Penerapan SVD dan Machine Learning untuk Sistem Pengenalan Suara pada Game Navigasi 2D Sederhana

Valentino Daniel Kusumo - 13524104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524104@itb.ac.id, danielkusumo550@gmail.com

***Abstrak*—*Machine learning* adalah cabang dari *Artificial Inteligence* yang memungkinkan program untuk belajar dari data, mengidentifikasi pola, dan membuat keputusan dengan sendirinya tanpa arahan dari penggunanya. Sistem Pengenalan Suara atau *Voice recognition* adalah teknologi yang memungkinkan program untuk memproses ucapan manusia menjadi perintah yang dapat dimengerti oleh mesin. Makalah ini berfokus pada pengembangan *game* navigasi 2 dimensi yang pergerakannya dikontrol melalui perintah suara. Dalam implementasinya, sinyal audio diubah menjadi representasi frekuensi menggunakan mel-spektrogram, kemudian direduksi menggunakan teknik *Singular Value Decomposition* (SVD) untuk mengekstraksi dan mereduksi dimensi fitur suara. Fitur yang telah direduksi kemudian diklasifikasikan menggunakan algoritma *machine learning*.**

Kata kunci*—Sistem pengenalan suara, ekstraksi fitur, SVD, mel-spektogram, *machine learning

I. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah mendorong kemunculan sistem interaksi manusia dengan mesin yang semakin canggih. Salah satu bidang yang berkembang pesat adalah sistem pengenalan suara (*voice recognition*), yaitu teknologi yang memungkinkan program untuk untuk memahami dan memproses ucapan manusia menjadi perintah yang dapat dipahami oleh mesin. Pengenalan suara kini banyak diterapkan dalam berbagai bidang, mulai dari *virtual assistant*, *smart home technology*, hingga aplikasi edukasi dan hiburan.

Sinyal suara memiliki karakteristik yang cukup kompleks sehingga membutuhkan representasi fitur yang tepat agar informasi penting dapat terekstraksi dengan baik. Teknik seperti **mel-spektogram** telah banyak digunakan dalam mereprsentasikan sinyal suara karena mampu mengekstraksi informasi yang relevan. Namun, data yang dihasilkan memiliki ukuran yang cukup besar sehingga dapat mempengaruhi efektivitas *machine learning* dalam proses klasifikasi, terutama pada *dataset* audio yang banyak dan besar.

Menghadapi masalah tersebut, teknik *singular value decomposition* digunakan untuk mereduksi dimensi dan mengekstraksi fitur utama suara. Dengan menggunakan SVD, *machine learning* dapat melakukan klasifikasi dengan lebih efisien tanpa menghilangkan informasi penting yang dibutuhkan untuk klasifikasi. Dengan demikian, sistem pengenalan suara dapat bekerja secara lebih akurat, cepat, dan stabil, sehingga berpotensi untuk dikembangkan dan diterapkan pada berbagai aplikasi di dunia nyata.

II. LANDASAN TEORI

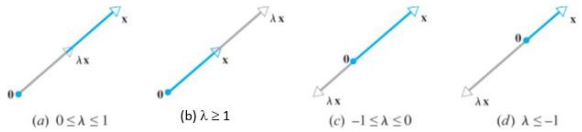
A. Nilai Eigen & Vektor Eigen

Jika A adalah matriks n x n, vektor tidak nol x di R^n disebut **vektor eigen** dari A jika A**x** sama dengan perkalian suatu skalar λ dengan x, yaitu

$$A\mathbf{x} = \lambda x$$

Skalar λ disebut **nilai eigen** dari *A*, dan *x* dinamakan vektor eigen yang berkoresponden dengan dengan λ.

Vektor eigen *x* menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks *n x n* menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri. Dengan kata lain, operasi $A\mathbf{x} = \lambda x$ menyebabkan vektor *x* menyusut atau memanjang dengan faktor λ dengan arah yang sama jika λ positif dan arah berkebalikan jika λ negatif.



