

LAPORAN TUGAS BESAR 2
IF2123 - ALJABAR LINIER DAN GEOMETRI

Kelompok 22 “Bukan Kelompok 22”

Ahmad Mudabbir Arief 13522072

Muhammad Neo Cicero Koda 13522108

William Glory Henderson 13522113



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

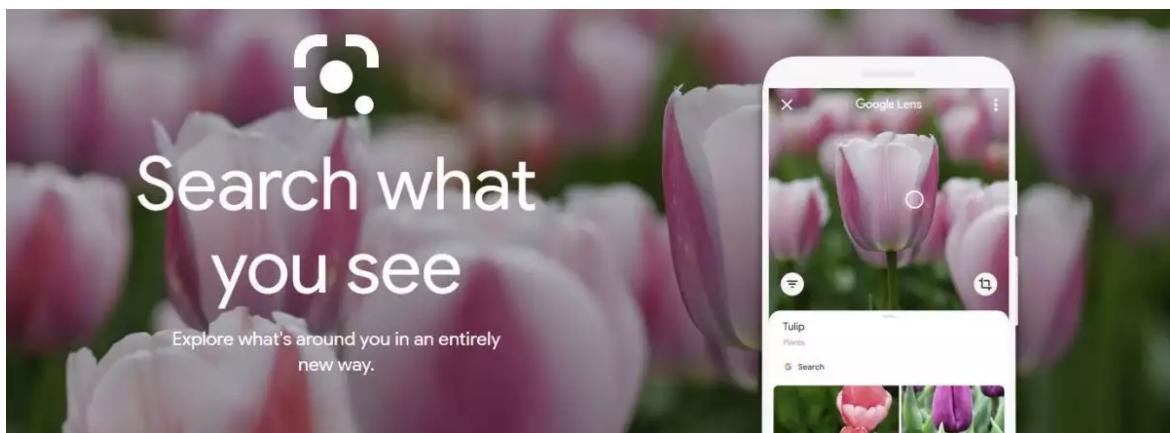
DAFTAR ISI

BAB I DESKRIPSI MASALAH	3
BAB II LANDASAN TEORI	4
1. CBIR dengan parameter warna	4
2. CBIR dengan parameter tekstur	6
3. Pengembangan Website	9
BAB III ANALISIS PEMECAHAN MASALAH	10
1. Langkah-langkah pemecahan masalah	10
2. Proses Pemetaan Masalah Menjadi Elemen-elemen pada Aljabar Geometri	11
3. Contoh Ilustrasi Kasus dan Penyelesaiannya	12
BAB IV IMPLEMENTASI DAN UJI COBA	13
1. Implementasi Program Utama	13
2. Penjelasan Struktur Program Berdasarkan Spesifikasi	18
3. Tata Cara Penggunaan Program	19
4. Hasil Pengujian	23
5. Analisis Desain Solusi Algoritma Pencarian	27
BAB V KESIMPULAN	28
1. Kesimpulan	28
2. Saran	28
3. Komentar	28
4. Refleksi	28
5. Ruang Perbaikan atau Pengembangan	29
DAFTAR PUSTAKA	30
LAMPIRAN	31

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.



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

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

BAB II

LANDASAN TEORI

Content-Based Image Retrieval (CBIR) merupakan sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan konten atau isi dari gambar tersebut. Proses tersebut dimulai dengan cara mengekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur tersebut diekstraksi dari gambar, fitur tersebut direpresentasikan sebagai vektor atau representasi numerik yang dapat dibandingkan dengan vektor gambar lain. Kemudian, CBIR memanfaatkan algoritma pencocokan untuk membandingkan vektor fitur dari gambar yang dicari dengan vektor fitur dari gambar dalam dataset. Hasil pencocokan tersebut digunakan untuk mengurutkan gambar pada kumpulan data dan menampilkan gambar yang paling mirip dengan gambar yang sedang dicari. Proses CBIR membantu pengguna mengakses dan menjelajahi koleksi gambar dengan lebih efektif karena tidak memerlukan pencarian berbasis teks atau kata kunci, tetapi mengandalkan kesamaan nilai visual antar gambar.

Dalam tugas besar ini, dibuat dua implementasi CBIR yang paling umum, yaitu:

1. CBIR dengan parameter warna

Pada Content-Based Image Retrieval (CBIR) dengan parameter warna, perbandingan dilakukan antara input dari suatu gambar dengan gambar-gambar yang terdapat dalam dataset. Pendekatan yang dilakukan adalah mengonversi representasi gambar dari format RGB ke metode histogram warna yang lebih umum.

Histogram warna menggambarkan frekuensi kemunculan berbagai warna dalam suatu ruang warna tertentu. Tujuan utama dilakukan hal tersebut adalah untuk mendistribusikan warna dari gambar. Akan tetapi, histogram warna tidak mampu mendeteksi objek spesifik dalam gambar atau memberikan informasi tentang posisi warna yang terdistribusi.

Pada dasarnya, pembentukan ruang warna dilakukan agar nilai citra menjadi beberapa rentang nilai yang lebih kecil. Hal ini bertujuan untuk menciptakan histogram warna di mana setiap interval atau rentang nilai dianggap sebagai *bin*. Perhitungan histogram warna dilakukan dengan menghitung jumlah piksel yang memiliki nilai warna

dalam setiap interval tersebut. Fitur warna meliputi histogram warna global dan histogram warna blok.

Dalam perhitungan histogram, penggunaan warna dalam format HSV (Hue, Saturation, Value) lebih dipilih karena dapat digunakan pada gambar-gambar dengan latar belakang berwarna putih, seperti pada kertas. Oleh karena itu, konversi warna dari RGB ke HSV perlu dilakukan dengan langkah sebagai berikut:

1. Nilai RGB dinormalisasikan dari rentang [0, 255] menjadi [0, 1]

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

Persamaan 2.1. Persamaan normalisasi RGB

2. Nilai C_{max} , C_{min} , dan Δ ditentukan dengan rumus:

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

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

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

Persamaan 2.2. Persamaan nilai C_{max} , C_{min} , dan Δ

3. Hasil perhitungan pada langkah 1 dan 2 digunakan untuk mendapat nilai HSV dengan rumus:

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

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

$$V = C_{max}$$

Persamaan 2.3. Persamaan nilai H, S, dan V

Nilai HSV diperoleh dan perbandingan antara gambar dari input dengan gambar-gambar pada dataset dilakukan dengan memanfaatkan *cosine similarity*

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

Persamaan 2.4. Persamaan nilai *cosinus* dua vektor

A dan B merupakan vektor dan n merupakan dimensi vektor-vektor tersebut. Semakin besar nilai hasil *cosine similarity* yang diperoleh, semakin besar pula tingkat kemiripan dua gambar.

Untuk melakukan pencarian histogram, gambar dibagi menjadi blok berukuran $n \times n$. Blok-blok yang terlalu besar dapat menghilangkan signifikansi blok tersebut. Akan tetapi, blok yang terlalu kecil dapat mengakibatkan peningkatan waktu pemrosesan. Pada CBIR dengan parameter warna ini, digunakan blok berukuran 4×4 agar pencarian blok menjadi lebih efektif. Representasi nilai dari setiap blok dapat dihasilkan dengan menghitung rata-rata nilai dalam ruang warna HSV dari blok terkait.

2. CBIR dengan parameter tekstur

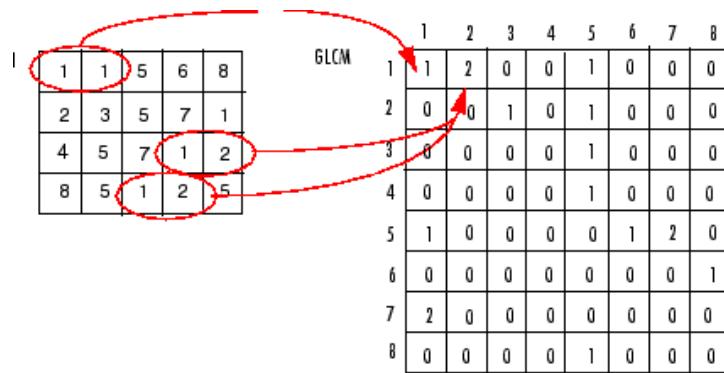
Content-Based Image Retrieval (CBIR) dengan parameter tekstur dilakukan dengan memanfaatkan *co-occurrence matrix*. Matriks ini dipilih karena kemudahan dan kecepatan pemrosesan yang dilakukan dengan matriks tersebut. Vektor yang dihasilkan juga memiliki ukuran yang lebih kecil. Jika terdapat gambar I dengan dimensi $n \times m$ piksel dan parameter *offset* ($\Delta x, \Delta y$), matriks dapat dirumuskan sebagai berikut:

Nilai i dan j digunakan untuk merepresentasikan intensitas gambar dan p dan q merupakan posisi gambar. Offset Δx dan Δy dipengaruhi oleh arah θ dan jarak ditentukan oleh persamaan berikut:

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

Persamaan 2.5. Persamaan pembuatan matriks *cooccurrence*

Nilai θ yang dapat digunakan adalah $0^\circ, 45^\circ, 90^\circ$, dan 135° . Pada tugas besar ini, nilai θ yang digunakan adalah 0° .



