第1章 绪 论

➢ **内容提要**

本章介绍数字图像的基本概念、数字图像的离散化表示、数字图像处理及软件环境配置。

➢ **知识要点**

- ✧ 图像、数字图像、像素及数字视频的定义。
- ✧ MATLAB 环境配置。
- ✧ 最常用的几个 MATLAB 函数命令。
- ✧ 内联函数与匿名函数。
- ✧ MATLAB GPU 编程基础。
- ✧ 数字图像处理所研究的主要内容。

➢ **教学建议**

- ✧ 本章教学安排建议 6 课时左右。
- ✧ 先修知识主要有线性代数、数学实验(MATLAB)、计算机基础等。
- ✧ MATLAB 常用函数是全书编程的基本工具,用 2 课时左右进行复习巩固。
- ✧ 内联函数与匿名函数用 2 课时左右进行练习。
- ✧ 考虑到具体实验环境,MATLAB GPU 编程基础部分可选学,用 2 课时左右。

1.1 基 本 概 念

人们对图像是很熟悉的,然而想对其给出一个科学的定义却不简单[1],诸多研究者一直进行着这方面的尝试。一个关于图像的最古老的定义是柏拉图提出的,他说:“我们首先把影子称为图像,然后把人们在水中或在模糊的、光滑的和闪亮的物体表面看到的反光和所有相似的再现看成图像”[2];图像是“对物体的表达、表象、模仿,一个生动的视觉描述,为了表达其他事物而引入的事物”[3];图像是用各种观测系统以不同形式和手段观测客观世界而获得的,可以直接或间接作用于人眼并进而产生视觉和知觉的实体[4];图像是对客观存在物体的一种相似性的生动模仿与描述,是物体的一种不完全的、不精确的,但在某种意义下是适当的表示[5-7]。因此,使用“图象”或许比“图像”更为合理[4]。

图像直观、丰富的特点决定了其可承载大量信息,故有“一图胜千言”“百闻不如一见”等说法。图 1-1 所示为一幅风景图像①,从图中可以看到天空、白云、水面、帆船等景色。尽管这幅图像展现的内容很有限,想把它们全部用文字描述清楚却十分困难。

---

① 除特殊说明,本书所用图像均来自 MATLAB 图像处理工具箱或南加利福尼亚大学的 USC-SIPI image database 图像库(http://sipi.usc.edu/database/)。

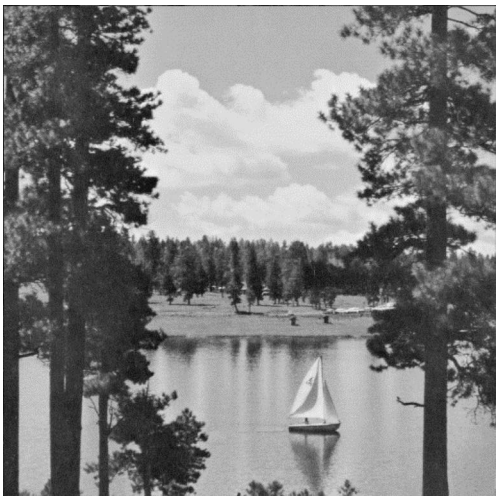


图 1-1　风景图像

根据图像的表达方式,可将其再细分为物理图像、数字图像、数字视频等。

物理图像是指物质或能量的实际分布[6]。根据是否能被人眼所感知,又可把物理图像分为可见图像与不可见图像。可见图像是能为人类视觉所感知的通常意义下的图像。例如,光学图像的光谱波段就能被人的肉眼所感知,属于可见图像。在医学诊断中使用的伽马射线波段在可见光之外,其原始的能量分布属于不可见图像。最终的医学图像是将上述不可见的物理量通过可视化手段转换成人眼能够方便识别的图像形式。

数字图像是用数字阵列来表示的图像。数字阵列中的元素称为像素。

数字视频是连续播放的数字图像序列。与单幅图像相比,数字视频可看成多幅数字图像在时间轴的叠加,其中每幅图像称为数字视频的一帧。图 1-2 是一个数字视频中连续的 4 帧。为了使视频播放保持视觉上的连续性,帧与帧之间往往具有很大的信息冗余。

图 1-2　数字视频的帧序列示例

由于数字视频的每一帧都是静态图像,因此一些数字图像处理方法很容易应用到数字视频的处理领域。

扩展阅读

视觉暂留:视觉暂留是指人眼在观察物体时,光信号传入大脑神经需要一定时间(约 1/24 秒),即使物体消失,视觉印象仍会短暂保留的现象。这一生理特性是动画、电影、LED 显示等技术的基础。

其实，早在公元前 4 世纪，古希腊哲学家亚里士多德就观察并记录了运动物体在视觉中产生暂留的现象。中国宋朝(10—13 世纪)的走马灯则是将这一现象转化为实用技术的成功案例。1824 年，英国科学家彼得·马克·罗杰特首次对这一现象进行科学定义和系统阐述，奠定了理论基础。1829 年，比利时物理学家约瑟夫·普拉托发明费纳奇镜，首次将视觉暂留理论转化为可重复验证的科学实验装置。1895 年法国卢米埃尔兄弟基于视觉暂留原理发明电影放映机，由此开创了现代影视时代。这一历程完整展现了“现象观察→经验技术→理论科学→工业应用”的科技创新典型路径。

标准图像：标准图像是经过学术界和工业界严格筛选的基准测试图像，主要用于算法的基础性能评估。最具代表性的包括 Lenna(纹理与细节分析)、Peppers(色彩基准)、Baboon(高频纹理测试)和 Cameraman(灰度基准)等。这些图像因其独特的视觉特征(如丰富的纹理、均衡的色调分布和清晰的边缘信息)，成为评估图像处理算法(如压缩、滤波和增强)的黄金标准。