Gambar 1. Kelipatan Vektor
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf

Diberikan sebuah matriks A berukuran *n x n*. Vektor eigen dan dan nilai eigen dari matriks A dihitung sebagai berikut:

$$A\mathbf{x} = \lambda x$$
$$IA\mathbf{x} = \lambda Ix$$
$$A\mathbf{x} = \lambda Ix$$
$$(\lambda I - A)\, x = 0$$

Persamaan tersebut merupakan sistem persamaan linier homogen, Solusi trivial $x = 0$ selalu ada, tetapi agar sistem memiliki solusi tidak nol, maka harus dipenuhi syarat

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

Persamaan det $(\lambda I - A) = 0$ disebut **persamaan karakteristik** dari matriks A, dan akar-akar persamaan tersebut, yaitu λ, dinamakan **akar-akar karateristik** atau **nilai-nilai eigen**.

Langkah selanjutnya adalah menentukan **vektor eigen** yang bersesuaian dengan masing-masing nilai eigen tersebut.

1. Substitusikan setiap nilai eigen λ ke dalam persamaan
$$(\lambda I - A) x = 0$$
2. Persamaan tersebut membentuk sebuah sistem **persamaan linier homogen**.
3. Sistem persamaan linier tersebut kemudian diselesaikan untuk mencari vektor *x* yang tidak sama dengan nol.
4. Solusi tidak nol dari sistem persamaan tersebut merupakan vektor eigen yang bersesuaian dengan nilai eigen λ.
5. Jika terdapat lebih dari satu solusi bebas, maka himpunan solusi tersebut membentuk ruang eigen (eigenspace) dari nilai eigen tersebut.

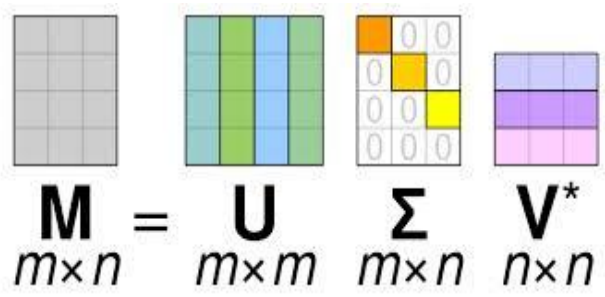
B. Singular Value Decomposition

Singular Value Decomposition adalah metode yang mendekomposisi sebuah matriks *A* berukuran *m x n* menjadi hasil perkalian 3 matriks, yaitu

$$A = U \Sigma V^{T}$$

dengan:

- *U* adalah matriks ortogonal berukuran *m x n* yang kolom – kolomnya merupakan **vektor singular kiri**
- Σ adalah matriks diagonal berukuran *m x n* yang berisi **nilai singular**
- V^T adalah transpose dari matriks ortogonal *V* berukuran *n x n* yang kolom-kolomnya merupakan **vektor singular kanan**.



Gambar 2. *Singular Value Decomposition* (SVD)
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-21-Singular-value-decomposition-Bagian1-2025.pdf

Nilai singular pada matriks Σ diperoleh dari akar kuadrat nilai eigen matriks A^TA atau AA^T sehingga konsep nilai eigen dan vektor eigen memiliki peran penting dalam pembentukan SVD.

Salah satu cara untuk melakukan dekomposisi SVD adalah sebagai berikut:

1. **Vektor singular kiri** diperoleh dengan menghitung nilai – nilai eigen dari matriks AA^T. Banyaknya nilai eigen yang tidak bernilai nol dari matriks tersebut menunjukkan **rank** dari matriks *A*, yang dinotasikan sebagai *k*.
2. Menentukan vektor – vektor eigen $u_1,u_2,..,u_m$ yang berkoresponden dengan nilai – nilai eigen dari matriks AA^T. Vektor tersebut kemudian dinormalisasi untuk mendapatkan *U*.
3. **Vektor singular kanan** diperoleh dengan menghitung nilai – nilai eigen dari matriks A^TA. Kemudian, nilai – nilai singular matriks *A* dari nilai – nilai eigen tersebut.
4. Menentukan vektor – vektor eigen $v_1,v_2,..,v_n$ yang berkoresponden dengan nilai – nilai eigen dari matriks A^TA. Vektor tersebut kemudian dinormalisasi untuk mendapatkan *V*.
5. Matriks diagonal Σ berukuran *m x n* kemudian dibentuk dengan elemen-elemen diagonal utama berupa nilai-nilai singular matriks *A* yang disusun dari nilai terbesar hingga terkecil. Nilai singular tersebut merupakan akar kuadrat dari nilai-nilai eigen tak nol dari matriks.
6. Dengan demikian, matriks *A* dapat dinyatakan sebagai hasil perkalian $A = U \Sigma V^{T}$.

