

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

TUGAS BESAR II
ALJABAR LINEAR DAN GEOMETRI
OCULAR REVERSE IMAGE SEARCHER

DISUSUN OLEH:

13522043 - Daniel Mulia Putra Manurung

13522084 - Dhafin Fawwaz Ikramullah

13522107 - Rayendra Althaf Taraka Noor



Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

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 mungkin kalian tahu adalah Google Lens.

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

BAB II

LANDASAN TEORI

1. CBIR

Content-Based Image Retrieval adalah penerapan teknik penglihatan komputer untuk masalah pemulihan gambar, yaitu, masalah pencarian gambar digital di database besar. Pengambilan gambar berbasis konten bertentangan dengan pendekatan berbasis konsep tradisional

CBIR dianggap sebagai salah satu metode terbaik untuk memperoleh data visual. Alih-alih menggunakan anotasi, ia menangani konten gambar secara langsung, termasuk aspek seperti warna, bentuk, dan struktur. Basis data teks-objek standar tidak dapat memenuhi kebutuhan database gambar, tetapi ditantang oleh volume data yang besar. Deskripsi otomatis dan efisien dari gambar tidak ada dalam metode konvensional anotasi gambar dengan teks. Sistem harus dapat memahami dan menafsirkan konten gambar yang dikelola untuk menerapkan CBIR. Indeks pencarian harus dihasilkan secara otomatis, memberikan pengguna antarmuka yang lebih visual untuk pencarian.

Kami akan menerapkan dua tipe CBIR, yaitu:

- CBIR dengan parameter warna
- CBIR dengan parameter tekstur

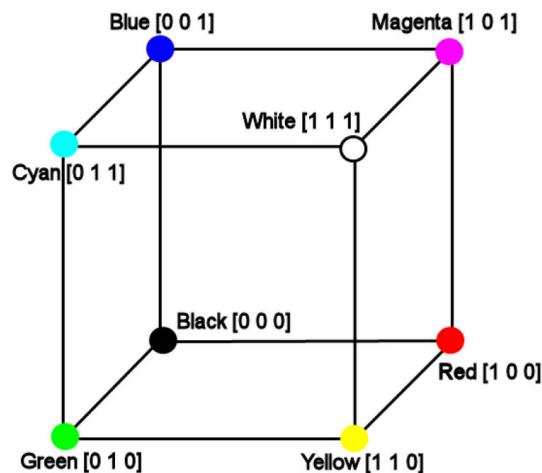
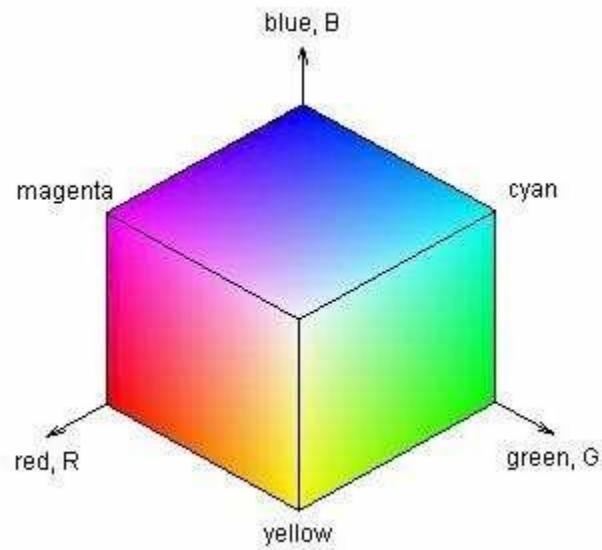
2. RGB

Model warna RGB adalah model warna aditif yang dapat melakukan reproduksi berbagai macam warna dengan menggabungkan warna dasar merah, hijau, dan biru dari cahaya dengan cara yang berbeda. Inisial dari tiga warna dasar tambahan merah, hijau, dan biru digunakan untuk membuat nama model.

Meskipun telah digunakan dalam fotografi tradisional, model warna RGB terutama digunakan untuk sensasi, representasi, dan tampilan gambar dalam sistem elektronik, seperti komputer dan televisi. Model warna RGB memiliki teori suara yang didasarkan pada bagaimana orang melihat warna sebelum penemuan elektronik.

Karena elemen warna dan tanggapan mereka terhadap tingkat merah, hijau, dan biru individu bervariasi dari produsen ke produsen, atau bahkan dalam perangkat yang sama dari waktu ke waktu, RGB adalah model warna yang tergantung pada perangkat. Perangkat yang berbeda akan mendeteksi atau mereproduksi nilai RGB yang diberikan secara berbeda. Oleh karena itu, tanpa manajemen warna, nilai RGB tidak mendefinisikan warna yang sama di seluruh perangkat.

Berikut adalah grafik RGB



3. HSL and HSV

HSL (*hue, saturation, lightness*) dan HSV (*hue, saturation, value*) adalah dua representasi alternatif dari model warna RGB yang dirancang pada tahun 1970-an oleh peneliti grafika komputer untuk lebih menyelaraskannya dengan cara membuat atribut penglihatan manusia yang memandang warna. Dalam model ini, warna dari masing-masing hue diatur dalam irisan radial, sekitar poros tengah dari warna-warna netral yang berkisar hitam di bagian bawah untuk di bagian putih atas. Dalam Tugas ini, akan menggunakan penerapan HSV, dan bukan HSL.

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

HSV memodelkan representasi dengan cara mencampurkan cat dari berbagai warna bersama-sama dengan saturasi dimensi menyerupai berbagai warna cerah cat berwarna dan nilai dimensi yang menyerupai campuran cat-cat dengan jumlah yang bervariasi dari cat hitam atau putih.

4. RGB to HSV Conversion

Dalam penerapan CBIR dengan parameter warna, kita akan mengonversi warna warna RGB ke HSV dikarenakan warna HSV dapat digunakan pada kertas dan lebih yang lebih umum untuk digunakan. Prosedur konversi RGB ke HSV adalah sebagai berikut.

a. Normalisasi RGB

Nilai RGB yang kita dapatkan akan dinormalisasi dengan range [0..1] dengan:

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

b. Mencari C_{max} , C_{min} , dan Δ

Ketiga hal ini didapatkan dengan:

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

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

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

c. Mendapatkan nilai HSV

Nilai HSV dapat didapatkan dengan nilai nilai yang telah kita cari di atas dengan:

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

5. Grayscale Conversion

Dalam bidang colorimetry, fotografi digital, dan gambar yang dihasilkan komputer, gambar skala abu-abu adalah yang di mana nilai masing-masing pixel adalah sampel tunggal yang hanya mewakili jumlah cahaya yang hadir - yaitu, ia berisi informasi tentang intensitas saja. Gambar dalam skala abu-abu, jenis monokrom hitam dan putih,

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

hanya terdiri dari warna abu. Pada yang terlemah, kontrasnya hitam; pada yang terkuat, itu putih.

Gambar dengan hanya dua warna – hitam dan putih – dalam konteks pencitraan komputer dikenal sebagai gambar bi-tonal hitam-putih satu-bit. Gambar berukuran abu-abu berbeda dengan gambar-gambar ini. Gambar-gambar dalam skala abu-abu berisi berbagai macam nada abu.

Pada CBIR dengan parameter tekstur, kita akan menggunakan grayscale, yang adalah hasil konversi RGB. Proses konversi RGB ke Grayscale adalah sebagai berikut:

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

6. Co - Occurrence Matrix

Matriks yang didefinisikan atas gambar sebagai distribusi nilai pixel yang terjadi bersama (nilai skala abu-abu, atau warna) pada offset yang diberikan disebut matriks atau distribusi co-occurrence. Ini juga dikenal sebagai matrix co-occurrence tingkat abu-abu, atau GLCM. Ini adalah metode untuk analisis tekstur dengan berbagai kegunaan, khususnya dalam analisis gambar medis.

Dengan gambar tingkat abu-abu I, matriks co-occurrence menghitung seberapa sering pasangan piksel dengan nilai spesifik dan offset terjadi dalam gambar. Metode pembentukan co-occurrence matrix adalah sebagai berikut:

- a. Offset, $\Delta x \Delta y$, adalah operator posisi yang dapat diterapkan pada setiap pixel dalam gambar (mengabaikan efek tepi): misalnya, (1, 2) dapat menunjukkan "satu ke bawah, dua ke kanan".
- b. Gambar dengan jumlah nilai pixel n yang berbeda akan menghasilkan matrix co-occurrence dengan dimensi $n \times n$.
- c. Nilai ke-(i,j) dari matriks co-occurrence memberikan jumlah kali dalam gambar bahwa nilai pixel ke-i dan ke-j muncul dalam hubungan yang diberikan oleh offset.

Gambar dengan jumlah nilai pixel p yang berbeda dengan matrix C dengan dimensi $p \times p$ didefinisikan dengan gambar I dengan ukuran $n \times m$, dengan parameter offset $\Delta x \Delta y$, direpresentasikan dengan:

$$C_{\Delta x, \Delta y}(i, j) = \sum_{x=1}^n \sum_{y=1}^m \begin{cases} 1, & \text{if } I(x, y) = i \text{ and } I(x + \Delta x, y + \Delta y) = j \\ 0, & \text{otherwise} \end{cases}$$

Dapat didapatkan juga matrix simetrik dari matrix co-occurrence dengan persamaan berikut:

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

7. Cosine Similarity

Sebuah ukuran kesamaan antara dua vektor non-zero yang didefinisikan dalam ruang produk internal dalam analisis data disebut kesamaan kosin. Kosin dari sudut antara vektor, atau produk titik dari vektor dibagi oleh produk panjangnya, adalah apa yang dikenal sebagai kesamaan kosin. Akibatnya, kesamaan kosin hanya tergantung pada sudut vektor dan bukan magnitudo nya. Interval $[-1, 1]$ selalu di mana kesamaan kosin jatuh.

Kesamaan cosine antara dua vektor proporsional adalah 1, yang dari dua vektor ortogonal adalah 0, dan yang dari 2 vektor yang berlawanan adalah -1. Situasi-situasi tertentu mencegah nilai komponen vektor menjadi negatif; cosine similarity terbatas pada $[0, 1]$.

Cosine dari dua vektor non-zero dapat dihasilkan dengan menggunakan rumus produk titik Euclidean:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$

Dengan vektor atribut n-dimensi, A dan B, kesamaan kosin, $\cos(\theta)$, diwakili dengan menggunakan produk titik dan magnitudo sebagai:

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

Dengan A_i dan B_i adalah komponen ke-i dari A dan B.

8. Pengembangan Website

Proses membuat situs web untuk intranet (jaringan pribadi) atau Internet (World Wide Web) dikenal sebagai pengembangan web atau biasa kita kenal dengan Web Development. Pengembangan web mencakup berbagai tugas, mulai dari membuat halaman statis dasar dengan teks hingga membuat aplikasi web yang rumit, perdagangan elektronik, dan platform media sosial. Web engineering, web design, pengembangan konten web, client liaison, client-side/server-side scripting, web server dan konfigurasi keamanan jaringan, dan pengembangan e-commerce adalah daftar yang lebih luas dari tugas yang sering dikaitkan dengan pengembangan web.

Tim pengembangan web untuk perusahaan dan organisasi yang lebih besar dapat mencakup ratusan developer web dan membuat situs web menggunakan teknik standar. Perusahaan yang lebih kecil mungkin hanya membutuhkan satu developer, baik secara kontrak atau permanen, atau mereka mungkin perlu mengalokasikan developer ke posisi

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

terkait seperti desain grafis atau teknisi sistem informasi. Alih-alih jatuh di bawah pengawasan satu departemen, pengembangan web mungkin hasil dari kolaborasi departemen. Spesialisasi web development terbagi menjadi tiga yaitu Front-End Developer, Back-End Developer, dan Full-Stack Developer.

a. Front-End Development

Front-End Development adalah pengembangan antarmuka pengguna grafis dari sebuah situs web, melalui penggunaan HTML, CSS, dan JavaScript, sehingga pengguna dapat melihat dan berinteraksi dengan situs tersebut.

b. Back-End Development

Jika front-end adalah apa yang dilihat pengguna, back-end itu apa yang tidak mereka lihat. Developer web back-end bekerja di server situs web, program, dan perangkat lunak untuk memastikan semuanya bekerja dengan benar di balik adegan. Developer ini bekerja dengan sistem seperti server, sistem operasi, API, dan database dan mengelola kode untuk keamanan, konten, dan arsitektur situs. Mereka bekerja sama dengan developer front-end untuk membawa produk mereka ke pengguna.

