Eigenvectors as Directions of Meaning in High-Dimensional Data: A Geometric Perspective for Drone State and Motion Analysis

Ariel Cornelius Sitorus - 13524085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524085@std.stei.itb.ac.id, [2]arielsitorus504@gmail.com

***Abstract*—This paper explores eigenvectors and eigenvalues from a geometric perspective, specifically how these concepts can be applied to analyze drone position data in 3D space. The core idea is simple: if we have a collection of data points forming a particular pattern (e.g., elongated in one direction), the eigenvectors of the covariance matrix will point toward that pattern. For drone analysis, the dominant eigenvector indicates the primary flight direction, while eigenvalues show how much variance exists in each direction. Using simulated linear and spiral trajectories, this paper demonstrates that eigenvectors can reconstruct flight direction with high accuracy (error around 1.42°). This approach forms the mathematical foundation of Principal Component Analysis (PCA), widely used in machine learning and data analysis.**

***Index Terms*—Eigenvector, Eigenvalue, Covariance Matrix, Spectral Theorem, Quadratic Forms, Orthogonal Projection, Euclidean Space, Basis Transformation, Drone Analysis.**

I. Introduction

Have you ever wondered how to extract "meaningful" information from a bunch of data points scattered in 3D space? For instance, if we have drone position data from point A to B, can we determine its flight direction without knowing the start and end points?

The answer lies in **Eigenvectors** and **Eigenvalues**, two topics that might seem abstract when first encountered in linear algebra class. But behind the mathematical definition $A\mathbf{v} = \lambda\mathbf{v}$, there's a powerful geometric meaning: eigenvectors show "special" directions where data has maximum variation.

Imagine a set of data points forming an elongated ellipse. If we compute the eigenvectors of the covariance matrix, the first eigenvector will point along the long axis, while the second points along the short axis. These are called the **Principal Axes**, the main directions that describe the "shape" of the data distribution.

In this paper, I will explain:

- How eigenvectors can "discover" drone flight direction just from position data
- Why eigenvalues represent variance (how spread out the data is in a certain direction)
- The connection between covariance matrix and hyper-ellipsoid geometry
- Practical implementation using Python and NumPy

The paper is structured as follows: Section II covers theoretical foundations on vector spaces and eigendecomposition. Section III explains the drone simulation methodology. Section IV presents the Python implementation. Section V discusses experimental results, Section VI is conclusion and future work, and Section VII is the appendix.

II. Theoretical Foundation

To provide a robust analysis, we must first establish the definitions and theorems governing the behavior of vectors in Euclidean space.

A. Linear Algebra and Eigenvectors

1) Euclidean Vector Spaces and Inner Products: We define our workspace as the real coordinate space $\mathbb{R}^n$, equipped with the standard inner product (dot product). For any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T\mathbf{v} = \sum_{i=1}^{n} u_i v_i \tag{1}$$

The inner product induces a norm $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$, which defines the geometric "length" of a vector. Crucially, the inner product allows us to define **Orthogonality**. Two vectors are orthogonal if and only if their inner product is zero.

2) Linear Transformations and Change of Basis: A linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ maps vectors while preserving vector addition and scalar multiplication. Every linear transformation can be represented by a matrix A.

Consider a vector $\mathbf{x}$. Its coordinates $[\mathbf{x}]_B$ relative to a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ are related to its standard coordinates $[\mathbf{x}]_E$ by the change-of-basis matrix P:

$$[\mathbf{x}]_E = P[\mathbf{x}]_B \tag{2}$$

where the columns of P are the basis vectors of B. If B is an **Orthonormal Basis**, then P is an orthogonal matrix ($P^T = P^{-1}$).

3) The Eigenvalue Problem: An eigenvector of a square matrix A is a non-zero vector $\mathbf{v}$ such that:

$$A\mathbf{v} = \lambda\mathbf{v} \tag{3}$$

where λ is the eigenvalue. Geometrically, eigenvectors represent the **invariant directions** of the transformation. To find eigenvalues, we solve the **characteristic equation**:

$$\det(A - \lambda I) = 0 \tag{4}$$

B. Geometric Interpretation of High-Dimensional Data

1) Derivation of the Covariance Matrix: Let X be an $N \times d$ matrix representing N data points in $\mathbb{R}^d$. The mean vector $\boldsymbol{\mu}$ is the centroid of the data. The sample covariance matrix Σ is:

$$\Sigma = \frac{1}{N-1}\bar{X}^T\bar{X} \tag{5}$$

where $\bar{X}$ is the centered data matrix. The element Σ_{ij} represents the covariance between dimension i and j.

