

**LAPORAN TUGAS BESAR 02**

**IF2123 ALJABAR LINEAR DAN GEOMETRI**

**Kelompok 13 (3 MUSKETEERS)**



Disusun oleh:

Denise Felicia Tiowanni (13522013)  
Muhammad Naufal Aulia (13522074)  
Abdullah Mubarak (13522101)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**BANDUNG**  
**2023**

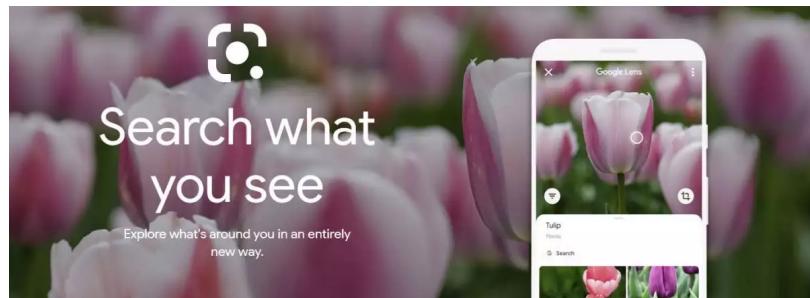
## **DAFTAR ISI**

BAB I DESKRIPSI MASALAH.....	2
BAB II LANDASAN TEORI.....	3
BAB III ANALISIS PEMECAHAN MASALAH.....	9
BAB IV IMPLEMENTASI DAN UJI COBA.....	12
BAB V PENUTUP.....	32
DAFTAR PUSTAKA.....	34

## BAB I

### 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 diketahui adalah Google Lens.



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

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

## **BAB II**

### **TEORI SINGKAT**

#### **2.1 CONTENT-BASED INFORMATION RETRIEVAL (CBIR)**

Content-Based Image Retrieval (CBIR) adalah suatu proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan mengekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur tersebut diekstraksi, fitur-fitur tersebut kemudian direpresentasikan 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 pencocokan ini digunakan untuk menyusun 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 lebih efisien, karena CBIR tidak bergantung pada pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai konten visual antara gambar-gambar tersebut. Pada tugas besar ini, diimplementasikan dua parameter CBIR yang paling populer, yaitu:

- a. CBIR dengan parameter warna

Dalam CBIR ini, perbandingan dilakukan antara input suatu gambar dengan gambar-gambar dari suatu dataset. Proses ini melibatkan transformasi gambar yang awalnya dalam format RGB menjadi metode histogram warna yang lebih umum.

Histogram warna mencatat frekuensi munculnya berbagai warna dalam suatu ruang warna tertentu, dengan tujuan mendistribusikan warna dalam gambar. Meskipun histogram warna tidak mampu mengidentifikasi objek spesifik dalam gambar atau menjelaskan posisi distribusi warna, pembentukan ruang warna diperlukan untuk membagi nilai citra menjadi beberapa rentang kecil. Hal ini bertujuan agar setiap interval rentang dapat dianggap sebagai bin dalam histogram warna.

Perhitungan histogram warna melibatkan penghitungan piksel yang mewakili nilai warna dalam setiap interval. Fitur warna mencakup histogram warna global dan histogram warna blok. Dalam perhitungan histogram, penggunaan warna global

HSV diutamakan karena warna tersebut dapat berlaku pada latar belakang kertas (berwarna putih), yang umumnya digunakan. Oleh karena itu, perlu dilakukan konversi warna dari RGB ke HSV dengan langkah-langkah sebagai berikut.

1. Normalisasi nilai RGB dengan mengubah nilai range [0, 255] menjadi [0, 1].

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

2. Cari  $C_{max}$ ,  $C_{min}$ , dan  $\Delta$ .

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

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

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

3. Gunakan hasil perhitungan poin 2 untuk mendapatkan nilai HSV dengan cara berikut.

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

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

$$V = C_{max}$$

4. Setelah mendapatkan nilai HSV, lakukan perbandingan antara *image* dari input dengan dataset menggunakan persamaan *cosine similarity*.

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

Pada persamaan ini, A dan B adalah vektor dan n adalah jumlah dimensi dari vektor. Tingkat kemiripan antar gambar dilihat dari seberapa besar hasil dari *cosine similarity*.

Pada tugas besar ini, pencarian histogram dilakukan dengan membagi gambar menjadi blok-blok berukuran  $n \times n$ . Hal ini dilakukan karena blok-blok tersebut akan kurang signifikan jika ukurannya terlalu besar, dan akan meningkatkan waktu pemrosesan jika terlalu kecil. Oleh karena itu, untuk pencarian blok yang lebih efektif, digunakan blok berukuran  $4 \times 4$ .

#### b. CBIR dengan parameter tekstur

Dalam CBIR ini, digunakan *co-occurrence matrix* yang memudahkan dan mempercepat proses pemrosesan dengan menghasilkan vektor yang berdimensi lebih kecil. Untuk suatu gambar  $I$  dengan  $n \times m$  piksel dan parameter offset  $(\Delta x, \Delta y)$ , suatu *co-occurrence matrix* dapat dirumuskan sebagai berikut:

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

dengan  $i$  dan  $j$  adalah nilai intensitas dari gambar, serta  $p$  dan  $q$  adalah posisi dalam gambar. Offset  $(\Delta x, \Delta y)$  bergantung pada arah  $\theta$  dan jarak yang diukur melalui persamaan  $\Delta x = p (\cos \theta)$  dan  $\Delta y = p (\sin \theta)$ . Nilai  $\theta$  yang umum digunakan adalah  $0^\circ, 45^\circ, 90^\circ$ , dan  $135^\circ$ .

Setelah didapatkan *co-occurrence matrix*, langkah selanjutnya adalah membuat matriks simetris dengan menjumlahkan *co-occurrence matrix* dengan transpose-nya. Kemudian, dicari matriks normalisasi dengan persamaan tertentu.

Langkah-langkah dalam CBIR dengan parameter tekstur melibatkan beberapa tahap, diantaranya:

1. Konversi warna gambar menjadi *grayscale* dengan rumus:

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

2. Kuantifikasi nilai *grayscale*. Karena citra grayscale berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran  $256 \times 256$ .
3. Ekstraksi komponen tekstur, diantaranya *contrast*, *entropy*, *dissimilarity*, *energy*, *correlation*, dan *homogeneity*. Persamaan untuk mendapatkan nilai ketiga komponen tersebut diberikan oleh:

*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:*

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

*Dissimilarity:*

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

*Energy:*

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

*Correlation:*

$$\sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} \left[ \frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

4. Buat sebuah vektor dari komponen-komponen tekstur yang telah diekstraksi.
5. Lakukan perbandingan antara image dari input dengan dataset menggunakan persamaan cosine similarity.

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

Sama seperti sebelumnya, A dan B adalah vektor dan n adalah jumlah dimensi dari vektor. Tingkat kemiripan antar gambar dilihat dari seberapa besar hasil dari *cosine similarity*.

## 2.2 BAHASA PEMOGRAMAN

### a. Python

Pada Tugas Besar II ini, python digunakan sebagai bahasa pemograman utama dalam implementasi algoritma pemrosesan gambar serta pada logika backend. Python memberikan kemudahan dalam proses manipulasi data seperti ketersediaan pustaka OpenCV, NumPy, ataupun Taichi sebagaimana digunakan dalam Tugas Besar ini.

#### 1. OpenCV (Open Source Computer Vision Library)

OpenCV adalah pustaka sumber terbuka yang menyediakan berbagai fungsi untuk pengolahan gambar dan penglihatan komputer. Dengan dukungan untuk banyak bahasa pemrograman, termasuk Python, OpenCV memungkinkan analisis gambar, deteksi objek, visi komputer, dan aplikasi pengolahan gambar lainnya. Selain itu, OpenCV juga mempermudah deteksi dan pelacakan objek dalam gambar atau video.

#### 2. NumPy

NumPy adalah pustaka Python yang menyediakan struktur data array multidimensi dan fungsi-fungsi matematika untuk melakukan operasi numerik. NumPy menjadi dasar untuk banyak pustaka ilmu data dan pengolahan data lainnya. Pada Tugas Besar ini, NumPy digunakan untuk melakukan operasi matriks serta membantu menghitung tiap-tiap fitur dari ekstraksi tekstur.

