多媒体技术 课程实验报告

学号：2019217771 班级：物联网工程 19-1 班 姓名：梁家玮

实验二 傅立叶描述子

一、实验原理

首先准备 0～9 的数字图像作为训练集，使用 OpenCV 找到图像的轮廓并标注。OpenCV 在寻找图像轮廓时不仅会包含图像的内部轮廓，而且会包含外部轮廓。所以需要找到图像的外轮廓并进行去除。对于如“0”这样的数字，内轮廓可能不止有一个，此时则找出面积最大的部分，并记录轮廓的数组编号。在对 0～9 数字依此进行分析之后，准备根据轮廓计算相应的傅立叶描述子。

用数学公式（推导）计算出图像轮廓的实际傅立叶描述子，将傅立叶描述子归一化得到 15 个归一化描述子，每个图像对应一个。将进行翻转处理的图像再次进行傅立叶描述子计算，得到与处理前的图像描述子差距不大的结果，符合了傅立叶描述子的特点（旋转不变）。

使用采用了其它字体的数字图片作为测试集进行测试，同样计算出每个图像的归一化后的傅立叶描述子，将两者进行比对（这里只采用了简单的绝对值比对），差距越小则越接近，取最接近的图像为识别结果。经过测试后得到的识别准确率为 85%左右。

二、实验代码

```
from cv2 import GaussianBlur, Mat, contourArea,CV_8SC2, sqrt
from numpy import *
#  最外侧轮廓对应的数组标
def picGenerate(img_path):
    o_index = list()
    idex = 0
```

```
    # 提取图像轮廓
    img = cv.imread(img_path)
    gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    ret,binary = cv.threshold(gray,128,255,cv.THRESH_BINARY)
    # 找所有轮廓
    contours, hierarchy =
cv.findContours(binary,cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE)
    # 找外轮廓
    contours_outside, hierarchy_outside =
cv.findContours(binary,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIMPLE)
    re = contours
    for i in range(0,len(contours)):
        for k in range(0,len(contours_outside)):
            if len(contours_outside[k]) == len(contours[i]):
                o_index.append(i)
    # print(o_index)
    # 计算面积最大的部分
    size_count_arr = list()
    for k in range(0,len(contours)):
        if k not in o_index:
            # 非外轮廓在图上标出
            cv.drawContours(img,contours,k,(0,0,255),3)
            size_count_arr.append([k,contourArea(contours[k])])
    # print(size_count_arr)
    idx = 0
    max_size = size_count_arr[idx][1]
    for m in range(0,len(size_count_arr)):
        if(size_count_arr[m][1] > max_size):
            idx = m
            max_size = size_count_arr[m][1]
```

```
plt.imshow(img)
use_index = size_count_arr[idx][0]
# print(use_index)
pass
# 计算傅立叶描述子
# 轮廓的实际描述子
f = [0] * 9000
# 归一化的描述子，取了前 15 个
fd = [0]    * 16
s = len(contours[use_index])
# print(s)
src_re = list()
for u in range(0,s):
    sumx = 0.0
    sumy = 0.0
    for v in range(0,s):
        p = contours[use_index][v]
        x = p[0][0]
        y = p[0][1]
        sumx += x*np.cos(2*np.pi*u*v/s)+y*np.sin(2*np.pi*u*v/s)
        sumy += y*np.cos(2*np.pi*u*v/s)-x*np.sin(2*np.pi*u*v/s)
    r = list()
    r.append(sumx)
    r.append(sumy)
    src_re.append(r)
    f[u] = sqrt((sumx*sumx)+(sumy*sumy))
    f[0] = 0
fd[0] = 0
# 傅立叶描述子归一化
for k in range(2,17):
```

```
            f[k] = f[k] / f[1]
            fd[k-1] = f[k]
        return fd
re_1 = picGenerate("0.png")
print(re_1)
re_2 = picGenerate("0_invert.png")
print(re_2)
# 计算所有的数字（0~9）的傅立叶描述子
number_group = list()
for i in range(0,10):
        file_path = str(i)+".png"
        re = picGenerate(file_path)
        number_group.append(re)
print(number_group)
# 数字识别测试
def compareFunction(path):
        print("File : "+path)
        result = -1
        input_file_repo = picGenerate(path)
        qa = -1
        for k in range(0,10):
            # 当前比较数字：k
            current = number_group[k]
            v = 0
            for m in range(0,16):
                v += abs(current[m] - input_file_repo[m])
            print(str(k)+" : "+str(v))
            if(qa==-1 or qa>v):
                result = k
                qa = v
```

```
        print("Result: " + str(result))
        return result
# 识别准确率测试
import csv
total = 0
right = 0
right_list = list()
wrong_list = list()
csv_reader = csv.reader(open("testlist.csv"))
for line in csv_reader:
        total += 1
        x = compareFunction(line[0])
        y = int(line[1])
        if(y==x):
                # 比较正确
                right += 1
                right_list.append(line[0]+" | "+str(x)+" | "+str(y))
        else:
                wrong_list.append(line[0]+" | "+str(x)+" | "+str(y))
```