BAB III

ANALISIS PEMECAHAN MASALAH

1. Langkah Pemecahan Masalah

Salah satu solusi untuk melakukan pencarian gambar dalam jumlah yang banyak adalah dengan memanfaatkan kekuatan komputasi dari sebuah komputer dan konsep Aljabar Vektor. Pada sebuah gambar terdapat informasi-informasi unik yang akan membedakan antara gambar yang satu dengan yang lain. Misalnya adalah warna dan teksturnya. Untuk warna, kita dapat membandingkan histogram warna dari gambar yang ingin dicari dengan histogram warna dari masing-masing dataset. Sedangkan untuk tekstur, kita dapat membandingkan komponen-komponen seperti *Contrast*, *Homogeneity*, *Entropy*, dan sebagainya di antara gambar yang ingin dicari dengan gambar pada masing-masing dataset.

2. Pemetaan Masalah

Untuk menghasilkan vektor - vektor yang nantinya digunakan untuk menghitung cosine similarity, matriks gambar akan diolah sesuai dengan parameter yg digunakan sebagai berikut.

a. Search by Color

Pada parameter warna, matriks citra yang awalnya direpresentasikan dalam bentuk RGB akan diubah menjadi HSV dengan cara yang sudah dijelaskan dalam bab 2. Kemudian, akan dibentuk array berukuran 72 yang nilainya akan sesuai dengan banyaknya HSV berada pada rentang tertentu. Terakhir, 72 array ini akan digunakan sebagai vektor-vektor yang dimiliki sebuah citra untuk dihitung kemiripannya dengan citra lain dengan cosine similarity.

Untuk meningkatkan akurasi, cara yang sama akan diterapkan dengan membagi gambar menjadi 4×4 blok berdasarkan ukuran pixelnya, setiap blok dari gambar tersebut akan dikonversi menjadi HSV dan memiliki histogramnya masing - masing. Dipilih 4×4 blok dikarenakan jika blok yang diambil terlalu banyak, maka akan sangat memperlama perhitungan dan menghasilkan hasil pool gambar yang tidak begitu luas, dan jika bloknya terlalu sedikit akan menghasilkan pool gambar yang tidak begitu akurat. Dikarenakan adanya $4 \times 4 = 16$ blok, maka panjang histogram tiap gambar adalah $16 \times 72 = 1152$. Untuk setiap blok dengan panjang 72, histogram akan dibandingkan (blok 1 - 72, 73 - 144, dst) dengan cosine similarity. Cosine similarity akhir adalah rata- rata dari setiap cosine similarity yang telah didapatkan. Hal ini meningkatkan akurasi pendapatan gambar dikarenakan akan lebih memperhatikan letak dari warna - warna pada gambar.

b. Search by Texture

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Pada parameter tekstur, matriks citra yang berbentuk RGB akan dikonversi ke bentuk grayscale dengan cara yang sudah ditunjukkan pada bab 2. Selanjutnya akan dibentuk matriks berukuran 256×256 yang memiliki nilai sesuai dengan co-occurrence matrix. Setelah itu, dengan matriks ini dikalkulasikan nilai-nilai yang komponen tekstur yakni, *contrast*, *dissimilarity*, *homogeneity*, *ASM*, *entropy*, *energy*.

Pada *search by texture* yang kami buat, co-occurrence matrix yang kami gunakan adalah co-occurrence dengan nilai theta 0 derajat. Hal ini dilakukan sehingga gambar-gambar terhitung mirip adalah gambar-gambar yang memiliki tekstur yang mirip dan dengan orientasi yang sama.

c. Caching

Jika gambar sudah pernah dicari, maka akan menjadi efektif jika perhitungan untuk pencarian gambar tersebut tidak dihitung ulang. Pada setiap kali melakukan search, hash berdasarkan Isi dari gambar yang dicari dan parameter search nya akan dikalkulasi terlebih dahulu. Lalu akan dimulai kalkulasi berdasarkan color atau texture. Url hasil pencarian tersebut yang akan disimpan ke database dan dikaitkan dengan hash tadi. Lalu jika user akan mencari gambar yang sama dan dengan parameter search yang sama pula, maka akan dihasilkan hash yang sama dengan sebelumnya. Lalu diterapkan database indexing untuk field hash tersebut sehingga pencarian berdasarkan hash yang sama akan menjadi cepat (Codecademy Team, n.d.). Hasil pencarian berdasarkan hash yang sama tersebut akan menjadi hasil pencarian yang dicari oleh user. Tentunya jika dataset baru diupload, semua cache ini harus dihapus.

d. Precomputed Dataset

Saat melakukan upload dataset, akan langsung dilakukan perhitungan untuk mendapatkan vektor untuk histogram warna dan vektor untuk komponen tekstur. Dengan cara ini, saat melakukan search, yang perlu dilakukan perhitungan untuk histogram warna dan komponen textur hanyalah untuk gambar yang sedang dicari saja. Sisanya tinggal perhitungan untuk Cosine Similarity antara gambar yang dicari dengan semua gambar pada dataset.

e. Pagination

Saat gambar selesai dicari, dari sisi user akan sulit jika ratusan gambar semuanya ditampilkan. Dengan memanfaatkan pagination, kita dapat menampilkan hanya beberapa gambar saja untuk gambar yang ditemukan.

f. Multiprocessing

Saat melakukan upload dataset dalam jumlah yang besar, akan memakan waktu yang lama. Padahal device yang kita gunakan sebenarnya masih sanggup untuk melakukan banyak pekerjaan lain. Dengan memanfaatkan multiprocessing, kita dapat melakukan banyak perhitungan histogram warna dan komponen tekstur

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

secara parallel atau dilakukan secara bersamaan dalam satu waktu dibanding yang awalnya satu per satu.

g. Web Scrapping

Akan sangat mempermudah jika kita dapat mengambil semua gambar yang terdapat pada sebuah website secara otomatis dan dijadikan dataset baru. Pada pengembangan website terdapat berbagai cara untuk menampilkan gambar. Yang dihandle di sini adalah source dari gambar yang ditampilkan dengan tag img pada html, gambar dengan encoding base64 pada tag script pada html, dan juga gambar yang ditampilkan dengan memanfaatkan styling dengan properti css background-image dengan value berupa url dari gambar.

h. RealTime Webcam

Pemilihan gambar untuk melakukan search juga dapat dilakukan dengan memanfaatkan realtime webcam dari device masing-masing. Dengan mengaktifkan hal tersebut, akan dilakukan penangkapan gambar dengan kamera setiap 5 detik, dan langsung digunakan untuk pencarian gambar.

i. Export PDF

Hasil pencarian gambar dapat diexport juga ke pdf.

3. Contoh Ilustrasi Kasus dan Penyelesaiannya

Adanya mesin pencarian gambar dapat memudahkan kita dalam berbagai hal. Berikut beberapa contoh kasus penggunaan mesin pencarian gambar dalam kehidupan sehari-hari.

a. Pencarian Barang Menggunakan E-Commerce

Dengan pesatnya perkembangan teknologi pada zaman sekarang, kita seringkali menemukan barang-barang menarik yang kita lihat di media sosial atau situs-situs lainnya di internet. Namun, sering juga kita menemukan barang yang kita tidak ketahui namanya dan tidak ada penjelasan mengenai barang yang kita lihat. Dan dengan adanya *image searching* kita bisa menemukan berbagai macam barang tersebut dengan sumber data gambar dari e-commerce. Hal ini bisa meningkatkan pembelian di e-commerce dan memudahkan kita dalam mencari barang yang dibutuhkan.

b. Face Recognition

Karena image searching melihat kemiripan gambar, ketika sudah cukup akurat, pemrosesan ini bisa digunakan juga untuk mengenali wajah. Pengenalan wajah ini bisa dimanfaatkan dalam berbagai kondisi, diantaranya membuka kunci perangkat elektronik dan pengelompokan foto.

BAB IV

IMPLEMENTASI DAN UJI COBA

1. Implementasi Program

SearchByColor.c

```
function normalizedRGB(n)
-> n / 255.0

function getCmax(r, g, b)
-> max(max(r, g), b)

function getCmin(r, g, b)
-> min(min(r, g), b)

function getCmaxLoc(r, g, b)
maxVal <- getCmax(r, g, b)

if maxVal = r then
-> 0
else if maxVal = g then
-> 1
else if maxVal = b then
-> 2

function getDelta(r, g, b)
normalizedR <- normalizedRGB(r)
normalizedG <- normalizedRGB(g)
normalizedB <- normalizedRGB(b)

maxVal <- getCmax(normalizedR, normalizedG, normalizedB)
minVal <- getCmin(normalizedR, normalizedG, normalizedB)

-> maxVal - minVal

function getHValue(R, G, B)
r <- toDouble(R)
g <- toDouble(G)
b <- toDouble(B)

if getDelta(r, g, b) = 0 then
degVal <- 0
else if getCmaxLoc(r, g, b) = 0 then
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
    degVal <- 60 * mod(((normalizedRGB(g) -  
normalizedRGB(b)) / getDelta(r, g, b)), 6)  
    else if getCmaxLoc(r, g, b) = 1 then  
        degVal <- 60 * (((normalizedRGB(b) - normalizedRGB(r))  
/ getDelta(r, g, b)) + 2)  
    else if getCmaxLoc(r, g, b) = 2 then  
        degVal <- 60 * (((normalizedRGB(r) - normalizedRGB(g))  
/ getDelta(r, g, b)) + 4)  
  
    if degVal < 0 then  
        degVal += 360  
  
    if (degVal > 315 and degVal <= 360) or degVal = 0 then  
        -> 0  
    else if degVal > 0 and degVal <= 25 then  
        -> 1  
    else if degVal > 25 and degVal <= 40 then  
        -> 2  
    else if degVal > 40 and degVal <= 120 then  
        -> 3  
    else if degVal > 120 and degVal <= 190 then  
        -> 4  
    else if degVal > 190 and degVal <= 270 then  
        -> 5  
    else if degVal > 270 and degVal <= 295 then  
        -> 6  
    else if degVal > 295 and degVal <= 315 then  
        -> 7  
    else  
        print(degVal)  
        print("Hgagal")  
  
function getSValue(R, G, B)  
    r <- toDouble(R)  
    g <- toDouble(G)  
    b <- toDouble(B)  
  
    if getCmax(normalizedRGB(r), normalizedRGB(g),  
normalizedRGB(b)) = 0 then  
        tempVal <- 0  
    else  
        tempVal <- (getDelta(r, g, b) /  
getCmax(normalizedRGB(r), normalizedRGB(g), normalizedRGB(b)))
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
if tempVal >= 0 and tempVal < 0.2 then
    -> 0
else if tempVal >= 0.2 and tempVal < 0.7 then
    -> 1
else if tempVal >= 0.7 and tempVal <= 1 then
    -> 2
else
    print("Sgagal")

function getValue(R, G, B)
    r <- toDouble(R)
    g <- toDouble(G)
    b <- toDouble(B)
    tempVal <- getCmax(normalizedRGB(r), normalizedRGB(g),
normalizedRGB(b))

    if tempVal >= 0 and tempVal < 0.2 then
        -> 0
    else if tempVal >= 0.2 and tempVal < 0.7 then
        -> 1
    else if tempVal >= 0.7 and tempVal <= 1 then
        -> 2
    else
        print(tempVal, "\n")
        print("Vgagal\n")

function getColorHistogram(matrix, row, col)
    colorHistogram <- malloc(sizeof(int) * HSV_HIST_SIZE_FULL +
5)

    if colorHistogram = NULL then
        print("gagal malloc\n")

    for i from 0 to HSV_HIST_SIZE_FULL - 1
        colorHistogram[i] <- 0

    rowcvrt <- toDouble(row)
    colcvrt <- toDouble(col)
    rowidx <- floor(rowcvrt / 4)
    colidx <- floor(colcvrt / 4)

    m <- 0
    n <- 0
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
for m from 0 to 3
    for n from 0 to 3
        for k from rowidx * m to rowidx * (m + 1) - 1
            for j from colidx * n to colidx * (n + 1) - 1
                R <- matrix[k * col * 3 + j * 3]
                G <- matrix[k * col * 3 + j * 3 + 1]
                B <- matrix[k * col * 3 + j * 3 + 2]

                H <- getHValue(R, G, B)
                S <- getSValue(R, G, B)
                V <- getVValue(R, G, B)

                idx <- foridx * HSV_HIST_SIZE + H * 9 + S *
3 + V
                colorHistogram[idx] += 1
            foridx++
        -> colorHistogram

function free_ptr(arr)
    free(arr)
```