C. Machine Learning

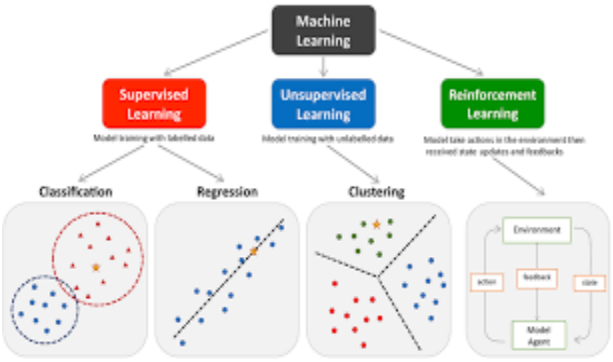
Machine Learning adalah salah cabang dari kecerdasan buatan (*Artificial Inteligence*) yang memungkinan program untuk melakukan analisis terhadap data tanpa harus diprogram secara eksplisit oleh pengguna. *Machine Learning* dapat memproses data dalam jumlah yang besar, mengidentifikasi pola, dan memprediksi hubungan antara data yang sebelumnya tidak diketahui. Tidak hanya itu, tetapi *machine learning* juga dapat melakukan klasifikasi dan prediksi pada suara, dokumen, gambar, angka, dan tipe data lainnya.

Secara umum, metode *machine learning* dapat diklasifikasikan menjadi beberapa jenis, antara lain **supervised learning**, **unsupervised learning**, **semi-supervised learning**, dan **reinforcement learning**.

Pada *supervised learning*, proses pembelajaran dilakukan menggunakan data latih yang telah memiliki label, sedangkan pada *unsupervised learning*, sistem belajar dari data tanpa label untuk menemukan pola atau struktur tertentu.

Semi-supervised learning merupakan pendekatan yang mengombinasikan data berlabel dan data tidak berlabel dalam proses pembelajaran. Metode ini digunakan ketika jumlah data berlabel terbatas, tetapi masih tersedia banyak data tanpa label yang dapat dimanfaatkan untuk meningkatkan kinerja model.

Adapun *reinforcement learning*, adalah metode pembelajaran di mana agen belajar melalui interaksi dengan lingkungan. Agen tersebut akan menerima umpan balik berupa *reward* atau *penalty* berdasarkan tindakan yang dilakukan, sehingga tujuan utama dari reinforcement learning adalah memaksimalkan total *reward* yang diperoleh.



Gambar 3. Tipe *Machine Learning* (*minus semi-supervised*)
Sumber: https://www.researchgate.net/figure/The-main-types-of-machine-learning-Main-approaches-include-classification-and-regression_fig1_354960266

III. IMPLEMENTASI

A. Perekaman Data Suara

Tahap awal dalam implementasi sistem pengenalan adalah perekaman data suara yang digunakan sebagai *dataset* pengenalan perintah pada *game* navigasi 2D. Perekaman dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan *library sounddevice* untuk merekam suara dari mikrofon dan *scipy.io.wavfile* untuk menyimpan hasil rekaman dalam format *.wav*.

Setiap sampel suara direkam dengan frekuensi *sampling* sebesar 16.000 Hz dan durasi 1,5 detik. Data suara disimpan dalam *folder* terpisah berdasarkan label perintah, yaitu *up*, *down*, *left*, dan *right*, sehingga memudahkan proses pelatihan model *machine learning*. Proses perekaman dilakukan secara interaktif, di mana pengguna menekan tombol *Enter* untuk memulai perekaman setiap sampel suara.