Gambar 2.1. Contoh Pembuatan Matriks *Cooccurrence*

Setelah didapatkan matriks *co-occurrence, symmetric matrix* dapat diperoleh dengan menjumlahkan *co-occurrence matrix* dengan hasil *transpose*-nya. Setelah itu, *matrix normalization* diperoleh dengan persamaan:

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

Persamaan 2.6. Persamaan normalisasi matriks

Berikut adalah langkah-langkah penerapan CBIR dengan parameter tekstur:

1. Karena warna tidak dipentingkan dalam penentuan tekstur, gambar berwarna dikonversi menjadi *grayscale* terlebih dahulu. Konversi RGB ke *grayscale* dapat dilakukan dengan rumus:

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

Persamaan 2.7. Persamaan konversi RGB ke *grayscale*

2. Nilai grayscale dikuantifikasi dengan membuat matriks berukuran 256 x 256 (diperoleh dari rentang nilai grayscale). Pengelihatan manusia menilai tingkat kemiripan antara dua gambar berdasarkan kekasaran tekstur gambar tersebut.
3. Dari *co-occurrence matrix* dapat diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Dapat juga diperoleh komponen lain berupa *energy*, *dissimilarity*, dan ASM. Komponen-komponen tersebut dapat diperoleh dengan menggunakan rumus:

Contrast:

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

Homogeneity :

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

Entropy :

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

Dissimilarity :

$$\sum_{i,j=0}^{levels-1} P_{i,j} |i - j|$$

ASM :

$$\sum_{i,j=0}^{levels-1} P_{i,j}^2$$

Energy :

$$: \sqrt{ASM}$$

Persamaan 2.8. Persamaan nilai *contrast*, *homogeneity*, *entropy*, *energy*, *dissimilarity*, dan ASM

Keterangan : P adalah matriks *co-occurrence*

Ketiga komponen tersebut akan membentuk suatu vektor yang dapat digunakan untuk menghitung tingkat kemiripan antara dua gambar.

4. Tingkat kemiripan dua gambar dihitung dengan menggunakan *cosine similarity* seperti pada persamaan 2.4.

Pada rumus ini, A dan B merupakan dua vektor dari dua gambar. Semakin besar hasil *Cosine Similarity* kedua vektor, semakin tinggi tingkat kemiripan kedua gambar tersebut.

3. Pengembangan Website

Proses pengembangan website *Reverse Image Search* berbasis *Content-Based Image Retrieval* (CBIR) adalah salah satu bentuk aplikasi atau pemanfaatan konsep Aljabar Linear dan Geometri. Website ini menggunakan *framework* React js dan Next js untuk *front-end* dan memakai Flask untuk *back-end*. Pengembangan dalam website ini menggunakan algoritma CBIR yang berbasis Python untuk melakukan pencarian gambar berdasarkan warna dan tekstur dari sebuah gambar yang di upload. Konsep dari aljabar linear akan dimasukkan ke dalam algoritma CBIR dan website ini memakai API untuk menghubungkan semua komponennya agar bisa menjadi sebuah website yang dapat dipakai. Desain web ini memiliki konsep yang simple, tetapi *futuristic* sehingga dapat memberikan kesan yang menarik bagi para pengguna. Pengembangan website juga memanfaatkan algoritma yang efektif sehingga dapat melakukan pencarian gambar dengan lebih efisien.

BAB III

ANALISIS PEMECAHAN MASALAH

1. Langkah-langkah pemecahan masalah

Pustaka (*library*) yang dimanfaatkan pada program ini adalah Numpy (untuk melakukan perhitungan matematis yang kompleks dengan cepat), math (menyediakan fungsi matematis pendukung), time (menghitung waktu pemrosesan gambar), dan OpenCV (pemrosesan gambar dasar).

Untuk CBIR dengan parameter warna, pertama-tama, gambar yang diterima akan diproses dari bentuk RGB menjadi bentuk HSV dengan melakukan langkah-langkah yang dijelaskan pada landasan teori. Gambar awal didapatkan nilai RGB kemudian dikonversikan menjadi bentuk HSV dan direpresentasikan sebagai *matrix of array* yaitu berukuran *tinggi* x *lebar* x 3 (masing-masing komponen H, S, dan V). Setelah didapatkan matriks HSV, matriks tersebut akan dipecah menjadi 16 kotak yang dipisah menjadi 4 x 4 blok sehingga bentuknya menjadi *array of matrix of array*. Setiap piksel dari masing-masing blok akan dikuantifikasi dan dikategorikan dalam *bin* sesuai nilai H, S, dan V yang dimiliki piksel tersebut. Jumlah total *bin* yang dihasilkan sebanyak $8 \times 3 \times 3 = 72$ *bin*, sehingga vektor yang digunakan untuk menghitung *cosine similarity* berupa array dengan 72 elemen. Karena matriks dipecah menjadi blok 4 x 4, jumlah total vektor yang dihasilkan adalah 16 untuk satu gambar. Setelah vektor dihasilkan, nilai cosinus untuk dua vektor yang berkoresponden akan dihitung dan tingkat kemiripan dihitung dengan merata-ratakan 16 nilai cosinus dari vektor blok kedua gambar.

Pada CBIR dengan parameter warna dilakukan pembagian menjadi 4 x 4 blok dikarenakan akan menghasilkan hasil yang lebih akurat dibandingkan 1 gambar full. Hal ini dikarenakan setiap blok pada image akan dibandingkan dengan setiap blok pada image lainnya dengan posisi bloknya yang sama contoh blok ujung kiri atas image 1 akan dibandingkan dengan blok ujung kiri atas image 2 dan seterusnya sampai blok ujung kanan bawah. Kemudian pada bagian *cosine similarity*, digunakan perbandingan 1 : 1 untuk semua blok sehingga untuk nilai akhir dari *cosine similarity* adalah penjumlahan semua cos yang sudah dibagi 16. Pemakaian perbandingan 1 : 1 untuk semua blok karena keseluruhan blok mempunyai pengaruh yang sama sehingga kemiripan suatu gambar bukan hanya dari pusat gambar itu saja tetapi juga dari bagian

ujung-ujung atau keseluruhan gambar meskipun mata manusia pada umumnya hanya fokus ke titik tengah.

Untuk CBIR dengan parameter tekstur, pertama-tama, komponen R, G, dan B dari tiap piksel gambar diekstrak terlebih dahulu. Setelah itu, akan dikonversi menjadi nilai *grayscale* untuk setiap piksel. Setelah mengonversi gambar ke bentuk grayscale, dengan menggunakan persamaan 2.5, matriks *cooccurrence* dapat dibuat dengan ukuran 256 x 256. Matriks diisi berdasarkan nilai *grayscale* dari 1 piksel dan piksel di sebelahnya. Contoh: terdapat angka 1 dan di sebelah kanannya juga angka 1, maka baris 1 dan kolom 1 pada matriks *cooccurrence* ditambahkan 1. Matriks *cooccurrence* digunakan untuk mendapatkan matriks *normalization*. Dari matriks *normalization*, didapatkan komponen ekstraksi *contrast*, *entropy*, dan *homogeneity*. Untuk meningkatkan akurasi dalam penentuan kemiripan gambar, pada perhitungan ini, ditambahkan juga tiga komponen ekstraksi lainnya, yaitu *dissimilarity*, *ASM*, dan *energy*.

Setelah merancang algoritma CBIR dengan parameter warna dan tekstur, algoritma tersebut perlu diintegrasikan ke *website*. Dalam tugas besar ini, digunakan *framework* React dan Next untuk *frontend* dan Flask untuk *backend*. Fitur-fitur yang harus dimiliki oleh website ini meliputi penjelasan singkat mengenai algoritma CBIR, kemampuan menerima folder dataset dari pengguna serta image dari pengguna, kemampuan memilih opsi pencarian berdasarkan warna atau tekstur, dan waktu pemrosesan yang cepat.

2. Proses Pemetaan Masalah Menjadi Elemen-elemen pada Aljabar Geometri

Penerapan Aljabar Geometri:

- Aplikasi *Cosine Similarity Theorem* untuk menentukan nilai kemiripan

Suatu konsep yang dipinjam dari aljabar geometri adalah *Cosine Similarity Theorem*. Konsep ini digunakan untuk menentukan kemiripan antara dua gambar yang sudah direpresentasikan dalam bentuk vektor, baik vektor komponen tekstur maupun vektor warna. Untuk nilai cosinus yang dihasilkan, nilai yang mendekati satu memiliki arti bahwa kedua gambar mirip, sedangkan nilai yang mendekati nol memiliki arti bahwa kedua gambar tidak mirip.

- Aplikasi *Symmetric Matrix*

Sebuah *Symmetric Matrix* adalah matriks persegi yang nilainya sama dengan nilai transposenya. Dalam CBIR dengan parameter tekstur, *symmetric matrix* dihasilkan dari matriks *cooccurrence* dan digunakan untuk mengekstraksi komponen tekstur.

