

Tugas Besar II IF2123 Aljabar Linier dan Geometri

Semester I Tahun 2023/2024

Laporan Tugas Besar II

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



Oleh

Maulana Muhammad Susetyo	13522127
Ahmad Rafi Maliki	13522137
Andi Marihot Sitorus	13522138

Kelompok ZilongHyper

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

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.

Di dalam Tugas Besar 2 ini, Kami 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.

Bab 2

Landasan Teori

1. Teori Dasar

1.1. CONTENT-BASED INFORMATION RETRIEVAL (CBIR)

Content-Based Information Retrieval (Temu-Balik informasi Berbasis Konten) adalah sebuah metode untuk mencari gambar yang serupa dari sebuah basis data dengan tujuan menutupi basis gambar dengan kemiripan terhadap kueri gambar.

1.2. CBIR DENGAN PARAMETER WARNA

CBIR dengan parameter warna memetakan warna dari gambar ke histogram warna yang membentuk vektor. dari dua gambar menghasilkan dua vektor yang akan dibandingkan ‘kemiripannya’ dengan cosine similarity.

Langkah pertama adalah menormalisasi gambar sehingga rentang nilai warna berada diantara (0,1).

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

Langkah kedua adalah mencari cmax, cmin, dan delta.

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

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

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

Langkah ketiga adalah mencari Hue, Saturation, dan value (HSV) dari gambar.

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

Langkah keempat adalah melakukan kuantifikasi dari hasil HSV sehingga dapat menjadi histogram.

$$H = \begin{cases} 0 & h \in [316, 360] \\ 1 & h \in [1, 25] \\ 2 & h \in [26, 40] \\ 3 & h \in [41, 120] \\ 4 & h \in [121, 190] \\ 5 & h \in [191, 270] \\ 6 & h \in [271, 295] \\ 7 & h \in [295, 315] \end{cases}$$

$$S = \begin{cases} 0 & s \in [0, 0.2) \\ 1 & s \in [0.2, 0.7) \\ 2 & s \in [0.7, 1] \end{cases}$$

$$V = \begin{cases} 0 & v \in [0, 0.2) \\ 1 & v \in [0.2, 0.7) \\ 2 & v \in [0.7, 1] \end{cases}.$$

Langkah terakhir adalah melakukan komparasi antara kueri dengan basis data dengan menggunakan cosine similarity.

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

Tingkat kemiripan dihitung berdasarkan jarak hasil dari 1 (hasil pasti berada diantara (0,1).

1.3. CBIR DENGAN PARAMETER TEKSTUR

Misalkan citra masukan memiliki N_c dan N_r piksel dalam arah horizontal dan vertikal secara berturut-turut. Anggap Z_c sebagai domain ruang horizontal dan Z_r sebagai domain ruang vertikal. Ketika arah dan jarak diberikan, elemen matriks dapat diungkapkan dengan menghitung logaritma piksel tingkat keabuan *co-occurrence* i dan j . Anggap jaraknya 1, dengan arah sama dengan 0° , 45° , 90° , 135° masing-masing, rumusnya adalah:

$$P(i, j/1, 0) = \# \left\{ [(k, l), (m, n)] \in (Z_r \times Z_c) \mid |k - m| = 0, |l - n| = 1, f(k, l) = i, f(m, n) = j \right\} \quad (3)$$

$$P(i, j/1, 90) = \# \left\{ [(k, l), (m, n)] \in (Z_r \times Z_c) \mid |k - m| = 1, |l - n| = 0, f(k, l) = i, f(m, n) = j \right\} \quad (4)$$

$$P(i, j/1, 45) = \# \left\{ [(k, l), (m, n)] \in (Z_r \times Z_c) \mid (k - m) = 1, (l - n) = -1 \text{ or } (k - m) = -1, (l - n) = 1, f(k, l) = i, f(m, n) = j \right\} \quad (5)$$

$$P(i, j/1, 135) = \# \left\{ [(k, l), (m, n)] \in (Z_r \times Z_c) \mid (k - m) = 1, (l - n) = 1 \text{ or } (k - m) = -1, (l - n) = -1, f(k, l) = i, f(m, n) = j \right\} \quad (6)$$

$$= \# \left\{ [(k, l), (m, n)] \in (Z_r \times Z_c) \mid (k - m) = 1, (l - n) = 1 \text{ or } (k - m) = -1, (l - n) = -1, f(k, l) = i, f(m, n) = j \right\}$$

Gambar warna akan diubah menjadi gambar skala abu-abu oleh rumus 7 , dengan jumlah skala abu-abu sebanyak 256.

$$Y = 0.29 \times R + 0.587 \times G + 0.114 \times B \quad (7)$$

di mana Y adalah nilai skala abu-abu. R,G,dan G mewakili nilai komponen merah, hijau, dan biru masing-masing.

