Tugas Besar 2 IF2123 Aljabar Linier dan Geometri

Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Semester I Tahun 2023/2024

Disusun oleh kelompok Nonstop Nubes November

Devinzen (13522064)

Sa’ad Abdul Hakim (13522092)

Matthew Vladimir Hutabarat (13522093)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

Daftar Isi

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.

A. Tujuan

1. Implementasi sistem temu balik gambar dengan memanfaatkan Aljabar Vektor dalam bentuk sebuah *website.*
2. Implementasi aljabar vektor untuk menggambarkan dan menganalisis data menggunakan pendekatan klasifikasi berbasis konten (*Content-Based Image Retrieval* atau CBIR), dengan mengidentifikasi gambar berdasarkan warna dan tekstur.

B. Spesifikasi

1. Program menerima input *folder dataset* dan sebuah citra gambar.
2. *Dataset* gambar dapat diunduh secara mandiri melalui pranala berikut. Akan tetapi peserta diperbolehkan untuk menggunakan *dataset* lain yang telah dipersiapkan oleh kelompok masing-masing.
3. Program menampilkan gambar citra gambar yang dipilih oleh pengguna.
4. Program dapat memberikan kebebasan pada pengguna untuk memilih parameter pencarian yang hendak digunakan (warna atau tekstur) melalui *toggle*. *Default* parameter yang digunakan di awal dibebaskan kepada masing-masing kelompok.
5. Program mulai melakukan perhitungan nilai kecocokan antara *image* masukan dengan *dataset image* berdasarkan parameter yang telah dipilih (warna atau tekstur).
6. Program dapat menampilkan nilai kecocokan antara *image* masukan dengan setiap gambar dalam dataset.
7. Program menampilkan hasil luaran dengan melakukan *descending sorting* berdasarkan nilai kecocokan tiap gambar. Cukup tampilkan seluruh gambar yang **memiliki tingkat kemiripan > 60%** dengan gambar masukan.

8. Program mengimplementasikan *pagination* agar jumlah gambar dapat dibatasi dengan halaman-halaman tertentu. Jumlah gambar dalam dataset yang akan digunakan oleh asisten saat penilaian mungkin berbeda dengan jumlah gambar pada dataset yang kalian gunakan, sehingga pastikan bahwa *pagination* dapat berjalan dengan baik.
9. Program dapat menampilkan **jumlah gambar** yang memenuhi kondisi tingkat kemiripan serta **waktu eksekusi.**

BAB 2

Landasan Teori

2.1 Dasar Teori

2.1.1 *CONTENT-BASED INFORMATION RETRIEVAL* (CBIR)

Content-Based Information Retrieval (CBIR) merupakan pendekatan dalam pengambilan informasi yang menekankan pada karakteristik visual atau konten dari data gambar. Dalam CBIR, pencarian informasi dilakukan berdasarkan fitur-fitur intrinsik gambar, seperti warna, tekstur, bentuk, dan relasi spasial antar objek. Implementasi CBIR yang praktis, umumnya bergantung pada fitur-fitur rendah seperti warna, bentuk, dan tekstur. Warna seringkali menjadi fitur yang efektif karena keberlanjutan, keefektifan, sederhana implementasinya, dan keuntungan penyimpanan yang rendah. CBIR juga memanfaatkan fitur-fitur tekstur untuk menangkap pola dan struktur repetitif dalam suatu gambar. Perkembangan teknologi komputer dan jaringan memungkinkan penyimpanan dan transmisi besar jumlah gambar, sehingga CBIR menjadi solusi untuk mengelola basis data gambar dengan fokus pada konten visual, bukan teks atau anotasi.

