# 数字图像处理实践
# ——基于 Python

主　编　郭　锐　杨成义
副主编　黎江枫　陈丽虹

清華大學出版社
北　京

## 内 容 简 介

本教材秉承理论结合实践的教学理念，旨在通过系统且深入的内容，帮助读者快速掌握数字图像处理的核心技术，为未来的工作、科研或深造奠定坚实基础。本教材共分 10 章，主要内容涵盖了数字图像的基本概念、数字图像处理的重要意义以及当前常用的处理工具，并介绍了 Python 在数字图像处理中的应用、图像的基本运算、图像变换领域，以及图像增强技术、图像复原技术、图像压缩编码技术、图像分割技术、彩色图像处理技术、图像表示与描述技术等。本教材以实战为导向，每章均配有大量可执行的代码与实例演示，确保读者能够边学边做，快速掌握数字图像处理的精髓。

本教材内容全面，层次分明，不仅适合作为应用型本科生的教材使用，也适合作为数字图像处理领域其他初学者的自学参考书。

**本书封面贴有清华大学出版社防伪标签，无标签者不得销售。**

**版权所有，侵权必究。举报：010-62782989，beiqinquan@tup.tsinghua.edu.cn。**

**图书在版编目(CIP)数据**

数字图像处理实践 ：基于 Python / 郭锐, 杨成义主编. -- 北京 ：清华大学出版社, 2025. 6. -- ISBN 978-7-302-69174-7

Ⅰ. TN911.73

中国国家版本馆 CIP 数据核字第 2025GH6517 号

**责任编辑：**王　定

**封面设计：**周晓亮

**版式设计：**思创景点

**责任校对：**马遥遥

**责任印制：**宋　林

**出版发行：**清华大学出版社

网　　址：https://www.tup.com.cn，https://www.wqxuetang.com

地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084

社 总 机：010-83470000　　邮　　购：010-62786544

投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

**印 装 者：**三河市君旺印务有限公司

**经　　销：**全国新华书店

**开　　本：**185mm×260mm　　**印　　张：**21.5　　**字　　数：**507 千字

**版　　次：**2025 年 6 月第 1 版　　**印　　次：**2025 年 6 月第 1 次印刷

**定　　价：**79.80 元

产品编号：110901-01

# 前言

PREFACE

党的二十大报告指出：“加快发展数字经济，促进数字经济和实体经济深度融合，打造具有国际竞争力的数字产业集群。”在当今数字化时代，图像数据的规模呈爆炸式增长，其应用场景也日益广泛。从医疗领域的疾病诊断、远程医疗，到工业生产中的质量检测、自动化控制；从智能安防的监控识别、智能交通的车辆检测，到文化娱乐的影视特效、虚拟现实；从农业生产的作物监测、病虫害防治，到电子商务的商品展示、图像搜索，数字图像处理技术无处不在。它为我们提供了更高效、更准确、更智能的信息获取、分析和处理手段，推动着各个领域的创新与进步，已然成为现代社会不可或缺的一部分。

本书秉承理论结合实践的教学理念，旨在通过系统且深入的内容，帮助读者快速掌握数字图像处理的核心技术，为未来的工作、科研或深造奠定坚实基础。本书共分为 10 章，第 1 章绪论，概述了数字图像的基本概念、数字图像处理的重要意义以及数字图像常用的处理工具。通过这章学习，读者将对数字图像有一个初步认识，并了解数字图像处理在各领域中的广泛应用，为后续章节的学习打下基础。第 2 章介绍 Python 在数字图像处理中的应用，详细介绍 Pillow 库、NumPy 库、scikit-image 库和 OpenCV 库这四大常用库的安装及基本功能，为后续章节的实战操作做好准备。第 3~6 章构成数字图像处理的核心技术体系：第 3 章详细讲解图像的基本运算，包括点运算、算术运算和几何运算，这些都是图像处理中最基础、最重要的操作。第 4 章深入探讨图像变换，重点介绍傅里叶变换及其频谱分析应用，让读者了解图像在频率域中的表示和处理方法。第 5 章探讨图像增强技术，包括直方图处理、空间域与频率域滤波等，这些方法能够改善图像的质量，使其更加清晰、易于分析。第 6 章研究图像复原技术，涉及退化模型、噪声模型及滤波复原等原理与方法，帮助读者掌握恢复受损图像、还原图像真实面貌的方法。第 7 章介绍图像压缩编码技术，讲解经典压缩方法如霍夫曼编码、算术编码等，并通过代码实现让读者掌握压缩原理。第 8 章专注于图像分割技术，详细介绍阈值分割、边缘检测等多种方法，并通过实例代码展示图像分割的实现方法。第 9 章探讨彩色图像处理技术，介绍彩色图像的基础、伪彩色与全彩色图像处理技术，并通过编程实例加深读者对彩色图像处理的理解。第 10 章关注图像描述技术，包括颜色、纹理、边界和区域描述等关键技术，为读者进一步掌握图像的分析与识别奠定基础。

本书以实战为导向，每章均配有大量可执行的代码与实例演示，确保读者能够边学边做，熟练运用 Python 进行数字图像处理，掌握数字图像处理的核心技术和方法。本书不仅适合作为应用型本科生的教材使用，也适合作为数字图像处理领域初学者的自学参考书。

本书由广东理工学院信息技术学院的教师共同编写。各章编写分工如下：第 1、第 3、第 8 章由郭锐编写，第 4、第 5、第 6 章由杨成义编写，第 2、第 7 章由黎江枫编写，第 9、第 10 章由陈丽虹编写。在此，我们向每一位参与编写的教师表示衷心的感谢，他们凭借丰富的教学经验和专业知识，为本书的质量提供了坚实的保障。同时感谢学院领导和同事们的支持与帮助，他们在教学和科研工作中给予了我们很多启发和鼓励。

尽管我们在编写过程中竭尽全力，但由于数字图像处理技术的不断发展和我们自身有限的水平，书中难免存在不足之处。我们真诚地希望读者能够批评指正，提出宝贵的意见和建议。我们将虚心接受并积极改进，以便在后续的版本中不断完善本书，为读者提供更优质的学习资源。

最后，希望本书能够帮助读者在数字图像处理领域迈出坚实的步伐，为未来的学习和研究打下良好的基础，共同推动数字图像处理技术的发展与应用。

本书提供教学大纲、教学课件、电子教案、习题参考答案和模拟试卷，读者可扫下列二维码进行下载。

教学大纲　教学课件　电子教案　习题参考答案　模拟试卷

编　者

2025 年 1 月

# 目录

CONTENTS


第 1 章 绪论 …… 1
1.1 认识数字图像 …… 2
1.1.1 数字图像的起源 …… 2
1.1.2 数字图像基本概念 …… 2
1.1.3 数字图像的分类 …… 3
1.1.4 数字图像的采样与量化 …… 5
1.2 认识数字图像处理 …… 6
1.2.1 数字图像处理的含义 …… 7
1.2.2 数字图像处理的应用领域 …… 7
1.2.3 常见的数字图像处理技术 …… 8
1.2.4 数字图像处理的基本步骤 …… 9
1.2.5 数字图像文件格式 …… 10
1.3 认识数字图像处理工具 …… 11
1.3.1 MATLAB …… 11
1.3.2 Visual C++ …… 13
1.3.3 Python …… 16
1.4 思考练习 …… 18

第 2 章 Python 与数字图像处理 …… 19
2.1 Python 环境部署 …… 20
2.1.1 Python 安装 …… 20
2.1.2 Pychram 安装 …… 20
2.2 了解 Pillow 库 …… 22
2.2.1 Pillow 库的基本信息 …… 22
2.2.2 Pillow 库的安装方法 …… 22
2.2.3 Pillow 库的主要作用 …… 23
2.3 了解 NumPy 库 …… 25
2.3.1 NumPy 库概述 …… 25
2.3.2 NumPy 库的安装方法 …… 26
2.3.3 NumPy 库的应用 …… 26
2.4 了解 scikit-image 库 …… 30
2.4.1 scikit-image 库的基础概念 …… 30
2.4.2 scikit-image 库的安装方法 …… 30
2.4.3 scikit-image 库的作用 …… 30
2.5 熟悉 OpenCV 库 …… 34
2.5.1 什么是 OpenCV …… 34
2.5.2 OpenCV 的历史与发展 …… 35
2.5.3 OpenCV 的应用领域 …… 35
2.5.4 OpenCV 安装 …… 35
2.5.5 OpenCV 基础操作 …… 36
2.6 python 其他的有关图像处理库 …… 42
2.6.1 Matplotlib 库 …… 42
2.6.2 PyTorch 库 …… 45
2.6.3 TensorFlow 库 …… 46
2.7 思考练习 …… 47

第 3 章 图像基本运算 …… 49
3.1 点运算 …… 50
3.1.1 线性点运算 …… 50
3.1.2 非线性点运算 …… 55
3.2 算术运算 …… 64
3.2.1 加法运算 …… 65
3.2.2 减法运算 …… 68
3.2.3 乘法运算 …… 70
3.2.4 除法运算 …… 71
3.3 几何运算 …… 73
3.3.1 仿射变换 …… 73
3.3.2 平移变换 …… 75
3.3.3 旋转变换 …… 77

3.3.4 镜像变换…… 78
3.3.5 缩放变换…… 80
3.3.6 透视变换…… 83
3.4 思考练习 …… 85

第 4 章 图像变换 ……86
4.1 图像变换概述 …… 87
4.1.1 图像变换的主要作用…… 87
4.1.2 图像变换的方法…… 87
4.1.3 图像变换的步骤…… 88
4.1.4 实例及代码实现…… 89
4.2 离散傅里叶变换 …… 90
4.2.1 离散傅里叶变换的定义…… 90
4.2.2 离散傅里叶变换的特点…… 90
4.2.3 实例及代码实现…… 91
4.2.4 应用领域…… 95
4.3 图像傅里叶变换频谱分析 …… 96
4.3.1 基本原理…… 96
4.3.2 常用的频谱分析方法…… 96
4.3.3 实例及代码实现…… 97
4.4 离散余弦变换 …… 103
4.4.1 定义和原理…… 103
4.4.2 特点和优势…… 104
4.4.3 实例及代码实现…… 104
4.5 思考练习 …… 107

第 5 章 图像增强 ……108
5.1 图像增强概述 …… 109
5.1.1 图像增强的主要作用…… 109
5.1.2 图像增强的方法…… 111
5.1.3 图像增强的步骤…… 112
5.2 直方图均衡 …… 113
5.2.1 直方图均衡的原理…… 113
5.2.2 直方图均衡的特点…… 114
5.2.3 实例及代码实现…… 115
5.3 空间域滤波增强 …… 118
5.3.1 平滑滤波…… 119
5.3.2 锐化滤波…… 126
5.4 频率域平滑滤波器 …… 133
5.4.1 概述…… 133
5.4.2 理想低通滤波器…… 133
5.4.3 Butterworth 低通滤波器 … 136
5.4.4 高斯低通滤波器…… 138
5.5 频率域锐化滤波器 …… 140
5.5.1 概述…… 141
5.5.2 理想高通滤波器…… 141
5.5.3 Butterworth 高通滤波器…… 143
5.5.4 高斯高通滤波器 …… 144
5.6 思考练习 …… 146

第 6 章 图像复原 ……148
6.1 图像复原及退化模型 …… 149
6.1.1 图像复原及相关概念…… 149
6.1.2 图像复原的方法和步骤…… 149
6.1.3 退化模型的表示…… 150
6.1.4 常见退化模型及形式…… 151
6.2 图像噪声 …… 152
6.2.1 图像噪声的分类…… 152
6.2.2 图像噪声模型应用领域…… 153
6.2.3 实例及代码实现…… 153
6.3 空间域滤波复原 …… 158
6.3.1 基本原理…… 158
6.3.2 空间域滤波复原的基本步骤…… 158
6.3.3 空间域滤波复原的分类…… 159
6.4 频率域滤波复原 …… 177
6.4.1 主要原理…… 177
6.4.2 滤波方法及实现步骤…… 177
6.4.3 频率域滤波的特点…… 177
6.4.4 带通滤波器…… 178
6.4.5 带阻滤波器…… 180
6.4.6 陷波滤波器…… 184
6.5 估计退化函数 …… 185
6.5.1 常见估计退化函数的方法…… 186
6.5.2 考虑因素…… 186
6.6 逆滤波和维纳滤波 …… 189

6.6.1 逆滤波 …… 190
6.6.2 维纳滤波 …… 194
6.7 思考练习 …… 197

第 7 章 图像压缩编码 …… 199
7.1 数字图像压缩编码基础 …… 200
7.1.1 图像压缩的定义和分类 …… 200
7.1.2 冗余与压缩效率 …… 203
7.1.3 信源编码与信道编码的区别与联系 …… 208
7.2 变长编码 …… 211
7.2.1 霍夫曼编码 …… 211
7.2.2 游程编码 …… 219
7.2.3 字典编码 …… 222
7.2.4 LZW 算法 …… 223
7.3 算术编码 …… 226
7.3.1 算术编码原理 …… 226
7.3.2 算术编码算法实现 …… 227
7.3.3 算术编码的优势和挑战 …… 230
7.4 变换编码 …… 231
7.4.1 DCT 的应用 …… 232
7.4.2 小波变换 …… 235
7.4.3 其他变换方法 …… 236
7.5 思考练习 …… 237

第 8 章 图像分割 …… 239
8.1 阈值分割 …… 240
8.1.1 基本原理 …… 240
8.1.2 阈值的选择 …… 240
8.1.3 阈值分割的类型 …… 240
8.1.4 代码实现 …… 241
8.1.5 应用领域 …… 243
8.2 边缘分割 …… 243
8.2.1 基本原理 …… 243
8.2.2 常用的边缘检测算法 …… 243
8.2.3 代码实现 …… 248
8.2.4 形态学运算函数 …… 252
8.2.5 边缘分割的优缺点分析 …… 254
8.2.6 边缘分割的应用场景 …… 254
8.3 区域分割 …… 254
8.3.1 区域生长法 …… 255
8.3.2 区域分裂与合并法 …… 260
8.4 聚类分割 …… 264
8.4.1 基本原理 …… 264
8.4.2 常用聚类算法在图像分割中的应用 …… 264
8.5 思考练习 …… 268

第 9 章 彩色图像处理 …… 269
9.1 彩色图像基础 …… 270
9.2 彩色模型 …… 271
9.2.1 彩色模型分类 …… 272
9.2.2 RGB 模型 …… 272
9.2.3 CMY 模型与 CMYK 模型 …… 275
9.2.4 HSI 模型和 HSV 模型 …… 276
9.2.5 CIELab 模型 …… 279
9.2.6 YCbCr 模型和 YUV 模型 …… 279
9.3 伪彩色处理 …… 284
9.3.1 伪彩色图像处理基础 …… 284
9.3.2 灰度级到彩色变换 …… 285
9.4 全彩色图像处理 …… 288
9.4.1 全彩色图像处理基础 …… 288
9.4.2 彩色图像增强 …… 288
9.4.3 彩色图像平滑 …… 297
9.4.4 彩色图像锐化 …… 300
9.5 思考练习 …… 305

