

# 目录

目录.....	I
摘要.....	i
<b>第一章 绪论</b> .....	<b>1</b>
1.1 实训目的与内容.....	1
1.2 图像检索系统工作流程.....	1
1.3 图像检索系统框架.....	1
<b>第二章 基于颜色信息的图像检索</b> .....	<b>2</b>
2.1 核心算法描述.....	2
2.2 系统架构说明.....	2
2.3 实验结果及分析.....	3
<b>第三章 基于纹理信息的图像检索</b> .....	<b>3</b>
3.1 核心算法描述.....	3
3.2 系统架构说明.....	3
3.3 实验结果及分析.....	3
<b>第四章 基于形状信息的图像检索</b> .....	<b>4</b>
4.1 核心算法描述.....	4
4.2 系统架构说明.....	4
4.3 实验结果及分析.....	4
<b>第五章 基于综合信息的图像检索</b> .....	<b>5</b>
5.1 核心算法描述.....	5
5.2 系统架构说明.....	5
5.3 实验结果及分析.....	5
<b>第六章 基于LBP的图像检索</b> .....	<b>6</b>
6.1 核心算法描述.....	6
6.2 系统架构说明.....	6
6.3 实验结果及分析.....	6
<b>第七章 基于VGG16的图像检索</b> .....	<b>7</b>
7.1 核心算法描述.....	7
7.2 系统架构说明.....	7
7.3 实验结果及分析.....	7
<b>第八章 基于Resnet50的图像检索</b> .....	<b>8</b>
8.1 核心算法描述.....	8
8.2 系统架构说明.....	8
8.3 实验结果及分析.....	8
<b>第九章 基于DenseNet121的图像检索</b> .....	<b>9</b>
9.1 核心算法描述.....	9

9.2 系统架构说明.....	9
9.3 实验结果及分析.....	9
<b>第十章 TOPSIS评价模型.....</b>	<b>10</b>
10.1 TOPSIS评价模型概念.....	10
10.2 TOPSIS方法的基本思路.....	10
10.3 TOPSIS方法的优点.....	10
10.4 TOPSIS算法的步骤.....	10
<b>第十一章 总结.....</b>	<b>11</b>
<b>参考文献.....</b>	<b>28</b>

## 摘要

本课题的基本目的是要求实现基于视觉特征的图像检索，具体包括：实现基于颜色信息、纹理信息、形状信息的图像检索。其中颜色信息采用的是HSV中心距法进行特征提取，纹理信息是采用灰度共生矩阵法（GLCM），形状信息采用的是形状Hu不变矩法。本文在颜色、纹理、形状的基础上进行三种信息的综合，实现了综合信息的图像检索。完成图像检索系统的基础功能后，本文又采用了LBP对图像进行局部纹理特征进行提取并集成到图像检索系统中。此外，在尝试了上述较为传统的特征提取与图像检索后，本文又引入了VGG16、Resnet50、Densenet121这三种经典的神经网络结构进行图像的特征提取与图像检索，并集成到图像检索系统中。为了评估上述各种图像检索方法的优劣，本文引入TOPSIS评价模型对各个图像检索方法进行客观的评分，实现对各图像检索方法的分析与比较。

关键词：CBIR；HSV；GLCM；Hu矩；LBP；VGG16；Resnet50；Densenet121；TOPSIS评价模型

# 第一章 绪论

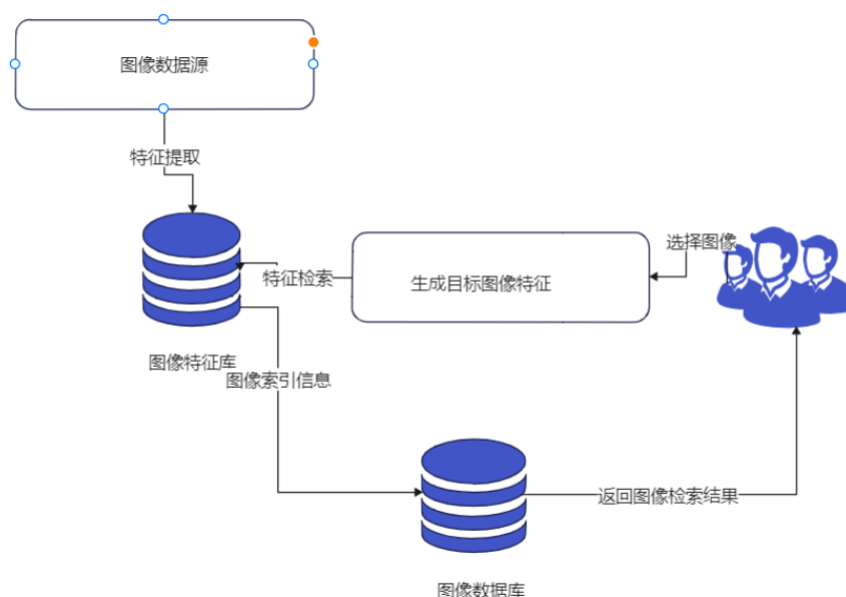
## 1.1 实训目的与内容

基于内容的图像检索系统（Content Based Image Retrieval，以下简称 CBIR），是计算机视觉领域中关注大规模数字图像内容检索的研究分支。典型的 CBIR 系统，允许用户输入一张图像，在图像数据库（或本地机、或网络）中查找具有相同或相似内容的其它图片。本实训的基本功能要求是实现基于视觉特征的图像检索。具体包括：（1）实现基于颜色信息的图像检索，可通过颜色直方图、颜色矩、颜色一致性矢量等方法来实现。（2）实现基于纹理特征的图像检索，可从四个方面进行：统计法、结构法、模型法、频谱法。（3）实现基于形状特征的图像检索，可分别从图像的边缘信息和区域信息来实现。（4）实现基于综合信息的图像检索。

## 1.2 图像检索工作流程

基于内容的图像检索技术是对输入的图像进行分析并分类统一建模，提取其颜色、形状、纹理、轮廓和空间位置等特征，建立特征索引，存储于特征数据库中。检索时，用户提交查询的源图像，通过用户接口设置查询条件，可以采用一种或几种的特征组合来表示，然后在图像数据库中提取出查询到的所需关联图像，按照相似度从大到小的顺序，反馈给用户。用户可根据自己的满意程度，选择是否修改查询条件，继续查询，以达到满意的查询结果。

## 1.3 图像检索系统框架



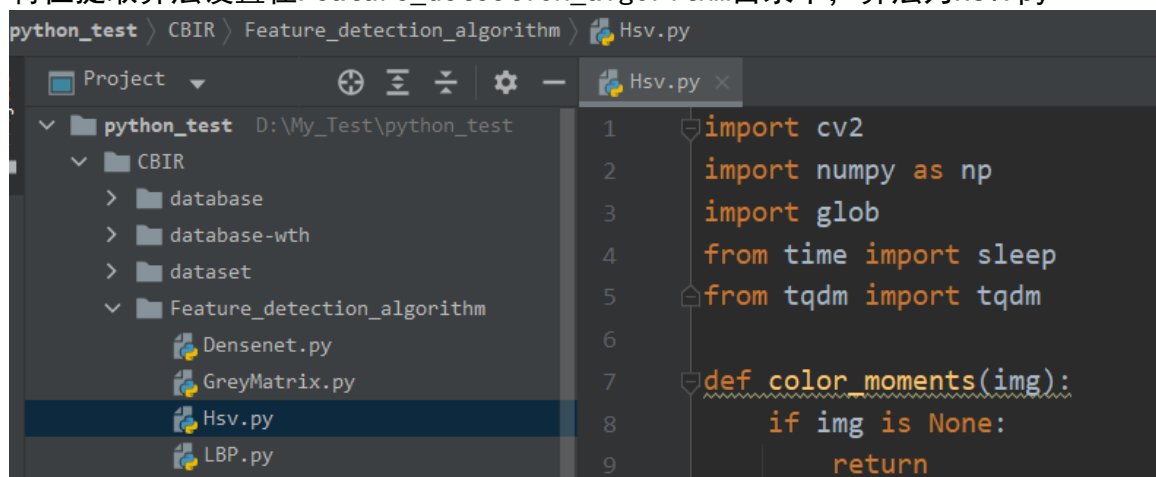
## 第二章 根据颜色信息的图像检索

### 2.1 核心算法描述

颜色信息图像检索部分采用的是HSV中心矩法。颜色矩 (color moments) 是由Stricker 和Orengo所提出的一种非常简单而有效的颜色特征。这种方法的数学基础在于图像中任何的颜色分布均可以用它的矩来表示。此外, 由于颜色分布信息主要集中在低阶矩中, 因此仅采用颜色的一阶矩 (mean)、二阶矩 (variance) 和三阶矩 (skewness) 就足以表达图像的颜色分布。与颜色直方图相比, 该方法的另一个好处在于无需对特征进行向量化。因此, 图像的颜色矩一共只需要9个分量 (3个颜色分量, 每个分量上3个低阶矩), 与其他的颜色特征相比是非常简洁的。在实际应用中, 为避免低次矩较弱的分辨能力, 颜色矩常和其它特征结合使用, 而且一般在使用其它特征前, 起到过滤缩小范围 (narrow down) 的作用。HSV 中心矩法是基于HSV空间的因此需要将RGB空间转换为HSV空间[1]。

### 2.2 系统架构说明

特征提取算法设置在Feature\_detection\_algorithm目录中, 算法为Hsv.py



The screenshot shows an IDE with a project named 'python\_test' located at 'D:\My\_Test\python\_test'. The project structure includes a 'CBIR' directory with sub-directories 'database', 'database-wth', 'dataset', and 'Feature\_detection\_algorithm'. The 'Feature\_detection\_algorithm' directory contains files 'Densenet.py', 'GreyMatrix.py', 'Hsv.py', and 'LBP.py'. The 'Hsv.py' file is open in the editor, showing the following code:

```
1 import cv2
2 import numpy as np
3 import glob
4 from time import sleep
5 from tqdm import tqdm
6
7 def color_moments(img):
8     if img is None:
9         return
```

核心算法为:

```
def color_moments(img):
    if img is None:
        return
    # 将BGR转换为HSV色彩空间
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # 分割通道 - h, s, v
    h, s, v = cv2.split(hsv)
    # 初始化颜色特征
    color_feature = []
    # 第一个中心矩--平均
    h_mean = np.mean(h) # np.sum(h)/float(N)
    s_mean = np.mean(s) # np.sum(s)/float(N)
    v_mean = np.mean(v) # np.sum(v)/float(N)
    color_feature.extend([h_mean, s_mean, v_mean])
```

