

# **LAPORAN TUGAS BESAR II**

**Aljabar Linier dan Geometri**

**Semester 2 Tahun Ajaran 2023/2024**

Oleh:

Shafiq Irvansyah 13522003

Ahmad Naufal Ramadhan 13522005

Yusuf Ardian Sandi 13522015



**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2023**

## DAFTAR ISI

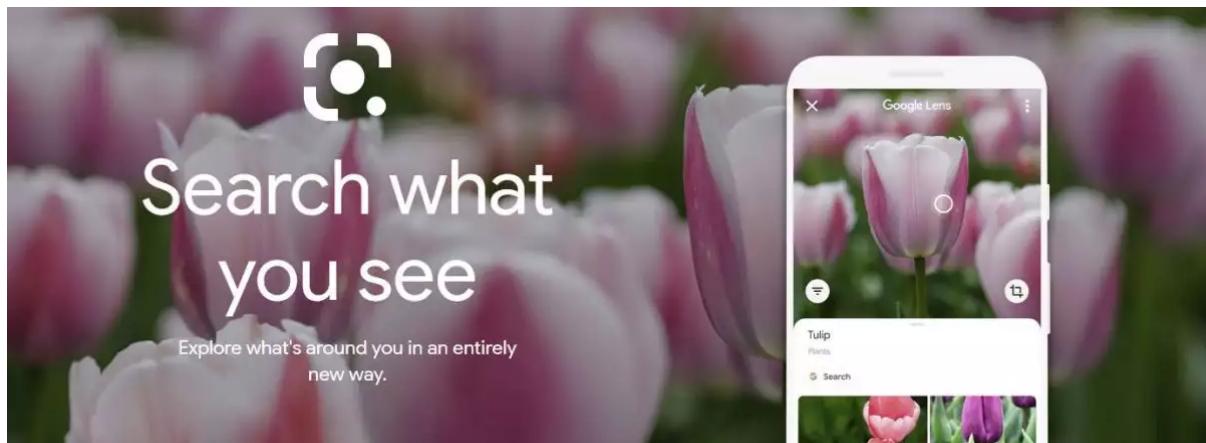
<b>LAPORAN TUGAS BESAR II.....</b>	<b>1</b>
DAFTAR ISI.....	2
BAB I	
DESKRIPSI MASALAH.....	3
Abstraksi.....	3
Spesifikasi Tugas.....	3
BAB II	
LANDASAN TEORI.....	5
Content-Base Information Retrieval (CBIR).....	5
Perkalian Dot Vektor.....	5
Norma/Magnitude Vektor.....	5
Cosine Similarity.....	6
Matriks Transpose.....	6
Penjelasan Singkat Mengenai Pengembangan Sebuah Website.....	6
BAB III	
ANALISIS PEMECAHAN MASALAH.....	7
Langkah-Langkah Pemecahan Masalah.....	7
Proses Pemetaan Masalah Menjadi Elemen-Elemen Pada Aljabar Geometri.....	7
Contoh Ilustrasi Kasus Dan Penyelesaiannya.....	8
BAB IV	
IMPLEMENTASI DAN UJI COBA.....	10
Implementasi Program Utama.....	10
Penjelasan Struktur Program Berdasarkan Spesifikasi.....	13
Penjelasan Tata Cara Penggunaan Program.....	14
Hasil Pengujian.....	14
Analisis Dari Desain Solusi Algoritma Pencarian.....	18
BAB V	
PENUTUP.....	20
Kesimpulan.....	20
Saran.....	20
Komentar atau Tanggapan.....	20
Refleksi Terhadap Tugas Besar.....	20
Ruang Perbaikan atau Pengembangan.....	20
DAFTAR PUSTAKA.....	22
LAMPIRAN.....	23

## BAB I

### DESKRIPSI MASALAH

#### 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, kami 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.

#### 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. Dataset gambar dapat diunduh secara mandiri melalui pranala [berikut](#). Akan tetapi peserta diperbolehkan untuk menggunakan dataset lain yang telah dipersiapkan oleh kelompok masing-masing.
3. Program menampilkan gambar citra gambar yang dipilih oleh pengguna.
4. 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.
5. Program mulai melakukan perhitungan nilai kecocokan antara image masukan dengan dataset image berdasarkan parameter yang telah dipilih (warna atau tekstur).
6. Program dapat menampilkan nilai kecocokan antara image masukan dengan setiap gambar dalam dataset.
7. 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.
8. 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.
9. Program dapat menampilkan **Jumlah gambar** yang memenuhi kondisi tingkat kemiripan serta **waktu eksekusi**.

## BAB II

### LANDASAN TEORI

#### ***Content-Based Information Retrieval (CBIR)***

Content-Based Information Retrieval (CBIR) adalah pendekatan dalam sistem temu kembali informasi yang berfokus pada analisis dan pemahaman terhadap konten atau isi dari data. Dalam CBIR, pencarian dilakukan berdasarkan karakteristik intrinsik dari data itu sendiri, seperti teks, gambar, suara, atau data multimedia lainnya. Sistem CBIR menggunakan representasi konten atau fitur-fitur khusus dari data untuk membangun indeks atau model yang memungkinkan pencarian berdasarkan kesamaan konten. Misalnya, dalam CBIR gambar, fitur-fitur seperti warna dan tekstur dapat diekstraksi dan digunakan untuk mencocokkan gambar-gambar yang serupa. Pendekatan ini memberikan solusi untuk pencarian yang lebih kontekstual dan mendalam, memungkinkan sistem untuk memahami dan merespon terhadap karakteristik intrinsik dari data yang dicari, tanpa tergantung pada metadata atau informasi eksternal.

