

Laporan Tugas Besar 2

IF2123 Aljabar Linier dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Disusun oleh:

Akbar Al Fattah	13522036
Dzaky Satrio Nugroho	13522059
Christopher Brian	13522106

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

Daftar Isi

Daftar Isi	2
Bab 1: Deskripsi Masalah	4
Bab 2: Landasan Teori	5
2.1. Content-Based Information Retrieval (CBIR)	5
2.1.1. CBIR dengan Parameter Warna	5
2.1.2. CBIR dengan Parameter Tekstur	6
2.2. Pengembangan Website	7
2.2.1. Website	7
2.2.2. HTTP	8
2.2.3. Front End	8
2.2.4. Back End	8
Bab 3: Analisis Pemecahan Masalah	9
3.1. Langkah-Langkah Pemecahan Masalah	9
3.2. Proses Pemetaan Masalah Menjadi Elemen-Elemen pada Aljabar Geometri	10
3.2.1. CBIR dengan Parameter Warna	10
3.2.2. CBIR dengan Parameter Tekstur	10
3.3. Contoh Ilustrasi Kasus dan Penyelesaiannya	11
3.3.1. CBIR dengan Parameter Warna	11
3.3.2. CBIR dengan Parameter Tekstur	12
Bab 4: Implementasi dan Uji Coba	14
4.1. Implementasi Program Utama	14
4.1.1 Tombol Upload Dataset	14
4.1.2 Tombol Upload Gambar Query	15
4.2. Penjelasan struktur program	15
4.3. Tata Cara Penggunaan Program	19
4.4. Hasil Pengujian	22
4.4.1 Dataset Acak	22
4.4.2. Dataset yang Mengandung Gambar yang Sama tetapi Dirotasi	23
4.4.3. Dataset yang Mengandung Gambar yang Diedit Ukurannya tetapi Gambar Sama	24
4.4.4. Dataset Gambar Grayscale tetapi Gambar Query Berwarna	26
4.4.5. Dataset Warna Solid	27
4.4.6. Dataset Gambar Berwarna tetapi Gambar Query Grayscale	28
4.5. Analisis Desain Solusi Algoritma	30
Bab 5: Kesimpulan	31
5.1. Kesimpulan	31
5.2. Saran	31
5.3. Komentar atau Tanggapan	31

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

5.4. Refleksi terhadap Tugas Besar	31
5.5. Ruang Perbaikan atau Pengembangan	32
Daftar Pustaka	33
Link Repository	33
Link Video	33

Bab 1: Deskripsi Masalah

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

Di dalam Tugas Besar 2 ini, kami 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 2: Landasan Teori

2.1. Content-Based Information Retrieval (CBIR)

Content-Based Image Retrieval (CBIR) adalah suatu metode yang digunakan untuk mencari dan mengambil gambar berdasarkan karakteristik visualnya. Proses ini dimulai dengan mengekstraksi ciri-ciri signifikan dari gambar, seperti warna, tekstur, dan bentuk. Setelah ekstraksi ciri-ciri, informasi tersebut diubah menjadi vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Selanjutnya, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor fitur dari gambar yang dicari dengan vektor fitur gambar dalam *dataset*. Hasil pencocokan ini digunakan untuk menyusun urutan gambar dalam *dataset*, menampilkan gambar yang paling serupa dengan gambar yang dicari. Pendekatan CBIR ini membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar secara efisien, tanpa perlu melakukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai visual antara gambar-gambar tersebut. Dalam Tugas Besar ini, kami mengimplementasikan dua parameter CBIR yang paling umum digunakan.

2.1.1. CBIR dengan Parameter Warna

Pada CBIR kali ini akan dibandingkan masukkan dari sebuah gambar dengan gambar yang dimiliki oleh *dataset*, hal ini dilakukan dengan cara mengubah gambar yang berbentuk RGB menjadi nilai rata-ratanya. Pada perhitungan rata-rata, 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]

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

2. Mencari C_{max} , C_{min} , dan Δ

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

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

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

3. Mencari nilai HSV

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

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

$$V = C_{max}$$

Setelah mendapatkan nilai HSV lakukanlah perbandingan antara gambar dari *input* dengan *dataset* dengan menggunakan *cosine similarity*.

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

Dengan A dan B adalah vektor dan n adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *cosine similarity*. Untuk melakukan perhitungan rata-rata, gambar akan dibagi menjadi 4×4 blok.

2.1.2. CBIR dengan Parameter Tekstur

CBIR dengan perbandingan tekstur dilakukan menggunakan suatu matriks yang dinamakan matriks *co-occurrence*. Matriks ini digunakan karena dapat melakukan pemrosesan yang lebih mudah dan cepat. Vektor yang dihasilkan juga mempunyai ukuran yang lebih kecil. Misalkan terdapat suatu gambar I dengan $n \times m$ piksel dan suatu parameter *offset* ($\Delta x, \Delta y$).

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:

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

Melalui persamaan tersebut, nilai θ yang kami gunakan adalah 0° dan 90° . Setelah didapat matriks *co-occurrence*, buatlah *symmetric matrix* dengan menjumlahkan matriks *co-occurrence* dengan hasil *transpose*-nya. Lalu cari matriks *normalized* dengan persamaan berikut.

$$MatrixNorm = \frac{MatrixOcc}{\Sigma MatrixOcc}$$

Langkah-langkah dalam CBIR dengan parameter tekstur adalah sebagai berikut :

1. Konversi warna gambar menjadi *grayscale*, ini dilakukan karena warna tidaklah penting dalam penentuan tekstur. Oleh karena itu, warna RGB dapat diubah menjadi suatu warna *grayscale* Y dengan perhitungan sebagai berikut.

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

2. Lakukan kuantifikasi dari nilai *grayscale*. Karena citra *grayscale* berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran 256 kali 256. Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut.
3. Dari matriks *co-occurrence* bisa diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Persamaan yang dapat digunakan untuk mendapatkan nilai tiga komponen tersebut antara lain :

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

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

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

4. Ukurlah kemiripan dari kedua gambar dengan menggunakan teorema *cosine similarity*, disini A dan B adalah dua vektor dari dua gambar. Semakin besar hasil *cosine similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

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

2.2. Pengembangan Website

Pengembangan *website* adalah proses pembangunan dan pemeliharaan *website*. Mulai dari pembuatan *website* berisi teks sederhana hingga yang bentuknya kompleks seperti *platform* media sosial atau aplikasi web.

2.2.1. Website

Website adalah sebuah halaman atau sekumpulan halaman *web* yang saling terhubung dan dapat diakses dari seluruh dunia, selama terkoneksi ke jaringan internet.

Setiap halaman *website* memiliki alamat unik yang dikenal sebagai URL (*Uniform Resource Locator*). Situs web dapat berisi berbagai jenis informasi, misalnya teks, gambar, video, dan suara. Selain itu, *website* juga bisa memuat fitur interaktif seperti *form* kontak, komentar, atau *chatting*.

2.2.2. HTTP

HTTP adalah protokol transfer data dari server website ke client atau komputer. Adanya transfer data membuat Anda dapat melihat berbagai informasi seperti dokumen, file, gambar dan video di browser.

2.2.3. *Front End*

Front end adalah bagian dari sebuah website yang langsung dilihat oleh pengguna. Front end dibangun menggunakan beberapa bahasa pemrograman seperti HTML, CSS, dan JavaScript. Atau, bisa juga menggunakan *front end framework* agar lebih mudah. Contohnya Vue.js, React, dan sebagainya. Ketika ada transfer data dari *server* ke *browser*, bahasa pemrograman untuk *front end* akan memuat tampilan *website*. Pemuatan dilakukan tanpa harus terus-menerus “berkomunikasi” dengan komputer berkat dukungan *cache*. Lewat *front end website*, pengguna bisa berinteraksi langsung dengan konten dan komponen *website*. Misalnya, menonton video, membuka gambar, menandai kalimat, dan lainnya.

2.2.4. *Back End*

Back end adalah bagian belakang layar dari sebuah website. Artinya, bagian ini tidak dapat dilihat user. *Back end* adalah data dan infrastruktur yang membuat *website* berfungsi. *Back end* menyimpan dan memproses data website untuk pengguna. Bahasa pemrograman yang dibutuhkan jauh lebih banyak daripada *front end*. *Front end* mungkin hanya membutuhkan JavaScript. Akan tetapi, server membutuhkan hampir semua bahasa. Bahasa yang digunakan untuk membangun *Back End* antara lain PHP, Ruby, bahasa Python dan lainnya. Contoh *back end* adalah aplikasi berbasis web.

