LAPORAN TUGAS BESAR 2

IF2123 ALJABAR LINEAR DAN GEOMETRI

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Disusun oleh:

Agil Fadillah Sabri (13522006)

Irfan Sidiq Permana (13522007)

Diero Arga Purnama (13522056)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR PERSAMAAN

BAB I
DESKRIPSI MASALAH

1.1 Abstraksi

Dalam era digital, jumlah gambar yang dihasilkan dan disimpan semakin meningkat dengan pesat, baik dalam konteks pribadi maupun profesional. Peningkatan ini mencakup berbagai jenis gambar, mulai dari foto pribadi, gambar medis, ilustrasi ilmiah, hingga gambar komersial. Terlepas dari keragaman sumber dan jenis gambar ini, sistem temu balik gambar (*image retrieval system*) menjadi sangat relevan dan penting dalam menghadapi tantangan ini. Dengan bantuan sistem temu balik gambar, pengguna dapat dengan mudah mencari, mengakses, dan mengelola koleksi gambar mereka. Sistem ini memungkinkan pengguna untuk menjelajahi informasi visual yang tersimpan di berbagai platform, baik itu dalam bentuk pencarian gambar pribadi, analisis gambar medis untuk diagnosis, pencarian ilustrasi ilmiah, hingga pencarian produk berdasarkan gambar komersial. Salah satu contoh penerapan sistem temu balik gambar yang mungkin kalian tahu adalah Google Lens. Di dalam Tugas Besar 2 ini, mahasiswa 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

2.1 *Content-Based Information Retrieval* (CBIR)

CBIR merupakan suatu metode pencarian yang membandingkan antara gambar *query* dengan gambar yang ada pada *dataset* (*query by example*). Sistem CBIR pada umumnya dibangun dengan melihat karakteristik atau ciri-ciri suatu gambar tersebut (Ramadijanti, 2006).

Metode yang dipakai pertama kali pada tahun 1992 ini digunakan oleh T. Koto untuk mendeskripsikan *experiment automatic retrieval image* dari sebuah *database* berdasarkan warna dan bentuk. Istilah untuk menggambarkan proses tersebut dapat diartikan dengan mengambil kembali citra yang diinginkan pada basis data berdasarkan fitur-fitur yang dimiliki oleh gambar (Irawan, 2010).

Content Based Image Retrieval (CBIR) merupakan proses pencarian gambar dengan membandingkan fitur-fitur yang terdapat pada gambar yang dicari dengan gambar yang terdapat dalam basis data (Hidayat dkk., 2017). Terdapat dua tingkatan fitur yang digunakan dalam CBIR yaitu fitur *high level* dan *low level*. Fitur *low level* yang sering digunakan pada CBIR adalah warna, tekstur dan bentuk. Pencarian dengan menggunakan fitur *low level* lebih mudah untuk dilakukan. Pencarian fitur *high level* yang sering digunakan yaitu kesan, emosi dan makna. Pencarian dengan menggunakan fitur *high level* cukup sulit dilakukan. Hal ini disebabkan adanya ruang pemisah semantik antara konsep pengguna dengan fitur *high level* yang terdapat pada gambar (Hidayat dkk., 2017).

2.2 Ekstraksi Fitur

Pada umumnya, fitur dinyatakan dengan suatu tanda yang khas, yang membedakan antara suatu gambar dengan gambar yang lain. Elemen pada sebuah gambar disebut fitur. Fitur dibagi menjadi dua tingkatan yaitu fitur *high level* dan fitur *low level*. Fitur *low level* lebih mudah diterapkan dibandingkan dengan fitur *high level*. Fitur *low level* yang sering digunakan adalah fitur warna dan fitur tekstur. Adapun ciri-ciri atau fitur dasar dari gambar sebagai berikut:

2.2.1 Warna

Ciri warna suatu gambar dapat dinyatakan dalam bentuk histogram dari gambar tersebut. Histogram warna adalah jumlah kemunculan berbagai warna yang ada pada ruang warna tertentu. Ada beberapa jenis ruang warna yang sering digunakan, seperti RGB, CMYK, HSV, dll.

Histogram warna dibuat untuk melakukan pendistribusian warna dari suatu gambar. Histogram warna tidak bisa mendeteksi sebuah objek yang spesifik yang terdapat pada gambar dan tidak bisa mendeskripsikan posisi dari warna yang didistribusikan.

Pembentukan ruang warna perlu dilakukan dalam rangka pembagian nilai citra menjadi beberapa *range* yang lebih kecil. Hal ini dilakukan dengan membuat sebuah histogram warna yang setiap interval tertentu disebut sebagai *bin*. Histogram warna dapat dihitung dengan menghitung piksel yang menyatakan nilai warna pada setiap interval. Fitur warna mencakup histogram warna global dan histogram warna blok.

Pada perhitungan histogram, warna global HSV lebih dipilih karena warna tersebut dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum untuk digunakan. Maka dari itu, perlu dilakukan konversi warna dari RGB ke HSV dengan prosedur sebagai berikut.

1. Nilai dari RGB dinormalisasi dengan mengubah nilai *range* [0, 255] menjadi [0, 1].

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

Persamaan 1 Normalisasi RGB

2. Mencari *Cmax, Cmin*, dan Δ.

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

Persamaan 2 *Cmax, Cmin*, dan Δ

3. Selanjutnya gunakan hasil perhitungan di atas untuk mendapatkan nilai HSV.

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

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

$$V = Cmaks$$

Persamaan 3 HSV

2.2.2 Tekstur

Tekstur merupakan karakteristik intrinsik dari suatu citra yang terkait dengan tingkat kekasaran (*roughness*), granularitas (*granulation*), dan keteraturan (*regularity*) susunan struktur pixel. Aspek tekstural dari sebuah citra dapat dimanfaatkan sebagai dasar dari segmentasi, klasifikasi, maupun interpretasi citra. Pada aspek tekstural ini, warna suatu gambar menjadi tidak terlalu penting, sehingga dalam pengambilan fitur tekstur, suatu gambar diubah menjadi citra *greyscale*-nya. Adapun untuk ruang warna RGB, nilai *greyscale* suatu piksel dinyatakan dalam persamaan berikut:

$$Y = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.144\mathrm{B}$$

Persamaan 4 Konversi RGB ke *Grayscale*

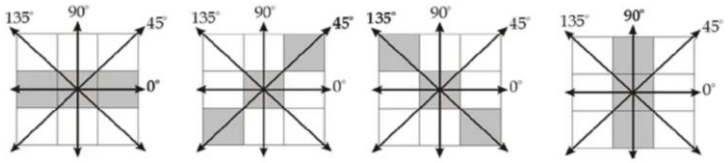
Salah satu metode pengekstrakan fitur tekstur suatu gambar adalam metode *Gray-Level Co-occurrence matrix* (GLCM). Menurut Lili dan Peng (2013), GLCM merupakan suatu metode untuk melakukan analisis terhadap suatu piksel pada citra dengan mengetahui tingkat keabuan yang terjadi. GLCM merupakan suatu metode untuk melakukan ekstraksi ciri berbasis statistika. Perolehan ciri diperoleh dari pengambilan nilai dua buah piksel, yang mempunyai nilai tertentu dan membentuk suatu sudut pola. Jarak kedua piksel dinyatakan sebagai parameter *offset* (Δx,Δy). Adapun parameter sudut (θ) yang bisa dipakai dari nilai piksel citra menggunakan GLCM adalah 0º, 45º, 90º, 135º.

Dengan menggunakan nilai i dan j sebagai nilai intensitas dari gambar dan p serta q sebagai posisi dari gambar, maka *offset* Δx dan Δy bergantung pada arah θ dan jarak yang digunakan melalui persamaan di bawah ini.

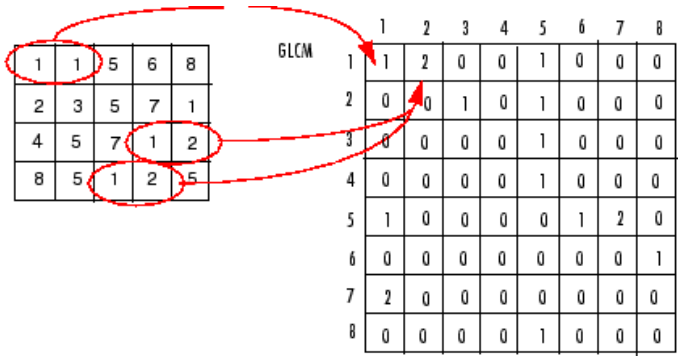
$$C_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1, & \text{jika } I(p,q) = i \text{ dan } I(p+\Delta x, q+\Delta y) = j \\ 0, & \text{jika lainnya} \end{cases}$$

Persamaan 5 Pembentukan *Co-Occurrence Matrix*

Dari persamaan tersebut terbentuk matrik *co-occurrence*. Adanya matrik tersebut didasarkan pada kondisi bahwa suatu piksel akan mempunyai nilai perulangan sehingga terdapat pasangan keabuannya (Thakare & N Patil, 2014). Kondisi nilai piksel tersebut dinotasikan sebagai matriks dengan jarak dua posisi (x_1, y_1) dan (x_2, y_2).



Gambar 1 *Offset* dan Nilai Sudut θ



Gambar 2 Proses Pembuatan Matriks *Occurrence*

Adapun dari matriks *co-occurrence* yang terbentuk, bisa dilakukan proses ekstraksi tekstur. Beberapa komponen tekstur yang sering digunakan yaitu *contrast*, *entropy*, *homogeneity*, *dissimilarity*, ASM (*angular second moment*), dan *energy*. Persamaan yang dapat digunakan untuk mendapatkan nilai komponen-komponen tersebut antara lain:

$\sum_{1,j=0}^{dimensi-1} P_{i,j}(i-j)^2$ Persamaan 6 *Contrast*	$\sum_{1,j=0}^{dimensi-1} P_{i,j}\lvert i-j\rvert^2$ Persamaan 9 *Dissimilarity*
$\sum_{1,j=0}^{dimensi-1} \frac{P_{i,j}}{1+(i-j)^2}$ Persamaan 7 *Homogeneity*	$\sum_{1,j=0}^{dimensi-1} {P_{i,j}}^2$ Persamaan 10 *ASM*
$-\left(\sum_{1,j=0}^{dimensi-1} P_{i,j} \times logP_{i,j}\right)$ Persamaan 8 *Entropy*	$\sqrt{ASM}$ Persamaan 11 *Energy*

2.3 Pengembangan Website

Website atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, atau gabungan dari semuanya baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman. Hubungan antara satu halaman web dengan halaman web yang lainnya disebut *hyperlink*, sedangkan teks yang dijadikan media penghubung disebut *hypertext*.

Dalam pengembangan sebuah website, umumnya dibagi menjadi dua, yaitu:

2.3.1 *Front-End*

Front-End merupakan istilah yang digunakan untuk pengembang yang bertugas untuk membuat tampilan bentuk web menggunakan bahasa pemrograman. *Front-End* bertanggung jawab pada pengelolaan desain murni sebuah website. Tampilan ini merupakan tampilan yang akan dilihat dan dioperasikan secara langsung oleh pengguna.

Pada bagian *Front-End*, bahasa yang sering digunakan adalah HTML dan CSS. *Hypertext Markup Language* (HTML) adalah bahasa markah standar untuk dokumen yang dirancang untuk ditampilkan di peramban internet. Ini dapat dibantu oleh teknologi seperti *Cascading Style Sheets* (CSS) yang merupakan aturan untuk mengatur beberapa komponen dalam sebuah web sehingga lebih terstruktur dan seragam.

2.3.2 *Back-End*

Back-End merupakan bagian dari web yang bertanggung jawab untuk menyediakan kebutuhan yang tak terlihat oleh pengguna (tidak berinteraksi langsung dengan pengguna). Ini adalah bagian yang bertanggung jawab atas pemrosesan data, logika bisnis, dan interaksi dengan database. Backend merupakan "otak" dari suatu aplikasi web yang menangani operasi-operasi yang tidak terlihat oleh pengguna secara langsung.

Bahasa pemrograman yang umum digunakan untuk mengembangkan bagian *Back-End* termasuk Python, PHP, Ruby, Java, JavaScript (Node.js), dan bahasa pemrograman tujuan umum lainnya. Framework-backend seperti Express.js (untuk Node.js), Django dan Flask (untuk Python), Laravel (untuk PHP), dan Spring (untuk Java) membantu mempercepat proses pengembangan dan menyediakan alat untuk mengatur logika bisnis, interaksi database, dan manajemen permintaan pengguna

BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Dalam sebuah program *Content-Based Image Retrieval* (CBIR), proses pencocokan suatu gambar *query* dengan kumpulan gambar pada *dataset* dimulai dengan ekstraksi fitur-fitur penting dari setiap gambar, yaitu warna dan tekstur. Fitur-fitur warna dan tekstur sebuah gambar diekstrak menggunakan langkah-langkah dan persamaan yang telah disebutkan di Bab II Landasan Teori Subbab 2.

Dalam ekstraksi fitur warna, suatu gambar dibagi menjadi 4×4 blok dengan tujuan untuk memperkecil jumlah fitur yang terbentuk. Ukuran blok 4×4 dipilih karena merupakan ukuran yang efektif dalam proses ekstraksi fitur warna. Jika ukuran blok terlalu besar, maka setiap blok menjadi tidak terlalu signifikan sehingga mengurangi keakuratan hasil pencarian walau mempercapat waktu ekstraksi. Adapun jika ukuran blok terlalu kecil dapat akan meningkatkan waktu dalam ekstraksi tetapi meningkatkan keakuratan hasil.

Dalam ekstraksi fitur tekstur, digunakan parameter *offset* (Δx,Δy) sebesar 1 piksel dengan sudut (θ) 0º. Kedua nilai tersebut dipilih karena dapat mempermudah proses implementasi kode yang dibuat. Kemudian, matriks co-occurrence yang terbentuk dari Persamaan 5 dijumlahkan dengan transposenya untuk membentuk matriks simetri. Untuk menghindari overflow bilangan saat perhitungan selanjutnya, matriks simetri yang terbentuk dinormalisasi dengan persamaan berikut:

$$MatriksNorm = \frac{MatriksOcc}{\sum MatriksOcc}$$

Persamaan 12 Normalisasi Matriks *Co-Occurence*

Setelah proses ekstraksi selesai, akan diperoleh sebuah vektor histogram warna (HSV) berdimenasi 64 (256/4 dengan 256 range dari nilai RGB dan 4 sebagai blok histogram warna yang dipilih), serta vektor tekstur berdimensi 6 (*contrast*, *entropy*, *homogeneity*, *dissimilarity*, ASM, dan *energy*) dari setiap gambar (*query* dan *dataset*).

