# NumDect_ProcessRealImage1

December 28, 2022

## 1 Handwritten Number Recognization

### 1.1 Group Information

Team Name: Participants:

### 1.2 0. Project Information

We have already creat the training dataset in *CreateTrainSet* . In this module, we will used the training dataset to recognize the handwritten digits.
We are going to complete the following steps: * Import the image with handwritten digits or take a photo of the handwritten digits to be detected. * Preprocess the image, including but not limited to converting the picture into a grayscale image, and then into a binary image; segmenting the picture to extract a single digit. * Recognize the handwritten digits based on kNN algorithm with the functions supplied by OpenCV. * Display the detected digits with digital tube.

### 1.3 1. Initialize the Environment

As usual, let's first initialize the environment. This step includes importing modules, geting project and file paths.

[1]:
```
%load_ext autoreload
%autoreload 1
# number detected related
import cv2
import os
import numpy as np
import math
from lib import imshow
import random

# get the project path
PRJ_PATH = os.getcwd()
# OPENCV_data.npz
TRAIN_DATA_NAME = "OPENCV_data.npz"
```

We will complete some independent functions in *my_function*.
Here let's first import the function we are going to work on.

[2]:
```
%aimport my_function
from my_function import image_split_row, image_split_column, led_display,␣
 ↪take_photo
```

### 1.4 2. Import the Training Dataset

In this module let's first import the training dataset which we generate in *CreateTrainSet* . Then let's create a kNN object and train it.

[3]:
```
# load the knn training data
with np.load(PRJ_PATH + '/TrainingData/' + TRAIN_DATA_NAME) as data:
    train = data["train"]
    train_labels = data["train_labels"]
train = train.astype(np.float32)
train_labels = train_labels.astype(np.float32)

# create KNN obj
knn = cv2.ml.KNearest_create()
knn.train(train,cv2.ml.ROW_SAMPLE,train_labels)
```

[3]:
```
True
```

### 1.5 3. Image Preprocessing

Before we start digits recognition, we need to do some preprocessing on the image. The preprocess includes the change of the image color gamut and the space domain. Let's start!

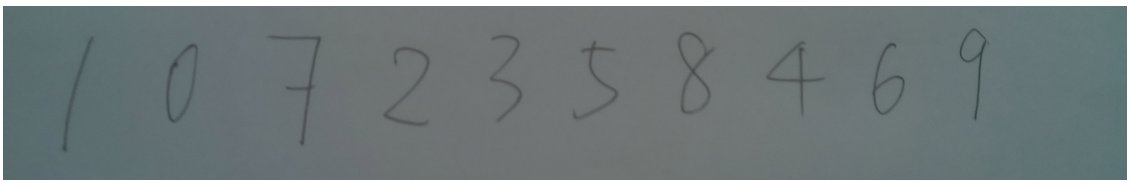
#### 1.5.1 Import Pending Pictures

- In first stage, we will use existing pictures with handwritten digits. Import the image with `dst = cv2.imread(filename)` and show the image with `imshow(src)`.

- In late stage, we will use the camera to take pictures. In this stage, you first need to build the camera control circuit on the breadboard. Then finish the function of taking pictures in `take_photo()` of *my_function*.

[4]:
```
image = take_photo()
img = cv2.imread(image)
imshow(img)
```

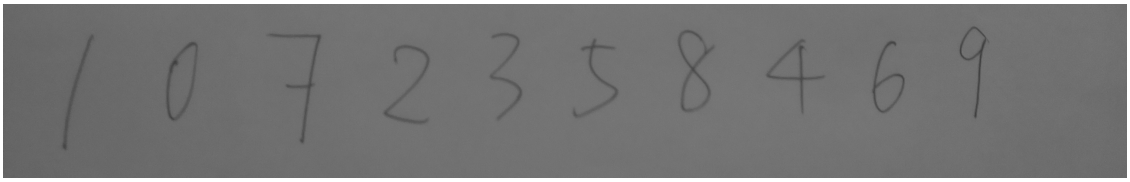
```
Camera activated
Folder already existed.
PATH: ./UserData/Pictures/
```



### 1.5.2 Get the Grayscale Image

After we get the image, we should first convert the color image into a grayscale image. Do it with `dst = cv2.cvtColor(src,code)`.

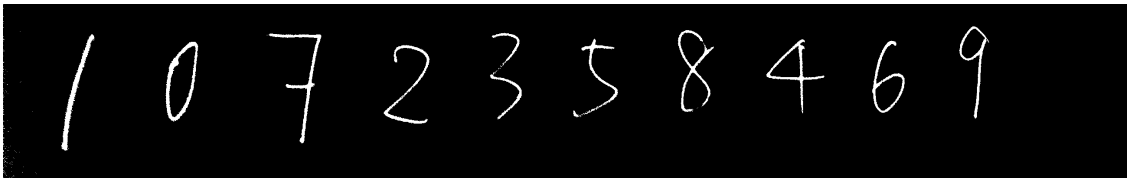
[5]:
```
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imshow(imgGray)
```



### 1.5.3 Get the Binary Image

After getting the grayscale image, we need to further convert the grayscale image into a binary image. Do it with `dst=cv2.threshold(src, thresh, maxval, type)`. **Note**: try to adjust parameter `thresh` to get a binary image without extra points or noise.

