# 第一部分 第1-2章作业

P9页

1.3 图像分析的基本特征是什么

**特征提取**：图像分析的第一步是从图像中提取重要的特征或信息。这些特征可以包括图像的边缘、颜色、纹理、形状等。特征提取的目标是将图像中的信息转化为数字化的数据，以便计算机能够理解和处理。

**分类和识别**：一旦特征被提取出来，接下来的任务是对图像进行分类或识别。分类涉及将图像分为不同的类别，而识别则是确定图像中的对象或场景是什么。这通常需要使用机器学习和深度学习算法来进行图像分类和识别。

**目标检测**：目标检测是一种更复杂的任务，它不仅要识别图像中的对象，还要确定它们在图像中的位置。目标检测可以用于识别图像中的多个对象，并为它们绘制边界框。

**分割**：图像分割是将图像划分为不同的区域或分割出感兴趣的对象的过程。这可以用于分离前景和背景，或者将图像中的对象分成不同的部分。

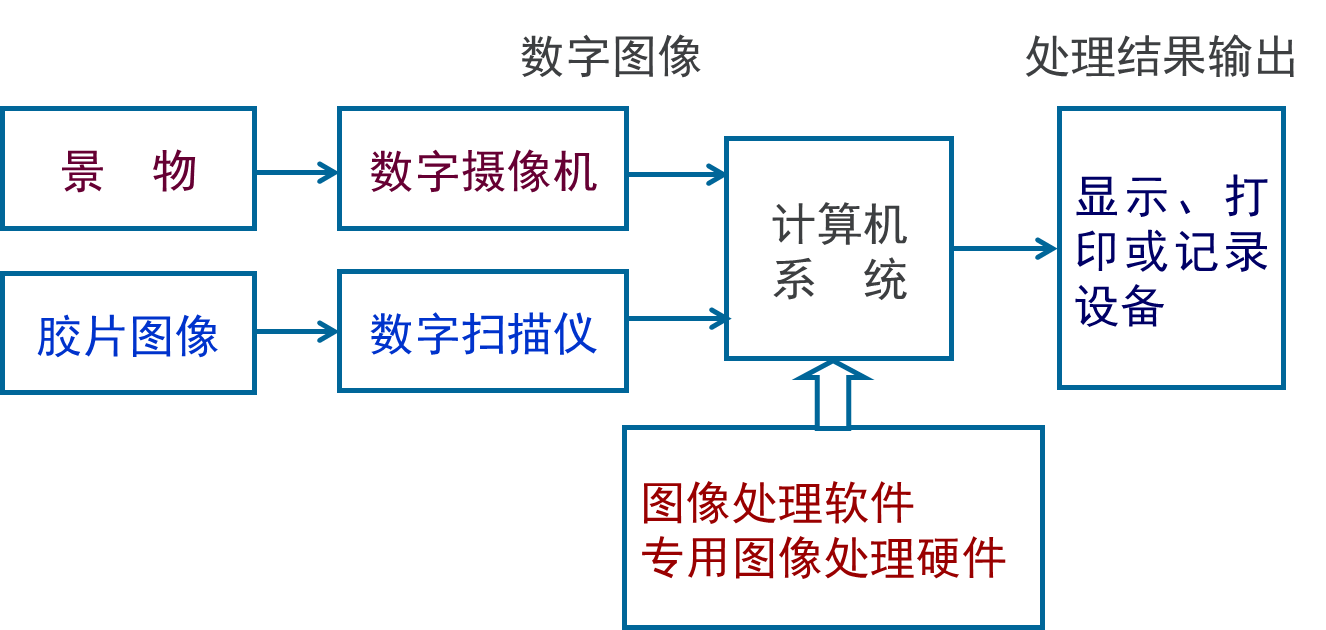
**物体跟踪**：物体跟踪是指在连续的图像帧中追踪对象的位置和运动。这在视频监控、自动驾驶和医学图像分析等领域中非常重要。

