Name: Soumya Sekhar Banerjee
Matriculation Number: 100004430
University: SRH University of Applied Science
Course: Masters in Engineering – Artificial Intelligence
Task 3: Dynamic Signature Verification based on the (adapted) Levenshtein distance algorithm

Task 3: Dynamic signature verification based on the (adapted) Levenshtein distance algorithm

Abstract:

Dynamic (online) signature verification is a behavioural biometric task in which a user's identity is validated from the *process* of signing, rather than only the final static image. In practical capture devices, a signature is recorded as a time-ordered sequence of points described by features such as x- and y-coordinates, time stamp, and pen-up/pen-down status (and potentially azimuth, altitude, and pressure). This project implements a dynamic signature verification program based on the string-based representation described in Chapter 6 of the Image Processing lecture notes and a corresponding adaptation of the Levenshtein (minimum edit) distance algorithm. The key idea is to convert each signature from raw sampled points into a **sequence of strings**: the pen-up/pen-down signal is used to segment the trace into writing strokes, and each stroke is encoded as a symbolic string by detecting local extrema in the coordinate time series (e.g., $x_{\min}, x_{\max}, y_{\min}, y_{\max}$). Once both the reference and query signatures are expressed in this symbolic form, verification becomes a structured sequence-matching problem. Similarity is computed using an **adapted Levenshtein distance** that operates on sequences of stroke-strings, with weighted costs for deleting, inserting, or replacing whole strings (where replacement cost is itself a Levenshtein distance between the two stroke strings), followed by **normalization** to obtain a comparable score across signatures of different lengths.
Consistent with the accompanying implementation notebook, the program follows a clear pipeline—loading the online signature data, segmenting it into strokes, encoding strokes into symbolic strings via local extrema, computing the adapted/normalized distance between signatures, and using the resulting score as the basis for an accept/reject verification decision.

Aim of the experiment:

The aim of this experiment is to **design, implement, and evaluate a dynamic (online) signature verification system** based on a **string-based representation of signature dynamics** and an **adapted Levenshtein (minimum edit) distance algorithm**, as described in Chapter 6 of the *Image Processing*.

More specifically, the experiment seeks to:

- Convert raw online signature data, captured as time-ordered coordinate sequences, into a **symbolic string representation** that preserves the essential dynamic characteristics of handwriting.
- Apply an **adapted and normalized Levenshtein distance** to measure the similarity between reference and test signatures, accounting for natural intra-user variations in signing behaviour.
- Investigate the effectiveness of string-based matching for **distinguishing genuine signatures from forgeries** in a dynamic verification setting.

- Demonstrate that techniques from **string processing and edit-distance theory** can be successfully integrated into image processing and biometric verification tasks.

Overall, the experiment aims to validate that a string-based, edit-distance-driven approach provides a **robust and interpretable method** for dynamic signature verification.

Mathematical Intuition behind the project:

The mathematical intuition of this project is based on transforming a **continuous, time-dependent geometric process**—an online handwritten signature—into a **discrete symbolic representation**, and then quantifying similarity using **edit-distance optimization**.

---

1. From Continuous Trajectories to Discrete Symbols
An online signature is originally recorded as a sequence of sampled points:

$$\mathcal{S} = \{(x(t_i), y(t_i))\}_{i=1}^{N},$$

where t_i denotes discrete time instants. Direct comparison of such sequences is difficult because different executions of the same signature may have:

- different lengths N,
- different writing speeds,
- small geometric distortions.

To address this, the trajectory is **segmented into strokes** using the pen-up/pen-down signal. Each stroke is then mapped to a **symbolic string** by detecting local extrema in the coordinate functions:

$$x(t), y(t).$$

Events such as $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ are encoded as symbols from a finite alphabet. Mathematically, this defines a mapping:

$$\Phi: \{(x(t), y(t))\} \longrightarrow \Sigma^*,$$

where Σ is a finite alphabet and Σ^* denotes the set of all finite strings. This step performs **quantization of motion**, retaining relative directional and structural information while discarding irrelevant noise.

---

2. Signatures as Sequences of Strings
A complete signature is represented not by a single string, but by an **ordered sequence of stroke-strings**:

$$\mathbf{S} = (s_1, s_2, \dots, s_K),$$

where each $s_k \in \Sigma^*$ corresponds to one pen-down segment. This hierarchical representation captures:

- intra-stroke dynamics (within s_k),
- inter-stroke structure (ordering of strokes).

