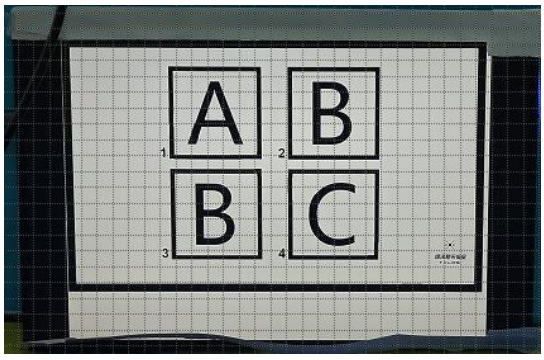
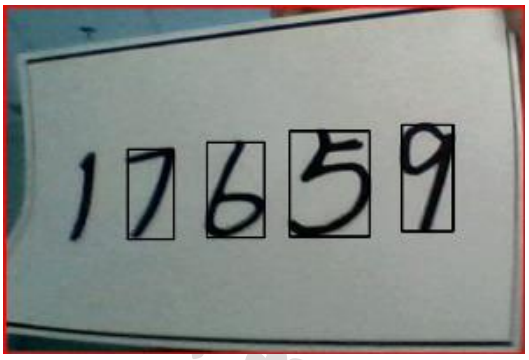
KNN数字字母分类识别

1.设计流程

1.1 应用场景





使用电脑自带的摄像头，对四个窗口的字母进行实时识别，其中每个窗口都是黑色小正方形，被一个大的外围黑色矩形框所包围。对数字进行识别，其中数字被一个大的外围黑色矩形框所包围。

1.2 程序设计思路

字母识别：首先对大的黑色矩形边框进行识别，并裁切图片保存，再对保存的图片进行尺寸处理（转换为等整数比例的尺寸），再对每个窗口字母进行识别。方法一：在矩形黑框内再对每个小正方形黑框进行识别，识别后作裁切并保存图片用于识别；优点是裁切后的图片完整方正，识别到正方形黑框后可直接使用于字母识别；缺点是同时对多个小尺寸的正方形黑框判定难度较大且不稳定，实时摄像头判定多个黑框实时性不高且容易误判漏判。方法二：在对大的黑框进行判定且进行尺寸处理后，使用像素值坐标定位的方法进行四个窗口的切割并保存为图片进行识别；优点是只对一个大的矩形黑框进行检测实时性好，基于坐标定位的方法编程较为简单；缺点是定位的方法受图片摆放和黑框位置判定的影响大，且定位需针对实际应用场景作对应调整。综合考虑，选用方法二进行窗口切割，最后对每张图片进行分类识别。

数字识别：沿用字母识别中识别大黑色矩形框并切割保存图片的方法，后续由左至右进行数字轮廓检测，分类识别数字。

1.3 环境配置（自行官网或搜索配置下载）

pycharm2021 社区版
Python3.6;
Opencv4.5.3
Numpy1.19.5
scikit-learn 0.19.2

2. 编程设计

2.1 字母分类模型训练和保存

```python
import os
import cv2
import numpy as np
import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# 读取数据集路径
dataset_path = {'A': r'D:\Ubuntu\ABC_DATA\ABC\Sample011',
                'B': r'D:\Ubuntu\ABC_DATA\ABC\Sample012',
                'C': r'D:\Ubuntu\ABC_DATA\ABC\Sample013'}

# 读取数据集图片
X, y = [], []                                          #创建数据集列表
for label, path in dataset_path.items():
    for f in os.listdir(path):                         #遍历数据集路径
        if f.endswith('.png'):                         #遍历所有png图片
            img = cv2.imread(os.path.join(path, f))    #读取每一张图片
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)   #图片转为灰度图
            resized = cv2.resize(gray, (20, 20))       #图片尺寸转为20×20
            X.append(resized.reshape(-1))              #转化为一维向量并保存到X列表
            y.append(label)                            #每张图片的标签保存到y列表

# 将数据集分为训练集和测试集
# test_size表示测试数据集的比例；random_state为偏差拟合因子
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 训练 KNN 模型
k = 5                                  # knn算法邻居因子，表示预测时与最近的多少个数据进行分类判定
knn = KNeighborsClassifier(n_neighbors=k)  # 初始化knn分类器
knn.fit(X_train, y_train)              # 训练knn模型并生成分类器

# 在测试集上进行模型评估
accuracy = knn.score(X_test, y_test)
print('Accuracy: {:.2f}%'.format(accuracy * 100))

# 运行程序模型将被生成保存，不保存模型无法被调用
with open('knn_model.pkl', 'wb') as f:
    pickle.dump(knn, f, protocol=2)
```