第 10 章 图像表示与描述 …… 307
10.1 图像表示描述的作用及应用场景 …… 308
10.1.1 图像表示与描述的作用 …… 308
10.1.2 图像表示描述的应用场景 …… 308
10.2 颜色描述 …… 309
10.2.1 颜色矩 …… 309

10.2.2 颜色直方图 …… 311
10.2.3 颜色集 …… 313
10.3 纹理描述 …… 315
10.3.1 矩分析法 …… 315
10.3.2 灰度差分统计 …… 316
10.3.3 灰度共生矩阵 …… 318
10.3.4 局部二值模式 …… 318
10.4 边界描述 …… 322
10.4.1 链码描述 …… 322
10.4.2 傅里叶描述 …… 323
10.5 区域描述 …… 326
10.5.1 几何特征 …… 326
10.5.2 不变矩 …… 330
10.6 思考练习 …… 332
参考文献 …… 333

# 第1章

# 绪　论

随着互联网和大数据技术的不断发展，其应用场景不断拓展，社会对数字图像数据的需求与依赖日益增长，在航天、遥感、工业检测、农业生产、智能安防、医疗健康、零售和娱乐等多个领域中，图像数据均占据了核心地位，而针对这些图像所运用的处理技术，无疑为这些领域提供了宝贵的信息支撑和深入分析的基础。本章将介绍与数字图像相关的基本概念、图像处理的基本步骤及方法，并介绍数字图像处理的工具。



## 本章学习目标

◎ 理解数字图像的构成、表示和存储方式，以及它们在不同领域的应用背景和重要性。

◎ 掌握数字图像处理的基本步骤，包括图像的读取、预处理、核心处理、保存、显示结果等关键步骤。

◎ 了解数字图像数据在航天、遥感、医学成像、工业检测、农业生产、智能安防、零售和娱乐等领域的核心作用和具体应用案例。

◎ 熟悉当前流行的数字图像处理工具的特点和适用场景，以及运用这些工具进行图像处理的基本代码结构。



## 素质要点

◎ 培养跨学科整合能力:在学习了数字图像相关的基本概念、应用领域和数字图像处理工具后，会发现数字图像处理与计算机科学、数学、物理学等多个学科是息息相关的，其应用也与这些学科领域紧密相连。高层次人才只有形成跨学科的知识体系，才能使技术在社会发展中发挥更大功效。

◎ 培养创新思维：积极运用创新思维，探索数字图像处理技术在新兴领域(如人工智能、大数据、虚拟现实等)的应用潜力，有意识地为解决复杂问题提供新颖的解决方案。这有利于学生提高个人能力和竞争力，为未来的发展奠定坚实基础。

# 1.1 认识数字图像

数字图像作为现代信息处理的重要部分，其基本的概念及涉及的原理和技术是理解后续图像处理技术的基础。本节将对数字图像的起源、基本概念、分类，以及数据图像的采样与量化过程进行全面介绍，以使读者清晰地认识到数字图像在现代社会中的重要地位，并掌握数字图像的基本属性和特点。

## 1.1.1 数字图像的起源

广义上讲，所有具有视觉效果的画面都可以称为图像。普通人接触到的原始的和自然的图像，是通过接受客观世界物体反射光或投射光在其视觉系统的成像而获取的，这些视觉信息约占其各个感官所能获取信息的 83%。

除了绘画的方式，人们一直在寻找能真实记录人物和场景的方式。公元前 5 世纪，古希腊哲学家亚里士多德描述了通过小孔在暗箱中成倒像的照相原理。据记载，世界上第一张记录实景的相片诞生于 19 世纪 20 年代，此后，随着科学技术和摄像技术的不断发展，照相机、录像机等能记录图像的机器不断被发明和升级。1975 年，柯达公司开发出了世界上第一台数码相机。

然而数字图像的出现却比数码相机早得多，数字图像最早诞生于电报业，随着 19 世纪电力时代的到来，电报、电缆被发明和使用，人们开始尝试通过采样、量化等方式将图像以数字的形式表达和传输，据记载，第一张通过海底电缆传输的数字新闻图像是 Bartlane 系统于 1921 年横跨大西洋传送的。此后，随着科技的发展和各领域的需求，为了提高图像的传输效率和图像质量，人们发明了一系列图像与数字的转译方法和设备。

## 1.1.2 数字图像基本概念

人们常说的数字图像一般指以有限的数值表示各像素值的二维图像，即一幅数字图像可以对应一个描述各坐标位置像素值的二维数组。

$$I_{\mathrm{n\times m}}=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,m-1) \\ f(1,0) & f(1,1) & \cdots & f(1,m-1) \\ & & \vdots & \\ f(n-1,0) & f(n-1,1) & \cdots & f(n-1,m-1) \end{bmatrix} \tag{1.1}$$

式(1.1)中，$I$为一幅二维图像对应的二维数组，将其以指定大小的单元进行纵向和横向划分，得到 $n$ 行 $m$ 列个单元，每个单元即为一个像素(pixel)，每个像素对应一个坐标位置$(i, j)$，该位置的数值称为像素值，可以表示图中该位置的亮暗程度，即灰度(gray)，用 $f(i, j)$表示。由于每个二维图像各坐标对应的灰度为有限的离散值，因此可以用以坐标$(x, y)$为自变量的二维离散函数 $f(x, y)$表示此图像。

### 1.1.3　数字图像的分类

上节中用 $f(x, y)$表示的数字图像为一般意义上的数字图像，即静态图像。若要表示动态的数字图像，则需要增加一个时间变量 $t$，即 $f(x, y, t)$，本教材主要介绍对静态图像的处理方法。

对静态图像来说，通常按图像能表现的颜色范围可以将图像分为二值图像(Binary Image)、灰度图像(Gray Scale Image)和 RGB 彩色图像(RGB Image)。

对一幅图像来说，每个像素的灰度值都有一个固定的取值范围，这个范围即每个像素能表达的灰度范围。

(1) 当图像为二值图像时，图像只能区分为两种颜色，即各像素的灰度取值范围仅有两个值。常见的仅有黑白两色的图像即为二值图像；0-1 二值图像的图像灰度为 0 或 1，则在计算机中存储时可以用 1 个 bit(比特)位表示一个像素的颜色。

(2) 当图像为灰度图像时，相当于增加了图像的颜色区分级别，即从二值图像的两种颜色扩展到多种颜色，这些颜色通常显示为从最暗的黑色到最亮的白色之间的渐变色，即不同强度的灰色，每种灰色称为一个灰度级，为便于表示及存储，一幅图像的灰度级一般表示为 2k，图像的灰度值取值范围为[0, 2k-1]。当 k=8 时，其能表达的颜色为 256 种(0~ 255)不同的灰度，即我们常见的灰阶丰富的 256 色灰度图，存储这种图的一个像素就需要 8 个 bit 位，即一个字节。

(3) 当图像为 RGB 图像时，则图像可以表示的颜色更加丰富。当计算机显示彩色图像时，通常采用以红色(R)、绿色(G)、蓝色(B)三原色按不同比例组成各种颜色的 RGB 模型。此类图像各像素点的灰度以红色分量、绿色分量及蓝色分量组成的三元组(R，G，B)表示，每个分量的取值范围也是 0~255，因此用此模型表示的颜色种类有 $2^8 \times 2^8 \times 2^8 = 2^{24}$ 种。存储 RGB 图像时，则至少需要 24 个 bit 位，即 3 个字节，每个颜色都可以以其十六进制代码表示。图 1-1 所示为电脑的颜色面板，在 RGB 模式下，每个颜色可以用三通道的分量值来指定，其中黑色(0，0，0)的十六进制代码为 0x000000。此类图像又称为 24 位真彩色图像。




图 1-1

图 1-1　电脑中设置颜色的面板组成

常见的颜色名、对应代码、RGB 值见表 1-1，其中左列颜色与右列颜色属于补色关系。

表 1-1

表 1-1　常见颜色的 RGB 取值表

| 颜色 | ( R,G,B ) | 显示及代码 | 显示及代码 | ( R,G,B ) | 颜色 |
|---|---|---|---|---|---|
| 黑色 | ( 0, 0, 0) | 0x000000 | 0xFFFFFF | (255,255,255) | 白色 |
| 红色 | (255, 0, 0) | 0xFF0000 | 0x00FFFF | ( 0,255,255) | 青色 |
| 绿色 | ( 0,255, 0) | 0x00FF00 | 0xFF00FF | (255, 0, 255) | 洋红 |
| 蓝色 | ( 0, 0,255) | 0x0000FF | 0xFFFF00 | (255,255, 0) | 黄色 |

根据不同的需要，彩色图像可以被分解为不同通道的图像分别处理，更常见的处理方法是将其转换为灰度图进行处理，用 Gray 表示灰度值，R，G，B 分别表示红、绿、蓝分量值，则 RGB 图到灰度图的转换公式如式(1.2)所示：

$$\mathrm{Gray} = \omega 1 \times \mathrm{R} + \omega 2 \times \mathrm{G} + \omega 3 \times \mathrm{B} \tag{1.2}$$

其中，$\omega 1$，$\omega 2$，$\omega 3$ 为三颜色通道分量值的权值，其取值可以根据需要进行人为调整，若无特殊需求，则一般默认取值为 $\omega 1 = 0.299$，$\omega 2 = 0.587$，$\omega 3 = 0.114$，这是根据人眼对红、绿、蓝三种颜色的感知程度取值的。由于默认情况下绿色分量的权值较高，因此用此权值转换得到的灰度图在视觉上与绿色通道的图更接近。

图 1-2 所示为对一幅原图进行三通道分离并分别显示的效果图，图 1-3 为将原图 1-2(a)按公式(1.2)转换成的灰度图。

图 1-2



(a) 原图　(b) 红色通道图　(c) 绿色通道图　(d) 蓝色通道图

图 1-2　彩色原图与 RGB 三通道图

为了减少数字图像的存储空间，一种索引图像(Indexed Image)被提出，用到了调色板(palatte)或称为颜色查找表(Look Up Table，LUT)技术，此类图像用颜色查找表存储当前图像用到的颜色对应的真实 RGB 值，而在表示图像的数组或矩阵中只用简单的序号指明各像素对应的颜色在查找表中的序号，从而大大缩减了图像数组的存储空间，很多图像采用索引方式存储。



图 1-3　图 1-2(a)转换得到的灰度图

除了 RGB 模型，彩色图像还可以根据需要采用其他彩色图像模型表示，如 HSV(Hue，Saturation，Value)、HSI(Hue，Saturation，Intensity)、HSL(Hue，Saturation，Lightness)颜色空间，具体的概念及各种模型之间的对应转换方法将在第 9 章详细介绍。

### 1.1.4　数字图像的采样与量化

图像的数字化一般指将连续图像在空间上分割为 M×N 个方格，每个方格称为一个像素，每个像素用一个灰度值表示，这样就使连续图像变成了离散的数字图像。采样(Sampling)与量化(Quantization)技术与图像数字化的质量有直接的关联。

图像数字化过程中，单位距离内取的像素点越多，则采样率(Sampling Rate)越高，图像的分辨率(Resolution)也越高，图像的细节保留越多，画质越细腻。反之，则图像画质越粗糙。为确定采样点位置，常见的采样方法包括最近邻插值(Nearest Neighbor Interpolation)、双线性插值(Bilinear Interpolation)和双立方插值(Bicubic Interpolation)等。

量化的方法决定了每个像素范围内的灰度取值，为使量化后的图像尽可能保留图像的颜色信息，常见的图像量化方法包括最近邻法(Nearest Neighbor Method，NNM)、误差扩散法(Error Diffusion Method，EDM)、中位值切割法(Median Cut Algorithm，MCA)、矢量量化法(Vector Quantization, VQ)和小波变换法(Wavelet Transform Method，WTM)等。这些方法各有优缺点，适用于不同的应用场景。

图 1-4 为采用不同的采样率对同样的图像进行数字化的效果对比，下面图像的采样频率高于上面图像的采样频率，可以明显看出采样频率高的图像能保留更多原图中颜色和位置的细节。

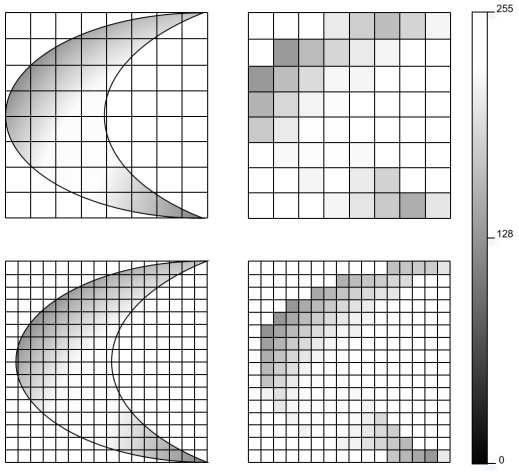



图 1-4　采样频率不同时采样效果对比图

量化后，灰度图像和 RGB 彩色图像均可用二维数组或矩阵表示，只是各元素的结构和所占存储空间有所不同，图 1-5 和图 1-6 分别展示了对单通道灰度图像和三通道 RGB 图像的量化结果。

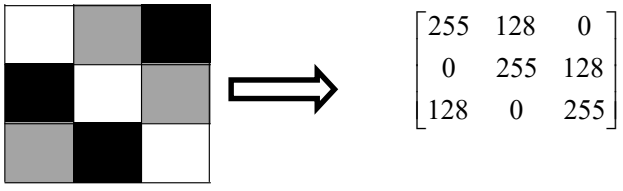



图 1-5　单通道灰度图像的量化表示

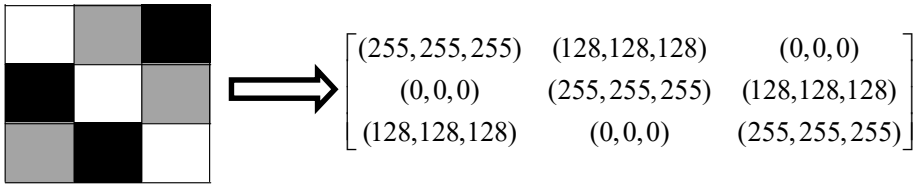



图 1-6　三通道 RGB 图像的量化表示

## 1.2 认识数字图像处理

数字图像处理作为现代信息处理的关键技术之一，其应用广泛且深入。本节将介绍数字图像处理的含义、应用领域、常见技术、基本步骤及数字图像文件格式，以使读者明确数字图像处理的目标和任务，了解其在各个领域的应用价值，并掌握常见的图像处理技术和基本处理步骤。

### 1.2.1　数字图像处理的含义

数字图像处理(Digital Image Processing，DIP) 指通过计算机对图像进行包括但不限于去除噪声、增强、压缩、复原、分割、提取特征等数学运算或加工处理的方法和技术，广泛意义上，利用计算机对图像进行的操作均可称为数字图像处理，但我们重点关注的是那些以达到提高图像视觉质量、提高图像可理解性、提取图像信息及减少图像存储空间等为目的的操作。

