

## **LAPORAN TUGAS BESAR 2**

### **IF2123 - ALJABAR LINIER DAN GEOMETRI**

**“Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar”**



#### **Dosen:**

Ir. Rila Mandala, M.Eng., Ph.D.

Arrival Dwi Sentosa, S.Kom., M.T.

#### **Kelompok 35:**

Owen Tobias Sinurat (13522131)

Farhan Raditya Aji (13522142)

Axel Santadi Warih (13522155)

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**SEMESTER I TAHUN 2023/2024**

# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	<b>4</b>
<b>1.1 ABSTRAKSI</b>	<b>4</b>
<b>1.2 CONTENT-BASED INFORMATION RETRIEVAL (CBIR)</b>	<b>5</b>
<b>1.3 PENGGUNAAN PROGRAM</b>	<b>11</b>
<b>1.4 SPESIFIKASI TUGAS</b>	<b>12</b>
<b>BAB II</b>	<b>13</b>
<b>2.1 CONTENT-BASED INFORMATION RETRIEVAL (CBIR)</b>	<b>13</b>
2.1.1 CBIR dengan parameter warna	13
2.1.2 CBIR dengan parameter tekstur	16
<b>2.2 WEBSITE DEVELOPMENT</b>	<b>19</b>
2.2.1 Pengertian	19
2.2.2 Mengapa Web Development Penting?	19
2.2.3 Jenis-Jenis Web Development	19
1. Frontend Web Development	20
2. Backend Web Development	20
3. Fullstack Web Development	21
2.2.4 Proses Kerja Web Development	21
1. Perencanaan (Planning)	21
2. Perancangan (Designing)	22
3. Implementasi (Coding)	22
4. Pengujian dan Penyebaran (Testing and Deployment)	23
5. Post-deployment dan Maintenance	23
<b>BAB III</b>	<b>24</b>
<b>3.1 Langkah-langkah Pemecahan Masalah</b>	<b>24</b>
3.1.1 CBIR dengan Parameter Warna	24
3.1.2 CBIR dengan parameter Tekstur	25
<b>3.2 Pemetaan Masalah ke Aljabar Geometri</b>	<b>26</b>
3.2.1 CBIR dengan Parameter Warna	26
3.2.2 CBIR dengan Parameter Tekstur	27
<b>3.3 Contoh Ilustrasi Kasus dan Penyelesaiannya</b>	<b>28</b>
3.3.1 CBIR dengan Parameter Warna	28
3.3.2 CBIR dengan Parameter Tekstur	28
<b>BAB IV</b>	<b>29</b>
<b>4.1 Pseudocode</b>	<b>29</b>
4.1.1 Warna	29
4.1.1.1 Pseudocode histHSV	29
4.1.1.2 Pseudocode similarity	30
4.1.1.3 Pseudocode colorCBIR	30

4.1.2 Tekstur	31
4.1.2.1 Pseudocode grayscale_image	31
4.1.2.2 Pseudocode calculate_texture_features	32
4.1.2.3 Pseudocode cosine_similarity	32
4.1.2.4 Pseudocode textureCBIR	32
4.1.3 Dataset Processing	33
4.1.2.5 Pseudocode datasetFeatureExtractor	33
<b>4.2 Penjelasan dan Tatacara Penggunaan Program</b>	<b>34</b>
4.2.1 Warna	34
4.2.1.1 histHSV	34
4.2.1.2 similarity	35
4.2.1.3 ubahArrayGambarJadiHistHSV	35
4.2.1.4 mainColor	35
4.2.2 Tekstur	36
4.2.2.1 ubahArrayGambarMenjadiListCooc	36
4.2.2.2 grayscale_image	36
4.2.2.3 calculate_texture_features	36
4.2.2.4 cosine_similarity	36
4.2.2.5 mainTekstur	36
4.2.3 Tata Cara Penggunaan	37
4.2.3.1 Tata Cara FrontEnd	37
4.2.3.2 Tata Cara BackEnd	37
<b>4.3 Hasil Pengujian</b>	<b>38</b>
4.3.1 Warna	38
4.3.2 Tekstur	39
<b>4.4 Analisis Color dan Tekstur</b>	<b>39</b>
<b>BAB V</b>	<b>41</b>
<b>5.1 Kesimpulan</b>	<b>41</b>
<b>5.2 Saran</b>	<b>41</b>
<b>5.3 Komentar</b>	<b>41</b>
<b>5.4 Refleksi</b>	<b>42</b>
<b>DAFTAR PUSTAKA</b>	<b>43</b>
<b>HASIL</b>	<b>44</b>

# BAB I

## DESKRIPSI MASALAH

### 1.1 ABSTRAKSI

Dalam era digital, jumlah gambar yang dihasilkan dan disimpan semakin meningkat dengan pesat, baik dalam konteks pribadi maupun profesional. Peningkatan ini mencakup berbagai jenis gambar, mulai dari foto pribadi, gambar medis, ilustrasi ilmiah, hingga gambar komersial. Terlepas dari keragaman sumber dan jenis gambar ini, sistem temu balik gambar (*image retrieval system*) menjadi sangat relevan dan penting dalam menghadapi tantangan ini. Dengan bantuan sistem temu balik gambar, pengguna dapat dengan mudah mencari, mengakses, dan mengelola koleksi gambar mereka. Sistem ini memungkinkan pengguna untuk menjelajahi informasi visual yang tersimpan di berbagai platform, baik itu dalam bentuk pencarian gambar pribadi, analisis gambar medis untuk diagnosis, pencarian ilustrasi ilmiah, hingga pencarian produk berdasarkan gambar komersial. Salah satu contoh penerapan sistem temu balik gambar yang mungkin kalian tahu adalah Google Lens.



**Gambar 1.** Contoh penerapan *information retrieval system* (Google Lens)

Di dalam Tugas Besar 2 ini, Anda diminta untuk mengimplementasikan sistem temu balik gambar yang sudah dijelaskan sebelumnya dengan memanfaatkan Aljabar Vektor dalam bentuk sebuah *website*, dimana hal ini merupakan pendekatan yang penting dalam dunia pemrosesan data dan pencarian informasi. Dalam konteks ini, aljabar vektor digunakan untuk menggambarkan dan menganalisis data menggunakan pendekatan klasifikasi berbasis konten (*Content-Based Image Retrieval* atau CBIR), di mana sistem temu balik gambar bekerja dengan mengidentifikasi gambar berdasarkan konten visualnya, seperti warna dan tekstur.

## **1.2 CONTENT-BASED INFORMATION RETRIEVAL (CBIR)**

*Content-Based Image Retrieval* (CBIR) adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan ekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur tersebut diekstraksi, mereka diwakili dalam bentuk vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Kemudian, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur gambar dalam dataset. Hasil dari pencocokan ini digunakan untuk mengurutkan gambar-gambar dalam dataset dan menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar dengan cara yang lebih efisien, karena tidak memerlukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai citra visual antara gambar-gambar tersebut. Pada Tugas Besar kali ini, Anda diminta untuk mengimplementasikan 2 parameter CBIR yang paling populer, antara lain :

### **1. CBIR dengan parameter warna**

Pada CBIR kali ini akan dibandingkan *input* dari sebuah *image* dengan *image* yang dimiliki oleh dataset, hal ini dilakukan dengan cara mengubah *image* yang berbentuk RGB menjadi sebuah metode histogram warna yang lebih umum.

Histogram warna adalah frekuensi dari berbagai warna yang ada pada ruang warna tertentu hal ini dilakukan untuk melakukan pendistribusian warna dari *image*. Histogram warna tidak bisa mendeteksi sebuah objek yang spesifik yang terdapat pada *image* dan tidak bisa mendeskripsikan posisi dari warna yang didistribusikan.

Pembentukan ruang warna perlu dilakukan dalam rangka pembagian nilai citra menjadi beberapa *range* yang lebih kecil. Hal itu dilakukan untuk membuat sebuah histogram warna yang setiap interval tiap *range* dianggap sebagai *bin*. Histogram warna dapat dihitung dengan menghitung piksel yang menyatakan nilai warna pada setiap interval. Fitur warna mencakup histogram warna global dan histogram warna blok.

Pada perhitungan histogram, warna global HSV lebih dipilih karena warna tersebut dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum

untuk digunakan. Maka dari itu, perlu dilakukan konversi warna dari RGB ke HSV dengan prosedur sebagai berikut.