2) The Spectral Theorem: The **Spectral Theorem** guarantees that for symmetric matrix Σ:

1) Eigenvalues are real.
2) The matrix is always diagonalizable.
3) Eigenvectors corresponding to distinct eigenvalues are orthogonal.

Thus, there exists an orthogonal matrix Q and diagonal matrix Λ such that:

$$\Sigma = Q\Lambda Q^T \tag{6}$$

3) Quadratic Forms and Hyper-Ellipsoids: The equation:

$$(\mathbf{x} - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}) = c^2 \tag{7}$$

defines a **Hyper-ellipsoid** in $\mathbb{R}^d$. By substituting $\Sigma = Q\Lambda Q^T$:

$$\sum_{i=1}^{d}\frac{y_i^2}{\lambda_i} = c^2 \tag{8}$$

This proves that:

- Eigenvectors (columns of Q) are the directions of the principal axes.
- Square roots of eigenvalues ($\sqrt{\lambda_i}$) are the lengths of the semi-axes.

4) Worked Example: 2D Eigendecomposition: Consider a simple 2D covariance matrix:

$$\Sigma = \begin{bmatrix}5 & 3\\ 3 & 5\end{bmatrix} \tag{9}$$

To find eigenvalues, we solve the characteristic equation:

$$\det(\Sigma - \lambda I) = \det\begin{bmatrix}5-\lambda & 3\\ 3 & 5-\lambda\end{bmatrix} = 0 \tag{10}$$

Expanding:

$$(5-\lambda)^2 - 9 = 0 \implies \lambda^2 - 10\lambda + 16 = 0 \tag{11}$$

Using the quadratic formula: $\lambda_1 = 8$ and $\lambda_2 = 2$.

For $\lambda_1 = 8$, we solve $(\Sigma - 8I)\mathbf{v} = 0$:

$$\begin{bmatrix}-3 & 3\\ 3 & -3\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix} = 0 \implies v_1 = v_2 \tag{12}$$

Normalized eigenvector: $\mathbf{v}_1 = \frac{1}{\sqrt{2}}[1,1]^T$ (45° direction).

For $\lambda_2 = 2$: $\mathbf{v}_2 = \frac{1}{\sqrt{2}}[1,-1]^T$ (135° direction).

This example illustrates how eigenvectors naturally align with the data's principal axes of variation. The eigenvalue ratio $\lambda_1/\lambda_2 = 4$ indicates that variance along $\mathbf{v}_1$ is four times greater than along $\mathbf{v}_2$.

C. Principal Component Analysis (PCA)

PCA is a direct application of eigendecomposition to dimensionality reduction. Given data matrix X with N samples in d dimensions:

1) Center the data: $\bar{X} = X - \boldsymbol{\mu}$
2) Compute covariance: $\Sigma = \frac{1}{N-1}\bar{X}^T\bar{X}$
3) Eigendecompose: $\Sigma = Q\Lambda Q^T$
4) Project: $Y = \bar{X}Q_k$ where Q_k contains the top k eigenvectors

The key insight is that projecting onto the eigenvectors with largest eigenvalues preserves maximum variance. This is precisely what we exploit in drone trajectory analysis: the dominant eigenvector captures the primary motion direction.

The explained variance ratio for k components is:

$$\text{Explained Variance} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{d}\lambda_i} \tag{13}$$

III. Methodology

A. Drone State and Motion Modeling

The drone state model is defined in Euclidean space $\mathbb{R}^3$ with position vector:

$$\mathbf{s}(t) = \begin{bmatrix}x(t)\\ y(t)\\ z(t)\end{bmatrix} \tag{14}$$

B. High-Dimensional Data Representation

1) Transformation Setup: We begin with a point cloud Z distributed as $\mathcal{N}(0, I)$ (unit sphere). We apply:

1. Scaling Transformation (S):

$$S = \begin{bmatrix}10 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{15}$$

with $\sigma_1 = 10$ (primary motion), $\sigma_2 = 3$ (drift), $\sigma_3 = 1$ (noise).

