

LAPORAN TUGAS BESAR II
IF2123 ALJABAR LINEAR DAN GEOMETRI
PENERAPAN METRIK BERBASISKAN VEKTOR DALAM CONTENT-BASED
INFORMATION RETRIEVAL (CBIR)



Disusun oleh:
Ga-Reela

Aland Mulia Pratama (13522124)
Rizqika Mulia Pratama (13522126)
Ikhwan Al Hakim (13522147)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

DAFTAR ISI

BAB I	2
DESKRIPSI MASALAH	2
BAB II	4
LANDASAN TEORI	4
I. Dasar Teori Content Based Information Retrieval (CBIR)	4
II. Pengembangan Website	5
BAB III	9
ANALISIS PEMECAHAN MASALAH	9
I. Mengumpulkan dan Menyimpan Dataset	9
II. Mengekstraksi Fitur-Fitur dari Dataset	9
III. Mencocokkan Fitur-Fitur Kueri dengan Dataset	10
IV. Graphical User Interface (GUI)	11
BAB IV	14
IMPLEMENTASI DAN UJI COBA	14
I. Implementasi Program (Pseudocode)	14
a) Fitur CBIR Dengan Parameter Color	14
b) Fitur CBIR Dengan Parameter Tekstur	17
II. Struktur Program	20
III. Tata cara Penggunaan Program	22
IV. Hasil Pengujian	23
V. Analisis Parameter Warna dan Tekstur	29
BAB V	31
PENUTUP	31
I. Kesimpulan	31
II. Saran	31
III. Komentar atau Tanggapan	31
IV. Refleksi terhadap Tugas Besar	31
V. Ruang Perbaikan atau Pengembangan	32
DAFTAR PUSTAKA	33

BAB I

DESKRIPSI MASALAH

I. Tujuan

Berikut adalah tujuan dari Tugas Besar 2 Aljabar Linear dan Geometri:

1. Implementasi CBIR dan Aljabar Vektor dalam sebuah Website
 - a) Pengembangan Sistem CBIR
 - b) Penerapan Aljabar Vektor
 - c) mengembangkan sistem temu balik gambar yang efisien
2. Fungsionalitas dan Keefektifan Sistem
 - a) Mengimplementasikan teknik ekstraksi fitur seperti warna, tekstur, dan bentuk dari gambar untuk membangun basis data gambar.
 - b) Mengembangkan antarmuka (*interface*) yang intuitif bagi pengguna untuk melakukan pencarian gambar berbasis konten.
 - c) Mengimplementasikan algoritma pencarian yang efisien berdasarkan representasi aljabar vektor untuk mendapatkan hasil pencarian yang akurat dan cepat.

II. Spesifikasi Tugas / Program

Buatlah program sistem temu balik gambar (*image retrieval system*) 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**.

III. Bonus / Fitur Tambahan Program

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*.
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. Tautan video yang dikumpulkan tidak boleh menggunakan url shortener seperti bit.ly, tinyurl.com, dll.

BAB II

LANDASAN TEORI

I. Dasar Teori *Content Based Information Retrieval* (CBIR)

Pencarian dan Pengambilan Gambar Berbasis Konten (CBIR) merupakan proses untuk mencari dan mengambil gambar berdasarkan isi visualnya. Proses ini dimulai dengan mengekstrak fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur ini diekstrak, mereka diwakili dalam bentuk vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Selanjutnya, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor fitur dari gambar yang dicari dengan vektor fitur gambar dalam dataset. Hasil dari perbandingan ini digunakan untuk menyusun peringkat gambar dalam dataset, menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar secara efisien, karena tidak bergantung pada pencarian berbasis teks atau kata kunci, melainkan pada kesamaan visual antara gambar-gambar tersebut. Berikut adalah 2 parameter *Content Based Information Retrieval* (CBIR) yang kami terapkan pada program:

1. CBIR dengan parameter Warna

Pada CBIR ini, akan dibandingkan input gambar dengan gambar-gambar dalam dataset, dengan cara mengubah gambar dalam format RGB menjadi histogram warna, suatu metode yang umum digunakan.

Histogram warna adalah distribusi frekuensi dari berbagai warna yang muncul dalam suatu ruang warna tertentu. Ini dilakukan untuk memetakan bagaimana warna didistribusikan di dalam gambar. Namun, histogram warna tidak mampu mengidentifikasi objek spesifik dalam gambar atau deskripsikan letak posisi warna yang terdistribusi.

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

Pembentukan ruang warna melibatkan pembagian nilai-nilai citra ke dalam rentang nilai yang lebih kecil, dimana tiap rentang dianggap sebagai bin dalam pembuatan histogram warna. Histogram warna dihitung dengan menghitung jumlah piksel yang mewakili nilai warna pada setiap rentang. Fitur warna termasuk histogram warna secara global dan histogram warna berbasis blok.

Ketika menghitung histogram, ruang warna HSV lebih sering dipilih karena dapat digunakan untuk gambar yang memiliki latar belakang putih, yang umum digunakan. Oleh karena itu, konversi warna dari RGB ke HSV diperlukan dalam proses ini.

2. CBIR dengan parameter Tekstur

Analisis Citra Berbasis Konten (CBIR) dengan perbandingan tekstur sering kali menggunakan suatu matriks yang disebut matriks kookurensi (co-occurrence matrix). Penggunaan matriks ini dipilih karena memfasilitasi pemrosesan yang efisien dan cepat, menghasilkan vektor dengan ukuran yang lebih kecil. Untuk menjelaskan konsep ini, mari asumsikan kita memiliki gambar I dengan dimensi $n \times m$ piksel dan suatu parameter offset ($\Delta x, \Delta y$).

Matriks kookurensi dapat dirumuskan sebagai berikut:

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

Dalam rumus ini, $C(p,q)$ mewakili elemen matriks kookurensi pada posisi p,q , sedangkan $I(i,j)$ adalah nilai intensitas piksel pada posisi i,j dalam gambar. Parameter Δx dan Δy bergantung pada arah θ dan jarak yang didefinisikan dalam suatu konteks tertentu.

Dengan menggunakan nilai intensitas i dan j sebagai representasi intensitas gambar, serta p dan q sebagai posisi dalam gambar, kita dapat menghitung matriks kookurensi untuk mengevaluasi hubungan antara piksel-piksel tersebut. Pendekatan ini memungkinkan ekstraksi fitur tekstur dengan representasi matriks yang lebih ringkas, memudahkan proses perbandingan dan analisis citra berbasis tekstur dalam CBIR.

II. Pengembangan Website

1. Dependency dan Framework

Luaran dari percobaan kali ini adalah sebuah *website*, maka ada beberapa dependency dan framework yang kami gunakan:

- FastAPI

Untuk bagian back-end, digunakan sebuah framework web untuk membangun API RESTful dengan Python yang bernama FastAPI. FastAPI adalah framework yang modern, cepat (berkinerja tinggi), dan mudah dipelajari, yang berdasarkan pada *type hints* standar Python untuk melakukan validasi, serialisasi, dan deserialisasi data. FastAPI juga secara otomatis menghasilkan dokumentasi OpenAPI untuk API yang dibangun dengannya. FastAPI mendukung sepenuhnya pemrograman asinkron dan dapat berjalan

dengan server ASGI seperti Uvicorn dan Gunicorn, sehingga cocok untuk lingkungan produksi.