- Nilai dari RGB harus dinormalisasi dengan mengubah nilai *range* [0, 255] menjadi [0, 1]

$$R' = \frac{R}{255} \quad G' = \frac{G}{255} \quad B' = \frac{B}{255}$$

- Mencari  $C_{max}$ ,  $C_{min}$ , dan  $\Delta$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

- Selanjutnya gunakan hasil perhitungan di atas untuk mendapatkan nilai HSV

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left( \frac{G' - B'}{\Delta} \bmod 6 \right) & , C' \text{ max} = R' \\ 60^\circ \times \left( \frac{B' - R'}{\Delta} + 2 \right) & , C' \text{ max} = G' \\ 60^\circ \times \left( \frac{R' - G'}{\Delta} + 4 \right) & , C' \text{ max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

$$V = C_{max}$$

Setelah mendapatkan nilai HSV lakukanlah perbandingan antara *image* dari input dengan dataset dengan menggunakan *cosine similarity*

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

Dengan  $A$  dan  $B$  adalah vektor dan  $n$  adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *Cosine Similarity*.

Untuk melakukan pencarian histogram, blok *image* dibagi menjadi  $n \times n$  blok. Setiap blok akan menjadi tidak terlalu signifikan jika blok-blok tersebut terlalu besar dan akan meningkatkan waktu dalam memprosesnya jika ukuran dari blok terlalu kecil. Pada pencarian blok ini agar lebih efektif disarankan menggunakan  $4 \times 4$  blok. Nyatakan nilai representatif dari sebuah blok dengan melakukan kalkulasi **nilai rata-rata HSV dari blok terkait** (sedikit berbeda dengan yang disampaikan pada jurnal referensi untuk menyederhanakan perhitungan. Akan tetapi, jika sudah telanjur mengimplementasikan metode pada referensi [**rata-rata histogram HSV**], diperbolehkan).

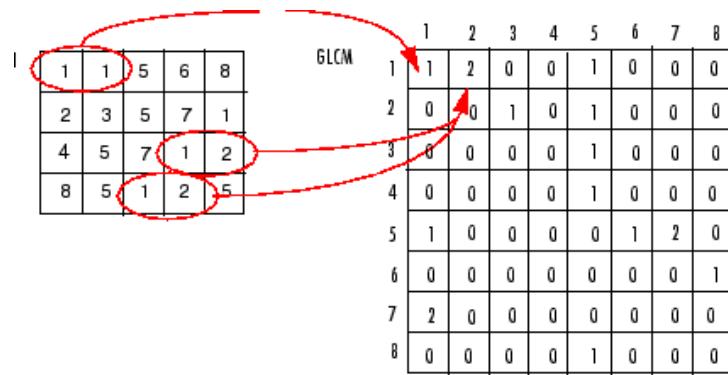
## 2. CBIR dengan parameter tekstur

CBIR dengan perbandingan tekstur dilakukan menggunakan suatu matriks yang dinamakan *co-occurrence matrix*. Matriks ini digunakan karena dapat melakukan pemrosesan yang lebih mudah dan cepat. Vektor yang dihasilkan juga mempunyai ukuran yang lebih kecil. Misalkan terdapat suatu gambar  $I$  dengan  $n \times m$  piksel dan suatu parameter offset  $(\Delta x, \Delta y)$ , Maka dapat dirumuskan matriksnya sebagai berikut:

Dengan menggunakan nilai  $i$  dan  $j$  sebagai nilai intensitas dari gambar dan  $p$  serta  $q$  sebagai posisi dari gambar, maka offset  $\Delta x$  dan  $\Delta y$  bergantung pada arah  $\theta$  dan jarak yang digunakan melalui persamaan di bawah ini.

$$C_{\Delta x, \Delta y} (i, j) = \sum_{p=1}^n \sum_{q=1}^m \begin{cases} 1, & \text{jika } I(p, q) = i \text{ dan } I(p + \Delta x, q + \Delta y) = j \\ 0, & \text{jika lainnya} \end{cases}$$

Melalui persamaan tersebut, digunakan nilai  $\theta$  adalah  $0^\circ, 45^\circ, 90^\circ$ , dan  $135^\circ$ .



**Gambar 2.** Cara Pembuatan Matrix *Occurrence*

Setelah didapat *co-occurrence matrix*, buatlah *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya. Lalu cari *matrix normalization* dengan persamaan.

$$\text{MatrixNorm} = \frac{\text{MatrixOcc}}{\sum \text{MatrixOcc}}$$

Langkah-langkah dalam CBIR dengan parameter tekstur adalah sebagai berikut :

1. Konversi warna gambar menjadi *grayscale*, ini dilakukan karena warna tidaklah penting dalam penentuan tekstur. Oleh karena itu, warna RGB dapat diubah menjadi suatu warna *grayscale*  $Y$  dengan rumus:

$$Y = 0.29 \times R + 0.587 \times G + 0.114 \times B$$

2. Lakukan kuantifikasi dari nilai *grayscale*. Karena citra *grayscale* berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran  $256 \times 256$ . Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut.
3. Dari *co-occurrence matrix* bisa diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Persamaan yang dapat digunakan untuk mendapatkan nilai 3 komponen tersebut antara lain :

*Contrast*:

$$\sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} (i - j)^2$$

*Homogeneity* :

$$\sum_{i,j=0}^{\text{dimensi}-1} \frac{P_{i,j}}{1 + (i - j)^2}$$

*Entropy* :

$$-\left( \sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} \times \log P_{i,j} \right)$$

**Keterangan :**  $P$  merupakan matriks *co-occurrence*

Dari ketiga komponen tersebut, dibuatlah sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan.

4. Ukurlah kemiripan dari kedua gambar dengan menggunakan Teorema *Cosine Similarity*, yaitu:

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

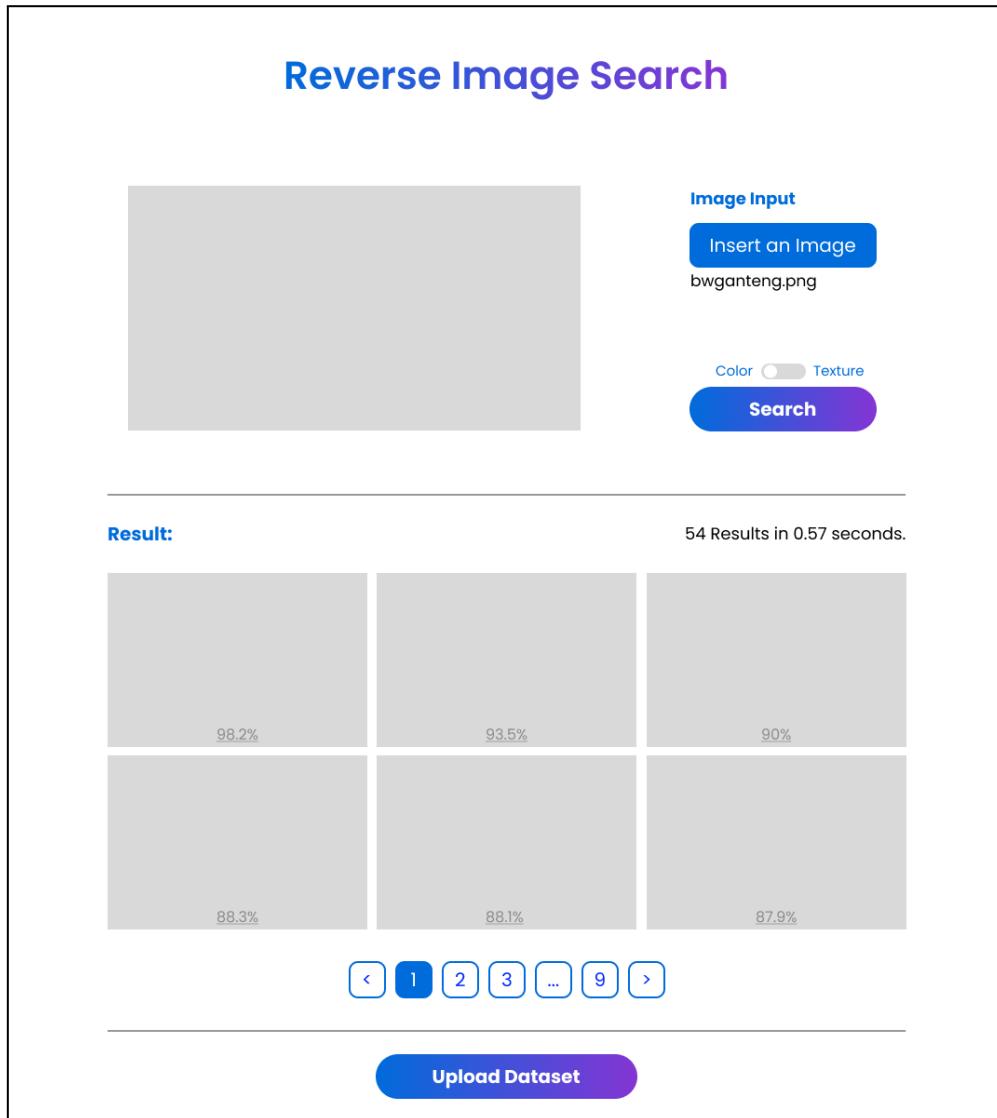
Disini  $A$  dan  $B$  adalah dua vektor dari dua gambar. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

### 1.3 PENGGUNAAN PROGRAM

Berikut ini adalah masukan yang akan diberikan oleh pengguna untuk eksekusi program.

