

**Laporan Tugas Besar 2**  
**IF2123 Aljabar Linier dan Geometri**  
**Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar**



Disusun oleh:  
Francesco Michael Kusuma (13522038)  
Marvel Pangondian (13522075)  
Julian Chandra Sutadi (13522080)

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2023**

# **Daftar Isi**

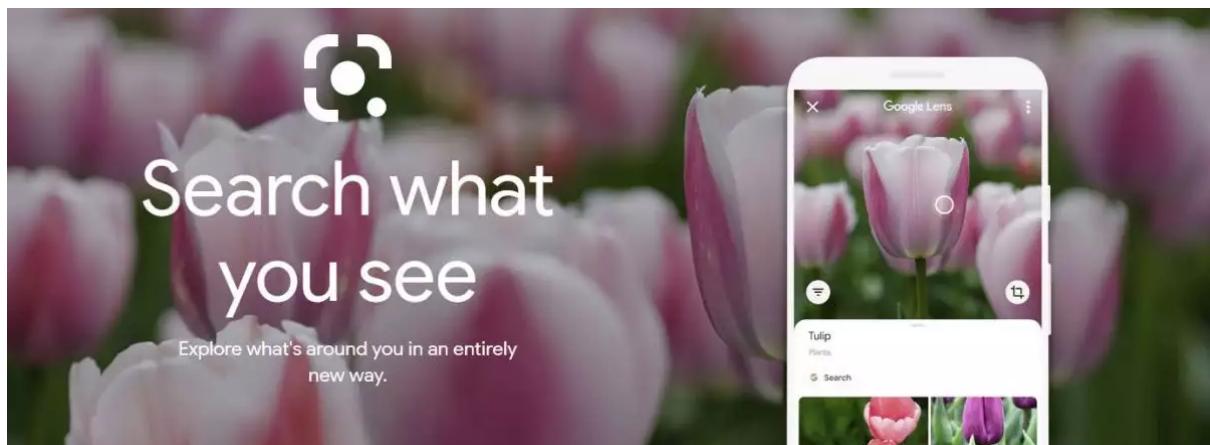
<b>Program Studi Teknik Informatika</b>	
<b>Sekolah Teknik Elektro dan Informatika</b>	
<b>Institut Teknologi Bandung</b>	
<b>2023.....</b>	<b>1</b>
<b>Daftar Isi.....</b>	<b>2</b>
<b>Bab I</b>	
<b>Deskripsi Masalah.....</b>	<b>4</b>
<b>Bab II</b>	
<b>Landasan Teori.....</b>	<b>5</b>
2.1 Content-Based Information Retrieval.....	5
2.2 Color Based CBIR.....	5
2.2 Texture-Based CBIR.....	6
2.3 Cosine Similarity.....	7
2.4 Pengembangan Web.....	7
<b>Bab III</b>	
<b>Analisis Pemecahan Masalah.....</b>	<b>9</b>
3.1 Langkah-langkah pemecahan masalah.....	9
3.1.1 Membaca Spesifikasi dan Referensi.....	9
3.1.2 Identifikasi Masalah.....	9
3.1.3 Tujuan Penelitian.....	9
3.1.4 Identifikasi Tools yang akan digunakan.....	9
3.1.5 Desain Program.....	9
3.1.6 Identifikasi Masalah dalam Implementasi Program.....	10
3.1.5 Optimalisasi.....	10
3.1.6 Analisis dan Evaluasi.....	10
3.1.7 Kesimpulan dan saran.....	10
3.2 Proses pemetaan masalah menjadi elemen-elemen pada aljabar geometri.....	11
3.3 Contoh ilustrasi kasus.....	12
<b>Bab IV</b>	
<b>Implementasi dan Uji Coba.....</b>	<b>13</b>
4.1 Implementasi Program Utama serta Struktur Program.....	13
4.1.1 CBIR dengan parameter color.....	13
4.1.2 Alur kerja program CBIR dengan parameter color.....	13
4.1.3 CBIR dengan parameter texture.....	14
4.1.4 Alur kerja program CBIR dengan parameter texture.....	15
4.2 Tata Cara Penggunaan Program.....	15
4.3 Hasil Pengujian.....	17
4.4 Analisis Desain Solusi Algoritma.....	28
<b>Bab V.....</b>	<b>29</b>
5.1 Kesimpulan.....	29
5.2 Saran.....	30
5.3 Komentar atau Tanggapan.....	30
5.4 Refeksi.....	30

5.5. Ruang Perbaikan atau Pengembangan.....	30
<b>Daftar Referensi.....</b>	<b>30</b>
<b>Link Repository.....</b>	<b>31</b>

# Bab I

## Deskripsi Masalah

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 populer adalah Google Lens.



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

Oleh karena itu, di dalam proyek ini, akan diimplementasikan sistem temu balik gambar 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.

## Bab II

### Landasan Teori

#### 2.1 Content-Based Information Retrieval

*Content-Based Information Retrieval* adalah sistem temu balik informasi dengan membandingkan fitur-fitur dasar dari suatu gambar dengan gambar-gambar lain yang terdapat di dalam basis data. CBIR akan mengekstrak fitur-fitur dari representasi RGB (*Red-Green-Blue*) citra kemudian mengubahnya menjadi fitur yang diinginkan. Secara umum, terdapat dua pendekatan CBIR, yaitu CBIR berbasis warna (*color based CBIR*) dan CBIR berbasis tekstur (*texture based CBIR*).

#### 2.2 Color Based CBIR

HSV (*hue, saturation, and value*) merupakan model warna yang digunakan dalam *computer graphics* serta *image processing*. HSV terdiri atas 3 komponen yakni :

1. Hue : mewakili jenis warna, seperti merah, hijau, atau biru. Hue dijelaskan sebagai sudut antara 0 dan 360 derajat pada roda warna, di mana 0 dan 360 keduanya sesuai dengan merah, 120 untuk hijau, dan 240 untuk biru.
2. Saturation : mewakili intensitas sebuah warna. Nilai 0% menyatakan warna merupakan abu-abu, sedangkan 100% menyatakan warna yang intens (*full color*)
3. Value : Menyatakan kecerahan warna. 0% menyatakan warna yang total gelap, sedangkan 100% menyatakan warna yang total terang

HSV digunakan sebagai pengganti RGB sebab HSV dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum untuk digunakan. Beberapa library bahasa program ketika membaca image, masih dalam bentuk RGB, sehingga untuk mengubah RGB menjadi HSV adalah sebagai berikut :

1. Nilai RGB perlu dinormalisasikan terlebih dahulu, yakni dengan membagi masing-masing komponen dengan 255 sehingga menjadi nilai dengan *range* [0,1]

$$R' = R/255 \quad G' = G/255 \quad B' = B/255$$

2. Setelah dinormalisasikan, perlu ditemukan Cmax (nilai terbesar antara hasil normalisasi RGB), Cmin (nilai terkecil antara hasil normalisasi RGB), serta delta (Cmax - Cmin)