- Vektor:

Suatu piksel dapat direpresentasikan sebagai vektor dengan komponen vektor berupa nilai R, G, dan B dari vektor tersebut.

- Matriks *cooccurrence*

Matriks *cooccurrence* juga diambil dari konsep matriks. Matriks *cooccurrence* merepresentasikan keadaan tekstur/kekasaran gambar tersebut.

3. Contoh Ilustrasi Kasus dan Penyelesaiannya

Contoh ilustrasi kasus yang bisa diaplikasikan kepada CBIR adalah pembelian produk pakaian yang cocok. Misalkan terdapat seseorang yang menyukai desain suatu produk dan ingin mencari produk yang memiliki desain yang mirip. Salah satu cara yang praktis agar pengguna tersebut dapat menemukan produk yang mirip adalah dengan memanfaatkan CBIR. Dengan CBIR berparameter warna, seseorang dapat meng-upload produk dengan warna yang diinginkan dan akan ditampilkan produk lain dengan warna yang serupa.

Suatu contoh kasus yang lain adalah pada *medical imaging* untuk melakukan diagnosis terhadap penyakit. Seandainya terdapat sebuah jaringan/organ yang terinfeksi sebuah penyakit yang tidak diketahui. Jaringan tersebut dapat diunggah ke dataset dan gambar-gambar yang serupa dan memiliki kasus yang mirip akan ditampilkan.

BAB IV

IMPLEMENTASI DAN UJI COBA

1. Implementasi Program Utama

- Pseudocode color.py (CBIR dengan parameter warna)

```
USE numpy as np
USE math
USE time
USE cv2
USE ThreadPoolExecutor

function rgb_to_hsv(image)
    Image <- convert_to_np_float(image) / 255.0
    maxc <- max_elements(image)
    minc <- min_elements(image)
    v <- maxc
    s <- np.where(maxc = 0, 0, (maxc - minc) / (maxc + 1e-20))
{ numpy if else for s value }

    rc <- (maxc - image[..., 2]) / (maxc - minc + 1e-20)
    gc <- (maxc - image[..., 1]) / (maxc - minc + 1e-20)
    bc <- (maxc - image[..., 0]) / (maxc - minc + 1e-20)

    h <- np.where(maxc = minc, 0,
                  np.where(maxc = image[..., 2], (bc - gc),
                           np.where(maxc == image[..., 1], 2.0 +
                           (rc - bc),
                           4.0 + (gc - rc))))
{ multiple if else for h }
```

```
h <- (h/6.0) mod 1.0
h <- np.where(h == 0, 360, h * 360)
-> np.stack([h, s * 100, v * 100))

function process_image_color(image)
  if image != Nil then:
    height <- image.height
    width <- image.width

    hsv_array <- rgb_to_hsv(image) # convert to hsv

    # set bins
    h_ranges <- [
      (316, 360),
      (1, 25),
      (26, 40),
      (41, 120),
      (121, 190),
      (191, 270),
      (271, 295),
      (295, 315)
    ]

    s_ranges <- [
      (0, 20),
      (21, 70),
      (71, 100)
    ]

    v_ranges <- [
```

```

        (0, 20),
        (21, 70),
        (71, 100)

    ]

{ initialize a 4x4 grid to divide the images }

subarrays <- np.array([hsv_array[i * (height div 4):(i + 1) * (height div 4), j * (width // 4):(j + 1) * (width div 4), :] for i in range(4) for j in range(4)])

{ create an array of 16 vectors }
vectors <- []
for subarray in subarrays:
    h_values, s_values, v_values <- subarray[:, :, 0].flatten(), subarray[:, :, 1].flatten(), subarray[:, :, 2].flatten()
    vector <- np.zeros(72)

    for h_category, (h_min, h_max) in enumerate(h_ranges):
        h_mask <- (h_values ≥ h_min) and (h_values ≤ h_max)

        for s_category, (s_min, s_max) in enumerate(s_ranges):
            s_mask <- (s_values ≥ s_min) & (s_values ≤ s_max)

            for v_category, (v_min, v_max) in enumerate(v_ranges):

```

```

v_mask <- (v_values ≥ v_min) &
(v_values ≤ v_max)

combination_mask <- h_mask and s_mask
and v_mask

vector[h_category * 9 + s_category * 3
+ v_category] <- np.sum(combination_mask)

vectors.append(vector)
-> np.array(vectors) { return array of 16 vectors}
else
    output("Failed to load the image.")

```

- Pseudocode tekstur.py (CBIR dengan parameter tekstur)

```

USE cv2
USE numpy as np
USE math
USE time
USE ThreadPoolExecutor

{ process images by texture }
Function process_image_texture(image)
    if image ≠ Nil then

        { extract rgb and convert to grayscale }
        b, g, r <- image[:, :, 0], image[:, :, 1], image[:, :, 2]
        gs <- r * 0.299 + g * 0.587 + b * 0.114
        grayscale_array <- gs

```

```

    { make a cooccurrence matrix }

    cooccurrence <- np.histogram2d(grayscale_array[:, :-1].ravel(),
    grayscale_array[:, 1:].ravel(), bins=256,
    range=[[0, 255], [0, 255]])[0]

    transpose_cooccurrence <- transpose(cooccurrence)
    sym_matrix <- cooccurrence + transpose_cooccurrence

    matrix_norm <- sym_matrix / np.sum(sym_matrix)

    i, j <- np.indices(matrix_norm.shape)
    diff <- i - j

    { create a mask to only process nonzero elements }
    non_zero_indices <- matrix_norm != 0

    che <- np.zeros(6, dtype='float64')
    che[0] <- np.sum(matrix_norm[non_zero_indices] *
    diff[non_zero_indices] ** 2) { contrast }
        che[1] <- np.sum(matrix_norm[non_zero_indices] / (1 +
    diff[non_zero_indices] ** 2)) { homogeneity }
        che[2] <- -np.sum(matrix_norm[non_zero_indices] * np.log10(
    matrix_norm[non_zero_indices])) { entropy }
        che[3] <- np.sum(matrix_norm[non_zero_indices] * np.abs(
    diff[non_zero_indices])) { dissimilarity }
        che[4] <- np.sum(matrix_norm[non_zero_indices] ** 2) {
    ASM }

        che[5] <- np.sqrt(che[4]) { energy }
        -> che

    else

```

```

        output("Failed to load the image.")

{ cosine function }
function cos(A, B)
    lengthA <- 0
    lengthB <- 0
    sum <- 0
    i traversal [0..(len(A))-1]:
        lengthA <- lengthA + A[i] ** 2
        lengthB <- lengthB + B[i] ** 2
        sum <- sum + A[i] * B[i]

    lengthA <- (lengthA) ** 0.5
    lengthB <- (lengthB) ** 0.5
    -> (sum / (lengthA * lengthB))

```

2. Penjelasan Struktur Program Berdasarkan Spesifikasi

Program ini terbagi menjadi dua folder utama. Folder pertama berupa folder *backend* yang berisi algoritma CBIR dengan parameter warna dan tekstur serta API untuk berhubungan ke *frontend* serta *script* untuk melakukan *web scraping*. Folder kedua berupa folder *algeo02* yang mengurus tampilan *website*. Folder *backend* terbagi lagi menjadi tiga folder, yaitu folder CBIR, folder Scraping, dan folder api. Folder *algeo02* terbagi lagi menjadi public dan src.

Pada folder CBIR, terdapat file color.py yang berisi algoritma untuk CBIR dengan parameter warna. Fungsi-fungsi yang terdapat pada file tersebut berupa:

- `rgb_to_hsv(image)`: Berfungsi untuk mengonversi gambar dalam bentuk RGB menjadi matriks yang tiap elemennya merupakan vektor dengan komponen H, S, dan V.
- `process_image_color(image)`: Berfungsi untuk membagi matriks HSV menjadi blok 4x4. Setelah terbagi menjadi blok 4x4, fungsi akan mengkategorikan tiap piksel dalam blok ke dalam salah satu dari 72 *bin* HSV dan menghasilkan vektor-vektor

yang merepresentasikan frekuensi kemunculan tiap rentang warna. Fungsi ini akan mengembalikan 16 vektor yang masing-masing memiliki 72 elemen.

- $\cos(A, B)$: Menghitung nilai *cosinus* dari dua vektor.
- $\text{cosine_similarity}(A, B)$: Menghitung rata-rata dari 16 vektor yang dihasilkan oleh proses `image_color(image)`. Fungsi ini mengembalikan nilai kemiripan antara dua gambar.

