Lab1 实验报告

李青雅 523030910004 电院 2301

2024 年 11 月 19 日

目录

1 实验概览

本次实验需掌握 OpenCV 的基本图像处理操作，学习如何提取图像的特征，包括颜色直方图、灰度直方图和梯度直方图和熟悉在 Python 中调用 OpenCV 库和 Matplotlib 绘图工具。

在练习 1 中，我们需要读取彩色图像，并绘制出其 RGB 三个通道的直方图。在练习 2 中，我们需要读取灰度图像，并绘制出灰度比例直方图和梯度强度直方图。

2 实验环境

Visual Studio Code 中的 Python，OpenCV 库用于图像处理，Matplotlib 库用于绘图，numpy 库用于数学处理，os 库用于调整工作目录。

3 解决思路

3.1 练习 1

首先使用 cv.imread 读取彩色图像，其中 imread 以 BGR 的形式读取图像。因为 OpenCV 中的颜色通道顺序是 BGR，而 Matplotlib 中的颜色通道顺序是 RGB，所以需要将 BGR 转换为 RGB。然后计算 RGB 三个颜色通道的总能量和每个颜色通道的比例。最后使用 Matplotlib 绘制出 RGB 三个通道的直方图，并且在柱状条上方标出每个通道的比例。

在写完核心代码后，我又在代码的前后分别加上了修改工作目录到脚本目录，让图表支持中文，以及将图标保存到特定路径文件夹下的代码，以便后续的实验。

```python
def plot_color_ratio(image_path, title = "颜色比例直方图"):
    # 读取图像并转换为 RGB 格式
    img_bgr = cv2.imread(image_path)
    # 防止读取失败
    if img_bgr is None:
        print(f"无法读取图像: {image_path}")
        return
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

    # 计算每个颜色通道的能量总和
    color_sum = [img_rgb[:, :, i].sum() for i in range(3)]
    total_energy = sum(color_sum)

    # 计算每个颜色通道的比例
    color_ratio = [energy / total_energy for energy in color_sum]

    # 绘制颜色比例直方图
    colors = ['红', '绿', '蓝']
    plt.bar(colors, color_ratio, color=['red', 'green', 'blue'])
    plt.xlabel("颜色通道")
    plt.ylabel("颜色比例")
    plt.title(title)
    # 设置 y 轴范围从 0 到 1
    plt.ylim(0, 1)

    # 在每个柱状图上方显示具体的比例值
    for i, ratio in enumerate(color_ratio):
        plt.text(i, ratio + 0.01, f"{ratio:.2f}", ha='center', color='black')
```

图 1: 颜色分布直方图的核心代码

3.2 练习 2

- 灰度比例直方图

  与练习 1 类似，不过练习 2 需要读入灰度图像，确保每个像素的灰度值在 0-255 范围内。通过直方图计算每个灰度值出现的频率，并将灰度值频率归一化为比例，使其总和为 1，方便对比不同图像的灰度特性。

```python
# 读取图像为灰度图
img_gray = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
if img_gray is None:
    print(f"无法读取图像: {image_path}")
    return

# 计算灰度直方图
hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])

# 归一化处理，使得直方图值代表比例
hist_normalized = hist / hist.sum()

# 绘制直方图并填充颜色
plt.figure(figsize=(8, 6))
x = range(256)  # 灰度值范围
plt.fill_between(x, hist_normalized.flatten(), color='teal', alpha=1)
```

图 2: 灰度比例直方图的核心代码

- 梯度强度直方图

  类似地，以灰度方式读入图像，然后计算梯度。先使用 Sobel 算子分别计算图像在水平方向和垂直方向上的梯度，然后通过梯度公式计算每个像素的梯度强度。接着统计梯度强度分布，并归一化处理。

```python
# 计算梯度（Sobel算子）
grad_x = cv2.Sobel(img_gray, cv2.CV_64F, 1, 0, ksize=3)  # 水平方向梯度
grad_y = cv2.Sobel(img_gray, cv2.CV_64F, 0, 1, ksize=3)  # 垂直方向梯度

# 计算梯度强度
magnitude = np.sqrt(grad_x**2 + grad_y**2)

# 将梯度强度分桶统计
max_magnitude = np.max(magnitude)
hist, bin_edges = np.histogram(magnitude, bins=bins, range=(0, max_magnitude))

# 归一化直方图
hist_normalized = hist / hist.sum()

# 绘制直方图
plt.figure(figsize=(8, 6))
plt.bar(bin_edges[:-1], hist_normalized, width=(bin_edges[1] - bin_edges[0]), color='teal', alpha=1)
```

图 3: 梯度强度直方图的核心代码

4 代码运行结果

具体代码运行的结果（每一张图片）放在 codes 中名为 output1 和 output2 的文件夹中，这里只展示三张对应类别的图片。其中颜色分布直方图放在 output1 文件夹中，灰度比例直方图和梯度强度直方图放在 output2 文件夹中。

