

LAPORAN TUGAS BESAR II
IF2123 ALJABAR LINIER DAN GEOMETRI
APLIKASI ALJABAR VEKTOR DALAM SISTEM TEMU BALIK
GAMBAR

Disusun untuk memenuhi tugas mata kuliah Aljabar Linear dan Geometri pada Semester 1 (satu) Tahun Akademik 2023/2024.



Oleh Kelompok Allens

Thea Josephine Halim	13522012
Diana Tri Handayani	13522104
Chelvadinda	13522154

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

DAFTAR ISI	1
BAB I	3
DESKRIPSI MASALAH	3
BAB II	5
TEORI SINGKAT	5
2.1 CBIR Dengan Parameter Warna	5
2.2 CBIR Dengan Parameter Tekstur	6
BAB III	9
ANALISIS PEMECAHAN MASALAH	9
3.1 Langkah-langkah Pemecahan Masalah	9
3.2 Pemetaan masalah	9
3.3 Ilustrasi Kasus dan Penyelesaiannya	11
BAB IV	12
IMPLEMENTASI DAN UJI COBA	12
4.1 Implementasi program utama	12
4.1.1 functions.py	12
4.1.2 mtexture.py	14
4.1.3 color.py	17
4.1.4 mcolor.py	24
4.1.5 directory.py	25
4.2 Struktur Program	26
4.3 Tata Cara Penggunaan Program dan Fitur-fitur	28
BAB V	32
KESIMPULAN, SARAN, DAN REFLEKSI	32
5.1 Kesimpulan	32
5.2 Saran	32
5.3 Komentar dan Tanggapan	33
5.4 Refleksi	33
5.5 Ruang Perbaikan dan Pengembangan	33
DAFTAR PUSTAKA	35

BAB I

DESKRIPSI MASALAH

Perkembangan era digital diiringi dengan munculnya semakin banyak foto atau citra digital yang berkualitas. Dari gambar-gambar yang bermacam-macam ini kita bisa melakukan analisis citra, seperti melakukan kompresi gambar, pemrosesan citra berwarna menjadi hitam-putih, atau mengenali objek gambar dengan sistem temu balik gambar. Sistem 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 adalah Google Lens.



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

Dengan menggunakan sistem temu balik gambar kita dapat menganalisis relevansi suatu citra dengan citra yang lain. 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

TEORI SINGKAT

Dalam Tugas Besar II ini terdapat dua bagian poin penyelesaian yang harus dicapai, yaitu *Content-Based Image Retrieval* (CBIR) dengan parameter warna dan *Content-Based Image Retrieval* (CBIR) dengan parameter tekstur. CBIR akan melakukan pembandingan satu gambar dengan gambar-gambar lain dalam dataset. Kemudian akan diambil gambar-gambar yang memiliki kemiripan >60%.

2.1 CBIR Dengan Parameter Warna

CBIR dengan parameter warna memanfaatkan parameter nilai RGB yang didapatkan dari gambar yang ada. Matriks RGB pertama-tama akan diubah menjadi matriks HSV dan dinormalisasi dari range [0,255] menjadi [0,1].

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

2.1 Normalisasi matriks RGB

Perlu juga dicari nilai C_{max} , C_{min} , dan Δ untuk perhitungan HSV

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

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

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

2.2 Nilai Cmax, Cmin, dan Δ

Dari matriks HSV inilah akan dibuat histogram warna dan *image* akan dibagi menjadi beberapa blok agar lebih efektif. Setelah didapatkan vektor berupa histogram warna (elemen-elemen HSV) akan dilakukan perhitungan dengan *cosine similarity*. Nilai *cosine similarity* dalam range [0..1] dengan nilai 1 menunjukkan kemiripan yang tinggi dan sebaliknya.

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

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

$$V = C_{\max}$$

2.3 Mencari nilai HSV

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

2.4 Cosine similarity

2.2 CBIR Dengan Parameter Tekstur

CBIR dengan parameter tekstur juga mengambil nilai RGB dari *image* yang ada. Perbandingan gambar dengan parameter tekstur tidak mempedulikan warna sehingga komponen warna dapat dihilangkan. Akan dibentuk matriks grayscale dari nilai RGB yang didapat.

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

2.5 Pengubahan RGB menjadi grayscale

Setelah didapat matriks grayscale, akan dibuat sebuah framework matriks berukuran 256 x 256, didapat dari range grey tone dari 0 hingga 255. Pada pengeraian kali ini, kami menggunakan nilai arah θ sebesar 0° . Akan dilakukan looping setiap 2 petak elemen arah horizontal, yang akan dijadikan sebagai penambahan jumlah elemen pada indeks baris i dan kolom j pada *co occurrence*, hingga semua elemen matriks ter-*cover*.

GLCM

1	1	2	0	0	1	0	0	0
2	0	0	1	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	0
5	1	0	0	0	0	1	2	0
6	0	0	0	0	0	0	0	1
7	2	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0

2.6 Pembuatan co occurrence

Selanjutnya perlu dilakukan perhitungan lebih lanjut dengan normalisasi *co occurrence*.

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

2.7 Normalisasi matriks co occurrence

Untuk mencari kesamaan antara dua gambar akan diambil beberapa *features* dari *co occurrence*. Pada pengeraian kali ini akan diambil nilai kontras, homogeneity, dan entropy. Dari ketiga komponen ini akan dibentuk sebuah vektor yang kemudian dijadikan parameter perbandingan dengan vektor gambar lain dengan *cosine similarity*.

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

2.8 Contrast

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

2.9 Homogeneity

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

2.10 Entropy

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-langkah Pemecahan Masalah

Website ini mengadopsi aljabar vektor sebagai dasar untuk menggambarkan dan menganalisis data terutama dalam konteks klasifikasi berbasis konten yang dikenal sebagai *Content-Based Image Retrieval* (CBIR). Fokus utama sistem ini adalah pada identifikasi gambar berdasarkan konten visual, seperti warna dan tekstur. Pengguna mengakses platform melalui *website* di browser lokal dan akan mengunggah folder dataset yang berisi gambar-gambar referensi dan gambar pembanding. Setiap gambar dalam dataset diinterpretasikan sebagai vektor dalam ruang fitur, dan analisis menggunakan aljabar vektor dilakukan untuk menganalisis karakteristik visual seperti warna dan tekstur. Pengguna dapat memilih satu gambar dari dataset sebagai referensi untuk memulai proses pencarian gambar serupa. Sistem menggunakan metode klasifikasi berbasis konten yang melibatkan aljabar vektor untuk mencocokkan gambar referensi dengan gambar lain dalam dataset. Hasil pencarian kemudian disajikan kepada pengguna. Dengan pendekatan ini, aljabar vektor menjadi fondasi yang kuat untuk menggambarkan dan menganalisis konten visual, memungkinkan sistem temu balik gambar mengidentifikasi gambar berdasarkan karakteristik warna dan tekstur dengan akurat.

