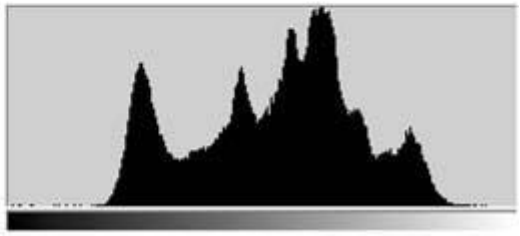
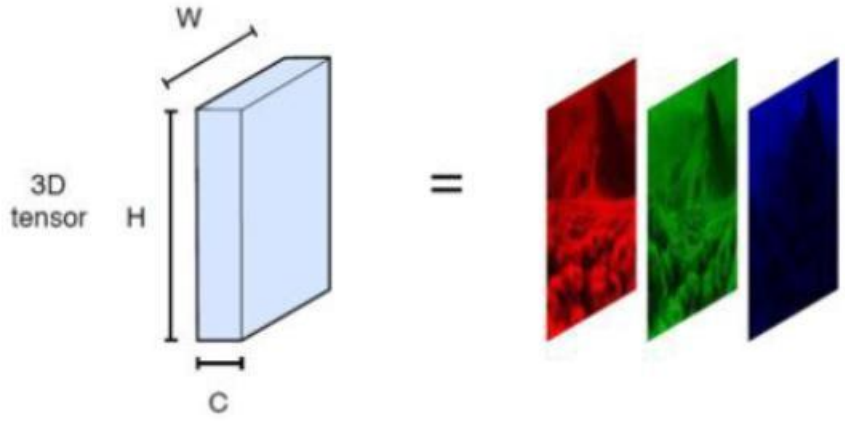
利用图像直方图特征计算图像相似度

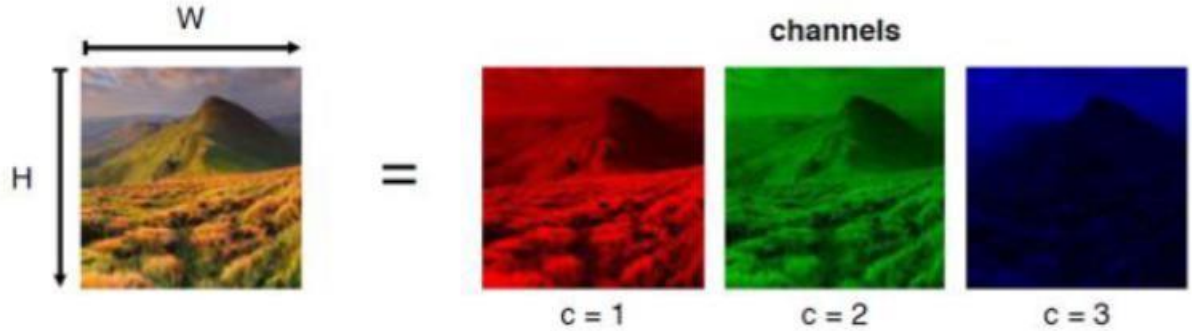
一、相关技术描述

（一）单通道图 俗称灰度图，把白色与黑色之间按对数关系分为若干等级，称为灰度，灰度分为 256 阶。任何颜色都有红、绿、蓝三原色组成，而灰度图只有一个通道，他有 256 个灰度等级，255 代表全白，0 表示全黑。从整幅图像的整体和局部的色彩以及亮度等级分布特征来看，灰度图描述与彩色图的描述是一致的特点。因此很多真彩色图片的分析，第一步就是转换为灰度图，然后再进行分析。



（二）三通道图 每个像素点都有 3 个值表示，所以就是 3 通道。 例如 RGB 图片即为三通道图片，RGB 色彩模式是工业界的一种颜色标准， 是通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的。RGB 即是代表红、绿、蓝三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一。





（三）图像直方图（英语：Image Histogram）是用以表示数字图像中亮度分布的直方图，标绘了图像中每个亮度值的像素数。可以借助观察该直方图了解需要如何调整亮度分布。这种直方图中，横坐标的左侧为

纯黑、较暗的区域，而右侧为较亮、纯白的区域。因此，一张较暗图片的图像直方图中的数据多集中于左侧和中间部分；而整体明亮、只有少量阴影的图像则相反。

（四）巴氏距离 在统计中，巴氏距离用于度量两个概率分布的相似性。它与 Bhattacharyya 系数密切相关，后者是两个统计样本或总体之间重叠量的度量。两种度量均以 1930 年代在印度统计研究所工作的统计学家 Anil Kumar Bhattacharya 的名字命名。Bhattacharyya 系数（Bhattacharyya Coefficient, 巴氏系数）是对两个统计样本的重叠量的近似计算。巴氏系数可用来对两组样本的相关性进行测量。

Bhattacharyya 距离(method = CV_COMP_BHATTACHARYYA)

$$d_{\text{Bhattacharyya}}(H_1,H_2)=\sqrt{1-\sum_I \frac{\sqrt{H_1(i)\cdot H_2(i)}}{\sum_I H_1(i)\cdot \sum_I H_2(i)}}$$

对于 Bhattacharyya 匹配[Bhattacharyya43]，低分数表示好匹配，而高分数表示坏的匹配。完全匹配是 0，完全不匹配是 1。

二、核心代码分析

（一）核心算法代码

单通道直方图计算图像相似度：

```
def calculate1(image1, image2):
    # 单通道直方图算法
    # 计算单通道的直方图并进行归一化
    hist1 = cv2.calcHist([image1], [0], None, [256], [0.0, 255.0])
    cv2.normalize(hist1, hist1,0,255*0.9,cv2.NORM_MINMAX)
    hist2 = cv2.calcHist([image2], [0], None, [256], [0.0, 255.0])
    cv2.normalize(hist2, hist2,0,255*0.9,cv2.NORM_MINMAX)

    # 计算直方图的重合度，利用巴氏距离法
    degree = 0
    degree = 1- cv2.compareHist(hist1, hist2, cv2.HISTCMP_BHATTACHARYYA)
    return degree
```