SearchByTexture.c

```
function getGrayS(r, g, b)
    -> round(0.29 * r + 0.587 * g + 0.114 * b)

function toGLCM(fromPicture, height, width)
    for i from 0 to height - 1
        for j from 0 to width - 2
            cidx1 <- i * width * 3 + j * 3
            cidx2 <- i * width * 3 + (j + 1) * 3
            val1 <- getGrayS(fromPicture[cidx1],
fromPicture[cidx1 + 1], fromPicture[cidx1 + 2])
            val2 <- getGrayS(fromPicture[cidx2],
fromPicture[cidx2 + 1], fromPicture[cidx2 + 2])
            Res[val1][val2] += 1
            Res[val2][val1] += 1

function generateTexture(sumElmt, contrast, dissimilarity,
homogeneity, ASM, entropy, energy, textureComponent, pictHeight,
pictWidth)
    for i from 0 to GLCM_SIZE - 1
        for j from 0 to GLCM_SIZE - 1
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
Res[i][j] /= 2 * sumElmt
contrast += Res[i][j] * (i - j) * (i - j)
dissimilarity += Res[i][j] * abs(i - j)
homogeneity += Res[i][j] / (1 + (i - j) * (i - j))
ASM += Res[i][j] * Res[i][j]
if Res[i][j] != 0
    entropy += Res[i][j] * log(Res[i][j])

entropy <- -entropy
energy <- sqrt(ASM)
textureComponent[0] <- contrast
textureComponent[1] <- dissimilarity
textureComponent[2] <- homogeneity
textureComponent[3] <- ASM
textureComponent[4] <- entropy
textureComponent[5] <- energy

function getTextureComponents(fromPicture, pictHeight,
pictWidth)
    textureComponents <- malloc(sizeof(double) *
COSINE_SIMILARITY_VECTOR_SIZE)
    toGLCM(fromPicture, pictHeight, pictWidth)
    generateTexture(pictHeight * (pictWidth - 1), 0, 0, 0, 0, 0,
0, textureComponents, pictHeight, pictWidth)
    -> textureComponents

function free_ptr(arr)
    free(arr)
```

calcSimilarity.c

```
function cosineSimilarity(A, B, vectorSize)
    dotProduct <- 0.0
    normA <- 0.0
    normB <- 0.0

    for i from 0 to vectorSize - 1
        dotProduct += A[i] * B[i]
        normA += A[i] * A[i]
        normB += B[i] * B[i]

    -> dotProduct / (sqrt(normA) * sqrt(normB))
function cosineSimilarityColor(A, B, vectorSize)
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
c <- malloc(sizeof(double) * 16)
dotProduct <- 0.0
normA <- 0.0
normB <- 0.0
total <- 0.0

for j from 0 to 15
    dotProduct <- 0.0
    normA <- 0.0
    normB <- 0.0
    for i from 0 to vectorSize - 1
        k <- j * 72
        dotProduct += toDouble(A[k + i]) * toDouble(B[k +
i])
        normA += toDouble(A[k + i]) * toDouble(A[k + i])
        normB += toDouble(B[k + i]) * toDouble(B[k + i])

    c[j] = dotProduct / (sqrt(normA) * sqrt(normB))
    if c[j] < 0 then
        c[j] <- -c[j]

for i from 0 to 15
    total += c[i]

-> total / 16.0
```

searcher.py

```
class Searcher:
{ Untuk urusan search image dan pdf }
    function generateHash(image_file: InMemoryUploadedFile,
search_type: str) -> str:
        image_content -> image_file.read()
        hash_input ->
f"{image_content}{search_type}".encode('utf-8')
        sha256_hash -> hashlib.sha256(hash_input).hexdigest()
        print("Hash generated:", sha256_hash)
        -> sha256_hash

    function searchByColorMultiprocess(dataset: DataSet, hash:
str, color_histogram: list[int]) -> SearchResult:
        color_histogram_c ->
```

```
color_histogram.ctypes.data_as(POINTER(c_int))
    sr -> SearchResult()
    sr.image_url -> dataset.image_request.url
    sr.hash -> hash

    color_histogram_res -> dataset.color_histogram
    color_histogram_res_v2 -> np.array(color_histogram_res,
dtype->np.int32)
    color_histogram_res_c ->
color_histogram_res_v2.ctypes.data_as(POINTER(c_int))

    similarity ->
ImageProcessing.cos_sim.cosineSimilarityColor(color_histogram_c,
color_histogram_res_c, 72)
    sr.similarity -> similarity
    -> sr

function getSearchResult(data: SearchRequest) ->
Tuple[List[SearchResult],bool]:
    if
SearchRequest.objects.filter(hash->data.hash).exists():
        search_result ->
SearchResult.objects.filter(hash->data.hash)
        -> (search_result,True)

    result: list[SearchResult] -> []
    if data.search_type = "0": { by texture }
        { Texture search logic }
    else: { by color }
        { Color search logic }

    result.sort(key->lambdax: x.similarity, reverse->True)
    -> (result,False)

function getPDFfromImageUrls(search_result_list:
list[SearchResult]) -> ContentFile:
    { PDF creation logic }
    -> pdf_content
```

uploader.py

```
class Uploader:
{ Untuk urusan upload dataset dan web scrapping }
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
function bulk_created_task_multiprocess(dataset: DataSet) ->
DataSet:
    image <- dataset.image_request
    img_tmp <- Image.open(image)
    img_matrix <- img_tmp.convert('RGB')
    img_matrix <- np.array(img_matrix, dtype = np.int32)
    c_img_matrix <-
img_matrix.ctypes.data_as(POINTER(c_int))

    row <- len(img_matrix)
    col <- len(img_matrix[0])

        c_texture_components_ptr <-
ImageProcessing.by_texture.getTextureComponents(c_img_matrix,
row, col)
        texture_components <-
np.ctypeslib.as_array(c_texture_components_ptr, shape=(6,))

        c_color_histogram_ptr <-
ImageProcessing.by_color.getColorHistogram(c_img_matrix, row,
col)
        color_histogram <-
np.ctypeslib.as_array(c_color_histogram_ptr, shape=(1152,))

        dataset.texture_components <-
texture_components.tolist()
        dataset.color_histogram <- color_histogram.tolist()
        dataset.save(update_fields=['texture_components',
'color_histogram'])

    -> dataset

function task_multiprocess(image_file: InMemoryUploadedFile,
image_name: str):
    image_byte <- BytesIO(image_file)
    img_tmp <- Image.open(image_byte)
    img_matrix <- img_tmp.convert('RGB')
    img_matrix <- np.array(img_matrix, dtype=np.int32)
    c_img_matrix <-
img_matrix.ctypes.data_as(POINTER(c_int))

    row <- len(img_matrix)
    col <- len(img_matrix[0])
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
c_texture_components_ptr <-
ImageProcessing.by_texture.getTextureComponents(c_img_matrix,
row, col)
    texture_components <-
np.ctypeslib.as_array(c_texture_components_ptr, shape=(6,))

    c_color_histogram_ptr <-
ImageProcessing.by_color.getColorHistogram(c_img_matrix, row,
col)
        color_histogram <-
np.ctypeslib.as_array(c_color_histogram_ptr, shape=(1152,))

    img_to_save <- InMemoryUploadedFile(
        image_byte,
        None,
        image_name,
        'image/jpeg',
        getsizeof(image_byte),
        None
    )

    DataSet.objects.create(image_request=img_to_save
        texture_components = texture_components.tolist(),
color_histogram=color_histogram.tolist())

function saveImages(image_list: list[InMemoryUploadedFile]):
    { Multiprocessing logic }

function scrapImages(web_url: str) -> (tuple[list[DataSet],
int]):
    { Web scraping logic }

function download_image(url):
    { Image download logic }

function get_all_image_url(url):
    { Get all image URLs from a web page }

function delete_all_request_result():
    { Delete all request results }
```

views.py

```
class SearchRequestAPIView(APIView):
{ Route: /api/search }
    permission_classes <- [permissions.AllowAny]
    parser_classes <- [MultiPartParser, FormParser]

    method post(request):
        { Extract data from the request }
        { Create a SearchRequest instance }
        { Generate hash and set other attributes }
        { Check if hash exists in the cache }
        { If not, save search results and request }
        { Serialize the data and send the response }

    method get(request):
        { Retrieve all SearchRequest instances }
        { Optionally filter by hash }
        { Optionally reverse the order }
        { Optionally limit the results }
        { Serialize the data and send the response }

class SearchResultAPIView(APIView):
{ Route: /api/cache }
    permission_classes <- [permissions.AllowAny]

    method get(request):
        { Retrieve all SearchResult instances }
        { Optionally filter by hash }
        { Optionally reverse the order }
        { Optionally limit the results }
        { Serialize the data and send the response }

class UploadDatasetAPIView(APIView):
{ Route: /api/upload/dataset }
    permission_classes <- [permissions.AllowAny]

    method get(request):
        { Retrieve all DataSet instances }
        { Optionally reverse the order }
        { Optionally limit the results }
        { Serialize the data and send the response }

    method post(request):
```

```
{ Check if overwriting or appending dataset }
{ Delete all existing SearchResult instances }
{ Save images and handle dataset creation }

class ScrapDatasetAPIView(APIView):
{ Route: /api/upload/scrap }
    permission_classes <- [permissions.AllowAny]

    method post(request):
        { Extract web URL from the request }
        { Scrape images from the web URL }
        { Generate a response with image URLs }

class PDFAPIView(APIView):
{ Route: /api/pdf }
    permission_classes <- [permissions.AllowAny]
    parser_classes <- [MultiPartParser, JSONParser]

    method post(request):
        { Extract hash from the request }
        { Check if PDF already exists for the hash }
        { If not, generate PDF from image URLs }
        { Update SearchResult with the PDF file }
        { Generate a response with the PDF URL }
```

2. Struktur Program

Proyek ini memanfaatkan 2 framework utama yaitu Next.js untuk frontend dan Django Rest Framework untuk Backend.

a. Frontend

Struktur folder untuk frontend secara garis besar adalah sebagai berikut.

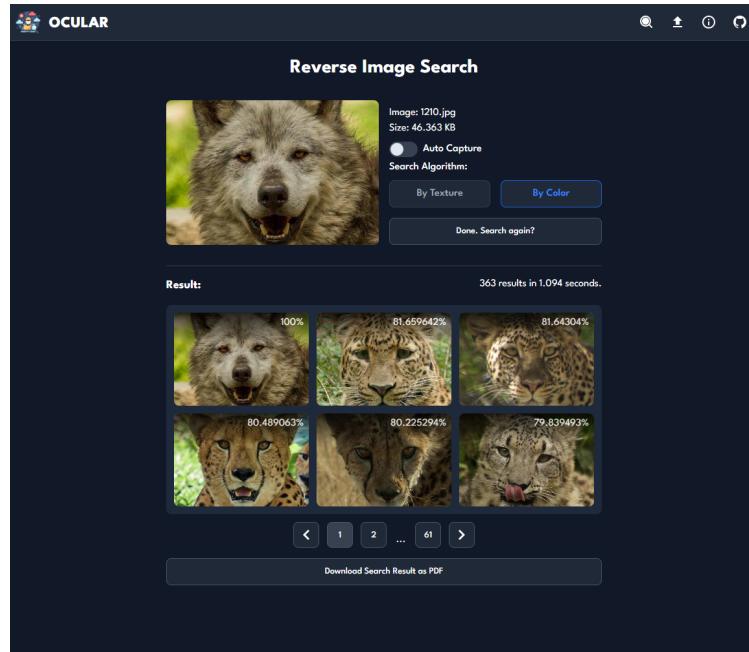
Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

The screenshot shows the file structure of a Next.js application. On the left, the 'src' directory contains 'client', '.next', 'app', 'component', and 'json' folders. 'client' has 'about' and 'uploaddataset' subfolders. 'app' has 'page.tsx', 'favicon.ico', 'globals.css', 'layout.tsx', 'page.tsx', 'certificates', and 'component' subfolders. 'component' has 'loadingskeleton.tsx'. 'json' has 'about.json'. On the right, there's a list of files: 'about.json', 'node_modules', 'public' folder containing 'althaf.jpg', 'daniel.jpg', 'dhafin.jpg', and 'icon.png', '.eslintrc.json', 'global.d.ts', 'next-env.d.ts', 'next.config.js', 'package-lock.json', 'package.json', 'postcss.config.js', 'tailwind.config.ts', 'tsconfig.json', and a 'server' folder.