3. Hasil HSV dapat diperoleh secara berikut :

a. Hue

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

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

$$V = C_{max}$$

b. Saturation

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

c. Value

$$V = C_{max}$$

## 2.2 Texture-Based CBIR

Dalam CBIR dengan parameter tekstur, terdapat beberapa fitur penting sebagai berikut:

1. Contrast: Ukuran penyebaran elemen - elemen matriks citra. Contrast dari suatu citra dapat dihitung dengan rumus

$$contrast = \sum_{i,j=0}^{N-1} P_{i,j} (i - j)$$

2. Homogeneity: Konsentrasi pasangan intensitas pada matriks co-occurrence. Homogeneity dari suatu citra dapat dihitung dengan rumus

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

3. Entropy : Mengukur keteracakkan distribusi intensitas.

$$entropy = \sum_{i,j=0}^{N-1} -P_{i,j} * \ln P_{i,j}$$

4. Energy : Mengukur pasangan intensitas

$$energy = \sqrt{\sum_{i,j=0}^{N-1} P_{i,j}^2}$$

5. Correlation : mengukur tingkat ketergantungan ketergantungan linier derajat keabuan dari piksel yang saling bertetangga

$$correlation = \sum_{i,j=0}^{N-1} P_{i,j} \left( \frac{(i-\mu)(j-\mu)}{\sigma^2} \right)$$

## 2.3 Cosine Similarity

Kesamaan kosinus mengukur kesamaan antara dua vektor ruang. Hal ini diukur dengan kosinus sudut antara dua vektor dan menentukan apakah dua vektor menunjuk ke arah yang kira-kira sama. Metode ini sering digunakan untuk mengukur kesamaan dokumen dalam analisis teks.

Di dalam sistem temu balik gambar dengan pendekatan CBIR, Cosine Similarity digunakan untuk menentukan kesamaan antar gambar. Dalam CBIR dengan parameter color, Cosine Similarity digunakan untuk menentukan kesamaan antara dua vektor yang terdiri atas komponen HSV yang sudah dalam bentuk histogram, sedangkan dalam CBIR dengan parameter tekstur digunakan untuk menghitung kesamaan dua gambar berdasarkan contrast, homogeneity, dan entropy. Berikut adalah rumus 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*.

## 2.4 Pengembangan Web

Situs web merupakan kumpulan halaman web, yaitu halaman yang menampilkan informasi, baik berupa teks, gambar, maupun media lainnya. Informasi dapat ditampilkan pada halaman web karena komputer yang menampilkan informasi (*client*) dapat berkomunikasi dengan komputer yang menyediakan informasi (*server*) melalui internet, baik dengan kabel maupun nirkabel.

Suatu web dapat menampilkan informasi karena adanya siklus *request* dan *response* antara *client* dan *server*. Data-data di dalam siklus tersebut dapat berupa berkas HTML, CSS, Javascript, dan aset-aset lain. Yang dimaksud dengan *server* adalah perangkat lunak dan perangkat keras. Ketika *client* sudah memberikan respon, maka informasi dapat ditampilkan di pihak *client*. HTML akan memberi struktur pada web, CSS memberi gaya pada web, sedangkan Javascript memberikan interaktivitas antarkomponen web.

Secara umum, pengembangan web dibagi menjadi dua, yaitu *frontend* dan *backend*. Pengembangan *frontend* berarti menyiapkan bagaimana informasi ditampilkan kepada *client*. Pengembang *frontend* akan banyak menggunakan HTM, CSS, dan Javascript untuk menampilkan informasi. Pengembangan *backend* berfokus pada pengolahan dan penyediaan informasi dan titik-titik akses kepada pengembang *frontend* yang akan mengambil informasi.

## Bab III

# Analisis Pemecahan Masalah

### 3.1 Langkah-langkah pemecahan masalah

#### 3.1.1 Membaca Spesifikasi dan Referensi

Sebelum memulai memecahkan masalah, terlebih dahulu mengetahui masalah tersebut. Untuk mengetahui masalah yang ingin dipecahkan, perlunya membaca spesifikasi ([Spesifikasi Tugas Besar 2 IF2123 Aljabar Linier dan Geometri](#)), melalui spesifikasi, akan diketahui masalah yang perlu diselesaikan. Masalah yang perlu diselesaikan yakni aplikasi CBIR dengan parameter warna dan gambar. Setelah mengetahui spesifikasi program/masalah, perlu diketahui cara menyelesaikan masalah tersebut yakni dengan membaca referensi. Referensi yang digunakan peneliti terdapat pada daftar referensi laporan

#### 3.1.2 Identifikasi Masalah

Setelah membaca spesifikasi dan referensi/literatur, dilakukan identifikasi masalah. Masalah yang dihadapi penulis adalah implementasikan sistem temu balik gambar dengan memanfaatkan Aljabar Vektor dalam bentuk sebuah *website*.

#### 3.1.3 Tujuan Penelitian

1. Mengimplementasikan materi aljabar vektor dalam sistem temu balik gambar
2. Mempelajari pembangunan web

#### 3.1.4 Identifikasi Tools yang akan digunakan

Setelah mengetahui masalah yang akan diselesaikan, perlunya identifikasi tools-tools yang akan digunakan dalam rangka menyelesaikan masalah tersebut. Untuk implementasi CBIR berdasarkan parameter color, ditemukan bahasa python sebagai bahasa yang paling sesuai berdasarkan tingkat kesulitan masalah dan kemampuan peneliti. Untuk implementasi CBIR berdasarkan parameter tekstur, digunakan bahasa golang dalam rangka mempercepat proses. Untuk implementasi website, ditemukan library React JS dengan framework Vite untuk mempermudah pembuatan web.

#### 3.1.5 Desain Program

Setelah mengetahui tools-tools yang akan digunakan, maka perlunya mendesain program. Desain program bertujuan untuk memecahkan masalah yang ingin diselesaikan. Desain program belum termasuk optimalisasi program, hal tersebut dapat dilakukan setelah dasar program selesai. Dalam desain program, perlu pengetahuan mengenai mengenai macam - macam library yang akan digunakan, data struktur, algoritma, dll.

### **3.1.6 Identifikasi Masalah dalam Implementasi Program**

Sejalan menyelesaikan masalah, ditemukan beberapa masalah yang perlu dihadapi, yakni runtime preprocessing image yang membutuhkan waktu yang lama. Salah satu solusi dalam menyelesaikan masalah tersebut adalah dengan resizing image dengan library Cv2 (cv2.resize()) yang mengubah size image menggunakan metode interpolasi. Ketahui bahwa pengubahan size image akan mempengaruhi akurasi similarity, tetapi sesuai dengan hasil penelitian oleh Miroslav Marinov; Yordan Kalmukov; dan Irena Valova ([Impact of image resolution on the search accuracy and results ordering](#)), hal tersebut hanya sedikit mempengaruhi akurasi dalam CBIR dengan parameter color. Dalam implementasi CBIR dengan parameter tekstur, peneliti tidak mengubah size image, sebab sudah menggunakan bahasa golang sehingga preprocessing image sudah cukup cepat. Masalah lain yang dihadapi peneliti yakni cara mengirimkan data ke front-end, cara yang ditemukan adalah dengan menggunakan API. Dalam membuat API, peneliti menggunakan web framework flask untuk bahasa python dan gin untuk bahasa golang. Masalah lain yang dihadapi peneliti adalah cara mengkoneksi dua bahasa berbeda ke dalam website.

