

IF2123 Aljabar Linear dan Geometri
APLIKASI ALJABAR VEKTOR DALAM SISTEM TEMU BALIK GAMBAR

Laporan Tugas Besar 2

Disusun untuk memenuhi tugas mata kuliah Aljabar Linear dan Geometri
pada Semester 1 (satu) Tahun Akademik 2023/2024



Oleh

Shabrina Maharani **13522134**

Atqiya Haydar Luqman **13522163**

Valentino Chryslie Triadi **13522164**

Kelompok GoMilk

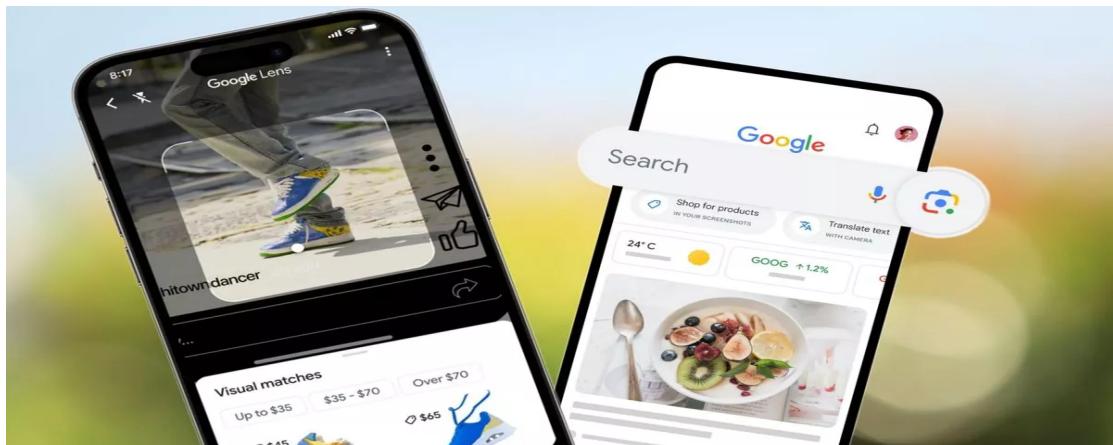
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
SPESIFIKASI TUGAS	3
BAB 2 LANDASAN TEORI	6
2.1 Pengolahan Citra	6
2.1.1 Pengenalan Citra	6
2.1.2 Citra Digital	7
2.1.2.5.1 RGB	11
2.1.2.5.2 HSV	13
2.1.3 Content-Based Information Retrieval (CBIR)	15
2.2 Website	19
2.2.1 Pengembangan Website	19
BAB 3 ANALISIS PEMECAHAN MASALAH	21
3.1 Langkah-Langkah Pemecahan Masalah	21
3.2 Proses Pemetaan Masalah	25
3.3 Contoh Ilustrasi Kasus	26
BAB 4 IMPLEMENTASI DAN UJI COBA	27
4.1 Implementasi Program Utama	27
4.1.1 CBIR dengan Parameter Warna	27
4.1.2 CBIR dengan Parameter Tekstur	32
4.2 Struktur Program Berdasarkan Spesifikasi	37
4.3 Tata Cara Penggunaan Program	39
4.4 Hasil Pengujian	40
4.5 Analisis Solusi	43
BAB 5	45
5.1 Kesimpulan	45
5.2 Saran	45
5.3 Komentar atau Tanggapan	46
5.4 Refleksi	46
5.5 Ruang Perbaikan atau Pengembangan	47
DAFTAR PUSTAKA	49
LAMPIRAN	50

BAB 1 DESKRIPSI MASALAH

Gambar merupakan suatu bentuk representasi, kesamaan, atau tiruan dari suatu objek. Gambar mengandung informasi terkait objek yang diwakilinya. Oleh karena itu, gambar memiliki kapasitas untuk menyampaikan informasi yang lebih kaya dibandingkan data teks. Pada era modern saat ini, jumlah gambar yang dihasilkan dan disimpan mengalami pertumbuhan yang signifikan, baik dalam ranah pribadi maupun profesional. Peningkatan ini mencakup beragam jenis gambar, mulai dari foto pribadi, gambar medis, ilustrasi ilmiah, hingga gambar komersial. Meskipun sumber dan jenis gambar sangat beragam, keberadaan sistem temu balik gambar (image retrieval system) menjadi sangat relevan dan esensial dalam mengatasi tantangan yang ada. Dengan dukungan dari 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, termasuk pencarian gambar pribadi, analisis gambar medis untuk diagnosis, pencarian ilustrasi ilmiah, dan bahkan pencarian produk berdasarkan gambar komersial. Sebagai contoh, Google Lens adalah salah satu implementasi sistem temu balik gambar yang mungkin sudah familiar bagi banyak orang.



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

Dalam Tugas Besar 2 Aljabar Linear dan Geometri, semua mahasiswa yang terdaftar di IF2123 diminta untuk menerapkan sistem temu balik gambar yang telah dijelaskan sebelumnya dengan memanfaatkan Aljabar Vektor dalam bentuk sebuah situs web. Pendekatan ini memiliki peran krusial dalam dunia pemrosesan data dan pencarian informasi. Dalam konteks ini, aljabar vektor digunakan untuk menggambarkan dan menganalisis data melalui pendekatan klasifikasi berbasis konten, yang dikenal sebagai Content-Based Image Retrieval (CBIR). Dengan metode ini, sistem temu balik gambar dapat mengidentifikasi gambar berdasarkan konten visualnya, seperti warna dan tekstur. Pada Tugas Besar 2 ini, hanya diminta untuk mengimplementasikan 2 parameter CBIR yang paling populer, yaitu warna dan tekstur.

SPESIFIKASI TUGAS

Program sistem temu balik gambar (*image retrieval system*) dibuat dengan spesifikasi sebagai berikut:

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**.

BONUS

Bagian ini hanya boleh dikerjakan apabila spesifikasi wajib dari Tugas Besar telah berhasil dipenuhi. Anda **tidak diharuskan untuk mengerjakan keseluruhan bonus**, tetapi semakin banyak bonus yang dikerjakan, maka akan semakin banyak tambahan nilai yang diperoleh.

Bagian A (Kamera)

1. Terdapat fitur kamera yang dapat melakukan penangkapan gambar secara *real-time* menggunakan *webcam* ketika program dijalankan.
2. Dilarang menambahkan *button* untuk *trigger* pada fitur kamera. **HINT:** Gunakan interval waktu untuk melakukan penangkapan gambar.
3. Fitur kamera merupakan fitur **tambahan**, sehingga fitur utama *upload* gambar melalui *website* tetap harus ada.

Bagian B (*Image Scraping*)

1. Implementasikan fitur *scraping* untuk melakukan ekstraksi gambar dari sebuah *website* tertentu.
2. Hasil dari *scraping* akan digunakan oleh program sebagai input dataset gambar.
3. Fitur *scraping* merupakan fitur **tambahan**, sehingga fitur utama *upload* gambar melalui *website* tetap harus ada.

Bagian C (Video)

1. Video penjelasan algoritma dan aplikasi program yang diunggah ke *youtube*. Referensi dapat kalian peroleh dengan kata kunci : Tubes 2 Algeo, Tugas Besar 2 Aljabar Linear dan Geometri, dan lain-lain. Berikut ini beberapa contoh referensi video yang disarankan:

- [Face Recognition using EigenFace Method - YouTube](#)

- [Video ALGEO02 - 21109 - Eigenface - YouTube](#)
 - [Tubes 2 IF2123 Algeo: Face Recognition with Eigenface \[YUKBISAYUK\] - YouTube](#)
 - [Tubes Algeo Face Recognition - YouTube](#)
2. Video dibuat sekreatif mungkin dengan target untuk mensosialisasikan ilmu yang kalian sudah pelajari dan terapkan pada program ini. Bukan hanya video mengenai penggunaan aplikasi.
 3. **Dilarang membuat video yang secara keseluruhan hanya dilakukan melalui online video conference, seperti: Google Meet, Zoom, dll.** Penggunaan *video conference* masih diperbolehkan pada saat **melakukan demonstrasi penggunaan program**. Hal ini dilakukan untuk mendorong interaksi antar masing-masing anggota kelompok.
 4. Tautan video yang dikumpulkan tidak boleh menggunakan *url shortener* seperti bit.ly, tinyurl.com, dll.

Bagian D (Kreatifitas)

Segala bentuk kreatifitas fungsionalitas yang kalian buat akan mendapat apresiasi dari asisten. Beberapa fungsionalitas yang **disarankan** antara lain:

1. *Caching result* dari perhitungan sebuah dataset menggunakan *output file* tertentu, misalnya: csv, fvecs, dll. Hal ini akan meningkatkan efisiensi dan efektifitas ketika suatu dataset hendak digunakan berulang kali.
2. *Export* hasil luaran program dalam format PDF yang dapat diunduh ke *local storage* oleh pengguna. Hasil luaran dapat mencakup gambar masukan serta gambar dalam dataset dan nilai kecocokannya.

BAB 2 LANDASAN TEORI

2.1 Pengolahan Citra

2.1.1 Pengenalan Citra

Citra atau sebuah gambar dikatakan memiliki makna lebih dari seribu kata yang berarti citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi.

Citra sering disebut juga gambar dalam bidang dwimatra (dua dimensi). Dari perspektif matematis, citra merupakan fungsi kontinu dari intensitas cahaya di bidang dua dimensi yang dapat diamati oleh sistem visual manusia. Secara matematis, citra pada bidang dwimatra adalah fungsi dwimatra yang merepresentasikan intesitas cahaya.

$$f(x, y)$$

(x, y) : koordinat pada bidang dwimatra

$f(x, y)$: intensitas cahaya (*brightness*) pada titik (x, y)

Gambar 2. Citra sebagai fungsi kontinu dari intensitas cahaya

Citra yang terlihat adalah hasil pantulan cahaya dari suatu objek, di mana sumber cahaya memberikan iluminasi, objek memantulkan sebagian cahaya tersebut, dan pantulan cahaya tersebut direkam oleh perangkat optik seperti mata manusia, kamera, scanner, sensor satelit, dan lain sebagainya. Citra merupakan output dari suatu sistem perekaman data yang bisa bersifat optik, analog, maupun digital. Citra dapat dibedakan menjadi citra diam dan citra bergerak. Citra diam adalah sebuah citra tunggal. Citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan sebagai gambar yang bergerak. Proses perekaman citra dapat dibagi menjadi dua kategori, yaitu:

1. Citra Analog:

Citra analog terdiri dari sinyal-sinyal elektromagnetik yang tidak dapat dibedakan secara spesifik sehingga umumnya tidak memiliki ukuran yang terukur. Citra analog memiliki fungsi yang kontinu. Hasil perekaman citra analog dapat berbentuk optik, seperti foto pada film foto konvensional, atau berupa sinyal video, seperti gambar pada layar televisi.

2. Citra Digital:

Citra digital terdiri dari sinyal-sinyal yang dapat dibedakan dan memiliki fungsi yang tidak kontinu, terbentuk oleh titik-titik warna. Hasil

perekaman citra digital dapat disimpan pada media magnetik (Munir, 2014:30). Biasanya dapat langsung disimpan pada disk atau pita magnetik.

2.1.2 Citra Digital

Citra digital adalah representasi citra melalui pencuplikan secara waktu dan ruang. Pencuplikan secara ruang berdasarkan koordinat sinyal (x,y) dan pencuplikan secara waktu merupakan sederetan citra yang bergerak atau biasanya disebut sebagai video digital. Citra digital biasanya berupa kumpulan nilai angka riil atau kompleks yang direpresentasikan dalam bentuk bit tertentu. Umumnya, citra ini memiliki bentuk persegi panjang dengan tinggi dan lebar sebagai dimensinya. Dalam konteks ini, citra digital dapat dianggap sebagai fungsi intensitas cahaya $f(x, y)$, di mana x dan y adalah koordinat spasial, dan nilai fungsi tersebut pada setiap titik (x, y) menunjukkan tingkat kecerahan citra pada titik tersebut. Ukuran citra digital $N \times M$ diwakili oleh matriks berukuran N baris dan M kolom, dengan setiap elemen disebut piksel atau *picture element*. Proses digitalisasi koordinat (x, y) dikenal sebagai pencuplikan citra (image sampling), sedangkan proses digitalisasi derajat keabuan $f(x, y)$ disebut kuantisasi derajat keabuan (gray-level quantization).