2.1.2 Perkalian titik (dot product)

Dasar teori dot product (atau dot product disebut juga inner product atau scalar product) adalah operasi matematika yang menghasilkan skalar dari dua vektor. Dot product antara dua vektor, misalnya **A** dan **B** dengan dimensi yang sama, dinyatakan sebagai:

Produk skalar dua vektor $\mathbf{A} = [A_1, A_2, ..., A_n]$ dan $\mathbf{B} = [B_1, B_2, ..., B_n]$

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n$$

Dot product memiliki aplikasi luas dalam berbagai bidang, termasuk aljabar linear, analisis vektor, dan machine learning. Misalnya, dalam machine learning, dot product sering digunakan dalam perhitungan kesamaan antar vektor, seperti dalam metode seperti cosine similarity.

2.1.3 Norma vektor

Norma vektor merupakan ukuran panjang atau magnitudo dari suatu vektor dalam ruang vektor. Norma vektor seringkali diukur menggunakan norma Euclidean, juga dikenal sebagai norma L2, yang dinyatakan sebagai akar kuadrat dari jumlah kuadrat setiap elemen vektor. Jika **V** adalah suatu vektor dengan

elemen V_1, V_2, …, V_n maka norma Euclidean dari vektor **V** (‖ **V** ‖) didefinisikan sebagai:

$$\|\mathbf{v}\|=\sqrt{v_1^2+v_2^2+\ \ldots+\ v_n^2}$$

Norma vektor memiliki banyak aplikasi, termasuk dalam machine learning, optimisasi, dan analisis numerik.

2.1.4 *Cosine Similarity*

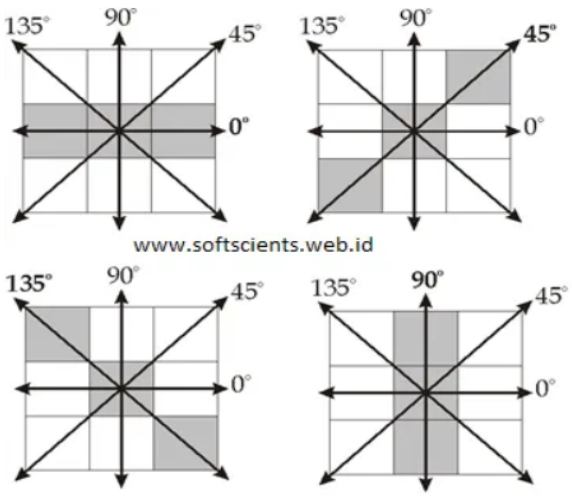
Cosine similarity adalah metrik kesamaan yang digunakan untuk mengukur sejauh mana dua vektor arah mendekati satu sama lain. Dalam konteks cosine similarity, vektor dianggap sebagai representasi dari dokumen atau fitur, dan metrik ini sering digunakan dalam Information Retrieval (IR), Text Mining, dan Machine Learning. Cosine similarity antara dua vektor **A** dan **B** dalam ruang vektor dihitung dengan menggunakan formula berikut:

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

Dengan *A* dan *B* adalah vektor dan *n* adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *Cosine Similarity*. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

2.1.5 *Gray-Level Co-occurrence matrix (GLCM)*

Gray-Level Co-occurrence matrix (GLCM) merupakan teknik analisis tekstur pada citra. GLCM merepresentasikan hubungan antara 2 pixel yang bertetanggaan (neighboring pixels) yang memiliki intensitas keabuan (grayscale intensity), jarak dan sudut. Terdapat 8 sudut yang dapat digunakan pada GLCM, diantaranya sudut 0°, 45°, 90°, 135°, 180°, 225°, 270°, atau 315°. Parameter jarak pada GLCM dihitung dengan banyaknya pixel antara pixel reference dan pixel neighbor.