**图像增强和恢复**：图像分析还包括改善图像质量、去噪、去模糊和图像恢复等任务，以便更好地分析和理解图像内容。

**特定应用领域的定制化特征**：根据具体的应用领域，图像分析还可以包括一些特定的任务和特征。例如，在医学图像分析中，可能需要检测肿瘤或病变；在地理信息系统中，可能需要提取地理特征。

1.5 数字图像处理系统主要由哪几部分组成？各部分的功用是什么？

组成如下：



作用如下：

图像采集设备：

摄像机：用于捕捉现实世界中的视觉信息，将其转换为数字图像数据。不同应用领域使用不同类型的摄像机，例如数字相机、工业摄像机、无人机载荷等。

图像预处理：

图像采样：将连续的图像转换为离散的像素点，以数字形式表示图像。

量化：确定每个像素的颜色或亮度值的离散级别，通常使用灰度级或颜色通道。

增强：通过应用滤波器、直方图均衡化等技术，改善图像的质量、对比度和清晰度。

去噪：降低图像中的噪声，以提高图像的质量和可分析性。

颜色空间转换：将图像从一种颜色表示空间转换为另一种，例如从RGB到灰度或HSV等。

特征提取：

边缘检测：识别图像中物体之间的边界。

角点检测：寻找图像中的角点，通常用于图像匹配和跟踪。

纹理分析：提取图像中的纹理信息，有助于对象识别和分类。

特征描述子：将图像中的特征点描述为数字向量，以便于比较和匹配。

图像处理和分析：

滤波和卷积：应用不同的滤波器来平滑、增强或检测图像中的特定特征。

对象检测和识别：使用机器学习和计算机视觉算法来识别图像中的对象。

图像分割：将图像划分为不同的区域或对象。

特定任务的处理：根据应用需求，执行特定的图像处理任务，例如医学图像分析、地理信息系统、安全监控等。

后处理和结果呈现：

结果合并和整合：将多个处理步骤的结果合并，以获得最终的图像或对象识别结果。

图像显示：将处理后的图像或结果显示给用户或其他系统。

存储和传输：将处理后的图像或数据存储到文件或传输给其他系统，以备后续分析或共享。

用户界面和控制：

用户交互：提供用户界面，以便用户可以与图像处理系统进行交互，调整参数或查看结果。

系统控制：管理图像采集设备、处理流程和输出，确保系统按照用户需求执行任务。

P32页

2.1 （4）灰度图像：

灰度图像是一种常见的数字图像类型，它使用灰度级来表示图像中的亮度或灰度信息，而**不包含彩色信息**。在灰度图像中，每个像素点的灰度级表示了该点的亮度或灰度强度，通常以0（黑色）到255（白色）之间的整数值来表示，其中0表示最暗的黑色，255表示最亮的白色。

（10）像素的8邻域：

像素的4邻域对角像素和4邻域像素组成的集合

2.10：

1. 黑白二值图像：使用1位来表示每个像素的颜色（黑/白）。占用的比特数：200×300×1=60000
2. 16灰度级图像：使用4位来表示每个像素的颜色（0000~1111），占用的比特数：200×300×4=240000
3. 256灰度级图像：使用8位表示每个像素的颜色（00000000~11111111），占用的比特数：200×300×8=480000
4. 真彩色图像：用24位来表示每个像素的颜色（8位用于红色通道，8位用于绿色通道，8位用于蓝色通道），占用的比特数：200×300×24=1,440,000

# 第二部分 Python与图像基本处理

## 第一题

### 代码与注释分析

from PIL import Image

import matplotlib.pyplot as plt

import numpy as np

# 读取RGB图像

input\_image = Image.open("input\_rgb\_image.jpg")#读入图像

# 显示原始RGB图像

plt.figure(figsize=(12, 4))#创建新的图形窗口，大小：宽度为12英寸，高度为4英寸

plt.subplot(131)#在窗口中创建子图：(131)中的1表示总行数，3表示总列数，1表示当前子图的位置。

plt.title("RGB Image")#当前子图标题

plt.imshow(input\_image)

plt.axis('off')#不显示坐标轴

# 将图像转换为灰度图像

gray\_image = input\_image.convert("L")

#显示灰度图像

plt.subplot(132)

plt.title("Gray Image")#当前子图标题

plt.imshow(gray\_image, cmap='gray')#告诉matplotlib使用灰度颜色映射来显示图像，否则可能会选择其他颜色映射

plt.axis('off')

# 将图像进行二值化处理

threshold = 128 # 二值化的阈值，改变其大小会得到不同的结果

binary\_image = gray\_image.point(lambda p: p > threshold and 255)#使用lambda表达式确定阈值

#显示二值化图像

plt.subplot(133)

plt.title("Binary Image")

plt.imshow(binary\_image, cmap='gray')