Vektor yang dihasilkan ini kemudian yang akan digunakan sebagai parameter perbandingan antara gambar *query* dengan gambar pada *dataset*. Proses pembandingan memanfaatkan konsep perkalian dot dalam konsep vektor. Dalam hal ini, akan dicari nilai cosinus antara vektor *query* dengan vektor masing-masing gambar pada *dataset* menggunakan persamaan berikut:

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

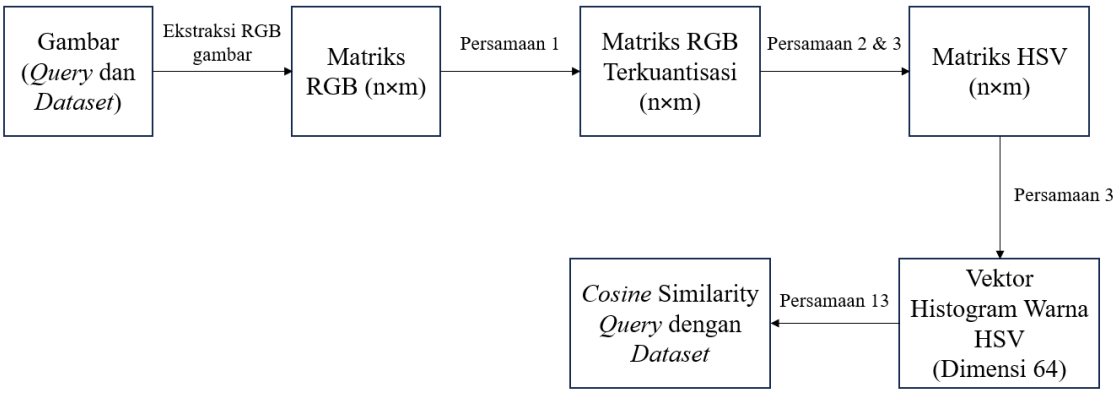
Persamaan 13 *Cosine Similarity*

Dengan A dan B adalah vektor dan n adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil nilai cosinusnya yang disebut *Cosine Similarity*. Suatu vektor dianggap mirip jika sudut antara kedua vektor tersebut kecil (hampir berhimpit). Karena nilai cosinus berbanding terbalik terhadap besar sudutnya (0^o-90^o), maka semakin besar nilai cosinus (mendekati 1), semakin kecil nilai sudutnya (mendekati 0^o), sehingga semakin mirip kedua vektor tersebut. Dengan demikian, suatu gambar akan dikatakan mirip jika hasil *Cosine Similarity* mendekati satu.

Setelah mendapatkan hasil *Cosine Similarity* dari gambar *query* dengan masing-masing gambar pada *dataset*, kemudian dilakukan proses perankingan dari yang terbesar hingga ke yang terkecil. Adapun hasil pencarian gambar yang akan ditampilkan kepada pengguna adalah gambar dengan persentase kemiripan diatas 60% (*Cosine Similarity* > 0,6).

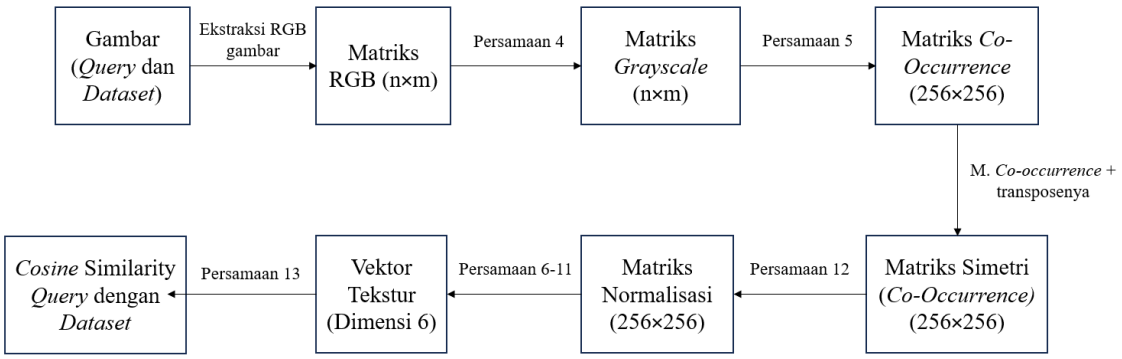
3.2 Pemetaan Masalah menjadi Elemen pada Aljabar Geometri

3.2.1 CBIR dengan Parameter Warna



Gambar 3 Pemetaan Masalah pada CBIR dengan Parameter Warna menjadi Elemen pada Aljabar Geometri

3.2.2 CBIR dengan Parameter Tekstur



Gambar 4 Pemetaan Masalah pada CBIR dengan Parameter Tekstur menjadi Elemen pada Aljabar Geometri

3.3 Contoh Ilustrasi Kasus dan Penyelesaiannya

Misalkan pengguna memiliki sekumpulan gambar *dataset* dan sebuah *query*. Pertama-tama, pengguna harus memasukkan *dataset* ke program terlebih dahulu. Setelah itu, program akan melakukan ekstraksi fitur-fitur pada setiap gambar yang ada pada *dataset* dan menyimpan hasil ekstraksinya ke dalam *database*. Setelah itu, pengguna bisa memasukkan gambar *query* yang ingin dicari ke dalam program. Program lalu mengekstraksi fitur-fitur yang terdapat pada gambar *query*.

Setelah selesai mengekstrak fitur, maka program sekarang tinggal melakukan pencarian ke dalam database. Jika pencarian yang diinginkan adalah dengan metode parameter warna, maka program akan membandingkan vektor fitur warna pada query dengan vektor fitur warna pada gambar *dataset* menggunakan persamaan *cosine similarity*. Begitu pula, jika pengguna ingin melakukan pencarian dengan metode parameter tekstur, maka program akan membandingkan vektor fitur tekstur pada query dengan vektor fitur tekstur pada gambar *dataset* menggunakan persamaan *cosine similarity*. Hasil perbandingannya kemudian disimpan ke dalam suatu list yang diurutkan secara menurun (dari besar ke kecil).

Setelah selesai melakukan pembandingan, program kemudian hanya perlu menampilkan gambar sesuai list terurut yang terbentuk kepada pengguna. Gambar dengan persentase tertinggi akan menjadi gambar yang paling mirip dengan *query*. Adapun dengan pembatasan gambar yang akan ditampilkan, hanya gambar yang tingkat kemiripannya > 60% yang akan ditampilkan ke pengguna.

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1 Implementasi Program Utama

4.1.1 CBIR dengan Parameter Warna

```
function rgb_to_hsv(R, G, B)
{ melakukan proses konversi nilai RGB ke HSV }
     r ← R / 255
     g ← G / 255
     b ← B / 255
     { pencarian statistik warna }
     cMax ← MAX(r, g, b)
     cMin ← MIN(r, g, b)
     delta ← cMax - cMin
     { Compute h }
     if delta = 0 then
          h ← 0
     else if cMax = r then
          h ← 60.0 * (((g - b) / delta) mod 6)
     else if cMax = g then
          h ← 60.0 * (((b - r) / delta) + 2)
     else
          h ← 60.0 * (((r - g) / delta) + 4)
     { Compute s }
     if cMax = 0 then
        s ← 0
     else
        s ← delta / cMax
     { Compute v }
     v ← cMax

     → (h, s, v)
```

```
function to_hsv(m)
{ melakukan konversi setiap piksel dari RGB ke HSV }
   i traversal [0, baris]
      j traversal [0, kolom]
         m_ij ← rgb_to_hsv(m_ij0, m_ij1, m_ij2)
   → m

function histogram(m, iM, jM)
{ membentuk setiap blok histogram warna (4x4) }
     sumH, sumS, sumV ← 0, 0, 0
     i traversal [iM*4, iM*4+4]
          j traversal [jM*4, jM*4+4]
               sumH += m_ij0
               sumS += m_ij1
               sumV += m_ij2
     → (sumH / 9, sumS / 9, sumV / 9)

function to_histogram(m, mHistogram)
{ mengubah matriks menjadi vektor histogram warna HSV }
     i traversal [0, 64]
          j traversal [0, 64]
               mHistogram_ij ← histogram(m, i, j)
     → mHistogram

function cosine_similarity(v1, v2)
{ mencari nilai cosinus teta antara dua buah vektor }
     dot, norm_a, norm_b ← 0, 0, 0
     i traversal [0, len(v1)]
          dot += v1[i] * v2[i]
          norm_a += math.pow(v1_i, 2)
          norm_b += math.pow(v2_i, 2)

     norm_a ← math.sqrt(norm_a)
     norm_b ← math.sqrt(norm_b)
```

```
    if norm_a = 0 or norm_b = 0 then
        cos_theta ← 0
    else
        cos_theta ← dot / (norm_a * norm_b)
    → cos_theta

function similarity(m1, m2)
{ melakukan pembandingan dua buah vektor }
    arr1 ← np.reshape(m1, -1)
    arr2 ← np.reshape(m2, -1)
    similarity ← cosine_similarity(arr1, arr2)
    → similarity
```

4.1.2 CBIR dengan Parameter Warna

```
function Penjumlahan_Matriks(Matriks1, Matriks2)
{ mengembalikan hasil penjumlahan 2 buah matriks }
    → np.add(Matriks1, Matriks2)

function Transpose(Matriks)
{ mengembalikan transpose dari matriks }
    → np.transpose(Matriks)

function RGB_to_GrayScale_Formula(R, G, B)
{ fungsi untuk mengubah RGB menjadi GrayScale }
    → 0.299*R + 0.587*G + 0.114*B

function Matriks_RGB_to_GrayScale(Mat_RGB)
{ Mengubah matriks RGB menjadi matriks GrayScale }
  height, width, vec ← Mat_RGB.shape
  Matriks_GrayScale ← np.empty((height, width))
  i traversal [height]
    j traversal [width]
```

```
        Matriks_GrayScale_ij ← RGB_to_GrayScale_Formula
                                    (Mat_RGB_ij0, Mat_RGB_ij1,
                                     Mat_RGB_ij2)
    → Matriks_GrayScale

function Matriks_GrayScale_to_Co_Occurence(Mat_GrayScale)
{ Mengubah matriks GrayScale menjadi matriks Co-Occurrence }
    Matriks_Co_Occurence = np.zeros((256, 256))
    i traversal [height]
        j traversal [width-1]
            baris ← Mat_GrayScale_ij
            kolom ← Mat_GrayScale_i(j+1)
            Matriks_Co_Occurence_ij += 1
    → Matriks_Co_Occurence

function Matrix_Normalisation(Matriks)
{ melakukan menormalisasi matriks }
    sum ← np.sum(Matriks)
    Matriks ← Matriks/sum
    → Matriks

function Texture_of_Image(Matriks)
{ mengembalikan texture sebuah matriks co-occurence yang
telah dinormalisasi }
  Vektor = np.zeros((6))
  i traversal [256]
      j traversal [256]
          Vektor_0 += Matriks_ij * (i-j)^2
          Vektor_1 += Matriks_ij / (1+(i-j)^2)
          if Matriks_ij != 0 then
              Vektor_2 += Matriks_ij * log(Matriks_ij)
          Vektor_3 += Matriks_ij * |i-j|
          Vektor_4 += Matriks_ij^2
```

```
  Vektor2 *= -1
  Vektor5 ← √Vektor4
  → Vektor

function Norm_Vektor(Vektor)
{ mengembalikan nilai norm/magnitude sebuah vektor }
    magnitude ← 0
    i traversal [6]
        magnitude += Vektori^2
    → √magnitude

function Cosine_Similarity(Vektor1, Vektor2):
{ mengembalikan nilai cosine similarity dari 2 vektor }
    Sum = 0
    i traversal [6]
       sum += Vektor1i * Vektor2i
    → sum / (Norm_Vektor(Vektor1) * Norm_Vektor(Vektor2))

function Hasil_CBIR_Tekstur(matriks):
{ mengembalikan hasil CBIR Tekstur dari gambar dalam bentuk
  Vektor }
    Matriks ← Matriks_RGB_to_GrayScale(matriks)
    Matriks ← Matriks_GrayScale_to_Co_Occurence(matriks)
    Matriks ← Penjumlahan_Matriks(matriks,
                Transpose(matriks))
    matriks ← Matrix_Normalisation(matriks)
    vektor_Texture ← Texture_of_Image(matriks)
    → vektor_Texture
```

4.2 Struktur Program

Program terdiri atas beberapa modul yaitu:

4.2.1 Main Program

Main program diberi nama server.py yang dibangun dengan menggunakan bahasa pemrograman python. Pada server.py inilah proses operasi untuk setiap gambar *dataset* dan *query* dilakukan memanfaatkan modul-modul fungsi yang ada. Untuk bisa menerima gambar dari web serta mengirimkan kembali gambar ke web, digunakan library *flask* sebagai API/framework.

4.2.2 Modul Fungsi CBIR dengan Parameter Warna

Modul ini diberi nama cbir_by_color.py. Modul ini berisi semua algoritma-algoritma (fungsi) yang dibutuhkan untuk mengekstrak fitur warna dari sebuah gambar. Di dalamnya terdapat 6 fungsi, yaitu:

- rgb_to_hsv, berisi persamaan untuk mengubah komponen RGB menjadi HSV.
- to_HSV, untuk mengubah matriks RGB menjadi matriks HSV.
- histogram, untuk mendapatkan satu blok histogram yang berasal dari piksel 4x4.
- to_histrogram, untuk mengubah matriks HSV menjadi matriks histogram.
- cosine_similarity, untuk mendapatkan nilai cosinus antara dua buah vektor.
- similarity, untuk mendapatkan hasil *similarity* dari dua buah gambar.

Untuk mempercepat proses penghitungan, digunakan library numba dengan memanfaatkan fungsi @njit. Dengan library ini, kode program akan dikompile menggunakan kompilator numba JIT, bukan interpreter python sehingga proses kompilasi menjadi lebih cepat.

4.2.3 Modul Fungsi CBIR dengan Parameter Tekstur

Modul ini diberi nama cbir_by_tekstur.py. Modul ini berisi semua algoritma-algoritma (fungsi) yang dibutuhkan untuk mengekstrak fitur tekstur dari sebuah gambar. Di dalamnya terdapat 10 fungsi, yaitu:

- Penjumlahan_Matriks, mengembalikan hasil penjumlahan dua buah matriks.
- Transpose, mengembalikan hasil transpose sebuah matriks.
- RGB_to_GrayScale_Formula, berisi persamaan yang mengubah nilai RGB menjadi nilai *grayscale*-nya.