#### **Perkalian Dot Vektor**

Perkalian dot vektor adalah operasi matematika yang menggabungkan elemen-elemen dua vektor untuk menghasilkan satu nilai tunggal. Dalam konteks matematika linier, perkalian dot vektor dilakukan dengan mengalikan setiap elemen yang sesuai dari dua vektor, dan kemudian menjumlahkan hasilnya. Misalnya, jika kita memiliki dua vektor A dan B dengan elemen-elemen  $[a_1, a_2, \dots, a_n]$  dan  $[b_1, b_2, \dots, b_n]$ , maka perkalian dot vektor ( $A \cdot B$ ) dihitung sebagai berikut:

$$A \cdot B = a_1 \cdot b_1 + a_2 \cdot b_2 + \dots + a_n \cdot b_n$$

Perkalian dot vektor sangat berguna dalam berbagai aplikasi, seperti komputasi linear algebra, machine learning, dan pemrosesan sinyal.

#### **Norma/Magnitude Vektor**

Norma atau magnitude vektor adalah konsep dalam matematika yang mengukur "ukuran" atau panjang vektor. Norma vektor sering kali dihitung untuk mendapatkan informasi tentang besar atau intensitas vektor dalam ruang. Terdapat beberapa jenis norma, tetapi yang paling umum adalah norma Euclidean (atau L2-norm) yang dihitung dengan menggunakan akar kuadrat dari jumlah kuadrat komponen vektor. Untuk suatu vektor  $v$  dengan komponen

$$v_1, v_2, \dots, v_n$$

, norma Euclidean didefinisikan sebagai

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

Norma vektor mencerminkan jarak dari titik awal vektor ke titik akhirnya dalam ruang.

## ***Cosine Similarity***

*Cosine similarity* adalah metode pengukuran kesamaan antara dua vektor dalam ruang berdimensi banyak dengan mengukur kosinus sudut antara vektor-vektor tersebut. Konsep ini banyak digunakan dalam pemrosesan teks dan analisis data untuk mengevaluasi sejauh mana dua dokumen atau set data serupa. Dalam cosine similarity, nilai similaritas antara dua vektor dihitung dengan membagi hasil perkalian dot vektor tersebut dengan produk norma Euclidean dari masing-masing vektor. Semakin dekat nilai cosine similarity ke 1, semakin serupa vektor-vektor tersebut. Metode ini sangat berguna dalam *clustering* dokumen, sistem rekomendasi, dan analisis pola dalam data multidimensi.

## **Matriks Transpose**

Teori matriks transpose merupakan konsep dasar dalam aljabar linear yang melibatkan operasi pengubahan baris dan kolom suatu matriks. Transpose dari suatu matriks diperoleh dengan menukar baris dan kolom matriks tersebut. Jika matriks awalnya dinyatakan sebagai matriks  $A$  dengan elemen-elemen  $a_{ij}$ , maka transpose dari  $A$ , disimbolkan sebagai  $A^T$ , akan memiliki elemen-elemen yang disusun ulang sehingga elemen pada baris ke- $i$  dan kolom ke- $j$  matriks  $A$  akan menjadi elemen pada baris ke- $j$  dan kolom ke- $i$  dalam matriks transpose  $A^T$ .

## **Penjelasan Singkat Mengenai Pengembangan Sebuah *Website***

Kami menggunakan *framework* Django untuk membuat *website* dengan harapan bisa mengeksplorasi *framework* lebih jauh dari apa yang sudah pernah kami pakai. Django adalah sebuah *framework web open-source* yang ditulis dalam bahasa pemrograman Python. Dengan bahasa Python, kami telah terbiasa menggunakan khususnya sehingga ini dapat mempermudah kami dalam mengembangkan program sesuai dengan [deskripsi masalah](#). Selain mudah dipelajari, bahasa Python ini juga terkenal dengan *library*-nya khusus untuk pemrosesan gambar, yaitu NumPy, PIL, dan OpenCV. Walaupun begitu, salah satu kelemahan bahasa ini adalah proses eksekusi program yang dilakukan tidak secepat bahasa lain. Dalam membuat *website* ini, kami juga menggunakan HTML dan CSS dalam membuat struktur konten, isi konten, serta gaya dari tampilan konten tersebut.

## **BAB III**

### **ANALISIS PEMECAHAN MASALAH**

#### **Langkah-Langkah Pemecahan Masalah**

Sebuah citra terbentuk dari elemen-elemen kecil yang disebut piksel. Setiap piksel terdiri dari tiga komponen warna utama, yaitu merah (red), hijau (green), dan biru (blue), yang direpresentasikan oleh angka-angka tertentu. Dengan menggunakan informasi ini, kita dapat membentuk suatu matriks yang menyimpan ketiga nilai tersebut, menciptakan representasi matematis dari sebuah citra.

Metode CBIR dengan fitur warna memanfaatkan informasi komponen warna tersebut untuk mengindikasikan sejauh mana kemiripan antara suatu citra dengan citra lainnya. Pendekatan dengan fitur warna menggunakan komponen citra dalam format Hue-Saturation-Value (HSV), sehingga citra harus diubah ke dalam format ini terlebih dahulu. Penilaian kemiripan dari suatu citra dapat dicari dengan menggunakan cosine similarity. Agar dapat dihitung kemiripannya, representasi matriks dari citra diubah menjadi bentuk histogram, menciptakan vektor histogram dari matriks tersebut. Cosine similarity dihitung dengan membandingkan kedua vektor tersebut dan mengevaluasi hasilnya. Semakin mendekati nilai 1, kedua citra dianggap semakin mirip, sedangkan semakin mendekati nilai 0 menunjukkan tingkat kemiripan yang lebih rendah.

Metode CBIR dengan fitur tekstur memanfaatkan pola, distribusi spasial, susunan warna, dan intensitas dalam membedakan tekstur gambar secara visual. Informasi-informasi tersebut tidak dapat diakses melalui kalkulasi statistik seperti mean, median, atau standar deviasi. Oleh karena itu, dalam pendekatan CBIR untuk fitur tekstur, digunakan Gray-level co-occurrence matrix (GLCM) sebagai solusi dari permasalahan tersebut. GLCM menggambarkan hubungan antara dua pixel sejajar dengan intensitas keabuan, jarak, dan sudut tertentu. Variasi sudut, seperti  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ ,  $135^\circ$ ,  $180^\circ$ ,  $225^\circ$ ,  $270^\circ$ , atau  $315^\circ$ , dapat diterapkan pada GLCM. Jarak, yang didefinisikan adalah sebagai jumlah pixel antara pixel referensi dan tetangga. GLCM kemudian dapat digunakan untuk mendapatkan komponen vektor ekstraksi tekstur, seperti contrast, homogeneity, dan entropy. Vektor tersebut kemudian dibandingkan menggunakan cosine similarity untuk menghitung persentase kemiripan antara gambar-gambar tersebut.