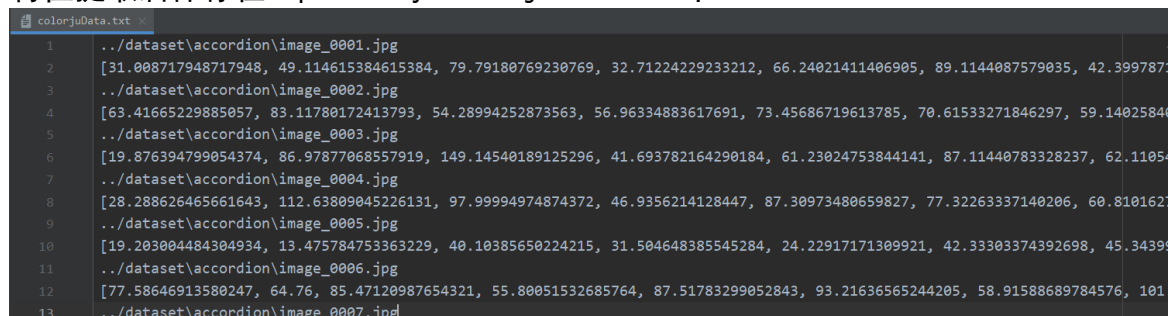
```

# 第二中心矩--标准差
h_std = np.std(h) # np.sqrt(np.mean(abs(h - h.mean())**2))
s_std = np.std(s) # np.sqrt(np.mean(abs(s - s.mean())**2))
v_std = np.std(v) # np.sqrt(np.mean(abs(v - v.mean())**2))
color_feature.extend([h_std, s_std, v_std])
# 第三中心矩--偏斜度的第三根
h_skewness = np.mean(abs(h - h.mean())**3)
s_skewness = np.mean(abs(s - s.mean())**3)
v_skewness = np.mean(abs(v - v.mean())**3)
h_thirdMoment = h_skewness**(1./3)
s_thirdMoment = s_skewness**(1./3)
v_thirdMoment = v_skewness**(1./3)
color_feature.extend([h_thirdMoment, s_thirdMoment, v_thirdMoment])

return color_feature

```

特征提取后保存在Repository/colorJuData.txt中

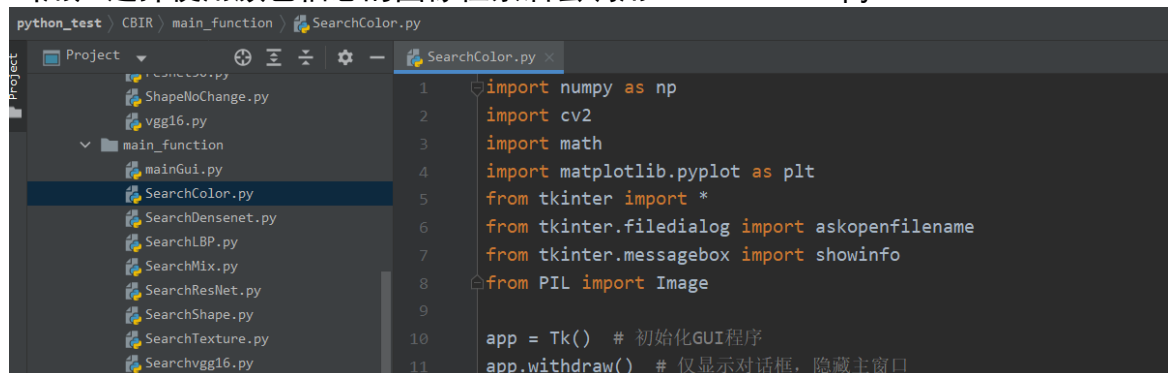


```

colorJuData.txt
1  ../dataset/accordion/image_0001.jpg
2  [31.008717948717948, 49.114615384615384, 79.79180769230769, 32.71224229233212, 66.24021411406905, 89.1144087579035, 42.399787
3  ../dataset/accordion/image_0002.jpg
4  [63.41665229885057, 83.11780172413793, 54.28994252873563, 56.96334883617691, 73.45686719613785, 70.61533271846297, 59.1402584
5  ../dataset/accordion/image_0003.jpg
6  [19.876394799054374, 86.97877068557919, 149.14540189125296, 41.693782164290184, 61.23024753844141, 87.11440783328237, 62.1105
7  ../dataset/accordion/image_0004.jpg
8  [28.288626465661643, 112.63809045226131, 97.99994974874372, 46.9356214128447, 87.30973480659827, 77.32263337140206, 60.810162
9  ../dataset/accordion/image_0005.jpg
10 [19.203004484304934, 13.475784753363229, 40.10385650224215, 31.504648385545284, 24.22917171309921, 42.33303374392698, 45.3439
11 ../dataset/accordion/image_0006.jpg
12 [77.58646913580247, 64.76, 85.47120987654321, 55.80051532685764, 87.51783299052843, 93.21636565244205, 58.91588689784576, 101
13 ../dataset/accordion/image_0007.jpg

```

当用户选择使用颜色信息的图像检索后会调用SearchColor.py

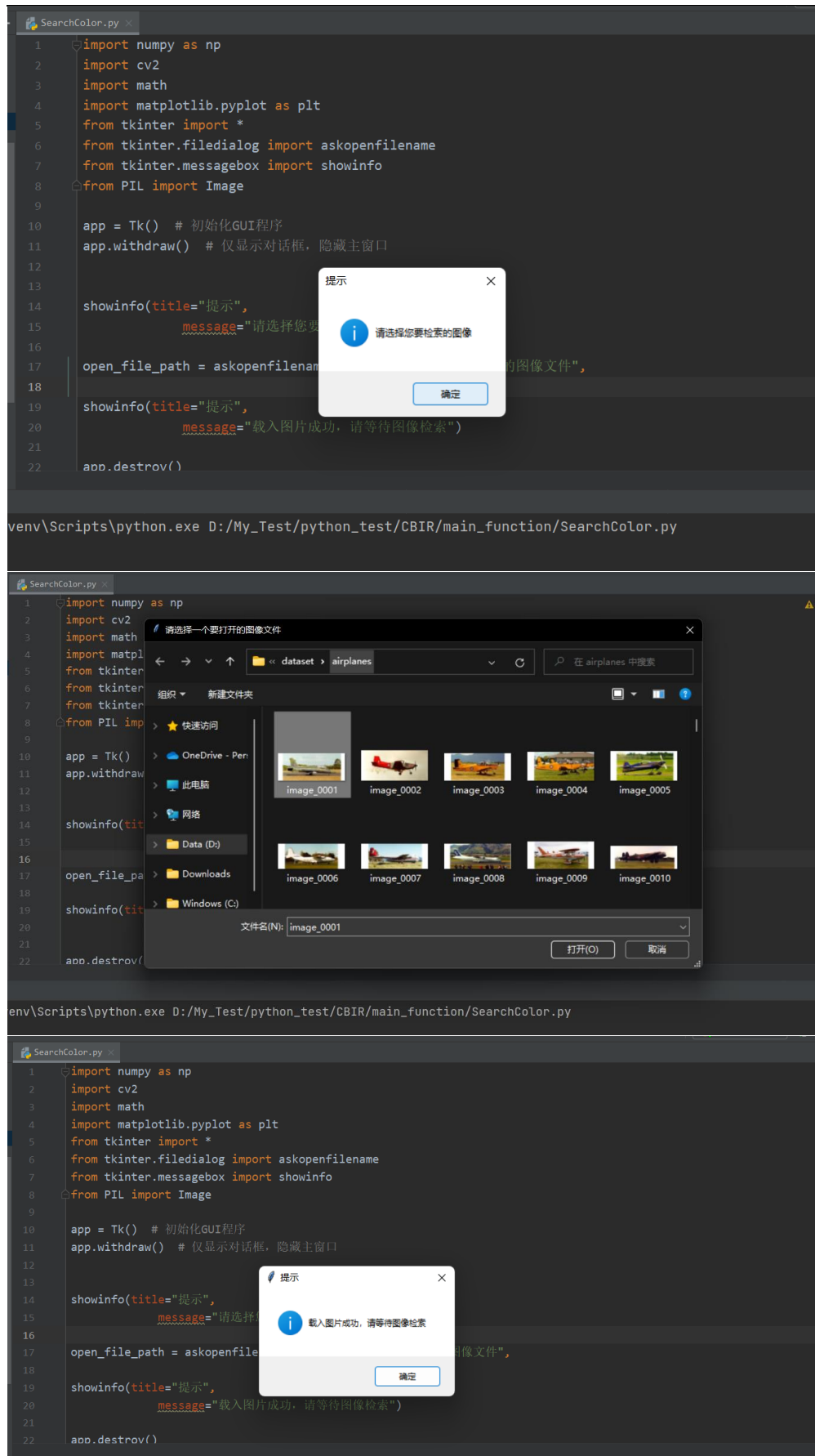


```

python_test > CBIR > main_function > SearchColor.py
Project
  SearchColor.py
  ShapeNoChange.py
  vgg16.py
  main_function
    mainGui.py
    SearchColor.py
    SearchDensenet.py
    SearchLBP.py
    SearchMix.py
    SearchResNet.py
    SearchShape.py
    SearchTexture.py
    Searchvgg16.py
1  import numpy as np
2  import cv2
3  import math
4  import matplotlib.pyplot as plt
5  from tkinter import *
6  from tkinter.filedialog import askopenfilename
7  from tkinter.messagebox import showinfo
8  from PIL import Image
9
10 app = Tk() # 初始化GUI程序
11 app.withdraw() # 仅显示对话框，隐藏主窗口

```

在SearchColor.py中先弹出GUI对话框，引导用户载入待检索的图片



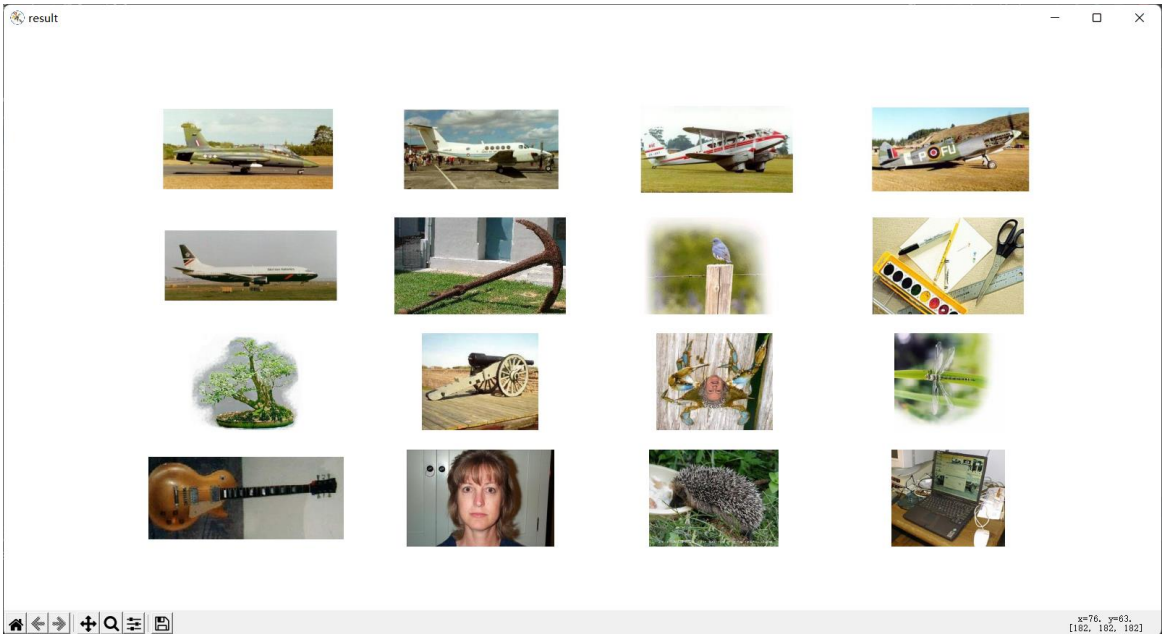
然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

