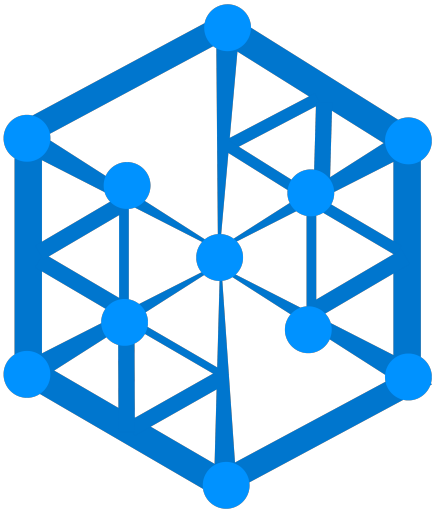
Spesifikasi Tugas Besar II

IF2123 Aljabar Linier dan Geometri

EigenTsar: Sistem Keamanan Biometrik dengan Pengenalan Wajah dan Pembaruan Dataset Dinamis Berbasis PCA-SVD

Semester I Tahun 2026/2027

v1.0
DEADLINE: RABU, 2026/11/25 21:23 WIB



Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

0 Daftar Revisi

- `v1.0` — [2026/11/02 16.00] — Versi rilis awal.

Daftar Isi

0 Daftar Revisi **1**

1 Pendahuluan **2**
1.1 Kisah . . . 2
1.2 Tujuan . . . 2

2 Dasar Teori **3**
2.1 Sistem Pengenalan Wajah Berbasis Eigenfaces . . . 3
2.2 Pengantar Aplikasi Web . . . 8

3 Spesifikasi **11**
3.1 Spesifikasi Wajib Program . . . 11
3.2 Spesifikasi Bonus . . . 14
3.3 Prosedur Pengerjaan . . . 15
3.4 Spesifikasi Laporan . . . 16
3.5 Kriteria Penilaian . . . 17

4 Pengumpulan **18**
4.1 *Deliverables* . . . 18
4.2 Teknis Pengumpulan . . . 18

5 Tautan Penting **19**

6 Referensi **20**

7 Surat Cinta dari Asisten **22**

1 Pendahuluan

1.1 Kisah

Berkat keandalan mesin kalkulasi matriks yang kalian bangun sebelumnya, roda riset Proyek Stuzha di bawah pimpinan Mitya kini melaju tanpa hambatan. Fluktuasi energi ekstrem di daratan Teyvat perlahan berhasil dijinakkan. Namun, keberhasilan besar ini justru memancing bahaya baru yang jauh lebih mengancam yaitu murka Renova. Menganggap Proyek Stuzha sebagai pembangkangan lancang yang merusak tatanan *Heavenly Principles*, ancaman balasan dari langit membuat seluruh lapisan masyarakat Snezhnaya harus merapatkan barisan dan memperkokoh persatuan mereka.

Demi memastikan proyek tetap berjalan di bawah bayang-bayang ancaman tersebut, Ratu kita, Sang Tsaritsa, menurunkan titah baru. Proyek Stuzha tidak boleh lagi terpusat di satu titik, melainkan harus dipecah dan dikerjakan secara terpisah. Tatanan baru ini menuntut satu hal yang paling mendasar dari cita-cita Sang Ratu yaitu rasa cinta yang hanya bisa tumbuh jika mereka saling percaya satu sama lain.



Gambar 1: Komisi dari Vesna
Sumber : Genshin Impact. 2026. Transcendence Trailer.

Namun, di tengah situasi genting yang menuntut solidaritas tinggi ini, selalu saja ada oknum licik yang mencari celah di dalam kesempitan. Alih-alih bersatu, segelintir pengkhianat justru memanfaatkan kekacauan ini untuk menyusup ke fasilitas riset demi mengeruk keuntungan pribadi dari dalam. Sadar bahwa rasa percaya saja tidak cukup tanpa sistem pertahanan yang ketat, petinggi organisasi Druzhna, **Vesna**, segera turun tangan. Ia membutuhkan sebuah alat keamanan canggih yang sanggup mengenali wajah personel secara akurat dari arsip data yang sudah ada, sehingga tidak ada lagi orang asing atau penyusup yang bisa seenaknya mengakses komputer-komputer vital milik Druzhna.

Masih ingat betul betapa handalnya kemampuan kalian, Mahasiswa Informatika yang mendalami IF2123 Aljabar Linier dan Geometri yang telah membantu Mitya dulu, Vesna pun kembali melayangkan komisi lewat Adventurers' Guild. Kali ini, kalian ditantang untuk merancang sistem pengenalan wajah (*Face Recognition*) otomatis berbasis **Eigenfaces** yang ditenagai oleh *Principal Component Analysis* (PCA) dan *Singular Value Decomposition* (SVD).

1.2 Tujuan

Pada Tugas Besar 2 ini, kita akan mengembangkan sebuah aplikasi *web* untuk pengenalan wajah menggunakan metode *Eigenfaces*. Anda akan diberikan *dataset* Georgia Tech Face Database yang berisi citra wajah dari sejumlah subjek. Aplikasi ini harus mampu menampilkan daftar subjek beserta citranya, membentuk ruang *eigenfaces* menggunakan *Principal Component Analysis* (PCA) melalui *Singular Value Decomposition* (SVD), serta mengenali wajah dari citra yang diunggah pengguna berdasarkan jarak Euclidean dalam ruang tersebut. Pengguna juga dapat menambahkan citra atau subjek baru, kemudian aplikasi memperbarui model secara otomatis. Melalui tugas besar ini, Anda diharapkan dapat menerapkan konsep nilai dan vektor eigen, SVD, perubahan basis, serta proyeksi vektor untuk menyelesaikan persoalan pengenalan wajah.

2 Dasar Teori

2.1 Sistem Pengenalan Wajah Berbasis Eigenfaces

Sebuah citra wajah dapat dipandang sebagai sekumpulan nilai intensitas piksel. Dengan demikian, pengenalan wajah dapat dirumuskan sebagai persoalan representasi dan perbandingan vektor. Metode *Eigenfaces* menggunakan *Principal Component Analysis* (PCA) untuk membentuk ruang berdimensi lebih kecil yang menangkap variasi pada kumpulan citra wajah. Citra masukan kemudian diproyeksikan ke ruang tersebut dan dibandingkan dengan citra yang telah terdaftar.

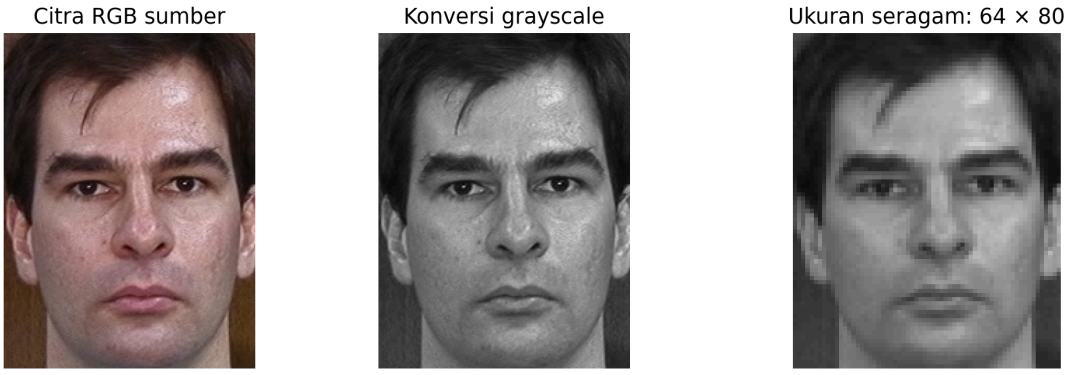
Pada bagian ini, setiap citra direpresentasikan sebagai vektor kolom. Misalkan terdapat N citra *training*, masing-masing berukuran $h \times w$ piksel setelah prapemrosesan. Banyak fitur pada setiap citra adalah $d = hw$. Notasi ini digunakan secara konsisten pada seluruh persamaan.

2.1.1 Persiapan Data Citra

Tiap piksel citra RGB memiliki komponen merah, hijau, dan biru. Pada citra 8 bit per kanal, intensitas setiap komponen berada pada rentang 0 hingga 255. Citra diubah menjadi skala abu-abu atau *grayscale* agar setiap piksel cukup direpresentasikan oleh satu nilai intensitas. Salah satu rumus konversi yang umum digunakan adalah

$$I = 0.299R + 0.587G + 0.114B.$$

Setelah itu, ukuran seluruh citra diseragamkan. Pilihan ukuran dan metode interpolasi saat pengubahan ukuran harus digunakan secara konsisten. Urutan kanal warna yang dibaca pustaka juga perlu diperhatikan, karena urutan RGB dan BGR berbeda.



Gambar 2: Citra RGB, hasil konversi *grayscale*, dan hasil penyeragaman ukuran. Citra bersumber dari bagian *training* Georgia Tech Face Database. Ukuran 64×80 hanya digunakan untuk ilustrasi, bukan ketentuan ukuran wajib.