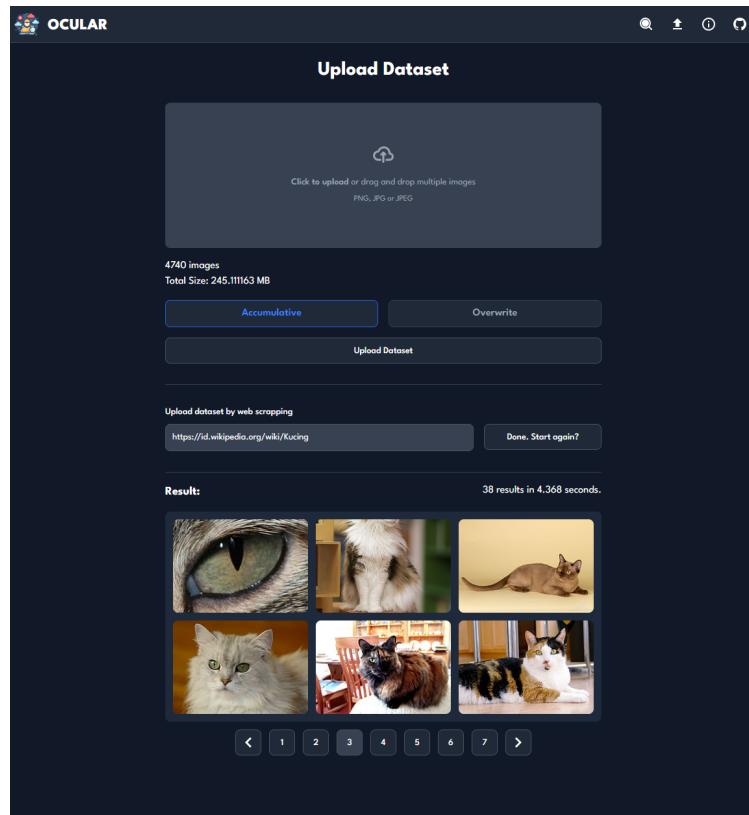
Terdapat beberapa file konfigurasi lain, tetapi yang perlu diperhatikan di sini adalah folder app, component, json, dan public. Folder component berisi component React yang dapat direuse. Folder json berisi data untuk halaman about untuk mempermudah pengubahan penjelasan halaman about. Folder public berisi gambar yang ditampilkan pada website. Sedangkan folder app merupakan halaman-halaman yang akan ditampilkan ke pengguna yaitu halaman search, upload dataset dan about. Berikut ini adalah tampilan dari ketiga halaman tersebut.

Halaman search:

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Terdapat sedikit inisiatif perubahan pada spesifikasi yaitu pada toggle antara color dan tekstur. Hal ini lebih sesuai dari sisi user experience (Anthony, 2019).
Halaman upload dataset:



Halaman about:

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

The screenshot shows the OCULAR web application interface. At the top, there is a navigation bar with icons for search, upload, and other functions. The main title is "Konsep Image Searching". Below the title, there is a descriptive text about Content-Based Image Retrieval (CBIR) and its implementation. A section titled "How to Use" contains four sub-sections: "Pencarian gambar relevan dengan upload gambar", "Upload dataset", "Pencarian gambar relevan dengan kamera", and "Web Scrapping". Each sub-section has a list of instructions. At the bottom, there is a section titled "Meet the Creators" featuring three team members with their names and student IDs.

Pada web ini, digunakan konsep Content-Based Image Retrieval (CBIR). Dengan dasar konsep ini kami mengolah sebuah citra sehingga didapatkan citra-citra yang memiliki kemiripan dengan citra awal. Untuk parameter yang digunakan ada 2 pilihan, yakni warna dan tekstur. Untuk parameter warna, gambar akan diubah menjadi ke bentuk HSV lalu dihitung color histogramnya. Sedangkan untuk parameter tekstur, gambar akan diubah ke bentuk grayscale lalu dihitung nilai komponen-komponen teksturnya. Setelah digunakan salah satu dari parameter tersebut, kemiripan gambar akan dikalkulasikan menggunakan cosine similarity.

How to Use

Pencarian gambar relevan dengan upload gambar

- Pilih laman searching
- Upload sebuah gambar pada kolom yang disediakan
- Pilih parameter pencarian
- Tekan tombol "Search Image"

Upload dataset

- Pilih laman upload
- Upload folder berisi gambar pada kolom yang disediakan
- Bisa juga dengan drag & drop beberapa gambar pada kolom tersebut
- Pilih opsi upload
- Tekan tombol "Upload Dataset"

Pencarian gambar relevan dengan kamera

- Pilih laman searching
- Nyalakan toggle "Auto Capture"
- Pilih allow untuk akses camera
- Gambar akan dimuat setiap 5 detik

Web Scrapping

- Pilih laman upload
- Masukkan alamat web pada kolom yang disediakan
- Tekan tombol "Start Scrapping"

Meet the Creators



Daniel Mulia Putra M.
13522043



Dhafin Fawwaz Ikramullah
13522084

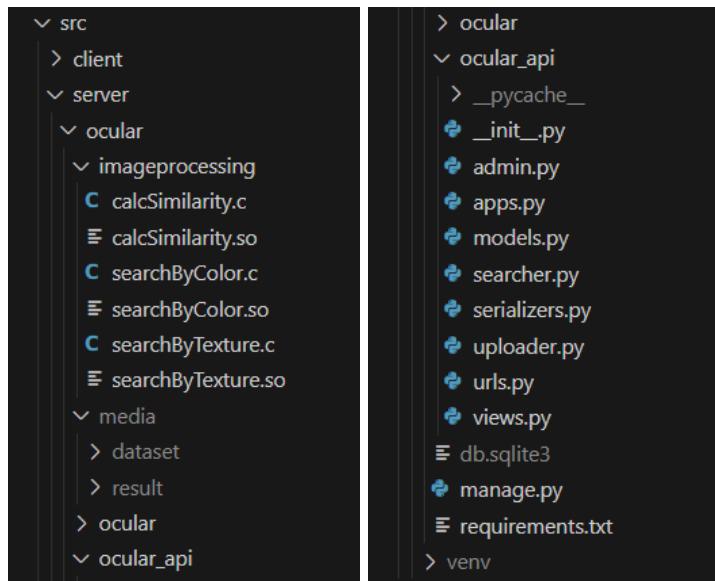


Rayendra Althaf Taraka N.
13522107

b. Backend

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

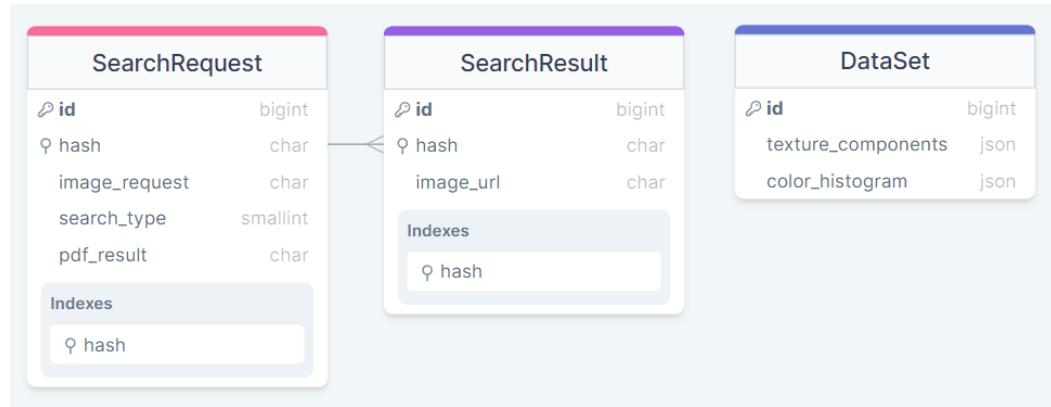
Struktur folder untuk backend secara garis besar adalah sebagai berikut.



Terdapat beberapa file konfigurasi lain, tetapi yang perlu diperhatikan di sini adalah folder imageprocessing, media, ocular_api, dan venv. Folder venv merupakan folder virtual environment yang isinya adalah dependencies dari proyek ini yang harus diinstal pengembang backend untuk menjalankan server. Folder media berisi folder dataset yang diupload beserta hasil pdf dari kumpulan gambar yang ditemukan saat melakukan pencarian. Folder imageprocessing berisi code dalam bahasa c dan hasil compilennya dalam format .so untuk melakukan berbagai perhitungan berat. Folder ocular_api merupakan folder utama dari program untuk menjalankan backend. Isinya terdapat __init__.py untuk inisialisasi library .so pada folder imageprocessing, models.py yang berisi model dari database, seerializer.py untuk serialization class pada models.py menjadi json, urls untuk definisi route yang ada, searcher.py untuk segala perhitungan dalam melakukan search yang juga akan memanfaatkan file .so pada folder imageprocessing, uploader.py untuk perhitungan dataset yang juga memanfaatkan file pada folder imageprocessing, dan views.py untuk menerima berbagai request seperti GET dan POST dari frontend.

Database yang digunakan memanfaatkan SQLite dengan model sebagai berikut:

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
 Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



API routes yang tersedia adalah sebagai berikut:

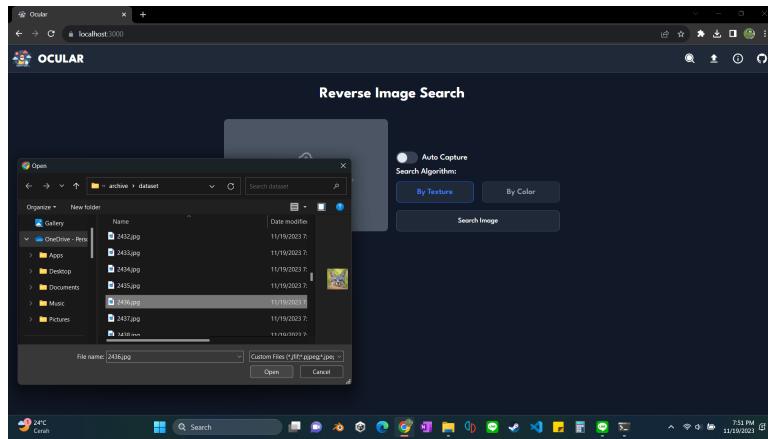
No	Route	Method	Request/Response	Deskripsi
1.	/api/search	POST	body: { search_type: int image: blob } response: { data: array <SearchResult> pdg_url: string }	Melakukan search berdasarkan image dan tipe algoritma search yang dipilih. search_type = 0 untuk dengan tekstur, search_type = 1 untuk dengan color.
2.	/api/search	GET	query parameters: { hash: string limit: int reverse: bool } response: array <SearchRequest>	Untuk debugging backend. Mengecek data yang ingin search.
3.	/api/cache	GET	query parameters: { hash: string limit: int reverse: bool } response: array <SearchResult>	Untuk debugging backend. Mengecek data hasil search.
4.	/api/upload /dataset	POST	body: { is_overwrite: bool images: array <blob> } response: string	Upload dataset dengan menentukan apakah ingin menambahkan dataset atau menghapus dataset sebelumnya tergantikan.

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

				nilai is_overwrite.
5.	/api/upload /dataset	GET	query parameters: { limit: int reverse: bool } response: array <DataSet>	Untuk debugging backend. Mengecek dataset hasil upload.
6.	/api/upload /scrap	POST	body: { web_url: string } response: array <SearchResult>	Melakukan webscrapping berdasarkan web_url beserta melakukan upload dataset.
7.	/api/pdf	POST	body: { hash: string } response: { pdf_url: string }	Mendapatkan pdf hasil search berdasarkan hash dari image yang dicari.