### 2.3 实验结果及分析

测试中以飞机 image\_0001 为例，



检索结果为



检索评分为：0.23571511894814806（1为满分）

```
SearchColor x
D:\My_Test\python_test\venv\Scri
检索评分为：
0.23571511894814806
```

分析：测试输入是一架飞机，返回的16个结果显示只有4个结果成功检索出了飞机，检索评分也不够理想。分析是因为采用的是图像颜色的特征进行检索，图像数据库中有若干图像的颜色会和目标图像的颜色相近，会造成干扰。



## 第三章 基于纹理信息的图像检索

### 3.1 核心算法描述

灰度共生矩阵法 (GLCM, Gray-level co-occurrence matrix), 就是通过计算灰度图像得到它的共生矩阵, 然后透过计算该共生矩阵得到矩阵的部分特征值, 来分别代表图像的某些纹理特征 (纹理的定义仍是难点)。灰度共生矩阵能反映图像灰度关于方向、相邻间隔、变化幅度等综合信息, 它是分析图像的局部模式和它们排列规则的基础。对于灰度共生矩阵的理解, 需要明确几个概念: 方向, 偏移量和灰度共生矩阵的阶数 [3]。

- 方向: 一般计算过程会分别选在几个不同的方向来进行, 常规的是水平方向  $0^\circ$ , 垂直  $90^\circ$ , 以及  $45^\circ$  和  $135^\circ$ ;
- 步距  $d$ : 中心像元 (在下面的例程中进行说明);
- 灰度共生矩阵的阶数: 与灰度图像灰度值的阶数相同, 即当灰度图像灰度值阶数为  $N$  时, 灰度共生矩阵为  $N \times N$  的矩阵;

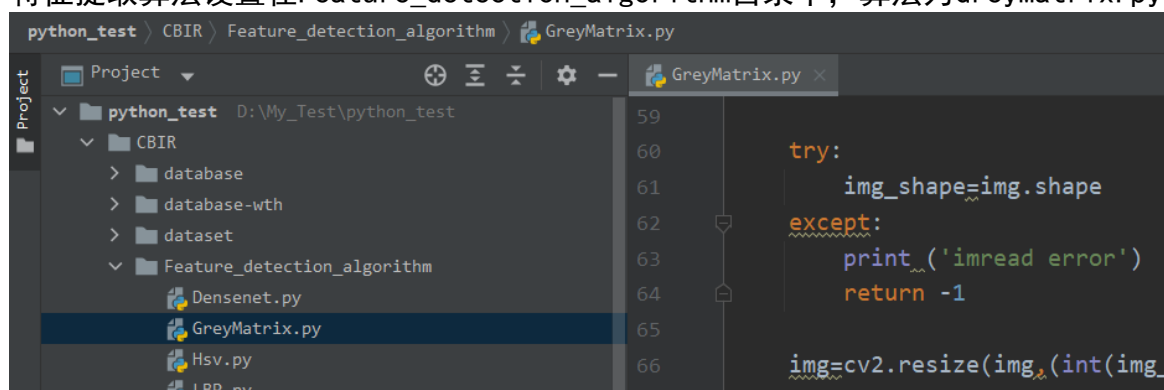
灰度共生矩阵 (Gray-Level Co-occurrence Matrix, GLCM) 统计了灰度图中像素间的灰度值分布规律以区分不同的纹理。

灰度共生矩阵可以定义为一个灰度为 [Math Processing Error] 的像素点与另一个与之对应位置上的像素点的灰度值为 [Math Processing Error] 的概率。那么所有估计的值可以表示成一个矩阵的形式, 以此被称为灰度共生矩阵。如: 根据图像中任意一点 [Math Processing Error] 的灰度值和它所对应的点 [Math Processing Error] 的灰度值可以得到一个灰度值组合 [Math Processing Error]。统计整幅图像每一种灰度值组合出现的概率矩阵 [Math Processing Error] 即为灰度共生矩阵。

由于灰度共生矩阵的维度较大, 一般不直接作为区分纹理的特征, 而是基于它构建的一些统计量作为纹理分类特征。例如 [Math Processing Error] 曾提出了 14 种基于灰度共生矩阵计算出来的统计量: 能量、熵、对比度、均匀性、相关性、方差、和平均、和方差、差平均、差熵、相关信息测度以及最大相关系数 [5]。

### 3.2 系统架构说明

特征提取算法设置在 Feature\_detection\_algorithm 目录中, 算法为 GreyMatrix.py



特征提取后保存在 Repository/GreyMatrixData.txt 中

```
GreyMatrixData.txt x
1 {../dataset/accordion/image_0001.jpg}
2 [0.10908930966469446,13.461179487179496,3.9170404995597865,0.5879656417625276]
3 {../dataset/accordion/image_0002.jpg}
4 [0.16653073721759787,10.920977011494244,3.2127824187583514,0.6665636479617992]
5 {../dataset/accordion/image_0003.jpg}
6 [0.0428307697466593,6.5448699763593385,4.12529482274617,0.6102867178517907]
7 {../dataset/accordion/image_0004.jpg}
8 [0.060282542597694104,5.293400673400676,4.054827169984367,0.614300258371434]
9 {../dataset/accordion/image_0005.jpg}
```


当用户选择使用基于纹理信息的图像检索后会调用SearchTexture.py

```
python_test CBIR main_function SearchTexture.py
Project
  Hsv.py
  LBP.py
  resnet50.py
  ShapelloChange.py
  vgg16.py
  main_function
    mainGui.py
    SearchColor.py
    SearchDensenet.py
    SearchLBP.py
    SearchMix.py
    SearchResNet.py
    SearchShape.py
    SearchTexture.py
    Searchvgg16.py
  models
  SearchTextureGUI.py
SearchTexture.py x
1 import cv2
2 import math
3 import matplotlib.pyplot as plt
4 from tkinter import *
5 from tkinter.filedialog import askopenfilename
6 from tkinter.messagebox import showinfo
7 from PIL import Image
8
9 app = Tk() # 初始化GUI程序
10 app.withdraw() # 仅显示对话框, 隐藏主窗口
11
12
13 showinfo(title="提示",
14          message="请选择您要检索的图像")
15
```

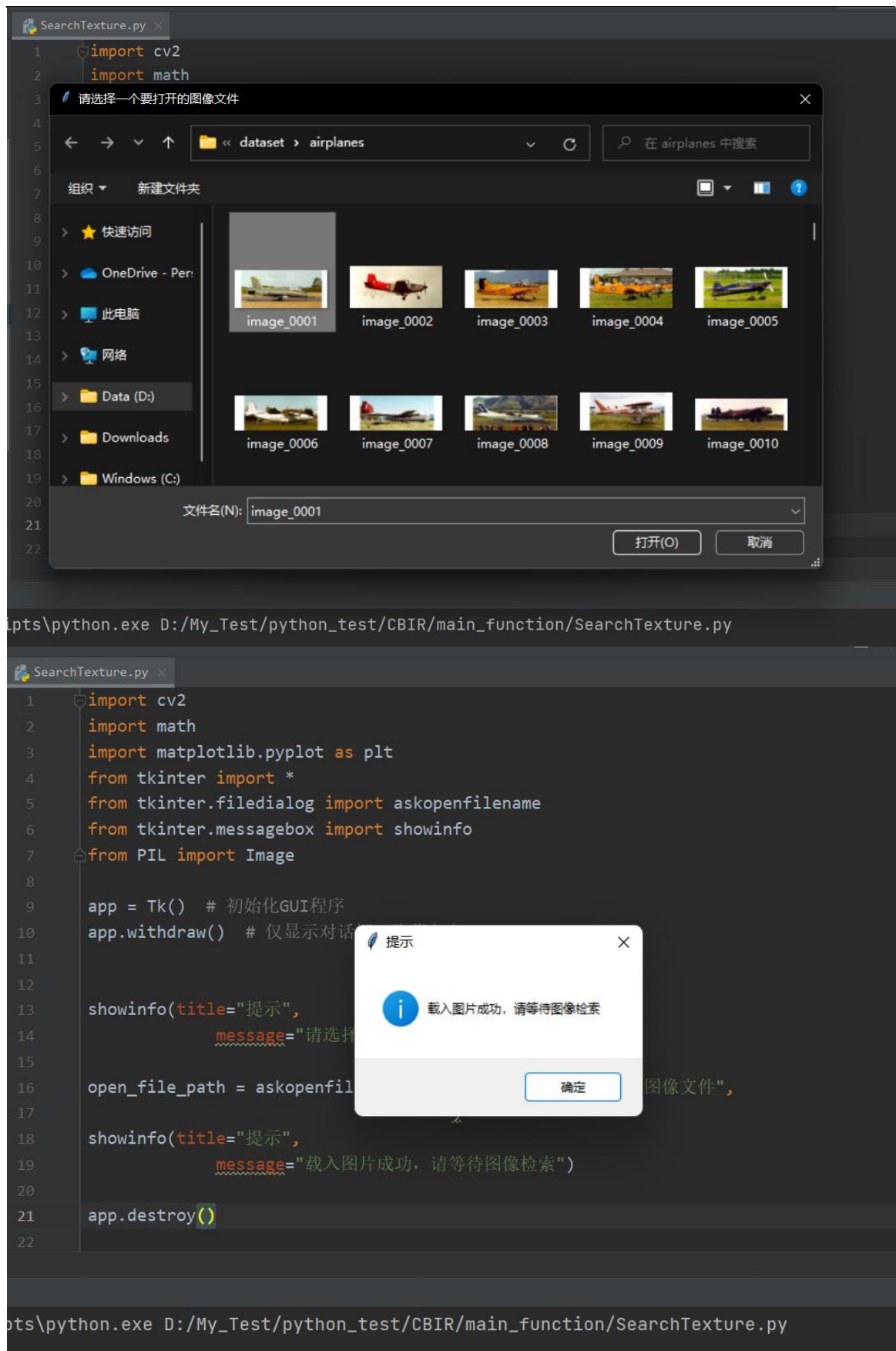
在SearchTexture.py中首先会弹出gui窗口引导用户输入待检索的图像

```
SearchTexture.py x
7 from PIL import Image
8
9 app = Tk() # 初始化GUI程序
10 app.withdraw() # 仅显示对话框, 隐藏主窗口
11
12
13 showinfo(title="提示",
14          message="请选择您要检索的图像")
15
16 open_file_path = askopenfilename(title="请选择您要打开的图像文件",
17                                   message="请选择您要检索的图像")
18
19 showinfo(title="提示",
20          message="请选择您要检索的图像")
21
22 app.destroy()
23
24 file = open("../Repository/GreyMatrixData.txt")
25
26 inputpath = open_file_path
27 inputimg = cv2.imread(inputpath)
28 crop_size = (224, 224)
29 img_new = cv2.resize(inputimg, crop_size, interpolation=cv2.INTER_CUBIC)