函数 `calculate1(image1, image2)`利用单通道直方图算法计算 `image1、image2` 的图像相似度。首先使用 `calcHist` 函数计算图像的直方图，之后使用 `normalize` 函数对计算出的直方图进行归一化处理。最后使用 `compareHist` 函数的巴氏距离法计算两张直方图的相似度。

`calcHist` 函数的定义如下

```
cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate ]]) -
>hist
```

其中，`images` 表示输入的图像或数组，且要求尺寸必须相同；`channels` 指需要计算直方图时需要统计的通道，第一个数组通道从 0 到 image[0].channels()-1，第二个数组从 image[0].channels() 到 images[0].channels()+images[1].channels()-1，以后的数组以此类推；`mask` 表示可选的操作掩码，一般用

None，表示处理整幅图像；histSize 表示每个维度的直方图尺寸，即这个直方图分成多少份，含有多少个直方柱；range 指直方图中各个像素的取值范围，[0.0, 256.0]表示直方图能表示像素值从 0.0 到 256 的像素；最后是两个可选参数，由于直方图作为函数结果返回了，所以第六个 hist 就没有意义了（待确定） 最后的参数 accumulate 是累计标识符，类型是布尔值，用来表示直方图是否叠加。

normalize 函数定义如下

```
cv2.normalize(src[, dst[, alpha[, beta[, norm_type[, dtype[, mask]]]]]]) → dst
```

其中，src 表示输入的数组；dst 表示与 src 大小相同的输出数组；alpha 指范数值在范围归一化的情况下归一化到较低的范围边界；beta 指在范围归一化的情况下的范围上限，不用于规范化；norm_type 是规范化类型，可选择的类型有 NORM_MINMAX（线性归一化）、NORM_INF（切比雪夫距离）、NORM_L1（曼哈顿距离）、
NORM_L2（欧几里得距离） compareHist 函数定义如

下

```
compareHist(H1, H2, method)
```

其中，H1 指的是用于比较第一张直方图；H2 指用于比较的第二章直方图；method 表示用于比较的方法策略，可选择的方式有 cv2.HISTCMP_CORREL（相关性比较）、cv2.HISTCMP_CHISQR（卡方比较）、cv2.HISTCMP_BHATTACHARYYA（巴氏距离比较）等**三通道直方图计算图像相似度**

```
def calculate2(image1, image2):
    # RGB每个通道的直方图相似度
    # 将图像分离为RGB三个通道，再计算每个通道的相似值
    sub_image1 = cv2.split(image1)
    sub_image2 = cv2.split(image2)
    sub_data = 0
    for im1, im2 in zip(sub_image1, sub_image2):
        sub_data += calculate1(im1, im2)
    sub_data = sub_data / 3
    return sub_data
```

函数 calculate2(image1, image2)利用三通道直方图算法计算 image1、image2 的图像相似度。首先使用 split 函数将图像分离成 R、G、B 三个颜色通道，再调用 calculate1 函数分别计算三个通道的相似度，最后得到图像相似度的平均值。

（二）图形界面代码

```
global img_display1
global img_display2
global img_display3
global filename1
global filename2
global filename3

vec_x1 = 0.05 # 第一张图片x坐标
vec_x2 = 0.25 # 第二张图片x坐标
vec_x3 = 0.5 # 第三张图片x坐标
vec_y1 = 0.1 # 按钮y坐标
vec_y2 = 0.2 # 图片y坐标
```

首先，定义了图形界面将要使用到的全局变量

```
# 设置界面的性质
root = Tk()
root.geometry('900x600') #界面大小
root.resizable(0,0)   # 禁止界面大小调整
root.title('图像相似度计算') # 界面标题
root.configure(bg = 'gray') #界面背景颜色
```

此处创建图形界面，并对图形界面的属性进行设置，`geometry` 设置界面大小为 900*600，`resizable(0,0)`禁止界面大小调整，相当于关闭了界面放大按钮，`title` 设置图形界面的标题，并使用 `configure` 将界面的背景颜色设置成了灰色。

`dispaly1` 和 `display2` 函数的作用相同，都是打开文件夹并对选中的图像进行显示。下面只对其中一个函数进行展示：

```
# 函数用于打开文件并显示图像
def display1():
    global img_display1
    global filename1

    filename1=tkinter.filedialog.askopenfilename() # 打开文件管理器
    if filename1 != '':
        lb.config(text='您选择的文件是'+filename1)

        img1 = Image.open(filename1) # 打开图片
        resized1 = img1.resize((150, 150)).convert('RGB') #调整图片大小
        img_display1 = ImageTk.PhotoImage(resized1)
        label_img1 = tkinter.Label(root, image = img_display1)
        label_img1.place(relx=vec_x1, rely=vec_y2)
    else:
        lb.config(text='您未选择任何文件')
```

`img_display1` 指的是将要展示的图片，`filename1` 是当前从文件夹中选中的文件名。函数首先使用 `filedialog` 打开文件夹管理器，并获取选中的文件名。之后使用 `Image.open` 打开图片，并将图片调整到合适的大小和规格，最后使用 `ImageTK.PhotoImage` 将调整后的图片转换成用作展示用的图片。