3.2 Pemetaan masalah

Masalah utama yang muncul adalah bagaimana kita melakukan pengubahan sebuah image menjadi sebuah nilai yang bisa dibandingkan dan bagaimana menganalisis sebuah image berdasarkan color dan tekstur. Membandingkan sebuah gambar dengan beberapa image dapat kita lakukan dengan konsep aljabar vektor. Aljabar vektor akan membantu proses menganalisis data dengan menggunakan konsep CBIR (*Content-Based Image Retrieval*). Dilansir dari Science Direct, CBIR artinya menganalisis

informasi fitur-fitur image yang dapat dihubungkan seperti warna, tekstur, dan bentuk dari suatu objek, lalu menjadikannya sebuah vektor fitur image. Fitur-fitur sebuah image dapat dibedakan menjadi 2: *low-level* dan *high-level features*. *Low-level* fitur yang paling sering digunakan adalah warna. CBIR warna mudah untuk dilakukan dan sederhana. Sebab paling terlihat dan tidak terpengaruh oleh rotasi, pergeseran, dan perbedaan skala. Warna yang diambil adalah HSV dan bukan RGB karena lebih konsisten dalam persepsi manusia. Nilai RGB didapatkan dengan memanfaatkan library numpy dan akan dilakukan normalisasi di rentang 0-255 karena shades warna berjumlah 255.

Pembuatan histogram warna membantu memetakan distribusi warna pada gambar, terutama untuk gambar yang sulit untuk di segmentasi. Image akan diambil nilai-nilai RGB nya dan akan diubah menjadi nilai HSV. Kemudian akan dikonversi menjadi sebuah vektor HSV. Dari nilai vektor HSV ini kita dapat membentuk histogram. Sebelumnya, kita perlu membagi image menjadi beberapa bagian, atau yang akan kita sebut sebagai bin. Histogram warna ini akan berisi jumlah pixel warna yang memenuhi sebuah interval tertentu. Dari sebuah histogram akan dilakukan pembandingan dengan histogram image lain dengan cosine similarity.

Fitur tekstur juga kerap digunakan dalam CBIR, untuk mencari pola repetisi dan granularity sebuah image. Berbeda sedikit dengan CBIR warna, di CBIR tekstur akan dilakukan pengubahan RGB menjadi matriks greyscale. Dari greyscale akan dilakukan normalisasi dan dibentuk vektor homogeneity, entropy, dan contrast. Sama seperti color based CBIR, lewat rumus cosine similarity kita dapat melihat tingkat kemiripan sebuah gambar dengan membandingkan vektor nilai gambar pertama dengan vektor gambar kedua. Nilai hasil cosine similarity merentang dari 0 hingga 1, dengan angka 1 yang menunjukkan kemiripan tinggi. Sebab jika $\cos \theta = 1$, kedua vektor berimpit, mengindikasikan image pembanding dengan image utama serupa. Dari hasil cosine similarity inilah akan dilakukan sorting

secara descending untuk menghasilkan mana data di database (image di dataset) yang paling relevan.

Konsep *information retrieval*, mengambil informasi yang dibutuhkan dari sebuah database, atau dalam kasus ini sebuah image, tidak hanya digunakan dalam perbandingan dua image, tetapi juga dalam melakukan pencarian pada kumpulan data atau *search engine*. Information retrieval ini dapat dimodelkan dengan ruang vektor. Sebuah query dapat diubah menjadi sebuah vektor. Dalam pencarian pada *search engine*, akan dibentuk sebuah vektor dengan tingkat kemunculan sebuah kata inputan di setiap dokumen, begitu pula akan terbentuk sebuah vektor query. Dari semua vektor yang ada akan dicari yang paling relevan antara vektor query, atau image inputan dalam hal ini, dengan vektor-vektor lain image di dataset. Di sinilah *cosine similarity*, bagian dari rumus perkalian dot berperan, menghitung kesamaan antara dua vektor.

3.3 Ilustrasi Kasus dan Penyelesaiannya

Proses dari pengguna mengunggah gambar hingga mendapatkan hasil dalam sistem Content-Based Image Retrieval (CBIR) berbasis aljabar vektor dimulai dengan pengguna yang membuka platform website untuk mencari kemiripan gambar, lalu pengguna memilih opsi untuk mengunggah gambar sebagai acuan pencarian. Setelah gambar diunggah, sistem melakukan proses pemrosesan untuk mengubahnya menjadi vektor dengan memanfaatkan konsep aljabar vektor. Aljabar vektor menjadi konsep untuk menggambarkan karakteristik visual gambar, seperti warna dan tekstur. Dalam analisis warna, sistem mengubah nilai warna dari format RGB ke format HSV guna memastikan konsistensi dalam representasi warna. Sementara itu, dalam menganalisis tekstur, sistem mengubah nilai RGB ke matriks *grayscale*.

Kemampuan sistem juga melibatkan pengelolaan dataset yang mencakup kumpulan data gambar sebagai dasar pencarian. Setelah gambar referensi pengguna diproses, langkah berikutnya melibatkan metode

klasifikasi berbasis konten dengan menggunakan aljabar vektor. Sistem memanfaatkan vektor untuk mencocokkan gambar referensi dengan gambar lain dalam dataset. Hasil pencarian kemudian disajikan kepada pengguna berdasarkan persentase kemiripan tertinggi.

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1 Implementasi program utama

4.1.1 functions.py

a. Atribut

Class ini tidak memiliki atribut.

b. Metode

<pre>function cosinesimilarity (tuple: vector1, vector2)-> float {I.S. Vector 1 dan vector 2 terdefinisi dan memiliki panjang yang sama} {F.s. Mengembalikan nilai hasil perhitungan cosine similarity kedua vektor}</pre> <p>KAMUS LOKAL</p> <p>result, total1, total2, i: int</p> <p>ALGORITMA</p> <pre>result <- 0 total1 <- 0 total2 <- 0 i <u>traversal</u> (len(vector1)) result <- result + vector1[i]*vector2[i] total1 <- total1 + vector1[i]*vector2[i] total2 <- total2 + vector1[i]*vector2[i] -> result/(math.sqrt(total1)*math.sqrt(total2))</pre>	<p>Metode menghitung kemiripan suatu gambar dengan cosine similarity</p>
<pre>function totalValue(arr of arr of int: symmetricMatrix)-> int {I.S. symmetricMatrix terdefinisi dan berukuran 256 x</pre>	<p>Metode untuk menghitung nilai total sebuah</p>

<pre> 256} {F.S. Mengembalikan nilai total semua elemen dalam symmetricMatrix} KAMUS LOKAL i, j, elmttotal: int ALGORITMA elmttotal <- 0 j <u>traversal</u> 256 i <u>traversal</u> 256 emltotal <- elmttotal + symmetricMatrix[i][j] -> elmttotal </pre>	<p>symmetricMatrix.</p>
<pre> function vectorcosine(arr of arr of int: cooccurrence)-> tuple {I.S. Matriks cooccurrence terdefinisi} {F.S. Mengirimkan tuple yang terdiri dari tuple homogeneity, entropy, dan contrast} KAMUS LOKAL cooccurrence: arr of arr of int homogeneity, entropy, contrast, i, j: int new_vector: tuple of ints ALGORITMA homogeneity <- 0 entropy <- 0 contrast <- 0 i <u>traversal</u> 0...len(cooccurrence) j <u>traversal</u> 0...len(cooccurrence) </pre>	<p>Metode untuk menghitung tuple homogeneity, entropy, dan contrast.</p>