Hasil dari tahap ini adalah sekumpulan berkas audio yang telah terorganisasi berdasarkan label perintah suara. Dataset ini selanjutnya digunakan pada tahap **normalisasi audio**, **ekstraksi fitur menggunakan SVD**, dan **pelatihan model machine learning** pada tahap berikutnya.

```
SAMPLE_RATE = 16000
DURATION = 1.5

def record_dataset(label, count=10):
    base_dir = Path(__file__).resolve().parent.parent / "data"
    label_dir = base_dir / label
    label_dir.mkdir(parents=True, exist_ok=True)

    for i in range(count):
        input(f"[{label}] Sample {i+1}/{count} - tekan ENTER untuk rekam ... ")
        print("Merekam ... ")

        audio = sd.rec(int(SAMPLE_RATE * DURATION), samplerate=SAMPLE_RATE, channels=1, dtype='float32')
        sd.wait()

        audio_int16 = np.int16(audio * 32767)
        filename = label_dir / f"{label}_{i+11}.wav"
        write(filename, SAMPLE_RATE, audio_int16)

        print(f"Disimpan ke {filename}\n")
```

Gambar 4. Kode Perekaman Data Suara
Sumber: Dokumen Penulis

B. Normalisasi Audio

Setelah proses perekaman, data suara yang diperoleh memiliki tingkat amplitudo yang bervariasi sehingga perlu dilakukan normalisasi audio untuk menyamakan tingkat energi sinyal suara agar perbedaan volume tidak memengaruhi proses pelatihan model *machine learning*.

Normalisasi dilakukan dengan menyesuaikan nilai **Root Mean Square (RMS)** setiap sinyal audio ke nilai target tertentu. Audio dengan amplitudo yang terlalu kecil diabaikan karena berpotensi mengandung informasi yang tidak representatif. Selanjutnya, sinyal audio diskalakan menggunakan faktor penguatan sehingga nilai RMS mendekati nilai target yang telah ditentukan. Untuk mencegah distorsi, amplitudo sinyal dibatasi pada rentang –1 hingga 1. Seluruh data suara dinormalisasi menggunakan frekuensi *sampling* yang sama, yaitu 16.000 Hz dan hasil normalisasi disimpan kembali dalam format *.wav* dengan tipe data *int16*.

```
SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))
DATA_DIR = os.path.join(SCRIPT_DIR, "data")
TARGET_RMS = 0.07

def normalize_audio(audio_path, target_rms=TARGET_RMS):
    """Normalize audio ke target RMS"""
    y, sr = librosa.load(audio_path, sr=16000)

    current_rms = np.sqrt(np.mean(y**2))

    if current_rms < 0.001:
        print("Audio terlalu lemah, skip")
        return False

    gain = target_rms / current_rms
    y_normalized = y * gain
    y_normalized = np.clip(y_normalized, -1.0, 1.0)

    # Save
    write(audio_path, sr, (y_normalized * 32767).astype(np.int16))

    new_rms = np.sqrt(np.mean(y_normalized**2))
    print(f"✓ {os.path.basename(audio_path)}: {current_rms:.4f} → {new_rms:.4f}")

    return True
```

Gambar 5. Kode Normalisasi Audio
Sumber: Dokumen Penulis

C. Ekstraksi Fitur Suara Menggunakan SVD

Audio yang telah dinormalisasi terlebih dahulu diubah ke dalam bentuk **Mel-spectrogram**, yang merepresentasikan energi sinyal suara pada skala frekuensi Mel sehingga lebih sesuai dengan karakteristik pendengaran manusia. Matriks Mel-spectrogram tersebut kemudian didekomposisi menggunakan SVD untuk memperoleh nilai-nilai singular yang merepresentasikan informasi utama dari sinyal suara. Sejumlah nilai singular terbesar dipilih sebagai fitur karena mengandung komponen energi dominan, sementara komponen dengan nilai kecil dianggap sebagai noise atau informasi yang kurang relevan.