1. *Image query*, berisi gambar yang akan digunakan untuk melakukan pencarian gambar.
2. Kumpulan gambar (dataset), dilakukan dengan cara mengunggah *multiple image* dalam bentuk *folder* ke dalam *web browser*. Setelah memasukkan *image query*, kumpulan gambar inilah yang akan diseleksi menjadi *result*.

Tampilan *layout* dari aplikasi *website* yang akan dibangun adalah sebagai berikut.



**Gambar 3.** Ilustrasi tampilan layout dari aplikasi *website*

Anda dapat menambahkan menu lainnya seperti gambar, logo, dan sebagainya. Tampilan *Front-End* dari *website* tidak harus sama persis dengan *layout* yang diberikan di atas, tetapi dibuat semenarik mungkin dan wajib mencakup komponen-komponen berikut:

- Judul Website
- Tombol Insert Gambar, beserta Display Gambar yang ingin dicari
- Toggle Button untuk memilih *searching* berdasarkan Warna atau Tekstur

- Tombol Search untuk memulai pencarian
- Kumpulan Gambar (result) yang didapat dari hasil pencarian
- Informasi mengenai Jumlah Result yang didapat dan Waktu Eksekusi
- Tingkat Kemiripan Setiap Gambar (result) dengan gambar yang ingin dicari, dalam persentase (%)
- Tombol Upload Dataset untuk mengunggah kumpulan gambar (dataset) dalam suatu folder
- Page-page tambahan: Konsep singkat search engine yang dibuat, How to Use, dan About us

Secara umum, berikut adalah cara umum penggunaan program:

1. Pengguna terlebih dahulu memasukkan dataset gambar dalam bentuk *folder* yang berisi kumpulan gambar. Dataset gambar ini diperlukan sebelum proses *searching* agar ada perbandingan untuk gambar yang ingin dicari.
2. Setelah dataset sudah dimasukkan, pengguna memasukkan sebuah gambar yang ingin di-*search* dari dataset.
3. Pilih opsi pencarian, ingin melakukan pencarian berdasarkan warna atau tekstur.
4. Tekan tombol *search*, program kemudian akan memproses, mencari gambar-gambar dari dataset yang memiliki kemiripan dengan gambar yang dimasukkan tadi.
5. Program akan menampilkan kumpulan gambar yang mirip, diurutkan dari yang memiliki kemiripan paling tinggi ke yang paling rendah. Setiap gambar yang muncul diberi persentase kemiripannya.
6. Terdapat informasi terkait jumlah gambar yang muncul, dan waktu eksekusi programnya.

#### **1.4 SPESIFIKASI TUGAS**

Buatlah program sistem temu balik gambar (*image retrieval system*) dengan spesifikasi sebagai berikut:

1. Program menerima input *folder dataset* dan sebuah citra gambar.
2. Program menampilkan gambar citra gambar yang dipilih oleh pengguna.
3. Program dapat memberikan kebebasan pada pengguna untuk memilih parameter pencarian yang hendak digunakan (warna atau tekstur) melalui *toggle*. *Default* parameter yang digunakan di awal dibebaskan kepada masing-masing kelompok.
4. Program mulai melakukan perhitungan nilai kecocokan antara *image* masukan dengan *dataset image* berdasarkan parameter yang telah dipilih (warna atau tekstur).
5. Program dapat menampilkan nilai kecocokan antara *image* masukan dengan setiap gambar dalam dataset.
6. Program menampilkan hasil luaran dengan melakukan *descending sorting* berdasarkan nilai kecocokan tiap gambar. Cukup tampilkan seluruh gambar yang **memiliki tingkat kemiripan > 60%** dengan gambar masukan.
7. Program mengimplementasikan *pagination* agar jumlah gambar dapat dibatasi dengan halaman-halaman tertentu. Jumlah gambar dalam dataset yang akan digunakan oleh asisten saat penilaian mungkin berbeda dengan jumlah gambar pada dataset yang kalian gunakan, sehingga pastikan bahwa *pagination* dapat berjalan dengan baik.
8. Program dapat menampilkan **jumlah gambar** yang memenuhi kondisi tingkat kemiripan serta **waktu eksekusi**.

# **BAB II**

## **LANDASAN TEORI**

### **2.1 *CONTENT-BASED INFORMATION RETRIEVAL (CBIR)***

*Content-Based Image Retrieval* (CBIR) adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan ekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur tersebut diekstraksi, mereka diwakili dalam bentuk vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Kemudian, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur gambar dalam dataset. Hasil dari pencocokan ini digunakan untuk mengurutkan gambar-gambar dalam dataset dan menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar dengan cara yang lebih efisien, karena tidak memerlukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai citra visual antara gambar-gambar tersebut. Pada Tugas Besar kali ini, Anda diminta untuk mengimplementasikan 2 parameter CBIR yang paling populer, antara lain :

#### **2.1.1 *CBIR dengan parameter warna***

Pada CBIR kali ini akan dibandingkan *input* dari sebuah *image* dengan *image* yang dimiliki oleh dataset, hal ini dilakukan dengan cara mengubah *image* yang berbentuk RGB menjadi sebuah metode histogram warna yang lebih umum.

Histogram warna adalah frekuensi dari berbagai warna yang ada pada ruang warna tertentu hal ini dilakukan untuk melakukan pendistribusian warna dari *image*. Histogram warna tidak bisa mendeteksi sebuah objek yang spesifik yang terdapat pada *image* dan tidak bisa mendeskripsikan posisi dari warna yang didistribusikan.

Pembentukan ruang warna perlu dilakukan dalam rangka pembagian nilai citra

menjadi beberapa *range* yang lebih kecil. Hal itu dilakukan untuk membuat sebuah histogram warna yang setiap interval tiap *range* dianggap sebagai *bin*. Histogram warna dapat dihitung dengan menghitung piksel yang menyatakan nilai warna pada setiap interval. Fitur warna mencakup histogram warna global dan histogram warna blok.

Pada perhitungan histogram, warna global HSV lebih dipilih karena warna tersebut dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum untuk digunakan. Maka dari itu, perlu dilakukan konversi warna dari RGB ke HSV dengan prosedur sebagai berikut.

1. Nilai dari RGB harus dinormalisasi dengan mengubah nilai *range* [0, 255] menjadi [0, 1]

$$R' = \frac{R}{255} \quad G' = \frac{G}{255} \quad B' = \frac{B}{255}$$

2. Mencari  $C_{max}$ ,  $C_{min}$ , dan  $\Delta$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

3. Selanjutnya gunakan hasil perhitungan di atas untuk mendapatkan nilai HSV

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left( \frac{G' - B'}{\Delta} \bmod 6 \right) , C' \max = R' \\ 60^\circ \times \left( \frac{B' - R'}{\Delta} + 2 \right) , C' \max = G' \\ 60^\circ \times \left( \frac{R' - G'}{\Delta} + 4 \right) , C' \max = B' \end{cases}$$

$$S = \begin{cases} 0 & C_{\max} = 0 \\ \frac{\Delta}{C_{\max}} & C_{\max} \neq 0 \end{cases}$$

$$V = C_{\max}$$

Setelah mendapatkan nilai HSV lakukanlah perbandingan antara *image* dari input dengan dataset dengan menggunakan *cosine similarity*

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

Dengan  $A$  dan  $B$  adalah vektor dan  $n$  adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *Cosine Similarity*.

Untuk melakukan pencarian histogram, blok *image* dibagi menjadi  $n \times n$  blok. Setiap blok akan menjadi tidak terlalu signifikan jika blok-blok tersebut terlalu besar dan akan meningkatkan waktu dalam memprosesnya jika ukuran dari blok terlalu kecil. Pada pencarian blok ini agar lebih efektif disarankan menggunakan  $4 \times 4$  blok.

