

LAPORAN TUGAS BESAR 2

IF2123 Aljabar Linier dan Geometri

Image Retrieval dan Music Information Retrieval Menggunakan PCA dan Vektor



Dipersiapkan oleh:

Kelompok 49:

M. Rayhan Farrukh 13523035

Shanice Feodora Tjahjono 13523097

Andrew Tedjapratama 13523148

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

Daftar Isi	2
Bab 1: Deskripsi Masalah	3
1.1. Pemrosesan Suara dan Gambar	3
1.2. Information Retrieval	4
1.3. Album Picture Finder - Principal Component Analysis	4
1.4. Music Information Retrieval - Query by Humming	9
Bab 2: Teori Singkat	13
2.1 Sistem Temu Balik Suara (MIR)	13
2.2 Metode Ekstraksi Fitur Berdasarkan Humming	13
2.3 Image Retrieval dengan PCA	14
Bab 3: Arsitektur Website (Frontend) dan Arsitektur Program Information Retrieval (Backend)	17
3.1 Frontend	17
3.2 Backend	20
Bab 4: Eksperimen	32
Bab 5: Kesimpulan, Saran, Komentar, dan Refleksi	35
5.1 Kesimpulan	35
5.2 Saran	35
5.3 Komentar	35
5.4 Refleksi terhadap Tugas Besar	35
Bab 6: Lampiran	36
6.1 Referensi	36
6.2 Link Video	36

Bab 1: Deskripsi Masalah

1.1. Pemrosesan Suara dan Gambar

Suara selalu menjadi hal yang paling penting dalam kehidupan manusia. Manusia berbicara mengeluarkan suara dan mendengarkan suatu suara untuk diresap ke otak dan mencari informasi dari suara tersebut. Suara juga bisa dijadikan orang-orang di dunia ini sebuah media untuk membuat karya seni. Contohnya adalah alat mendeteksi lagu. Manusia bisa mendeteksi suara dengan menggunakan indera pendengar dan memberikan kesimpulan akan apa jenis suara tersebut melalui respon dari otak. Sama seperti manusia, teknologi juga bisa mendeteksi suara dan memberikan jawaban mereka melalui algoritma-algoritma yang beragam bahkan bisa melebihi kapabilitas manusia. Dengan menggunakan algoritma apapun, konsep dari pendeksi dan interpretasi suara itu bisa juga disebut dengan sistem temu balik suara atau bisa disebut juga dengan *audio retrieval system*. Banyak aplikasi yang menggunakan konsep sistem temu balik contohnya adalah Shazam.

Selain suara, manusia juga memiliki penglihatan sebagai salah satu inderanya dan bisa melihat warna dan gambar yang bermacam-macam. Teknologi komputasi juga memiliki kapabilitas yang sama dan bisa melihat gambar sama seperti kita, tetapi teknologi seperti ini juga bisa merepresentasikan gambar tersebut sebagai beragam-ragam angka yang bisa disebut juga fitur. Tahun ke tahun, *image processing* selalu menjadi fokus utama dari tugas besar 2 Algeo. Algoritma yang digunakan adalah Eigenvalue, Cosine Similarity, Euclidean Distance, dll.

Anda sudah melewati Tugas Besar 1 yaitu tentang matriks dan implementasi terhadap berbagai hal. Matriks adalah salah satu komponen yang penting dalam aplikasi aljabar vektor. Di dalam Tugas Besar 2 ini, anda diminta untuk membuat semacam aplikasi Shazam yaitu sebuah aplikasi yang meminta input lagu dan aplikasi tersebut mendeteksi apa nama dari lagu tersebut dan beberapa detail lainnya. Pada tugas besar ini, anda akan menggunakan aljabar vektor untuk mencari perbandingan antar satu audio dengan audio yang lain. Anda akan menggunakan konsep yang bernama *Music Information Retrieval* atau MIR untuk mencari dan mengidentifikasi suara berdasarkan fitur-fitur yang dimilikinya. Tidak hanya itu, anda juga akan menggunakan konsep Principal Component Analysis (PCA) untuk

mencari kumpulan audio melalui deteksi wajah berbagai orang (anggap saja mereka sebagai seorang penyanyi).

1.2. Information Retrieval

Information Retrieval adalah konsep meminta informasi dari sebuah data dengan memasukkan data tertentu. Pada tugas besar ini, anda akan berkutik dengan 2 jenis Information Retrieval. *Image Retrieval* dan *Music Information Retrieval*. *Image Retrieval* adalah konsep untuk memasukkan sebuah input gambar dan berharap mendapatkan gambar yang ada di data sesuai dengan informasi dan perhitungan yang diinginkan. Sedangkan *Music Information Retrieval* (MIR) adalah konsep untuk memasukkan sebuah input audio dan berharap mendapatkan audio yang ada di data sesuai dengan informasi dan perhitungan yang diinginkan. Pada tugas besar kali ini, kalian akan mengimplementasikan *Image Retrieval* dengan menggunakan Principal Component Analysis dan *Music Information Retrieval* dengan menggunakan humming.

1.3. Album Picture Finder - Principal Component Analysis

Setiap audio pastinya memiliki gambar albumnya sendiri masing-masing. Terkadang kita lebih mengingat suatu visual dibandingkan nama dari lagu itu sendiri. Untuk memudahkan pengguna yang hanya memiliki gambar dari suatu album, maka dibutuhkan album finder dengan menggunakan teknik Principal Component Analysis (PCA).

Sebelum implementasi PCA, jangan lupa untuk memasukkan pemetaan nama audio dengan nama gambar yang bersangkutan, gunakan file .txt atau .json untuk melakukan pemetaan audio dengan gambar yang bersangkutan. Gunakan kreativitas anda untuk memetakan audio dengan gambar tersebut.

```
mapper.txt
```

```
-----  
audio_file pic_name  
audio_1.mid pic_1.png
```

```
audio_2.mid pic_2.png
```

```
audio_3.mid pic_3.png
```

```
audio_4.mid pic_4.png
```

```
.
```

```
mapper.json
```

```
[
```

```
{
```

```
    "audio_file": audio_1.mid,
```

```
    "pic_name": pic_1.png
```

```
},
```

```
{
```

```
    "audio_file": audio_2.mid,
```

```
    "pic_name": pic_2.png
```

```
},
```

```
{
```

```
    "audio_file": audio_3.mid,
```

```
    "pic_name": pic_3.png
```

```
},
```

```
{
```

```
    "audio_file": audio_4.mid,
```

```
    "pic_name": pic_4.png
```

```
}
```

```
]
```

Principal Component Analysis (PCA) adalah teknik statistik yang digunakan untuk mereduksi dimensi data dengan tetap mempertahankan sebanyak mungkin informasi yang ada. PCA mengubah data berdimensi tinggi menjadi beberapa dimensi yang lebih kecil, disebut principal components, tanpa kehilangan esensi atau pola utama dalam data tersebut. Hasil data yang didapatkan dari PCA ini akan berupa eigenvector dan proyeksi data.

Langkah-langkah untuk melakukan pencarian gambar menggunakan PCA adalah sebagai berikut:

1. Image Processing and Loading

Lakukan pemrosesan seluruh gambar yang ada pada dataset. Ubah gambar menjadi grayscale untuk mengurangi kompleksitas gambar dan membuat fokus menjadi bagian terang dan gelap gambar. Yang berarti setiap gambar direpresentasikan dalam intensitas pixel saja tanpa informasi warna.

$$I(x,y) = 0.2989 \cdot R(x,y) + 0.5870 \cdot G(x,y) + 0.1140 \cdot B(x,y)$$

Selanjutnya, gambar akan diubah besarnya sehingga ukurannya sama untuk seluruh gambar. Ukuran seluruh gambar harus konsisten untuk membuat perhitungan semakin akurat.