#### 3. Taichi

Taichi adalah pustaka komputasi fisika yang dirancang untuk performa tinggi dan mudah digunakan. Dengan fokus pada simulasi fisika, Taichi mendukung komputasi GPU dan CPU, membuatnya ideal untuk pengembangan aplikasi simulasi fisika yang kompleks. Pada Tugas Besar ini, Taichi membantu proses perhitungan dan membuat waktu eksekusi program menjadi lebih singkat.

b. BeautifulSoup4

Beautiful Soup adalah pustaka Python yang digunakan untuk tujuan *web scraping* untuk menarik data dari file HTML dan XML. Beautiful Soup ditunjang oleh parser Python populer seperti lxml dan html5lib. Pada tugas besar ini, BeautifulSoup4 dan html5lib digunakan untuk mengambil data html bertipe image dari web yang kemudian disimpan di folder lokal.

c. react-webcam

React-webcam adalah pustaka React js yang memungkinkan penggunaan kamera device untuk menangkap gambar. Pada tugas besar ini, react-webcam digunakan untuk menangkap gambar dari kamera device setiap 5 detik yang kemudian dikirim ke backend.

d. Flask

Flask digunakan sebagai framework backend untuk membangun API dan mengelola logika server. Dengan kemampuannya yang ringan dan fleksibel, Flask memfasilitasi pengembangan sistem dengan menangani permintaan HTTP, seperti pencarian gambar dan pengolahan data.

e. React

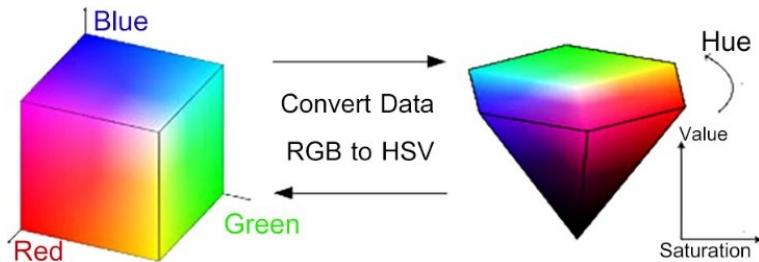
React digunakan sebagai library JavaScript untuk mengembangkan antarmuka pengguna (frontend). Dengan konsep komponen yang dapat digunakan ulang dan kemampuan pembaruan tanpa perlu me-refresh halaman, React memastikan antarmuka pengguna responsif dan efisien. Pada Tugas Besar ini, digunakan pula suatu framework bernama Tailwind CSS yang dihubungkan dengan React guna membantu proses *styling*.

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### 3.1 ANALISIS GAMBAR DENGAN WARNA

Analisis gambar dengan warna diawali dengan proses konversi warna gambar dari format RGB ke HSV. Proses ini melibatkan normalisasi nilai RGB, pencarian nilai Cmax, Cmin,  $\Delta$ , serta konversi ke ruang warna HSV. Hal ini dilakukan karena ruang warna HSV lebih memanfaatkan mekanisme persepsi visual manusia.



**Gambar 2.** Ruang RGB dan Ruang HSV

Selanjutnya, pembentukan histogram warna dilakukan dengan menghitung frekuensi munculnya berbagai warna dalam gambar. Dalam hal ini, digunakan histogram warna blok dimana pencarian dimulai dengan membagi gambar menjadi blok-blok berukuran 4x4. Hal ini dipilih untuk menjaga signifikansi blok serta meningkatkan efisiensi pemrosesan.

Tahap yang terakhir adalah membandingkan gambar dari input dengan dataset menggunakan cosine similarity. Besar cosine similarity menentukan tingkat kemiripan antar gambar.

#### 3.2 ANALISIS GAMBAR DENGAN TEKSTUR

Berbeda dengan warna, analisis gambar dengan tekstur dimulai dengan mengkonversi warna gambar ke skala *grayscale* atau abu-abu. Hal ini dikarenakan pada pemrosesan tekstur, analisis warna tidaklah diperlukan. Dengan mengubah gambar ke

skala abu-abu, ini akan mempercepat waktu eksekusi program dengan mengeliminasi bagian yang tidak diperlukan.



Gambar 3. Konversi RGB menjadi *Grayscale*

Setelah gambar telah berhasil dikonversi ke skala abu-abu, akan dilakukan kuantifikasi nilai *grayscale* dengan membangun matriks 256x256 yang berkorespondensi dengan citra *grayscale* yaitu sebanyak 256 piksel. Selanjutnya barulah dilakukan ekstraksi komponen tekstur seperti *contrast*, *homogeneity*, *dissimilarity*, *energy*, *correlation*, maupun *entropy* yang kemudian dibentuk menjadi suatu vektor.

Terakhir, dilakukan perbandingan gambar antara input dengan dataset menggunakan *cosine similarity*.

### 3.3 ILUSTRASI KASUS DAN PENYELESAIANNYA

Misalkan kita memiliki suatu dataset gambar medis yang terdiri dari citra hasil pemindaian X-ray. Tujuan kita adalah membuat sistem temu balik gambar (CBIR) untuk membantu dokter atau peneliti menemukan gambar-gambar dengan kondisi serupa berdasarkan konten visual, seperti warna atau tekstur.

#### 1. Analisis Gambar dengan Warna

Gambar hasil pemindaian X-ray masih dalam format RGB di konversi menjadi HSV. Selanjutnya dibentuk histogram warna blok dari gambar X-ray untuk menghitung frekuensi munculnya warna dalam gambar. Terakhir, dilakukan perbandingan menggunakan *cosine similarity*. Dengan ini, sistem akan membantu mengenali gambar dengan kondisi serupa dan menampilkan hasil dengan benar.

#### 2. Analisis Gambar dengan Tekstur

Gambar hasil pemindaian X-ray dalam format RGB dikonversi menjadi *grayscale* untuk kemudian diekstraksi komponen teksturnya. Dilakukan pula perbandingan *cosine similarity*.

Dalam kasus ini, CBIR akan membantu dokter memahami gambar-gambar dengan kondisi serupa untuk memahami perkembangan penyakit dan memilih terapi yang paling sesuai. Selain itu, dokter dan peneliti pun dapat mencari kasus serupa dalam database hanya dengan memasukkan satu buah foto referensi sebagai bandingan.

## BAB IV

### IMPLEMENTASI DAN UJI COBA

#### 4.1 IMPLEMENTASI

##### 4.1.1 CBIR Texture

<b>Nama Fungsi (parameter)</b>	<b>Deskripsi</b>
toGrayscale(image)	Mengkonversi gambar RGB menjadi gambar dengan citra <i>grayscale</i>
matrixGLCM(image, d=1)	Membuat <i>framework</i> matriks (matriks <i>co-occurrence</i> ), dimana pada kasus ini sudut yang digunakan adalah $0^\circ$ dan <i>distance</i> sama dengan 0
symmetricGLCM(glcm_matrix)	Mengkonversi matriks menjadi matriks simetri
normalizeGLCM(glcm_matrix)	Menormalisasi matriks
extract_texture(glcm_matrix)	Melakukan ekstraksi tekstur seperti <i>contrast</i> , <i>homogeneity</i> , <i>entropy</i> , <i>dissimilarity</i> , <i>energy</i> , maupun <i>correlation</i>
createVector(a, b, c, d, e, f)	Membuat vektor dari komponen-komponen ekstraksi tekstur
CBIR_tekstur(image)	Menggabungkan semua fungsi CBIR tekstur dan mengembalikan vektor komponen-komponen ekstraksi tekstur

##### 4.1.2 CBIR Color

<b>Nama Fungsi (parameter)</b>	<b>Deskripsi</b>
warna_csv()	Menghapus/mereset file warna.csv saat mengupload dataset baru
rgb_to_index(r, g, b)	Mengubah RGB menjadi HSV, lalu memformulasikan HSV menjadi index
rgb_to_histogram(gambar1, hist)	Mengubah RGB menjadi index dengan fungsi <i>rgb_to_index</i> lalu digunakan untuk

	menghasilkan histogram dengan panjang 72 yang mewakili gambar1
fitur()	Membandingkan <i>cosine similarity</i> gambar <i>query</i> dengan dataset

#### 4.1.3 Backend

##### a. finder.py

Nama Fungsi (parameter)	Deskripsi
cosine_sim(vector1, vector2)	Mengembalikan hasil <i>cosine similarity</i> antara dua vektor masukan.
find(inputFeature, option)	Membaca vektor ekstraksi fitur gambar dan mengembalikan <i>dictionary</i> nama gambar serta nilai hasil kemiripannya dari <i>cosine similarity</i>

##### b. init.py

File untuk menginisiasi ekstraksi awal dataset, dengan memanggil fungsi-fungsi pada CBIR Texture & Color pada tabel di atas.