[11]:
```
_threshold, imgBin = cv2.threshold(imgGray,98,255,cv2.THRESH_BINARY_INV)
imshow(imgBin)
```



### 1.5.4 Split the Image

When we start to recognize handwritten digits, we should find out the corresponding digits one by one. So we need to segment the binary image to obtain a single digit.

Remeber how we split the image in *CreatTrainSet*? Since the numbers in *digits.png* have the same size (20x20) and are arranged neatly, we can simply divide *digits.png* according to the size of the numbers. But this time, the numbers to be detected are not restricted to strict positions, and their size may also be different, which means we can no longer use `np.hsplit()` and `np.vsplit()` for segmentation.

However in the previous steps, we have already get the binary image which only contains 0 and 255. In our project, 0 represents black, which is the background of the image. And 255 represents

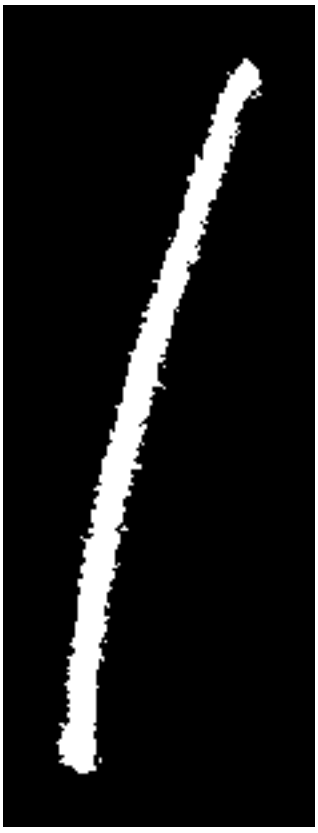
white, which constitutes the number on the picture. Therefor, we can determine the boundary of the handwritten number through the transformation of 0 and 255.

Head to my_function.py to complete function `image_split_row` and `image_split_column`.

- In the first stage, we will use the picture with only one line of numbers. Just run the following cell to view the split image.
- In the later stage, we will use the picture with multiple lines of numbers. Adjust the code in the cell below, and then run it to see the splitted image.

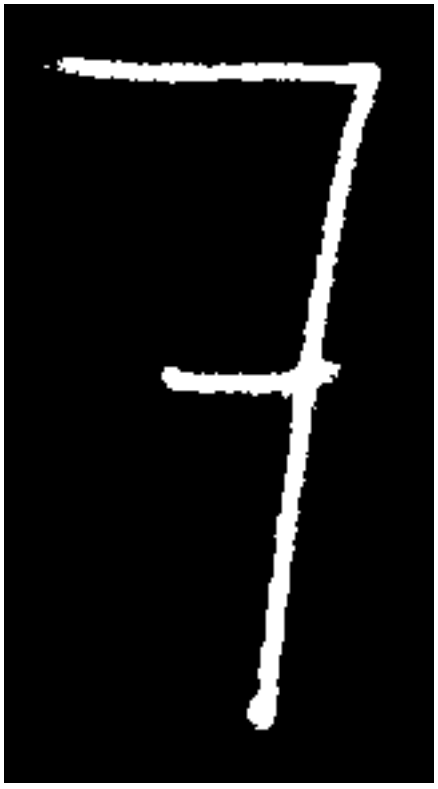
[12]:
```
imgCol = image_split_column(imgBin)
imgMonos = []
for col in range(0,len(imgCol)):
    imgMono = image_split_row(imgCol[col])
    print(f"{(col)}:")
    imshow(imgMono[0])
    imgMonos.append(imgMono[0])
```

```
0:
```



```
1:
```

2:



3:

4:

5:

6:

7:

8:

9:

10:

Here comes another problem: these images have different size. We need to resize them to a uniform shape. Also we should reshape the image to make it consistent with the kNN lib. Fill in the two variables

[8]:
```
resizeSize = (20,20)
reShapeSize = (1,400)
```

### 1.6 4. Recognize the Handwritten Number

Let's complete the resize and reshape operations of the image, and do the recognization. - In the first stage, we only have one line of numbers. Run the following cell to get the recognized numbers. - In the later stage, we will deal with multipul lines of numbers. Adjust the cell below to display the multiple rows of numbers.

[13]:
```
# resize and reshape the image with single number
# then recognize the number with knn.findNearest(imgReshape,k=?)
numberList = []
for i in range(0,len(imgMonos)):
    imgResize = cv2.resize(imgMonos[i], resizeSize)
    imshow(imgResize)
    imgReshape = imgResize.reshape(reShapeSize).astype(np.float32)
    _,result,_,_ = knn.findNearest(imgReshape,k=8)
    numberList.append(int(result))
print("The "+str(1)+"th row has:" + str(numberList))
```

```
The 1th row has:[6, 0, 4, 1, 2, 4, 5, 5, 4, 5, 7]
```

### 1.7 5. Display the Number by Digital Tube

Next we will use a digital tube to display the numbers we finally recognize.

We will use a common anode digital tube to display the result. Please complete the following operations: 1. Use breadboard, DuPont cables and GPIO pins of Raspberry Pi to complete the circuit for digital tube. 2. Complete the relevant code for digital tube in function `led_display` of *my_function.py*.

About the cell below: - In the first stage with only one line of numbers,run the cell below to display the result. - In the later stage with multiple lines of numbers, adjust the code in the cell to display the total results.

[10]:
```
# do the image splite and reshape
# then recognize the number with knn.findNearest(imgReshape,k=?)

led_display(numberList)
```

## 2 Congratulate!

### 2.1 You have completed all the tasks.