标准图像集：标准图像集主要指结构化、大规模且带有标注的图像集合，支撑着计算机视觉各领域的研究。一些经典的标准数据集包括 MNIST(手写数字识别)、CIFAR-10(基础物体分类)、ImageNet(大规模图像分类)、PASCAL VOC(通用目标检测)、COCO(复杂场景理解)和 Cityscapes(自动驾驶场景解析)。这些数据集不仅提供标准化评估基准，其构建方法(如 ImageNet 的 WordNet 体系、COCO 的密集标注策略)更成为后续研究的范本，持续推动着计算机视觉技术的发展。随着技术发展，研究者也在持续创建新的专业数据集(如医疗影像领域的 CheXpert、自动驾驶领域的 Waymo Open Dataset)，推动计算机视觉技术向更专业化、精细化方向发展。在实际研究中，根据具体需求选择合适的基准数据集或构建新数据集，是确保研究质量的重要环节。

1.2 MATLAB 基础

MATLAB 是“Matrix Laboratory”两个词的缩写组合，意为“矩阵实验室”。是由美国 Mathworks 公司发布的主要面对科学计算、可视化以及交互式程序设计的软件系统。除基本模块外，MATLAB 还提供了许多可选工具箱，如图像处理工具箱、优化工具箱、统计工具箱、控制系统工具箱等。工具箱的使用能够大大提高用户在具体领域的编程效率，从而节省出更多的时间用于创新思维。有人把 MATLAB 称为继机器语言、汇编语言、高级语言之后的“第四代计算机编程语言”。

1.2.1 MATLAB 工具箱安装

MATLAB 中可以添加大量工具箱，这些工具箱可以是 MATLAB 官方提供的专业工具箱，也可以是第三方的共享代码。以图像处理软件包 DIPUM[8] 为例，介绍 MATLAB 工具箱的安装步骤：