Terdapat file lain pada folder CBIR berupa `tekstur.py` yang berisi algoritma untuk CBIR dengan parameter tekstur. Fungsi-fungsi yang terdapat pada file tersebut berupa:

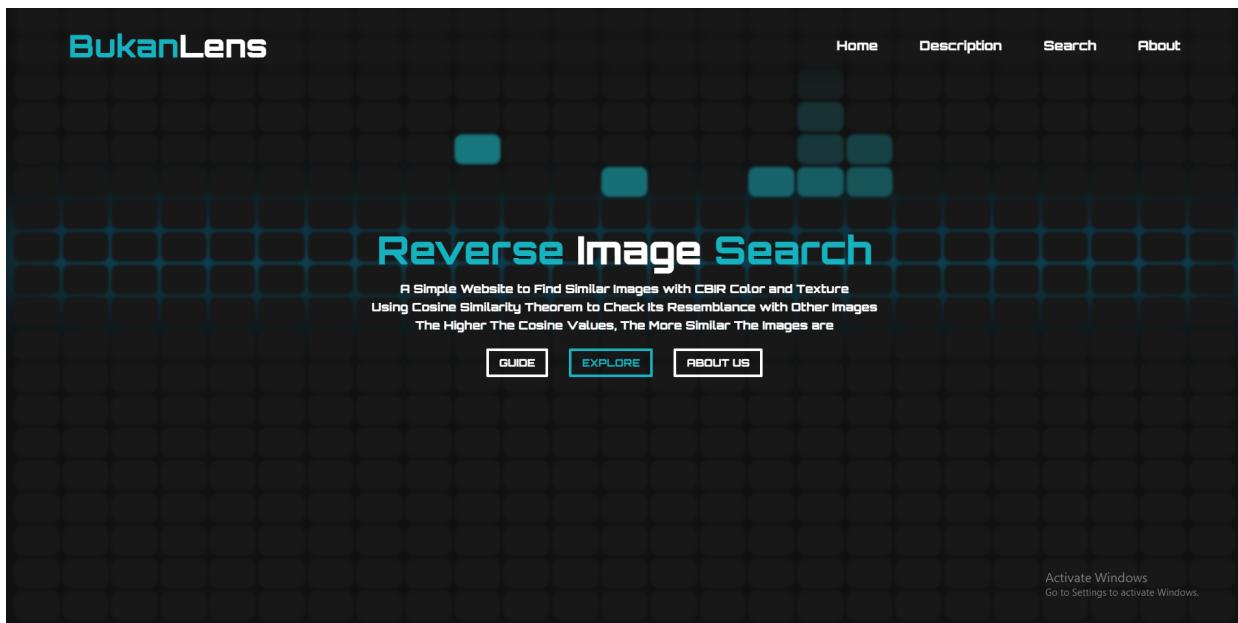
- `process_image_texture(image)`: Fungsi ini akan mengekstraksi nilai R, G, dan B dari sebuah gambar dan menghasilkan sebuah matriks yang merupakan representasi gambar tersebut dengan nilai *grayscale*. Setelah itu, fungsi akan membangun matriks *cooccurrence* dan menghitung *matrix normalization* dari matriks *cooccurrence* tersebut. Dari *matrix normalization*, fungsi akan menghitung komponen-komponen tekstur, yaitu *contrast*, *homogeneity*, *entropy*, *dissimilarity*, *ASM*, dan *energy*.
- $\cos(A, B)$: Menghitung nilai cosinus dari dua vektor.

Dalam folder algeo02, terdapat folder public yang berisi asset publik yang digunakan secara umum. Terdapat juga folder src yang berisi empat folder lain, yaitu api, app, components, dan utils. Folder api berisi script untuk menghubungkan program ke *backend*. Folder app merupakan folder utama dari *frontend* dan kode untuk page-page yang ada dalam program, seperti landing page, page About Us, page Search, dan page Description. Folder components berisi komponen-komponen yang dapat dipakai berkali-kali dalam website, seperti tombol-tombol dan *navigation bar*. Folder utils berisi fungsi pembantu dalam mengambil gambar.

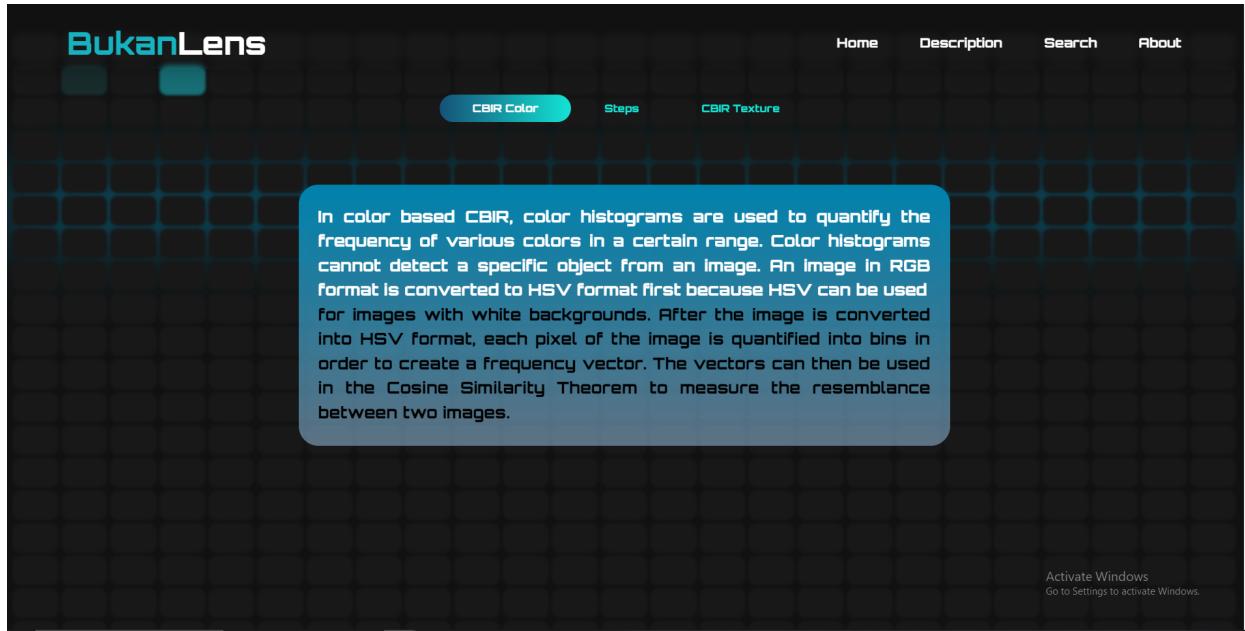
3. Tata Cara Penggunaan Program

Clone repository dari github terlebih dahulu. Kemudian nyalakan website dengan npm run dev dan nyalakan node server.js serta run fungsi python `image_processing`. Kemudian masuk ke browser masing-masing dan ketik link yang tertera ketika melakukan npm run dev. Setelah itu, tampilan layar akan langsung berada pada home page dari website. Background dari website dapat menyala biru ketika mengarahkan mouse ke kotak-kotak hitam. Terdapat penjelasan singkat mengenai reverse image search, navigation bar, dan 3 tombol yaitu description, search,

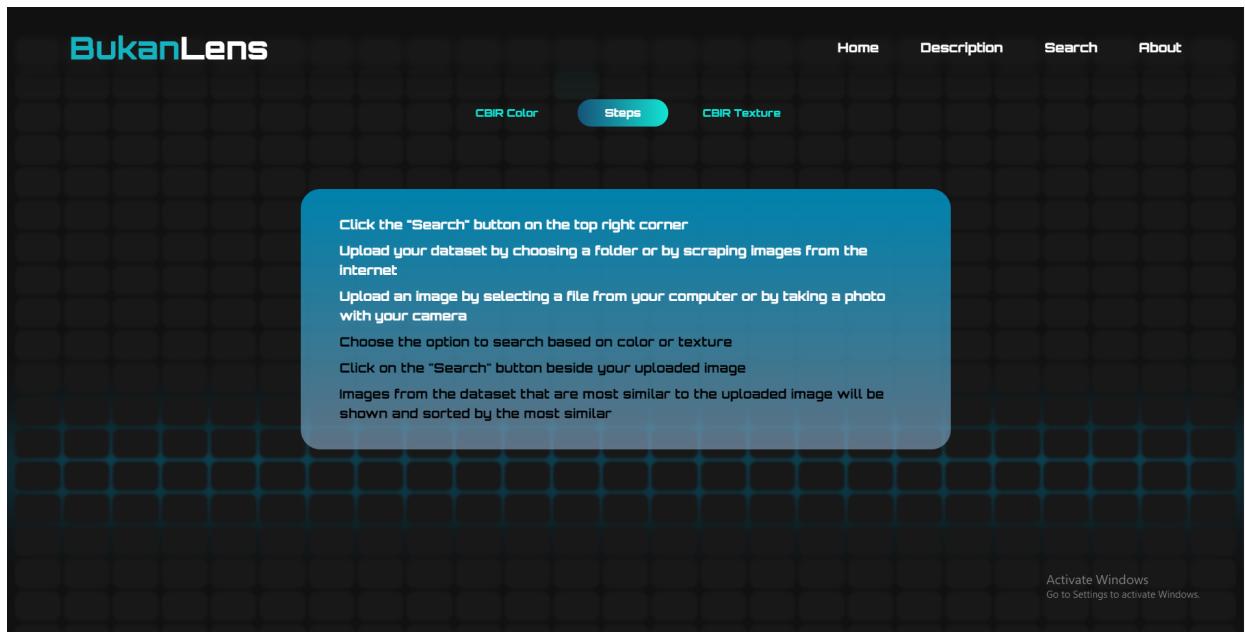
dan about us. Ketiga tombol itu akan mengarahkan ke page yang berbeda sesuai dengan namanya. Jika memilih description maka page akan berpindah ke description page yang berisi penjelasan mengenai CBIR color dan tekstur serta penjelasan tata cara menggunakan fitur searching gambar pada website. Jika memilih search maka page akan berpindah ke search page yang berisi fitur utama dari website ini. Terdapat tempat untuk upload image dan upload dataset, tombol search, tombol untuk memilih color atau tekstur, dan tombol untuk memilih fitur kamera. Pertama-tama upload data set terlebih dahulu baru upload image dan kemudian search. Lalu pada bagian bawah akan keluar foto dengan kemiripan secara berurut dari persentase tertinggi hingga persentase 60. Jika ingin memilih tekstur maka klik tombol tekstur dan sebaliknya. Serta untuk kamera dilakukan dengan menekan tombol Use Camera dan kamera akan langsung menyala. Proses pencarian kemiripan gambar akan berlangsung secara otomatis dan foto akan terupdate setiap 10 detik secara otomatis. Image scraping dilakukan dengan memasukkan link website dan kemudian akan terupload menjadi dataset. Jika memilih About us maka akan berpindah ke page about us yang berisi foto, nama, dan NIM dari pembuat website.