Lalu ubah vektor grayscale pada gambar menjadi 1D untuk membuat dimensi vektor dapat dilakukan pemrosesan data. Jika gambar memiliki dimensi $M \times N$, maka hasilnya adalah vektor dengan panjang $M \cdot N$:

$$I = [I_1, I_2, \dots, I_{M \cdot N}]$$

2. Data Centering (Standardization)

Pada step ini lakukan standarisasi data di sekitar nilai 0. Hitung rata rata dari setiap gambar untuk suatu piksel.

$$\mu_j = \frac{1}{N} \sum_{i=1}^N x_{ij}$$

di mana:

- x_{ij} : nilai piksel ke-j pada gambar ke-i,
- N: jumlah total gambar dalam dataset.

Lalu kurangi piksel tersebut dengan rata-rata yang sudah dihitung untuk melakukan standarisasi.

$$x_{ij}' = x_{ij} - \mu_j$$

3. PCA Computation Using Singular Value Decomposition (SVD)

Lakukan perhitungan SVD pada gambar-gambar yang sudah distandarisasi. Buat matriks kovarians dari data yang sudah distandarisasi:

$$C = \frac{1}{N} X'^T X'$$

di mana:

- X' : matriks data yang sudah distandarisasi.

Lalu lakukan dekomposisi nilai singular untuk mendapatkan komponen utama:

$$\mathbf{C} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{U}^T$$

- \mathbf{U} : matriks eigenvector (komponen utama),
- $\boldsymbol{\Sigma}$: matriks eigenvalue (menunjukkan varian data di sepanjang komponen utama).

Lalu ambil n jumlah component utama teratas dari hasil SVD dan lakukan proyeksikan gambar ke komponen utama.

Pilih k-komponen utama teratas ($k \ll M \cdot N$) dan proyeksikan data:

$$\mathbf{Z} = \mathbf{X}' \mathbf{U}_k$$

di mana:

- U_k : matriks eigenvector dengan n-dimensi.

4. Similarity Computation

Lakukan pencarian kesamaan dengan mencari jarak euclidean dari tiap dataset dengan gambar query. Lalu lakukan pengurutan kecocokan dari yang paling tinggi.

Pertama-tama, representasikan gambar query dalam ruang komponen utama dengan proyeksi yang sama:

$$\mathbf{q} = (\mathbf{q}' - \mu)\mathbf{U}_k$$

Dimana:

- \mathbf{q} : Vektor proyeksi dari gambar query ke ruang komponen utama (PCA).
- \mathbf{q}' : Gambar query dalam format vektor (setelah grayscale, resize, dan flattening).
- μ : Rata-rata piksel dari dataset (per piksel).
- \mathbf{U}_k : Matriks eigenvector dengan k dimensi utama dari PCA.

Kemudian, hitung jarak Euclidean antara gambar query dengan semua gambar dalam dataset:

$$d(\mathbf{q}, \mathbf{z}_i) = \sqrt{\sum_{j=1}^k (q_j - z_{ij})^2}$$

- $d(\mathbf{q}, \mathbf{z}_i)$ = Jarak antara gambar query \mathbf{q} dan gambar ke- i dalam ruang komponen utama.
- \mathbf{z}_i = Vektor proyeksi dari gambar ke- i dalam dataset ke ruang komponen utama.
- q_j : Elemen ke- j dari vektor proyeksi query \mathbf{q} .
- z_{ij} : Elemen ke- j dari vektor proyeksi gambar ke- i , yaitu \mathbf{z}_i .
- k : Jumlah dimensi ruang komponen utama yang dipilih.

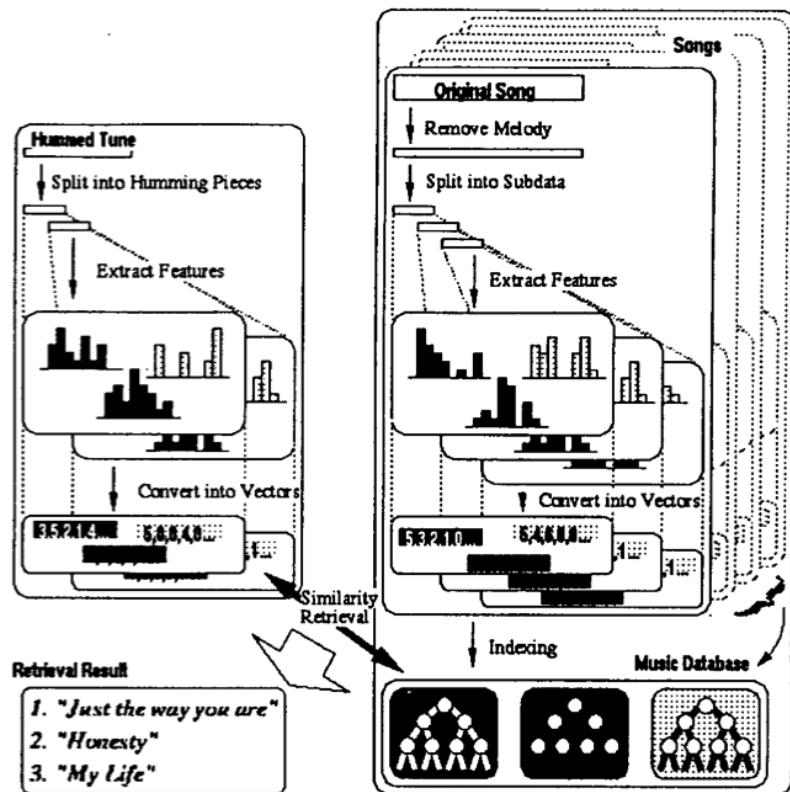
Lalu, Urutkan hasil berdasarkan jarak terkecil.

5. Retrieval and Output

Kumpulkan gambar-gambar yang mirip dengan query masukan dengan cara melakukan limitasi jumlah atau dengan memberikan batas jarak euclidean.

Hasil dari pencarian gambar dapat digabungkan dengan hasil pencarian suara ataupun dijalankan sendiri.

1.4. Music Information Retrieval - Query by Humming



Gambar 2. Tahapan pada MIR dengan metode Query By Humming

Pada query by humming akan dilakukan beberapa langkah utama. Pertama, dilakukan pemrosesan audio, di mana suara yang diinput direkam atau diterima dan dipersiapkan untuk tahap berikutnya. Selanjutnya, data audio tersebut melalui tahap ekstraksi fitur vektor, yaitu proses konversi data audio menjadi representasi numerik berupa vektor fitur yang dapat dianalisis. Terakhir, pada tahap pencarian similaritas vektor, vektor fitur yang dihasilkan dibandingkan dengan dataset untuk menemukan hasil yang paling sesuai atau diatas nilai kemiripan/similaritas yang telah ditentukan.

1. Pemrosesan Audio

Pemrosesan audio dalam sistem query by humming menggunakan file MIDI dengan fokus pada track melodi utama, umumnya di Channel 1. Setiap file MIDI diproses menggunakan metode windowing yang membagi melodi menjadi segmen 20-40 beat dengan sliding window 4-8 beat. Teknik ini memungkinkan pencocokan fleksibel dari berbagai bagian lagu yang mungkin diingat pengguna.

Proses windowing disertai normalisasi tempo dan pitch untuk mengurangi variasi humming. Setiap note event dikonversi menjadi representasi numerik yang mempertimbangkan durasi dan urutan nada, memungkinkan sistem membandingkan potongan melodi dengan database.

Berikut adalah formula untuk melakukan normalisasi tempo yang dibutuhkan :

$$NP(note) = \frac{(note - \mu)}{\sigma}$$

Dimana μ adalah rata rata dari pitch dan σ adalah standar deviasi dari pitch.

2. Ekstraksi Fitur

2.1. Distribusi tone

Distribusi tone diukur berdasarkan tiga viewpoints.