### 1.2.2　数字图像处理的应用领域

为了方便、准确地获取各类图像蕴含的重要信息，人们开始尝试用计算机处理图像。随着计算机技术不断发展，20 世纪 60 年代出现了具备处理数字图像能力的大型计算机，数字图像处理发展为一门学科。其技术第一次的成功运用也是在那时被记载下来的：美国喷气推进实验室(Jet Propulsion Laboratory，JPL)对1964 年传回的月球照片进行了畸变矫正、去除噪声、灰度变换等操作，获取了月面信息，这是数字图像处理技术被应用到了空间探索领域的重要里程碑。在此之后，数字图像处理的应用很快扩展到了天文学、遥感监测和医学成像等领域。

(1) 在空间探索、天文、遥感、航天航空领域，数字图像作为研究对象信息的重要来源，其处理技术的应用是各项研究的基础。例如，通过分析获取的太空影像，可以分析月球(图 1-7)、火星等其他星球的地形、地貌，辅助太空探索；通过分析卫星图像或航空摄影图像(图 1-8)，监测环境变化、农业发展、城市扩张、气候变化等情况，并监控土地使用、森林砍伐和自然灾害等现象；飞机遥感技术和卫星遥感技术也在很大程度上依赖数字图像处理技术。

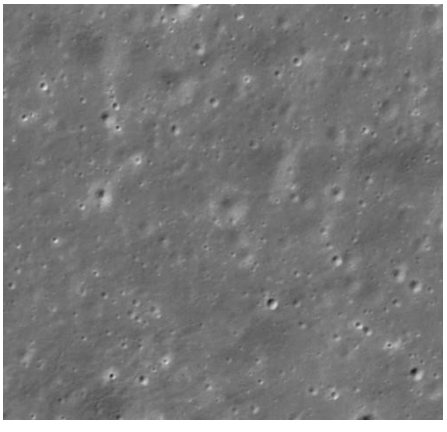


图 1-7　月背地表照片



图 1-8　地球地貌照片

图 1-8

(2) 在医学成像领域，数字图像处理技术自从在 1972 年发明的计算机断层技术(CT 技术)中起到了重要作用开始，就一直在现代医学无损诊断中发挥着不可替代的作用；医学影像的处理、分析及智能诊断一直是数字图像处理领域关注的一个重要分支；数字图像处理技术在分析药物对细胞和组织的影响方面起到的作用，使其在药物研发和监测过程中也是

必不可少的。

(3) 工业检测领域涉及的图像处理技术包括应用日益广泛的自动检测、无损检测、智能生产、质量控制等技术。数字图像处理技术既能在军工、电力等人工检测不便的场景下顺利、准确地完成检测任务，也能帮助普通工厂大幅度提高常规生产线的检测效率和产品质量。火炮身管缺陷检测、深海沉管质量检测、产品表面缺陷检测、零件标准测量等均属数字图像处理应用的常规场景。

(4) 在农业领域，通过对农田图像、农作物图像进行分析，可以监测农作物健康和产量、识别农作物病害以及时采取防治措施；通过对农产品图像的分析，还可以实现对农产品的质量检测和分级。

(5) 在智能安防领域，人脸识别、指纹识别等技术依赖于对摄像头实时拍摄的人脸图像进行特征提取和匹配等数字图像处理技术，它们是目前流行的智能安防领域的核心技术；监控视频的内容分析和行为识别也依赖于图像数据的处理和分析。

(6) 在零售业领域，对商品图像进行识别和检测，可以实现商品的自动识别、计数和分类，可以实现自动结账和库存管理等功能；大型商场通过对人流监控摄像头拍摄的人群图像进行识别和分析，可以实现对客流量、年龄、性别等信息的自动统计。

(7) 在娱乐及社交媒体领域，数字图像处理技术可以帮助实时识别和分析玩家动作以完成游戏互动；通过脸部表情识别技术还能实现情感交互和个性化推荐。

总之，从高空到深海、从生产到生活，图像数据为多个领域提供了关键信息，数字图像处理的技术触角几乎深入了人类社会的方方面面。近年来，计算机技术和人工智能的迅速发展，推动了智能化服务的进步，对数字图像处理技术的研究也向更深、更高的层次发展，目前热度逐年攀高的**计算机视觉**(Computer Vision，CV)的研究目标就是在对图像进行一系列处理的基础上，深入分析、理解和解释图像信息，以模拟人类视觉系统。

### 1.2.3 常见的数字图像处理技术

数字图像处理技术涵盖了一系列用于改善、分析和理解图像的方法。随着数字图像技术的发展，人们一直在研究图像处理的新方法和新技术。以下列出最基本、最重要的数字图像处理技术。

(1) 图像的基本运算。图像的基本运算包括点运算、算术运算、几何运算，可以用以改善图像质量、提取信息或根据特定的应用需求调节图像，是许多高级图像处理任务的基础。

(2) 图像变换。图像变换包括傅里叶变换(Fourier Transform,FT)、沃尔什变换(Walsh Transform，WT)、离散余弦变换(Discrete Cosine Transform，DCT)等，用于将图像在空域与频域中的转换及处理，以提取特征、改善质量或实现压缩。例如，在频域中利用数学方法改变图像数据的排列或表示可以很方便地过滤掉在空间域中不便处理的规律噪声等。

(3) 图像增强。图像增强旨在改善图像质量，增强图像中的有用信息，抑制不感兴趣的信息，以便于后续的分析和处理。图像增强包括去除噪声、提高清晰度等，是图像处理与分析中的重要处理步骤。

(4) 图像复原。图像复原旨在恢复图像的原始状态，消除由各种原因(如噪声、模糊、退化等)导致的图像质量下降。常见的图像复原方法是根据退化的先验知识或降质过程建立模型，并据此模型对图像进行恢复或重建。图像复原广泛应用于医学成像、卫星图像处理、视频增强和历史文献的数字化恢复等领域。

(5) 图像编码压缩。图像编码压缩是减少图像数据量的技术，目的是在保证可接受的图像质量的前提下，提升数字图像的传输效率、处理效率和存储效率，优化资源使用率和用户体验。图像编码压缩广泛应用于多媒体通信、互联网传输、数字存储、视频监控、医学影像、卫星遥感、移动设备及云计算等多个领域。

(6) 图像分割。图像分割旨在分离图像中的目标与背景，以便于后续的分析和处理。它通过图像分割将图像分成若干个具有特定性质的区域或对象，并从图像中提取有意义的特征部分，如边缘、区域等，是图像识别和分析的基础。图像分割广泛应用于智能识别、无人驾驶、医学影像分析等各类图像处理领域。

(7) 彩色图像处理。彩色图像处理是图像处理领域的一个重要分支，它专注于对图像中的颜色信息进行专门处理。该技术涉及色度学和编码学的知识，可以实现图像颜色校正、颜色空间转换、颜色量化、色彩恢复等操作，可有效支持进一步的图像分析和识别任务。

(8) 图像表示与描述。图像表示与描述是指根据图像的区域特性及区域间的联系(如几何特性或纹理特征等)，使用特定的数据结构来反映区域、边界等图像信息，一般包括纹理描述、边界描述和区域描述等。好的表示与描述方法能显著提高图像处理的效率和准确性。

除上述图像处理技术外，还有一些与大数据、大模型关系密切的新兴技术，这些技术既可以单独使用，也可以组合使用，以满足特定的应用需求。本教材将着重介绍上述技术的具体应用。

### 1.2.4　数字图像处理的基本步骤

一般来说，数字图像处理包括读取图像、预处理、核心处理、保存、显示结果等步骤，步骤是通用的。本节采用通用的伪码[①]形式说明数字图像处理的过程。

```
/*****************************************
    程序名：eg 1.1
    描　述：数字图像处理的一般过程伪码示例
*****************************************/
1.  BEGIN
2.      // 步骤 1: 输入图像
3.      image = READ_IMAGE(image_path)
4.
5.      IF image IS NOT FOUND THEN
6.          PRINT "Image not found or the path is incorrect"
```

① 伪码是一种描述算法或过程的高级抽象语言，它不依赖于任何特定的编程语言。

```
        EXIT
END IF

// 步骤 2: 预处理
preprocessed_image = PREPROCESS_IMAGE(image)
        // 转换为灰度图像
        gray_image = CONVERT_TO_GRAYSCALE(image)
        // 应用高斯模糊以减少图像噪声
        blurred_image = APPLY_GAUSSIAN_BLUR(gray_image)
        // 可选：其他预处理步骤，如直方图均衡化等
        RETURN blurred_image

// 步骤 3: 图像分割
segmented_image = SEGMENT_IMAGE(preprocessed_image)
        // 应用阈值化进行图像分割
        thresholded_image = APPLY_THRESHOLDING(preprocessed_image)
        // 可选：其他分割方法，如基于边缘的分割等
        RETURN thresholded_image

// 步骤 4: 显示结果
DISPLAY_IMAGE("Original Image", image)
DISPLAY_IMAGE("Preprocessed Image", preprocessed_image)
DISPLAY_IMAGE("Segmented Image", segmented_image)

END
```

此伪码框架提供了一个通用的数字图像处理流程，包括图像的读取、预处理、分割和显示。在实际编程中，可以根据所使用的编程语言和图像处理库来实现这些步骤的具体细节。

### 1.2.5 数字图像文件格式

目前数字图像文件有多种文件格式，常见的有以下几种。

(1) BMP (Bitmap)(.bmp)。这是一种位图图像文件格式，通常不压缩，能够保持图像的原始质量。

(2) JPEG(Joint Photographic Experts Group) (.jpeg, .jpg)。这是一种广泛使用的有损压缩的图像文件格式。它通过去除图像中的冗余数据和颜色信息来实现压缩，常用于捕捉和存储高清环境图像，尽管存在一定的质量损失，但其高压缩比使得存储和传输更为高效。

(3) PNG(Portable Network Graphics) (.png)：这是一种无损压缩的图像文件格式，支持透明背景，能够保持图像的原始质量，对需要保持图像完整性和透明度的增强现实、人机交互界面等场景是理想的选择。

(4) GIF(Graphics Interchange Format) (.gif)。这是一种支持动画的图像文件格式，使用

无损压缩技术，但颜色深度较低(最多 256 色)，适用于色彩简单、需要动画效果的场景。

(5) TIFF(Tagged Image File Format) (.tiff, .tif)。这是一种灵活的高质量图像文件格式，支持多种颜色模式和无损压缩，适用于图像编辑和高质量打印，以及需要高精度图像处理的场景，如机器视觉检测、地图构建等。

(6) WebP (Web Picture)(.webp)。这是由谷歌(Google)开发的一种图像格式，旨在提供更好的压缩和质量，广泛应用于前端开发、网站开发和移动应用程序中。

(7) RAW(Raw Image Format)(.nef, .cr2, .cr3 等)。这是由相机或图像传感器捕获的未经处理的原始数据格式，包含大量图像信息，但需要进行后期处理才能转换为可查看的格式，因此一般不直接用于实时处理，但可以作为原始数据保存，以便后续进行高精度分析和处理。

## 1.3 认识数字图像处理工具

目前，在进行数字图像处理实践和实验时，从易用性、库的支持及可视化能力方面，最适合、最流行的编程语言有 MATLAB、Visual C++和 Python。这些语言不仅提供了丰富的图像处理库和工具，还具备易于学习、高效执行等特点，能够满足学生在学习和实践中的需求。

### 1.3.1 MATLAB

#### 1. 简介

MATLAB 是美国 MathWorks 公司开发的一款数学软件，集成了算法开发、数据可视化、数据分析及数值计算的高级计算语言和交互式环境。其编程语言以矩阵和数学操作为基础，简洁明了，在描述及处理大量数据或进行复杂计算时表现出色。MATLAB 不仅适用于工程计算，还广泛应用于数学、物理、化学、生物学、金融、经济学等多个领域。

MATLAB 在数字图像处理方面发挥着极其重要的作用，其强大的图像处理工具箱(Image Processing Toolbox)为用户提供了丰富的图像处理和分析功能。

下列函数为 MATLAB 在数字图像处理方面的常用基础处理函数。

**常用 MATLAB 函数简介**

- imread 函数：能够轻松实现多种格式的图像文件(如 JPEG，PNG，BMP 等)的读取，读取后的图像数据可以被存储为矩阵形式，其中每个元素代表图像中的一个像素点。
- imshow 函数：可以方便地显示图像，直观地查看图像处理的效果。
- imwrite 函数：用于将处理后的图像保存为文件。
- rgb2gray 函数：图像灰度化。
- imbinarize 函数：图像二值化。

- imfilter 函数：MATLAB 中用于图像滤波的一个非常强大的函数。它允许用户对图像进行各种线性滤波操作，如平滑、锐化、边缘检测等。imfilter 函数通过指定的滤波器(也称为卷积核或模板)对图像进行卷积操作，以达到所需的滤波效果。
- histeq 函数：直方图均衡化。
- imadjust 函数：图像对比度调整。
- fft 函数：傅里叶变换。
- dct 函数：离散余弦变换。

除此之外，MATLAB 还提供了 Canny 边缘检测、Harris 角点检测等多种特征提取方法，支持基于阈值、区域生长、分水岭等多种图像分割方法，MATLAB 的图像处理工具箱还提供了丰富的图像识别和分类功能。结合 MATLAB 的其他工具箱［如统计和机器学习工具箱(Statistics and Machine Learning Toolbox)、计算机视觉工具箱(Computer Vision Toolbox)等］，还可以实现使用支持向量机(Support Vector Machine，SVM)、神经网络等机器学习算法对图像进行分类和识别的复杂任务。

### 2. 示例代码

下面代码为以 MATLAB 语言编写的图像处理程序。

```
/*****************************************
    程序名：eg 1.2
    描　述：Matlab 数字图像处理过程代码示例
*****************************************/
% 读取图像
I = imread('example.jpg');  % 读取文件名为'example.jpg'的图像
if size(I, 3) == 3
        % 将图像转换为灰度图像
        I_gray = rgb2gray(I);
else
        I_gray = I;
end

% 预处理：使用高斯滤波去除噪声
h = fspecial('gaussian', 5, 1.5);
I_smooth = imfilter(I_gray, h);

% 核心处理：使用 Canny 算法进行边缘检测
edges = edge(I_smooth, 'Canny');

% 显示原始图像和边缘检测结果
subplot(1, 2, 1);
imshow(I_gray); title('Original Grayscale Image');
```

```
subplot(1, 2, 2);
imshow(edges); title('Edge Detected Image');

% 保存处理后的图像
imwrite(edges, 'edges_detected.jpg');  % 保存为文件名为'edges_detected.jpg'的图像
```

这段代码首先读取一个图像文件，检查图像是否为彩色图像，如果是，则将其转换为灰度图像；接着使用高斯滤波器对灰度图像进行平滑处理，以去除噪声；然后使用 Canny 算法进行边缘检测；最后将原始的灰度图像和检测到的边缘在两个子图中显示出来，并将边缘检测的结果保存为新的图像文件。

#### 3. 特点

优势：MATLAB 提供了丰富的图像处理函数和工具箱，能够快速、高效地完成图像处理任务；语法简单、容易上手，在矩阵运算方面表现出色；可视化能力强，能通过函数直接展示可视化界面，对图像处理结果的观测更加直观，便于调试和验证算法。