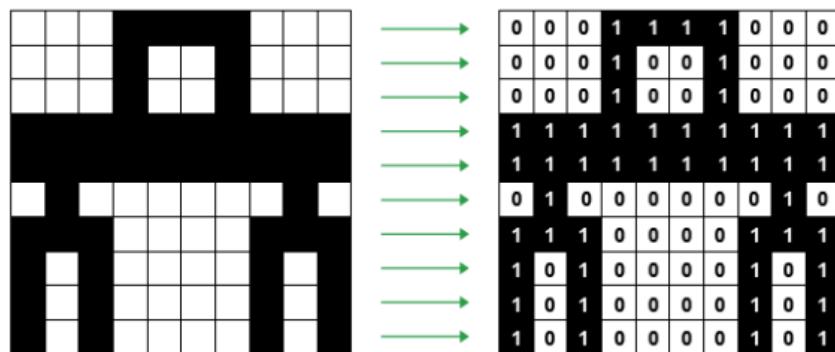
Citra, sebagai representasi optis objek yang disinari oleh sumber radiasi, terdiri dari berkas cahaya yang dipantulkan oleh objek. Iluminasi sumber cahaya dan koefisien pantul objek berperan dalam menentukan intensitas cahaya pada setiap titik citra. Citra digital memiliki beberapa karakteristik, termasuk ukuran citra, resolusi, dan format nilai. Format citra digital dapat berupa citra biner, skala keabuan (grayscale), warna, warna berindeks, atau film radiografi, yang merupakan citra fisik menunjukkan distribusi materi atau energi dari radiasi pengion. Pengolahan citra digital melibatkan transformasi citra pada format digital dan melakukan operasi-operasi pemrosesan sinyal oleh komputer. Format input dan output dari sistem pengolahan citra digital adalah citra digital.

2.1.2.1 Jenis Citra Digital

Setiap piksel dalam citra digital memiliki nilai yang berada dalam rentang tertentu, mulai dari nilai minimum hingga nilai maksimum. Rentang ini bervariasi tergantung pada jenis warna yang digunakan, dan umumnya berkisar antara 0 hingga 255. Terdapat beberapa jenis citra berdasarkan nilai pikselnya.

1. Citra Biner adalah citra digital yang hanya memiliki dua kemungkinan nilai piksel, yakni hitam dan putih. Citra biner adalah citra yang memiliki hanya dua nilai *graylevel*, 0 (Putih) dan 1 (Hitam). Representasi nilai piksel dalam citra biner hanya

membutuhkan satu bit. Contoh citra biner dapat dilihat pada gambar di bawah ini.



Gambar 3. Citra Biner

2. Citra Grayscale adalah citra digital yang memiliki satu nilai kanal pada setiap pikselnya, artinya nilai bagian merah (red), hijau (green), dan biru (blue) sama. Citra grayscale memiliki kedalaman warna sebanyak 8 bit, mencakup 256 kombinasi warna keabuan. Contoh citra grayscale dapat dilihat pada gambar 2.2 (Putra, 2017:23). Citra grayscale merupakan gambar yang hanya memiliki satu saluran (channel). Setiap piksel di dalam citra ini hanya merepresentasikan jumlah cahaya dalam gambar, memberikan informasi intensitas. Beberapa metode dapat digunakan untuk mengubah citra berwarna menjadi citra grayscale, seperti Metode Rerata dan Metode Berbobot.



Gambar 4. Citra Grayscale

Proses konversi citra berwarna menggunakan metode rerata sesuai dengan formula pada Rumus (1). Namun, metode rerata dianggap kurang sesuai dengan persepsi mata manusia terhadap citra, sehingga pembobotan untuk setiap warna perlu disesuaikan [1].

Sebagai opsi lain, dikembangkan Metode Berbobot, yang menitikberatkan pada komponen warna hijau karena mata manusia lebih responsif terhadap warna tersebut. Formula yang digunakan untuk Metode Berbobot diberikan pada Rumus (2).

$$g = \frac{1}{3}(R + G + B) \quad (1)$$

$$g = 0,299R + 0,587G + 0,114B \quad (2)$$

Gambar 5. Rumus Citra Grayscale

3. Citra Warna (24-Bit) adalah citra yang setiap pikselnya direpresentasikan dengan 24-bit, menghasilkan total 16.777.216 variasi warna. Jumlah variasi ini lebih dari cukup untuk merepresentasikan seluruh warna yang dapat dilihat oleh manusia. Setiap informasi warna disimpan dalam 1-byte data, dengan 8-bit pertama untuk nilai biru, 8-bit kedua untuk nilai hijau, dan 8-bit terakhir untuk nilai merah (Santoso, 2018:10).



Gambar 6. Citra Warna

2.1.2.2 Format File Citra Digital

File gambar berperan sebagai wadah untuk menyimpan gambar yang dapat ditampilkan di layar, disimpan dalam suatu media penyimpanan data dengan menggunakan format gambar tertentu. Setiap format gambar memiliki ciri khasnya sendiri. Beberapa format umum yang digunakan saat ini meliputi bitmap (.bmp), tagged image format (.tif, tiff), portable network graphics (.png), graphics interchange format (.gif), jpeg (.jpg), mpeg (.mpg), dan lainnya. Dalam tugas besar ini, hanya format gambar jpeg (.jpg) yang digunakan dan akan dibahas dalam landasan teori. Hal ini karena format gambar lainnya memiliki nilai yang tidak hanya terbatas pada kombinasi warna RGB.

2.1.2.2.1 JPEG (*Joint Photographic Experts Group*)

Format jpeg atau jpg adalah jenis format citra yang memiliki sejumlah kelebihan dan kekurangan jika dibandingkan dengan format citra lainnya. Salah satu kelebihan yang umum adalah fleksibilitas dalam ukuran file dan kualitas citra yang dihasilkan. Namun, kekurangan yang dapat disoroti adalah bahwa kualitas citra mungkin tidak selalu sesuai dengan citra aslinya. Format JPEG mendukung mode warna RGB, CMYK, dan Grayscale, tetapi tidak mendukung citra dengan latar belakang transparan. Format JPEG menguraikan informasi tersebut menjadi komponen luminance (komponen cahaya) dan dua komponen chromatic (komponen perubahan warna dari hijau ke merah dan dari biru ke kuning). Proses kompresi file citra dalam format ini menggunakan metode kompresi JPEG (Prihatini, 2018:45).

2.1.2.3 Elemen-Elemen Citra Digital

Citra digital melibatkan sejumlah elemen dasar yang menjadi fokus dalam pengolahan citra dan aplikasi dalam *computer vision*. Elemen-elemen kunci tersebut melibatkan:

1. Kecerahan: Istilah lain untuk intensitas cahaya, di mana kecerahan pada titik tertentu dalam citra sebenarnya mencakup intensitas rata-rata area sekitarnya. Sistem visual manusia memiliki kemampuan menyesuaikan tingkat kecerahan dari yang paling rendah hingga tertinggi.
2. Kontras: Istilah yang menyatakan distribusi antara terang dan gelap dalam gambar. Citra dengan kontras rendah cenderung memiliki komposisi dominan yang terang atau gelap, sementara citra dengan kontras baik menyebar secara merata.
3. Kontur (Contour): Terbentuk oleh perubahan intensitas pada piksel-piksel bertetangga, memungkinkan mata manusia mendekripsi tepi atau kontur objek dalam citra.
4. Warna: Merupakan persepsi sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna memiliki panjang gelombang yang berbeda, dan kombinasi warna tertentu seperti merah, hijau, dan biru memberikan rentang warna yang luas.
5. Bentuk (Shape): Properti intrinsik objek tiga dimensi yang sering kali diidentifikasi oleh manusia dengan bentuknya. Meskipun citra umumnya dwimatra, informasi bentuk objek dapat diekstraksi pada tahap awal pengolahan citra.

6. Tekstur: Dicirikan oleh distribusi spasial derajat keabuan di antara piksel-piksel yang bertetangga. Tekstur tidak dapat didefinisikan untuk satu piksel saja, karena sistem visual manusia memandang citra sebagai suatu kesatuan dan menerima informasi citra secara holistik pada skala tertentu.

2.1.2.4 Warna

Manusia mampu melihat radiasi elektromagnetik pada rentang panjang gelombang 400 hingga 700 nanometer sebagai warna. Beberapa hewan juga memiliki kemampuan melihat spektrum elektromagnetik dengan sisi yang berbeda, memungkinkan mereka melihat warna yang tidak dapat dilihat oleh manusia.

Pengalaman manusia terhadap warna merupakan hasil dari proses kombinasi antara mata dan otak. Mata berfungsi sebagai penerima cahaya, sementara otak menginterpretasikan data dari mata sebagai informasi visual dan menerjemahkannya menjadi warna. Penglihatan manusia didasarkan pada tiga penerima, satu untuk merah, satu lagi untuk hijau, dan sisanya untuk biru. Berbagai model warna biasanya memiliki tiga atau empat saluran (channel) yang merepresentasikan perbedaan lingkup warna.

2.1.2.5.1 RGB

Sistem yang umum digunakan untuk merepresentasikan warna adalah sistem RGB (Red, Green, Blue). RGB merupakan sistem penggabungan warna primer (additive primary colours), yaitu merah, hijau, dan biru, untuk menghasilkan warna tertentu. Contohnya, warna putih diperoleh dari kombinasi merah = 255, hijau = 255, dan biru = 255, dan dinyatakan dalam sistem RGB sebagai RGB (255, 255, 255). Rentang nilai setiap warna primer dalam sistem RGB adalah 0 hingga 255. Dengan sistem RGB, terdapat sekitar 16,7 juta kemungkinan warna yang dapat terbentuk ($256 \times 256 \times 256$). Tabel warna untuk sistem RGB menampilkan beberapa kode warna yang dihasilkan dari kombinasi warna RGB (Siregar Arifin, 2016:11).

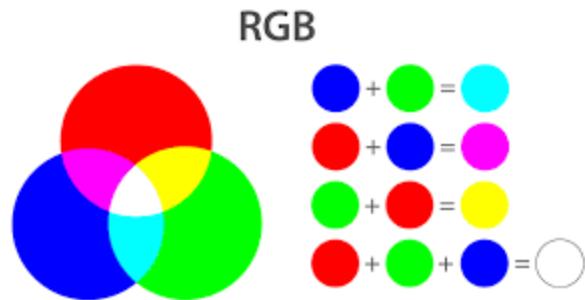
TABEL SISTEM WARNA	R	G	B
	0	0	0
Hitam	0	0	0
Coklat	165	42	42
Merah	255	0	0
Oranye	255	165	0
Kuning	255	255	0
Hijau	0	255	0
Biru	0	0	255
Ungu	127	0	255
Abu-Abu	128	128	128
Putih	255	255	255
Emas	255	215	0
Perak	192	192	192

Gambar 6. Tabel RGB

Dalam ruangan yang sepenuhnya gelap, tanpa adanya cahaya, kondisi tersebut disebut sebagai gelap total. Pada situasi ini, mata tidak menyerap sinyal gelombang cahaya, dan representasinya dalam model warna RGB adalah (0,0,0). Jika kita memasukkan cahaya merah ke dalam ruangan tersebut, warna ruangan akan berubah menjadi merah, seperti yang diwakili oleh nilai RGB (255,0,0). Dalam kondisi ini, semua objek dalam ruangan hanya terlihat dalam warna merah.

Prinsip dasar ini juga berlaku ketika menggunakan cahaya hijau atau biru. Berdasarkan teori tri-stimulus vision yang menyatakan bahwa manusia melihat warna dengan membandingkan cahaya yang diterima oleh sensor peka cahaya di retina (yang berbentuk kerucut), di mana sensor ini paling responsif terhadap cahaya dengan panjang gelombang 630 nm (merah), 530 nm (hijau), dan 450 nm (biru). Model ini dapat dijelaskan sebagai sebuah kubus dengan sumbu R, G, dan B (Santoso, 2018:50).

Seiring dengan itu, penggunaan sistem RGB memberikan fleksibilitas dalam menghasilkan berbagai warna dengan mengatur intensitas masing-masing warna primer. Sistem ini mendasari representasi warna pada berbagai perangkat elektronik dan aplikasi grafis.



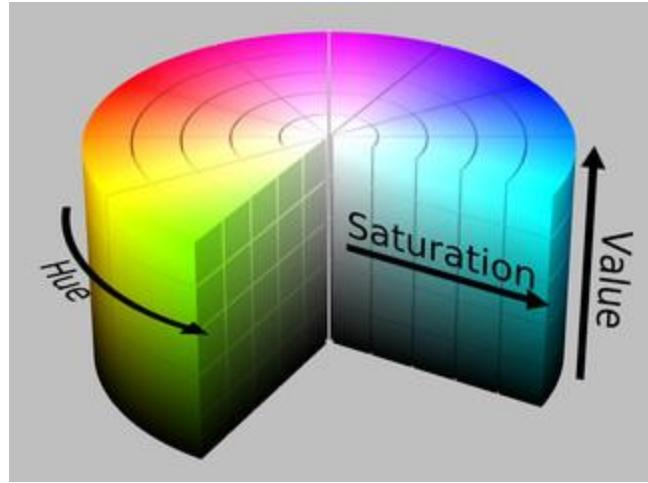
Gambar 7. RGB

2.1.2.5.2 HSV

Model warna HSV menggambarkan warna dalam istilah Hue, Saturation, dan Value. Hue mengindikasikan warna sejati seperti merah, violet, dan kuning, membedakan warna-warna dan menentukan tingkat kemerahan, kehijauan, dan sebagainya, dari cahaya. Hue berhubungan dengan panjang gelombang cahaya. Saturation menunjukkan tingkat kemurnian suatu warna, menandakan seberapa banyak warna putih dicampurkan ke dalam warna tersebut. Value adalah atribut yang mencerminkan sejauh mana cahaya diterima oleh mata, tanpa mempertimbangkan warna.