Misalkan citra *grayscale* direpresentasikan oleh matriks $I \in \mathbb{R}^{h \times w}$. Vektor $x \in \mathbb{R}^d$ dibentuk dengan menggabungkan kolom-kolom matriks secara berurutan. Sebagai contoh,

$$I = \begin{bmatrix} 10 & 40 \\ 20 & 50 \\ 30 & 60 \end{bmatrix} \quad \longrightarrow \quad x = \begin{bmatrix} 10 & 20 & 30 & 40 & 50 & 60 \end{bmatrix}^T.$$

Proses ini disebut vektorisasi atau *flatten*. Seluruh citra *training* kemudian disusun menjadi

$$X = \begin{bmatrix} x_1 \mid x_2 \mid \cdots \mid x_N \end{bmatrix} \in \mathbb{R}^{d \times N}.$$

Identitas subjek disimpan sebagai label ℓ_i yang berpasangan dengan kolom x_i. Citra *training*, *validation*, dan kueri harus menggunakan ukuran, konversi warna, skala intensitas, dan urutan vektorisasi yang sama. Apabila intensitas dibagi 255 sehingga berada pada rentang $[0, 1]$, pembagian tersebut juga diterapkan pada seluruh citra. Semua operasi pengurangan rata-rata selanjutnya dilakukan menggunakan bilangan pecahan, karena hasilnya dapat bernilai negatif.

2.1.2 Pemusatan Data

Citra rata-rata atau *mean face* diperoleh dengan merata-ratakan setiap posisi piksel pada seluruh citra *training*, yaitu

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \in \mathbb{R}^d.$$

Setiap citra kemudian dipusatkan melalui *mean-centering*,

$$a_i = x_i - \mu, \qquad A = \begin{bmatrix} a_1 \mid a_2 \mid \cdots \mid a_N \end{bmatrix} \in \mathbb{R}^{d\times N}.$$

Akibatnya, rata-rata setiap baris A mendekati nol. Sebagai contoh, jika dua vektor citra adalah $x_1 = [10, 20]^T$ dan $x_2 = [30, 40]^T$, diperoleh $\mu = [20, 30]^T$, $a_1 = [-10, -10]^T$, dan $a_2 = [10, 10]^T$.

Pemusatan membuat analisis berfokus pada variasi terhadap citra rata-rata. *Mean-centering* tidak menyamakan variansi setiap piksel dan tidak dilakukan dengan mengurangi rata-rata piksel masing-masing citra secara terpisah. Vektor μ hanya dihitung dari data *training*. Citra *validation* dan kueri menggunakan vektor rata-rata tersebut.

2.1.3 Nilai Eigen dan Vektor Eigen

Untuk matriks persegi B, sebuah vektor tak nol v disebut vektor eigen apabila

$$Bv = \lambda v.$$

Nilai λ disebut nilai eigen. Persamaan tersebut menyatakan bahwa transformasi B memetakan v menjadi kelipatan dirinya sendiri. Pada matriks simetris riil, basis vektor eigen dapat dipilih ortonormal, sehingga

$$B = Q\Lambda Q^T, \qquad Q^T Q = I.$$

Sifat ini digunakan untuk memproses matriks simetris seperti $A^T A$. Nilai dan vektor eigen matriks tersebut dapat dihitung menggunakan metode iteratif. Salah satu pilihannya adalah metode rotasi Jacobi, yang berbeda dari metode iterasi Jacobi untuk menyelesaikan SPL.

Secara garis besar, rotasi Jacobi dapat dilakukan sebagai berikut.

1. Inisialisasi $D = B$ dan $Q = I$.
2. Pilih pasangan indeks $p \neq q$ dengan elemen luar diagonal D_{pq} yang belum cukup kecil. Jika elemen tersebut sudah nol, pasangan dapat dilewati.
3. Hitung sudut dan koefisien rotasi

$$\theta = \frac{1}{2}\operatorname{atan2}(2D_{pq}, D_{qq} - D_{pp}), \qquad c = \cos\theta, \qquad s = \sin\theta.$$

4. Bentuk transformasi J yang sama dengan identitas, kecuali blok baris dan kolom p, q, yaitu

$$J_{\{p,q\},\{p,q\}} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}.$$

5. Perbarui $D \leftarrow J^T D J$ dan $Q \leftarrow QJ$, lalu ulangi hingga elemen luar diagonal cukup kecil atau batas iterasi tercapai.

Salah satu kriteria penghentian adalah

$$\sqrt{\sum_{p\neq q} D_{pq}^2} \leq \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}}\|B\|_F,$$

dengan $\|B\|_F = \sqrt{\sum_{i,j} B_{ij}^2}$. Diagonal akhir D memberikan pendekatan nilai eigen, sedangkan kolom Q memberikan vektor eigen yang bersesuaian. Untuk efisiensi, pembaruan cukup dilakukan pada baris dan kolom yang terlibat, tanpa membentuk dan mengalikan matriks J penuh setiap kali. Program perlu memeriksa konvergensi, bukan hanya berhenti pada jumlah iterasi tertentu.

Uraian ini merupakan salah satu pilihan algoritma, bukan kewajiban menggunakan rotasi Jacobi. Algoritma lain tetap dapat digunakan sepanjang memenuhi batasan implementasi pada spesifikasi.

2.1.4 Singular Value Decomposition (SVD)

Singular Value Decomposition memfaktorkan matriks A menjadi

$$A = U\Sigma V^T.$$

Dalam bentuk penuh, ukuran U adalah $d \times d$, ukuran Σ adalah $d \times N$, dan ukuran V adalah $N \times N$. Kolom U dan V ortonormal. Elemen diagonal Σ merupakan nilai singular tak negatif yang diurutkan dari terbesar ke terkecil.

Jika $r = \text{rank}(A)$, bentuk ringkasnya adalah

$$A = U_r\Sigma_r V_r^T, \qquad U_r \in \mathbb{R}^{d\times r}, \quad \Sigma_r \in \mathbb{R}^{r\times r}, \quad V_r \in \mathbb{R}^{N\times r}.$$

Hubungan SVD dengan dekomposisi eigen diberikan oleh

$$A^TAv_i = \sigma_i^2 v_i, \qquad AA^Tu_i = \sigma_i^2 u_i.$$

Pada citra, umumnya d jauh lebih besar daripada N. Karena itu, membentuk AA^T berukuran $d \times d$ dapat memerlukan memori yang besar. Alternatifnya, bentuk matriks Gram

$$G = A^TA \in \mathbb{R}^{N\times N}.$$

Setelah memperoleh pasangan (γ_i, v_i) dari $Gv_i = \gamma_i v_i$, hitung

$$\sigma_i = \sqrt{\gamma_i}, \qquad u_i = \frac{Av_i}{\sigma_i}, \qquad \gamma_i > 0.$$

Sebagai ilustrasi, jika citra berukuran 64×80 dan terdapat 550 citra *training*, AA^T berukuran 5120×5120, sedangkan G hanya berukuran 550×550. Tidak perlu membentuk kedua matriks tersebut sekaligus.

Nilai eigen γ_i diurutkan secara menurun, kemudian urutan yang sama diterapkan pada vektor v_i, nilai σ_i, dan vektor u_i. Nilai singular nol atau terlalu kecil tidak digunakan sebagai pembagi. Dalam komputasi numerik, nilai eigen negatif yang sangat kecil akibat pembulatan dapat dianggap nol sesuai toleransi, tetapi nilai negatif yang besar perlu diperiksa sebagai indikasi masalah perhitungan.

Pendekatan matriks Gram menghemat ukuran persoalan eigen, tetapi dapat mengurangi ketelitian nilai singular yang sangat kecil. Ketelitian perhitungan diperiksa menggunakan toleransi yang sesuai, dengan memastikan bahwa $U_r^TU_r \approx I$ dan $A \approx U_r\Sigma_r V_r^T$. Tidak diperlukan penyimpanan matriks U penuh hanya untuk mengambil sejumlah kecil komponen.

2.1.5 Principal Component Analysis (PCA)

PCA mencari arah ortogonal yang menangkap variansi terbesar dari data yang telah dipusatkan. Arah pertama menangkap variansi terbesar, arah kedua menangkap variansi terbesar berikutnya dengan syarat ortogonal terhadap arah pertama, dan demikian seterusnya.

Dengan konvensi pada bagian ini, matriks kovariansi didefinisikan sebagai

$$C = \frac{1}{N}AA^T.$$

Dari SVD, diperoleh

$$C = U_r\frac{\Sigma_r^2}{N}U_r^T.$$

Dengan demikian, kolom U_r merupakan arah komponen utama, sedangkan variansi pada arah ke-i adalah $\lambda_i = \sigma_i^2/N$. Beberapa referensi menggunakan pembagi $N-1$. Perbedaan pembagi tersebut mengubah skala nilai eigen kovariansi, tetapi tidak mengubah arah komponen utama atau urutannya.