Bab 3: Analisis Pemecahan Masalah

Untuk menyelesaikan masalah ini, kami menggunakan teknologi berikut:

Front-end:

1. HTML5
2. CSS

Back-end:

1. Python dan Numpy
2. Flask (Framework Web Development)
3. PIL (untuk mengolah gambar)

3.1. Langkah-Langkah Pemecahan Masalah

1. Unggah *dataset* dari *folder* yang hanya berisi gambar dengan salah satu format berikut: .png, .jpg, .jpeg, atau .bmp. (*folder* yang mengandung *file* yang tidak termasuk dalam ekstensi tersebut tidak akan diterima).
2. Setiap gambar dalam dataset akan memasuki tahap pra-proses yang meliputi resizing menjadi ukuran 256 kali 256 piksel. Selanjutnya, gambar yang sudah di-resize akan diubah menjadi matriks yang berisi nilai RGB tiap piksel.
3. Untuk tahap pra-proses selanjutnya terdapat dua proses utama, yaitu CBIR dengan parameter warna dan CBIR dengan parameter tekstur. Untuk CBIR dengan parameter warna, matriks RGB akan diubah menjadi matriks HSV. Selanjutnya, matriks HSV akan dibagi menjadi 4 kali 4 blok lalu dihitung nilai rata-rata HSV untuk setiap blok. Nilai rata-rata HSV setiap blok akan diubah ke dalam vektor dengan tiga komponen dan disimpan dalam *file* .txt.
4. Untuk CBIR dengan parameter tekstur, matriks RGB gambar akan diubah menjadi *Gray Level Co-Occurrence Matrix* (GLCM) yang diproses sehingga hasil akhirnya berupa matriks simetri yang dinormalisasi. Matriks tersebut kemudian dihitung nilai *contrast*, *homogeneity*, dan *entropy*-nya, lalu disimpan sebagai vektor dengan tiga komponen dalam *file* .txt.
5. Unggah gambar *query* (selain file dengan ekstensi yang terdaftar pada bagian 3.1 nomor 1 tidak akan diterima).
6. Pilih parameter CBIR, dengan pilihan parameter warna atau parameter tekstur.
7. Server akan membaca *cache* yang sudah dibuat pada bagian 3.1 langkah 2 dan 3. Jika pada langkah 6 pengguna memilih parameter warna, maka yang dibaca oleh server adalah baris pertama pada *file cache*. Jika pengguna memilih parameter tekstur, maka yang dibaca oleh server adalah baris kedua.
8. Jika pengguna memilih parameter warna, gambar *query* akan melalui proses yang sama dengan pra-proses gambar dalam *dataset*, kemudian dihitung nilai *cosine similarity* antar setiap blok dalam gambar *query* dan setiap gambar dalam *dataset*, lalu dihitung rata-rata *cosine similarity* dengan setiap gambar dalam *dataset* dan diurutkan berdasarkan nilainya.
9. Jika pengguna memilih parameter tekstur, gambar *query* akan melalui proses yang sama dengan pra-proses gambar dalam *dataset*, kemudian dihitung cosine similarity

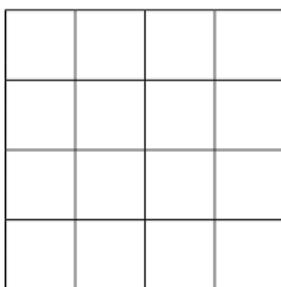
antara gambar *query* dengan setiap gambar dalam *dataset* dan diurutkan berdasarkan nilainya.

10. Server akan menampilkan gambar-gambar dengan hasil perhitungan *cosine similarity* di atas 60% dan diurutkan dari tertinggi ke terendah pada website.

3.2. Proses Pemetaan Masalah Menjadi Elemen-Elemen pada Aljabar Geometri

3.2.1. CBIR dengan Parameter Warna

1. Ubah gambar menjadi matriks yang berisi vektor yang memiliki komponen nilai RGB.
2. Ubah semua vektor yang menjadi elemen matriks menjadi nilai HSV.



3. Bagi gambar menjadi 4x4 blok seperti berikut.
4. Buat list untuk menyimpan vektor fitur rata-rata HSV.
5. Untuk setiap blok, hitung rata-rata nilai H, rata-rata nilai S, dan rata-rata nilai V dan simpan sebagai vektor lalu dimasukkan ke *list*.
6. Lakukan langkah 1 sampai 5 untuk gambar *query* dan semua gambar yang ada di *dataset*. Kemudian, list vektor rata-rata nilai HSV disimpan sebagai *cache*.
7. Hitung *cosine similarity* dari setiap blok gambar di antara kedua gambar. Perlu dicatat bahwa ada total 16 nilai *cosine similarity* yang bermakna ada 1 nilai *cosine similarity* untuk setiap blok dan harus dibandingkan dengan blok yang sama.
8. Hitung rata-rata dari nilai *cosine similarity* dari 16 nilai *cosine similarity* tersebut.
9. Kemiripan gambar dengan gambar *query* dalam persen adalah *Similarity (%)* = $(\cos \theta * 100)\%$ dengan $\cos \theta$ adalah *cosine similarity* rata-rata yang didapatkan dari langkah 8.

3.2.2. CBIR dengan Parameter Tekstur

1. Ubah gambar menjadi matriks yang berisi vektor yang memiliki komponen nilai RGB
2. Ubah nilai RGB pada matriks menjadi *grayscale*
3. Ubah matriks *grayscale* menjadi *Gray Level Co-occurrence Matrix* (GLCM) berukuran 256 kali 256. Elemen-elemen yang dicek adalah elemen yang bersebelahan secara horizontal dan vertikal. Sudut yang dipilih pada website ini adalah 0° dan 90°

4. Ubah matriks GLCM menjadi bentuk simetrinya dengan cara menambahkan matriks GLCM dan matriks GLCM yang sudah di-*transpose*.
5. Ubah menjadi matriks simetri menjadi bentuk *normalized*-nya dengan membagi nilai setiap elemen dengan jumlah setiap elemen dalam matriks simetri.
6. Hitung nilai *contrast*, *homogeneity*, dan *entropy* menggunakan matriks *normalized* dan simpan ke dalam list.
7. Lakukan langkah 1 sampai 6 untuk gambar *query* dan semua gambar yang ada di *dataset*. Kemudian, list vektor nilai *contrast*, *homogeneity*, dan *entropy* disimpan sebagai *cache*.
8. Hitung nilai *cosine similarity* antara gambar *query* dengan setiap gambar dalam *dataset* menggunakan vektor nilai *contrast*, *homogeneity*, dan *entropy*.
9. Kemiripan gambar dengan gambar *query* dalam persen adalah *Similarity (%)* = $(\cos \theta * 100)\%$ dengan $\cos \theta$ adalah *cosine similarity* yang didapatkan dari langkah 8.

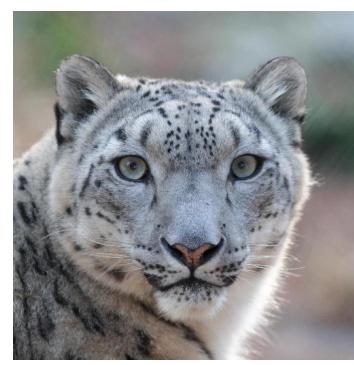
3.3. Contoh Ilustrasi Kasus dan Penyelesaiannya

3.3.1. CBIR dengan Parameter Warna

Misalkan ada gambar *query* dan salah satu gambar *dataset* berikut:



Query



Dataset

Kita ubah kedua gambar tersebut menjadi matriks vektor HSV. Setelah diubah, kedua gambar tersebut kita bagi menjadi 4x4 blok.

