

LAPORAN TUGAS BESAR 02

IF2123 ALJABAR LINEAR GEOMETRI

Kelompok LOOGLE GENS



Disusun Oleh :

Mohammad Andhika Fadillah (13522128)

Justin Aditya Putra Prabakti (13522130)

Mohammad Akmal Ramadhan (13522161)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA - KOMPUTASI

INSTITUT TEKNOLOGI BANDUNG

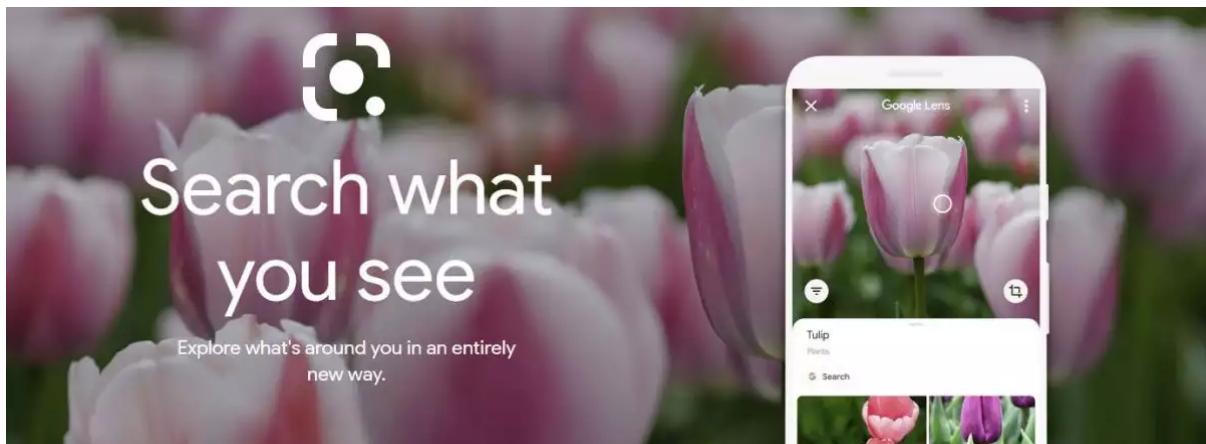
DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
BAB 2 TEORI SINGKAT	6
BAB 3 IMPLEMENTASI PUSTAKA & PROGRAM	13
BAB 4 EKSPERIMEN	24
BAB 5 PENUTUP	42
DAFTAR REFERENSI	44

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.



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

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

CONTENT-BASED INFORMATION RETRIEVAL (CBIR)

Content-Based Image Retrieval (CBIR) adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan ekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur tersebut diekstraksi, mereka diwakili dalam bentuk vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Kemudian, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur gambar dalam dataset. Hasil dari pencocokan ini digunakan untuk mengurutkan gambar-gambar dalam dataset dan menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar dengan cara yang lebih efisien, karena tidak memerlukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai citra visual antara gambar-gambar tersebut. Pada Tugas Besar kali ini, Anda diminta untuk mengimplementasikan 2 parameter CBIR yang paling populer, antara lain :

1. CBIR dengan parameter warna

Pada CBIR kali ini akan dibandingkan *input* dari sebuah *image* dengan *image* yang dimiliki oleh dataset, hal ini dilakukan dengan cara mengubah *image* yang berbentuk RGB menjadi sebuah metode histogram warna yang lebih umum.

Histogram warna adalah frekuensi dari berbagai warna yang ada pada ruang warna tertentu hal ini dilakukan untuk melakukan pendistribusian warna dari *image*. Histogram warna tidak bisa mendeteksi sebuah objek yang spesifik yang terdapat pada *image* dan tidak bisa mendeskripsikan posisi dari warna yang didistribusikan.

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

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

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

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

2. Mencari *Cmax*, *Cmin*, dan Δ

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

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

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

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

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

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

$$V = C_{max}$$

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

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

Dengan *A* dan *B* adalah vektor dan *n* adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *Cosine Similarity*.

Untuk melakukan pencarian histogram, blok *image* dibagi menjadi $n \times n$ blok. Setiap blok akan menjadi tidak terlalu signifikan jika blok-blok tersebut terlalu besar dan akan meningkatkan waktu dalam memprosesnya jika ukuran dari blok terlalu kecil. Pada pencarian blok ini agar lebih efektif disarankan menggunakan 3×3 blok.

2. CBIR dengan parameter tekstur

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

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

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

Melalui persamaan tersebut, digunakan nilai θ adalah 0° , 45° , 90° , dan 135° . Sebagai gambaran, berikut diberikan contoh cara pembuatan *co-occurrence matrix* dan [link cara pembuatannya](#) (Contoh ini dapat digunakan sebagai referensi, bukan acuan sebagai acuan utama).

	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 2. Cara Pembuatan Matrix *Occurrence*

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

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

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

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

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

2. Lakukan kuantifikasi dari nilai *grayscale*. Karena citra *grayscale* berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran 256×256 . Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut. *Grayscale* semula dari suatu

gambar akan dikompresi untuk mengurangi operasi perhitungan sebelum dibentuknya *co-occurrence matrix*.

3. Dari *co-occurrence matrix* bisa diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Persamaan yang dapat digunakan untuk mendapatkan nilai 3 komponen tersebut antara lain :

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 :

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

Keterangan : P merupakan matriks *co-occurrence*

Dari ketiga komponen tersebut, dibuatlah sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan.