<pre> homogeneity <- homogeneity + occurrence[i][j]/(1+((i-j)*(i-j)) contrast <- contrast + occurrence[i][j] * (i-j) * (i-j) if (cooccurrence[i][j]>0) then entropy <- entropy + cooccurrence[i][j] * math.log(cooccurrence[i][j],10) new_vector <- (homogeneity, -(entropy), contrast) -> new_vector </pre>	
void sort_dictionary (dictionary_arr, key, ascending)-> arr {I.S. dictionary_arr terdefinisi berisi tuple nama image dan nilai cosine similarity} {F.S. Mengembalikan dictionary_arr yang sudah terurut secara descending} KAMUS LOKAL dictionary_arr: arr ascending: boolean ALGORITMA -> sorted(dictionary_arr, key <- sort_key, reverse <- <u>not</u> (ascending)	Metode untuk mengurutkan array tuple image dan cosine similarity secara descending.

4.1.2 mtexture.py

a. Atribut

Class ini tidak memiliki atribut.

b. Metode

<pre>function konversigray (image) -> arr of arr of int {I.S. sebuah image terdefinisi} {F.S. Mengembalikan matriks greyscale dari image} KAMUS LOKAL greyscale: arr of arr of int i, j, R, G, B: int ALGORITMA greyscale <- [[0...re]...re] i traversal 0...len(image) j traversal 0...len(image) R <- image[i][j][2] G <- image[i][j][1] B <- image[i][j][0] greyscale[i][j] <- round(0.29*R + 0.587*G + 0.114*B) greyscale <- np.resize(greyscale,(256,256)) -> greyscale</pre>	<p>Metode untuk mengambil nilai RGB pada suatu image dan mengkonversinya menjadi matriks greyscale</p>
<pre>function createcooccurence -> arr of arr of int(greyscale) {I.S. matriks greyscale terdefinisi} {F.S. Mengembalikan sebuah matriks cooccurence} KAMUS LOKAL cooccurence: arr of arr of int i, j: int ALGORITMA occurrence <- arr of arr of 256</pre>	<p>Metode untuk menghitung matriks cooccurence dari matriks greyscale.</p>

```

i traversal 256
j traversal 255
    cooccurence [(greyscale[i][j])]
    [(greyscale[i][j+1])] ++
cooccurence <- cooccurence +
np.transpose(cooccurence)
cooccurence <- cooccurence +
fn.totalValue(cooccurence)
-> cooccurence

```

function searchtexture (image)-> tuple of int
{I.S. image terdefinisi}
{F.S. Memproses setiap image dalam folder dataset_dir
dan mengurutkannya sesuai cosine similarity berdasar
CBIR tekstur}

KAMUS LOKAL

ALGORITMA

```

dataset_dir <- get_upload_dir()
start_time <- time.time()
mimage <- cv2.imdecode(image,1)
Mgreyscal <- konversigray(mimage)
Mcooccurence <- createcooccurence(mgreyscale)
Mtoople <- fn.vectorcosine(mcooccurence)
final <- []
if not(exists(dataset-dir) and isdir(dataset_dir)) then
    -> {"success":False, "error":"Path doesn't exist"}
image_files <- listdir(dataset_dir)
filename traversal image_files

```

Metode untuk memproses setiap image dalam sebuah folder image, dan mengurutkannya sesuai kemiripan tekstur.

```

if filename.lower().endswith(('.png', '.jpg', '.jpeg'))
then
    Img_path <- os.path.join(dataset_dir,
                           filename)
    image <- cv2.imread(img_path)
    greyscale <- konversigray(image)
    cooccurrence <- createcooccurrence(greyscale)
    toople <- fn.vectorcosine(cooccurrence)
    similarity <-
        fn.cosinesimilarity(mtoople,toople)
    if(similarity>=0.6) then
        final.append( {"filename":filename,
                      "similarity":similarity})
-> {"files": fn.sort_dictionary(final, "similarity", False),
    "time": time.time() - start_time}

```

4.1.3 color.py

a. Atribut

Class ini tidak memiliki atribut.

b. Metode

void RGBtoHSV(path, re, H, S, V) {I.S. path, re (ukuran resize), matriks H, S, dan V terdefinisi} {F.S. Melakukan pembacaan image dari path, melakukan resize, pembentukan matriks RGB, dan proses konversi menjadi HSV}	Metode untuk melakukan pembacaan image dari path yang terdefinisi dan mengkonversinya menjadi HSV.
KAMUS LOKAL R, G, B, Cmax, Cmin, delta: array of array of int i, j: int	

ALGORITMA

```
image <- image.open(path)
width <- image.size
height <- image.size
new_size <- (width/2, height/2)
resized_image <- image.resize(new_size)
i traversal 0...len(imageC)
    j traversal 0...len(imageC)
        R[i][j] <- imageC[i][j][2] / 255
        G[i][j] <- imageC[i][j][1] / 255
        B[i][j] <- imageC[i][j][0] / 255
    i traversal 0...len(re)
        j traversal 0...len(re)
            Cmax[i][j] <- MAX(R[i][j], G[i][j], B[i][j])
            Cmin[i][j] <- MIN(R[i][j], G[i][j], B[i][j])
            delta[i][j] <- Cmax[i][j] - Cmin[i][j]
        if delta[i][j] = 0 then
            H[i][j] <- 0
        elif Cmax[i][j] = R[i][j]) then
            H[i][j] <- 60 *
                (((G[i][j]-B[i][j])/delta[i][j]) % 6)
        elif Cmax[i][j] = G[i][j]) then
            H[i][j] <- 60 *
                (((B[i][j]-R[i][j])/delta[i][j]) + 2)
        elif Cmax[i][j] = B[i][j]) then
            H[i][j] <- 60 *
                (((R[i][j]-G[i][j])/delta[i][j]) + 4)
```

