**Assignment 1. On-Line Signature Data Handling**

**student information**

Student ID: m5281054

Name: WANG Binghao

**Required Libraries:**

import tkinter as tk  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import numpy as np  
from matplotlib.ticker import MultipleLocator

**Run the Program:**

Python: Python3.12

I have put the executable file into ITA09\_Assignment\_1\Otherfiles\code

This file also includes all data sets. Make sure you open this folder and install the required libraries. Click Run.

**About Code Explanation:**

The code primarily achieves the following key functionalities:

1: Read handwritten signature data from a file.

2: Use tkinter to create a main window and add a canvas to the window for drawing handwritten signatures.

3: Draw the animation effect of a handwritten signature and simulate the process of signing by connecting different coordinate points.

4: Extract features of handwritten signatures, including horizontal and vertical coordinates, pressure, direction, height and time.

5: Use matplotlib to create a new graphics window and draw five subgraphs of the handwritten signature features. Each subgraph represents the change of a feature over time.

6: Set the ordinate range of each subgraph to make the image clearer and easier to read.

7: Runs the main loop, displays the graphics window and main window, and waits for user interaction.

**Some functions defined**

**read\_signature\_data()**

This function is used to read the handwritten signature data file and parse it into data in list form. Each line in the file represents a data point, and each line contains 6 values: x coordinate, y coordinate, pressure, direction, height and time. If the file contains empty lines, the function will skip them. If the file contains the stroke delimiter -1, the function inserts a special value -999999 as the delimiter.

def read\_signature\_data(file\_path):

signature\_data = []

with open(file\_path, 'r') as file:

for line in file:

if line.strip(): # 跳过空行

data = line.strip().split()

if len(data) == 6: # 确保每行有6个值

signature\_data.append(list(map(int, data)))

elif len(data) == 1 and int(data[0]) == -1: # 笔划分隔符

signature\_data.append([-1, -999999, -999999, -999999, 999999, -999999])

return signature\_data

**draw\_signature()**

This function is used to draw the animation effect of a handwritten signature on the specified canvas. The function first sets the style for drawing the signature (line color and width), and then iterates through the signature data in a loop. If the stroke separator -1 is encountered, the record of the previous point will be cleared and the drawing delay will be increased to achieve the animation effect; otherwise, a straight line will be drawn based on the coordinate information of the current point and the previous point, and the delay time will be determined based on the time interval.

def draw\_signature(canvas, signature\_data, delay=100):

line\_color = "black"

line\_width = 3

prev\_x, prev\_y, prev\_time = None, None, None

total\_time = signature\_data[-1][-1]

for data in signature\_data:

if data[0] == -1:

prev\_x, prev\_y, prev\_time = None, None, None

canvas.after(delay)

canvas.update()

else:

x, y, pressure, direction, height, time = data

if prev\_x is not None and prev\_y is not None and prev\_time is not None:

canvas.create\_line(prev\_x, prev\_y, x, y, fill=line\_color, width=line\_width)

canvas.after(int((time - prev\_time) \* delay / total\_time))

canvas.update()

prev\_x, prev\_y, prev\_time = x, y, time

**extract\_features()**

This function takes a list containing signature data and extracts individual feature values from it, including x-coordinate, y-coordinate, pressure, direction, height, and time. It returns a list of these characteristic values.

def extract\_features(signatures):

x\_coordinates = []

y\_coordinates = []

pressures = []

directions = []

heights = []

times = []

for signature in signatures:

for data in signature:

if data[0] != -1:

x\_coordinates.append(data[0])

y\_coordinates.append(data[1])

pressures.append(data[2])

directions.append(data[3])

heights.append(data[4])

times.append(data[5])

return x\_coordinates, y\_coordinates, pressures, directions, heights, times

**plot\_data()**

This function is used to plot feature data. It accepts a matplotlib axis object ax, feature data data, time data times and figure title title. The function uses the plot method to plot feature data on the axis and set the title, abscissa and ordinate labels

def plot\_data(ax, data, times, title):

ax.plot(times, data)

ax.set\_title(title)

ax.set\_xlabel('Time')

ax.set\_ylabel('Value')

**some problems**

I encountered some problems while working on this project. I still failed after a lot of modifications and debugging. I think I still have to continue to learn python language and some logical thinking to solve it.

1. I set a breakpoint by stroke and then restarted drawing, but during the drawing process, the drawing was still continuous.
2. I want to find a way to put the data change graph in the same window as my graph, and draw it together based on the time data, but I can't do that no matter how I modify the code.