Untuk mengambil k komponen teratas, dibentuk

$$U_k = \begin{bmatrix} u_1 \mid u_2 \mid \cdots \mid u_k \end{bmatrix} \in \mathbb{R}^{d\times k}, \qquad U_k^TU_k = I_k.$$

Karena $\sum_i a_i = 0$, kolom-kolom A bergantung linear. Oleh sebab itu,

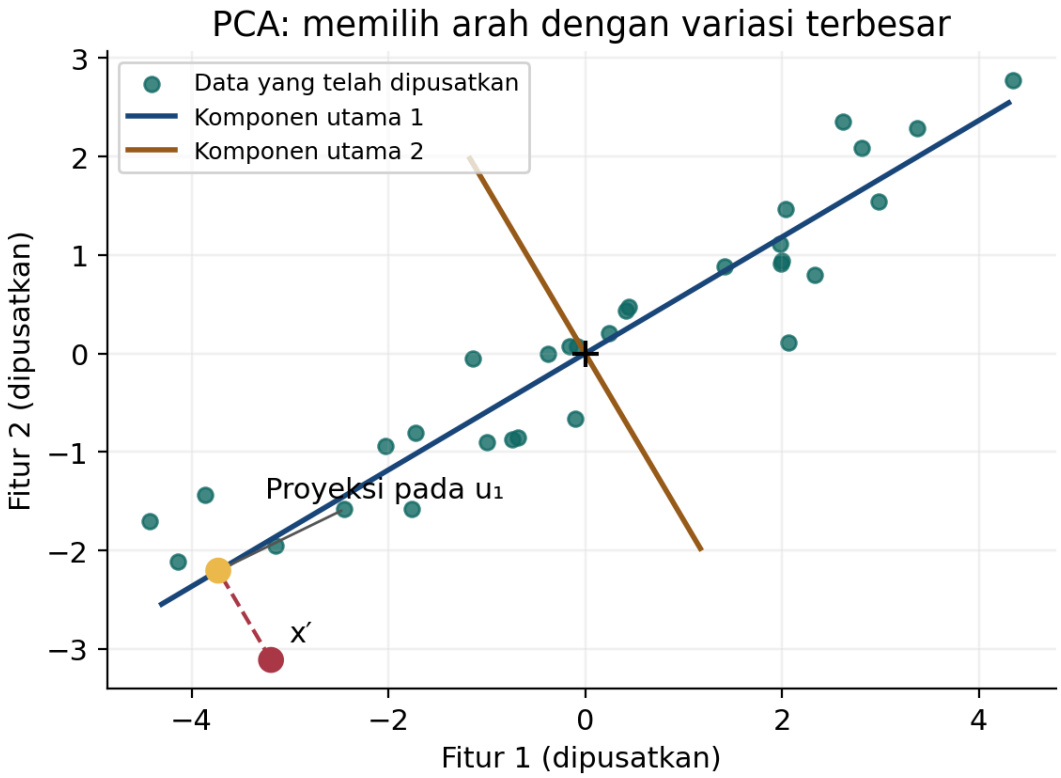
$$1 \le k \le r \le \min(d, N-1),$$

dengan r pada implementasi diperkirakan berdasarkan toleransi numerik. Jika seluruh citra identik sehingga $A = 0$, tidak tersedia komponen utama dengan variansi positif dan program perlu memberi pesan yang sesuai.

Proporsi variansi yang dipertahankan dapat dihitung sebagai

$$\eta(k) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

Sebagai contoh, nilai singular $4, 2, 1$ memberikan $\eta(2) = 20/21 \approx 95.24\%$. Persentase ini menyatakan variansi yang dipertahankan, bukan akurasi pengenalan wajah. Nilai k tetap perlu dievaluasi menggunakan data *validation* karena penambahan komponen tidak selalu meningkatkan ketepatan identifikasi.



Gambar 3: Ilustrasi PCA pada data sintetis dua dimensi. Proyeksi ke komponen pertama mempertahankan arah variasi terbesar dan mengabaikan variasi pada arah tegak lurusnya. Ilustrasi dibuat untuk menjelaskan konsep PCA, bukan hasil pengujian wajah.

2.1.6 Pembentukan dan Visualisasi Eigenfaces

Setiap kolom u_i mempunyai d elemen sehingga dapat dibentuk kembali menjadi citra berukuran $h \times w$, menggunakan kebalikan urutan vektorisasi. Citra tersebut disebut *eigenface*. *Eigenface* merupakan pola perubahan terhadap *mean face*, bukan foto seorang subjek tertentu.

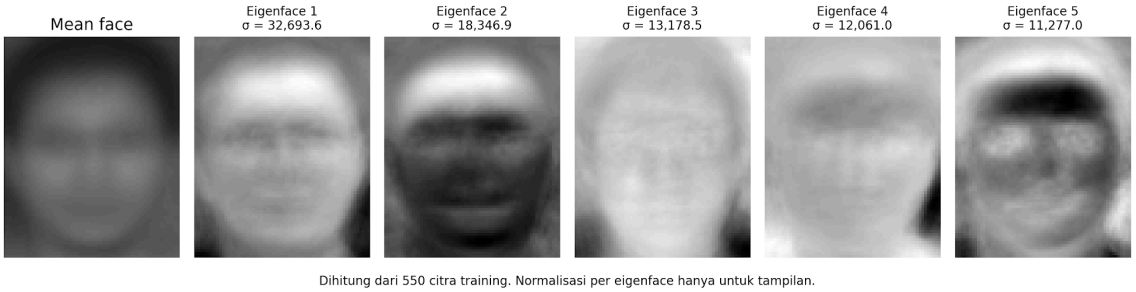
Representasi citra dapat didekati dengan kombinasi linear

$$x_i \approx \mu + \sum_{j=1}^{k} c_{ij} u_j, \qquad c_i = U_k^T (x_i - \mu).$$

Elemen u_i dapat bernilai negatif. Untuk menampilkan *eigenface* sebagai citra, salah satu pilihan normalisasi adalah

$$\widetilde{u}_i[p] = 255 \, \frac{u_i[p] - \min(u_i)}{\max(u_i) - \min(u_i)},$$

dengan p sebagai indeks piksel. Jika penyebut bernilai nol, citra ditampilkan dengan intensitas konstan. Normalisasi tampilan hanya diterapkan pada salinan untuk visualisasi. Basis perhitungan tetap menggunakan nilai asli yang ortonormal.



Gambar 4: *Mean face* dan lima *eigenfaces* teratas yang dihitung dari 550 citra *training* Georgia Tech, setelah diubah menjadi *grayscale* dan berukuran 64×80. *Eigenfaces* diurutkan berdasarkan nilai singular, dengan normalisasi tampilan terpisah untuk setiap citra.

Vektor eigen dapat memiliki tanda yang berlawanan pada dua implementasi. Pembalikan tanda yang diterapkan konsisten pada suatu basis dan seluruh proyeksi tidak mengubah jarak Euclidean. Apabila nilai eigen berulang, basis ortonormal pada subruang yang bersesuaian juga dapat berbeda. Karena itu, tampilan *eigenface* tidak harus identik secara piksel untuk menunjukkan perhitungan yang benar.

2.1.7 Proyeksi dan Pencocokan Wajah

Setelah basis diperoleh, seluruh citra *training* diproyeksikan menjadi

$$Z = U_k^T A \in \mathbb{R}^{k \times N}.$$

Kolom c_i pada Z merupakan koordinat citra ke-i dalam ruang *eigenfaces*. Untuk citra kueri x_q, hitung

$$a_q = x_q - \mu, \qquad c_q = U_k^T a_q \in \mathbb{R}^k.$$

Rata-rata dan basis untuk kueri harus berasal dari model *training* yang sama. Jarak terhadap citra *training* ke-i dihitung dengan

$$d_i = \|c_q - c_i\|_2 = \sqrt{\sum_{j=1}^{k} (c_{qj} - c_{ij})^2}.$$

Citra pembanding terdekat dan identitas prediksi ditentukan sebagai

$$i^* = \arg\min_{1 \le i \le N} d_i, \qquad \widehat{\ell} = \ell_{i^*}.$$

Pada tugas besar ini, pembandingan dilakukan terhadap setiap citra *training*, kemudian label citra terdekat digunakan sebagai identitas prediksi. Ini merupakan pilihan pencocokan dalam spesifikasi, bukan kewajiban menggunakan rata-rata proyeksi per subjek.

Sebagai contoh, jika $c_q = [2, 1]^T$, $c_1 = [1, 1]^T$, dan $c_2 = [5, 5]^T$, diperoleh $d_1 = 1$ dan $d_2 = 5$. Dengan demikian, label citra pertama dipilih. Apabila jaraknya sama, pemilihan dilakukan menggunakan aturan yang deterministik dan didokumentasikan. Jarak ini bukan probabilitas atau persentase keyakinan.

Akurasi pengenalan pada data *validation* dihitung sebagai

$$\text{akurasi} = \frac{\text{jumlah kueri } \textit{validation} \text{ dengan identitas benar}}{\text{jumlah seluruh kueri } \textit{validation}} \times 100\%.$$

Pengujian dilakukan tanpa memasukkan citra *validation* ke dalam model *training*. Ketika membandingkan beberapa nilai k, gunakan kumpulan data dan batas pengukuran waktu yang sama. Waktu pembentukan model dibedakan dari waktu pengenalan sebuah kueri agar hasil perbandingan dapat dijelaskan.