缺点：虽然 MATLAB 在矩阵运算方面表现出色，但在多线程和多核处理器支持方面相对较弱，因此对于大规模图像处理的速度一般相对较慢；MATLAB 的面向对象编程支持相对较弱，当需要用户自行编写代码或函数来实现特定功能时，其扩展性可能在一定程度上被限制。

### 1.3.2 Visual C++

#### 1. 简介

Visual C++是微软公司开发的一款便捷实用的 C 语言编程软件，软件功能强大，由多种组件组成，是一种集成开发环境(Intergrated Development Invironment，IDE)。它是 Visual Studio 系列工具集中的一个组件，用于 C、C++和 C++/CLI(C++的一种扩展，支持.NET Framework)等编程语言的软件开发。Visual C++提供了一系列的工具和特性，使 C++编程更加高效和便捷。最重要的是，Visual C++提供了一个图形用户界面(Graphical User Interface，GUI)，使得编码、调试、编译和部署应用程序变得更加直观和高效。

Visual C++在图像处理方面扮演着重要的角色，其强大的编程能力和与 Windows 操作系统的紧密集成，使得它成为开发图像处理应用程序的优选工具之一。

#### 2. 示例代码

Visual C++本身不直接提供图像处理算法，但开发者可以利用 C++语言的高效性和灵活性，编写自己的图像处理算法。而且 Visual C++支持集成第三方图像处理库，其中在工业和科研中应用较为广泛的有 OpenCV，HALCON，MIL 等图像处理库。此外，还有 CxImage，CImg，Dlib，ITK (Insight Segmentation and Registration Toolkit)，Boost.GIL 等图像处理库也可以与 Visual C++结合使用。这些图像处理库各有特点，提供了不同的图像处理和可视化功能。

以下代码为用 Visual C++结合 OpenCV 库执行与 eg 1.2 中同样处理过程的代码示例。

```
/*************************************************************
    程序名：eg 1.3
    描　述：Visual C++结合 OpenCV 进行与 eg 1.2 同样过程的代码示例
*************************************************************/
#  include <OpenCV2/OpenCV.hpp>
#  include <iostream>
using namespace cv;
using namespace std;

int main() {
    // 读取图像(默认以彩色模式读取)
    string imagePath = "path_to_your_image.jpg";
    Mat colorImage = imread(imagePath);

    if (colorImage.empty()) {
        cout << "Error loading image!" << endl;
        return -1;
    }

    // 检查图像是否为灰度图(通道数为 1)
    Mat grayImage;
    if (colorImage.channels() == 1) {
        // 已经是灰度图，直接赋值
        grayImage = colorImage;
    } else {
        // 是彩色图，转换为灰度图
        cvtColor(colorImage, grayImage, COLOR_BGR2GRAY);
    }

    // 核心处理(高斯滤波和 Canny 边缘检测)
    Mat blurredImage, edgesImage;
    GaussianBlur(grayImage, blurredImage, Size(5, 5), 1.4);
    Canny(blurredImage, edgesImage, 50, 150);

    // 创建窗口显示原始灰度图像和边缘检测结果
    namedWindow("Original Gray Image", WINDOW_AUTOSIZE);
    imshow("Original Gray Image", blurredImage);  // 显示平滑后的灰度图像

    namedWindow("Canny Edges", WINDOW_AUTOSIZE);
    imshow("Canny Edges", edgesImage);
```

```
37.
38.         // 等待按键再关闭窗口
39.         waitKey(0);
40.
41.         // 保存边缘检测的结果为新的图像文件
42.         string outputPath = "path_to_save_edgesImage.jpg";   //指定保存位置及文件名
43.         bool isSaved = imwrite(outputPath, edgesImage);
44.         if (!isSaved) {
45.             cout << "保存边缘检测结果失败" << endl;
46.         } else {
47.             cout << "边缘检测结果已保存为  edges_detected.jpg" << endl;
48.         }
49.
50.         return 0;
51.     }
```

**常用 OpenCV 函数简介**

- imread 函数：用于读取参数指定的图像文件。
- cvtColor 函数：用于在不同颜色空间之间转换图像，如从 BGR 转换到灰度图或 HSV 等。
- GaussianBlur 函数：对图像进行高斯模糊处理，常用于去除图像噪声和边缘平滑。
- Canny 函数：使用 Canny 算法检测图像中的边缘。
- namedWindow 函数：创建一个窗口，用于显示图像。
- imshow 函数：在指定的窗口中显示图像。
- waitKey 函数：等待用户按键事件，通常用于控制图像显示的持续时间，若参数设置为 0，则无限期等待。
- destroyAllWindows 函数：销毁所有由 OpenCV 创建的窗口。
- imwrite 函数：将图像写入文件，用于保存图像到磁盘。

### 3. 特点

优势：Visual C++的基础编程语言 C++以其高性能著称，C++支持直接硬件访问和底层操作，执行速度通常比 Python 和 MATLAB 等解释型语言要快得多，在需要快速处理大规模数据或进行复杂计算的图像处理任务时，能够提供更好的性能表现。另外，C++具有强大的面向对象编程能力，在构建大型、复杂的图像处理系统时具有更好的可扩展性和灵活性。Visual C++提供的图形化界面也可以使开发、部署和测试算法更为直观。

缺点：相比于 Python 和 MATLAB，Visual C++的语法相对复杂，学习起来需要一定的时间和精力，且需要手动管理内存和进行类型检查等操作。因此，使用 Visual C++语法可能会增加开发过程中的复杂性和出错率，使开发周期可能更长。

### 1.3.3 Python

#### 1. 简介

Python 语言是一种开放源代码的编程语言，是由荷兰计算机科学家吉多·范罗苏姆(Guido van Rossum)在 1989 年开发的，由于受到了他之前参与的 ABC 语言项目的影响，其设计理念是开发一种介于 C 和 shell 之间的功能全面、易学易用、可拓展的语言。Python 语言的后续发展离不开社区、组织和公司的支持和贡献，Python 软件基金会(Python Software Foundation，PSF)是一个非营利组织，致力于促进、维护和保护 Python 语言的发展，负责管理 Python 语言的商标和版权、组织 Python 开发者社区的活动，并提供资金和资源支持 Python 相关项目。此外，谷歌、脸书(Facebook)、多宝箱(Dropbox)和照片墙(Instagram)等许多知名的科技公司也是 Python 语言的积极支持者和贡献者。

Python 在图像处理领域的应用非常广泛，从基本的图像读取、显示、转换到复杂的图像分析、识别、压缩和编码，都可以通过 Python 及其强大的图像处理库和工具集来实现。用 Python 语言进行数字图像处理时，常用到的图像处理库有 OpenCV，Pillow (PIL Fork)，scikit-image，NumPy，Matplotlib，SimpleCV 等。Python 简洁的语法、丰富的库支持和活跃的社区使得它成为图像处理研究和开发的理想选择。

#### 2. 示例代码

```
/*************************************************************
    程序名：eg 1.4
    描　述：Python 结合 OpenCV 进行与 eg 1.2 同样过程的代码示例
*************************************************************/
import cv2
import numpy as np

# 读取图像文件
image_path = 'your_image_path.jpg'  # 指定图像文件路径
image = cv2.imread(image_path)

# 检查图像是否为彩色图像
if len(image.shape) == 3:  # 彩色图像有 3 个通道(高度、宽度、通道数)
        print("图像是彩色的，将转换为灰度图像。")
        # 转换为灰度图像
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
else:
        print("图像已经是灰度图像。")
        gray_image = image

# 使用高斯滤波器对灰度图像进行平滑处理
blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
```

```
19.
20.  # 使用 Canny 算法进行边缘检测
21.  edges = cv2.Canny(blurred_image, threshold1=50, threshold2=150)
22.
23.  # 显示原始的灰度图像和检测到的边缘
24.  # 创建一个带有两个子图的窗口
25.  cv2.namedWindow('Original vs Edges', cv2.WINDOW_NORMAL)
26.  cv2.imshow('Original vs Edges', np.hstack((gray_image, edges)))
27.
28.  # 等待用户按键后关闭窗口
29.  cv2.waitKey(0)
30.  cv2.destroyAllWindows()
31.
32.  # 将边缘检测的结果保存为新的图像文件
33.  output_path = 'edges_detected.jpg'   # 指定输出的图像文件及路径
34.  cv2.imwrite(output_path, edges)
35.  print(f"边缘检测结果已保存为：{output_path}")
```

### 3. 特点

优势：Python 的语法简洁，代码具有很好的可读性，且可移植性强；具有高效率的数据结构和丰富的第三方图像处理库支持，提供了广泛的图像处理功能，能够满足大多数图像处理实践的需求；拥有丰富的社区支持和大量的在线资源，使学习和使用更加方便；作为跨平台的编程语言，可以在多个操作系统上运行，在数字图像处理实践中具有更好的可移植性和兼容性。

缺点：Python 作为解释型语言(Interpreted Languages)在处理大规模数据或进行复杂计算时，其性能通常不如 Visula C++等编译型语言(Compiled Languages)，占用内存空间可能更大，运行速度可能较慢。

经过对比上述三种流行的图像处理工具，可以看出 MATLAB 因其算法研究与开发的便利性和直观编程风格而受到青睐，适合需要快速算法测试的预研场景，但运算速度较慢，且无法直接形成可执行程序；Visula C++以高效执行和底层控制能力在性能和实时处理要求较高的场合占据优势；而 Python 具有易学易用、拥有强大的社区支持和丰富的图像处理库资源的优势。因此，本书将 Python 作为数字图像处理教学及进行快速实践的首选，后续章节中的代码将以 Python 为主要使用语言，感兴趣的读者也可以另外学习 MATLAB 和 Visual C++实践技术。

【扩展阅读】
中国光学之父王大珩



## 1.4 思考练习

### 一、选择题

1. 采样和量化是数字图像处理中的两个关键步骤，其中哪个步骤决定了图像中最多可以包含的颜色数量？(　　)

A. 采样　　B. 量化　　C. 两者都决定　　D. 两者都不决定

2. 下列哪项不是数字图像处理技术的常见应用？(　　)

A. 医学成像分析　　B. 安全监控与识别

C. 文本编辑(如 Word 文档编辑)　　D. 卫星图像处理

3. 在数字图像处理的基本步骤中，哪一步通常用于改善图像质量或突出图像中的特定特征？(　　)

A. 图像获取　　B. 图像增强　　C. 图像编码　　D. 图像显示

4. 下列哪种文件格式主要用于无损压缩图像？(　　)

A. JPEG　　B. PNG　　C. GIF　　D. BMP

5. MATLAB 中用于读取图像文件的函数通常是下列哪个？(　　)

A. imread　　B. readimage　　C. loadimg　　D. imageinput

6. 在使用 Visual C++进行数字图像处理时，可以用下列哪个库来提供图像处理功能？(　　)

A. OpenCV　　B. NET Framework

C. MFC(Microsoft Foundation Classes)　　D. DirectX

7. 关于数字图像处理工具 MATLAB、Visual C++和 Python 的比较，以下哪个说法不正确？(　　)

A. MATLAB 提供了丰富的内置函数和工具箱，便于快速开发图像处理算法

B. Visual C++因其高效性和可控性，常用于对性能要求极高的图像处理任务

C. Python 因其简洁的语法和丰富的第三方库(如 OpenCV、Pillow)，成为学习和研究图像处理的热门选择

D. MATLAB 因其开源特性，广泛应用于商业软件开发和学术研究中

### 二、分析题

1. 解释什么是数字图像，并列出其三个基本属性。
2. 描述数字图像的采样过程和量化过程，并解释它们对图像质量的影响。
3. 列出进行数字图像处理的五个基本步骤，并简要说明每个步骤的目的。
4. 比较 MATLAB、Visual C++和 Python 在数字图像处理中各自的优势和局限性。
5. 选择一个你感兴趣的数字图像处理应用领域，调研其应用场景。

# 第2章

# Python与数字图像处理

本章将深入探讨使用 Python 进行数字图像处理的关键知识和技能。图像处理是现代计算机视觉和数据分析的重要组成部分，掌握相关工具将使读者在这些领域具有竞争力。本章主要围绕几个重要的图像处理库展开，包括 Pillow 库、NumPy 库、scikit-image 库、OpenCV 库等，并且介绍了如何在 Windows 系统上部署 Python 环境及安装编译器。



## 本章学习目标

◎ 了解 Python 与数字图像处理的关系。

◎ 掌握在 Windows 系统上部署 Python 环境。

◎ 了解 Pillow 库、NumPy 库、scikit-image 库、OpenCV 库等几个常用图像处理库的运用。



## 素质要点

◎ 培养创新能力：学生在学习 Pillow、NumPy、OpenCV 等图像处理库的过程中，应注重结合实际问题进行创新应用。例如，在图像滤镜或颜色调整等基础功能的基础上，尝试设计个性化的图像处理算法，探索新的图像分析方法，提高解决问题的能力。

◎ 提升实践精神：学生通过动手编写代码并反复调试，掌握数字图像处理技术的实际应用方法。在实验中，积极尝试图像的加载、转换、滤波等操作，将理论知识与实际应用紧密结合，为未来参与复杂项目奠定基础。

# 2.1 Python 环境部署

本节主要介绍 Python 的环境部署，包括 Python 的安装及 Pycharm 软件的安装。

## 2.1.1 Python 安装

在 window 系统上部署 Python，可以通过 Python 官网下载安装程序：首先选择合适的版本进行下载，下载完成后按照以下步骤进行安装。

(1) 双击 Python 安装程序，启动安装，如图 2-1 所示。

(2) 勾选“Add python.exe to PATH”，把 Python 的安装路径添加到系统的环境变量中，然后单击“Install Now”进行下一步操作。

(3) 单击“Close”完成 Python 的安装，如图 2-2 所示。




图 2-1 运行 Python 安装程序

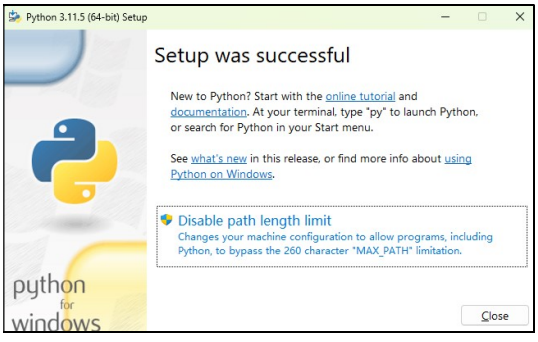



图 2-2 完成 Python 安装

## 2.1.2 Pychram 安装

在 window 系统上部署 Pycharm，可以通过 Pycharm 官网下载安装程序：首先选择合适的版本进行下载，下载完成后按照以下步骤进行安装。

(1) 双击 Python 安装程序，启动安装，如图 2-3 所示。单击“Next”进行下一步操作。

(2) 根据用户需求选择安装目录，无特殊要求，默认路径即可，并单击“Next”进行下一步操作，如图 2-4 所示。

(3) 勾选“Pycharm Community Edition”和“ADD "bin" folder to the PATH”为 Pycharm 添加桌面快捷方式及添加环境变量，如图 2-5 所示。