Dalam model warna HSV, Hue menggambarkan dimensi lingkaran warna, di mana setiap titik di lingkaran tersebut mewakili warna dasar, dan perpindahan sepanjang lingkaran menghasilkan variasi warna. Saturation mengukur seberapa jauh suatu warna dari warna netral (putih atau abu-abu), sedangkan Value menentukan seberapa terang atau gelap suatu warna.

Secara umum, model warna HSV memberikan representasi yang lebih intuitif dan mudah dimengerti terhadap karakteristik warna, terutama dalam konteks desain grafis dan pengolahan citra. Hal ini memungkinkan pengguna untuk dengan lebih mudah memanipulasi dan memahami aspek-aspek visual warna pada suatu gambar. Model warna ini sering digunakan dalam berbagai aplikasi, termasuk pengeditan gambar dan desain grafis, karena memfasilitasi kontrol yang lebih baik atas parameter warna. (Sutoyo, 2017:60).



Gambar 8. HSV

Model warna HSV (Hue, Saturation, Value) merupakan salah satu model warna alternatif selain model warna RGB dan HSL. Pada model HSV, Value adalah nilai yang mengindikasikan seberapa banyak warna yang terdapat pada suatu warna tertentu. Sama seperti Saturation, nilai Value ini berkisar antara 0 hingga 1. Jika nilai Value adalah 0, warna tersebut akan menjadi hitam.

Sebaliknya, jika nilai Value adalah 1, tingkat kandungan warna hitam pada warna tersebut akan hilang. Untuk mengkonversi nilai model warna HSV dari model warna RGB, dapat menggunakan rumus berikut.

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

$$S = \begin{cases} \frac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq S \leq 1$$

$$H = \begin{cases} \frac{60(G-B)}{V - \min(R, G, B)} & \text{if } V = R \\ 120 + \frac{60(B-R)}{V - \min(R, G, B)} & \text{if } V = G \\ 240 + \frac{60(R-G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases}$$

$$\text{if } H < 0 \text{ then } H = H + 360$$

$$0 \leq H \leq 360$$

Gambar 9. Rumus HSV

Dalam rumus tersebut, V mewakili nilai Value, S mewakili nilai Saturation, dan H mewakili nilai Hue. Rentang nilai untuk Value dan Saturation adalah antara 0 hingga 1, sedangkan rentang nilai untuk Hue adalah antara 0 hingga 360. Model warna HSV memiliki struktur ruang warna yang berbeda dari model warna RGB. Gambar 3 di bawah ini memberikan gambaran yang lebih jelas. Pada gambar tersebut, puncak nilai Value mencakup seluruh warna yang tidak mengandung warna hitam. Oleh karena itu, yang mempengaruhi warna adalah nilai Hue dan Saturation. Ini sangat berbeda dengan model HSL, di mana nilai L setara dengan 1 dan tidak dapat diubah oleh nilai Hue dan Saturation.

2.1.3 Content-Based Information Retrieval (CBIR)

Sistem temu balik gambar berbasis konten (Content-Based Image Retrieval atau CBIR) merupakan suatu proses untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan mengekstrak fitur-fitur kunci dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur ini diambil, mereka diubah menjadi vektor atau deskripsi numerik untuk dibandingkan dengan gambar lain. CBIR kemudian menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur gambar dalam dataset. Hasil pencocokan ini digunakan untuk menyusun dan menampilkan gambar-gambar dalam dataset, dengan mengutamakan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna mengakses dan menjelajahi koleksi gambar secara lebih efisien, tanpa perlu menggunakan pencarian berdasarkan teks atau kata kunci, melainkan dengan mempertimbangkan kesamaan nilai citra visual antara gambar-gambar tersebut. Pada Tugas Besar kali ini, hanya diminta untuk mengimplementasikan 2 parameter CBIR yang paling populer, yaitu parameter warna dan tekstur.

1. CBIR dengan parameter warna

Dalam CBIR ini, perbandingan dilakukan antara input dari suatu gambar dengan gambar-gambar yang ada dalam dataset. Pendekatan yang digunakan melibatkan konversi gambar RGB menjadi metode histogram warna yang lebih umum. Histogram warna merupakan representasi frekuensi warna yang ada dalam suatu ruang warna tertentu. Tujuan dari pendekatan ini adalah untuk mendistribusikan warna dalam gambar. Meskipun histogram warna tidak mampu mengidentifikasi objek spesifik dalam gambar atau menggambarkan posisi distribusi warna, tetapi memberikan informasi tentang frekuensi kemunculan berbagai warna.

Proses pembentukan ruang warna dilakukan untuk membagi nilai citra menjadi beberapa rentang yang lebih kecil, sehingga membentuk histogram warna dengan setiap interval rentang dianggap sebagai "bin". Perhitungan histogram warna melibatkan penghitungan jumlah piksel yang memiliki nilai warna dalam setiap interval.

Dalam menghitung histogram warna, penggunaan warna global dalam format HSV lebih disukai. Pilihan ini diambil karena warna HSV lebih cocok untuk latar belakang berwarna putih yang umumnya digunakan pada kertas. Oleh karena itu, konversi warna dari RGB ke HSV diperlukan sebagai langkah awal dalam prosedur ini.

Proses content-based image retrieval (CBIR) dengan parameter warna melibatkan beberapa langkah kunci. Pertama, untuk memudahkan perhitungan, gambar diubah menjadi *array of RGB* terlebih dahulu. Kemudian, array diubah menjadi nxn blok. Setelah itu, nilai-nilai RGB dalam setiap blok perlu dinormalisasi untuk mengubah rentang [0, 255] menjadi [0, 1].

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

Gambar 9. Normalisasi RGB

Selanjutnya, Cmax (nilai maksimum), Cmin (nilai minimum), dan Δ (delta) dihitung untuk nilai-nilai yang sudah dinormalisasi.

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

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

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

Gambar 10. Mencari Delta

Setelah itu, nilai-nilai HSV dihitung berdasarkan hasil perhitungan sebelumnya.

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

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

$$V = C_{max}$$

Gambar 11. Menghitung nilai HSV

Proses selanjutnya melibatkan perbandingan antara gambar input dan dataset menggunakan cosine similarity, di mana A dan B mewakili vektor, dan n adalah jumlah dimensi vektor. Tingkat kemiripan diukur berdasarkan hasil cosine similarity. Perbandingan dilakukan antar blok ke-n di gambar A dan gambar B menggunakan rumus di bawah ini.

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

Gambar 12. Cosine Similarity

Seluruh proses ini membantu dalam melakukan pencarian dan pengambilan gambar berdasarkan kontennya dengan fokus pada parameter warna.

2. CBIR dengan parameter tekstur

Pada CBIR dengan perbandingan tekstur, digunakan suatu matriks yang disebut co-occurrence matrix. Penggunaan matriks ini dipilih karena memberikan kemudahan dan kecepatan dalam pemrosesan, serta menghasilkan vektor dengan ukuran yang lebih kecil. Misalnya, dalam gambar I dengan $n \times m$ piksel dan parameter offset $(\Delta x, \Delta y)$, co-occurrence matrix dapat dirumuskan sebagai berikut:

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

Gambar 13. Pembentukan co-occurrence matrix

Dengan menggunakan nilai i dan j sebagai intensitas gambar, serta p dan q sebagai posisi dalam gambar, offset Δx dan Δy bergantung pada arah θ dan jarak yang ditentukan oleh persamaan di atas.

Langkah pertama yang dapat dilakukan untuk membuat CBIR dengan parameter tekstur adalah dengan melakukan konversi warna gambar ke dalam skala abu-abu (grayscale) karena warna tidak dianggap penting dalam menentukan tekstur. Proses ini melibatkan pengubahan warna RGB menjadi warna abu-abu Y dengan menggunakan rumus berikut.

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

Gambar 14. Cosine Similarity

Setelah konversi warna, elemen pada matrix grayscale yang tadinya bertipe bilangan riil dibulatkan menjadi nilai integer dan dilakukan kuantifikasi nilai grayscale, dimana matriks yang berkoresponden akan berukuran 256×256 sesuai dengan ukuran citra grayscale yang memiliki 256 piksel. Berdasarkan persepsi manusia, tingkat kemiripan gambar dievaluasi berdasarkan kekasaran teksturnya.

I	1	2	3	4	5	6	7	8
1	1	1	5	6	8			
2	2	3	5	7	1			
4	4	5	7	1	2			
8	8	5	1	2	5			

GLCM	1	2	3	4	5	6	7	8
1	1	2	0	0	1	0	0	0
2	0	0	1	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	0
5	1	0	0	0	0	1	2	0
6	0	0	0	0	0	0	0	1
7	2	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0

Gambar 15. Pembentukan co-occurrence matrix

Setelah didapat *co-occurrence matrix*, dapat dibuat *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya. Lalu cari *matrix normalization* dengan persamaan.

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

Gambar 16. Normalisasi

Dari *symmetric matrix*, dapat diekstrak tiga komponen ekstraksi tekstur, yaitu contrast, entropy, dan homogeneity. Masing-masing komponen dapat dihitung menggunakan persamaan yang disediakan, dengan P sebagai matriks co-occurrence.

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 :

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

Gambar 17. Contrast, Homogeneity, Entropy

Selanjutnya, dari ketiga komponen tersebut, dibentuk sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan antara dua gambar. Kemiripan antara kedua gambar diukur menggunakan Teorema Cosine Similarity, dengan vektor A dan B sebagai representasi dari dua gambar yang dibandingkan. Semakin besar hasil Cosine Similarity dari kedua vektor, semakin tinggi tingkat kemiripannya.

2.2 Website

Website merupakan suatu kumpulan halaman atau dokumen yang dapat diakses melalui internet dan disajikan dalam format teks, gambar, audio, atau video. Secara umum, website adalah medium digital yang memungkinkan individu atau organisasi untuk berbagi informasi, menyampaikan pesan, atau menyediakan layanan kepada pengguna. Dalam konteks khusus, website dapat berfungsi sebagai alat pemasaran, platform bisnis, atau wadah untuk berinteraksi dengan pengguna.

Pengertian website secara umum mencakup struktur dasar, yaitu halaman-halaman yang saling terkait dan dapat diakses dengan menggunakan peramban web. Konten dalam website dapat melibatkan berbagai elemen, seperti teks informatif, grafik, formulir interaktif, dan aplikasi web yang lebih kompleks. Seiring dengan perkembangan teknologi, website tidak hanya sebagai sumber informasi, tetapi juga sebagai medium dinamis yang memfasilitasi interaksi dua arah antara pengguna dan penyedia konten.

2.2.1 Pengembangan Website

Pengembangan website adalah suatu proses yang melibatkan tahapan merancang, membuat, dan memelihara struktur serta konten suatu situs web. Proses ini mencakup berbagai aspek seperti desain antarmuka, fungsionalitas,

keamanan, dan performa, dengan tujuan menciptakan pengalaman pengguna yang optimal. Tahap awal dalam pengembangan website dimulai dengan perencanaan, di mana tujuan situs, audiens target, dan fitur-fitur yang diperlukan ditentukan. Selanjutnya, desain antarmuka dan struktur informasi situs web dirancang untuk memberikan pengalaman pengguna yang baik.

Pengembangan dilanjutkan dengan implementasi desain melalui pemrograman, pengembangan fungsionalitas, dan integrasi sistem. Pengujian fungsionalitas, keamanan, dan kinerja menjadi langkah kritis sebelum peluncuran, yang melibatkan penyusunan situs web agar dapat diakses oleh pengguna. Dalam hal teknologi dan platform, pengembangan website memanfaatkan berbagai teknologi seperti HTML, CSS, JavaScript, serta backend frameworks seperti Django, Ruby on Rails, atau Express.js. Penggunaan CMS seperti WordPress atau Drupal mempermudah manajemen konten.

Dalam konteks pengembangan modern, responsivitas dan ramah seluler sangat penting. Website harus dapat diakses dengan baik dari berbagai perangkat, termasuk smartphone dan tablet. Aspek keamanan juga diperhatikan, termasuk enkripsi data, manajemen hak akses, dan perlindungan terhadap serangan siber.

Setelah diluncurkan, pemeliharaan dan pembaruan rutin diperlukan untuk menjaga keberlanjutan dan keamanan website. Proses ini melibatkan pemantauan kinerja, penanganan bug, dan peningkatan fungsionalitas. Tak kalah penting, optimasi SEO menjadi pertimbangan dalam pengembangan website untuk meningkatkan visibilitas di mesin pencari dan menarik lebih banyak pengguna. Dengan pemahaman mendalam tentang langkah-langkah ini, pengembangan website dapat dilakukan secara efektif dan efisien, mendukung keberhasilan dan pertumbuhan suatu entitas di dunia digital.

BAB 3 ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Bab 3 ini akan membahas secara mendalam langkah-langkah yang ditempuh dalam pemecahan masalah saat membuat implementasi program Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar dengan fokus pada konteks Content-Based Image Retrieval (CBIR) untuk pencarian gambar. Berikut adalah langkah-langkah serta analisis pemecahan masalah dalam pembuatan implementasi program.