2.1.8 Pembaruan Model

Penambahan citra mengubah kumpulan data yang digunakan untuk membentuk model. Jika terdapat m citra baru $z_1, \ldots, z_m$, rata-rata baru adalah

$$\mu_{\text{baru}} = \frac{N\mu + \sum_{j=1}^{m} z_j}{N + m}.$$

Selanjutnya, bentuk kembali matriks data terpusat dari seluruh citra *training*, hitung SVD, pilih basis yang sesuai, dan proyeksikan kembali seluruh citra. Langkah ini mengikuti langsung perubahan A, U_k, dan Z pada persamaan sebelumnya.

Proyeksi lama tidak boleh langsung digabungkan dengan proyeksi model baru, karena rata-rata atau basisnya dapat berbeda. Rata-rata, basis, nilai singular, proyeksi, label, parameter prapemrosesan, dan nilai k disimpan sebagai satu versi model yang konsisten. Model baru digunakan untuk pengenalan setelah seluruh perhitungannya selesai.

Perhitungan ulang dilakukan ketika citra ditambahkan melalui fitur penambahan *dataset*. Mengunggah citra sebagai kueri pengenalan tidak secara otomatis menambah data *training*. Jika parameter prapemrosesan berubah, citra sumber perlu diproses kembali sebelum pembentukan model.

2.1.9 Penolakan Wajah Tidak Dikenal (Bonus)

Tanpa mekanisme penolakan, pencarian tetangga terdekat akan memilih salah satu identitas terdaftar untuk setiap kueri. Bonus ini menambahkan ambang τ terhadap jarak terdekat. Misalkan

$$d_{\min} = \min_i d_i.$$

Keputusan pengenalan adalah

$$\text{hasil}(x_q) = \begin{cases} \ell_{i^*}, & d_{\min} \le \tau, \\ \text{wajah tidak dikenali}, & d_{\min} > \tau. \end{cases}$$

Ambang yang terlalu kecil dapat menolak wajah subjek yang sebenarnya terdaftar. Sebaliknya, ambang yang terlalu besar dapat menerima wajah dari identitas yang belum terdaftar. Nilai ambang dapat dipilih dengan mempertimbangkan citra *validation* untuk identitas terdaftar dan citra identitas lain yang disiapkan sendiri oleh kelompok. Citra identitas lain tersebut tidak dimasukkan ke data *training*.

Evaluasi dapat melaporkan proporsi kueri dikenal yang diidentifikasi dengan benar dan proporsi kueri tidak dikenal yang berhasil ditolak. Sebagai contoh, jika 18 dari 20 wajah tidak dikenal ditolak, tingkat penolakan pada kumpulan tersebut adalah 90%. Kedua jenis kueri perlu diuji agar ambang tidak tampak baik hanya karena program menolak seluruh masukan.

Nilai ambang bergantung pada skala intensitas, ukuran citra, jumlah komponen, dan model yang digunakan. Ambang perlu dievaluasi kembali apabila konfigurasi tersebut berubah. Mekanisme ini merupakan penolakan identitas berdasarkan jarak, bukan jaminan bahwa masukan merupakan citra wajah.

2.2 Pengantar Aplikasi Web

2.2.1 Arsitektur Client–Server

Aplikasi web dibangun berdasarkan arsitektur *client–server*, yaitu model komunikasi di mana terdapat dua pihak utama yang saling berinteraksi:

- **Client**: Perangkat pengguna yang mengakses aplikasi melalui *web browser*. Klien menampilkan antarmuka pengguna dan mengirimkan permintaan ke server.
- **Server**: Program yang berjalan di suatu tempat, baik di internet maupun jaringan lokal, yang menunggu permintaan dari klien, memproses permintaan tersebut, kemudian mengirimkan hasil kembali.

Komunikasi antara client dan server mengikuti pola *request–response*. Klien mengirim permintaan (*request*), server memprosesnya, lalu mengirim respons (*response*) kembali.

2.2.2 HTTP dan Metodenya

Lalu, bagaimana cara client dan server "berbicara"? Jawabannya adalah dengan menggunakan protokol bernama HTTP (*HyperText Transfer Protocol*). HTTP mendefinisikan format dan aturan pertukaran pesan antara client dan server. Setiap kali kalian membuka halaman web, menekan tombol login, atau mengirim formulir, peramban kalian sebenarnya sedang mengirimkan HTTP request ke server.

Struktur HTTP Request Sebuah permintaan HTTP biasanya berisi:

- **Method**: Tindakan yang ingin dilakukan, misalnya GET atau POST.
- **URL atau endpoint**: Alamat tujuan di server.
- **Headers**: Metadata tambahan, seperti tipe konten atau informasi autentikasi.
- **Body (opsional)**: Data yang dikirimkan ke server, biasanya dalam format JSON.

Contoh sederhana:

```
POST /api/login HTTP/1.1
Content-Type: application/json

{ "email": "asep@gmail.com", "password": "rahasia" }
```

Struktur HTTP Response Server kemudian merespons permintaan tersebut dengan pesan HTTP yang berisi:

- **Status code**: Menunjukkan hasil pemrosesan, misalnya 200 OK atau 404 Not Found.
- **Headers**: Informasi tambahan, seperti jenis data dan panjang konten.
- **Body**: Data hasil pemrosesan, sering kali dalam format JSON atau HTML.

Contoh respons:

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "status": "success", "token": "abc123xyz" }
```

HTTP memiliki beberapa metode yang menunjukkan tujuan dari sebuah permintaan. Berikut beberapa yang paling umum:

- **GET**: Digunakan untuk mengambil data dari server, misalnya ketika peramban memuat halaman web.
- **POST**: Digunakan untuk mengirim data ke server, misalnya ketika mengisi formulir pendaftaran.
- **PUT**: Digunakan untuk memperbarui data yang sudah ada di server.
- **DELETE**: Digunakan untuk menghapus data di server.

Dengan metode ini, komunikasi client–server menjadi lebih terstruktur dan mudah dipahami. Misalnya, ketika frontend mengirimkan:

```
POST /api/login
```

Server dapat memprosesnya sebagai permintaan untuk login pengguna sesuai dengan rute yang didefinisikan pada aplikasi.

Status Code Umum Selain metode, penting juga memahami kode status HTTP yang dikembalikan server:

- **200 OK**: Permintaan berhasil.
- **201 Created**: Data baru berhasil dibuat.
- **400 Bad Request**: Format permintaan salah atau data tidak valid.

- **401 Unauthorized**: Permintaan tidak memiliki kredensial autentikasi yang valid, misalnya karena pengguna belum login.
- **404 Not Found**: Sumber daya tidak ditemukan.
- **500 Internal Server Error**: Terjadi kesalahan di sisi server.

Dengan memahami struktur request, response, dan kode status, kalian bisa lebih mudah membaca atau menelusuri kesalahan komunikasi antara frontend dan backend.

2.2.3 Frontend dan Backend

Aplikasi web modern dibagi menjadi dua bagian utama:

Frontend (Sisi Klien) Bagian yang berjalan di peramban pengguna, terdiri atas:

- **HTML**: Struktur dan konten halaman.
- **CSS**: Tampilan visual halaman.
- **JavaScript**: Logika interaktif dan komunikasi dengan backend.

Backend (Sisi Server) Bagian yang berjalan di server, bertanggung jawab atas logika bisnis dan pemrosesan data. Backend menerima permintaan dari frontend melalui API, melakukan komputasi yang diperlukan, lalu mengembalikan hasil ke frontend.

2.2.4 Format Komunikasi JSON

Komunikasi antara frontend dan backend biasanya menggunakan format JSON (*JavaScript Object Notation*). JSON adalah format data ringan yang mudah dibaca manusia dan mudah diurai atau di-*parse* oleh program. Objek JSON disusun dari pasangan *key–value*. Perhatikan contoh berikut.

1. Frontend mengirim request JSON ke backend untuk mencari data pengguna dengan nama “asep”:

```
{ "name": "asep" }
```

2. Backend memproses request dan mengembalikan response JSON berupa daftar pengguna yang sesuai:

```
{
  "status": "success",
  "results": [
    { "id": 1, "name": "Asep", "age": 25 },
    { "id": 3, "name": "Asep Surasep", "age": 30 }
  ],
  "total": 2
}
```

3 Spesifikasi

3.1 Spesifikasi Wajib Program

Secara ringkas, luaran minimum yang diharapkan pada tugas besar ini adalah:

1. Buatlah aplikasi pengenalan wajah berbasis *web* menggunakan metode *Eigenfaces*. Aplikasi harus mampu mengolah *dataset* wajah, membentuk ruang *eigenfaces* menggunakan *Principal Component Analysis* (PCA) dengan *Singular Value Decomposition* (SVD), mengenali wajah berdasarkan gambar yang diunggah, serta menambahkan data wajah dan memperbarui model secara otomatis. Aplikasi ini menggunakan *framework frontend* berbasis JavaScript/TypeScript. Bahasa dan *framework backend* dibebaskan.
2. Laporan mengenai aplikasi *web* yang dibangun beserta teori-teori yang mendasari pembuatannya.
3. Video berdurasi maksimal 30 menit yang memuat penjelasan struktur dan implementasi program, demonstrasi fitur, serta jawaban atas pertanyaan analisis pada Subbab 3.1.4.

3.1.1 Batasan Umum Tugas Besar

1. Bahasa pemrograman yang digunakan adalah JavaScript/TypeScript untuk *frontend*, sedangkan bahasa pemrograman untuk *backend* dibebaskan. Anda dapat menggunakan Python, Java, Go, Rust, atau bahasa lainnya yang mendukung kebutuhan aplikasi.
2. *Framework frontend* dan *backend* yang digunakan dibebaskan. *Framework frontend* harus berbasis JavaScript/TypeScript, misalnya React, Vue, Angular, Svelte, dan sebagainya.
3. Semua pustaka yang tersedia untuk bahasa pemrograman yang digunakan diperbolehkan, kecuali pustaka yang menggantikan implementasi perhitungan matriks tingkat lanjut dan inti algoritma *Eigenfaces*, dengan ketentuan sebagai berikut.
   - Seluruh perhitungan *eigenvalue* dan *eigenvector* serta implementasi SVD dan PCA harus diimplementasikan sendiri pada bahasa pemrograman yang digunakan. Ketentuan ini juga berlaku terhadap algoritma dekomposisi pendukung yang digunakan dalam perhitungan tersebut.
   - Perhitungan *mean face*, *mean-centering*, pembentukan *eigenfaces*, proyeksi wajah, pengukuran jarak, dan penentuan hasil pengenalan harus diimplementasikan sendiri. Pustaka operasi dasar boleh digunakan sebagai bagian dari implementasi perhitungan tersebut.
   - Pustaka yang melibatkan operasi dasar matriks, seperti penjumlahan, pengurangan, perkalian, dan transpos, diperbolehkan.
   - Pustaka untuk membaca, menulis, dan memanipulasi gambar, seperti OpenCV, Pillow, dan pustaka sejenis, diperbolehkan.
   - Fungsi siap pakai untuk PCA, SVD, *eigendecomposition*, dan pengenalan wajah, seperti `sklearn.decomposition.PCA`, `numpy.linalg.svd`, `numpy.linalg.eigh`, dan `cv2.face.EigenFaceRecognizer`, tidak diperbolehkan. Larangan ini berlaku pula terhadap fungsi setara pada bahasa atau pustaka lain.
   - Model atau *embedding* wajah *pretrained* yang menggantikan representasi *Eigenfaces* tidak diperbolehkan.
   - Silakan konfirmasi melalui QnA apabila terdapat keraguan mengenai pustaka yang diperbolehkan.
4. *Dataset* yang digunakan adalah Georgia Tech Face Database versi *cropped*, yang memuat 50 subjek dengan 15 citra untuk setiap subjek. *Dataset* dibagi oleh asisten sebagai berikut:
   - *Train*: 11 citra per subjek, dengan total 550 citra.
   - *Validation*: 2 citra per subjek, dengan total 100 citra.
   - *Test*: 2 citra per subjek, dengan total 100 citra.

   Dataset train dan *validation* dapat diakses melalui *dataset*. *Dataset test* disimpan oleh asisten untuk pengujian *black box*.

5. *Dataset train* digunakan untuk menghitung *mean face*, membentuk basis *eigenfaces*, dan memperoleh vektor proyeksi citra *training*. *Dataset validation* digunakan untuk mengevaluasi hasil dan memilih parameter program. Citra *validation* tidak boleh dimasukkan ke dalam data *training* atau digunakan untuk membentuk model.

6. Pengolahan citra harus mengikuti tahapan yang dijelaskan pada Bab 2. Citra *training*, *validation*, dan kueri harus diproses secara konsisten, termasuk dalam hal ukuran, konversi *grayscale*, skala intensitas, serta urutan *flatten*. Citra sumber memiliki ukuran yang bervariasi sehingga program perlu menyeragamkan ukurannya sebelum membentuk matriks data.

7. Model dibentuk atau dimuat pada saat inisialisasi aplikasi *web*. Hasil prapemrosesan, seperti *mean face*, *eigenfaces*, dan proyeksi *dataset*, boleh disimpan dalam berkas untuk digunakan kembali. Program tetap harus dapat membentuk model dari *dataset* mentah. Data yang disimpan harus diperbarui apabila *dataset* atau parameter pembentukan model berubah.

8. Citra yang digunakan sebagai kueri pengenalan tidak boleh secara otomatis dimasukkan ke dalam data *training*. Penambahan data *training* hanya dilakukan melalui fitur penambahan *dataset*. Proses pengenalan menggunakan *mean face* dan basis *eigenfaces* yang diperoleh dari data *training*.

9. Aplikasi *web* harus mampu menangani kesalahan dan menampilkan pesan yang informatif. Program tidak boleh berhenti secara tidak terkendali ketika menerima masukan yang tidak valid. Pesan kesalahan harus menjelaskan masalah yang terjadi, misalnya berkas gambar tidak dapat dibaca, jumlah komponen tidak valid, atau model belum siap digunakan.

10. Aplikasi harus dapat dijalankan secara lokal oleh asisten berdasarkan petunjuk pada `README.md`. Seluruh kebutuhan perangkat lunak, versi *runtime*, dependensi, serta perintah untuk menginstal dependensi, melakukan *build*, dan menjalankan program harus didokumentasikan dengan jelas.

3.1.2 Rincian Umum Program

Aplikasi *web* harus memenuhi kebutuhan fungsional berikut.

1. Halaman utama menampilkan daftar subjek pada *dataset* beserta identitas dan citra wajahnya. Bentuk penyajian dan jumlah citra per halaman dibebaskan.

2. Program membaca *dataset* beserta identitas subjeknya, melakukan prapemrosesan, mengubah setiap citra menjadi vektor kolom, kemudian menyusunnya menjadi matriks data.

3. Program menghitung *mean face*, melakukan *mean-centering*, serta membentuk basis *eigenfaces* menggunakan PCA dengan SVD sesuai dengan dasar teori pada Bab 2. Jumlah komponen utama k dapat diatur melalui antarmuka atau konfigurasi program dan harus ditampilkan atau didokumentasikan. Program harus menangani nilai k yang tidak valid.

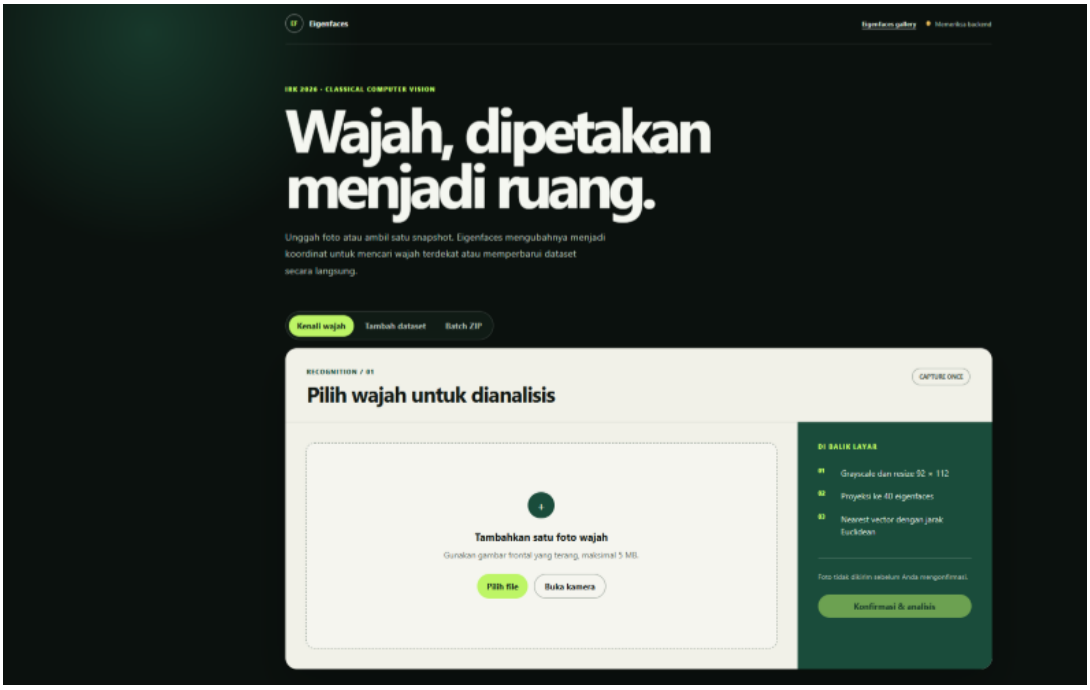
4. Citra *training* dan kueri diproyeksikan ke ruang *eigenfaces* berdimensi k. Proyeksi kueri menggunakan *mean face* dan basis *eigenfaces* dari data *training*.

5. Terdapat fitur untuk mengunggah satu gambar wajah berformat PNG atau JPEG. Gambar ditampilkan kembali sebelum pengenalan dilakukan. Pengujian menggunakan citra wajah yang telah dipotong. Deteksi wajah otomatis tidak diwajibkan.

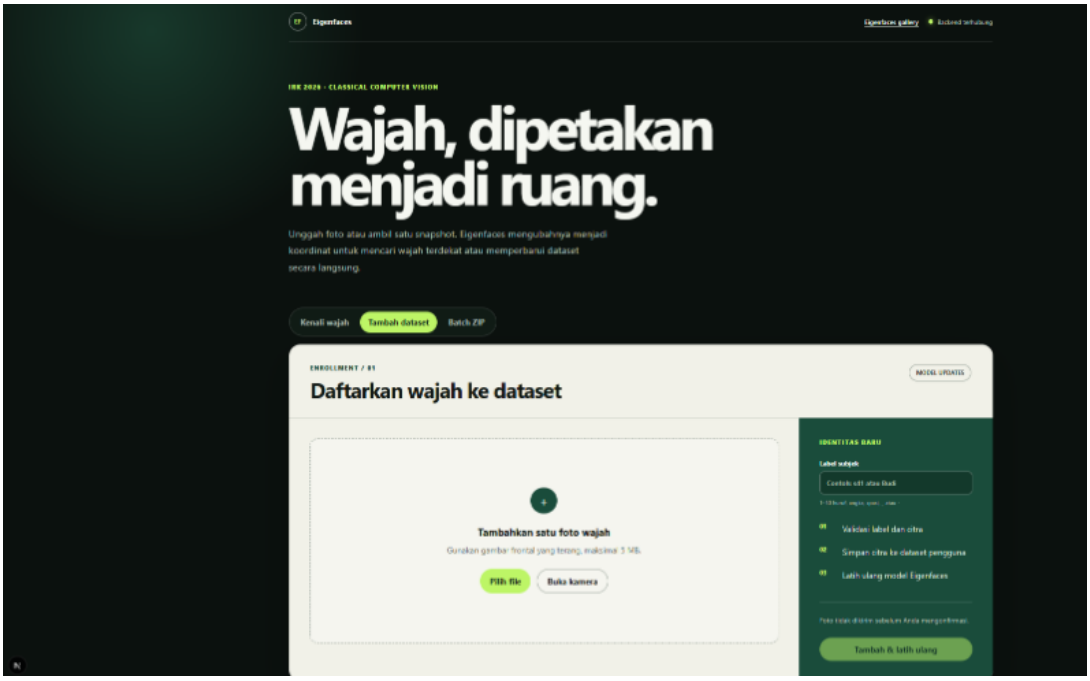
6. Program menentukan citra *training* terdekat berdasarkan jarak Euclidean antara vektor proyeksinya dan vektor proyeksi kueri. Keluaran menampilkan gambar kueri, identitas subjek hasil pencocokan, citra pembanding, dan nilai jaraknya. Aturan pemilihan hasil dengan jarak yang sama harus konsisten dan didokumentasikan pada `README.md`.