- Matriks_RGB_to_GrayScale, mengubah matriks RGB menjadi matriks *grayscale*.
- Matriks_GrayScale_to_Co_Occurence, mengembalikan matriks *co-occurrence* yang dibentuk menggunakan matriks *grayscale*.
- Matrix_Normalisation, mengambalikan hasil normalisasi sebuah matriks.
- Texture_of_Image, mengambalikan vektor hasil ekstraksi fitur sebuah matriks *co-occurrence*.
- Norm_Vektor, mengembalikan norm sebuah vektor.
- Cosine_Similarity, untuk mendapatkan nilai cosinus antara dua buah vektor.
- Hasil_CBIR_Tekstur, proses mengubah sebuah matriks RGB menjadi vektor teksturnya menggunakan fungsi-fungsi sebelumnya.

Untuk mempercepat proses penghitungan, digunakan library numba dengan memanfaatkan fungsi @njit. Dengan library ini, kode program akan dikompile menggunakan kompilator numba JIT, bukan interpreter python sehingga proses kompilasi menjadi lebih cepat.

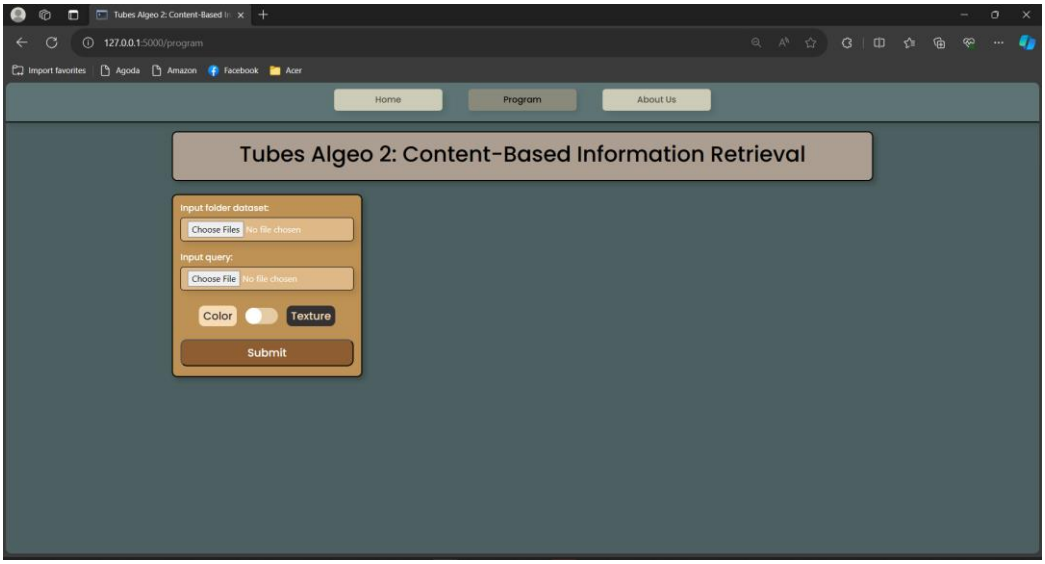
4.2.4 Modul Pengekstrak Fitur

Modul ini diberi nama feature_extractor.py. Modul ini dalam bentuk class objek yang digunakan untuk menjalankan proses ekstrak fitur dari sebuah gambar memanfaatkan modul pada 4.2.2 dan 4.2.3 yang akan digunakan pada main program.

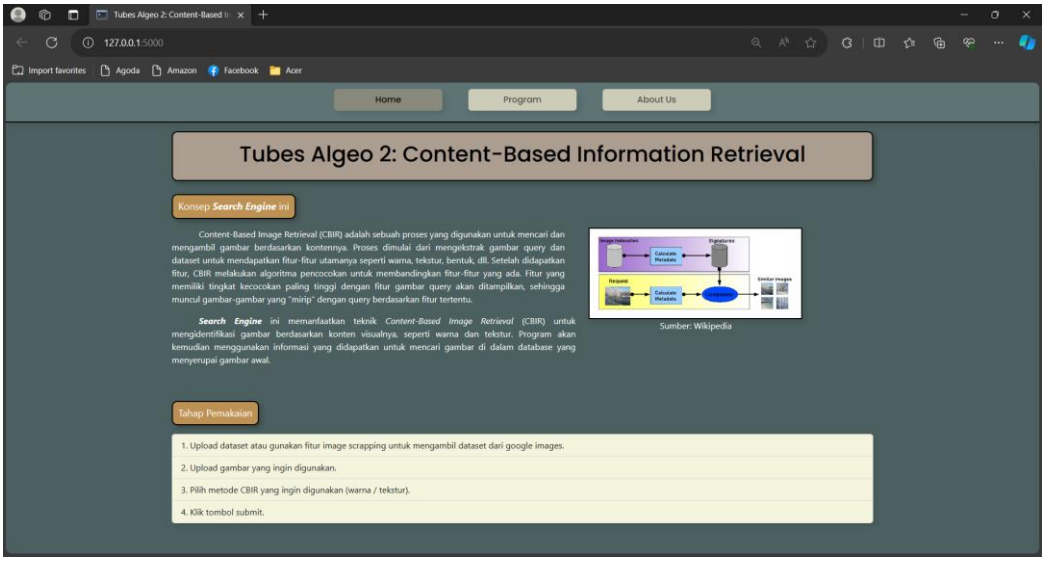
Pada bagian ekstrak fitur warna, sebuah gambar terlebih dahulu di *resize* ukurannya menjadi 256×256, yang bertujuan untuk mempermudah proses pembentukan *bin*/blok warna 4×4. Jika tidak dilakukan resize, untuk gambar-gambar yang ukurannya dimensinya bukan kelipatan 4, dapat terjadi *index out of range* saat proses pengekstrakan fitur HSV. Adapun pada ekstrak fitur tekstur tidak diperlukan proses *resize* sebuah gambar.

4.2.5 Modul Pembangunan Web

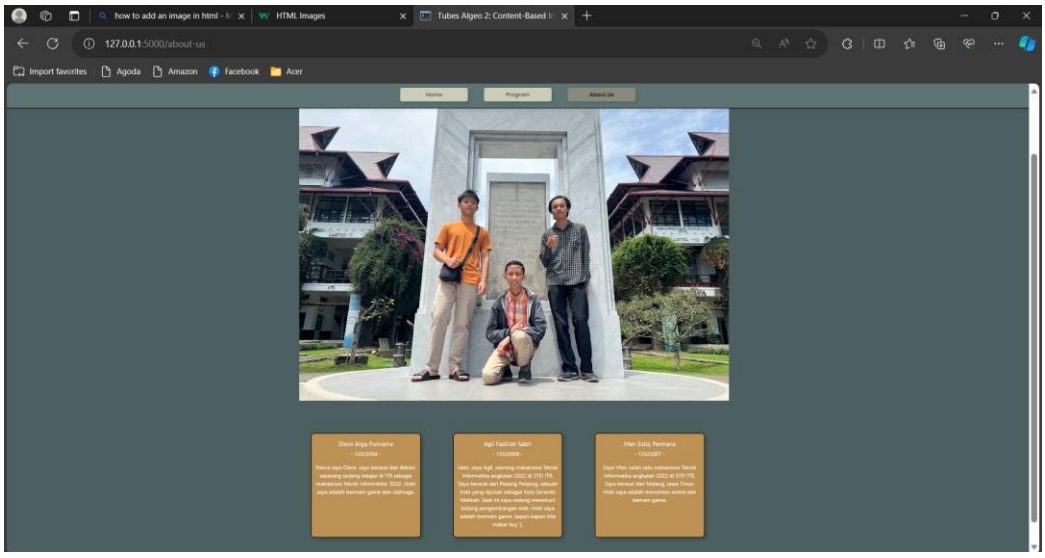
Dibangun dengan menggunakan HTML, modul ini terdiri atas 3 file, yaitu laman utama tempat proses CBIR dilakukan yang diberi nama index.html, laman beranda tempat penjelasan singkat CBIR dan cara menggunakan program yang diberi nama home.html, serta laman tentang penulis yang diberi nama about-us.html. untuk merapikan tampilan web, digunakan *cascading style sheets* yang diberi nama style.css.



Gambar 5 Laman Utama Program



Gambar 6 Laman Beranda Program



Gambar 7 Laman *About Us*

4.2.6 Database

Database menggunakan 3 buah folder, yang pertama folder tempat menyimpan semua gambar dataset (.jpg) yang diinputkan pengguna yang diberi nama dataset. Yang kedua folder tempat menyimpan hasil ekstraksi fitur (.npy) yang diberi nama feature. Didalamnya terdapat 2 subfolder, satu untuk menyimpan hasil ekstrak fitur warna dan satu lagi untuk menyimpan hasil ekstrak fitur tekstur. Folder ketiga adalah tempat menyimpan gambar query (.jpg) yang dimasukkan oleh pengguna.

4.3 Tata Cara Penggunaan Program

Sebelum menjalankan program, terlebih dahulu harus dipastikan versi python yang digunakan adalan 11.6 ke bawah, karena versi python diatas versi tersebut tidak mendukung library numba. Selain itu, pastikan bahwa dalam python yang digunakan telah terinstal library pillow, numpy, flask, pathlib, shutil, dan requests.

Adapun langkah-langkah penggunaan program yaitu:

1. Jalankan server.py di terminal dengan mengetikkan 2 hal berikut:
   >> cd src
   >> python server.py
2. Akan muncul link **http://127.0.0.1:5000**, klik kiri+ctrl link tersebut dan kemudian Anda beralih ke laman web. Disini anda telah masuk ke dalam web program CBIR.
3. Tekan tombol **Program** pada bagian atas web untuk masuk ke program utama.
4. Pada bagian input folder dataset, masukkan folder dataset yang akan dijadikan sebagai acuan dalam CBIR.
5. Selanjutnya, masukkan gambar *query* pada bagian input *query*.
6. Setelah itu, silahkan pilih metode pencarian yang hendak digunakan, apakah dengan menggunakan parameter warna (tekan *toggle button* ke kata *color*) atau menggunakan parameter tekstur (tekan *toggle button* ke kata *tekstur*).
7. Setelah itu, silahkan tekan tombol *submit* dan tunggu beberapa saat hingga hasil keluar.
8. Setelah selesai, Anda akan dapat melihat hasil CBIR yang dilakukan dengan ditampilkannya gambar-gambar pada dataset yang persentase kemiripannya dengan *query* > 60%. Pada setiap gambar, akan ditampilkan nama gambar serta persentase kemiripannya. Selain itu, juga ditampilkan lama waktu proses pencarian serta total jumlah gambar yang diperoleh.

4.4 Hasil Pengeujian

4.4.1 Dataset 1

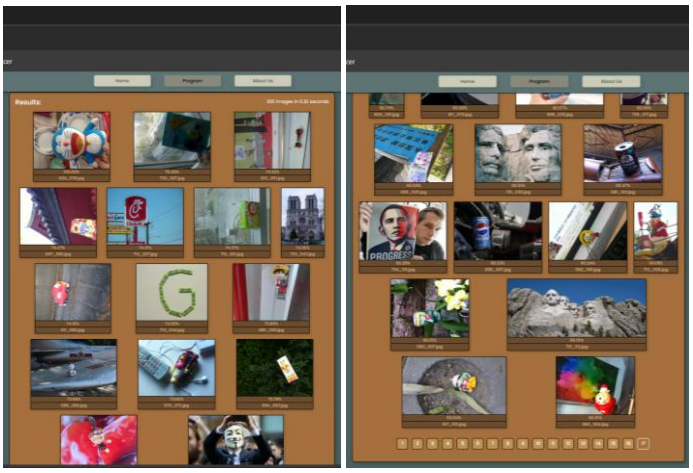
Dataset 1 berisi gambar-gambar acak, mulai dari mainan-mainan, figur karakter, benda-benda kecil, bendera, dll. Berikut beberapa hasil pengujian dengan dataset 1 sebagai sumber datasetnya:

Perkiraan waktu ekstrak fitur dataset: 44,67 detik, total gambar: 500.

- Query 1 (ada di dalam dataset)

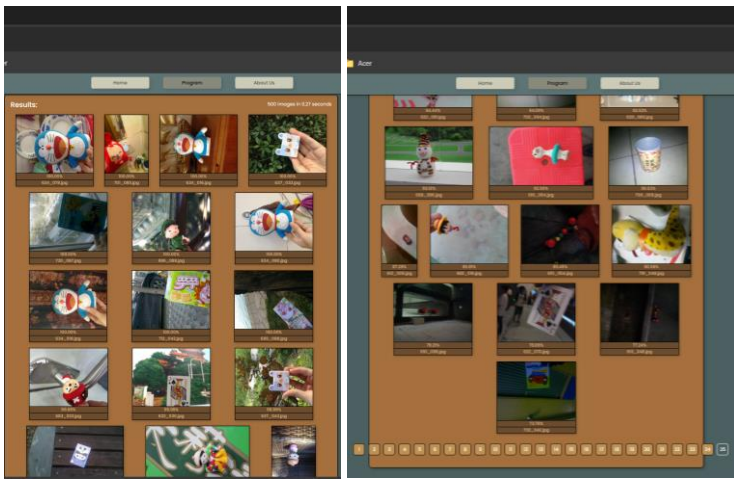
Gambar 8 *Query* 1 Dataset 1

Hasil: CBIR Warna (335 gambar dalam 0.32 detik)



Gambar 9 Hasil CBIR Warna *Query* 1 Dataset 1

Hasil : CBIR Tekstur (500 gambar dalam 0,27 detik)

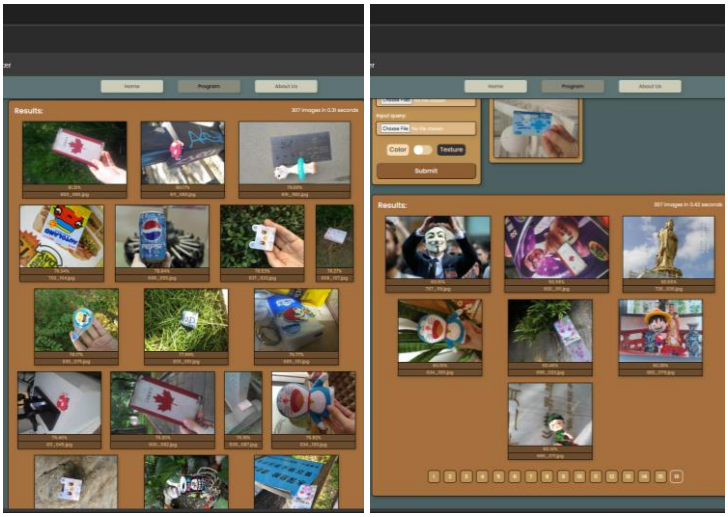


Gambar 10 Hasil CBIR Tekstur *Query* 1 Dataset 1

- Query 2 (tidak ada di dalam dataset)