- 1. Menerima input data dari pengguna di Website yang tersedia untuk mengubahnya agar dapat diproses dalam pemrograman pencarian gambar yang sudah dibuat.**

Pada langkah pertama ini, program menerima input data dari pengguna melalui web. Pengguna dapat memasukkan gambar dan data set untuk menemukan gambar yang memiliki kemiripan antara gambar yang diinput dengan gambar yang tersedia dalam sebuah data set.. Input ini kemudian diubah agar sesuai dengan format yang dapat diproses dalam pemrograman pencarian. Proses pengubahan ini memastikan bahwa data yang dimasukkan oleh pengguna dapat diintegrasikan dengan langkah-langkah selanjutnya dalam aplikasi.

Proses penerimaan input data dari pengguna dilakukan melalui antarmuka web yang interaktif dan mudah digunakan. Setelah pengguna memasukkan input data, program akan mengubahnya agar sesuai dengan format yang dapat diproses dalam pemrograman pencarian. Proses pengubahan ini dilakukan dengan cara berikut gambar yang diunggah oleh pengguna akan dikonversi menjadi format digital yang dapat diproses oleh komputer. Proses pengubahan ini memastikan bahwa data yang dimasukkan oleh pengguna dapat diintegrasikan dengan langkah-langkah selanjutnya dalam aplikasi.

Proses penerimaan input data dari pengguna ini dirancang untuk memberikan pengalaman pengguna yang menyenangkan dan interaktif. Antarmuka web yang digunakan dibuat dengan tampilan yang menarik dan fitur-fitur yang mudah digunakan. Selain itu, proses pengubahan data dilakukan dengan cara yang diusahakan efisien dan menuju akurat, sehingga hasil pencarian dapat diperoleh dengan cepat dan akurat.

Misalnya, seorang pengguna ingin menemukan gambar harimau sumatera. Pengguna dapat mengunggah gambar harimau sumatera dari perangkat mereka, dan memasukkan data set yang ingin dicari. Setelah itu, program akan mengubah input data tersebut menjadi format yang dapat diproses dalam pemrograman pencarian.

Setelah data diubah, program akan membandingkan gambar yang diunggah oleh pengguna dengan setiap gambar dalam basis data menggunakan

metrik cosine similarity. Metriks ini mengukur kemiripan antara dua vektor, yang dalam hal ini adalah vektor representasi gambar. Gambar yang memiliki kemiripan tertinggi dengan gambar yang diunggah oleh pengguna akan ditampilkan kepada pengguna sebagai hasil pencarian.

Dalam CBIR warna, tantangan utama terletak pada variasi warna yang dapat muncul dalam gambar, dipengaruhi oleh faktor-faktor seperti pencahayaan, kondisi pengambilan gambar, dan perbedaan kamera yang digunakan.

2. Membandingkan satu buah gambar dengan gambar lainnya menggunakan konsep *Content-Based Information Retrieval* (CBIR)

Dalam dunia pemrosesan gambar, tantangan mendasar adalah bagaimana secara efektif membandingkan dua buah gambar, terutama saat mencari kesamaan visual. Salah satu solusi yang muncul adalah Content-Based Image Retrieval (CBIR), suatu konsep yang memungkinkan identifikasi dan pencarian gambar berdasarkan konten visual. CBIR memanfaatkan ekstraksi fitur dari gambar untuk membangun representasi numerik yang dapat dibandingkan secara matematis. Untuk mengatasi kerumitan ini, CBIR warna memanfaatkan ekstraksi fitur yang berfokus pada distribusi warna. Setiap gambar diwakili dalam suatu ruang warna, seperti RGB, yang merupakan model matematis untuk menggambarkan relasi antar berbagai warna. CBIR memiliki dua parameter utama: warna dan tekstur.

1. Pemecahan Masalah dalam CBIR Warna

Permasalahan utama dalam perbandingan warna adalah variasi warna yang mungkin terjadi dalam gambar. CBIR menggunakan metode ekstraksi fitur warna untuk menghasilkan representasi numerik dari distribusi warna gambar. Pemilihan metode ekstraksi yang efektif akan memastikan bahwa perbedaan warna diakui dan diukur dengan tepat.

Pertama-tama, dalam CBIR warna, langkah awalnya adalah mengubah array dari warna RGB yang dihasilkan gambar. Proses ini melibatkan pemisahan array RGB menjadi blok-blok 3x3. Menggunakan blok-blok ini membantu percepatan perhitungan karena fokus pada area kecil dalam gambar, meminimalkan kompleksitas perbandingan.

Setelah pemisahan blok dilakukan, langkah selanjutnya adalah mengonversi setiap blok ke dalam ruang warna HSV (Hue, Saturation, Value). Transformasi ke ruang warna HSV diperlukan karena HSV lebih intuitif dalam merepresentasikan informasi warna. Hue mencerminkan jenis warna, Saturation menunjukkan intensitas warna, dan Value mencerminkan kecerahan.

Selanjutnya, dilakukan pembuatan histogram dari setiap blok dalam gambar. Histogram merepresentasikan distribusi frekuensi warna dalam blok tersebut. Pembuatan histogram diperlukan untuk menyederhanakan informasi warna dan memberikan representasi yang lebih mudah dibandingkan. Dengan

menggambarkan frekuensi kemunculan setiap nilai warna, histogram menjadi dasar untuk perbandingan distribusi warna antar blok. Pentingnya penggunaan histogram terletak pada kemampuannya untuk merepresentasikan distribusi warna secara ringkas dan memberikan pandangan yang komprehensif tentang karakteristik warna dalam suatu blok. Ini memungkinkan kita untuk mengenali pola warna yang relevan tanpa harus memeriksa setiap piksel secara individual. Setelah memiliki array histogram untuk setiap blok pada suatu gambar, langkah selanjutnya adalah membandingkan gambar dengan menggunakan metrik cosine similarity. Cosine similarity digunakan untuk mengukur sejauh mana dua vektor (dalam hal ini, vektor representasi histogram dua blok) memiliki arah yang sama. Semakin tinggi nilai cosine similarity, semakin mirip distribusi warna kedua blok tersebut.

Melalui langkah-langkah ini, CBIR warna menciptakan representasi numerik yang efektif dari distribusi warna gambar, memungkinkan perbandingan yang akurat dan efisien untuk pencarian gambar yang serupa berdasarkan kesamaan warna dalam blok-bloknya.

2. Pemecahan Masalah dalam CBIR Tekstur

Tantangan dalam perbandingan tekstur adalah diversitas karakteristik tekstur yang dapat muncul dalam gambar. Langkah pertama dalam pemecahan masalah tekstur pada CBIR adalah mengubah array dari warna RGB menjadi citra grayscale. Proses ini penting karena tekstur dapat lebih baik diidentifikasi dalam citra grayscale, yang fokus pada tingkat kecerahan atau intensitas piksel tanpa mempertimbangkan warna.

Setelah itu, pembuatan framework matrix dilakukan dengan menggunakan 256 level kuantisasi. Kuantisasi dilakukan untuk mengurangi kompleksitas dan membuat representasi lebih efisien, mempertahankan informasi tekstur utama dengan membagi tingkat intensitas piksel menjadi level-level yang lebih sedikit. Framework matrix ini memberikan landasan untuk proses selanjutnya. Framework matrix diisi dengan mengukur frekuensi kemunculan nilai piksel pada citra grayscale. Hal ini menciptakan representasi numerik yang lebih mudah dicerna, memungkinkan identifikasi pola dan struktur tekstur.

Proses selanjutnya adalah membuat Gray-Level Co-occurrence Matrix (GLCM) dari framework matrix. GLCM merepresentasikan seberapa sering sepasang piksel dengan tingkat intensitas tertentu muncul bersamaan dalam suatu arah. Transpose dari GLCM juga diambil untuk memperhitungkan kemunculan dalam arah yang berlawanan.

Setelah GLCM didapatkan, langkah berikutnya adalah normalisasi. Normalisasi dilakukan untuk menyamakan skala dan membuat matriks lebih

mudah dibandingkan. Normalisasi GLCM menghasilkan Co-occurrence Matrix yang dapat digunakan sebagai dasar untuk mengekstrak fitur tekstur.

Dengan Co-occurrence Matrix, dapat dihitung nilai *contrast*, *homogeneity*, dan *entropy*. *Contrast* mencerminkan variasi intensitas piksel, *homogeneity* mencerminkan seberapa seragam tekstur, dan *entropy* mencerminkan tingkat keacakan tekstur. Ketiga nilai ini membentuk vektor yang merepresentasikan fitur tekstur dari gambar.

Terakhir, vektor fitur tekstur dari gambar yang akan dicari kemiripannya dapat dibandingkan menggunakan metrik cosine similarity. Semakin tinggi nilai cosine similarity, semakin mirip tekstur kedua gambar. Dengan merinci proses ekstraksi fitur dan perbandingan tekstur menggunakan CBIR, kami jadi memahami bagaimana konsep ini mengatasi kompleksitas dalam mencari gambar berdasarkan karakteristik tekstur. Pemahaman mendalam ini memberikan dasar untuk pencarian gambar yang lebih kontekstual dan mendalam berdasarkan tekstur visual.

3. Mengumpulkan Data Hasil Perbandingan dengan Cosine Similarity dan Integrasi dengan Web

Cosine similarity digunakan dalam pemrograman untuk membandingkan kemiripan dua buah gambar karena memberikan ukuran sejauh mana dua vektor (dalam hal ini, vektor representasi fitur gambar) memiliki arah yang sama. Dalam konteks CBIR, vektor ini mewakili distribusi fitur gambar, dan cosine similarity memberikan nilai antara 0 (tidak ada kesamaan) hingga 1 (kesamaan sempurna). Keuntungan dari cosine similarity adalah ketahanannya terhadap perubahan skala, sehingga perbedaan dalam magnitudo fitur tidak mempengaruhi hasil. Hal ini penting karena distribusi fitur dapat bervariasi tanpa mengubah signifikansi kesamaan. Cosine similarity juga efektif dalam mencerminkan hubungan antar fitur dalam gambar. Jika dua gambar memiliki distribusi fitur yang serupa, nilai cosine similarity akan tinggi, menunjukkan kesamaan visual yang relevan.

Setelah perhitungan cosine similarity dilakukan untuk setiap pasangan gambar dalam basis data, hasilnya dapat disimpan dalam bentuk matriks similaritas. Matriks ini memiliki ukuran NxN, di mana N adalah jumlah gambar dalam basis data. Setiap elemen matriks merepresentasikan nilai cosine similarity antara gambar i dan gambar j. Dalam praktiknya, hasil cosine similarity dapat disimpan dalam struktur data yang efisien, seperti tabel atau array. Hal ini memudahkan pengelolaan dan pengambilan data saat diintegrasikan dengan web.

Data hasil perbandingan dapat diolah agar sesuai dengan format yang dapat digunakan oleh aplikasi web. Ini mungkin melibatkan penyortiran hasil berdasarkan nilai cosine similarity tertinggi atau pembuatan indeks untuk mempercepat akses.

Hasil perbandingan kemiripan dapat diakses melalui API (Application Programming Interface) atau disimpan dalam database yang dapat diakses oleh aplikasi web. Ini memungkinkan aplikasi web untuk dengan mudah mengambil dan menampilkan hasil pencarian kepada pengguna.

Web dapat mengintegrasikan hasil cosine similarity ke dalam antarmuka pengguna untuk memberikan pengalaman pencarian gambar yang interaktif. Pengguna dapat melihat gambar yang paling mirip, mendapatkan informasi tambahan, atau menyortir hasil berdasarkan preferensi mereka. Melalui langkah-langkah ini, hasil perbandingan dengan cosine similarity tidak hanya dihitung, tetapi juga disimpan dan disiapkan agar dapat diintegrasikan dengan lancar ke dalam aplikasi web, meningkatkan responsivitas dan keterlibatan pengguna.

3.2 Proses Pemetaan Masalah

Proses pemetaan masalah dalam program ini melibatkan konsep matriks, vektor, dekomposisi, dan normalisasi untuk mengatasi tantangan dalam pemrosesan dan perbandingan gambar. Dalam konteks pengolahan gambar, matriks sangat relevan karena gambar dapat diinterpretasikan sebagai matriks piksel dengan baris dan kolom yang merepresentasikan dimensi gambar. Penerapan matriks dalam program ini terkait erat dengan pelajaran aljabar linier dan geometri yang diajarkan di kelas,, khususnya dalam transformasi dan representasi data.

Konsep aljabar linier, seperti matriks, berperan penting dalam pemrosesan gambar. Misalnya, dalam ekstraksi fitur warna, array RGB gambar diubah menjadi matriks, yang memfasilitasi manipulasi dan analisis lebih lanjut. Pelajaran aljabar linier geometri memberikan dasar untuk memahami bagaimana matriks dapat merepresentasikan transformasi dalam ruang warna. Sementara itu, penggunaan vektor dalam pemecahan masalah tekstur membuka pintu untuk penerapan konsep aljabar geometri. Vektor fitur tekstur, seperti hasil dari ekstraksi GLCM, memungkinkan perbandingan dan pencarian gambar berdasarkan karakteristik tekstur yang direpresentasikan sebagai vektor. Pelajaran tentang vektor dalam aljabar geometri membantu memahami bagaimana vektor dapat digunakan sebagai alat representasi dalam analisis tekstur gambar.