`compare` 函数用于对比图像 `image1、iamge2` 的相似度，并画出图像相似度对比的折线图，具体代码如下所示

```
# 函数用于比较图像相似度
def compare():
    global filename1
    global filename2
    global filename3
    global img_display3

    if(filename1 != '' and filename2 != ''):
        img1_path = filename1
        img2_path = filename2

        # 使用opencv方法计算图像相似度
        imgobj1 = cv2.imread(img1_path)
        imgobj2 = cv2.imread(img2_path)

        # 将图片大小进行统一
        img1 = cv2.resize(imgobj1, (256, 256))
        img2 = cv2.resize(imgobj2, (256, 256))

        res1 = calculate1(img1, img2) #单通道直方图计算
        res2 = calculate2(img1, img2) #三通道直方图计算
        string2 = "单通道直方图相似度：%s" % res1 + "\n" + "三直方图算法相似度：%s" %
res2
        lb_cmp.config(text=string2)


        lb.config(text='')
        if res1 < 0.6:
            lb_res.config(text="哎呀，这两张图片看起来不太像呢≈( •`ω •´)✧")
        else:
            lb_res.config(text="哇塞，这两张图片看起来真像≈o(*^@^*)o")

        # 画出单通道直方图图像相似度比较的折线图
        hist1 = cv2.calcHist([img1], [0], None, [256], [0.0, 255.0])
        hist2 = cv2.calcHist([img2], [0], None, [256], [0.0, 255.0])
        cv2.normalize(hist1, hist1,0,255*0.9,cv2.NORM_MINMAX)
        cv2.normalize(hist2, hist2,0,255*0.9,cv2.NORM_MINMAX)
        pyplot.plot(range(256), hist1, 'r')
        pyplot.plot(range(256), hist2, 'b')
        pyplot.title('单通道图像对比度')
        pyplot.rcParams['font.sans-serif']=['SimHei']
        pyplot.rcParams['axes.unicode_minus'] = False
        filename3 = r'C:\test\test.png'
        pyplot.savefig(filename3)   # 将图片保存

        # 在图形界面显示折线图
```

```
            img3 = Image.open(filename3)
            resized3 = img3.resize((430, 280)).convert('RGB')
            img_display3 = ImageTk.PhotoImage(resized3)
            label_img3 = tkinter.Label(root, image = img_display3)
            label_img3.place(relx=vec_x3, rely=vec_y2)

            pyplot.clf() #清空数据图
        else:
            lb_res.config(text='--这是一个利用图像直方图特征计算图像相似度的程序--',
    bg='gray')
```

首先使用 `imread` 函数读取相关文件路径的图片对象，然后调用 `resize` 函数将图片大小进行统一，之后分别调用 `calculate1` 和 `calculate2` 方法，进行单通道和三通道直方图的图像相似度计算，并将结果输出到图形界面相关标签上。

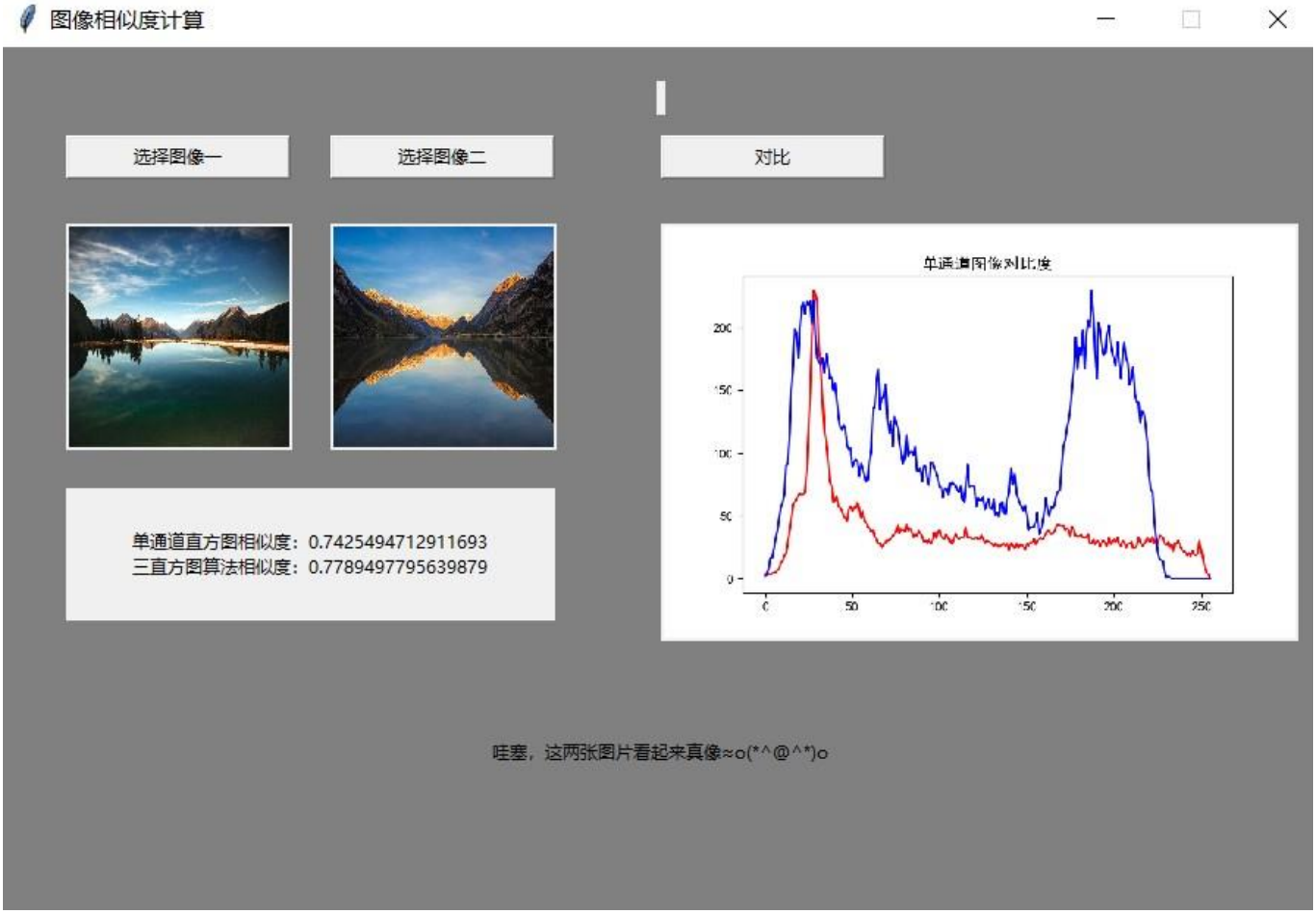
另外，函数利用 `pyplot` 的方法画出单通道图像对比的折线图，并将折线图的图片保存到路径 `C:\test\test.png` 中，之后从文件中读取图片并将它展示到直方图的展示界面上。