三、运行结果

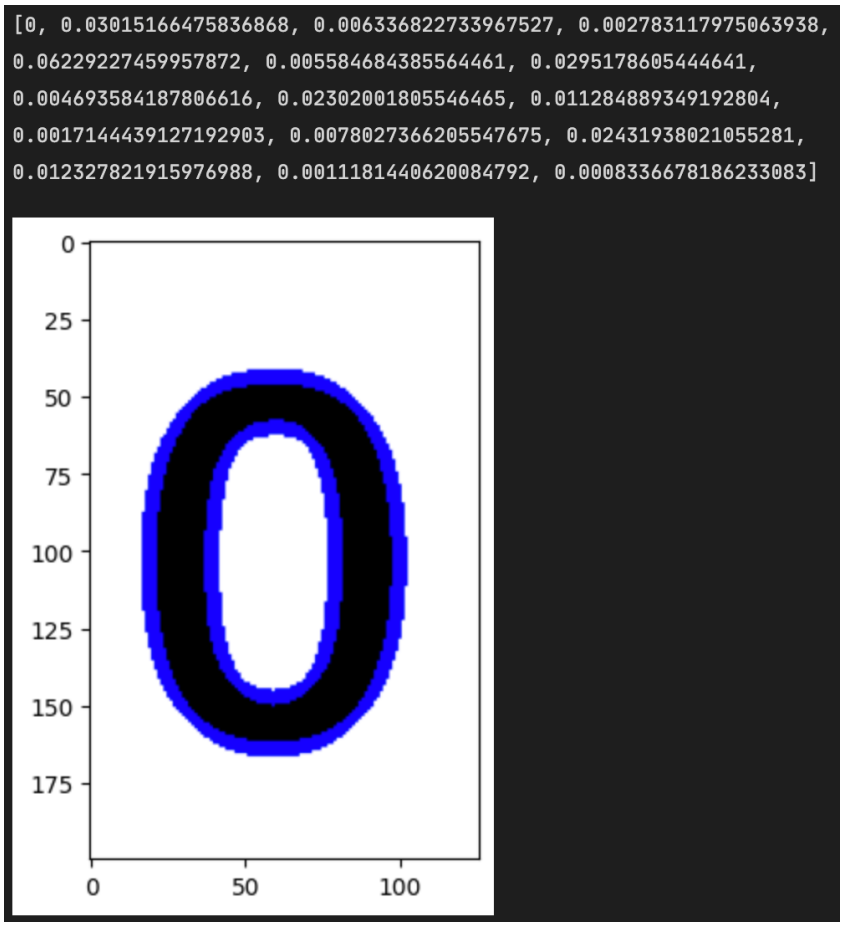

图 1 对图像进行轮廓标注，以及计算出的傅立叶描述子

[0, 0.030151664758368838, 0.006336822733967417, 0.0027831179750642267,
0.062292274599578666, 0.00558468438556395, 0.029517860544465244,
0.0046935841878062365, 0.023020018055465705, 0.011284889349191734,
0.001714443912718056, 0.007802736620554404, 0.02431938021055411,
0.012327821915976133, 0.0011181440620084198, 0.0008336678186236054]