- React-JS

Untuk bagian front-end, kami menggunakan sebuah framework pengembangan website yaitu React-JS yang merupakan salah satu *library* atau pustaka JavaScript. React-JS dapat memberikan pengalaman website yang cepat dan interaktif bagi pengguna. React merupakan arsitektur berbasis komponen, di mana tampilan website dirancang menggunakan komponen yang nantinya diimpor ke dalam halaman utama website. React menggunakan format file JSX yaitu *syntaks extension* yang memungkinkan kami menulis elemen HTML dalam kode JavaScript. Format file JSX di-transpile menjadi JavaScript reguler sebelum di-render nantinya ke dalam website. React-JS sangat mendukung pengerjaan tugas ini dengan beberapa *library* atau pustaka pendukungnya yaitu:

- React-Router-DOM

react-router-dom adalah pustaka tambahan untuk React yang dirancang khusus untuk menangani perutean (routing) dalam aplikasi React. Dengan menggunakan react-router-dom, Anda dapat membuat aplikasi React yang memiliki navigasi berbasis komponen dan memungkinkan perubahan tampilan tanpa perlu melakukan permintaan ulang halaman. *BrowserRouter* adalah komponen yang menyediakan konteks untuk aplikasi Anda dan mengelola keadaan riwayat peramban (browser). Ini adalah pembungkus utama yang membungkus seluruh aplikasi dan memungkinkan penggunaan fitur-fitur perutean. *Route* adalah komponen yang digunakan untuk menentukan pencocokan URL dan menentukan komponen yang harus di render ketika URL sesuai. *Link* adalah komponen yang digunakan untuk membuat tautan yang memicu perubahan URL tanpa perlu memuat ulang halaman.

- Tailwind-CSS

Tailwind CSS adalah kerangka kerja CSS yang dirancang untuk mempermudah pembuatan antarmuka pengguna dengan memberikan sekumpulan kelas utilitas yang dapat digunakan untuk menggambarkan gaya dan tata letak elemen HTML. Berbeda dengan kerangka kerja CSS tradisional yang cenderung menyediakan komponen-komponen siap pakai, Tailwind lebih fokus pada penggunaan kelas-kelas kecil yang dapat diterapkan langsung pada elemen HTML. Tailwind menyediakan utilitas kelas yang mencakup berbagai properti CSS. Sebagai contoh, untuk memberikan warna teks merah, Anda dapat menggunakan kelas '*text-red-500*'. Tailwind dirancang untuk menjadi ringan dan cepat. Dengan menggunakan hanya kelas-kelas yang diperlukan, ukuran file CSS yang dihasilkan dapat diperkecil, mengoptimalkan waktu pemuat halaman.

- Python

Untuk pengolahan data, digunakan bahasa pemrograman Python, hal ini dikarenakan Python memiliki kelebihan dalam memproses berbagai jenis data dalam jumlah yang besar serta dibekali dengan banyak library yang mendukung dan mempermudah pengolahan data secara *seamless*. Dalam percobaan kali ini python banyak membantu dalam proses pengolahan matriks gambar berkat beberapa library pendukungnya:

- OpenCV

OpenCV merupakan library yang dapat melakukan computer vision dan image processing dengan cepat dan akurat. OpenCV menyediakan fungsi-fungsi yang dapat digunakan untuk membaca, menulis, mengubah, menampilkan, dan menganalisis gambar. OpenCV juga dapat melakukan operasi gambar, seperti deteksi wajah, pengenalan objek, segmentasi gambar, ekstraksi fitur, dan lain-lain. OpenCV juga dapat berinteraksi dengan library lain yang menggunakan array sebagai struktur data utama, seperti NumPy, SciPy, Matplotlib, dan lain-lain.

- NumPy

NumPy merupakan library yang dapat melakukan operasi matematika dan statistika pada array dan matriks multidimensi dengan cepat dan efisien. NumPy juga dapat berinteraksi dengan library lain yang menggunakan array sebagai struktur data utama, seperti SciPy, Matplotlib, Pandas, dan lain-lain. NumPy sangat berguna untuk mengolah matriks gambar, karena gambar dapat direpresentasikan sebagai array dua atau tiga dimensi yang berisi nilai piksel. NumPy dapat melakukan operasi matriks, seperti perkalian, invers, transpose, dan lain-lain, dengan mudah dan cepat.

- Joblib

Joblib merupakan library yang dapat melakukan parallel computing dengan mudah dan efektif. Joblib menyediakan fungsi-fungsi yang dapat digunakan untuk menjalankan beberapa proses secara bersamaan, seperti looping, mapping, dan lain-lain. Joblib juga dapat melakukan caching dan persistensi data, sehingga dapat menghemat waktu dan memori. Joblib sangat berguna untuk mengolah matriks gambar, karena dapat mempercepat proses pengolahan yang membutuhkan banyak komputasi, seperti ekstraksi fitur, klasifikasi, dan lain-lain.

- Shutil

Shutil merupakan library yang dapat melakukan operasi file dan direktori dengan mudah dan efektif. Shutil menyediakan fungsi-fungsi yang dapat digunakan untuk menyalin, memindahkan, menghapus, dan mengubah nama file dan direktori. Shutil juga dapat melakukan kompresi dan dekompresi file, sehingga dapat menghemat ruang penyimpanan. Shutil sangat berguna untuk mengolah matriks gambar, karena dapat membantu Anda mengatur file-file gambar yang ada dalam direktori, seperti mengelompokkan, mengurutkan, dan memfilter gambar.

- Subprocess

Subprocess: library yang dapat menjalankan perintah shell atau program eksternal dengan mudah dan efektif. Subprocess menyediakan fungsi-fungsi yang dapat digunakan untuk membuat, menjalankan, dan mengontrol proses-proses yang berjalan di luar interpreter Python. Subprocess juga dapat melakukan komunikasi antara proses-proses tersebut, seperti mengirim dan menerima input dan output. Subprocess sangat berguna untuk mengolah matriks gambar, karena dapat membantu Anda menjalankan program-program yang tidak tersedia dalam Python, seperti program C, C++, Java, dan lain-lain, yang mungkin lebih cepat atau lebih akurat dalam mengolah gambar.

- Concurrently

Concurrently adalah sebuah library Python dan Node yang memungkinkan pengguna untuk menjalankan dua proses atau lebih secara bersamaan yang pada percobaan kali ini digunakan untuk menjalankan front-end dan back-end secara bersamaan.

- Axios

Axios adalah sebuah library yang dapat digunakan untuk melakukan HTTP request dengan menggunakan promise berbasis JavaScript. Axios dapat digunakan baik di browser maupun di node.js, dan dapat berinteraksi dengan berbagai format data, seperti JSON, XML, URL encoded, dan lain-lain. Axios juga dapat melakukan konfigurasi, intercept, transform, dan cancel request, serta menghasilkan dokumentasi API secara otomatis

BAB III

ANALISIS PEMECAHAN MASALAH

I. Mengumpulkan dan Menyimpan Dataset

Sebelum kami dapat melakukan ekstraksi fitur-fitur yang relevan dari dataset yang kami miliki, kami harus terlebih dahulu melakukan proses pengumpulan dan penempatan dataset tersebut. Dataset ini nantinya akan digunakan sebagai sumber data yang akan memberikan hasil output sesuai dengan kueri yang diberikan oleh pengguna melalui antarmuka web. Dataset kami tempatkan di folder src/database yang berada di dalam struktur direktori proyek kami. Dengan demikian, dataset dapat diakses dengan mudah oleh front-end yang bertanggung jawab untuk menampilkan hasil output kepada pengguna, sekaligus dapat dilacak dan diatur dengan baik oleh back-end yang bertanggung jawab untuk melakukan proses ekstraksi fitur dan pencocokan kueri.