##### c. tekstur.py

Berisikan semua fungsi kebutuhan CBIR Texture seperti tertera pada tabel di atas

##### d. warna.py

Berisikan semua fungsi kebutuhan CBIR Color seperti tertera pada tabel di atas

##### e. warna\_individual.py

Implikasi dari penggunaan Taichi, tidak bisa melakukan *multithreading*, oleh karena itu menjalankan fungsi yang mengandung Taichi harus dilakukan pada masing-masing file .py-nya sendiri sehingga untuk kebutuhan ekstraksi fitur warna gambar *query* ditempatkan pada file warna\_individual.py

##### f. app.py

Nama Route	Nama Fungsi	Deskripsi
/dataset	upload()	Endpoint ketika user melakukan upload dataset, menjalankan fungsi upload (ekstraksi fitur gambar pada

		dataset), mengembalikan pesan ekstraksi selesai.
/search	search()	Endpoint ketika user melakukan search gambar <i>query</i> , memanggil fungsi search (membandingkan vektor fitur gambar, menyimpan sementara gambar yang akan dikembalikan), mengembalikan waktu durasi proses ke route /durasi melalui file csv*
/durasi	durasi()	Endpoint lanjutan dari route /search, memanggil fungsi durasi (membaca waktu durasi dari file csv), mengembalikan waktu durasi perbandingan.
/retrieve-images	retrieve_images()	Endpoint lanjutan dari route /search, mengembalikan <i>similar images</i> yang sudah terurut ke <i>frontend</i> .
/img/<path:path>	send_report()	Endpoint untuk mengembalikan gambar direktori lokal ke <i>frontend</i> melalui <i>backend</i> .
/generate-pdf	generate_pdf(), get_next_pdf_filename()	Untuk bonus <i>export PDF</i> , mengambil <i>similar images</i> hasil pencarian dan menuliskan di luaran PDF ke direktori lokal.
/scrape	scrape()	Untuk bonus <i>image scraping</i> , mengambil gambar sebagai dataset dari <i>url</i> dengan bantuan <i>BeautifulSoup4</i> , dan melakukan ekstraksi.
/camera	capture()	Untuk bonus kamera, endpoint yang menerima gambar <i>query</i> dari kamera

		melalui <i>frontend</i> .
--	--	---------------------------

#### 4.1.4 Frontend

##### a. App.js

‘App.js’ adalah file utama dalam tampilan pengguna yang menggunakan React Router untuk menangani navigasi. Terdapat beberapa komponen utama dalam file ini, diantaranya Header, Search, Dataset, Images, Dora-Button-Page, serta Nobi-Button-Page.

##### b. index.js

‘index.js’ menyiapkan aplikasi React dengan mengimpor modul, membuat elemen root, serta merender komponen utama App ke dalam elemen root yang telah ditentukan.

##### c. dataset.js

Pada ‘dataset.js’, diimpor modul ‘styles.css’ untuk mengatur gaya dan tampilan yang diperlukan untuk komponen, serta modul ‘axios’ untuk melakukan permintaan HTTP. File ini mencakup pengunggahan dataset yang kemudian di-pass ke backend server.

##### d. header.js

File ‘header.js’ mengatur tampilan *header* dari website. File ini mencakup beberapa komponen yang terdapat pada *header*, diantaranya *button* menuju page About Us serta How to Use.

##### e. nobi-button.js

File ‘nobi-button.js’ mengatur tampilan *button* Nobita atau *button* yang mengantar pengguna pada halaman How to Use.

##### f. nobi-button-page.js

File ‘nobi-button-page.js’ mengatur isi atau konten dari halaman How to Use.

##### g. dora-button.js

File ‘dora-button.js’ mengatur tampilan *button* Doraemon atau *button* yang mengantar pengguna pada halaman About Us.

##### h. dora-button-page.js

File ‘dora-button-page.js’ mengatur isi atau konten dari halaman About Us.

**i. toggleoptions.js**

File ‘toggleoptions.js’ mengatur tampilan *toggle options*, yaitu tombol Texture dan Color. File ini juga menangani kasus ketika tombol Texture ataupun Color tersebut di klik.

**j. search.js**

File ‘search.js’ mengatur pencarian pengguna. File ini menerima input berupa *query image* dari pengguna, serta menangani kasus ketika pengguna menekan tombol Search tanpa memilih opsi Texture atau Color, serta ketika pengguna menekan Search tanpa mengunggah *query image* sama sekali.

**k. images.js**

File ‘images.js’ menerima dan menampilkan hasil pencarian gambar, baik itu Texture ataupun Color, kepada pengguna. Pada file ini pula, terdapat opsi Try Again ataupun Download Result in PDF bagi pengguna untuk melakukan kembali pencarian dan mengunduh hasil pencarian dalam bentuk PDF.

**l. scraping.js**

File ‘scraping.js’ menerima url website dari pengguna lalu mengirimnya ke backend.

**m. camera.js**

File ‘camera.js’ mengambil gambar dari kamera device setiap 5 detik, lalu menyimpan gambar ke folder uploaded dengan format jpg.

**n. styles.css**

File ‘styles.css’ menampung seluruh *design* dari komponen-komponen file lainnya. Secara keseluruhan, semua file frontend mengimpor modul ‘styles.css’ guna mengatur layout dari setiap komponennya.

## **4.2 PENJELASAN STRUKTUR PROGRAM**

Struktur program Reverse Image Search kami mencakup beberapa direktori dan file, yang dapat dijabarkan sebagai berikut.

| README.md

|

|——doc

```
|  
|   └── img  
|       └── pdf  
|  
|   └── src  
|       ├── Backend  
|       |   └── durasi  
|       |       └── durasi.csv  
|       |   └── fitur  
|       |       └── tekstur.csv  
|       |       └── warna.csv  
|       |   └── img  
|       |       └── dataset  
|       |           └── retrieved (temp)  
|       |           └── uploaded (temp)  
|       |   ├── app.py  
|       |   ├── finder.py  
|       |   ├── init.py  
|       |   ├── tekstur.py  
|       |   ├── warna_individual.py  
|       |   └── warna.py  
|  
|   └── Website  
|       └── src  
|           └── components  
|               ├── dataset.js  
|               ├── dora-button-page.js  
|               ├── dora-button.js  
|               ├── header.js  
|               ├── images.js  
|               └── kamera.js
```

```

|   nobi-button-page.js
|   nobi-button.js
|   scraping.js
|   search.js
|   style.css
|   toggleoptions.js
|
|   App.js
|   index.js
|
└── test
    ├── dataset
    └── query

```

dimana

- a. README.md berisi dokumentasi dasar yang memberikan informasi tentang cara menggunakan, menginstal, serta memahami program.
- b. doc berisi file laporan.
- c. img berisi *screenshot* hasil ujicoba program yang ditampilkan serta image yang digunakan oleh README.
- d. src berisi *source code* pembangun program secara keseluruhan, terdiri dari backend dan frontend. Pada website kami, digunakan *framework* Flask untuk *backend* dan React untuk *frontend*
- e. Di dalam src, terdapat backend yang berisi folder penyimpanan hasil ekstraksi fitur, folder img untuk dataset dan keperluan gambar, serta file-file algoritma utama. Hasil ekstraksi fitur dari gambar dataset disimpan dalam folder berisi file .csv (tekstur.csv dan warna.csv). Hal ini dilakukan sebagai bentuk *caching* yang mempermudah pengambilan vektor hasil ekstraksi ketika dibutuhkan dalam pencarian, tanpa harus selalu melakukan proses ekstraksi berulang.
- f. Di dalam src, terdapat frontend yang berisi kebutuhan struktur badan website beserta semua komponen yang membangunnya.
- g. test berisi folder untuk pengujian, baik dataset pengujian maupun gambar *query* yang mau dicari

### **4.3 TATA CARA PENGGUNAAN PROGRAM**

- a. Upload dataset

Pengguna dapat mengunggah dataset gambar melalui *upload* folder berisi kumpulan gambar. Dataset ini menjadi sumber ekstraksi fitur dan pengindeksan hasil gambar.