<pre> if (Cmax[i][j] == 0) then S[i][j] <- 0 else S[i][j] <- delta[i][j]/Cmax[i][j] V[i][j] <- Cmax[i][j] </pre>	
<p>function color(Ha, Sa, Va, path2)-> float</p> <p>{I.S. Matriks Ha, Sa, Va serta path2 (path image lain) terdefinisi}</p> <p>{F.S. Melakukan pembandingan dua image dan mengembalikan nilai similarity kedua gambar}</p> <p>KAMUS LOKAL</p> <p>Ha, Hb, Sb, Sa, Vb, Va: arr of arr of int re: integer</p> <p>ALGORITMA</p> <pre> Re <- 0 Hb <- [[0...re]...re] Sb <- [[0...re]...re] Vb <- [[0...re]...re] i traversal 0...len(re) j traversal 0...len(re) Ha <- np.array(Ha) Sa <- np.array(Sa) Va <- np.array(Va) Hb <- np.array(Hb) Sb <- np.array(Sb) Vb <- np.array(Vb) re1 <- re // 4 re2 <- re // 4 * 2 </pre>	<p>Metode untuk melakukan pembandingan dua gambar dan menghitung nilai cosine similaritynya.</p>

```

re3 <- re // 4 * 3
re4 <- re

H <- fn.cosinesimilarity(np.ravel(Ha[0:re1,
0:re1]), np.ravel(Hb[0:re1, 0:re1]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[0:re1,
re1:re2]), np.ravel(Hb[0:re1, re1:re2]))

H += fn.cosinesimilarity(np.ravel(Ha[0:re1,
re2:re3]), np.ravel(Hb[0:re1, re2:re3]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[0:re1,
re3:re4]), np.ravel(Hb[0:re1, re3:re4]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re1:re2,
0:re1]), np.ravel(Hb[re1:re2, 0:re1]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re1:re2,
re1:re2]), np.ravel(Hb[re1:re2, re1:re2]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re1:re2,
re2:re3]), np.ravel(Hb[re1:re2, re2:re3]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re1:re2,
re3:re4]), np.ravel(Hb[re1:re2, re3:re4]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re2:re3,
0:re1]), np.ravel(Hb[re2:re3, 0:re1]))

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re2:re3,
re1:re2]), np.ravel(Hb[re2:re3, re1:re2]))

```

```

H <- H +
fn.cosinesimilarity(np.ravel(Ha[re2:re3,
re2:re3]), np.ravel(Hb[re2:re3, re2:re3]))
H <- H +
fn.cosinesimilarity(np.ravel(Ha[re2:re3,
re3:re4]), np.ravel(Hb[re2:re3, re3:re4]))
H <- H +
fn.cosinesimilarity(np.ravel(Ha[re3:re4,
0:re1]), np.ravel(Hb[re3:re4, 0:re1]))
H <- H +
fn.cosinesimilarity(np.ravel(Ha[re3:re4,
re1:re2]), np.ravel(Hb[re3:re4, re1:re2]))
H <- H +
fn.cosinesimilarity(np.ravel(Ha[re3:re4,
re2:re3]), np.ravel(Hb[re3:re4, re2:re3]))
H <- H +
fn.cosinesimilarity(np.ravel(Ha[re3:re4,
re3:re4]), np.ravel(Hb[re3:re4, re3:re4]))
meanH <- H / 16
S <- fn.cosinesimilarity(np.ravel(Sa[0:re1,
0:re1]), np.ravel(Sb[0:re1, 0:re1]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[0:re1,
re1:re2]), np.ravel(Sb[0:re1, re1:re2]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[0:re1,
re2:re3]), np.ravel(Sb[0:re1, re2:re3]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[0:re1,
re3:re4]), np.ravel(Sb[0:re1, re3:re4]))

```

```

S <- S +
fn.cosinesimilarity(np.ravel(Sa[re1:re2,
0:re1]), np.ravel(Sb[re1:re2, 0:re1]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re1:re2,
re1:re2]),np.ravel(Sb[re1:re2, re1:re2]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re1:re2,
re2:re3]), np.ravel(Sb[re1:re2, re2:re3]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re1:re2,
re3:re4]), np.ravel(Sb[re1:re2, re3:re4]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re2:re3,
0:re1]), np.ravel(Sb[re2:re3, 0:re1]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re2:re3,
re1:re2]), np.ravel(Sb[re2:re3, re1:re2]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re2:re3,
re2:re3]), np.ravel(Sb[re2:re3, re2:re3]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re2:re3,
re3:re4]), np.ravel(Sb[re2:re3, re3:re4]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re3:re4,
0:re1]), np.ravel(Sb[re3:re4, 0:re1]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re3:re4,
re1:re2]), np.ravel(Sb[re3:re4, re1:re2]))

```

```

S <- S +
fn.cosinesimilarity(np.ravel(Sa[re3:re4,
re2:re3]), np.ravel(Sb[re3:re4, re2:re3]))
S <- S +
fn.cosinesimilarity(np.ravel(Sa[re3:re4,
re3:re4]), np.ravel(Sb[re3:re4, re3:re4]))
meanS <- S / 16
V <- fn.cosinesimilarity(np.ravel(Va[0:re1,
0:re1]), np.ravel(Vb[0:re1, 0:re1]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[0:re1,
re1:re2]), np.ravel(Vb[0:re1, re1:re2]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[0:re1,
re2:re3]), np.ravel(Vb[0:re1, re2:re3]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[0:re1,
re3:re4]), np.ravel(Vb[0:re1, re3:re4]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re1:re2,
0:re1]), np.ravel(Vb[re1:re2, 0:re1]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re1:re2,
re1:re2]), np.ravel(Vb[re1:re2, re1:re2]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re1:re2,
re2:re3]), np.ravel(Vb[re1:re2, re2:re3]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re1:re2,
re3:re4]), np.ravel(Vb[re1:re2, re3:re4]))

```

```

V <- V +
fn.cosinesimilarity(np.ravel(Va[re2:re3,
0:re1]), np.ravel(Vb[re2:re3, 0:re1]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re2:re3,
re1:re2]), np.ravel(Vb[re2:re3, re1:re2]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re2:re3,
re2:re3]), np.ravel(Vb[re2:re3, re2:re3]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re2:re3,
re3:re4]), np.ravel(Vb[re2:re3, re3:re4]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re3:re4,
0:re1]), np.ravel(Vb[re3:re4, 0:re1]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re3:re4,
re1:re2]), np.ravel(Vb[re3:re4, re1:re2]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re3:re4,
re2:re3]), np.ravel(Vb[re3:re4, re2:re3]))
V <- V +
fn.cosinesimilarity(np.ravel(Va[re3:re4,
re3:re4]), np.ravel(Vb[re3:re4, re3:re4]))
meanV <- V / 16
-> (meanH + meanS + meanV) / 3

```

4.1.4 mcolor.py

- a. Atribut

Class ini tidak memiliki atribut.