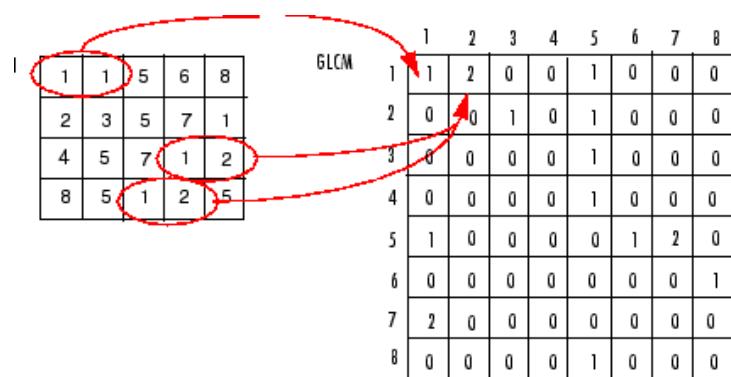
### 2.1.2 CBIR dengan parameter tekstur

CBIR dengan perbandingan tekstur dilakukan menggunakan suatu matriks yang dinamakan *co-occurrence matrix*. Matriks ini digunakan karena dapat melakukan pemrosesan yang lebih mudah dan cepat. Vektor yang dihasilkan juga mempunyai ukuran yang lebih kecil. Misalkan terdapat suatu gambar  $I$  dengan  $n \times m$  piksel dan suatu parameter offset  $(\Delta x, \Delta y)$ , Maka dapat dirumuskan matriksnya sebagai berikut:

Dengan menggunakan nilai  $i$  dan  $j$  sebagai nilai intensitas dari gambar dan  $p$  serta  $q$  sebagai posisi dari gambar, maka *offset*  $\Delta x$  dan  $\Delta y$  bergantung pada arah  $\theta$  dan jarak yang digunakan melalui persamaan di bawah ini.

$$C_{\Delta x, \Delta y} (i, j) = \sum_{p=1}^n \sum_{q=1}^m \begin{cases} 1, & \text{jika } I(p, q) = i \text{ dan } I(p + \Delta x, q + \Delta y) = j \\ 0, & \text{jika lainnya} \end{cases}$$

Melalui persamaan tersebut, digunakan nilai  $\theta$  adalah  $0^\circ, 45^\circ, 90^\circ$ , dan  $135^\circ$ .



**Gambar 2.** Cara Pembuatan Matrix *Occurrence*

Setelah didapat *co-occurrence matrix*, buatlah *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya. Lalu cari *matrix normalization* dengan persamaan.

$$\text{MatrixNorm} = \frac{\text{MatrixOcc}}{\sum \text{MatrixOcc}}$$

Langkah-langkah dalam CBIR dengan parameter tekstur adalah sebagai berikut :

1. Konversi warna gambar menjadi *grayscale*, ini dilakukan karena warna tidaklah penting dalam penentuan tekstur. Oleh karena itu, warna RGB dapat diubah menjadi suatu warna *grayscale*  $Y$  dengan rumus:

$$Y = 0.29 \times R + 0.587 \times G + 0.114 \times B$$

2. Lakukan kuantifikasi dari nilai *grayscale*. Karena citra *grayscale* berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran  $256 \times 256$ . Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut.
3. Dari *co-occurrence matrix* bisa diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Persamaan yang dapat digunakan untuk mendapatkan nilai 3 komponen tersebut antara lain :

*Contrast*:

$$\sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} (i - j)^2$$

*Homogeneity* :

$$\sum_{i,j=0}^{\text{dimensi}-1} \frac{P_{i,j}}{1 + (i - j)^2}$$

*Entropy* :

$$-\left( \sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} \times \log P_{i,j} \right)$$

**Keterangan :**  $P$  merupakan matriks *co-occurrence*

Dari ketiga komponen tersebut, dibuatlah sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan.

4. Ukurlah kemiripan dari kedua gambar dengan menggunakan Teorema *Cosine Similarity*, yaitu:

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

Disini  $A$  dan  $B$  adalah dua vektor dari dua gambar. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

## **2.2 WEBSITE DEVELOPMENT**

### **2.2.1 Pengertian**

Web development adalah istilah umum untuk pekerjaan yang dilaksanakan untuk membangun situs web. Hal ini mencakup segala hal, mulai dari markup, koding hingga scripting, konfigurasi jaringan, dan pengembangan CMS. Menurut Techopedia, web development dalam arti yang lebih luas mencakup semua tindakan, pembaruan, dan operasi yang dibutuhkan untuk membangun, memelihara, dan mengelola situs web. Hal ini bertujuan untuk memastikan website memiliki performa, pengalaman pengguna (user experience) dan kecepatan yang optimal.

### **2.2.2 Mengapa Web Development Penting?**

Jumlah pengguna internet meningkat dengan sangat pesat. Bisa dikatakan bahwa internet telah menjadi portal dan media utama untuk melakukan penelitian, edukasi, ataupun sekadar mencari hiburan. Dilansir dari [We Are Social](#), pengguna internet global per April 2022 mencapai lebih dari 5 miliar orang atau sekitar 63 persen dari populasi di dunia.

Saat ini website menjadi tempat bagi seseorang untuk menemukan berbagai informasi. Selain itu, website juga sangat berguna bagi sebuah bisnis sebagai sarana branding dan promosi. Untuk itulah web development menjadi industri yang menjanjikan dan sangat berkembang. Menurut [Bureau of Labor Statistics](#), pekerjaan di bidang web development pada kurun waktu 2020-2030 akan tumbuh sebesar 13% lebih cepat dibandingkan kebanyakan karir di bidang teknologi lainnya.

### **2.2.3 Jenis-Jenis Web Development**

Ingin menjadi web developer? Eits, tunggu dulu. Kamu harus tahu dan paham terlebih dahulu bahwa ada berbagai jenis karir di bidang web development dengan tugas yang juga berbeda-beda. Lalu, apa sajakah itu?

## ***1. Frontend Web Development***

Frontend development adalah jenis web development yang berkaitan dengan tampilan website. Misal seperti navigasi menu website, tata letak, grafis dan lainnya. Tugas ini dilaksanakan oleh seorang frontend web developer.

Frontend web developer bertugas untuk membangun antarmuka (interface) website yang bisa membantu pengguna menggunakan website dan mencari informasi dengan mudah.

Untuk menjadi seorang frontend web developer, kamu setidaknya harus menguasai 3 bahasa pemrograman: HTML, CSS, dan JavaScript. Selain itu, frontend web developer juga harus memastikan website memiliki tampilan yang responsif. Sehingga, apapun perangkat yang digunakan, tampilan website akan tampak rapi dan bisa menyesuaikan dengan ukuran layar perangkat.

Maka dari itu, frontend web developer juga harus bisa menggunakan framework seperti Angular dan React. Dengan menggunakan framework tersebut, frontend web developer bisa lebih mudah dalam mengembangkan tampilan antarmuka website dengan lebih menarik dan profesional.

## ***2. Backend Web Development***

Jika frontend web development berhubungan dengan apa yang bisa dilihat oleh pengguna, maka backend web development adalah kebalikannya. Backend web development dan frontend web development harus selalu bekerja sama supaya proses pengembangan website dapat berjalan dengan baik.

Backend web development adalah proses membangun website dari “balik layar” alias yang tidak dapat dilihat oleh pengguna. Mulai dari database, server sistem operasi, sistem keamanan, hingga Application Programming Interface (API).

Untuk melakukan proses pengembangan web dari sisi backend, kamu harus menguasai bahasa pemrograman untuk sisi server, seperti PHP, Python, dan SQL. Di

backend juga terdapat berbagai framework yang perlu dikuasai oleh developer seperti Django, CodeIgniter, dan Rails. Seperti halnya frontend, framework di backend development juga berfungsi untuk mempermudah workflow di dalam penggerjaan website, mengingat framework sudah memiliki kerangka dan struktur yang berguna untuk pembuatan website.

### ***3. Fullstack Web Development***

Jenis web development selanjutnya adalah fullstack web development. Di sini, developer melakukan pekerjaan frontend dan backend sekaligus. Artinya, seorang fullstack web developer harus bisa membuat tampilan dan fitur website sekaligus memastikan keamanan dan kecepatan website dari sisi server bisa berjalan optimal.

Developer yang bertugas di fullstack web development harus menguasai lebih banyak bahasa pemrograman dan framework. Tidak hanya itu, fullstack web developer juga harus bisa melakukan debugging, mengembangkan aplikasi, troubleshooting, dan bahkan menciptakan fitur baru.