3. Tata Cara Penggunaan Program

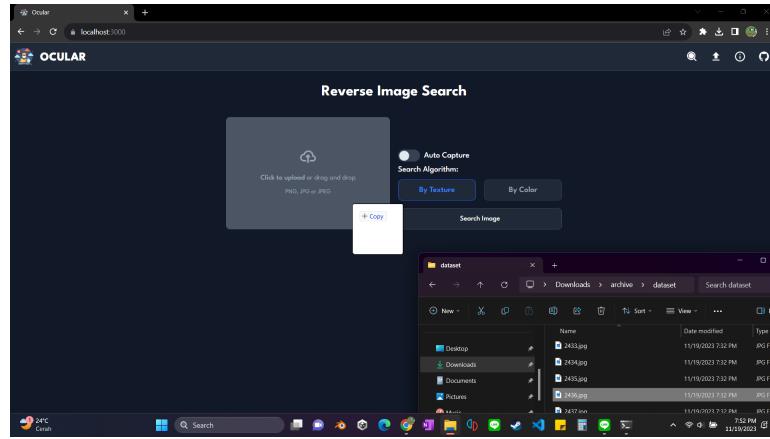
- a. Pencarian gambar relevan dengan upload gambar
 - Pilih laman searching
 - Upload sebuah gambar pada kolom yang disediakan



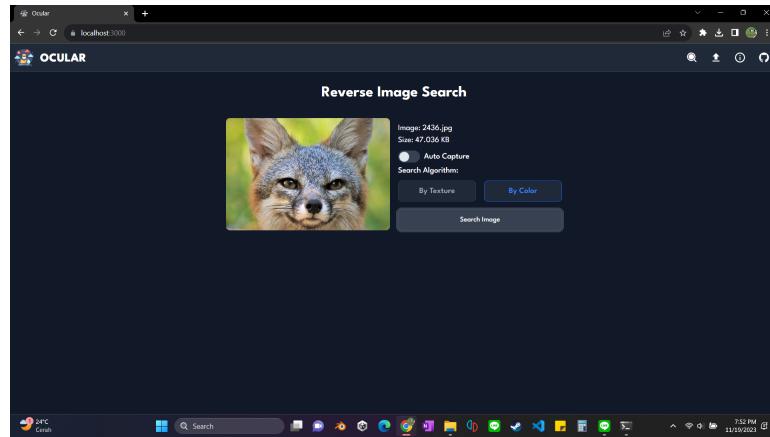
- Bisa juga dengan drag & drop

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

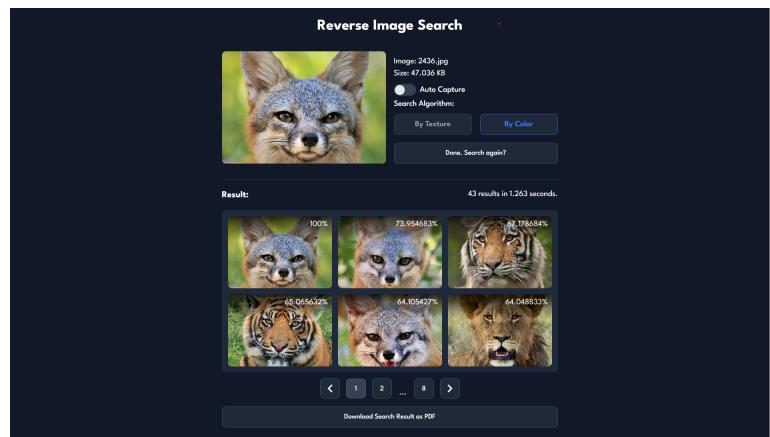
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



- Pilih parameter pencarian, misalnya "By Color"
- Tekan tombol "Search Image"



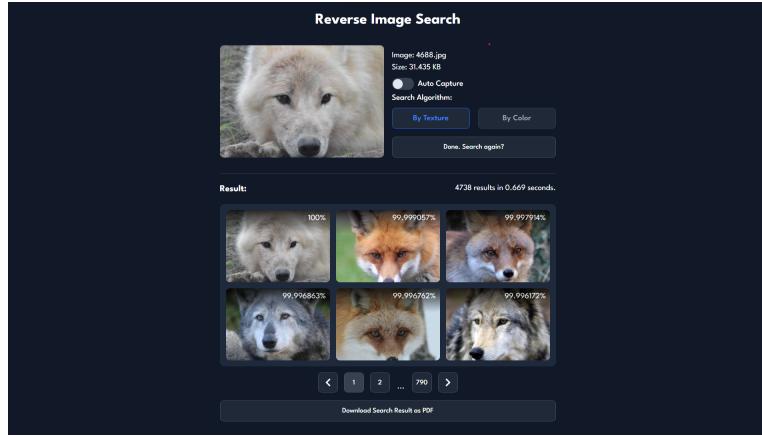
- Akan muncul hasilnya dan bisa melihat gambar selanjutnya dengan pagination



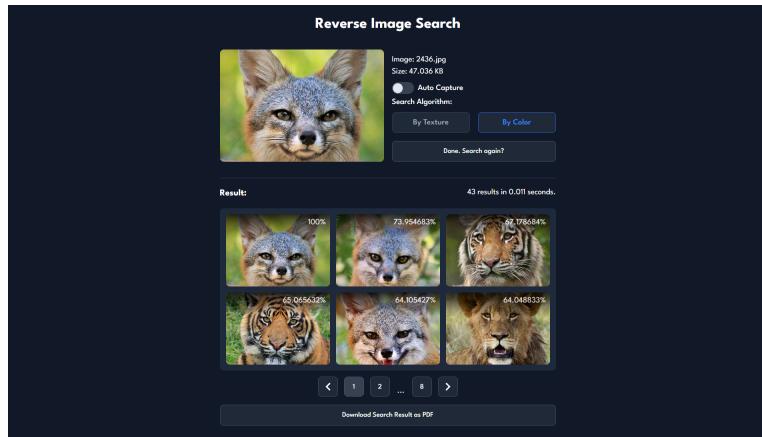
- Coba dengan gambar lain dan parameter lain yaitu "By Texture"

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

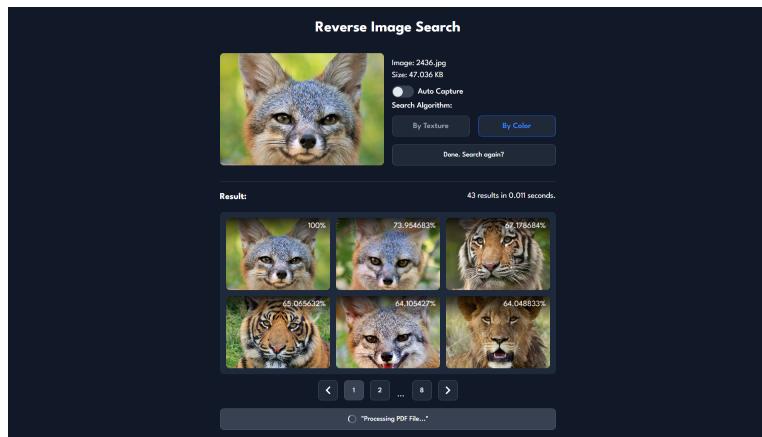
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



- Coba lagi dengan gambar pertama dan parameter "By Color". Maka hasil akan muncul dengan sangat cepat karena cache



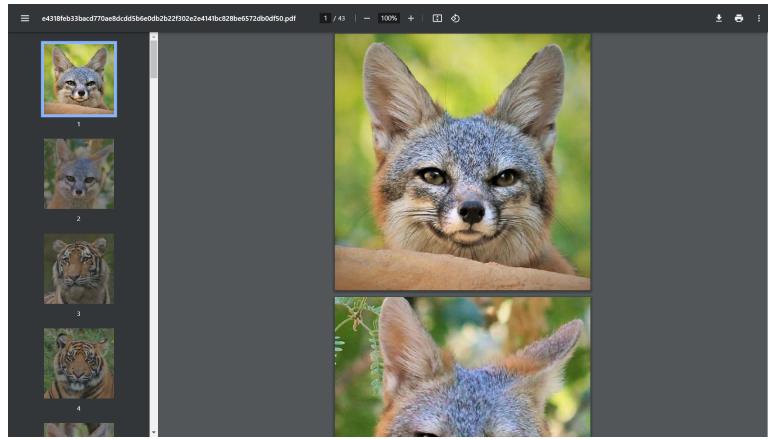
- Untuk tambahan, tekan "Download Search Result as PDF".
- Server akan memproses File PDF nya sebentar.



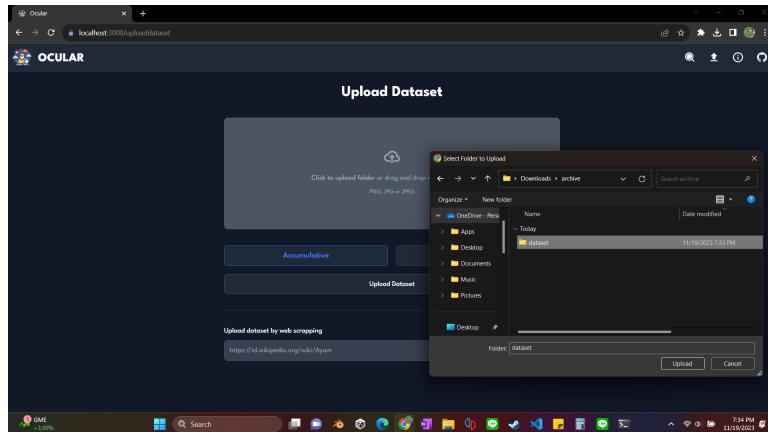
- Saat selesai akan diarahkan ke File PDF tersebut.

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

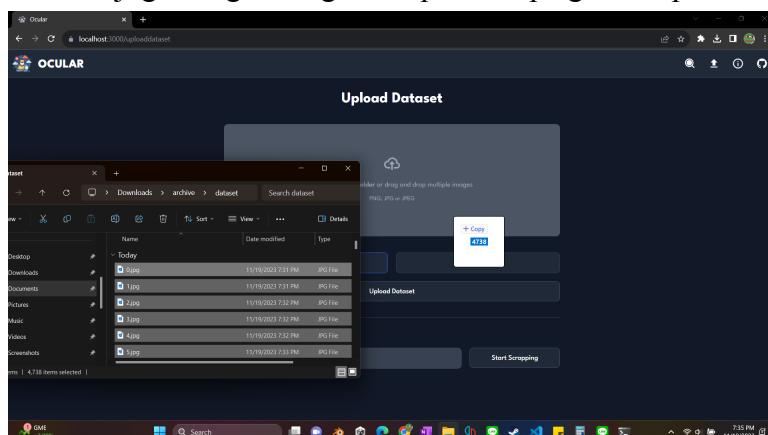
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



- Jika PDF sudah pernah diproses, maka
- b. Upload dataset
 - Pilih laman upload
 - Upload folder berisi gambar pada kolom yang disediakan



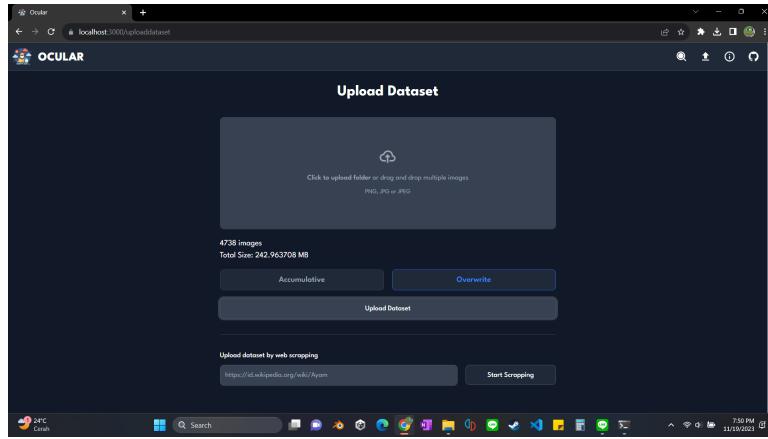
- Bisa juga dengan drag & drop beberapa gambar pada kolom tersebut



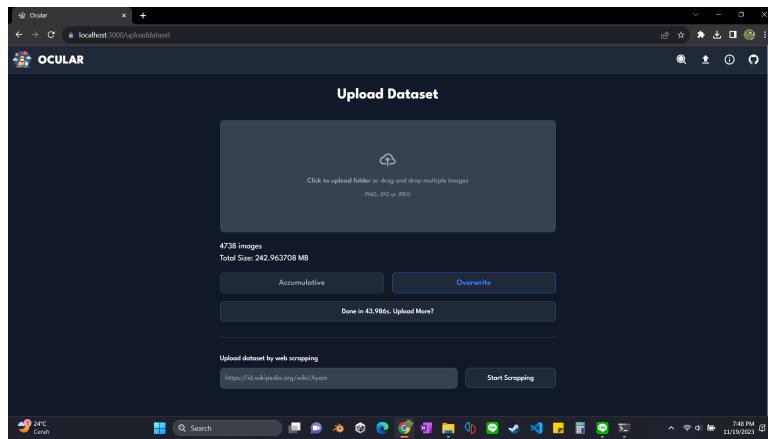
- Pilih opsi upload, misalnya overwrite, yaitu untuk menghapus dataset sebelumnya.
- Tekan tombol "Upload Dataset"

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

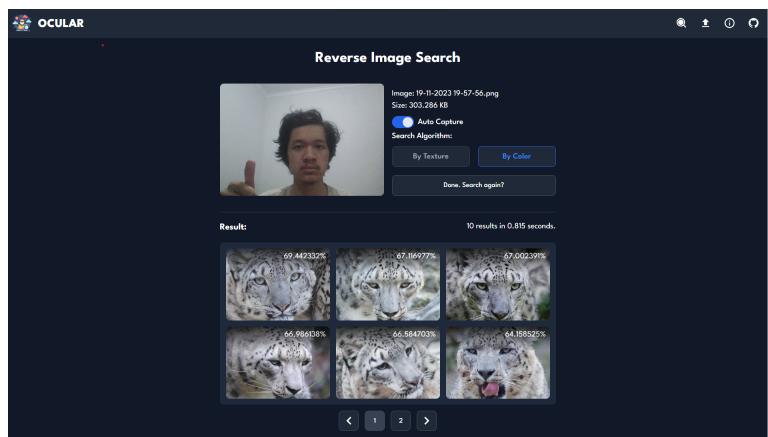
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



- Tunggu hingga upload selesai.