Empat matriks ko-oksuransi terbentuk sesuai dengan rumus 3 hingga rumus 6 dalam empat arah. Empat parameter tekstur: kapasitas, entropi, momen inersia, dan keterkaitan dihitung. Akhirnya, rata-rata dan deviasi standar masing-masing parameter diambil sebagai setiap komponen fitur tekstur.

Untuk gambar dan vektor fitur yang sesuai, asumsikan nilai komponen fitur memenuhi distribusi Gaussian. Pendekatan normalisasi Gaussian digunakan untuk menerapkan normalisasi internal agar setiap fitur memiliki bobot yang sama.

$$h^{i,j'} = \frac{h^{i,j} - m_i}{\sigma_i} \quad (8)$$

di mana adalah rata-rata dan adalah deviasi standar. akan diunitkan dalam rentang $[-1,1]$.

Fitur tekstur setiap gambar dihitung sesuai dengan langkah-langkah di atas. Nilai-nilai tekstur dibandingkan dengan cosine similarity, semakin kecil perbedaannya, semakin besar nilai cosine similaritynya.

2. Pengembangan *Website*

Pengembangan *Website* atau yang biasa dikenal dengan *webdev* adalah sebuah kegiatan yang berkaitan dengan menciptakan, membangun, dan menjaga *website* dan *web applications* yang berjalan secara daring di peramban. Selain itu *web-dev* juga dapat mencakup kegiatan seperti *web design*, *web programming*, dan *database management*.

Pengembangan *website* sangat erat dengan pekerjaan mendesain fitur dan fungsionalitas suatu aplikasi. Istilah pengembangan biasanya diperuntukan untuk kegiatan konstruksi yaitu memprogram sebuah *website*.

Alat-alat dasar yang terlibat dalam pengembangan *website* adalah bahasa pemrograman yang disebut HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), dan JavaScript. Namun, ada sejumlah program lain yang digunakan untuk "mengelola" atau memfasilitasi pembangunan situs yang jika tidak, harus dilakukan "dari awal" dengan menulis kode. Sejumlah sistem manajemen konten (CMS) termasuk dalam kategori ini, termasuk WordPress, Joomla!, Drupal, TYPO3, dan Adobe Experience Manager.

Bab 3

Analisis Pemecahan Masalah

1. Langkah-langkah Pemecahan Masalah

1.1. Langkah-langkah Pemecahan Masalah CBIR dengan Parameter Warna

Tahap-tahap yang perlu dilakukan untuk mencari similaritas dua gambar sama seperti yang tertera pada Teori Dasar, yaitu:

1. Mengubah gambar menjadi sebuah matrix
2. Menormalisasi Matrix
3. Membagi Matrix menjadi submatrix berukuran seperenambelas matrix asli
4. Mencari Cmax, Cmin, dan delta
5. Mengubah nilai RGB yang sudah di-normalisasi menjadi nilai HSV
6. Memetakan nilai HSV menjadi histogram HSV(vektor berdimensi 14) dimana 8 dimensi pertama (bins) adalah untuk Hue, 3 dimensi selanjutnya untuk Saturation, dan 3 bins terakhir untuk Value.
7. Mencari nilai kemiripan menggunakan cosine similarity
8. mengulang langkah 1-6 untuk setiap image pada database

Untuk preprocessing matrix (membagi matrix menjadi 16 submatrix), kami menggunakan list slicing. Karena banyaknya kemungkinan kelebihan pixel yang muncul (jika ukuran gambar tidak habis dibagi empat (ada 15 kemungkinan berbeda)), kami memutuskan untuk menempatkan pixel-pixel pada submatrix terluar kanan bawah .

