及模型在未知数据上的推广能力，来评估随机仿射变换的效果。

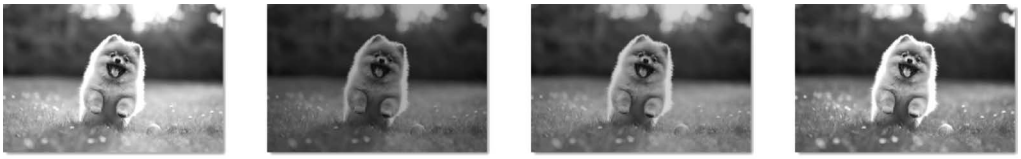
尽管随机仿射变换具有诸多优势，如增加图像多样性、预防模型过拟合和提升模型泛化能力，但它也存在一些不足之处。例如，过度应用随机仿射变换可能会改变图像的语义信息，进而影响模型的性能表现。因此，需要在训练过程中找到一个平衡点，使得模型既能学习丰富的特征，又能保持良好的性能。

然而，随机仿射变换的发展并非一帆风顺。在早期的研究中，由于计算资源的限制，随机仿射变换的应用受到了很大的限制。但随着计算机硬件的发展和深度学习技术的兴起，随机仿射变换开始在计算机视觉领域得到广泛的应用。特别是在深度学习中，随机仿射变换已经成为一种标准的数据增强技术，被广泛应用于各种计算机视觉任务中。

## 3.3.2　随机明度/色相/饱和度变换

随机明度/色相/饱和度变换是一种图像增广技术，主要用于模拟光照条件的变化。这种变换包括随机明度调整、随机色相调整和随机饱和度调整等操作，如图 3.15 所示。这些操作可以有效地模拟不同光照条件下的图像变化，使模型能够学习在各种光照条件下识别物体的能力。在实际操作过程中，一般需要先将 RGB 空间的图像变换到 HSV 空间（参考

3.2.4 节中介绍的色彩空间转换算法)。



(a) 原图　(b) 随机明度变化　(c) 随机色相变化　(d) 随机饱和度变化

图 3.15　随机明度/色相/饱和度变换效果

**随机明度调整**是指随机地改变图像的明暗程度。可以通过随机选择一个明度值,然后加到图像的每个像素上或从图像的每个像素中减去来实现。这种方法可以模拟不同光照强度下的图像变化,使模型能够学习在不同光照强度下识别物体的能力。数学上,可以表示为

$$L' = L + \eta(H - L) \tag{3.48}$$

式中,$L$ 是原始图像的明度值;$L'$是变换后的明度值;$\eta$ 是随机选择的明度调整因子;$H$ 是最大明度值。

**随机色相调整**是指随机地改变图像的色相。可以通过随机选择一个色相因子,然后乘以图像的每个像素来实现。这种方法可以模拟不同色相下的图像变化,使模型能够学习在不同色相下识别物体的能力。数学上,可以表示为

$$C' = \alpha C \tag{3.49}$$

式中,$C$ 是原始图像的色相值;$C'$是变换后的色相值;$\alpha$ 是随机选择的色相因子。

**随机饱和度调整**是指随机地改变图像的饱和度。可以通过随机选择一个饱和度因子,然后乘以图像的每个像素的颜色分量来实现。这种方法可以模拟不同饱和度下的图像变化,使模型能够学习在不同饱和度下识别物体的能力。数学上,可以表示为

$$S' = \beta S \tag{3.50}$$

式中,$S$ 是原始图像的饱和度值;$S'$是变换后的饱和度值;$\beta$ 是随机选择的饱和度调整因子。

随机明度/色相/饱和度变换在图像分类和目标检测等任务中有广泛的应用。在这些任务中,随机明度/色相/饱和度变换可以增加图像的多样性,防止模型过拟合,提高模型的泛化能力。例如,在图像分类任务中,可以使模型学习到在各种光照条件下识别物体的能力,从而提高模型在未知光照条件下的分类准确性。在目标检测任务中,可以使模型学习到在各种光照条件下识别物体的能力,从而提高模型在未知光照条件下的检测准确性。