7. Pengguna dapat menambahkan citra untuk subjek yang sudah terdaftar maupun subjek baru melalui antarmuka. Program otomatis memperbarui *mean face*, *eigenfaces*, dan proyeksi *dataset* tanpa memulai ulang *server*. Antarmuka menampilkan status pembaruan, dan model terbaru digunakan pada pengenalan berikutnya.

Berikut adalah contoh tampilan antarmuka aplikasi *web* yang dapat digunakan sebagai referensi. Tampilan dibebaskan. Fitur wajib tetap mengikuti ketentuan pada Subbab 3.1.2.



Gambar 5: Contoh antarmuka pengenalan wajah.



Gambar 6: Contoh antarmuka penambahan *dataset*.

3.1.3 Video Penjelasan dan Demonstrasi Program

Setiap kelompok wajib mengumpulkan satu video dengan durasi maksimal 30 menit. Video harus memuat seluruh bagian berikut.

1. Jelaskan arsitektur aplikasi, pembagian modul, serta hubungan antara *frontend* dan *backend*. Tunjukkan alur pemrosesan sejak citra diterima hingga hasil pengenalan ditampilkan.
2. Tunjukkan dan jelaskan bagian kode yang mengimplementasikan representasi *dataset*, *mean-centering*, SVD dan pembentukan *eigenfaces*, pemilihan komponen utama, proyeksi, perhitungan jarak, serta pembaruan model ketika *dataset* bertambah.

3. Demonstrasikan pengenalan beberapa citra wajah, penanganan masukan yang tidak valid, serta penambahan citra atau subjek baru yang diikuti pembaruan model dan pengenalan menggunakan model tersebut.
4. Apabila mengerjakan fitur bonus, kelompok wajib menunjukkan cara penggunaan dan hasilnya dalam video.
5. Jawab seluruh pertanyaan analisis pada Subbab 3.1.4. Penjelasan harus dikaitkan dengan implementasi dan hasil eksperimen kelompok.
6. Seluruh anggota kelompok harus berpartisipasi dalam penjelasan video. Tautan video dicantumkan pada `README.md` dan laporan, serta harus dapat diakses oleh asisten tanpa permintaan izin tambahan. Penilaian dilakukan berdasarkan video dan pengujian program secara *black box* oleh asisten.

3.1.4 Pertanyaan Analisis

Jawab pertanyaan-pertanyaan berikut dalam video penjelasan dan demonstrasi program. Sertakan hasil eksperimen atau bagian implementasi yang relevan untuk mendukung jawaban Anda.

1. Mengapa *mean-centering* perlu dilakukan sebelum membentuk *eigenfaces*? Jelaskan pengaruhnya terhadap representasi data dan apa yang dapat terjadi apabila tahap tersebut dilewati.
2. Apa makna *eigenfaces* dan *singular value* dalam representasi citra wajah? Jelaskan bagaimana hasil SVD digunakan untuk membentuk basis *eigenfaces* dan menentukan komponen utama yang dipertahankan pada program Anda.
3. Lakukan pengujian menggunakan sekurang-kurangnya tiga nilai k yang berbeda dan valid. Gunakan data *training* dan *validation* yang sama untuk setiap percobaan, kemudian bandingkan akurasi pada *validation* serta waktu pengenalan pada lingkungan eksekusi yang sama. Apakah penggunaan lebih banyak *eigenfaces* selalu menghasilkan pengenalan yang lebih akurat? Jelaskan berdasarkan hasil eksperimen Anda.
4. Mengapa pengujian dengan citra *training* belum cukup untuk menilai kemampuan pengenalan program? Jelaskan pula pengaruhnya terhadap hasil evaluasi apabila citra *validation* ikut digunakan dalam perhitungan *mean face* dan pembentukan basis *eigenfaces*.
5. *Dataset* Georgia Tech memiliki variasi pencahayaan, ekspresi, dan sudut wajah. Bagaimana variasi tersebut dapat memengaruhi pengenalan menggunakan *Eigenfaces*? Tunjukkan satu citra *validation* yang menurut Anda paling sulit dikenali, lalu analisis hasilnya berdasarkan prapemrosesan dan representasi citra dalam ruang *eigenfaces*.
6. Ketika citra atau subjek baru ditambahkan ke *dataset*, mengapa *mean face*, basis *eigenfaces*, dan proyeksi citra *training* perlu diperbarui? Apa yang dapat terjadi apabila program menggunakan *mean face* atau basis yang baru, tetapi tetap mempertahankan vektor proyeksi dari model sebelumnya?

3.2 Spesifikasi Bonus

Anda dapat mengerjakan spesifikasi berikut untuk memperoleh poin tambahan. Fitur bonus program yang dikerjakan harus dijelaskan dalam laporan dan didemonstrasikan dalam video wajib.

3.2.1 Kontainerisasi dengan Docker dan *Deployment*

Aplikasi *web* yang dibuat dapat dikontainerisasi menggunakan Docker sehingga dapat dijalankan di berbagai lingkungan dengan konfigurasi yang telah disediakan. Sertakan `Dockerfile` dan `docker-compose.yml` pada repositori kelompok agar seluruh layanan aplikasi dapat dijalankan menggunakan perintah `docker compose up --build`, dengan petunjuk penggunaan pada `README.md`. Aplikasi juga dapat di-*deploy* pada layanan *hosting* dan harus dapat diakses oleh asisten serta menjalankan fitur wajib program. Cantumkan tautan aplikasi pada `README.md` dan bagian lampiran laporan. Kontainerisasi dan *deployment* dinilai sebagai satu komponen bonus.

3.2.2 *Beautiful UI/UX*

Aplikasi *web* memiliki UI/UX yang responsif, tampilan yang menarik, serta menerapkan prinsip-prinsip desain UI/UX yang baik. Penilaian memperhatikan keterbacaan, konsistensi elemen antarmuka, kejelasan navigasi, kesesuaian tampilan pada perangkat dengan ukuran layar berbeda, serta penyajian status pemrosesan dan pesan kesalahan. Jelaskan prinsip-prinsip desain UI/UX yang diterapkan pada Bab 3 laporan sebagai subbab tersendiri, disertai contoh penerapannya pada antarmuka program.

3.2.3 Penulisan Laporan dengan LaTeX

LaTeX merupakan bahasa markah (*markup language*) yang banyak digunakan untuk penulisan dokumen ilmiah dan membantu pengelolaan referensi, gambar, tabel, serta persamaan matematika secara konsisten. Tulislah laporan menggunakan LaTeX dan sertakan berkas sumber `.tex` beserta berkas pendukung yang diperlukan untuk kompilasi pada repositori kelompok, dalam folder yang sama dengan laporan PDF.

3.2.4 Galeri *Eigenfaces*

Aplikasi menyediakan halaman atau bagian khusus yang menampilkan *mean face* dan sejumlah *eigenfaces* teratas dari model yang sedang digunakan. *Eigenfaces* ditampilkan berdasarkan urutan *singular value* dari yang terbesar, disertai nomor komponen dan *singular value* yang bersesuaian. Nilai piksel boleh dinormalisasi untuk keperluan visualisasi, tetapi normalisasi tampilan tidak boleh mengubah nilai basis yang digunakan dalam perhitungan pengenalan wajah. Galeri harus menampilkan hasil perhitungan program dan diperbarui setelah model selesai dilatih ulang akibat penambahan citra atau subjek.

3.2.5 *Threshold* untuk *Unknown Face*