Selain SVD, ditambahkan pula beberapa fitur lain, seperti **Mel-Frequency Cepstral Coefficients (MFCC)**, **Zero Crossing Rate (ZCR)**, dan **Spectral Centroid** untuk memperbanyak representasi karakteristik suara. Seluruh fitur yang diperoleh kemudian digabungkan menjadi satu vektor fitur dan dinormalisasi agar memiliki skala yang seragam sebelum digunakan pada tahap pelatihan model *machine learning*.

```
DATA_DIR = "data"
OUTPUT_X = "features/X.npy"
OUTPUT_Y = "features/y.npy"

N_MELS = 64          # jumlah filter mel
K_COMPONENTS = 40    # lebih banyak komponen untuk fitur lebih kaya

def extract_svd_features(audio_path):
    """Load audio -> Mel spectrogram -> SVD + MFCC -> fitur."""
    y, sr = librosa.load(audio_path, sr=16000)

    # Mel-spectrogram + SVD
    S = librosa.feature.melspectrogram(
        y=y,
        sr=sr,
        n_mels=N_MELS
    )
    S_db = librosa.power_to_db(S, ref=np.max)
    U, s, Vt = np.linalg.svd(S_db, full_matrices=False)
    svd_features = s[:K_COMPONENTS]

    # MFCC features (tambahan untuk meningkatkan akurasi)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    mfcc_mean = np.mean(mfcc, axis=1)
    mfcc_std = np.std(mfcc, axis=1)

    # Zero Crossing Rate (karakteristik suara)
    zcr = np.mean(librosa.feature.zero_crossing_rate(y))

    # Spectral Centroid (brightness suara)
    spectral_centroid = np.mean(librosa.feature.spectral_centroid(y=y,
sr=sr))
    # Gabungkan semua fitur
    feature = np.concatenate([
        svd_features,      # 40 features
        mfcc_mean,         # 13 features
        mfcc_std,          # 13 features
        [zcr],             # 1 feature
        [spectral_centroid] # 1 feature
    ])  # Total 68 features

    # Normalisasi fitur
    feature = feature / (np.linalg.norm(feature) + 1e-8)

    return feature
```

Gambar 6. Kode Ekstraksi Fitur Suara
Sumber: Dokumen Penulis

Penjelasan Singkat Kode :

- **Melspectogram**
  Mengubah sinyal audio menjadi matriks waktu–frekuensi berbasis skala Mel.
- **np.linalg.svd(S_db, full_matrices=False)**
  Melakukan dekomposisi SVD pada Mel-spectrogram. Nilai singular (s) merepresentasikan kekuatan komponen utama sinyal.
- **s[:K_COMPONENTS]**
  Mengambil sejumlah nilai singular terbesar sebagai fitur utama hasil SVD.
- **MFCC (*Mel-Frequency Cepstral Coefficients*)**
  Menangkap karakteristik spektral suara yang sering digunakan pada sistem pengenalan suara.
- **ZCR dan Spectral Centroid**
  Digunakan untuk merepresentasikan karakteristik temporal dan kecerahan suara.

D. Training

Pada tahap ini, dilakukan proses *training* model untuk mengklasifikasikan perintah suara yang ada, yaitu *up*, *down*, *left*, dan *right*. Data suara yang telah melalui tahap normalisasi dan ekstraksi fitur menggunakan SVD selanjutnya digunakan sebagai data latih.

Setiap audio diproses untuk menghasilkan vektor fitur, kemudian dikumpulkan dalam dataset beserta label perintahnya. Sebelum proses *training*, fitur – fitur tersebut dinormalisasi terlebih dahulu menggunakan **StandardScaler** agar memiliki distribusi yang seragam. Normalisasi ini bertujuan untuk meningkatkan efektivitas algoritma klasifikasi yang sensitif terhadap perbedaan skala fitur.

Model klasifikasi yang digunakan adalah **Support Vector Machine (SVM)** dengan kernel **Radial Basis Function (RBF)**. SVM dipilih karena memiliki performa yang baik dalam menangani data besar dan mampu mengambil keputusan yang optimal. Setelah proses *training* selesai, model yang telah terbentuk disimpan ke dalam *file .pkl* agar dapat digunakan kembali pada tahap implementasi akhirnya.

```
def train():
    print("Loading dataset...")
    X, y = load_dataset() # fungsi untuk load dataset
    print(f"Dataset: {X.shape[0]} sampel, {X.shape[1]} features\n")

    print("Training SVM...")
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    model = SVC(kernel='rbf', C=10, gamma='scale',
probability=True)
    model.fit(X_scaled, y)

    model_data = {'classifier': model, 'scaler': scaler}

    os.makedirs("model", exist_ok=True)
    with open(MODEL_PATH, "wb") as f:
        pickle.dump(model_data, f)

    print("Model telah disimpan")
```