Blok Query

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Blok Dataset

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Setelah kita bagi matriks menjadi 4x4, didapat bahwa list vektor rata-rata nilai H,S,dan V adalah:

Query: [(89.66, 37.73, 35.47), (70.91, 36.02, 45.14), (68.99, 40.05, 40.63), (94.35, 47.54, 24.88), (35.95, 24.25, 62.17), (34.26, 19.25, 60.64), (33.75, 23.92, 56.18), (56.63, 31.81, 42.87), (38.34, 32.25, 58.49), (39.35, 27.48, 50.31), (35.59, 23.8, 53.49), (41.42, 35.09, 44.09), (52.19, 47.65, 49.5), (70.98, 73.13, 40.68), (62.59, 56.3, 39.79), (60.23, 68.97, 36.34)]

Sampel *dataset*: [(106.06, 14.34, 51.52), (180.15, 5.53, 56.91), (178.43, 7.06, 64.12), (181.59, 8.72, 60.92), (151.43, 8.27, 62.35), (210.04, 13.26, 58.62), (209.49, 14.84, 58.91), (163.47, 12.26, 44.76), (175.64, 5.69, 46.47), (185.69, 11.93, 55.79), (196.19, 14.21, 59.3), (121.17, 9.33, 64.01), (145.73, 4.68, 31.68), (61.24, 15.58, 40.25), (122.0, 5.47, 68.0), (164.05, 8.94, 59.14)]

Kemudian, kita hitung cosine similarity untuk masing-masing vektor yang ada di setiap blok yang bersesuaian dengan kedua gambar. Didapatkan bahwa nilai *cosine similarity* untuk setiap blok yang bersesuaian adalah:

[0.9657922714919092,	0.8992024721171504,	0.8947538388917274,
0.9160208648991355,	0.7660089639907505,	0.6971687334911862,
0.7054210666864429,	0.8745851486336967,	0.6883061648851739,
0.7730455949945578,	0.7454100081524033,	0.8503580520745063,
0.7299132209426967,	0.8664870538312663,	0.8182311071830866,
0.735176092620134]		

Setelah mendapatkan data di atas, kita hitung rata-rata *cosine similarity*-nya. Didapat bahwa rata-rata *cosine similarity*-nya adalah 0.8078675409303641. Jadi, kemiripannya adalah 80,78675409303641%.

3.3.2. CBIR dengan Parameter Tekstur

Misalkan ada gambar *query* dan salah satu gambar *dataset* berikut:



Query



Dataset

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

Kita ubah kedua gambar tersebut menjadi matriks grayscale yang selanjutnya diolah menjadi *Gray Level Co-occurrence Matrix* (GLCM). Matriks GLCM tersebut selanjutnya diproses menjadi matriks simetri versi *normalized*.

Berikut adalah sampel matriks simetri GLCM versi *normalized* dari kedua gambar. (Data di bawah ini hanya sebagian karena ukuran matriks GLCM pada metode ini adalah 256x256, terlalu besar untuk dimasukkan ke laporan ini)

Query: [[0.00000000e+00 3.31906137e-05 0.00000000e+00 0.00000000e+00
0.00000000e+00] [3.31906137e-05 0.00000000e+00 4.97859205e-05 4.97859205e-05
4.97859205e-05] [0.00000000e+00 4.97859205e-05 6.63812274e-05 3.31906137e-05
4.97859205e-05] [0.00000000e+00 4.97859205e-05 3.31906137e-05 3.31906137e-05
4.97859205e-05] [0.00000000e+00 4.97859205e-05 4.97859205e-05 4.97859205e-05
6.63812274e-05]]
Sampel dataset: [[4.23387950e-05 4.23387950e-05 4.23387950e-05 2.11693975e-05
2.11693975e-05] [4.23387950e-05 0.00000000e+00 4.23387950e-05 6.35081926e-05
2.11693975e-05] [4.23387950e-05 4.23387950e-05 0.00000000e+00 4.23387950e-05
2.11693975e-05] [2.11693975e-05 6.35081926e-05 4.23387950e-05 0.00000000e+00
0.00000000e+00] [2.11693975e-05 2.11693975e-05 2.11693975e-05 0.00000000e+00
0.00000000e+00]]

Dari matriks tersebut, didapatkan vektor nilai *contrast*, *homogeneity*, dan *entropy* sebagai berikut.

Query: (781.6361313020677, 0.048635486616697005, 14.274403699109929)
Sampel dataset: (564.9596087895337, 0.058541056908990245, 13.958344548482696)

Selanjutnya, kita hitung nilai *cosine similarity* dari vektor kedua gambar ini. Didapatkan bahwa nilai *cosine similarity*-nya adalah 0.9999792521454273.

Dengan demikian, kemiripannya adalah 99,99792521454273%.

Bab 4: Implementasi dan Uji Coba

4.1. Implementasi Program Utama

Berikut adalah garis besar dari implementasi program utama yang esensial.

4.1.1 Tombol Upload Dataset

```
procedure upload_dataset() {prosedur yang dijalankan saat menekan tombol upload dataset}
    dataset <- getFromRequest("Dataset") {dapatkan dataset dari upload folder}
    if(not isEmpty(dataset)) then {jika dataset yang diupload tidak kosong}
        datasetValid <- isDatasetValid(dataset) {cek kevalidan dataset, yaitu hanya mengandung gambar}
        if(datasetValid) then
            saveToServer(dataset) {simpan dataset ke penyimpanan server}
            output("Folder berhasil diunggah") {pesan bahwa folder dataset berhasil diunggah}
            cacheCBIRFeatures(dataset) {simpan fitur CBIR sebagai cache di dalam penyimpanan server}
        else
            output("Folder invalid") {Folder invalid karena mengandung file yang bukan merupakan gambar}
    else
        output("Folder yang diunggah kosong") {Folder invalid karena folder kosong}
```

4.1.2 Tombol Upload Gambar Query

```
procedure upload_image() {prosedur saat menekan tombol unggah gambar}
    file <- getFromRequest("File") {dapatkan file yang diupload}
    if(file = NULL) then {tidak ada file yang diunggah}
        output ("Tidak ada gambar query yang diunggah")
        return

    if(file != NULL and isFileValid(file))then {jika file valid (berupa gambar) dan ada}
        saveToServer(file) {simpan gambar ke server}

        if(parameter = "texture")then {jika parameter yang dipilih di website adalah tekstur}
            featureQuery <- ContrastHomogeneityEntropy(imageToNormalizedGLCM(file)) {cari vektor
c,h,dan e dari gambar query}

            for images in dataset: {baca semua file di dataset yang sudah diunggah}
                featureDataset <- readCache(images,"texture") {baca cache vektor tekstur dari dataset}
                cosSimilarity <- textureCosineSimilarity(featureQuery,featureDataset) * 100 {hitung cosine
similarity (tekstur) dari kedua gambar tersebut}
                if(cosSimilarity>60)then {jika nilai cosine similarity>60, masukkan ke priority queue}
                    push(imagesPriorityQueue,(cosSim,images)) {priority queue dipilih agar terurut
otomatis berdasarkan nilai cosine similarity}
                else {jika parameter yang dipilih di website adalah warna}
                    featureQuery<- HSVAverage(imageToHSVMatrix(file)) {cari vektor rata-rata nilai HSV dari
gambar query}
                    for images in dataset: {baca semua file di dataset yang sudah diunggah}
                        featureDataset <- readCache(images,"color") {baca cache vektor warna dari dataset}
                        cosSimilarity <- colorCosineSimilarity(featureQuery,featureDataset) * 100 {hitung cosine
similarity (warna) dari kedua gambar tersebut}
                        if(cosSimilarity>60)then{jika nilai cosine similarity>60, masukkan ke priority queue}
                            push(imagesPriorityQueue,(cosSim,images))
                return

            else {gambar query tidak valid}
                output("Gambar query tidak valid")
                return
```

4.2. Penjelasan struktur program

Website ini secara umum memiliki 4 buah *source code* utama, yaitu *server.py*, *index.html*, *styles.css*, dan *images.py*.

a. *server.py*

server.py adalah *source code* yang mengandung *framework* Flask. Artinya, *source code* ini adalah *server* untuk *website* ini yang berfungsi untuk menerima *upload*, menampilkan paginasi, dan menampilkan seluruh fitur yang ada di *website* ini termasuk mendownload file PDF. *server.py* juga merupakan program utama dari *website* ini. Untuk kerangka *server.py* kurang lebih sudah dijelaskan di bagian [4.1. Implementasi Program Utama](#).

Tugas Besar 2 IF2123

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Kelompok 09 Tahun Ajaran 2023/2024

b. index.html

index.html hanya berfungsi sebagai kerangka dan *front end* website ini. Tidak akan dijelaskan secara rinci di laporan ini.

c. styles.css

styles.css berfungsi sebagai *front end* yang menampilkan tampilan antarmuka website ini. Tidak dijelaskan di laporan ini.

d. images.py

images.py adalah *file* yang menyimpan semua fungsi untuk melakukan CBIR, termasuk *caching*. Berikut adalah penjelasan setiap fungsinya.