第一步，将解压后的工具箱复制到 MATLAB 安装目录下的 toolbox 子目录。

第二步，在 MATLAB 运行界面的 HOME 选项卡中单击 Set Path 快捷按钮，打开 Set Path 窗口，如图 1-3 所示。

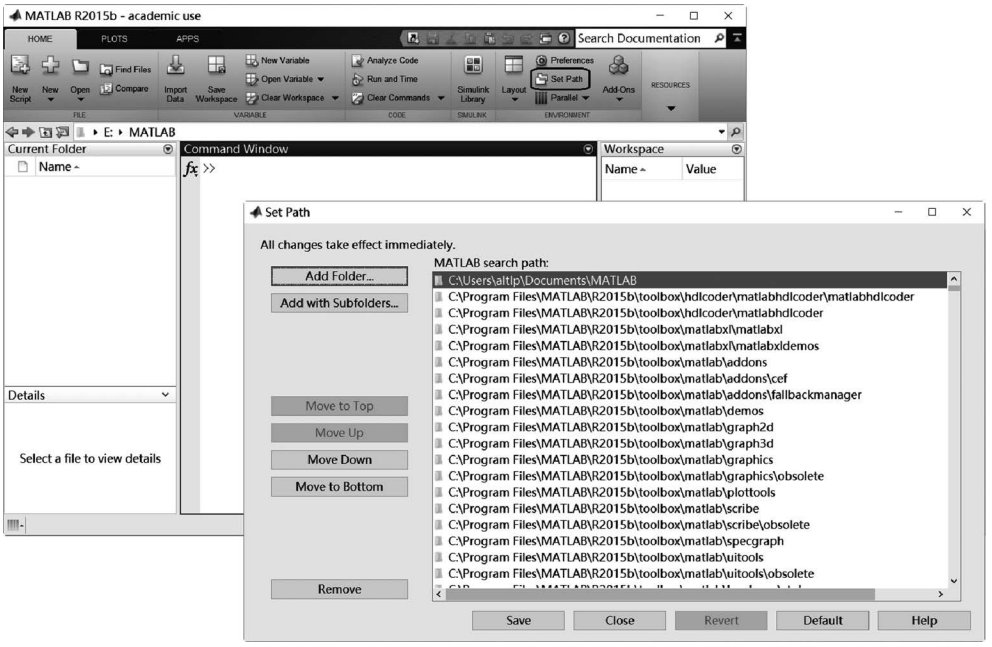


图 1-3　MATLAB 工具箱安装示例第二步

第三步,单击 Set Path 窗口中的 Add Folder 按钮,选择 DIPUM 所对应的目录,如图 1-4 所示。

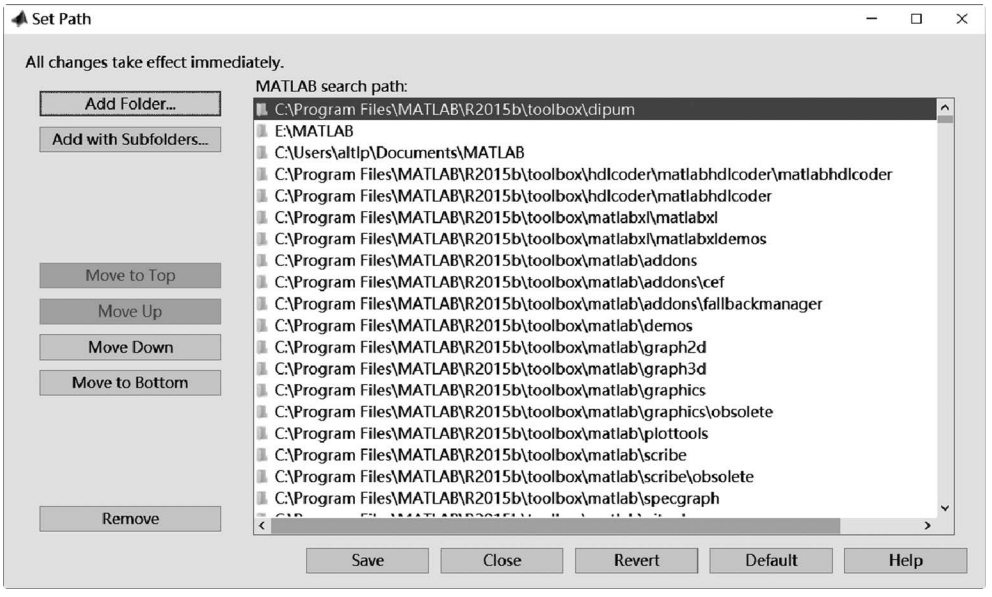


图 1-4　MATLAB 工具箱安装示例第三步

第四步,依次单击上面窗口中的 Save 和 Close 按钮,完成 DIPUM 工具箱的安装。

1.2.2 几个常用的 MATLAB 命令

为方便后续内容的程序调试，几个常用的 MATLAB 命令列举如下：

“help 命令名称/程序名称”用来在命令窗口中显示相应 MATLAB 命令或文件的帮助信息。

“doc 命令名称/程序名称”用来在帮助窗口中以网页形式获取 MATLAB 命令或 M 程序的帮助信息，显示风格较“help”更为丰富和灵活。若在命令窗口中只输入“doc”，则直接打开帮助网页，操作十分方便。

“type 程序名称”用于在命令窗口中显示 MATLAB 程序内容。

“edit 程序名称”用于在 MATLAB 程序编辑器中打开相应程序以进行编辑。

“pcode 程序名称”用于将 MATLAB 和 M 程序转换为扩展名为.p 的加密格式。

“clc”命令用于清除命令窗口所显示的内容。

“clear”命令用于清除内存变量。

“cat”命令用于按指定方向连接矩阵。

“save”和“load”命令则分别用于将内存变量放入外存和将放入外存的内存变量重新读入内存。

“tic”和“toc”命令，用来记录 MATLAB 命令执行的时间。常用“tic”来保存当前时间，而后使用“toc”来记录程序完成时间。

上面命令的具体实例可通过 doc 函数获取，建议读者结合实例，加深对这些命令的掌握。

1.2.3 MATLAB 的内联函数与匿名函数

此处对 MATLAB 的内联函数和匿名函数实现方法进行介绍，MATLAB 引入这两个函数主要是为了减少程序访问外存时间，提高运算效率。

1. 内联函数

内联(inline)函数是 MATLAB 7.0 以前经常使用的一种构造函数对象的方法。在命令窗口、程序或函数中创建局部函数时，通过使用 inline 构造函数，而不用将其存储为一个 M 文件，同时又可以像使用一般函数那样调用它。常用语法为

```
fhandle = inline(expr,arg1,arg2,…)
```

其中，fhandle 是调用该函数的函数句柄；expr 为函数表达式；arg$i$ 为函数参数。由于内联函数是存储于内存中而不是在 M 文件中，因此省去了文件访问的时间，加快了程序的运行效率。

需注意，内联函数在使用过程中还会受到一些制约。首先，不能在内联函数中调用另一个 inline 函数；其次，只能由一个 MATLAB 表达式组成，并且只能返回一个变量。

2. 匿名函数

匿名函数(anonymous function)是 MATLAB 7.0 提出的一种全新的函数描述形式，和内联函数类似，可以让用户编写简单的函数而不需要创建 M 文件，匿名函数具有 inline 函数的所有优点，并且还具有一些独有特点，运行效率也比 inline 函数高。定义一个匿名函数

常用语法是

```
fhandle = @(arglist) expression
```


其中,fhandle 依然是相应的函数句柄; arglist 为参数列表。

例 1-1 分别采用内联函数和匿名函数定义:

$$f(x,y)=x^2+y^2$$

并测试运行。

```
>> f1 = inline('x^2 + y^2', 'x', 'y'); ①
>> f1(2,3)
ans =
    13
>> f2 = @(x,y) x^2 + y^2;
>> f2(2,3)
ans =
    13
```

匿名函数会让前面的内联函数逐步退出 MATLAB 舞台,事实上在设计这种类型的函数时就带有这一目的,目前保留内联函数无疑会使程序具有更好的向下兼容性。

1.2.4 MATLAB GPU 编程基础

GPU 英文全称为 Graphic Processing Unit,中文翻译为“图形处理器”或“图像处理单元”。与 CPU 相比,GPU 更适合大规模并发计算,在处理图像时往往具有更高的工作效率。

从 MATLAB 2013 版本开始,MATLAB 可以直接调用 GPU 进行并行计算。只要数据是 gpuArray 格式的,许多 MATLAB 内置函数就会自动地调用 GPU 进行计算。下面介绍 MATLAB 的 GPU 编程时的几个基础函数。

1. GPU 设备确认

(1) gpuDeviceCount 函数用于获取当前设备上的 GPU 数目,语法格式为

```
n = gpuDeviceCount
```

(2) gpuDevice 函数用于选择 GPU 设备,语法格式为

```
D = gpuDevice or gpuDevice()
```

如果当前还未设置选择的 GPU,则选择默认的 GPU,D 是返回对象; 如果已经设置了 GPU,则返回设置的 GPU 对象。

```
D = gpuDevice(IDX):
```

表示选择 IDX 对应的 GPU 设置(分别用 1 和 2 等表示),D 依然为返回对象。

(3) reset 函数用于清空 GPU 内存,语法格式为

```
reset(gpudev)
```

---

① 也可忽略参数而写成 f1=inline('x^2+y^2');

与 MATLAB 的 clear 功能类似，gpudev 是 gpuDevice 返回的对象。

例 1-2 获取当前计算机的 GPU 数目，指定 GPU 设备，并练习清空所指定 GPU 的内存。

```
>> n = gpuDeviceCount
>> D = gpuDevice
>> reset(D)
```

2. GPU 与 CPU 之间的数据交互

(1) gpuArray 用于将 CPU 内存数据传到 GPU 内存中，语法格式为

```
GX = gpuArray(X)
```

其中，X 为 CPU 内存变量；GX 为得到的 GPU 内存变量。

(2) gather 函数用于将 GPU 内存数据传到 CPU 内存中，语法格式为

```
X = gather(GX)
```

其中，GX 为 GPU 内存变量；X 为得到的 CPU 内存变量。

例 1-3 编程实现 CPU 与 GPU 之间内存变量的交互。

```
X = rand(10,'single');    % 定义在 CPU 上的一个 10×10 的随机初始化数组
GX = gpuArray(X);         % 在 GPU 初始化数组 GX,并且将 X 的值赋给 GX
GX2 = GX. * GX;           % GPU 上执行数组对应位置的点乘
CX2 = gather(GX2);        % 将 GPU 上的运算结果 GX2 返回到 CPU 的 CX2
```