(4) 单击“Next”，软件默认会添加到开始菜单，如图 2-6 所示。

(5) 单击“Install”对 Pycharm 软件进行安装。Pycharm 安装完成，如图 2-7 所示。

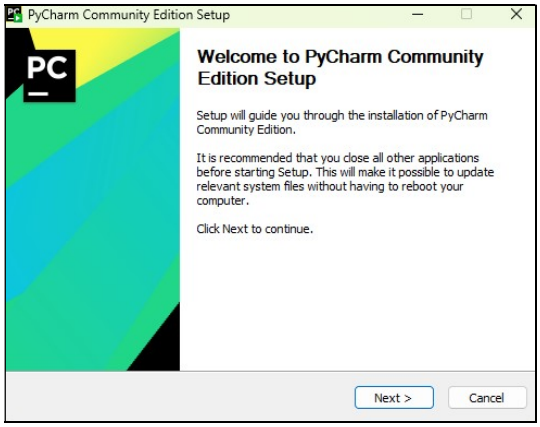



图 2-3　Pycharm 安装程序

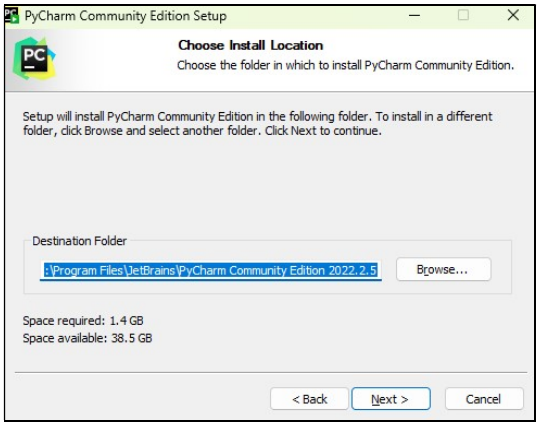



图 2-4　选择 Pycharm 的路径

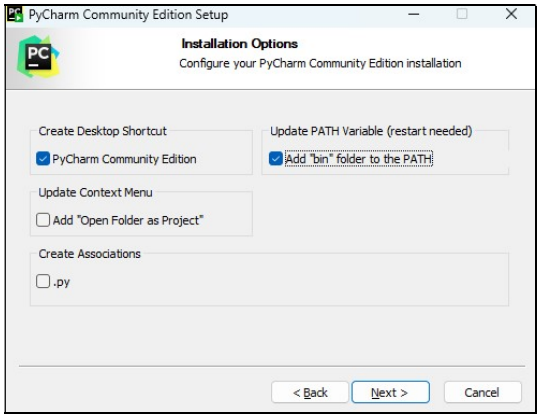



图 2-5　配置 Pycharm 安装环境

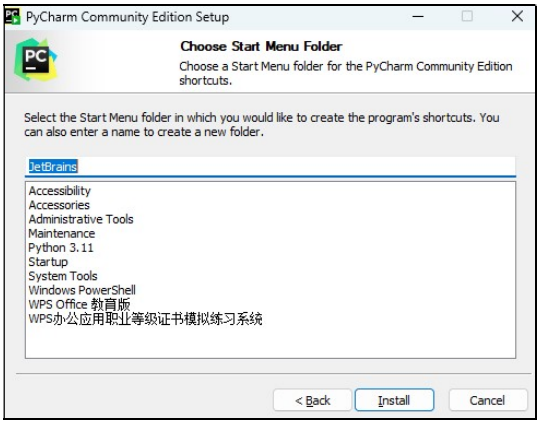



图 2-6　将 Pycharm 添加到开始菜单

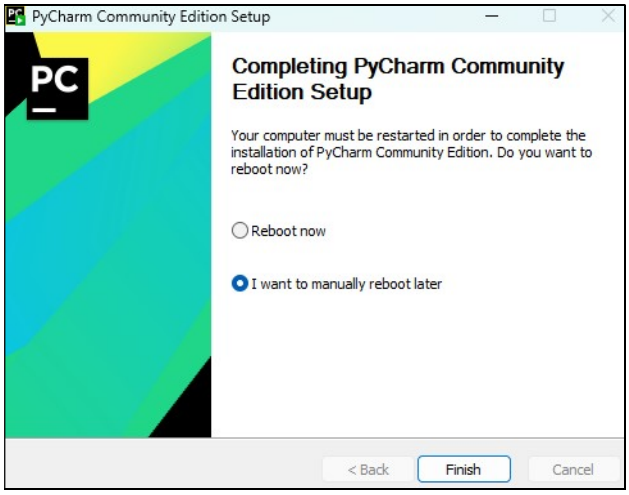



图 2-7　Pycharm 安装完成

# 2.2 了解 Pillow 库

本节介绍 Python 中的图像处理库 Pillow，包括其安装方法、基本功能(如图像加载、保存、转换、滤镜应用等)及应用示例。Pillow 是 PIL(Python Imaging Library)库的分支，广泛用于简单的图像处理操作。

## 2.2.1 Pillow 库的基本信息

Pillow 库是 Python 中的一个图像处理库，是 PIL 库的一个分支。Pillow 库支持许多图像格式，如 JPEG、PNG、BMP、GIF 等，能够执行各种图像处理任务，如裁剪、旋转、滤镜应用等。由于其简洁和易用的特性，Pillow 库被广泛用于图像处理、计算机视觉项目中。

本节主要讲述如何利用 Python 对数字图像进行基础的处理，并且介绍常用的几个处理库，主要以 Pycharm 编译器为基础示例。

## 2.2.2 Pillow 库的安装方法

方法一：在 Pycharm 编译器终端或者 Windows 命令提示符中输入“pip install Pillow”即可自动开始安装，如图 2-8 所示。

方法二：登录 Python Pillow(python-pillow.org)等网站，下载安装包，在终端处输入“pip install【安装包目录】”指令进行安装。

方法三：打开 Pycharm，点击文件→项目→Python 解析器→ ＋ →搜索栏输入 Pillow→安装软件包，如图 2-9 所示。




图 2-8




图 2-9　Python 解析器中安装 Pillow 库

注意：安装完后输入代码：from PIL import Image 验证 Pillow 有无正确安装。

### 2.2.3 Pillow 库的主要作用

Pillow 库主要包括以下常用功能。

(1) 加载和保存图像。Pillow 库支持从多种格式中读取图像并将其保存为不同格式的文件。常见的支持格式包括 JPEG、PNG、BMP、GIF、TIFF 等。开发者可以轻松地在不同图像格式之间转换。

例如，简单地读取一个名字为“2.jpg”的图片，并将该图像转换为 PNG 格式，代码如下：

```
/*****************************************
    程序名：eg 2.1
    描　述：图像格式转换
*****************************************/
1.  from PIL import Image
2.
3.  # 加载图像
4.  img = Image.open("2.jpg")
5.
6.  # 保存为不同格式
7.  img.save("tupian1.png")
```

如图 2-10 所示，利用 Pillow 库可以对图像进行常见的格式转换。



(a) 2.jpg　　(b) 2.png

图 2-10　Pillow 转换格式

(2) 图像转换。Pillow 库允许对图像进行多种转换操作，如旋转、翻转、缩放、裁剪、转换图像模式等。如图 2-11 所示，利用 Pillow 库进行图片旋转。



(a) 原图　　(b) 旋转图片

图 2-11　利用 Pillow 旋转图片

(3) 像滤镜。Pillow 提供了多种图像滤镜，允许开发者在图像上应用不同的效果，如锐化、模糊、边缘增强等。这些滤镜可以用于图像增强或美化。

代码示例如下：

```
/*****************************************
    程序名：eg 2.2
    描　述：图像模糊处理
*****************************************/
1.  from PIL import Image, ImageFilter
2.
3.  # 打开图像
4.  img = Image.open("image.jpg")  # 请将路径替换为你的实际图像路径
5.
6.  # 应用高斯模糊滤镜，其中数值控制模糊程度
7.  blurred_img = img.filter(ImageFilter.GaussianBlur(50))
8.
9.  # 保存模糊后的图像
10. blurred_img.save("mohu1.jpg")
```

(4) 颜色调整。Pillow 库支持对图像的亮度(Intensity)、对比度(Contrast)、色调(又称色相，Hue)、饱和度(Saturation)等进行调整。通过这些操作，可以实现图像的颜色校正、增强效果。调整亮度等参数，还需要导入“ImageEnhance”模块进行图像处理。

代码示例如下：

```
/*****************************************
    程序名：eg 2.3
    描　述：调整图像颜色
*****************************************/
1.  from PIL import Image, ImageEnhance  # 确保导入 ImageEnhance
2.
3.  # 打开图像
4.  img = Image.open("image.jpg")  # 请将路径替换为你的实际图像路径
5.
6.  # 调整亮度
7.  enhancer = ImageEnhance.Brightness(img)
8.  bright_img = enhancer.enhance(5)  # 5 表示大幅度增强亮度
9.
10. # 保存调整后的图像
11. bright_img.save("liangdu.jpg")
```

(5) 图像合成与透明度处理。Pillow 库支持多幅图像的合成操作。比如，可以将一张图像叠加到另一张图像上，应用透明度或 Alpha 通道处理，创建复合图像效果。透明度处理

在处理 PNG 或 RGBA 模式图像时非常有用。

代码示例如下：

```
/*****************************************
    程序名：eg 2.4
    描　述：合成图像
*****************************************/
1.  from PIL import Image
2.  # 合成两张图像
3.  background = Image.open("diejia1.jpg")
4.  foreground = Image.open("diejia2.jpg")
5.  # 使用透明度进行合成
6.  background.paste(foreground, (50, 50), foreground)
7.  background.save("combined_image.png")
```

(6) 绘图功能。Pillow 库提供了一个 ImageDraw 模块，支持在图像上绘制文本、线条、矩形、圆形等基本图形。该功能在生成标记图像、图像水印等场景下非常有用。

(7) 通道处理。Pillow 库允许对图像的每个通道(如 RGB 的红色通道、绿色通道、蓝色通道)进行独立处理。通道处理在需要对图像特定颜色进行操作时非常有用，如在图像中分离某个通道进行分析。

(8) 多帧图像处理。对于 GIF 等多帧图像，Pillow 库支持逐帧处理，可以提取和操作每一帧图像，并保存为新的 GIF 或视频格式。

## 2.3 了解 NumPy 库

本节讲解 NumPy 库的核心概念及在图像处理中的应用，重点介绍了其多维数组操作、数学运算及线性代数功能，并通过代码展示如何进行图像灰度化和模糊处理。

### 2.3.1 NumPy 库概述

NumPy(Numerical Python)库是一个用于 Python 编程语言的核心科学计算库。它提供了对大型多维数组和矩阵的支持以及对这些数据结构执行数学运算的工具。NumPy 库是许多数据分析、机器学习和科学计算库的基础，如 Pandas、SciPy、TensorFlow 和 scikit-learn。

#### 1. 基本信息

NumPy 库提供了高效的数组操作功能，包括许多用于数组运算的数学函数。其主要数据结构是 ndarray，即 n 维数组，支持多维数据的存储和操作。NumPy 还包含了线性代数、傅里叶变换、随机数生成等多种数学功能，极大地扩展了 Python 在科学计算领域的能力。

2. 主要作用

(1) 多维数组操作。NumPy 库的核心是多维数组对象(ndarray)，支持高效的多维数组操作，包括数组创建、索引、切片、变形、合并和分割等。

(2) 数学运算。数学运算提供了广泛的数学函数，包括基本的算术运算、线性代数操作、统计运算等。

(3) 线性代数。线性代数包括矩阵乘法、特征值计算、奇异值分解等线性代数功能。

(4) 傅里叶变换。傅里叶变换支持快速傅里叶变换(Fast Fourier Transform，FFT)及其逆变换，用于信号处理等应用。

(5) 随机数生成。随机数生成提供了多种随机数生成方法，用于数据模拟和统计分析。

### 2.3.2 NumPy 库的安装方法

Numpy 库的安装方法和 Pillow 库的安装方法类似。

(1) 在 Pycharm 编译器终端或者 Windows 命令提示符中输入："pip install Pillow"，如果使用的是 Python3 则需要使用"pip3 install numpy"即可开始自动安装。

(2) 打开 Pycharm，点击文件→项目→Python 解析器→➕→搜索栏输入 numpy→安装软件包，如图 2-12 所示：

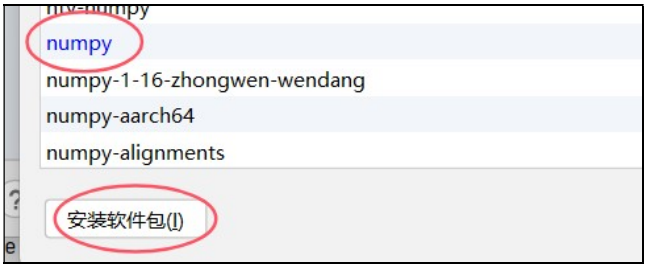



图 2-12 Python 解析器中安装 Numpy 库

### 2.3.3 NumPy 库的应用

NumPy 库是 Python 语言中用于科学计算的核心库。它为高性能的多维数组对象和与数组相关的操作提供了支持，并为执行数学和逻辑操作、傅里叶变换、线性代数等科学计算任务提供了便捷的工具。以下是 NumPy 库的主要作用。

1. ndarray

NumPy 库的核心 ndarray 是一个高效的 n 维数组，用于存储同类型元素。与 Python 内置的列表不同，ndarray 在内存上是连续的，允许更快地访问和操作。此外，NumPy 数组支持高维度的数据操作，如二维矩阵、三维张量等。

2. 数学运算

NumPy 库提供了丰富的数学函数，支持在数组上直接进行各种数学运算，如加减乘除、幂运算、对数、三角函数等。由于 NumPy 库的操作是针对整个数组进行的，因而其效率非常高。常见的数学操作包括：元素级运算(如加法、乘法)；数组的聚合操作(如求和、求平

均值)；广播机制，使不同形状的数组能够进行运算。

### 3. 线性代数运算

NumPy 库包含强大的线性代数功能，支持矩阵乘法、矩阵转置、求逆、特征值分解、奇异值分解等常见的线性代数操作。NumPy 库的 linalg 模块提供了线性代数计算的功能，方便处理矩阵和向量之间的操作。

### 4. 随机数生成

NumPy 库的 random 模块提供了生成随机数的功能，支持生成随机的标量、向量、矩阵以及符合特定分布的随机数。这对于蒙特卡罗模拟、随机抽样和统计建模等应用非常有用。

### 5. 高效的数据操作

NumPy 库支持对数组进行切片、索引、连接、拆分、重塑等操作，这些操作都是在 C 语言的基础上实现的，因此其性能非常高。此外，NumPy 库还支持布尔索引和花式索引，允许灵活地操作数组中的元素。

### 6. 兼容性和集成

NumPy 库与其他科学计算库(如 SciPy、Pandas、Matplotlib)紧密集成，是 Python 数据科学和机器学习生态系统的基础。NumPy 数组常常作为数据在不同库之间传递的标准格式，使得各种库能够无缝协作。

