## Laboratory # 2

***Biometric data acquisition: Signature***

# 1 Introduction

The purpose of this laboratory exercise is to become acquainted with signature data acquisition and statistical analysis, prior to verification or recognition.

## 1.1 Digital Tablet

To collect the signatures in this lab we will use a Wacom's digitizing tablet (`http://www.wacom.com`). In a lab kit you will have one of the models *Wacom Intuos* (CTL-4100WL) or *Wacom Bamboo Fun* (CTE-450). The tablet is capable of capturing coordinate values sampled at 200 points per second during pen movement. Using time values, kinematic properties of writing can be calculated (such as speed of the pen). Tablet writing area is 2540 lines per inch, and there are 1024 levels of pressure data. It is a USB device; when plugged into a computer that shall automatically find the Wacom's driver. If it does not, download and install the 64-bit tablet's driver (pay attention to the device model you are using) from: `http://us.wacom.com/en/support/drivers`. We also included the correct drivers to the Lab 2 D2L folder.

In the Biometric Technologies Lab at UofC (`http://www.ucalgary.ca/btlab`), we have developed the software SigGet to capture, and save, data in `.csv` format. This file contains information of the sampled points in the signature such as: the coordinates, pressure and time (used to computer the velocity) of writing.

You will need this data to plot the pressure map of the signature, velocity map, 3D plot (with pressure as a third coordinate), and statistics derived from velocity of your real signature and the forged one, to be used in the next labs for verification of the signatures. If you cannot collect the data using the tablet, some pre-recorded data will be provided.

## 1.2 Software to capture signatures: SigGet

SigGet, shown in Fig. 1, is a software that converts data captured by the tablet into `.csv` file.

Download the SigGet `.zip` file from D2L, unpack and run the file `SigGet V3.exe`. It will open the SigGet window (accept the security message), and you can use the tablet pen to acquire your signature on the tablet, reflected on the screen. The SigGet interface (see Fig. 1) has two buttons:

- *Save*: will save the acquired signature in `.csv` (coordinates, pressure and time) or `.png` (image) formats. In this course we will work only on the `.csv` format.
- *Clear*: reset the workspace and start new signature.

When a `.csv` file is saved, the following variables are created (assume $n$ is the number of points captured):

**coord** - $n$-by-2 matrix of `float` numbers containing coordinates $X$ and $Y$ of points captured;

**time** - $n$-by-1 matrix (column array) of `int` numbers containing time stamps in milliseconds for each point with respect to the first data point (which is time zero);

**prs** - $n$-by-1 matrix (column array) of `float` numbers containing pressure value for each points (even though the tablet is able to recognize 1024 levels of pressure, the value is usually normalized to 256).

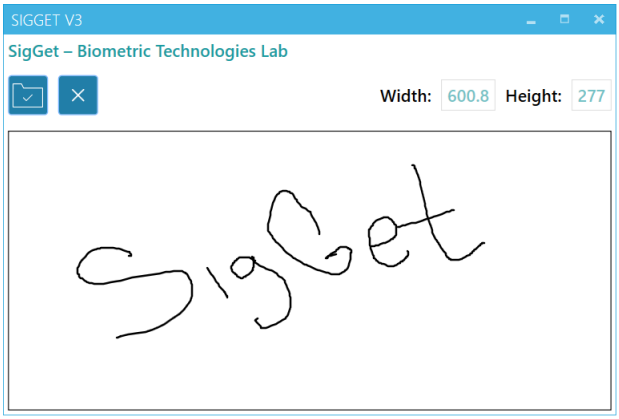



Figure 1: Software SigGet used to collect signatures.

# 2 The laboratory procedure

## 2.1 Acquisition of the sample data

The collected files shall be saved in the same folder as your Jupyter Notebook. Collect and save:

- at least 30 of your own (genuine) signatures,
- at least 30 of your "forged" (impostor) signatures (signed by someone else who try to replicate their shape),
- at least 30 some other signatures (also impostor, but some completely different signature or a written word).

The pre-recorded samples of signatures in `.csv` format collected by the TA using the tablet, can be found in the D2L under the name *prerecorded_signature_samples*.

## 2.2 Report

The Lab, performed in Jupyter Notebook / Python (file extension `.ipynb`) shall include the following graded components (10 marks total):

- Introduction (a paragraph about the purpose of the lab).
- (10 marks) Description of the result on each exercise with illustrations/graphs and analysis of the results (marks are distributed as shown in the Exercise section).
- Conclusion (a paragraph on what is the main take-out of the lab).

Your Notebook is your report that shall be saved (use menu "Download As") as `.ipynb`, and submitted through D2L dropbox for Lab 2, by the deadline (the following Thursday). Note that 10% of the lab grade will be deducted for each late submission day.