例 1-4 编程比较 CPU 与 GPU 对大规模计算的运行效率。

```
A1 = rand(12000,400);
B1 = rand(400,12000);
tic
  f1 = sum(A1.'. * B1, 1);
tCPU = toc

A2 = rand(12000,400,'gpuArray');
B2 = rand(400,12000,'gpuArray');
tic
  f2 = sum(A2.'. * B2, 1);
tGPU = toc
CGTime = tCPU/tGPU
```

上面程序的运行结果为

```
tCPU =
    0.0414
tGPU =
    3.0278e-04
CGTime =
    136.6621
```

由程序运行结果可以看出，针对这一计算任务，在实验所用的计算机上，GPU 的运算速度达到 CPU 的 130 多倍。另外需要注意，本程序在调用随机数函数 rand 时使用“gpuArray”

参数,直接生成了 GPU 内存变量。

GNU Octave:GNU Octave 是一款免费开源的数值计算软件,语法与 MATLAB 高度兼容,支持矩阵运算、数据可视化和算法开发。它提供了类似 MATLAB 的编程环境,包含数百个内置数学函数,能直接运行大多数 MATLAB 脚本(.m 文件),是学术研究和工程计算的经济替代方案。

CUDA:CUDA 是一个由 NVIDIA 公司推出的 GPU 并行计算平台,本质上是一套连接软件与 GPU 硬件的编程接口和生态系统。它能与一些 AI 框架(PyTorch/TensorFlow)、科学软件(MATLAB)等深度集成,为开发者提供调用 GPU 计算核心的标准方法,既隐藏了硬件细节,又保留了并行优化能力。

1.3 本书内容

数字图像处理就是用计算机处理数字图像,其着重强调在图像之间进行变换,即处理的对象和结果均为图像。

本书将介绍数字图像处理的一些基本内容,主要包括图像的基本操作、图像的基本运算、图像变换、图像的形态学操作、图像增强、图像去噪、图像分割等。

图像的基本操作,主要是图像的读取、显示与存储操作。

图像的基本运算,包括图像的代数运算和几何变换两部分。

图像变换是指通过数学映射的方法,将空域中的信息转换到变换域,如频域、时频域等。通过对图像变换域信息进行处理,再进行逆变换,最终获取处理后的图像。

图像的形态学操作是指采用数学形态学的仿生方法对图像进行处理。

图像增强是为了改善图像的视觉效果,突出图像中有用的部分(即所感兴趣的部分)。

图像去噪的目的是将退化图像原有信息进行恢复,使之更接近实际。

需要注意,图像增强与图像去噪的目的不同,前者是为了满足用户的主观偏爱,而后者则倾向于逼近客观真相。

图像分割则是将一幅图像分成若干有意义或感兴趣区域的过程。通过图像分割,有利于将图像中有意义的特征部分提取出来,而这些特征是进一步进行图像分析和理解的基础[7]。

由于篇幅所限,本书主要对图像处理的最基本方法进行介绍。相信通过对本书的学习,读者在进行相关文献阅读及解决实际问题时,能有一个较为清晰的问题背景及物理直觉,有利于对问题的正确分析和最终解决。

图像处理、图像分析与图像理解:图像处理聚焦像素级操作(如去噪、锐化),通过算法直接修改图像数据,实现从图像到图像的转换;图像分析在此基础上提取结构化信息(如目标检测、特征分类),实现从图像到结构化数据的转换;而图像理解则进一步融合场景上下文和语义推理(如行为识别、关系判断),实现从图像到人类可解释的高层描述/决的策转换。

三者呈递进关系：图像处理是基础，为分析提供高质量输入；图像分析生成的中层表征支撑理解任务；而理解层的反馈又可优化处理和分析策略。随着深度学习发展，传统界限逐渐模糊，端到端模型已能直接实现“像素→语义”的跨层次映射，但三者分工仍可为系统设计提供重要理论框架。

本章实验

实验一　MATLAB 工具箱安装

一、实验目的

掌握 MATLAB 工具箱的安装。

二、实验原理

MATLAB 工具箱安装，详见 1.2.1 节。

三、实验内容

完成网上电子资源教辅材料中所给 MyTool 工具箱的安装。

四、实验报告要求

(1) 描述必要的实验原理和基本步骤。

(2) 用数据和图片给出各个步骤中取得的实验结果。

(3) 验证工具箱是否可用。

实验二　MATLAB 基本操作

一、实验目的

(1) 熟悉常用 MATLAB 基础命令。

(2) 掌握匿名函数与内联函数的使用。

二、实验原理

MATLAB 基本操作，详见 1.2.2 节与 1.2.3 节。

三、实验内容

(1) 练习 1.2.2 节中几个常用的 MATLAB 命令。

(2) 分别使用内联函数、匿名函数和 MATLAB 外部函数文件完成一个函数的编程并测试运行。

四、实验报告要求

(1) 描述必要的实验原理和基本步骤。

(2) 用截图给出各个步骤中取得的实验结果。

(3) 比较三种函数对同一算法的运行效率(时间)。

实验三　经典图像数据集

一、实验目的

了解一些经典数据集及申请下载方式。

二、实验原理

参考1.1节“基本概念”的扩展阅读的相关内容。

三、实验内容

(1) 了解1～2个经典数据集。

(2) 尝试下载1～2个经典数据集。

四、实验报告要求

(1) 对数据集进行简要介绍。

(2) 对数据集的了解及下载过程通过截图配合说明。

第 2 章 图像的基本操作

➢ **内容提要**

本章介绍 MATLAB 环境中数字图像的基本操作，包括数字图像的离散化表示、图像的读取、显示、存储(写)操作，以及图像的邻域操作、块操作。

➢ **知识要点**

- ✧ 图像的数值矩阵模型，包括灰度图像、二值图像及彩色图像。
- ✧ 图像的读取、显示及存储操作。
- ✧ 图像的邻域操作与块操作的具体实现。

➢ **教学建议**

- ✧ 本章教学安排建议 6 课时左右。
- ✧ 图像数值矩阵模型的理解，图像的读、写、显示等基本操作共 2 课时。
- ✧ 图像的邻域操作与块操作对后续章节中的滤波器设计十分重要，用 2 课时进行实践练习。
- ✧ 2 课时左右的时间专门用于实验及讲评。