### 7. 高效存储和操作大规模数据

NumPy 库能够高效地处理大规模数据，特别是在数据分析和机器学习中，使用 NumPy 库可以显著提高代码的执行效率。同时，NumPy 库提供了高效的数据存储和读取功能，支持将数组数据保存到磁盘并从磁盘加载。

### 8. 支持广播和矢量化

NumPy 库的广播机制允许对不同形状的数组进行数学运算，而无须显式地扩展数据的形状。矢量化操作则使得循环的计算任务能够转化为数组级别的操作，从而大幅提高性能。

### 9. 文件输入输出

NumPy 库支持将数组数据保存到文件中，并可以从文件中读取数据。这包括文本文件(如 CSV)和二进制文件(如.npy，.npz)的读写操作。

### 10. 支持傅里叶变换和随机采样

NumPy 库还包含快速傅里叶变换和随机采样等功能，因而在信号处理、统计建模等方面具有广泛的应用。

### 11. 内存管理

NumPy 库通过对数组进行内存分配和管理，使得数组的操作更加高效。同时，NumPy 库还支持共享内存的多数组操作，避免了不必要的数据复制，提高了性能。

在数字图像处理中，NumPy 库经常用于图像的基本操作和处理，如图像的翻转、裁剪、滤波等。下面是一个基于数字图像处理的实际应用案例：将图像转换为灰度图，并应用高

斯模糊滤波。

在图像处理任务中，灰度化和模糊是常见的预处理步骤。灰度化可以减少图像的复杂性，而高斯模糊则可以平滑图像，减少噪声。例如，要将图 2-13 进行灰度化和模糊处理，则需要使用以下原理进行处理。处理该图像还需要调动 Matplotlib 和 scipy 模块，需要提前进行安装。



图 2-13

(1) 图像灰度化彩色图像通常由三个通道(红色通道、绿色通道、蓝色通道)组成，灰度化过程是将图像的 RGB 颜色转换为单通道灰度值。常用的转换公式是：

$$\text{Gray} = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2.1}$$

(2) 高斯模糊是一种基于正态分布的低通滤波器，用于平滑图像并减少细节。高斯模糊的效果是通过卷积操作实现的。

① 代码如下：

```
/****************************************
    程序名：eg 2.5
    描　述：图像模糊和灰度化处理
****************************************/
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  from scipy.ndimage import gaussian_filter
4.  from PIL import Image
5.  # 读取图像并转换为 NumPy 数组
6.  image = Image.open('n.jpg')  # 替换为你的图像路径
7.  image_np = np.array(image)
8.
9.  #  1. 将图像转换为灰度图
10. def rgb2gray(image):
11.         return 0.299 * image[:, :, 0] + 0.587 * image[:, :, 1] + 0.114 * image[:, :, 2]
12.
13. gray_image = rgb2gray(image_np)
14.
```

```
# 2. 应用高斯模糊滤波
blurred_image = gaussian_filter(gray_image, sigma=2)  # sigma 控制模糊程度

# 显示原图、灰度图和模糊后的图像
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.title("原图")
plt.imshow(image_np)

plt.subplot(1, 3, 2)
plt.title("灰度图")
plt.imshow(gray_image, cmap='gray')

plt.subplot(1, 3, 3)
plt.title("高斯模糊图")
plt.imshow(blurred_image, cmap='gray')

plt.show()
```

② 代码解释。

图像读取与转换：使用 PIL.Image 读取图像并将其转换为 NumPy 数组进行处理。

灰度化：将彩色图像转换为灰度图，使用线性加权的方式结合 RGB 通道。

高斯模糊：使用 scipy.ndimage 中的 gaussian_filter 函数对灰度图应用高斯模糊，sigma 参数控制模糊的程度。

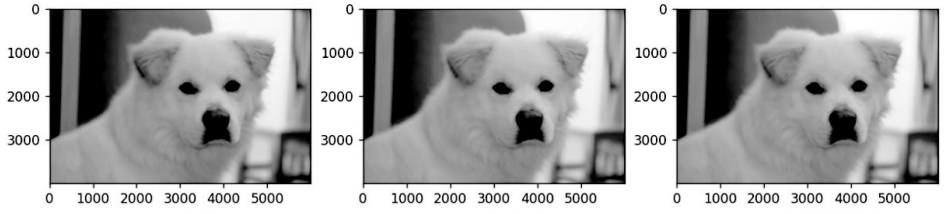
结果展示：使用 Matplotlib 显示原图、灰度图和模糊后的图像。

③ 应用场景。

灰度化在特征提取、边缘检测等任务中常用，因为灰度图比彩色图像更简单且保留了足够的信息。

高斯模糊广泛应用于去噪声处理、图像平滑和特征检测的预处理步骤。

此代码展示了如何使用 NumPy 库和相关库进行简单的图像处理操作，是数字图像处理中的基础应用，处理结果如图 2-14 所示。




(a) 原图　(b) 灰度图　(c) 高斯模糊图

图 2-14　Numpy 库处理完成的图像


图 2-14

# 2.4 了解 scikit-image 库

本节介绍了专用于图像处理的 scikit-image 库，涵盖其在图像预处理、特征提取、分割等方面的应用。

## 2.4.1 scikit-image 库的基础概念

scikit-image 库是一个专为图像处理设计的 Python 库，基于 NumPy 和 SciPy 构建。它提供了丰富的功能，用于图像的预处理、过滤、形态学操作、分割、特征提取等。scikit-image 库简单易用，且功能强大。

## 2.4.2 scikit-image 库的安装方法

scikit-image 库的安装方法和 Pillow 库的安装方法类似。

(1) 在 Pycharm 编译器终端或者 Windows 命令提示符中输入“pip install scikit-image”。

(2) 打开 Pycharm，点击文件→项目→Python 解析器→+→搜索栏输入 scikit-image 安装软件包，如图 2-15 所示。

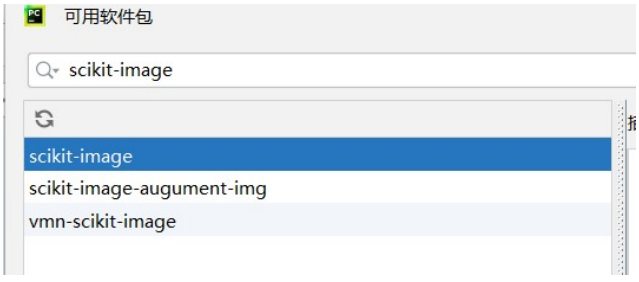



图 2-15　Python 解析器中安装 scikit-image 库

## 2.4.3 scikit-image 库的作用

scikit-image 库是一个专为图像处理设计的 Python 库，广泛应用于计算机视觉、图像分析、医学成像、遥感等领域。它提供了丰富的图像处理功能，可以帮助开发者和研究人员完成从简单的图像操作到复杂的图像分析任务。

scikit-image 库提供了多个子模块，涵盖各种图像处理功能。io 模块支持图像的加载与保存，color 模块支持 RGB、HSV、Lab 等颜色空间的转换，filters 提供边缘检测、平滑等图像滤波器，transform 实现旋转、缩放等几何变换，segmentation 负责图像区域分割，morphology 执行形态学操作如膨胀与腐蚀，feature 检测图像特征如边缘与角点，measure 用于测量图像中对象的属性，draw 则用于在图像上绘制几何图形。

scikit-image 库的主要作用有以下几点。

1. 图像预处理

(1) 灰度化：将彩色图像转换为灰度图，简化后续的处理任务。

(2) 缩放与旋转：支持图像的缩放、旋转、翻转等基本操作，用于数据增强或调整图像大小。

(3) 图像滤波：提供多种滤波器，如高斯滤波(CV2.GaussianBlur)、均值滤波(CV2.blur)、边缘检测滤波(CV2.Sobel)等，用于去噪、平滑或增强图像。

2. 图像增强

(1) 对比度调整：包括直方图均衡、自适应直方图均衡等，用于增强图像的对比度。

(2) 去噪：利用不同的滤波技术，如中值滤波、非局部均值滤波等，减少图像中的噪声。

3. 图像分割

(1) 基于阈值的分割：如 Otsu 阈值法，用于将图像分割为前景和背景。

(2) 区域生长：利用像素相似性将图像划分为不同区域。

(3) 聚类方法：如 K-means 聚类，用于将图像中的像素分组。

(4) 轮廓检测：用于提取图像中的轮廓和边界。

4. 边缘检测与特征提取

(1) 边缘检测：使用 Canny、Sobel(索贝尔)、Prewitt(普鲁伊特)等边缘检测算法，提取图像中的边缘信息。

(2) 特征提取：如角点检测、HOG(方向梯度直方图)、SIFT(尺度不变特征变换)等技术，用于在图像中检测和描述局部特征。

5. 形态学操作

(1) 膨胀与腐蚀：基于形态学原理的图像处理操作，用于调整图像中的结构元素。

(2) 开闭操作：用于消除图像中的小物体或填补小孔。

6. 图像复原与重建

(1) 去模糊：利用反卷积等方法恢复图像中的模糊区域。

(2) 图像插值与重建：基于采样点重建图像，如通过插值生成更高分辨率的图像。

7. 频域变换与分析

(1) 傅里叶变换：将图像从空间域转换到频域，用于频率分析和滤波操作。

(2) 小波变换：用于多分辨率分析和图像压缩。

8. 统计与度量

(1) 图像统计：计算图像的直方图、均值、方差等统计特性。

(2) 形状度量：计算图像中物体的面积、周长、紧致度等几何属性。

9. 高级应用

(1) 图像匹配与对齐：用于多幅图像的配准与拼接。

(2) 医学图像分析：应用于 CT、MRI(核磁共振)等医学图像的分割、测量与诊断。

10. 应用场景

(1) 计算机视觉：如物体识别、图像分类、自动驾驶等领域中的图像处理任务。

(2) 医学成像：如医学图像的预处理、分割和分析，用于辅助诊断。

(3) 遥感：用于分析卫星图像，提取地理特征或进行土地覆盖分类。

(4) 工业检测：用于检测生产线上的产品缺陷或质量问题。

(5) 科研实验：在科学研究中用于分析和处理实验数据中的图像信息。

我们可以使用 scikit-image 库对图 2-16(a)这张照片进行一些图像处理操作，如图像缩放、滤波、边缘检测等。

① 以下是利用 scikit-image 库对图片进行处理的 Python 代码示例。

```
/*****************************************
    程序名：eg 2.6
    描　述：图像模糊和灰度化处理
*****************************************/
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  from scipy.ndimage import gaussian_filter
4.  from PIL import Image
5.  # 读取图像并转换为 NumPy 数组
6.  image = Image.open('n.jpg')  # 替换为你的图像路径
7.  image_np = np.array(image)
8.
9.  # 1. 将图像转换为灰度图
10. def rgb2gray(image):
11.         return 0.299 * image[:, :, 0] + 0.587 * image[:, :, 1] + 0.114 * image[:, :, 2]
12.
13. gray_image = rgb2gray(image_np)
14.
15. # 2. 应用高斯模糊滤波
16. blurred_image = gaussian_filter(gray_image, sigma=2)  # sigma 控制模糊程度
17.
18. # 显示原图、灰度图和模糊后的图像
19. plt.figure(figsize=(12, 6)
20.
21. plt.subplot(1, 3, 1)
22. plt.title("原图")
23. plt.imshow(image_np)
24.
25. plt.subplot(1, 3, 2)
26. plt.title("灰度图")
```

```
27.  plt.imshow(gray_image, cmap='gray')
28.
29.  plt.subplot(1, 3, 3)
30.  plt.title("高斯模糊图")
31.  plt.imshow(blurred_image, cmap='gray')
32.
33.  plt.show()
```

② 代码解释。

灰度化：将彩色图像转换为灰度图，以简化后续的图像处理步骤。

高斯模糊：使用高斯模糊滤波器对图像进行平滑处理，减少噪声和细节。

边缘检测：使用索贝尔算子(Sobel Operator)进行边缘检测，提取图像中的边缘信息。

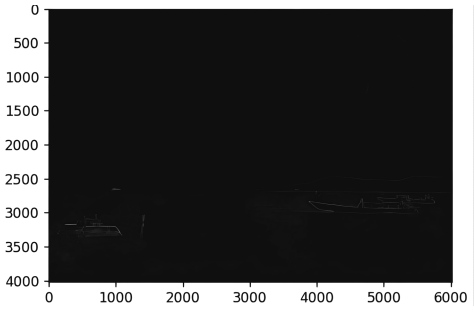
这个示例展示了如何通过 scikit-image 库对图像进行基本处理，并可视化处理结果。也可以根据需要调整参数，如高斯模糊的 sigma 值或 Sobel 算子(CV2.Sobel)的设置，以获得不同的效果，模糊处理的结果如图 2-16(b)所示，边缘检测处理后如图 2-16(c)所示。




(a) scikit-image 库处理原图




(b) 利用 scikit-image 库模糊处理




(c) 利用 scikit-image 库边缘检测

图 2-16　利用 scikit-image 库对图像进行基本处理

## scikit-image 库常用函数介绍

1. 图像读取与保存

- io.imread()：读取图像文件。
- io.imsave()：保存图像文件。
- io.imread_collection()：读取多张图像为一个集合。

- io.imshow()：显示图像。

2. 图像转换

- color.rgb2gray()：将彩色图像转换为灰度图像。
- color.gray2rgb()：将灰度图像转换为 RGB 图像。
- color.rgb2hsv()：将 RGB 图像转换为 HSV 图像。
- color.hsv2rgb()：将 HSV 图像转换为 RGB 图像。

3. 几何变换

- transform.resize()：调整图像大小。
- transform.rotate()：旋转图像。
- transform.warp()：进行任意图像变换。
- transform.rescale()：按比例缩放图像。
- transform.swirl()：对图像进行漩涡变换。
- transform.hough_line()：霍夫线检测。

4. 图像过滤

- filters.gaussian()：高斯模糊滤波。
- filters.sobel()：边缘检测滤波器。
- filters.threshold_otsu()：大津阈值分割。
- filters.rank.mean()：局部均值滤波。

5. 图像分割

- segmentation.slic()：使用 SLIC 算法进行超像素分割。
- segmentation.felzenszwalb()：Felzenszwalb 图像分割。
- segmentation.active_contour()：活动轮廓模型。
- segmentation.flood()：洪水填充算法。
- segmentation.watershed()：分水岭算法。

## 2.5 熟悉 OpenCV 库

本节概述了计算机视觉领域核心工具 OpenCV，包含图像读取、基本操作、通道处理、色彩空间转换等。

### 2.5.1 什么是 OpenCV

OpenCV(Open Source Computer Vision Library)是一个开源的计算机视觉和机器学习软件库。它由 Intel 公司于 2000 年推出，旨在为计算机视觉应用提供快速、稳定的工具。OpenCV 提供了丰富的图像处理功能，如图像的读取、处理、分析和保存，还包括许多机器学习算法，可以帮助开发人员构建图像和视频分析应用程序。由于其强大的功能和广泛的应用，