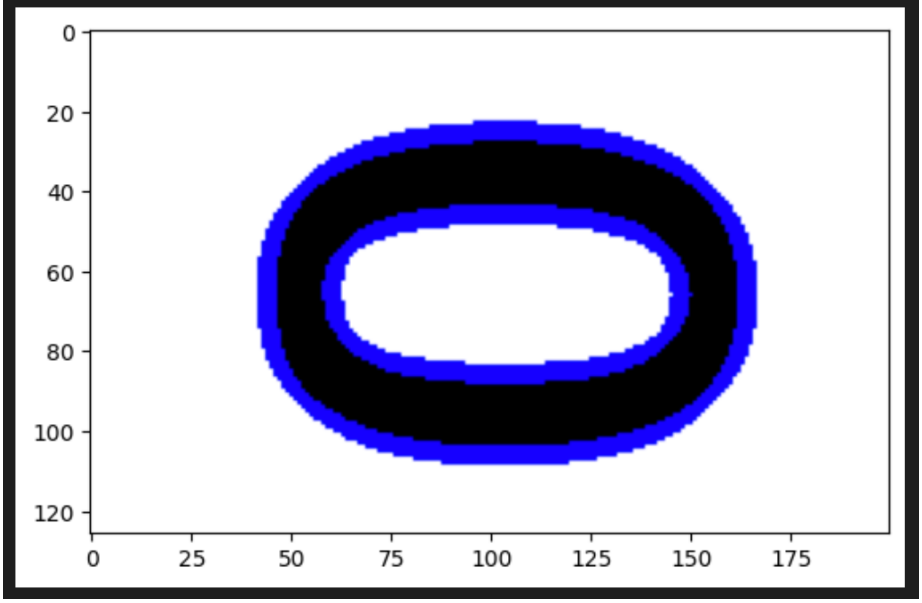

图 2 将图像旋转处理，傅立叶描述子未发生较大变化

```
print(number_group)
```

Python

[[0, 0.030151664758368868, 0.006336822733967527, 0.002783117975063938, 0.06229227459957872, 0.005584684385564461, 0.029517860544464l, 0.004693584187806616,
0.023020018055546465, 0.011284889349192804, 0.0017144439127192903, 0.0078027366205547675, 0.024319380210552B1, 0.012327821915976988, 0.0011181440620084792,
0.0008336678186233083], [0, 0.8338008746766528, 0.6937646641162881, 0.4675744460680104, 0.27682835712992554, 0.08971805689240366, 0.003390477956155111,
0.062715953020106145, 0.037724189580919S7, 0.038677646535195527, 0.11471203544244023, 0.11809364687647804, 0.133678178776718G, 0.08211625946940303, 0.0496210141883127,
0.012467388820178298], [0, 0.6815811095329191, 0.171796392264109S2, 0.10805113610412667, 0.0976192885674142, 0.073657630055264?1, 0.052255729042962225,
0.05880344149650761, 0.0828094467673137, 0.093993615076118, 0.06827726642885344, 0.0803225793184702A, 0.05381071531289039, 0.06928897283000188, 0.05987283918850487,
0.050171083970963B2], [0, 0.45121364340424835, 0.06858737895765134, 0.3557598001940345, 0.047237831572630?8, 0.00563037738116367?, 0.040571650002048154,
0.05046193577267405, 0.02325189861613007, 0.03821230679419703, 0.014925437561772874, 0.02360804480811462, 0.020242689693234G7, 0.03526742853681203,
0.018033339447312086, 0.015050876501487635], [0, 0.6151293692460144, 0.41523416517444?8, 0.2552972336317616, 0.12153021110964765, 0.02565012512528014,
0.050173748212438G5, 0.07566376918102505, 0.073066125254829O8, 0.05000001931560607, 0.022526877682376132, 0.032987139662478S5, 0.040588826187746625,
0.042028083258344l9, 0.027765674551393723, 0.018886747789592753], [0, 0.5727076611446745, 0.16344410673293G3, 0.046464401715178454, 0.1722425889611345,
0.144774768534243, 0.12728298476610185, 0.1447710785787092, 0.09836917666597276, 0.08334996891651682, 0.090047055151094?, 0.06551033946080177, 0.022347219943496855,
0.06432435805048875, 0.03063650523443190?, 0.04466466150101789], [0, 0.7382355927033235, 0.03636375764513382, 0.05346269296797364S, 0.0637785888375887,
0.05083918943184549, 0.019234737420885363, 0.041357334222809?1, 0.026702914539370?, 0.011766043093007597, 0.020999103995385?9, 0.026567352568878117,
0.020712104191136567, 0.013978221437725294, 0.020109703534850355, 0.014365181168674749], [0, 0.058295584562464246, 0.2511679104796679, 0.06494289937793918,
0.14866626611130672, 0.07041593260801825, 0.12263939932964761, 0.06834295874680375, 0.107628994248973S3, 0.067423416972821l4, 0.09721669161676987, 0.0662311932047104A,
0.08739588331573626, 0.06419331945977548, 0.07967411624564069, 0.06853680473008822], [0, 0.03296039642842038, 0.22889490537421?8, 0.0027032095246785504,
0.05155940308265051S, 0.010346728288070498, 0.04022891170372?2, 0.010530183554628784, 0.007109977105665261, 0.00557831650223221S, 0.016255200235461197,
0.0014557515089454324, 0.004241790760813?3, 0.00254165553649536B, 0.024856021536007612, 0.00333374764139427835], [0, 0.715963376209182?, 0.05622611058792871S,
0.05420948506206606, 0.061564151845793356, 0.061075179163069905, 0.014439629374117462, 0.040752187825162885, 0.025361730083852l6, 0.007948215231715988,
0.017560278462565394, 0.018676592763813674, 0.007969676656835?5, 0.006503839226252383, 0.013982305790489?2, 0.016213992010594435]]