- b. Image scraping dari URL

Selain *upload* folder dataset dari direktori lokal, pengguna juga dapat memasukkan URL website untuk mengambil gambar-gambar yang terdapat di dalamnya sebagai dataset.

- c. Insert image

Pengguna dapat memasukkan gambar yang hendak dicari.

- d. Capture image

Fitur *capture* memungkinkan pengguna memasukkan gambar yang hendak dicari melalui kamera perangkat pengguna sendiri.

- e. Search by texture

Fitur yang memungkinkan proses pencarian gambar dari gambar masukkan menggunakan analisis fitur tekstur, kemudian menampilkan kembali hasil *similar image* ke pengguna.

- f. Search by color

Fitur yang memungkinkan proses pencarian gambar dari gambar masukkan menggunakan analisis fitur warna, kemudian menampilkan kembali hasil *similar image* ke pengguna.

- g. Export PDF

Memungkinkan pengguna mengunduh hasil *similar image* yang ditampilkan website ke dalam direktori lokal perangkat pengguna dalam bentuk file PDF.

- h. How to use

Page tambahan berisi penjelasan singkat tentang website, konsep dasar, dan cara penggunaannya secara umum.

- i. About us

Page tambahan berisi identitas kelompok perancang.

## 4.3 UJI COBA PROGRAM

### 4.3.1 Upload Dataset

**Upload dataset ketika belum memilih folder dataset**

Please choose a file before uploading

Insert an Image    Search

or

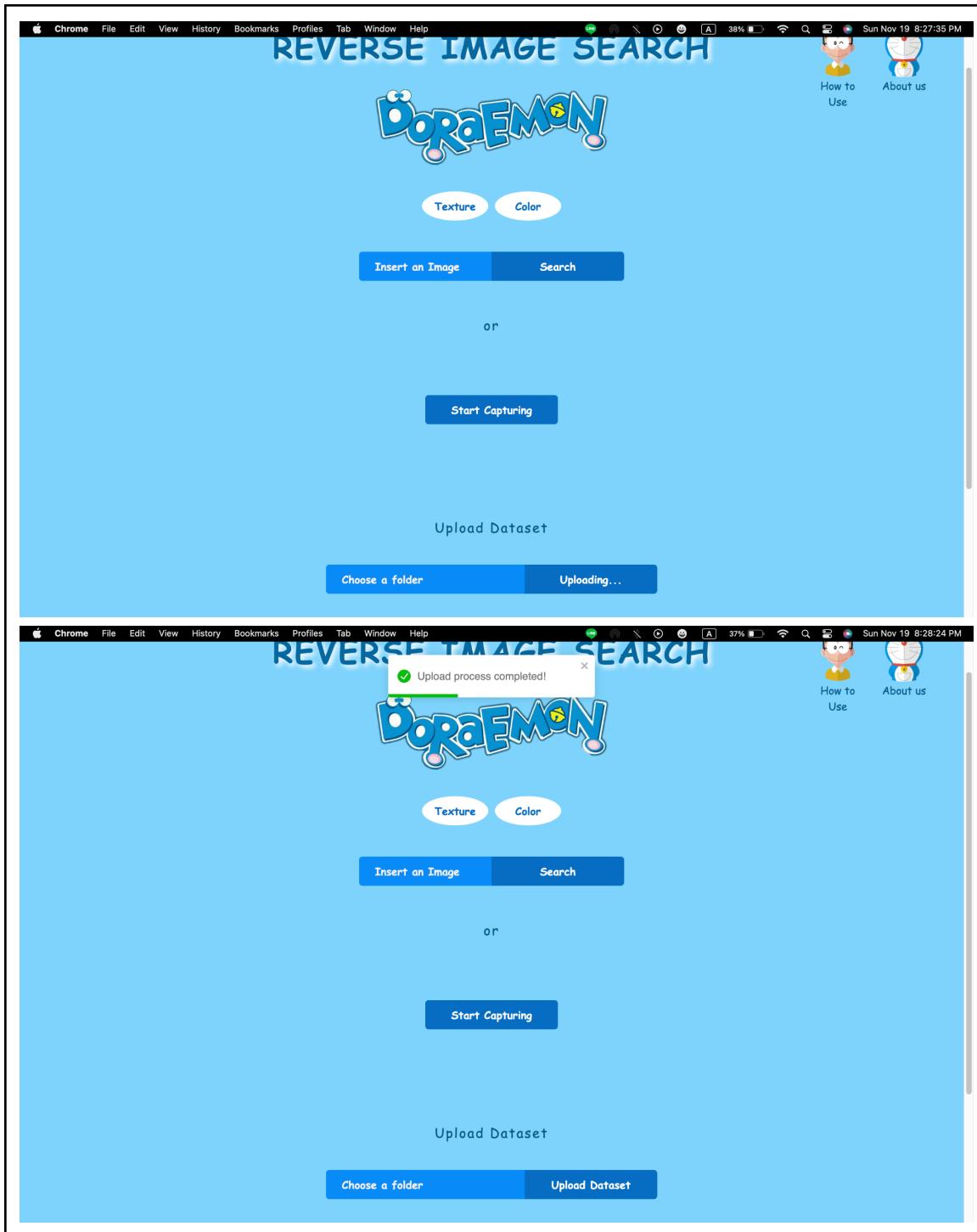
Start Capturing

Upload Dataset

Choose a folder    Upload Dataset

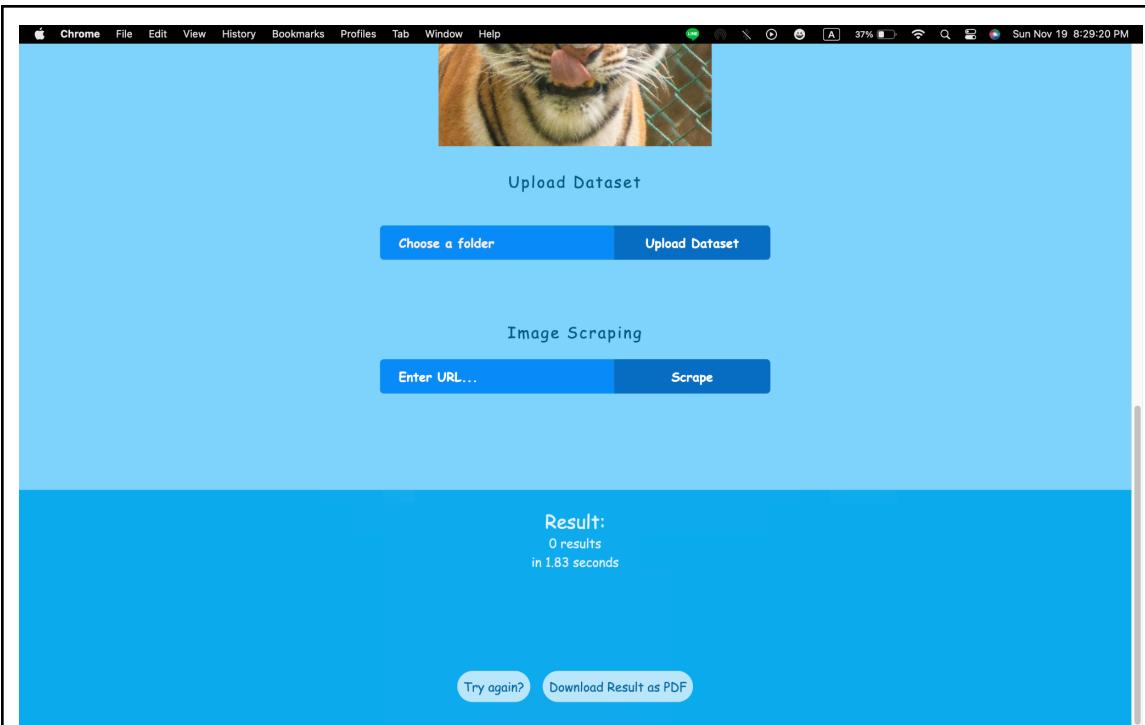
Image Scraping

**Upload dataset ketika sudah memilih folder dataset**

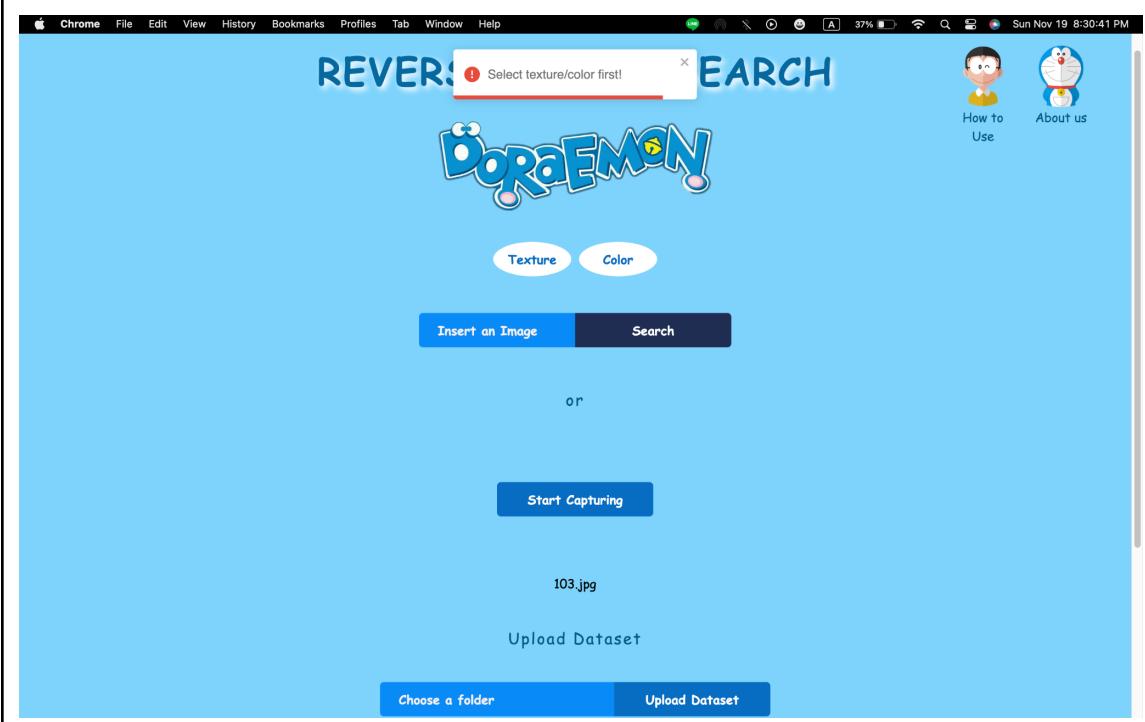


#### 4.3.2 Pencarian Gambar

Pencarian gambar ketika dataset masih kosong



### Pencarian gambar ketika dataset sudah ada, namun tidak menekan tombol Texture/Color



### Pencarian gambar ketika dataset sudah ada, sudah menekan tombol Texture/Color, namun belum mengunggah *query image*

**REVERSE SEARCH**

Please upload a file before searching!



Texture Color

Insert an Image Search

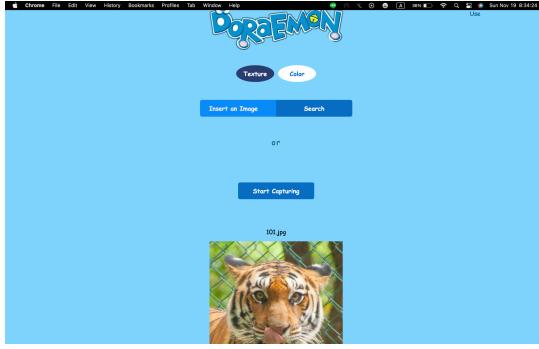
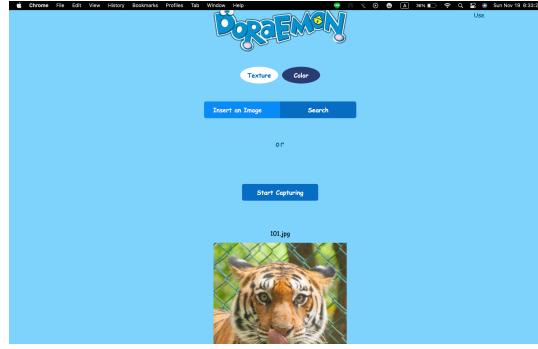
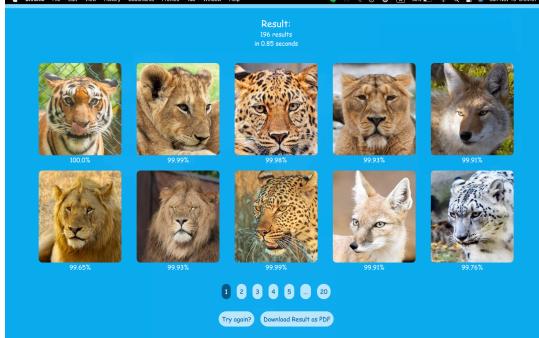
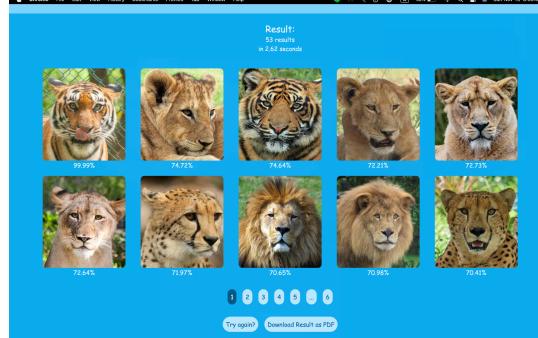
or