#### **Proses Pemetaan Masalah Menjadi Elemen-Elemen Pada Aljabar Geometri**

CBIR dengan parameter warna:

1. Diambil nilai RGB dari suatu gambar dengan format array sehingga data ini bisa dianggap sebagai sebuah vektor.
2. Vektor tersebut dinormalisasi dengan melakukan salah satu operasi vektor, yaitu pembagian vektor.
3. Dicari nilai maksimum dan nilai minimum dari setiap indeks *array*-nya dimana format dari kedua nilai tersebut juga seperti sebuah vektor.
4. Variabel delta dihasilkan dengan melakukan operasi pengurangan vektor nilai maksimum oleh nilai minimum.
5. Berdasarkan rumus yang telah diberikan, hitung masing-masing nilai H, S, dan V dari vektor-vektor R, G, B yang dibuat.

6. Dari data H, S, dan V, masing-masing dibuat histogramnya menggunakan *library* NumPy dan data histogram ini juga bisa dianggap sebagai vektor.
7. Untuk membandingkan dengan *cosine similarity*, data histogram dari dataset akan dikalikan dengan data histogram dari gambar yang ingin dicari dengan perkalian dot vektor serta membaginya dengan norma/*magnitude* vektor/data tersebut.

CBIR dengan parameter tekstur:

1. Diambil nilai RGB dari suatu gambar dalam format array sehingga data ini dapat dianggap sebagai vektor. Dari keseluruhan gambar diperoleh *Matrix of RGB*.
2. Vektor nilai RGB dikonversi menjadi *Grayscale* sehingga diperoleh Matriks of *Grayscale*.
3. Dari *Matrix of grayscale*, Matriks *co-occurrence* dicari dengan nilai  $\theta = 0$  dan *distance* = 1.
4. Matriks *co-occurrence* tersebut kemudian dijumlahkan dengan transposnya dan dinormalisasi dengan membagi semua elemen matriks dengan jumlah dari semua elemen matriks.
5. Dari Matriks co-occurrence, nilai Contrast, Homogeneity, dan Entropy dicari dengan rumusnya masing-masing. Ketiga nilai tersebut dibentuk menjadi vektor komponen ekstraksi tekstur.
6. Vektor komponen ekstraksi tekstur kemudian dinormalisasi dengan *Gaussian Normalization* sehingga bobot tiap komponen vektor sama, yaitu dengan mengurangi semua elemen vektor dengan *mean* dari vektor tersebut kemudian dibagi dengan standar deviasi dari vektor tersebut .
7. Untuk membandingkan dengan *cosine similarity*, vektor komponen ekstraksi tekstur yang sudah di-*Gaussian Normalize* akan dikalikan dengan vektor komponen ekstraksi tekstur yang sudah di-*Gaussian Normalize* dari gambar yang ingin dicari dengan perkalian dot vektor, serta membaginya dengan norma/*magnitude* vektor/data tersebut.

### **Contoh Ilustrasi Kasus Dan Penyelesaiannya**

CBIR dengan parameter warna

Salah satu ilustrasi kasus dari website yang kami buat adalah ingin membandingkan citra dengan parameter warna. Semakin mirip komposisi warna dari suatu citra maka semakin mirip citra yang dibandingkan.

Penyelesaian:

1. Ubah citra yang ingin dibandingkan ke dalam matriks warna RGB lalu ubah ke dalam format HSV.
2. Hitung histogram nilai HSV dari masing-masing citra.
3. Bandingkan kedua histogram citra. Semakin mirip histogram yang dihasilkan maka semakin mirip citra yang dibandingkan.
4. Tingkat kemiripan citra yang dibandingkan dapat direpresentasikan dengan nilai *cosine* dari kedua histogram citra yang dibandingkan.

CBIR dengan parameter tekstur

Salah satu ilustrasi kasus dari website yang kami buat adalah ingin membandingkan citra dengan parameter tekstur. Ingin diambil nilai tekstur citra yang paling mirip dengan tekstur citra yang dibandingkan.

Penyelesaian:

1. Ubah citra yang ingin dibandingkan ke dalam matriks warna RGB lalu ubah ke dalam format grayscale.
2. Hitung co-occurrence dari masing-masing citra, kemudian tambahkan dengan transposnya lalu dinormalisasi.
3. Hitung komponen vektor ekstraksi tekstur, pada kasus ini, komponen yang dihitung adalah contrast, homogeneity, dan entropy.
4. Bandingkan vektor komponen ekstraksi tekstur antara dua gambar yang ingin dibandingkan dengan menggunakan cosine similarity.

## BAB IV

### IMPLEMENTASI DAN UJI COBA

#### Implementasi Program Utama

```
### color.py ###

use numpy as np

type: <histogram: (list of integer: hist,
                    list of float: bin_edges)>

function rgb_hsv_hist(list of list of integer: rgb_image) ->
tuple: (histogram: hist_h, histogram: hist_s, histogram:
hist_v)
{- Fungsi ini mengubah data RGB dari suatu image menjadi data histogram H, S, dan V. Menerima masukkan berupa image dengan format list of RGB. Menghasilkan keluaran berupa histogram hist_h, hist_s, dan hist_v. -}
```

#### KAMUS LOKAL

normalized\_image, Cmax, Cmin, delta: list of list of float  
H, V, S: list of float  
Hist\_h, hist\_s, hist\_v: histogram