### **3.1.5 Optimalisasi**

Setelah dasar kode sudah benar dan dapat berjalan, maka diperlukan optimalisasi kode. Tujuan dari optimasi kode adalah memperkecil waktu yang dibutuhkan untuk program mengkalkulasi dan process image. Salah satu optimalisasi yang dilakukan peneliti adalah penggunaan multiprocessing.

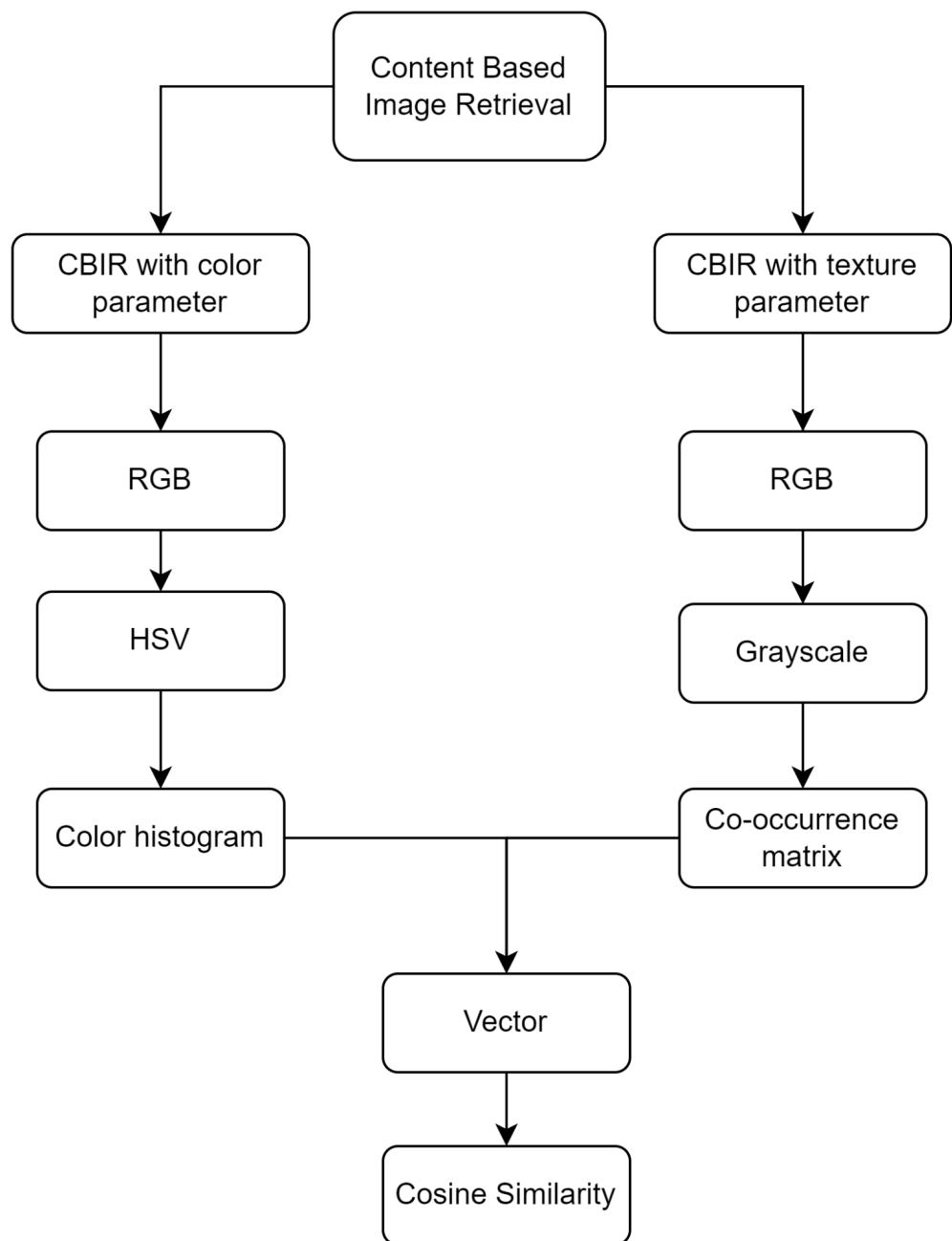
### **3.1.6 Analisis dan Evaluasi**

Setelah mengoptimalkan program, perlu dilakukannya analisis dan evaluasi dari hasil implementasi. Pada tahap ini, perlu dijawab beberapa pertanyaan yakni : Apakah terdapat bagian yang bisa diperbaiki?; apakah terdapat solusi alternatif?; apakah implementasi sudah dapat memecahkan masalah?; apakah implementasi akurat?; dll. Jika terdapat hal - hal yang dapat dikembangkan, maka hal tersebut dilakukan.

### **3.1.7 Kesimpulan dan saran**

Tahap ini merupakan tahap terakhir dari penelitian dan dilakukan kesimpulan serta saran dari seluruh tahap yang telah dilakukan.n. Kesimpulan dan saran dapat dilihat pada bab 5.

