Use of Singular Value Decomposition for Signature Verification in WarioWare Gold

Sebastian Hung Yansen - 13523070[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523070@mahasiswa.itb.ac.id, sebastianhung25@gmail.com

***Abstract*— This paper explores the application of Singular Value Decomposition (SVD) for signature verification, as exemplified in the "Autograph" minigame from *WarioWare Gold*. Using SVD, signatures are decomposed into principal components, allowing efficient comparison via Euclidean distance. The technique is implemented in a custom web interface for experimentation, demonstrating rapid processing and resource efficiency. The findings validate the viability of SVD for signature verification especially for achieving high accuracy while minimizing computational overhead.**

***Keywords*—Signature, SVD, Videogame, Verification**

I. Introduction

A person or their handwriting can be identified through behavioral biometrics based on their signature. Signature verification is important in establishing a person's authority. As more and more items require signatures, it can be quite a challenge in verifying each and every signature one by one manually. Therefore, it is important that an automated system is placed in order to automatically detect whether a signature corresponds to the person's original signature or not.

In a study conducted by Kamel, Sayeed, and Ellis [1] on a glove-based method for online signature verification, Singular Value Decomposition (SVD) is applied to extract singular vectors that capture the highest energy of the signature data. This results in the creation of a principal subspace, which represents the majority of the original data's variation. By characterizing the signature data through its -dimensional principal subspace, authenticity is assessed by measuring the angles between these subspaces. The SVD-based signature verification approach achieved an Equal Error Rate (EER) of 2.46%, highlighting its promising effectiveness in the proposed methodology.

An example of a signature verification being used can be seen in the game WarioWare Gold, more specifically the minigame Autograph. In the minigame, the player is prompted to make their signature which will be used as a reference. Once the player has wrote their signature as the reference, they are then tasked to sign various items with their signature as much as possible. The items can vary from a note, a t-shirt, a toilet paper roll, and even a game system. If the signature doesn't match the reference signature, the player earns less points and can even earn no points at all.

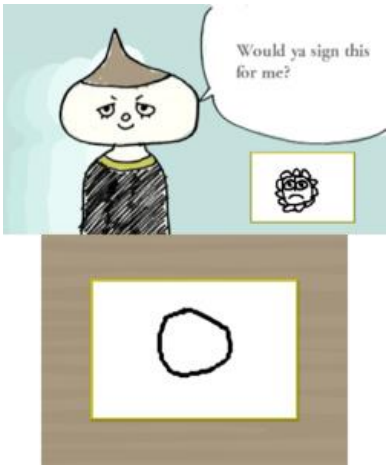

Fig. 1.1. Autograph! Gameplay
Source: https://www.mariowiki.com/Autograph*!*

II. Theoretical Basis

A. Matrix

A matrix is a square or rectangular array of numbers. The matrix size is represented with m x n with m being number of rows and n being number of columns. The number on a specific row and column can be denoted by a_{ij} i representing the rows and j representing the columns of the number. Matrices can be used for storing information and for arithmetic operations. Addition and subtraction of matrices require 2 matrices with the same dimensions and both operations work by adding or subtracting the corresponding elements. Multiple types of matrix multiplication exist, one of which is the scalar multiplication. With scalar multiplication, each element in a matrix is multiplied by a scalar r

A matrix can be transposed by flipping the rows and columns of the original matrix. Suppose that a matrix exists called Matrix A, the matrix turns into A^T when transposed. Formally, the transpose of an m x n matrix is the n x m of the transposed matrix.

There also exists the identity matrix, a matrix shaped as a square n x n with ones filling up the main diagonal and zeros elsewhere. Suppose that n = 3, the identity

matrix would be:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The identity matrix is special and has many important uses which include matrix inversion, basis in vector spaces, eigenvalues and eigenvectors, diagonalization and decomposition, and many more.

The matrix inverse is denoted as A^{-1} for matrix A shaped as a square. It is defined such that

$$A \cdot A^{-1} = A^{-1} \cdot A = I$$

with I as the identity matrix. Other than the square shape, for a matrix to have an inverse requires said matrix to be non-singular. If and only if a square matrix's determinant is zero, the matrix is considered singular.