```



```
pts\python.exe D:/My_Test/python_test/CBIR/main_function/SearchTexture.py
```



然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后

使用TOPSIS评价模型对此次检索进行评分。

### 3.3 实验结果及分析

测试中以飞机image\_0001为例



检索结果为：

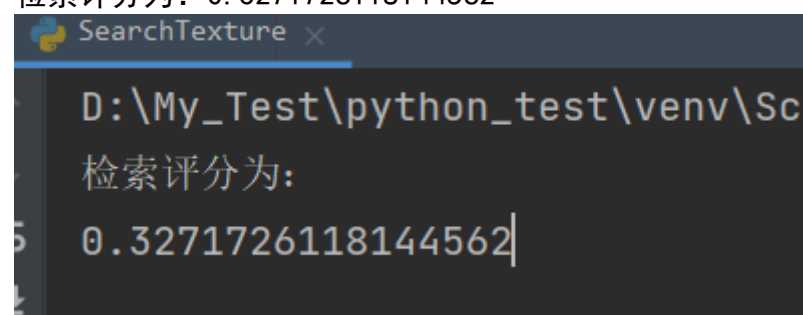
result

- □ ×



检索评分为：0.3271726118144562

检索评分为：0.3271726118144562



分析：纹理信息的检索成功检索出了飞机，相比于颜色信息检索，检索评分也有了一定的提升。尽管肉眼无法辨别出飞机的纹理是否相同，但成功的全部检索出了飞机，准确度上还是可以接受的。

## 第四章 基于形状信息的图像检索

### 4.1 核心算法描述

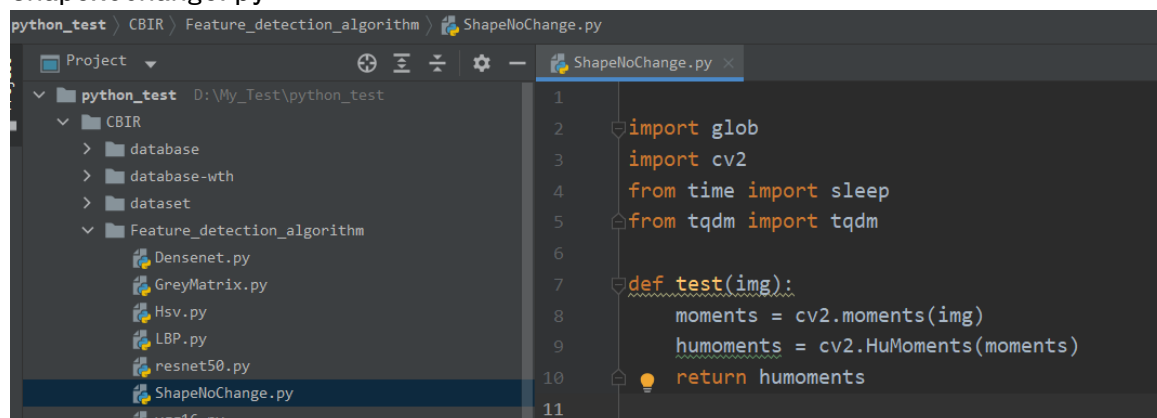
几何矩是由Hu (Visual pattern recognition by moment invariants) 在1962年提出的, 具有平移、旋转和尺度不变性。

由Hu矩组成的特征量对图片进行识别, 优点就是速度很快, 缺点是识别率比较低, 我做过手势识别, 对于已经分割好的手势轮廓图, 识别率也就30%左右, 对于纹理比较丰富的图片, 识别率更是不堪入目, 只有10%左右。这一部分原因是由于Hu不变矩只用到低阶矩(最多也就用到三阶矩), 对于图像的细节未能很好的描述出来, 导致对图像的描述不够完整。

Hu不变矩一般用来识别图像中大的物体, 对于物体的形状描述得比较好, 图像的纹理特征不能太复杂, 像识别水果的形状, 或者对于车牌中的简单字符的识别效果会相对好一些[4]。

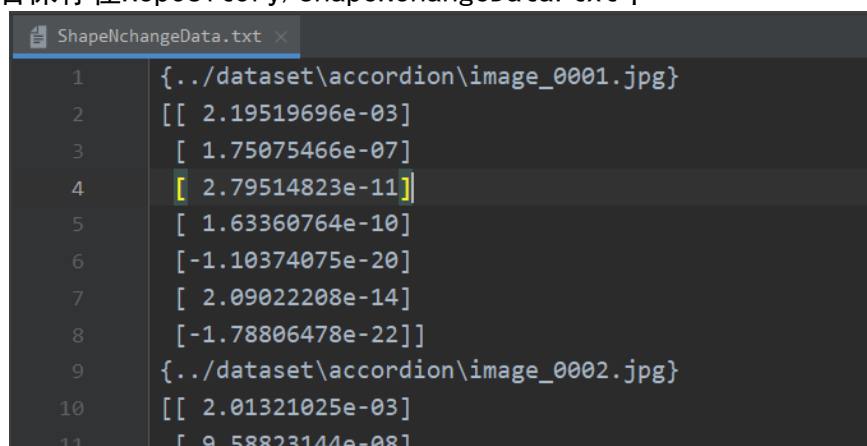
### 4.2 系统架构说明

特征提取算法设置在Feature\_detection\_algorithm目录中, 算法为ShapeNoChange.py



```
python_test > CBIR > Feature_detection_algorithm > ShapeNoChange.py
Project
  python_test D:\My_Test\python_test
    CBIR
      database
      database-with
      dataset
      Feature_detection_algorithm
        Densenet.py
        GreyMatrix.py
        Hsv.py
        LBP.py
        resnet50.py
        ShapeNoChange.py
        vgg16.py
    ShapeNoChange.py
1
2 import glob
3 import cv2
4 from time import sleep
5 from tqdm import tqdm
6
7 def test(img):
8     moments = cv2.moments(img)
9     humoments = cv2.HuMoments(moments)
10    return humoments
11
```

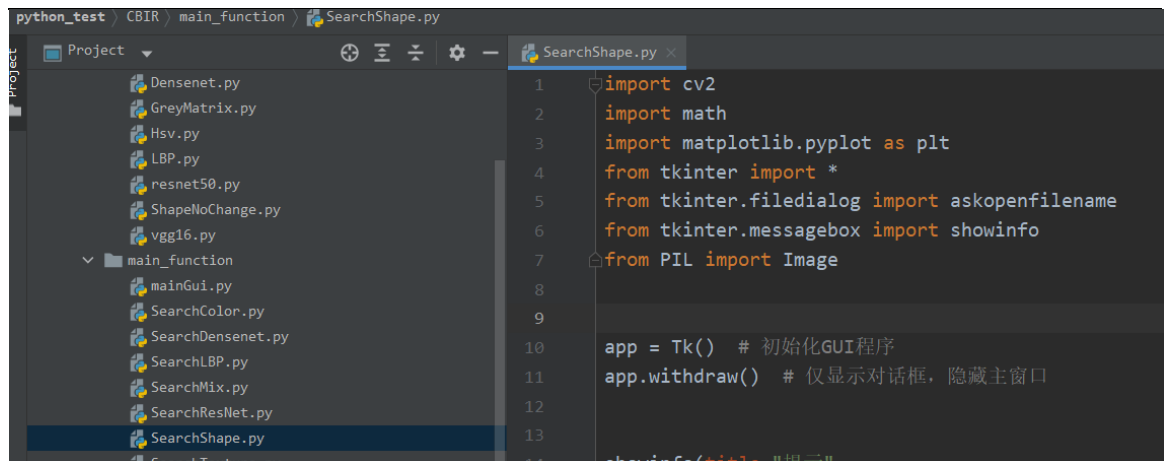
特征提取后保存在Repository/ShapeNchangeData.txt中



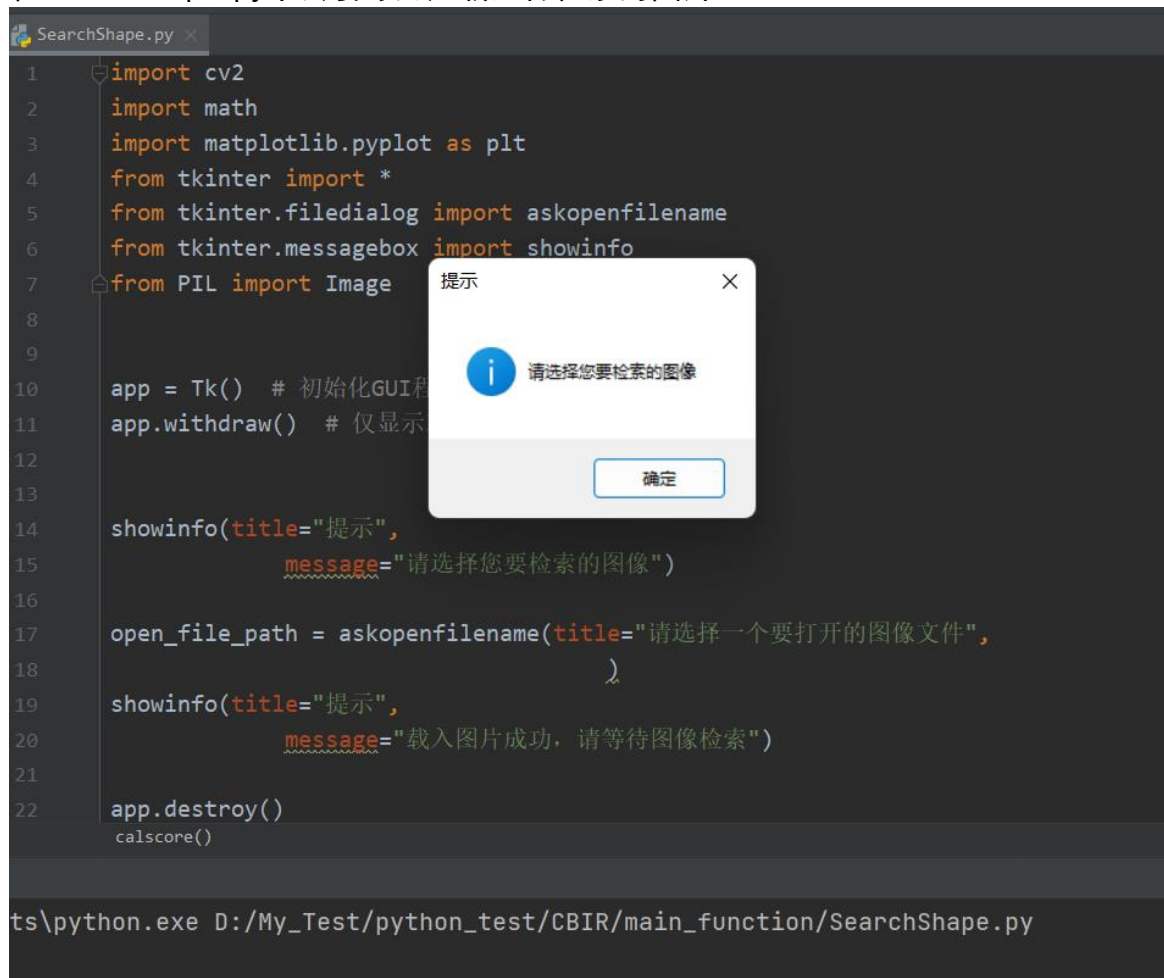
```
ShapeNchangeData.txt
1 {../dataset\accordion\image_0001.jpg}
2 [[ 2.19519696e-03]
3  [ 1.75075466e-07]
4  [ 2.79514823e-11]
5  [ 1.63360764e-10]
6  [-1.10374075e-20]
7  [ 2.09022208e-14]
8  [-1.78806478e-22]]
9 {../dataset\accordion\image_0002.jpg}
10 [[ 2.01321025e-03]
11  [ 9.58823144e-08]
```