#### ***2.2.4 Proses Kerja Web Development***

Dalam mengembangkan sebuah website, ada banyak sekali proses yang harus dilakukan, dan tentu saja tidak mudah. Mengingat, setiap pengembangan website mempunyai proses yang bervariasi bergantung pada tipe website, sumber daya yang dimiliki, dan bahasa pemrograman yang digunakan.

Walau begitu, secara umum proses kerja web development dibagi menjadi tujuh tahapan. Berikut penjelasannya.

##### ***1. Perencanaan (Planning)***

Agar proses pengembangan web dapat berjalan lancar, kamu perlu menentukan rencana pembuatan website. Mulai dari tujuan pembuatan website, target audiens, struktur navigasi, hingga budget yang dimiliki.

## ***2. Perancangan (Designing)***

Setelah perencanaan web development sudah direncanakan dengan matang, langkah selanjutnya adalah perancangan website. Di tahap ini, tampilan website dirancang menggunakan sitemap dan wireframe. Sitemap merupakan sketsa yang berisi korelasi antara berbagai halaman di dalam sebuah website. Melalui sitemap, developer dapat memperkirakan tingkat kemudahan pengguna dalam menemukan informasi atau layanan dalam sebuah website. Sedangkan wireframe adalah visual antarmuka website yang nantinya akan dibuat oleh developer. Fitur ini tidak memiliki elemen desain seperti logo dan warna. Wireframe berkaitan dengan elemen yang akan ditambahkan ke halaman website.

## ***3. Implementasi (Coding)***

Tahap selanjutnya adalah tahap implementasi pada web development, yakni penyusunan kode program. Pada tahap ini, developer akan menggunakan berbagai bahasa pemrograman beserta framework. Tahap ini ditangani oleh frontend developer yang bekerja sama dengan backend developer atau langsung oleh fullstack developer.

- Membuat Frontend Website

Dalam proses mengerjakan frontend web development, kamu memerlukan kombinasi dari HTML, CSS, dan JavaScript. Gabungan ketiga coding tersebut adalah untuk membuat navigasi menu, font website, dan desain website yang responsif. Singkatnya, tahap ini akan mengonversikan wireframe menjadi proyek website yang sebenarnya dengan berbagai unsur seperti grafik warna, tipografi, tombol, animasi, menu, dan lain sebagainya.

- Membuat Backend Website

Agar frontend dapat berfungsi, diperlukan back-end development. Dibutuhkan dua komponen dalam pengembangan backend, yakni database dan server.

Database berfungsi untuk menata, menyimpan, dan memproses data yang diminta oleh server. Sedangkan server adalah hardware atau software yang memiliki fungsi untuk mengirim, menerima, dan memproses permintaan data dari browser atau client. Setelah

memiliki kedua komponen tersebut, backend developer akan mengerjakan proses coding, database management, dan pengelolaan website (infrastruktur).

#### ***4. Pengujian dan Penyebaran (Testing and Deployment)***

Untuk mendeteksi apakah terdapat bug dalam sistem, harus ada tahap testing atau pengujian yang teliti, ketat, dan berulang. Tepat sebelum kamu melakukan setting website untuk terhubung ke server. Dalam tahap ini, website diuji dari segi fungsionalitas, kompatibilitas, kegunaan, kinerja, dan lain sebagainya. Tidak hanya itu, semua script kode program juga harus diuji untuk memastikan bahwa website bisa dimuat dan tampil dengan baik di semua platform dan perangkat.

#### ***5. Post-deployment dan Maintenance***

Begitu website diluncurkan, proses pengembangan web tidak berhenti begitu saja. Masih ada yang harus dilakukan setelah website sudah diluncurkan dan disebarluaskan. Mulai dari pembaruan umum (general updates), maintenance website, dan juga penambahan fitur baru apabila diperlukan. Post-deployment dan maintenance website ini berfungsi untuk terus meningkatkan fungsi dan fitur website seiring dengan adanya kebutuhan baru.

# **BAB III**

## **ANALISIS PEMECAHAN MASALAH**

### ***3.1 Langkah-langkah Pemecahan Masalah***

Dalam pengaplikasian sistem temu balik gambar yang berbasis aljabar vektor untuk Content-Based Image Retrieval (CBIR), langkah-langkah pemecahan masalahnya sebagai berikut:

#### ***3.1.1 CBIR dengan Parameter Warna***

##### **A. Upload dan Memproses Semua Gambar ke Ruang HSV dalam Bentuk Histogram**

Pada saat awal website berjalan, maka user diharuskan menginput dataset gambar terlebih dahulu. Setelah upload berhasil maka program akan langsung memproses dataset tersebut dari RGB ke ruang HSV dan nilainya akan disimpan semua.

##### **B. Proses konversi dari RGB ke HSV**

Proses ini bermula dengan membaca tiap gambar dari dataset dengan menggunakan library cv2 untuk membacanya menjadi sebuah matrix RGB. Setelah mendapatkan matrix gambarnya lakukan resizing image dengan library cv2 agar waktu *processing* lebih efisien .Setelah itu normalisasi nilai ke rentang 0-1 dan cari Cmax , Cmin , dan selisihnya. Setelah didapat lakukan proses mencari nilai HSV-nya.

##### **C. Ubah Menjadi Histogram HSV**

Proses menyimpan dalam bentuk histogram ini dipisah menjadi 3 histogram (H,S,V). Mula-mula siapkan 3 array kosong dengan panjang = bins,disini saya memilih bins = 8 karena mengikuti dengan referensi yang diberikan. Lalu cari inddengan mengiterasi membagi nilai H dengan nilai 360/bins (360 karena rentang H hingga 360) dan di cari minimal dari 8 untuk menjaga jaga agar nilainya melebih dari 8. Begitu juga dengan S

dan V namun karena rentangnya 0-1 maka 1/bins. Setelah itu setiap histogram pada indeks yang didapat ditambah 1, proses berlangsung hingga semua sudah masuk dalam histogramnya.

#### D. Compare dengan *Cosine Similarity*

Setelah semua dataset gambar diubah dalam bentuk histogram HSV maka berikutnya user menginput gambar yang akan dicari kemiripannya dengan gambar yang ada di dataset. Setelah gambar di input maka gambar akan di proses menjadi histogram HSV dulu. Setelah itu Histogram gambar inputan akan dicompare dengan semua gambar di dataset dengan menggunakan *Cosine Similarity*. Setelah itu setiap nilai kemiripannya akan disimpan dan di *sort* untuk mendapatkan nilai tertingginya. Maka setelah itu hasilnya akan terlihat gambar dengan persentase dari yang paling besar ke paling rendah.

### **3.1.2 CBIR dengan parameter Tekstur**

#### A. Upload dan memproses gambar menjadi greyscale

Setelah menerima image dari hasil upload user, maka program akan memproses dataset tersebut dari RGB menjadi greyscale dengan rumus sendiri.

#### B. Mengkonversi menjadi matrix co-occurrence

Setelah image diubah menjadi wujud greyscale, lalu setiap pixel yang ada dalam image greyscale tersebut akan dipakai untuk membuat matrix co-occurrence tersebut, dengan cara menjadikan pixel tersebut sebagai indeks dan menambahkan satu elemen setiap didapatkan indeks dari dua buah pixel yang bersebelahan.

#### C. Ekstraksi

Dari matrix co-occurrence yang telah dibuat, kini bisa didapatkan berbagai nilai ekstraksi greyscale, seperti *contrast*, *entropy*, dan *homogeneity* dengan rumus rumus tertentu.

#### D. *Cosine similarity*

Setelah mendapatkan nilai *contrast*, *entropy*, dan juga *homogeneity*, kita dapat membuat suatu vektor yang berdimensi 3 hal tersebut. Setelah itu, program hanya perlu menarik *contrast*, *entropy*, *homogeneity* dari image - image dataset, lalu membuatnya menjadi sebuah vektor juga, dan selanjutnya membandingkannya dengan vektor dari image yang diunggah.

### **3.2 Pemetaan Masalah ke Aljabar Geometri**

#### **3.2.1 CBIR dengan Parameter Warna**

Masalah pada CBIR warna ini adalah mencari solusi untuk mengimplementasikan kode programnya. Tujuan utama pada CBIR parameter warna ini adalah menemukan gambar-gambar dalam dataset yang mirip dengan gambar query berdasarkan karakteristik warnanya. Untuk itu pemetaan masalahnya dalam parameter warnanya sebagai berikut:

##### A. Konversi Warna RGB ke HSV

- a. Normalisasi semua nilai RGB ke rentang [0, 1].
- b. Hitung nilai Cmax, Cmin, dan  $\Delta$ .
- c. Konversi dari RGB ke ruang warna HSV.