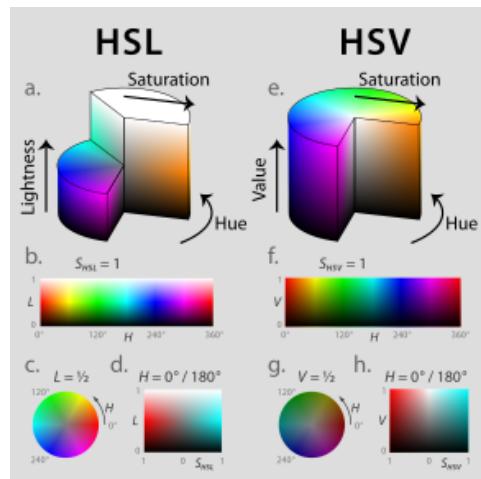
1.2. Langkah-langkah Pemecahan Masalah CBIR dengan Parameter Tekstur

Seperti yang telah dijelaskan pada teori dasar, terdapat beberapa langkah untuk menemukan kemiripan dua gambar dengan parameter tekstur

1. Mengubah gambar menjadi grayscale image
2. Mengubah grayscale image menjadi matrix berisi gray pixel
3. Membandingkan gray pixel yang bersebelahan dengan arah tertentu lalu menambahkan nilai pada co-occurrence matrix di indeks yang sesuai dengan nilai pixel yang bersebelahan.
4. Mencari fitur-fitur tekstur seperti contrast, homogeneity, dan entropy yang kemudian dimasukkan ke sebuah vektor.
5. Membandingkan vektor dua gambar dengan cosine similarity

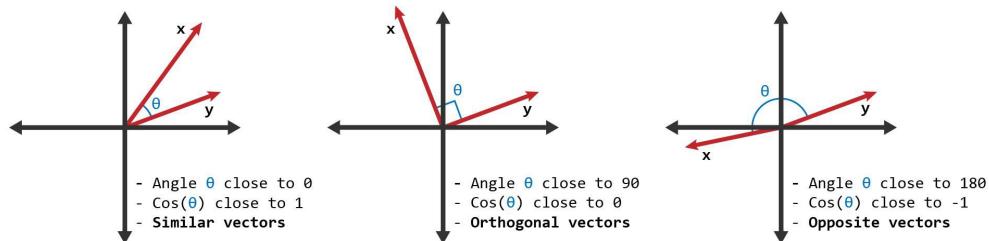
2. Proses Pemetaan Masalah Menjadi Elemen-Elemen Pada Aljabar Geometri

2.1. Pemetaan Masalah Menjadi Elemen Aljabar Geometri pada CBIR Warna



Dalam ruang warna HSV, HSV sendiri dapat direpresentasikan sebagai vektor dari color circle yang memiliki depth yaitu Value yang berupa kecerahan warna, Saturation yang merupakan jarak dari titik tengah, dan Hue yang berupa derajat dari 0, yaitu merah.

Untuk mencari kemiripan dari kedua gambar, kita dapat membandingkan dua vektor HSV yang didapat dari gambar dan menggunakan cosine similarity untuk menilai similaritas. semakin kecil sudut kedua vektor maka nilai cosine semakin besar ($\cos 0=1$) dan semakin besar sudutnya semakin mengecil.



2.2. Pemetaan Masalah Menjadi Elemen Aljabar Geometri pada CBIR Tekstur

Dari matriks *co-occurrence* bisa didapatkan nilai-nilai komponen tekstur yang beragam. Dalam program ini, diambil 3 nilai yaitu contrast, homogeneity, dan entropy yang dimasukkan kedalam sebuah vektor. Untuk mencari kemiripan dari dua gambar, kita dapat membandingkan vektor dari kedua gambar tersebut dengan cosine similarity.