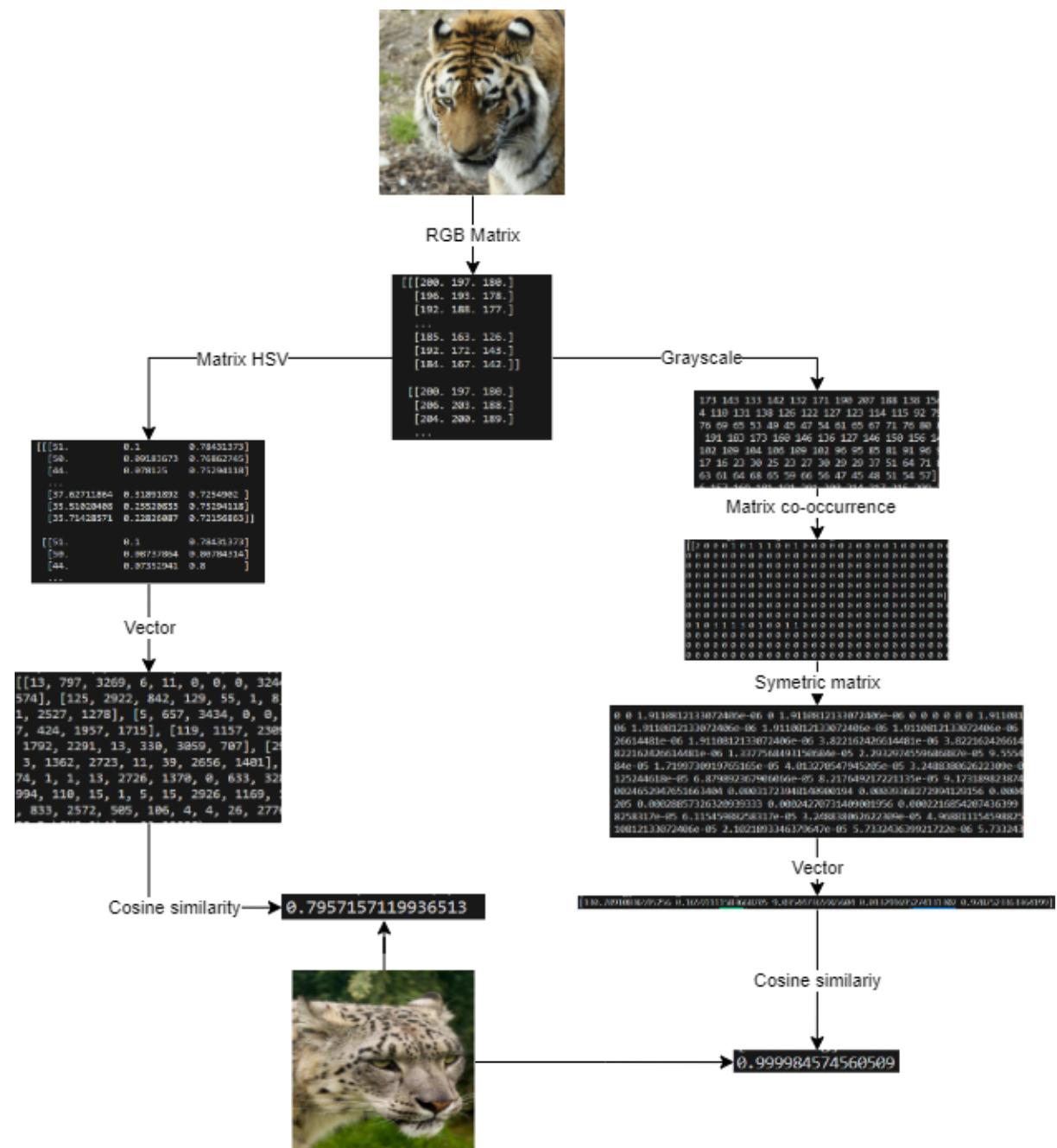
### 3.2 Proses pemetaan masalah menjadi elemen-elemen pada aljabar geometri



Dari hasil pemetaan masalah, peneliti menemukan dua elemen pada aljabar geometri yakni matrix dan vektor. Dari hasil pemetaan masalah tersebut, peneliti perlu menambah wawasan mengenai vektor dan matrix.

### 3.3 Contoh ilustrasi kasus

Pada contoh kasus ini, kita akan mencoba mencari similarity antara dua gambar. Berikut merupakan contoh ilustrasinya :



## Bab IV

### Implementasi dan Uji Coba

#### 4.1 Implementasi Program Utama serta Struktur Program

##### 4.1.1 CBIR dengan parameter color

```
{module hsvsimilarity }

USE NUMPY
USE CV2

KAMUS UMUM

function imgToVector(path : string) -> vector
{mengembalikan hasil vector color histogram sebuah image}

function imgToMatrix(path : string) -> matrix
{menerima path dan mengeluarkan matrix RGB sebuah image}

function matrixRGBtoHSV(RGB : matrix) -> matrix
{menerima matrix RGB dan mengeluarkan hsv matrix sebuah image}

function hsvToVector(HSV : matrix ) -> vector
{menerima matrix HSV dan mengeluarkan vector color histogram}

function cosineSimilarity(vector1 : vector, vector2 : vector) -> float
{menerima dua vector, lalu mengeluarkan cosine similarity antara dua vector tersebut}

function loadVectorData(path : string, dir_list : array of string) ->array
of vectors
{menerima path dan nama-nama images dan mengembalikan array of vectors semua image tersebut}
```

##### 4.1.2 Alur kerja program CBIR dengan parameter color

Awalnya, akan di cek folder imgDataset dan imgUpload pada folder public. Folder imgDataset berisi dataset yang akan digunakan sedangkan imgUpload berisi sebuah foto yang kita akan gunakan untuk compare dengan image-image pada imgDataset. Pertama akan dipanggil fungsi loadVectorData yang berfungsi untuk mengubah seluruh image dalam imgDataset menjadi vector dan mengembalikan array of vectors. Array of vectors tersebut akan di bandingkan satu per satu dengan vector hasil image pada folder imgUpload. Hasil tersebut kemudian di kirim ke front-end melalui API. Berikut galias besar alu program menggunakan pseudocode

Program utama CBIR parameter color  
USE hsvsimilarity

### KAMUS

dir\_list,l : array of string  
data\_vector : array of vectors  
Img\_vector : vector  
cosine\_result : float

### ALGORITMA UTAMA

```
dir_list <- get_all_path_of_images
data_vectr <- loadVectorData(path_to_folder_imgDataset,dir_list)
l <- get_path_of_image_upload {this will result in a list with only one component}
Img_vector <- hsvToVector(matrixRGBtoHSV(imgToMatrix(path_to_folder_imgUpload + l[0])))
i traversal[0..len_of_dir_list]:
    Cosine_result <- cosineSimilarity(data_vector[i],img_vector)
    Store_cosine_result
```