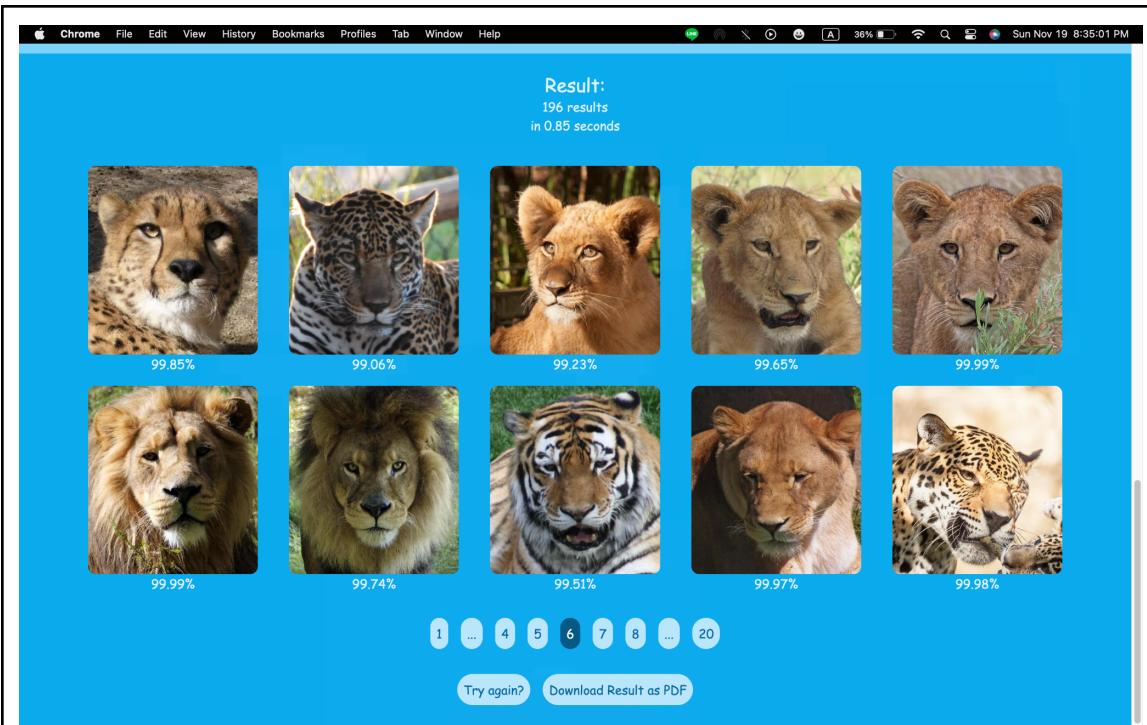
Start Capturing

Upload Dataset

Choose a folder Upload Dataset

### Pencarian gambar ketika dataset sudah ada

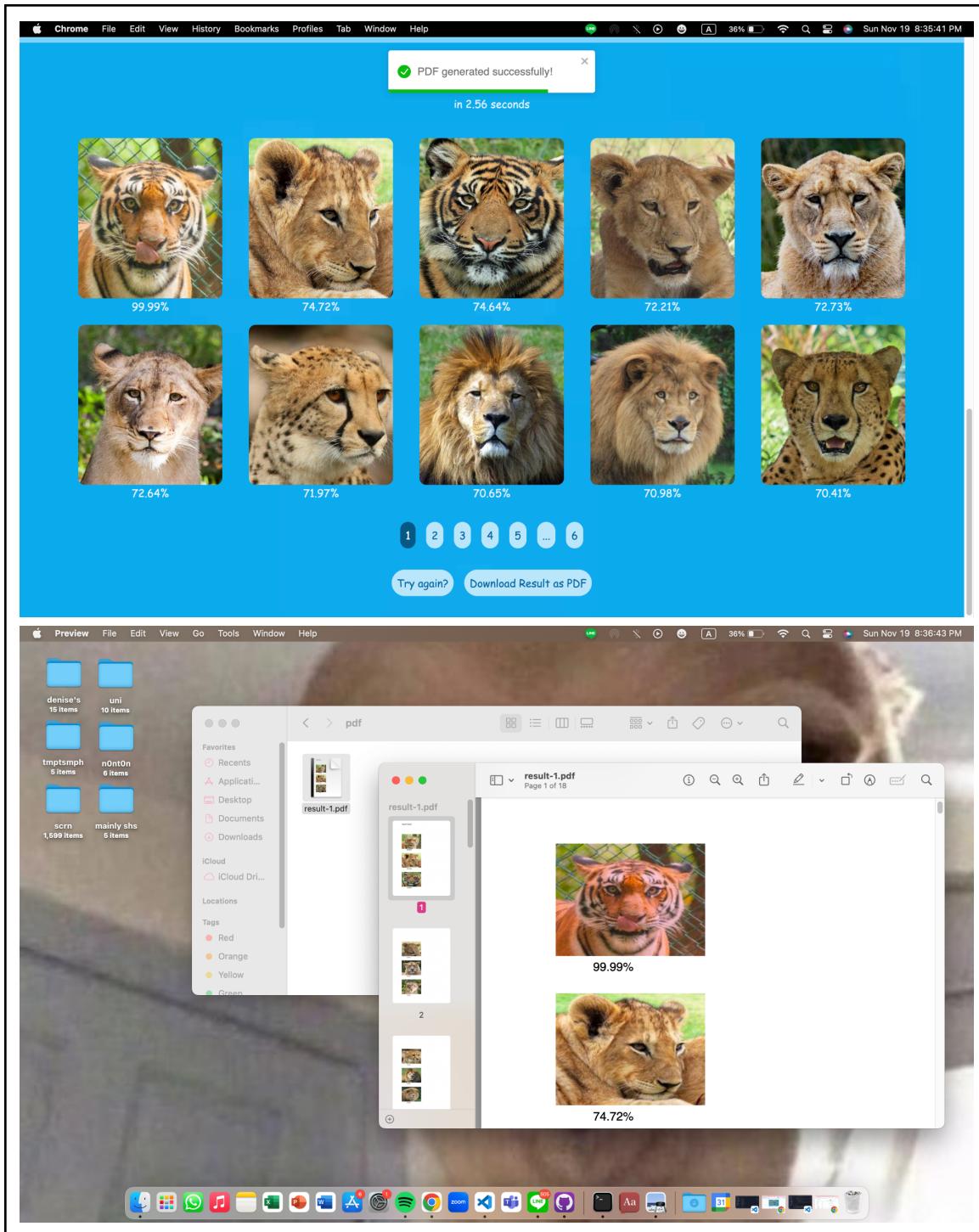
Texture	Color
	
	
<b>Pagination</b>	



Try Again (kembali ke *home page*)

A screenshot of the Doraemon Reverse Image Search homepage. The title bar indicates it's running on a Mac (Chrome, File, Edit, View, History, Bookmarks, Profiles, Tab, Window, Help) with a battery level of 36% and the date Sun Nov 19 8:35:16 PM. The main header is "REVERSE IMAGE SEARCH" with the Doraemon logo below it. There are two buttons for "Texture" and "Color". Below the logo are two buttons: "Insert an Image" and "Search". A "How to Use" link and an "About us" link are located on the right side. In the center, there's a "Start Capturing" button and a "Upload Dataset" button at the bottom. A progress bar is visible at the bottom of the page.

Download Result as PDF



### 4.3.3 Kamera

Start/End Capturing

Chrome File Edit View History Bookmarks Profiles Tab Window Help Sun Nov 19 8:37:19 PM

How to Use About us

# DoraEMON!

Texture Color

Insert an Image Search

African Lion Stop Capturing

Upload Dataset

Choose a folder Upload Dataset

Chrome File Edit View History Bookmarks Profiles Tab Window Help Sun Nov 19 8:37:28 PM