Bab 4

Implementasi dan Uji Coba

1. *Pseudocode Program Utama*

a. Pseudocode CBIR dengan Warna

```
{Program CBIR dengan Warna}
Use numpy as np
Use asarray from numpy
Use os
Use process_time from time
Use PIL from image

function RGBNorm(input Pict : Array of array of int) -> array of array of int
    {Algoritma}
    -> Pict/255.0

Function HSV(input Pict : Array of array of int) -> array
    {Kamus Lokal}
    red_channel, green_channel, blue_channel : array of array of int
    cmax, cmin, delta : array of array of int
    boolmaxR, boolmaxG, boolmaxB, boolG : array of array of bool
    htemp, stemp, vtemp, array of array of float
    vector : array of int (0..14)

    {Algoritma}
    red_channel <- Pict(:,:,:,0)
    green_channel <- Pict(:,:,:,1)
    blue_channel <- Pict(:,:,:,2)

    cmax <- np.max(Pict, axis=2)
    cmin <- np.min(Pict, axis=2)
    delta <- np.subtract(cmax, cmin)

    boolmaxR <- np.logical_and(delta!=0,cmax=red_channel)
    boolmaxG <- np.logical_and(delta!=0,cmax=green_channel)
    boolmaxB <- np.logical_and(delta!=0,cmax=blue_channel)
    boolG     <- np.logical_and(delta!=0,delta!=0)

    htemp=np.zeros_like(red_channel,dtype<-np.float32)
    stemp=np.zeros_like(red_channel,dtype<-np.float32)
    vtemp=np.zeros_like(red_channel,dtype<-np.float32)

    red_channel<-RGBNorm(red_channel)
    green_channel<-RGBNorm(green_channel)
```

```

blue_channel<-RGBNorm(green_channel)

cmax<-RGBNorm(cmax)
delta<-RGBNorm(delta)

htemp[boolG]=0

htemp[boolmaxR]=
(60*((green_channel[boolmaxR]-blue_channel[boolmaxR])/delta[boolmaxR])
mod 6))

htemp[boolmaxG]=
(60*((blue_channel[boolmaxG]-red_channel[boolmaxG])/delta[boolmaxG])+2))

htemp[boolmaxB]=
(60*((red_channel[boolmaxB]-green_channel[boolmaxB])/delta[boolmaxB])+4)
)

stemp[cmax=0]<-0
stemp[cmax!=0]<-delta[cmax!=0]/cmax[cmax!=0]

vtemp=cmax

vector<-Quantify(htemp,stemp,vtemp)
->vector

function Quantify(input hue: Array of array of int,sat: Array of array of
int,val: Array of array of int)->array of array of int (0..14)
{kamus lokal}
h,s,v : array of array of float
hueRange : array of int (0..8)
satRange : array of int (0..3)
valRange : array of int (0..3)
huedict : dict of int (0..8)
huedict : dict of int (0..3)
huedict : dict of int (0..3)
H : array of int (0..8)
S : array of int (0..3)
V : array of int (0..3)

{algoritma}
h<-np.copy(hue)
s<-np.copy(sat)
v<-np.copy(val)

h[np.logical_and(h>315, h=0)]<-0
h[np.logical_and(h>0, h<=25)]<-1
h[np.logical_and(h>25, h<=40)]<-2
h[np.logical_and(h>40, h<=120)]<-3

```

```

h[np.logical_and(h>120, h<=190)]<-4
h[np.logical_and(h>190, h<=270)]<-5
h[np.logical_and(h>270, h<=295)]<-6
h[np.logical_and(h>295, h<=315)]<-7

hueRange <- [0,1,2,3,4,5,6,7]
huedict <- {value: np.count_nonzero(h=value) for value in hueRange}
H = np.array(list(huedict.values()), dtype=np.uint32)

s[np.logical_and(s>0.7, s<=1)]<-2
s[np.logical_and(s>0.2, s<=0.7)]<-1
s[np.logical_and(s>=0, s<=0.2)]<-0

satRange <- [0,1,2]
satdict <- {value: np.count_nonzero(s=value) for value in satRange}
S <- np.array(list(satdict.values()), dtype=np.uint32)

v[np.logical_and(v>0.7, v<=1)]<-2
v[np.logical_and(v>0.2, v<=0.7)]<-1
v[np.logical_and(v>=0, v<=0.2)]<-0

valRange <- [0,1,2]
valdict <- {value: np.count_nonzero(v=value) for value in valRange}
V <- np.array(list(valdict.values()), dtype=np.uint32)

-> np.concatenate((H,S,V))

def cosineSimAvg(v1: array of array of int ,v2: array of array of int)
-> float
    {kamus lokal}
    hasil : float
    i : int
    vDot : int
    vNorm1, vNorm2 : float
    {algoritma}
    hasil=0.0
    i traversal 0..16:
        vDot<-np.dot(v1[i],v2[i])
        vNorm1<-np.linalg.norm(v1[i])
        vNorm2<-np.linalg.norm(v2[i])
        hasil<-hasil + np.divide(vDot,np.multiply(vNorm1,vNorm2)))
    -> hasil/16

def Result(Pict: String, n: int):
    {kamus lokal}
    result_array: array of array of int
    h,w : int
    i,j,nth : int