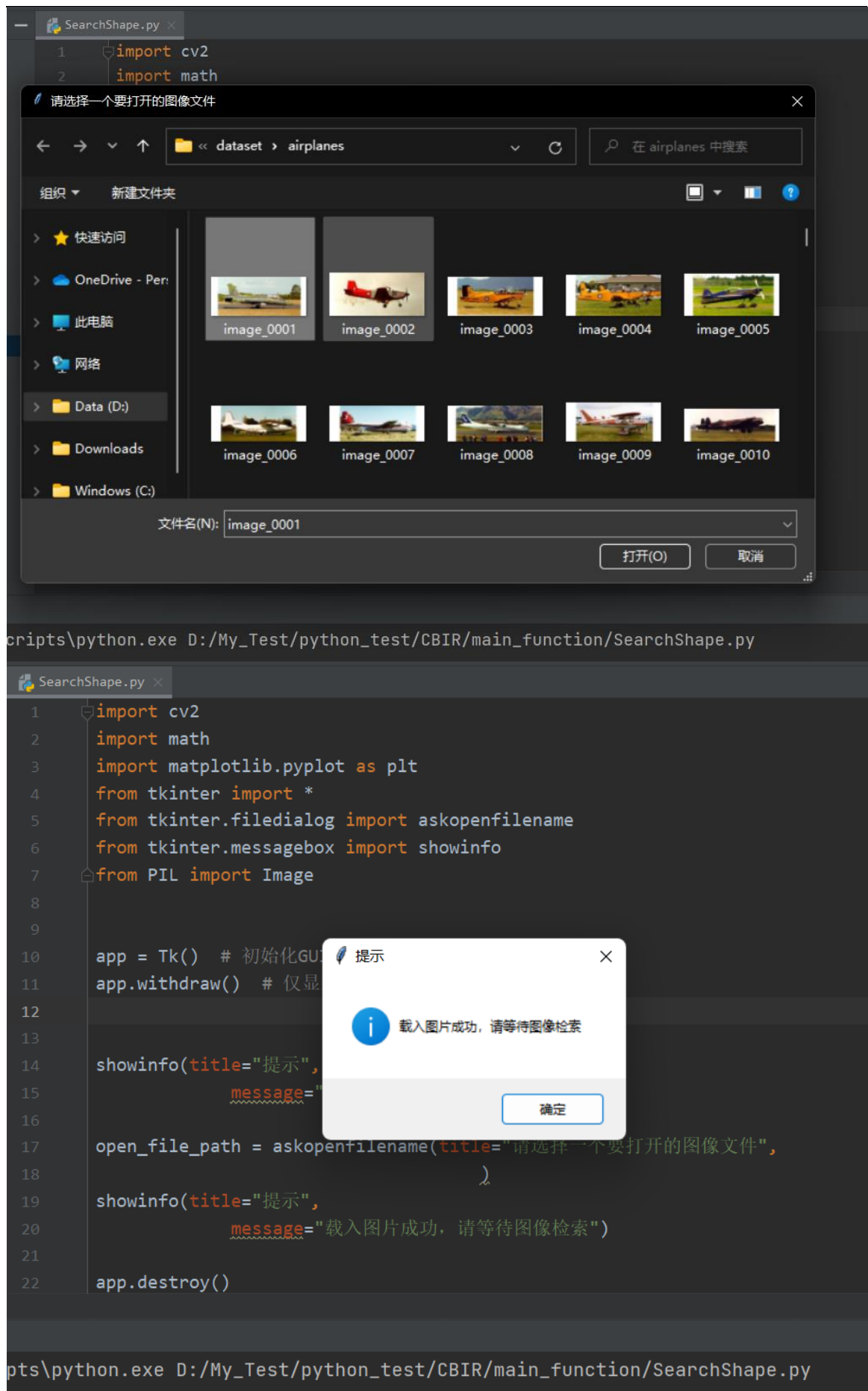
当用户选择使用形状信息进行图像检索时会调用SearchShape.py





在SearchShape.py中会引导用户载入待检索的图片





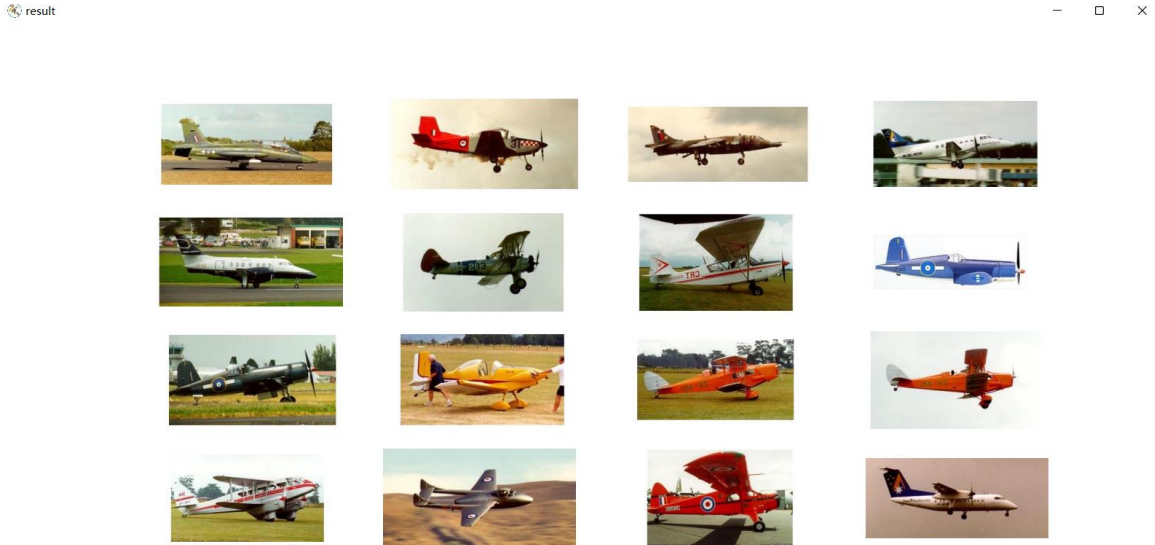
然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

4.3 实验结果及分析

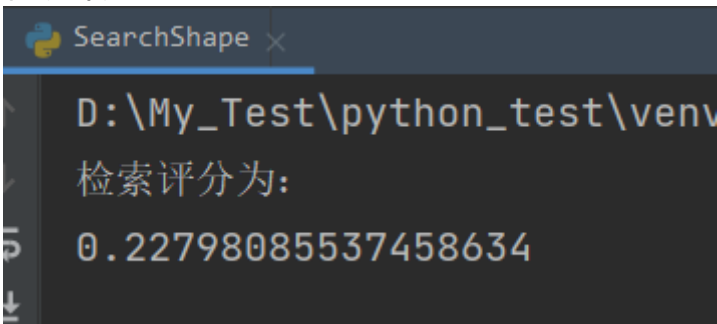
测试中以飞机 image\_0001 为例



检索结果为：



检索评分：0.22798085537458634



分析：

基于形状信息的图像检索评分相比纹理信息的图像检索有了一定的下降，从客观上来说检索效果是变差的，但好的一方面是检索出来的全部都是飞机，从主观上来说检索效果还是可以接受的。



## 第五章 基于综合信息的图像检索

### 5.1 核心算法描述

基于综合信息的图像检索是将颜色、纹理、形状算法结合起来，先分别使用颜色、纹理、形状算法对待图像进行处理，然后载入颜色、纹理、形状的特征库对图像进行检索，当颜色、纹理、形状三方面达到检索要求后才认定为是相似的图像。