### 4.1.3 CBIR dengan parameter texture

#### USE IMAGE

```
function imgToRGB(string filename) -> array_3d
{menerima path image dan mengembalikan array berdimensi 3d yang merupakan
array RGB}

function sliceRGB(rgba : array_3d ) -> array_3d
{menerima array_3d rgba dan mengembalikan array_3d rgb}

function rgbToGrayscale(rgb : array_3d) -> array_3d
{menerima array_3d RGB dan mengembalikan array_3d grayscale}

function createCoocurenceMatrix(grayscale : array_3d) -> matrix
{menerima sebuah array_3d grayscale dan mengembalikan matrix co-ocurene}

function createSymetricMatrix(coocurence : matrix) -> matrix
{menerima matrix co-ocurrence dan mengembalikan matrix yang sudah simetrik}

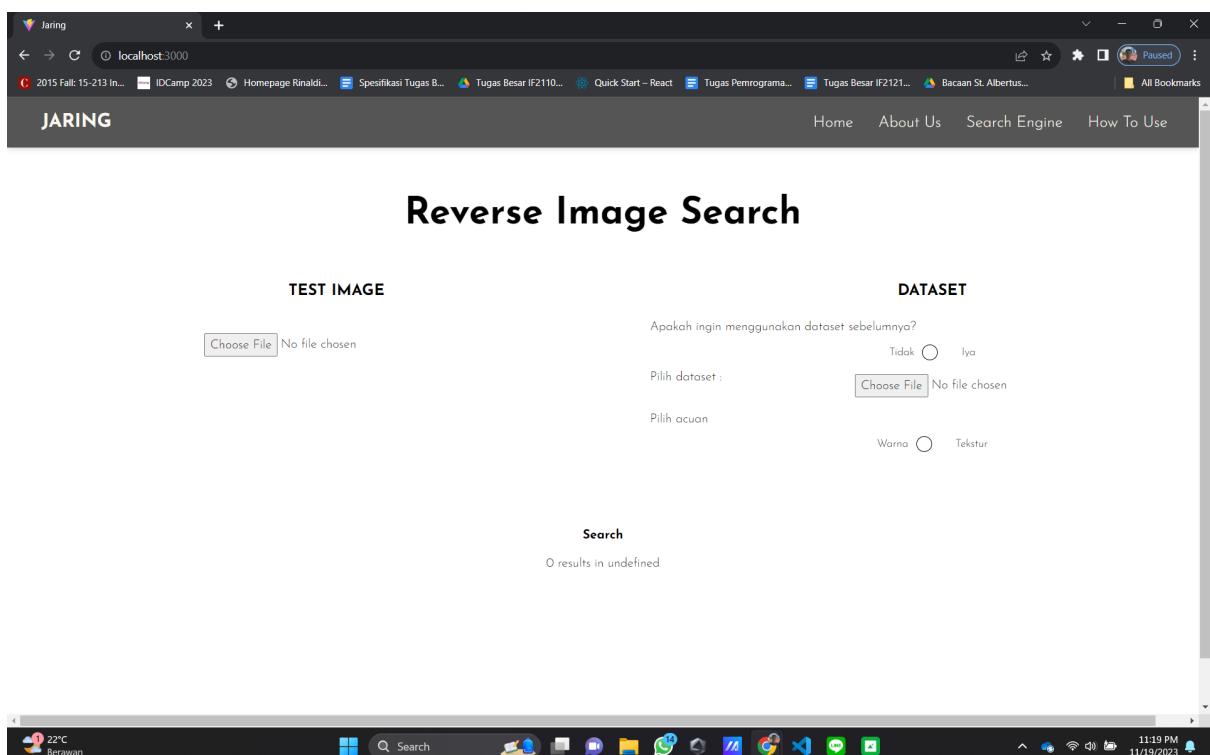
function createFeatureVector(symmetric : matrix) -> array of floats
{menerima sebuah matrix symetric dan mengeluarkan array of floats yang
berisi feature sebuah gambar}
function cosineSimilarity(featureVector1 : array of floats, feaureVector2
: array of floats) -> float
```

```
{menerima dua vector dan mengembalikan hasil cosine similarity kedua  
vector}
```

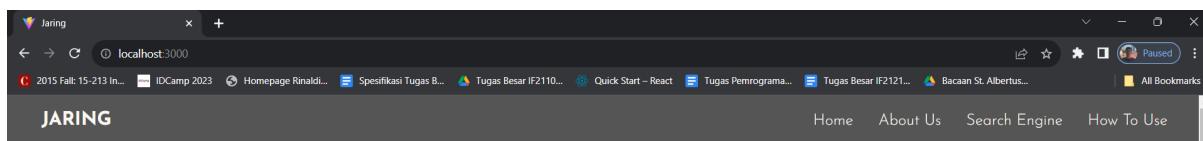
#### 4.1.4 Alur kerja program CBIR dengan parameter texture

Program akan membuka image dari folder imgUpload dan folder imgDataset, kemudian mengubah gambar menjadi representasi RGBA, kemudian mengambil komponen RGB dari representasi RGBA. Kemudian, matriks RGB diubah menjadi matriks greyscale. Setelah itu, dibentuk co-ocurrence matrix berukuran 256\*256 yang menyimpan kemunculan bersebelahan dari tiap pasang elemen greyscale matriks. Matriks tersebut kemudian dibuat simetris dan dinormalisasi. Proses ini dilakukan untuk tiap gambar. Kemudian, dihitung nilai *cosine similarity* antara gambar di dalam imgUpload dengan tiap gambar di dalam imgDataset. Hasil akan dikembalikan ke *frontend* dengan API.

## 4.2 Tata Cara Penggunaan Program



Upload Test Image



## Reverse Image Search

### TEST IMAGE

Choose File 0.jpg



### DATASET

Apakah ingin menggunakan dataset sebelumnya?

Tidak  Iya

Pilih dataset :

Choose File No file chosen

Pilih acuan

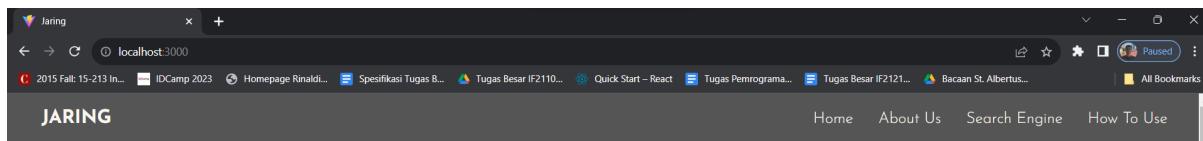
Warna  Tekstur

### Search

0 results in undefined



Upload Dataset (harus .zip)



## Reverse Image Search

### TEST IMAGE

Choose File 0.jpg



### DATASET

Apakah ingin menggunakan dataset sebelumnya?

Tidak  Iya

Pilih dataset :