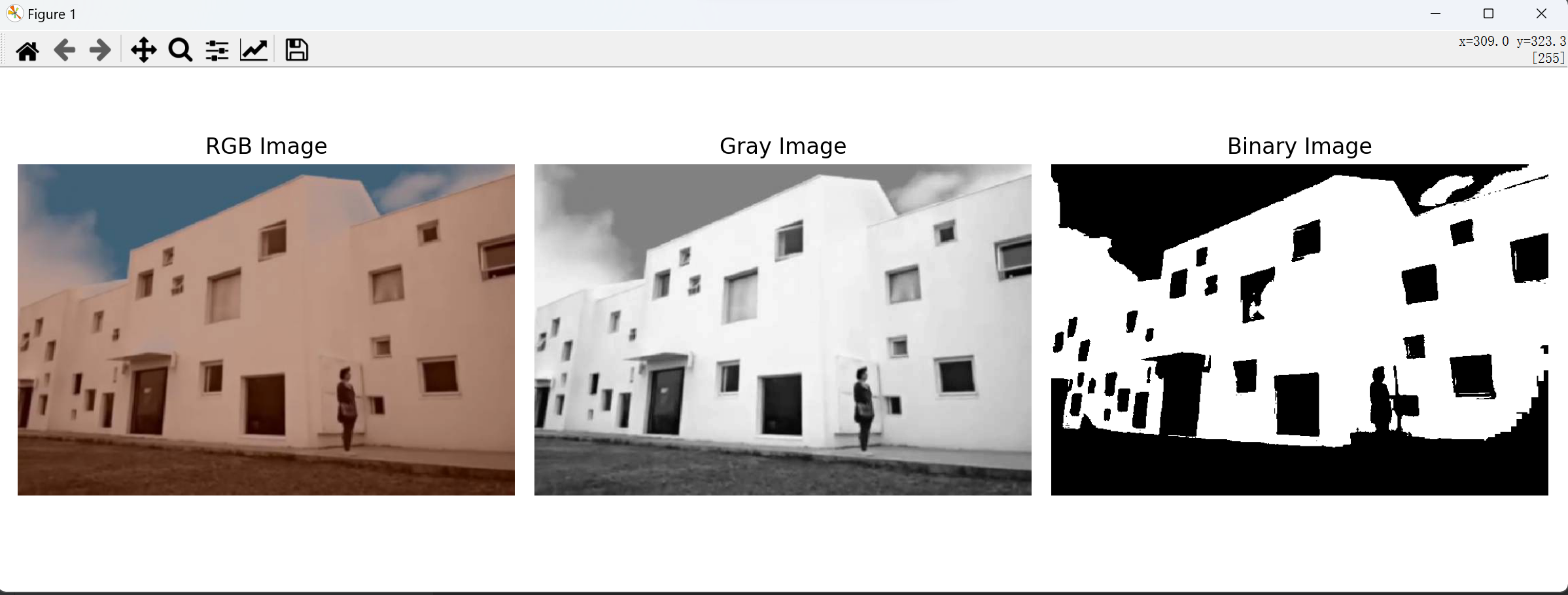
plt.axis('off')

# 展示图像

plt.tight\_layout()

plt.show()

### 输出



## 第二题

### 代码与注释分析

from PIL import Image

import matplotlib.pyplot as plt

import numpy as np

#读入图像并转换为灰度图像

input\_image = Image.open("input\_rgb\_image.jpg")

gray\_image = input\_image.convert("L")

# 将灰度图像转换为NumPy数组

gray\_array = np.array(gray\_image)

# print(gray\_array)

# 计算灰度直方图

hist, bins = np.histogram(gray\_array, bins=256, range=(0, 256))#参数解析：bins：直方图的柱子数量；range：灰度级别范围

#返回两个numpy数组：一个表示灰度值，另一个表示不同的灰度

# 计算归一化灰度直方图

normalized\_hist = hist / float(np.sum(hist))

# 绘制灰度直方图

plt.figure(figsize=(10, 4))

plt.subplot(121)

plt.title("gray-scale histogram")

plt.xlabel("grayscale level")

plt.ylabel("pixel count")

plt.bar(bins[:-1], hist, width=1, color='gray', align='center')

# 绘制归一化灰度直方图

plt.subplot(122)

plt.title("Normalized grayscale histogram")

plt.xlabel("grayscale level")