Fitur Absolute Tone Based (ATB) menghitung frekuensi kemunculan setiap nada berdasarkan skala MIDI (0-127). Histogram yang dihasilkan

memberikan gambaran distribusi absolut nada dalam data. Hal ini penting untuk menangkap karakteristik statis melodi dalam sinyal audio. Langkah pertama adalah membuat histogram dengan 128 bin, sesuai dengan rentang nada MIDI dari 0 hingga 127. Kemudian, hitung frekuensi kemunculan masing-masing nada MIDI dalam data. Setelah itu, normalisasi histogram untuk mendapatkan distribusi yang terstandarisasi.

Fitur Relative Tone Based (RTB) menganalisis perubahan antara nada yang berurutan, menghasilkan histogram dengan nilai dari -127 hingga +127. RTB berguna untuk memahami pola interval melodi, yang lebih relevan dalam mencocokkan humming dengan dataset yang tidak bergantung pada pitch absolut. Dimulai dengan membangun histogram yang memiliki 255 bin dengan rentang nilai dari -127 hingga +127. Selanjutnya, hitung selisih antara nada-nada yang berurutan dalam data. Terakhir, lakukan normalisasi pada histogram yang telah dibuat.

Fitur First Tone Based (FTB) fokus pada perbedaan antara setiap nada dengan nada pertama, menciptakan histogram yang mencerminkan hubungan relatif terhadap titik referensi awal. Pendekatan ini membantu menangkap struktur relatif nada yang lebih stabil terhadap variasi pitch pengguna. Histogram dibuat dengan 255 bin, juga mencakup rentang nilai dari -127 hingga +127. Kemudian, hitung selisih antara setiap nada dalam data dengan nada pertama. Histogram yang dihasilkan kemudian dinormalisasi untuk menghasilkan distribusi yang seimbang.

2.2. Normalisasi

Normalisasi memastikan bahwa semua nilai dalam histogram berada dalam skala probabilitas. Berikut adalah formula umum dari normalisasi yang digunakan:

$$H_{norm} = \frac{H[d]}{\sum_{d=1}^{127} H[d]}$$

Dimana H adalah Histogram dan d adalah bin dari histogram tersebut.

3. Penghitungan Similaritas

Ubah setiap histogram menjadi sebuah vektor dan lakukan perhitungan kemiripannya menggunakan cosine similarity. Pada jurnal terkait, metode yang digunakan adalah euclidean distance, tetapi pada tugas kali ini metode perhitungan similaritas yang akan digunakan adalah cosine similarity. Cosine Similarity adalah ukuran untuk menentukan seberapa mirip dua vektor dalam ruang berdimensi tinggi, dengan menghitung sudut cosinus di antara keduanya. Semakin kecil sudutnya (semakin dekat ke 1 hasilnya), semakin mirip kedua vektor tersebut. Sehingga cosine similarity bisa dijadikan salah satu metode lain dalam perhitungan similaritas. Silahkan lakukan eksplorasi eksperimen dengan pembobotan berbeda untuk setiap fitur dan tentukan bobot terbaiknya.

Berikut adalah formula dari cosine similarity :

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Gambar 3. Cosine Similarity Formula

Bab 2: Teori Singkat

2.1 Sistem Temu Balik Suara (MIR)

Sistem Temu Balik Suara, atau *Music Information Retrieval* (MIR) adalah bidang studi interdisipliner yang berfokus pada pengambilan, analisis, dan pengorganisasian informasi dari data terkait musik. Bidang ini memadukan metode dari ilmu komputer, musikologi, pembelajaran mesin, dan pemrosesan sinyal untuk memungkinkan sistem memproses musik dalam berbagai formatnya, seperti sinyal audio, skor, dan metadata.

Area utama MIR meliputi:

1. Ekstraksi Fitur: Menganalisis sinyal musik untuk memperoleh karakteristik seperti tempo, nada, harmoni, ritme, dan timbre.
2. Kesamaan Musik: Mengukur kesamaan antara cuplikan musik untuk tugas seperti pembuatan *playlist* atau rekomendasi.
3. Klasifikasi dan Penandaan: Secara otomatis mengkategorikan musik berdasarkan genre, suasana, atau instrumentasi.
4. Pengambilan Musik: Mencari musik melalui kueri seperti *humming* atau menyediakan klip lagu.
5. Rekomendasi Musik: Personalisasi saran musik berdasarkan preferensi pengguna.
6. Konteks Pengguna: Memahami bagaimana pendengar berinteraksi dengan musik berdasarkan faktor-faktor seperti lokasi, waktu, dan suasana.

Kemajuan terkini dalam MIR meliputi pendekatan yang berpusat pada pengguna untuk personalisasi yang lebih baik dan teknik evaluasi tingkat lanjut untuk meningkatkan akurasi.

2.2 Metode Ekstraksi Fitur Berdasarkan Humming

Metode Ekstraksi Fitur Berdasarkan Humming adalah sistem pengambilan berbasis kesamaan yang memungkinkan pengguna untuk *humming* bagian dari melodi sebagai kueri untuk mencari lagu dalam basis data. Sistem ini terkenal karena penggunaan ruang vektor fitur berdimensi tinggi, yang menggabungkan fitur transisi nada dan distribusi nada untuk meningkatkan akurasi dan fleksibilitas. Adapun beberapa fitur utama dari metode ini, antara lain:

1. Ekstraksi Fitur
 - a. Transisi Nada: Menangkap perkembangan waktu nada musik.
 - b. Distribusi Nada: Mewakili nada menggunakan histogram yang memperhitungkan nada absolut (ATB), perbedaan relatif antara nada berurutan (RTB), dan perbedaan antara nada dan nada pertama dalam urutan (FTB).
2. Penanganan *Variability*
 - a. Histogram Fuzzy: Menggabungkan nilai nada di sekitar, mengurangi efek variasi nada dan tempo yang umum dalam melodi *humming*.
 - b. Normalisasi Tempo: Menyesuaikan kueri *humming* dan *database* musik ke tempo dasar untuk perbandingan yang konsisten.

3. Konstruksi Basis Data

Musik disimpan dalam format MIDI, dan hanya *channel* melodi yang digunakan untuk perbandingan. Lagu dibagi menjadi subdata yang tumpang tindih menggunakan metode *sliding window*, yang memungkinkan perbandingan/pencocokan sebagian dari bagian mana pun dari sebuah lagu.

4. Pengambilan Kesamaan

Fitur diubah menjadi vektor berdimensi tinggi dan pengambilan menggunakan penjumlahan linier tertimbang dari jarak antara vektor fitur untuk memberi peringkat hasil berdasarkan kesamaan.

Adapun beberapa keuntungan dari metode *humming*, antara lain fleksibilitas, efisiensi, dan kemampuan beradaptasi.

2.3 Image Retrieval dengan PCA

Image Retrieval adalah proses mencari dan mengambil gambar dari suatu database berdasarkan kemiripan fitur yang diekstraksi dari gambar terhadap fitur pada gambar-gambar dalam database tersebut. Salah satu cara yang sering digunakan untuk melakukan proses ini adalah metode *Principal Component Analysis* (PCA) yang digunakan untuk menyederhanakan fitur pada gambar sehingga membuat perhitungan dan pemrosesan menjadi lebih efisien.

Berikut tahapan pada *Principal Component Analysis*.

1. Ekstraksi Fitur Gambar

Setiap gambar dalam database direpresentasikan sebagai matriks piksel dalam format grayscale. Gambar kemudian diubah menjadi vektor satu dimensi (*flattened*) untuk mempermudah perhitungan. Setelah di *flatten*, akan diambil rata-rata dari setiap piksel dan dilakukan standarisasi dengan cara mengurangi setiap piksel pada setiap gambar di database dengan rata-rata piksel tersebut.