Dekomposisi dan normalisasi adalah langkah-langkah kritis dalam memastikan kualitas dan keseragaman data. Penerapan dekomposisi matriks, seperti dalam proses ekstraksi fitur warna, memungkinkan pemisahan komponen warna untuk analisis yang lebih mendalam. Normalisasi, di sisi lain, membantu menjaga konsistensi dan membawa data ke skala yang dapat dibandingkan, memastikan bahwa hasil perbandingan dapat diandalkan.

Secara keseluruhan, konsep-konsep aljabar linier dan geometri, terutama matriks dan vektor, memberikan dasar matematis yang kuat untuk merumuskan dan memecahkan

masalah dalam pemrosesan gambar. Penerapan dekomposisi dan normalisasi sebagai langkah-langkah *pre-processing* menunjukkan pemahaman yang mendalam tentang bagaimana data gambar dapat dioptimalkan untuk analisis yang efektif. Dengan menggunakan landasan ini, program dapat menghadirkan solusi yang kokoh dan efisien dalam pencarian dan perbandingan gambar berbasis konten.

3.3 Contoh Ilustrasi Kasus

Pengguna memiliki kebebasan untuk menggali opsi yang kami tawarkan dan membuat pilihan yang sesuai dengan kebutuhan mereka. Web kami didesain agar mudah dinavigasi, memberikan pengalaman yang jelas dan nyaman.

Sebagai contoh, misalkan pengguna memilih untuk mencari gambar dengan menggunakan data set. Dalam proses ini, pengguna diminta untuk mengunggah file data set dan gambar yang ingin dicari. Namun, dalam petualangan pencarian ini, terdapat juga mekanisme pengaman yang inovatif yang kami tanamkan. Jika, misalnya, file yang diunggah pengguna tidak sesuai dengan format yang diperlukan, program kami dilengkapi dengan sistem penanganan masalah. Pengguna akan diberitahu secara jelas dan diberikan arahan untuk mengatasi kendala tersebut, menciptakan pengalaman yang ramah dan informatif. Kemudian kami juga mendapatkan kasus lain dimana pengguna ingin memasukan dataset dari gambar yang terdapat di beberapa website, sehingga kami menambahkan fitur *image-scraping* juga.

Tidak hanya itu, program kami memberikan kebebasan bagi pengguna untuk memilih berbagai metode pencarian, seperti menggunakan kamera atau melakukan web scraping. Meskipun jalannya berbeda, konsep pencarian tetap terjaga, dan program kami dapat menyesuaikan diri untuk memproses input dari metode pencarian yang dipilih pengguna. Setelah memilih dan mengunggah data set, program kami memulai perjalanan pencarian gambar. Gambar input diproses dengan cermat, dipecah menjadi blok-blok kecil untuk mendapatkan gambaran warna yang lebih rinci. Hasilnya, pengguna dapat dengan mudah melihat hasil pencarian yang ditampilkan di web terintegrasi, memberikan gambar-gambar yang memiliki kemiripan signifikan dengan gambar input. Selain itu, kami juga menambahkan fitur *caching* dimana dataset yang pernah diolah akan disimpan datanya. Sehingga proses pencarian akan lebih cepat jika pengguna ingin melakukan pencarian berulang dengan dataset yang sama.

Dengan menyajikan pengalaman yang lebih terbuka, menangani masalah input secara cerdas, dan memberikan kebebasan dalam metode pencarian, program atau web kami tidak hanya memberikan solusi efisien bagi pengguna yang mencari gambar, tetapi juga menciptakan petualangan yang menarik dan informatif.

BAB 4 IMPLEMENTASI DAN UJI COBA

4.1 Implementasi Program Utama

4.1.1 CBIR dengan Parameter Warna

1. Fungi Konversi dari RGB menjadi HSV

```
function RGBtoHSV (rgb : Matrix
[0..imagerowsizeafterdecompose-1][0..imagecolumnsizelafterdecompose-1] of array [0..2] of integer) -> Matrix
[0..imagerowsizelafterdecompose-1][0..imagecolumnsizelafterdecompose-1] of array [0..2] of float
{Mengonversi matriks RGB menjadi matriks HSV}
```

KAMUS LOKAL

```
Cmax : float
Cmax : float
d : float
r : float
g : float
b : float
hsv : Matrix
[0..imagerowsizelafterdecompose-1][0..imagecolumnsizelafterdecompose-1] of array [0..2] of float
{Fungsi max dari library Numpy}
function max (r: float, g: float, b: float)-> float
{Fungsi min dari library Numpy}
function min (r: float, g: float, b: float)-> float
```

ALGORITMA

```
i traversal [0..imagerowsizelafterdecompose-1]
j traversal [0..imagecolumnsizelafterdecompose-1]
r <- rgb[i][j][0]/255
g <- rgb[i][j][1]/255
b <- rgb[i][j][2]/255

Cmax <- max(r,g,b)
Cmin <- min(r,g,b)
d <- Cmax - Cmin

{ Find H }
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)
elif (Cmax = b) then
        h <- 60 * (((r-g)/d)+4)

{ Find S }
if (Cmax = 0) then
    s <- 0
else
    s <- d/Cmax

{ Find V }
v <- Cmax

hsv[i][j] <- [h,s,v]
-> hsv

```

2. Fungsi Konversi dari HSV menjadi Histogram (bingung)

```

function hsv_to_histogram (hsv : Matrix
[0..imagerowsizeafterdecompose-1][0..imagecolumnsizelafterdecompose
-1] of array [0..2] of float) -> array of [0..13] of integer
{Mengonversi matriks HSV menjadi array of histogram}

```

KAMUS LOKAL

```

histH : integer
histS : integer
histV : integer
hist : array of [0..13] of integer
hsv : Matrix
[0..imagerowsizeafterdecompose-1][0..imagecolumnsizelafterdecompose
-1] of array [0..2] of float
function histogramH (h: float, bins: array of [0..1] of
float) -> integer
{Fungsi histogramS dari library Numpy}
function histogramS (s: float, bins: array of [0..1] of
float) -> integer
{Fungsi histogramV dari library Numpy}
function histogramV (v: float, bins: array of [0..1] of
float) -> integer

```

ALGORITMA

```

i traversal [0..imagerowsizeafterdecompose-1]
j traversal [0..imagecolumnsizelafterdecompose-1]

```

```

[h,s,v] <- hsv[i][j]
hist[0] <- histogramH(h,[315,360])
hist[1] <- histogramH(h,[0,25])
hist[2] <- histogramH(h,[25,40])
hist[3] <- histogramH(h,[40,120])
hist[4] <- histogramH(h,[120,190])
hist[5] <- histogramH(h,[190,270])
hist[6] <- histogramH(h,[270,295])
hist[7] <- histogramH(h,[295,315])
hist[8] <- histogramS(s,[0,0.2])
hist[9] <- histogramS(s,[0.2,0.7])
hist[10] <- histogramS(s,[0.7,0.1])
hist[11] <- histogramV(s,[0,0.2])
hist[12] <- histogramV(s,[0.2,0.7])
hist[13] <- histogramV(s,[0.7,0.1])
->hist

```

3. Fungsi Menghitung Cosine Similarity

```

function cosine_similarity (a : array of [0..13] of integer, b :  

array of [0..13] of integer) -> float  

{Menghitung cosine similarity dari histogram per block nya}

```

KAMUS LOKAL

```

sum_a : integer  

sum_b : integer  

dot : integer  

cosine: float

```

ALGORITMA

```

sum_a <- 0  

sum_b <- 0  

dot <- 0  

i traversal [0..13]  

    sum_a <- sum_a + a[i]  

    sum_b <- sum_b + b[i]  

    dot <- dot + a[i]*b[i]  

cosine <- dot/(sum_a*sum_b)
->cosine

```

4. Fungsi untuk membagi gambar menjadi 3x3 blok

```

function decompose_block (img : Matrix  

[0..imagerowsize-1][0..imagecolumnszie-1] of array [0..2] of  

integer ) -> Matrix  

[0..(newarrayrowsize-1)/9][0..(newarraycolumnszie-1)/9] of array  

[0..2] of integer

```

{Membagi array rgb menjadi 3x3 blok}

KAMUS LOKAL

```
new_array : Matrix  
[0..(imagerowsize-1)/3][0..(imagecolumnszie-1)/3] of array [0..2]  
of integer  
new2_array : Matrix  
[0..(newarrayrowsize-1)/3][0..(newarraycolumnszie-1)/3] of array  
[0..2] of integer  
{Fungsi array_split dari library Numpy}  
function array_split (rgb: Matrix  
[0..imagerowsize-1][0..imagecolumnszie-1] of array [0..2] of  
integer, b: integer) -> Matrix  
[0..(imagerowsize-1)/b][0..(imagecolumnszie-1)/b] of array [0..2]  
of integer
```

ALGORITMA

```
new_array <- array_split(img,3)  
new2_array <- array_split(new_array,3)  
-> new2_array
```

5. Fungsi untuk membandingkan antar blok

```
function compare_blocks (histogram1:array of [0..13] of integer  
, histogram2: array of [0..13] of integer) -> float  
{Menghasilkan nilai similarity dalam bentuk persen dari dua  
gambar dengan penempatan blok yang sama}
```

KAMUS LOKAL

```
sim : float  
similarity: float  
i : integer
```

ALGORITMA

```
sim <- 0  
i traversal [0..8]  
    sim <- sim+cosine_similarity(histogram1[i],histogram2[i])  
similarity <- (sim/9)*100  
->similarity
```

6. Fungsi menghitung color similarity

```

function colorSimilarity (img :Matrix
[0..imagerowsize-1][0..imagecolumnsize-1] of array [0..2] of
integer, isUpload : boolean, filename: string) -> array of [0..8]
of array [0..13] of integer
{Menghasilkan array of histogram dari 9 blok dari 1 gambar}

```

KAMUS LOKAL

```

block : Matrix
[0..(newarrayrowsize-1)/9][0..(newarraycolumnszie-1)/9] of array
[0..2] of integer
    i : integer
    j : integer
    hist : array of [0..8] of array [0..13] of integer
    function decompose_block (img : Matrix
[0..imagerowsize-1][0..imagecolumnsize-1] of array [0..2] of
integer ) -> Matrix
[0..(newarrayrowsize-1)/9][0..(newarraycolumnszie-1)/9] of array
[0..2] of integer
    function RGBtoHSV (rgb : Matrix
[0..imagerowsizeafterdecompose-1][0..imagecolumnszieafterdecompose-1] of array [0..2] of integer) -> Matrix
[0..imagerowsizeafterdecompose-1][0..imagecolumnszieafterdecompose-1] of array [0..2] of float

```

ALGORITMA

```

block <- decompose_block(img)
i traversal [0..2]
    j traversal [0..2]
        block[u][j] <- RGBtoHSV(block[u][j])
        hist[u] <- block[u][j]
    if (isUpload = true) then
        open('cache.txt', f)
        rewrite(f)
        write(f, filename+"|"+hist+"\n")
        close('cache.txt')
    else
        ->hist

```

7. Fungsi untuk membandingkan warna

```

function compareWarna (filename : string) -> float
{Menghasilkan array of histogram dari 9 blok dari 1 gambar}

```

KAMUS LOKAL

```

start : float
end : float

```

```

function time() -> float
function colorSimiliarity (img :Matrix
[0..imagerowsize-1][0..imagecolumnszie-1] of array [0..2] of
integer, isUpload : boolean, filename: string) -> array of [0..8]
of array [0..13] of integer
procedure readlines(input : string)
{Fungsi untuk membulatkan ke bawah dari python}
function round (sim : float)-> integer
function parseTXT (line : string)-> integer
function Sort (line : string)-> integer

```

ALGORITMA

```

start <- time()
open("static/" + filename, img)
img <- colorSimiliarity(img, false, filename)
open('cache.txt', f)
rewrite(f)
line <- readlines(f)
i traversal [0..length(line)-1]
    temp <- parseTXT(line[i])
    sim <- compare_blocks(img, temp[1])
    if (sim ≥ 60) then
        ret[temp(0)] <- round(sim,2)
end <- time()
ret <- Sort(ret)
ret["Time"] <- round(end-start,2)
open('hasil.json', f)
rewrite(f)
write(f, json.dumps(ret))
close('hasil.json')
->ret

```

4.1.2 CBIR dengan Parameter Tekstur

1. Fungsi untuk mengonversi gambar rgb menjadi citra grayscale

```

function convertToGrayscale (rgb : Matrix
[0..imagerowsize-1][0..imagecolumnszie-1] of array [0..2] of
integer) -> Matrix [0..imagerowsize-1][0..imagecolumnszie-1] of
array [0..255] of float
{Mengonversi array rgb menjadi citra grayscale}

```

KAMUS LOKAL

```

new : Matrix [0..imagerowsize-1][0..imagecolumnszie-1] of
array [0..255] of float
i : integer
j : integer