# 3 Exercise in Jupyter Notebook with Python

Below we provide a sample code which you can use when creating your Notebook for this lab.

- Import libraries to load the data (Pandas), perform numerical and scientific calculations (NumPy and SciPy) and plot the graphs (Matplotlib):

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits import mplot3d
import pandas as pd
```

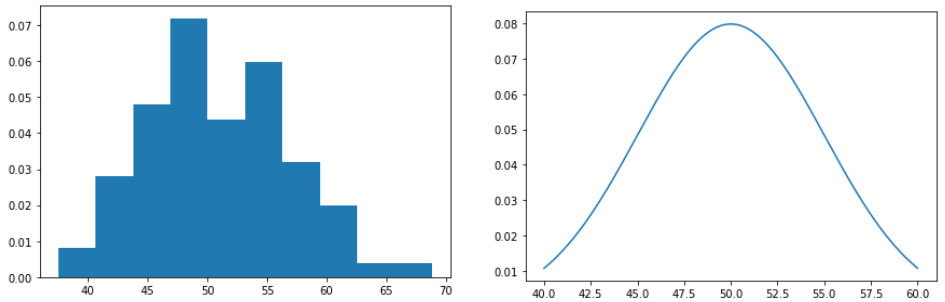
- **Exercise 1** (2 marks): Consider a random sample of a normally distributed data with mean $\mu = 50$ and standard deviation $\sigma = 5$. The Scipy's function `norm.rvs(...)` below is used to draw samples from a normal distribution.

```
# Theoretical Mean and Std
mu = 50
std = 5
qtt_samples = 80

pop_norm = norm.rvs(mu, std, size=qtt_samples)
pop_norm

# Output (yours output might be different!):
array([46.91990838, 59.40379394, 52.4235294 , 53.89375851, 52.16204872,
       49.17677246, 47.98578232, 45.05658579, 50.29725694, 56.11002371, ...])
```

To plot a histogram, considering 10 bins of the generated data, use the code:

```
plt.hist(pop_norm, bins=10, density=True);
print('REAL mean: %.2f; std: %.2f' % (np.mean(pop_norm), np.std(pop_norm)))
```

The result shall be similar to the one shown below in Fig. 2.




(a) Histogram (b) Normal curve

Figure 2: Graphical representation of the generated data.

Perform the following:

- Create another set of 1000 samples with $\mu = 50$ and $\sigma = 5$.
- Plot the histogram with 100 bins.
- Compare the results of the calculations of both sets. What conclusions can be drawn from the amount of samples compared with the mean and standard deviation calculated and the number of bins considered?

- **Exercise 2** (1 marks): Consider a sample data of the word `Biometrics` written by a "genuine" person, and an "impostor" as a signature that replicates the shape but not the original pressure. Load the sample signature from a directory listed below:

```
# make sure to put the correct path to where your data is.
# The dataset share on D2L has the following folders:
# 'signature_samples/biometrics/'
# 'signature_samples/calgary/'
# 'signature_samples/hello/'

# 'genuine' or 'impostor'
subdir = 'impostor/'
data_dir = 'signature_samples/biometrics/' + subdir


csv_file = pd.read_csv(data_dir + '1.csv')

# show the first lines of the data loaded
# the file contains 4 columns: [X, Y, Time, Pressure]
csv_file.head()

# splitting the original data/matrix into 3 variables
coord1 = csv_file[['X', 'Y']].to_numpy()
time1 = csv_file['Time'].to_numpy().reshape((-1,1))
prs1 = csv_file['Pressure'].to_numpy().reshape((-1,1))
```

- Set up the Matplotlib colormap, so your signature can be plotted in color representing the pressure:

```
# to correctly map the pressure into the colormap the normalization is required
prs1 = prs1 / np.max(prs1) * 255
prs1 = prs1.astype(int)

pressuremap = cm.get_cmap('jet', np.max(prs1)+1)
```

- Plot the signature in 2D with the color representing the pressure on each point:

```
for i in range(len(prs1)—1):
    c = pressuremap(prs1[i]).squeeze()
    im = plt.plot(coord1[i:i+2,0], —coord1[i:i+2,1], linewidth=2, c=c)
sm = cm.ScalarMappable(cmap='jet', norm=plt.Normalize(vmin=0, vmax=1))
plt.colorbar(sm)
```

Similarly to calculation of the mean and std of the random data, calculate the mean and std of the pressure for one signature, and plot the Normal distribution.

- **Exercise 3** (1 mark):

Below is the code to calculate velocity and plot velocity map of one signature: Explain why do we check for the condition `if time_diff != 0:`?