2. Rotation Transformation (R):

$$R_z(45°) = \begin{bmatrix}\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0\\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0\\ 0 & 0 & 1\end{bmatrix} \tag{16}$$

The synthetic dataset is:

$$\mathbf{x}_i = R \cdot S \cdot \mathbf{z}_i + \mathbf{t} \tag{17}$$

C. Eigenvector Analysis Pipeline

The reconstruction algorithm:

1) Compute centroid $\hat{\mu}$
2) Compute centered matrix $\bar{X}$
3) Compute covariance matrix Σ
4) Solve $\det(\Sigma - \lambda I) = 0$ for eigenvalues
5) Find eigenvectors from nullspace of $(\Sigma - \lambda I)$

IV. Implementation

This section presents the modular Python implementation. The code is organized into separate modules for maintainability and reusability.

A. Project Structure

The implementation follows a modular architecture:

- `data_generation.py`: Synthetic data generation
- `eigenanalysis.py`: Eigenstructure and PCA analysis
- `interpretation.py`: Drone dynamics interpretation
- `visualization.py`: 3D visualization
- `main.py`: Main entry point

B. Data Generation Module

```python
import numpy as np

def generate_linear_flight(n_samples=500, seed=42):
    np.random.seed(seed)
    white_data = np.random.randn(3, n_samples)

    scale = np.array([[10, 0, 0], [0, 3, 0], [0, 0, 1]])
    theta = np.radians(45)
    c, s = np.cos(theta), np.sin(theta)
    rotation_z = np.array([[c, -s, 0], [s, c, 0], [0, 0, 1]])

    transform = rotation_z @ scale
    return (transform @ white_data).T

def generate_spiral_flight(n_samples=500, seed=42, radius=10, height=30):
    np.random.seed(seed)
    t = np.linspace(0, 6*np.pi, n_samples)
    x = radius * np.cos(t) + np.random.randn(n_samples)*0.5
    y = radius * np.sin(t) + np.random.randn(n_samples)*0.5
    z = (height/(6*np.pi)) * t + np.random.randn(n_samples)*0.3
    return np.column_stack([x, y, z])
```

Listing 1. data_generation.py

C. Eigenanalysis Module

```python
import numpy as np

def compute_covariance(data):
    mean_vec = np.mean(data, axis=0)
    centered = data - mean_vec
    cov_matrix = np.cov(centered, rowvar=False)
    return mean_vec, centered, cov_matrix

def eigendecompose(matrix):
    eigenvalues, eigenvectors = np.linalg.eig(matrix)
    idx = eigenvalues.argsort()[::-1]
    return eigenvalues[idx].real, eigenvectors[:, idx].real

def analyze_eigenstructure(data):
    mean_vec, centered, cov_matrix = compute_covariance(data)
    eigenvalues, eigenvectors = eigendecompose(cov_matrix)
    explained_var = eigenvalues / eigenvalues.sum()
    return {
        'mean': mean_vec, 'covariance': cov_matrix,
        'eigenvalues': eigenvalues, 'eigenvectors': eigenvectors,
        'explained_variance': explained_var
    }

def pca_transform(data, n_components=2):
    result = analyze_eigenstructure(data)
    centered = data - result['mean']
    W = result['eigenvectors'][:, :n_components]
    return centered @ W, result['explained_variance'][:n_components], W
```

Listing 2. eigenanalysis.py

D. Interpretation Module

```python
import numpy as np

def compute_heading_pitch(eigenvector):
    v = eigenvector
    heading = np.degrees(np.arctan2(v[1], v[0]))
    horiz = np.sqrt(v[0]**2 + v[1]**2)
    pitch = np.degrees(np.arctan2(v[2], horiz))
    return heading, pitch

def interpret_eigenvalues(eigenvalues):
    ratios = eigenvalues / eigenvalues.sum()
    labels = ["Primary motion", "Lateral drift", "Vertical noise"]
    return [{'eigenvalue': val, 'variance_percent': ratio * 100,
             'interpretation': labels[i] if i < 3 else f"PC{i+1}"}
            for i, (val, ratio) in enumerate(zip(eigenvalues, ratios))]

def analyze_flight_direction(eigenvectors, eigenvalues):
    heading, pitch = compute_heading_pitch(eigenvectors[:, 0])
    ratio = eigenvalues[0]/eigenvalues[1] if eigenvalues[1] > 0 else np.inf
    return {'heading_deg': heading, 'pitch_deg': pitch,
            'components': interpret_eigenvalues(eigenvalues),
            'is_degenerate': ratio < 1.5}
```