Choose File 50foto.zip

Pilih acuan

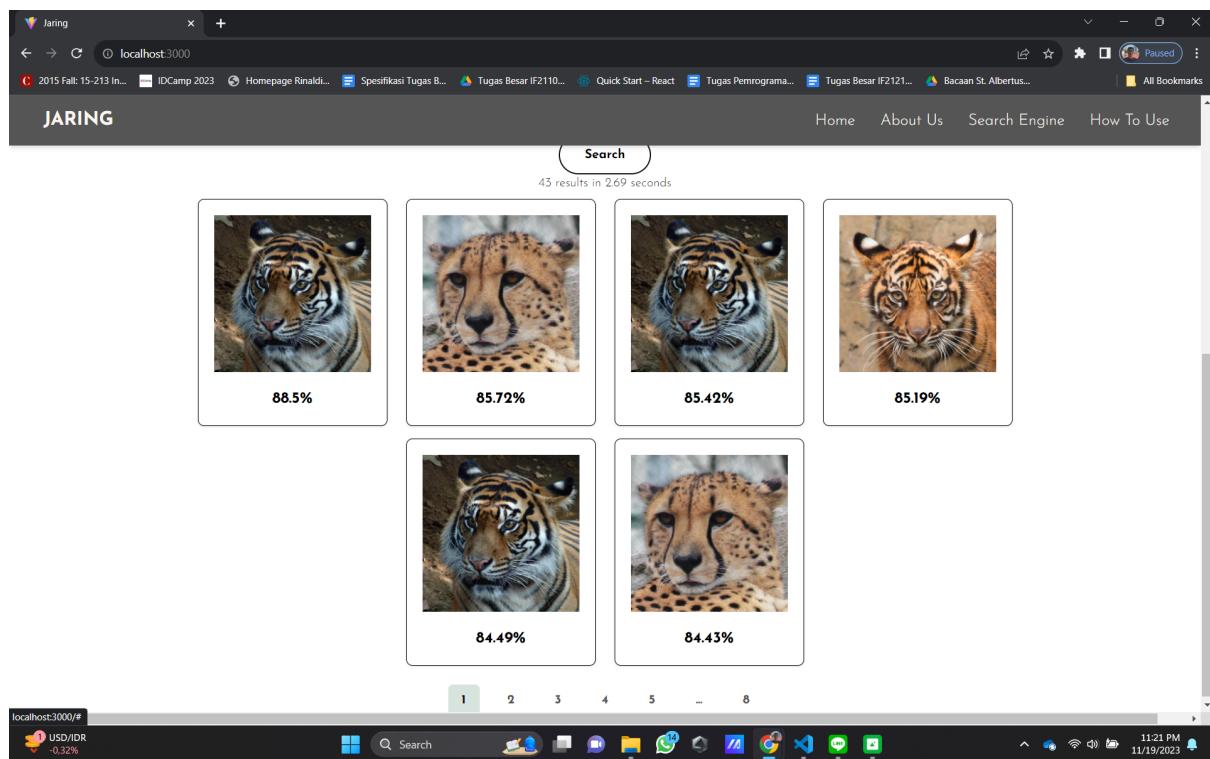
Warna  Tekstur

### Search

Uploaded File: 50foto.zip  
0 results in undefined



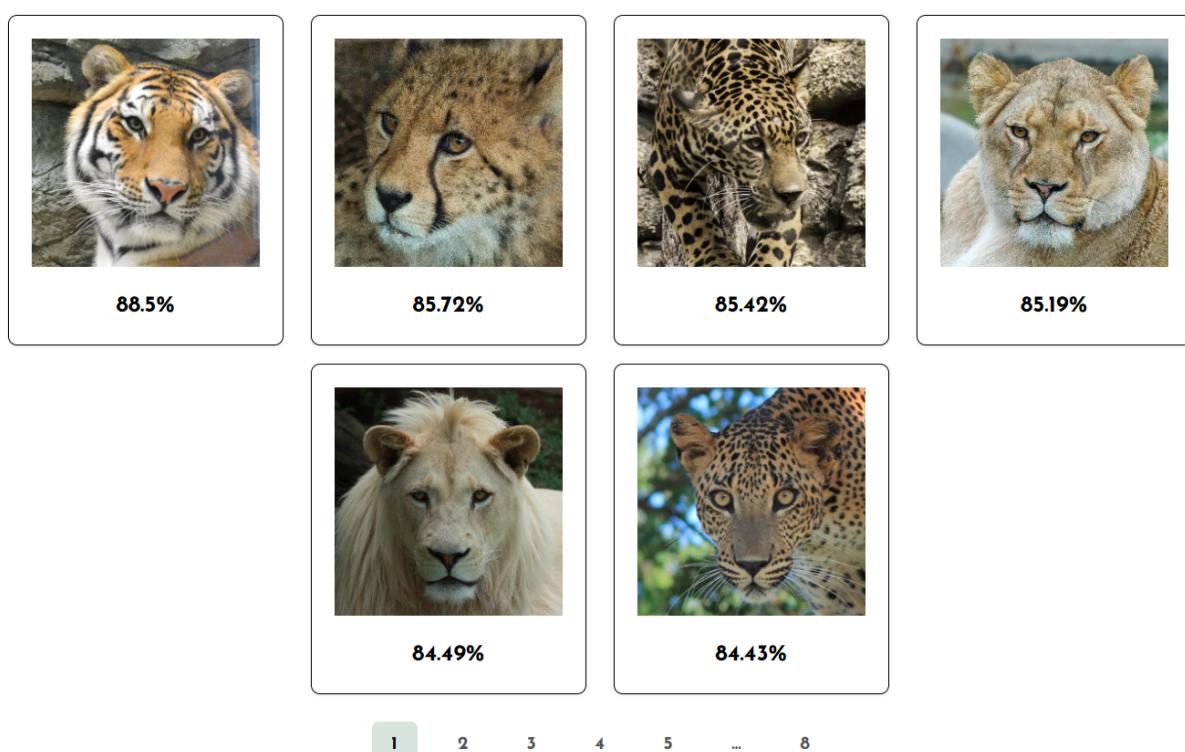
Tekan tombol search



Hasil akan ditampilkan

### 4.3 Hasil Pengujian

Tes 1 : 0.jpg dengan 50foto.zip (color-based)





83.07%



82.59%



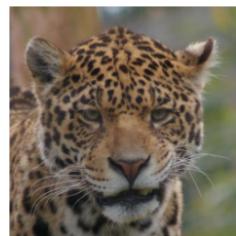
82.36%



82.19%



81.88%

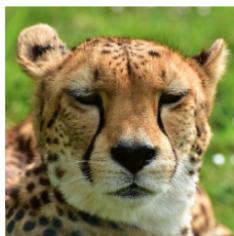


80.71%

1    2    3    4    5    ...    8



80.51%



79.91%



79.76%



79.57%



79.57%

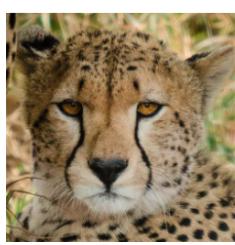


79.56%

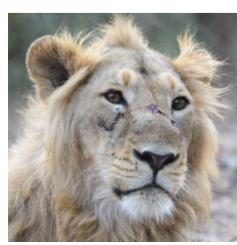
1    2    3    4    5    ...    8



**79.35%**



**78.48%**



**76.43%**



**76.42%**



**75.93%**



**75.81%**

1

2

3

4

5

6

...

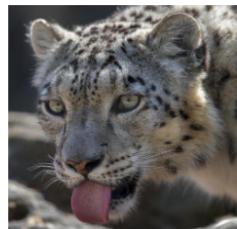
8



**75.31%**



**74.87%**



**74.65%**



**73.96%**



**73.59%**



**73.32%**

1

...