2. Reduksi Dimensi Menggunakan PCA

PCA adalah teknik statistika yang digunakan untuk mereduksi dimensi data dengan tetap mempertahankan informasi penting (*principal component*). Untuk melakukan PCA, pertama harus dihitung matriks *Singular Value Decomposition* dari matriks kovarian piksel. Matriks kovarian didapat dengan formula berikut.

$$C = \frac{1}{N} X' X$$

X' : Matriks data yang sudah distandarisasi

Matriks kovarian akan didekomposisi sebagai berikut.

$$C = U \Sigma U^T$$

Kemudian pilih sebuah angka k yaitu jumlah komponen utama teratas dari matriks U yang akan digunakan untuk proyeksi PCA. Lalu matriks PCA didapat dengan formula berikut.

$$Z = X' U_k$$

3. Pencarian Gambar

Gambar yang ingin dicari akan diproses dan direduksi dimensinya sebagaimana gambar lainnya pada database, kemudian dihitung sampai mendapatkan PCA dengan menggunakan U_k yang sama. Jarak antara proyeksi PCA gambar query dan proyeksi PCA gambar dalam database dihitung menggunakan *Euclidean distance*:

$$d(q, z_i) = \sqrt{\sum_{j=1}^k (q_j - z_{ij})^2}$$

Gambar pada database yang memiliki jarak *euclidean* terkecil terhadap gambar yang di *query* adalah hasil pencarian gambar yang termirip.

Bab 3: Arsitektur Website (Frontend) dan Arsitektur Program Information Retrieval (Backend)

3.1 Frontend

Pengembangan website bagian frontend bertujuan untuk mengembangkan bagian website yang dilihat dan berinteraksi langsung dengan pengguna. Dalam pembuatan bagian frontend, kami menggunakan **Next.js**, **TypeScript**, dan **Tailwind CSS**. Kami juga menggunakan beberapa teknologi pendukung seperti **shadcn/ui** (UI Framework), **Radix** (UI Library), dan **Lucide** (Icon Library).

Berikut beberapa komponen yang terdapat pada website kami:

1. Alert Dialog (shadcn/ui)

Digunakan sebagai peringatan yang meminta pengguna untuk mengambil tindakan, seperti melanjutkan atau membatalkan proses. Contohnya, saat pengguna membatalkan pencarian lagu setelah mengunggah file.

2. Alert (shadcn/ui)

Digunakan sebagai tanda konfirmasi selesainya suatu proses, contohnya proses pengunggahan mappor.

3. Button (shadcn/ui)

Button adalah komponen UI yang dapat digunakan untuk sejumlah aksi, seperti *navigate*, *cancel*, dan lain sebagainya. Beberapa button yang terdapat pada website kami, antara lain button memilih *search by humming* dan/atau *search by album cover*, button untuk melakukan pengunggahan file dan/atau dataset, button untuk melakukan search, button untuk membatalkan search.

4. Badge (shadcn/ui)

Berisi informasi minor. Contohnya dalam menampilkan persentase kemiripan hasil search dengan file yang diunggah.

5. Card (shadcn/ui)

Digunakan untuk menampilkan daftar lagu beserta gambar album yang ada di dataset.

6. Pagination (shadcn/ui)

Digunakan agar dalam satu page, hasil search yang keluar dibatasi sehingga pengguna tidak perlu melakukan *scrolling* terlalu lama.

7. Home Page

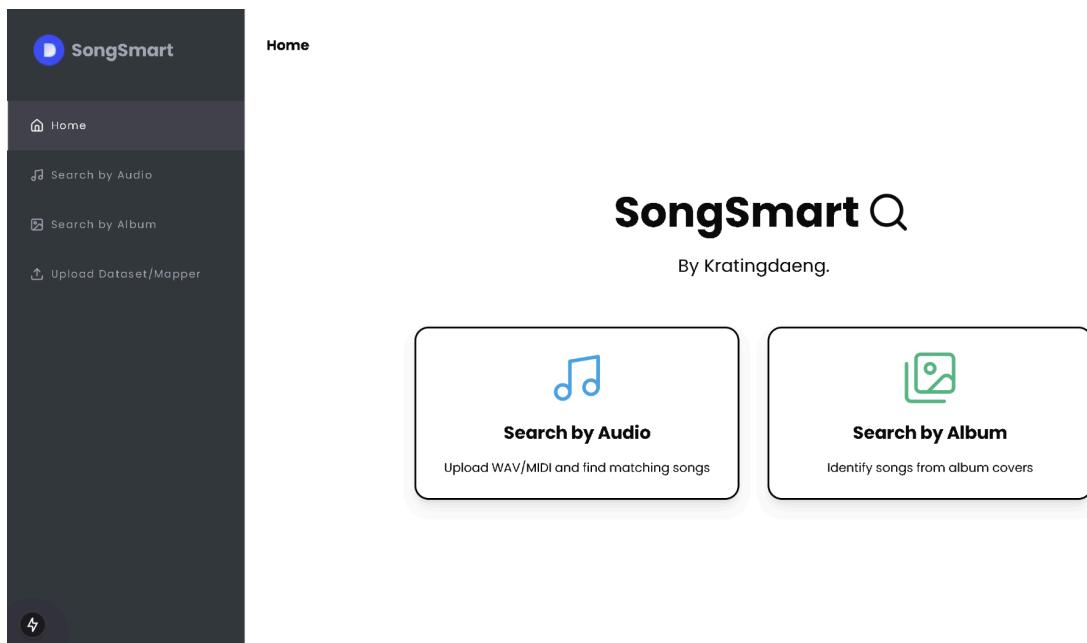
Tampilan pertama saat website dibuka pertama kali. Berisi pilihan untuk melakukan *search by audio* atau *search by album*.

8. Side Bar

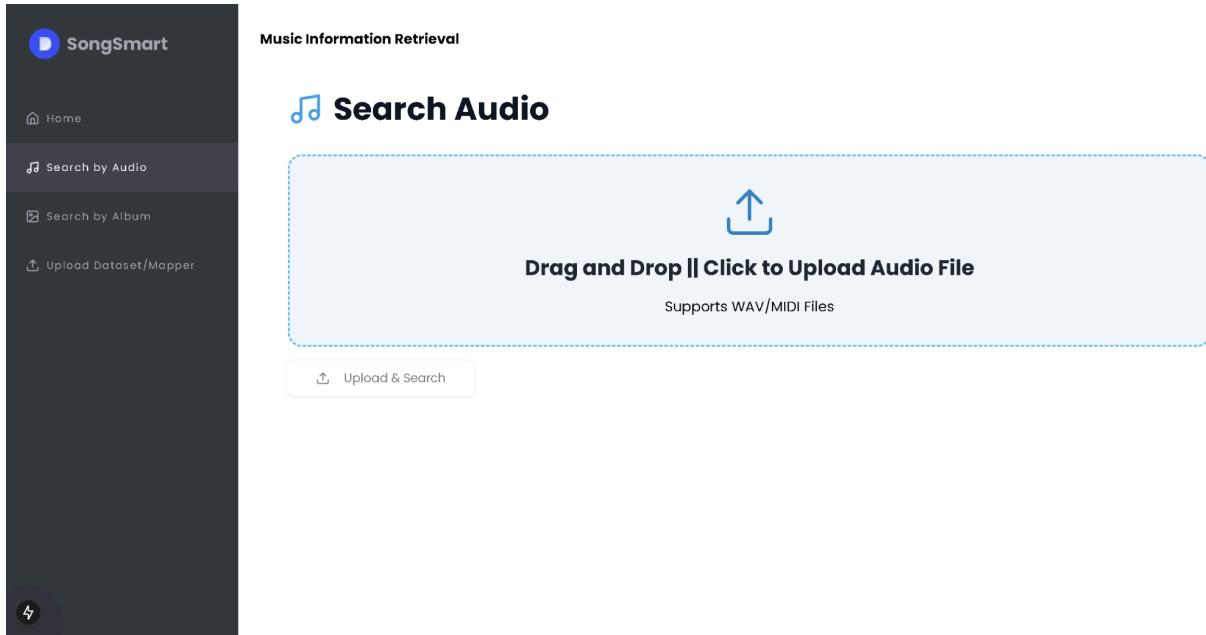
Berisi pilihan-pilihan page yang dapat dituju, yaitu Home, Search by Audio, Search by Album, dan Upload Dataset/Mapper.