图 3 计算每张数字图片的傅立叶描述子

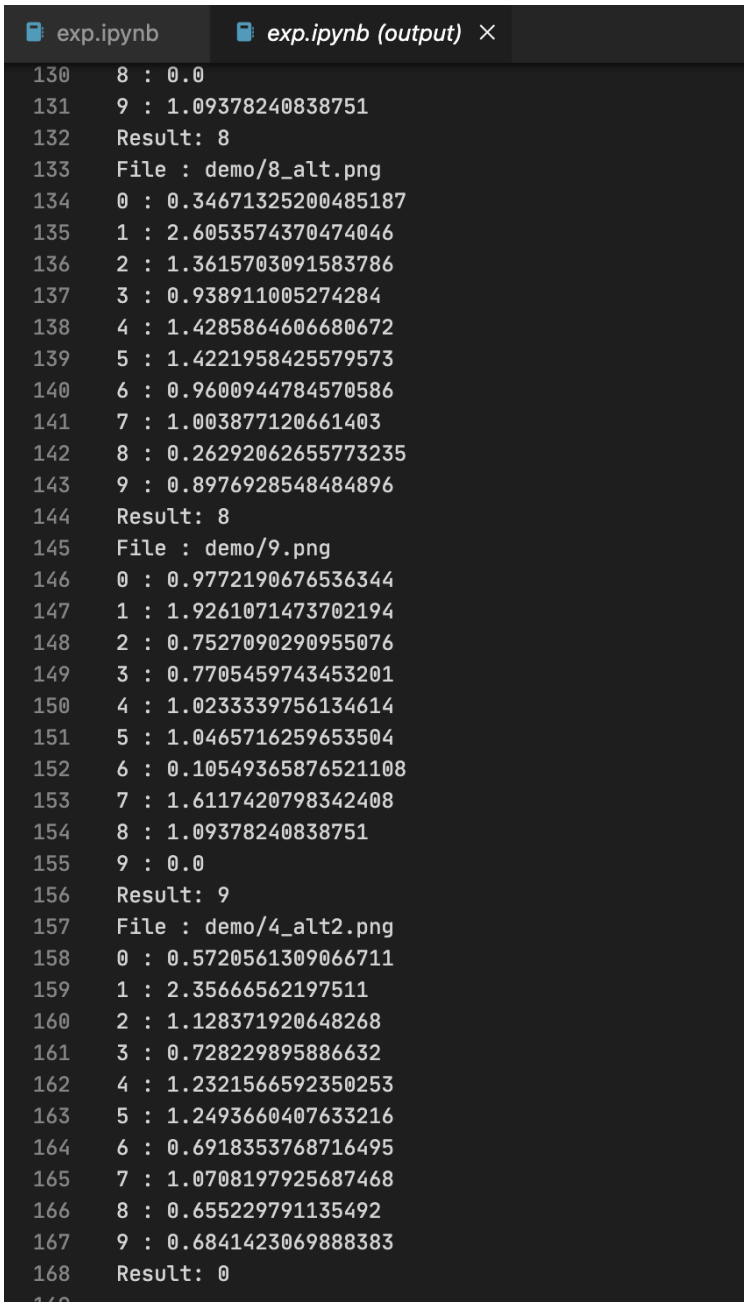

图 4 测试集+训练集图片的对比结果

```
Accuracy : 0.8571428571428571
======== Right ========
FilePath | Recognize | Answer
demo/0.png | 0 | 0
demo/0_invert.png | 0 | 0
demo/1.png | 1 | 1
demo/2.png | 2 | 2
demo/3.png | 3 | 3
demo/4_alt.png | 4 | 4
demo/5.png | 5 | 5
demo/6.png | 6 | 6
demo/7.png | 7 | 7
demo/8.png | 8 | 8
demo/8_alt.png | 8 | 8
demo/9.png | 9 | 9
======== Wrong ========
FilePath | Recognize | Answer
demo/6_alt.png | 8 | 6
demo/4_alt2.png | 0 | 4
```

图 5 识别的准确率与识别状况