1. CBIR parameter warna

- cosine_similarity()

```
# Cosine Similarity
def cosine_similarity(vector_a, vector_b):
    dot_product = sum(a * b for a, b in zip(vector_a, vector_b)) #nilai dot product dari vektor a dan vektor b
    norm_a = sum(a**2 for a in vector_a) ** 0.5 #nilai vektor a yang sudah dinormalisasi
    norm_b = sum(b**2 for b in vector_b) ** 0.5 #nilai vektor a yang sudah dinormalisasi
    if norm_a == 0 and norm_b == 0: #jika normalisasi vektor a = normalisasi vektor b = 0
        return 1.0 #kembalikan nilai 1 karena vektor a dan b sudah pasti berimpit
    elif norm_a == 0 or norm_b == 0: #jika hanya salah satunya yang bernilai 0
        return 0.0 #kembalikan nilai 0 karena nilai dot productnya adalah 0
    cosine_similarity_value = dot_product / (norm_a * norm_b) #hitung cosine similarity
    return cosine_similarity_value
```

- rgb_to_hsv() (ini hanyalah implementasi menghitung nilai HSV dari bagian [2.1.1. CBIR dengan Parameter Warna](#))

```
# CBIR dengan parameter warna
def rgb_to_hsv(rgb_tuple): #mengubah RGB menjadi HSV
    r, g, b = rgb_tuple
    r = r / 255.0
    g = g / 255.0
    b = b / 255.0
    Cmax = max(r, g, b)
    Cmin = min(r, g, b)
    v = Cmax
    if (Cmax != 0):
        s = (Cmax - Cmin) / Cmax
    else:
        s = 0
    if (s == 0):
        h = 0
    else:
        delta = Cmax - Cmin
        if (Cmax == r):
            h = 60 * (((g - b) / delta) % 6)
        elif (Cmax == g):
            h = 60 * (((b - r) / delta) + 2)
        else: # Cmax == b
            h = 60 * (((r - g) / delta) + 4)
    h = (h + 360) % 360
    return int(h), int(s * 100), int(v * 100)
```

- `image_to_hsv_matrix()`

```
def image_to_hsv_matrix(image_path): #mengubah gambar menjadi matriks HSV
    image = Image.open(image_path).convert("RGB")
    resized_image = image.resize((256, 256)) #resize gambar menjadi 256x256
    rgb_matrix = np.array(resized_image)
    hsv_matrix = np.array([[rgb_to_hsv(pixel) for pixel in row] for row in rgb_matrix])
    #ubah matriks RGB menjadi HSV
    return hsv_matrix
```

Catatan: Kami perlu melakukan *resizing* menjadi 256 kali 256 agar *pre-processing* dan *caching* berjalan lebih cepat. Dampak dari hal ini juga sangat kecil dalam perhitungan kemiripan gambar.

- `hsv_average()`

```
def hsv_average(hsv_matrix): #menghitung 16 buah vektor yang berisi rata-rata nilai H,S,dan V
    rows, cols, _ = hsv_matrix.shape
    part_rows = rows // 4 #bagi matriks menjadi 4x4 blok
    part_cols = cols // 4 #bagi matriks menjadi 4x4 blok
    hsv_average = [] #16 vektor rata-rata nilai HSV akan disimpan di sini
    for i in range(4):
        for j in range(4):
            part = hsv_matrix[i * part_rows : (i + 1) * part_rows, j * part_cols : (j + 1) * part_cols]
            #pilih blok yang akan dihitung vektor rata-rata nilai HSV nya
            average_h = round(np.mean(part[:, :, 0]), 2) #hitung rata-rata nilai H dari blok yang dipilih
            average_s = round(np.mean(part[:, :, 1]), 2) #hitung rata-rata nilai S dari blok yang dipilih
            average_v = round(np.mean(part[:, :, 2]), 2) #hitung rata-rata nilai V dari blok yang dipilih
            hsv_average.append((average_h, average_s, average_v)) #simpan ke list
    return hsv_average
```

- `color_average_cosine_similarity()`

```
def color_average_cosine_similarity(hsv_average1, hsv_average2):
    cosine_similarity_values = [] #nilai cosine similarity dari 16 blok akan diletakkan di list ini
    for avg1, avg2 in zip(hsv_average1, hsv_average2):
        vector_avg1 = np.array(avg1).flatten()
        vector_avg2 = np.array(avg2).flatten()
        similarity = cosine_similarity(vector_avg1, vector_avg2) #hitung nilai cosine similarity dari dua buah vektor yang bersesuaian
        cosine_similarity_values.append(similarity) #simpan nilai tersebut ke list
    average_cosine_similarity = np.mean(cosine_similarity_values) #hitung rata-rata nilai cosine similarity dari semua blok
    return average_cosine_similarity
```

2. CBIR parameter tekstur

- `rgb_to_grayscale()`

```
def rgb_to_grayscale(r, g, b): #mengubah RGB menjadi grayscale
    y = 0.29 * r + 0.587 * g + 0.114 * b
    return y
```

Ini hanya implementasi dari konversi menjadi grayscale dari bagian [2.1.2. CBIR dengan Parameter Tekstur](#)

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

- `image_to_normalized_glcm()`

```
def image_to_normalized_glcm(image_path):
    #sudut yang dipilih adalah 0 derajat dan 90 derajat
    image = Image.open(image_path).convert("RGB")
    resized_image = image.resize((256, 256))
    rgb_matrix = np.array(resized_image)
    grayscale_matrix = np.apply_along_axis(lambda pixel: int(round(rgb_to_grayscale(*pixel))), axis=-1, arr=rgb_matrix)
    #ubah gambar menjadi matriks grayscale
    #susun matriks GLCM
    glcm_matrix = np.zeros((256,256), dtype=int)
    right_neighbors = np.roll(grayscale_matrix, shift=-1, axis=1)
    down_neighbors = np.roll(grayscale_matrix, shift=-1, axis=0)
    glcm_matrix[grayscale_matrix, right_neighbors] += 1
    glcm_matrix[right_neighbors, grayscale_matrix] += 1
    glcm_matrix[grayscale_matrix, down_neighbors] += 1
    glcm_matrix[down_neighbors, grayscale_matrix] += 1
    symmetric_matrix = glcm_matrix + glcm_matrix.T #hitung symmetric matrix
    symmetric_matrix_normalized = symmetric_matrix / np.sum(symmetric_matrix) #normalisasi symmetry matrix
    return symmetric_matrix_normalized
```

Catatan: Kami perlu melakukan *resizing* menjadi 256 kali 256 agar *pre-processing* dan *caching* berjalan lebih cepat. Dampak dari hal ini juga sangat kecil dalam perhitungan kemiripan gambar. Sudut 0 derajat dan 90 derajat dipilih untuk memudahkan implementasi dan mempercepat waktu kerja program.

- `contrast_homogeneity_entropy()`

```
def contrast_homogeneity_entropy(symmetric_matrix_normalized):
    contrast = np.sum(symmetric_matrix_normalized * np.square(np.subtract.outer(range(256), range(256)))) #hitung nilai contrast
    homogeneity = np.sum(symmetric_matrix_normalized / (1 + np.square(np.subtract.outer(range(256), range(256))))) #hitung nilai homogeneity
    entropy = -np.sum(symmetric_matrix_normalized * np.log2(symmetric_matrix_normalized + 1e-10)) #hitung nilai entropy
    return contrast, homogeneity, entropy
```

- `texture_cosine_similarity()`

```
def texture_cosine_similarity(v1, v2): #hitung nilai cosine similarity
    vector1 = list(v1)
    vector2 = list(v2)
    cosine_similarity_value = cosine_similarity(vector1, vector2)
    return cosine_similarity_value
```

- Simpan preprocessing sebagai cache

```
def save_cbir_results(image_path): #simpan semua vektor sebagai cache
    hsv_matrix = image_to_hsv_matrix(image_path)
    text_file_path = "static/cache/" + os.path.splitext(os.path.basename(image_path))[0] + '.txt'

    hsv_average_result = hsv_average(hsv_matrix)
    glcm_matrix_normalized = image_to_normalized_glcm(image_path)
    contrast, homogeneity, entropy = contrast_homogeneity_entropy(glcm_matrix_normalized)
    glcm_result = (contrast, homogeneity, entropy)

    with open(text_file_path, 'w') as file:
        file.write('\t'.join(map(str, hsv_average_result)) + '\n')
        file.write('\t'.join(map(str, glcm_result)) + '\n')
```

- Muat data cache

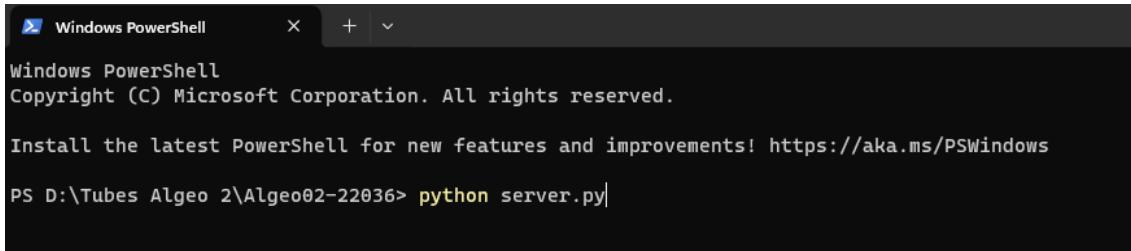
```
def get_cbir_results(image_path, cbir_type): #muat file cache
    text_file_path = "static/cache/" + os.path.splitext(os.path.basename(image_path))[0] + '.txt'
    with open(text_file_path, 'r') as file:
        lines = file.readlines()
    if cbir_type == "color":
        result_color = lines[0].strip().split('\t')
        result_color_out = [eval(i) for a,i in enumerate(result_color)]
    else:
        result_texture = lines[1].strip().split('\t')

    return float(result_texture[0]), float(result_texture[1]), float(result_texture[2])
```

4.3. Tata Cara Penggunaan Program

Catatan: Seluruh tangkapan layar website di laporan ini memiliki frontend versi lama, namun backendnya tetap sama dan tidak ada perbedaan dengan versi final. Jika Anda ingin melihat tangkapan layar dari frontend website versi final, lihat di README repository ini.