4. Ukurlah kemiripan dari kedua gambar dengan menggunakan Teorema *Cosine Similarity*, yaitu:

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

Disini A dan B adalah dua vektor dari dua gambar. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

BAB 2

LANDASAN TEORI

2.1 RGB

RGB adalah singkatan dari Red, Green, dan Blue. Ini merujuk pada model warna yang digunakan dalam tampilan grafis, televisi, dan layar komputer. Model warna ini berdasarkan campuran intensitas dari tiga warna primer: merah, hijau, dan biru, yang digunakan dalam proporsi yang berbeda untuk menciptakan berbagai warna. Dengan mengontrol intensitas ketiga warna ini, kita dapat membuat sejumlah besar warna yang berbeda di layar. Model warna RGB sangat umum dalam teknologi tampilan seperti monitor komputer, televisi, dan berbagai jenis layar lainnya.



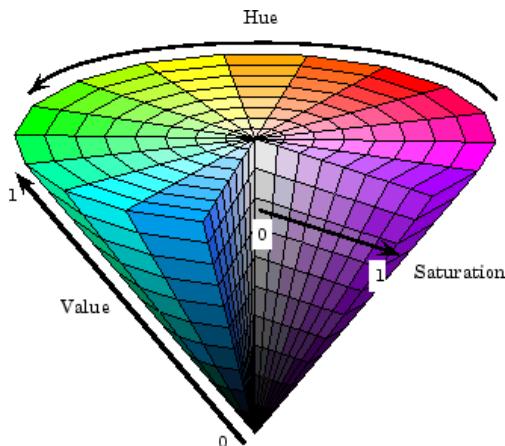
Gambar 3. Warna RGB

2.2 HSV

HSV adalah singkatan dari Hue (Tinggi), Saturation (Saturasi), dan Value (Nilai). Ini adalah model warna alternatif yang digunakan untuk mendefinisikan warna dalam bentuk yang lebih intuitif bagi manusia dibandingkan dengan model warna RGB.

1. **Hue (Tinggi)** mengacu pada "warna" sebenarnya dalam spektrum warna, seperti merah, hijau, biru, dan sebagainya. Dalam model HSV, Hue diukur dalam derajat (0° hingga 360°), membentang melalui roda warna.

2. **Saturation (Saturasi)** menggambarkan kecerahan atau kekuatan warna. Semakin tinggi nilai saturasi, semakin hidup dan jenuh warnanya. Saturasi diukur dalam skala persentase, dari 0% (abu-abu atau monokromatik) hingga 100% (warna penuh).
3. **Value (Nilai)**, sering juga disebut kecerahan, mengacu pada seberapa terang atau gelapnya warna. Value juga diukur dalam skala persentase, dengan 0% mewakili warna hitam dan 100% mewakili warna yang sepenuhnya terang.



Gambar 4. Warna HSV

Model warna HSV memungkinkan pengguna untuk secara intuitif memanipulasi dan mendefinisikan warna berdasarkan aspek-aspek yang lebih dekat dengan persepsi visual manusia. Ini berguna dalam berbagai aplikasi, termasuk grafika komputer, pengolahan citra, dan manipulasi warna dalam desain.

2.3 Histogram

Sebuah histogram adalah representasi grafis dari distribusi data. Ini menunjukkan sebaran frekuensi atau jumlah kemunculan nilai-nilai dalam satu set data. Histogram dibuat

dengan mengelompokkan data ke dalam "bin" atau interval nilai, dan kemudian menggambarkan jumlah pengamatan yang jatuh ke dalam setiap interval tersebut.

Dalam konteks fotografi atau pengolahan citra digital, histogram sering digunakan untuk menganalisis distribusi intensitas piksel dalam gambar. Histogram gambar menunjukkan sebaran intensitas piksel (misalnya, dalam model warna RGB, grayscale, HSV, dll.) dari 0 (nilai minimum) hingga 255 (nilai maksimum).

Histogram gambar memungkinkan untuk melihat sebaran intensitas piksel pada rentang warna tertentu. Misalnya, pada histogram grayscale, sumbu horizontal mewakili intensitas piksel dari hitam (0) hingga putih (255), sedangkan sumbu vertikal menunjukkan jumlah piksel pada setiap tingkat intensitas tersebut.

Histogram memberikan wawasan tentang:

1. **Kontras gambar:** Histogram yang terdistribusi luas menunjukkan gambar dengan rentang tonal yang luas, sedangkan histogram yang terkonsentrasi menunjukkan gambar dengan rentang tonal yang lebih sempit.
2. **Kecerahan:** Histogram dapat membantu menentukan apakah gambar terlalu gelap (kurangnya piksel pada sisi kanan histogram) atau terlalu terang (kurangnya piksel pada sisi kiri histogram).
3. **Dominasi warna:** Dalam histogram warna (seperti dalam model warna RGB atau HSV), puncak pada satu saluran warna menunjukkan dominasi warna tertentu dalam gambar.

2.4 Cosine Similarity

Cosine similarity merupakan suatu rumus yang dapat digunakan untuk menentukan seberapa mirip dua vektor berada dalam ruang multidimensi. Konsep ini sangat berguna dalam analisis data, pengelompokan, pemrosesan bahasa alami, sistem rekomendasi, dan berbagai bidang lain yang melibatkan perbandingan antara elemen-elemen berdasarkan representasi vektor mereka.

