

LAPORAN TUGAS BESAR 2
IF2123 ALJABAR LINEAR DAN GEOMETRI
SISTEM PERSAMAAN LINIER, DETERMINAN, DAN
APLIKASINYA
SEMESTER I TAHUN 2023/2024



Disusun oleh:

dizkon

Tazkia Nizami	13522032
Dhidit Abdi Aziz	13522040
Vanson Kurnialim	13522049

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

BAB I

DESKRIPSI MASALAH

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

BAB II

LANDASAN TEORI

A. Dasar Teori Secara Umum

Image adalah sebuah visualisasi representasi dari suatu objek atau biasa disebut sebagai gambar yang terdiri dari pixel-pixel. Pixel ini memiliki nilai ‘warna’ yang jika dikombinasikan dengan keseluruhan pixel lain akan membentuk suatu gambar atau image. CBIR warna dan tekstur ini memanfaatkan aspek dasar dari pixel ini untuk menarik suatu sifat dari suatu image sehingga dengan metode yang tepat, kita dapat membandingkan dua atau lebih gambar berdasarkan sifat atau atribut yang telah didapatkan.

B. Pengembangan Website

Website adalah serangkaian halaman web berisi informasi yang terhubung satu sama lain dan diakses melalui internet. Website terdiri dari komponen berupa teks, gambar, suara, animasi dan jenis lainnya yang dapat dimanfaatkan oleh pengguna untuk mencapai suatu tujuan. Semua orang dapat mengembangkan sebuah website sesuai dengan preferensinya sendiri, bisa untuk tujuan pribadi ataupun perusahaan.

Dalam mengembangkan sebuah website, tahap paling awal dalam pengembangannya adalah menentukan perencanaan. Perencanaan ini meliputi menentukan tujuan dari website, menentukan fitur yang harus ada pada website, alat yang digunakan untuk mengembangkan website, hingga tampilan dari website. Tahap ini juga harus memperhatikan waktu *deadline* dari pengembangan website. Perencanaan harus dilakukan sebaik dan sematang mungkin agar setiap bagian dari website akan bekerja berdampingan untuk memuaskan pengguna.

Langkah selanjutnya adalah penulisan kode. Penulisan kode harus tetap mengacu pada hasil dari perencanaan. Hal ini untuk menghindari konflik dengan bagian lain yang sudah direncanakan. Pada saat ini juga akan lebih baik jika pengembang menuliskan beberapa komentar penjelasan pada kodennya. Hal ini untuk membantu orang lain yang tidak terlibat langsung dengan bagian yang sedang ditulis kodennya untuk memahami maksud dan bagaimana cara kerja kode yang ditulis.

Setelah penulisan kode sudah selesai, proses testing dimulai. Testing dilakukan dengan cara menjalankan website kemudian melakukan berbagai interaksi dengan website. Interaksi yang dilakukan harus sebanyak mungkin untuk memastikan bahwa pengguna tidak akan menemukan awakutu pada website yang sudah ditulis. Setiap awakutu sebaiknya dicatat untuk membantu pengembang dalam menyempurnakan website di tahap selanjutnya.

Tahap selanjutnya adalah menghilangkan setiap awakutu dengan cara menyempurnakan kode yang sudah ditulis sebelumnya. Proses ini membutuhkan wawasan yang luas dan mungkin sumber dari internet untuk memastikan bahwa awakutu yang ada dapat teratasi dengan baik. Kemudian tahap testing dilakukan lagi untuk memastikan bahwa hasil dari penyempurnaan kode sudah dilakukan dengan benar. Tahapan memperbaiki kode dan testing ini dilakukan berkali-kali hingga pengembang merasa sudah cukup menyelesaikan awakutu yang ada.

Setelah pengembang merasa bahwa kode untuk websitenya sudah cukup baik, pengembang bisa mulai meluncurkan webnya di internet. Tahap ini bisa dibilang merupakan tahap terakhir dari pengembangan website, namun pengembang harus melakukan pemeliharaan terhadap website yang sudah dibuat. Tahap pemeliharaan sendiri berarti memastikan bahwa setiap bagian dari website bisa berjalan dengan baik meskipun teknologi yang digunakan pada website mungkin sudah mengalami pembaharuan, ataupun menambahkan bagian baru yang lebih sesuai dengan perkembangan zaman saat ini pada website.

BAB III

ANALISIS PEMECAHAN MASALAH

A. Langkah-langkah Pemecahan Masalah

- Membentuk vektor warna dari suatu image

Data RGB dari suatu image diambil lalu dinormalisasi, yaitu dibagi dengan 255, agar range data menjadi 0-1. Tiap pixel akan dicari nilai maksimum, minimum, dan selisihnya. Selanjutnya, nilai RGB dari tiap pixel tersebut akan diubah menjadi HSV, bergantung pada nilai maksimum RGB pixel tersebut. Warna global HSV lebih dipilih karena warna tersebut dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum untuk digunakan.

Untuk memudahkan perbandingan matrix HSV dari dua gambar, matrix tersebut akan disimplifikasi dan diseragamkan menjadi satu *array* dengan 14 elemen. Array ini berupa histogram, yaitu hasil perhitungan jumlah nilai H, S, dan V pada rentang tertentu. Terakhir, dua *array* dari gambar yang berbeda akan dihitung *cosine similarity*-nya.

- Membentuk vektor tekstur dari suatu image

Data warna atau RGB dari suatu image akan diambil. Kemudian, data tersebut ditransformasi menjadi matrix grayscale atau matrix hitam putih dengan cara mengalikan masing-masing aspek warna dengan faktor warna masing-masing dan dijumlahkan satu sama lain sehingga didapatkan sebuah value yang hanya menunjukkan nilai hitam-putih dari suatu image.

Dari matrix grayscale ini, akan dibentuk matrix Co-occurrence yang merepresentasikan keseringan suatu nilai muncul. Dikarenakan range nilai dari grayscale adalah 0-255, maka besar matrix Co-occurrence adalah 256x256. Distance yang digunakan adalah sebesar 1 karena jarak tersebut merupakan yang paling cocok untuk CBIR. Angle yang digunakan adalah 0 derajat karena setelah dicoba, tidak ada sudut paling optimal untuk CBIR ini.

Matrix symmetry merupakan matrix yang dijumlahkan dengan transpose dari matrix tersebut. Setelah membuat matrix symmetry dari matrix Co-occurrence yang kita punya, kita akan normalisasi matrix tersebut sehingga data di dalamnya tidak terlalu besar. Sekarang, baru kita bentuk vektor tekstur dengan 5 komponen sebagai berikut : Contrast, Homogeneity, dan Entropy, Energy, dan Correlation.

- Mencari kesamaan atau similarity dari 2 gambar

Menggunakan metode CBIR warna maupun CBIR tekstur, cosine similarity merupakan jawaban untuk mencari tingkat kesamaan gambar. Cosine similarity menghitung derajat kesamaan arah dari kedua vektor sehingga dalam hal ini bisa dijadikan sebagai penentu tingkat kesamaan dari 2 gambar.

- Menyusun segala komponen menjadi bagian dari website

Website akan dibangun menggunakan framework Django. Setelah bagian display seperti tombol dan tempat unggah selesai, algoritma pencarian gambar berdasarkan kedua parameter warna dan tekstur akan diimplementasikan ke dalam website. Lalu, hasil dari pencarian yang memenuhi batasan akan ditampilkan pada halaman website. Selanjutnya, website akan diperbaiki dan dikembangkan lagi dengan fitur-fitur serta fungsionalitas tambahan seperti kamera dan *cache*.