接下来，对图形界面的相关内容进行补充，代码内容如下所示

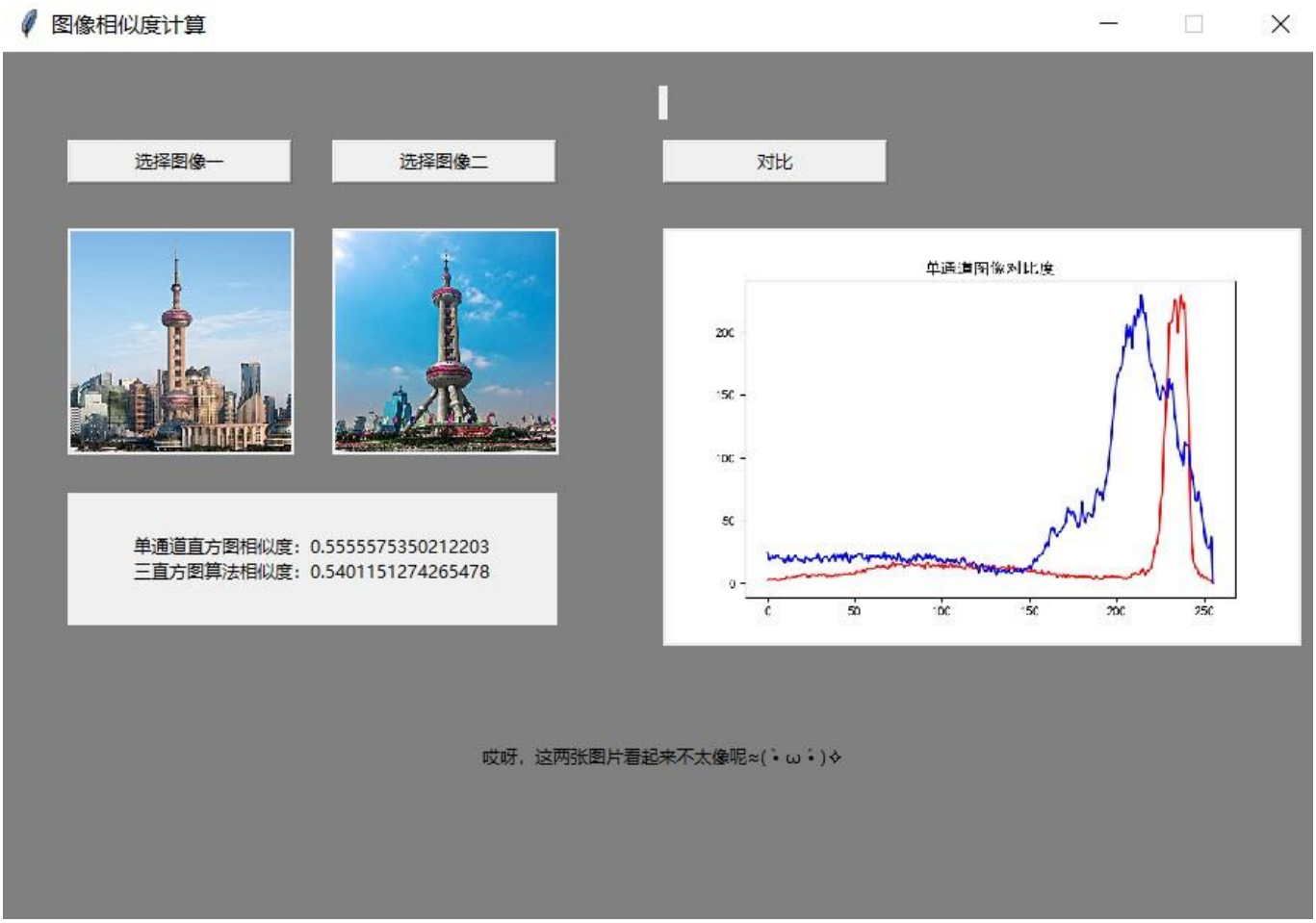
```
# 提示标签
lb = Label(root, text='请选择两张图像，进行图像相似度计算', bg='gray')
lb.place(relx=0.1, rely=vec_y1, relwidth=0.8, relheight=0.1)
lb.pack()
lb_file = Label(root,text='')
lb_file.pack()

# 输出结果的标签
lb_res = Label(root, text='--这是一个利用图像直方图特征计算图像相似度的程序--',
bg='gray')
lb_res.place(relx=0.1, rely=0.7, relwidth=0.8, relheight=0.2)

# 按钮一选择第一张图片
btn1 = Button(root, text='选择图像一', command=display1)
btn1.place(relx=vec_x1, rely=vec_y1, relwidth=0.17, relheight=0.05)

# 按钮二选择第二张图片
btn2 = Button(root, text='选择图像二', command=display2)
btn2.place(relx=vec_x2, rely=vec_y1, relwidth=0.17, relheight=0.05)

# 按钮三进行图片相似度对比
btn3 = Button(root, text='对比', command=compare)
btn3.place(relx=vec_x3, rely=vec_y1, relwidth=0.17, relheight=0.05)

# 设置提示1
txt1 = Text(root)
txt1.place(relx=vec_x1, rely=vec_y2, relwidth=0.17, relheight=0.25)
txt1.insert(END, "    请选择第一张图像")
txt1.config(state=DISABLED)
```

```
# 设置提示2
txt2 = Text(root)
txt2.place(relx=vec_x2, rely=vec_y2, relwidth=0.17, relheight=0.25)
txt2.insert(END, "    请选择第二张图像")
txt2.config(state=DISABLED)

# 设置结果对比区域
txt3 = Text(root)
txt3.place(relx=vec_x1, rely=0.5, relwidth=0.37, relheight=0.15)
txt3.insert(END, "")
txt3.config(state=DISABLED)

lb_cmp = Label(root, text='')
lb_cmp.place(relx=vec_x1, rely=0.5, relwidth=0.37, relheight=0.15)

# 设置直方图展示区域
txt3 = Text(root)
txt3.place(relx=0.5, rely=0.2, width=430, height=280)
txt3.insert(END, "    直方图展示区")
txt3.config(state=DISABLED)
```