Pada dasarnya, cosine similarity mengukur kesamaan arah dari dua vektor dalam ruang multidimensi, bukan jarak sebenarnya. Perhitungan cosine similarity dilakukan dengan menggunakan formula cos dari sudut antara dua vektor.

Misalkan kita memiliki dua vektor A dan B, setiap vektor mewakili elemen atau fitur dalam ruang yang mungkin memiliki banyak dimensi. Cosine similarity antara vektor A dan B dihitung sebagai hasil dari perkalian dot (dot product) antara dua vektor tersebut, dibagi oleh hasil kali dari magnitudo (panjang) masing-masing vektor:

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

Gambar 3. Rumus Cosine Similarity

Hasil cosine similarity berkisar dari -1 hingga 1. Nilai 1 menunjukkan bahwa kedua vektor memiliki arah yang sama, nilai -1 menunjukkan bahwa arah kedua vektor berlawanan, sementara nilai 0 menunjukkan bahwa vektor-vektor tersebut tegak lurus satu sama lain.

2.5 Co-occurrence Matrix

Matriks co-occurrence (co-occurrence matrix) adalah representasi statistik dari frekuensi kemunculan bersama antara dua elemen dalam suatu konteks atau rangkaian data.

Biasanya digunakan dalam pemrosesan bahasa alami dan analisis citra untuk menangkap hubungan antara elemen-elemen dalam suatu domain tertentu.

Dalam analisis citra, matriks co-occurrence juga digunakan untuk menggambarkan hubungan spasial antara piksel dalam citra. Dalam hal ini, matriks merepresentasikan

frekuensi kemunculan pasangan nilai intensitas piksel pada jarak atau arah tertentu di dalam citra.

2.6 Symmetric Matrix

Matriks simetris adalah jenis matriks persegi di mana elemen-elemennya simetris terhadap diagonal utama. Diagonal utama matriks persegi adalah garis dari sudut kiri atas ke sudut kanan bawah, yang memisahkan matriks menjadi dua bagian yang simetris secara reflektif.

Elemen-elemen pada matriks simetrik saling simetrik sepanjang diagonal utamanya. Secara lebih formal, misal a_{ij} menyatakan menyatakan elemen matriks A pada baris ke-i dan kolom ke-j. Matriks A simetri jika dan hanya jika untuk setiap i, j berlaku $a_{ij} = a_{ji}$.

2.7 Matrix Normalization

Matrix normalization dalam konteks co-occurrence matrix adalah proses mengubah nilai-nilai dalam matriks co-occurrence untuk menghasilkan representasi yang lebih terstruktur, terukur, atau lebih relevan untuk analisis yang dilakukan. Co-occurrence matrix merepresentasikan seberapa sering pasangan elemen tertentu muncul bersama dalam sebuah domain, seperti kata-kata dalam teks atau intensitas piksel dalam citra.

Kegunaan matrix normalization dalam co-occurrence matrix adalah untuk:

1. **Mengontrol Skala:** Matriks co-occurrence bisa memiliki nilai-nilai dengan skala yang beragam, yang bisa mempengaruhi analisis. Normalisasi membantu dalam mengendalikan rentang nilai agar tidak terlalu besar atau kecil.
2. **Memudahkan Perbandingan:** Normalisasi membantu dalam perbandingan yang lebih akurat antara elemen-elemen dalam matriks, terutama jika ada variasi skala yang besar.

3. **Penyederhanaan Interpretasi:** Normalisasi bisa membantu dalam penyederhanaan atau peningkatan interpretasi hasil analisis, membuat data lebih mudah dipahami atau diinterpretasikan.

2.8 Grayscale

Grayscale adalah representasi dari gambar atau citra yang hanya menggunakan level keabuan (grayscale) tanpa warna. Dalam gambar grayscale, setiap piksel direpresentasikan dengan tingkat keabuan tunggal, biasanya dari 0 (hitam) hingga 255 (putih) dalam sistem skala abu-abu.

Gambar grayscale menghilangkan informasi warna sehingga hanya menyisakan informasi tentang tingkat kecerahan atau intensitas di setiap piksel. Setiap piksel dalam gambar grayscale memiliki satu saluran warna tunggal (misalnya, dalam model warna RGB, kecerahan ditunjukkan oleh nilai yang sama untuk saluran merah, hijau, dan biru).

2.9 Contrast

Contrast mengacu pada perbedaan tingkat kecerahan atau warna antara area atau elemen dalam citra. Ini adalah salah satu fitur penting yang digunakan untuk mengekstraksi informasi tekstur dari citra. Kontras pada dasarnya mencerminkan seberapa jauh dan tajam perbedaan antara elemen-elemen citra.

Dalam analisis citra, *contrast* dihitung dengan mempertimbangkan distribusi nilai intensitas piksel di area tertentu dalam citra. Area dengan kontras tinggi akan menampilkan perbedaan yang tajam antara tingkat kecerahan di dalamnya, sementara area dengan kontras rendah akan menampilkan perbedaan yang lebih samar atau kurang tajam antara tingkat kecerahan.

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

Gambar 4. Persamaan untuk mencari contrast

Dengan \mathbf{P} adalah matriks *co-occurrence*

2.10 Homogeneity

Homogenitas adalah ukuran seberapa seragam atau serupa tekstur dalam suatu area pada citra. Ini menggambarkan sejauh mana piksel-piksel di dalam area tertentu memiliki pola atau intensitas yang mirip atau seragam.