9. Upload

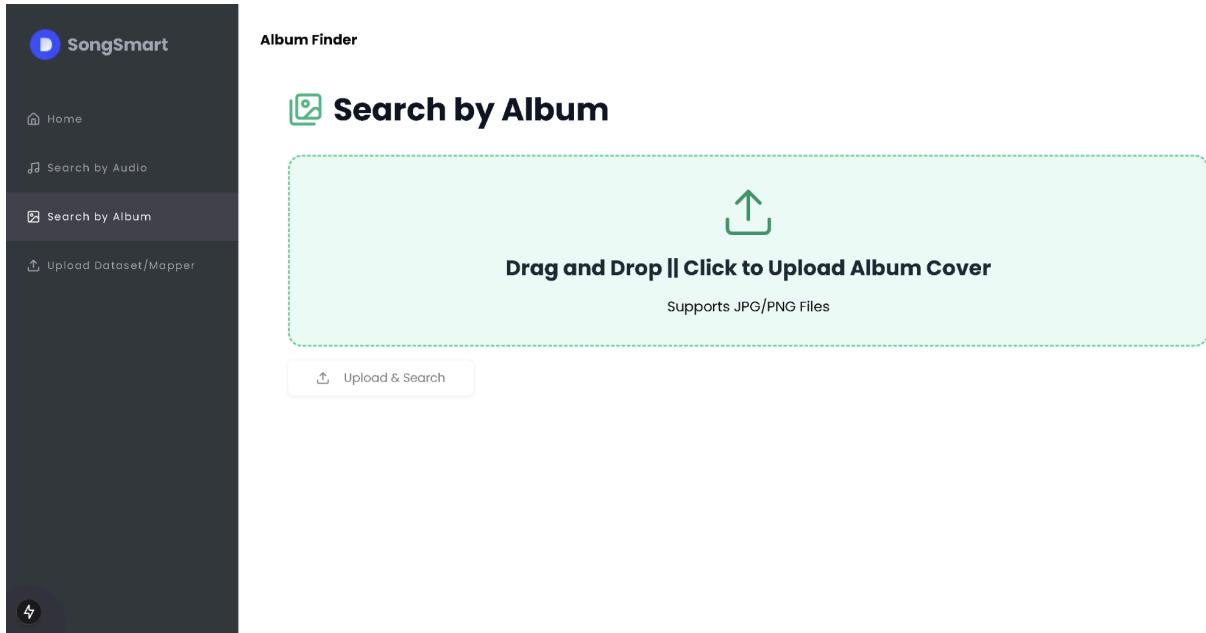
Digunakan untuk mengeksekusi proses pengunggahan, seperti *upload* dataset, mapper, audio query, dan image query.



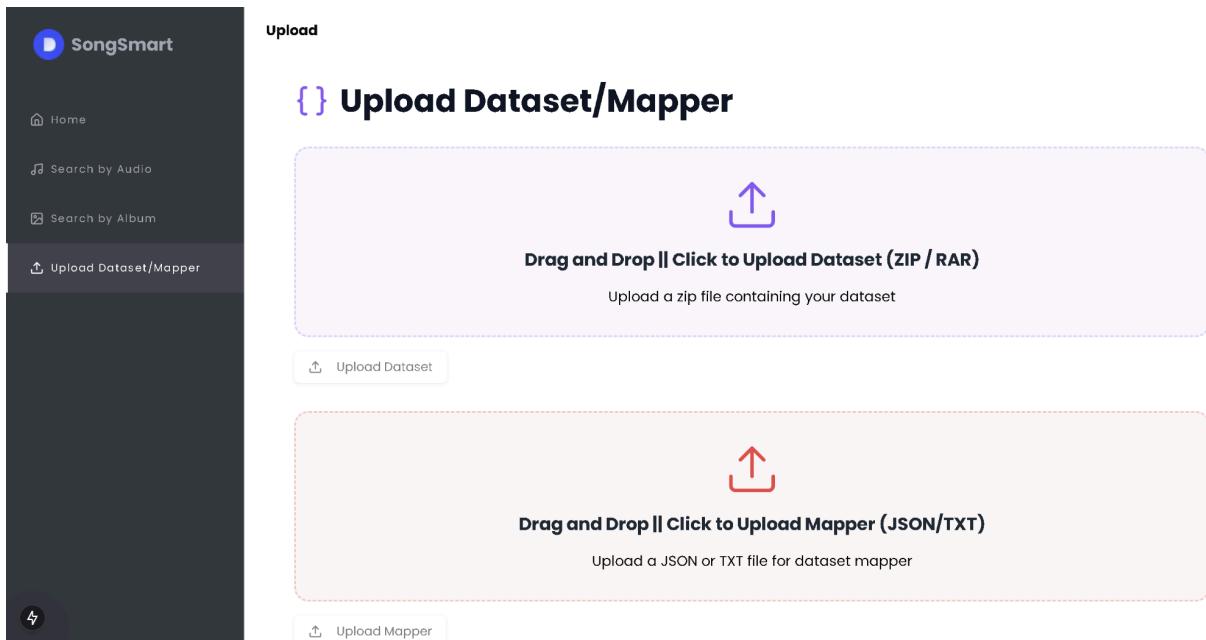
Gambar 4. Tampilan home page SongSmart



Gambar 5. Tampilan page Search by Audio



Gambar 6. Tampilan page Search by Album



Gambar 7. Tampilan page Search by Album

3.2 Backend

Dalam pembuatan bagian backend, kami menggunakan Python sebagai language utama dalam perhitungan *Music Information Retrieval* maupun *Album Finder*. Setelah itu, kami menggunakan FastAPI sebagai web framework pembuatan API sebagai penghubung antara Frontend, Backend, dan Database. Setelah itu, kami menggunakan SQLite sebagai database untuk perhitungan yang

lebih teratur dan cepat. Kami juga memanfaatkan beberapa library/teknologi pendukung seperti os, shutil, dan PIL, serta library-library Python seperti numpy, mido, dan librosa.

1. Album Picture Finder - Principal Component Analysis

Principal Component Analysis digunakan untuk mencari gambar terdekat atau gambar yang paling mirip dengan suatu gambar query. Proses ini dilakukan dengan mengekstraksi piksel-piksel yang ada pada gambar, kemudian dilakukan berbagai perhitungan agar dimensi piksel yang digunakan tidak terlalu besar sehingga membuat pencarian efisien.

Untuk fitur ini, digunakan *library* numpy untuk membantu perhitungan serta *library* Pillow (PIL) untuk mempermudah manipulasi piksel-piksel pada gambar. Selain itu digunakan sebuah *class* untuk menyimpan data-data perhitungan pada gambar, dimana satu objek *class* merepresentasikan satu gambar pada dataset. Berikut kode definisi *class* tersebut.

```
# CLASS
class ImageData:
    def __init__(self, filename: String, pixels: Vector, mod_pixels: Vector) -> None:
        self.filename = filename
        self.pixels = pixels
        self.modified_pixels = mod_pixels
        self.pca = None
        self.k = None # Principal Component Count
        self.euclid_distance = None
        self.similarity = None

    def compress_pixels(self):
        buffer = io.BytesIO()
        img = PIL.fromarray(self.pixels) # Convert NumPy array to image
        img = img.convert('RGB')
        img.save(buffer, format="JPEG", quality=75, optimize=True) # Compress image
        return buffer.getvalue()
```