How to Use About us

# DoraEMON!

Texture Color

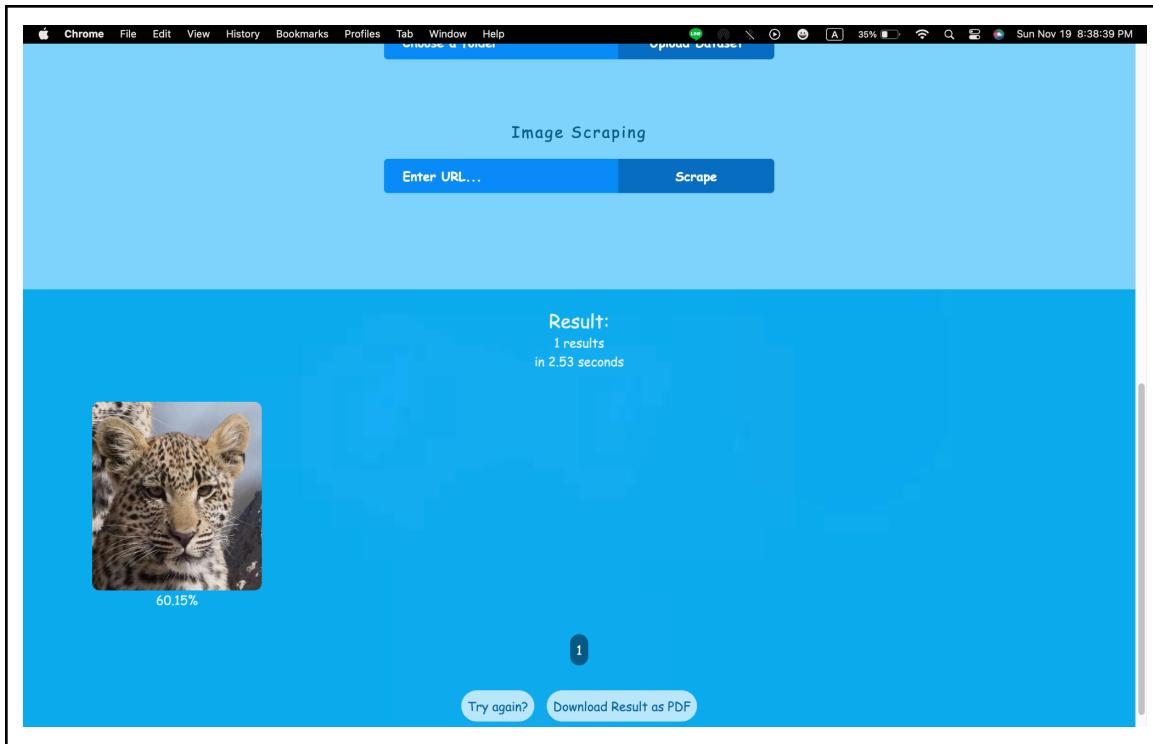
Insert an Image Search

African Lion Start Capturing

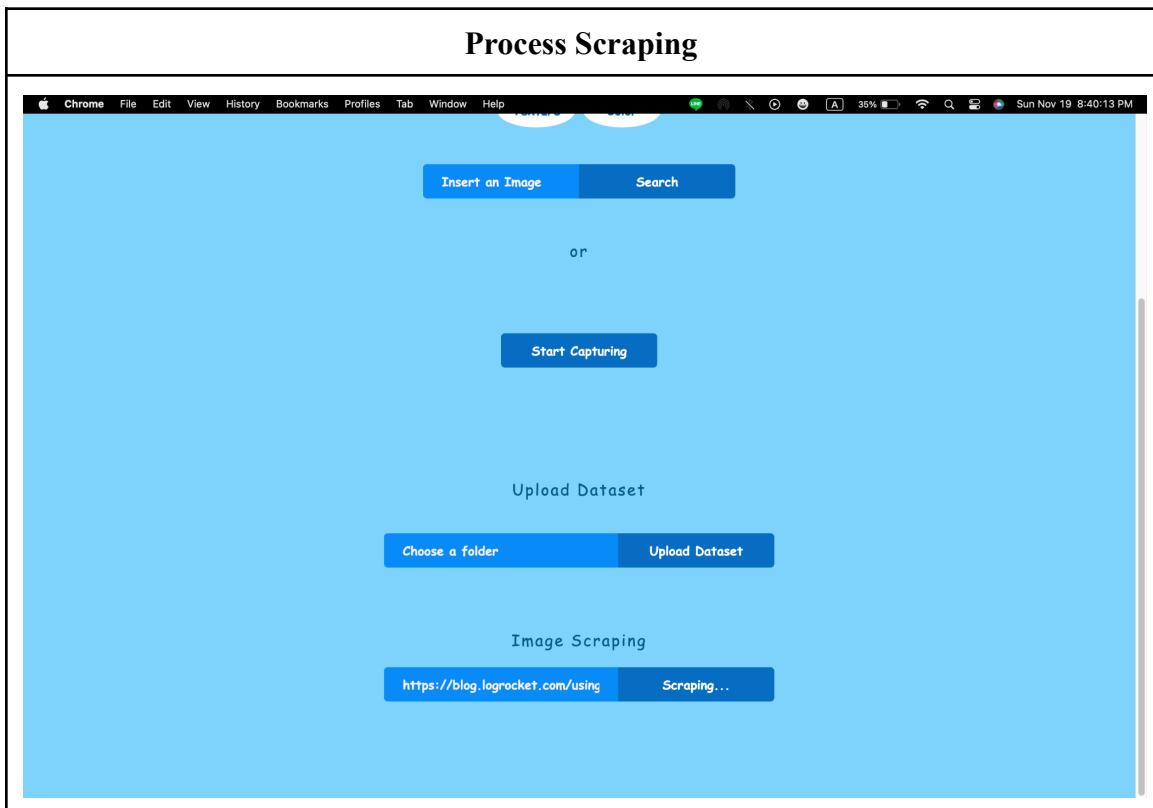
Upload Dataset

Choose a folder Upload Dataset

Searching Result



#### 4.3.4 Image Scraping



## Scraping Berhasil (gambar berhasil masuk ke dataset)

Scraping process completed!

Insert an Image Search

Start Capturing

Upload Dataset

Choose a folder Upload Dataset

Image Scraping

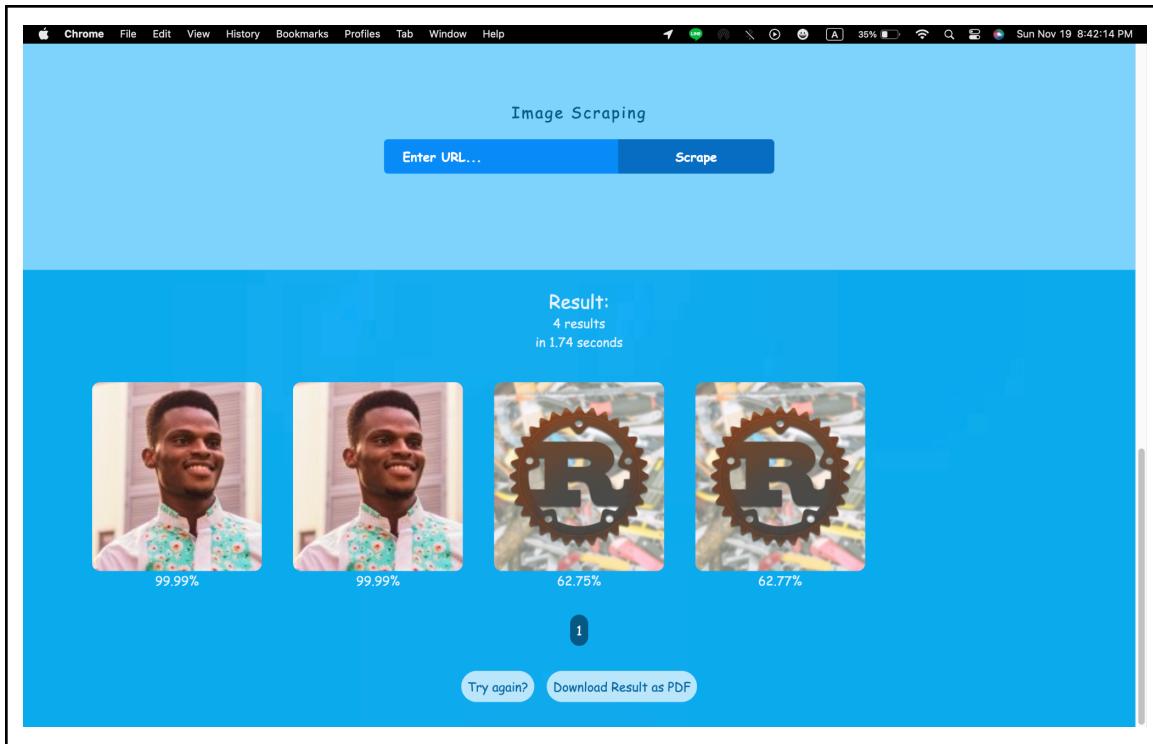
Enter URL... Scrape

ALGEO... .vscode doc img dataset 0.jpg 1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg 9.jpg 10.jpg 11.jpg

File "/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/taichi/lang/kernel\_impl.py", line 976, in wrapped raise type(e)("N" + str(e)) from None taichi.lang.exception.TaichiRuntimeTypeError: Invalid argument into ti.types.ndarray(), got None /bin/sh: python: command not found Ekstraksi selesai! 127.0.0.1 - [19/Nov/2023 20:40:19] "POST /scrape HTTP/1.1" 200 -

Whole Image 150x150 6.05KB Background

### Searching Result



#### 4.3.5 Page Tambahan

A screenshot of the "About Us" page of the website. The page has a light blue header with the title "About Us". Below the header is a navigation bar with "Home" and "How to Use" buttons. The main content area has a light blue background and features the title "ABOUT 3 MUSKETEERS" in large, bold, white letters. Below the title is a paragraph of text: "3 Musketeers adalah sebuah kelompok Tugas Besar II Aljabar Linear dan Geometri (IF2123) yang terdiri dari Denise Felicia Tiowanni (NIM 13522013), Muhammad Naufal Aulia (NIM 13522074), dan Abdullah Mubarak (NIM 13522101). Projek untuk Tugas Besar II IF2123 kali ini adalah 'Website Sistem Temu Balik Gambar' yang diimplementasikan menggunakan pendekatan klasifikasi berbasis konten dengan aljabar vektor." To the right of the text is a large, white Doraemon character with a red collar and bell, standing with one hand raised. At the bottom of the page is a white footer bar with the text "Overview dan Cara Kerja".