Untuk mengukur homogenitas, sering kali digunakan metrik yang mengukur seberapa dekat atau serupa nilai piksel di dalam suatu area. Area yang homogen memiliki perbedaan intensitas yang rendah antara piksel-pikselnya, sedangkan area yang tidak homogen memiliki variasi intensitas yang lebih besar.

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

Gambar 5. Persamaan untuk mencari Homogeneity

Dengan \mathbf{P} adalah matriks *co-occurrence*

2.11 Entropy

Entropi adalah ukuran keacakan atau ketidakpastian distribusi intensitas piksel di dalam suatu area pada citra. Ini mengukur sejauh mana informasi yang terdapat dalam distribusi intensitas piksel tersebut. Entropi yang tinggi menunjukkan bahwa distribusi piksel lebih acak atau tidak

teratur, sementara entropi yang rendah menunjukkan adanya pola yang terstruktur atau teratur.

Konsep entropi berasal dari teori informasi dan statistik. Di dalam konteks analisis citra, entropi sering digunakan untuk mengevaluasi tingkat keacakan atau keragaman intensitas piksel di dalam suatu area.

Perhitungan entropi biasanya melibatkan distribusi probabilitas dari nilai intensitas piksel. Jika distribusi probabilitas piksel lebih merata, atau jika tidak ada nilai yang dominan, entropi akan tinggi. Namun, jika distribusinya lebih terkonsentrasi atau teratur, entropi akan rendah.

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

Gambar 6. Persamaan untuk mencari Entropy

Dengan \mathbf{P} adalah matriks *co-occurrence*

2.12 Pengembangan Website

Berikut adalah tahap-tahap umum dalam menciptakan suatu website:

1. Perencanaan: Ketika ingin membuat website, tujuan dan target pengguna website harus dipertimbangkan agar alat yang digunakan dapat memadai tujuan tersebut. Agar suatu website dapat berdiri, diperlukan 2 hal:
 - o Backend : Backend adalah bagian dari website yang memungkinkan suatu website untuk menerima *request*, dan mengirimkan *response* yang diminta oleh pengguna, dalam konteks tugas ini, request yang dikirimkan berbentuk gambar dan dataset yang kemiripannya ingin dihitung, lalu *response* berupa “daftar kemiripan gambar secara terurut” yang nantinya akan ditampilkan pada Frontend
 - o Frontend : Frontend adalah bagian dari suatu website yang sepenuhnya berada di komputer pengguna, tujuan frontend adalah untuk menyajikan informasi yang diminta oleh pengguna. Dalam konteks tugas ini, frontend

akan menunjukkan gambar dan angka persentase kemiripan suatu gambar dataset dengan gambar acuan

2. Desain: Agar website terlihat bagus dan nyaman untuk digunakan, perlu adanya tahap desain. Desain untuk suatu website bisa berupa gambaran tangan, wireframe, atau desain visual lengkap menggunakan alat seperti Adobe XD atau Sketch.
3. Pengembangan: Tahap ini melibatkan penerjemahan desain menjadi kode. Pengembangan umumnya menggunakan bahasa pemrograman seperti HTML, CSS, dan JavaScript untuk membangun halaman web yang interaktif dan responsif. Namun, terdapat berbagai macam alternatif yang disebut “framework” yang bertujuan untuk memudahkan pemrograman hingga memperluas batas kompleksitas suatu frontend.
4. Pengujian: Setelah pengembangan selesai, website diuji untuk memastikan bahwa semua fitur berfungsi dengan baik, tidak ada bug, dan tampilan konsisten di berbagai perangkat dan browser.
5. Launching: Setelah lulus uji, website siap untuk di launch. Ini melibatkan proses memindahkan website dari lingkungan pengembangan ke server atau hosting yang dapat diakses secara publik.
6. Maintenance: Sewaktu-waktu, website dapat mengalami kerusakan atau resiko keamanan, oleh karena itu suatu website memerlukan maintenance/perawatan yang rutin agar tetap nyaman dan aman untuk digunakan melalui tambahan fitur, reset server, atau “security patch”.

Proses ini bisa melibatkan banyak orang dengan keahlian yang berbeda, seperti desainer UI/UX, pengembang front-end dan back-end, serta manajer proyek untuk memastikan semua tahapan berjalan dengan lancar sesuai rencana.

BAB 3

Analisis Pemecahan Masalah

3.1 Langkah-langkah pemecahan masalah.

Pada program *Content-Based Image Retrieval (CBIR)* dengan parameter warna, diperlukan pengubahan dari *image* yang berbentuk RGB menjadi sebuah metode histogram warna yang lebih umum. Hal tersebut dilakukan untuk mempermudah perhitungan untuk menyelesaikan program tersebut.

3.2 Proses pemetaan masalah menjadi elemen-elemen pada aljabar geometri.

Dalam kuliah Aljabar Linier dan Geometri diberitahukan bahwa untuk mencari Kesamaan (sim) antara dua vektor $Q = (q_1, q_2, \dots, q_n)$ dan $D = (d_1, d_2, \dots, d_n)$ diukur dengan rumus cosine similarity yang merupakan bagian dari rumus perkalian titik (dot product) dua buah vektor:

$$Q \cdot D = \|Q\| \|D\| \cos \theta \quad \longrightarrow \quad sim(Q, D) = \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|}$$