3

4

5

6

7

8



72.75%



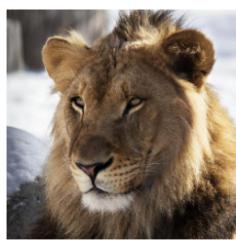
71.97%



71.63%



71.47%



71.09%



70.74%

1 ... 4 5 6 7 8



67.98%



67.16%



66.62%



66.14%

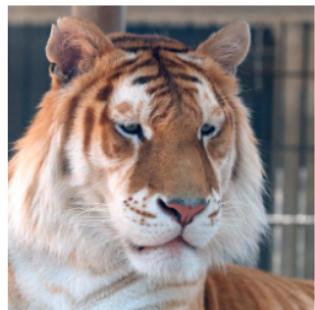


65.86%

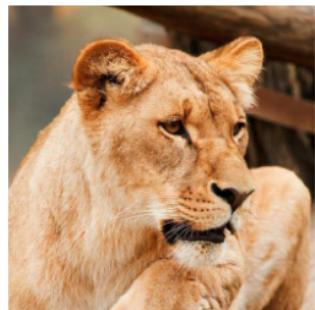


65.85%

1 ... 5 6 7 8



**65.3%**



**63.01%**



**61.94%**

1 ... 6 7 8

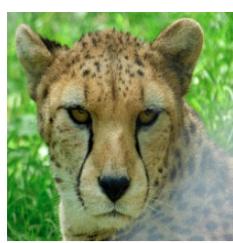
Test 2: julian.jpg dan 50foto.zip (color based)



**73.32%**

1

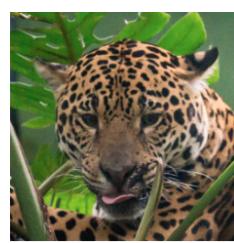
Test 3: 222.jpg dan 50lagi.zip (color-based)



86.94%



84.97%



83.06%



81.16%



80.75%



80.05%

1

2

3

4

5



77.77%



75.71%



75.09%



73.56%



73.28%



72.52%

1

2

3

4

5



70.35%



69.45%



69.24%



68.93%

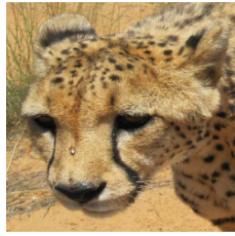


68.11%



68.03%

1 2 3 4 5



67.65%



65.13%



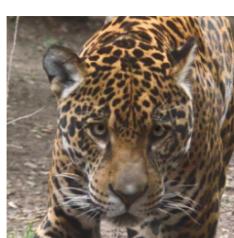
64.69%



63.71%



61.6%



60.72%

1 2 3 4 5



**60.5%**



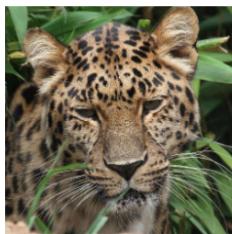
**60.34%**



**60.04%**

1 ... 3 4 5

Test 4: 7.jpg dan 50lagi.zip



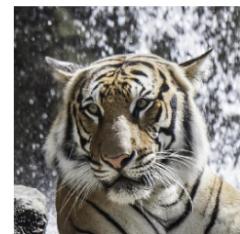
**99.99%**



**99.99%**



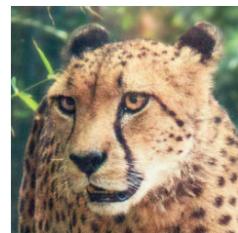
**99.99%**



**99.99%**



**99.99%**



**99.99%**

1 2 3 4 5 ... 9



99.99%



99.99%



99.99%



99.99%



99.98%



99.98%

1 2 3 4 5 ... 9



99.98%



99.98%



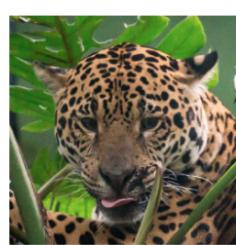
99.98%



99.96%



99.96%



99.94%

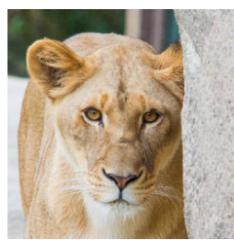
1 2 3 4 5 ... 9



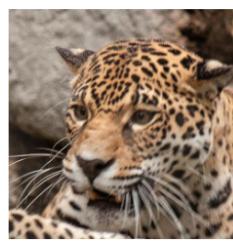
99.94%



99.94%



99.94%



99.93%



99.93%



99.91%

1

2

3

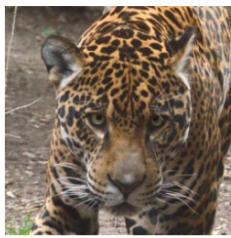
4

5

6

...

9



99.91%



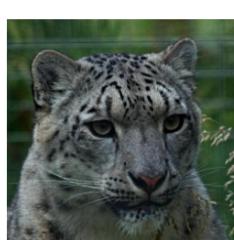
99.91%



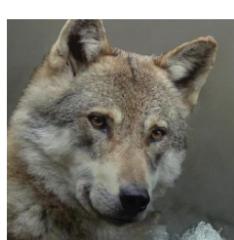
99.91%



99.90%



99.89%



99.89%

1

...

3

4

5

6

7

...