#### ALGORITMA

```
normalized_image <- rgb_image / 255.0

Cmax <- np.max(normalized_image, axis <- (-1))
Cmin <- np.min(normalized_image, axis <- (-1))
delta <- Cmax - Cmin

H <- np.zeros_like(Cmax)

i traversal [0 ... length(H)-1]
if (Cmax = R) then
    H[i] <- 60 * ( ((G - B) / delta) mod 6 )
else if (Cmax = G) then
    H[i] <- 60 * ( ((B - R) / delta) + 2 )
else if (Cmax = B) then
    H[i] <- 60 * ( ((R - G) / delta) + 4 )

H <- (H / 6.0) % 1.0

S <- np.zeros_like(Cmax)
S[Cmax == 0] <- delta[Cmax == 0] / Cmax[Cmax == 0]
```

```

V <- Cmax

hist_h <- np.histogram(
    H,
    Bins <- [
        0,
        25 / 360,
        40 / 360,
        120 / 360,
        190 / 360,
        270 / 360,
        295 / 360,
        315 / 360,
        360 / 360,
    ],
)
hist_s = np.histogram(S, bins <- [0, 0.2, 0.7, 1])
hist_v = np.histogram(V, bins <- [0, 0.2, 0.7, 1])

-> hist_h, hist_v, hist_s

```

```

#### texture.py ####
USE numpy as np

function img_to_grayscale(string image_path) -> Matrix of
integer
{- Fungsi ini mengubah data RGB dari suatu image menjadi
grayscale. Menerima masukkan berupa path image dengan format
string. Menghasilkan keluaran berupa matriks grayscale. -}

KAMUS LOKAL
img, grayscale: Matrix of integer
rgb_channels : Matrix of list of integer

ALGORITMA
img <- np.array(Image.open(image_path))
rgb_channels <- img[..., :3]

grayscale <- np.dot(rgb_channels, [0.299, 0.587,
0.114]).astype(integer)

-> grayscale

```

```
function co_occurrence (Matrix of integer gray_pict)->Matrix of float
```

{- Fungsi ini membuat Matriks co\_occurrence dari Matriks grayscale. Menerima masukan berupa *gray picture* dalam format Matriks integer. Menghasilkan keluaran berupa Matriks co\_occurrence -}

#### KAMUS LOKAL

```
gray_pict : Matrix of integer
co_occurrence : Matrix of float
i, j, height, width : integer
```

#### ALGORITMA

```
co_occurrence <- np.zeros((256, 256), dtype=integer)
```

```
gray_pict <- np.array(gray_pict)
height, width <- gray_pict.shape
```

```
i traversal [0..height-1]
j traversal [0..width-1]
    co_occurrence[gray_pict[i][j]][gray_pict[i][j+1]] <-
        co_occurrence[gray_pict[i][j]] [gray_pict[i][j+1]] + 1

co_occurrence <- (co_occurrence + co_occurrence.T) /
                    np.sum(co_occurrence)
-> co_occurrence
```

```
function gaussian_normalize (list of float data)->list of float
```

{- Fungsi ini melakukan normalisasi Gaussian pada vektor dalam bentuk list of float sehingga bobot tiap elemen vektor setara. Menerima masukan berupa data/vektor dalam bentuk list of float. Menghasilkan keluaran berupa vektor dalam bentuk list of float -}

#### KAMUS LOKAL

```
mean, std_dev, normalized_data : list of float
```

#### ALGORITMA

```
mean <- np.mean(data)
std_dev <- np.std(data)
normalized_data <- (data-mean) / std_dev
```

```

-> normalized_data

function component (Matrix of float matrix) -> list of float
{- Fungsi ini membentuk komponen ekstraksi tekstur yaitu
contrast, homogeneity, dan entropy. Menerima masukan berupa
matrix co_occurrence dalam bentuk Matrix of float. Menghasilkan
keluaran berupa vektor komponen ekstraksi tekstur dalam bentuk
list of float -}

KAMUS LOKAL
i, j, diff : list of integer
contrast, homogeneity, entropy : float
nonzero_elements : Matrix of float

ALGORITMA
matrix <- np.array(matrix)
i,j <- np.indices(matrix.shape)
diff <- i-j

contrast <- np.sum(matrix * pow(diff,2))
homogeneity <- np.sum(matrix / (1 + pow(diff,2)))
nonzero_elements <- (matrix[matrix != 0])
entropy <- np.sum(nonzero_elements*np.log10(nonzero_elements))

-> [contrast, homogeneity, entropy]

```

## Penjelasan Struktur Program Berdasarkan Spesifikasi

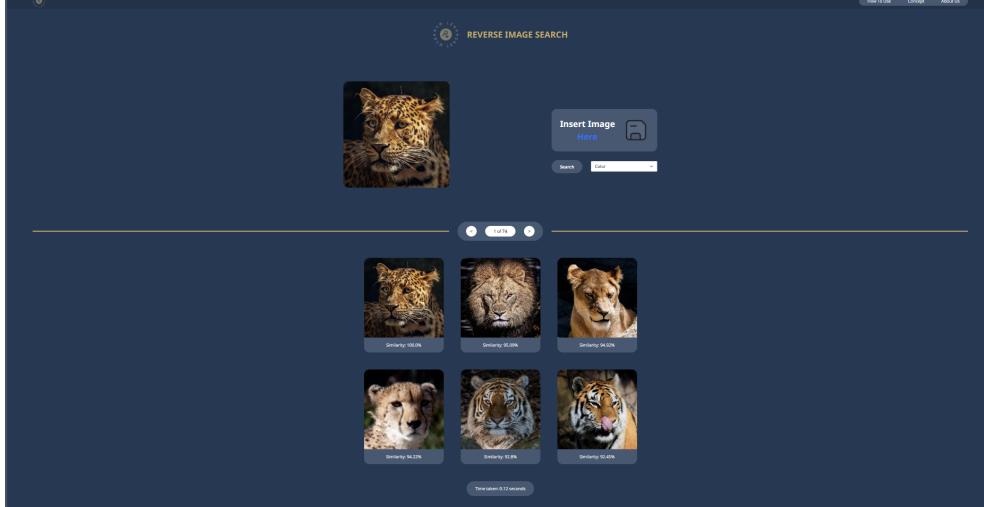
Struktur program yang kami terapkan mengikuti format program Django. Kami membuat struktur folder menggunakan perintah django-admin di dalam direktori src. Aplikasi Django kami diberi nama pages. Di dalam direktori pages, kami membuat folder static dan templates yang berfungsi untuk menyimpan data statis dari situs web. Folder static berisi data statis seperti dataset dan cache hasil pemrosesan dalam format json. Sementara itu, folder templates berisi file HTML yang berfungsi sebagai templat yang dapat disesuaikan sesuai kebutuhan. Kami juga menyertakan file forms.py untuk membuat form yang dapat mengirimkan POST request ke server. Logika utama dalam pemrosesan citra terdapat di dalam views.py. Logika-logika tersebut diorganisir berdasarkan fungsi dari masing-masing proses, kemudian digabungkan dalam views index. Setiap fungsi dalam views.py yang mengembalikan hasil dari fungsi render berfungsi untuk me-render halaman situs web.