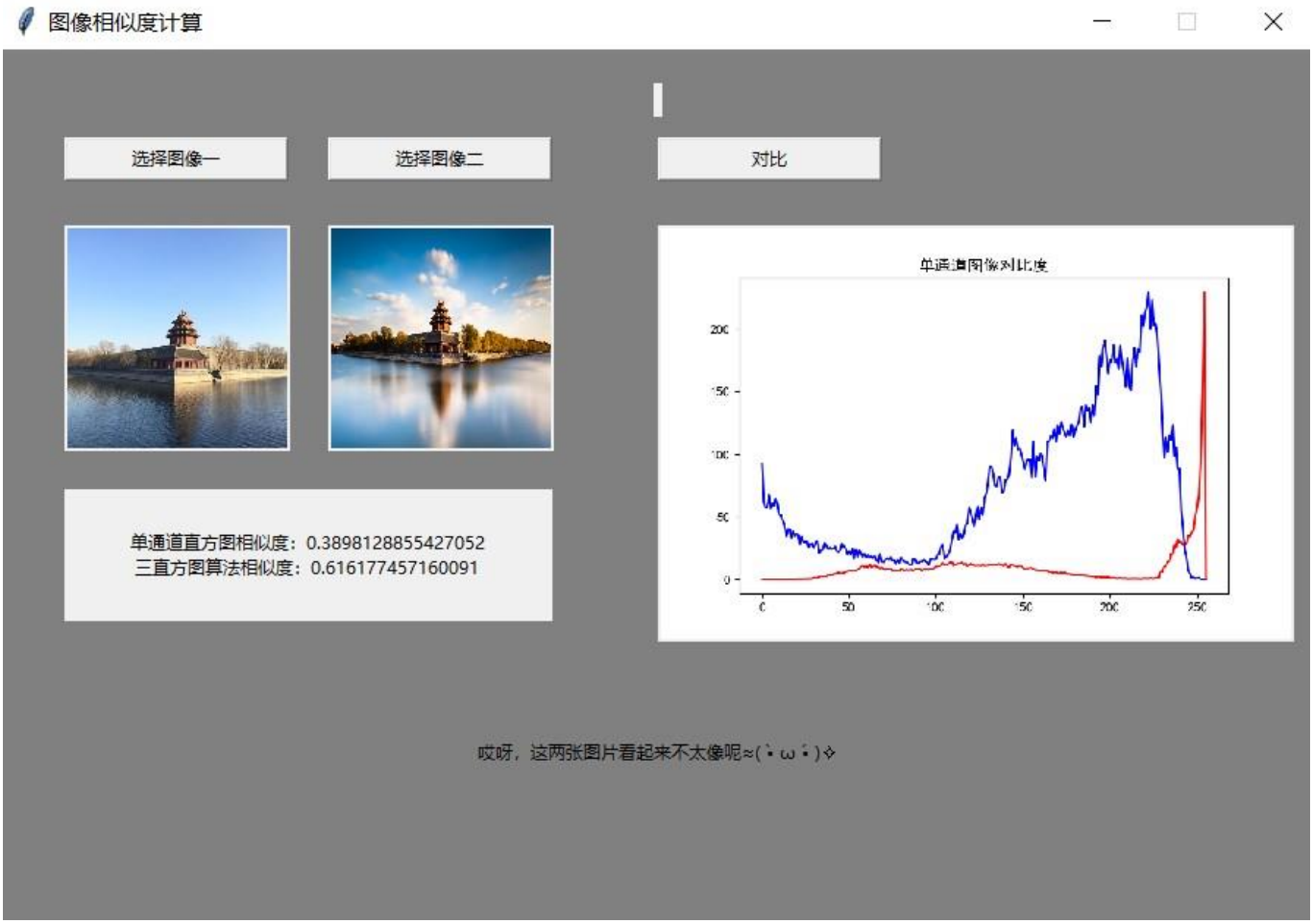
三、实验结果分析第一组



第二组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.5555575350212203
三直方图算法相似度：0.5401151274265478
单通道图像对比度
哎呀，这两张图片看起来不太像呢≈(ˉ ω ˉ)✧

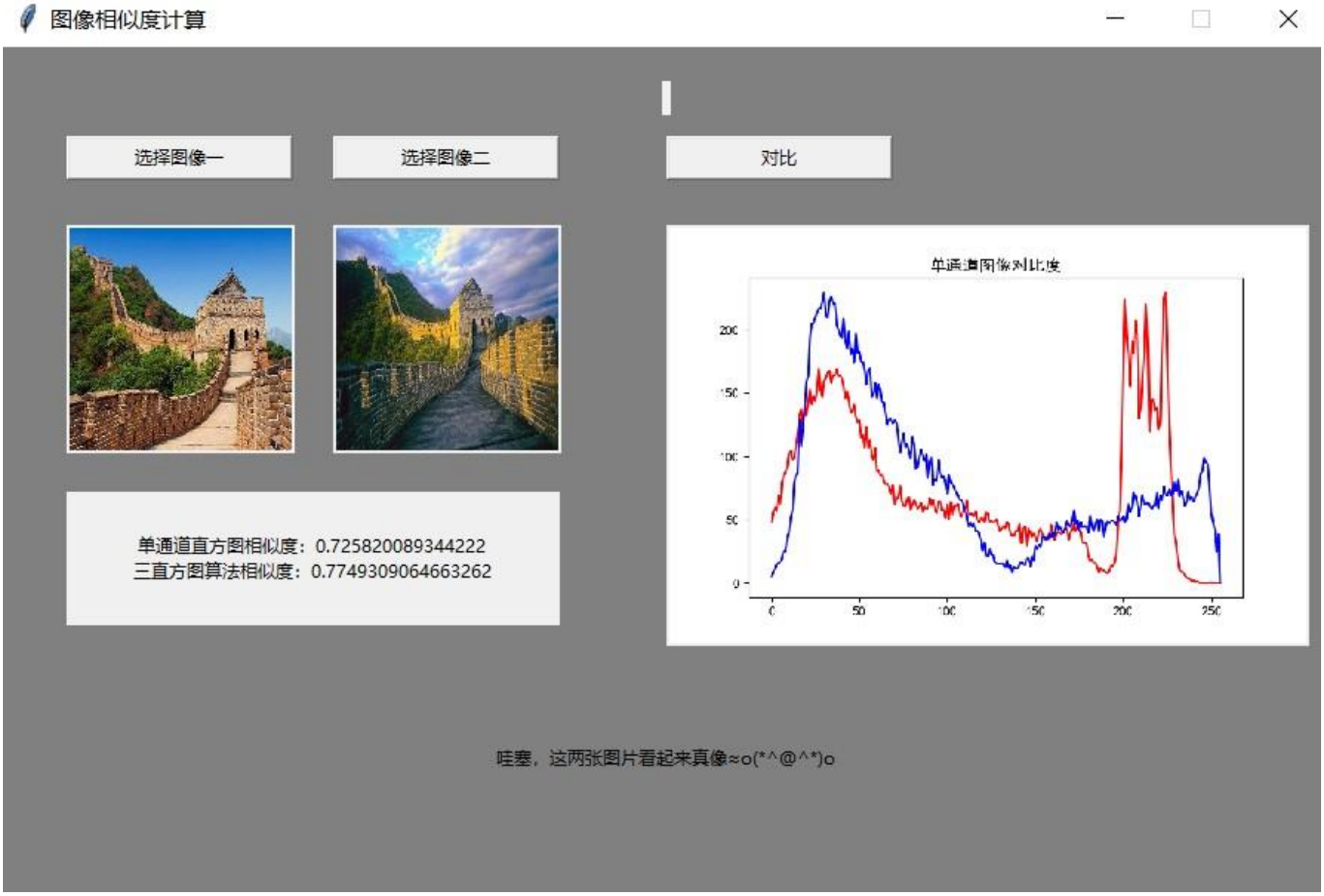
第三组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.3898128855427052
三直方图算法相似度：0.616177457160091
单通道图像对比度
哎呀，这两张图片看起来不太像呢≈(ˉ ω ˉ)✧