2.1 数字图像的离散化表示

如第 1 章所述，数字图像是用数字阵列来表示的，具体到不同的图像类型，这些数字阵列也具有不同形式。

2.1.1 灰度图像

对一幅分辨率为 $M\times N$ 的灰度图像 $\boldsymbol{I}$，在 MATLAB 中是以二维矩阵的形式来表示的：

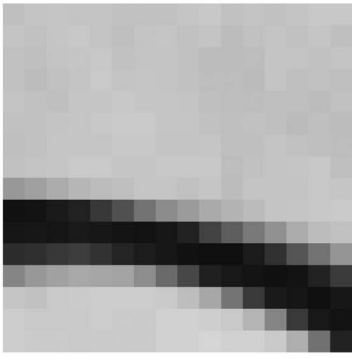
$$\boldsymbol{I}=\begin{bmatrix} I(1,1) & I(1,2) & \cdots & I(1,N) \\ I(2,1) & I(2,2) & \cdots & I(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ I(M,1) & I(M,2) & \cdots & I(M,N) \end{bmatrix} \tag{2.1}$$

矩阵中的每个元素代表一个像素，该矩阵也称为像素矩阵。元素值越大，对应像素越亮。当数字矩阵中元素的取值只有两种(常为 0 或 1)时，相应灰度图像被称为二值图像。

需注意的是，在编程时 MATALB 中矩阵元素的下标约定从(1,1)开始，不同于 C 语言中下标从(0,0)开始的约定方式。

图 2-1(a)是大小为 256×256 的测试图像，图 2-1(b)是取图 2-1(a)中的钟表提手部分并进行了 16 倍的放大显示，图 2-1(c)是 MATLAB 中与图 2-1(b)相对应的数值矩阵。

(a) 数字图像



(b) 钟表提手部分16×16子图

223	225	226	225	225	224	224	224	226	227	224	225	224	225	227	227
225	224	226	227	225	223	225	226	224	225	221	224	225	224	226	223
224	223	225	226	227	224	225	225	224	221	221	222	225	225	224	225
224	221	224	225	228	225	226	225	224	222	222	220	225	223	224	223
225	224	223	226	226	227	226	226	225	223	222	222	223	223	221	224
224	224	225	226	227	228	228	226	226	224	223	224	223	220	223	221
223	224	226	226	226	227	227	225	225	226	223	223	225	224	223	222
224	225	226	228	228	226	226	227	227	227	222	224	225	226	227	223
200	205	212	218	221	223	226	226	224	226	221	225	225	225	227	226
69	71	84	96	123	147	175	192	202	211	216	220	225	226	227	228
76	79	77	83	80	74	70	78	96	134	159	179	195	212	221	224
111	114	123	130	118	100	88	78	69	63	63	86	115	151	175	198
188	200	206	210	208	207	193	168	138	93	81	73	65	72	97	127
226	224	228	229	230	230	230	228	222	209	177	125	85	77	63	78
230	228	228	228	230	229	229	232	232	231	228	217	187	125	81	71
230	229	229	228	229	229	229	232	232	234	232	232	228	213	174	97

(c) 子图像在MATLAB中的矩阵表示

图 2-1　灰度图像在 MATLAB 中的离散化表示示例

2.1.2　彩色图像

RGB 图像是一种最常见的彩色图像格式，一幅 RGB 图像就是彩色像素的一个 $M\times N\times 3$ 数组，与灰度图像类似，$M\times N$ 为像素的多少，即分辨率。如图 2-2 所示，形成一幅 RGB 彩色图像的三个分量矩阵通常称为红、绿或蓝分量图像。

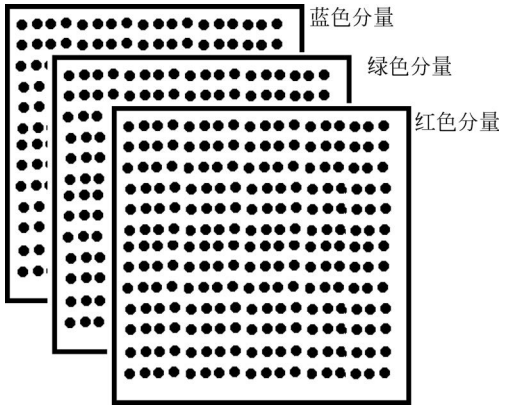


图 2-2　由红、绿、蓝三个分量图像形成 RGB 彩色图像示意图

图 2-3 是一幅实际图像(原图见本书配套电子资源)的红、绿、蓝分量示意图。以服装部分为例,其绿色分量的亮度(分量值)明显大于红色分量和蓝色分量部分,这表明原图中的服装色彩主要呈现为绿色。

(a) 红色分量　　(b) 绿色分量　　(c) 蓝色分量

图 2-3　一幅实际 RGB 图像的红、绿、蓝分量示意图

由于 RGB 图像的每个分量都是一个二维矩阵,因此很多灰度图像的算法也可改进后用于彩色图像的处理。

索引图像有两个分量,即整数的数据矩阵 $\boldsymbol{X}$ 和彩色映射矩阵 map。map 为 $m\times 3$ 的浮点类型,其中 m 为其所定义的颜色数目,每一行定义了颜色的红、绿、蓝三个分量。这些概念在图 2-4 中给予说明。