B. Proses pemetaan masalah menjadi elemen-elemen pada aljabar geometri.

Berbagai data yang dipergunakan dalam tugas besar ini merupakan array of array atau bisa dibilang matrix. Sehingga pemrosesan data yang kita punya merupakan pemanfaatan dari operasi matrix. Seperti pada CBIR dengan parameter tekstur, diperlukan adanya pembentukan matrix simetri yaitu matrix yang dijumlahkan oleh transpose dari matrix itu sendiri. Kemudian matrix simetri tersebut juga diubah lagi menjadi ‘matrix satuan’.

Fungsi cosine similarity yang digunakan untuk mencari kesamaan dari kedua gambar juga merupakan elemen dari aljabar geometri dimana kedua vektor dicari tingkat kesamaannya dengan rumusan

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

Gambar 1. Rumus cosine similarity.

C. Contoh ilustrasi kasus dan penyelesaiannya.

Misal user ingin mencari gambar dari dataset dengan gambar dari query. User akan mengunggah query dan dataset pada website. Pada saat user mengunggah dataset, website akan otomatis menghitung data dari tiap dataset dan disimpan ke dalam database dengan kedua algoritma warna dan tekstur. Proses ini memang sengaja dilakukan pada saat user mengunggah dataset dan bukan saat tombol search ditekan untuk efisiensi waktu dan meminimalisir waktu

yang eksekusi yang berlebih dan bertumpuk. Data tersebut juga disimpan di dalam database atau menggunakan *cache* dengan alasan yang sama serta untuk reusability data.

User dapat memilih ingin menggunakan CBIR metode apa dengan menekan tombol *toggle* warna dan teksur. Saat user menekan tombol search, data dari query akan dihitung dengan cara yang sesuai dengan tombol *toggle*. Kemudian data dari query akan dibandingkan dengan setiap data dari dataset menggunakan algoritma cosine similarity dan dataset yang sesuai akan disimpan oleh website. Data tersebutlah yang akan ditampilkan kepada pengguna pada layar website.

BAB IV

IMPLEMENTASI DAN UJI COBA

A. Implementasi Program Utama

Mengambil data Warna :

```
function convert_rgb_hsv(input string: path) → array
[0..13] of integer
{Menerima masukan path directory dari image dan
mengembalikan histogram hsv}
{Versi tanpa numpy}

{Kamus Lokal}
matrix_rgb: array [0 .. pixel-1] of array [0 .. pixel -1 ]
of array [0..2] of integer
hsv: array [0 .. 13] of integer
i, j, k, cmin, cmax, indikatormax, delta: integer

{Algoritma}
i traversal [0..13]
    hsv[i] ← 0

{Mengambil matrix rgb dari suatu gambar}
matrix_rgb ← ImagetORGB(path_gambar)

i traversal[0..pixel-1]
    j traversal [0 .. pixel-1]
        cmin ← 1
        cmax ← 0
        k traversal [0..2]
            matrix_rgb[i][j][k] ← matrix_rgb[i][j][k] /
255.0
            if (matrix_rgb[i][j][k] >= cmax) then
                cmax ← matrix_rgb[i][j][k]
                if (j = 0) then
                    indikatormax ← 0
                else if (j = 1) then
                    indikatormax ← 1
                else
                    indikatormax ← 2
            if (matrix_rgb[i][j][k] <= cmin) then
                cmin ← matrix_rgb[i][j]
delta ← cmax - cmin

{Mengubah nilai r menjadi h}
```

```

if delta=0 then
    matrix_rgb[i][j][0] ← 0
else if indikatormax = 0 then
    matrix_rgb[i][j][0] ← 60*((matrix_rgb[i][j][1] - matrix_rgb[i][j][2] / delta) mod 6)
else if indikatormax = 1 then
    matrix_rgb[i][j][0] ← 60*((matrix_rgb[i][j][2] - matrix_rgb[i][j][0])/delta) + 2
else
    matrix_rgb[i][j][0] ← 60*((matrix_rgb[i][j][0] - matrix_rgb[i][j][1]) / delta) + 4

matrix_rgb[i][j][0] ← round(matrix_rgb[i][j][0])

{Mengubah nilai g menjadi s}
if cmax = 0 then
    matrix_rgb[i][j][1] ← 0
else
    matrix_rgb[i][j][1] ← delta / cmax

{Mengubah nilai b menjadi v}
matrix_rgb[i][j][2] ← cmax

{Mengisi histogram hsv}
if (matrix_rgb[i][j][0] >= 316) and (matrix_rgb[i][j][0] <= 360) then
    h[0] ← h[0] + 1
else if (matrix_rgb[i][j][0] >= 1) and (matrix_rgb[i][j][0] <= 25) then
    h[1] ← h[1] + 1
else if (matrix_rgb[i][j][0] >= 26) and (matrix_rgb[i][j][0] <= 40) then
    h[2] ← h[2] + 1
else if (matrix_rgb[i][j][0] >= 41) and (matrix_rgb[i][j][0] <= 120) then
    h[3] ← h[3] + 1
else if (matrix_rgb[i][j][0] >= 121) and (matrix_rgb[i][j][0] <= 190) then
    h[4] ← h[4] + 1
else if (matrix_rgb[i][j][0] >= 191) and

```

```

(matrix_rgb[i][j][0] <= 270):
    h[5] ← h[5] + 1
else if (matrix_rgb[i][j][0] >= 271) and
(matrix_rgb[i][j][0] <= 295):
    h[6] ← h[6] + 1
else if (matrix_rgb[i][j][0] >= 295) and
(matrix_rgb[i][j][0] <= 315):
    h[7] ← h[7] + 1

if (matrix_rgb[i][j][1] >= 0) and
(matrix_rgb[i][j][1] < 0.2) then
    h[8] ← h[8] + 1
else if (matrix_rgb[i][j][1] >= 0.2) and
(matrix_rgb[i][j][1] < 0.7):
    h[9] ← h[9] + 1
else if (matrix_rgb[i][j][1] >= 0.7) and
(matrix_rgb[i][j][1] <= 1):
    h[10] ← h[10] + 1

if (matrix_rgb[i][j][2] >= 0) and
(matrix_rgb[i][j][2] < 0.2):
    h[11] ← h[11] + 1
else if (matrix_rgb[i][j][2] >= 0.2) and
(matrix_rgb[i][j][2] < 0.7):
    h[12] ← h[12] + 1
else if (matrix_rgb[i][j][2] >= 0.7) and
(matrix_rgb[i][j][2] <= 1):
    h[13] ← h[13] + 1

```

→ hsv

Mengambil data Tekstur :

```

function getData (input string : path) → array of integer[0..4]
{Menerima masukan path directory dari image dan mengembalikan
vektor tekstur}

```

Kamus Lokal
image_array : array of integer[][]
kontras, homo, entro, energy, correlation : real
vektor : array of integer[0..4]

```

image_array ← ImagetorRGB(path)
image_array ← Grayscale(image_array)