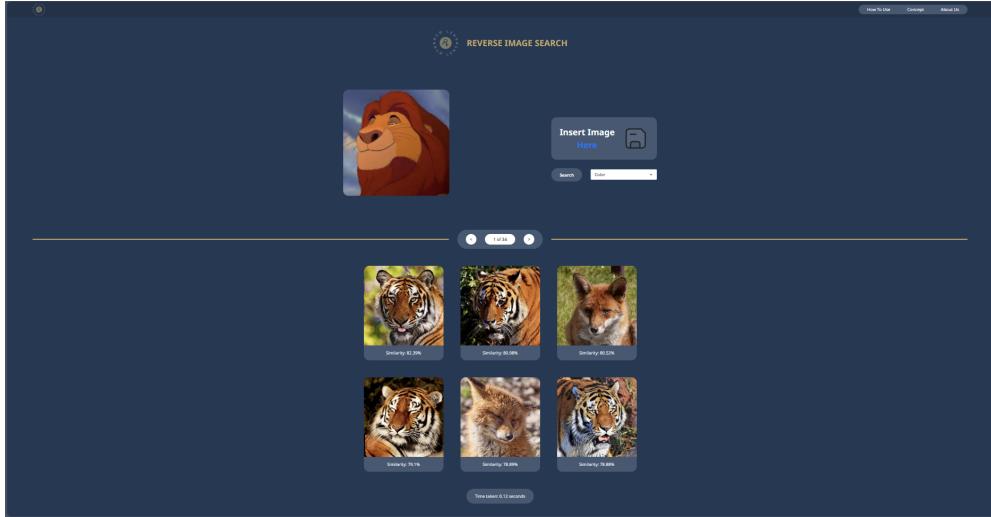
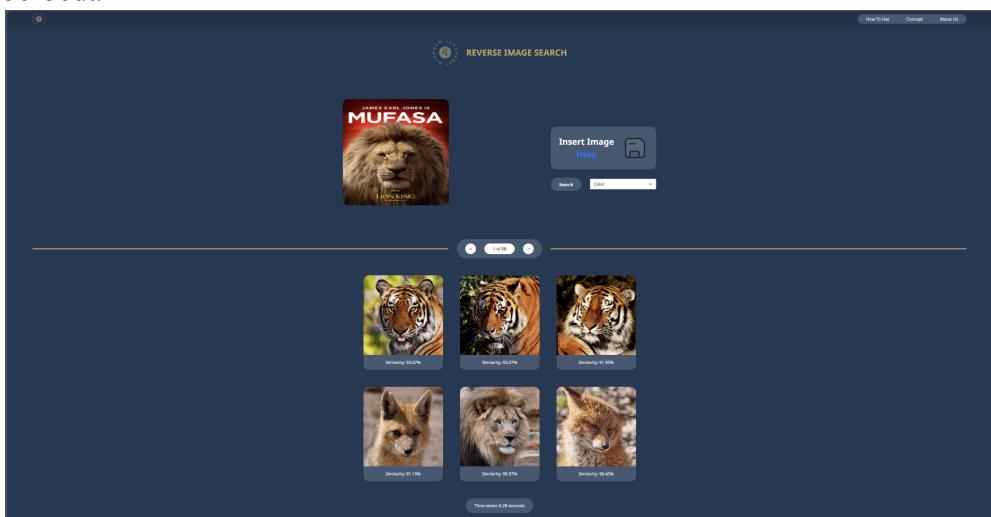
## Penjelasan Tata Cara Penggunaan Program

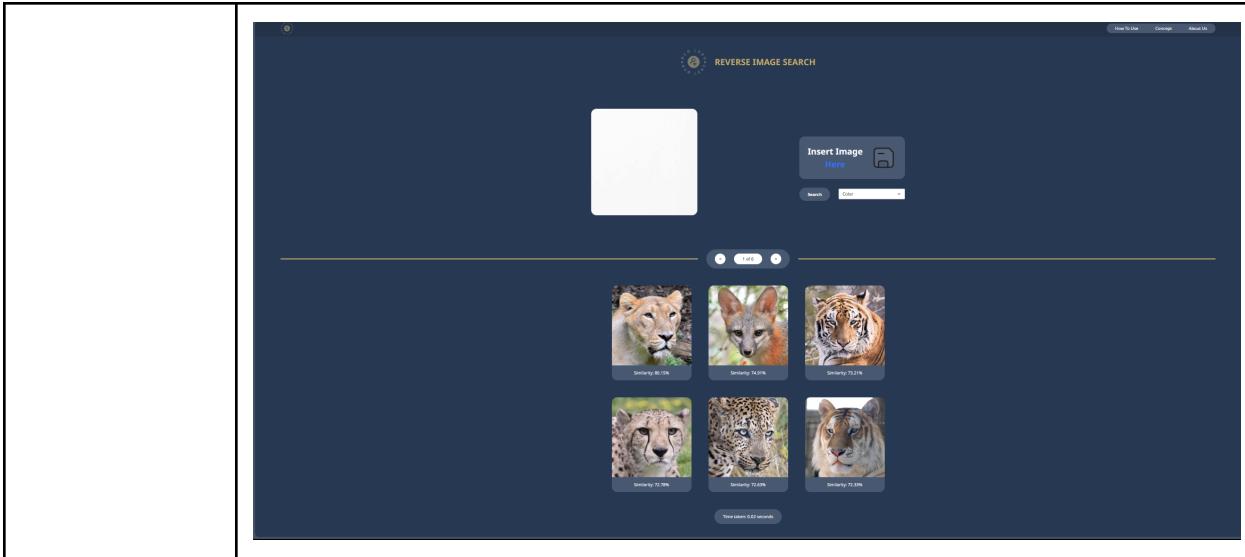
Pada [github kami](#), sudah dijelaskan bagaimana menjalankan program melalui *website*. Setelah melakukan semua prosedur yang tertera, akan langsung ditampilkan halaman *Home* dari *website* kami. Pada *website* ini, kami membagi menjadi empat halaman, yaitu *Home*, *Concept*, *How to use*, dan *About us*. Halaman *Home* berisi program utama yaitu *search engine* dari CBIR dimana *user* akan memberikan masukkan dataset gambar beserta gambar yang ingin dicari. Halaman *Concept* berisi penjelasan konsep *search engine* yang kami gunakan secara singkat seputar CBIR. Halaman *How to use* ialah halaman yang berisi tata cara menggunakan *search engine* pada *website* ini dengan menyertakan visualisasinya. Terakhir, halaman *About us* memuat profil pembuat *website*. Pada navigasi bar yang ada di atas, terdapat pilihan halaman *Home*, *Concept*, *How to use*, dan *About us* yang dapat dituju oleh user dengan tombol logo sebagai *Home page*.