- b. Metode

<pre> function searchcolor(path_image) {I.S. path_image terdefinisi} {F.S. Memproses setiap image dalam path_image dan menguratkannya sesuai cosine similarity berdasar color} KAMUS LOKAL start_time, similarity:float path_dataset:string Ha, Sa, Va: arr of arr of int final: arr of int re: integer ALGORITMA path_dataset <- get_upload_dir() start_time <- time.time() re <- 200 Ha <- [[0...re]...re] Sa <- [[0...re]...re] Va <- [[0...re]...re] RGBtoHSV(path_image,re,Ha,Sa,Va) if not(exists(path_dataset) <u>and</u> isdir(path_dataset)) then -> {"files": sort_dictionary(final, "similarity", False), "time": time() - start_time} </pre>	Metode untuk menghitung dan menghasilkan array berisi image yang mirip berdasarkan similarity.
--	--

4.1.5 directory.py

a. Atribut

Class ini tidak memiliki atribut.

b. Metode

<pre>function get_upload_dir() -> str {I.S.} {F.S. mengembalikan path directory untuk menyimpan dataset yang diupload}</pre> <p>KAMUS LOKAL</p> <p>ALGORITMA</p> <pre>-> './dataset_upload'</pre>	Mengembalikan path penyimpanan dataset
---	---

4.2 Struktur Program

Website kami menggunakan framework React untuk *frontend* dan Flask untuk *backend*. Setelah terbentuk FE dan algoritma pemrosesan image color dan texture. Pertama-tama akan dibuat API routes, dengan tujuan membantu mengaitkan FE dan BE. Konsepnya secara sederhana adalah mengupload image dan membacanya dalam string melalui `readasBinaryString` function dan men-decode data image dengan melakukan konversi dari binary ke ASCII dengan `btoa` function. Data image yang didapat akan di-post dengan endpoint upload ke lokal folder kita. Setelah itu akan dilakukan proses dalam function `search_color/texture` dan hasilnya akan disimpan dalam sebuah JSON file. Dengan begitu kita dapat mempostingnya di website nanti.

Pada FE kami terdapat 6 file utama: page dan Pagination (css) dan Header, ImageSearch, index, dan Pagination (jsx).

a. ImageSearch.jsx

Memuat berbagai const untuk declare state awal beberapa variabel dan fetching data. Pada function `uploadDataset`, mula-mula terdapat alert jika pengguna belum memasukkan data image inputan.

b. Page.css

Mengatur tampilan dari website secara keseluruhan, mencakup header (navigasi), pagination, dan lain-lain.

c. Pagination.css

Mengatur tampilan dari button pagination.

d. Header.jsx

Lalu dengan menggunakan fileReader, akan dilakukan konversi file yang terbaca menjadi binary string sambil melakukan update persentase upload progress. Function searchImage fokus pada tombol toggle antara color dan texture. Jika tombol toggle mengarah pada color akan dilakukan pemanggilan pada api search_color begitu pula sebaliknya. Bagian selanjutnya adalah struktur lain-lain dari website seperti buttons dan input. Data hasil perhitungan cosine akan ditampilkan dengan menggunakan fungsi .map(). Penggunaan fungsi slice() untuk pagination dan pemanggilan class/function pagination pada Pagination.jsx.

e. Index.jsx

Memuat tambahan-tambahan website seperti petunjuk penggunaan website, basic concepts, dan about us.

f. Pagination.jsx

Memuat function pagination buttons yang terdiri dari tombol first, before, next, dan last.

Sedangkan pada *backend* kami menggunakan beberapa files: color, mtexture, mcolor, main, dan directory. Files-files *backend* kami tempatkan pada folder terpisah “api”.

a. Color.py

File color.py berisi fungsi-fungsi yang digunakan pada pemrosesan CBIR color.

b. Mtexture.py

Berisi fungsi utama yang dibutuhkan untuk mengolah data dari beberapa image dengan memanfaatkan fungsi konversigray dan createcooccurrence yang terdefinisi.

c. Mcolor.py

Berisi fungsi utama penggabungan fungsi-fungsi lain yang dibutuhkan pada color dan akan mereturn sebuah dictionary yang akan diurutkan berdasarkan key nilai cosine similarity secara descending.

d. Main.py

Merupakan kumpulan API routes: search_texture, search_color, upload, dan image. Route search_texture melakukan proses pencarian berdasar tekstur dan mengubah hasilnya dalam JSON, begitu pula route search_color. Sedangkan route upload akan membuat file dengan mkdir pada lokal kita dengan isi image-image dataset yang diupload.

e. Directory.py

Return path folder untuk memasukkan hasil upload dataset pada lokal.

4.3 Tata Cara Penggunaan Program dan Fitur-fitur

Website Allens merupakan platform yang digunakan untuk mencari gambar yang serupa dan relevan. Program ini menggunakan konsep aljabar vektor dalam algoritma pencocokan dengan dua parameter CBIR.

Langkah-langkah penggunaan website sebagai berikut:

a. Unggah dataset dan file gambar

Unggah folder dataset yang berisi gambar-gambar yang ingin digunakan untuk mencari gambar serupa dengan menekan tombol “Select Dataset File”. Selanjutnya, unggah file gambar yang akan dicari dengan menekan tombol “choose a file”.

b. Preview image inputan

Setelah berhasil mengunggah file gambar, gambar beserta nama file akan ditampilkan. Gambar ini akan digunakan untuk mencari gambar yang serupa dan relevan.

c. Pencarian gambar serupa berdasarkan warna dan tekstur

Pencarian gambar yang serupa berdasarkan dua parameter, yaitu warna dan tekstur. Pengguna dapat memilih parameter ini dengan menekan *toggle button* yang telah disediakan di *website*.

d. Pagination untuk hasil

Hasil pencarian akan ditampilkan berdasarkan persentase kemiripan tertinggi. Jika ingin melihat hasil lainnya, pengguna dapat menekan tombol pagination berikutnya yang terdapat di situs web untuk melihat halaman hasil selanjutnya.

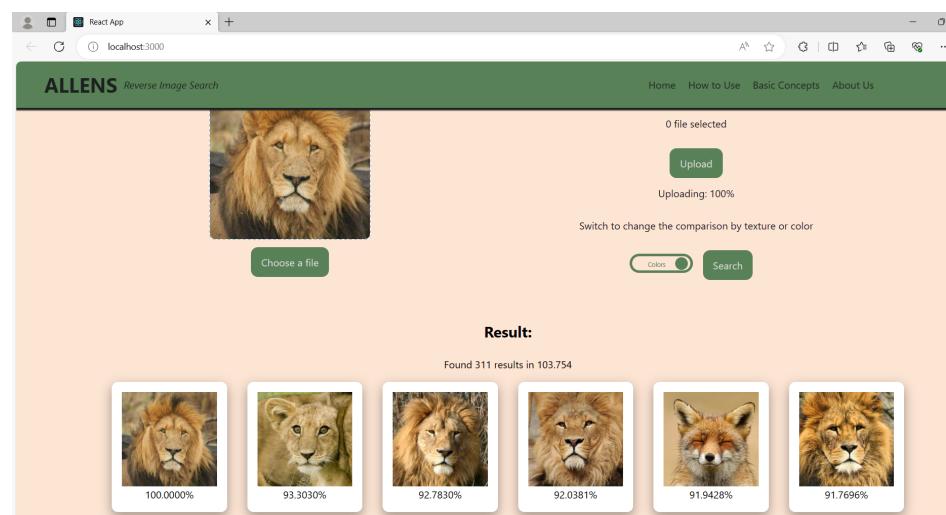
e. Navigasi bar

Website Allens juga dilengkapi dengan navigasi bar di bagian kanan atas, yang terdiri dari menu "Home", "Basic Concepts", "How to Use", dan "About Us". Pengguna dapat mengklik masing-masing menu untuk menavigasi sesuai keinginan. Menu "Home" menampilkan bagian awal website, "Basic Concepts" membawa pengguna ke halaman dengan konsep dasar dari program website, "How to Use" memberikan penjelasan penggunaan website, dan "About Us" menampilkan informasi tentang tim pembuat website.