Aplikasi memiliki mekanisme ambang batas (*threshold*) agar dapat menyatakan bahwa wajah masukan tidak dikenali, alih-alih selalu memilih salah satu subjek pada data *training*. Program menggunakan jarak Euclidean terdekat dalam ruang *eigenfaces* sebagai dasar keputusan. Apabila jarak tersebut melebihi nilai ambang, program menampilkan status wajah tidak dikenali. Jika jaraknya tidak melebihi nilai ambang, program menampilkan hasil pencocokan sesuai dengan fitur wajib. Nilai ambang dapat ditentukan melalui antarmuka atau konfigurasi program, dan keluaran harus menampilkan status pengenalan, jarak terdekat, serta nilai ambang yang digunakan. Pengujian menggunakan citra *validation* untuk wajah yang dikenal serta citra lain yang disiapkan sendiri oleh kelompok untuk wajah yang tidak dikenal. Citra tersebut harus berasal dari identitas yang belum terdapat pada data *training* dan diunggah melalui fitur pengenalan tanpa terlebih dahulu ditambahkan ke dalam *dataset*. Jelaskan alasan pemilihan ambang dan hasil pengujiannya dalam laporan serta video, dengan membahas kemampuan menerima wajah yang dikenal dan menolak wajah yang tidak dikenal.

3.3 Prosedur Pengerjaan

1. Tugas dikerjakan secara berkelompok dengan anggota sebanyak 3 orang yang dipilih sendiri tetapi tidak boleh sama dengan tugas besar 1. Anggota kelompok boleh berasal dari kelas yang berbeda selama mendaftar sebagai peserta mata kuliah IF2123 Aljabar Linier dan Geometri. Setiap kelompok wajib mendaftarkan anggotanya dan memilih asisten pembimbing melalui Kelompok Tugas Besar IF2123 Aljabar Linier dan Geometri 2026/2027 sebelum 8 November 2026 pukul 09.00 WIB. Peserta yang belum terdaftar setelah tenggat akan dipasangkan ke dalam kelompok secara acak.

2. Salah satu perwakilan kelompok bertindak sebagai pemilik repositori. Perwakilan kelompok membuat satu repositori baru pada akun GitHub pribadinya dengan ketentuan sebagai berikut:

   - Nama repositori adalah `algeo26-tb2-<nama-kelompok>`.
   - Repositori bersifat *private*.
   - Setiap kelompok hanya membuat satu repositori.

3. Pemilik repositori wajib mengundang seluruh anggota kelompok dan asisten pembimbing pilihan sebagai *collaborator*. Seluruh undangan harus diterima sebelum 11 November 2026 pukul 23.59 WIB. Akses *collaborator* tidak boleh dicabut sampai seluruh proses penilaian tugas besar selesai.

4. Setiap anggota wajib menggunakan akun GitHub masing-masing dalam mengerjakan tugas. Pembagian kontribusi anggota harus dapat dilihat melalui riwayat *commit* repositori.
5. Pertanyaan mengenai spesifikasi tugas dapat disampaikan melalui *sheet* QnA Tugas Besar IF2123 Aljabar Linier dan Geometri 2026/2027.
6. Kelompok dapat mengajukan asistensi kepada asisten yang telah dipilih apabila diperlukan. Asisten tersebut nantinya juga akan berperan sebagai pemeriksa hasil program, laporan, serta video kalian.
7. Demo program akan dilaksanakan secara *black box* oleh asisten pemeriksa dan setiap kelompok diwajibkan membuat video sesuai dengan ketentuan yang telah diberikan.
8. DILARANG menyontek pekerjaan kelompok lain atau menggunakan bantuan *generative AI*, seperti ChatGPT, DeepSeek, Claude, Gemini, dan layanan sejenis, kecuali untuk penggunaan yang secara eksplisit diizinkan pada spesifikasi. Segala bentuk kecurangan akan ditindaklanjuti sesuai dengan ketentuan akademik yang berlaku.
    a. Gunakan AI secara bijaksana, misalnya untuk membantu mencari referensi, memperbaiki tata bahasa pada laporan, atau *debugging*. Namun, pastikan bahwa inti dari pekerjaan Anda adalah hasil pemikiran dan usaha kelompok Anda sendiri.
    b. Sertakan lampiran Form Penggunaan Generative AI yang diisi sejujur-jujurnya (Halaman 19 pada Panduan AI ITB). Apabila terdapat indikasi kecurangan atau ketidakjujuran, maka akan ditindaklanjuti sesuai hukum yang berlaku.

3.4 Spesifikasi Laporan

1. Sampul
    - mirip seperti sampul dokumen ini tanpa keterangan tenggat
    - ganti tulisan "Spesifikasi" menjadi "Laporan"
    - ganti logo IRK dengan foto bersama seluruh anggota kelompok
    - cantumkan nama kelompok serta nama dan NIM seluruh anggota di bawah foto
2. Bab 1 (Pendahuluan)
    - deskripsi persoalan, boleh menyalin bagian pendahuluan dokumen ini
    - tujuan pembuatan program
3. Bab 2 (Dasar Teori), teori singkat mengenai
    - *eigenvalue* dan *eigenvector* beserta cara mencari dan makna geometrisnya
    - *Singular Value Decomposition* (SVD) dan *Principal Component Analysis* (PCA)
    - penerapan PCA melalui SVD pada *Eigenfaces*, meliputi *mean-centering*, pembentukan basis, dan proyeksi citra
    - pencocokan wajah menggunakan jarak Euclidean
    - teori pendukung fitur bonus yang dikerjakan, apabila diperlukan
4. Bab 3 (Desain dan Implementasi), penjelasan desain aplikasi *web* yang dibuat, meliputi
    - arsitektur aplikasi dan teknologi yang digunakan
    - struktur program dan modul penting
    - alur prapemrosesan citra, pembentukan model, dan pengenalan wajah
    - pembaruan model ketika citra atau subjek ditambahkan
    - implementasi fitur bonus yang dikerjakan
    - subbab khusus mengenai prinsip desain dan contoh penerapannya, apabila mengerjakan bonus *Beautiful UI/UX*
5. Bab 4 (Eksperimen)
    - hasil pengujian pengenalan wajah, penambahan citra, dan penambahan subjek baru, disertai tangkapan layar yang mewakili

- tabel perbandingan akurasi pada data *validation* dan waktu pengenalan untuk sekurang-kurangnya tiga nilai k, menggunakan data *training* dan lingkungan eksekusi yang sama
- analisis hasil pengujian, termasuk kesalahan pengenalan dan keterbatasan program
- hasil pengujian fitur bonus yang dikerjakan
- pemilihan ambang serta hasil pengujian wajah yang dikenal dan tidak dikenal, apabila mengerjakan bonus *unknown face*
- ringkasan hasil eksperimen yang mendukung pertanyaan analisis pada Subbab 3.1.4, sedangkan jawaban lengkap disampaikan dalam video wajib

6. Bab 5 (Penutup)
   - kesimpulan dari seluruh pengerjaan tugas besar
   - saran pengembangan program dan saran kepada asisten
   - refleksi pribadi setiap anggota kelompok
7. Daftar Pustaka
   - seluruh referensi yang digunakan dalam pengerjaan tugas besar, termasuk referensi teori dan sumber *dataset*
   - menggunakan format sitasi IEEE
8. Lampiran
   - tautan menuju repositori GitHub kelompok
   - tautan menuju video wajib
   - tautan menuju aplikasi *web* yang telah di-*deploy*, apabila mengerjakan bonus *deployment*

3.5 Kriteria Penilaian

Penilaian tugas besar terdiri atas implementasi program (70 poin), video wajib (15 poin), laporan (15 poin), dan bonus (25 poin). Rincian skor maksimum setiap komponen penilaian ditunjukkan pada tabel berikut.

Tabel 1: Rincian Skor Maksimum untuk Tiap Komponen Penilaian

Komponen	**Skor Maksimum**
Implementasi program sesuai spesifikasi wajib	70
Video penjelasan dan demonstrasi program	15
Laporan	15
(Bonus) Kontainerisasi dengan Docker dan *Deployment*	10
(Bonus) *Beautiful UI/UX*	3
(Bonus) Penulisan laporan dengan LaTeX	2
(Bonus) Galeri *Eigenfaces*	5
(Bonus) *Threshold* untuk *Unknown Face*	5
Total	**120**

4 Pengumpulan

4.1 *Deliverables*

Berkas yang dikumpulkan oleh setiap kelompok terdiri atas kode sumber (*source code*) dan laporan tugas besar dengan ketentuan sebagai berikut.

1. Program dan laporan disimpan di dalam repositori GitHub *private*.
2. Repositori sekurang-kurangnya memuat tiga folder berikut:
   - Folder `docs` yang berisi laporan dalam format PDF dengan nama `<nama-kelompok>.pdf`.