##### B. Histogram Warna

- a. Hitung histogram HSV.
- b. Membuat histogramnya secara terpisah.
- c. Histogram H, Histogram V, Histogram S.

##### C. Cosine Similarity

- a. Hitung tingkat kemiripan dengan menggunakan Cosine Similarity.
- b. Gunakan vektor dan aljabar vektor untuk membandingkan gambar input dan dataset.

### **3.2.2 CBIR dengan Parameter Tekstur**

Masalah yang dihadapi adalah mencari solusi untuk Content-Based Image Retrieval (CBIR) dengan penekanan khusus pada parameter tekstur. Dalam konteks ini, tujuan akhirnya adalah berhasil menemukan gambar-gambar dalam dataset yang memiliki kemiripan tinggi dengan gambar query, dengan penilaian kesamaan yang dilakukan berdasarkan karakteristik tekstur yang dimiliki oleh gambar tersebut. Maka untuk itu, pemetaan masalahnya adalah sebagai berikut:

A. Konversi ke Grayscale:

- a. Konversi gambar ke skala keabuan untuk menghilangkan pengaruh warna dalam penentuan tekstur.

B. Kuantifikasi Grayscale:

- a. Lakukan kuantifikasi pada nilai grayscale.
- b. Gunakan matriks  $256 \times 256$  yang berkorespondensi dengan citra grayscale.

C. Co-occurrence Matrix:

- a. Hitung matriks co-occurrence dengan offset  $\Delta x$ .
- b. Gunakan nilai intensitas gambar dan posisi sebagai parameter.
- c. Gunakan nilai  $\theta$  pada  $0^\circ$ .

D. Ekstraksi Tekstur:

- a. Dapatkan kontrast, homogeneity, dan entropy dari matriks co-occurrence.
- b. Implementasikan persamaan yang sesuai dengan mendapatkan nilai-nilai ini.

E. Cosine Similarity:

- a. Gunakan vektor untuk merepresentasikan tekstur gambar.
- b. Hitung Cosine Similarity antara gambar input dan dataset.

### **3.3 Contoh Ilustrasi Kasus dan Penyelesaiannya**

#### **3.3.1 CBIR dengan Parameter Warna**

##### a. Ilustrasi Kasus

- User input gambar.
- Program mengonversi warna ke HSV.
- Menghitung histogram HSV.
- Hitung tingkat kemiripan dengan *Cosine Similarity*.

##### b. Penyelesaian:

Mengimplementasikan algoritma untuk konversi warna, perhitungan histogram, dan perhitungan Cosine Similarity.

#### **3.3.2 CBIR dengan Parameter Tekstur**

##### a. Contoh Ilustrasi Kasus:

- Pengguna memasukkan gambar.
- Program mengonversi ke Grayscale dan melakukan kuantifikasi.
- Matriks co-occurrence dihitung dengan offset yang sesuai.
- Ekstraksi tekstur (kontrast, homogeneity, entropy) dilakukan.
- Pengukuran tingkat kemiripan menggunakan Cosine Similarity.

##### b. Penyelesaian:

Implementasi algoritma untuk konversi Grayscale, kuantifikasi, perhitungan co-occurrence matrix, ekstraksi tekstur, dan perhitungan Cosine Similarity.

# BAB IV

## Implementasi dan Uji Coba

### 4.1 Pseudocode

#### 4.1.1 Warna

##### 4.1.1.1 Pseudocode histHSV

```
async function histHSV(gambar : image) -> array, array, array
    USE numpy
    bins <- 8
    gambar <- gambar / 255.0
    tinggi <- gambar.shape[0]
    lebar <- gambar.shape[1]
    hist_h <- numpy.zeros(bins)
    hist_s <- numpy.zeros(bins)
    hist_v <- numpy.zeros(bins)

    i traversal [0..tinggi-1] do
        j traversal [0..lebar-1] do
            r <- gambar[i, j, 2]
            g <- gambar[i, j, 1]
            b <- gambar[i, j, 0]
            Cmax <- max(r,g,b)
            Cmin <- min(r,g,b)
            selisih <- Cmax - Cmin
            if (selisih = 0) then
                h <- 0
            else if (Cmax = r1) then
                h <- 60 * (((g1 - b1) / selisih) % 6)
            else if (Cmax = g1) then
                h <- 60 * (((b1 - r1) / selisih) + 2)
            else
                h <- 60 * (((r1 - g1) / selisih) + 4)

            if (Cmax = 0) then
                s <- 0
            else
                s <- selisih / Cmax

            v <- Cmax
```

```

idx_h <- min(int(h / (360 / bins)), bins - 1)
idx_s <- min(int(s / (1 / bins)), bins - 1)
idx_v <- min(int(v / (1 / bins)), bins - 1)

hist_h[idx_h] <- hist_h[idx_h] + 1
hist_s[idx_s] <- hist_s[idx_s] + 1
hist_v[idx_v] <- hist_v[idx_v] + 1

totalh <- sum(hist_h)
totals <- sum(hist_s)
totalv <- sum(hist_v)

hist_h <- [h / totalh for h in hist_h]
hist_s <- [s / totals for s in hist_s]
hist_v <- [v / totalv for v in hist_v]

-> hist_h, hist_s, hist_v

```

#### 4.1.1.2 Pseudocode similarity

```

function similarity(hist1 : array, hist2 : array) -> float
    USE numpy
    summ <- 0
    panjangvek1 <- numpy.linalg.norm(hist1)
    panjangvek2 <- numpy.linalg.norm(hist2)

    i traversal [0..length(hist1)-1] do
        summ <- summ + hist1[i] * hist2[i]
    -> summ / (panjangvek1 * panjangvek2)

```

#### 4.1.1.3 Pseudocode colorCBIR

```

function colorCBIR(minim : float, file : file image, arrayobject : arrayobject) ->
arrayobject
    FROM .colorProcessor USE histHSV,similarity
    USE cv2
    USE test
    USE time
    USE numpy as np

    content <- await img.read()
    gambar <- cv2.imdecode(np.frombuffer(content, np.uint8),
cv2.IMREAD_COLOR)
    gambar_resized <- cv2.resize(gambar, (0, 0), fx = 0.5, fy = 0.5)

```

```

H1, S1, V1 <- warna.histHSV(gambar_resized)
arrayImgdanSim <- []
mulai <- time.time()
{for row in arrayobject}
data traversal arrayobject do
    H2 <- data['hist_h'],
    S2 <- data['hist_s']
    V2 <- data['hist_v']
    threshold <- int(minim)
    similarityValue <- (warna.similarity(H1, H2) + warna.similarity(S1, S2) +
    warna.similarity(V1, V2)) / 3 * 100

    if similarityValue >= threshold then
        if (not arrayImgdanSim) then
            arrayImgdanSim.append({
                "image": data['image'],
                "persentasePersamaan": similarityValue
            })
        else
            indeks <- 0
            while indeks < len(arrayImgdanSim) and
arrayImgdanSim[indeks]['persentasePersamaan'] > similarity do
                indeks < indeks + 1
                masuk <- {
                    "image": data['image'],
                    "persentasePersamaan": similarityValue
                }
                arrayImgdanSim.insert(indeks, masuk)
selesai <- time.time()
selisih <- selesai - mulai

hasil <-({{
    "result": arrayImgdanSim,
    "time": selisih
}})
-> hasil

```

#### 4.1.2 Tekstur

##### 4.1.2.1 Pseudocode grayscale\_image

```
function grayscale_image(image) ->
r, g, b <- splitImageChannels(image)
```

```

r <- r * 0.29
g <- g * 0.587
b <- b * 0.114
y <- calculateWeightedSum(r, g, b)
yInt <- convertToInteger(y)
-> yInt

```

#### 4.1.2.2 Pseudocode calculate\_texture\_features

```

function calculate_texture_features(image) -> vektor
    image <- grayscale_image(image)
    co_occurrence_matrix <- createEmptyMatrix(256, 256, dtype=int)
    height, width <- getDimensions(image)

    i traversal height-1 do
        j traversal width-2 do
            if j < width:
                co_occurrence_matrix[image[i, j], image[i, j+1]] += 1

    contrast <- calculateContrast(co_occurrence_matrix)
    entropy <- calculateEntropy(co_occurrence_matrix)
    homogeneity <- calculateHomogeneity(co_occurrence_matrix)

    -> contrast, entropy, homogeneity

```

#### 4.1.2.3 Pseudocode cosine\_similarity

```

function cosine_similarity(vector1, vector2) -> float
    dot_product <- dotProduct(vector1, vector2)
    norm_vector1 <- calculateNorm(vector1)
    norm_vector2 <- calculateNorm(vector2)

    similarity <- dot_product / (norm_vector1 * norm_vector2)

    -> similarity