Pada *Home page*, terdapat lima fitur utama yang disediakan. Pertama, pengguna dapat mengunggah dataset yang dimiliki ke *website*. Kedua, pengguna dapat mengunggah gambar yang ingin dicari ke *website*. Ketiga, pengguna dapat memilih dua parameter pencarian, yaitu warna atau tekstur. Keempat, hasil pencarian dengan kemiripan diatas 60% akan ditampilkan dengan memberikan persentase kemiripannya dan mengurutkannya dari persentase terbesar ke persentase terkecil. Terakhir, dalam mengeksplorasi hasil pencarian, pengguna dapat menggunakan fitur *pagination* yang ada pada *website* sehingga pengguna tidak perlu terus-menerus melakukan *scrolling* hingga hasil pencarian yang terakhir.

## Hasil Pengujian

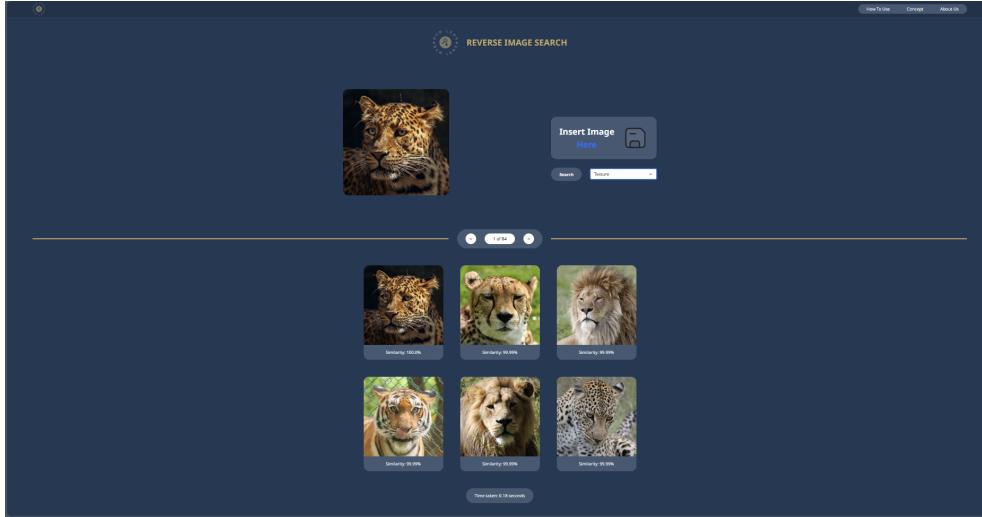
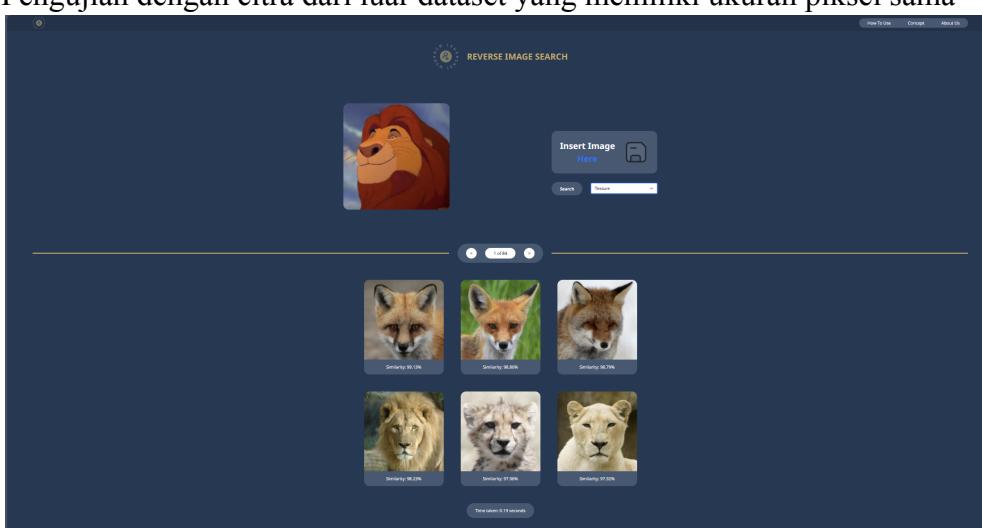
Algoritma Color	
Testcase 1	Pengujian dengan salah satu citra yang ada dalam dataset  A screenshot of a web application titled "REVERSE IMAGE SEARCH". At the top, there is a placeholder box labeled "Insert Image Here" with a file icon. Below it is a search bar with the placeholder "Search" and a dropdown menu. The main area displays a grid of six images. The first image is a leopard's face. To its right, three smaller images are shown with their similarity scores: "Similarity: 98.0%" (lion), "Similarity: 95.5%" (lion), and "Similarity: 94.0%" (leopard). Below these are two more images: a cheetah's face ("Similarity: 94.2%") and a tiger's face ("Similarity: 93.8%"). A third tiger image is also present with a similarity score of 94.0%. At the bottom of the grid, a note says "Time taken: 0.12 seconds".
Testcase 2	Pengujian dengan citra dari luar dataset yang memiliki ukuran piksel sama