2.2 Penjelasan singkat mengenai pengembangan sebuah *website*

Framework yang kami gunakan dalam pengembangan website adalah Flask. Flask adalah framework web yang ringan dan bersifat mikro untuk bahasa pemrograman Python. Dengan fokus pada kesederhanaan dan fleksibilitas, Flask memudahkan pengembang dalam membangun aplikasi web dengan cepat. Dengan menggunakan Flask, pengembang dapat dengan mudah membuat rute URL, mengelola permintaan HTTP, dan merender tampilan HTML. Flask juga mendukung ekstensibilitas melalui penggunaan ekstensi, memungkinkan integrasi dengan basis data, otentikasi, dan fitur-fitur lainnya. Kesederhanaan Flask membuatnya cocok untuk proyek-proyek kecil hingga menengah, dan fleksibilitasnya memungkinkan pengembang untuk memilih alat dan library yang sesuai dengan kebutuhan spesifik proyek.

BAB 3

Analisis Pemecahan Masalah

3.1 Langkah-langkah pemecahan masalah

Sebuah gambar terdiri dari unit-unit kecil yang disebut piksel. Setiap piksel memiliki tiga elemen warna utama, yakni *red* (merah), *green* (hijau), dan *blue* (biru), yang diwakili oleh nilai numerik tertentu. Dengan menggunakan data ini, kita dapat membentuk sebuah matriks yang menyimpan ketiga nilai tersebut, menciptakan suatu representasi matematis dari gambar tersebut.

CBIR dengan parameter warna memanfaatkan informasi komponen warna untuk menunjukkan sejauh mana suatu gambar mirip dengan gambar lainnya dapat dilakukan dengan menggunakan pendekatan fitur warna. Dalam hal ini, kita dapat menggunakan komponen gambar dalam format Hue-Saturation-Value (HSV). Sebelum menghitung kemiripan, gambar perlu diubah ke format HSV. Pada pendekatan ini, penilaian kemiripan dilakukan menggunakan cosine similarity. Proses ini melibatkan representasi matriks dari gambar yang diubah menjadi histogram, yang kemudian membentuk vektor histogram. Cosine similarity dihitung dengan membandingkan kedua vektor tersebut, dan hasilnya dievaluasi. Nilai yang mendekati 1 menunjukkan semakin tinggi tingkat kemiripan antara kedua gambar dan sebaliknya.

CBIR dengan parameter tekstur menggunakan tiga elemen warna tersebut untuk mengubahnya menjadi suatu warna grayscale karena warna tidaklah penting dalam penentuan tekstur. Selanjutnya dari nilai grayscale tersebut akan dibentuk sebuah

co-occurrence matrix agar bisa memperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Dari ketiga komponen tersebut, dibuatlah sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan menggunakan *cosine similarity*.

3.2 Proses pemetaan masalah menjadi elemen-elemen pada aljabar geometri

CBIR dengan parameter warna:

1. Didapat nilai RGB dari sebuah gambar yang berbentuk array yang nilainya dirata-ratakan setiap blok 4x4
2. Setiap elemen di array RGB tersebut diubah ke bentuk HSV
3. Dibuatlah array histogram kemunculan warna-warna HSV itu
4. Array histogram itu akan digunakan sebagai vektor untuk menentukan *Cosine Similarity*

CBIR dengan parameter tekstur:

1. Didapat nilai RGB dari sebuah gambar yang berbentuk array nilai red, green, dan blue yang kemudian membentuk matriks RGB dari gambar dengan ukuran sesuai piksel gambar tersebut
2. Setiap array RGB tersebut kemudian dikonversi menjadi sebuah nilai grayscale yang kemudian membentuk matriks grayscale.
3. Dari matriks grayscale kemudian bisa dibentuk *co-occurrence matrix* dengan $\theta = 0^o$ dan jarak = 1.
4. Dari matriks *co-occurrence* tersebut kemudian dibuat *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya.
5. Kemudian dibentuk matriks normalisasi dengan membagi *symmetric matrix* dengan jumlah semua elemen *symmetric matrix.*
6. Selanjutnya akan diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity* menggunakan persamaan masing-masing dan didapat vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan dengan menggabungkan 3 komponen tersebut.
7. Vektor tersebut kemudian digunakan untuk mengukur kemiripan dari kedua gambar dengan menggunakan Teorema *Cosine Similarity* yang menggunakan perkalian titik dan norma vektor dalam perhitungannya.. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

3.3 Contoh ilustrasi kasus dan penyelesaiannya

CBIR dengan parameter warna

Contoh ilustrasi kasus adalah membandingkan gambar berdasarkan parameter warna. Semakin serupa komposisi warna suatu gambar, maka semakin mirip gambar yang dibandingkan.