```

ALGORITMA

```

    i traversal [0..imagerowsize-1]
        j traversal [0..imagecolumnsize-1]
            new <- rgb[i][j][0]*0.29 + rgb[i][j][1]*0.587 +
            rgb[i][j][2]*0.114
            ->new

```

2. Fungsi untuk membuat co-occurrence matrix

```

function create_co_occurrence_matrix (grayscaleMatrix : Matrix
[0..imagerowsize-1][0..imagecolumnsize-1] of array [0..255] of
integer, quantization_level : integer) -> Matrix [0..255][0..255]
of array [0..255] of integer
{Membuat glcm matrix}

```

KAMUS LOKAL

```

    i : integer
    j : integer
    framework_matrix : Matrix [0..255][0..255] of array [0..255]
of integer
    glcm_matrix : Matrix [0..255][0..255] of array [0..255] of
integer
        function add (matrix1 : Matrix [0..255][0..255] of array
[0..255] of integer, matrix2 : Matrix [0..255][0..255] of array
[0..255] of integer)->Matrix [0..255][0..255] of array [0..255]
of integer
        function transpose (matrix : Matrix [0..255][0..255] of
array [0..255] of integer)->Matrix [0..255][0..255] of array
[0..255] of integer

```

ALGORITMA

```

    i traversal [0..imagerowsize-1]
        j traversal [0..imagecolumnsize-1]
            framework_matrix[i][j]<- framework_matrix[i][j]+1
            glcm_matrix <- add(glcm_matrix, transpose(glcm_matrix))
    return glcm_matrix

```

3. Fungsi untuk mendapatkan nilai contrast dari sebuah gambar

```

function calculate_contrast_from_glcm (grayscaleMatrix : Matrix
[0..255][0..255] of array [0..255] of integer) -> float
{Mendapatkan nilai contrast}

```

KAMUS LOKAL

```

    i : integer
    j : integer

```

```

contrast : float

ALGORITMA
    i traversal [0..255]
        j traversal [0..255]
            contrast <- grayscaleMatrix[i][j]*((i-j)(i-j))
    -> contrast

```

4. Fungsi untuk mendapatkan nilai entropy dari sebuah gambar

```

function calculate_entropy_from_glcM (grayscaleMatrix : Matrix
[0..255][0..255] of array [0..255] of integer) -> float
{Mendapatkan nilai contrast}

KAMUS LOKAL
    i      : integer
    j      : integer
    entropy : float

ALGORITMA
    i traversal [0..255]
        j traversal [0..255]
            entropy <- grayscaleMatrix[i][j]/1+((i-j)(i-j))
    -> entropy

```

5. Fungsi untuk mendapatkan nilai homogeneity dari sebuah gambar

```

function calculate_homogeneity_from_glcM (grayscaleMatrix :
Matrix [0..255][0..255] of array [0..255] of integer) -> float
{Mendapatkan nilai contrast}

KAMUS LOKAL
    i      : integer
    j      : integer
    homogeneity : float
    function log (elmtMtx : float)->float

ALGORITMA
    i traversal [0..255]
        j traversal [0..255]
            homogeneity<-
            -(grayscaleMatrix[i][j]*log(grayscaleMatrix[i][j]))
    -> homogeneity

```

6. Fungsi untuk membuat vektor tekstur dari matriks yang telah dibuat

```

function create_texture_vector (glcmMatrix : Matrix
[0..255][0..255] of array [0..255] of integer) -> array [0..2]
float
{Membuat vektor tektur dengan nilai contrast,homogeneity, dan
entropy}

```

KAMUS LOKAL

```
texture_vector : array [0..2] of float
```

ALGORITMA

```

    texture_vector[0] <- calculate_contrast_from_glcm(glcmMatrix)
    texture_vector[1] <-
calculate_homogeneity_from_glcm(glcmMatrix)
    texture_vector[2] <- calculate_entropy_from_glcm(glcmMatrix)
-> texture_vector

```

7. Fungsi untuk mengubah gambar menjadi vektor

```

function pictureToTextureVector (img :Matrix
[0..imagerowsize-1][0..imagecolumnszie-1] of array [0..2] of
integer, isUpload : boolean, filename: string) -> array of [0..8]
of array [0..2] of float
{Mengubah gambar menjadi vektor}

```

KAMUS LOKAL

```

sum_a : integer
sum_b : integer
dot : integer
cosine: float

```

ALGORITMA

```

grayscale_matrix <- convertToGrayscale(img)
glcm_matrix <- create_co_occurrence_matrix(grayscale_matrix)
i traversal [0..255]
    j traversal [0..255]
        norm_glcm_matrix[i][j]<-
glcm_matrix[i][j]/sum(glcm_matrix)
    texture_vector <- create_texture_vector(norm_glcm_matrix)
    if (isUpload = true) then
        open ('cache.txt',f)
        rewrite(f)
        write(f, filename + texture_vector[0] + "," +
texture_vector[1] + "," + texture_vector[2] +"\n")
        close('cache.txt')
->texture_vector

```

8. Fungsi untuk mendapatkan cosine similarity

```

function cosine_similarity (a : array of [0..2] of float, b : array of [0..2] of float) -> float
{Menghitung cosine similarity dari vektor tekstur sebuah gambar dalam bentuk persen}

KAMUS LOKAL
  sum_a : integer
  sum_b : integer
  dot : integer
  cosine: float

ALGORITMA
  sum_a <- 0
  sum_b <- 0
  dot <- 0
  i traversal [0..2]
    sum_a <- sum_a + a[i]
    sum_b <- sum_b + b[i]
    dot <- dot + a[i]*b[i]
  cosine <- (dot/(sum_a*sum_b))*100
->cosine

```

9. Fungsi untuk membandingkan tekstur antara dua gambar

```

function compareTekstur (filename : string) -> float
{Menghasilkan array of similarity dari 2 gambar}

KAMUS LOKAL
  start : float
  end : float
  count : integer
  function time() -> float
  function colorSimiliarity (img :Matrix
[0..imagerowsiz-1][0..imagecolumnsiz-1] of array [0..2] of integer, isUpload : boolean, filename: string) -> array of [0..8] of array [0..13] of integer
  procedure readlines(input : string)
  {Fungsi untuk membulatkan ke bawah dari python}
  function round (sim : float)-> integer
  function parseTXT (line : string)-> integer
  function Sort (line : string)-> integer

ALGORITMA
  start <- time()
  open("static/" +filename, img)
  img <- colorSimiliarity(img, false, filename)

```

```

open('cache.txt', f)
rewrite(f)
line <- readlines(f)
count <- 0
i traversal [0..length(line)-1]
    temp <- parseTXT(line[i])
    sim <- compare_blocks(img, temp[1])
    output(sim, temp[0])
    if (sim ≥ 60) then
        count <- count+1
        ret[temp(0)] <- round(sim,2)
    ret <- Sort(ret)
end <- time()
ret["Time"]<- round(end-start,2)
open('hasil.json', f)
rewrite(f)
write(f, json.dumps(ret))
close('hasil.json')
->ret

```

4.2 Struktur Program Berdasarkan Spesifikasi

Dalam tugas besar kali ini, kami menggunakan next js app, dengan folder api merupakan *backend* dari program kami dan folder *routes* berisi *page* dari website kami. Kami membangun website ini dengan tidak terlalu memperhatikan struktur *directory* antara *frontend* dan *backend*. Alasan kami tidak terlalu memperhatikan struktur *directory* antara *frontend* dan *backend* adalah karena kami menggunakan *endpoint* sebagai perantara komunikasi antara *frontend* dan *backend*.

Pendekatan struktur program dengan fokus utama *endpoint* sebagai penghubung antara *frontend* dan *backend* dilakukan dengan cara *client-side (frontend)* menarik/*fetch endpoint* dari *server-side (backend)*. Setelah diproses, *server-side* akan mengirimkan *response* yang berupa hasil atau pun status pemrosesan. Dengan struktur ini, pengaruh dari hierarki *directory* tidak terlalu besar dan struktur ini dapat lebih memudahkan kami dalam mengerjakan tugas ini secara terpisah karena memberi kebebasan dalam bekerja sama tanpa menimbulkan *conflict* / saling tumpuk menumpuk kode.

Dengan demikian, apabila kami ingin mengembangkan program ini lebih lanjut lagi dengan men-*deploy* program ini akan lebih mudah bagi kami untuk bisa menghubungkan *backend* dengan *frontend* nya.

Berikut adalah struktur directory dari program GoMilk yang telah kami buat.

```

. └── Algeo02-22134
    └── src
        └── tubes-algeo-02

```

```
|- app
  |- api
    |- main.py
    |- Bonus.py
    |- CBIRWarna.py
    |- CBIRTekstur.py
    |- pdf.html
    |- requirement.txt
    |- static
      |- dataset
        |- search.jpg
    |- (routes)
      |- Camera
        |- page.tsx
      |- Footer
        |- page.tsx
      |- Navbar
        |- page.tsx
      |- about-tubes
        |- page.tsx
      |- content-based-information-retrieval
        |- page.tsx
      |- image-scrapper
        |- page.tsx
    |- favicon.ico
    |- globals.css
    |- layout.tsx
    |- page.tsx
  |- components
    |- ui
      |- button.tsx
      |- input.tsx
      |- switch.tsx
  |- public
    |- 52008104_0.webp
    |- webBG.jpg
    |- chevron-right.svg
    |- dummy.png
    |- dummy.png
    |- furina.png
    |- image-placeholder.webp
    |- next.svg
    |- vercel.svg
```

```
└── components.json
└── next.config.js
└── package-lock.json
└── package.json
└── postcss.config.js
└── tailwind.config.js
└── tailwind.config.ts
└── tsconfig.json
└── test
    ├── 0-199.zip
    └── 0-49.zip
└── img
    ├── About.png
    ├── Home.png
    ├── Tools.png
    └── logo.png
└── docs
└── README.md
```

4.3 Tata Cara Penggunaan Program

Untuk memanfaatkan program GoMilk yang menggunakan mengimplementasikan Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar dengan metode Content Based Image Retrieval (CBIR) dengan optimal, pengguna harus memastikan node.js, *python dependencies* dan npm (Node Package Manager) telah diinstall dalam komputer. Untuk melakukan *set-up* pada komputer, pengguna dapat mengikuti langkah-langkah yang sudah tersedia pada README (<https://github.com/ValentinoTriadi/Algeo02-22134/blob/main/README.md>) yang ada di repository github kelompok kami. Berikut adalah tatacara penggunaan program yang telah kami buat.

Pengguna terlebih dahulu diminta untuk memasukkan dataset gambar dalam bentuk folder atau link website untuk image scraping yang berisi kumpulan gambar. Dataset gambar ini diperlukan sebelum melakukan proses searching agar program dapat melakukan perbandingan dengan gambar yang akan dicari.

Setelah pengguna memasukkan dataset, langkah berikutnya adalah mengunggah gambar yang ingin dicari dari dataset tersebut. Proses penginputan gambar dapat dilakukan dengan mengunggah file gambar ke dalam sistem. Pengguna kemudian diberikan opsi untuk memilih jenis pencarian, apakah berdasarkan warna atau tekstur.

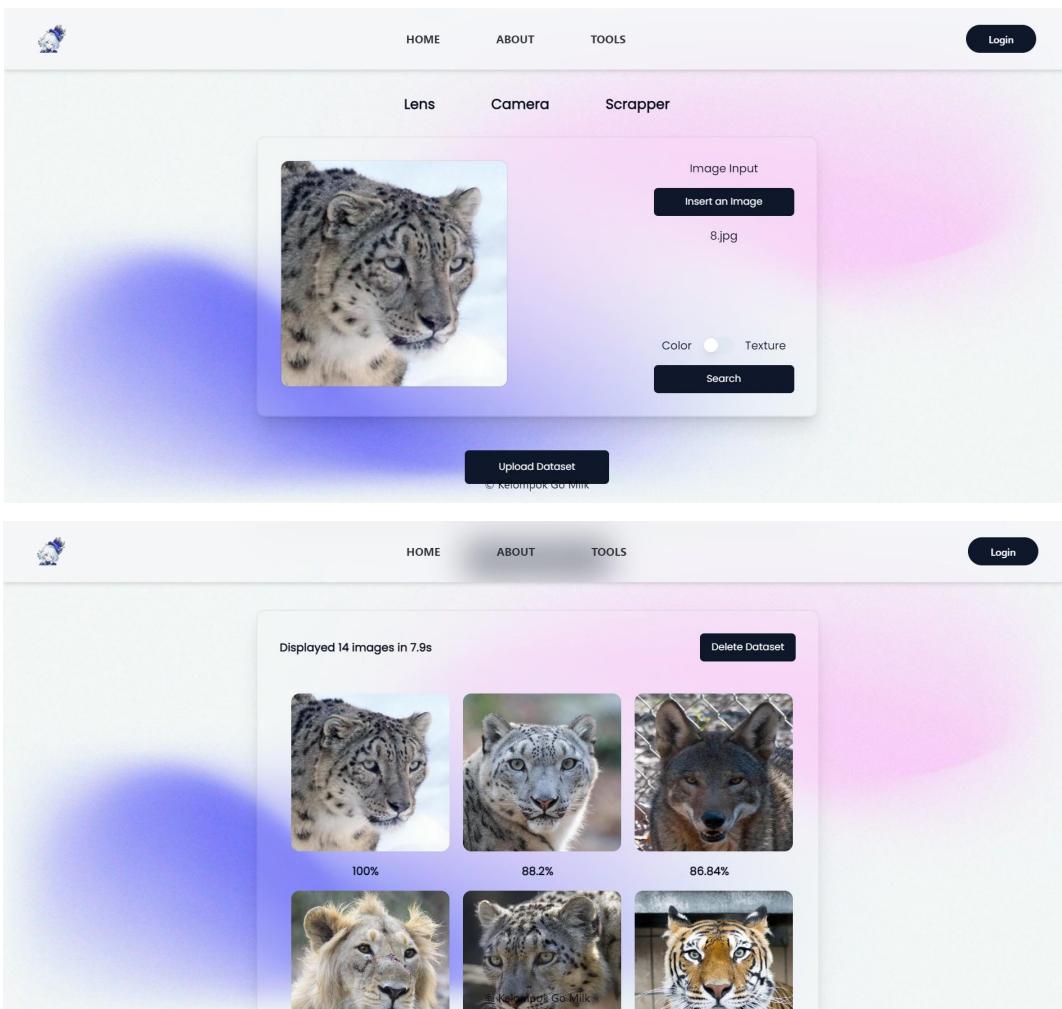
Setelah memilih opsi pencarian, pengguna dapat menekan tombol "search", dan program akan mulai memproses gambar yang telah diunggah. Proses ini mencakup

pencarian gambar-gambar dari dataset yang memiliki kemiripan dengan gambar yang diunggah oleh pengguna.