Dataset didapatkan dari dua sumber utama, yaitu unggahan dari pengguna yang ingin menggunakan data mereka, dan hasil scraping dari internet yang dilakukan secara otomatis menggunakan skrip Python.

II. Mengekstraksi Fitur-Fitur dari Dataset

a. Fitur Warna

Untuk fitur warna, kami menggunakan kelas ColorDescriptor yang kami buat sendiri untuk menghitung deskripsi warna dari gambar. Kelas ini memiliki dua metode utama, yaitu describe dan histogram. Metode describe bertugas untuk membagi gambar menjadi dua blok yang sama besar, dan menghitung histogram warna untuk setiap blok. Histogram warna dihitung dalam ruang warna HSV, yang memiliki tiga komponen, yaitu hue, saturation, dan value. Hue merepresentasikan warna dasar, saturation merepresentasikan kejemuhan warna, dan value merepresentasikan kecerahan warna.

Metode histogram bertugas untuk menghitung frekuensi kemunculan nilai-nilai HSV dalam setiap blok, dan menghasilkan array satu dimensi yang berisi nilai-nilai histogram yang sudah dinormalisasi. Array ini kemudian digabungkan menjadi satu array yang berisi deskripsi warna dari seluruh gambar. Deskripsi warna ini nantinya akan digunakan untuk mencocokan gambar dengan kueri pengguna.

b. Fitur Tekstur

Untuk fitur tekstur, kami menggunakan beberapa fungsi yang kami buat sendiri untuk menghitung deskripsi tekstur dari gambar. Fungsi-fungsi ini adalah Normalisasi, coocurrenceMatrix, symmetricMatrix, normalizationMatrix, contrast, entropy, homogeneity, dan similarity. Fungsi Normalisasi bertugas untuk mengubah gambar berwarna menjadi gambar grayscale sesuai dengan spesifikasi yang kami tentukan.

Fungsi coocurrenceMatrix bertugas untuk menghitung matriks ko-korensi dari gambar grayscale, yaitu matriks yang merepresentasikan frekuensi kemunculan pasangan nilai pixel yang bersebelahan secara horizontal.

Fungsi `symmetricMatrix` bertugas untuk membuat matriks ko-korensi menjadi simetris, yaitu dengan menambahkan matriks ko-korensi dengan transposenya.

Fungsi `normalizationMatrix` bertugas untuk menormalisasi matriks simetris, yaitu dengan membagi setiap elemen matriks dengan jumlah seluruh elemen matriks.

Fungsi `contrast`, `entropy`, dan `homogeneity` bertugas untuk menghitung nilai kontras, entropi, dan homogenitas dari matriks normalisasi. Nilai-nilai ini merupakan fitur tekstur yang kami gunakan untuk menggambarkan karakteristik tekstur dari gambar. Fungsi `similarity` bertugas untuk menghitung kesamaan antara dua vektor fitur tekstur, yaitu dengan menggunakan rumus cosine similarity. Kesamaan ini nantinya akan digunakan untuk mencocokan gambar dengan kueri pengguna.

III. Mencocokkan Fitur-Fitur Kueri dengan Dataset

a. Fitur Warna

Untuk mencocokkan kueri dengan dataset berdasarkan fitur warna, kami menggunakan dua kelas yang kami buat sendiri, yaitu `ColorDescriptor` dan `ColorSearcher`. Kelas `ColorDescriptor` bertugas untuk menghitung deskripsi warna dari gambar, seperti yang telah dijelaskan sebelumnya. Kelas `ColorSearcher` bertugas untuk mencari gambar-gambar yang memiliki fitur warna yang mirip dengan fitur warna dari gambar kueri. Kelas ini memiliki dua metode utama, yaitu `init` dan `search`. Metode `init` bertugas untuk membaca indeks fitur warna dari dataset, yang telah kami simpan di file `src/conf/conf_color.csv`. File ini berisi nama file dan nilai-nilai fitur warna dari setiap gambar dalam dataset. Metode `search` bertugas untuk mencari gambar-gambar yang memiliki nilai fitur warna yang paling dekat dengan nilai fitur warna dari gambar kueri. Metode ini menggunakan rumus jarak chi-squared untuk menghitung jarak antara dua vektor fitur warna. Semakin kecil jaraknya, semakin mirip fitur warnanya. Metode ini mengembalikan daftar hasil pencarian yang berisi skor kemiripan dan nama file dari gambar-gambar yang mirip dengan gambar kueri.

Setelah kami mendapatkan daftar hasil pencarian, kami melakukan beberapa langkah tambahan untuk menampilkan hasil pencarian kepada pengguna. Pertama, kami hanya menampilkan gambar-gambar yang memiliki skor kemiripan di atas 60%, yaitu gambar-gambar yang memiliki fitur warna yang cukup mirip dengan gambar kueri. Kedua, kami mengurutkan gambar-gambar tersebut berdasarkan skor kemiripan dari yang tertinggi ke yang terendah. Ketiga, kami menyimpan nama file dan skor kemiripan dari gambar-gambar tersebut dalam sebuah dictionary bernama `result_color`. Kami juga menyimpan waktu yang dibutuhkan untuk melakukan pencarian dan jumlah gambar yang cocok dalam dictionary tersebut. Keempat, kami menulis dictionary tersebut ke dalam file JSON bernama `src/conf/result_color.json`, yang nantinya akan digunakan oleh front-end untuk menampilkan hasil pencarian kepada pengguna.

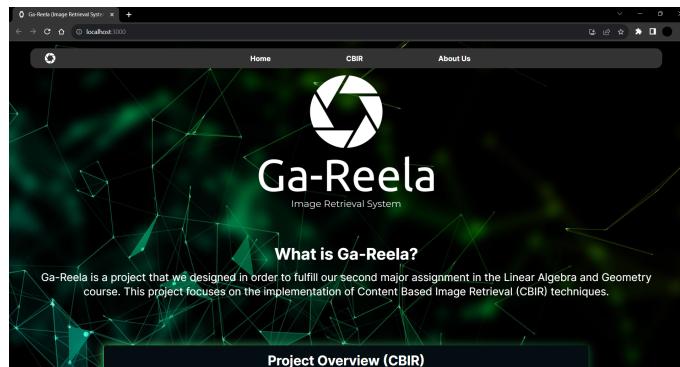
b. Fitur Tekstur

Untuk mencocokkan kueri dengan dataset berdasarkan fitur tekstur, kami menggunakan dua kelas yang kami buat sendiri, yaitu `TextureDescriptor` dan `TextureSearcher`. Kelas `TextureDescriptor` bertugas untuk menghitung deskripsi tekstur

dari gambar, seperti yang telah dijelaskan sebelumnya. Kelas TextureSearcher bertugas untuk mencari gambar-gambar yang memiliki fitur tekstur yang mirip dengan fitur tekstur dari gambar kueri. Kelas ini memiliki dua metode utama, yaitu init dan search. Metode init bertugas untuk membaca indeks fitur tekstur dari dataset, yang telah kami simpan di file src/conf/conf_texture.csv. File ini berisi nama file dan nilai-nilai fitur tekstur dari setiap gambar dalam dataset. Metode search bertugas untuk mencari gambar-gambar yang memiliki nilai fitur tekstur yang paling dekat dengan nilai fitur tekstur dari gambar kueri. Metode ini menggunakan rumus cosine similarity untuk menghitung kesamaan antara dua vektor fitur tekstur. Semakin besar kesamaannya, semakin mirip fitur tekturnya. Metode ini mengembalikan daftar hasil pencarian yang berisi skor kesamaan dan nama file dari gambar-gambar yang mirip dengan gambar kueri.