图 3　字母分类模型训练

2.2 数字分类模型训练和保存

```python
import os
import cv2
import numpy as np
import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

dataset_path = {'0': r'D:\Ubuntu\ABC_DATA\0_9\data_0',
                '1': r'D:\Ubuntu\ABC_DATA\0_9\data_1',
                '2': r'D:\Ubuntu\ABC_DATA\0_9\data_2',
                '3': r'D:\Ubuntu\ABC_DATA\0_9\data_3',
                '4': r'D:\Ubuntu\ABC_DATA\0_9\data_4',
                '5': r'D:\Ubuntu\ABC_DATA\0_9\data_5',
                '6': r'D:\Ubuntu\ABC_DATA\0_9\data_6',
                '7': r'D:\Ubuntu\ABC_DATA\0_9\data_7',
                '8': r'D:\Ubuntu\ABC_DATA\0_9\data_8',
                '9': r'D:\Ubuntu\ABC_DATA\0_9\data_9'}

X, y = [], []
for label, path in dataset_path.items():
    for f in os.listdir(path):
        if f.endswith('.png'):
            img = cv2.imread(os.path.join(path, f))
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            resized = cv2.resize(gray, (20, 20))
            X.append(resized.reshape(-1))
            y.append(label)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

accuracy = knn.score(X_test, y_test)
print('Accuracy: {:.2f}%'.format(accuracy * 100))

with open('knn_data.pkl', 'wb') as f:
    pickle.dump(knn, f, protocol=2)
```

图 4　数字分类模型训练

2.3 数字识别代码

代码见 box_forDATA.py

参数修改：

①初始化置信度为 0.70；

把单引号里的路径修改为‘knn_data.pkl’所在的路径，注意‘\’；

```
################置信度设置和调用模型#####################
confidence_threshold = 0.70
with open(r'D:\Ubuntu\ABC_DATA\knn_for_data_char\knn_data.pkl', 'rb') as f:
    knn = pickle.load(f)
```

②判断每个数字后的图片裁切尺寸为 100×100；

```
################数字定位函数#########################
def crop_image(img, x_offset, y_offset):
    half_width, half_height = 100, 100
    x1, y1 = x_offset - half_width, y_offset - half_height
```

③读取摄像头初始化为 0 代表读取电脑自带摄像头，可通过读取 IP 地址的方式读取手机摄像头或其它设备摄像头，感兴趣请自行搜索；

```
#############读取电脑自带摄像头#######
cap = cv2.VideoCapture(0)
```

④初始化二值化处理的分割值为 127，表示色彩值大于 127 的变为 255，小于 127 的变为 0；其中 1 代表白色，0 代表黑色；

```
while True:
    ret, frame = cap.read()
    img = frame.copy()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

⑤对矩形框的最小面积进行限定，如果矩形框的像素面积小于 10000 则无法检测到；

```
    max_rect = None
    max_area = 10000
    for cnt in contours:
```

⑥我将矩形黑框大小转换为了‘600×400’，可根据矩形框实际比例进行调整；

把‘box_data.jpg’更改为自己想要的图片命名格式，这是裁切到的大矩形框的图片；

```
        cv2.rectangle(img, (max_rect[0], max_rect[1]), (max_rect[0]+max_rect[2], max_rec
        cropped = img[y_min:y_max, x_min:x_max]
        dim = (600, 400)
        resized = cv2.resize(cropped, dim, interpolation=cv2.INTER_AREA)
        cv2.imwrite('box_data.jpg', resized)
        img = cv2.imread('box_data.jpg')
```

⑦把“双引号内的路径更改为自己想保存裁切后的各个数字小图片的路径”，digit_{index}.jpg 采用了索引的命名方法，即从 0 到“index”依次保存图片，前缀名为 digit_’，后缀为序号，格式为 jpg；

```
roi = 255 - roi
roi_resized = cv2.resize(roi, (40, 40), interpolation=cv2.INTER_AREA)
# cv2.imwrite('digit_{}.jpg'.format(index), roi_resized)
cv2.imwrite(f'C:\\Users\\aifei\\Desktop\\Robot_sofa_file\\knn_for_data_char\\digit_data\\digit_{index}.jpg', roi_resized)
index += 1
```

⑧路径和上面小图片保存的路径一致；

```
# 获取所有符合条件的文件路径
file_paths = glob.glob(r'C:\Users\aifei\Desktop\Robot_sofa_file\knn_for_data_char\digit_data\digit_*.jpg')
predicted_str = ''
# 遍历文件路径列表，依次处理每张图片
```

⑨启动‘os.remove’可删除上面定义路径下的所有图片（包括其它文件）。

```
    predicted_str += predicted_letter_str
    # os.remove(file_path)

print(f"预测结果为：{predicted_str}")
```