Penyelesaian:

1. Mengubah gambar yang dibandingkan menjadi sebuah matriks RGB kemudian ubah nilai RGB menjadi HSV.
2. Plotting nilai-nilai HSV itu ke sebuah histogram yang menyatakan kemunculan warna-warna pada interval HSV tertentu.
3. Nilai-nilai di histogram itu dijadikan sebuah vektor
4. Hitung kemiripan dua gambar itu dengan *cosine similarity* dari vektor yang didapatkan

CBIR dengan parameter tekstur

Contoh ilustrasi kasus adalah membandingkan gambar berdasarkan parameter tekstur. Semakin serupa tekstur dari suatu gambar, maka semakin mirip gambar yang dibandingkan.

Penyelesaian:

1. Mengubah gambar yang dibandingkan menjadi sebuah matriks RGB kemudian ubah nilai RGB sehingga menjadi sebuah matriks grayscale.
2. Bentuk sebuah matriks *co-occurrence* dari matriks grayscale tersebut kemudian didapat sebuah matriks normalisasi dari sebuah *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya.
3. Kemudian hitung 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity* dan bentuk menjadi sebuah vektor.
4. Bandingkan masing-masing vektor gambar dengan menggunakan *cosine similarity.*
5. Semakin besar sebuah nilai *cosine similarity* maka semakin mirip kedua vektor gambar yang berarti semakin mirip kedua gambar.

BAB 4

Implementasi dan Uji Coba

4.1 Implementasi program utama

CBIR_warna.py

```
USE numpy as np
function rgb_to_hsv (real r, real g, real b) → (real, real, real)
        KAMUS LOKAL
        Cmax, Cmin, d, H, S: real

        ALGORITMA
        Cmax ← max(r, g, b)
        Cmin ← min(r, g, b)
        d ← Cmax - Cmin
        if (d = 0) then
                H ← 0
        else
                if (Cmax = r) then
                        H ← 60 * (((g - b) / d) % 6)
                else
                        if (Cmax = g) then
                                H ← 60 * (((b - r) / d) + 2)
                        else
                                H ← 60 * (((r - g) / d) + 4)
        if (Cmax = 0) then
                S ← 0
        else
                S ← d / Cmax
        → H, S, Cmax

function cosine_similarity (array of integer Vector1, array of integer Vector2) → real
        KAMUS LOKAL
        dot_product: integer
        Vector1_Length, Vector2_Length, similarity: real

        ALGORITMA
        dot_product ← np.sum(np.multiply(Vector1, Vector2))
        Vector1_Length ← np.sqrt(np.sum(np.square(Vector1)))
        Vector2_Length ← np.sqrt(np.sum(np.square(Vector2)))
        similarity ← dot_product / (Vector1_Length * Vector2_Length)
        → similarity
```