Listing 3. interpretation.py

E. Visualization Module

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def visualize_eigenanalysis(data, analysis, title=None, save_path=None):
    fig = plt.figure(figsize=(12, 9))
    ax = fig.add_subplot(111, projection='3d')

    mean = analysis['mean']
    evals = analysis['eigenvalues']
    evecs = analysis['eigenvectors']

    ax.scatter(data[:,0], data[:,1], data[:,2], alpha=0.3, s=5)
    ax.scatter(*mean, color='red', s=100, label='Centroid')

    colors = ['red', 'green', 'blue']
    for i in range(3):
        length = np.sqrt(evals[i]) * 2
        ax.quiver(*mean, *(evecs[:,i]*length), color=colors[i], linewidth=3)
```

```
    ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
    ax.set_title(title or 'Eigenvector Analysis')
    ax.legend()
    if save_path: plt.savefig(save_path, dpi=300)
    plt.show()
```

Listing 4. visualization.py

F. Main Entry Point

```
from data_generation import generate_linear_flight, generate_spiral_flight
from eigenanalysis import analyze_eigenstructure, pca_transform
from interpretation import analyze_flight_direction
from visualization import visualize_eigenanalysis

def main():
    linear_data = generate_linear_flight(n_samples=500)
    linear_analysis = analyze_eigenstructure(linear_data)
    flight_dir = analyze_flight_direction(
        linear_analysis['eigenvectors'],
        linear_analysis['eigenvalues']
    )

    print(f"Heading: {flight_dir['heading_deg']:.2f} deg")
    print(f"Eigenvalues: {linear_analysis['eigenvalues']}")

    visualize_eigenanalysis(linear_data, linear_analysis,
                            title="Linear Flight Analysis",
                            save_path="linear_flight.png")