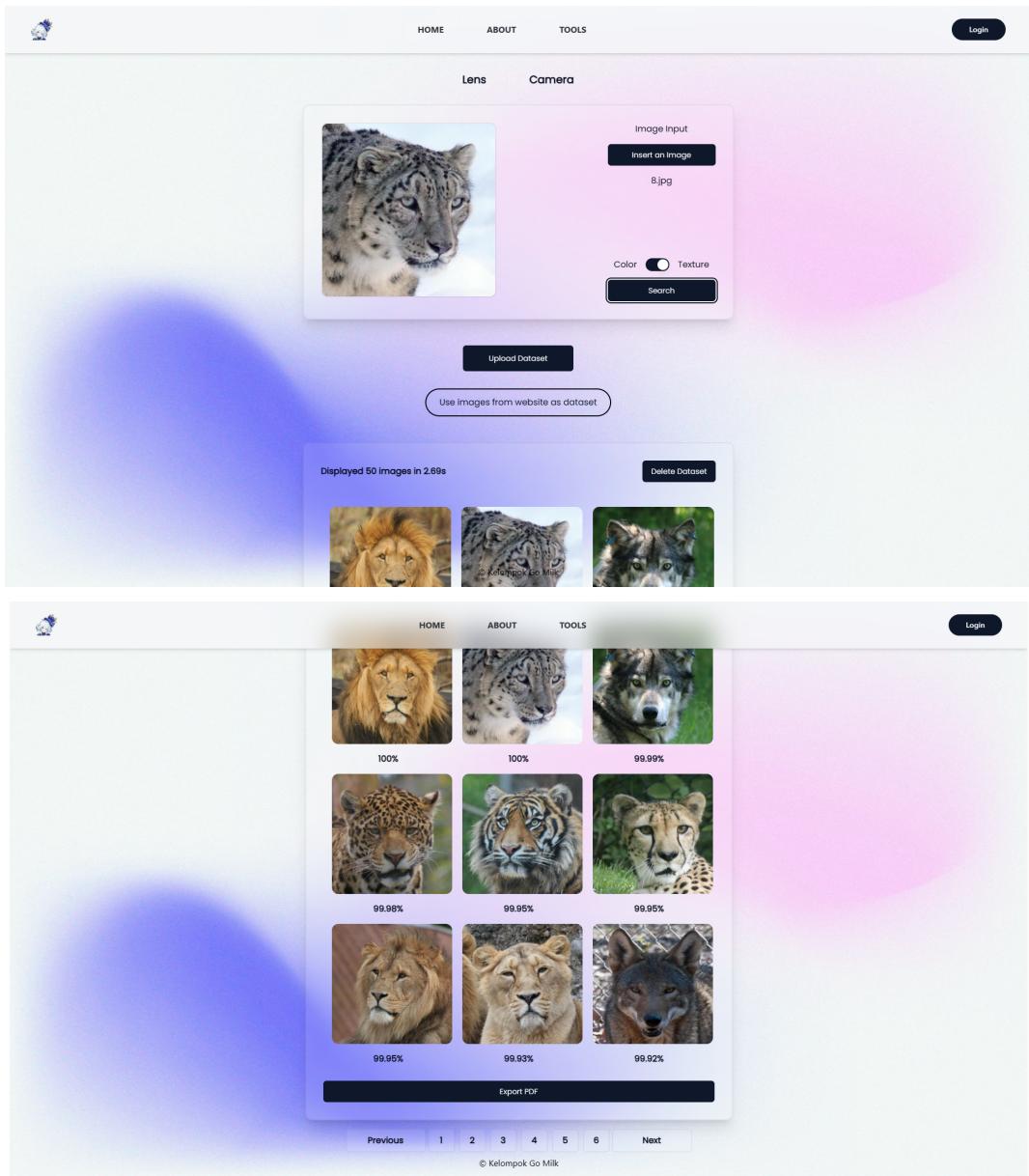
Hasil pencarian akan ditampilkan sebagai kumpulan gambar yang mirip, diurutkan berdasarkan tingkat kemiripan dari yang paling tinggi hingga yang paling rendah. Setiap gambar yang muncul akan disertai dengan persentase kemiripannya terhadap gambar yang diunggah oleh pengguna.

Program juga akan menyajikan informasi terkait, seperti jumlah total gambar yang muncul sebagai hasil pencarian, dan waktu eksekusi programnya. Dengan demikian, pengguna dapat dengan mudah mengevaluasi hasil pencarian dan menemukan gambar-gambar yang paling relevan sesuai dengan preferensi mereka.

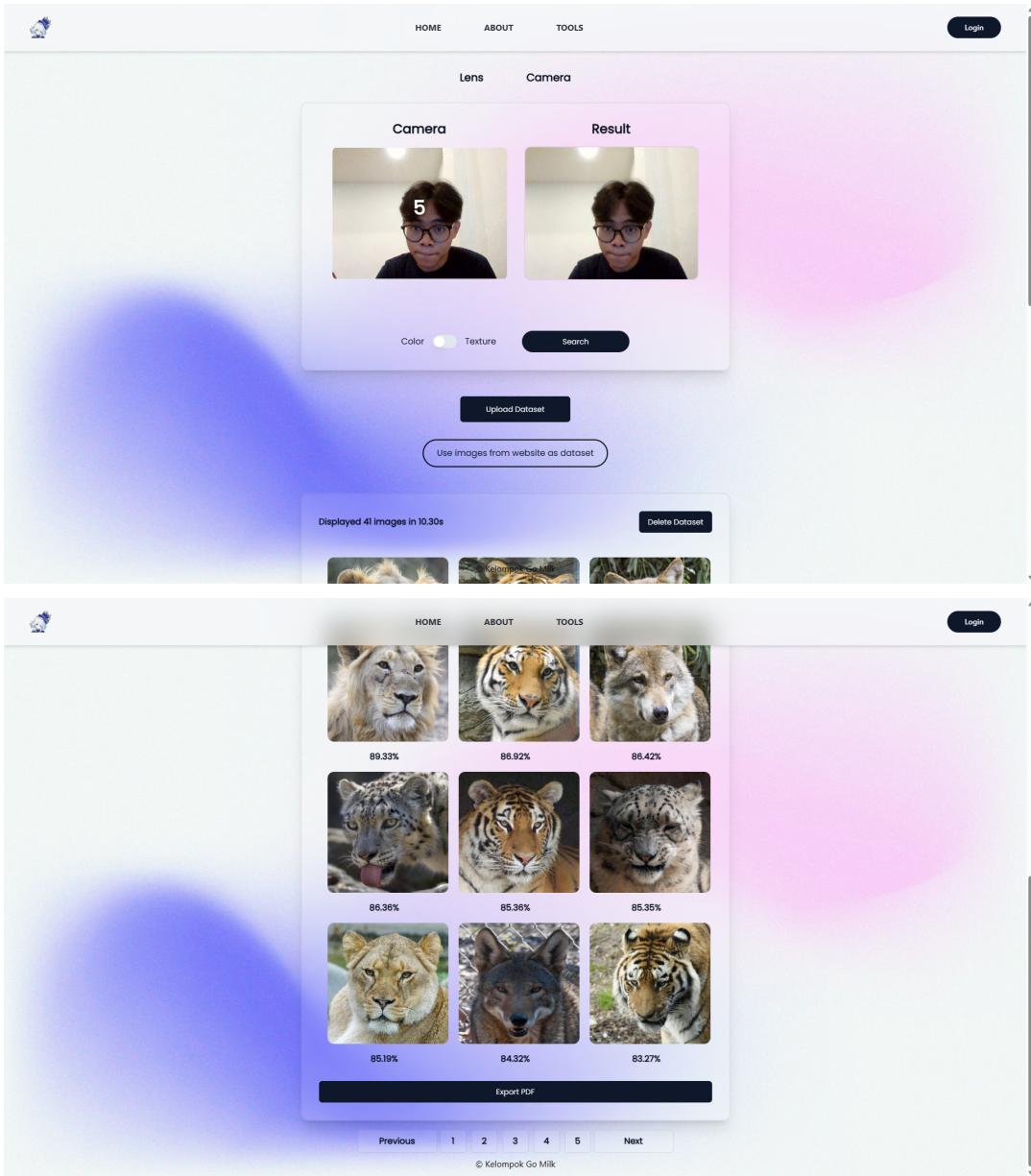
4.4 Hasil Pengujian



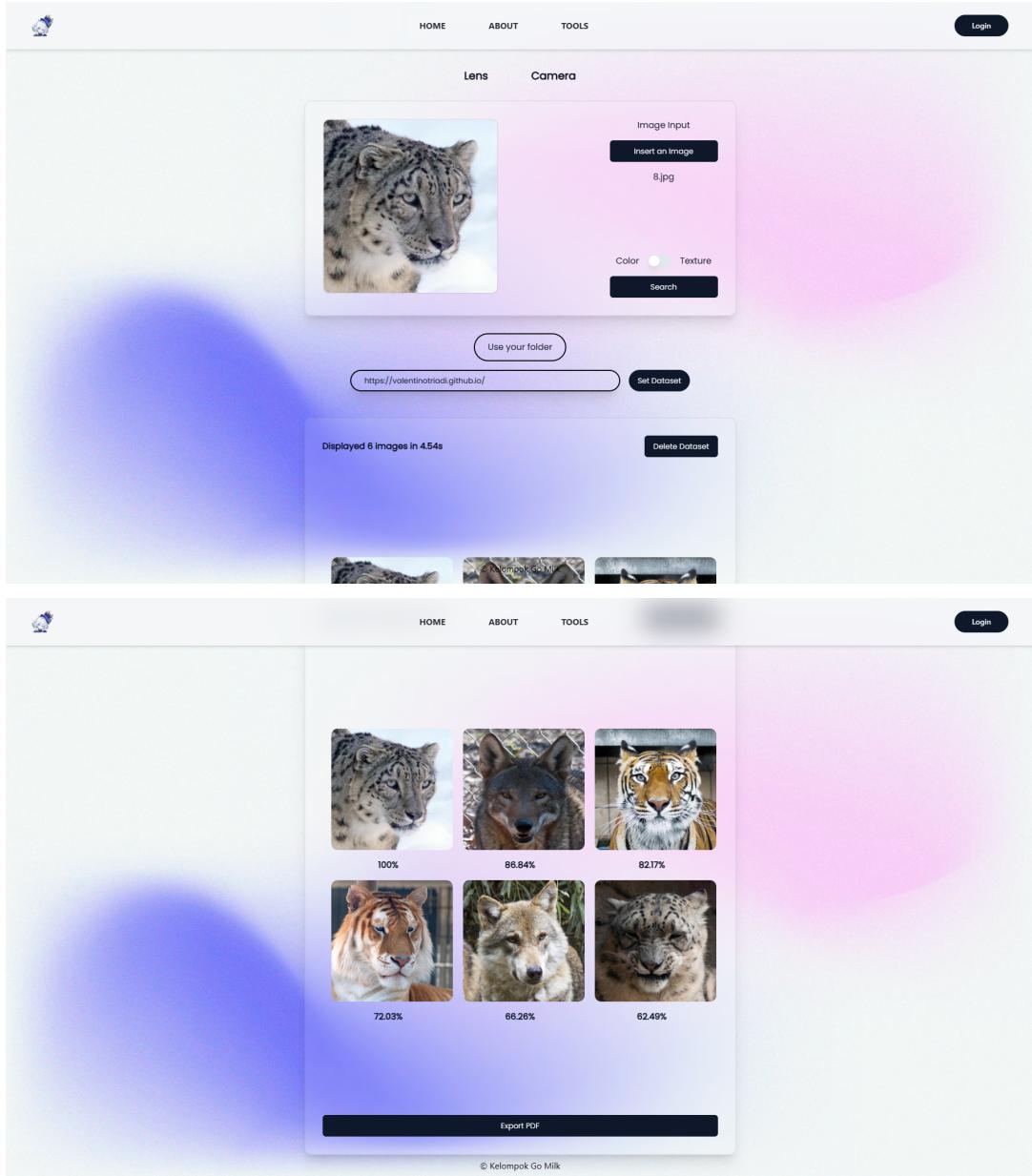
Kami melakukan pencarian berdasarkan parameter warna menggunakan gambar 8.jpg dengan dataset berisi 50 gambar dari 0.jpg – 49.jpg.