Home About us

## ABOUT THIS WEBSITE

Website ini adalah sebuah web sistem temu balik gambar yang diimplementasikan menggunakan pendekatan klasifikasi berbasis konten dengan aljabar vektor. 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.

- Warna:  
Ekstraksi fitur warna (RGB) → Ubah ke HSV → Kuantifikasi ke histogram → Bandingkan histogram dua gambar dengan cosine similarity.
- Tekstur:  
Ekstraksi fitur warna (RGB) → Ubah ke grayscale → Co-occurrence matrix → Ekstraksi tekstur → Bandingkan vektor tekstur dengan cosine similarity.

## CARA PENGGUNAAN

1. Pengguna terlebih dahulu memasukkan dataset gambar dalam bentuk folder 'dataset' yang berisi kumpulan gambar. Dataset gambar ini diperlukan sebelum proses searching agar terdapat pembanding untuk gambar yang ingin dicari.
2. Setelah dataset sudah dimasukkan, pengguna memasukkan sebuah gambar yang ingin di-search dari dataset.
3. Pilih opsi pencarian, ingin melakukan pencarian berdasarkan warna atau tekstur.
4. Tekan tombol search, program kemudian akan memproses, mencari gambar-gambar dari dataset yang memiliki kemiripan dengan gambar yang dimasukkan tadi.
5. Program akan menampilkan kumpulan gambar yang mirip, diurutkan dari yang memiliki kemiripan paling tinggi ke yang paling rendah. Setiap gambar yang muncul diberi persentase kemiripannya.
6. Terdapat informasi terkait jumlah gambar yang muncul, dan waktu eksekusi programnya.



#### 4.4 ANALISIS

Menurut kami, secara umum pembangunan CBIR berdasarkan fitur warna lebih unggul dibandingkan dengan fitur tekstur. Hal ini dapat terjadi karena fitur warna tetap dapat berfungsi meskipun kualitas gambar kurang baik. Dalam situasi dimana kamera mungkin gagal menangkap tekstur dengan jelas, warna tetap dapat menjadi indikator yang dapat berfungsi karena cenderung selalu tampak pada berbagai kondisi pencahayaan dan kualitas gambar.

Pertimbangan lainnya adalah aspek kepraktisan implementasi. Ekstraksi fitur warna dianggap lebih sederhana dan memerlukan sumber daya komputasi yang lebih rendah dibandingkan dengan ekstraksi fitur tekstur yang memerlukan pendekatan yang lebih kompleks, contohnya seperti untuk menghitung tiap-tiap *contrast*, *entropy*, *dissimilarity*, *energy*, *correlation*, ataupun *homogeneity* dari suatu gambar. Selain itu, pembangunan *co-occurrence matrix* pada CBIR tekstur juga memerlukan operasi matematis yang kompleks.

Namun, preferensi terhadap keunggulan CBIR berbasis warna atau tekstur dapat bervariasi sesuai dengan kepentingan individu. Pada beberapa situasi, pengguna yang memiliki minat lebih pada informasi warna, seperti mereka yang bekerja pada bidang desain ataupun industri mode, pasti akan memprioritaskan warna sebagai faktor penentu yang lebih penting. Sebaliknya, dalam konteks pengguna yang lebih mempertimbangkan tekstur sebagai elemen penting, seperti dalam bidang pengolahan citra medis dimana dilakukan identifikasi tekstur kulit atau struktur organ, CBIR tekstur mungkin menjadi pilihan yang lebih unggul.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Melalui pembelajaran pada kuliah Aljabar Linier dan Geometri IF2123, kami berhasil mengimplementasikan materi pada suatu sistem temu balik gambar (CBIR) dengan memanfaatkan Aljabar Vektor. Dua parameter CBIR yang diimplementasikan adalah CBIR dengan parameter warna dan CBIR dengan parameter tekstur.

Analisis gambar dengan parameter warna melibatkan konversi warna dari RGB ke HSV, pembentukan histogram warna blok, dan perbandingan menggunakan *cosine similarity*. Sebaliknya, analisis gambar dengan tekstur dimulai dengan konversi ke skala abu-abu (*grayscale*), kuantifikasi nilai *grayscale*, ekstraksi komponen tekstur, dan perbandingan menggunakan *cosine similarity*.

Dalam implementasi *website*, digunakan bahasa pemrograman Python dengan bantuan pustaka seperti OpenCV, NumPy, Taichi, BeautifulSoup4, Flask, React, dan react-webcam. Fungsi-fungsi yang dibuat mencakup proses konversi warna, pembentukan matriks *co-occurrence*, normalisasi, ekstraksi tekstur, dan perbandingan *cosine similarity*, serta beberapa fitur tambahan : *website scraping* dan *auto-capture camera*.

#### **5.2 Saran**

Kami memiliki beberapa saran dalam penggeraan proyek yang sejenis dengan tugas besar ini:

- a. Sebaiknya membuat tugas besar lebih berkaitan dengan materi perkuliahan serta *timeline* penggeraan yang jelas.
- b. Memanfaatkan A.I., seperti chatGPT, bard, sebaik mungkin untuk membantu membuat kode.
- c. Meningkatkan kecepatan kode dengan *multi processing* dengan fungsi Pool di python, atau dengan memanfaatkan library seperti Numpy, Taichi, Numba, dan Mypy.

### **5.3 Refleksi**

Tugas besar ini telah memberikan pengalaman yang berharga bagi kami atas situasi dan tantangan yang muncul. Salah satu kendala yang kami hadapi adalah waktu penggerjaan yang kurang efektif dan efisien. Ditambah kemampuan adaptasi, khusus dalam pengembangan *website*, yang masih belum cukup baik. Terlepas dari itu, pembagian tugas yang merata dan kerja sama yang baik dari kami menjadi kunci keberhasilan. Pada akhirnya, kemampuan *problem solving*, *computational thinking*, dekomposisi masalah, komunikasi, serta kerjasama kami dapat meningkat. Selain itu, masing-masing dari kami juga bisa mengetahui lebih dalam mengenai pemrograman, khususnya dalam bidang *website development*.

### **5.4 Ruang Perbaikan atau Pengembangan**

Program yang kami buat masih jauh dari kata sempurna. Untuk kedepannya, perlu dilakukan pengembangan optimasi kinerja sistem, terutama dalam hal waktu eksekusi program, agar sistem dapat memberikan respons yang lebih cepat namun tetap akurat. Selain itu, perlu juga dilakukan penyempurnaan antarmuka pengguna (frontend) untuk meningkatkan kegunaan dan daya tarik visual sistem. Terakhir, perlu dilakukan uji coba lebih lanjut untuk memastikan keandalan sistem dalam berbagai kasus penggunaan guna memperbaiki *bug* atau kesalahan yang mungkin terjadi.

### **5.5 Link Repository**

Link Github *repository* Tugas Besar 2 Mata Kuliah IF2123 Aljabar Linier dan Geometri Kelompok 13 (3 Musketeers) dapat diakses pada tautan berikut:  
<https://github.com/NopalAul/Algeo02-22013>.

### **5.6 Link Video**

Link video demo Tugas Besar 2 Mata Kuliah IF2123 Aljabar Linier dan Geometri Kelompok 13 (3 Musketeers) dapat diakses pada tautan berikut:  
<https://youtu.be/kTaZdwUF400?si=VW5VN6nE9lOqFUDk>.

## **DAFTAR PUSTAKA**

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/algeo23-24.htm>

<https://docs.google.com/document/d/1HVDyywnUdNz9hStgx5ZLqHypK89hWH8qfERJOiDw6KA/edit>

<https://stackoverflow.com/questions/34660385/how-to-position-a-react-component-relative-to-its-parent#:~:text=To%20position%20a%20React%20component%20relative%20to%20its%20parent%2C%20you,an%20specify%20the%20positioning%20properties.&text=In%20this%20CSS%20code%2C%20we,set%20its%20position%20to%20absolute.>

<https://stackoverflow.com/questions/46498041/is-the-opencv-3d-histogram-3-axis-histogram>

<https://www.sciencedirect.com/science/article/pii/S0895717710005352>

[https://www.ripublication.com/ijaer17/ijaerv12n24\\_271.pdf](https://www.ripublication.com/ijaer17/ijaerv12n24_271.pdf)