Gambar 7. Dot Product & Cosine Similarity

3.3 Contoh ilustrasi kasus dan penyelesaiannya

Menghitung Cosine Similarity:

$$Q = (1, 4, 3)$$

$$D = (2, 7, 5)$$

$$Q \cdot D = 1 \times 2 + 4 \times 7 + 3 \times 5 = 45$$

$$\|Q\| = \sqrt[2]{1^2 + 4^2 + 3^2} = 5,09$$

$$\|D\| = \sqrt[2]{2^2 + 7^2 + 5^2} = 8,83$$

$$\begin{aligned} \text{Cosine similarity} &= \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= 1,001 \end{aligned}$$

BAB 4

Implementasi dan Uji Coba

4.1 Pseudocode

Pseudocode pemrosesan CBIR melalui tekstur:

```
using math
using read

function Contrast(p: int, i: int, j:int) -> float:
-> p * (i-j)**2

function Homogeneity(p: int, i: int, j:int) -> float:
-> p / (1 + (i-j)**2)

function Entropy(p: int, i: int, j:int) -> float:
    retval : float

    if (p != 0) then
        retval <- p * math.log10(p)
        retval <- retval * (-1)
    else
        retval <- 0
    -> retval

function Sigma(m: list of list of int,f: function) -> float:
    total : float

    baris <- length(m)
    kolom <- length(m[1])
    float <- 0
    i traversal [1..baris]:
        j traversal [1..kolom]:
            total <- total + f(m[i][j],i,j)
    -> total

function RGBtoGrayscale(r: list of list of int,g: list of list of int,b: list of list of int) -> list of list of int:
    gray: float
    result: list of list of float

    i traversal [1..length(r)]
        j traversal [1..length(r[1])]
```

```

result[i][j] <- r[i][j] * 0.299 + g[i][j] * 0.587 + b[i][j] * 0.114
-> result

function coocurrence(m: list of list of int, depth: int, distance: int = 1, angle: int = 0) -> list of float:

baris : int (jumlah baris matrix awal)
kolom : int (jumlah baris matrix akhir)
GLCM : list of list of int
totalValue : int (sigma dari nilai dalam GLCM)

baris = length(m)
kolom = length(m[1])
i traversal [1..depth]
    j traversal [1..depth]
        GLCM[i][j] <- 0

{offset baris}
row_offset = distance * math.sin(math.radians(angle))

{rounding ke arah int terjauh dari 0}
if row_offset >= 0 then:
    row_offset <- math.ceil(row_offset)
else:
    row_offset <- math.floor(row_offset)

{offset kolom dengan beberapa case khusus karena cos = inf/NaN}
if (angle == 90):
    col_offset <- 1
elif (angle == 270):
    col_offset <- -1
else:
    col_offset <- distance * math.cos(math.radians(angle))
if row_offset >= 0 then:
    col_offset <- math.ceil(col_offset)
else:
    col_offset <- math.floor(col_offset)

totalValue <- 0

#iterasi matrix dan penambahan langsung dengan transposenya
i traversal [1..baris-row_offset]:
    j traversal [1..kolom-col_offset]:
        row <- int(m[i][j])+1
        col <- int(m[i + row_offset][j + col_offset])+1
        GLCM[row][col] <- 1
        {karena matrix akan ditambahkan dengan transpose nya sendiri maka}
        GLCM[col][row] <- 1

#normalisasi
i traversal [1..depth]:
    j traversal [1..depth]:

```

```

GLCM[i][j] <- float(GLCM[i][j] / totalValue)

-> GLCM

```

```

function process_texture(filepath: str) -> list[float]:
    read.readImgtoRGB(filepath)
    grayscale <- RGBtoGrayscale(read.R, read.G, read.B)
    GLCM <- coocurrence(grayscale,256)
    contrast <- Sigma(GLCM,Contrast)
    homogeneity <- Sigma(GLCM,Homogeneity)
    entropy <- Sigma(GLCM, Entropy)
    -> [contrast, homogeneity, entropy]

```

Pseudocode pemrosesan CBIR melalui Warna :

```

using cv2
using os
using numpy
using time

function hue(r, g, b, delta, Cmax)
    epsilon <- 1e-8
    delta[delta is equal to 0] <- epsilon
    h <- np.zeros_like(delta)
    h[Cmax is equal to r] <- (60 * (((g - b) / delta) % 6))[Cmax is equal to r]
    h[Cmax is equal to r] <- (60 * (((b - r) / delta) + 2))[Cmax is equal to g]
    h[Cmax is equal to r] <- (60 * (((r - g) / delta) + 4))[Cmax is equal to b]
    h <- h/360.0

    -> h

function saturation(cmax, delta)
    epsilon <- 1e-8
    s <- np.where(cmax is equal to 0, 0, delta / (cmax +epsilon))

    -> s

function delta(r, g, b)
    delta <- np.maximum.reduce([r, g, b]) - np.minimum.reduce([r, g, b])