Gambar 8. Class definisi ImageData

Hal pertama yang dilakukan untuk menghitung PCA gambar adalah mengekstraksi piksel-piksel pada gambar, kemudian mengubah RGB menjadi grayscale, kemudian piksel-piksel tersebut dinormalisasi dengan cara mengurangi resolusi atau dimensi piksel gambar. Setelah itu dilakukan *flattening* pada matriks piksel sehingga menjadi matriks 1D atau sebuah vektor. Berikut kode untuk proses-proses tersebut.

```

✓ def normalize_image(cropped: Matrix) -> Matrix:
    dim = (16, 16)
    img = PIL.fromarray(cropped)
    resized_img = img.resize(dim, PIL.Resampling.BILINEAR) # Resizing
    return np.array(resized_img) # Return numpy array

✓ def matrix_to_1d(resized: Matrix) -> Vector:
    flattened = []
    for row in resized:
        for value in row:
            flattened.append(value) # append setiap row ke dalam 1 row saja
    return np.array(flattened) # numpy array untuk speed efficiency

# Menyatukan semua proses diatas
✓ def preprocess_image(file_path: String) -> Vector:
    grayscale = extract_grayscale(file_path)
    cropped = crop_image(grayscale)
    resized = normalize_image(cropped)
    vector = matrix_to_1d(resized)
    return vector

# Memproses data-data gambar dataset
✓ def load_dataset(directory: String) -> list[ImageData]:
    dataset_list = []
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)
        img = PIL.open(file_path)
        image = ImageData(filename, np.array(img), preprocess_image(file_path))
        dataset_list.append(image)
    return dataset_list

```

Gambar 9. Program proses perhitungan PCA dan Album Finder

Setelah vektor piksel didapatkan, akan dilakukan standarisasi piksel dengan cara menghitung piksel rata-rata dari seluruh gambar pada dataset kemudian mengurangi setiap piksel dengan rata-rata tersebut. Berikut kode untuk proses ini.

```

#####
# Data Centering (Standardization) #####
# Menghitung rata2 pixel gambar dataset

def get_pixel_means(dataset: list[ImageData]) -> Vector:
    pixel_matrix = np.array([image.modified_pixels for image in dataset])
    img_total = len(dataset)
    pixel_means = []

    for j in range(pixel_matrix.shape[1]):
        pixel_sum = 0
        pixel_sum = np.sum(pixel_matrix, dtype=np.int32)
        pixel_means.append(pixel_sum / img_total)

    return np.array(pixel_means)

def standardize_images(dataset: list[ImageData], pixel_means) -> None:
    # Ubah vector menjadi matrix, dimana setiap baris adalah image berbeda
    # dan setiap kolom adalah pixel2 nya matrix N*jumlah_pixel
    pixel_matrix = np.array([image.modified_pixels for image in dataset])
    standardized_pixels_list = pixel_matrix - pixel_means # Standarisasi Pixel

    # Perbarui pixel pada dataset
    for image, standardized_pixels in zip(dataset, standardized_pixels_list):
        image.modified_pixels = standardized_pixels

```

Gambar 10. Program standarisasi pixels

Setelah piksel distandarisasi, dilanjutkan kepada perhitungan PCA yang dimulai dengan mencari matriks-matriks dekomposisi (SVD) dari matriks kovarian. Untuk mencari SVD, diperlukan untuk mencari nilai-nilai eigen dari matriks kovarian terlebih dahulu. Nilai-nilai eigen dan vektor eigen didapatkan dengan memanfaatkan algoritma dekomposisi QR ([Youtube](#)). Berikut kode untuk tahap ini.

```

# Helper to get eigenvalues
def qr_decomposition(A: Matrix) -> Tuple[Matrix, Matrix]:
    m, n = A.shape # Ambil row dan col
    Q = np.zeros((m, n))
    R = np.zeros((n, n))

    for i in range(n):
        v = A[:, i].copy()

        for j in range(i):
            R[j, i] = np.dot(Q[:, j], v)
            v = v - R[j, i] * Q[:, j]

        R[i, i] = np.sqrt(np.sum(v**2))
        Q[:, i] = v / R[i, i]

    return Q, R

# Menggunakan QR Algorithm https://www.youtube.com/watch?v=McHW22IJ3UM
def get_eigen(A: Matrix) -> Tuple[Vector, Matrix]:
    iter = 1000
    tol = 10 - 6 # Tolerance (presisi)

    n = A.shape[0]
    Q_total = np.eye(n)

    for _ in range(iter):
        Q, R = qr_decomposition(A)

        A = R @ Q
        Q_total = Q_total @ Q

        # Convergence check
        off_diag = A - np.diag(np.diagonal(A))
        if np.sqrt(np.sum(off_diag**2)) < tol:
            break

    eigenvalues = np.maximum(np.diagonal(A), 0) # Round negatives to zero
    return eigenvalues, Q_total

```

Gambar 11. Fungsi QR Decomposition dan perhitungan EigenValue

Setelah nilai eigen didapatkan, dilanjutkan dengan perhitungan SVD dan PCA. SVD didapatkan dengan membentuk matriks diagonal singular dari nilai eigen yang sudah didapatkan untuk membentuk matriks sigma. Setelah itu, matriks U didapatkan dengan

mengurutkan vektor-vektor eigen berdasarkan nilai singular eigen yang berkorespondensi dengan vektor eigen dengan urutan yang sama pada matriks sigma (terurut membesar).

Setelah itu, matriks PCA adalah hasil perkalian matriks piksel gambar dengan matriks U yang diambil k komponen teratas dengan k adalah bilangan bulat arbiter yang dipilih sebagai komponen penting teratas pada matriks. Adapun kode untuk tahap ini adalah sebagai berikut.

```
# I.S. M adalah matriks n x n (square)
# Mengembalikan Matriks U
def singular_value_decomposition(C: Matrix) -> Tuple[Matrix, int]:
    eigenvalues, eigenvectors = np.linalg.eig(C)
    # eigenvalues, eigenvectors = get_eigen(C)
    # Urut eigenvalue berdasarkan nilai singular
    sorted_indices = np.argsort(eigenvalues)[::-1]
    U = eigenvectors[:, sorted_indices]
    # U, s, vh = np.linalg.svd(C)
    k = choose_k(eigenvalues)
    return U, k

def get_top_k_components(U: Matrix, k: int) -> Matrix:
    return U[:, :k]

# Digunakan pada dataset yang sudah distandarisasi
# Set atribut pca pada List ImageData
def principal_component_analysis_dataset(dataset: list[ImageData]) -> Matrix:
    X = np.array([image.modified_pixels for image in dataset])

    C = get_covariance_matrix(dataset)
    U, k = singular_value_decomposition(C)

    if dataset[0].k is not None:      # Overwrite k (ini hanya untuk query)
        k = dataset[0].k
    Uk = get_top_k_components(U, k)
    Z_matrix = X @ Uk

    for image, projection in zip(dataset, Z_matrix):
        image.pca = np.real(projection)
        image.k = k
    return Uk
```

Gambar 12. Fungsi perhitungan PCA

Langkah terakhir pada fitur ini adalah untuk menghitung jarak nilai-nilai matriks PCA dari gambar-gambar dataset terhadap matriks PCA gambar yang di *query*. Gambar yang dianggap mirip adalah gambar-gambar yang memiliki jarak *euclidean* terkecil. Berikut kode untuk tahap ini.

```
##### Similarity Computation #####
def calculate_euclidian_distance(dataset: list[ImageData], query: ImageData) -> None:
    q = query.pca
    for images in dataset:
        z = images.pca
        distance = np.sqrt(np.sum((q - z) ** 2))
        images.euclid_distance = distance
        images.similarity = (2000 - distance) / 2000
```