Setelah kami mendapatkan daftar hasil pencarian, kami melakukan beberapa langkah tambahan untuk menampilkan hasil pencarian kepada pengguna. Pertama, kami hanya menampilkan gambar-gambar yang memiliki skor kesamaan di atas 60%, yaitu gambar-gambar yang memiliki fitur tekstur yang cukup mirip dengan gambar kueri. Kedua, kami mengurutkan gambar-gambar tersebut berdasarkan skor kesamaan dari yang tertinggi ke yang terendah. Ketiga, kami menyimpan nama file dan skor kesamaan dari gambar-gambar tersebut dalam sebuah dictionary bernama result_texture. Kami juga menyimpan waktu yang dibutuhkan untuk melakukan pencarian dan jumlah gambar yang cocok dalam dictionary tersebut. Keempat, kami menulis dictionary tersebut ke dalam file JSON bernama src/conf/result_texture.json, yang nantinya akan digunakan oleh front-end untuk menampilkan hasil pencarian kepada pengguna.

IV. Graphical User Interface (GUI)



Program kami mengintegrasikan website sebagai *Graphical User Interface*(GUI), dengan menggunakan React JS untuk front-end dan FastAPI untuk back-end pada eksperimen ini.

1. Upload Panel



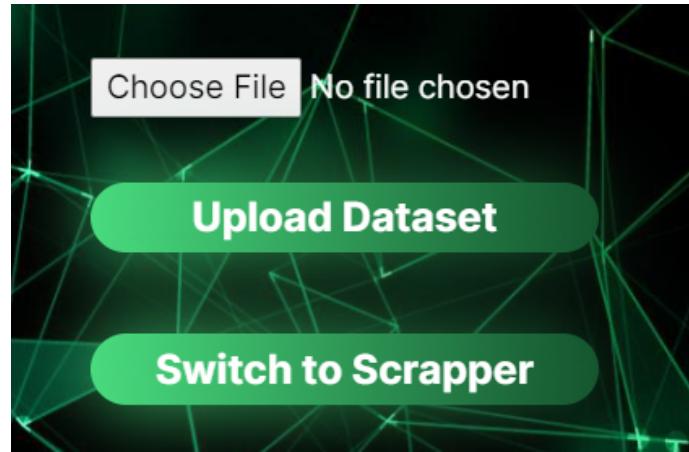
Panel ini menyajikan judul, tempat untuk memasukkan kueri gambar, tombol unggah, serta pratinjau gambar yang telah diunggah. Saat pengguna menekan tombol pencarian, program secara otomatis akan menjalankan kode Python untuk mencari persentase kemiripan kueri dengan setiap gambar dalam dataset. Hasil pengolahan tersebut kemudian dicatat dalam file JSON untuk diakses oleh panel tampilan gambar.

2. Camera Panel



Panel ini serupa dengan panel unggah, namun memungkinkan pengguna memasukkan kueri langsung melalui kamera. Pengguna hanya perlu menekan "Start Camera," dan kamera akan secara otomatis mengambil gambar setiap 15 detik, menampilkan pratinjau hingga pengguna menekan "Stop Camera." Proses pengolahan gambar tetap sama dengan panel unggah.

3. Upload Dataset Panel



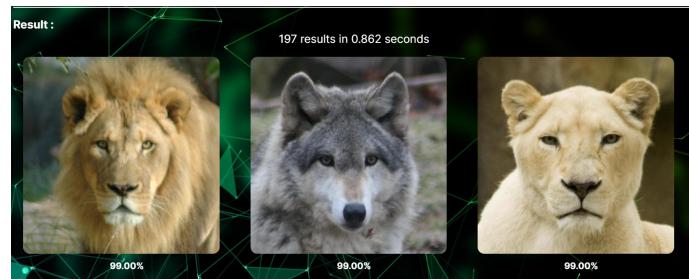
Panel ini berguna untuk mengunggah dataset; pengguna hanya perlu memasukkan folder dari perangkat lokal. Saat tombol unggah dataset ditekan, program akan menjalankan fungsi database_init untuk mengekstraksi fitur warna dan tekstur dari semua gambar dalam dataset, menyimpan hasilnya dalam file CSV.

4. Image Scraping Panel



Panel ini memiliki proses pengolahan yang serupa dengan panel unggah dataset, namun dataset yang digunakan berasal dari internet. Terdapat kolom untuk memasukkan alamat web yang berisi gambar yang ingin diambil. Program akan menyimpan gambar hasilnya ke dalam folder database dan mengekstrak fitur warna serta tekstur dari dataset.

5. Show Image Panel



Panel ini terdiri dari tiga bagian utama. Pertama, terdapat saklar untuk memilih apakah ingin menampilkan hasil berdasarkan fitur warna atau fitur tekstur. Kedua, terdapat tombol "Show Result" dan "Hide Result" untuk menampilkan atau menyembunyikan hasil. Terakhir, terdapat bagian untuk menampilkan gambar berdasarkan persentase kemiripan, mulai dari yang tertinggi hingga terendah, beserta jumlah hasil dan waktu yang diperlukan untuk menemukan semua hasil.

BAB IV

IMPLEMENTASI DAN UJI COBA

I. Implementasi Program (*Pseudocode*)

- a) Fitur CBIR Dengan Parameter Color

Fungsi Pembagi Gambar Menjadi Blok 2x2

```
function describe(input self, image : array) -> list
{Menghitung deskripsi warna dari gambar, membagi gambar
jadi dua blok dan histogram warna dihitung untuk setiap
blok.}

KAMUS LOKAL
USE Numpy as np
USE cv2

i, j : integer
block_size_x, block_size_y, block_startX, block_endX,
block_startY, block_endY : real
image, h, w, block, hsv_block : array
hist, features : list

ALGORITMA
image <- image.astype("float") /255.0
features <- []
h, w <- image.shape[:2]
block_size_x <- w // self.blocks
block_size_y <- h // self.blocks

i traversal [0...self.blocks]
    j traversal [0...self.blocks]
        block_startX <- i * block_size_x
        block_endX <- (i*1) * block_size_x
        block_startY <- j * block_size_y
        block_endY <- (j*1) * block_size_y

            block <- image[block_startY..block_endY,
block_startX..block_endX]
            block <- (block * 255).astype(np.uint8)

            hsv_block <- cv2.cvtColor(block,
cv2.COLOR_RGB2HSV)

            hist <-self.histogram(hsv_block)
```

```

        features.extend(hist)
-> features

```

Fungsi Histogram

```

function histogram(input r, g, b : real)
{Fungsi untuk hitung histogram dalam ruang warna hsv.}

```

KAMUS LOKAL

USE Numpy as np
USE cv2

self : class
hsv : array

ALGORITMA

```

hist <- cv2.calcHist([hsv], [0, 1, 2], None, self.bins,
[0, 180, 0, 256, 0, 256])
hist <- cv2.normalize(hist, hist).flatten()
-> hist