```

```

image_array ← CoccurrenceMatrix(image_array)

image_array ← CoccurrenceMatrix(image_array)
image_array ← symmetric(image_array)
image_array ← normalize(image_array)

kontras ← contrast(image_array)
homo ← homogeneity(image_array)
entro ← entropy(image_array)
energy ← Energy(image_array)
correlation ← Correlation(image_array)

vektor[0] ← kontras
vektor[1] ← homo
vektor[2] ← entro
vektor[3] ← energy
vektor[4] ← correlation

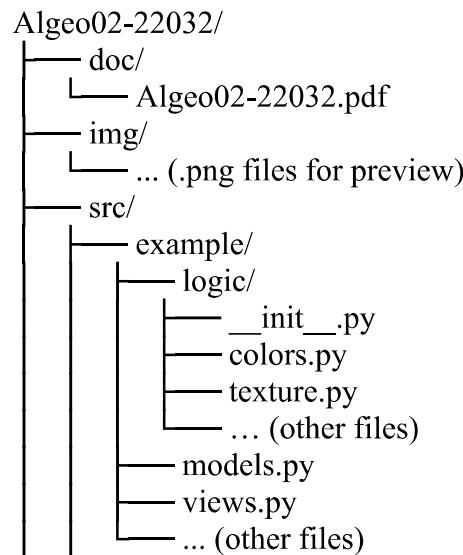
→ vektor

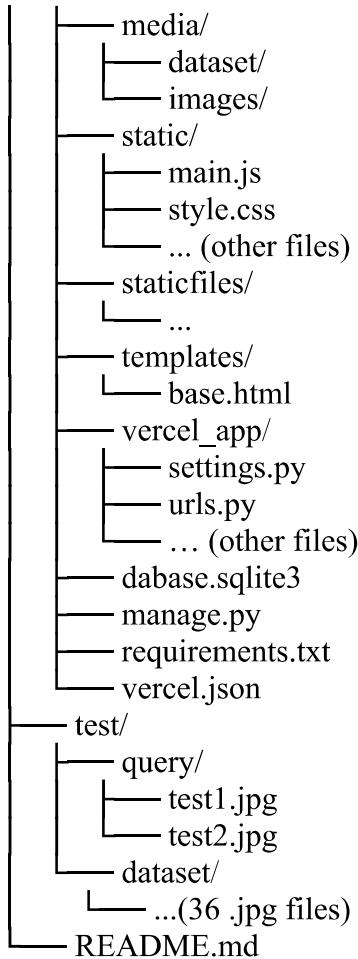
```

B. Penjelasan Struktur Program berdasarkan Spesifikasi

Website ini dikembangkan dengan framework Django. Django dipilih karena salah satu artikel menuliskan tentang Django dengan quote “Django likes to call itself the web framework ‘for perfectionists with deadlines,’” Python sebagai bahasa yang sudah pernah dipelajari sebelumnya dan memiliki syntax yang mendekati bahasa manusia juga menjadi alasan kuat untuk menuliskan program kami dalam Python.

Menggunakan template yang sudah disediakan oleh Vercel, tahap pengembangan website langsung lompat dan fokus ke bagian mengembangkan mekanisme CBIR dan tampilan website. Setelah pengembangan dirasa cukup, berikut adalah gambaran struktur dari repository yang telah selesai dibuat





Penjelasan dari setiap folder dari repository adalah sebagai berikut:

- doc adalah folder yang digunakan untuk menyimpan laporan hasil tugas besar yang kemudian akan dikumpulkan
- img adalah folder yang berisi foto .png yang menampilkan preview penggunaan dari website CBIR yang telah dibuat
- test adalah folder yang memuat gambar query contoh dan beberapa foto yang dapat digunakan sebagai dataset untuk pengguna mencoba website CBIR yang telah dibuat
- src adalah folder yang memuat seluruh komponen website yang dibangun dengan framework Django.

Folder yang menjadi fokus di sini adalah file texture.py dan color.py yang terletak di Algeo02-22032/src/example/logic/. Kedua file ini menyimpan algoritma yang digunakan untuk melakukan CBIR dari gambar query dari semua gambar dataset.

Saat website dijalankan (dengan menggunakan perintah “py manage.py runserver” di folder src), pengguna akan melihat tampilan utama website yang dapat

dilihat pada bagian D. Hasil Pengujian. Pengguna dapat melakukan CBIR dengan menggunakan website yang sedang dijalankan.

Pengguna dapat mengunggah gambar query dengan cara melakukan drag & drop di daerah unggah yang terletak di kiri atas. Program ini kemudian akan langsung mengunggah gambar tersebut ke folder media/images. Selain itu, gambar juga akan diambil data tekstur dan warnanya, yang kemudian akan digunakan dalam proses CBIR. proses ini diatur pada file views.py, tepatnya pada fungsi bernama upload. Website juga kemudian akan menampilkan gambar tersebut di posisi tempat pengguna mengupload gambar query.

Pada bagian kanan atas, terdapat sebuah *switch/toggle* yang digunakan untuk menentukan metode CBIR yang akan digunakan. Terdapat dua pilihan yaitu *color* atau *texture* dengan pilihan default saat program dimulai adalah *color*. Pengguna hanya perlu menekan switch tersebut untuk mengubah metode CBIR yang akan digunakan dalam pencarian gambar.

Tepat di bawahnya, terdapat sebuah tombol bertuliskan “Search” yang digunakan untuk menampilkan hasil CBIR query dengan dataset. Ketika tombol ini ditekan, website akan menampilkan gambar dari dataset yang merupakan hasil dari CBIR dengan metode yang terpilih pada *switch*. Tombol ini akan menjalankan sebuah fungsi pada views.py yaitu update_result. Fungsi update_result akan mengirimkan sebuah data Json ke frontend yang berisi gambar yang akan ditampilkan, seberapa besar similaritasnya, berapa banyak gambar yang memiliki similaritas diatas 60% dan waktu eksekusi proses CBIR. Sebuah script javascript kemudian akan dijalankan dan melakukan pembaharuan halaman dan menampilkan gambar-gambar tersebut secara terurut dari yang memiliki similaritas terbesar hingga terkecil. Halaman web akan menampilkan 6 gambar per halaman dan pengguna cukup menekan tombol next atau previous untuk melihat 6 gambar selanjutnya atau sebelumnya.

Pada bagian yang sama, terdapat pula sebuah tombol yang dapat digunakan untuk mengakses kamera pengguna. Setelah ditekan, maka akan muncul tampilan dari kamera pengguna pada layar. Setiap 5 detik, kamera akan mengambil gambar secara otomatis dan kemudian menjalankan fungsi upload pada views.py yang mengakibatkan foto yang diambil menjadi sebuah query yang kemudian akan digunakan pada CBIR. Setelah itu, fungsi update_result di views.py akan dijalankan dan menampilkan hasil CBIR dari gambar query terhadap dataset. Pengguna cukup menekan tombol akses kamera lagi untuk mematikan kamera.