Gambar 13. Fungsi menghitung euclidean distance untuk kemiripan

2. Music Information Retrieval - Query by Humming

Metode Ekstraksi Fitur Berdasarkan Humming digunakan untuk mencari lagu yang paling mirip dengan *query* yang di-*upload* oleh pengguna. Proses ini dilakukan dengan mengekstraksi data pitch dari file audio, baik dalam format .wav maupun .midi, lalu menghitung beberapa fitur utama seperti Absolute Tone Bin (ATB), Relative Tone Bin (RTB), dan First Tone Bin (FTB). Fitur ini membantu dalam merepresentasikan melodi sebagai vektor berdimensi tinggi yang dapat dibandingkan dengan data dalam basis data musik. Pertama, dilakukan pemrosesan pada file audio berdasarkan tipe filenya *wav* atau *midi*. Berikut kode untuk proses ini.

```

# Convert wav to pitch data
def preprocess_wav(file_path):
    audio_data, sample_rate = librosa.load(file_path)

    # Get pitches from wav file
    pitches = librosa.yin(audio_data, fmin=librosa.note_to_hz('C1'), fmax=librosa.note_to_hz('C8'), sr=sample_rate)

    # Convert pitch (Hz) to MIDI note numbers
    pitch_data = []
    for pitch in pitches:
        if pitch > 0:
            pitch_data.append(int(librosa.hz_to_midi(pitch)))
        else:
            pitch_data.append(0)
    return pitch_data

# Convert midi to pitch data
def preprocess_midi(file_path):
    midi = MidiFile(file_path)
    pitch_data = []

    for track in midi.tracks:
        for msg in track:
            if msg.type == 'note_on' and msg.velocity > 0:
                pitch_data.append(msg.note)
    return pitch_data

```

Gambar 14. Fungsi preprocess midi dan wav

Untuk file .wav, digunakan Librosa untuk mengekstrak frekuensi nada dengan algoritma YIN, yang kemudian dikonversi menjadi nada MIDI. Sedangkan untuk file .midi, nada langsung diambil dari event *note_on* pada channel melodi.

Pitch yang diekstrak kemudian diolah menggunakan metode *sliding window* yang membagi pitch menjadi segmen-segmen kecil yang berhimpitan. Hal ini memungkinkan sistem untuk melakukan pencocokan sebagian saja sehingga *query* pendek tetap dapat dicocokkan dengan

melodi lengkap dalam database. Berikut kode yang digunakan untuk tahap ini.

```
# Windowing
def apply_sliding_window(data, window, step):
    segments = []

    for i in range(0, len(data) + 1 - window, step):
        segment = data[i:i + window]
        segments.append(segment)
    return segments

# Audio Processing
def get_processed_audio(file_path):
    if file_path.endswith('.wav'):
        pitch_data = preprocess_wav(file_path)
    elif file_path.lower().endswith('.mid', '.midi'):
        pitch_data = preprocess_midi(file_path)
    else:
        raise ValueError("Unsupported file format")

    # Windowing
    segments = apply_sliding_window(pitch_data, 40, 8)
    pitches = [p for segment in segments for p in segment if p > 0]
    if len(pitches) == 0:
        print("Pitches list is empty.")
        return pitch_data

    return segments
```

Gambar 15. Fungsi Audio Processing

Selanjutnya, fitur pitch yang telah di-segmentasi akan dikalkulasi menggunakan histogram. Histogram ini dibagi menjadi tiga fitur utama:

- ATB (Absolute Tone Based): Histogram distribusi nada absolut dalam rentang 0-127.
- RTB (Relative Tone Based): Histogram perbedaan nada antara nada-nada berurutan dalam rentang -127 hingga 127.
- FTB (First Tone Based): Histogram perbedaan nada terhadap nada pertama dalam setiap segmen melodi.

Setelah histogram-histogram tersebut dihitung, dilakukan normalisasi agar distribusi nilainya konsisten meskipun muncul banyak variasi nada atau tempo. Rata-rata dari setiap histogram

kemudian digunakan untuk merepresentasikan setiap segmen dalam bentuk vektor fitur berdimensi tinggi. Berikut implementasinya.

```
# Create Histogram and Normalize
def create_and_normalize_histogram(data, num_bins, val_range=None):
    # Create
    if val_range is None:
        val_range = (0, num_bins)
    histogram, _ = np.histogram(data, bins=num_bins, range=val_range)

    # Normalize
    total = np.sum(histogram)
    if total > 0:
        return histogram / total
    else:
        print("Error Warning: Histogram sum 0 atau negatif")
        return histogram

def extract_features(pitch_data):
    atb_list = []
    rtb_list = []
    ftb_list = []

    for windowed_data in pitch_data:
        # ATB [0, 127]
        bins_atb = 128
        atb_normalized = create_and_normalize_histogram(windowed_data, bins_atb)
        atb_list.append(atb_normalized)

        # RTB [-127, 127] (selisih antara nada-nada berurutan)
        bins_rtb = 255
        rtb_data = [windowed_data[i] - windowed_data[i - 1] for i in range(1, len(windowed_data))]
        rtb_normalized = create_and_normalize_histogram(rtb_data, bins_rtb, (-127, 127))
        rtb_list.append(rtb_normalized)

        # FTB [-127, 127] (selisih antara nada-nada dengan nada pertama)
        bins_ftb = 255
        if len(windowed_data) > 0:
            ftb_data = [windowed_data[i] - windowed_data[0] for i in range(1, len(windowed_data))]
            ftb_normalized = create_and_normalize_histogram(ftb_data, bins_ftb, (-127, 127))
        else:
            ftb_normalized = np.zeros(bins_ftb, dtype=float)
        ftb_list.append(ftb_normalized)

    # Find avg of each method window histograms
    atb_combined = np.mean(atb_list, axis=0)
    rtb_combined = np.mean(rtb_list, axis=0)
    ftb_combined = np.mean(ftb_list, axis=0)

    return atb_combined, rtb_combined, ftb_combined
```

Gambar 16. Fungsi extract features dan windowing

Proses pencocokan dilakukan dengan menghitung cosine similarity antara vektor fitur dari kueri humming dan vektor fitur lagu dalam basis data menggunakan fungsi `get_cosine_similarity`. Cosine similarity digunakan karena mampu mengukur kesamaan arah antar vektor, yang ideal digunakan untuk membandingkan melodi. Hasil dari setiap perhitungan *cosine similarity* (ATB, RTB, dan FTB) kemudian dirata-ratakan untuk mendapatkan skor *similarity* akhir.

```
def get_cosine_similarity(vector_A, vector_B):
    dot_product = np.dot(vector_A, vector_B)
    norm_vector_A = np.linalg.norm(vector_A)
    norm_vector_B = np.linalg.norm(vector_B)
    norm_product = norm_vector_A * norm_vector_B

    # check handle division by zero
    if norm_product == 0:
        return 0

    cosine_similarity = dot_product / norm_product
    return cosine_similarity
```