- c. Pencarian gambar relevan dengan kamera
 - Pilih laman searching
 - Nyalakan toggle "Auto Capture"

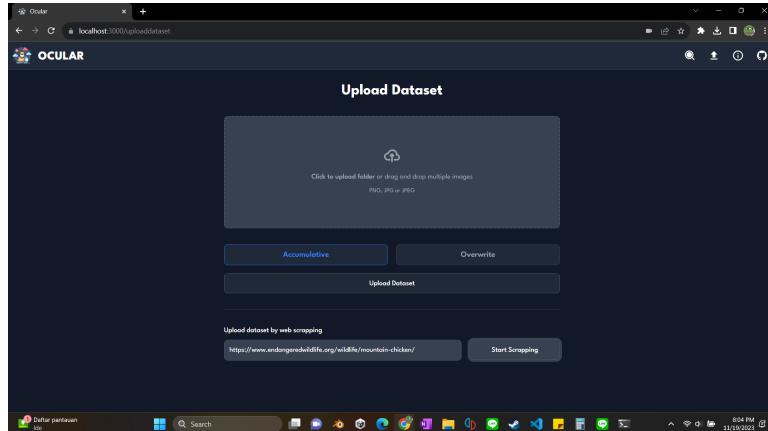


- Pilih allow untuk akses camera jika muncul prompt
 - Gambar akan dimuat setiap 5 detik
- d. Web Scrapping
 - Pilih laman upload

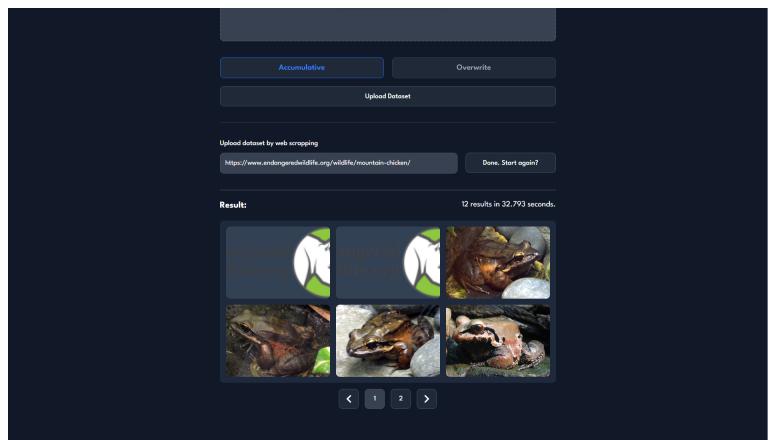
Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

- Masukkan alamat web pada kolom yang disediakan
- Tekan tombol "Start Scrapping"



- Akan muncul hasil scrapping dan bisa melihat gambar selanjutnya dengan pagination

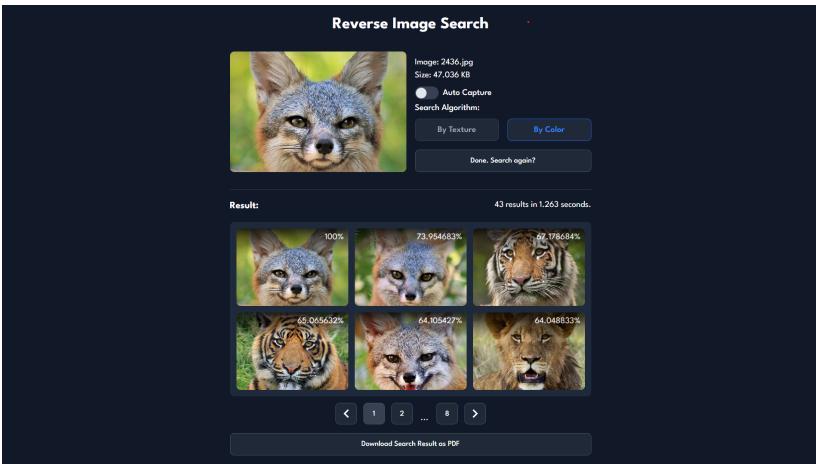
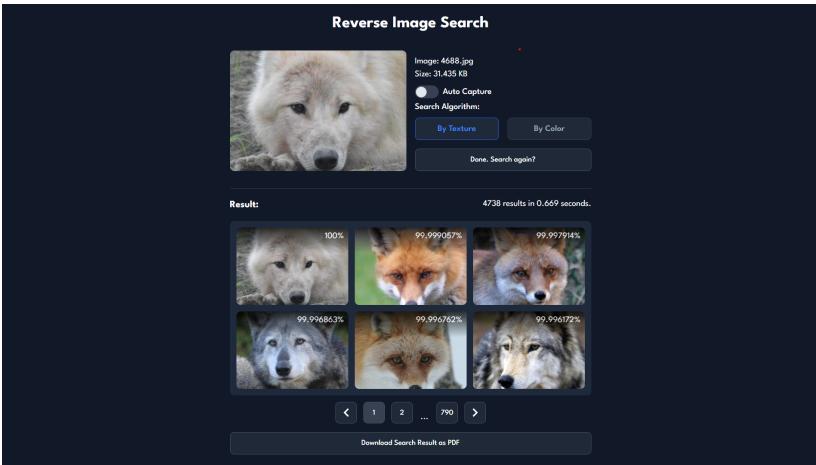
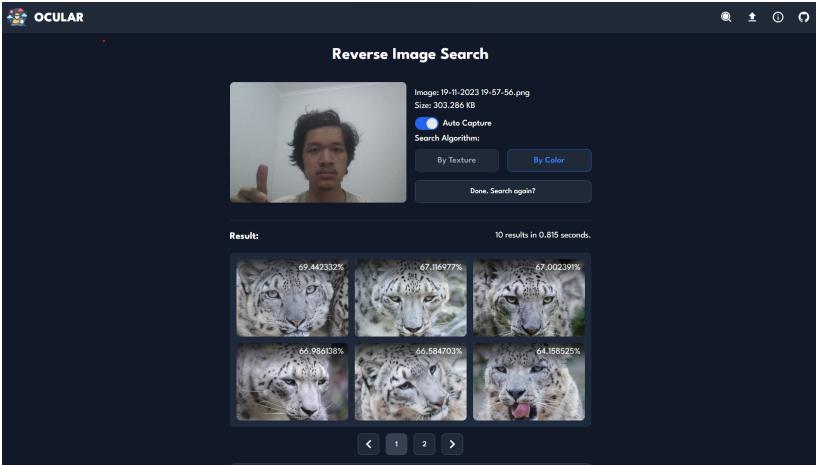


4. Hasil Pengujian

Uji berikut ini dilakukan secara berurutan sehingga jika setelah web scrapping bisa saja terdapat dataset dari hasil web scrapping tersebut

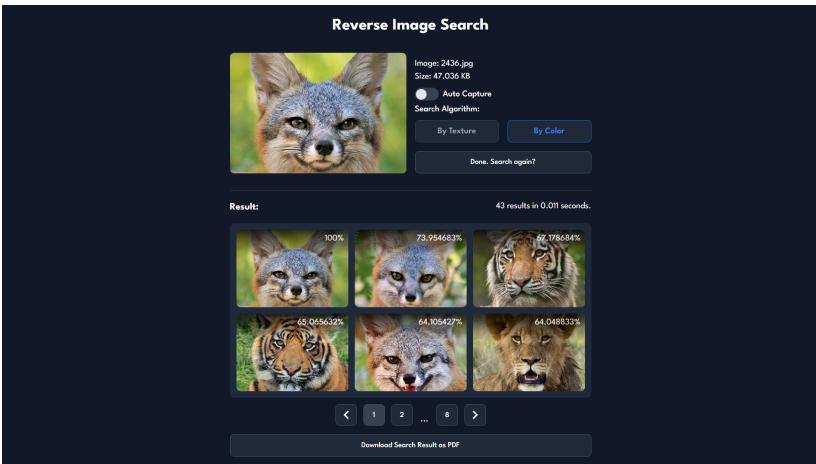
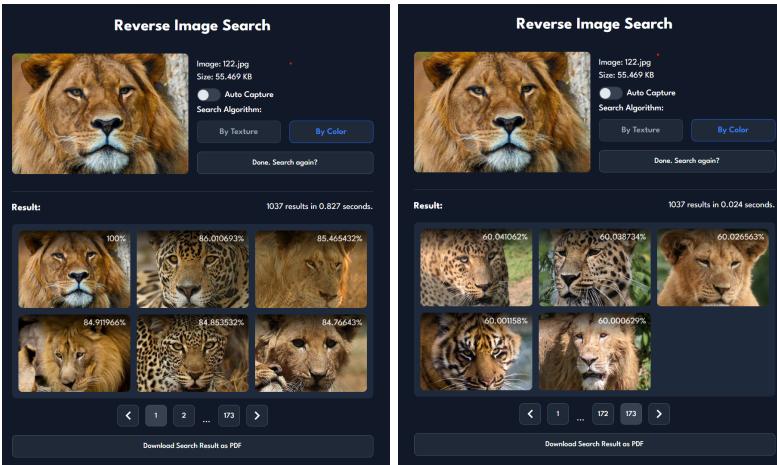
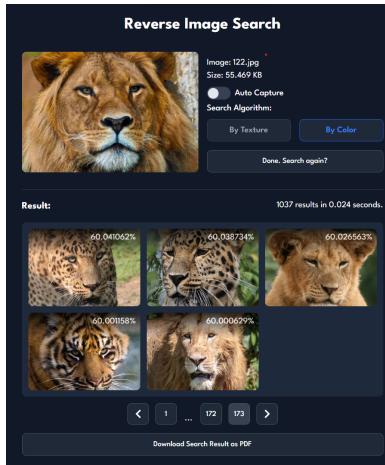
Hasil Tes	Deskripsi
-----------	-----------