Gambar 7. Kode Train Model
Sumber: Dokumen Penulis

E. Implementasi Game Navigasi 2D Sederhana

Pada tahap ini, dilakukan implementasi sistem pengenalan suara (*Voice Recognition*) secara *real time* pada game navigasi 2D sederhana ini. Sistem bekerja dengan merekam suara pengguna melalui mikrofon, kemudian melakukan ekstraksi fitur menggunakan metode yang sama seperti pada tahap sebelumnya, yaitu fitur berbasis SVD yang dikombinasikan dengan MFCC dan fitur lainnya.

Model *machine learning* yang telah *ditrain* sebelumnya di *load* dari file *voice_model.pkl* dan digunakan untuk memprediksi perintah suara yang diucapkan pengguna. Hasil prediksi berupa salah satu perintah navigasi, yaitu *up*, *down*, *left*, atau *right*. Selain hasil prediksi, sistem juga dapat menampilkan probabilitas klasifikasi untuk setiap kelas sebagai indikator tingkat keakuratan model.

Perintah suara yang dikenali kemudian digunakan sebagai input untuk mengendalikan pergerakan pada game navigasi 2D. Setiap perintah suara akan mengubah posisi objek sesuai arah yang dikenali oleh sistem. Proses ini dilakukan secara berulang hingga pengguna menghentikan program (*menekan tombol 'q'*).

```
GRID_SIZE = 10

class GameSimulator:
    def __init__(self):
        self.x = 5
        self.y = 5
        self.last_move = ""

    def print_grid(self):
        # Header angka
        header = "     "
        for i in range(1, GRID_SIZE + 1):
            header += f" {i}  "
        print(header)

        # Garis
        print("    +" + "---+" * GRID_SIZE)

        for row in range(1, GRID_SIZE + 1):
            if row == GRID_SIZE:
                line = f" {row} |"
            else:
                line = f" {row}  |"
            for col in range(1, GRID_SIZE + 1):
                if col == self.x and row ==
self.y:
                    line += " O |"  # Player
                else:
                    line += "   |"
            print(line)
            print("    +" + "---+" * GRID_SIZE)

        print(f"\nPosisi: ({self.x}, {self.y})")
        if self.last_move:
            print(f"Move: {self.last_move}")

    def move(self, direction):
        if direction == "up":
            self.y = max(0, self.y - 1)
        elif direction == "down":
            self.y = min(GRID_SIZE - 1, self.y +
1)
        elif direction == "left":
            self.x = max(0, self.x - 1)
        elif direction == "right":
            self.x = min(GRID_SIZE - 1, self.x +
1)
        self.last_move = direction.upper()
```

Gambar 8. Kode Program Utama
Sumber: Dokumen Penulis

IV. HASIL DAN ANALISIS

Berdasarkan hasil pengujian sistem, tahap normalisasi audio berhasil menyeragamkan tingkat energi sinyal suara pada seluruh dataset (*hasil yang ditampilkan menunjukkan tidak ada perubahan karena dataset ini telah dinormalisasi sebelumnya sehingga tidak ada perubahan yang terlihat*). Proses ini membuat perbedaan volume antar rekaman menjadi lebih konsisten sehingga data suara yang digunakan lebih stabil untuk tahap ekstraksi fitur dan *training* model. Dengan normalisasi ini, pengaruh variasi amplitudo suara terhadap kinerja sistem dapat diminimalkan.

```
--- NORMALISASI AUDIO ---

Normalisasi semua audio? (ya/tidak): ya
--- NORMALISASI DATASET AUDIO ---
Target RMS: 0.07

--------------------------------------------------
DOWN
--------------------------------------------------
✓ down_1.wav: 0.0700 -> 0.0700
✓ down_10.wav: 0.0700 -> 0.0700
✓ down_11.wav: 0.0700 -> 0.0700
✓ down_12.wav: 0.0700 -> 0.0700
✓ down_13.wav: 0.0700 -> 0.0700
✓ down_14.wav: 0.0700 -> 0.0700
✓ down_15.wav: 0.0700 -> 0.0700
✓ down_16.wav: 0.0700 -> 0.0700
✓ down_17.wav: 0.0700 -> 0.0700
✓ down_18.wav: 0.0700 -> 0.0700
✓ down_19.wav: 0.0700 -> 0.0700
✓ down_2.wav: 0.0700 -> 0.0700
✓ down_20.wav: 0.0700 -> 0.0700
✓ down_3.wav: 0.0700 -> 0.0700
✓ down_4.wav: 0.0700 -> 0.0700
✓ down_5.wav: 0.0700 -> 0.0700
✓ down_6.wav: 0.0700 -> 0.0700
✓ down_7.wav: 0.0700 -> 0.0700
✓ down_8.wav: 0.0700 -> 0.0700
✓ down_9.wav: 0.0700 -> 0.0700
JUMLAH FILE : 20 files
```