Kami melakukan pencarian berdasarkan parameter tekstur menggunakan gambar 8.jpg dengan dataset berisi 50 gambar dari 0.jpg – 49.jpg.



Kami melakukan pencarian berdasarkan parameter warna menggunakan gambar hasil tangkapan kamera dengan dataset berisi 50 gambar dari 0.jpg – 49.jpg.



Kami melakukan pencarian berdasarkan parameter warna menggunakan gambar 8.jpg dengan dataset yang diambil dari website portofolio miliki Valen..

4.5 Analisis Solusi

Berdasarkan hasil pengujian yang kami lakukan, pembangunan Content-Based Image Retrieval (CBIR) dengan fokus pada fitur warna lebih unggul dibandingkan dengan menggunakan fitur tekstur. Terdapat beberapa alasan yang mungkin dapat menjelaskan hasil ini.

Pertama, pemilihan parameter warna dalam CBIR cenderung lebih efektif dalam menggambarkan karakteristik visual dari suatu gambar. Warna dapat menyampaikan informasi

yang kaya dan mudah dipahami oleh program. Dengan fokus pada fitur warna, CBIR dapat dengan lebih akurat memahami dan membandingkan perbedaan warna antar gambar, sehingga meningkatkan tingkat kesesuaian hasil pencarian.

Adanya kejelasan dan ketepatan dalam mewakili warna juga dapat meminimalkan ambiguitas dalam proses pencocokan gambar. Sebaliknya, fitur tekstur mungkin lebih kompleks untuk diekstraksi dan diinterpretasikan, dan interpretasi tekstur dapat bervariasi antar individu.

Dalam konteks ini, pemilihan parameter warna dalam CBIR memberikan pendekatan yang lebih mendekati akurat dan dapat diandalkan dalam menemukan kesesuaian antara gambar yang dicari dan gambar dalam dataset. Oleh karena itu, hasil pengujian menunjukkan bahwa CBIR dengan parameter warna memberikan kinerja yang lebih baik dalam hal ketepatan dan keberlanjutan pencarian gambar.

BAB 5

5.1 Kesimpulan

Dalam proyek Tugas Besar 2 IF2123 Aljabar Linier dan Geometri ini, kami berhasil mengimplementasikan aplikasi Aljabar Vektor untuk Sistem Temu Balik Gambar. Program kami menggunakan salah satu metode perbandingan gambar yaitu CBIR warna dan tekstur dengan ekstraksi fitur, dekomposisi matriks, dan normalisasi. Hasilnya, program mampu membandingkan dua gambar dengan memberikan persentase kemiripan yang dihitung menggunakan *cosine similarity* dan menampilkan hasilnya secara interaktif di web. Fitur utama program mencakup upload gambar, upload dataset, penggunaan kamera, dan *image-scraping*. Program ini memberikan pengalaman pencarian gambar berbasis konten yang efisien dan informatif, menciptakan solusi kuat bagi pengguna yang mencari gambar dengan kemiripan visual.

5.2 Saran

Pelaksanaan Tugas Besar 2 IF2123 Aljabar Linier dan Geometri di Semester I Tahun 2023/2024 merupakan pengalaman yang sangat berharga bagi kami. Dari pengalaman ini, kami ingin berbagi beberapa saran kepada pembaca yang mungkin akan menghadapi tugas serupa di masa depan:

1. Pemahaman Bahasa Pemrograman: Tugas ini menuntut penulisan program dalam bahasa Python, yang mungkin belum familiar bagi sebagian mahasiswa. Saya sangat menyarankan untuk meluangkan waktu yang cukup guna mempelajari bahasa ini dengan baik, terutama jika Anda belum memiliki pengalaman sebelumnya. Karena tugas ini melibatkan banyak fitur yang perlu dikuasai dengan rinci.
2. Kerja Sama Tim: Efektivitas dalam kerja sama tim memiliki peran penting dalam menyelesaikan tugas ini. Kolaborasi secara real-time melalui alat seperti VSCode untuk pembuatan program dan kolaborasi langsung pada Google Docs untuk pembuatan laporan bisa sangat membantu. Selain itu, dalam pengembangan source code, konflik antara anggota tim dapat terjadi. Oleh karena itu, menggunakan alat pengelola versi seperti Github sangat direkomendasikan agar mempermudah manajemen proyek secara asinkron.
3. Pemahaman Fungsi dari Library yang Mempermudah: Tugas ini mungkin melibatkan penggunaan library Python tertentu untuk menyelesaikan permasalahan tertentu. Penting untuk memahami dengan baik fungsi-fungsi yang disediakan oleh library tersebut agar dapat memanfaatkannya secara efektif dalam pengembangan program. Sumber daya online dan dokumentasi resmi library

dapat menjadi panduan yang berguna dalam memahami cara menggunakan fitur-fitur tersebut.

Semoga saran-saran ini membantu pembaca dalam menyiapkan diri untuk menangani tugas serupa di masa depan.

5.3 Komentar atau Tanggapan

Tanggapan kami terhadap Tugas Besar 2 Aljabar Linier dan Geometri di Semester 3 Tahun 2023/2024 sungguh sangat menambah wawasan. Meskipun menghadapi berbagai kesulitan dalam perjalanan penggerjaan tugas ini, kami merasa terbantu oleh dedikasi para asisten yang selalu siap memberikan panduan dan dukungan. Keberadaan teman-teman kelompok juga turut menjadikan proses ini begitu menarik dan membuat kami belajar lebih banyak. Kesulitan yang dihadapi ternyata menjadi tantangan yang memicu semangat eksplorasi kami terhadap konsep-konsep baru dalam aljabar linier dan geometri. Hal ini semakin memperkaya pengalaman pembelajaran kami dan memberikan wawasan lebih mendalam terkait materi yang diajarkan.

5.4 Refleksi

Tugas Besar 2 IF2123 Aljabar Linier dan Geometri di Semester I Tahun 2023/2024 menjadi tantangan berarti pada awal semester ini. Saat menghadapi tugas ini, kami tidak hanya dihadapkan pada kompleksitas materi, tetapi juga melihat bagaimana penerapan konsep aljabar linier dalam sistem temu balik gambar terutama metode CBIR membuka wawasan baru. Melalui eksplorasi ini, kami semakin menyadari relevansi matriks dalam pengembangan teknologi, terutama dalam kehidupan sehari-hari.

Penerapan sistem temu balik gambar (CBIR) menjadi sebuah pusat perhatian yang menarik. Kami menemukan bahwa konsep matriks dapat diaplikasikan dengan cara yang unik, khususnya dalam konteks aplikasi seperti Google Lens. Melalui CBIR, teknologi ini memungkinkan kita untuk mencari dan mengenali objek dari dunia nyata menggunakan gambar sebagai dasar pencarian. Hal ini tidak hanya mencerminkan kemajuan dalam ilmu komputer, tetapi juga memberikan pengalaman praktis yang kita temui sehari-hari, seperti ketika kita menggunakan Google Lens untuk mengidentifikasi objek di sekitar kita.

Seiring dengan penerapan CBIR dalam kehidupan sehari-hari, kami menyadari betapa esensialnya pemahaman konsep aljabar linier. Misalnya, dalam konteks Google Lens, matriks dapat digunakan untuk merepresentasikan fitur-fitur kunci suatu objek dalam gambar, memungkinkan sistem mengenali pola dan karakteristik yang unik. Ini memberikan perspektif baru bagi kami tentang bagaimana fondasi matematika dapat diaplikasikan dengan sangat nyata dalam teknologi yang kita gunakan setiap hari.

Dalam perjalanan kami menggali konsep-konsep ini, kami juga mengevaluasi cara kami berkolaborasi sebagai tim. Menariknya, kerja sama yang solid di antara anggota

kelompok mencerminkan pentingnya sinergi dalam pengembangan teknologi. Dalam konteks CBIR, keseluruhan sistem harus bekerja secara bersama-sama untuk mencapai hasil yang diinginkan, mengingat kompleksitas dan keragaman informasi dalam gambar yang dicari.

Sebagai kesimpulan, tugas besar ini tidak hanya menjadi ujian pemahaman konsep aljabar linier, tetapi juga menjadi jendela yang membuka cakrawala baru terkait dengan aplikasi sistem temu balik gambar dalam kehidupan sehari-hari. Dengan demikian, kami tidak hanya memahami materi secara teoritis, tetapi juga mendapatkan apresiasi yang lebih dalam terhadap dampak nyata dan kemungkinan inovasi teknologi di masa depan.

5.5 Ruang Perbaikan atau Pengembangan

Ruang perbaikan dan pengembangan dalam mengerjakan tugas dapat difokuskan pada beberapa aspek yang dapat ditingkatkan. Pertama-tama, pengaturan waktu menjadi kunci utama. Kami menyadari bahwa perencanaan waktu yang lebih baik dapat meningkatkan efisiensi penggerjaan. Menerapkan strategi manajemen waktu, seperti membuat jadwal yang terstruktur dan menetapkan tenggat waktu internal untuk setiap tahap pekerjaan, dapat membantu menghindari tekanan waktu yang tidak perlu.

Selain itu, ketelitian membaca spesifikasi dari awal menjadi aspek yang perlu diperhatikan. Memahami secara menyeluruh tentang apa yang diminta dalam tugas, termasuk detail-detail kecil, dapat mengurangi risiko kesalahan dan memastikan pekerjaan berjalan sesuai dengan harapan. Menempatkan perhatian ekstra pada spesifikasi tugas dapat meminimalkan revisi dan penyesuaian yang mungkin diperlukan.

Komunikasi tim yang baik juga dapat dioptimalkan. Membuat saluran komunikasi yang jelas dan terbuka di antara anggota tim dapat menghindari kebingungan dan memastikan bahwa semua anggota memiliki pemahaman yang sama tentang tujuan dan tanggung jawab masing-masing. Diskusi reguler dan pembahasan mengenai kemajuan proyek dapat meningkatkan keterlibatan semua anggota tim.

Dalam konteks pengembangan web, penambahan fitur-fitur yang mendukung dapat memberikan nilai tambah. Memperhatikan kebutuhan pengguna dan mengidentifikasi area yang dapat diperbaiki atau ditingkatkan dalam hal fungsionalitas dapat meningkatkan kualitas tugas. Pemikiran kreatif dalam mengimplementasikan fitur-fitur baru yang relevan dengan tujuan proyek dapat membuat proyek lebih bermanfaat dan menarik.

Dalam konteks pengolahan gambar, dapat melakukan eksplorasi lebih jauh lagi mengenai efisiensi kinerja program. Dengan memahami cara agar kinerja program lebih efisien, program pengolahan gambar akan menjadi lebih cepat dan lebih baik lagi. Hal tersebut sangat diperlukan agar program kami bisa menjadi program yang layak untuk dipublikasikan.

Terakhir, evaluasi diri secara berkala dapat menjadi langkah penting. Menerima umpan balik, baik dari anggota tim maupun asisten, dan memanfaatkannya sebagai dasar untuk perbaikan lebih lanjut adalah cara efektif untuk terus berkembang. Selalu terbuka terhadap saran dan berkomitmen untuk belajar dari setiap pengalaman dapat membantu memperbaiki kinerja secara berkelanjutan.

DAFTAR PUSTAKA

Rinaldi Munir. "Aljabar dan Geometri untuk Informatika 2023/2024."

informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/algeo23-24.html

Yue, Jun, Zhenbo Li, Lu Liu, dan Zetian Fu. 2011. "Content-based image retrieval using color and texture fused features." Mathematical and Computer Modelling 54, no. 3-4: 1121-1127. ISSN 0895-7177.

<https://doi.org/10.1016/j.mcm.2010.11.044>: <https://doi.org/10.1016/j.mcm.2010.11.044>.

Yunus, Muhammad. 2020. Feature Extraction: Gray Level Co-occurrence Matrix (GLCM). Medium. Diakses 11 November 2023.

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

LAMPIRAN

- 1. Link Repository Github :** <https://github.com/ValentinoTriadi/Algeo02-22134.git>
- 2. Link Video :**
<https://www.youtube.com/watch?v=wFbkLxOYAZM&feature=youtu.be>