```

Fungsi Search

```

function search(input self, queryFeatures : class limit : integer)
{Mencari tingkat kemiripan dan menentukan gambar yang
mirip dengan query.}

```

KAMUS LOKAL

USE cosine_milarity
USE numpy as np
USE csv

features : list
d, v, k : real
result : dictionary

ALGORITMA

```

with assign(self.indexpath) as f
    reader <- csv.reader(f)
    next(reader)
    row traversal [reader]
    features <- [float(x) x traversal row[1:]]

```

```

d <- self.cosine_similarity(features,
queryFeatures)
    Result[row[0]] <- d
    close(f)
Results <- sorted([(v, k) (k,v) traversal
[result.items()]], reverse = true)
-> results[:limit]

```

Fungsi *Process Image*

```

function process_image(input imagePath : string) ->
dictionary
{Fungsi ekstraksi fitur dari database yang diupload.}

KAMUS LOKAL
USE Joblib
USE csv
USE glob
USE cv2
USE time
USE texture_descriptor
USE color_descriptor

imagePath, imageID : string
features_color, features_texture : list
image : array

ALGORITMA
imageID <- imagePath[imagePath.rfind("\\") + 1:]
image <- cv2.imread(imagePath)
r, g, b <- cv2.split(image)
features_color <- cd.describe(image)
Features_texture <-
texture_descriptor.Tekstur_Feature(image, r, g, b)
-> {'imageID': imageID, 'features_color': features_color,
'features_texture': features_texture}

```

Fungsi Cosine Similarity

```

function cosine_similarity(input self : class vector1,
vector2 : array)
{Mencari kemiripan berdasarkan perbandingan perbandingan

```

```
histogram dua vektor.}
```

KAMUS LOKAL

```
USE Numpy as np
```

```
USE cv2
```

```
self : class
```

```
vector1, vector2 : array
```

```
norm_vector1, norm_vector2, similarity : real
```

ALGORITMA

```
dot_product <- np.dot(vector1, vector2)
```

```
norm_vector1 <- np.linalg.norm(vector1)
```

```
norm_vector2 <- np.linalg.norm(vector2)
```

```
similarity <- dot_product / (norm_vector1 * norm_vector2)
```

```
-> similarity
```

b) Fitur CBIR Dengan Parameter Tekstur**Fungsi Normalisasi Warna (RGB)**

```
function normalisasi(input r, g, b : real)
```

```
{Konversi warna gambar menjadi grayscale, ini dilakukan  
karena warna tidaklah penting dalam penentuan tekstur.}
```

KAMUS LOKAL

```
y : real
```

ALGORITMA

```
r <- r * 0.29
```

```
g <- g * 0.587
```

```
b <- b * 0.114
```

```
y <- r + g + b
```

```
-> y
```

Fungsi *co-occurrence* Matrix

```
function coocurrenceMatrix(input list : Matriks, tinggi,  
lebar: integer)
```

```
{Konversi warna gambar menjadi grayscale, ini dilakukan  
karena warna tidaklah penting dalam penentuan tekstur.}
```

KAMUS LOKAL

USE numpy
coocurence : Matriks
X, Y : Integer

ALGORITMA

```
coocurence <- np.zeros((256,256), dtype = real)
cekuser <- False
cekpass <- False
i traversal [0..tinggi]
    j traversal [0..lebar-1]
        X <- round(listi,j)
        Y <- round(listi,j+1)
        coocurence[X,Y] += 1
-> coocurence
```

Fungsi Matriks Simetri

```
function symmetricMatrix(input matrix : Matriks)
{Mengubah matriks menjadi matriks symmetric.
I.S. : matrix terdefinisi
F.S. : mengembalikan nilai matriks
}
```

KAMUS LOKAL

USE numpy
symMatrix : Matriks

ALGORITMA

```
symMatrix <- np.zeros((256,256), dtype=float)
symMatrix <- matrix + np.transpose(matrix)
-> symMatrix
```

Fungsi Normalisasi Matriks

```
function normalizationMatrix(input matrix : Matriks)
{Mengembalikan matriks yang telah di normalisasi.}
```

KAMUS LOKAL

USE numpy

ALGORITMA

```
divisor <- np.sum(matrix)
```

```

normalization <- np.zeros((256,256),dtype=float)
normalization <- matrix / divisor

-> normalization

```

Fungsi Contrast

function contrast(<u>input</u> Matriks : Matriks) {ekstraksi komponen entropy dari parameter tekstur.}
KAMUS LOKAL <u>USE</u> numpy
ALGORITMA -> (np.sum(np.square(np.arange(256)[:, np.newaxis] - np.arange(256)) * matrix))

Fungsi Homogeneity

function homogeneity(<u>input</u> Matriks : Matriks) {ekstraksi komponen entropy dari parameter tekstur.}
KAMUS LOKAL <u>USE</u> numpy
ALGORITMA -> np.sum(matrix / (1+np.abs(np.arange(256)[:,np.newaxis] - np.arange(256))))

Fungsi Entropy

function entropy(<u>input</u> Matriks : Matriks) {ekstraksi komponen entropy dari parameter tekstur.}
KAMUS LOKAL <u>USE</u> numpy
ALGORITMA -> -(np.sum(matrix * np.log(matrix + 1e-8)))

Cosine Similarity

```
function similarity(input e1, h1, c1, e2, h2, c2: Real)
{mencari nilai kemiripan berdasarkan komponen Homogeneity,
entropy, dan contrast.}
```

KAMUS LOKAL

```
use numpy
VectorA, VectorB : Array of Real
dot_product : real
norm_A, norm_B : real
```

ALGORITMA

```
VectorA <- np.array([Contrast1, Homogeneity1, Entropy1])
VectorB <- np.array([Contrast2, Homogeneity2, Entropy2])

dot_product <- np.sum(vectorA * vectorB)
norm_A <- np.linalg.norm(vectorA)
norm_B <- np.linalg.norm(vectorB)
-> (dot_product / (norm_A * norm_B))
```

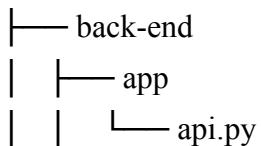
II. Struktur Program

Dalam Tugas Besar ini, kami telah mengadopsi pendekatan yang memisahkan folder back-end dan front-end dalam proyek kami. Struktur folder disusun dengan folder back-end yang terpisah dengan folder front-end. Tujuan dari langkah ini adalah untuk menghindari kebingungan saat melakukan pengembangan baik pada bagian front-end maupun back-end.

Dengan struktur ini, *resource* front-end dan back-end tetap terpisah secara jelas, memudahkan tim dalam mengelola dan memahami setiap aspek dari proyek. Meskipun kedua bagian tersebut berada dalam hierarki folder yang berbeda, kemampuan untuk menjalankan keduanya secara bersamaan tetap terjamin. Hal ini dimungkinkan melalui penggunaan library concurrently, yang memungkinkan komputer untuk mengeksekusi dua proses secara simultan.

Langkah ini tidak hanya meningkatkan keteraturan proyek, tetapi juga memperjelas struktur kerja tim. Dengan demikian, kami yakin bahwa pendekatan ini akan memberikan kontribusi positif terhadap efisiensi pengembangan proyek kami.