Gambar 9. Normalisasi Audio
Sumber: Dokumen Penulis

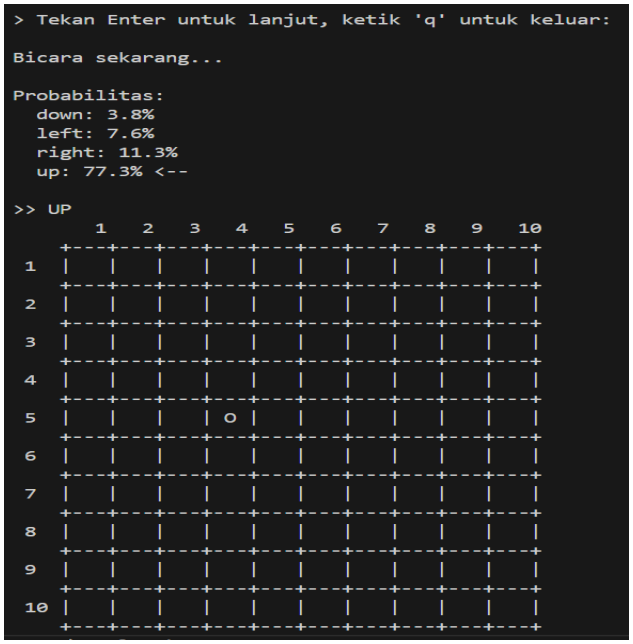
Pada tahap *training* model *machine learning*, sistem berhasil membangun model klasifikasi perintah suara menggunakan fitur hasil ekstraksi SVD yang dikombinasikan dengan fitur lainnya. Proses *training* berjalan dengan baik tanpa ada kendala apa pun.

```
Loading dataset...
Memproses down: 20 files
Memproses left: 20 files
Memproses right: 20 files
Memproses up: 20 files
Dataset: 80 sampel, 68 features

Training SVM...
Model telah disimpan
```

Gambar 10. Train Model
Sumber: Dokumen Penulis

Hasil pengujian pada program utama menunjukkan bahwa sistem pengenalan suara (*Voice recognition*) telah berhasil diimplementasikan pada game navigasi 2D sederhana dan dapat dijalankan secara *real-time*. Sistem berhasil menerjemahkan perintah suara, tetapi masih memiliki kekurangan dalam hal keakuratan.



Gambar 11. Program Utama
Sumber: Dokumen Penulis

Untuk mengevaluasi efektivitas program, dilakukan 10 kali pengujian sistem pengenalan suara yang hasilnya disajikan dalam bentuk tabel pada bagian berikut

Nomor percobaan	Perintah Suara	Hasil Prediksi	Sesuai / Tidak
1	*Right*	*Right*	Sesuai
2	*Left*	*Left*	Sesuai
3	*Down*	*Left*	Tidak
4	*Up*	*Up*	Sesuai
5	*Right*	*Right*	Sesuai
6	*Left*	*Left*	Sesuai
7	*Right*	*Right*	Sesuai
8	*Up*	*Up*	Sesuai
9	*Left*	*Left*	Sesuai
10	*Down*	*Left*	Tidak

Tabel 1. Tabel Pengujian Sistem Pengenalan Suara 10 Kali
Sumber : Dokumen Penulis

Berdasarkan hasil pengujian yang dilakukan, sistem pengenalan suara mampu mengenali perintah suara dengan tingkat keberhasilan sebesar 8 dari 10 kali percobaan atau sebesar 80%. Hasil ini menunjukkan bahwa sistem telah mampu bekerja dengan baik. Namun, masih terdapat beberapa kesalahan pada sebagian percobaan, yang mungkin dipengaruhi oleh berbagai faktor seperti perbedaan intonasi suara, tingkat kebisingan lingkungan, serta keterbatasan jumlah dataset.