```

```

-> delta

function calculateAverageHSV(image)
    image <- image / 255.0
    r, g, b <- cv2.split(image)
    Cmax <- np.maximum.reduce([r, g, b])
    delta_values <- delta(r, g, b)
    h_avg <- np.mean(hue(r, g, b, delta_values, Cmax))
    s_avg <- np.mean(saturation(Cmax, delta_values))
    V <- Cmax
    v_avg >- np.mean(V)

-> h_avg, s_avg, v_avg

function calculateHistogram(image, n <- 4, bins <- 8)
    hist <- []
    i traversal [1..n]
    j traversal [1..n]
        block <- image[i * (image.shape[0] // n):(i + 1)
                      *(image.shape[0] // n), j* (image.shape[1] // n)
                      :(j + 1) * (image.shape[1] // n)]
        block_avg_hsv <- calculateAverageHSV(block)
        hist.extend(block_avg_hsv)

-> np.divide(hist, np.sum(hist))

function cosineSimilarity(hist1, hist2)
    dot <- np.dot(hist1, hist2)
    norm1 <- np.sqrt(np.dot(hist1, hist1))
    norm2 <- np.sqrt(np.dot(hist2, hist2))
    similarity <- dot / (norm1 * norm2) if (norm1 * norm2) != 0 else 0

-> similarity

function cbirColor(input_image, dataset_dir, bins <- 8)
    input_hist <- calculateHistogram(input_image, bins)
    similarities <- []

    Entry traversal os.scandir(dataset_dir)
        If entry.is_file() then
            dataset_image <- cv2.imread(entry.path)
            dataset_hist <- calculateHistogram(dataset_image, bins)
            similarity <- cosineSimilarity(input_hist, dataset_hist)
            similarities.append(similarity)
        sorted_indices <- np.argsort(similarities)[::-1]
        sorted_similarities <- np.sort(similarities)[::-1]

-> sorted_indices, sorted_similarities
function run():
    dataset_dir <- "src/website/images"
    input_image <- cv2.imread("src/we ")
    start_time <- time.time()

```

```

sorted_indices, sorted_similarities <- cbirColor(input_image, dataset_dir, bins=8)
end_time <- time.time()
elapsed_time <- end_time - start_time

# Counter for images with similarity above 60%
count <- 0

# Iterate over the sorted indices and similarities
i, similarity traversal zip(sorted_indices, sorted_similarities)
    # If the similarity is above 60%
    if similarity > 0.6 then
        # Read the image
        image_path <- os.path.join(dataset_dir, os.listdir(dataset_dir)[i])
        image <- cv2.imread(image_path)
        # Display the image
        cv2.imshow(f"Image {i} - Similarity: {similarity}", image)
        output similarity
        count <- count + 1
    # Print the number of images with similarity above 60% and the execution time
    output Number of images with similarity above 60%: {count - 1}
    Output Elapsed time: {elapsed_time:.2f} seconds
    cv2.waitKey(0)
    cv2.destroyAllWindows()

function process_color(filepath: str) -> list[float]:
    image <- cv2.imread(filepath)
    h_avg, s_avg, v_avg <- calculateAverageHSV(image)

    -> [h_avg, s_avg, v_avg]

if __name__ == "__main__":
    run()

```

4.2 Penjelasan struktur program berdasarkan spesifikasi

Program ini terdiri dari 2 bagian, yaitu front-end dan back-end.

Front-end adalah website yang dibuka secara langsung oleh pengguna, yang terdiri dari file HTML, CSS, dan Javascript. Tugas dari front end adalah memudahkan penggunaan aplikasi.

Back-end menggunakan FastAPI, dan bertugas untuk melakukan semua kalkulasi yang diperlukan dan mengirimkannya ke front end.

4.3 Penjelasan tata cara penggunaan program

Berikut cara penggunaan program

1. Unggah Gambar:

Tekan tombol “*Upload gambar*” untuk meng-*upload* gambar yang ingin dijadikan sebagai acuan untuk *search*

2. Pilih metode pencarian:

Ketika dibuka, *default* sistem adalah mode pencarian Tekstur, jika ingin mengganti ke mode pencarian Warna, tekan tombol “*Swap mode*”

3. Unggah dataset:

Tekan tombol “*Upload dataset*” untuk mengunggah folder yang berisi gambar, atau sambil menekan “*Control*”, klik semua gambar yang hendak dijadikan dataset. Secara default, mengupload *database* tidak akan menghapus dataset yang sebelumnya telah diupload, jika ingin mengganti dataset, tekan tombol “*Delete existing dataset?*” supaya program menghapus dataset setiap kali menerima dataset baru

4. Hasil:

Algoritma kemudian akan menghitung parameter-parameter gambar acuan dan dataset yang sesuai dengan mode. Lalu, menggunakan *cosine similarity*, setiap

gambar pada dataset akan diberi nilai kemiripan dari 0% hingga 100%. Terakhir, gambar akan ditampilkan kepada pengguna secara berurut dari paling mirip, hingga yang paling berbeda.

4.4 Hasil Pengujian

Tekstur

LOOGLE GENS

Home Result How To Use About Us

Reverse Image Search

[Upload Image](#)

[Upload Dataset](#)

[Swap Mode](#)

Current mode: TEKSTUR

[Keep Dataset](#)