```

```

{algoritma}
Pict=Image.open(Pict).convert("RGB")
Pict=asarray(Pict)
result_array = np.zeros((n*n,14), dtype=np.int64)
h=Pict.shape[0]//n
w=Pict.shape[1]//n
nth=0
for i in range (0,n):
    for j in range (0,n):
        result_array[nth]=HSV(Pict[i*h:h*(i+1)-1, j*w:w*(j+1)-1 ,:])
        nth+=1
return result_array

function get_result_image_texture() -> array of array of (String, float)
{Kamus Lokal}
current_folder, folder_name, input_name, input_path, folder_path: String
dataset_paths, submit_path : array of String
input_glc : array of integer
val : float

{Algoritma}
current_folder <- os.path.dirname(os.path.abspath(__file__))
folder_name <- 'static/dataset_picture'
input_name <- 'static/submitted_picture'
input_path <- os.path.join(current_folder, input_name)
folder_path <- os.path.join(current_folder, folder_name)
dataset_paths <- [os.path.join(folder_path, file) for file in
os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path,
file))]
submit_path <- [os.path.join(input_path, file) for file in
os.listdir(input_path) if os.path.isfile(os.path.join(input_path, file))]
v1 <- Result(submit_path[0])
i traversal (0...dataset_paths.length - 1)
    val < cosineSimilarityAvg(v1, Result( dataset_paths[i]))
if (val > 0.6) then
    similarities.append([val*100, dataset_paths[i]])
-> similarities

```

b. Pseudocode CBIR dengan Tekstur

```

{Program CBIR dengan Tekstur}
Use numpy as np
Use os
Use math
Use PIL from image

function convert_to_grayscale(input input_path : String) -> Image

```

```

{Kamus Lokal}
image : Image
grayscale_image : array of array of integer
grayscale_image_ret : Image
width, height, x, y, gray : integer

{Algoritma}
image <- Image.open(input_path).convert("RGB")
<width, height> <- grayscale_image.size
for x traversal (0...width-1)
    for y traversal (0...height-1):
        r, g, b <- grayscale_image.getpixel((x, y))
        gray <- int(0.299 * r + 0.587 * g + 0.114 * b)
        grayscale_image_ret <- grayscale_image.putpixel((x,
y), (gray, gray, gray))
-> grayscale_image_ret

function glcm(input input: Image) -> array of integers [0...2]
{Kamus Lokal}
vektor : array of integers [0...2]
Contrast, Homogeneity, Entropy : integer
glcm : array of integers [0...255]

{Algoritma}
image <- Gray.convert_to_grayscale(input)
distance <- 1
angle <- 0
height, width <- image.size
level <- 256
image <- np.array(image)
glcm <- np.zeros((level, level), dtype <- np.uint32)
for i traversal (0...width-3):
    for j traversal (0...height-2):
        current_pixel <- image[i, j][0]
        neighbor_pixel <- image[i, j + distance][0] if angle = 0 else None
        glcm[current_pixel, neighbor_pixel] <- glcm[current_pixel,
neighbor_pixel] + 1

        transpose_glcml <- np.transpose(glcm)
        glcm <- glcm + transpose_glcml
        glcm <- glcm.astype(float)
        glcm <- glcm / np.sum(glcm)

Contrast <-0
Homogeneity <-0
Entropy <-0

for i traversal (0...255) :
    for j traversal (0...255) :