1. Clone repository ini ke *local repository*.
2. Jika **source code index.html, style.css, server.py, dan image.py** masih berada di dalam *folder src*, keluarkan semuanya dari *folder src* dan pindahkan setiap *file* di atas dengan petunjuk berikut.
 - a. index.html dipindahkan ke *folder templates/*.
 - b. style.css dipindahkan ke *folder static/css/* (*folder css* terletak di dalam *folder static*)
 - c. server.py dan images.py dipindahkan ke *parent* dari *folder src* (atau di *folder root* dari *local repository* ini).
3. Buka *terminal/powershell* di *folder root local repository* ini dan ketik **python server.py** lalu tekan *enter*.

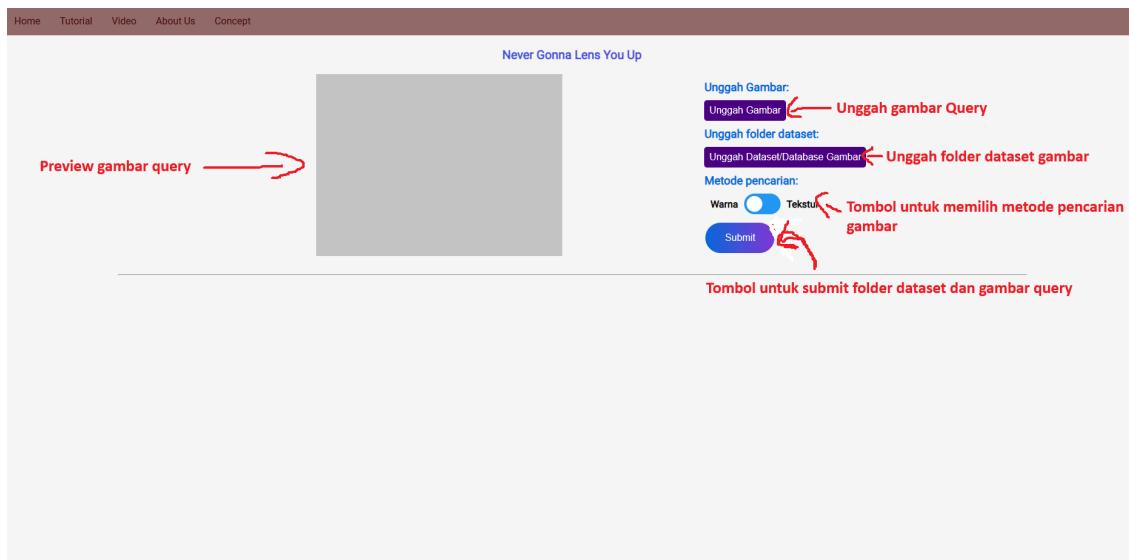


4. *Terminal/powershell* akan memberikan Anda sebuah *link* untuk menuju *website*. Ctrl + Klik *link* tersebut untuk membuka *website*.

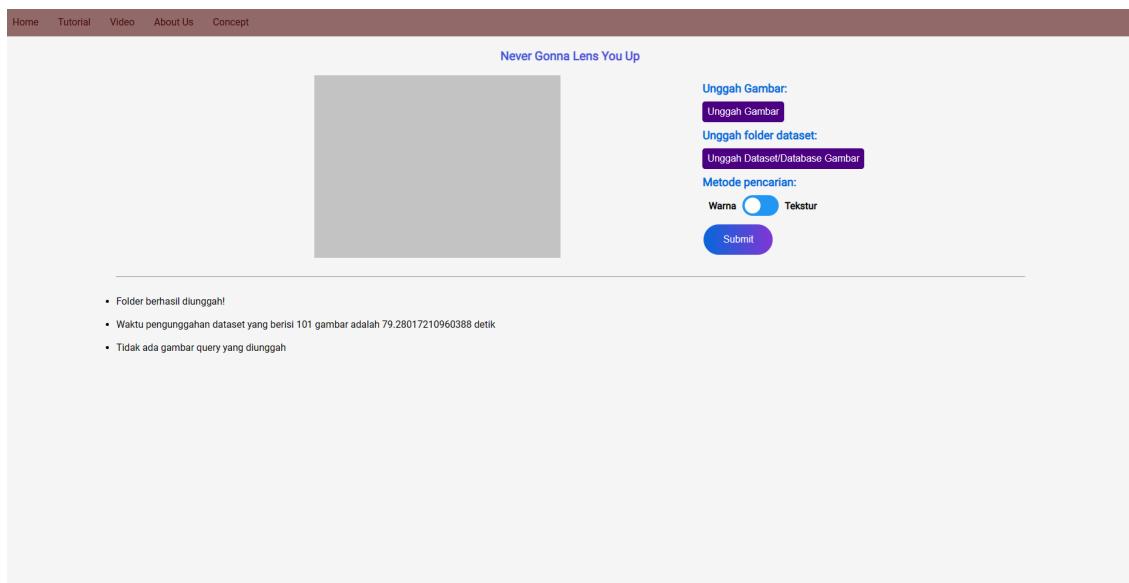


Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

5. Berikut adalah tampilan antarmuka *website* beserta fungsinya jika Anda baru pertama kali membuka website ini.



6. Unggah *dataset* dengan klik tombol **Unggah Dataset/Database Gambar** lalu pilih **folder** (bukan memilih kumpulan *file*). Pastikan *dataset* hanya berisi kumpulan *file* yang memiliki ekstensi berikut: .png, .jpg, .jpeg, .bmp. Setelah itu, tekan tombol **Submit** untuk mengunggah *dataset* tersebut ke server.
7. Setelah anda berhasil mengunggah *dataset* gambar, tampilan *website* akan berubah menjadi di bawah ini.



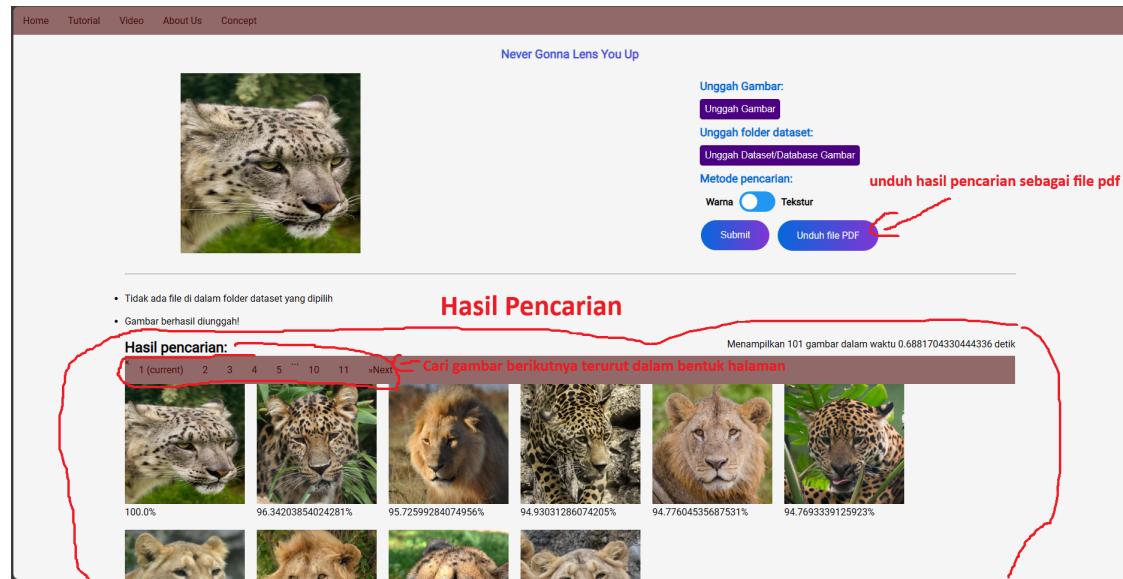
8. Unggah gambar *query* dengan tombol **Unggah Gambar** lalu pilih satu *file* gambar yang ingin dicari. Pastikan gambar *query* berformat salah satu dari ekstensi berikut: .png, .jpg, .jpeg, .bmp.