第四组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.7548165070343693
三直方图算法相似度：0.6946210936079839
单通道图像对比度
哇塞，这两张图片看起来真像≈o(*^@^*)o

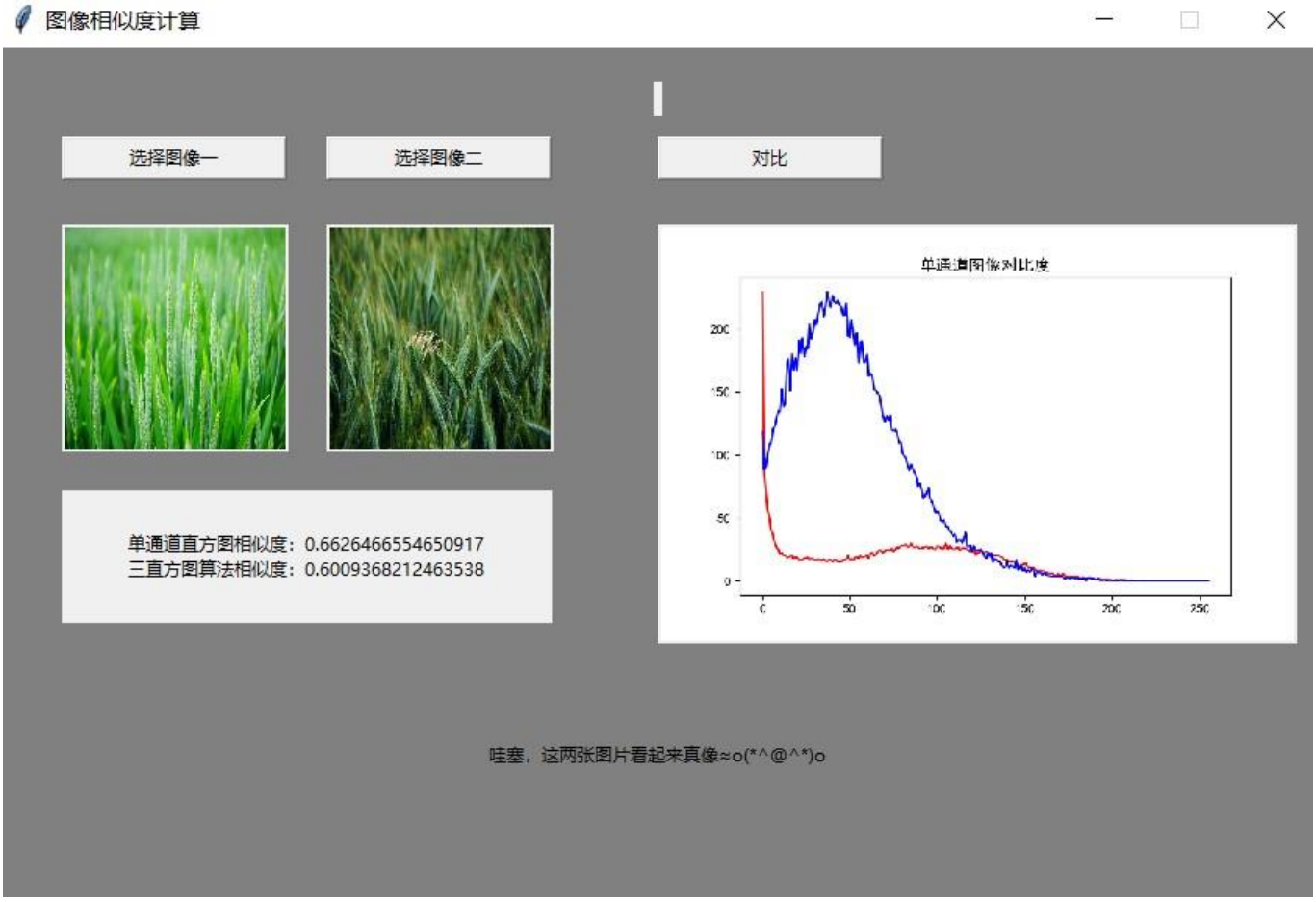
第五组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.725820089344222
三直方图算法相似度：0.7749309064663262
单通道图像对比度
哇塞，这两张图片看起来真像≈o(*^@^*)o