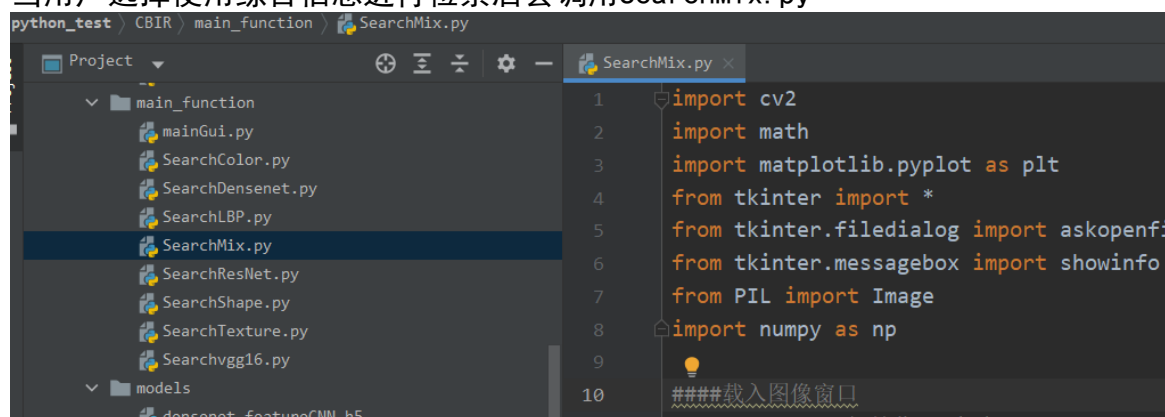
### 5.2 系统架构说明

特征提取的算法分别为Hsv.py、GreyMatrix.py、ShapeNoChange.py

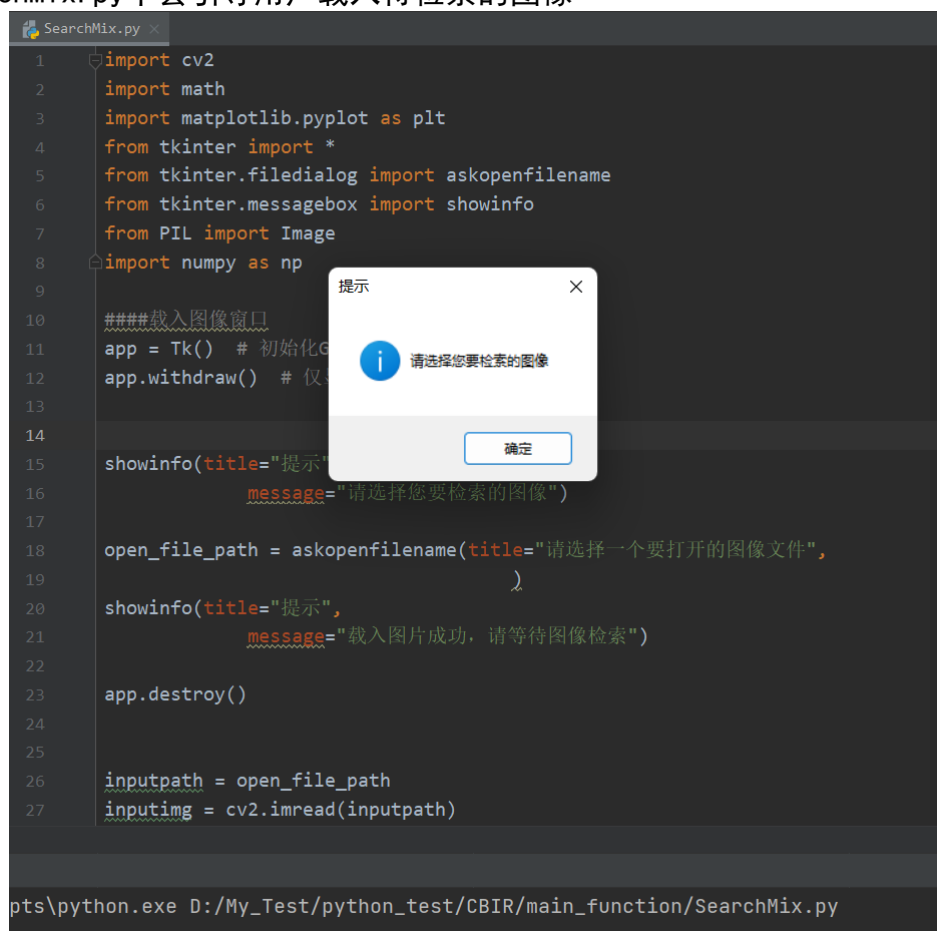
特征提取后保存在colorJuData.txt、GreyMatrixData.txt、

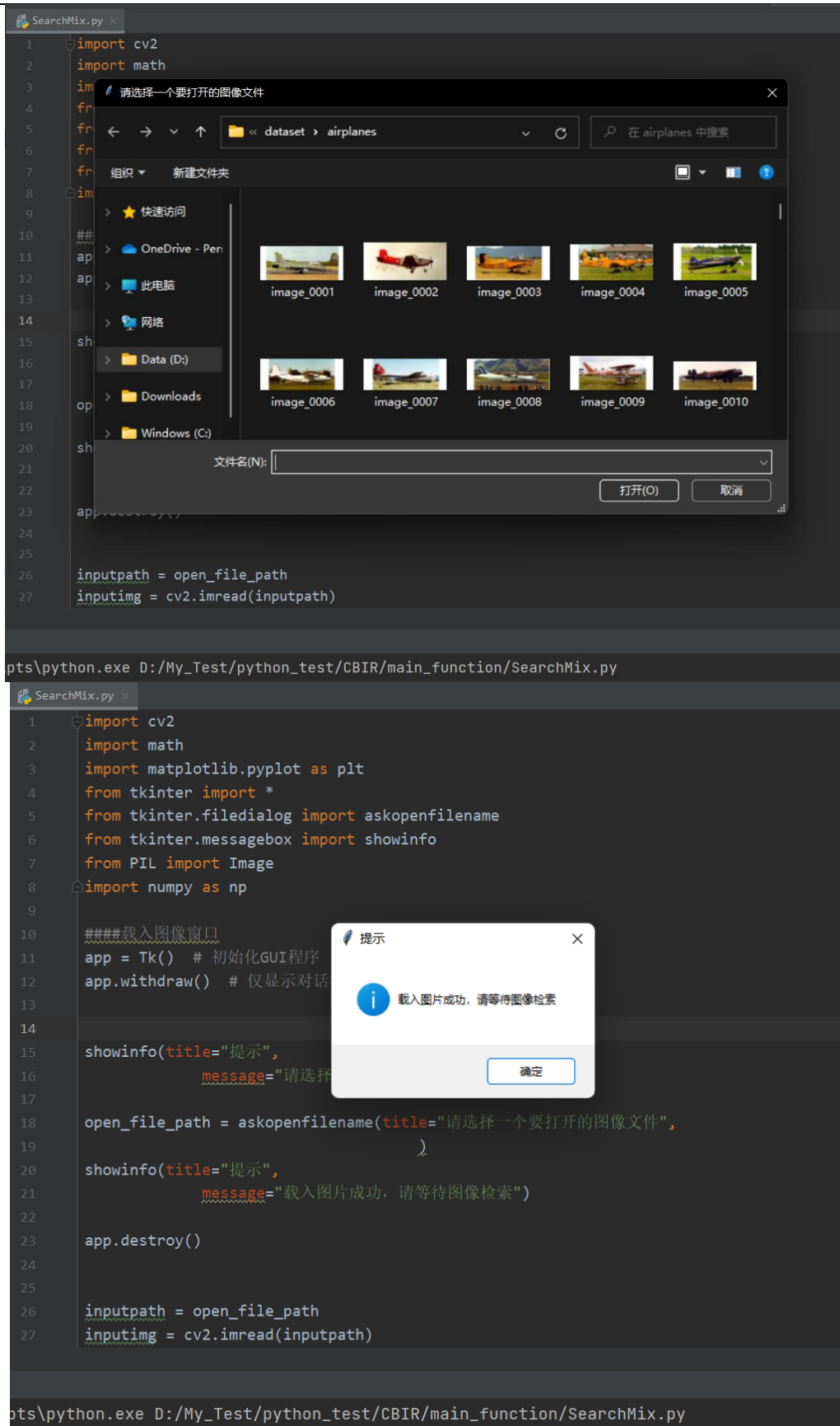
ShapeNchangeData.txt中。

当用户选择使用综合信息进行检索后会调用SearchMix.py



在SearchMix.py中会引导用户载入待检索的图像





然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

### 5.3 实验结果及分析

测试中以飞机image\_0001为例



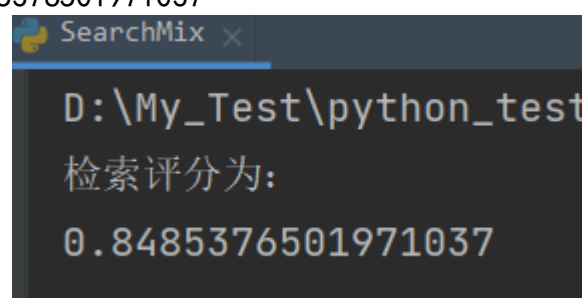
检索结果为:

result



检索评分为: 0.8485376501971037

x=331, y=69,  
[228, 248, 249]



分析:

首先从客观检索评分上看, 综合信息检索的效果相比于颜色、纹理、形状信息检索有了极大的提升, 从主观上来看检索出来的对象确实都是飞机, 因而从客观上还有主观上来说综合信息的检索效果是十分优秀的。

## 第六章 基于LBP的图像检索

### 6.1 核心算法描述

LBP (Local Binary Patterns, 局部二值模式) 是提取局部特征作为判别依据的, 一种有效的纹理描述算子, 度量和提取图像局部的纹理信息, 它具有旋转不变性和灰度不变性等显著的优点, 对光照具有不变性。用于纹理特征提取。LBP的

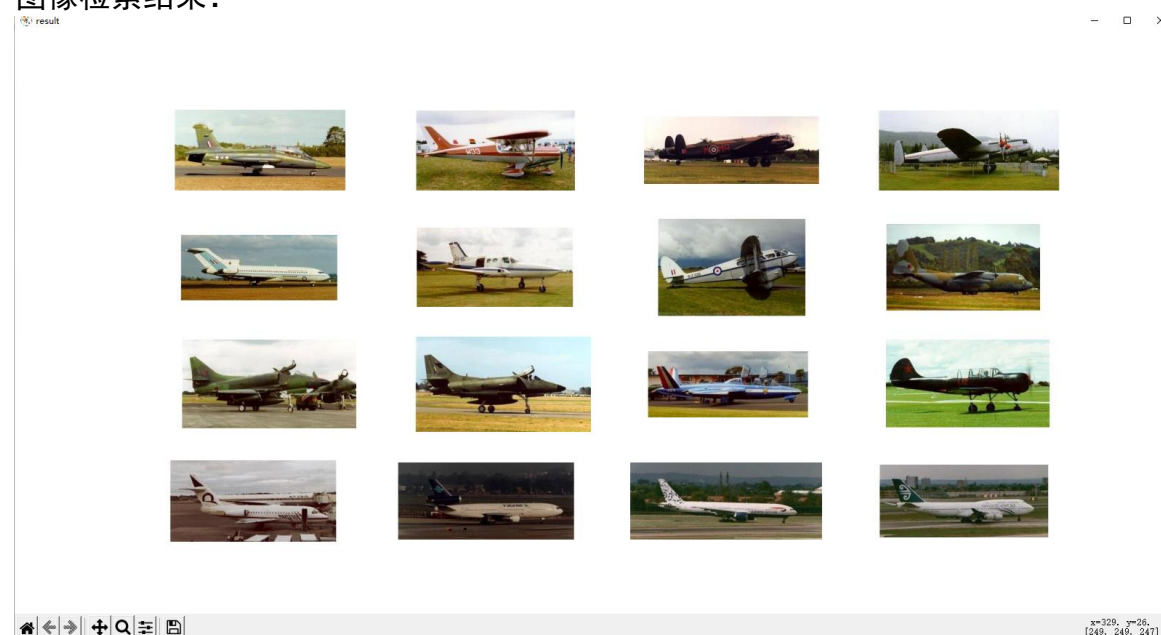


## 6.3 实验结果及分析

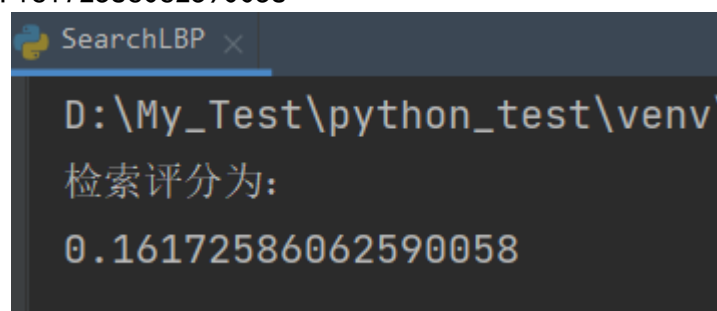
测试中以飞机image\_0001为例



图像检索结果:



检索评分为: 0.16172586062590058



分析: LBP是对图像的局部特征进行提取并进行检索的, 这样必然会损失一部分有用的信息, 主观上来看确实将若干飞机检索了出来, 但客观的评分上却比较低, 反映出局部特征检索的缺陷。

## 第七章 基于VGG16的图像检索

### 7.1 核心算法描述

本文使用了经典的vgg16网络结构。在python中直接调用高层神经网络API—Keras, 提取vgg16最后一层卷积特征作为图像的特征。算法会对图像数据库中的图像进行批量特征抽取, 当需要图像检索时抽取目标图像的特征, 然后和批量的图像

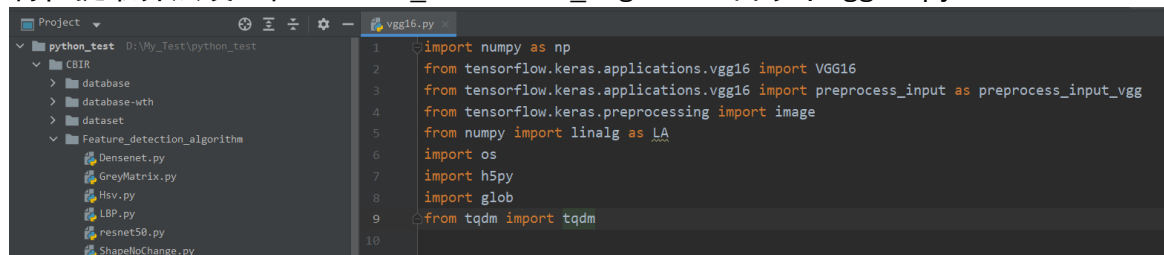


数据库特征进行相似度计算，查找索引并返回结果。

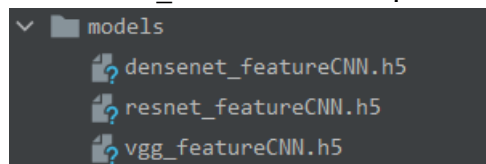
```
# 提取vgg16最后一层卷积特征
def vgg_extract_feat(self, img_path):
    img = image.load_img(img_path, target_size=(self.input_shape[0], self.input_shape[1]))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input_vgg(img)
    feat = self.model_vgg.predict(img)
    norm_feat = feat[0] / LA.norm(feat[0])
    return norm_feat
```

## 7.2 系统架构说明

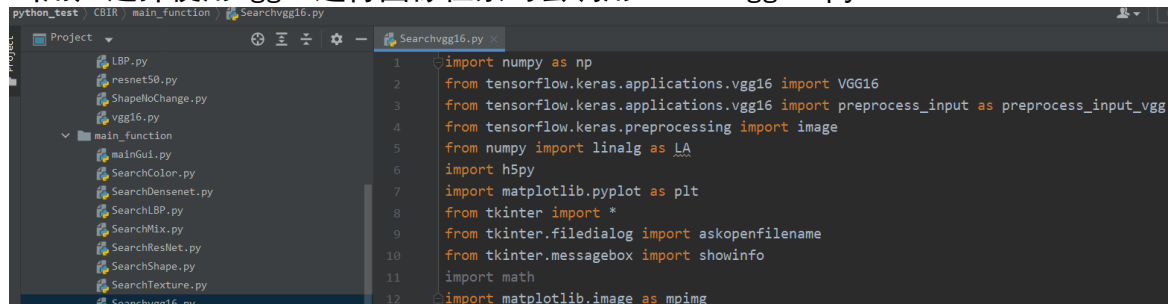
特征提取算法设置在Feature\_detection\_algorithm目录中vgg16.py



特征提取后保存在/models/resnet\_featureCNN.h5中



当用户选择使用vgg16进行图像检索时会调用Searchvgg16.py



在Searchvgg16.py中会先引导用户载入图像（过程与前五章一致，不再赘述）然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

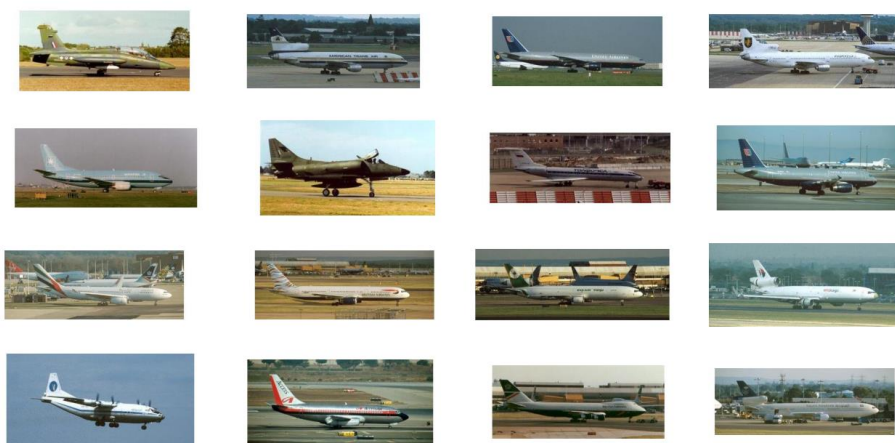
## 7.3 实验结果及分析

测试中以飞机image\_0001为例

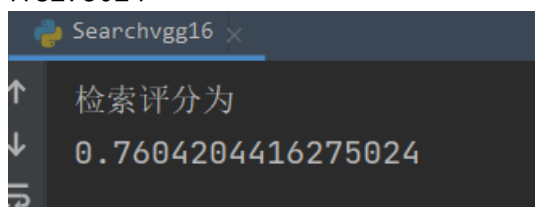


检索结果为:

result



检索评分为0.7604204416275024



分析: vgg16神经网络成功的检索出了若干飞机的图像, 并且根据TOPSIS评分也比较高, 可以认为vgg16拥有比较好的特征提取以及图像检索的能力

## 第八章 基于Resnet50的图像检索

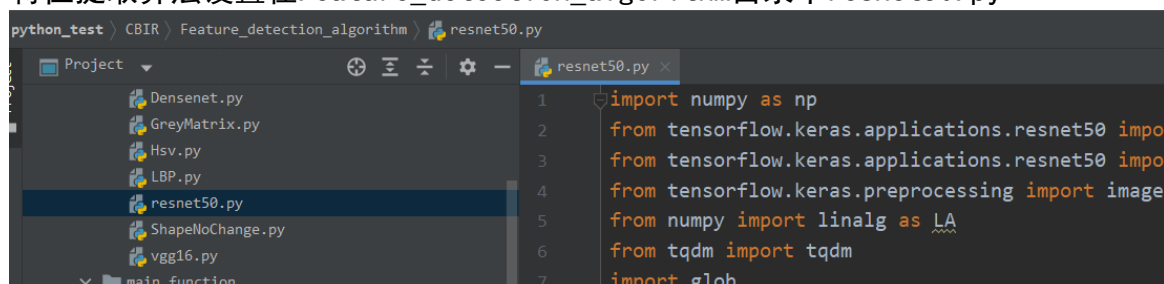
### 8.1 核心算法描述

本文使用了经典的Resnet50网络结构。在python中直接调用高层神经网络API—Keras, 提取Resnet50最后一层卷积特征作为图像的特征。算法会对图像数据库中的图像进行批量特征抽取, 当需要图像检索时抽取目标图像的特征, 然后和批量的图像数据库特征进行相似度计算, 查找索引并返回结果。

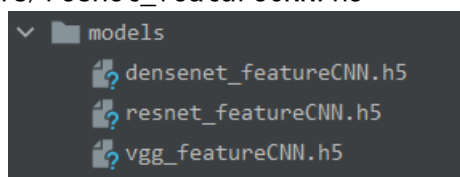
```
#提取resnet50最后一层卷积特征
def resnet_extract_feat(self, img_path):
    img = image.load_img(img_path, target_size=(self.input_shape[0], self.input_shape[1]))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input_resnet(img)
    feat = self.model_resnet.predict(img)
    norm_feat = feat[0]/LA.norm(feat[0])
    return norm_feat
```

## 8.2 系统架构说明

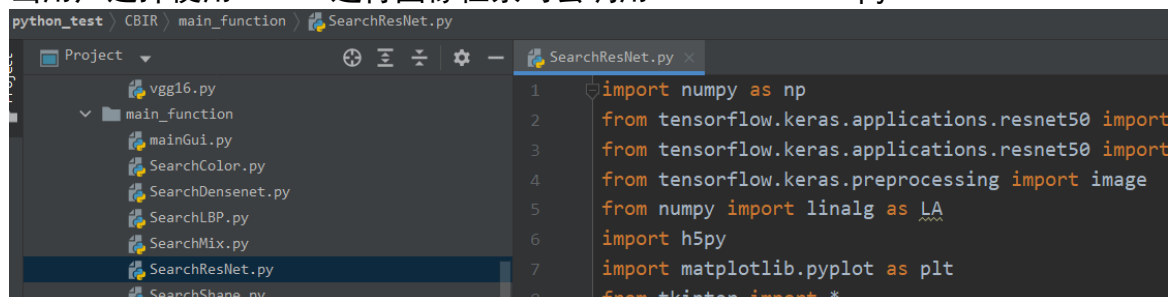
特征提取算法设置在Feature\_detection\_algorithm目录中resnet50.py



特征提取后保存在/models/resnet\_featureCNN.h5



当用户选择使用resnet进行图像检索时会调用SearchResNet.py



在SearchResNet.py中会先引导用户载入图像（过程与前五章一致，不再赘述）然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

## 8.3 实验结果及分析

测试中以飞机image\_0001为例





检索结果为:

result



检索得分为: 0.7700089477002621



分析: resnet50神经网络成功的检索出了若干飞机的图像, 并且根据TOPSIS评分也比较高, 可以认为resnet50拥有比较好的特征提取以及图像检索的能力