4.4 Hasil Pengujian

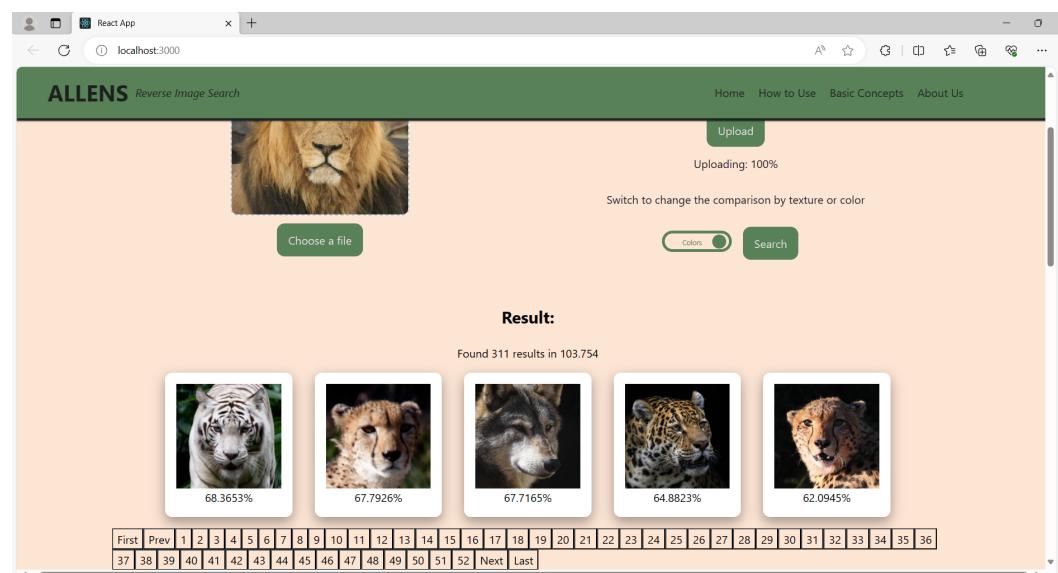
a. Dataset harimau dari referensi [kaggle](#).

Diambil dataset harimau bersumber dari kaggle dan diambil sekitar 311 gambar, kami bandingkan berdasarkan *color* dengan suatu gambar dari dataset yang sama dan didapatkan hasil dari halaman pertama pagination (tingkat kemiripan paling tinggi) sebagai berikut:



4.1 CBIR color image harimau *first* slide

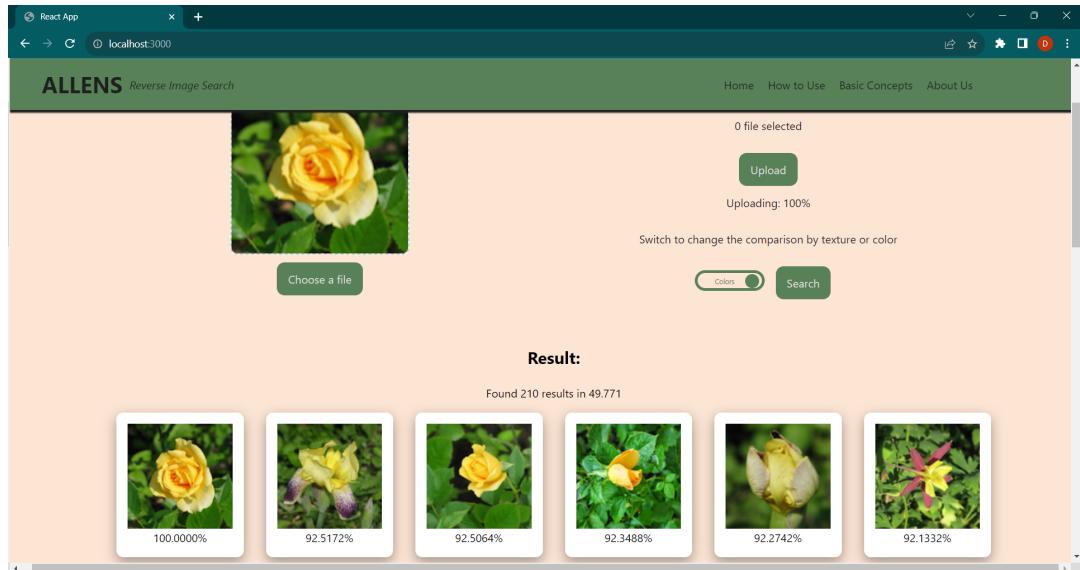
Sedangkan untuk halaman terakhir pagination (tingkat kemiripan paling rendah) didapatkan hasil sebagai berikut:



4.2 CBIR color image harimau *last* slide

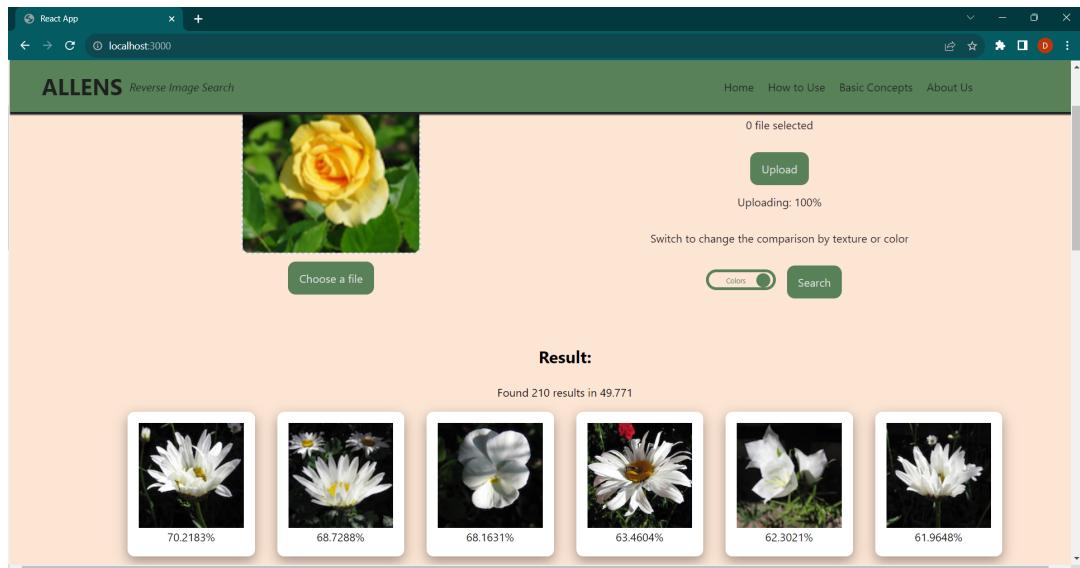
- b. Dataset bunga dari referensi [kaggle](#).