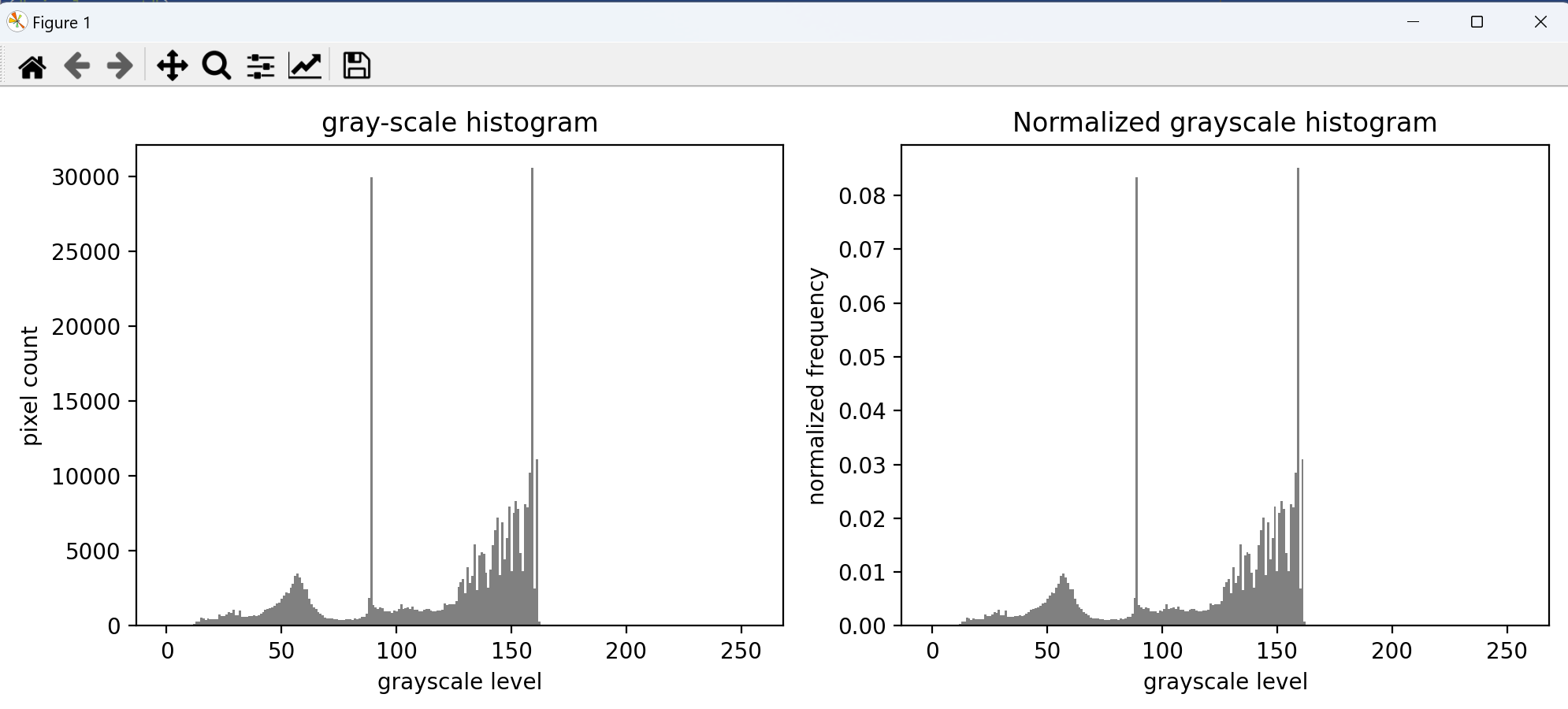
plt.ylabel("normalized frequency")

plt.bar(bins[:-1], normalized\_hist, width=1, color='gray', align='center')

plt.tight\_layout()#自动调整布局，以显示在图形窗口中

plt.show()

### 输出



## 第三题

### 代码与注释分析

import cv2

import matplotlib.pyplot as plt

# 读取第一幅图像

image1 = cv2.imread("image1.jpg")

# 读取第二幅图像

image2 = cv2.imread("image2.jpg")

#用addWeighted进行加权图像相加。0表示不增加图像的亮度

result\_image = cv2.addWeighted(image1, 0.5, image2, 0.5, 0)

# 转换BGR颜色通道为RGB，以便matplotlib正确显示

result\_image\_rgb = cv2.cvtColor(result\_image, cv2.COLOR\_BGR2RGB)

#展示相加前后图片

plt.figure(figsize=(12, 4))#创建新的图形窗口，大小：宽度为12英寸，高度为4英寸

plt.subplot(131)

plt.title("image1")

plt.imshow(image1, cmap='viridis')

plt.axis('off')

plt.subplot(132)

plt.title("image2")

plt.imshow(image2, cmap='viridis')

plt.axis('off')

plt.subplot(133)

plt.title("result image")

plt.imshow(result\_image\_rgb)

plt.axis('off')

plt.tight\_layout()#自动调整布局，以显示在图形窗口中

plt.show()

### 输出