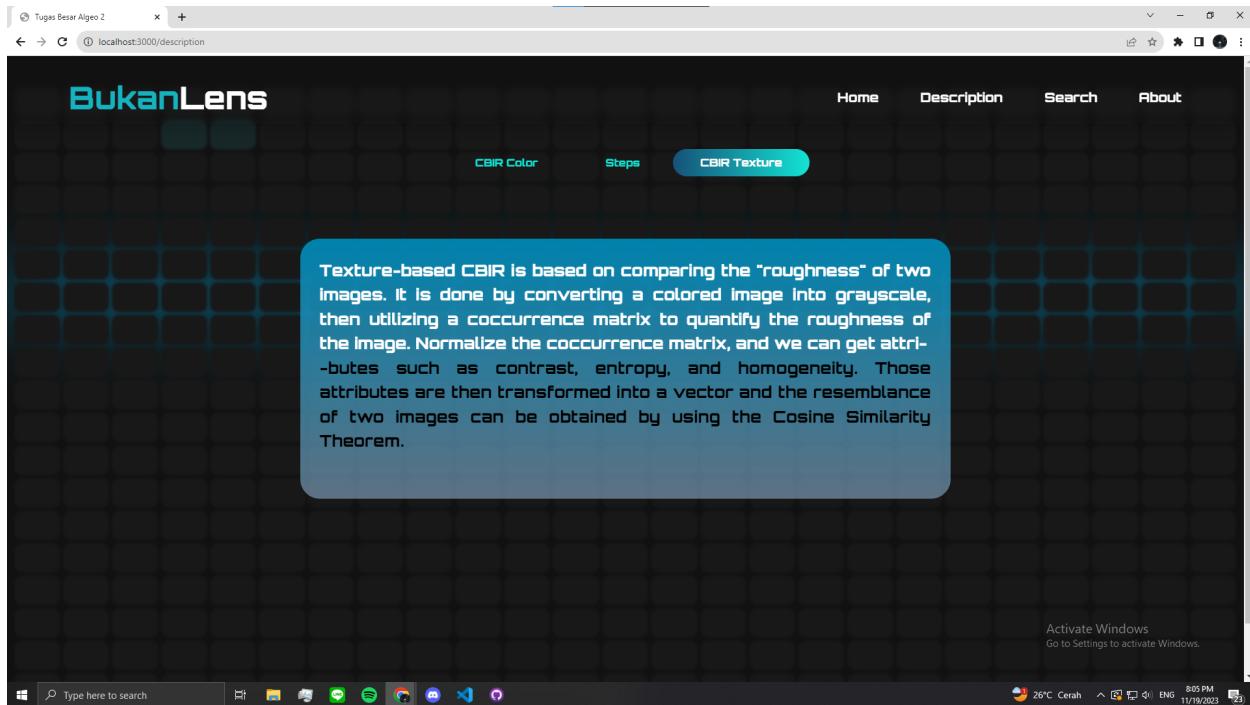
Gambar 4.1. *Landing page website*



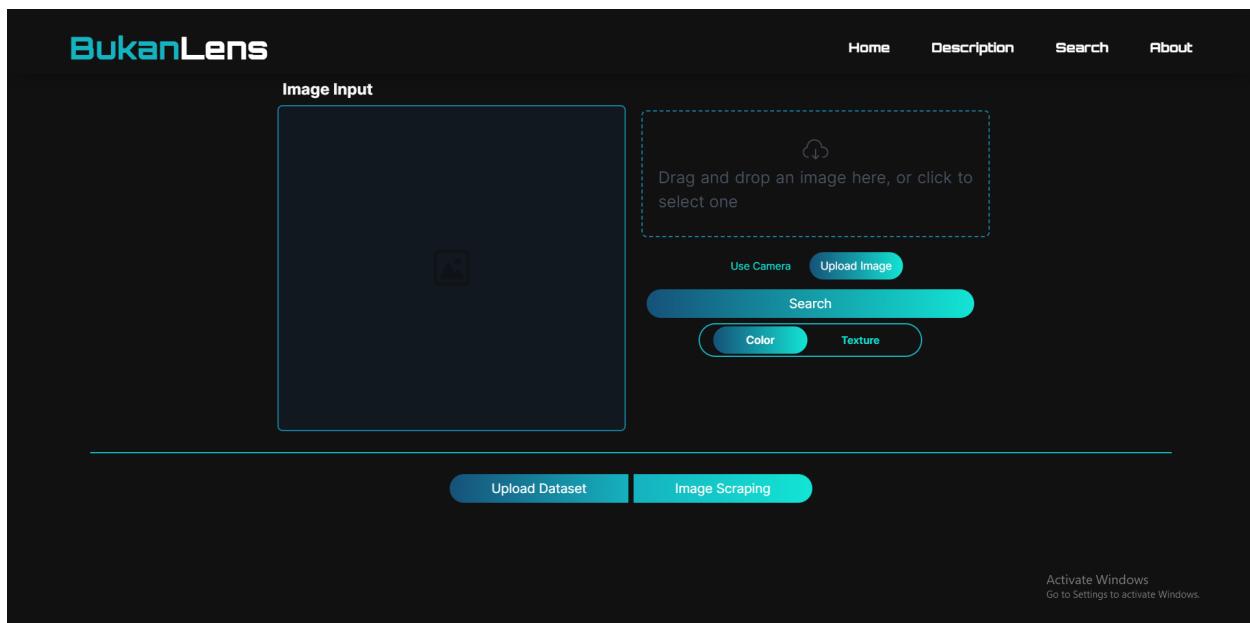
Gambar 4.2. Penjelasan CBIR warna dalam website



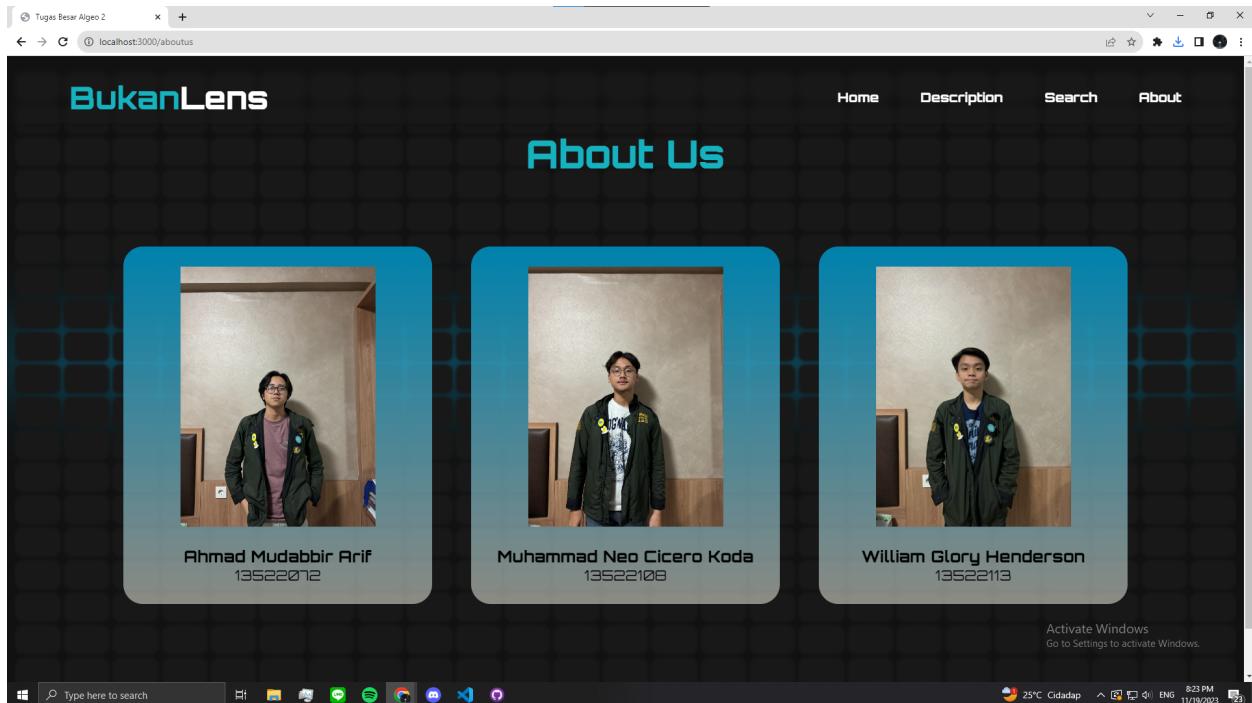
Gambar 4.3. Langkah-langkah penggunaan website