CBIR_texture.py

```
USE numpy as np
procedure gscale_to_co_occur(matrix, mat_cooccur)
        KAMUS LOKAL
        height, width : integer

        ALGORITMA
        matrix ← np.array(matrix)
        height, width ← get_matrix_shape(matrix)

        i traversal [0..height-1]
           j traversal  [0..width-2]
              mat_cooccur[matrix[i][j]][matrix[i][j+1]] ←
              mat_cooccur[matrix[i][j]][matrix[i][j+1] + 1

function sym_matrix(matrix) →  Matrix of Integer
        KAMUS LOKAL

        ALGORITMA
        matrix ← np.array(matrix)
        → matrix + matrix.T

procedure normalized_matrix(matrix, mat_norm)
        KAMUS LOKAL
        total_sum : integer

        ALGORITMA
        matrix ← np.array(matrix)
        total_sum ← matrix.sum()
        mat_norm[:] ← matrix / total_sum

function contrast(matrix) → float
        KAMUS LOKAL
        i, j : integer

        ALGORITMA
        matrix ← np.array(matrix)
        i, j ← np.indices(matrix.shape)
        → np.sum(matrix * (pow(i-j,2)))

function homogeneity(matrix) → float
        KAMUS LOKAL
        i, j : integer

        ALGORITMA
        matrix ← np.array(matrix)
        i, j ← np.indices(matrix.shape)
        → np.sum(matrix / (1 + (pow(i-j,2))))
```

```
function entropy(matrix) → float
        KAMUS LOKAL
        not_zero_elements : Matrix of float

        ALGORITMA
        matrix ← convert_to_numpy_array(matrix)
        non_zero_elements ← matrix[matrix ≠ 0]
        → return -np.sum(not_zero_elmts * np.log10(not_zero_elmts))

function rgb_to_gscale(R, G, B) → integer
        KAMUS LOKAL

        ALGORITMA
        → round(0.299 * R + 0.587 * G + 0.114 * B)
```

4.2 Penjelasan struktur program berdasarkan spesifikasi (dapat membahas terkait arsitektur kode secara keseluruhan, struktur pembangunan website, atau hal-hal lain yang berkaitan dengan struktur program).

Tubes kali ini diharuskan untuk membuat website yang membutuhkan frontend dan backend. Pada backend kami memilih python karena memiliki library terkait image processing dan juga bahasa yang nyaman bagi kami, untuk frontend kami memilih framework bernama React.js. React membantu kami membuat website antarmuka dengan cepat karena bisa dibuat oleh component component yang lebih kecil untuk menggabungkan frontend dan backend kami menggunakan framework flask beserta sqlalchemy dari flask juga untuk menyimpan dataset.

4.3 Penjelasan tata cara penggunaan program (dapat membahas terkait interface program, fitur-fitur yang disediakan program, atau hal-hal lain yang berkaitan dengan cara penggunaan program).

Untuk menjalankan program kita harus ke membuat 2 terminal, flask-server dan website-algeo. Berikut cara menjalankan program:

1. Di flask-server kita harus membuat virtual environment dengan mengetikkan py -3 -m venv venv
2. Selanjutnya diaktivasi dengan venv\Scripts\activate
3. Pastikan terlebih dahulu untuk menginstall semua package yang diperlukan
4. Jalankan program dengan mengetik python app.py lalu flask run
5. Terminal flask-server sudah bisa berjalan, selanjutnya masuk ke terminal website-algeo
6. Kita dapat langsung mengetik npm start untuk mulai.

Di website terdapat menu Reverse Image Research (program), How to Use, dan About Us.

4.4 Hasil pengujian (screenshot antarmuka program dan beberapa data uji beserta skenario pengujian). Diharapkan bahwa setiap kelompok menyampaikan hasil pengujian untuk beberapa test case yang berbeda, mulai dari variasi gambar masukan pencarian hingga dataset.

CBIR warna

Test case 1	
Test case 2	
Test case 3	
Test case 4	

CBIR tekstur

Test case 1	
Test case 2	
Test case 3	
Test case 4	

4.5 Analisis dari desain solusi algoritma pencarian yang diimplementasikan pada setiap pengujian yang dilakukan.

Berdasarkan hasil pengujian menggunakan algoritma yang kami buat, CBIR berdasarkan fitur warna cenderung kurang baik dibandingkan fitur tekstur pada kasus gambar yang memiliki warna yang mirip.