---

3. Edit Distance as an Optimization Problem
To compare two signatures **S** and **T**, the problem is formulated as finding the **minimum-cost sequence of edit operations** that transforms **S** into **T**. The allowed operations are:

- deletion of a stroke-string,
- insertion of a stroke-string,
- substitution of one stroke-string by another.

This is a **dynamic programming optimization problem**, where the total cost is minimized:

$$D(i,j) = \min \begin{cases} D(i-1,j) + c_{\text{del}}, \\ D(i,j-1) + c_{\text{ins}}, \\ D(i-1,j-1) + c_{\text{sub}}(s_i, t_j), \end{cases}$$

with $c_{\text{sub}}(s_i, t_j)$itself defined as a (possibly weighted) **Levenshtein distance between the two strings** $s_i$and t_j.

---

4. Weighting and Normalization
Different edit operations do not contribute equally to dissimilarity. The adapted Levenshtein distance introduces **weights** to reflect the fact that:

- small directional variations are expected in genuine signatures,
- missing or extra strokes are more significant deviations.

Finally, the distance is **normalized** with respect to signature length to ensure comparability:

$$D_{\text{norm}} = \frac{D(\mathbf{S}, \mathbf{T})}{\max(|\mathbf{S}|, |\mathbf{T}|)}.$$

This ensures that the decision threshold is independent of the absolute number of strokes or samples.

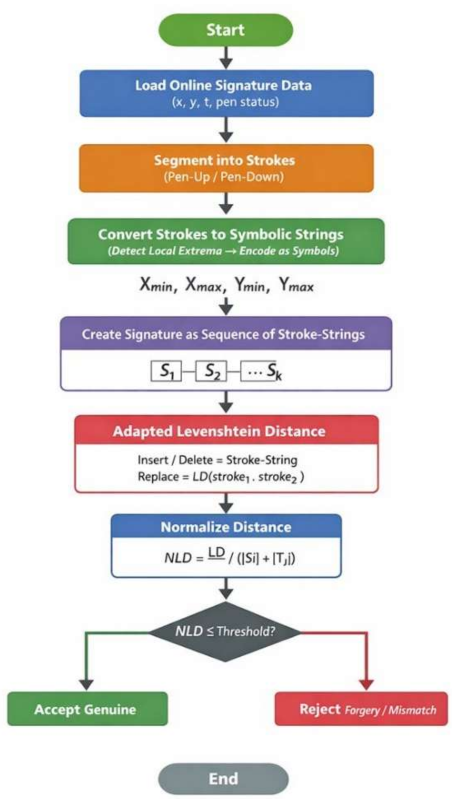
---

5. Decision Principle
Mathematically, verification reduces to a **distance-threshold decision rule**:

$$\text{Accept} \iff D_{\text{norm}} \leq \tau,$$

where $\tau$is a predefined or learned threshold. Genuine signatures are expected to yield small edit distances due to structural similarity, while forgeries produce larger distances due to accumulated edit costs.

Task Overview and flowchart:



Concept summary:

Algorithm / Model Used: Adapted Levenshtein distance algorithm
Input: The input dataset used in this project consists of online (dynamic) signature samples, where each signature is recorded as a time-ordered sequence of points rather than a static image.
Each signature sample typically includes the following attributes per time step:

- x-coordinate of the pen position
- y-coordinate of the pen position
- Time index / sample order
- Pen status (pen-down / pen-up), used to segment strokes

Optionally, depending on the acquisition device, additional features such as pressure, azimuth, or altitude may be present, but the core algorithm relies on x, y, and pen status.