Gambar 4.4. Penjelasan CBIR Color dalam website



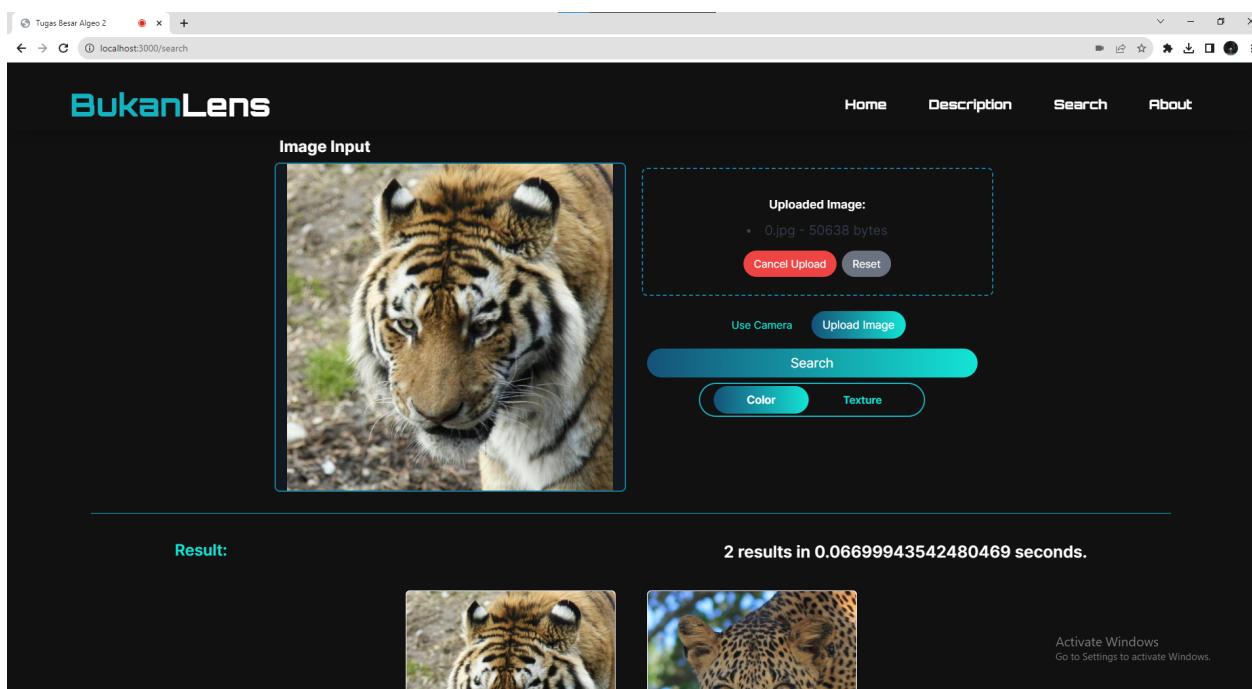
Gambar 4.5. Tampilan page *search*



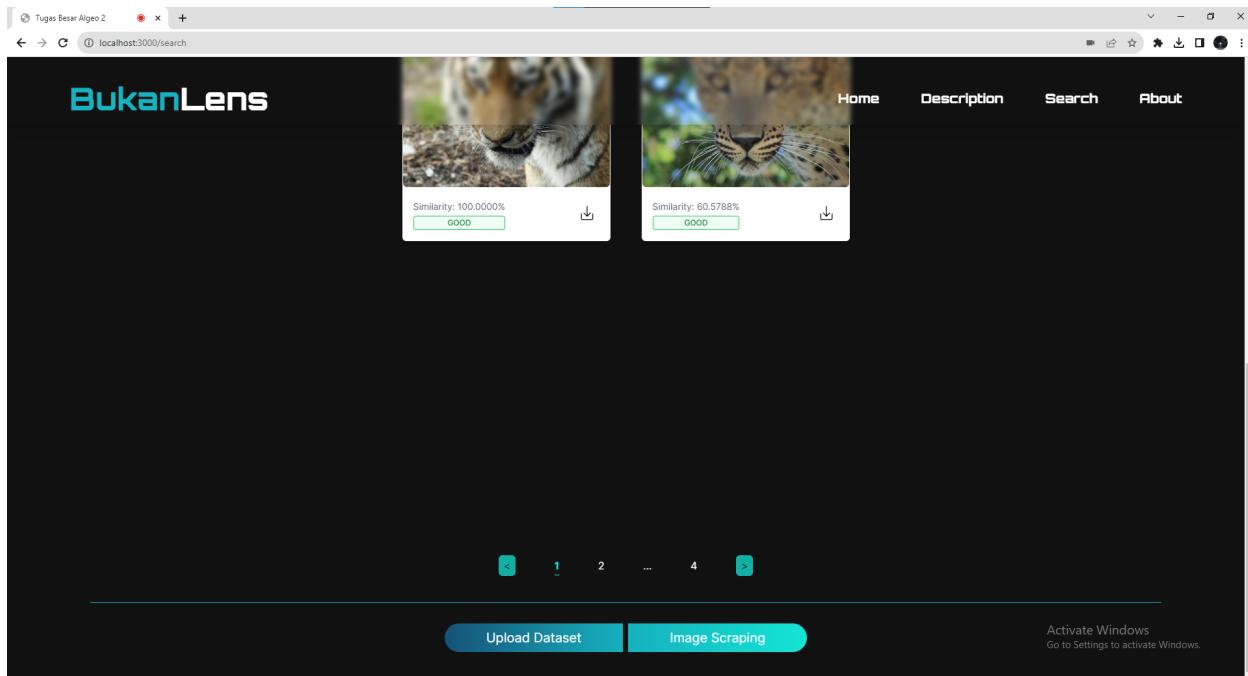
Gambar 4.6. Tampilan *about us* dalam website

4. Hasil Pengujian

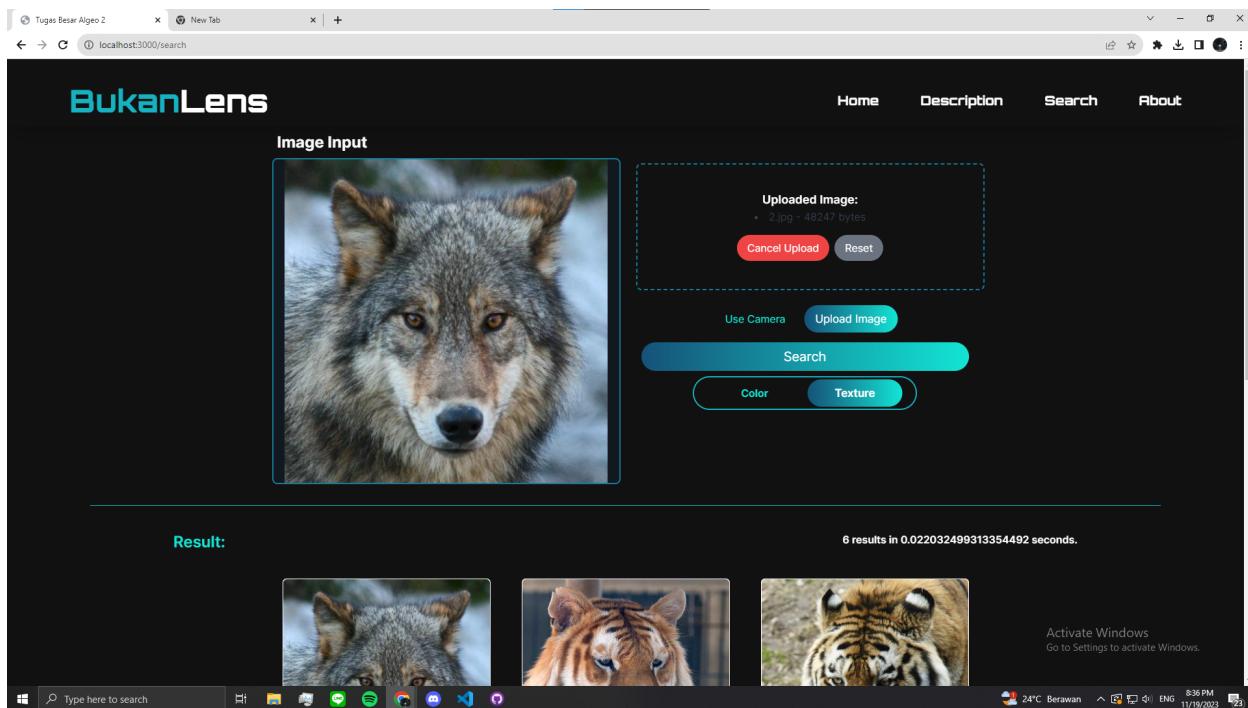
Test case warna (20 gambar sebagai dataset):