Delete existing dataset?: FALSE



Search

LOOGLE GENS

Home Result How To Use About Us

Result



100.00%



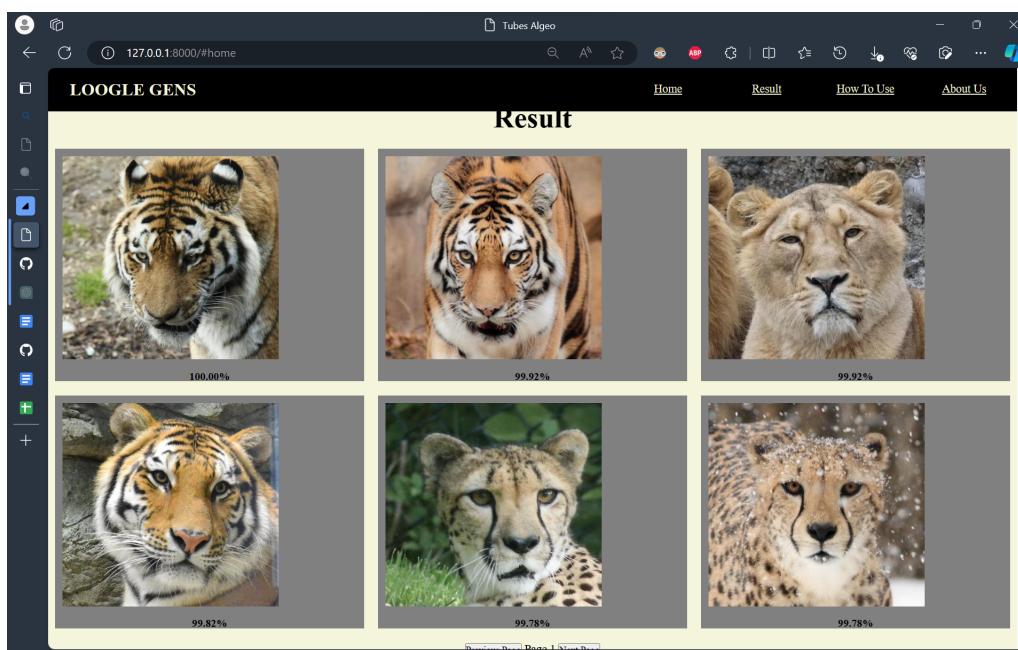
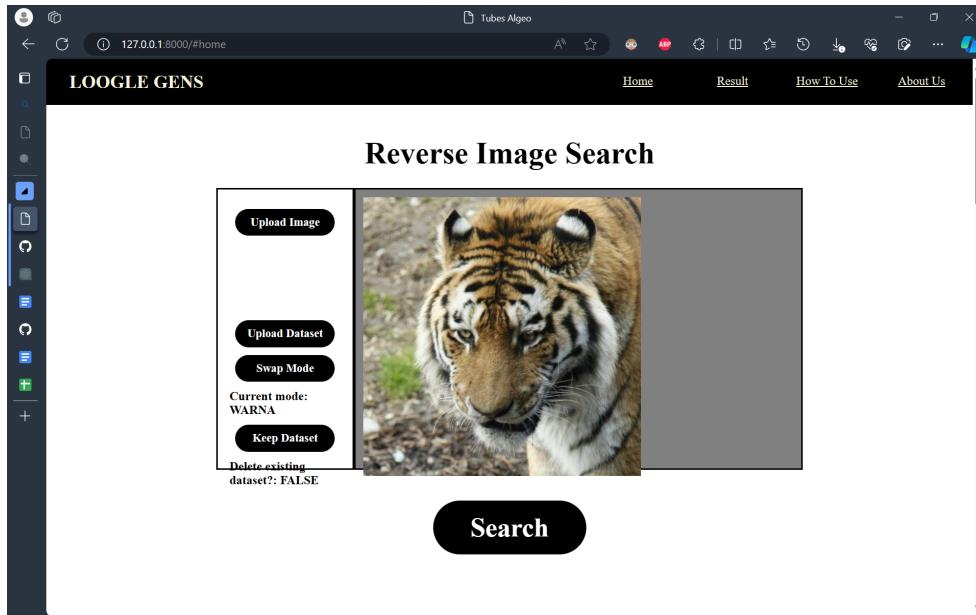
100.00%



100.00%

[Previous Page](#) Page 1 [Next Page](#)

Warna



4.5 Analisis

Analisis dari desain solusi algoritma pencarian yang diimplementasikan pada setiap pengujian yang dilakukan. Misalnya, apakah pembangunan CBIR berdasarkan fitur warna lebih baik dari tekstur pada kasus-kasus tertentu beserta analisis mengenai mengapa hal itu bisa terjadi jika benar.

Terdapat kejanggalan pada hasil CBIR Tekstur karena selalu menghasilkan angka yang tinggi. Ini disebabkan oleh nilai contrast yang sangat tinggi jika dibandingkan dengan nilai Homogeneity dan Entropy



Sampel gambar yang digunakan

```
+ <> + iT
[2] 1s
RGB shape: (355, 200, 3)
[234.1915351404909, 0.39253224090768374, 3.0960555661330345]

0s
print(process_texture('Kucing.jpg'))

RGB shape: (266, 200, 3)
[292.3325839724953, 0.203741863951901, 3.922597779151431]
```

Hasil kalkulasi

Besarnya contrast ini menyebabkan kedua vektor terlihat sama, sama seperti ketika melihat 2 titik berdekatan dari jarak yang cukup jauh, titik akan terlihat menyatu menjadi satu titik.

Untuk mendapatkan hasil yang lebih akurat, diperlukan normalisasi. Salah satu cara yang dapat dilakukan adalah dengan membagi hasil kalkulasi setiap image dataset dengan hasil kalkulasi image acuan, dengan metode ini, persentase yang dihasilkan lebih baik

Dengan perubahan berikut:

```
    return [contrast, homogeneity, entropy]
else:
    return [contrast/norm[0], homogeneity/norm[1], entropy/norm[2]])
```