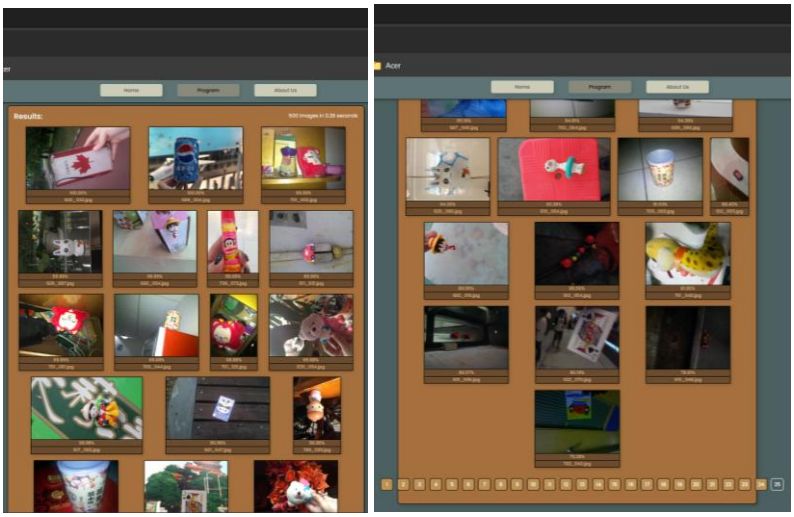
Gambar 11 *Query* 2 Dataset 1

Hasil: CBIR Warna (307 gambar dalam 0.31 detik)



Gambar 12 Hasil CBIR Warna *Query* 2 Dataset 1

Hasil : CBIR Tekstur (500 gambar dalam 0.26 detik)

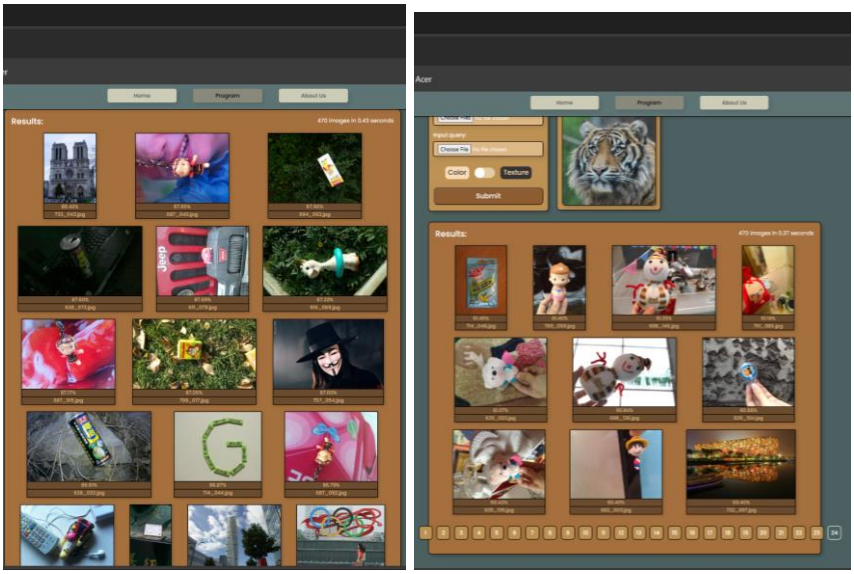


Gambar 13 Hasil CBIR Tekstur *Query* 2 Dataset 1

- Query 3 (tidak ada di dalam dataset)

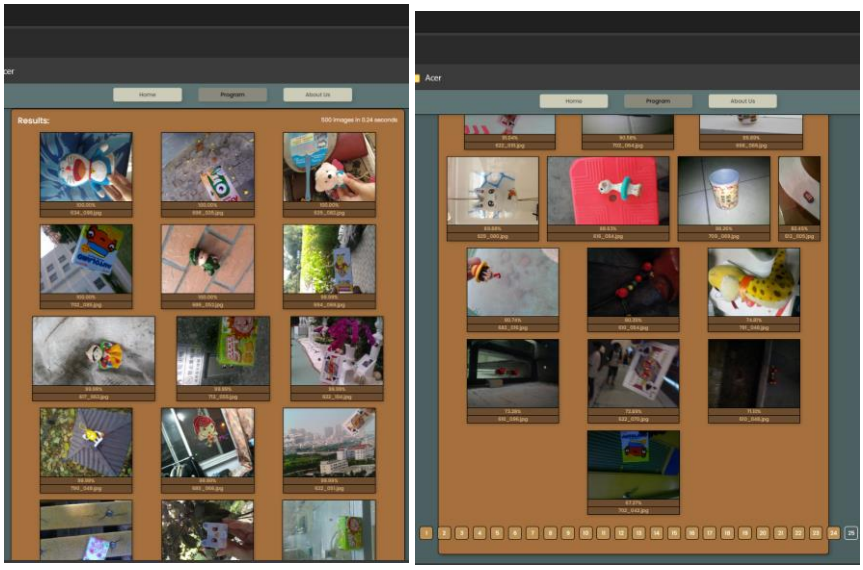
Gambar 14 *Query* 3 Dataset 1

Hasil: CBIR Warna (470 gambar dalam 0.43 detik)



Gambar 15 Hasil CBIR Warna *Query* 3 Dataset 1

Hasil : CBIR Tekstur (500 gambar dalam 0.24 detik)

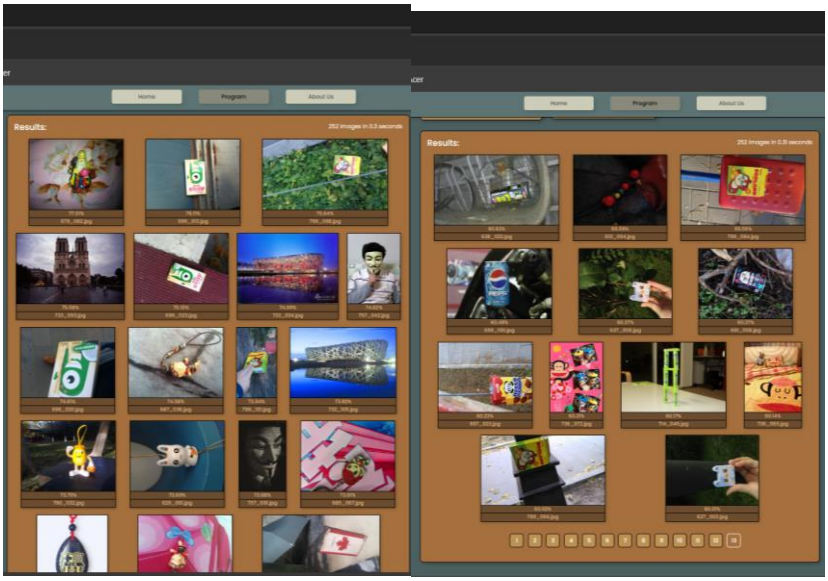


Gambar 16 Hasil CBIR Tekstur *Query* 3 Dataset 1

- Query 4 (tidak adak di dalam dataset)

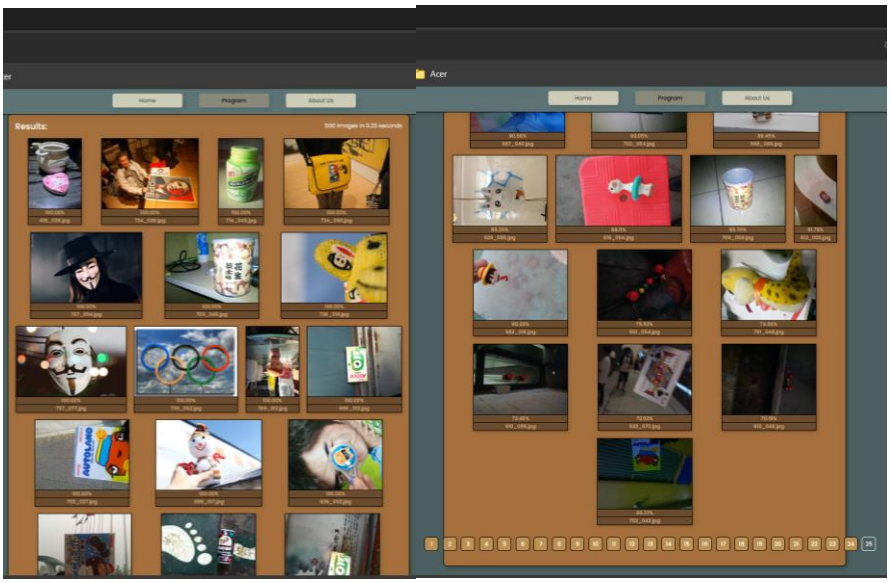
Gambar 17 *Query* 4 Dataset 1

Hasil: CBIR Warna (252 gambar dalam 0.3 detik)



Gambar 18 Hasil CBIR Warna *Query* 4 Dataset 1

Hasil : CBIR Tekstur (500 gambar dalam 0.23 detik)



Gambar 19 Hasil CBIR Tekstur *Query* 4 Dataset 1

4.4.2 Dataset 2

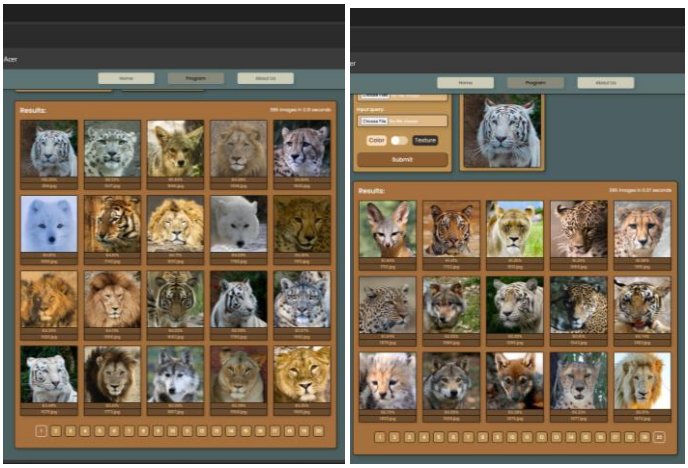
Dataset 2 berisi gambar-gambar kucing-kucing besar (singa, harimau, cheetah, dll), rubah, dan serigala. Berikut beberapa hasil pengujian dengan dataset 2 sebagai sumber datasetnya:

Perkiraan waktu ekstrak fitur dataset: 23,65 detik, total gambar: 500.

- Query 1 (ada di dalam dataset)

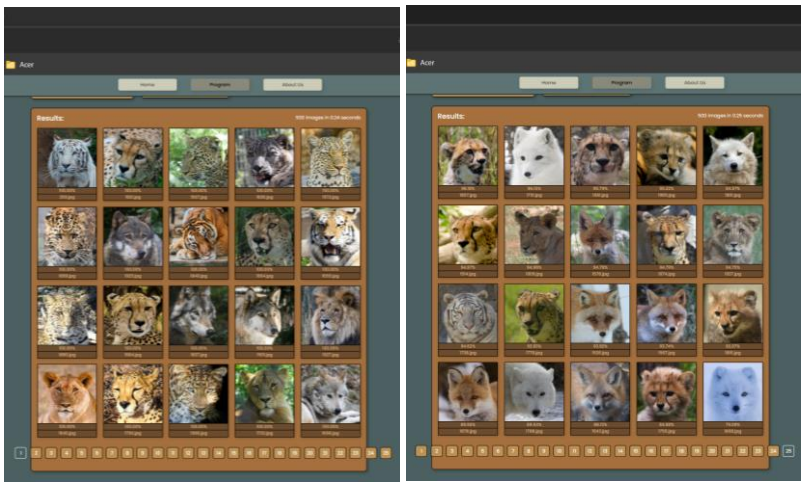
Gambar 20 *Query* 1 Dataset 2

Hasil: CBIR Warna (395 gambar dalam 0.31 detik)



Gambar 21 Hasil CBIR Warna *Query* 1 Dataset 2

Hasil : CBIR Tekstur (500 gambar dalam 0,24 detik)

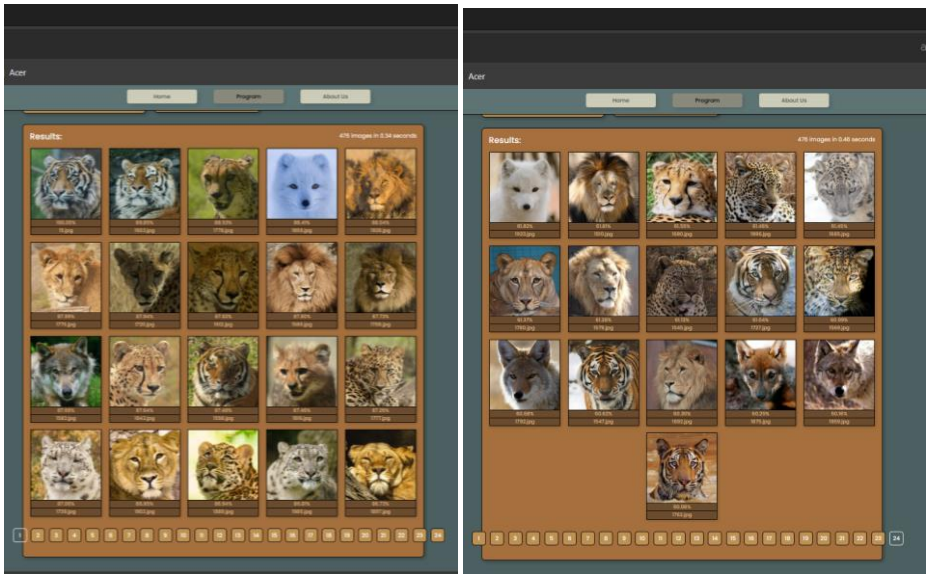


Gambar 22 Hasil CBIR Tekstur *Query* 1 Dataset 2

- Query 2 (ada di dalam dataset)

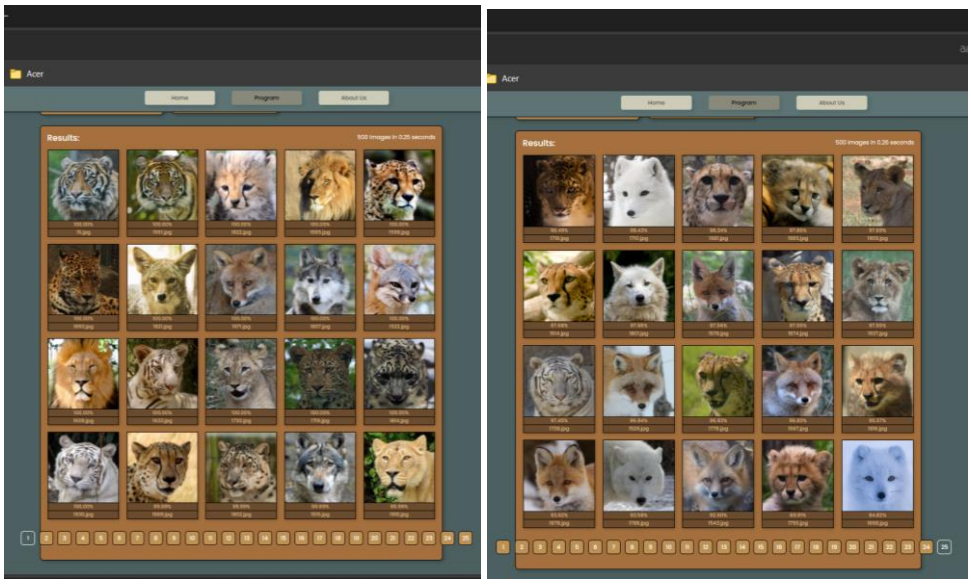
Gambar 23 *Query* 2 Dataset 2

Hasil: CBIR Warna (476 gambar dalam 0.34 detik)