Selanjutnya, terdapat sebuah daerah unggah lebih besar yang digunakan untuk mengunggah folder dataset yang ingin digunakan pengguna. Pengguna cukup melakukan drag & drop folder dataset yang ingin digunakan ke daerah tersebut. Ketika pengguna mengunggah gambar di daerah tersebut, fungsi uploadDataset di views.py akan dijalankan. Fungsi ini akan mengunggah gambar-gambar tersebut satu per satu ke folder media/dataset/ dan mengekstrak data tekstur dan warna setiap gambar. Perbedaan fungsi uploadDataset dengan upload adalah fungsi ini juga akan menempatkan hasil ekstrak data warna dan tekstur tersebut ke database. Hal ini dilakukan untuk mempersingkat proses CBIR karena data tekstur dan warna dari dataset sudah dihitung sebelumnya. Sehingga,

saat pengguna menjalankan fungsi update_result (melalui tombol search ataupun dengan akses kamera) data gambar dataset sudah siap untuk dibandingkan dengan gambar query dan waktu eksekusi akan semakin singkat.

C. Penjelasan Tata Cara Penggunaan Program

1. Upload gambar dataset.

Untuk mengunggah gambar dataset yang akan digunakan, pengguna bisa melakukan drag & drop folder yang berisi gambar-gambar ke bagian upload dataset. Secara otomatis, gambar akan terupload satu per satu ke database. Proses ini membutuhkan waktu yang lumayan lama karena jumlah gambar dataset sudah sewajarnya berjumlah banyak dan gambar juga langsung di ekstrak data warna dan teksturnya untuk kemudian dilakukan perbandingan gambar.

Setelah mengunggah gambar yang cukup banyak, website akan terasa cukup berat. Pengguna dapat melakukan refresh halaman untuk menyegarkan kembali website menjadi lancar lagi.

2. Upload gambar Query

a. Melalui upload gambar

Pengguna bisa mengunggah gambar Query dengan melakukan drag & drop ke kotak upload yang berada paling atas. Untuk gambar berdimensi sekitar 1100 x 1400 pixel sendiri akan membutuhkan waktu maksimal 4 detik untuk diunggah. Hal ini cukup singkat mengingat gambar yang diunggah langsung diambil data warna dan teksturnya.

b. Melalui camera pengguna

Pengguna bisa mengunggah gambar Query dengan menggunakan kamera yang dimiliki oleh pengguna. Pengguna cukup menekan tombol “open camera” untuk mengakses kamera. Kamera kemudian akan mengambil gambar setiap 5 detik dan langsung menampilkan hasilnya di bagian hasil sesuai dengan metode yang terpilih saat ini.

3. Pilih Metode CBIR yang ingin digunakan (Tekstur / Warna)

Pengguna dapat memilih metode untuk melakukan CBIR antara Query dengan gambar di Dataset. Pengguna cukup mengatur “switch” yang terletak pada bagian kanan atas website dan bertuliskan color dan texture. fitur kamera sendiri akan menampilkan gambar sesuai dengan metode CBIR yang sedang dipilih.

4. Tekan tombol Search

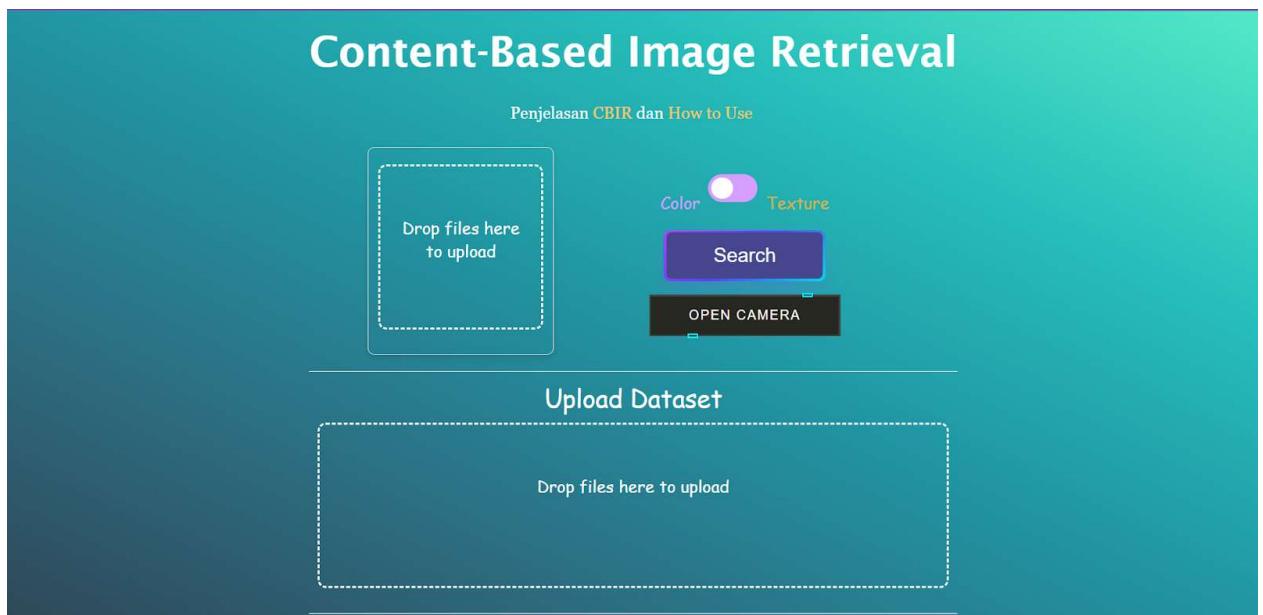
Langkah terakhir adalah menekan tombol Search, setelah tombol Search ditekan, website akan menampilkan 6 gambar pertama yang memiliki similaritas

tertinggi. Pengguna bisa melihat gambar-gambar selanjutnya dengan menekan tombol next di bawah gambar-gambar yang tertampil pada bagian result.

Pada bagian result juga akan tertulis berapa lama proses pencarian gambar dilakukan dan berapa banyak gambar yang diperoleh.

Setelah hasil CBIR ditampilkan, pengguna masih bisa menggunakan website ini dengan cara mengunggah gambar baru di bagian query dengan cara mengklik di bagian sekitar gambar Query saat ini atau dengan drag & drop gambar query baru ke gambar Query saat ini. Pengguna juga bisa menambahkan dataset tambahan dengan cara yang relatif sama. Pengguna juga bisa mengganti metode CBIR dengan menekan switch color-texture.