```

#### 4.1.2.4 Pseudocode textureCBIR

```

async function textureCBIR(threshold, image, dataset) -> array
    content <- await image.read()
    gambar <- decodeImage(content)
    image_resize <- resizeImage(gambar, scale_factor=0.5)
    imageFeatures <- calculateTextureFeatures(image_resize)
    arrayakhir <- []

```

```

mulai <- time.time()

entry traversal dataset do
    comparedFeatures <- entry['contrast'], entry['entropy'], entry['homogeneity']
    similarityValue <- cosineSimilarity(imageFeatures, comparedFeatures) * 100
    threshold <- int(threshold)

    if similarityValue >= threshold:
        if not arrayakhir:
            arrayakhir.append({
                "image": entry['image'],
                "percentasePersamaan": similarityValue
            })
        else:
            indeks <- 0
            while indeks < len(arrayakhir) and arrayakhir[indeks]['percentasePersamaan'] >
similarityValue:
                indeks += 1
            masuk <- {
                "image": entry['image'],
                "percentasePersamaan": similarityValue
            }
            arrayakhir.insert(indeks, masuk)

selesai <- time.time()
selisih <- selesai - mulai
hasil <- {
    "result": arrayakhir,
    "time": selisih
}
-> hasil

```

#### ***4.1.3 Dataset Processing***

#### ***4.1.2.5 Pseudocode datasetFeatureExtractor***

```

async function datasetFeatureExtractor(arr) -> array ,array
    USE base64
    from .colorProcessor USE histHSV
    USE numpy as np
    USE cv2
    USE numpy as np
    USE base64
    from .textureProcessor USE calculate_texture_features

```

```

colorArray <- []
textureArray <- []

i,file traversal enumerate(arr) do
    content <- await file.read()
    gambar <- decodeImage(content)
    image_resize <- resizeImage(gambar, scale_factor=0.5)
    H, S, V <- calculateHSVHistogram(image_resize)
    _, buffer <- encodeImage(gambar)
    img_encode <- encodeImageToBase64(buffer)

    colorArray.append({
        "image": img_encode,
        "hist_h": H,
        "hist_s": S,
        "hist_v": V,
    })

contrast, entropy, homogeneity <- calculateTextureFeatures(image_resize)

textureArray.append({
    "image": img_encode,
    "contrast": contrast,
    "entropy": entropy,
    "homogeneity": homogeneity,
})

return colorArray, textureArray

```

## **4.2 Penjelasan dan Tatacara Penggunaan Program**

### **4.2.1 Warna**

Pada kode program warna terdapat 4 fungsi pembangun sistem temu balik gambar yang berfungsi sebagai berikut:

#### **4.2.1.1 histHSV**

Fungsi yang digunakan untuk mengekstrak nilai RGB dari suatu gambar lalu mengubahnya dalam bentuk HSV. Setelah menjadi bentuk HSV nilai-nilai tersebut dijadikan sebuah histogram secara terpisah (histogram H, histogram V, histogram S). Penggunaan fungsi ini hanya perlu memasukkan gambar yang akan diolah, maka return yang dihasilkan langsung sebuah histogram HSV.

#### **4.2.1.2 similarity**

Fungsi ini digunakan untuk menghitung cosine similarity dari 2 histogram yang dibandingkan. Penggunaan ketika mencari similarity yaitu dengan memasukkan kedua histogram H , histogram S , histogram V yang akan dibandingkan. Setelah itu hanya perlu menjumlah semua hasilnya dan dibagi dengan 3 untuk mendapatkan hasil akhir dari cosine similaritynya.

#### **4.2.1.3 ubahArrayGambarJadiHistHSV**

Fungsi ini digunakan untuk mengubah masing masing file gambar yang ada di array inputan lalu memproses setiap gambarnya menjadi histogram HSV dan menyimpannya dalam array of object yang berisi gambar asli dan histogram HSVnya. Penggunaannya dengan memasukkan array of file dan akan menerima hasil array of object yang berisi gambar dan histogramnya.

#### **4.2.1.4 mainColor**

Fungsi main ini digunakan untuk membandingkan gambar yang akan dicari dengan membandingkan dengan gambar di dataset yang sudah diolah menjadi histogram HSV. Setelah membandingkannya maka hasil akan di simpan dalam sebuah array lalu di simpan lagi dalam array of object yang akan menyimpan array compare dan waktu prosesnya. Penggunaan fungsi ini menerima 3 parameter , yaitu minim yang berarti hasil minimal compare persentase yang akan dikeluarkan, lalu file image yang akan dicompare dan yang terakhir ada arrayobject yang berisi gambar dan histogram HSV.

### **4.2.2 Tekstur**

Pada kode program tekstur terdapat 5 fungsi pembangun sistem temu balik gambar yang berfungsi sebagai berikut:

#### **4.2.2.1 ubahArrayGambarMenjadiListCooc**

fungsi ini digunakan untuk mengubah array berisi file image (dataset) menjadi array of object yang berisikan ['image'], ['contrast'], ['entropy'], ['homogeneity']. Hal ini bertujuan agar perbandingan gambar menjadi lebih cepat dan lebih efisien.

#### **4.2.2.2 grayscale\_image**

fungsi ini digunakan untuk mengubah parameter image yang sudah dimasukkan agar menjadi image greyscale, dimana warna RGB pada image tersebut dimanipulasi seperti ini: Grey = Red \* 0.29 + Green \* 0.587 + Blue \* 0.114. hal ini bertujuan untuk pembentukan co-occurrence matrix dimana

#### **4.2.2.3 calculate\_texture\_features**

fungsi ini digunakan untuk mencari nilai contrast, entropy, dan juga homogeneity dari sebuah co-occurrence matrix dari sebuah image yang sudah di greyscale. user tinggal memasukkan parameter image dan akan fungsi ini akan langsung mengeluarkan

#### **4.2.2.4 cosine\_similarity**

fungsi ini tentu saja bertujuan untuk menghitung persamaan dari vektor dua gambar berbeda, yaitu contrast, entropy dan juga homogeneity. fungsi ini kemudian menghitung dua vektor tersebut dengan menggunakan rumus *cosine similarity* dan nantinya hasil tersebut akan dipakai untuk menyatakan persentase persamaan dari kedua gambar tersebut.

#### **4.2.2.5 mainTekstur**

fungsi ini adalah akar dari semua fungsi untuk CBIR tekstur. Fungsi mainTekstur ini menerima input treshold, array of objects yang terdiri dari 4 elemen, yaitu ['image'], ['contrast'], ['entropy'], ['homogeneity'], dan juga input terakhir yaitu image referensi sebagai gambar yang dibandingkan dengan seluruh dataset. Selanjutnya, fungsi ini akan menghitung dan juga membandingkan 3 vektor tersebut dengan vektor milik foto referensi, lalu jika persamaannya melebihi treshold, image dari dataset tersebut lalu akan dimasukkan ke dalam array of object lagi dimana elemennya yaitu ['image'], dan ['persentasePersamaan'].

#### **4.2.3 Tata Cara Penggunaan**

##### **4.2.3.1 Tata Cara FrontEnd**

1. buka di terminal.
2. ketik “cd src\website” lalu enter:

```
>> cd src\website
```

3. ketiklah “np i” dan enter

```
>> npm i
```

4. setelah itu, ketiklah “npm run dev” lalu enter, dan tunggu sampai terminal berhenti.

```
>> npm run dev
```

5. lalu, bukalah browser an ketik “Localhost:3000”

```
>> Localhost:3000
```

##### **4.2.3.2 Tata Cara BackEnd**

1. buka di terminal.
2. ketik “cd src\backend” lalu enter:

```
>> cd src\backend
```

3. lalu, ketiklah “pip install -r requirements.txt”