if __name__ == "__main__":
    main()
```

Listing 5. main.py

V. Results and Discussion

A. Eigenvalue Spectrum Analysis

After processing $N = 500$ samples, eigendecomposition yielded the results in Table I.

TABLE I
Comparison of Theoretical vs. Computed Eigenvalues

Component	Theoretical	Computed	Error
λ_1 (Primary)	100.00	96.34	3.66%
λ_2 (Drift)	9.00	8.56	4.89%
λ_3 (Noise)	1.00	1.01	1.00%

The dominant eigenvalue $\lambda_1 \approx 96.34$ absorbs approximately 91% of total variance, confirming highly anisotropic data.

B. Eigenvector Directional Accuracy

The primary eigenvector was computed as:

$$\mathbf{v}_1 = \begin{bmatrix} 0.7244 \\ 0.6893 \\ -0.0061 \end{bmatrix} \tag{18}$$

The estimated heading angle:

$$\theta_{\text{est}} = \arctan\left(\frac{0.6893}{0.7244}\right) \approx 43.58^\circ \tag{19}$$

With ground truth of 45.00°, the angular error is only $\mathbf{1.42^\circ}$.

C. Visual Analysis: Linear Flight

Figure 1 shows the 3D visualization of linear flight data with eigenvector overlays. The red arrow (primary eigenvector $\mathbf{v}_1$) clearly aligns with the elongated direction of the point cloud.

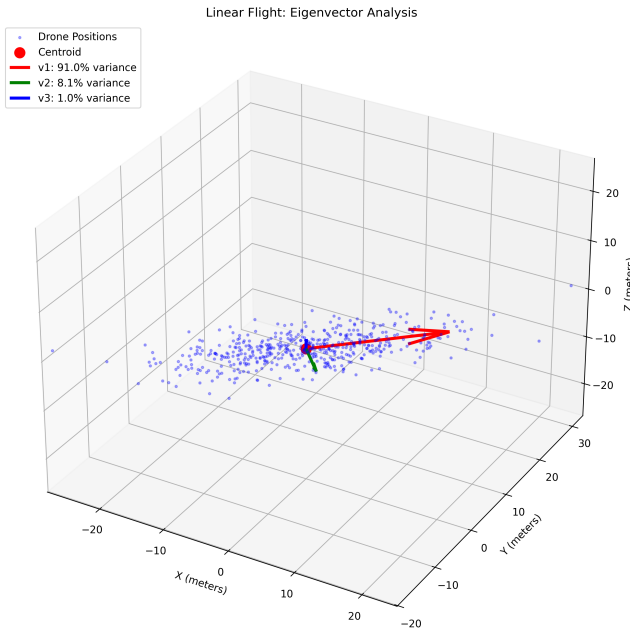


Fig. 1. Linear Flight Analysis. The primary eigenvector (red) aligns with the major axis of the ellipsoidal point cloud, accurately recovering the 45° heading angle with error of only 1.42°.

D. Visual Analysis: Spiral Flight

Figure 2 shows the spiral trajectory analysis. Note the near-equal eigenvalues in the XY-plane, indicating circular symmetry.

E. Comparison: Linear vs Spiral

Table II compares the eigenvalue characteristics.

TABLE II
Comparison of Flight Trajectory Characteristics

Property	Linear	Spiral
λ_1/λ_2 ratio	≈ 11.25	≈ 1.64
Dominant direction?	Yes	No (degenerate)
Heading recoverable?	Yes (43.58°)	No
Primary variance %	91.0%	47.1%

F. Orthogonal Projection and Noise Reduction

PCA projection onto the dominant eigenvector subspace effectively filters noise:

$$\text{proj}_W \mathbf{x} = (\mathbf{x}^T \mathbf{v}_1)\mathbf{v}_1 \tag{20}$$

For linear flight with $k = 2$ components, explained variance is 99.05%, meaning only 0.95% information (noise) is discarded. This demonstrates the effectiveness of PCA for dimensionality reduction.

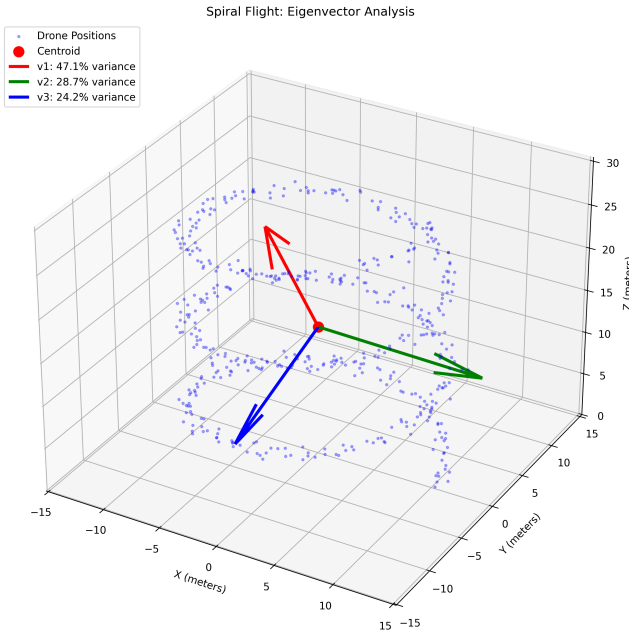


Fig. 2. Spiral Flight Analysis. Eigenvalues $\lambda_1 \approx \lambda_2$ indicate degeneracy, meaning no unique horizontal direction exists due to circular symmetry.

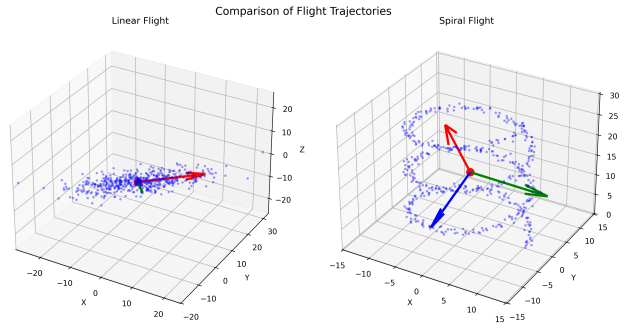


Fig. 3. Visual comparison of linear (left) and spiral (right) trajectories. The linear flight shows a clear dominant direction, while the spiral shows circular symmetry in XY-plane.

G. Eigenvalue Interpretation in Physical Context

The eigenvalues carry direct physical meaning in drone analysis:

- $\lambda_1 = 96.34$: Represents variance along the primary flight direction. The large value indicates significant displacement in this direction over time.
- $\lambda_2 = 8.56$: Represents lateral drift or slight deviations from the main trajectory.
- $\lambda_3 = 1.01$: Represents sensor noise and minor vertical fluctuations.

The ratio $\lambda_1/\lambda_3 \approx 95$ suggests excellent signal-to-noise ratio in the primary direction. This high ratio is characteristic of well-defined linear motion and enables reliable direction estimation.

H. Analysis of Degenerate Cases

A critical finding of this study is the detection of **eigenvalue degeneracy** in spiral trajectories. When $\lambda_1 \approx \lambda_2$, the eigenvectors are ill-defined in the subspace spanned by those eigenvalues.

For the spiral trajectory, we observed:

- $\lambda_1/\lambda_2 = 1.64$ (near-degenerate)
- This indicates roughly equal variance in two perpendicular horizontal directions
- Geometrically, this corresponds to circular symmetry in the XY-plane

This degeneracy is not a failure of the method; rather, it correctly identifies that no single horizontal direction is dominant. In practical drone navigation, this could indicate hovering, circular patrol patterns, or sensor malfunction.