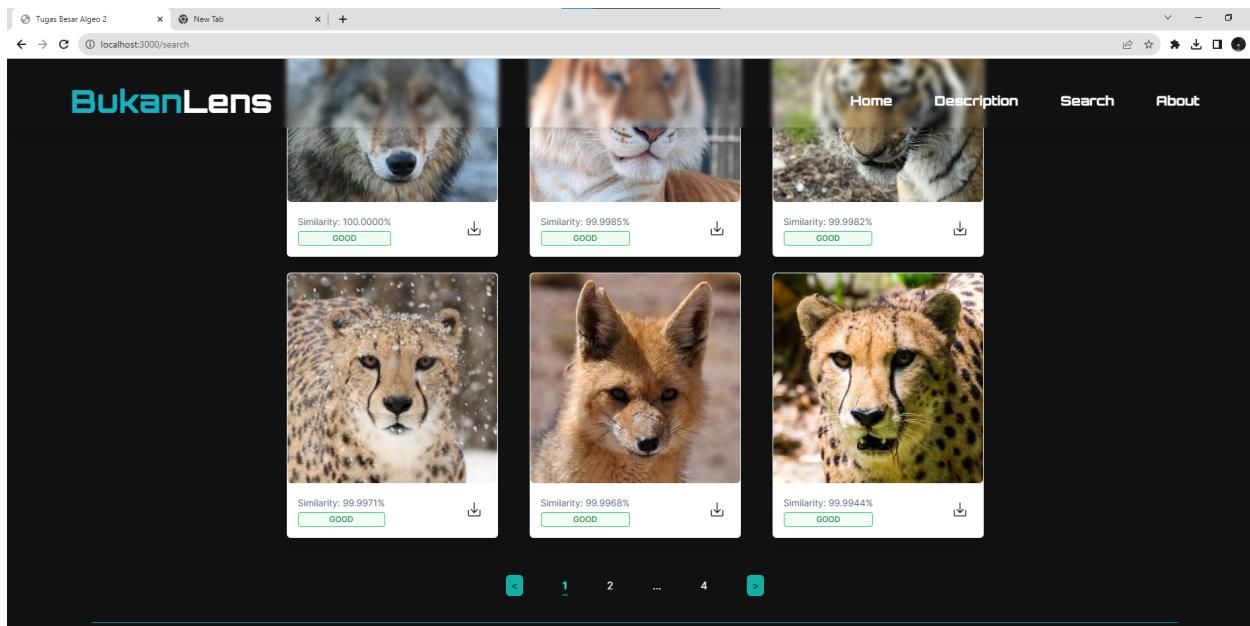
Gambar 4.7. Tampilan setelah meng-upload sebuah image (parameter warna)



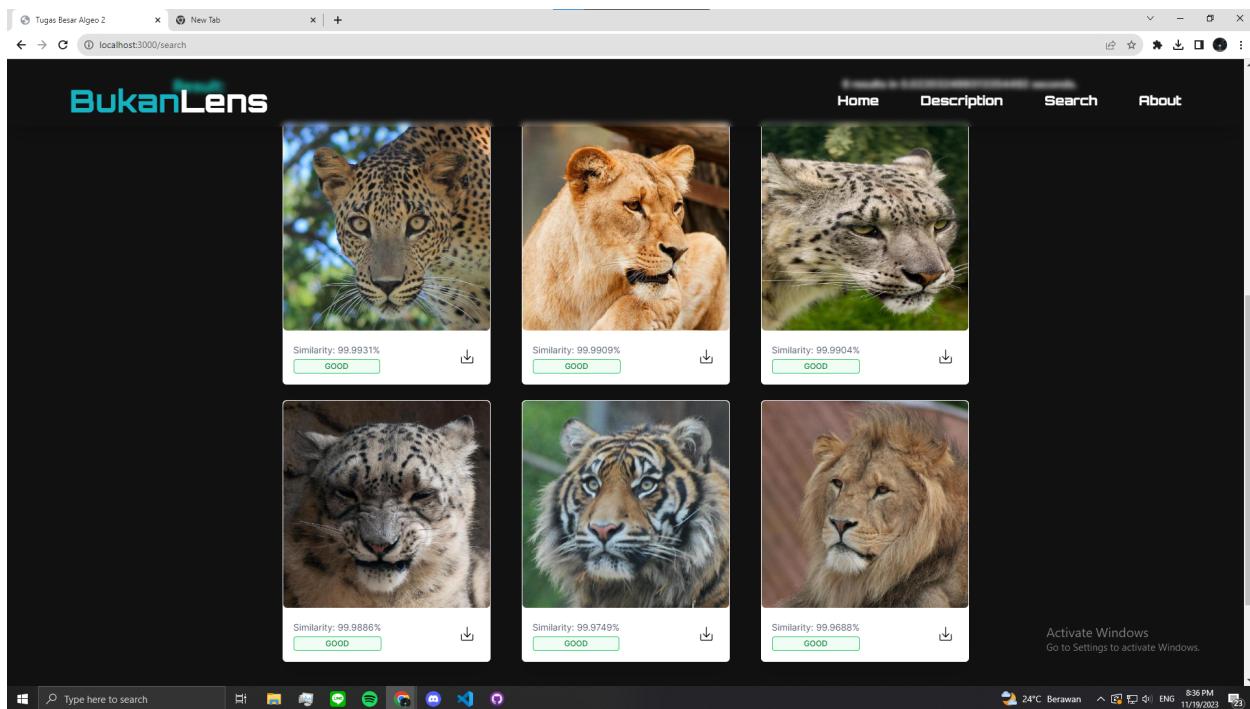
Gambar 4.8. Tampilan setelah meng-upload sebuah image (parameter warna) (1)



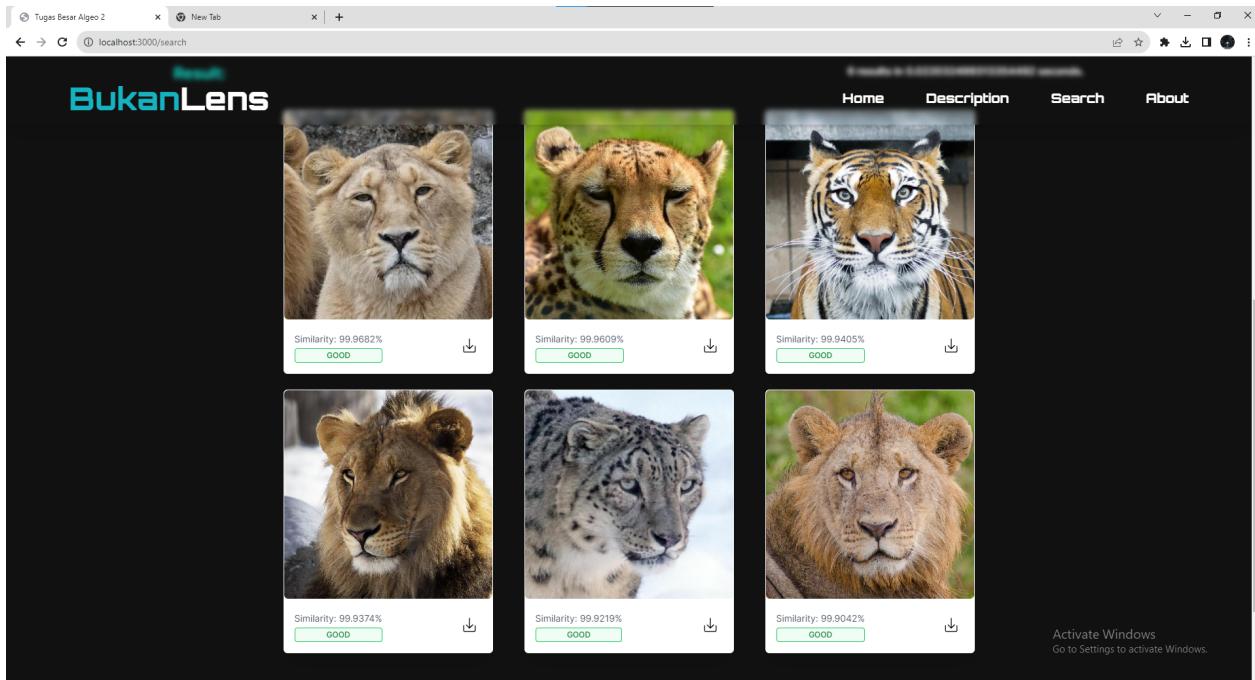
Gambar 4.9. Tampilan setelah meng-upload sebuah image (parameter tekstur)