Algeo02-22124



```
├── feature
│   ├── color_descriptor.py
│   ├── color_runner.py
│   ├── color_searcher.py
│   ├── database_init.py
│   ├── texture_descriptor.py
│   ├── texture_runner.py
│   └── texture_searcher.py
└── main.py
├── src (front-end)
│   ├── assets
│   ├── components
│   │   ├── Aboutus.jsx
│   │   ├── CameraUploader.js
│   │   ├── Cbir.jsx
│   │   ├── Cybereye.jsx
│   │   ├── DatasetUploader.js
│   │   ├── Home.jsx
│   │   ├── ImageScrapper.js
│   │   ├── ImageUploader.js
│   │   ├── MainPagination.jsx
│   │   └── Navbar.jsx
│   ├── conf
│   │   ├── conf_color.csv
│   │   ├── conf_texture.csv
│   │   ├── hasil.json
│   │   ├── result_color.json
│   │   └── result_texture.json
└── doc
    └── Algeo02-22124.pdf
├── database
├── fonts
└── uploads
```

```

    └── uploads.jpg
    ├── img
    │   ├── tampilan-aboutus.jpg
    │   ├── tampilan-cbir.jpg
    │   └── rampilan-home.jpg
    ├── test
    │   ├── dataset1
    │   ├── dataset2
    │   ├── scraping1
    │   ├── scraping2
    │   ├── dataset1
    │   ├── input-gradient.png
    │   └── input-naruto.png
    ├── App.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── reportWebVitals
    └── setupTests.js

```

III. Tata cara Penggunaan Program

Untuk memanfaatkan program Ga-Reela *Content Based Image Retrieval* (CBIR) dengan optimal pastikan node.js dan npm (Node Package Manager) telah terinstall dalam sistem. Berikut adalah langkah-langkah untuk menjalankan program Ga-Reela yang telah kami buat:

1. Clone repository pada local files anda, repository dapat diakses melalui pranala [berikut](#).
2. Buka terminal, pastikan *directory* terminal berada di dalam folder repository yang telah diclone.

Konfigurasi Front-End

3. Jalankan perintah `npm install` untuk mengaktifkan framework react-JS di dalam local anda.
4. Jalankan perintah `npm install react-router-dom` untuk mengaktifkan pustaka routing ke path-path lain.

Konfigurasi Back-End

5. Jalankan perintah `pip install opencv-python` untuk mengaktifkan pustaka pengolahan citra.
6. Jalankan perintah `pip install numpy` untuk mengaktifkan pustaka perhitungan matematis dari python.
7. Jalankan perintah `pip install joblib` yang kami gunakan guna membantu pemrosesan parallel.
8. Jalankan perintah `npm install -g concurrently` untuk menjalankan frontend dan backend dalam satu localhost yang sama.
9. Jalankan perintah `npm install axios` untuk menerima ataupun mengirim response.

Cara menjalankan program

10. Ubahlah directory terminal ke dalam folder `Algeo02-22124`.
11. Jalankan `npm start`.
12. Pindah ke halaman CBIR melalui Navigation Bar pada website dan unggah dataset pada tombol `upload dataset`.
13. Anda juga bisa menggunakan dataset melalui sebuah situs web dengan tombol `switch to image scraper`.
14. Pilih gambar yang ingin dicari dalam dataset dengan menekan tombol `upload image`.
15. Anda juga dapat memilih alternatif input gambar yaitu dengan kamera secara *real time* dengan menekan tombol `switch to camera`.

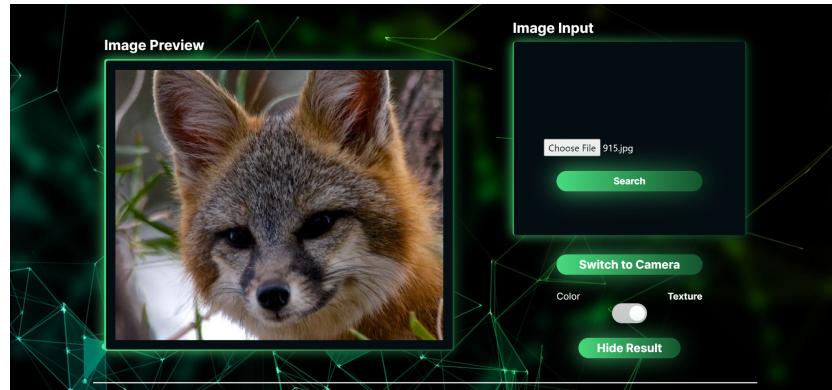
IV. Hasil Pengujian

Disediakan dua dataset yang akan digunakan untuk fitur pencarian gambar dengan parameter warna maupun tekstur dengan dataset yang diupload melalui *local files* atau melalui sebuah situs dan data diperoleh melalui *image scraping*. Input terhadap program dapat melalui gambar yang diunggah dari *local* maupun gambar yang diperoleh dari camera secara *real time*. Dataset yang disediakan melalui local files diberi nama dataset1 dan dataset2. Data yang digunakan untuk fitur *image scraping* adalah [link1](#) dan [link2](#).

a) Pencarian dari Dataset (dataset1, upload)

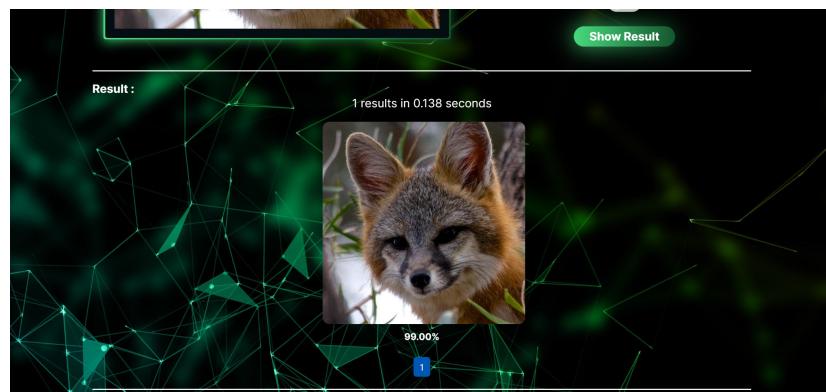
Pada percobaan ini, kita akan mencoba melakukan proses pencarian citra atau gambar dengan input atau masukan gambar sama dengan salah satu dataset yang digunakan. Kami akan menggunakan dataset1 untuk melakukan pengujian pencarian dari dataset.