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

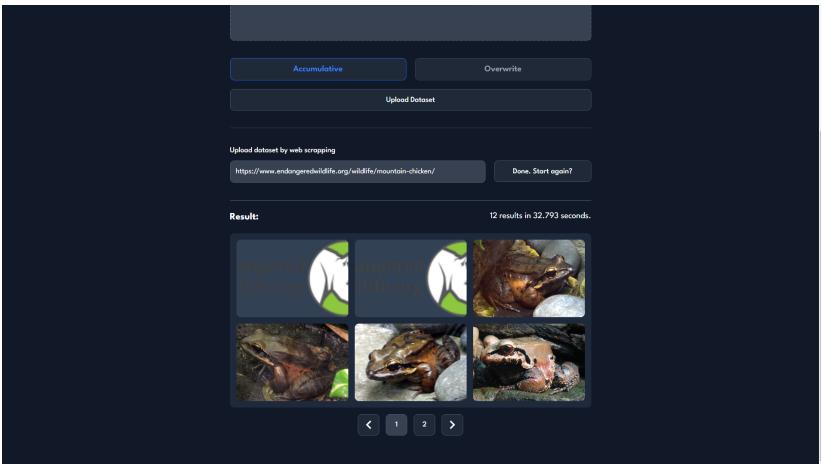
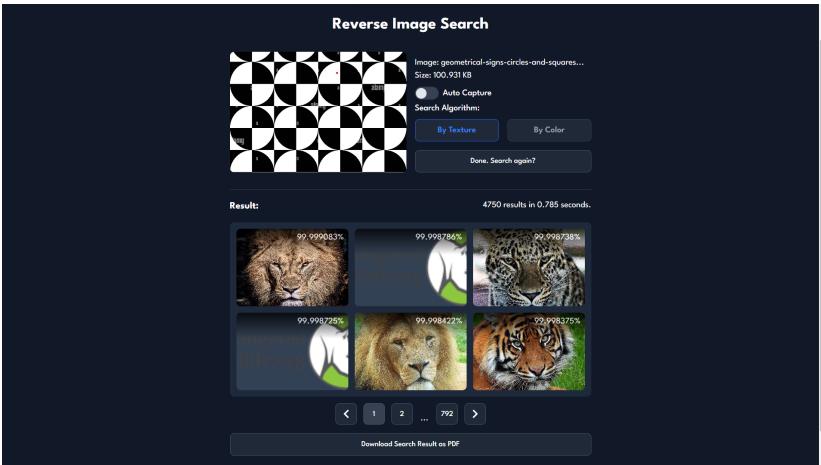
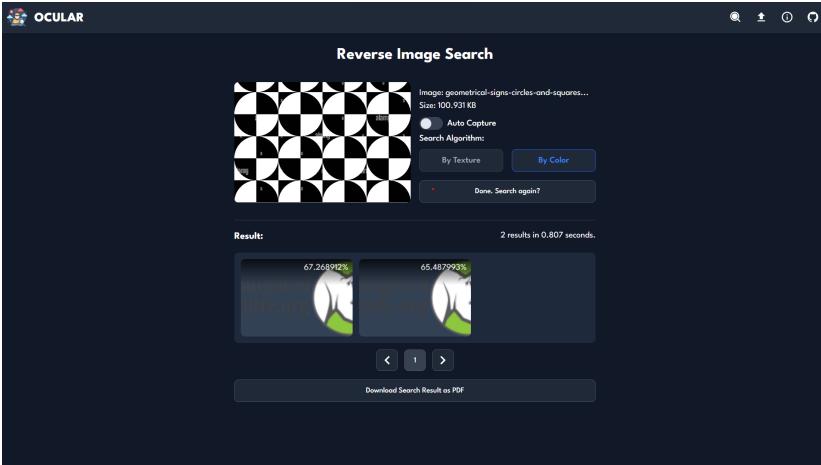
	Search by color
	Search by texture
	Search by color dengan auto capture

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

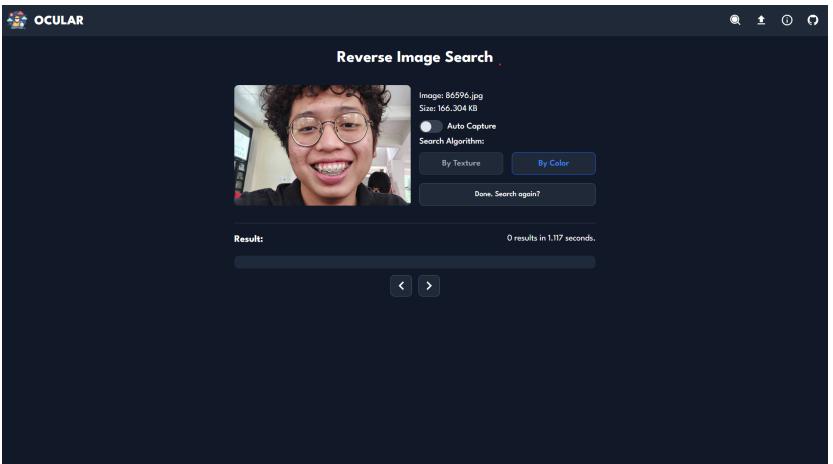
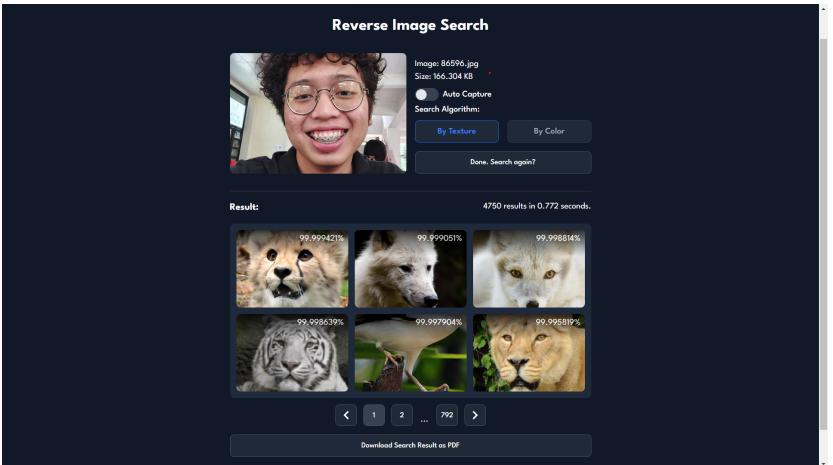
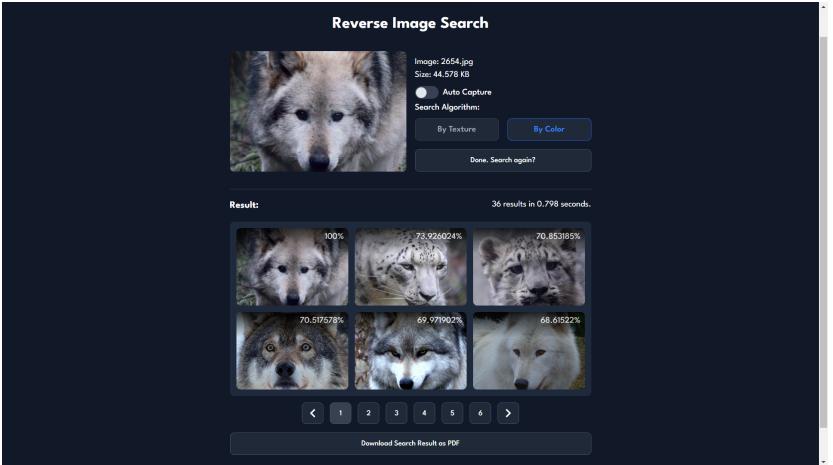
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

	Search by color gambar yang pernah disearch sebelumnya, sehingga akan menggunakan cache untuk mempersingkat waktu pencarian
	Kiri: Search by color Kanan: Page akhir search by color kembali gambar yang sama, sehingga menggunakan cache
	Kiri: Search by texture Kanan: Page akhir search by texture kembali gambar yang sama, sehingga menggunakan cache

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

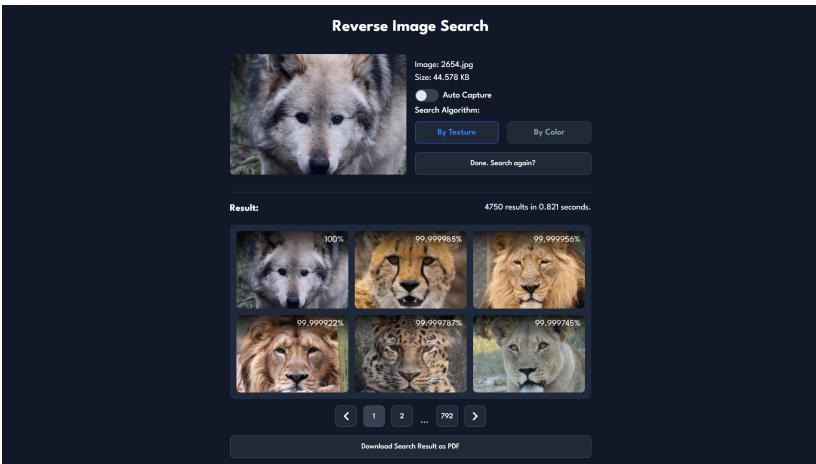
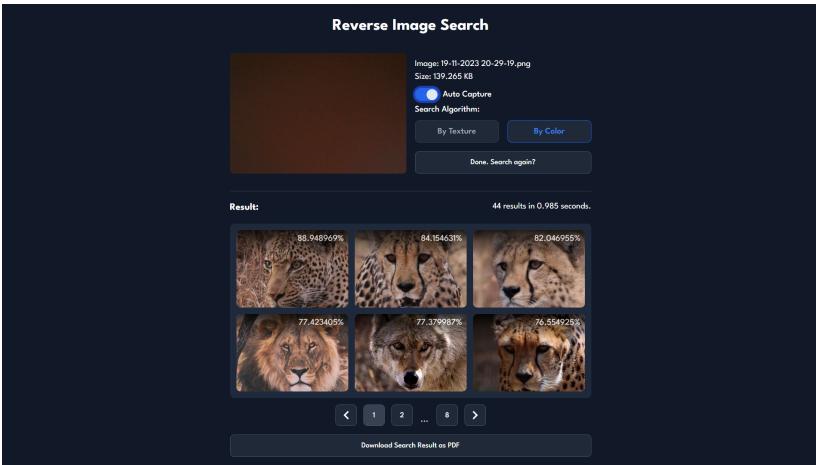
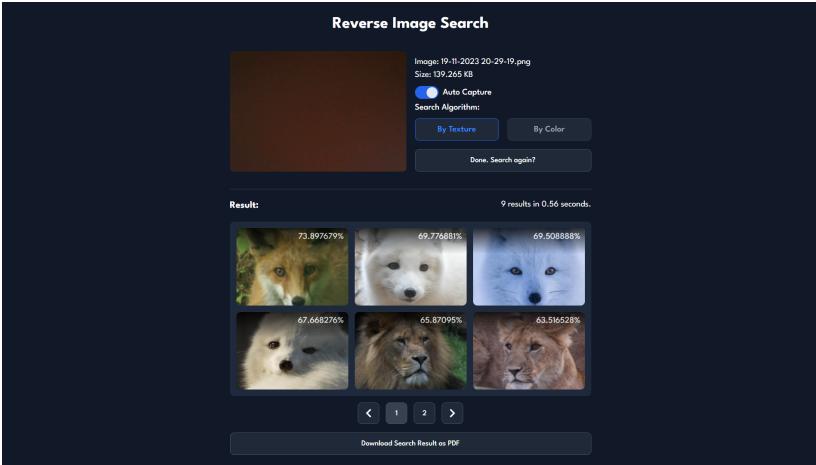
	Hasil dari web scrapping dari url: https://www.endangeredwildlife.org/wildlife/mountain-chicken/
	Search by texture dengan gambar yang memiliki kontras tinggi (gambar yang dicari tidak terdapat dalam dataset)
	Search by color dengan gambar yang memiliki kontras tinggi (gambar yang dicari tidak terdapat dalam dataset)

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

	Search by color gambar Althaf (gambar yang dicari tidak terdapat dalam dataset)
	Search by texture gambar Althaf (gambar yang dicari tidak terdapat dalam dataset)
	Search by color