D. Hasil Pengujian

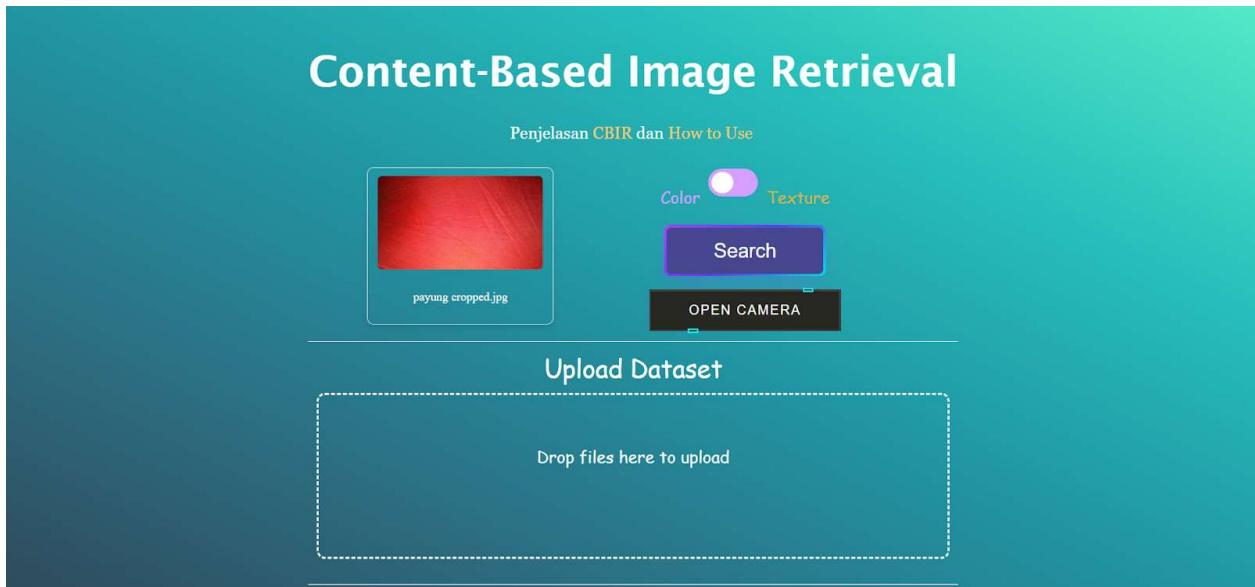


Gambar 2. Tampilan utama ketika pengguna mengakses website.

1. Pengujian Warna

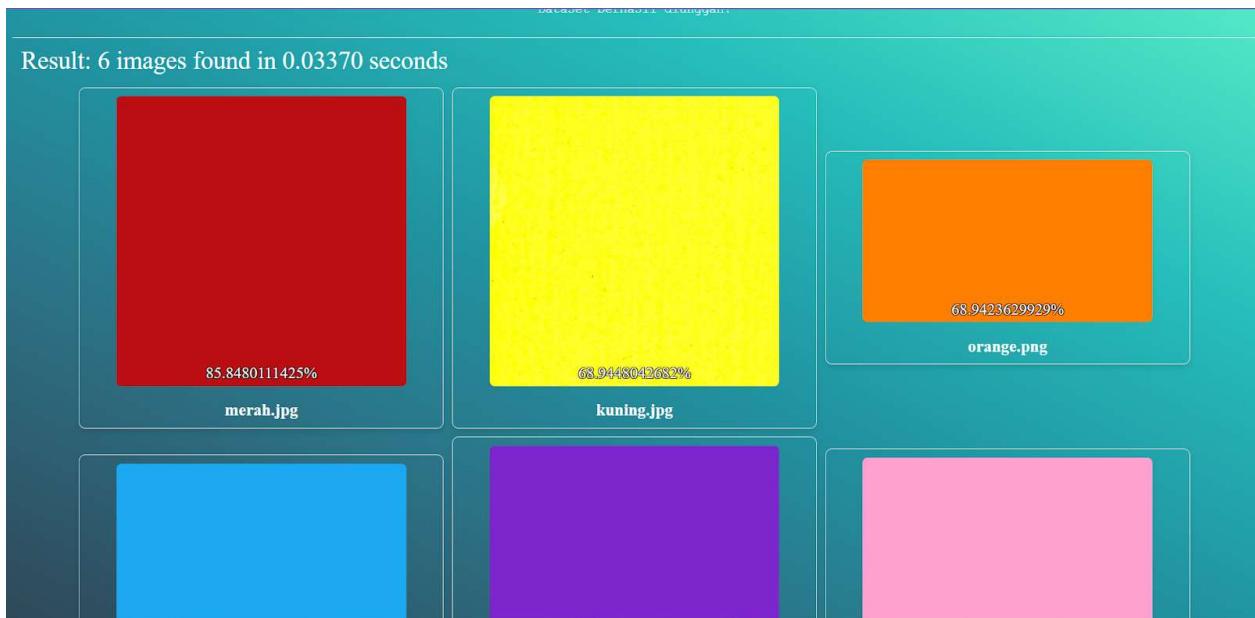
Pengujian dilakukan dengan gambar query adalah foto payung berwarna merah yang difoto secara dekat, dan dataset yang digunakan adalah warna-warna merah, oranye, kuning, hijau, biru, pink, dan ungu.

a. Query



Gambar 3. Tampilan query metode warna.

b. Hasil



Gambar 4. Tampilan hasil metode warna.

c. Penjelasan Tambahan

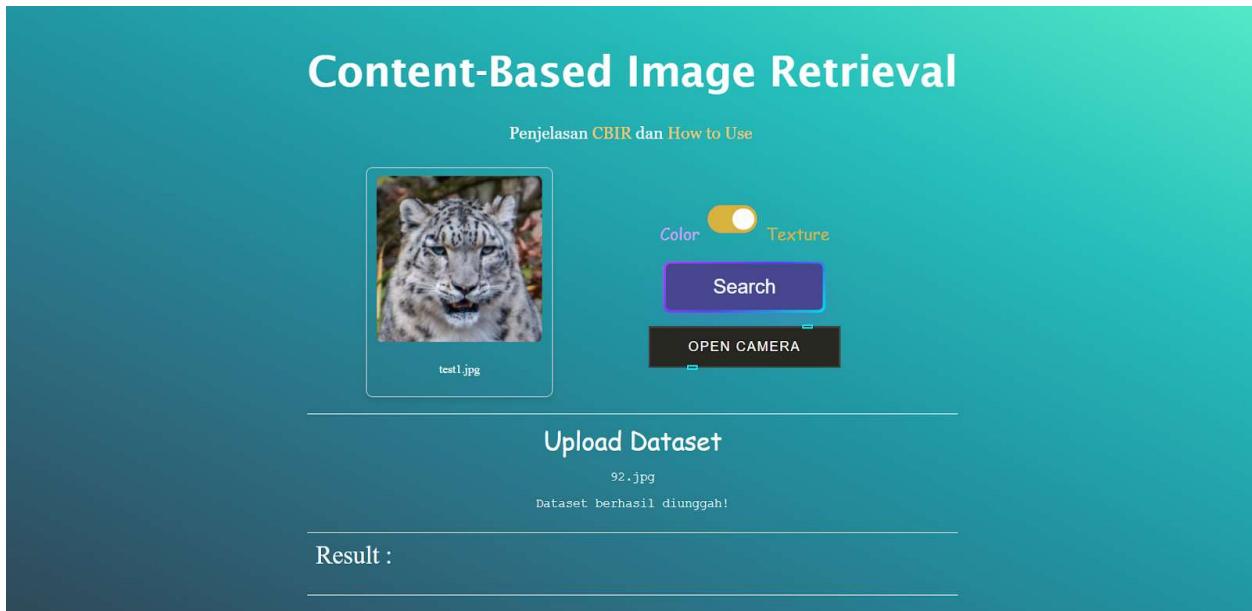
Karena foto yang diambil dan dijadikan query tidak sempurna dalam menunjukkan warna dari payung, muncul hasil warna lain yang memiliki kemiripan yang sama. Namun warna dengan kemiripan lainnya yang tinggi memang masih dekat dengan warna merah (seperti kuning dan oranye). Warna

merah juga tidak menunjukkan similaritas 100% karena *range* warna merah sendiri cukup luas, warna merah yang ada pada gambar query dan pada dataset berbeda namun masih bisa dibilang merah.