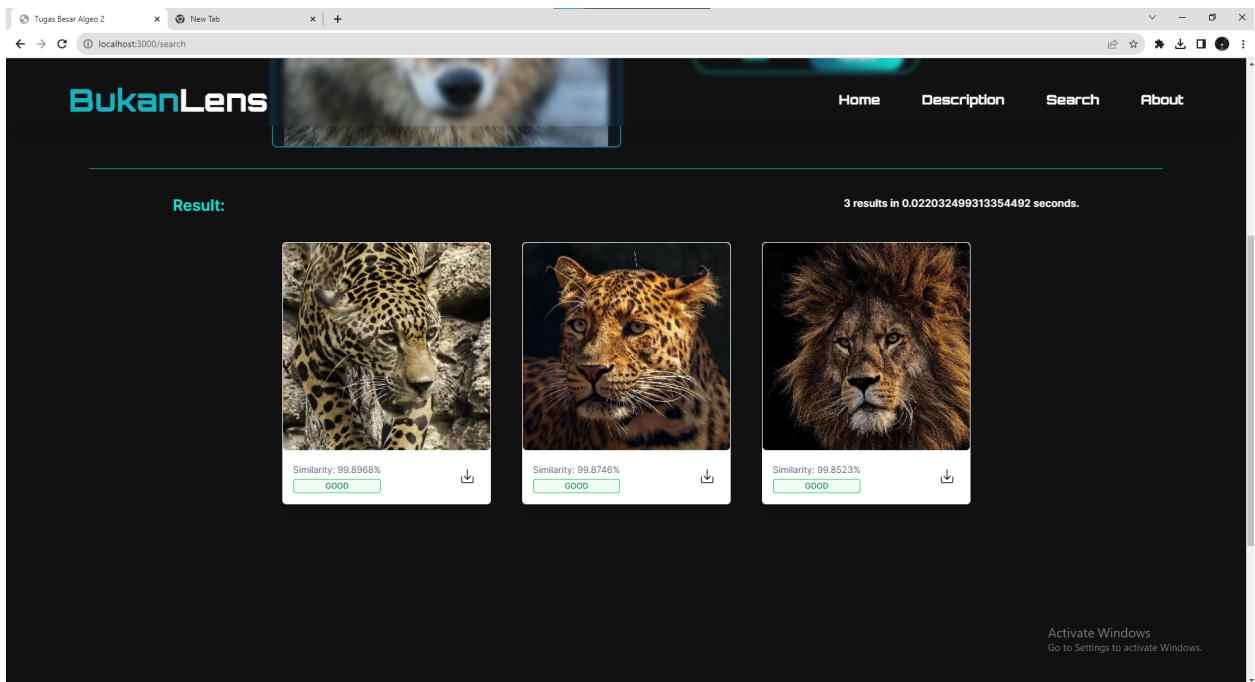
Gambar 4.10. Tampilan setelah meng-*upload* sebuah image (parameter tekstur) (1)



Gambar 4.11. Tampilan setelah meng-*upload* sebuah image (parameter tekstur) (2)



Gambar 4.12. Tampilan setelah meng-*upload* sebuah image (parameter tekstur) (3)



Gambar 4.12. Tampilan setelah meng-*upload* sebuah image (parameter tekstur) (4)

5. Analisis Desain Solusi Algoritma Pencarian

CBIR dengan parameter warna dan CBIR dengan parameter tekstur keduanya memiliki keunggulan masing-masing yang bergantung dengan spesifikasi pencarian dan karakteristik gambar yang dipakai.

Untuk CBIR dengan parameter warna, keunggulannya terletak pada konsepnya yang intuitif bagi manusia karena warna merupakan hal yang mudah dibedakan oleh pandangan manusia. Warna efektif dalam membeda-bedakan objek atau daerah dalam suatu gambar. Warna juga peka terhadap perubahan pada tingkat kecerahan suatu objek. Kepakaannya terhadap pencerahan dapat mengeluarkan hasil yang kurang baik pada penilaian gambar/objek yang sama, namun dalam pencerahan yang beda. Selain itu dua gambar/objek yang berbeda, namun memiliki pewarnaan yang sama akan dinilai memiliki kemiripan yang tinggi walaupun seharusnya tidak.

CBIR dengan parameter warna biasa digunakan ketika warna merupakan aspek yang penting dalam bidang tersebut. Contoh aplikasi CBIR dengan parameter warna adalah dalam bidang *fashion*, contohnya mencari pakaian dengan warna yang sama. Selain itu, CBIR dengan parameter warna juga berguna dalam bidang seni rupa dan kesehatan (*medical imaging*).

Untuk CBIR dengan parameter tekstur, keunggulannya terletak pada kemampuannya untuk mendeteksi pola dan/atau detil yang tidak bisa dideteksi oleh CBIR dengan parameter warna. Selain itu, perubahan pada warna, seperti perubahan pada pencerahan juga tidak memengaruhi CBIR dengan parameter tekstur. Kekurangan CBIR adalah algoritmanya yang tidak sesederhana CBIR dengan parameter tekstur. Selain itu, konsep tekstur kurang intuitif bagi manusia jika dibandingkan dengan konsep warna. Konsep tekstur juga lebih subjektif karena setiap orang memiliki penafsiran tersendiri terhadap definisi dari tekstur.

CBIR dengan parameter tekstur biasa digunakan ketika detil-detil dan pola dalam gambar lebih dilihat dan warna suatu gambar tidak terlalu dipentingkan. Contoh aplikasi CBIR dengan parameter tekstur adalah dalam analisis bahan/material, citra satelit, dan ilmu geologi.

Dapat disimpulkan bahwa kedua metode CBIR memiliki keunggulannya masing-masing dan pemilihan metode yang tepat bergantung kepada tujuan pemakaiannya dan karakteristik gambar-gambar yang akan dipakai.

BAB V

KESIMPULAN

1. Kesimpulan

Kesimpulan dari tugas besar website Reverse Image Search adalah telah dibuat suatu website yang dapat menerima sebuah gambar dan dataset dan mencari gambar-gambar yang serupa dalam dataset tersebut dengan menggunakan algoritma CBIR berbasis warna dan CBIR berbasis tekstur.

2. Saran

Saran untuk tugas besar ini adalah pemberian waktu yang lebih lama dikarenakan masih dalam transisi dan masih baru belajar membuat website dengan framework baru karena adanya spesifikasi tertentu seperti waktu yang membuat pemrosesan pada backend harus lebih cepat. Kemudian perbaikan atau pembaharuan spesifikasi bisa diinformasikan lebih cepat dan disebarluaskan secara luas karena mungkin beberapa sudah jadi dan tidak membaca spek dengan teliti sehingga perlu diganti-ganti lagi yang dimana memakan waktu dan merubah kode.

3. Komentar

Komentar untuk tugas besar ini adalah cukup seru karena mengeksplor framework baru dan dapat bekerja secara kelompok dimana ini bisa menjadi ilustrasi untuk bekerjasama dengan orang lain dalam perusahaan.

4. Refleksi

Refleksi terhadap tugas besar ini adalah kita bisa lebih teliti lagi dalam membaca spesifikasi dari fungsi serta bisa lebih berhati-hati dalam membuat kode karena banyak terjadi typo dalam proses pembuatan sehingga terkadang terjadi error. Pembagian waktu juga diperlukan karena banyaknya tugas besar sehingga memaksa kita untuk lebih bisa bekerja dalam tekanan dan semaksimal mungkin.

5. Ruang Perbaikan atau Pengembagan

Ruang Perbaikan atau Pengembangan terhadap tugas besar ini adalah websitenya tidak responsif terhadap PC lain sehingga jika ukuran layar kecil, tampilan website menjadi kurang bagus. Bonus tidak semua dibuat sehingga masih bisa dimaksimalkan. Tampilan website *description* mungkin bisa dibuat lebih menarik dengan menambahkan gambar-gambar atau animasi lain.

DAFTAR PUSTAKA

- Anton, H., & Rorres, C. (2013). *Elementary Linear Algebra with Applications*. John Wiley & Sons.
- Y. Jun, et al. (2011) *Content-based image retrieval using color and texture fused features*,
- Mathematical and Computer Modelling,
<https://www.sciencedirect.com/science/article/pii/S0895717710005352>
- Hughes, J., et al.. (2014). *Computer Graphics: Principles and Practice*. Addison-Wesley.
- Chan, J, et al. (2019). *Python API Development Fundamentals*. Packt Publishing Ltd.
- Baraldi, A, Pannigiani, F. (1995). *An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters*. IEEE Transactions on Geoscience and Remote Sensing.

LAMPIRAN

Link menuju *repository*: <https://github.com/Dabbir/Algeo02-22072>

Link video youtube: <https://youtu.be/LEySWcDpDHw?si=OXcxGdsRfEvu9-7X>