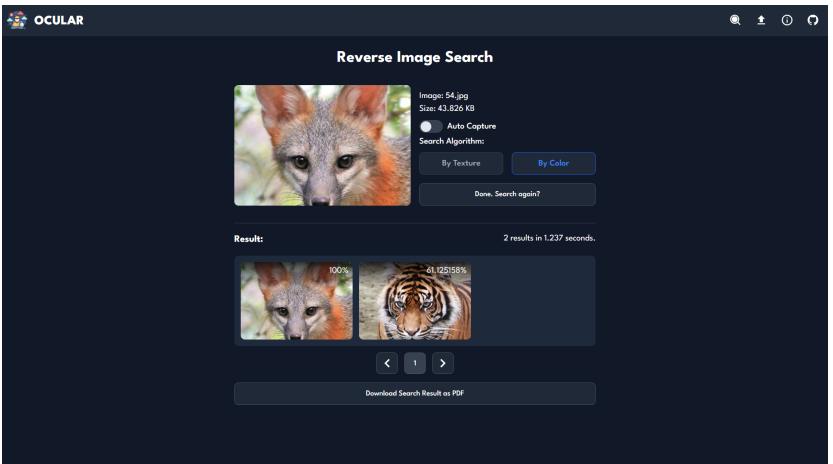
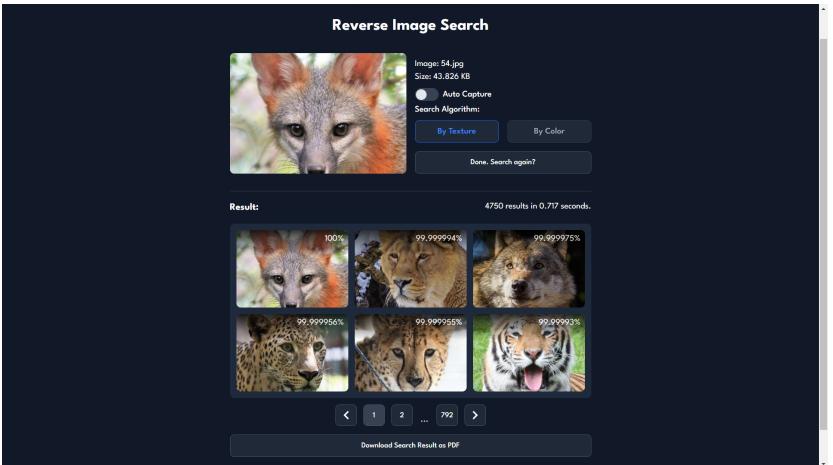
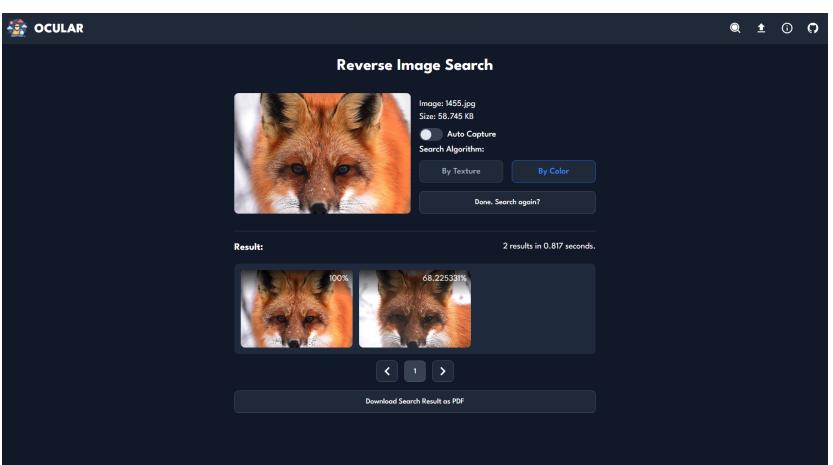
Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

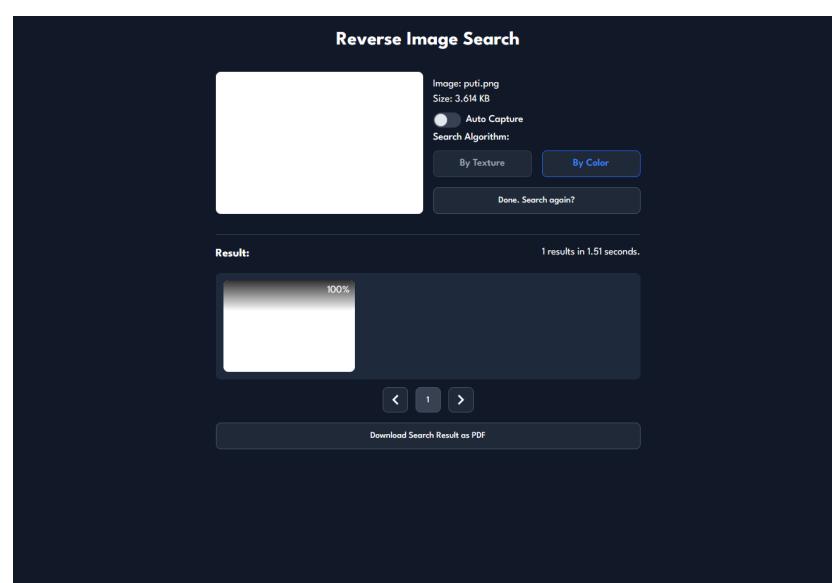
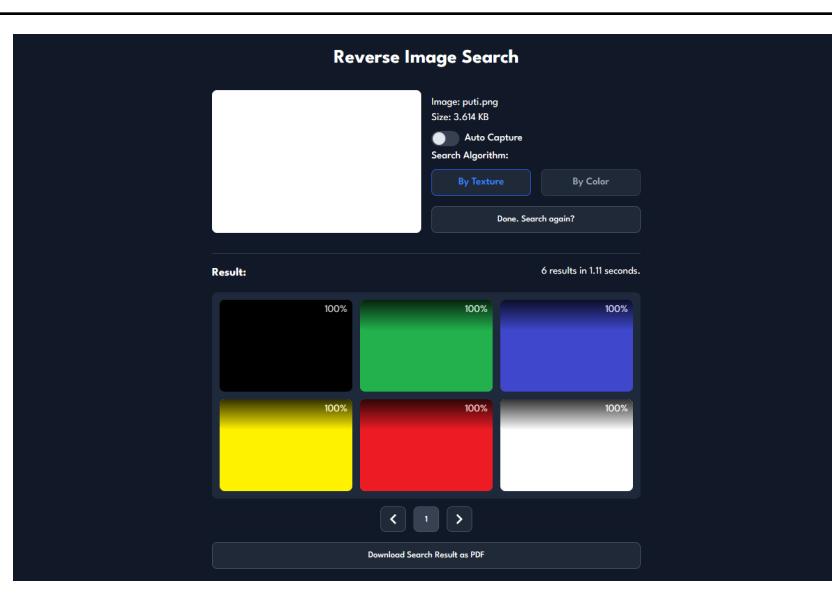
	Search by texture
	Search by color, dengan auto capture dan dengan camera yang ditutup dengan jari
	Search by texture, dengan auto capture dan dengan camera yang ditutup dengan jari

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

	Search by color
	Search by texture
	Search by color

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

 <p>The screenshot shows the 'Reverse Image Search' interface. At the top, there's a placeholder for an image, followed by search parameters: 'Image: puti.png', 'Size: 3.614 KB', 'Auto Capture' (selected), and 'Search Algorithm'. Below these are two buttons: 'By Texture' (highlighted in blue) and 'By Color'. A 'Done. Search again?' button is at the bottom right. The 'Result:' section displays 1 result in 1.51 seconds. It shows a single image of a white rectangle (100%). Navigation buttons <, 1, > are below the image, along with a 'Download Search Result as PDF' button.</p>	Search by color gambar dengan warna full putih. Dataset dimodifikasi menjadi gambar dengan full hanya warna.
 <p>The screenshot shows the 'Reverse Image Search' interface. At the top, there's a placeholder for an image, followed by search parameters: 'Image: puti.png', 'Size: 3.614 KB', 'Auto Capture' (selected), and 'Search Algorithm'. Below these are two buttons: 'By Texture' (highlighted in blue) and 'By Color'. A 'Done. Search again?' button is at the bottom right. The 'Result:' section displays 6 results in 1.11 seconds. It shows a grid of six colored squares: black (100%), green (100%), blue (100%), yellow (100%), red (100%), and white (100%). Navigation buttons <, 1, > are below the grid, along with a 'Download Search Result as PDF' button.</p>	Search by texture gambar dengan warna full putih. Dataset dimodifikasi menjadi gambar dengan full hanya warna.

5. Analisis

Berdasarkan test case yang kami lakukan, terlihat bahwa hasil dari pencarian berdasarkan warna dan berdasarkan tekstur keduanya cukup akurat, dan menghasilkan gambar dari dataset yang terlihat cukup mirip dan menyerupai gambar search request, dan pasti mengembalikan gambar yang sama dengan gambar search request dengan similarity 100%.

Terlihat dari test case juga bahwa search by color dan tekstur memiliki tingkat akurasi yang berbeda di situasi yang berbeda juga. Pada search by color, gambar yang memiliki tingkat kemiripan tinggi adalah gambar yang memiliki total warna yang menyerupai search request pada blok-bloknya. Sedangkan pada search by texture, cenderung mengembalikan gambar yang memiliki tingkat kekontrasan yang mirip, jika sebuah gambar memiliki warna yang mirip tetapi tingkat kemiripan kekontrasan yang rendah, maka akan memiliki tingkat kemiripan yang rendah juga. Misalnya pada search by texture jika kita melakukan search gambar full berwarna putih, maka ia akan mengembalikan gambar (jika terdapat di dataset) gambar full berwarna hitam, merah, hijau, dan lain-lain, dikarenakan gambar-gambar tersebut memiliki tingkat kekontrasan yang sama, walaupun memiliki warna yang sepenuhnya berbeda.

Hal yang mencolok dari pencarian berdasarkan tekstur adalah, pencarian ini menghasilkan persentase kemiripan yang cukup tinggi untuk setiap pencariannya. Hal ini dikarenakan adanya komponen dalam pencarian berdasarkan tekstur yang cukup dominan dibandingkan yang lain, yaitu komponen contrast. Sehingga ketika vektor komponen tekstur dimasukkan dalam persamaan cosine similarity akan menghasilkan angka yang cukup besar. Tetapi hal ini tidak mengubah keakuratan pencarian gambar, dikarenakan hanya representasi persentasenya yang besar, tetapi kemiripannya dibandingkan gambar lain tetap sesuai. Salah satu cara untuk memperkecil angka persentase ini adalah dengan menerapkan normalisasi pada komponen tersebut, karena range dari komponen contrast tidaklah pasti, maka terdapat kemungkinan gambar dengan cosine similarity diatas 60% yang tidak termasuk dalam pool hasil, untuk itu kami memilih untuk tidak menerapkannya.

BAB V

1. Kesimpulan

Dari pengerjaan Tugas Besar II ini, kami berhasil menerapkan pelajaran Aljabar Linear dan Geometri pada program komputer, dan juga berhasil melakukan eksplorasi dan penerapan pengembangan website. Permasalahan yang dapat kami selesaikan adalah membuat sebuah reverse image searcher yang bekerja dengan baik dan akurat. Dengan mengimplementasikan metode - metode yang telah dipelajari, akhirnya Tugas Besar II ini berhasil direalisasikan.

2. Saran dan Komentar

Saran kami untuk Tugas Besar II ini adalah, untuk kedepannya, sebaiknya diberikan waktu yang lebih lama sesuai dengan beban tugasnya, considering tugas besar ini melibatkan beberapa metode atau materi yang (mungkin) beberapa dari kami belum mempelajari sama sekali, maka menurut kami akan lebih baik jika waktu yang diberikan lebih lama.

3. Refleksi

Dari Tugas Besar II ini, selain kami mempelajari banyak hal, kami juga dapat belajar untuk bekerja sama dan berkomunikasi antara anggota kelompok, dan berusaha memanfaatkan waktu sebaik - baiknya.

4. Ruang Perbaikan

Masih terdapat cara untuk mempercepat perhitungan dataset. Walau pun sudah memanfaatkan code dari bahasa c untuk perhitungan berat dan multiprocessing python, karena framework yang digunakan adalah Django, maka masih akan terdapat limitasi dari python. Contoh dari limitasi ini misalnya pada initial setup multiprocessing dan saat menyimpan gambar ke database dengan SQLite.

DAFTAR PUSTAKA

Link Repository: <https://github.com/DhafinFawwaz/Algeo02-22043>

Link Video: <https://youtu.be/lEwqx5FcK60>

Anthony. (2019, August 5). *Stop Misusing Toggle Switches*. UX Movement. Retrieved November 19, 2023, from

<https://uxmovement.com/mobile/stop-misusing-toggle-switches/>

Codecademy Team. (n.d.). *What is a Database Index?* Codecademy. Retrieved November 19, 2023, from <https://www.codecademy.com/article/sql-indexes>

Content-based image retrieval. (n.d.). Wikipedia. Retrieved November 19, 2023, from https://en.wikipedia.org/wiki/Content-based_image_retrieval

Content-based image retrieval using color and texture fused features. (2023, June 16). YouTube. Retrieved November 19, 2023, from

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

Grayscale. (n.d.). Wikipedia. Retrieved November 19, 2023, from <https://en.wikipedia.org/wiki/Grayscale>

Web development. (n.d.). Wikipedia. Retrieved November 19, 2023, from https://en.wikipedia.org/wiki/Web_development