The determinant of a matrix (suppose in this case, matrix A) is denoted by $|A|$ or det(A). It is a scalar value and has numerous applications including solving systems of equations, understanding transformations, and determining invertibility.

B. Vector Space

A vector space consists of a collection of vectors along with operations of vector addition and scalar multiplication [4]. Those operations must satisfy a list of axioms for all vectors which include:

1. Closure
2. Commutativity
3. Associativity
4. Identity
5. Inverses
6. Distributive Properties
7. Associativity of Scalar Multiplication
8. Scalar Identity

A basis is a set of linearly independent vectors that span the space. The number of vectors in the basis is the dimension of the vector space.

One important property of vector spaces is the ability to measure distances between vectors. The Euclidean distance is a commonly used metric for this purpose. For two vectors u,v∈Rnu,v∈Rn, the Euclidean distance is defined as:

$$d(\boldsymbol{u}, \boldsymbol{v}) = ||\boldsymbol{u} - \boldsymbol{v}|| = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

This distance represents the straight-line distance between two points in a Euclidean space. It is foundational in many applications, including analyzing the behavior of transformations and comparing data points in vector spaces.

The mapping between two vector spaces is represented through linear transformation. It preserves the operations of vector addition and scalar multiplication. Linear transformations itself can be represented by matrices and modifies vectors while preserving their underlying linear structure.

The behavior of linear transformations is described using eigenvalues and eigenvectors. An eigenvector is a non-zero vector (let's say vector v) of a matrix A if applying A to v scales the vector without changing its direction:

$$Av = \lambda v$$

the λ in the equation is called the eigenvalue corresponding to v. It represents how much v is stretched or compressed during the transformation. An eigenvector is always associated with its eigenvalue and the eigenvalue can be zero, positive, or negative. Not only that, the eigenvalue and eigenvector can show the vector's direction where:

λ > 0 : The vector's direction is preserved
λ < 0 : The vector's direction is reversed
λ = 0 : The vector maps to the zero vector

Finding eigenvalues and eigenvectors are quite simple by using these equations:

$$\det(A - \lambda I) = 0$$

$$(A - \lambda I)v = 0$$

In geometric interpretation, eigenvectors indicate the directions along which a linear transformation acts as simple scaling or correspond to directions of unchanged orientation. The eigenvalue tells how much scaling or compression occurs in that direction.

Vector spaces provide the required framework for studying solutions to linear equations, transformations, and more advanced structures including Singular Value Decomposition. Euclidean distance, eigenvalues, and eigenvectors collectively help in understanding and quantifying the effects of transformations in vector spaces.

C. Singular Value Decomposition

Singular Value Decomposition (SVD) is a concept in linear algebra that provides a way to decompose a given matrix into simpler constituent components. It is widely used for data analysis, dimensionality reduction, and matrix approximation. SVD is represented as:

$$A = U \sum V^T$$

where:

- U is an orthogonal matrix with columns representing the left singular vectors of A.
- Σ is a diagonal matrix containing the singular values of A in descending order along its diagonal.

- V is an orthogonal matrix with columns representing the right singular vectors of A.

SVD itself has a few properties which include:
- Rank Approximation
- Energy Preservation
- Orthogonality

It is required to solve for the eigenvalues and eigenvectors before computing SVD. The eigenvalues and eigenvectors are counted from A^TA and AA^T and they form the columns of V and U respectively.

C. Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method designed to reduce dimensionality, extract features, and facilitate data visualization. By transforming high-dimensional datasets into lower-dimensional spaces, PCA retains as much of the original variance as possible. The process begins by centering the data through mean subtraction for each feature. Following this, the covariance matrix of the centered data is calculated to capture relationships and variances among dimensions. The eigenvectors of this covariance matrix, termed principal components, indicate the directions of maximum variance. Their corresponding eigenvalues quantify the magnitude of variance in these directions. Finally, the data is projected onto a subspace formed by the top eigenvectors, corresponding to the largest eigenvalues, thereby preserving the most significant variance in the dataset. PCA can be implemented with multiple methods with SVD being one of them. SVD is used for implementation for numerical stability and efficiency.