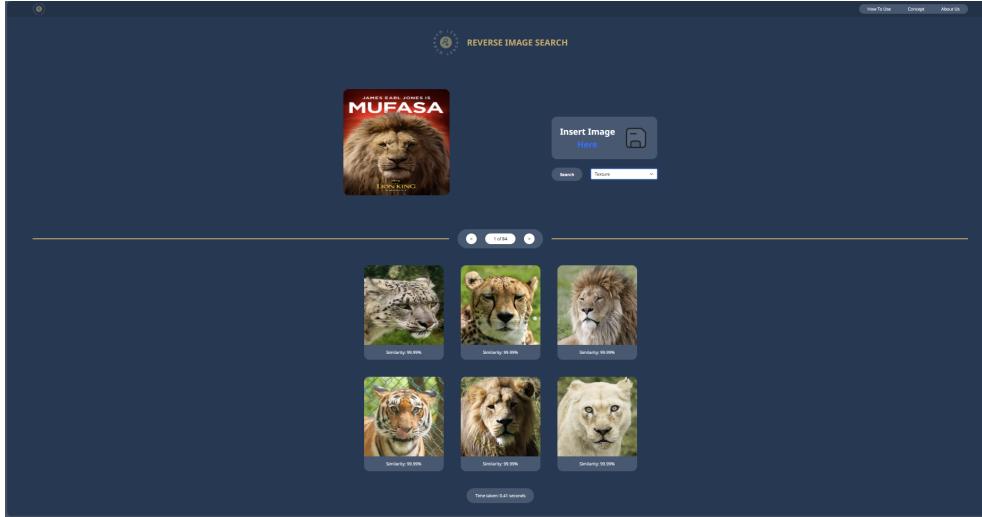
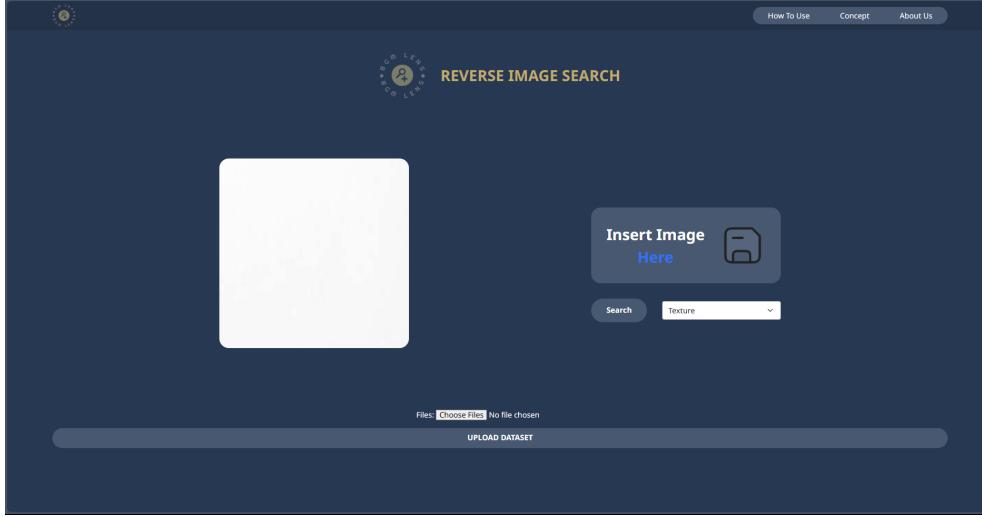
	 <p>The screenshot shows a user interface for a reverse image search tool. At the top, there is a placeholder for an image with the text "Insert Image Here". Below it is a search bar with the word "Color". A central image of a lion's head is displayed. Below the main image, there are two rows of smaller images. The first row contains three images: a tiger, another tiger, and a fox. The second row also contains three images: a tiger, a fox, and another tiger. Each image has a similarity percentage below it: 92.39%, 92.04%, 92.12% for the first row; and 78.15%, 78.08%, 78.08% for the second row. At the bottom of the interface, a message says "Time taken: 0.12 seconds".</p>
Testcase 3	Pengujian dengan citra dari luar dataset yang memiliki ukuran piksel berbeda
	 <p>The screenshot shows a user interface for a reverse image search tool. At the top, there is a placeholder for an image with the text "Insert Image Here". Below it is a search bar with the word "Color". A central image of a lion from a movie poster is displayed. Below the main image, there are two rows of smaller images. The first row contains three images: a tiger, another tiger, and a fox. The second row also contains three images: a fox, a lion, and another fox. Each image has a similarity percentage below it: 91.47%, 91.52%, 91.58% for the first row; and 91.13%, 91.37%, 91.43% for the second row. At the bottom of the interface, a message says "Time taken: 0.28 seconds".</p>
Testcase 4	Pengujian dengan citra berwarna putih



### Algoritma Tekstur

Testcase 1	Pengujian dengan salah satu citra yang ada dalam dataset
------------	--

	
Testcase 2	Pengujian dengan citra dari luar dataset yang memiliki ukuran piksel sama
	

	
Testcase 4	<p>Pengujian dengan citra berwarna putih</p> 

### Analisis Dari Desain Solusi Algoritma Pencarian

Analisis dari desain solusi algoritma pencarian yang diimplementasikan pada setiap pengujian yang dilakukan. Misalnya, apakah pembangunan CBIR berdasarkan fitur warna lebih baik dari tekstur pada kasus-kasus tertentu beserta analisis mengenai mengapa hal itu bisa terjadi jika benar.