Gambar 24 Hasil CBIR Warna *Query* 2 Dataset 2

Hasil : CBIR Tekstur (500 gambar dalam 0,25 detik)

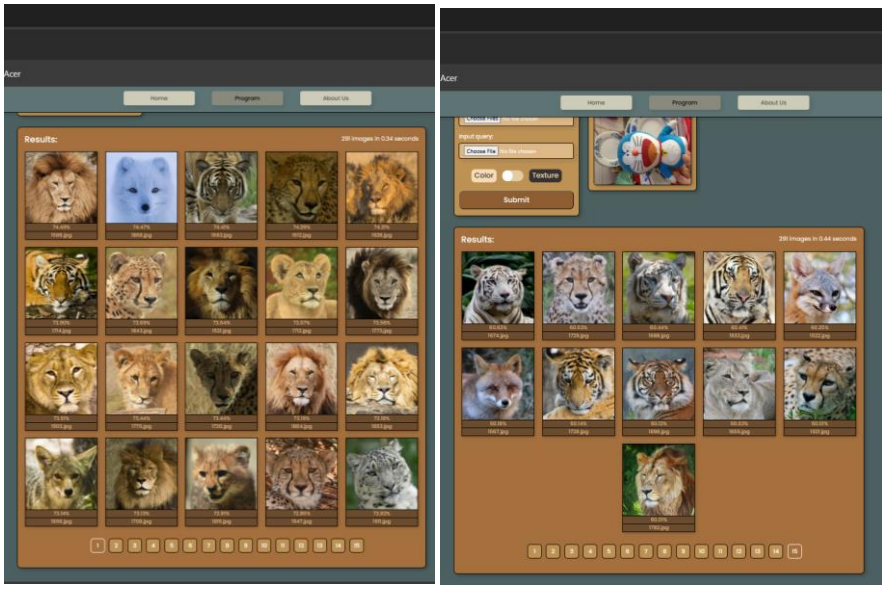


Gambar 25 Hasil CBIR Tekstur *Query* 2 Dataset 2

- Query 3 (tidak ada di dalam dataset)

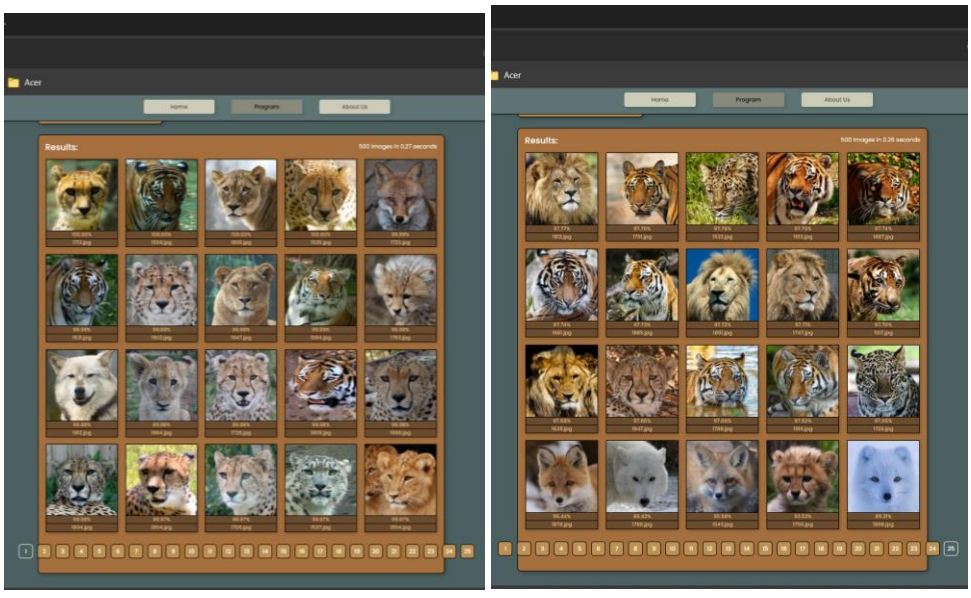
Gambar 26 *Query* 3 Dataset 2

Hasil: CBIR Warna (291 gambar dalam 0.34 detik)



Gambar 27 Hasil CBIR Warna *Query* 3 Dataset 2

Hasil : CBIR Tekstur (500 gambar dalam 0,27 detik)

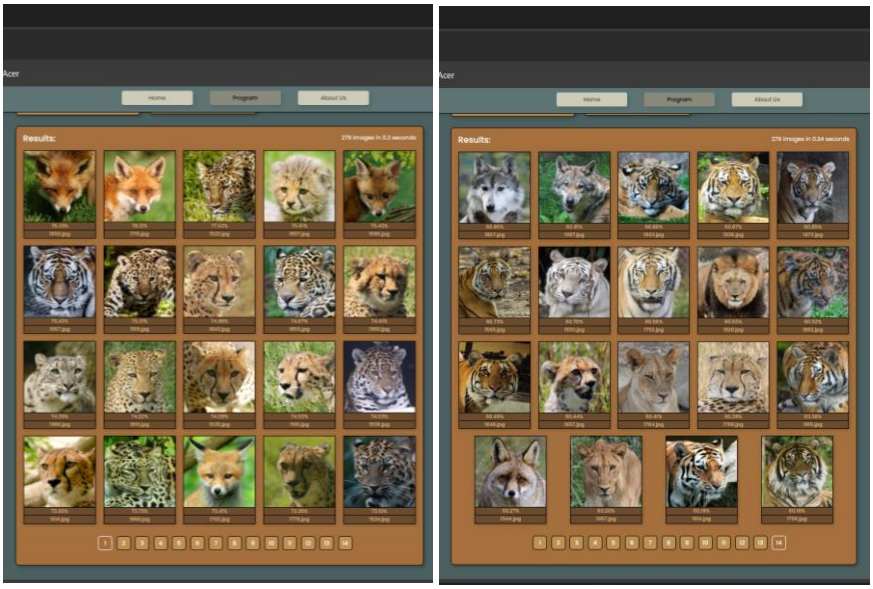


Gambar 28 Hasil CBIR Tekstur *Query* 3 Dataset 2

- Query 4 (tidak ada di dalam dataset)

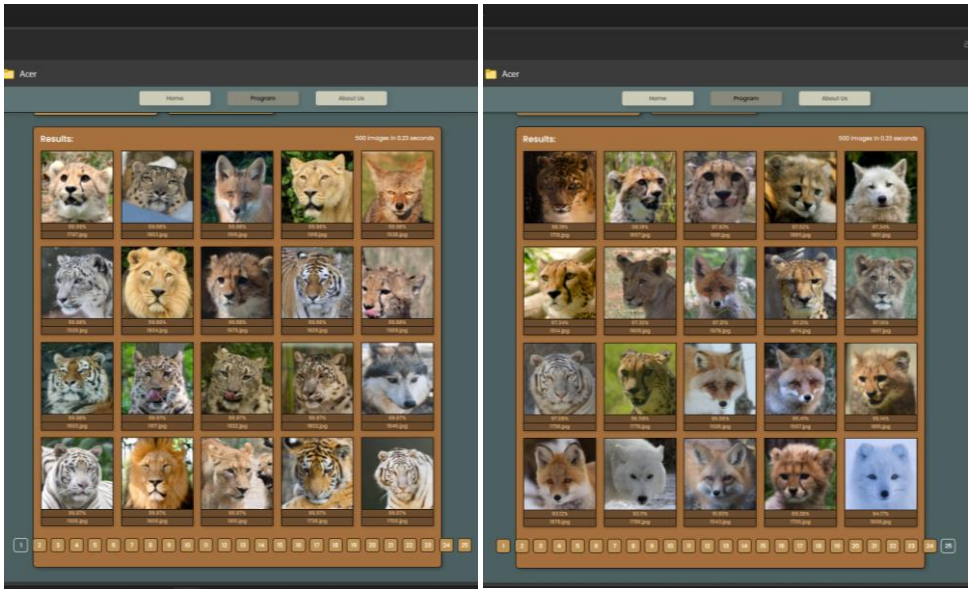
Gambar 29 *Query* 4 Dataset 2

Hasil: CBIR Warna (279 gambar dalam 0.3 detik)



Gambar 30 Hasil CBIR Warna *Query* 4 Dataset 2

Hasil : CBIR Tekstur (500 gambar dalam 0,23 detik)



Gambar 31 Hasil CBIR Tekstur *Query* 4 Dataset 2

4.4.3 Dataset 3

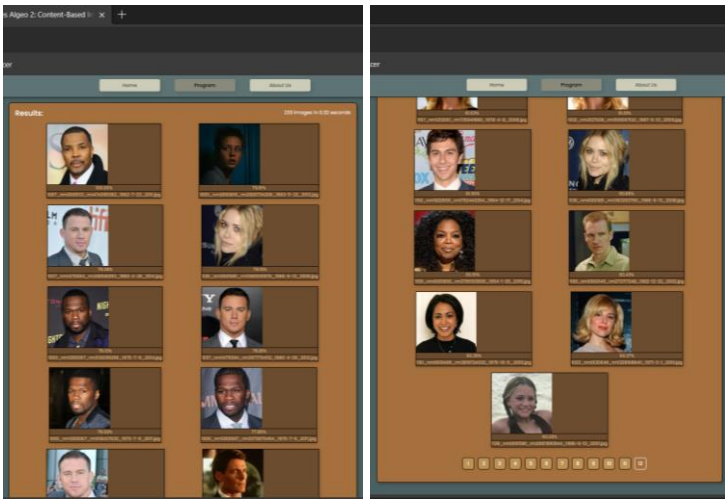
Dataset 3 berisi gambar-gambar para publik figur/artis. Berikut beberapa hasil pengujian dengan dataset 3 sebagai sumber datasetnya:

Perkiraan waktu ekstrak fitur dataset: 23,65 detik, total gambar: 500.

- Query 1 (ada di dalam dataset)

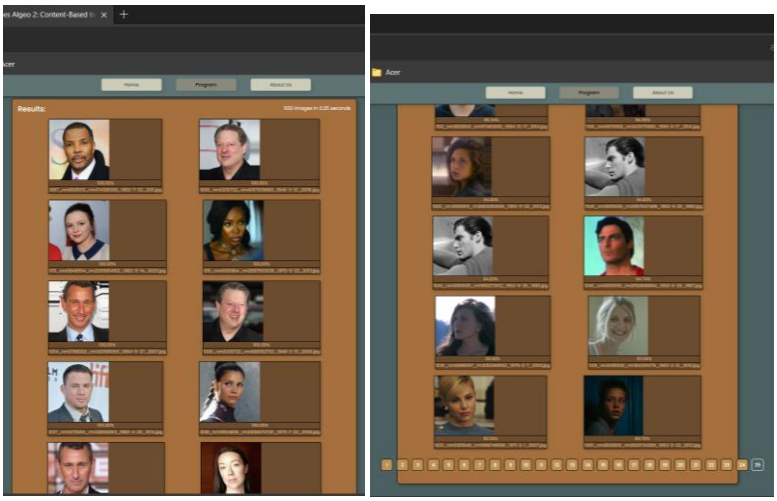
Gambar 32 *Query* 1 Dataset 3

Hasil: CBIR Warna (233 gambar dalam 0,32 detik)



Gambar 33 Hasil CBIR Warna *Query* 1 Dataset 3

Hasil : CBIR Tekstur (500 gambar dalam 0,25 detik)

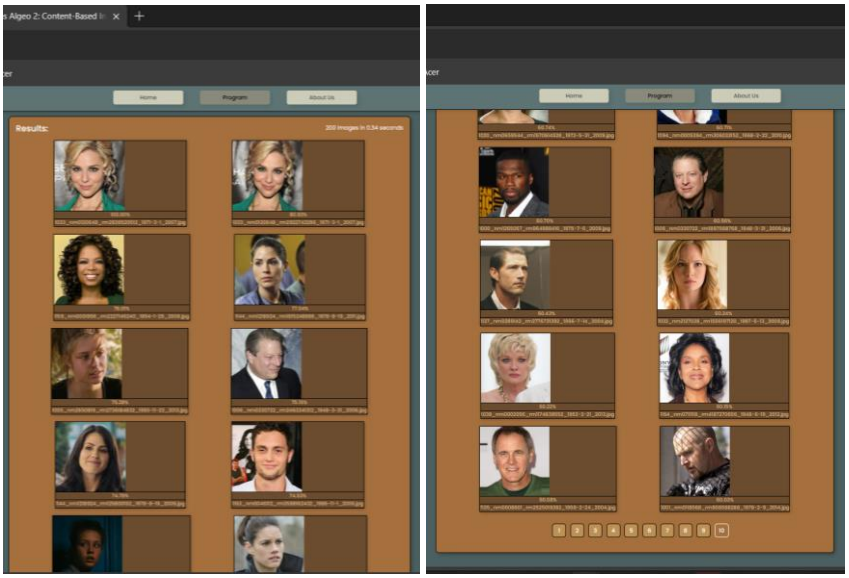


Gambar 34 Hasil CBIR Tekstur *Query* 1 Dataset 3

- Query 2 (ada di dalam dataset)