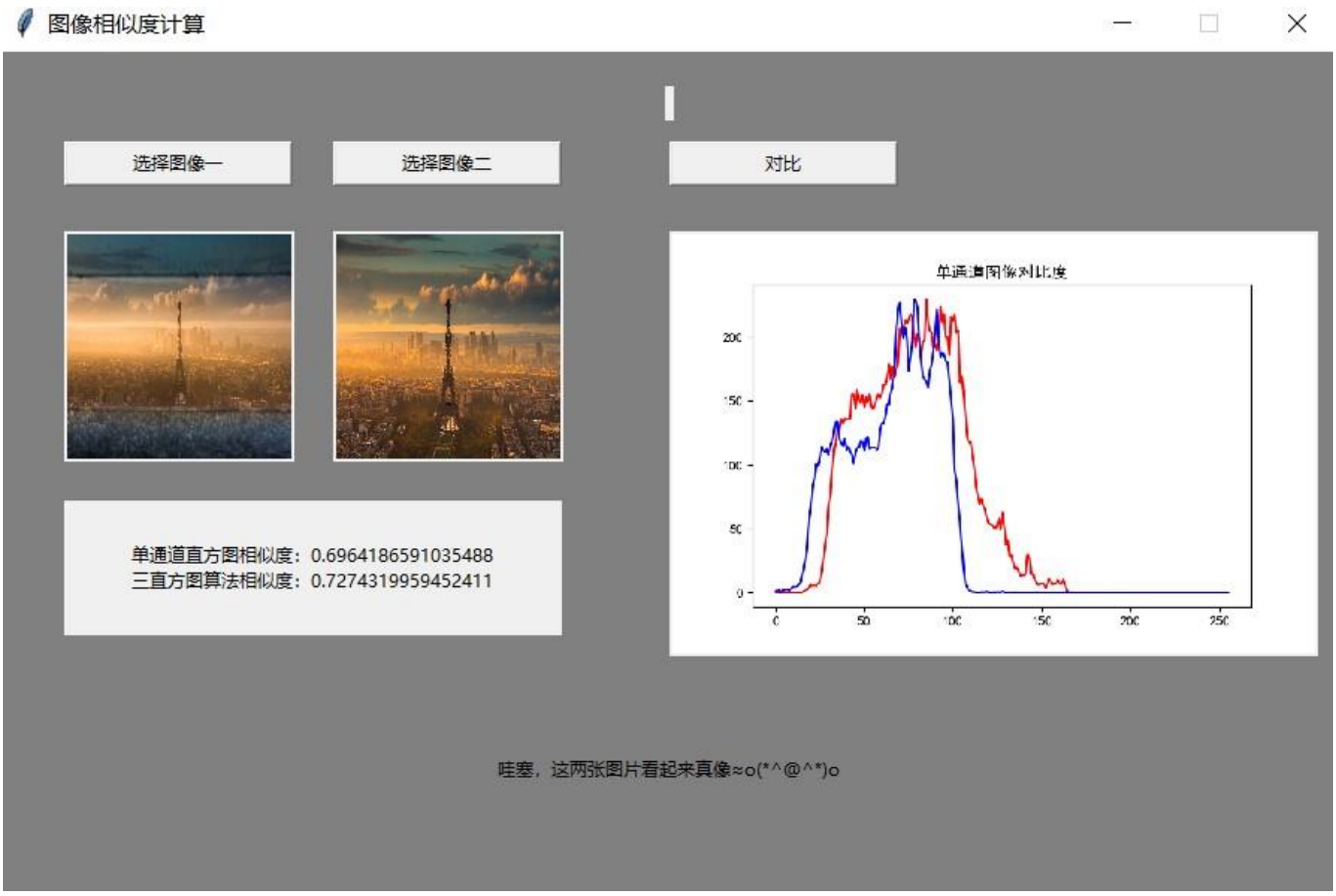
第六组


图像相似度计算
选择图像一
选择图像二
对比
单通道图像对比度
单通道直方图相似度：0.614485819618771
三直方图算法相似度：0.6366483360253117
哇塞，这两张图片看起来真像≈o(*^@^*)o