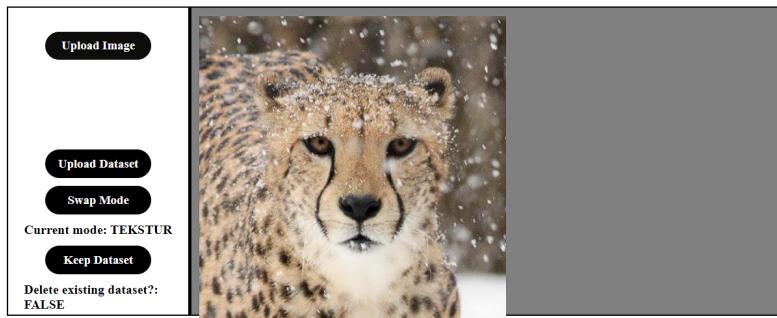
(norm adalah hasil dari image acuan yang akan di search)

Dan untuk cosine similarity:

```
#print(img_data_dict[filename])
if mode == "TEKSTUR":
    value = cosineSimilarity([1,1,1],img_data_dict[filename]["value"])
else:
```

Didapatkan hasil Tekstur yang lebih memuaskan

Reverse Image Search



The screenshot shows the 'Result' page of the LOOGLE GENS application. It displays three search results in a grid format:

- Result 1:** An image of a cheetah's face with the caption "100.00%".
- Result 2:** An image of several Coca-Cola cans with the caption "93.95%".
- Result 3:** An image of a person wearing glasses and a green hoodie, sitting at a desk with a laptop, with the caption "90.55%".

At the bottom of the page, there are navigation links: 'Previous Page', 'Page 1', and 'Next Page'.

BAB 5

PENUTUP

5.1 Kesimpulan

Dalam tugas besar kali ini, kami mempelajari banyak hal baru, yaitu mengimplementasikan Content-Based Image Retrieval (CBIR) dengan dua parameter utama: warna dan tekstur.

Pertama, untuk CBIR berbasis warna, prosesnya dimulai dengan ekstraksi fitur-fitur warna seperti histogram warna global dan blok. Konversi warna dari RGB ke HSV dilakukan untuk memudahkan perhitungan. Selanjutnya, pencarian kesamaan dilakukan menggunakan cosine similarity antara vektor-fitur gambar input dan dataset, dengan pemilihan blok 4x4 untuk pencarian histogram.

Kedua, CBIR berbasis tekstur memanfaatkan co-occurrence matrix untuk mengekstraksi tekstur dari gambar. Langkah-langkahnya mencakup konversi ke citra grayscale, pembuatan co-occurrence matrix dengan nilai offset yang berbeda, pembuatan matriks simetris, dan normalisasi matrix. Dari co-occurrence matrix, diekstraksi tiga komponen tekstur: contrast, entropy, dan homogeneity, yang kemudian membentuk vektor untuk pengukuran kemiripan.

Aplikasi dari CBIR ini mengharuskan pengguna untuk memasukkan gambar query dan dataset gambar, kemudian memilih parameter pencarian berdasarkan warna atau tekstur. Hasilnya akan menampilkan gambar-gambar dari dataset yang mirip dengan gambar query, diurutkan berdasarkan tingkat kemiripan.

Kesimpulannya, CBIR ini menawarkan cara untuk mencari gambar dengan cara yang lebih efisien, tanpa menggunakan kata kunci atau teks, melainkan dengan membandingkan nilai-nilai citra visual. Ini adalah pendekatan yang cocok untuk pencarian gambar berdasarkan konten visual daripada metode pencarian berbasis teks.

5.2 Saran

- Pemaparan akan materi yang dijadikan tugas seharusnya dijelaskan terlebih dahulu sehingga tidak terjadi kekeliruan saat proses pengerajan.
- Pemaparan materi yang berkaitan dengan mata kuliah Aljabar Geometri, seharusnya dicantumkan dengan jelas pada spesifikasi agar tidak terjadi kekeliruan.

5.3 Komentar atau Tanggapan

Kami membuka segala bentuk komentar atau tanggapan terkait proses dan hasil pengerajan tugas besar kami ini. Kami yakin bahwa komentar atau tanggapan dari orang lain akan membawa perkembangan bagi pengeraaan tugas besar kami. Kami juga ingin berkomentar terhadap tugas besar Algeo kedua ini. Menurut kami, tugas besar ini memiliki banyak manfaat bagi masing - masing dari kami. Kami mempelajari banyak sekali ilmu baru yang tentunya berguna bagi kami.

5.4 Refleksi

Melalui tugas besar ini, kami mendapatkan berbagai macam pengalaman. Kami dapat mengasah kemampuan bekerjasama, berkomunikasi, mendekomposisikan berbagai macam permasalahan yang ada, serta melatih kedisiplinan dalam mengerjakan tugas masing-masing. Kami juga mengetahui lebih dalam tentang bahasa pemrograman python beserta library baru yang digunakan seperti OpenCV2.

5.5 Ruang Perbaikan

Semua hal tidaklah bisa untuk mencapai kesempurnaan. Kami yakin penggerjaan yang kami lakukan masih jauh dari kata sempurna. Masih banyak hal yang dapat dikembangkan seperti di bagian website, kami yakin masih banyak di luar sana yang memiliki tampilan dan fungsionalitas website yang lebih baik dari kami. Tetapi, itu tidak membuat kami putus asa, kami bersyukur atas pencapaian dan penggerjaan yang kami buat.

DAFTAR PUSTAKA

- <https://www.pinecone.io/learn/series/image-search/color-histograms/>
- https://en.wikipedia.org/wiki/Content-based_image_retrieval
- <https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm>
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/>
- Link Repository

Berikut kami lampirkan tautan repository yang kami buat untuk tugas besar 2 Aljabar Linear dan Geometri : <https://github.com/akmalrmn/Algeo02-22128>