Gambar 16. Fungsi perhitungan cosine similarity

Dalam implementasi kami, perhitungan *cosine similarity* digunakan langsung untuk mencari skor *similarity* akhir dengan mengimplementasikan sebuah *method* dari *class* `AudioData`. Penggunaan *class* disini dengan maksud mempermudah penyimpanan representasi sebuah file audio agar manajemen file menjadi simpel. Berikut implementasi *class* `AudioData` beserta *method-method*-nya.

```

class AudioData:
    def __init__(self, filename, type=None):
        if (type == "q"):
            self.dir = DIR + "query/"
        else:
            self.dir = AUD_DIR
        self.filename = filename
        self.pitch_data = None
        self.atb = None
        self.rtb = None
        self.ftb = None
        self.similarity = None

    def extract_features(self):
        self.pitch_data = get_processed_audio(self.dir + self.filename)
        self.atb, self.rtb, self.ftb = extract_features(self.pitch_data)

    def calculate_similarity(self, query: 'AudioData'):
        # Hitung cosine similarity ATB, RTB, FTB terpisah
        atb_query, rtb_query, ftb_query = query.atb, query.rtb, query.ftb
        atb_dataset, rtb_dataset, ftb_dataset = self.atb, self.rtb, self.ftb

        similarity_atb = get_cosine_similarity(atb_query, atb_dataset)
        similarity_rtb = get_cosine_similarity(rtb_query, rtb_dataset)
        similarity_ftb = get_cosine_similarity(ftb_query, ftb_dataset)

        # Rata-rata cosine similarity
        avg_similarity = (similarity_atb + similarity_rtb + similarity_ftb) / 3
        self.similarity = float(avg_similarity)

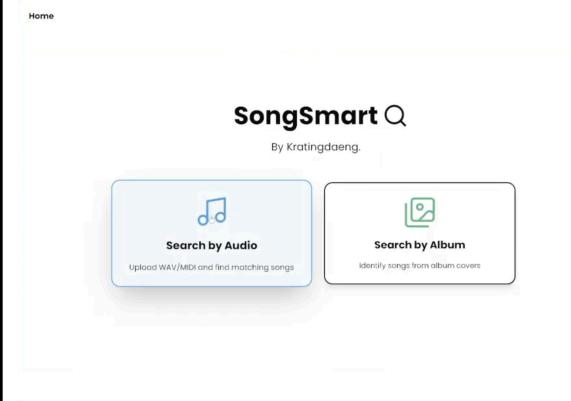
    def __str__(self):
        # return '\n'.join(f'{key}: {value}' for key, value in self.__dict__.items())
        return f"File: {self.filename}\nATB: {self.atb}\nRTB: {self.rtb}\nFTB: {self.ftb}\n"

```

Gambar 16. Class AudioData dan perhitungan kemiripan

Setiap *instance* AudioData akan di-set skor *similarity*-nya ketika ada *query*, audio-audio dengan similarity di atas 50% akan ditampilkan sebagai *closest match* pada website.

Bab 4: Eksperimen

Pengujian yang dilakukan	Hasil Pengujian
Home Page	<p>Home page sebagai halaman judul dan navigation page.</p>  <p>The screenshot shows the SongSmart homepage. At the top, it says "SongSmart" with a magnifying glass icon. Below it, it says "By Kratingdaeng.". There are two main buttons: "Search by Audio" with a microphone icon and "Search by Album" with a square and camera icon. Below each button is a small explanatory text: "Upload WAV/MIDI and find matching songs" for audio and "Identify songs from album covers" for albums.</p>
Search by Audio	<p>Screen untuk upload query file audio dan hasilnya melalui pagination untuk 10 hasil tiap page, disertakan waktu runtime.</p>

Music Information Retrieval

Search Audio

Selected: locked_out_of_heaven.wav
Supports WAV/MIDI Files

Upload & Search

Search Results Runtime: 1.863 seconds

Image	Title	Artist	Match
	locked_out_of_heaven.wav	Bruno Mars	Match: 100.00%
	fearless.wav	Taylor Swift	Match: 93.80%
	blank_space.wav	Bruno Mars	Match: 93.16%
	shake_it_off.wav	Taylor Swift	Match: 91.04%
	cardigan.wav	Taylor Swift	Match: 87.26%
	love_story.wav	Bruno Mars	Match: 87.15%
	leave_the_door_open.wav	Halsey	Match: 86.17%
	apt.wav	Bruno Mars	Match: 84.10%
	bed_bound.wav	Taylor Swift	Match: 81.21%
	die_with_a_smile.wav	Taylor Swift	Match: 81.04%

< Previous 1 Next >

Search by Album

Gambar yang di-query:



Hasil:

Search by Album

Upload Mapper

Upload Dataset

Page untuk mengupload mapper dan dataset,
disertakan runtime upload database.

Upload

{ } Upload Dataset/Mapper

 Drag and Drop || Click to Upload Dataset (ZIP / RAR)

Upload a zip file containing your dataset

 Drag and Drop || Click to Upload Mapper (JSON/TXT)

Upload a JSON or TXT file for dataset mapper

Bab 5: Kesimpulan, Saran, Komentar, dan Refleksi

5.1 Kesimpulan

Pada tugas besar ini kami berhasil mengembangkan sebuah website Music Information Retrieval (MIR) dan Image Retrieval menggunakan metode Principal Component Analysis (PCA). Website ini memungkinkan pengguna untuk mencari lagu berdasarkan terhadap suatu dataset serta mencocokkan gambar melalui fitur vektor piksel yang menggunakan PCA. Dengan menggunakan pengurangan dimensi melalui PCA, proses pencarian menjadi lebih efisien dan akurat.

5.2 Saran

1. Pengembangan Fitur Tambahan: Sistem dapat dikembangkan lebih lanjut dengan menambahkan fitur mencari nama musik aslinya serta nama artis. Namun karena keterbatasan pada implementasi mapper, dan karena dataset cenderung tidak memiliki metadata yang benar, fitur ini menjadi sulit untuk diimplementasikan.
2. Optimasi Performa: Implementasi algoritma dapat lebih dioptimalkan, terutama untuk audio dengan dataset yang lebih besar. Selain itu, pada pemrosesan gambar, gambar di *resize* menjadi sangat kecil yaitu 16x16 pixel saja. Ini membuat algoritma sangat cepat namun mengurangi akurasi.
3. Pengujian Lebih Lanjut: Perlu dilakukan pengujian pada dataset yang lebih beragam dan kompleks untuk memastikan keandalan sistem dalam berbagai skenario pencarian. Selain itu, tipe-tipe *file* lainnya masih belum dapat diimplementasikan, sehingga *input validation* pada website dibutuhkan untuk menghindari *upload* file yang tidak dapat di-*handle* oleh program.

5.3 Komentar

Tugas besar ini memberikan kesempatan kepada kami untuk mempelajari konsep-konsep seperti Music Information Retrieval dan PCA, serta proses pembuatan website secara lebih mendalam. Tugas ini tentunya tidak mudah karena banyaknya hal yang bersifat asing bagi kami; tetapi melalui tugas ini, kami dapat belajar untuk meningkatkan kemampuan berkomunikasi serta bekerja dalam tim.

5.4 Refleksi terhadap Tugas Besar

Pengerjaan tugas besar ini menjadi pengalaman yang cukup berkesan karena adanya konsep-konsep baru seperti MIR dan PCA. Meskipun awalnya terasa menantang dan asing, proses pembelajaran dan implementasi membuka wawasan kami. Tugas besar ini memperkaya pemahaman kami mengenai materi-materi yang diajarkan di kelas, antara lain nilai eigen dan

vektor eigen, SVD, jarak euclidean, dan lainnya. Oleh karena itu, hal ini memungkinkan kami untuk memahami materi lebih dalam karena kami dapat melihat penerapannya dalam kehidupan nyata.

Bab 6: Lampiran

6.1 Referensi

Kosugi, N., Nishihara, Y., Kon'ya, S., Yamamuro, M., & Kushima, K. (1999). *Image Compression Techniques: A Closer Look at Principal Component Analysis*. Medium. <https://ujangriswanto08.medium.com/image-compression-techniques-a-closer-look-at-principal-component-analysis-67cf7a29fdb9>.

Kosugi, N., Nishihara, Y., Kon'ya, S., Yamamuro, M., & Kushima, K. (1999). *Music Retrieval by Humming: Using Similarity Retrieval Over High Dimensional Feature Vector Space*. IEEE International Conference on Multimedia Computing and Systems. <https://ieeexplore.ieee.org/document/799561>.

Schedl, M., Gómez, E., & Urbano, J. (2014). *Music Information Retrieval: Recent Developments and Applications. Foundations and Trends in Information Retrieval*, 8(2-3), 127–261. https://repositori.upf.edu/bitstream/10230/27565/1/Urbano_ftir_mus.pdf.

Smith, J., & Doe, R. (2014). *Principal Component Analysis for Image Processing*. IEEE International Conference on Multimedia Computing and Systems. <https://ieeexplore.ieee.org/document/699557>.

6.2 Link Video

Berikut adalah link video

 [Video Tugas Besar 2 Aljabar Linier dan Geometri](#)