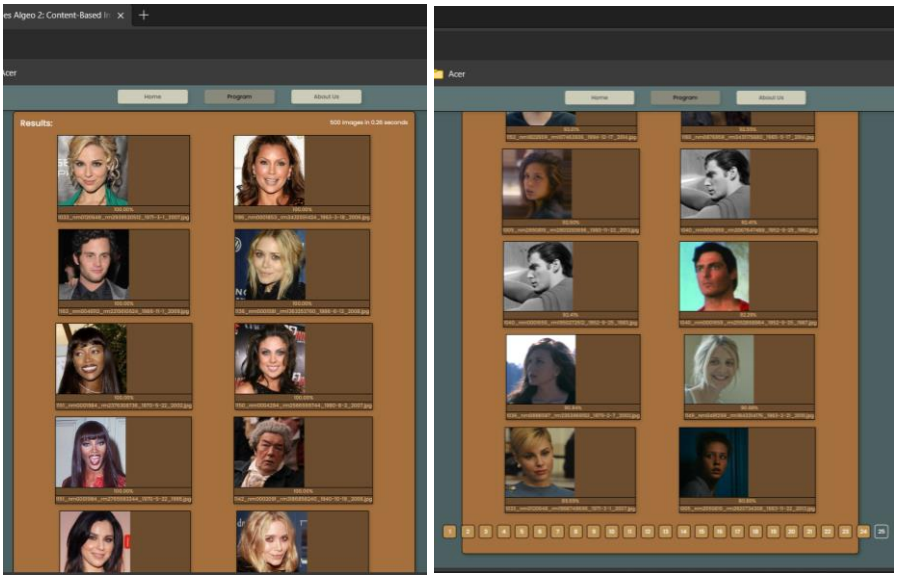
Gambar 35 *Query* 2 Dataset 3

Hasil: CBIR Warna (200 gambar dalam 0.34 detik)



Gambar 36 Hasil CBIR Warna *Query* 2 Dataset 3

Hasil : CBIR Tekstur (500 gambar dalam 0,26 detik)

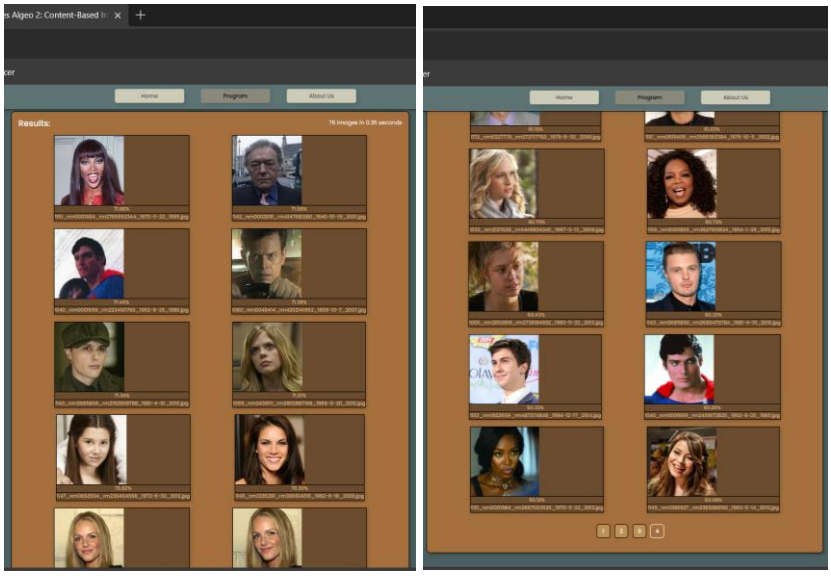


Gambar 37 Hasil CBIR Tekstur *Query* 2 Dataset 3

- Query 3 (tidak ada di dalam dataset)

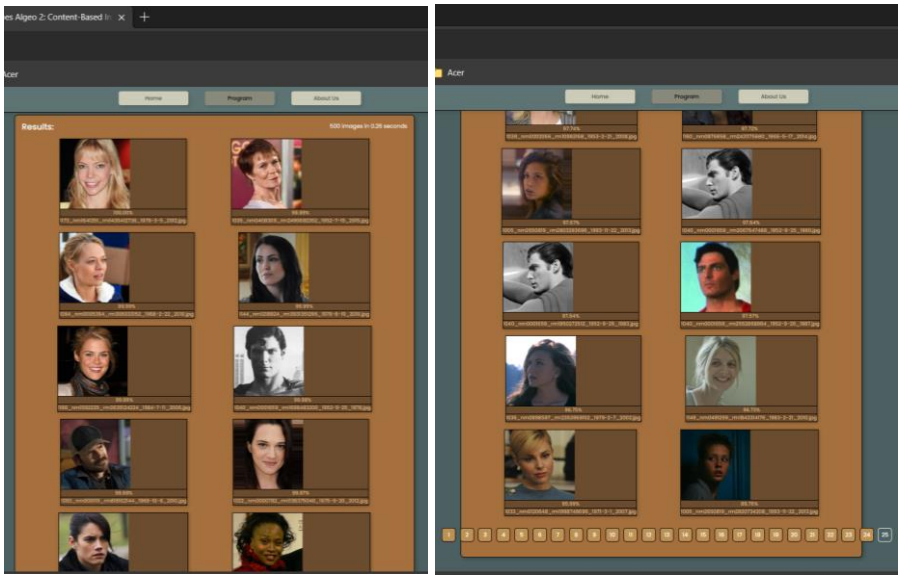
Gambar 38 *Query* 3 Dataset 3

Hasil: CBIR Warna (76 gambar dalam 0,35 detik)



Gambar 39 Hasil CBIR Warna *Query* 3 Dataset 3

Hasil : CBIR Tekstur (500 gambar dalam 0,26 detik)

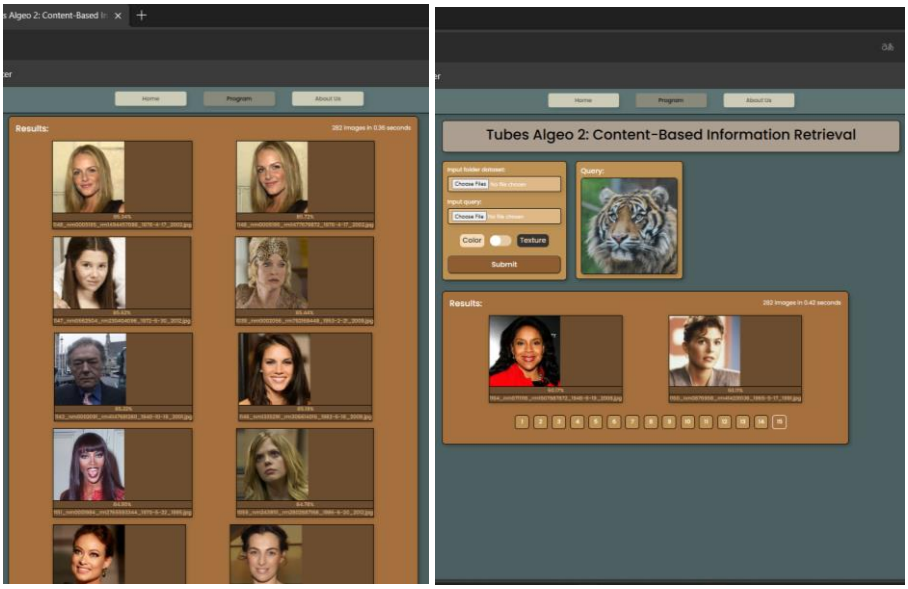


Gambar 40 Hasil CBIR Tekstur *Query* 3 Dataset 3

- Query 4 (tidak ada di dalam dataset)

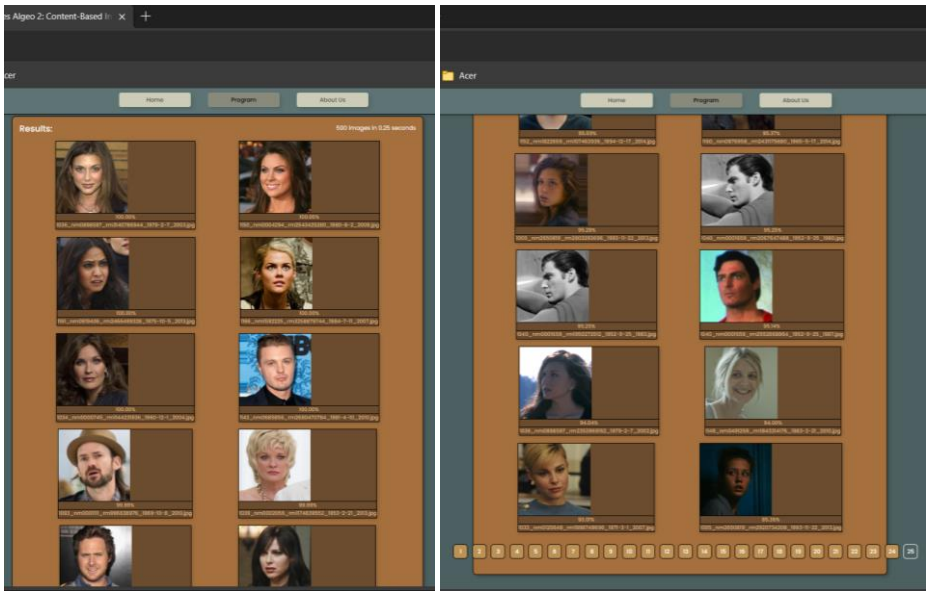
Gambar 41 *Query* 4 Dataset 3

Hasil: CBIR Warna (282 gambar dalam 0.36 detik)



Gambar 42 Hasil CBIR Warna *Query* 4 Dataset 3

Hasil : CBIR Tekstur (500 gambar dalam 0,25 detik)



Gambar 43 Hasil CBIR Tekstur *Query* 4 Dataset 3

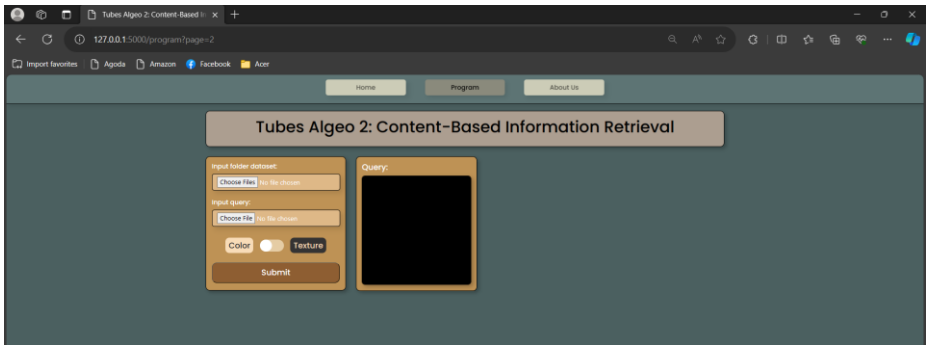
4.4.4 *Query* Unik dengan Dataset 1

Dengan menggunakan kembali gambar-gambar pada dataset 1, akan dilakukan pengujian untuk beberapa *query* berikut ini:

- Query 1 (tidak ada di dalam dataset)

Gambar 44 *Query* Warna Hitam

Hasil: CBIR Warna dan Tekstur (0 gambar dalam (?) detik)

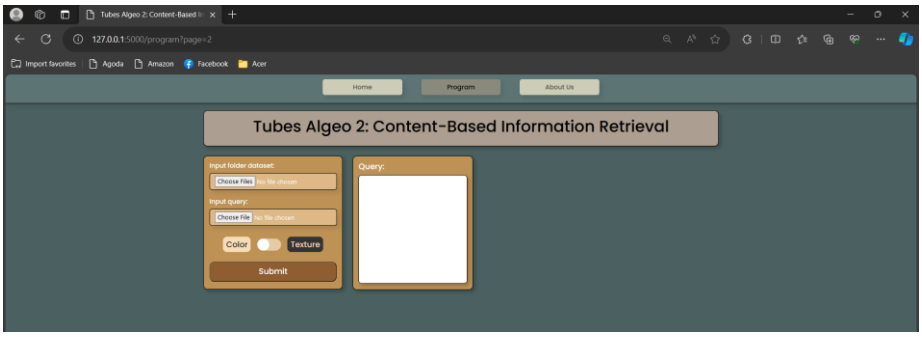


Gambar 45 Hasil CBIR Warna dan Tekstur *Query* 41

- Query 2 (tidak ada di dalam dataset)

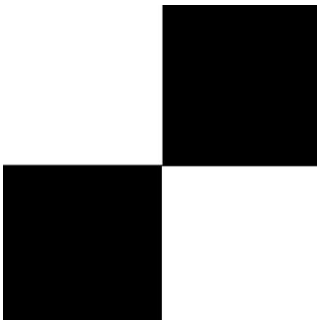
Gambar 46 *Query* Warna Putih

Hasil: CBIR Warna dan Tekstur (0 gambar dalam (?) detik)



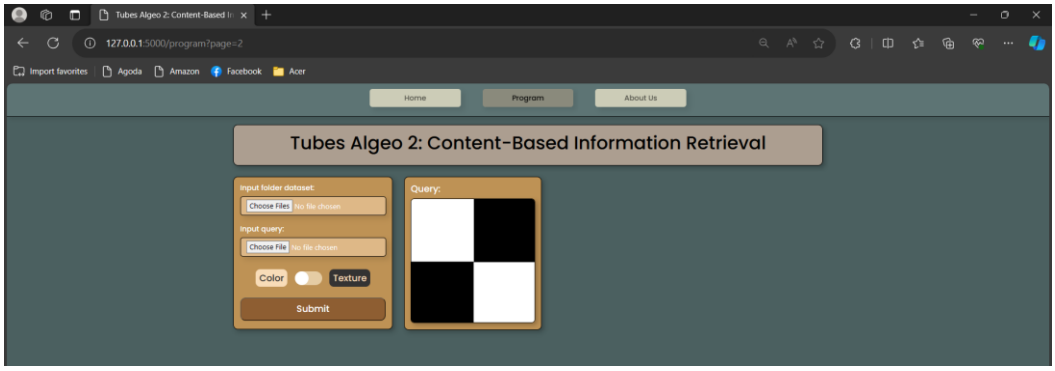
Gambar 47 Hasil CBIR Warna dan Tekstur *Query* 42

- Query 3 (tidak ada di dalam dataset)



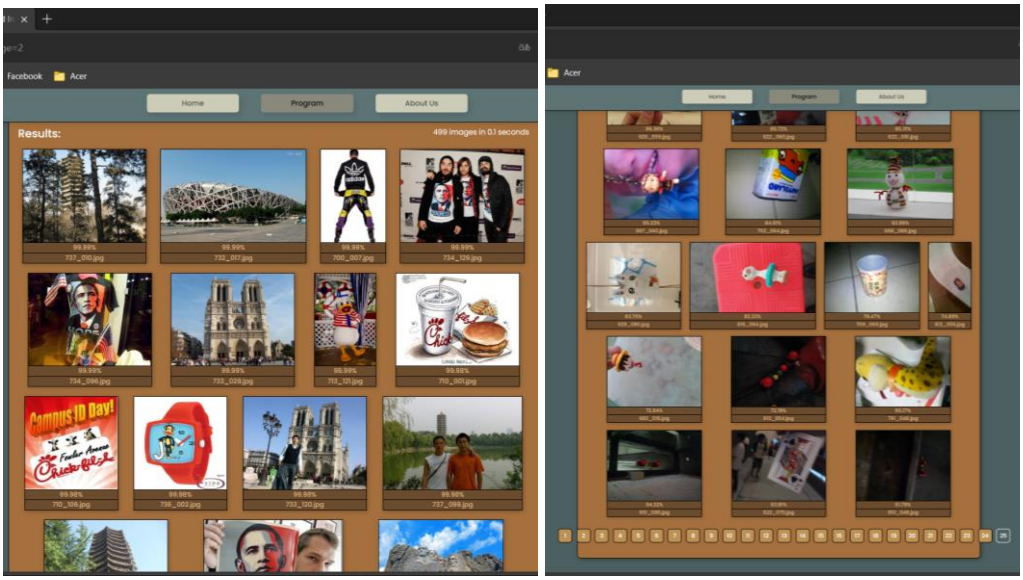
Gambar 48 *Query* Warna Hitam-Putih

Hasil: CBIR Warna (0 gambar dalam (?) detik)