2. Pengujian Tekstur

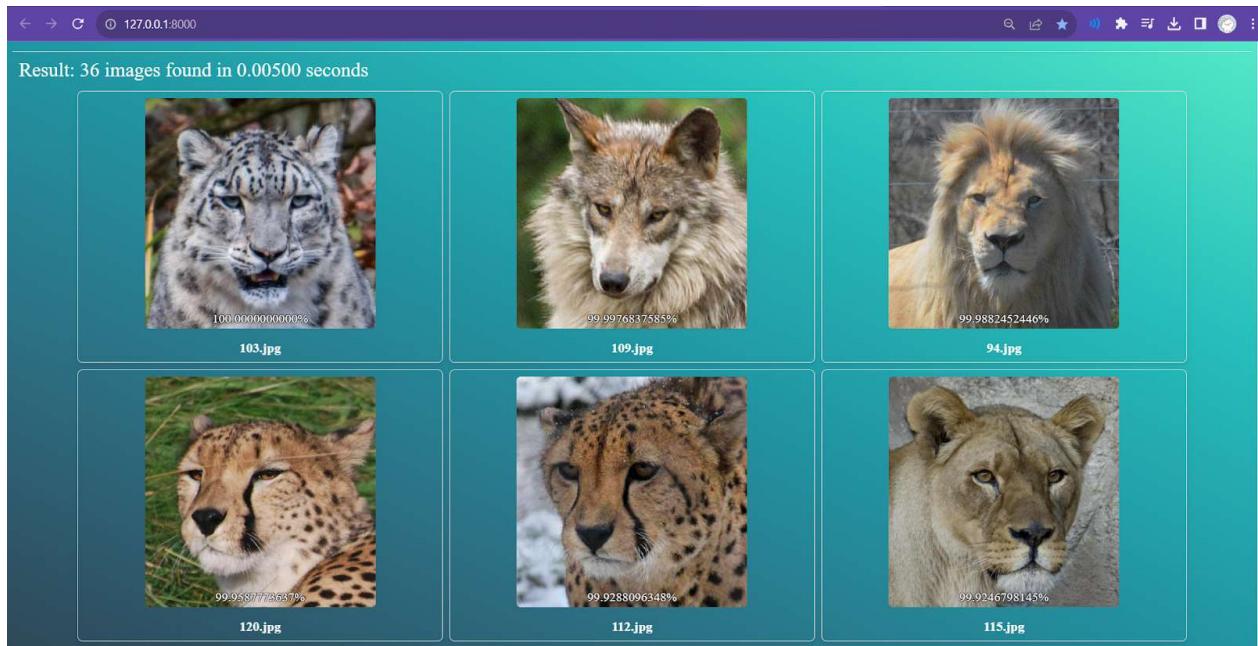
Pengujian dilakukan dengan gambar query adalah salah salah satu foto tes yang terdapat pada folder test, dan dataset yang digunakan adalah semua gambar dataset yang ada pada folder test.

a. Query

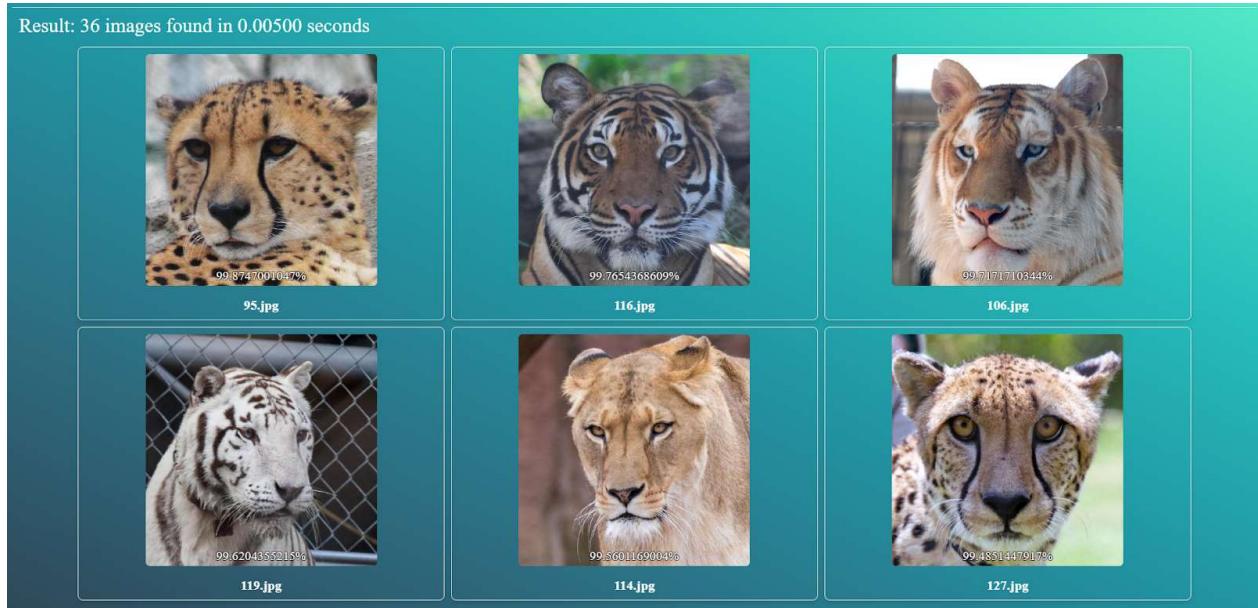


Gambar 5. Tampilan query metode tekstur.

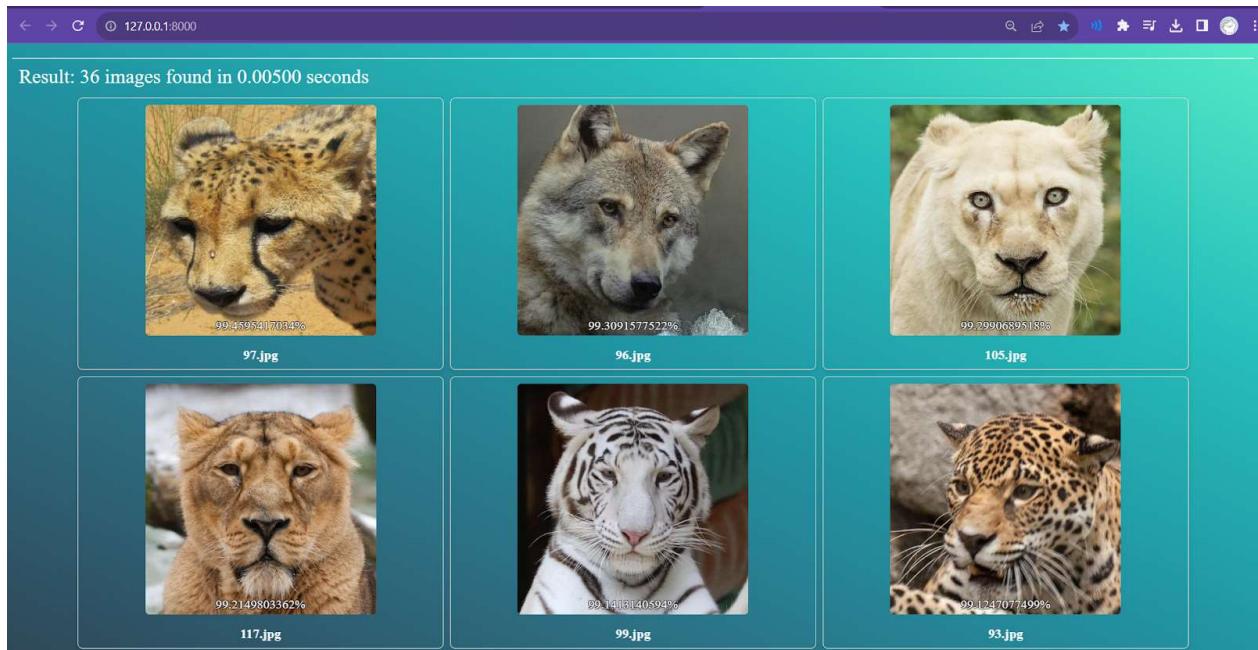
b. Hasil



Gambar 6. Tampilan hasil metode tekstur (1).