Input atau masukan gambar



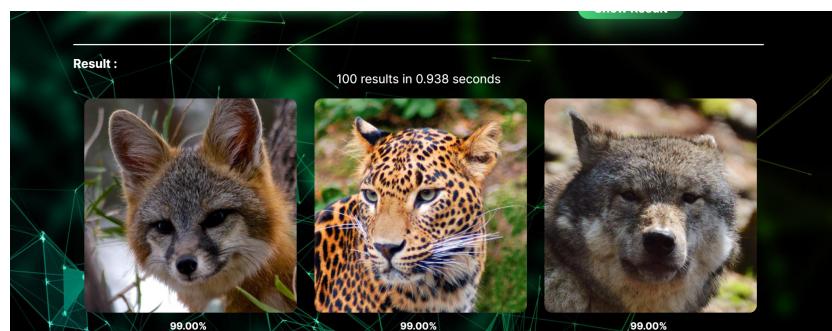
Gambar 4.1. Input citra pencarian dari dataset

Pencarian berdasarkan Parameter Color



Gambar 4.2. Hasil pencarian dari dataset parameter warna

Pencarian berdasarkan Parameter Tekstur



Gambar 4.3. Hasil pencarian dari dataset parameter tekstur

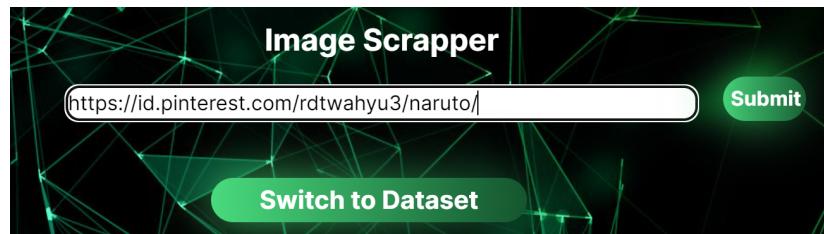
Ketika kita melakukan pencarian menggunakan gambar yang terdapat dalam dataset maka program akan menampilkan gambar yang dimasukkan sebagai input dari hasil pencarian.

b) Pencarian dari luar dataset

Kami akan mencoba juga pencarian gambar dengan gambar yang serupa namun input atau masukan gambar serupa tetapi tidak terdapat dalam dataset.

Kami akan menggunakan fitur image scraping untuk perolehan dataset melalui [link1](https://id.pinterest.com/rdt wahyu3/naruto/) berikut.

Input Dataset melalui *Image Scraping*



Gambar 4.4. Input dataset naruto dari melalui *image scraping*

Input atau masukan gambar



Gambar 4.5. Input citra pencarian dari luar dataset

Pencarian berdasarkan parameter warna



Gambar 4.6. Hasil pencarian dari luar dataset parameter warna

Pencarian berdasarkan parameter tekstur



Gambar 4.7. Hasil pencarian dari luar dataset parameter tekstur

Ketika kita melakukan pencarian berdasarkan parameter warna, tidak dapat ditemukan gambar yang ingin dicari dalam dataset melalui *image scraping*. Hal ini disebabkan karena perbedaan pencahayaan gambar memberikan perubahan yang signifikan terhadap nilai warna pada citra yang ingin dicari. Parameter tekstur dapat memberikan gambar yang ingin dicari dalam dataset. Namun, gambar Naruto tidak muncul pada peringkat pertama hasil pencarian berdasarkan parameter tekstur. Gambar Naruto pertama yang muncul pada hasil berada pada peringkat ke-5 pencarian.

c) Pencarian gambar yang tidak dikenal dari dataset (link2, camera)

Kami akan mencoba pencarian gambar dengan gambar yang tidak dikenal dari dataset. Pengertian dari gambar tidak dikenal disini adalah menggunakan input atau masukan yang berbeda dengan dataset. Kami akan menguji gambar tidak dikenal yang memiliki wajah dan juga gambar yang tidak memiliki wajah. Input atau masukan gambar yang ingin dicari menggunakan fitur unggah gambar dan juga camera. Kami akan menggunakan fitur *image scraping* untuk perolehan dataset melalui [link2](#) berikut.

Input Dataset melalui *Image Scraping*



Gambar 4.8. Input dataset pokemon melalui *image scraping*

Percobaan dengan gambar tidak dikenal yang memiliki wajah:

Input atau masukan gambar (camera)



Gambar 4.9. Input citra pencarian dari luar dataset menggunakan camera

Pencarian berdasarkan parameter warna



Gambar 4.10. Hasil pencarian gambar yang memiliki wajah berdasarkan parameter warna

Pencarian berdasarkan parameter tekstur

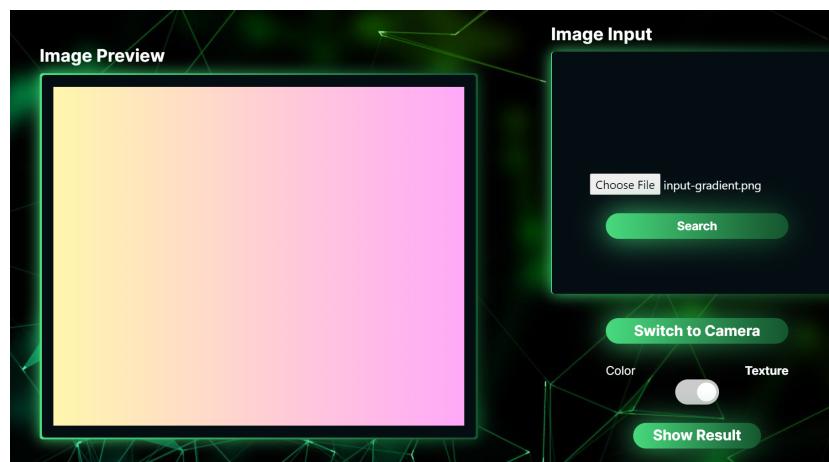


Gambar 4.11. Hasil pencarian gambar yang memiliki wajah berdasarkan parameter tekstur

Ketika melakukan pencarian dengan masukan gambar tidak dikenali yang memiliki wajah dari dataset berdasarkan parameter warna, program tidak menemukan gambar yang ingin dicari. Hal ini disebabkan parameter warna sensitif terhadap perubahan pada komponen warna dari citra. Pencarian gambar dengan parameter tekstur tetap memberikan hasil pencarian tetapi dengan persentase kemiripan yang lebih rendah dari percobaan bagian b. Hal ini disebabkan karena parameter tekstur sulit membedakan tekstur dari wajah yang dimiliki citra. Sehingga mayoritas gambar yang muncul pada peringkat atas hasil pencarian adalah gambar/citra yang memiliki wajah.

Percobaan dengan gambar yang tidak memiliki wajah:

Input atau masukan gambar (camera)



Gambar 4.12. Input citra yang tidak memiliki wajah

Pencarian berdasarkan parameter warna



Gambar 4.10. Hasil pencarian gambar yang tidak memiliki wajah berdasarkan parameter warna

Pencarian berdasarkan parameter tekstur



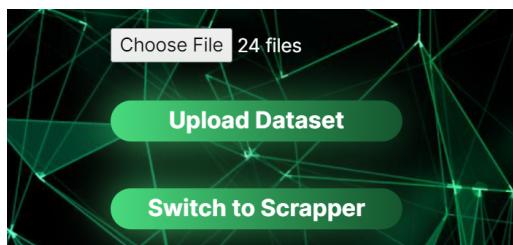
Gambar 4.11. Hasil pencarian gambar yang tidak memiliki wajah berdasarkan parameter tekstur

Ketika melakukan pencarian dengan masukan gambar yang tidak memiliki wajah berdasarkan parameter warna, program tidak menemukan gambar yang ingin dicari. Hal ini terjadi juga untuk kasus pencarian dengan parameter tekstur. Hal ini disebabkan karena citra yang tidak memiliki wajah memiliki perbedaan yang sangat signifikan dengan citra yang memiliki wajah.

d) Penggunaan Kamera sebagai *input*

Kami akan mencoba pencarian gambar untuk mengidentifikasi anggota kelompok Ga-Reela. *Input* atau masukan gambar kami peroleh melalui fitur tambahan camera. Kami akan menggunakan dataset2 yang berisi gambar dari masing-masing anggota kelompok Ga-Reela.