OpenCV 已成为计算机视觉领域的核心工具之一，被广泛应用于各种领域，如自动驾驶、安防监控、医疗影像处理等。

### 2.5.2 OpenCV 的历史与发展

OpenCV 最初是由 Intel 研究院开发的，并于 2000 年首次发布。当时的目标是为研究人员和开发人员提供一个免费的、便于使用的计算机视觉工具库，促进计算机视觉技术的普及和应用。在此后的几年中，OpenCV 经历了多次版本更新和功能扩展，逐渐成为业界标准。

2006 年，OpenCV 1.0 发布，标志着 OpenCV 进入相对成熟的阶段。

2009 年，OpenCV 2.0 发布，引入了 C++接口，使开发更加便捷，同时扩展了许多新的功能和算法。

2015 年，OpenCV 3.0 发布，引入了模块化设计，并大幅提升了性能。随着深度学习的兴起，OpenCV 也增加了对深度学习框架的支持，如 TensorFlow 和 Caffe。

2018 年，OpenCV 4.0 发布，进一步优化了性能，并引入了更多的深度学习功能和硬件加速支持。

如今，OpenCV 已经发展成为一个支持多平台(如 Windows、Linux、macOS、Android 和 iOS)的大型库，拥有超过 2500 个优化的算法，并且被广泛应用于学术研究和工业领域。

### 2.5.3 OpenCV 的应用领域

OpenCV 的应用领域非常广泛，涵盖了多个行业和技术方向。以下是 OpenCV 主要的应用领域。

(1) 自动驾驶。OpenCV 在自动驾驶领域有广泛应用，特别是在图像识别、目标检测、车道检测、物体跟踪等方面，为自动驾驶系统提供了关键的计算机视觉支持。

(2) 安防监控。在安防监控领域，OpenCV 被用来实现人脸识别、行为分析、入侵检测等功能，大大提高了监控系统的智能化水平。

(3) 医疗影像处理。OpenCV 被广泛应用于医疗影像的处理与分析，如 CT、MRI 图像的处理，帮助医生更好地进行诊断和治疗。

(4) 增强现实(AR) 。OpenCV 支持增强现实技术，通过图像识别和处理，实现虚拟信息与现实环境的融合，广泛应用于游戏、教育、广告等领域。

(5) 工业自动化。在工业自动化中，OpenCV 被用来实现机器视觉系统，如产品质量检测、机器人导航、自动化装配等，极大地提高了生产效率和产品质量。

(6) 无人机。无人机技术的发展离不开计算机视觉的支持，OpenCV 在无人机图像处理、目标跟踪、障碍物检测等方面有着重要的应用。

通过这些应用，OpenCV 成为推动计算机视觉技术发展的重要力量，并将在未来继续为更多的创新领域提供支持。

### 2.5.4 OpenCV 安装

(1) 通过 pip 安装。如果使用的是 Python，最简单的方法是通过 pip 安装 OpenCV。这

个方法会安装 OpenCV 的 Python 接口(cv2 模块)。

安装命令为：pip install opencv-python。

如果需要安装包含更多功能的版本(如包括图像和视频编解码器的支持)，可以使用以下命令安装完整版：pip install opencv-python-headless。

(2) 如果使用的是 pycharm 编译器，打开 pycharm，点击文件→项目→python 解析器→+→搜索栏输入 Pillow→安装软件包，如图 2-17 所示。

(3) 安装完成后，通过输入以下代码进行验证。

```
1.  import cv2
2.  # 输出 OpenCV 的版本信息
3.  print("OpenCV 版本:", cv2.__version__)
```

如果安装成功，会提示如图 2-18 所示的信息。

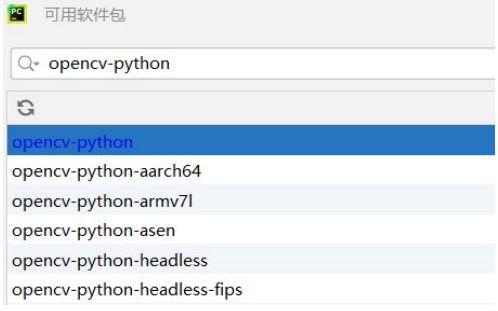



图 2-17　Python 解析器中安装 OpenCV 库




图 2-18　OpenCV 安装成功

## 2.5.5　OpenCV 基础操作

### 1. 图像读取与显示

图像读取与显示是使用 OpenCV 进行图像处理的第一步。通过 cv2.imread()函数可以读取图像文件，并将其存储为 NumPy 数组。通过 cv2.imshow()函数可以在窗口中显示图像。读取时可以指定图像的读取模式，如彩色、灰度或不带透明度的彩色图像。图 2-19 所示是利用 OpenCV 进行基本图像读取，后续都是对图 2-19 进行操作。



图 2-19　OpenCV 读取图像

读取图像的代码为：

```
/*****************************************
    程序名：eg 2.7
    描　述：OpenCV 图像读取与显示
*****************************************/
1.  import cv2
2.
3.  # 读取图像
4.  image = cv2.imread("E:/sztxcl/OpenCV/1.jpg")
5.
6.  # 显示图像
7.  cv2.imshow("Image", image)
8.
9.  # 等待按键输入以关闭窗口
10. cv2.waitKey(0)
11. cv2.destroyAllWindows()
```

通过 OpenCV 可以对图片执行一系列的操作，如图 2-20 所示，利用 OpenCV 把图像缩小到原图的 25%。




图 2-20　利用 OpenCV 对图像进行裁剪、旋转、缩放

(1) 图像裁剪。

① 焦点提取。裁剪图像可以帮助提取图像中的感兴趣区域(Region of Interest，ROI)，忽略不相关的部分。比如，在图像识别中，仅关注目标对象的部分，而不处理背景区域。

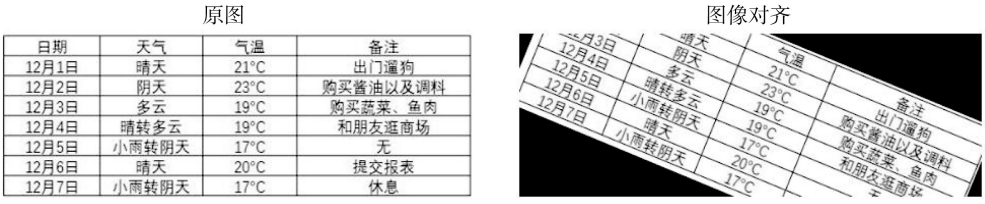
② 数据优化。在处理大尺寸图像时，通过裁剪将图像缩小，可以降低计算成本，提高处理速度。在移动设备或实时处理系统中，这种方法尤为常用。

(2) 图像旋转。

① 图像对齐。旋转图像可以用于对齐图像中的对象。例如，在文档扫描和文本识别中，要确保文本的方向是水平的，这对于 OCR(光学字符识别)非常重要，如图 2-21(a)所示。

② 视角调整。在某些场景中，通过旋转图像可以模拟不同的视角，从而实现更好的视

觉效果，或准备多视角训练数据供机器学习模型使用，如图 2-21(b)所示。




(a) 图像对齐




(b) 视角调整

图 2-21　图像旋转

(3) 图像缩放。

① 分辨率调整。图像缩放用于调整图像的分辨率，以适应不同的显示设备或应用需求。例如，在 Web 开发中，根据屏幕尺寸调整图片大小。

② 数据增强。图像缩放可以作为数据增强的一部分，用于深度学习模型的训练，增加模型对不同大小物体的鲁棒性。

### 2. 图像通道分离与合并

彩色图像通常由多个通道组成，如 RGB 图像包含红、绿、蓝三个通道。可以通过 cv2.split() 函数将图像分离为各个通道，并通过 cv2.merge()函数将其合并为一幅完整的图像。如图 2-22 所示，左上角为一幅完整的图像，剩下三幅图像分别为 RBG 三个通道的颜色。

图 2-22




图 2-22　图像通道分离与合并

以上案例表明图像通道分离与合并在实际应用中有非常重要的意义(尤其是在图像处理和计算机视觉领域)。以下是一些实际意义和应用场景。

(1) 单通道处理。

① 颜色增强或修改。分离图像通道后，可以单独调整某个通道的强度。例如，增强红色通道以突出图像中的红色区域，然后重新合并通道生成调整后的图像。

② 滤波处理。某些图像滤波或增强操作可能仅需要在一个通道上进行。分离通道后，可以在单个通道上执行滤波，然后合并结果。

(2) 特定颜色的检测和分割。

① 颜色检测。通过分离通道，可以分析图像中某种颜色的分布。例如，在 RGB 图像中，如果你想检测蓝色物体，可以仅提取蓝色通道，然后基于该通道进行二值化处理生成掩码。

②对象分割。图像通道分离可以用于基于颜色的图像分割。通过对每个通道进行处理，可以提取出图像中具有特定颜色的对象。

(3) 彩色通道合成。

① 多光谱图像处理。在遥感或医学图像处理中，可能需要将多个光谱或成像通道合并成一幅图像，以便于分析和展示。

② 虚拟通道创建。某些应用场景中，可能需要创建一个虚拟通道来增强图像的某些特征，然后与其他通道合并以生成新的图像。

(4) 数据增强与数据扩充。

在训练深度学习模型时，通道分离与合并可被用作一种数据增强技术。例如，通过交换图像的颜色通道，可以生成新的训练数据，提高模型的泛化能力。

(5) 隐写术与水印。

在图像的某个通道中嵌入信息(如水印或秘密数据)，然后重新合并通道，以创建包含隐藏信息的图像。

(6) 灰度图像转换与处理。

将 RGB 图像转换为灰度图像通常依赖于通道分离后的加权平均操作。通过分析每个通道的权重，可以生成符合要求的灰度图像。

这些实际应用表明，图像通道的分离与合并并不仅仅是一个理论上的操作，而是为解决复杂的图像处理任务提供了灵活和强大的工具。

### 3. 色彩空间转换

色彩空间转换是指将图像从一种色彩空间转换为另一种色彩空间，如从 RGB 转换为灰度图像或 HSV 色彩空间。OpenCV 提供了 cv2.cvtColor()函数用于实现各种色彩空间的转换。利用 OpenCV 进行色彩空间转换的效果如图 2-23 所示。

图 2-23

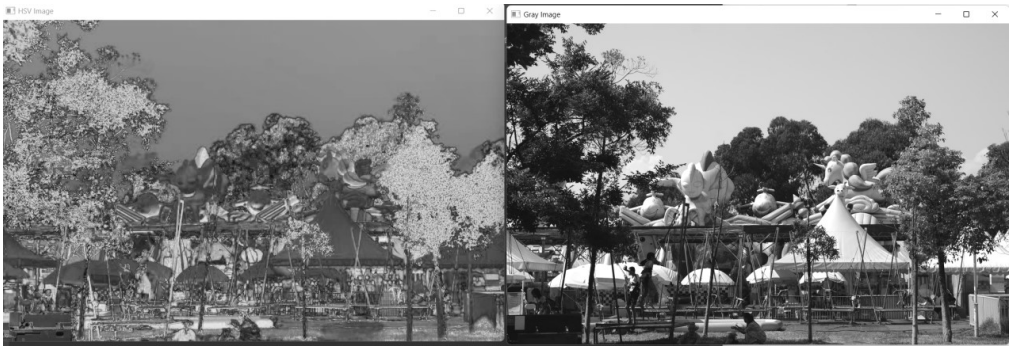



图 2-23　利用 OpenCV 进行色彩空间转换效果

代码示例如下：

```
/****************************************
    程序名：eg 2.8
    描　述：OpenCV 色彩空间转换
****************************************/
import cv2

# 读取图像
image = cv2.imread("E:/sztxcl/OpenCV/1.jpg")

# 转换为灰度图像
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 转换为 HSV 色彩空间
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# 显示结果
cv2.imshow("Gray Image", gray_image)
cv2.imshow("HSV Image", hsv_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

通过以上案例，可以得知色彩空间转换的处理形式。

色彩空间转换还有以下意义。

(1) 灰度图像转换。

减少计算复杂度。灰度图像仅包含亮度信息，没有颜色信息，相较于彩色图像，其数据量更小，计算开销也较低。这对于计算资源有限的系统(如嵌入式设备)非常有用。

图像分析。某些图像处理任务(如边缘检测、图像二值化)只依赖于亮度信息，灰度图

像能更好地突出这些特征，减少不必要的颜色干扰。

(2) HSV 色彩空间转换。

颜色分割与检测。HSV 色彩空间将颜色分为色调、饱和度和亮度三个独立的维度，便于基于颜色的分割和检测。例如，基于色调的分割可以更精确地识别特定颜色的物体，无论其亮度和饱和度如何。

颜色增强。通过 HSV 色彩空间，可以单独调整图像的亮度和饱和度，增强图像的视觉效果，而不影响其色调。

(3) 其他色彩空间。

特定应用需求。一些特定领域(如医学图像处理、卫星遥感)需要使用特定的色彩空间更好地突出图像的某些特征。例如，YUV 色彩空间在视频处理领域应用广泛，便于分离亮度和色度信息。

(4) 综合处理案例：图像预处理与对象检测。

在本案例中，我们将对位于 E:\sztxcl\OpenCV\3.jpg 的图像进行综合处理，包括图像的预处理(灰度化、降噪)、边缘检测、轮廓检测以及对象识别。该案例展示了 OpenCV 在图像预处理和对象检测中的实际应用。

① 图像读取与灰度化。首先，我们读取图像并将其转换为灰度图像，减少颜色信息的干扰，为后续处理打下基础。

代码示例如下：

```
/*****************************************
    程序名：eg 2.9
    描　述：OpenCV 图像灰度化
*****************************************/
1.  import cv2
2.
3.  # 读取图像
4.  image = cv2.imread('E:\\sztxcl\\OpenCV\\4.jpg')
5.
6.  # 转换为灰度图像
7.  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

② 图像降噪。为了减少图像中的噪声，我们应用高斯模糊滤波器，使得图像更加平滑，从而提高边缘检测的效果。

```
1.  blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
```

③ 边缘检测。使用 Canny 边缘检测算法提取图像中的边缘，便于后续的轮廓检测。

```
1.  edges = cv2.Canny(blurred_image, 50, 150)
```

④ 轮廓检测与对象识别。通过边缘检测结果，我们使用 OpenCV 库的 findContours 函

数检测图像中的轮廓，并标注出识别的对象。

代码示例如下：

```
/*****************************************
    程序名：eg 2.10
    描　述：OpenCV 轮廓检测与对象识别
*****************************************/
1.  # 检测轮廓
2.  contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
3.  cv2.CHAIN_APPROX_SIMPLE)
4.
5.  # 绘制轮廓
6.  output_image = image.copy()
7.  cv2.drawContours(output_image, contours, -1, (0, 255, 0), 2)
8.
9.  # 显示结果
10. cv2.imshow('Detected Objects', output_image)
11. cv2.waitKey(0)
12. cv2.destroyAllWindows()
```

此案例展示了如何使用 OpenCV 库从图像读取到对象识别的完整处理流程。我们首先通过灰度化和降噪预处理图像，然后使用边缘检测提取图像中的重要特征，最后通过轮廓检测定位和标注图像中的对象。这一综合处理流程在图像分析、对象检测和计算机视觉应用方面具有广泛的实际意义。