Gambar 7. Tampilan hasil metode tekstur (2).

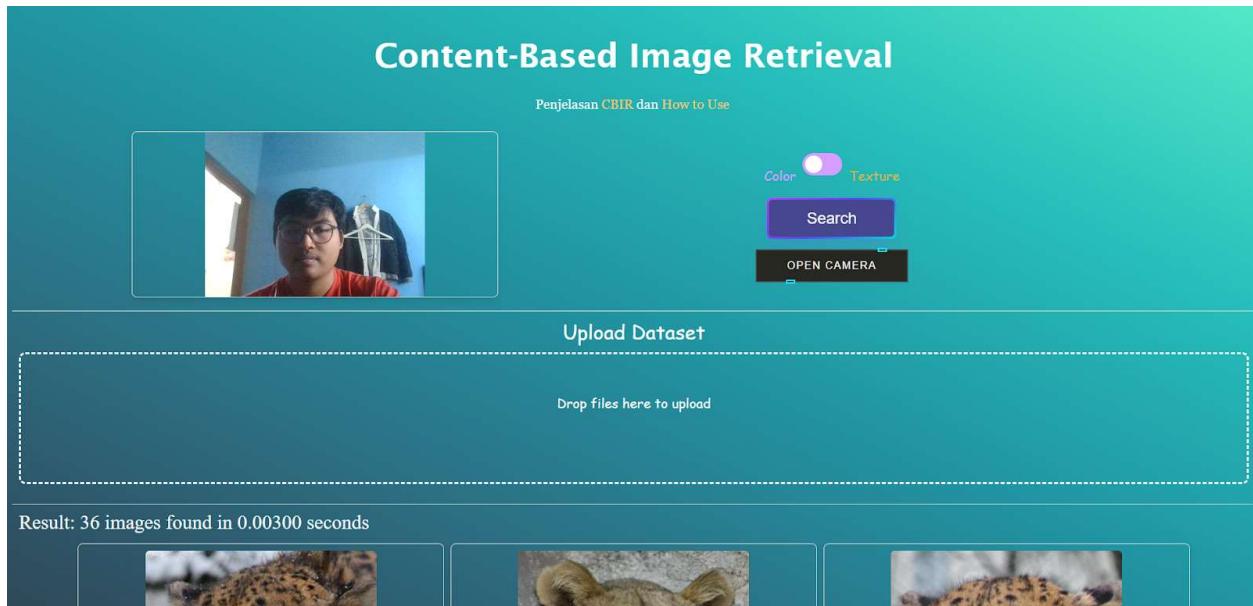


Gambar 8. Tampilan hasil metode tekstur (3).

3. Pengujian Kamera

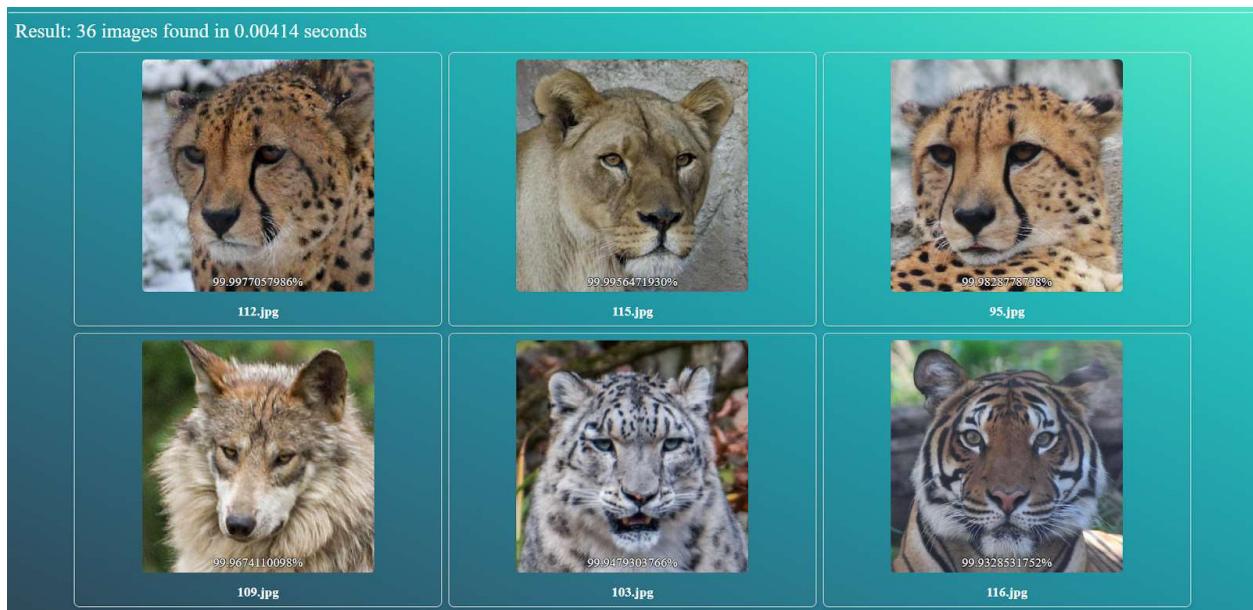
Pengujian dilakukan dengan metode tekstur, gambar query adalah gambar yang diambil dari kamera pengguna, dan dataset yang digunakan adalah semua gambar dataset yang ada pada folder test.