III. IMPLEMENTATION

The player is first required to make their signature which will be used as a reference. If the signature is too simple or too large, it will be rejected until the player makes a suitable signature.

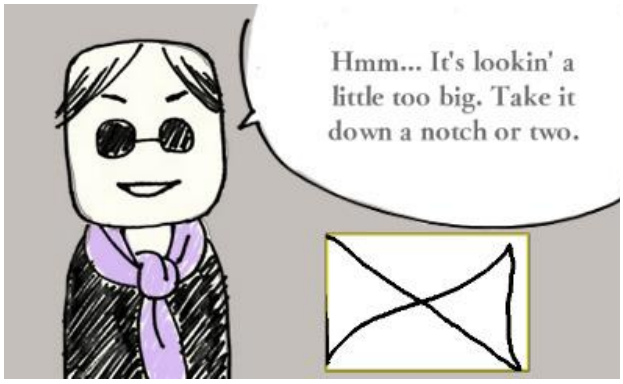

Fig. 3.1. Large signature submitted

Fig. 3.2. Simple signature submitted

Fig. 3.3. Valid signature submitted

The valid signature then follows the procedure for PCA. The image goes through data centering, SVD computation, as well as data projection. The projected data is stored and will be compared to the upcoming signatures.

After submitting a valid signature, the player is immediately put into the main game where they are required to sign items correctly. The player earns points based on how close the signature is to the reference. The score is communicated to the player through the in-game dialogue from the fans themselves. It is possible for the player to earn less points and can even score zero points if the signature doesn't match the reference.

Fig. 3.4. Signature almost similar to reference

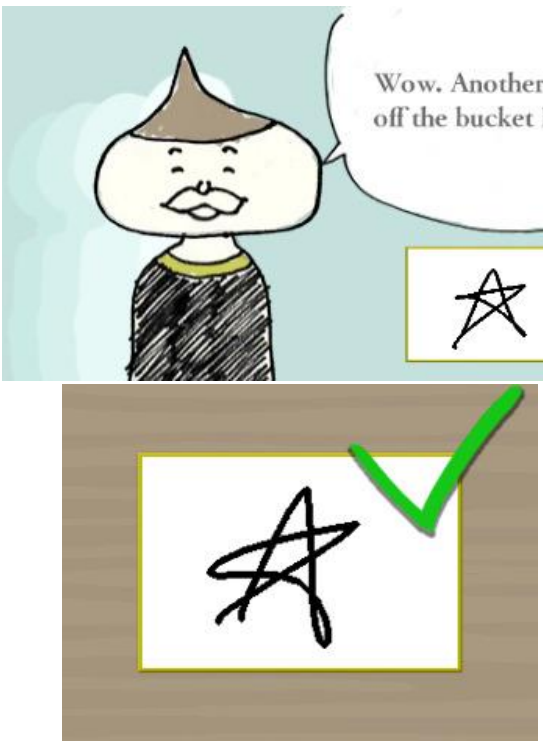

Fig. 3.5. Signature very similar to reference

Fig. 3.6. Signature not similar to reference

For every signature that the player submits, it follows the same PCA process. After going through the PCA process, the Euclidean distance is then counted. Some Euclidean distance limits are already set from the start and it is compared to the Euclidean distance that was counted before. The player then gains points based on how similar their signature is to the reference.

Based on how the player performs, the game calculates the score earned at the end.

Fig. 3.7. The final score

IV. EXPERIMENT

For the experiment an extremely simple website is created that has its own canvas so the user can draw within the website instead of having to use a separate application. The frontend uses React and the backend uses Python. The website is used for to compare signatures and output the

similarity percentage.