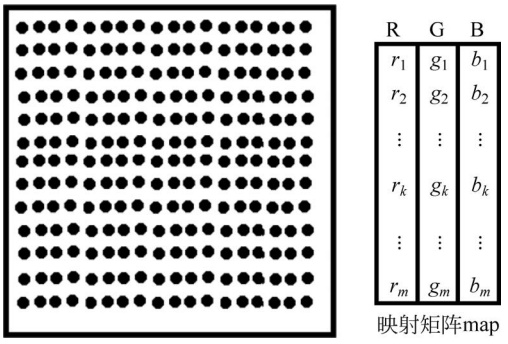


图 2-4　索引图像示例

除 RGB 和索引模式,彩色图像还有 CMYK 模式、HSB 模式、Lab 模式等,此处不再一一介绍。

扩展阅读

图像格式:图像格式是数字图像的存储规范,常见的包括:JPEG(适合照片)、PNG(支持透明背景)、GIF(动图专用)、WebP(高压缩率),以及专业领域用的 TIFF(印刷)和 DICOM(医疗)等。

矢量图:矢量图是通过数学公式定义的图像格式,其核心特点是无限缩放不失真。与像素构成的位图不同,矢量图由路径、锚点和填充属性描述,适合存储 Logo、图标、工程图纸等需要频繁缩放的图形。主流格式包括 SVG(网页友好)、EPS(跨平台印刷标准)和 PDF(文档通用)等。

色彩空间:色彩空间是用于数字化表示颜色的数学框架,主要包括 RGB(显示器使用的红绿蓝加色模型)、CMYK(印刷采用的青品黄黑减色模型)、HSV(基于色调-饱和度-亮度

的直观调色系统)、Lab(设备无关的广色域科学模型)以及YUV(视频压缩用的亮度-色度分离模型)。不同模型各有优势,使用时需根据应用场景(如显示/印刷/分析)具体权衡使用哪种颜色模型。

2.2 数字图像的读、写和显示

MATLAB的图像处理工具箱由大量支持图像处理操作的函数组成,借助这些函数可以方便地进行图像处理操作。

2.2.1 图像的读取

使用imread函数可以将图像数据读入MATLAB环境。常用语法格式为

```
[A,map] = imread(filename)
```

其中,A为所得到的数据图像矩阵;map为彩色映射矩阵(若读取非索引图像则此参数可省略);filename为被读入图像的文件全名(含扩展名)。当filename中不包含任何路径信息时,imread默认从当前目录中读取文件;若当前目录中没有所需文件,则自动尝试在MATALB搜索路径中寻找该文件。要想读入指定路径中的图像,则filename中须包含相应路径信息。

例2-1 使用imread函数读取当前目录下一幅名为Clock.tiff的图像,并显示图像矩阵。可使用命令:

```
>> x = imread('Clock.tiff');
>> x
```

在第一步中若去掉分号,则可省去第二步,直接显示矩阵。图2-5为原始图像及其在MATLAB命令窗口中的图像数据矩阵显示。

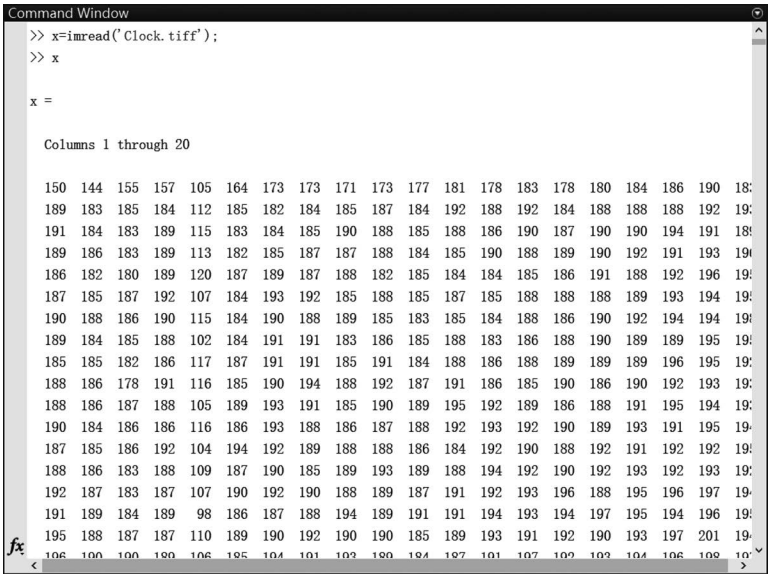


图2-5 原始图像及其在MATLAB命令窗口中的图像数据矩阵显示

可通过 whos 命令查看 x 的附加信息

```
>> whos x
  Name         Size              Bytes  Class    Attributes
  x          256×256             65536  uint8
```

结果表明，x 中的数据元素为 8 位无符号整数，这是为了节省空间，MATLAB 为图像提供的特殊数据类型。该数据类型所允许的数据范围，最小值为 0，最大值为 255。

2.2.2 图像的显示

imshow 函数是最常用的图像显示函数，其基本语法为

```
imshow(I)
```

其中，I 为一个图像矩阵[①]，语法为

```
imshow(I,[low high])
```

则将小于或等于 low 的灰度显示为黑色，将大于或等于 high 的灰度显示为白色，介于 low 与 high 之间的灰度值将以默认的级数显示为不同亮度值。语法为

```
imshow(I,[  ])
```