Tugas Besar 2 IF2123

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Kelompok 09 Tahun Ajaran 2023/2024

9. Pilih salah satu metode pencarian gambar dengan tombol yang berada di atas tombol **Submit**. Pilih antara metode CBIR dengan parameter warna atau parameter tekstur.
10. Setelah memilih metode pencarian gambar, klik tombol **Submit** untuk memulai pencarian gambar.
11. Jika pencarian selesai, maka akan muncul gambar-gambar yang sudah terurut berdasarkan kemiripannya dengan gambar *query*. Anda bisa melihat halaman berikutnya Anda juga bisa mengunduh hasil pencarian gambar sebagai *file pdf*.



12. Anda bisa mengulang proses mengunggah *dataset*, gambar *query*, dan pencarian sesuka hati Anda.

4.4. Hasil Pengujian

4.4.1 Dataset Acak

Parameter warna:

The screenshot shows a search interface for image datasets. At the top, there is a file upload section with buttons for "Unggah Gambar" (Upload Image), "Unggah folder dataset" (Upload Dataset/Folder), and "Unggah Dataset/Database Gambar" (Upload Dataset/Database). Below this is a "Metode pencarian:" (Search Method) section with a toggle switch between "Warna" (Color) and "Tekstur" (Texture), currently set to "Warna". There are also "Submit" and "Unduh file PDF" buttons. On the left, a large image of a lion's head is displayed. To its right is a list of search results showing various animal faces with their similarity scores. The first result is a lion with 100.0% similarity. The search results are paginated at the bottom with links for 1 (current), 2, 3, 4, 5, ..., 10, 11, and "Next". The total time taken for the search is 0.39798569679260254 detik.

Rank	Image	Similarity (%)
1	Lion (Current)	100.0%
2	Lion	97.90561000859965%
3	Lion	97.8342331822395%
4	Lion	97.7941489735904%
5	Cheetah	97.76745053065314%
6	Lion	97.6080849216102%
7	Lion	97.53863914734441%
8	Lion	97.28987552634312%
9	Cheetah	97.13641752315853%
10	Lion	97.10030233046825%

Parameter tekstur:

The screenshot shows a search interface for image datasets, similar to the previous one but using texture as the search method. The layout includes a file upload section, a search method toggle between "Warna" and "Tekstur" (set to "Tekstur"), and "Submit" and "Unduh file PDF" buttons. A large image of a lion's head is on the left. The search results show various animal faces with their similarity scores. The first result is a lion with 99.99999999999997% similarity. The search results are paginated at the bottom with links for 1 (current), 2, 3, 4, 5, ..., 10, 11, and "Next". The total time taken for the search is 0.48397111892700195 detik.

Rank	Image	Similarity (%)
1	Lion (Current)	99.99999999999997%
2	Lion	99.99999530928423%
3	Lion	99.99999491828379%
4	Lion	99.9999858847626%
5	Lion	99.99997461711354%
6	Wolf	99.99996338866627%
7	Cheetah	99.9999524556381%
8	Cheetah	99.99994179068638%
9	Wolf	99.9999336656394%
10	Tiger	99.99993059542639%

4.4.2. Dataset yang Mengandung Gambar yang Sama tetapi Dirotasi

Parameter warna:

The screenshot shows a search interface for image datasets. At the top, there is a large input field containing a blue Bulbasaur with green vines. To the right of the input field are several buttons: "Unggah Gambar" (Upload Image), "Unggah folder dataset" (Upload Dataset Folder), "Unggah Dataset/Database Gambar" (Upload Database/Image Database), "Metode pencarian:" (Search Method:), a toggle switch between "Warna" (Color) and "Tekstur" (Texture), a "Submit" button, and a "Unduh file PDF" (Download PDF File) button.

Below the search bar, a message states: "• Tidak ada file di dalam folder dataset yang dipilih" (No file selected in the dataset folder) and "• Gambar berhasil diunggah!" (Image uploaded successfully!).

The search results are titled "Hasil pencarian:" (Search Results:) and show a grid of 12 images. The first row contains six images of Bulbasaur with different orientations and slight variations in color. The second row contains four images: a green Bulbasaur, a red Bulbasaur, a black and white version of a character, and a blue version of a character. Below each image is its corresponding confidence score: 100.0%, 100.0%, 100.0%, 100.0%, 98.37110802929456%, 96.95456430270963%, 96.74404598173405%, 96.59356472739576%, 96.57809724580969%, and 96.29529754469189%.

At the bottom right of the results area, it says "Menampilkan 344 gambar dalam waktu 1.5825402736663818 detik" (Displaying 344 images in 1.5825402736663818 seconds).

Parameter tekstur:

This screenshot shows the same search interface but with the "Tekstur" (Texture) search method selected. The search bar still contains the image of a blue Bulbasaur.

The search results for texture-based search are also titled "Hasil pencarian:" and show a grid of 12 images. The first row contains six images of Bulbasaur with different orientations and slight variations in color. The second row contains four images: a red Bulbasaur, a blue Bulbasaur, a yellow and brown dragon-like creature, and a blue and purple dragon-like creature. Below each image is its corresponding confidence score: 100.0%, 100.0%, 100.0%, 100.0%, 99.99999999960062%, 99.99999999954834%, 99.9999999994927%, 99.9999999955878%, 99.99999999466927%, 99.99999999381532%, and 99.99999999381532%.

At the bottom right of the results area, it says "Menampilkan 344 gambar dalam waktu 0.6018581390380859 detik" (Displaying 344 images in 0.6018581390380859 seconds).

4.4.3. Dataset yang Mengandung Gambar yang Diedit Ukurannya tetapi Gambar Sama

Parameter warna:

Hasil pencarian:

- Tidak ada file di dalam folder dataset yang dipilih
- Gambar berhasil diunggah!

Diperbesar 200% Diperbesar 300% Diperkecil 50%

Menampilkan 345 gambar dalam waktu 0.9081704616546631 detik

Hasil pencarian:

1 (current) 2 3 4 5 ... 34 35 »Next

Asli

Diperbesar 200% Diperbesar 300% Diperkecil 50%

Menampilkan 345 gambar dalam waktu 0.9081704616546631 detik

Hasil pencarian:

«Previous 1 2 (current) 3 4 5 ... 34 35 »Next

Menampilkan 345 gambar dalam waktu 0.9081704616546631 detik

Unggah Gambar: Unggah Gambar
Unggah folder dataset: Unggah Dataset/Database Gambar
Metode pencarian: Warna Tekstur
Submit Unduh file PDF

Unggah Gambar: Unggah Gambar
Unggah folder dataset: Unggah Dataset/Database Gambar
Metode pencarian: Warna Tekstur
Submit Unduh file PDF