Website yang kami bangun sudah sesuai dengan UI yang didesain. Namun, website kami belum membuat website menjadi responsif, untuk sekarang hanya bisa untuk tampilan desktop. Website yang kami buat juga kurang interaktif dan terlihat lebih statis. Untuk pengunggahan dataset belum bisa mengunggah dalam bentuk folder, untuk sekarang hanya bisa untuk mengunggah banyak file sekaligus.

Pada fitur CBIR dengan parameter warna, terdapat kecacatan dimana dua gambar hanya mirip dari jumlah warnanya, tetapi tidak dengan letak warnanya. Hal ini disebabkan oleh perbandingan data histogram yang tidak menggunakan letak-letak blok warna tertentu untuk menentukan kemiripannya sehingga letak dari suatu warna diabaikan. Ini mengakibatkan dua gambar dinyatakan mirip walaupun letak warna yang sesuai tidak mirip antara dua gambar.

Pada fitur CBIR dengan parameter tekstur, terdapat ketidaksesuaian di mana persentase gambar dalam dataset menunjukkan rata-rata di atas 99%. Fenomena ini dipicu oleh dominasi pembobotan komponen kontras dibandingkan dengan komponen lainnya, yang mengakibatkan representasi visual yang tidak seimbang dalam proses pencarian citra. Untuk mengatasi hal ini, dilakukan normalisasi Gaussian guna menyeimbangkan pembobotan. Meskipun demikian, rata-rata persentase kemiripan dengan dataset justru mengalami peningkatan yang marginal dari kondisi sebelumnya. Hal ini dimungkinkan perlunya kalkulasi untuk menentukan pembobotan yang ideal.

Berdasarkan hasil pengujian yang dilakukan dari metode CBIR yang kami buat, pembangunan CBIR berdasarkan fitur warna terlihat lebih akurat dibandingkan dengan fitur tekstur. Perbandingan data histogram persebaran warna dapat memberikan gambaran yang lebih akurat dibandingkan dengan parameter komponen ekstraksi vektor tekstur. Untuk meningkatkan performa fitur tekstur, diperlukan penambahan parameter pada komponen ekstraksi vektor tekstur sehingga hasil yang didapat lebih akurat.

## **BAB V**

## **PENUTUP**

### **Kesimpulan**

Dari kuliah IF2123 Aljabar Linier dan Geometri, kami mengambil konsep-konsep materi yang kami dapat dan mengimplementasikannya dalam aplikasi pengolahan citra. Sebuah citra dapat direpresentasikan sebagai suatu matriks yang kemudian dapat diolah sesuai dengan kaidah operasi matriks.

Melalui tugas besar ini, kami dapat memahami proses pengolahan citra dengan membangun aplikasi website menggunakan bahasa python dengan framework Django. Kami belajar banyak hal, baik yang berhubungan dengan materi Aljabar Linier dan Geometri, pengolahan citra, atau pembuatan website.

### **Saran**

- Lebih mendalami kembali mengenai pemrosesan citra yang efisien dan lebih akurat.
- Melakukan riset lebih dalam mengenai algoritma dan framework yang digunakan

### **Komentar atau Tanggapan**

Tugas besar ini sangat membantu kami untuk mengeksplorasi pembuatan *website* menggunakan Python. Namun, tugas besar ini dirasa kurang mengaplikasikan teori Aljabar Linear dan Geometri yang begitu banyak jika dibandingkan dengan tugas besar 1 dan tugas besar tahun sebelumnya yaitu Face Recognition yang utamanya memanfaatkan teori Eigen.

### **Refleksi Terhadap Tugas Besar**

Kami diberikan kesempatan untuk mengeksplorasi aspek pengolahan citra berdasarkan materi yang dipelajari dalam kuliah Aljabar Linear dan Geometri IF2123, serta untuk mengembangkan keterampilan dalam membuat website dengan menggunakan framework Django. Melalui tugas besar ini, kami mendapatkan pemahaman yang lebih mendalam tentang proses pengolahan citra dan langkah-langkah pembuatan website. Pengalaman ini membuka wawasan kami, mulai dari penerapan algoritma yang diajarkan dalam kuliah hingga kerjasama tim dalam membangun sebuah website. Keterampilan berkolaborasi kami semakin terasah dan terarah, terutama dalam penggunaan GitHub untuk manajemen proyek yang efektif.

### **Ruang Perbaikan atau Pengembangan**

Pada pembuatan *website* ini, kami belum merancangnya dengan maksimal. Hal ini dapat dilihat dari UI yang kurang interaktif sehingga tidak ramah dari sudut pandang user. Pada masa yang akan datang, selain perancangan program utama, perancangan *website* juga perlu diperhatikan agar UI lebih interaktif.

Algoritma CBIR yang kami rancang tentu belum mencapai kesempurnaan dan masih terdapat ruang untuk pengembangan lebih lanjut. Pendekatan CBIR dengan fitur warna dapat ditingkatkan dengan menerapkan pembobotan pada blok piksel. Dengan membagi citra menjadi

blok-blok dan memberikan bobot dari perhitungan nilai HSV, akurasi dari metode CBIR dengan fitur warna akan menjadi lebih akurat. Pendekatan CBIR dengan algoritma tekstur, dapat dikembangkan dengan mengintegrasikan komponen ekstraksi tambahan pada vektor, yang dapat memberikan efek signifikan dalam membandingkan dua gambar. Selain itu, dapat ditambahkan juga pengaturan pembobotan pada setiap komponen vektor ekstraksi tekstur , yang disesuaikan dengan kondisi khusus dari masing-masing gambar.

## **DAFTAR PUSTAKA**

“RGB to HSV color conversion”

<https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm>

Proses konversi warna RGB ke HSV.

“Content-based image retrieval using color and texture fused features”

<https://www.sciencedirect.com/science/article/pii/S0895717710005352>

Penjelasan lebih jauh mengenai CBIR tekstur dan warna.

## **LAMPIRAN**

Pranala Github

<https://github.com/shafiqIrv/Algeo02-22003.git>

Pranala Youtube

[https://youtu.be/\\_32MEZO2XNQ](https://youtu.be/_32MEZO2XNQ)