Input Dataset2 melalui *upload dataset*



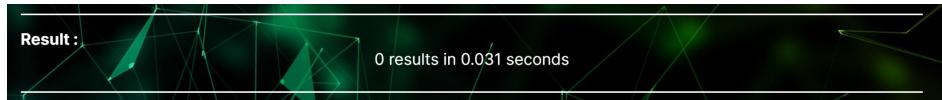
Gambar 4.12. Input dataset2 dengan unggah gambar

Input atau masukan gambar (camera)



Gambar 4.13. Input gambar yang ingin dicari dengan fitur kamera

Hasil pencarian berdasarkan parameter warna



Gambar 4.14. Hasil pencarian parameter warna fitur kamera

Hasil pencarian berdasarkan parameter tekstur



Gambar 4.15. Hasil pencarian parameter tekstur fitur kamera

Ketika melakukan pencarian dengan masukan gambar melalui kamera berdasarkan parameter warna, program tidak menemukan gambar yang ingin dicari. Hal ini disebabkan karena pencahayaan gambar saya pada dataset2 berbeda dengan pencahayaan saat saya mengambil gambar melalui kamera. Program menampilkan hasil yang relevan saat melakukan pencarian berdasarkan tekstur. Namun, persentase kemiripan saya pada hasil program tidak terlalu besar dikarenakan pencahayaan juga memberikan pengaruh terhadap parameter tekstur tetapi tidak signifikan.

V. Analisis Parameter Warna dan Tekstur

Parameter Warna memberikan representasi visual yang intuitif dari gambar, memudahkan program untuk mengenali dan memahami informasi visual. Warna dapat membantu dalam deteksi objek atau identifikasi pola tertentu dalam gambar. Namun, parameter ini memiliki beberapa kelemahan yaitu sangat sensitif terhadap perubahan kondisi pencahayaan, mempengaruhi akurasi sistem, terutama jika data pengujian diambil dalam kondisi pencahayaan yang berbeda. Analisis warna juga memerlukan perhitungan yang lebih kompleks dibandingkan dengan beberapa parameter lainnya.

Tekstur menyediakan informasi tambahan tentang pola atau struktur dalam gambar, yang dapat berguna dalam identifikasi dan pencarian gambar. Tekstur cenderung lebih tahan terhadap perubahan pencahayaan dibandingkan dengan parameter warna karena perubahan warna tidak memberikan peran yang berarti dalam analisis ini. Namun,

hasil dari analisis tekstur ini seringkali bersifat subjektif dan tidak merepresentasikan hasil yang sebenarnya.

Ukuran blok dalam analisis warna dan tekstur dapat mempengaruhi tingkat detail yang dihasilkan. Blok yang lebih kecil dapat menangkap detail yang lebih halus tetapi meningkatkan waktu pemrosesan, sedangkan blok yang lebih besar mungkin lebih efisien secara komputasi tetapi dapat melewatkannya variasi yang halus. Setiap parameter yang kita atur memberikan hubungan yang berlawanan antara performa dan akurasi. Hasil yang lebih baik dapat dilakukan dengan mengkombinasikan parameter warna dan juga tekstur untuk representasi gambar yang lebih komprehensif.

BAB V

PENUTUP

I. Kesimpulan

1. Pengembangan sistem *content based information retrieval* (CBIR) yang telah diintegrasikan dengan Website.
2. Menerapkan aljabar vektor untuk menyelesaikan permasalahan program yang dapat dilihat pada Bab 2.
3. Menghasilkan sistem temu balik gambar yang efektif berdasarkan waktu dan kinerja program.
4. Ekstraksi fitur seperti warna dan tekstur telah diimplementasikan ke dataset yang diunggah melalui website.
5. Mengembangkan Website yang memiliki *interface* (antarmuka) yang menarik
6. Caching result dari perhitungan dataset ke dalam output file .csv untuk menghasilkan algoritma pencarian yang efisien.

II. Saran

1. Program dapat di-improve dengan menerapkan *Usability Heuristics* dalam mendesain *user interface* (antarmuka pengguna) untuk mencapai *user experience* yang lebih seamless.
2. Dapat diimplementasikan *Image Scraping* yang terintegrasi dengan penangkapan gambar secara *real-time*.

III. Komentar atau Tanggapan

Pemanfaatan website untuk implementasi CBIR(*Content Based Image Retrieval*) kami rasa kurang relevan dikarenakan basic atau dasar pengetahuan mengenai pengembangan website sangatlah minim. Hal ini tentunya menyebabkan hal ini sebagai penghambat dan tugas ini menjadi tidak berfokus pada bahasan mata kuliah Aljabar Linier dan Geometri. Menurut kami, untuk tampilan antarmuka pengguna dapat dimudahkan dengan library atau pustaka antarmuka pengguna bawaan tanpa menggunakan web.

IV. Refleksi terhadap Tugas Besar

Kami belajar banyak dalam tugas besar ini. Pengalaman kami selama mengerjakan tugas besar ini cukup memuaskan karena banyak hal baru yang kami pelajari seperti pengembangan website dan juga *library* yang dimiliki suatu bahasa pemrograman. Namun, tekanan dari mata kuliah lain, menyebabkan tugas besar ini terasa lebih berat. Komunikasi sudah optimal sehingga menghasilkan output yang baik pada penyelesaian dan pengembangan program. Dari segi kekurangan, kerja sama kami dapat di-improve sebagai suatu tim.

V. Ruang Perbaikan atau Pengembangan

Menggunakan dynamic import dapat memberikan overhead pada performa karena setiap kali halaman dimuat, komponen gambar akan diimpor secara dinamis. Ini mungkin tidak efisien jika jumlah gambar atau komponen sangat besar. Mungkin pertimbangkan untuk melakukan prefetching atau preloading untuk meningkatkan performa. Penanganan loading untuk memberi tahu pengguna bahwa gambar sedang dimuat, terutama saat menggunakan dynamic import. Tampilan dan fungsi dapat dibuat responsif, terutama jika proyek ini diharapkan dapat berjalan di berbagai perangkat.

Proyek atau tugas besar ini kedepannya dapat menjadi dasar dalam mengembangkan *machine learning*. Pengolahan citra dengan teknik ini dapat dapat dimanfaat menjadi data train untuk model machine learning contohnya KNN, LSTM, Naive Bayes, dan lain-lain.

DAFTAR PUSTAKA

concurrently. (n.d.). npm. <https://www.npmjs.com/package/concurrently>

Eriansyah, A. (2015). Perbandingan Content Based Image Retrieval dengan Fitur Warna Menggunakan Metode Colour Histogram dan Fitur Tekstur Menggunakan Metode Grey Level Co-Occurrence Matrices. [usu.ac.id.](http://repositori.usu.ac.id/handle/123456789/20451) doi: <http://repositori.usu.ac.id/handle/123456789/20451>

FastAPI. (n.d.). FastAPI. <https://fastapi.tiangolo.com/>

Muhammad, Y. (Tanggal publikasi). Feature Extraction: Gray Level Co-Occurrence Matrix (GLCM). Medium. <https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1>

RGB to HSV Color Conversion Algorithm. (n.d.). Mathematics Stack Exchange. [https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm.](https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm) (Accessed Nov. 18, 2023).

React. (n.d.). Documentation. <https://reactjs.org/docs>

Tailwind CSS. (n.d.). Documentation. <https://tailwindcss.com/docs>

Repository GitHub

Link Repository : <https://github.com/alandmprtma/Algeo02-22124>

Penjelasan Algoritma dan Bonus

Link Youtube : <https://youtu.be/qON9X1uillo>