V. Kesimpulan Dan Saran

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem pengenalan suara berbasis Singular Value Decomposition (SVD) dan metode *machine learning* berhasil diimplementasikan pada game navigasi 2D sederhana. Sistem mampu mengenali perintah suara berupa arah gerak dan menerjemahkannya menjadi pergerakan objek.

Meskipun begitu, percobaan yang telah dilakukan sebelumnya menunjukkan masih adanya kekurangan dalam mengenali perintah suara yang diberikan. Oleh karena itu, sistem masih perlu pengembangan lebih lanjut agar dapat memiliki performa yang lebih baik, khususnya dalam mendeteksi perintah suara, seperti menambahkan jumlah *dataset*, menambahkan metode pra pemrosesan untuk memastikan *dataset* cukup akurat, dan mencari berbagai metode lainnya yang mungkin dapat menambahkan keakuratan pada model.

VI. Lampiran

Program Game Navigasi 2D Sederhana dapat diakses pada link berikut : https://github.com/ValentinoDan/Makalah-Algeo.git

Video penjelasan dapat diakses pada link berikut : https://youtu.be/s77e9dxw9lY?si=pSiCSdPWd2dRSCNf

VII. Ucapan Terima Kasih

Puji syukur kepada Tuhan Yang Maha Esa atas segala berkat dan rahmat-Nya sehingga makalah ini dapat diselesaikan dengan baik. Penulis juga menyampaikan terima kasih kepada dosen pengampu, teman-teman penulis yang telah memberikan dukungan, serta kedua orang tua penulis yang selalu memberikan semangat dan motivasi selama proses penyusunan makalah ini.

Referensi

[1] Amazon Web Services, Inc. 2025. *Apa itu Machine Learning?* Amazon Web Services. Diperoleh dari https://aws.amazon.com/id/what-is/machine-learning/. Diakses pada 13 Desember 2025.

[2] Ghazi, T. 2022. *Ekstraksi low-level feature audio untuk machine learning. Medium*. Diperoleh dari https://taqiyyaghazi.medium.com/ekstraksi-low-level-feature-audio-untuk-machine-learning-402bc20e9a3a. Diakses pada 14 Desember 2025.

[3] GN, Sidharth. 2025. *The RBF kernel in SVM: A complete guide*. Quark Machine Learning. Diperoleh dari https://www.quarkml.com/2022/10/the-rbf-kernel-in-svm-complete-guide.html. Diakses pada 16 Desember 2025.
[4] McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. 2015. librosa: *Audio and music signal analysis in python*. Proceedings of the 14th python in science conference (hal. 18–25). Diperoleh dari https://librosa.org/doc/. Diakses pada 14 Desember 2025.
[5] Munir, Rinaldi. 2025. Nilai Eigen dan Vektor Eigen (Bagian 1) : "Bahan Kuliah IF2123 Aljabar Linier dan Geometri." Bandung. Diakses pada 12 Desember 2025.
[6] Munir, Rinaldi. 2025. Singular Value Decomposition (SVD) (Bagian 1) : "Bahan Kuliah IF2123 Aljabar Linier dan Geometri." Bandung. Diakses pada 13 Desember 2025.
[7] Vaj, T. 2023. *MFCC vs. Mel-Spectrogram*. Medium. Diperoleh dari https://vtiya.medium.com/mfcc-vs-mel-spectrogram-8f1dc0abbc62. Diakses pada 14 Desember 2025.
[8] W3Schools. (n.d.). *Python machine learning - Getting started*. W3Schools. Diperoleh dari https://www.w3schools.com/python/python_ml_getting_started.asp. Diakses pada 16 Desember 2025.

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, 24 Desember 2025

Valentino Daniel Kusumo
13524104