## 第九章 基于DenseNet121的图像检索

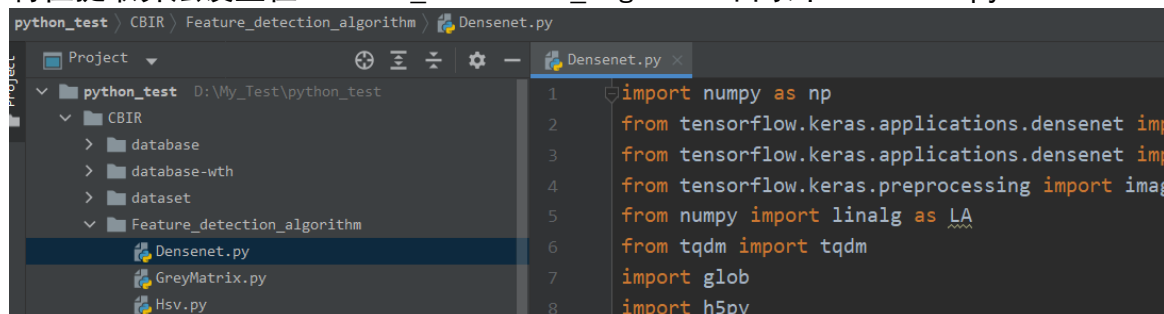
### 9.1 核心算法描述

本文使用了经典的Densenet121网络结构。在python中直接调用高层神经网络API—Keras, 提取Densenet121最后一层卷积特征作为图像的特征。算法会对图像数据库中的图像进行批量特征抽取, 当需要图像检索时抽取目标图像的特征, 然后和批量的图像数据库特征进行相似度计算, 查找索引并返回结果。

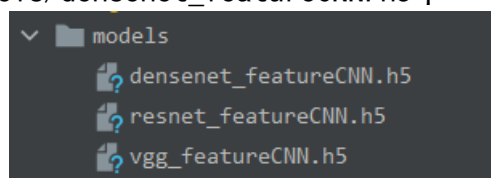
```
#提取densenet121最后一层卷积特征
def densenet_extract_feat(self, img_path):
    img = image.load_img(img_path, target_size=(self.input_shape[0], self.input_shape[1]))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input_densenet(img)
    feat = self.model_densenet.predict(img)
    norm_feat = feat[0]/LA.norm(feat[0])
    return norm_feat
```

## 9.2 系统架构说明

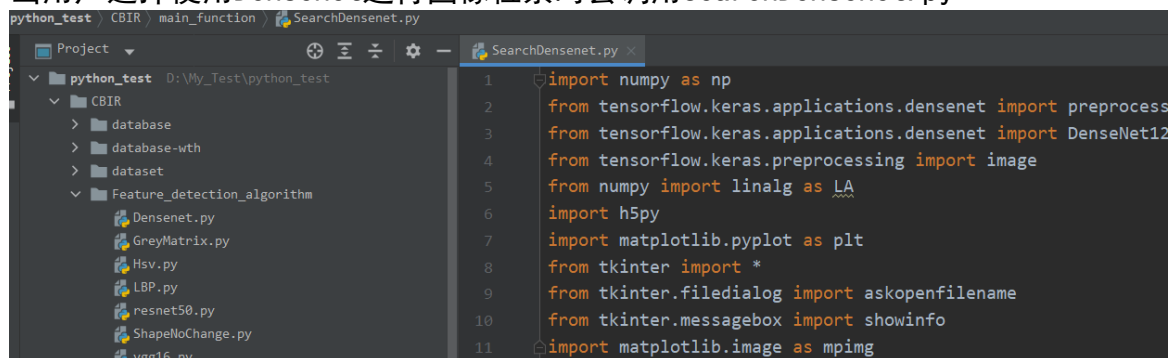
特征提取算法设置在Feature\_detection\_algorithm目录中Densenet.py



特征提取后保存在/models/densenet\_featureCNN.h5中



当用户选择使用Densenet进行图像检索时会调用SearchDensenet.py



在SearchDensenet.py中会先引导用户载入图像（过程与前五章一致，不再赘述）然后系统对输入的图片进行特征提取，接着与图像特征库中的所有图像特征进行计算，求出所有与目标图片相似的图像索引，返回给用户若干检索出来的图像，最后使用TOPSIS评价模型对此次检索进行评分。

## 9.3 实验结果及分析

测试中以飞机image\_0001为例



检索结果为:

result



检索得分为: 0.8218799717724323

```
SearchDensenet x
image names: b'airplane
image names: b'airplane
top 16 images in order
检索得分为
0.8218799717724323
```

分析: Densenet121神经网络成功的检索出了若干飞机的图像, 并且根据TOPSIS评分也比较高, 可以认为Densenet121拥有比较好的特征提取以及图像检索的能力

## 第十章 TOPSIS评价模型

### 10.1 TOPSIS评价模型概念

TOPSIS(Technique for Order Preference by Similarity to an Ideal Solution)法, 即逼近于理想解的排序技术, 又称为优劣解距离法、理想解法、理想点法, 由C. L. Hwang和K. Yoon于1981年首次提出。它是多目标决策分析中一种常用的有效方法。

TOPSIS法根据有限个评价对象与理想化目标的接近程度进行排序的方法, 是在现有的对象中进行相对优劣的评价。

## 10.2 TOPSIS方法的基本思路

### ①定义决策问题的理想化目标/理想解(Ideal Solution)

正理想解 (positive ideal solution):又称为最优解,是一个设想的最优的解(方案),其各个属性值(指标)都达到各备选方案中的最好的值;

负理想解(negative ideal solution)又称为最劣解,是一个设想的最劣的解(方案),它的各个属性值(指标)都达到各备选方案中的最坏的值。

### ②检测各个方案与理想解的相对接近度;

③通过把各备选方案与正理想解和负理想解做比较,找到那个距理想解的距离最近、而距负理想解的距离最远的方案。

## 10.3 TOPSIS法的优点

1、TOPSIS法对原始数据的信息利用最为充分,其结果能精确的反映各评价方案之间的差距。

2、对数据分布及样本含量,指标多少没有严格的限制,数据计算亦简单易行。

3、不仅适合小样本资料,也适用于多评价对象、多指标的大样本资料。

4、可得出良好的可比性评价排序结果。

## 10.4 TOPSIS算法步骤

1.构造决策矩阵 $v_{ij}$ ,  $i$ 为各方案,  $j$ 为各指标

2.对决策矩阵进行规范化处理

3.构建权重 $w_j$ , 可通过 熵权法 $Q$ 、FAHP、相关性等方法确定权重

4.计算加权决策矩阵

$$r_{ij} = w_j v_{ij}$$

5.计算正负理想解

$$S_j^+ = \begin{cases} \max_{1 \leq i \leq n} \{r_{ij}\} & j = 1, \dots, m; \text{越大越优型指标} \\ \min_{1 \leq i \leq n} \{r_{ij}\} & j = 1, \dots, m; \text{越小越优型指标} \end{cases}$$

$$S_j^- = \begin{cases} \min_{1 \leq i \leq n} \{r_{ij}\} & j = 1, \dots, m; \text{越大越优型指标} \\ \max_{1 \leq i \leq n} \{r_{ij}\} & j = 1, \dots, m; \text{越小越优型指标} \end{cases}$$

## 6.计算各方案与正负理想解间的距离

$$Sd_i^+ = \sqrt{\sum_{j=1}^m (S_j^+ - r_{ij})^2} \quad i = 1, \dots, n$$

$$Sd_i^- = \sqrt{\sum_{j=1}^m (S_j^- - r_{ij})^2} \quad i = 1, \dots, n$$

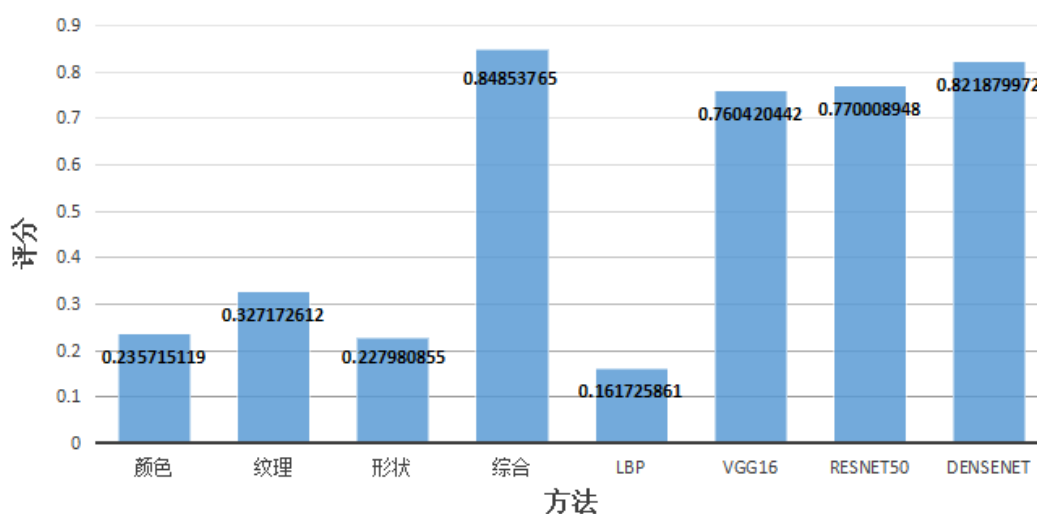
## 7.计算各方案与正理想解的相对贴近度

$$\eta_i = \frac{Sd_i^-}{Sd_i^+ + Sd_i^-}$$

# 第十一章 总结

TOPSIS综合评价模型对系统采用的八种图像检索方法进行评分，结果如下：

**TOPSIS评价得分**



根据TOPSIS模型可以得出从客观上综合信息的检索的效果是最好的，三种神经网络的检索效果率低于综合信息检索的评分，但是也远高于单个信息的图像检索。以飞机作为检索测试，从检索效果的主观上进行判断只有颜色存在干扰结果，其他检索方法都成功检索出来了若干飞机。但是客观上颜色检索方法并不是最低的，这也启示我不能单纯依靠主观或客观一方面的判断，而应该将二者结合起来，得出一个相对正确的结论。

本次项目我们完成了颜色、纹理、形状、综合信息图像检索的基础功能，并对纹理特征额外使用了LBP算法进行检索。此外我们还引入了三种经典的神经网络结构进行特征提取与图像检索。在完成了图像检索功能后我们采用了TOPSIS评价模

---

型对各种检索方法进行客观上的评分。根据图像检索的主观印象与TOPSIS模型的客观打分，完成对各种检索方法分析与比较。这次项目的经历教会了我许多图像特征提取的方法，并且还使用了三种经典的神经网络完成了特征提取与图像识别，最后使用的topsis评价模型也让我从客观的角度上看到了不同方法的区别，这启示我未来如果面对图像处理的时候，如果从传统的特征提取的方法入手，应当多考虑不同方面的信息，并将不同方面的信息综合起来，如果从神经网络入手，应当在经典的网络中做出适应性的改变，来不断提高图像处理的效果。