Diambil dataset bunga bersumber dari kaggle dan diambil 210 gambar, kami bandingkan berdasarkan *color* dengan suatu gambar dari dataset yang sama dan didapatkan hasil dari halaman pertama pagination (tingkat kemiripan paling tinggi) sebagai berikut:



4.3 CBIR color image bunga *first* slide

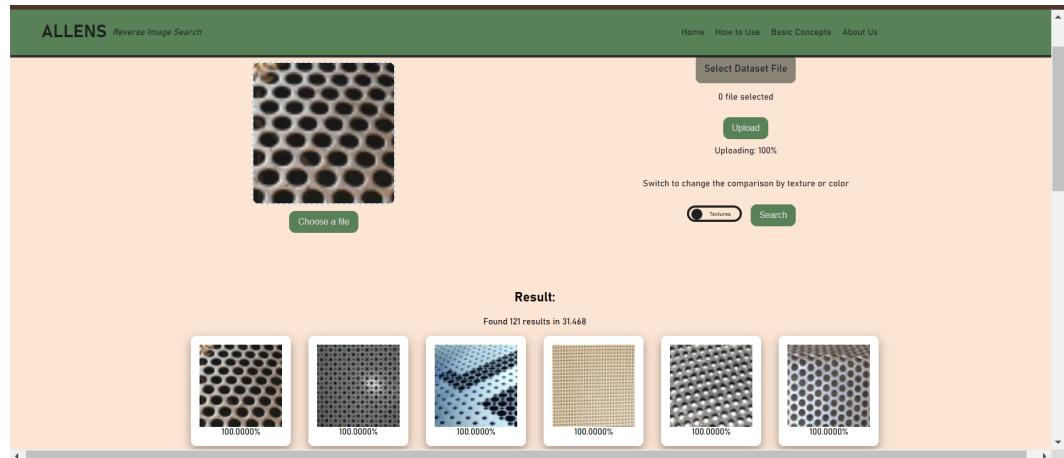
Sedangkan untuk halaman terakhir pagination (tingkat kemiripan paling rendah) didapatkan hasil sebagai berikut:



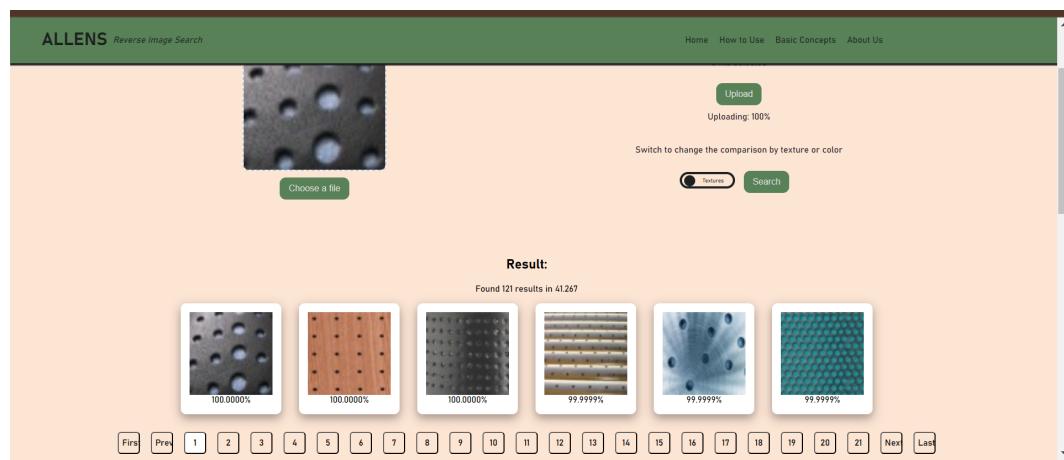
4.4 CBIR color image bunga *last* slide

- c. Dataset perforated dari referensi [github](#) (123 files)

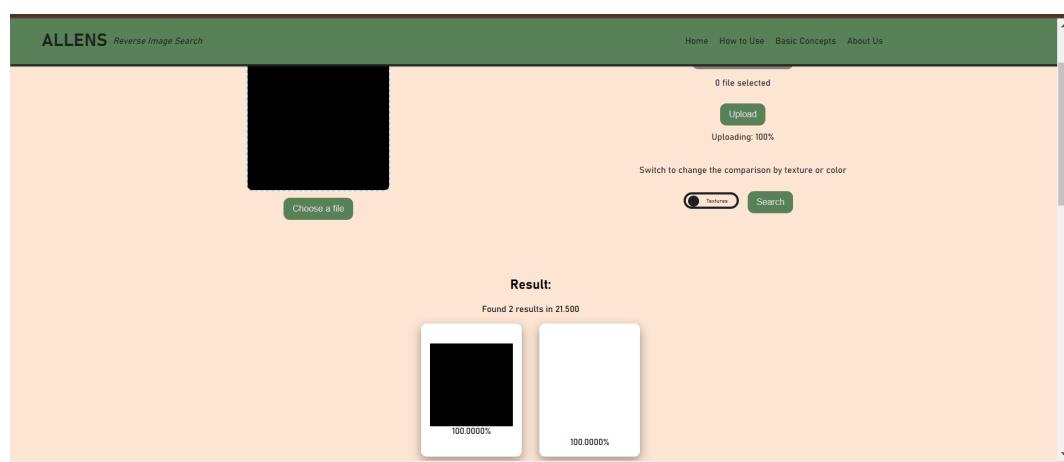
Diambil dataset perforated bersumber dari [github ahmed-BH](#) dan diambil sekitar 123 gambar (ada image tambahan), kami bandingkan berdasarkan *color* dengan suatu gambar dari dataset yang sama dan didapatkan hasil dari halaman pertama pagination (tingkat kemiripan paling tinggi) sebagai berikut:



4.5 CBIR texture



4.6 CBIR texture part 2



4.7 CBIR texture image hitam polos

4.5 Analisis Solusi Algoritma

Dari hasil algoritma dan website yang kami buat, kami menemukan bahwa keakuratan tergantung pada image yang mau kita bandingkan. Apabila image tersebut mengandung banyak pattern dan tekstur yang khusus, sebaiknya kita membandingkannya berdasarkan tekstur. Seperti contoh aplikasi CBIR tekstur pada dataset perforated, yang lebih mementingkan tekstur dan pola yang berepetisi daripada warna. Sedangkan color based CBIR akan lebih relevan untuk analisis image dengan warna yang bermacam-macam dan ramai, seperti dataset *wildlife* dari kaggle di atas. Akan tetapi dalam pembuatan website ini kami menemukan fakta bahwa nilai cosine similarity tekstur selalu saja bernilai tinggi, rata-rata berada di atas 90% (kecuali solid color). Pada solid color (Gambar 4.5), seluruh gambar perforated yang lain tidak akan memenuhi kecuali gambar hitam itu sendiri dan gambar *solid white*. Setelah melakukan analisis lebih lanjut, kami temukan bahwa nilai kontras sebuah gambar dapat melonjak hingga ribuan. Hal inilah yang memungkinkan terjadinya error dalam perhitungan similarity. Vektor yang terbentuk jadi jauh condong ke arah kontras image, sebab nilai homogeneity dan entropy sangatlah kecil (rata-rata satu digit). Akibatnya nilai homogeneity dan entropy seolah-olah menjadi tidak berpengaruh lagi dan nilai kontras lah yang mendominasi. Maka dari itu, kami dapat menyimpulkan bahwa kedua CBIR efektif pada kasus image-image tertentu.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Tugas besar 2 IF2123 Aljabar Linier dan Geometri fokus pada implementasi sistem temu balik gambar dengan memanfaatkan aljabar vektor yang dibentuk dalam sebuah *website*. Implementasi tersebut mencakup proses upload dataset yang memberikan akses dan kemampuan untuk memproses dataset gambar dalam sistem. Selain itu, fitur pratinjau gambar memberikan fleksibilitas kepada pengguna untuk memeriksa gambar input sebelum diproses. Implementasi pencarian berbasis warna dan tekstur menggunakan konsep aljabar vektor, memungkinkan sistem untuk menghasilkan hasil pencarian yang akurat dan relevan. Secara keseluruhan, website kami memuat fitur-fitur berikut:

- a. Upload dataset dan file image
- b. Preview image inputan
- c. Pencarian image yang mirip berdasar warna dan tekstur
- d. Pagination untuk hasil
- e. Navigation bar

5.2 Saran

Dalam tugas besar 2 IF2123 Aljabar Linier dan Geometri kami banyak belajar tentang aplikasi aljabar vektor dengan pembuatan sebuah *website*. Berikut beberapa saran dari kami dalam tugas besar ini:

- a. Pemberian spesifikasi yang lebih jelas dengan cara-cara perhitungannya karena banyak *step-step* yang seperti terlewat atau bahkan belum dicantumkan dalam spesifikasi.
- b. Pemberian lebih banyak *guide* untuk pembuatan website terutama di bagian menghubungkan *back end* dan *front end*.

5.3 Komentar dan Tanggapan

Tugas besar ini menunjukkan pendekatan yang komprehensif dalam mengembangkan sistem Content-Based Image Retrieval (CBIR) dengan menggunakan aljabar vektor sebagai dasar. Integrasi antarmuka pengguna yang mudah dipahami, metode analisis vektor, dan algoritma klasifikasi untuk konten visual menunjukkan perhatian yang teliti terhadap kebutuhan pengguna. Aspek penting seperti penanganan dataset besar, keamanan data, pelatihan pengguna, dan pemantauan kinerja telah diperhatikan dengan baik.

5.4 Refleksi

Melalui penyelesaian Tugas Besar 2 IF2123 Aljabar Linier dan Geometri, kami merasakan pengalaman belajar yang sangat berharga. Tugas besar ini tidak hanya memberikan pemahaman mendalam tentang konsep aljabar vektor, tetapi juga membuka wawasan terhadap penerapannya dalam dunia nyata, khususnya dalam sistem temu balik gambar dan pembuatan sebuah *website*. Selama pembuatan aplikasi web, kami juga mengalami berbagai masalah dan tantangan sehingga kami harus lebih banyak mengeksplor tentang pembuatan *website* agar dapat mengatasi masalah tersebut. Selain itu, menerjemahkan konsep aljabar vektor ke dalam kode program tidak hanya menguji pemahaman konsep, tetapi juga mengasah kemampuan pemrograman. Kolaborasi dalam merancang dan mengimplementasikan fitur-fitur aplikasi memperkuat keterampilan komunikasi dan koordinasi di antara anggota tim. Secara keseluruhan, tugas besar ini tidak hanya menjadi evaluasi akademis, tetapi juga mengasah kemampuan kerjasama dalam tim.

5.5 Ruang Perbaikan dan Pengembangan

Pada bagian fitur pilihan variabel pembanding *texture* dan *color* dapat diperluas dengan pilihan variabel pembanding yang lain. Bahkan

mungkin dapat dikembangkan dengan mengkombinasikan banyak variabel. Sehingga tingkat kemiripan gambar yang dikeluarkan akan semakin akurat.

Penting juga untuk memperhatikan umpan balik dari pengguna guna memahami lebih dalam kebutuhan terhadap sistem. Dengan memperhatikan saran dan pengalaman pengguna maka dapat dilakukan perbaikan atau peningkatan pada fitur-fitur sistem agar lebih sesuai dengan kebutuhan pengguna.

Selanjutnya, pengembangan sistem dapat difokuskan pada peningkatan keamanan data, terutama jika melibatkan gambar-gambar yang bersifat sensitif. Penerapan metode enkripsi yang kuat, kontrol akses yang ketat, dan langkah-langkah keamanan data lainnya dapat membantu melindungi privasi dan integritas informasi. Pengembangan ini penting untuk menjaga kepercayaan dan keamanan pengguna.

Selain itu, diperlukan pengembangan fitur-fitur baru atau teknologi terbaru yang dapat meningkatkan kinerja dan fungsionalitas sistem. Dengan tetap berfokus pada evolusi teknologi, sistem CBIR dapat tetap relevan dan terus berkembang. Contoh pengembangan yang dapat dilakukan, seperti berikut:

- a. Mencari cara mempercepat processing image hingga kecepatan kurang dari 1 detik per image (catatan ukuran image besar).
- b. Memperbaiki tampilan website menjadi lebih baik dan lebih lengkap lagi dengan menambah berbagai fitur.

Dengan demikian, melalui umpan balik, perbaikan keamanan data, dan integrasi fitur baru, sistem CBIR ini dapat terus berkembang menjadi solusi yang lebih relevan dan efektif bagi pengguna.

DAFTAR PUSTAKA

Link github: <https://github.com/dianatrihy/Algeo02-22012>

Link video: https://youtu.be/iouKJjNTCZ0?si=j0H4Vq_Nx31fJifT

Feature Extraction : Gray Level Co-occurrence Matrix (GLCM). (2020, July 16).

Muhammad Yunus. Retrieved November 12, 2023, from
<https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1>

Grinberg, M. (2020, February 21). *How To Create a React + Flask Project*. Miguel Grinberg. Retrieved November 12, 2023, from
<https://blog.miguelgrinberg.com/post/how-to-create-a-react--flask-project>

How to Compress Images Using Python and PIL? (2023, July 20). Tutorialspoint.
Retrieved November 10, 2023, from
<https://www.tutorialspoint.com/how-to-compress-images-using-python-and-pil>

IF2123 Aljabar Geometri - Semester I Tahun 2023/2024. (n.d.). Informatika.
Retrieved November 9, 2023, from
<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/algeo23-24.htm>

RGB to HSV Color Conversion Algorithm - Mathematics Stack Exchange. (2013, November 8). Math Stack Exchange. Retrieved November 6, 2023, from
<https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm>

Smyth, P. (2018, April 2). *Creating Web APIs with Python and Flask*. Programming

Historian. Retrieved November 13, 2023, from

<https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask>

SQL Tutorial. (n.d.). W3Schools. Retrieved November 9, 2023, from

<https://www.w3schools.com/sql/default.asp>