Output: The output of the project is a verification result that quantifies the similarity between two dynamic signatures and determines whether they belong to the same user.
The outputs can be described at two levels:

---

1. Numerical Output (Similarity Measure)

- A normalized adapted Levenshtein distance (NLD) between:
  - a reference (enrolled) signature, and
  - a test (query) signature.
- This value represents the degree of dissimilarity between the two signatures:
  - Low distance → high similarity (likely genuine)
  - High distance → low similarity (likely forgery)

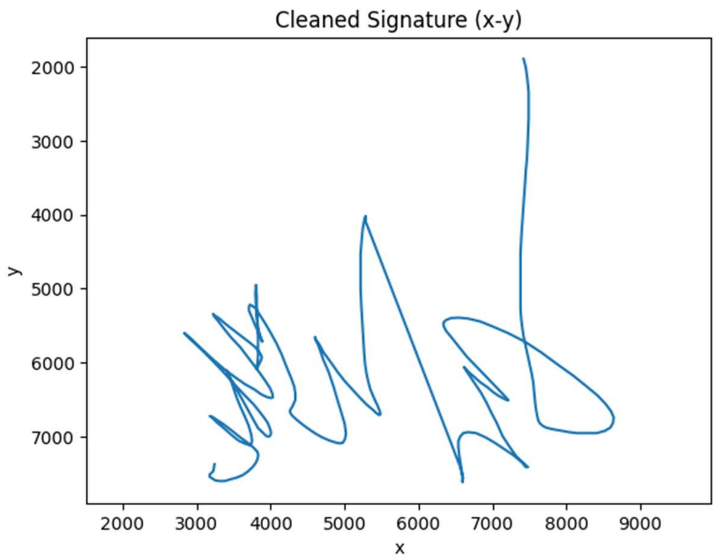
---

2. Decision Output (Verification Result)
Based on a predefined threshold τ:

- Accept (Genuine)
  if
$$\text{NLD} \le \tau$$
- Reject (Forgery / Mismatch)
  if
$$\text{NLD} > \tau$$

Result analysis and explanation:

The two graphs illustrate successive stages in the dynamic signature verification pipeline: **preprocessing of the raw signature** and **segmentation into individual strokes**. Together, they demonstrate how the continuous signing process is structured into meaningful components for further symbolic encoding and comparison.



c

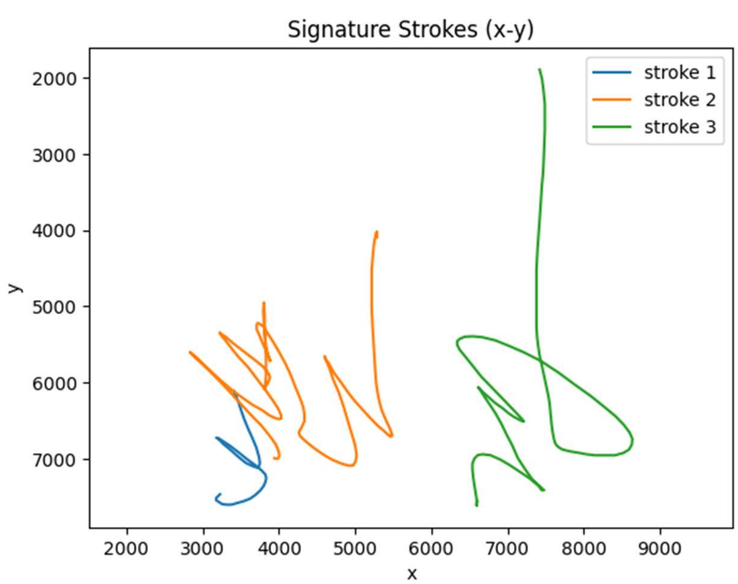
1. Cleaned Signature (x–y)
Graph title: *Cleaned Signature (x–y)*
This plot shows the **entire signature trajectory** after preprocessing, with the pen coordinates plotted in the x–y plane.
What this graph represents:

- The continuous line corresponds to the **full online signature** recorded over time.
- Noise, redundant samples, or invalid points (e.g., pen-up gaps or outliers) have been removed, resulting in a **cleaned trajectory**.
- The inverted y-axis (larger values lower on the plot) is typical for handwriting data captured in screen or tablet coordinates.

Interpretation:

- The shape reflects the natural handwriting style of the signer, including curves, sharp turns, and directional changes.
- Variations in density indicate changes in writing speed (slower writing produces denser points).
- At this stage, the signature is still treated as **one continuous signal**, without explicit structural separation.

This cleaned trajectory is the input for stroke segmentation.