Gambar 49 Hasil CBIR Warna *Query* 44

Hasil : CBIR Tekstur (499 gambar dalam 0,1 detik)

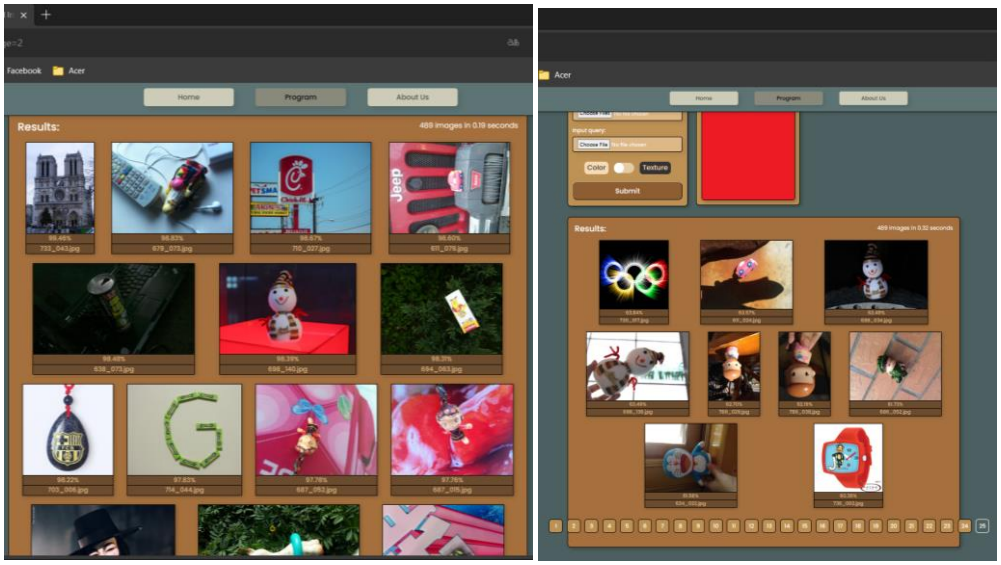


Gambar 50 Hasil CBIR Tekstur *Query* 44

- Query 4 (tidak ada di dalam dataset)

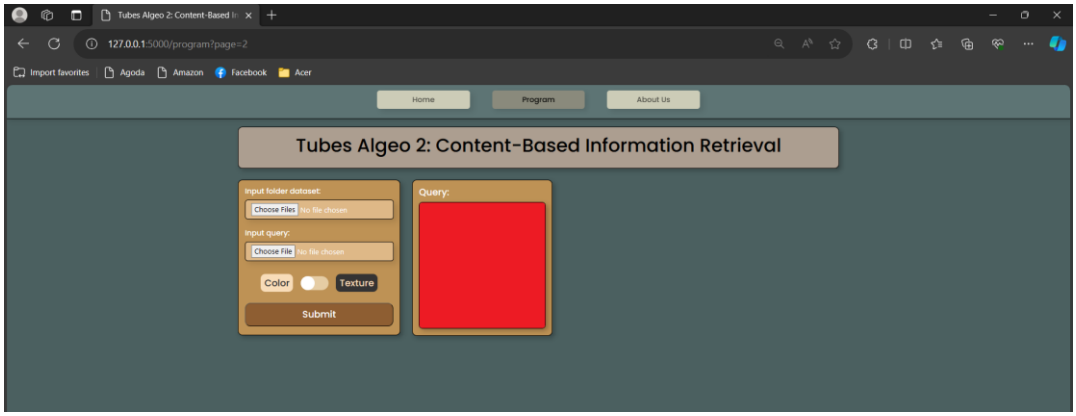
Gambar 51 *Query* Warna Merah

Hasil: CBIR Warna (489 gambar dalam 0.19 detik)



Gambar 52 Hasil CBIR Warna *Query* 47

Hasil : CBIR Tekstur (0 gambar dalam (?) detik).

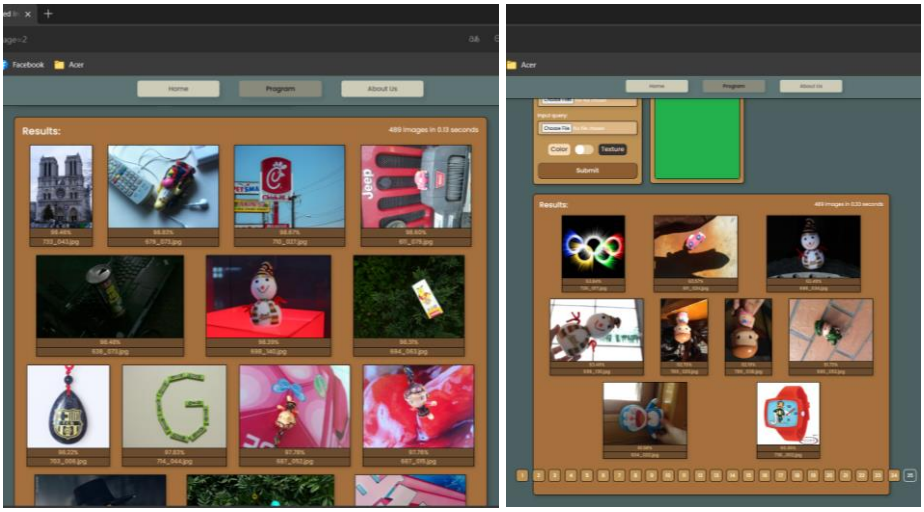


Gambar 53 Hasil CBIR Tekstur *Query* 47

- Query 5 (tidak ada di dalam dataset)

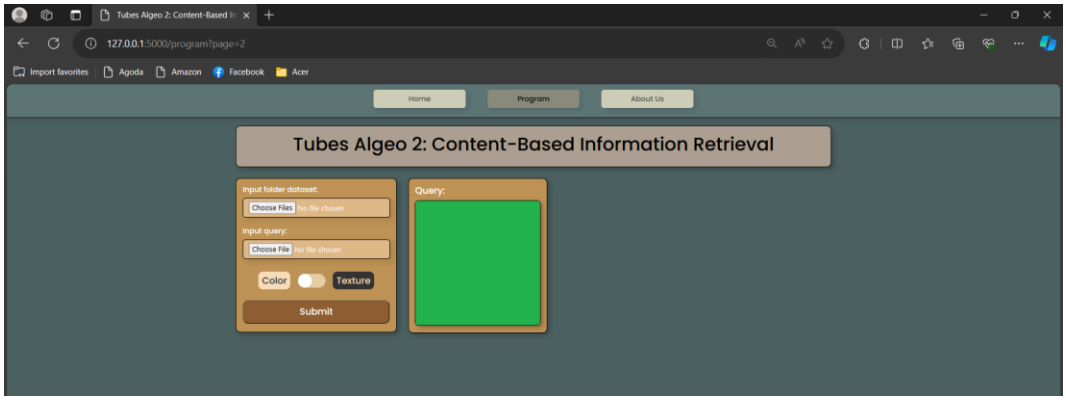
Gambar 54 *Query* Warna Hijau

Hasil: CBIR Warna (489 gambar dalam 0.13 detik)



Gambar 55 Hasil CBIR Warna *Query* 50

Hasil : CBIR Tekstur (0 gambar dalam (?) detik)

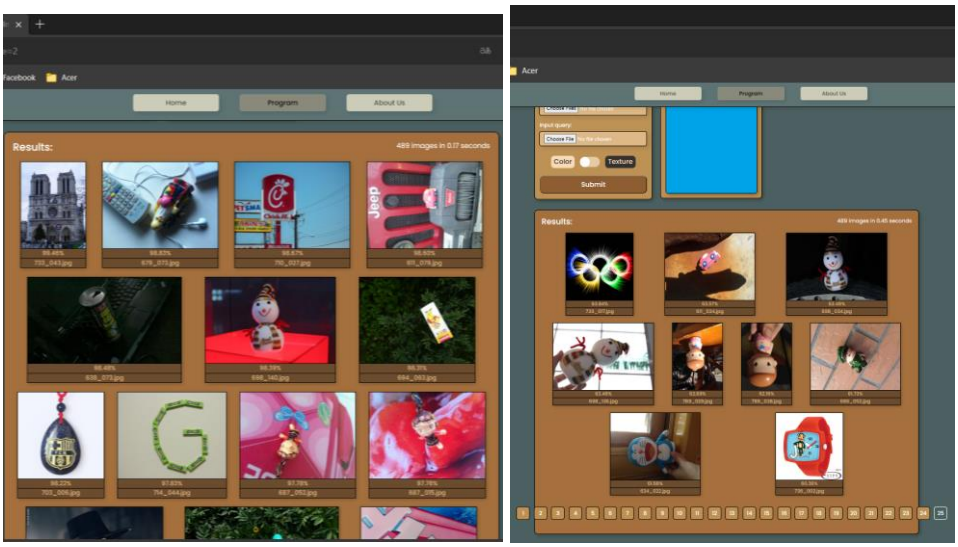


Gambar 56 Hasil CBIR Tekstur *Query* 50

- Query 6 (tidak ada di dalam dataset)

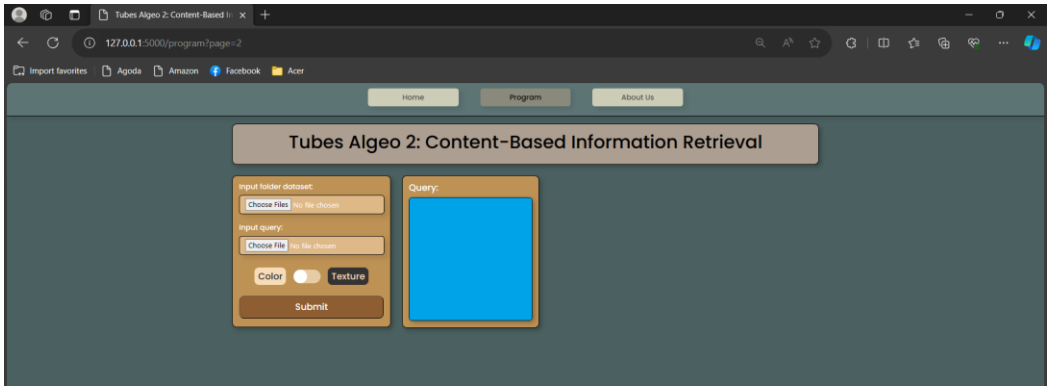
Gambar 57 *Query* Warna Biru

Hasil: CBIR Warna (489 gambar dalam 0.17 detik)



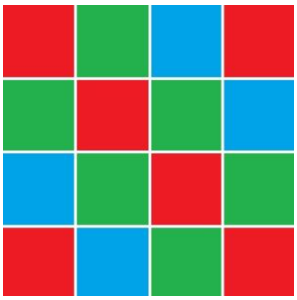
Gambar 58 Hasil CBIR Warna *Query* 53

Hasil : CBIR Tekstur (0 gambar dalam (?) detik)



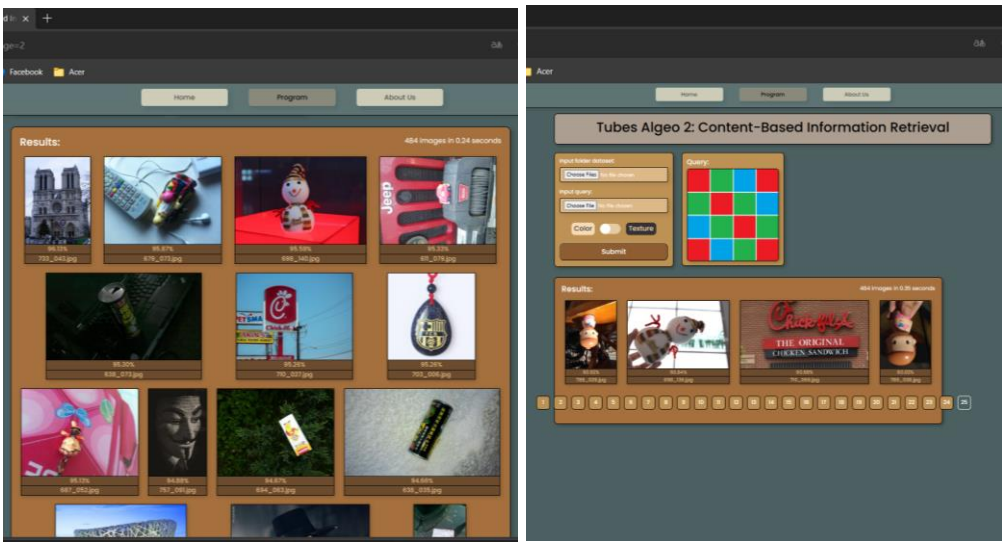
Gambar 59 Hasil CBIR Tekstur *Query* 53

- Query 7 (tidak ada di dalam dataset)



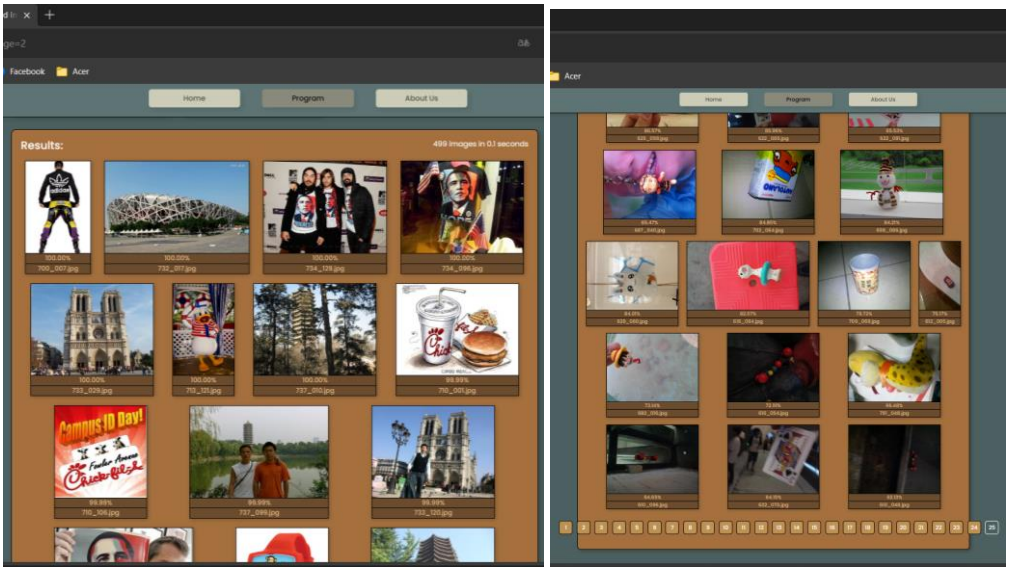
Gambar 60 *Query* Warna RGB

Hasil: CBIR Warna (489 gambar dalam 0.17 detik)