然而,随机明度/色相/饱和度变换可能会改变图像的语义信息,导致模型的性能下降。例如,如果将一张明亮的图片变为一张昏暗的图片,那么模型可能会无法识别出图片中的物体。为了解决这些问题,研究人员提出了一些改进的随机明度/色相/饱和度变换方法。例如,有些方法只对图像的一部分区域进行随机明度/色相/饱和度变换,而不是对整个图像进行变换,这样可以保留更多的语义信息。有些方法使用更复杂的模型来模拟光照条件的变化,如基于物理的光照模型,这样可以更准确地模拟真实世界的光照条件。

### 3.3.3 随机裁剪与拼接

随机裁剪与拼接是一种图像增广技术，主要用于模拟不同的观察角度和场景。这种技术在近年来得到了广泛的关注和应用。

随机裁剪是指随机地在图像上选择一部分进行裁剪。可以通过随机选择裁剪的位置和大小来实现。这种方法可以模拟不同的观察角度和场景，使模型能够学习到在不同的观察角度和场景下识别物体的能力。数学上，可以表示为

$$I' = I(x_1, y_1, x_2, y_2) \tag{3.51}$$

式中，$I$ 是原始图像；$I'$是裁剪后的图像；$(x_1, y_1)$是裁剪的左上角坐标，$(x_2, y_2)$是裁剪的右下角坐标，具体操作效果如图 3.16 所示。

图 3.16　随机裁剪效果

随机裁剪在图像分类和目标检测等任务中有广泛的应用。在这些任务中，随机裁剪可以增加图像的多样性，防止模型过拟合，提高模型的泛化能力，从而提高模型在未知观察角度和场景下的准确性。

然而仅对图像内部像素进行裁剪只能关联图片内部像素之间的关系，为了进一步考虑图像间像素的关联性，研究人员提出了一些随机拼接方法。近年来，Mixup(Zhang H，2017)和 CutMix(Yun S，2019)等技术被提出，进一步提高了图像增广的效果。Mixup 通过线性插值的方式，将两个图像及其标签进行混合，生成新的图像和标签。这种方法可以平滑模型的决策边界，防止过拟合，提高模型的泛化能力。CutMix 则是在 Mixup 的基础上，通过裁剪和粘贴的方式，将两个图像的部分区域进行混合，生成新的图像和标签。这种方法可以保留更多的图像特征，提高模型的性能。随机拼接效果见图 3.17。

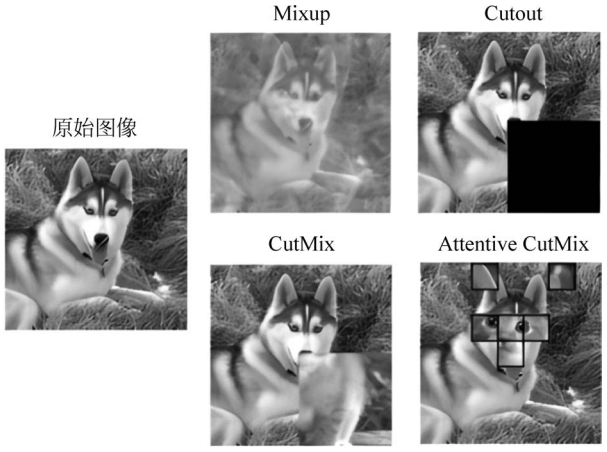



图 3.17　随机拼接效果

## 思考题

1. 二维离散傅里叶变换是数字信号处理中分析图像频率特性的重要工具。请证明二维离散傅里叶变换的时域卷积定理。

2. 直方图均衡化是一种用于改善数字图像对比度的技术。已知某幅数字图像已经通过直方图均衡化技术进行了增强，请证明再使用一次直方图均衡化时图像不变。