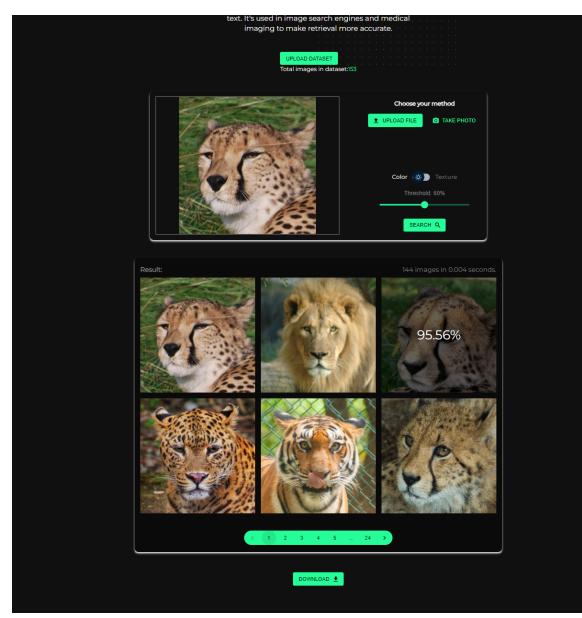
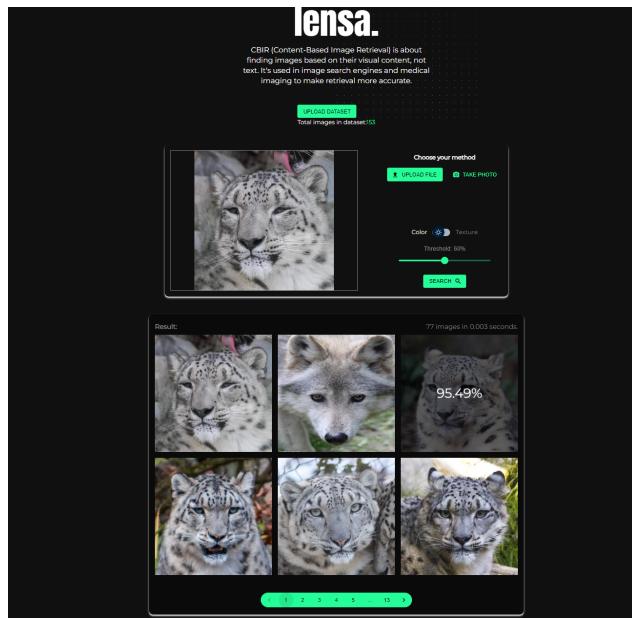
```
>> pip install -r requirements.txt
```

4. setelah itu “python3 -m uvicorn main:app -- reload” lalu ketik enter.

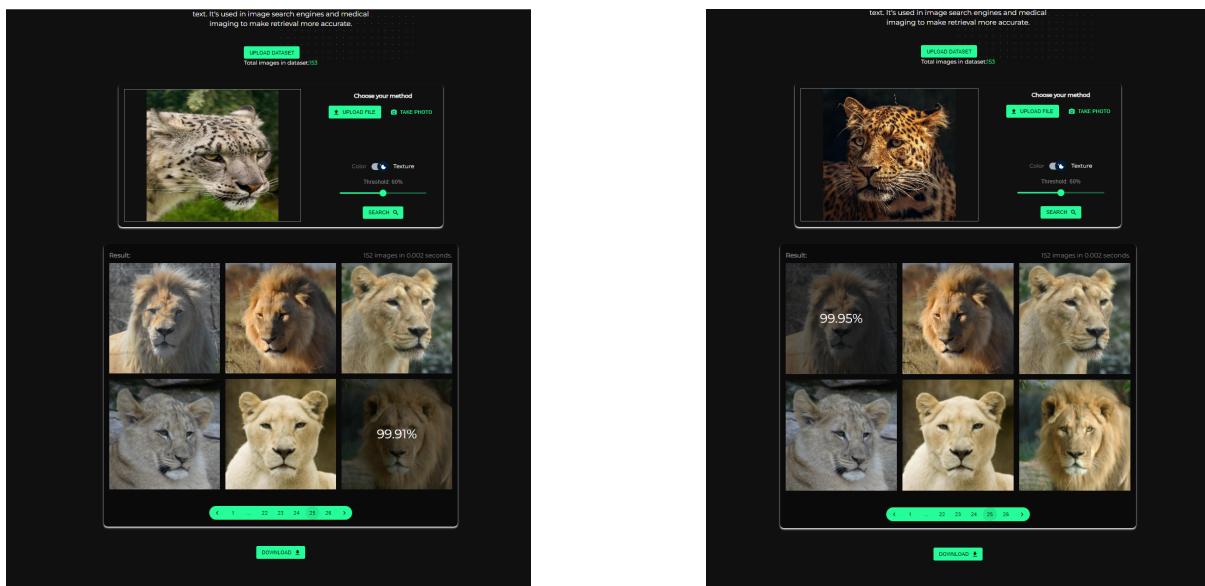
```
>> python3 -m uvicorn main:app --reload
```

## 4.3 Hasil Pengujian

### 4.3.1 Warna



### 4.3.2 Tekstur



## 4.4 Analisis Color dan Tekstur

Untuk membandingkan mana yang lebih akurat antara CBIR warna dan tekstur ini tidak memiliki jawaban yang valid atau pasti. Hal ini dikarenakan CBIR warna atau tekstur dapat dipengaruhi oleh beberapa faktor, seperti sifat data, jenis gambar, dan tujuan yang akan dicapai. Baik CBIR berbasis warna maupun tekstur memiliki kelebihan dan kelemahan masing-masing sebagai berikut:

### A. Warna

#### a. Kelebihan:

- i. Warna merupakan fitur yang sangat mencolok, jelas, dan mudah diterima manusia.
- ii. Warna sangat bagus untuk mencari sesuatu yang sangat bergantung pada warna seperti fashion.

#### b. Kelemahan:

- i. Sangat rentan dengan kondisi lingkungan di sekitar dan akan memengaruhi hasilnya.
- ii. Tidak efektif untuk gambar yang memiliki struktur serupa tetapi dengan perbedaan warna yang signifikan.
- iii. Kurang mampu menangkap informasi semantik yang terkandung dalam gambar.

### B. Tekstur

#### a. Kelebihan:

- i. Lebih stabil dalam menghadapi perubahan pencahayaan karena tidak terkait dengan perubahan warna.
- ii. Sesuai untuk gambar dengan pola yang serupa meskipun memiliki variasi warna yang signifikan.

- iii. Mampu menggambarkan informasi semantik dengan lebih baik dalam beberapa situasi.
- b. Kelebihan:
  - i. Pengolahan tekstur mungkin lebih rumit dibandingkan dengan pemrosesan warna.
  - ii. Bersifat subjektif dalam beberapa konteks, karena persepsi terhadap tekstur dapat bervariasi antara individu.
  - iii. Kurang efektif untuk gambar yang lebih dipengaruhi oleh warna daripada oleh tekstur.

Maka untuk tekstur maupun warna tidak dapat disimpulkan salah satunya lebih baik. Pada suatu *case* tertentu dapat tekstur yang lebih baik, dan begitu juga sebaliknya warna juga dapat menjadi lebih baik ketika *case* tertentu.

# **BAB V**

## **KESIMPULAN**

### **5.1 Kesimpulan**

Dari tugas besar ini, banyak hal yang dapat kami pelajari, seperti implementasi CBIR menggunakan python dan library-library nya, beberapa cara menerapkan search dengan CBIR, cara mengembangkan website, penggunaan teknologi-teknologi website, *best practice* dalam proyek pemrograman bersama tim, penggunaan git, dan lainnya. Kami menggunakan Github sebagai *version control system* dengan tujuan mempermudah proses pengerjaan. Kesimpulannya, kami berhasil mengimplementasikan setiap spesifikasi wajib dan beberapa spesifikasi bonus yang diberikan dalam waktu yang disediakan.

### **5.2 Saran**

Berikut adalah beberapa saran pengembangan dari kelompok kami.

- a. Mempelajari git untuk mempermudah proses pengerjaan tugas besar secara bersama-sama.
- b. Mendiskusikan *flow* program terlebih dahulu untuk menghindari miskomunikasi saat proses integrasi.
- c. Mempelajari terlebih dahulu teori terkait masalah yang diberikan.
- d. Memecah tugas besar menjadi beberapa milestone.

### **5.3 Komentar**

Berikut adalah beberapa komentar kami terhadap tugas ini.

- a. Tugas besar cukup mudah dan menurut kami sangat menyenangkan.
- b. Relevansi dengan materi Algeo sangat besar.
- c. Bisa dibilang tugas ini cukup erat hubungannya dengan web development, sehingga bagi yang tidak mempunyai pengalaman akan kesusahan.

#### **5.4 Refleksi**

Berikut adalah beberapa refleksi dari kami.

- a. Tugas seharusnya bisa lebih cepat diselesaikan jika terlebih dahulu mempelajari teori terkait CBIR.
- b. Pembagian tugas sudah benar secara porsi dan kemampuan.

## **DAFTAR PUSTAKA**

Yunus, M. 2020. Feature Extraction: Gray Level Co-occurrence Matrix (GLCM).

<https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1>

# **HASIL**

Link GITHUB:

<https://github.com/owenthe10x/Algeo02-22131.git>

Link Youtube:

<https://youtu.be/D4FQh5uTbtk>