## 2.6 Python 其他的有关图像处理库

本节简述了 Matplotlib、PyTorch 和 TensorFlow 等库在图像处理中的作用，涵盖数据可视化、深度学习应用及模型训练，适用于处理图像数据及构建深度学习模型。

### 2.6.1 Matplotlib 库

#### 1. Matplotlib 库简介

Matplotlib 库是一个广泛使用的 Python 2D 绘图库，主要用于生成各种图表和可视化数据。它的主要作用包括以下几点。

(1) 数据可视化。Matplotlib 库能够生成多种类型的图表，如折线图、柱状图、散点图、饼图、直方图等，帮助用户将数据以直观的方式展现。

(2) 支持多种文件格式。绘制的图表可以导出多种常见的图像格式，如 PNG、JPEG、SVG、PDF 等，方便在不同平台上展示。

(3) 与科学计算库的集成。Matplotlib 与 NumPy、Pandas 等科学计算库高度集成，能够非常方便地处理和可视化大型数据集。

(4) 自定义图表。用户可以根据需要对图表进行高度自定义，如设置图例、坐标轴标签、标题、颜色、线型、标记等细节，满足不同场景下的需求。

(5) 交互式图形界面。支持交互式的图形界面，用户可以对生成的图表进行缩放、平移、保存等操作，这对于数据探索和分析非常有帮助。

(6) 嵌入其他应用。可以将生成的图形嵌入 GUI 应用程序中，如 Tkinter、PyQt、GTK、wxPython 等，方便开发可视化界面应用。

### 2. Matplotlib 库安装方法

方法一：在 Pycharm 编译器终端或者 Windows 命令提示符中输入“pip install matplotlib”。

方法二：打开 Pycharm，点击文件→项目→Python 解析器→→搜索栏输入 matplotlib→安装软件包。

### 3. Matplotlib 库的具体应用

下面利用一个例子，展示利用 Matplotlib 库生成包含折线图、柱状图和饼图的综合图表，并进行自定义样式设置和子图布局管理。

① 代码示例如下：

```
/*****************************************
    程序名：eg 2.11
    描　述：利用 Matplotlib 库生成图表
*****************************************/
1.  import matplotlib.pyplot as plt
2.
3.  # 创建数据
4.  x = [1, 2, 3, 4, 5]
5.  y1 = [10, 14, 20, 25, 30]  # 折线图数据
6.  y2 = [5, 9, 13, 17, 21]  # 柱状图数据
7.  labels = ['A', 'B', 'C', 'D', 'E']  # 饼图标签
8.  sizes = [15, 25, 35, 10, 15]  # 饼图数据
9.
10. # 创建一个 2x2 布局的图表
11. fig, axs = plt.subplots(2, 2, figsize=(10, 8))
12.
13. # 子图 1：折线图
14. axs[0, 0].plot(x, y1, color='blue', marker='o', linestyle='-', linewidth=2)
15. axs[0, 0].set_title('Line Chart')
16. axs[0, 0].set_xlabel('X-axis')
```

```
17.  axs[0, 0].set_ylabel('Y1-axis')
18.
19.  # 子图 2：柱状图
20.  axs[0, 1].bar(x, y2, color='orange')
21.  axs[0, 1].set_title('Bar Chart')
22.  axs[0, 1].set_xlabel('X-axis')
23.  axs[0, 1].set_ylabel('Y2-axis')
24.
25.  # 子图 3：饼图
26.  axs[1, 0].pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
27.  axs[1, 0].set_title('Pie Chart')
28.
29.  # 子图 4：混合图表(折线图和柱状图)
30.  axs[1, 1].plot(x, y1, color='blue', marker='o', label='Line Data')
31.  axs[1, 1].bar(x, y2, color='green', alpha=0.5, label='Bar Data')
32.  axs[1, 1].set_title('Combined Line and Bar Chart')
33.  axs[1, 1].set_xlabel('X-axis')
34.  axs[1, 1].set_ylabel('Y-axis')
35.  axs[1, 1].legend()
36.
37.  # 调整子图间的距离
38.  plt.tight_layout()
39.
40.  # 显示图表
41.  plt.show()
```

② 代码解释。

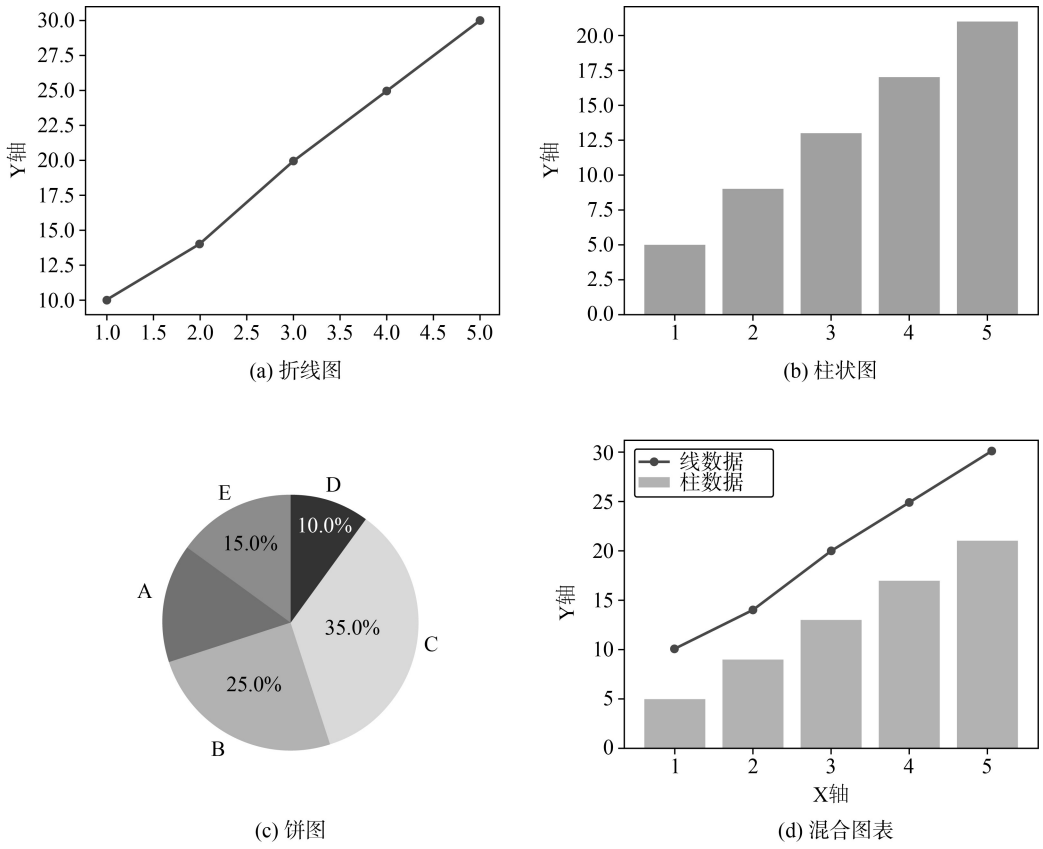
执行完以上代码，可以利用 Matplotlib 库绘制相对应的图片，如图 2-24 所示。

折线图。图 2-24(a)展示了一条简单的折线图，设置了蓝色线条、圆形标记点，并调整了线条宽度。

柱状图。图 2-24(b)展示了柱状图，颜色为橙色。

饼图。图 2-24(c)展示了饼图，标签显示了不同分类，并在饼图中标注了各个部分的百分比。

混合图表。图 2-24(d)展示了折线图和柱状图的组合，在同一张图中展示两种数据类型，使用不同的颜色和透明度区分数据。




(a) 折线图　(b) 柱状图

(c) 饼图　(d) 混合图表

图 2-24　利用 Matplotlib 库绘制图片

通过以上代码，将在一个窗口中展示 4 个不同类型的子图。该案例展示了 Matplotlib 库的灵活性，以及它如何结合不同的图表类型进行数据的可视化表达。

## 2.6.2　PyTorch 库

PyTorch 库是一个开源的深度学习框架，由 Facebook AI Research 开发，广泛用于研究和生产环境中。它以灵活的动态计算图(dynamic computational graph)为特色，支持高效的张量计算、自动微分以及强大的 GPU 加速，使其成为深度学习和神经网络模型开发的首选工具之一。PyTorch 库非常适合研究和开发新的神经网络模型，且在可扩展性和调试便利性方面表现出色。

### 1. PyTorch 库的主要功能

(1) 张量运算。PyTorch 库的核心是 Tensor，类似于 NumPy 库的 ndarray，但支持 GPU 加速。用户可以使用 PyTorch 库轻松进行矩阵运算和线性代数计算。

(2) 自动微分。PyTorch 库内置的自动求导引擎(Autograd)能够通过记录操作构建动态计算图，然后通过反向传播自动计算梯度。它为模型优化提供了基础。

(3) 神经网络模块。PyTorch 库提供了一个模块化的神经网络库 torch.nn，可以帮助用户轻松构建和训练神经网络模型。常用的层、激活函数、损失函数、优化器等都可以直接

使用。

(4) GPU 加速。PyTorch 库支持通过 CUDA 进行 GPU 加速，用户可以很容易地在 CPU 和 GPU 之间切换计算资源，并利用 GPU 进行高效的并行计算。

(5) 动态计算图。与 TensorFlow 等库的静态计算图不同，PyTorch 库采用了动态计算图的方式，允许用户在运行时修改计算图，这使得调试更加灵活。

(6) 强大的生态系统。PyTorch 库拥有丰富的生态系统，支持包括计算机视觉、自然语言处理、强化学习等领域。常见库如 torchvision(计算机视觉)、torchtext(文本处理)、torchaudio(音频处理)、Detectron2(对象检测)等都可以无缝集成。

(7) 社区支持与应用。PyTorch 库得到了广泛的社区支持，被顶级研究机构和公司用于开发创新的 AI 应用。它常被用来开发前沿的深度学习模型，特别是在计算机视觉、自然语言处理、生成对抗网络(GANs)、强化学习等领域。

#### 2. PyTorch 的安装方法

利用 Window 终端或者 Pycharm 终端输入：

```
1.  pip install torch torchvision torchaudio
```

### 2.6.3 TensorFlow 库

#### 1. TensorFlow 库简介

TensorFlow 库是一个开源的机器学习框架，由谷歌开发，旨在高效地构建和训练深度学习模型。其核心数据结构是张量(多维数组)，通过计算图来表示复杂的计算过程。计算图中的节点表示操作，边表示数据流动，使得并行计算和资源优化变得可行。

在工作流程中，用户首先定义模型结构和损失函数，然后通过优化算法(如梯度下降)进行训练，不断更新模型参数。TensorFlow 库利用自动微分技术简化了梯度计算的过程，支持多种优化算法来提高模型的性能和准确性。

TensorFlow 库拥有丰富的生态系统，包括 Keras(高层 API)、TensorBoard(可视化工具)和 TensorFlow Lite(轻量级部署工具)，广泛应用于计算机视觉、自然语言处理和强化学习等领域。其灵活性和可扩展性使其成为研究和生产环境中热门的选择。

#### 2. TensorFlow 库在数字图像处理中的应用

(1) 图像分类。TensorFlow 库可以用于训练深度学习模型进行图像分类任务，如识别猫、狗或其他对象。通过卷积神经网络(Convolutional Neural Networks，CNN)，模型能够自动提取图像中的特征，从而提高分类准确性。

(2) 目标检测。目标检测技术能够识别图像中的多个对象并定位其位置。使用 TensorFlow 库，开发者可以实现如 YOLO(You Only Look Once)或 Faster R-CNN 等模型，应用于监控、自动驾驶等领域。

(3) 图像分割。图像分割是将图像分成多个区域的过程，常用于医学影像分析和自动驾驶。TensorFlow 库支持 U-Net 和 Mask R-CNN 等模型，能够精确分割出感兴趣的区域或对象。

(4) 图像生成。TensorFlow 库还可用于生成新的图像，如使用生成对抗网络。这类技术

可以应用于艺术创作、虚拟现实等领域，生成逼真的图像或艺术风格的转换。

(5) 图像增强。通过应用各种图像处理技术(如旋转、缩放、颜色调整等)，TensorFlow 库可以用于数据增强，以提高模型在训练过程中的鲁棒性。

### 3. 安装方法

在 Pycharm 编译器终端或者 Windows 命令提示符中输入“pip install tensorflow”，安装完成后输入以下代码验证：

```
1.  import tensorflow as tf
2.  print(tf.__version__)
```

【扩展阅读】
Python 在数字图像处理领域的应用潜力



## 2.7 思考练习

### 一、选择题

1. Pillow 库的前身是以下哪个库？(　　)
   A. OpenCV　　B. Scikit-Image
   C. NumPy　　D. PIL
2. NumPy 库的核心数据结构是什么？(　　)
   A. DataFrame　　B. ArrayList
   C. ndarray　　D. Tensor
3. 在 scikit-image 库中，以下哪项不属于形态学操作？(　　)
   A. 膨胀　　B. 腐蚀
   C. 开闭操作　　D. 旋转
4. OpenCV 的初始版本由哪家公司开发？(　　)
   A. Google　　B. Intel
   C. Microsoft　　D. IBM
5. 以下哪种方法可以在 OpenCV 库中将彩色图像转换为灰度图像？(　　)
   A. cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   B. cv2.convertToGray(img)
   C. cv2.toGray(img)
   D. cv2.grayConvert(img)

## 二、填空题

1. Pillow 库是 Python 中用于处理图像的库，它的主要功能包括图像的加载、保存、转换、_____、滤镜应用等操作。

2. NumPy 库的核心数据结构是_____，即_____，它支持高效的多维数组操作。

3. scikit-image 库可以进行图像分割操作，常用的方法有基于阈值的分割、_____、聚类方法、_____等。

4. OpenCV 库支持的主要色彩空间转换操作包括RGB到灰度图像的转换和RGB 到_____色彩空间的转换。

5. 在 NumPy 库中，数组操作可以通过_____函数进行切片、索引、连接等操作。

## 三、判断题

1. Pillow 库只能处理 JPEG 格式的图像文件。 (　　)

2. NumPy 库可以进行快速傅里叶变换，用于信号处理等应用。 (　　)

3. scikit-image 库是基于 TensorFlow 库构建的，用于图像处理。 (　　)

4. OpenCV 库的 cv2.imread()函数用于读取图像文件，并将其存储为 NumPy 数组。 (　　)

5. Pillow 库不支持透明度处理和图像合成。 (　　)

## 四、简答题

1. 简述 Pillow 库在图像处理中常用的几种操作，并举例说明其应用。

2. NumPy 库在数字图像处理中的作用是什么？请举例说明。

3. 简述 OpenCV 库的应用领域，并列举至少两个实际应用。

## 五、综合运用题

1. 使用 Pillow 库编写代码，读取一张图像，将其旋转 90°并保存为新的文件格式，然后对该图像应用高斯模糊滤镜，并保存处理后的图像。

2. 利用 OpenCV 库编写代码，读取一张图像，将其转换为灰度图像，进行边缘检测，并将检测结果显示出来。