a. Query

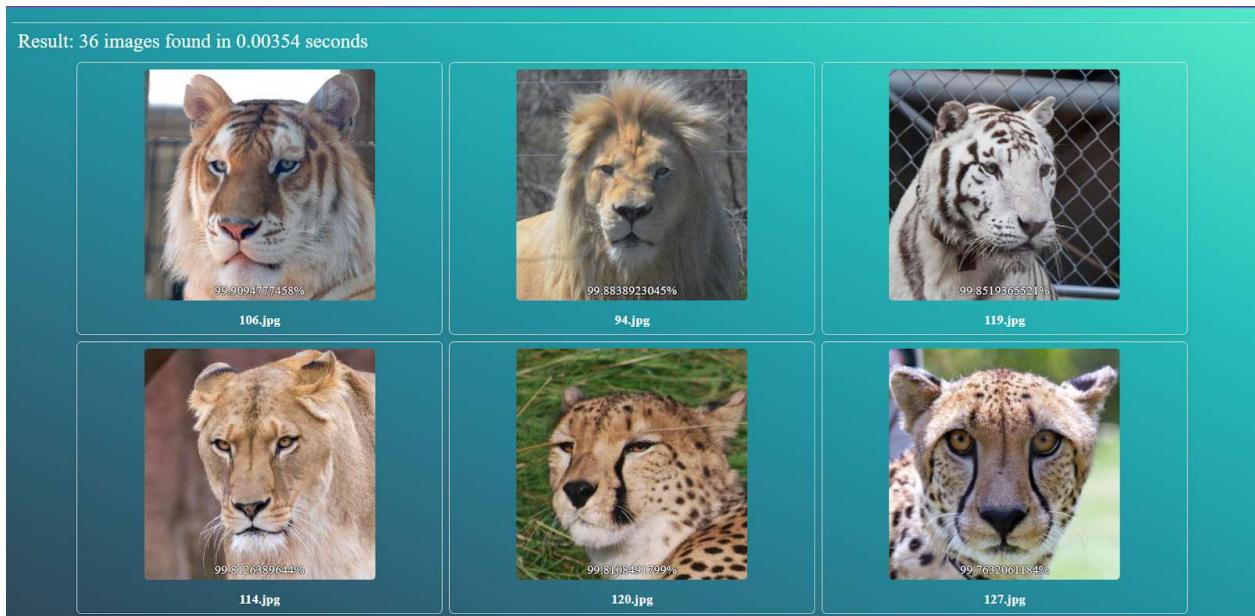


Gambar 9. Tampilan open camera.

b. Hasil



Gambar 10. Tampilan hasil dengan kamera (1).



Gambar 11. Tampilan hasil dengan kamera (2).

c. Penjelasan Tambahan

Karena foto yang diambil dan dijadikan query bukanlah gambar hewan, melainkan gambar seorang manusia di dalam kamar, maka seharusnya perhitungan similaritasnya cukup rendah. Namun karena metode yang digunakan adalah tekstur dan parameternya tidak begitu sempurna, maka hasil perhitungan similaritas akan menunjukkan nilai yang lumayan besar sehingga gambar hewan-hewan ini tertampilkan pada result.

E. Analisis dari Desain Solusi Algoritma pencarian yang diimplementasikan

CBIR menggunakan parameter warna memiliki rata-rata waktu eksekusi ± 0.2 detik untuk perbandingan 2 buah gambar dengan dimensi 512x512. Sedangkan untuk CBIR tekstur, rata-rata waktu eksekusi adalah ± 0.5 detik dengan kondisi yang sama. Tentu saja *execution time* tersebut tidak lepas dari kapabilitas dari masing-masing perangkat yang menjalankan program. Apabila dibandingkan, fitur warna akan lebih baik pada kasus-kasus yang mementingkan warna dibandingkan bentuk, misalnya pada test case kami ketika mencocokkan warna payung dengan database warna. Sedangkan, fitur tekstur akan lebih baik pada kasus-kasus yang mementingkan tekstur dan bentuk, misalnya saat mencocokkan gambar-gambar hewan. Penggunaan warna pada kasus gambar hewan akan kurang efisien karena bisa saja terdapat dua hewan yang berbeda tetapi dengan warna yang sejenis tetapi akan dianggap dua gambar yang mirip.

Pada CBIR parameter warna, pengolahan gambar dapat terjadi dengan sangat cepat berkat penggunaan library numpy. Dengan demikian, proses percabangan dalam

perhitungan hsv hingga pembuatan histogram tidak perlu menggunakan fungsi-fungsi dasar.

Untuk CBIR dengan tekstur, hasil yang didapat secara konsisten berkisar antara 98-99 %. Maka dari itu, kami menambahkan komponen tekstur di luar dari spesifikasi yaitu energy dan korelasi. Energy menghitung keseragaman dan keteraturan pixel, semakin tinggi energy nya maka semakin teratur pixel. Correlation mencari koefisien kesamaan untuk tiap pixel dengan suatu algoritma sehingga secara keseluruhan dapat membandingkan similarity dari 2 gambar. Kami memilih 2 komponen tambahan ini setelah membaca jurnal-jurnal dan referensi dan mengambil kesimpulan bahwa kedua parameter ini sering digunakan sebagai parameter tekstur untuk image processing. Namun, setelah menerapkan 2 parameter baru ini, sebagian besar hasil kemiripan tetap berada pada radius yang sama dengan sedikit sekali improvement dari sebelumnya.

Fungsi kamera yang diterapkan dalam website berjalan atau mengambil gambar setiap 5 detik. Tanpa perlu menekan tombol search, hasil pencocokan akan otomatis keluar pada bawah halaman.

Pada penerapannya, setelah mengunggah dataset pada laman website, kita mengunggah image query dan menekan tombol search. Lalu jika kita ingin menggunakan query image yang lain namun tetap ingin menggunakan dataset yang sama, kita tidak perlu mengunggah ulang dataset dan hanya perlu mengunggah query terbaru. Hal ini dikarenakan kami menggunakan *cache* yang menyimpan data dari dataset pada database sehingga dapat digunakan kembali. Penggunaan *cache* ini membuat web menjadi lebih efisien dan efektif ketika dataset ingin digunakan beberapa kali.

BAB V

PENUTUP

A. Kesimpulan

Kami telah berhasil membuat sebuah website untuk membandingkan gambar *query* dengan dataset sesuai input dari user. Website tersebut juga memiliki 2 metode pencarian atau teknik pemrosesan gambar yang berbeda yaitu pemrosesan dengan parameter warna dan tekstur yang dapat di-*toggle*. Hasil yang didapat dari tekstur sebagian besar dalam kisaran 90-99 % dalam kemiripan dikarenakan parameter atau komponen yang digunakan memanglah kurang sesuai. Waktu eksekusi kedua jenis CBIR bisa dibilang tergolong cepat dan website CBIR berjalan lancar dengan tidak adanya *bug*.

B. Saran

Saran untuk orang lain yang ingin mengerjakan projek seperti ini, gunakanlah library pembantu perhitungan seperti numpy karena sangat membantu dalam waktu eksekusi dan efisiensi program. Jika menggunakan algoritma dasar seperti nested loop, hasil yang dicapai kurang cepat.

C. Komentar

Tugas besar kali ini cukup menarik. Kami jadi bisa mencari dan belajar sendiri terkait topik ini dan bekerja bersama membangun proyek ini, mulai dari kode warna, kode tekstur, sampai websitenya. Topik yang diberikan sangat membantu semangat kami karena topiknya tidak membosankan.

D. Refleksi

Manajemen waktu dan prioritas sebaiknya ditingkatkan sehingga program dapat diselesaikan dari jauh-jauh hari.

E. Ruang Perbaikan atau Pengembangan

Masih ada banyak hal yang bisa kita kembangkan dan bahkan belum sempat kami eksplor. Salah satunya adalah web scraping, export hasil ke PDF, dan juga fungsionalitas object detector. Begitu pula dengan komponen pada tekstur, kita belum bisa menemukan komponen tekstur yang cocok agar hasil similarity menjadi sesuai.

DAFTAR PUSTAKA

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

https://www.sciencedirect.com/science/article/pii/S0895717710005352?ref=pdf_download&fr=RR-2&rr=8205463fef285ea7#s000030

https://eudl.eu/pdf/10.1007/978-3-642-35615-5_12

<https://sevima.com/tahapan-membuat-website-baru/>

LAMPIRAN

Link Repository Github: <https://github.com/TazakiN/Algeo02-22032>

Link Video Youtube: <https://youtu.be/AI44S7ate6I>