9



99.89%



99.86%



99.84%



99.84%



99.83%



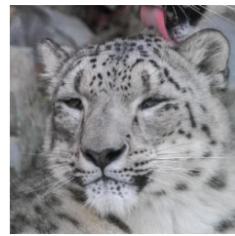
99.82%

1 ... 4 5

6 7 8 9



99.80%



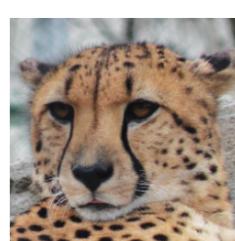
99.79%



99.78%



99.62%



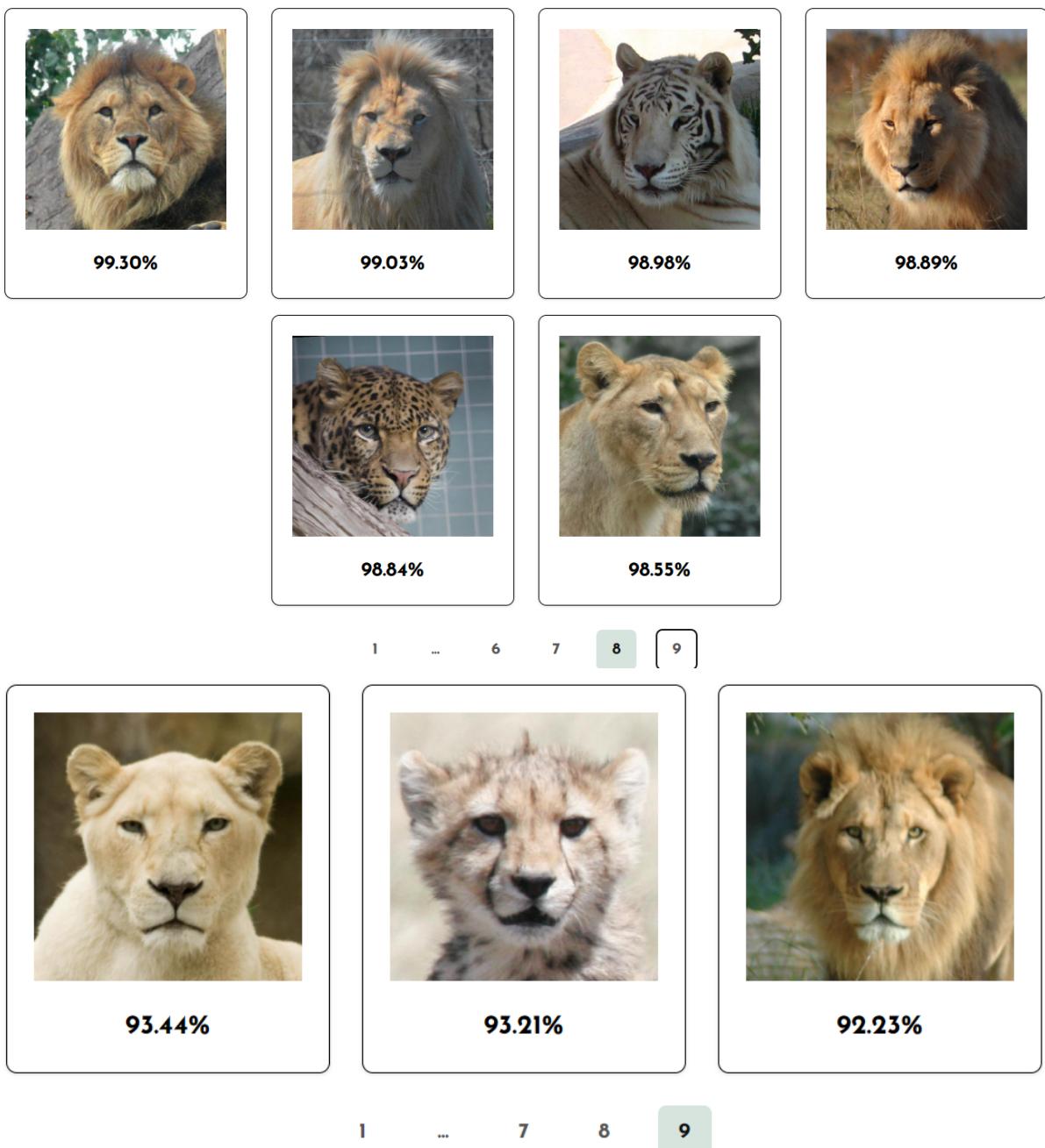
99.56%



99.40%

1 ... 5 6

7 8 9



#### 4.4 Analisis Desain Solusi Algoritma

Berdasarkan hasil pengujian, didapatkan bahwa implementasi CBIR dengan parameter color memberikan hasil *similarity* yang lebih baik/presisi dibandingkan dengan CBIR texture. Dari berbagai percobaan, didapatkan bahwa hasil CBIR texture selalu mendekati 90%, hal tersebut dikarenakan value contrast tiap gambar sangat besar sehingga hasil cosine similarity sangat bergantung pada nilai kontras. Elemen-elemen vektor fitur yang lainnya, walau bisa memiliki perbedaan yang signifikan, menjadi tidak signifikan di dalam perhitungan jika dibandingkan dengan nilai *contrast*.

## Bab V

### 5.1 Kesimpulan

Dalam mencari similarity gambar, terdapat sebuah kesulitan, yakni ketika membandingkan gambar, kedua gambar tersebut memiliki komponen yang dominan sehingga hasil cosine similarity tidak terlalu akurat. Sebagai contoh, ketika membandingkan dua gambar yang memiliki feature yang berbeda menggunakan CBIR texture/color, ada kemungkinan hasil cosine similarity kedua gambar tersebut besar. Untuk CBIR color, hal tersebut dapat terjadi karena terdapat di antara HSV kedua gambar yang sangat sama sehingga value HSV yang lain tidak dapat mempengaruhi akurasi cosine similarity. Dalam CBIR texture, kedua gambar yang berbeda dapat memiliki cosine similarity yang besar dikarenakan dari 5 feature (Contrast, Homogeneity, Entropy, Energy, and Correlation) yang peniliti gunakan, terdapat sebuah feature yang sangat dekat antara kedua gambar sehingga mempengaruhi akurasi cosine similarity.

Implementasi website yang kita buat tidak sempurna, tetapi untuk membandingkan kesamaan dua gambar secara *surface level* sudah cukup baik. Terdapat beberapa faktor yang dapat mempengaruhi hasil kesamaan dua gambar, antara lain posisi object pada gambar yang tidak di tengah (dapat di samping kanan / kiri), kecerahan dan intensitas yang sama antara gambar sehingga mempengaruhi hasil CBIR parameter color, feature contrast yang terdang sama antar gambar juga dapat mempengaruhi hasil similarity antar gambar.

### 5.2 Saran

Saran untuk kelompok kami antara lain:

1. Meningkatkan awareness mengenai waktu deadline
2. Membaca berbagai referensi terlebih dahulu sebelum mengimplementasikan program
3. Memperbaiki komunikasi antar anggota

### 5.3 Komentar atau Tanggapan

Tugas besar ini mengajarkan kita untuk improvise dalam menyelesaikan masalah, terkadang kita harus begadang agar dapat mencapai tujuan, tetapi pengalaman dari perjuangan tersebut merupakan hal yang sangat berguna di masa depan

### 5.4 Refeksi

Refleksi yang kita dapat dari tugas besar ini adalah untuk memperbaiki pengaturan waktu kami lagi. Salah satu alasan mengapa hasil implementasi web kita kurang memuaskan adalah buruknya kami dalam memanage waktu. Hal lain yang kami dapatkan dari tugas besar ini adalah pengetahuan mengenai *Content Based Image Retrieval*, pengembangan web,

aplikasi cosine similarity, aplikasi matrix occurrence, dan team work. Kami harus semakin berani dalam memulai, termasuk dalam mempelajari hal baru.

## 5.5. Ruang Perbaikan atau Pengembangan

Bagian utama yang harus kami perbaiki adalah *time management*. Sebagian besar bug kami temukan di hari terakhir deadline hingga penyelesaian implementasi web tidak maksimal. Bagian lain yang perlu kami perbaiki adalah kemampuan pengembangan web kami. Kami tidak memiliki pengalaman dengan web development sehingga ketika membuat web harus melalui trial and error sebab deadline yang dekat dengan release tugas besar. Hal lain yang perlu kita perbaiki adalah komunikasi kita. Terkadang ketika menyusun web, terdapat miskomunikasi sehingga sebagian besar waktu kami terhabiskan mencoba menyelesaikan masalah-masalah akibat hal tersebut.

## **Daftar Referensi**

M. Marinov, Y. Kalmukov and I. Valova, "Content-Based Image Retrieval: Impact of image resolution on the search accuracy and results ordering," 2021 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2021, pp. 72-75, doi: 10.1109/ICAI52893.2021.9639858.

<https://ieeexplore.ieee.org/abstract/document/9639858>

<https://go.dev/doc/tutorial/web-service-gin>

<https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=Cosine%20similarity%20measures%20the%20similarity,document%20similarity%20in%20text%20analysis.>

<https://www.coursera.org/articles/web-developer>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf>

[https://medium-com.translate.goog/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39?\\_x\\_tr\\_sl=en&\\_x\\_tr\\_tl=id&\\_x\\_tr\\_hl=id&\\_x\\_tr\\_pto=tc&\\_x\\_tr\\_hist=true](https://medium-com.translate.goog/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc&_x_tr_hist=true)

## **Link Repository**

<https://github.com/FrancescoMichael/Algeo02-22038.git>