   - Folder `src` yang berisi seluruh kode sumber program.
   - Folder `test` yang berisi data citra kasus uji serta berkas pendukung pengujian lainnya.
3. Setiap kelompok wajib menggunakan *semantic commit message* dan menjelaskan perubahan yang dilakukan.
4. Repositori wajib memiliki berkas `README.md` yang sekurang-kurangnya memuat penjelasan singkat program, alur program, tata cara menjalankan program, tautan YouTube menuju video demonstrasi, serta rincian/daftar fitur bonus yang dikerjakan bila memang mengerjakan. Gunakan `README` ini sebagai referensi.
5. Pastikan repositori *private*[1] selama pengerjaan tugas besar.
6. Program akan dijalankan asisten untuk melakukan penilaian. Asisten pemeriksa tidak akan melakukan pengaturan atau kompilasi lagi agar program dapat berjalan. Program yang tidak dapat dijalankan tidak akan diberi nilai.
7. Apabila kelompok mengerjakan fitur bonus, seluruh kode sumber, berkas pendukung, dan petunjuk penggunaan fitur tersebut harus disertakan di dalam repositori.

4.2 Teknis Pengumpulan

1. Program dan laporan dikumpulkan dengan membuat rilis pada repositori GitHub untuk tiap kelompok. Sebelum membuat rilis, pastikan seluruh *deliverables* telah tersedia dan tag rilis mengacu pada *commit* terakhir yang akan dinilai.
2. Tag rilis menggunakan format `v1.x`, dengan `x` merupakan nomor revisi (`x` = 0 untuk rilis pertama). Judul rilis menggunakan format `v1.x-<nama-kelompok>`. Deskripsi rilis sekurang-kurangnya memuat nama kelompok, daftar anggota, dan ringkasan perubahan apabila rilis tersebut merupakan revisi.
3. Pengumpulan dilakukan melalui form berikut. Pengumpulan cukup dilakukan satu kali oleh perwakilan kelompok. Setiap kelompok wajib memastikan bahwa tautan rilis dapat diakses oleh asisten pembimbing dan koordinator asisten. Kesalahan tautan atau akses *repository* menjadi tanggung jawab kelompok.
4. Kelompok boleh melakukan revisi dengan membuat rilis baru dan mengirimkan kembali Google Form. Versi yang digunakan untuk penilaian adalah rilis terakhir yang dikumpulkan melalui Google Form. Perubahan pada *repository* yang tidak disertai pembuatan dan pengumpulan rilis baru tidak akan dinilai.
5. Waktu pengumpulan ditentukan berdasarkan waktu pengiriman Google Form. Rilis tidak boleh diubah setelah tautannya dikumpulkan. Apabila diperlukan perubahan, kelompok wajib membuat nomor revisi baru dan mengirimkan kembali formulir. Tenggat pengumpulan tugas besar adalah **Rabu, 25 November 2026 pukul 21.23 WIB**.
6. Keterlambatan dikenai penalti sebesar 1 poin untuk setiap 6 menit keterlambatan. Durasi keterlambatan dihitung dari selisih antara tenggat pengumpulan dan waktu pengiriman terakhir pada Google Form, kemudian dibulatkan ke atas ke kelipatan 6 menit.[2]

[1] Bisa diubah ke *public* setelah tenggat pengumpulan tugas besar berlalu.
[2] Misalnya, keterlambatan 3 menit maupun 6 menit sama-sama menghasilkan penalti 1 poin.

5 Tautan Penting

- Salinan Dokumen Spesifikasi Terbaru
- Linktree Algeo 2026
- QnA Tugas Besar IF2123 Aljabar Linier dan Geometri 2026/2027
- Kelompok Tugas Besar IF2123 Aljabar Linier dan Geometri 2026/2027
- Dataset Eigenfaces
- Situs Pak Rinaldi Munir

6 Referensi

Silakan lakukan eksplorasi mandiri melalui referensi-referensi berikut untuk memperdalam pemahaman Anda mengenai teori-teori yang telah dibahas pada Bab 2 serta pengembangan aplikasi pada tugas besar ini.

1. **Nilai Eigen, Vektor Eigen, dan SVD**
    a. T. Roughgarden dan G. Valiant, "Lecture #9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations," *CS168: The Modern Algorithmic Toolbox*, Stanford University, 30 April 2024. [Daring]. Tersedia: https://web.stanford.edu/class/cs168/l/l9.pdf.
    b. D. Bindel, "Solvers for SEP and SVD," *CS 6210: Matrix Computations*, Cornell University, 5 November 2025. [Daring]. Tersedia: https://www.cs.cornell.edu/courses/cs6210/2025fa/lec/2025-11-05.html.
2. **PCA**
    a. J. Shlens, "A Tutorial on Principal Component Analysis," *arXiv preprint arXiv:1404.1100*, April 2014, doi: 10.48550/arXiv.1404.1100. [Daring]. Tersedia: https://arxiv.org/abs/1404.1100.
    b. I. T. Jolliffe dan J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, art. 20150202, April 2016, doi: 10.1098/rsta.2015.0202.
3. **Eigenfaces**
    a. M. Turk dan A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, hlm. 71–86, 1991, doi: 10.1162/jocn.1991.3.1.71. [Daring]. Tersedia: salinan artikel pada MIT.
    b. M. A. Turk dan A. P. Pentland, "Face Recognition Using Eigenfaces," dalam *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Maui, HI, Amerika Serikat, 1991, hlm. 586–591, doi: 10.1109/CVPR.1991.139758. [Daring]. Tersedia: salinan artikel pada University of Texas at Austin.
    c. OpenCV, "Face Recognition with OpenCV," *OpenCV Documentation*, versi 4.13.0. [Daring]. Tersedia: https://docs.opencv.org/4.13.0/da/d60/tutorial_face_main.html.
    d. OpenCV, "cv::face::EigenFaceRecognizer Class Reference," *OpenCV Documentation*, versi 4.13.0. [Daring]. Tersedia: dokumentasi EigenFaceRecognizer.
4. **Prapemrosesan Citra dan Dataset**
    a. OpenCV, "Color conversions," *OpenCV Documentation*, versi 4.13.0. [Daring]. Tersedia: https://docs.opencv.org/4.13.0/de/d25/imgproc_color_conversions.html.
    b. A. V. Nefian, "Georgia Tech face database," *Face Recognition*. [Daring]. Tersedia: https://www.anefian.com/research/face_reco.htm.
5. **Pengembangan Aplikasi Web**
    a. MDN contributors, "Overview of HTTP," *MDN Web Docs*, Mozilla. [Daring]. Tersedia: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview.
    b. MDN contributors, "Client-server overview," *MDN Web Docs*, Mozilla. [Daring]. Tersedia: dokumentasi arsitektur client–server.
    c. MDN contributors, "POST request method," *MDN Web Docs*, Mozilla. [Daring]. Tersedia: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/POST.
    d. T. Bray, ed., "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, Desember 2017, doi: 10.17487/RFC8259. [Daring]. Tersedia: https://www.rfc-editor.org/rfc/rfc8259.
6. **Kontainerisasi dengan Docker dan Deployment**

a. Docker Inc., "Why use Compose?," *Docker Docs*. [Daring]. Tersedia: https://docs.docker.com/compose/intro/features-uses/.

7. **Dokumen Acuan Penyusunan**

a. "Spesifikasi Tugas Besar 2 IF2123 Aljabar Linier dan Geometri: EigenPustaka, Sistem Perpustakaan Digital dengan Rekomendasi Semantik dan Pencarian Visual Berbasis SVD," Semester I Tahun 2025/2026, versi 1.2, 2025.

b. I. M. W. Putera, "OHL_Eigenfaces," spesifikasi Tugas Seleksi Laboratorium Ilmu Rekayasa Komputasi 2026, versi 10 Juli 2026.

7 Surat Cinta dari Asisten

Komentar 7.1: Ares

semangatt bantai tubes 2 nyaa, pixelll!

Komentar 7.2: Dzaki

Yahh tugas tubes tarakhir Algeo dengerin ini ya
https://youtu.be/_OfRiOucWwc?si=jqCeBmQ4Z3-HTG7V

Komentar 7.3: Daniel



Komentar 7.4: An-Dafa

https://www.instagram.com/reels/De_szqCiSDa - Brick by brick 😭

Komentar 7.5: Manuel



Komentar 7.6: Jonathan



goodluck gng