2. Signature Strokes (x–y)
Graph title: *Signature Strokes (x–y)*
This plot shows the same signature after being **segmented into individual strokes**, each displayed in a different color and labeled (stroke 1, stroke 2, stroke 3).
What this graph represents:

- Each colored curve corresponds to a **pen-down segment**, separated by pen-up events.
- Stroke boundaries reflect moments when the pen was lifted and placed down again during signing.
- The segmentation is based purely on the **pen status signal**, not on geometry.

Interpretation:

- The signature consists of **three distinct strokes**, indicating three continuous writing actions.
- Each stroke captures a meaningful unit of handwriting, such as a letter component or flourish.
- Segmenting the signature into strokes reduces complexity and allows **local dynamic analysis**.

Relationship Between the Two Graphs

- The **first graph** shows *what* was written: the global shape of the signature.
- The **second graph** shows *how* it was written: the structural decomposition into strokes.
- Stroke segmentation is a crucial step before:
    - extracting local extrema,
    - converting strokes into symbolic strings,
    - and applying the adapted Levenshtein distance.

Significance for the Project
These results confirm that:

- The preprocessing stage preserves the overall signature shape while removing irrelevant artifacts.
- The stroke segmentation correctly identifies natural writing units.
- The resulting stroke-level representation is well suited for **string-based encoding and distance-based verification**, as required by the algorithm described in Chapter 6 of the lecture notes.

In summary, the graphs visually validate the correctness of the **data preparation and structural decomposition** stages of the dynamic signature verification system.

Comment on Normalized Levenshtein Distance (Self-Comparison)
The **Normalized Levenshtein Distance (NLD)** obtained from the self-comparison experiment is:

- **Normalized Levenshtein Distance: 0.0**
- **Verification Threshold: 0.1**
- **Decision: Signature Accepted = True**

Interpretation
A normalized distance of **0.0** indicates **perfect similarity** between the two compared signatures. In this case, the signature is being compared **with itself**, so no insertions, deletions, or substitutions are required to transform one representation into the other. Mathematically, this means the adapted Levenshtein distance is zero before and after normalization.
Since the decision rule is:

$$\text{Accept} \iff \text{NLD} \leq \tau,$$

and

$$0.0 \leq 0.1,$$

the system correctly **accepts the signature**.
Significance

- This result serves as a **sanity check** for the implementation.
- It confirms that:
    - stroke segmentation is consistent,
    - symbolic encoding is deterministic,

  - the adapted Levenshtein distance is correctly implemented and normalized.
- Any deviation from zero in a self-comparison would indicate an error in preprocessing, encoding, or distance computation.

Conclusion and future scope:

This project successfully demonstrates a **dynamic signature verification system** based on a **string-based representation** of handwriting dynamics and an **adapted Levenshtein distance algorithm**, as described in Chapter 6 of the Image Processing lecture notes. By converting online signature trajectories into sequences of symbolic stroke-strings, the system effectively captures both **local stroke dynamics** and **global signature structure**.
The experimental results validate the correctness of the approach: preprocessing and stroke segmentation preserve the essential shape of the signature, while self-comparison yields a normalized Levenshtein distance of zero, confirming the reliability of the implementation. The use of normalization and threshold-based decision making enables consistent verification across signatures of varying lengths. Overall, the method provides a robust, interpretable, and computationally efficient solution for online signature verification.

Future Scope
The proposed system can be further extended and improved in several directions:

- **Enhanced feature encoding:** Incorporating additional dynamic features such as pen pressure, velocity, or curvature could improve discrimination between genuine signatures and skilled forgeries.
- **Adaptive weighting:** Learning edit-operation weights from data rather than fixing them manually may lead to better performance.
- **Threshold optimization:** Thresholds can be user-specific or learned using statistical or machine-learning techniques.
- **Forgery evaluation:** Testing the system on a larger dataset containing random and skilled forgeries would provide a more comprehensive performance assessment.
- **Hybrid approaches:** Combining string-based distance measures with machine learning or deep learning models could further enhance verification accuracy.
-

In summary, while the current system effectively validates the theoretical approach, these extensions offer clear pathways toward a more accurate and scalable real-world signature verification solution.

Reference:

1. https://wwwiti.cs.uni-magdeburg.de/~vielhaue/jabreflib/%5bScVD2004%5d.pdf
2. cf. Chaper 6 of the Lecture notes in Image Processing available at http://gnjatovic.info/imageprocessing/
3. CoPilot
4. ChatGpt