则自动将 low 设置为 I 中的最小值，将 high 设置为 I 中的最大值。这一显示形式在图像矩阵的动态范围较小或有负值出现时经常用到。

继续例 2-1 中的操作，在命令窗口中输入

```
>> imshow(X)
```

得到相应图像的显示结果如图 2-6 所示。

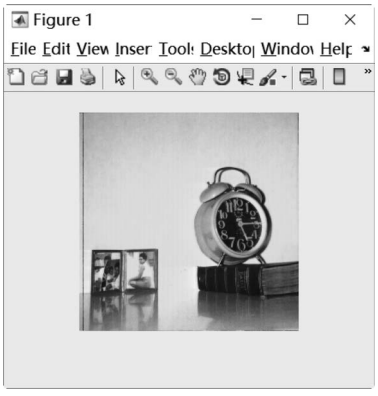


图 2-6 Clock 图像的显示结果

读者可以尝试用 imshow 的其他语法格式对该图像进行显示。

MATLAB 还允许将图像窗口划分为几部分，分别显示不同的图像，此功能可用

① 若 I 为索引图像，则采用 imshow(I,map)格式，其中 map 为色彩映射表。

subplot 范围来实现。语法为

```
subplot(m,n,p)
```

其中,前两个参数的意义是将图像窗口划分为 $m \times n$ 块,第三个参数 p 表示图像的序号。

例 2-2 用 subplot 函数在一个 2×2 的窗口中同时显示多幅图像。

```
>> X1 = imread('Boat.tiff');
>> X2 = imread('Clock.tiff');
>> X3 = imread('Elaine.tiff');
>> X4 = imread('Man.tiff');
>> subplot(2,2,1);imshow(X1);title('Boat');
>> subplot(2,2,2);imshow(X2);title('Clock');
>> subplot(2,2,3);imshow(X3);title('Elaine');
>> subplot(2,2,4);imshow(X4);title('Man');
```

其运行结果如图 2-7 所示。

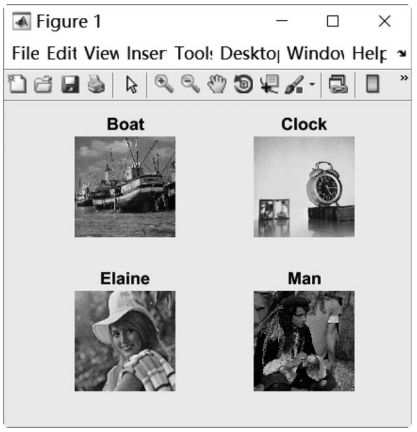


图 2-7 用 subplot 函数显示多幅图像示例

2.2.3 图像的保存

imwrite 函数可用于将图像保存到指定目录中。其常用的语法格式为

```
imwrite(I,filename)
```

表示将图像矩阵 I 以文件名 filename 保存到外存储器中,其中,filename 应当包含扩展名,用于指定文件类型。如果 filename 不包含路径,则将该图像文件保存于当前文件夹。

亮度范围:实际图像处理中,不同来源的图像可能具有不同的亮度范围(如常用的 8 位 JPEG 为 0～255、医学影像的 12 位 DICOM 为 0～4095)。为了方便算法处理,在具体应用中有时会将像素值归一化到区间[0,1]。这种归一化操作既能统一不同位深图像的数值尺度,又保留了与原始数据的对应关系。

2.3 邻域操作与块操作

2.3.1 图像的邻域操作

邻域操作是将每个像素值利用其相应邻域信息进行处理的过程。这里所选的像素邻域通常远小于原图像大小，且形状规则。

邻域操作包括滑动邻域操作和分离邻域操作两种。滑动邻域是一个像素集，其所包含的元素由中心像素的位置所决定，操作结果也作为该中心像素新的灰度值。滑动邻域操作一个邻域只处理一个像素。为方便确定中心像素，滑动邻域通常采用奇数行、奇数列的矩形窗口。

实现一个滑动邻域操作需要以下 5 个步骤[9]：

(1) 选择一个单独的像素。

(2) 确定该像素的滑动邻域。

(3) 对邻域中的像素值应用一个函数求值，该函数将返回标量计算结果。

(4) 将计算结果作为输出图像中对应的像素的值。

(5) 对输入图像的每一个像素都重复上面 4 个步骤。

MATLAB 图像处理工具箱提供了 nlfilter 函数进行滑动邻域操作，其语法如下：

```
J = nlfilter(I, [m n], fun)
J = nlfilter(I, 'indexed', …)
```

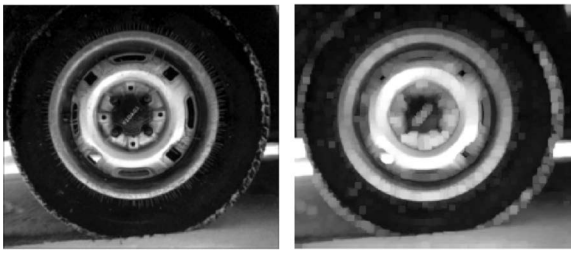
其中，I 为原图像；[m n]指定了大小为 $m \times n$ 的滑动邻域；fun 是对滑动邻域矩阵进行操作的函数；indexed 是可选参数，指定这个参数，则表示原图像为索引图像。

nlfilter 语法格式中的 fun 参数还可以为函数句柄，从而使用户可以通过自编 M 文件来实现邻域操作中的具体计算。

如果用户定义函数比较简单，则可直接采用内联函数或匿名函数定义，从而省去 M 文件，提高代码运行效率。

例 2-3 使用 nlfilter 函数实现了将每个像素设置为 5×5 邻域的最大值，如图 2-8 所示。

```
x = imread('tire.tif');
myf = @(x) max(x(:));
y = nlfilter(x,[5 5],myf);
subplot(1,2,1);imshow(x);
subplot(1,2,2);imshow(y);
```



(a) 原始图像　　(b) 邻域操作结果

图 2-8　nlfilter 函数示例

MATLAB还提供了一种用于快速邻域块操作的函数colfilt,这个函数通过为每个像素建立一个列向量,向量各元素对应该像素的邻域元素,减少了图像操作过程中的邻域定义的耗时。具体语法格式为

```
J = colfilt(I,[m n],block_type,fun)
J = colfilt(I,[m n],[mblock nblock],block_type,fun)
J = colfilt(I,'indexed',…)
```

该函数的用法与nlfilter函数相似,I表示原图像,[m n]指定大小为 $m \times n$ 的滑动邻域,[mblock nblock]表示一次将 $m$block$\times n$block大小的图像读入内存;fun为邻域矩阵操作函数,indexed依然为用于指明索引图像的可选参数。block_type用于定义块的移动方式,有两个取值:distinct和sliding,分别用于表示分离块操作和滑动块操作。例2-4给出了例2-3操作的colfilt函数实现方法。