Fig. 4.1. Website Implementation

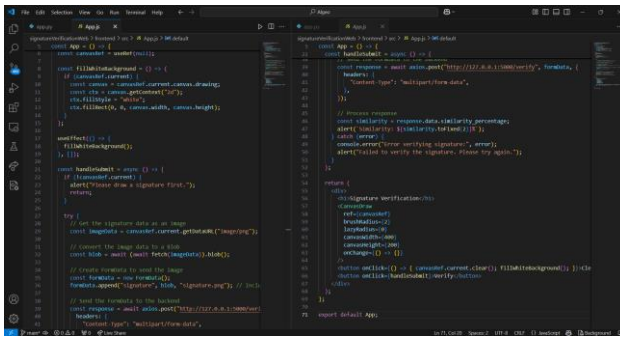

Fig. 4.2. Frontend Implementation

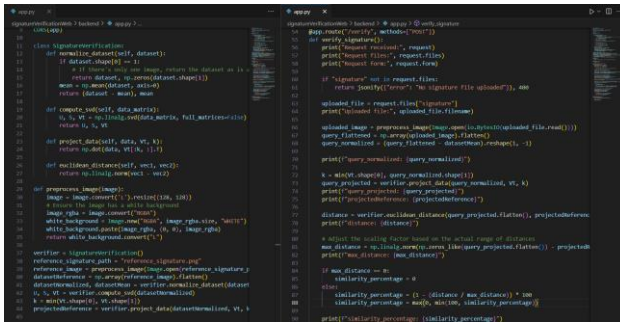

Fig. 4.3. Backend Implementation

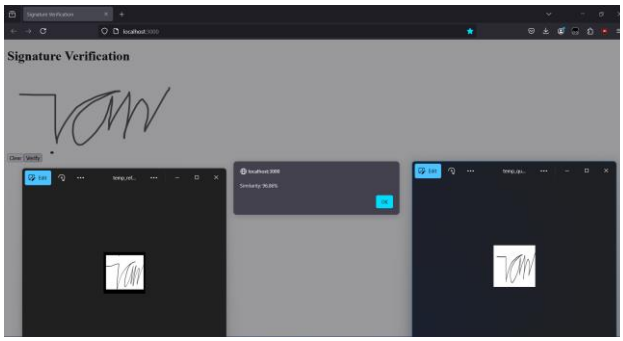

Fig. 4.4. Implementation Results with reference and query image

Based on the results, the SVD-based approach achieved 96.86% accuracy with a processing time of less than half a second. The process has thresholds that can be changed such as the maximum Euclidean distance, k, etc. The exact numbers within the minigame is unknown and the system currently uses automated numbers. The thresholds should be able to be tuned to achieve the best results.

The SVD method is quick and allows the user to get the similarity score almost immediately. Not only that, it saves on overall computational process and uses fewer resources. This is especially useful for its use in the game to not make the player wait too long. The game is also played on a 3DS which has relatively weak hardware compared to current-gen systems.

V. Conclusion

Based on the analysis and testing that has been done, it can be concluded that verifying signatures using the SVD method is applicable to compare signatures. While there are other methods out there that are probably more effective, it is still quite impressive for such a weak system to be able to compute a signature verification minigame within a game.

VII. Acknowledgment

The writer of this paper would like to thank the Lord for His guidance and mercy for giving me the ability and strength to finish this paper. The writer would also like to thank their parents, friends, and teacher for their support in knowledge and material. The writer is grateful to everyone that they've met and those who've helped the writer go through tough parts of their life.

References

[1] Rahmat, R., "Online Signature Verification using SVD Method," 2009. [Online]. Available: http://utpedia.utp.edu.my/id/eprint/8809/. [Accessed: Jan. 2, 2025].

[2] G. Strang, Linear Algebra and Its Applications. 4th ed. Brooks Cole, 2005.

[3] G. Strang, *Introduction to Linear Algebra*, 4th ed. Wellesley-Cambridge Press, 2009. ISBN: 978-0-9802327-1-4.

[4] "Matrices," in Fundamental Data Compression. Amsterdam: Elsevier, 2006, pp. 223-229. doi: 10.1016/B978-075066310-6/50016-7

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Januari 2025

Sebastian Hung Yansen/13523070