Gambar 61 Hasil CBIR Warna *Query* 56

Hasil : CBIR Tekstur (0 gambar dalam (?) detik)



Gambar 62 Hasil CBIR Tekstur *Query* 56

4.5 Analisis Hasil

Dari hasil pengujian yang dilakukan pada bagian 4.4.1, 4.4.2, dan 4.4.3, terlihat bahwa hasil pencarian menggunakan metode parameter warna selalu menghasilkan hasil pencarian gambar dengan persentase kemiripan yang bervariasi diantara angka 60% hingga 100% (hanya gambar yang ada di dataset yang akan memiliki persentase 100%) dengan jumlah gambar selalu di bawah 500. Sedangkan ketika digunakan metode parameter tekstur, selalu dihasilkan hasil pencarian dengan total gambar 500 buah hasil dengan persentase kemiripan yang sangat tinggi (diatas 90%), bahkan bisa terdapat lebih dari 1 gambar dengan persentase kemiripan 100%. Hanya ada sedikit sekali gambar yang persentase kemiripannya dibawah 90%. Hal ini terjadi karena pada metode parameter warna, vektor histogram HSV yang dihasilkan sangat bervariasi nilainya antara yang satu dengan yang lainnya, sehingga nilai *cosine similarity*-nya juga bervariasi. Sedangkan pada metode parameter tekstur, terdapat 1 komponen nilai vektor, yaitu *contrast*, yang sangat besar nilainya (ratusan, bahkan dapat mencapai ribuan), sedangkan 5 komponen sisanya sering dibawah 100, bahkan dibawah 10. Hal ini menyebabkan 1 komponen tersebut sangat mendominasi, sehingga jika dua vektor digambarkan ke ruang 3 dimensi (jika diambil 3 komponen yang wajib saja, komponen bonus diabaikan untuk kemudahan visualisasi), kedua vektor tersebut nyaris berhimpit. Hal ini berakibat pada hasil *cosine similarity* yang tinggi untuk banyak gambar.

Walaupun begitu, pada beberapa hasil pencarian, terkadang parameter tekstur menghasilkan hasil pencarian yang baik, seperti terlihat pada bagian 4.4.1 *Query* 1 (Gambar 8), dimana parameter tekstur memberikan 4 gambar yang mirip dengan *query*, sedangkan parameter warna hanya memberikan 1 gambar yang mirip dengan *query*. Di lain contoh, pada bagian 4.4.3 *Query* 2 (Gambar 35), pada parameter warna dihasilkan dua buah gambar yang mirip dengan *query*, sedangkan pada parameter tekstur, hanya dihasilkan 1 gambar yang mirip dengan *query*. Namun secara umum, baik parameter warna maupun tekstur memberikan hasil yang kurang mirip dengan *query* yang diberikan. Parameter warna sering menghasilkan gambar yang distribusi warnanya mirip dengan distribusi warna pada *query*, walau mungkin bentuk objek di dalamnya tidak mirip. Sedangkan parameter tekstur sering menghasilkan gambar dengan ciri khas pola yang mirip dengan pola pada *query*, walau mungkin warna objek di dalamnya sangat berbeda.

Namun hal unik terjadi ketika dilakukan pengujian dengan warna yang polos, seperti pada bagian 4.4.4. Pada *query* dengan warna hitam saja (Gambar 44) atau putih saja (Gambar 46), sama sekali tidak diperoleh hasil pencarian, baik dengan parameter warna maupun tekstur.

Hal ini kemungkinan terjadi karena pada kedua *query* tersebut, fitur warna memberikan hasil minimal untuk warna hitam dan hasil maksimal untuk warna putih, sehingga ketika dibandingkan dengan gambar lainnya memberikan hasil yang sangat rendah. Selain itu, gambar polos yang hanya memiliki 1 warna saja tidak memiliki pola apapun sehingga tidak terdapat fitur tekstur di dalamnya (seoalah-olah seperti lantai yang sangat halus). Hal ini berakibat pada vektor tekstur yang dihasilkan adalah vektor **0.** Namun jika kedua warna digabungkan (gambar 48), vektor tekstur mulai diperoleh sehingga terdapat hasil pencarian pada CBIR dengan parameter tekstur, walau dengan parameter warna masih tetap tidak menghasilkam apa-apa.

Beralih ke gambar merah (Gambar 51), hijau (Gambar 54), dan biru (Gambar 57) yang polos. Seperti sebelumnya, gambar polos akan menghasilkan vektor tekstur **0**, sehingga parameter tekstur tidak memberikan hasil apapun. Namun di sini parameter warna mulai memberikan hasil pencarian, walau untuk ketiga warna tersebut, sama-sama menghasilkan hasil pencarian yang sama persis. Hal ini terjadi karena walau warna RGB ketiga warna tersebut berbeda, tetapi ketika dikonversi menjadi HSV, nilainya akan sama. adapun ketika ketiga gambar tersebut digabungkan (gambar 60), baik parameter warna dan tekstur sama-sama menghasilkan hasil pencarian. Pada gambar 60, mulai terdapat tekstur di dalamnya karena terdapat perbedaan warna di antara piksel-pikselnya.

BAB V
KESIMPULAN

5.1 Simpulan

CBIR (*Content-based Information Retrieval*) adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses CBIR pada gambar dilakukan dengan mengekstrak fitur-fitur utama dari gambar seperti warna, tekstur, bentuk, dan lain-lain yang digunakan sebagai data untuk diproses dan dilakukan pencocokan satu sama lain agar diperoleh gambar yang paling cocok dengan gambar *query*. CBIR merupakan salah satu aplikasi materi aljabar linear, dimana data fitur yang diperoleh digunakan sebagai vektor yang kemudian dihitung kemiripan vektornya satu sama lain, yaitu berdasarkan sudut yang dibentuk oleh kedua vektor yang dibandingkan. Dalam Tugas Besar 2 ini, kami mengimplementasikan salah dua dari parameter CBIR yang paling kerap digunakan, yaitu parameter warna dan tekstur.

Program yang kami implementasikan menggunakan framework Flask sebagai API yang menghubungkan antara algoritma pemrosesan dengan website, sedangkan library eksternal yang kami gunakan diantaranya yaitu numpy untuk pemrosesan matriks, numba untuk mempercepat jalannya algoritma, serta pillow untuk membuka dan menyimpan gambar. Setelah dilakukan uji coba program, hasil gambar yang didapatkan melalui implementasi algoritma CBIR sayangnya masih kurang akurat terutama pada parameter tekstur yang tingkat kemiripan gambarnya selalu berada pada kisaran 90% hingga 100%. Hal ini dikarenakan tahapan pemrosesan untuk CBIR tekstur masih belum lengkap, sehingga perlu langkah tambahan seperti menerapkan *gaussian filter* pada gambar *grayscale*, pembagian hasil filter menjadi blok 4x4 untuk dihitung nilai tekstur dari masing-masing blok, dan sebagainya.

5.2 Saran

Selama pengerjaan Tugas Besar 2 ini, semua anggota kami aktif berkontribusi dalam progress pengerjaan tugas besar sehingga beban yang diemban masing-masing anggota tidak terlalu berat. Namun pada awal pengerjaan tugas besar, kami belum menyusun rencana yang pasti untuk alur pengerjaan website maupun target deadline progress yang kami harapkan, sehingga untuk selanjutnya dapat dibenahi lagi di tugas besar mata kuliah lain agar alur pengerjaan tugas besar menjadi lebih konsisten.

5.3 Komentar atau Tanggapan

Tugas Besar 2 ini sangatlah mengejutkan bagi kami pada awalnya, apalagi pada saat hari rilis tugas besar ini pada tanggal 31 Oktober 2023, karena kami diperintahkan untuk membuat sebuah website, ditambah lagi tidak ada diantara kami yang benar-benar pernah membuat sebuah web sebelumnya. Selain itu, tugas ini juga berbarengan dengan tugas-tugas mata kuliah lainnya yang semakin memaksa kami untuk bisa memanfaatkan waktu seefisien dan seefektif mungkin.

Walau begitu, dengan adanya tugas besar ini, menyadarkan kami bahwa ternyata apa yang kami pelajari selama ini sangat berguna. Sebelumnya kami bahkan tidak menyangka apa yang telah kami pelajari sejak SMA, yang kami tidak tahu kegunaannya pada saat itu, ternyata sangat berguna untuk proses pengembangan suatu sistem temu balik informasi. Kami belajar banyak dari tugas besar ini.

Selain itu, kami merasa sangat tertantang untuk bisa menyelesaikan Tugas Besar 2 ini sebaik mungkin, walau disibukkan dengan tugas-tugas lainnya yang juga sama beratnya. Seperti yang selalu kita dengungkan “Informatika berjiwa satria. Tidak pernah mengenal keluh kesah. Tugas yang melimpah bukanlah rintangan. Lautan ujian sudahlah biasa”.

5.4 Refleksi

Pada proses pengerjaan Tugas Besar 2 ini, terdapat beberapa kendala yang kami hadapi. Diantara kendala terbesar yang kami rasakan yaitu kesulitan dalam mengembangkan website serta mengintegrasikan website dengan algoritma yang telah kami buat. Hal ini dikarenakan kurangnya pemahaman kami mengenai *web development*, karena sebelum diluncurkannya tugas besar ini tidak ada seorang dari kami yang telah berpengalaman dalam membuat website. Mengingat banyaknya tugas besar lain dengan deadline yang berdekatan, pada akhirnya kami memilih untuk menggunakan framework yang paling *beginner-friendly* yaitu Flask sebagai API, serta tidak menggunakan *framework front-end* apapun. Namun di sisi lain, dengan adanya Tugas Besar 2 ini kami jadi belajar banyak mengenai pengembangan website yang tentunya akan sangat berguna di dunia kerja nanti, mengingat banyak sekali lowongan kerja yang membutuhkan *software engineer* di bidang *web development*. Secara keseluruhan, pengalaman mengerjakan Tugas Besar 2 ini adalah pengalaman yang berharga bagi kami sebagai langkah awal dalam mengeksplor dunia pengembangan web.

5.5 Ruang Perbaikan dan Pengembangan

Dikarenakan hasil gambar yang masih kurang akurat, maka hal yang dapat diperbaiki selanjutnya terutama adalah menambahkan langkah-langkah algoritma CBIR yang belum diimplementasikan, terutama pada algoritma CBIR dengan parameter tekstur. Selain itu, agar hasil algoritma dapat lebih akurat lagi maka selanjutnya dapat ditambahkan parameter tambahan seperti parameter bentuk, parameter dimensi, dan lain-lain serta user dapat memilih salah satu atau beberapa parameter sekaligus untuk memperoleh gambar yang paling sesuai berdasarkan kriteria yang diinginkan. Dalam hal implementasi kode website, hal yang dapat ditingkatkan lagi yaitu melakukan *refactor* pada *source code* agar struktur kode lebih rapi dan tertata, sehingga lebih mudah untuk di-*maintain* kedepannya. Selain itu agar website dapat menjadi lebih *scalable* untuk pengembangan lebih lanjut, dapat digunakan *framework* maupun *database* yang umum menjadi standar website perusahaan seperti React, Django, SQL, MongoDB, dan sebagainya.

DAFTAR PUSTAKA

https://codingstudio.id/blog/mengenal-back-end-developer/ (Diakses pada 17 November 2023).

https://codingstudio.id/blog/pengertian-front-end-developer/ (Diakses pada 17 November 2023).

https://dewey.petra.ac.id/repository/jiunkpe/jiunkpe/s1/info/2016/jiunkpe-is-s1-2016-26411170-36750-website-chapter2.pdf (Diakses pada 18 November 2023).

https://www.domainesia.com/berita/website-adalah/ (Diakses pada 17 November 2023).

https://dspace.uii.ac.id/bitstream/handle/123456789/29158/13523270%20Dian%20Febrianto.pdf?sequence=1 (Diakses pada 17 November 2023).

https://ichi.pro/id/numba-kompilasi-jit-tapi-untuk-python-60746993518806 (Diakses pada 17 November 2023).

https://ieeexplore.ieee.org/document/6745402 (Diakses pada 18 November 2023).

https://ilmukomputer.org/wp-content/uploads/2009/10/yanuwid-cbir.pdf (Diakses pada 10 November 2023).

https://journals.tubitak.gov.tr/elektrik/vol19/iss1/8/ (Diakses pada 18 November 2023).

https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm (Diakses pada 3 November 2023).

https://numba.pydata.org/#:~:text=Numba%20translates%20Python%20functions%20to%20optimized%20machine%20code,can%20approach%20the%20speeds%20of%20C%20or%20FORTRAN (Diakses pada 17 November 2023).

https://ojs.uajy.ac.id/index.php/jbi/article/view/1077 (Diakses pada 18 November 2023).

https://prism.ucalgary.ca/server/api/core/bitstreams/8f9de234-cc94-401d-b701-f08ceee6cfdf/content (Diakses pada 10 November 2023).

https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask (Diakses pada 5 November 2023).

http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth06.pdf (Diakses pada 10 November 2023).

https://www.dropbox.com/s/yt3zrd4y9nhd3pv/PCD%20Lanjut%20Pertemuan%208%20-%20Fitur%20Tekstur.pdf?dl=0 (Diakses pada 10 November 2023).

https://www.kaggle.com/datasets/mathurinache/gpr1200-dataset (Diakses pada 18 November 2023).

https://www.kaggle.com/datasets/theaayushbajaj/cbir-dataset (Diakses pada 11 November 2023).

https://www.linuxid.net/32381/cara-menghapus-file-dan-direktori-di-python/ (Diakses pada 18 November 2023).

https://www.sciencedirect.com/science/article/pii/S0895717710005352 (Diakses pada 4 November 2023).

https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1 (Diakses pada 4 November 2023).

LAMPIRAN

Link repository:

https://github.com/DieroA/Algeo02-22006.git

Link video youtube:

https://youtu.be/F0TXIfab-6w?si=9aqw75weUarlztCr