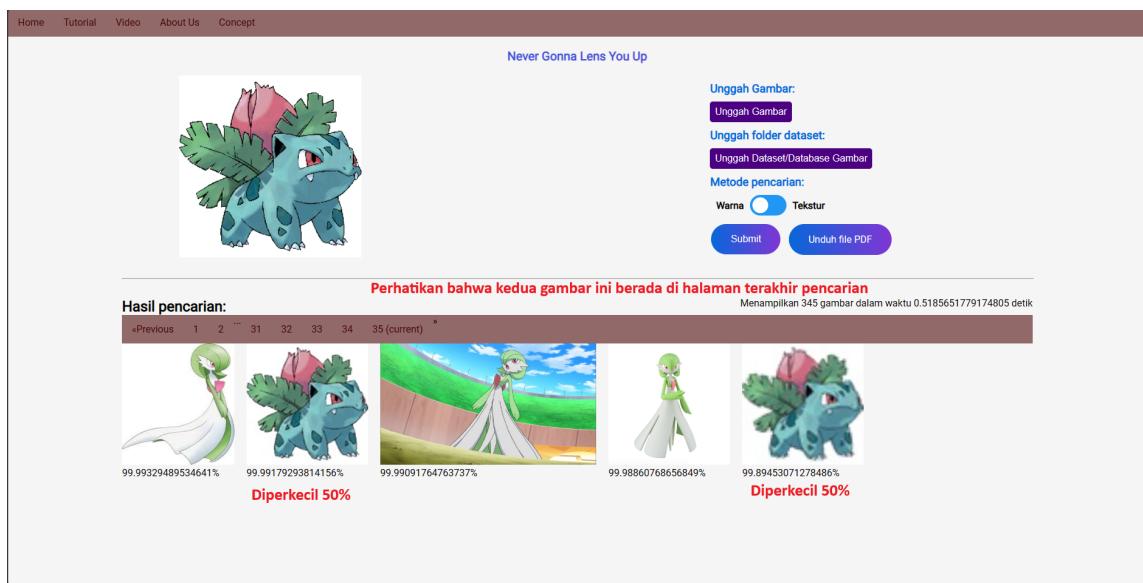
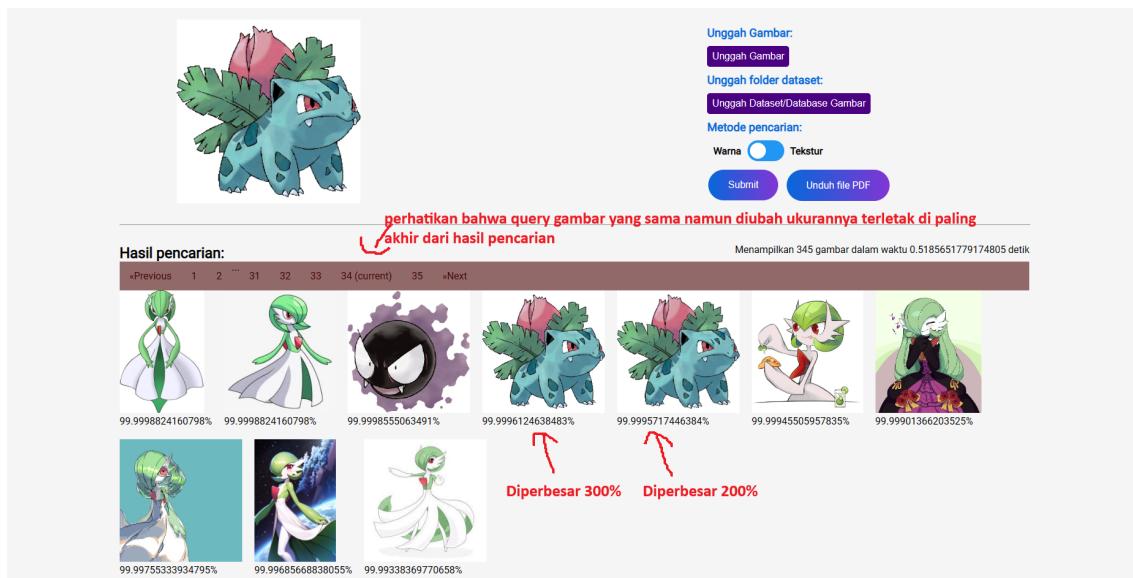
Hasil pencarian:

Menampilkan 345 gambar dalam waktu 0.9081704616546631 detik

Diperkecil 25%

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

Parameter teksur:



4.4.4. Dataset Gambar Grayscale tetapi Gambar Query Berwarna

Parameter warna:

The screenshot shows a search interface with a header bar containing 'Home', 'Tutorial', 'Video', 'About Us', and 'Concept'. Below the header, there's a title 'Never Gonna Lens You Up' above a color photograph of a street scene. To the right, there are upload options ('Unggah Gambar' and 'Unggah folder dataset') and a toggle switch for 'Metode pencarian' between 'Warna' (selected) and 'Tekstur'. Buttons for 'Submit' and 'Unduh file PDF' are also present. Below the main image, a list of bullet points says: '• Tidak ada file di dalam folder dataset yang dipilih' and '• Gambar berhasil diunggah!'. A message at the bottom states 'Menampilkan 1 gambar dalam waktu 0.5230734348297119 detik'. Under the heading 'Hasil pencarian:', a grayscale version of the same street scene is shown with a caption 'Parameter warna tidak mampu untuk menemukan gambar versi grayscale dari gambar query'.

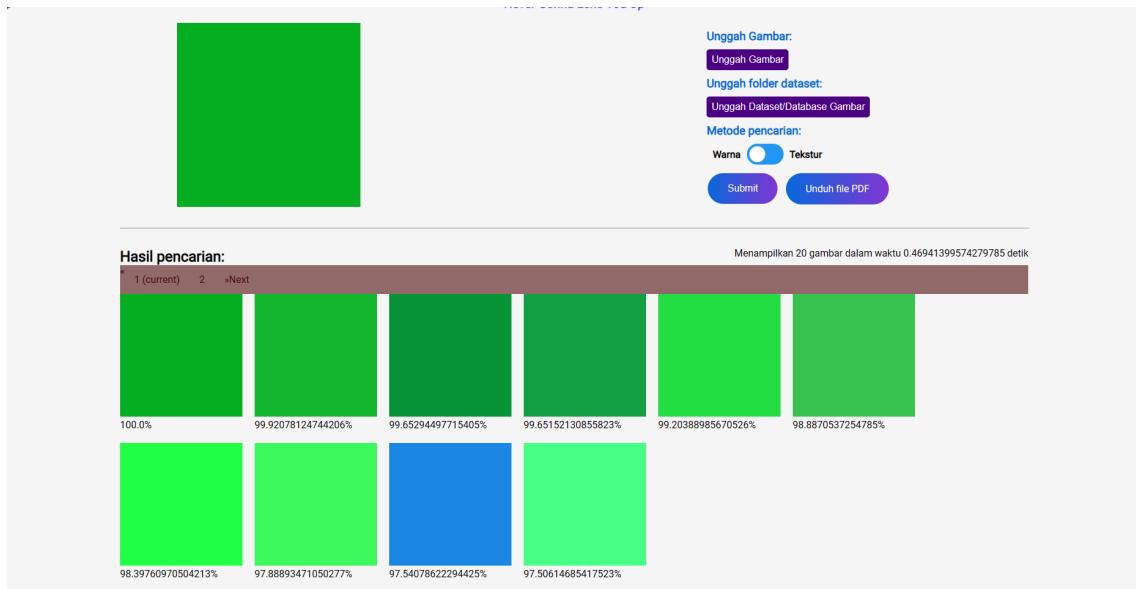
Parameter tekstur:

The screenshot shows a search interface similar to the previous one, but with the 'Tekstur' option selected. It displays a color photograph of a street scene as the query image. Below it, a row of six grayscale images is labeled 'Gambar versi grayscale dari query' with a red arrow pointing to it. The text 'terletak di halaman depan' is written above the first image in the results. The results page includes a navigation bar with '«Previous', page numbers (1, 2, 3, 4, 5, ..., 50, 51), and '»Next'. Below the navigation, there are two rows of grayscale images, each with a percentage value underneath. The top row includes images with percentages like 99.999990064713%, 99.99999874020739%, etc. The bottom row includes images with percentages like 99.9999964345646%, 99.99999609900787%, etc. A message at the bottom right says 'Menampilkan 502 gambar dalam waktu 2.571495294570923 detik'.

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

4.4.5. Dataset Warna Solid

Parameter warna:



Parameter tekstur:



4.4.6. Dataset Gambar Berwarna tetapi Gambar Query Grayscale

Parameter warna:

Unggah Gambar:

Unggah folder dataset:

Metode pencarian:
Warna Tekstur

Hasil pencarian:

Menampilkan 25 gambar dalam waktu 1.9430856704711914 detik

1 (current) 2 3 »Next

ID	Confidence (%)
73.68112983068052%	100.0%
72.2235123540208%	100.0%
71.34096542691819%	100.0%
67.68483149907915%	100.0%
80.1317075194912%	80.1317075194912%
66.17278111088375%	66.17278111088375%
63.94727494769656%	63.94727494769656%
63.86674413119261%	63.86674413119261%
63.566924743292766%	63.566924743292766%
63.125423171649274%	63.125423171649274%
62.7338725768391%	62.7338725768391%
62.34921706702037%	62.34921706702037%
62.251318319053084%	62.251318319053084%
62.06775313628887%	62.06775313628887%
61.63003644046019%	61.63003644046019%

Unggah Gambar:

Unggah folder dataset:

Metode pencarian:
Warna Tekstur

Hasil pencarian:

Menampilkan 25 gambar dalam waktu 1.9430856704711914 detik

«Previous 1 2 (current) 3 »Next

ID	Confidence (%)
66.17278111088375%	66.17278111088375%
63.94727494769656%	63.94727494769656%
63.86674413119261%	63.86674413119261%
63.566924743292766%	63.566924743292766%
63.125423171649274%	63.125423171649274%
62.7338725768391%	62.7338725768391%
62.34921706702037%	62.34921706702037%
62.251318319053084%	62.251318319053084%
62.06775313628887%	62.06775313628887%
61.63003644046019%	61.63003644046019%

Tugas Besar 2 IF2123
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar
Kelompok 09 Tahun Ajaran 2023/2024

The screenshot shows a search interface with a dark header bar containing navigation links: Home, Tutorial, Video, About Us, and Concept. Below the header is a search input field with the placeholder "Never Gonna Lens You Up". To the right of the input are several buttons: "Unggah Gambar" (Upload Image), "Unggah folder dataset" (Upload Dataset/Folder), and "Metode pencarian" (Search Method). The "Metode pencarian" section includes two radio buttons: "Warna" (Color) and "Tekstur" (Texture), with "Tekstur" being selected. Below these are "Submit" and "Unduh file PDF" buttons.