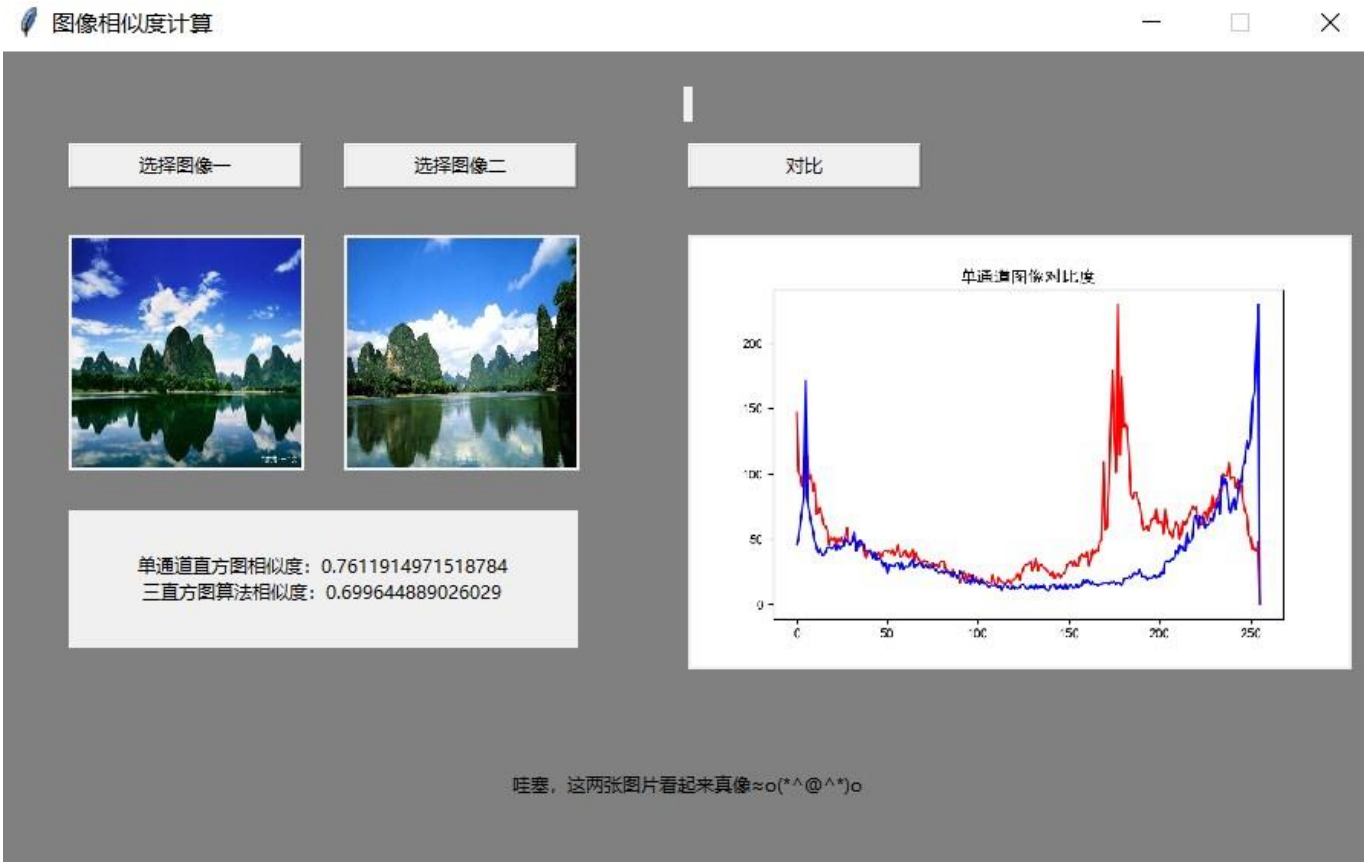
第七组


图像相似度计算
选择图像一
选择图像二
对比
单通道图像对比度
单通道直方图相似度：0.6626466554650917
三直方图算法相似度：0.6009368212463538
哇塞，这两张图片看起来真像≈o(*^@^*)o

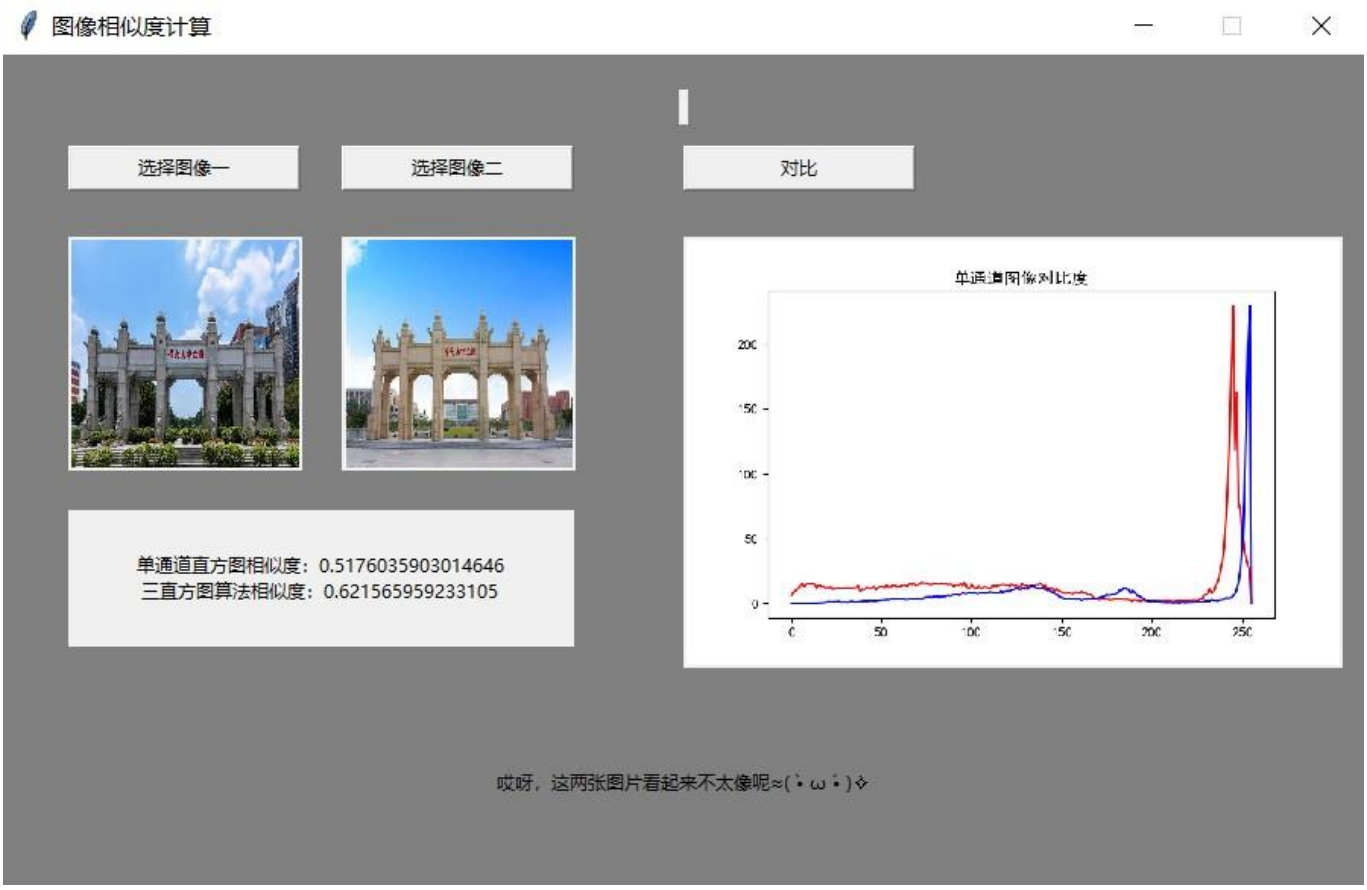
第八组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.6964186591035488
三直方图算法相似度：0.7274319959452411
单通道图像对比度
哇塞，这两张图片看起来真像≈o(*^@^*)o

第九组


图像相似度计算
选择图像一
选择图像二
对比
单通道直方图相似度：0.7611914971518784
三直方图算法相似度：0.699644889026029
单通道图像对比度
哇塞，这两张图片看起来真像≈o(*^@^*)o

第十组



通过以上十组图像对的对比，可以看到，对于颜色分布近似的图像对，单通道直方图和三通道直方图的结果相差不大；而对于颜色分布差异较大的图像对，两者的预测结果出现较大偏差，以第三组表现最为明显。

以上结果显示出直方图对比图像相似度的可行性。然而，虽然比较直方图是最简单、最快捷的方法，且在大部分示例中，都能得到较好的相似度预测，但它的应用场景受限。直方图主要根据图片亮度分布进行比较，所选用的参数较少，为了得到更加精确的结果，应当在图像相似度对比中引入更多参考变量。