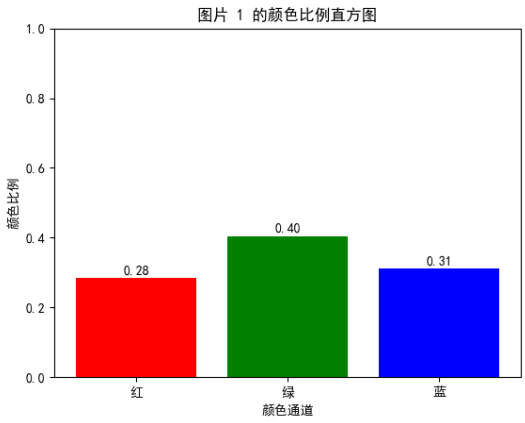

图 4: 颜色分布直方图

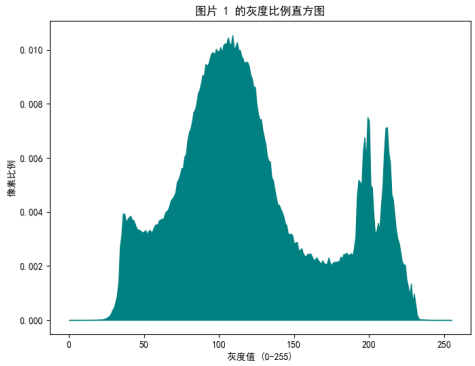

图 5: 灰度比例直方图

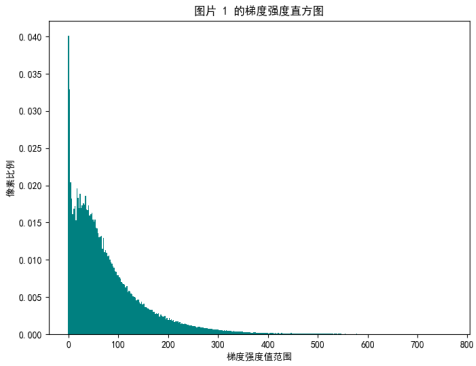

图 6: 梯度强度直方图

5 实验结果分析与思考

5.1 练习 1

在读入彩色图像后，我尝试用 plt.show() 显示图像，但是发现无法显示。后面发现是因为当前工作目录和脚本所在的目录不一样，读入的 img_bgr 是 NONE，所以无法显示。为了避免人工 cd 进脚本目录和读取失败的情况，我添加了如下代码。

```
# 更改工作目录到脚本所在目录
os.chdir(os.path.dirname(os.path.abspath(__file__)))
```

图 7: 修改工作目录到脚本目录

```
# 防止读取失败
if img_bgr is None:
    print(f"无法读取图像: {image_path}")
    return
```

图 8: 防止图片读取失败

三张照片的颜色分布都不一样，主观来看，哪一个通道里的的颜色占比越大（即直方图中对应的通道越高），这张图片给人的感觉就越像是这个颜色。

5.2 练习 2

灰度图像灰度值的分布情况，反映了灰度图像的明暗程度。图中 x 轴表示灰度值，y 轴表示灰度值的比例。相同 y 的高度情况下，x 轴数值越小，整个图片看起来越暗；x 轴数值越大，整个图片看起来越亮。而梯度强度直方图则反映了图像纹理的疏密程度，纹理简单的图在小 x 轴数值附近函数值较大，纹理复杂的图在大 x 轴附近函数值较大。

6 拓展思考

6.1 示例代码中的 cv2.cvtColor（img_bgr, cv2.COLOR_BGR2RGB）的作用是什么？

用于将图像从一种颜色空间转换为另一种颜色空间。在这里，cv2.cvtColor 将在 BGR 颜色空间上的 img_bgr 通过参数 cv2.COLOR_BGR2RGB 转化为 RGB 颜色空间。

6.2 如何使得 pyplot 正确显示灰度图的颜色？

在 plt.imshow() 中，需要加上 cmap='gray' 参数，表示将图像以灰度图的形式显示, 而不是以伪彩色的形式显示。

7 实验感想

练习 1 中，计算每个颜色的通道总和时，查阅资料找到了更简洁的写法，也算是我不太熟练 Python 进阶语法。并且在一开始，我将 BGR 的图像转化成了 RGB，其实严格上来说不需要这个转换，因为最后 matplotlib 显示的是直方图，而不是原图。不转换的话，只不过是最后直方图通道颜色柱的位置不一样而已。就是按照“蓝、绿、红”的顺序显示，而不是现在的“红、绿、蓝”。

与练习 1 类似,但是这里计算灰度直方图的时候,我直接用了 OpenCV 内置的函数 cv2.calcHist 计算灰度图像的直方图。而在计算梯度强度直方图的时候，我不太熟悉这里的梯度算法。通过查询资料，我了解到 Sobel 算子和分桶统计。可以用 cv.Sobel 来直接计算某个像素点在 x，y 两个方向上的方向梯度，然后计算梯度强度。而练习 2 中的分桶统计可以将 256 个灰度值分为 256 个区间，每个区间宽度为 1，然后统计每个区间内的梯度强度的个数，最后归一化处理。