Hasil pencarian: «Previous 1 2 3 (current) »

Menampilkan 25 gambar dalam waktu 1.9430856704711914 detik

Below the search results, there is a horizontal row of five small images with their respective IDs below them:

- 61.14875254326929% (grayscale image of a city street)
- 61.10233271126397% (color image of a building)
- 60.84975148931490% (color image of a street scene)
- 60.71725997884852% (color image of a skyscraper)
- 60.61980717756967% (color image of a mountain landscape)

Perhatikan bahwa parameter warna tidak mampu untuk mencari gambar query versi berwarna dari query grayscale karena tidak ditemukan (kemiripan $\leq 60\%$).

Parameter tekstur:

The screenshot shows the same search interface as the previous one, but with the "Tekstur" (Texture) search method selected. The results are annotated with red arrows and text:

- A red arrow points to the first result with the text "terletak di depan".
- A red arrow points to the second result with the text "Versi berwarna dari gambar query".

Hasil pencarian: «Previous 1 2 (current) 3 4 5 ... 50 51 »Next

Menampilkan 502 gambar dalam waktu 0.5898258686065674 detik

Below the search results, there are two rows of small images with their IDs below them:

- 99.9999953030097% (grayscale image of a building)
- 99.99999892250936% (color image of a glacier)
- 99.9999985713156% (color image of a building)
- 99.99999848332158% (color image of a building)
- 99.99999841576994% (color image of a building)
- 99.99999781921628% (color image of a mountain landscape)

- 99.99999778652462% (color image of a snow-covered mountain)
- 99.99999747543093% (color image of a building)
- 99.99999694146337% (color image of a narrow street)
- 99.99999672025139% (color image of a street at night)

4.5. Analisis Desain Solusi Algoritma

1. Perhatikan bahwa nilai kemiripan gambar query dengan seluruh gambar dataset adalah 99% jika menggunakan parameter tekstur dan nilai kemiripan gambar query akan terus menurun hingga paling rendah sekitar 60% jika menggunakan parameter warna. Ini menunjukkan bahwa parameter tekstur kurang baik untuk digunakan jika kita ingin menilai kemiripan antara dua buah gambar secara umum. Akibatnya, untuk menilai kemiripan antara dua buah gambar, lebih baik menggunakan parameter warna.
2. Hasil dataset random menunjukkan bahwa parameter warna menampilkan gambar yang termirip yaitu berdasarkan warnanya, sedangkan parameter tekstur menunjukkan gambar yang termirip berdasarkan kontur/permukaan dan ukuran gambar.
3. Kedua parameter melakukan performa yang baik untuk mencari gambar yang sama namun dirotasi.
4. Parameter warna bekerja dengan sangat baik untuk mencari gambar yang sama namun diubah ukurannya. Parameter tekstur justru bekerja dengan sangat buruk untuk mencari gambar yang sama namun diubah ukurannya. Hal ini terjadi karena ukuran gambar akan sangat berpengaruh terhadap hasil yang ditampilkan oleh parameter tekstur, sedangkan parameter warna tidak karena parameter warna menilai kemiripan berdasarkan warnanya.
5. Parameter tekstur tidak bisa digunakan untuk mencari gambar warna solid karena tidak mengandung tekstur. Untuk mencari gambar warna solid harus menggunakan parameter warna.
6. Parameter warna tidak bisa digunakan untuk mencari versi grayscale dari dataset berwarna dan sebaliknya, karena yang dibandingkan pada kasus ini adalah tekstur dan kontur dari gambar. Untuk itu, digunakan parameter tekstur untuk mencarinya.

Bab 5: Kesimpulan

5.1. Kesimpulan

CBIR dengan parameter warna bekerja dengan cara membandingkan warna dari setiap piksel di antara dua gambar. Akibatnya, ukuran gambar dan rotasi tidak terlalu berpengaruh terhadap hasil pencarian gambar. Namun, parameter warna memiliki kekurangan, yaitu tidak bisa mencari versi *grayscale* dari *query* berwarna dan sebaliknya karena *grayscale* tidak memiliki warna sehingga sangat berdampak pada hasil dari parameter warna.

CBIR dengan parameter tekstur bekerja dengan cara membandingkan kontur dan tekstur dari dua buah gambar. Akibatnya, CBIR dengan parameter tekstur mampu untuk mencari versi *grayscale* dari gambar berwarna dan sebaliknya. Akan tetapi, CBIR dengan parameter tekstur tidak bisa digunakan untuk mencari gambar warna solid dan mencari gambar yang sama dengan ukuran berbeda karena ukuran gambar sangat mempengaruhi kemiripan di antara kedua gambar.

CBIR parameter warna dapat digunakan untuk membandingkan kemiripan gambar yang berbeda karena nilai *similarity* yang terus menurun, sedangkan CBIR parameter tekstur tidak dapat digunakan untuk menilai kemiripan gambar yang berbeda karena *similarity*-nya hampir selalu 99%. Oleh karena itu, dapat disimpulkan bahwa kedua parameter tersebut berguna di kondisi yang berbeda pula.

5.2. Saran

Sebaiknya revisi yang berkaitan dengan penilaian konsep utama (CBIR) dilakukan di sekitar awal pengerjaan tugas besar atau saat baru saja rilis, bukan di tengah-tengah pengerjaan tugas besar agar tidak menimbulkan kebingungan dan perubahan backend mendadak.

5.3. Komentar atau Tanggapan

Tubesnya menarik juga karena kita bisa membuat Google Images/Google Lens versi mini. ~~Mayan buat portofolio~~ - Akbar

Cukup menarik dan menambah minat saya pada penerapan Aljabar Linear dan Geometri. - Brian

Keren, tubes pertama yang disuruh bikin website sungguhan. - Dzaky

5.4. Refleksi terhadap Tugas Besar

Ada kekurangan dari *website* yang kami buat, yaitu *website* kami tidak mampu untuk mengunggah *file-file* berukuran sangat besar dan tidak mampu untuk mengunggah *dataset* yang sangat banyak (lebih dari sekitar 900 gambar berformat .jpg). Kami sudah mencoba untuk

Tugas Besar 2 IF2123

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Kelompok 09 Tahun Ajaran 2023/2024

memperbesar kapasitas *upload file* tapi tetap tidak bisa dilakukan meskipun sudah mencoba mengubah konfigurasi kapasitas unggah *file* di *server Flask*. Hal ini disebabkan oleh keterbatasan Python dan Flask itu sendiri. Bahasa Python termasuk *interpreted language*, akibatnya Python berjalan lebih lambat daripada *compiled language* seperti C++, C, dan Go. Selain itu, Flask adalah *framework web development* yang bersifat *lightweight* sehingga Flask kurang cocok digunakan untuk *website* berskala besar. Namun, kami berhasil membuat sebuah *website* kecil yang mampu berfungsi seperti Google Images dan Google Lens dengan bahasa dan *framework* yang sederhana.

5.5. Ruang Perbaikan atau Pengembangan

Setelah mengetahui kekurangan dari Python dan Flask, kami akan membuka diri untuk mengeksplorasi lebih jauh tentang bahasa pemrograman untuk *back end* yang lain untuk *web development* seperti Go, C++, C, Java, Javascript, dan Typescript, serta *framework web development* yang lain, seperti Django, Vue, dan React.

Daftar Pustaka

[Kumar, Kamlesh & Li, Jian-Ping & Zain-ul-abidin, & Shaikh, Riaz. \(2016\). Content Based Image Retrieval Using Gray Scale Weighted Average Method. International Journal of Advanced Computer Science and Applications. 7. 10.14569/IJACSA.2016.070101.](#)
[<https://niqahahoster.co.id/blog/website-development-adalah/>](#)
[<http://www.fpdf.org/>](#)
[<https://www.geeksforgeeks.org/os-module-python-examples/>](#)
[<https://www.pinecone.io/learn/series/image-search/color-histograms/>](#)
[<https://www.scaler.com/topics/np-vectorize/>](#)
[<https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1>](#)

Link Repository

[<https://github.com/Kizaaaa/Algeo02-22036>](#)

Link Video

[<https://www.youtube.com/watch?v=hf5lr5d7ctY>](#)