例2-4 用colfilt函数对例2-3中的操作实现快速算法。

```
x = imread('tire.tif');
myf = @(x) max(x);
y = uint8(colfilt(x,[5 5],'sliding',myf));
subplot(1,2,1);imshow(x);
subplot(1,2,2);imshow(y);
```

由于colfilt函数操作中每个像素的邻域均已排成列向量,故此算法中的匿名函数参数为max(x),不同于例2-3 nlfilter中的参数max(x(:))。

为方便用户编制更复杂的邻域操作,例2-5给出例2-4算法的外部函数仿真实现。

例2-5 采用外部函数实现例2-4算法代码。

```
x = imread('tire.tif');
y = uint8(colfilt(x,[5 5],'sliding',@MAX));
subplot(1,2,1);imshow(x);
subplot(1,2,2);imshow(y);

function y = MAX(x)
y = max(x);
```

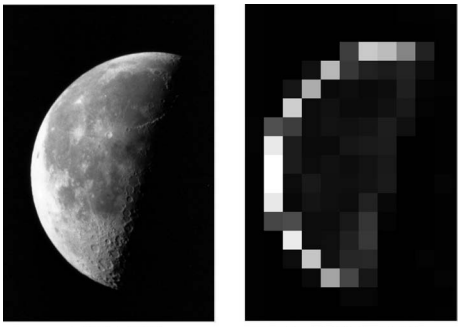
2.3.2 图像的块操作

图像的分离块操作是将图像划分为大小相同的矩形区域,不同图像块在图像中无重叠地排列。

在MATLAB图像处理工具箱中,对应分离块操作的函数是blockproc,其语法格式如下:

```
J = blockproc(I, blockSize, fun)
```

其中,I为原始图像;blockSize用于指定图像块的大小,图像块的信息由块结构定义,相应的数据矩阵放在.data域中。例2-6为MATLAB帮助文档中的块操作示例程序(参见图2-9)。



(a) 原始图像　　(b) 块操作结果

图 2-9　块操作示例

例 2-6　将 32×32 图像块中所有像素的灰度值设为该图像块的标准方差。

```
fun = @(X) std2(X.data) * ones(size(X.data));
I2 = blockproc('moon.tif',[32 32],fun);
figure;
imshow('moon.tif');
figure;
imshow(I2,[]);
```

其中，moon.tif 图像为 MATLAB 图像工具箱自带，故不用指定所在目录。请读者分析代码及运行结果的物理意义。

扩展阅读

邻域操作：邻域操作的技术演进完整展现了计算机视觉的发展轨迹。20 世纪 50—70 年代，以均值滤波、高斯滤波和 Canny 边缘检测为代表的固定核操作奠定了传统图像处理的理论基础；20 世纪 80 年代卷积神经网络(CNN)的诞生首次实现了从人工设计到可学习邻域操作的历史性突破；进入 21 世纪后，深度学习推动邻域操作持续革新——大感受野卷积核拓展了空间建模能力，可变形卷积实现了几何自适应采样，而注意力机制则彻底突破了局部性限制。这一从固定模板到动态学习、从局部处理到全局建模的技术跃迁，不仅完成了特征工程的范式转换，更构建了连接底层视觉与高层语义理解的关键桥梁，深刻影响了现代计算机视觉的发展方向。

本 章 实 验

实验一　灰度图像的基本操作

一、实验目的

(1) 掌握灰度图像的读、写与显示。

(2) 理解灰度图像矩阵的物理意义。

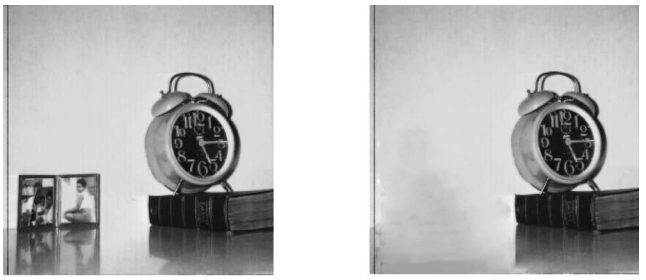
二、实验原理

(1) 数字图像的离散化表示。

(2) 数字图像的读、写和显示。

三、实验内容

如图 2-10 所示,对本书电子资源中的 Clock.tiff 图像,尝试通过 MATLAB 矩阵基本操作,剪切掉左下角的日历部分内容,并保持原图像视觉上的光滑性。



(a) Clock原图　　(b) 处理结果

图 2-10　图像的矩阵操作

四、实验报告要求

(1) 描述实验的基本原理和步骤。

(2) 用数据和图片给出各个步骤中取得的实验结果,并进行必要的讨论。

(3) 完整的原代码。

(4) 将处理后的图像写到外存储器中。

(5) 若不考虑编程实现,可否给出能更好地保持这种光滑性的矩阵操作方法?

实验二　彩色图像的基本操作

一、实验目的

(1) 掌握两种彩色图像的读取方式。

(2) 通过对彩色图像的基本操作进一步理解图像处理的实际应用。

二、实验原理

(1) 数字图像的离散化表示。

(2) 数字图像的读、写和显示。

三、实验内容

(1) 读入 MATLAB 图像处理工具箱中的一幅 RGB 彩色图像(onion.png),编程对其三种彩色分量进行轮换,即将 R 分量换成 G 分量,G 分量换成 B 分量,B 分量换成 R 分量。显示并保存处理结果。

(2) 读入 MATLAB 图像处理工具箱中的一幅索引图像(kids.tif),分析数据矩阵与映射矩阵的结合机理。此照片已有些发“黄”,尝试通过修改映射矩阵,使此图像看上去不那么显旧。

(3) 有没有类似方法帮助红绿色盲识别交通指示灯?请读者介绍自己在这方面的想法。

四、实验报告要求

(1) 描述实验的基本原理和步骤。

(2) 用数据和图片给出各个步骤中取得的实验结果,并进行必要的讨论。