目录

1 实验介绍

1.1 实验内容

1. 将 img1.png 和 img2.png 两幅图像以彩色图像方式读入, 并计算颜色直方图.
2. 将 img1.png 和 img2.png 两幅图像以灰度图像方式读入, 并计算灰度直方图和梯度直方图.

1.2 实验环境

操作系统:Ubuntu 18.04 LTS
Python 版本:3.7
NumPy:1.14.5
OpenCV-Python:3.4.3.18
Matplotlib:2.2.2

2 实验过程

2.1 灰度直方图

2.1.1 原理

灰度图像 $I(x,y)$ 的灰度直方图定义为各灰度值像素数目的相对比例, 反映了图像的明暗程度. 图像中灰度值为 i 的像素总个数为:

$$N(i) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} (I(x,y) == i ? 1 : 0)$$

则灰度直方图:

$$H(i) = \frac{N(i)}{\sum_{j=0}^{255} N(j)}, i = 0, \ldots, 255$$

2.1.2 程序实现

在 Python 中, 可以使用 NumPy 进行向量化编程, 避免对图像进行二层循环. 例如, `np.bincount()` 函数可以方便地实现计数操作. 在得到直方图数据后, 使用 Matplotlib 绘图即可.

```python
def grayscale_hist(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    hist = np.bincount(img.ravel(), minlength=256)
    hist = hist / np.sum(hist)
    plt.bar(range(0, 256), hist, width=1.0)
    plt.title("Grayscale Histogram of {}".format(image_path))
    plt.show()
```

2.2 颜色直方图

2.2.1 原理

彩色图像 $I(x,y,c)$ 的颜色直方图是它三个颜色分量总能量的相对比例, 反映了图像的总体色调分布. 某一颜色分量的总能量:

$$E(c) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x,y,c)$$

某一颜色分量的能量相对比例:

$$H(c) = \frac{E(c)}{\sum_{i=0}^{2} E(i)}$$

2.2.2 程序实现

同样的, 利用 NumPy 来操作图像对应的张量, 并使用 Matplotlib 来绘制直方图. 为了直观地表现色彩分布,RGB 的三个分量在直方图中分别用对应的颜色表示.

```python
def color_hist(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_COLOR)
    hist = np.sum(img.reshape(-1, 3), axis=0)
    hist = hist / np.sum(hist)
    plt.bar(["Blue", "Green", "Red"], hist, color=["blue", "green", "red"])
    plt.title("Color Histogram of {}".format(image_path))
    plt.show()
```

2.3 梯度直方图

2.3.1 原理

灰度图像 $I(x,y)$ 的梯度直方图定义为其各像素点的梯度值 (边界像素点除外), 反映了图像纹理的复杂程度. 一个像素点的梯度定义为:

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x} = I(x+1,y) - I(x-1,y)$$

$$I_y(x,y) = \frac{\partial I(x,y)}{\partial y} = I(x,y+1) - I(x,y-1)$$

而梯度强度定义为:

$$M(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2}$$

由于 $-255 \leq I_x, I_y \leq 255$,$0 \leq M(x,y) \leq 255\sqrt{2} < 361$, 则将梯度强度均匀分成 361 个区间, 定义 (x,y) 处的像素所在区间为:

$$B(x,y) = i, i \leq M(x,y) < i+1, 0 \leq i \leq 360$$

故落在第 i 个区间总的像素数目为:

$$N(i) = \sum_{x=1}^{W-2} \sum_{y=1}^{H-2} (B(x,y) == i ? 1 : 0)$$

比例为:

$$H(i) = \frac{N(i)}{\sum_{j=0}^{360} N(j)}$$

2.3.2 程序实现

在使用 NumPy 操作图像张量时, 需要注意其中的数据类型: 通过 OpenCV 读取的图像中默认 `dtype="uint8"`, 即为 8 位无符号整型数, 在进行求梯度操作时会出现溢出的问题, 导致最终出现错误的结果.

因此, 在读取图像之后, 需要改变其数据类型为 `"int32"`. 同时, 在计算出 `M` 后, 也应当将其从浮点类型转换为整型数, 以方便计数.

```python
def gradient_hist(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = img.astype("int32")
```

```
    Ix = (img[:, 2:] - img[:, :-2])[1:-1, :]
    Iy = (img[2:, :] - img[:-2, :])[:, 1:-1]
    M = np.sqrt(Ix * Ix + Iy * Iy).astype("int32")
    hist = np.bincount(M.ravel(), minlength=361)
    hist = hist / np.sum(hist)
    plt.bar(range(0, 361), hist, width=1.0)
    plt.title("Gradient Histogram of {}".format(image_path))
    plt.show()
```

2.4 主程序

在完成以上绘制 3 种直方图的函数后, 只需在主程序中分别调用即可:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


if __name__ == '__main__':
    images_list = ["img1.png", "img2.png"]
    histograms_list = [color_hist, grayscale_hist, gradient_hist]
    for image in images_list:
        for histogram in histograms_list:
            histogram(image)
```

绘制的直方图见随报告一起提交的图片文件.

3 总结与分析

本次实验中学习了基本的图像特征提取, 同时练习了 NumPy 的使用.