```
vel = np.zeros((len(time1)—1, 1))

for i in range(len(time1)—1):
    distance = np.sqrt((coord1[i+1,0] — coord1[i,0])**2 +
                       (coord1[i+1,1] — coord1[i,1])**2)

    # if too fast, time_diff = 0
    time_diff = abs(time1[i+1] — time1[i])
    if time_diff != 0:
        vel[i] = distance / time_diff
    vel[i] = int(vel[i] * 1000) + 1


vel = np.insert(vel, 0, 1)
vel = vel / np.max(vel) * 255
vel = vel.astype(int)

velmap = cm.get_cmap('jet', np.max(vel)+1)

for i in range(len(vel)—1):
    c = velmap(vel[i])
    if time_diff < 17: # You may need to change this threshold
        im = plt.plot(coord1[i:i+2,0], —coord1[i:i+2,1], c=c)
plt.title('Velocity')

sm = cm.ScalarMappable(cmap='jet', norm=plt.Normalize(vmin=0, vmax=1))
plt.colorbar(sm)
```

NOTE: Velocity can be calculated as shown above, but you can also define a function that returns velocity as shown below:

```
def calc_velocity(time, coords):
    vel = np.zeros((len(time)—1, 1))
```

```
    for i in range(len(time)-1):
        distance = np.sqrt((coords[i+1,0] - coords[i,0])**2 +
                           (coords[i+1,1] - coords[i,1])**2)

        # if too fast time_diff = 0
        time_diff = time[i+1] - time[i]
        if time_diff == 0:
            time_diff = 0.0001

        vel[i] = distance / time_diff
        vel[i] = int(vel[i] * 1000) + 1

    vel = np.insert(vel, 0, 1)
    vel = vel / np.max(vel) * 255
    vel = vel.astype(int)

return vel
```

- Plot in 3D, where the third coordinate represents pressure, using the code below:

```
ax = plt.axes(projection='3d')

# the line below is  to reshape to vector using .flatten method
ax.plot3D(coord1[:,0].flatten(), -coord1[:,1].flatten(), prs1.flatten())
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Pressure')
ax.set_title('Pen pressure in 3D')
```

- **Exercise 4** (3 marks): Now perform statistical analysis of pressure and velocity distribution across average values of pressure or 30 signatures of the same person. We will represent the pressure of each signature by its 'average (mean) value', and then consider the distribution of these average values across genuine signatures.

  Consider 30 genuine signatures (yours or pre-recorded sample set, part a), and calculate average (mean) values and deviation from average (which will be calculated using standard deviation) for pressure. Plot a normal distribution of the average pressure values across the 30 signatures of the same individual. Perform the same for velocity.

  Below is an example that defines an array with columns representing mean and standard deviation of the velocity and pressure:

```
# columns: mean_vel, std_vec, mean_prs, std_prs
stats_measures = np.zeros((10,4))

# using 10 signatures
for i in range(10):
    csv_file = pd.read_csv(data_dir + '%d.csv' % (i+1))
```

```
coord1 = csv_file[['X', 'Y']].to_numpy()
time1 = csv_file['Time'].to_numpy().reshape((-1,1))
prs1 = csv_file['Pressure'].to_numpy().reshape((-1,1))

vel = calc_velocity(time1, coord1)

stats_measures[i, 0] = np.mean(vel)
stats_measures[i, 1] = np.std(vel)

stats_measures[i, 2] = np.mean(prs1)
stats_measures[i, 3] = np.std(prs1)

print('#%d -- pressure: %.2f; velocity: %.2f' %\
        (i, np.mean(prs1), np.mean(vel)))
```

- **Exercise 5** (3 marks): Perform statistical analysis of average pressure and velocity distribution across 30 impostor signatures. As an "impostor", you can use either the 30 "forged" signatures (or part *b* of the pre-recorded set), or 30 writings by yourself, of different shape/word (or part *c* of the pre-recorded set). Represent the pressure (or velocity) of each signature by its average (mean), and then consider the distribution of these averages across 30 impostor signatures. How different are those distributions for "genuine" signatures from Exercise 4 and the "impostor" from Exercise 5?

## Acknowledgments

We acknowledge E. Reentov and J. Graf who contributed to the SigGet software development. The template for this Lab was developed by Dr. H. C. R. Oliveira (postdoc in the Biometric Technologies Laboratory in 2020-2022). We also acknowledge this course TAs, O. Shaposhnyk, I. Yankovyi and M. Zakir, for verifying this lab code.

---

*Dr. S. Yanushkevich*
*Biometric Technologies Laboratory*
*January 16, 2024.*