Berdasarkan hasil pengujian menggunakan algoritma yang kami buat, CBIR berdasarkan fitur tekstur cenderung kurang baik dibandingkan fitur warna pada kasus gambar yang memiliki bentuk yang mirip.

BAB 5

5.1 Kesimpulan

Dari kuliah IF2123 Aljabar Linier dan Geometri, kami mengambil konsep-konsep materi yang kami dapat dan mengimplementasikannya dalam aplikasi sistem temu balik gambar dalam bentuk sebuah website. Program yang kami buat mampu menggambarkan dan menganalisis data menggunakan pendekatan klasifikasi berbasis konten (*Content-Based Image Retrieval* atau CBIR) dengan memanfaatkan aljabar vektor dan beberapa operasi matriks.

Melalui tugas besar ini, kami dapat memahami cara pemrosesan sebuah gambar dan mengaplikasikannya dalam sistem temu balik gambar dengan bahasa Python dalam bentuk sebuah website yang menggunakan framework Flask. Kami juga dapat mempelajari banyak hal seperti pengaplikasian materi Aljabar Linier dan Geometri terutama aljabar vektor dan matriks, mengolah dan membandingkan gambar, dan pembuatan sebuah website.

5.2 Saran

- Mempelajari lebih dalam algoritma dalam memproses dan membandingkan gambar agar algoritma yang dibuat dapat memiliki akurasi yang tepat serta lebih efisien dalam memproses gambar
- Mempelajari lebih dalam mengenai pembuatan sebuah website serta framework yang digunakan

5.3 Komentar atau Tanggapan

Tugas besar ini membantu kami mengetahui contoh pengaplikasian dari materi aljabar vektor dan matriks yang sudah kami pelajari dalam bentuk proses dan membandingkan gambar. Tugas ini juga membuat kami mempelajari pembuatan sebuah website baik *front-end* maupun *back-end*.

5.4 Refleksi terhadap tugas besar ini.

Kami dapat mengimplementasikan materi kuliah Aljabar Linear dan Geometri IF2123 yang dipelajari di kelas ke dalam sebuah aplikasi sistem temu balik gambar berbahasa python dalam bentuk website. Melalui tugas besar ini, kami mengetahui lebih dalam mengenai aljabar vektor, operas-operasi matriks, dan aplikasinya. Kami belajar banyak mulai dari bagaimana cara menggunakan algoritma untuk mengambil data dari sebuah gambar hingga membandingkannya dengan gambar lain menggunakan bahasa Python. Selain itu kami juga belajar bagaimana membuat sebuah website menggunakan framework Flask. Kami juga belajar bagaimana cara berkoordinasi dan bekerja sama dengan baik dalam kelompok menggunakan github.

5.5 Ruang Perbaikan atau Pengembangan

Algoritma yang kami buat masih dapat tentu masih dapat dikembangkan lebih lanjut. Seperti, algoritma CBIR yang kami rancang belum mencapai tingkat kesempurnaan, sehingga masih terbuka peluang untuk pengembangan lebih lanjut. Pendekatan CBIR dengan fokus pada fitur warna dapat ditingkatkan melalui implementasi pembobotan pada blok piksel. Dengan membagi citra ke dalam blok-blok dan menetapkan bobot berdasarkan perhitungan nilai HSV, kami percaya bahwa akurasi metode CBIR dengan fitur warna dapat meningkat secara signifikan.

Selain itu, pada algoritma CBIR tekstur, sistem temu balik gambar masih belum begitu akurat sehingga bisa ditingkatkan dengan menggunakan beberapa nilai θ dalam pembuatan *co-occurence matrix* dan menggunakan lebih banyak komponen ekstraksi tekstur dalam vektor yang dihasilkan sehingga akurasi dari CBIR tekstur dalam sistem temu balik gambar menjadi lebih akurat.

Daftar Pustaka

https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1

Link repository: https://github.com/NgokNgok04/Algeo02-22064