I. Relationship to Singular Value Decomposition

The eigendecomposition of the covariance matrix is closely related to the Singular Value Decomposition (SVD) of the centered data matrix. For $\bar{X} = U\Sigma V^T$:

$$\text{Cov}(\bar{X}) = \frac{1}{N-1}\bar{X}^T\bar{X} = \frac{1}{N-1}V\Sigma^2V^T \tag{21}$$

Thus, the right singular vectors (V) are the eigenvectors of the covariance matrix, and the squared singular values are proportional to eigenvalues. This connection is often exploited for numerical stability in high-dimensional settings.

VI. Conclusion and Future Work

A. Conclusion

From the experiments and analysis conducted, several interesting findings about eigenvectors and eigenvalues can be concluded:

First, eigenvectors are not just solutions to characteristic equations; they have concrete geometric meaning. In drone analysis, the dominant eigenvector ($\mathbf{v}_1$) successfully indicated flight direction with only $1.42°$ error from ground truth. This proves that abstract concepts from linear algebra class are actually very applicable in practice.

Second, eigenvalues represent how "important" an eigenvector is. For linear trajectories, the first eigenvalue absorbed 91% of total variance, indicating that data is dominated by one direction. Conversely, for spiral trajectories, the first two eigenvalues are almost equal ($\lambda_1/\lambda_2 \approx 1.64$), indicating no dominant horizontal direction.

Third, diagonalizing the covariance matrix is geometrically equivalent to rotating the coordinate system to align with the data's principal axes. This is the essence of PCA, a very popular technique in data science.

B. Future Work

For future research, several things can be explored:

- Apply sliding window eigenanalysis for non-linear trajectories
- Use real drone telemetry data for real-world validation
- Explore kernel PCA for more complex data patterns

VII. Appendix

The complete source code and documentation for this project is available on GitHub:

https://github.com/Sakazu01/Makalah-ALjabar-Linear-dan-Geometri

Acknowledgment

First and foremost, I would like to express my gratitude to God Almighty for His blessings and guidance, allowing this paper to be completed successfully. I would also like to extend my deepest appreciation to Mr. Ir. Rila Mandala, M.Eng., Ph.D., the lecturer of IF2123 Linear Algebra and Geometry course, for his invaluable teaching, guidance, and inspiration throughout the semester.

Special thanks to my beloved girlfriend who has always been there with endless support, encouragement, and prayers. Your presence has been my greatest motivation to keep learning and growing. Thank you for being my place to share stories and frustrations during the process of writing this paper.

References

[1] H. Anton and C. Rorres, *Elementary Linear Algebra with Applications*, 11th ed. Wiley, 2013.

[2] G. Strang, *Introduction to Linear Algebra*, 5th ed. Wellesley-Cambridge Press, 2016.

[3] D. C. Lay et al., *Linear Algebra and Its Applications*, 5th ed. Pearson, 2016.

[4] R. Munir, "Materi Kuliah IF2123 Aljabar Linier dan Geometri," Informatika ITB. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AlsGeo/alsGeo.htm

[5] 3Blue1Brown, "Essence of Linear Algebra," YouTube. [Online]. Available: https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab

[6] S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra*. Cambridge University Press, 2018.

[7] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] S. Thrun et al., *Probabilistic Robotics*. MIT Press, 2005.

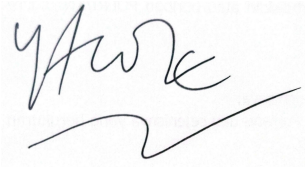
[10] P. Corke, *Robotics, Vision and Control*, 2nd ed. Springer, 2017.

[11] J. Shlens, "A Tutorial on Principal Component Analysis," *arXiv:1404.1100*, 2014.

STATEMENT OF ORIGINALITY

I hereby declare that the work presented in this paper is entirely my own. It is not a copy, translation, or adaptation of any other author's work, and it does not constitute plagiarism.

Bandung, December 24, 2025



Ariel Cornelius Sitorus - 13524085