```

```

    Contrast <- Contrast + glcm[i][j] * ((i-j)**2)
    Homogeneity <- Homogeneity + glcm[i][j] / (1 + (i-j)**2)
    if (glcm[i, j] != 0) then
        Entropy <- Entropy - glcm[i, j] * math.log(float(glcm[i, j]))
    vektor.append(Contrast)
    vektor.append(Homogeneity)
    vektor.append(Entropy)
-> vektor

function cosineSimilarity(input vektorA, vektorB : array of integers [0...2])
-> float
{Kamus Lokal}
atas, bawahA, bawahB, cosine : float
i : integer

{Algoritma}
atas <- 0
bawahA <- 0
bawahB <- 0
for i traversal (0...2):
    atas <- atas + vektorA[i]*vektorB[i]
    bawahA <- bawahA vektorA[i]*vektorA[i]
    bawahB <- bawahB vektorB[i]*vektorB[i]
cosine <- atas / (math.sqrt(bawahA) * math.sqrt(bawahB))
-> cosine

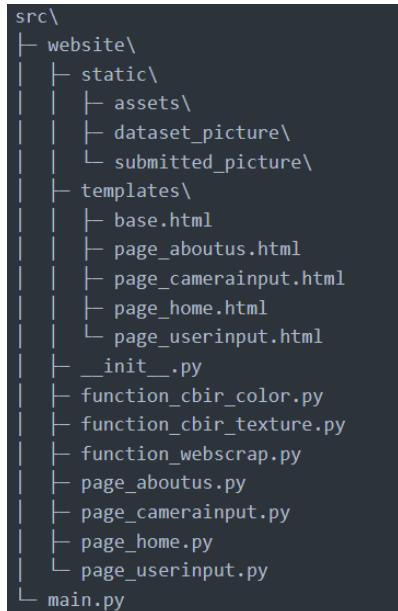
function get_result_image_texture() -> array of array of (String, float)
{Kamus Lokal}
current_folder, folder_name, input_name, input_path, folder_path : String
dataset_paths, submit_path : array of String
input_glcml : array of integer
val : float

{Algoritma}
current_folder <- os.path.dirname(os.path.abspath(__file__))
folder_name <- 'static/dataset_picture'
input_name <- 'static/submitted_picture'
input_path <- os.path.join(current_folder, input_name)
folder_path <- os.path.join(current_folder, folder_name)
dataset_paths <- [os.path.join(folder_path, file) for file in
os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path,
file))]
submit_path <- [os.path.join(input_path, file) for file in
os.listdir(input_path) if os.path.isfile(os.path.join(input_path, file))]
input_glcml <- glcm(convert_to_grayscale(submit_path[0]))
for i traversal (0...dataset_paths.length - 1)
    val < cosineSimilarity(input_glcml,
glcm(convert_to_grayscale(dataset_paths[i])))
    if (val > 0.6) then

```

```
    similarities.append([val*100, dataset_paths[i]])  
-> similarities
```

2. Struktur Program



Gambar *Struktur Program*

Seluruh program yang diperlukan dalam membangun Website CBIR berada di dalam folder src. Di dalam folder src terdapat folder website yang berisi source code web dan file main.py yang jika di run akan menjalankan web pada server lokal.

Didalam folder website terdapat folder static yang berisi asset gambar yang diperlukan dalam pembangunan web, dan folder dataset_picture serta submitted_picture sebagai tempat penyimpanan sementara gambar-gambar yang akan dilakukan pencarian menggunakan algoritma CBIR. Selain itu terdapat juga folder templates yang berisi file .html untuk front-end website. Terakhir terdapat file function_* kode python untuk algoritma CBIR dan webscraping serta file page_* yang berisi kode flask python untuk back-end website.

3. Cara Penggunaan Program

Program disajikan dalam bentuk web yang dibangun menggunakan Flask Python dan Bootstrap5 HTML CSS.

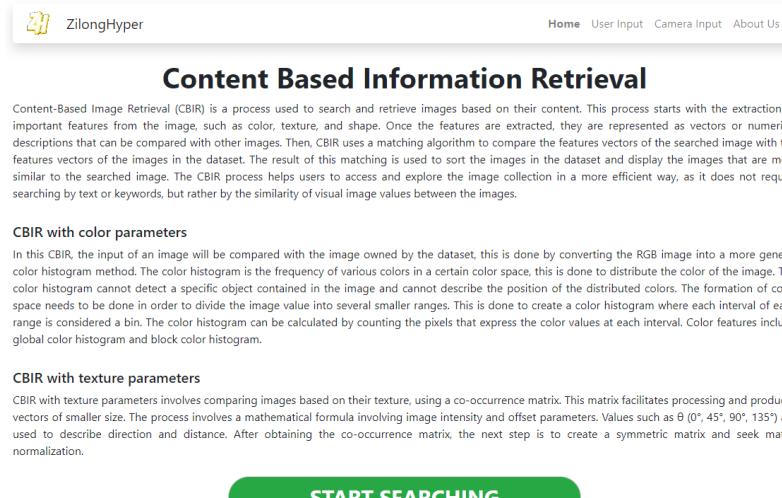
Agar program dapat berjalan seperti seharusnya, *directory* terminal harus berada pada folder src jika ingin menjalankan *web* pada server lokal.

Sebelum menjalankan program pada server lokal, diperlukan beberapa modul python yang harus di-instal:

1. Flask
2. Numpy
3. Requests
4. beautifulsoup4

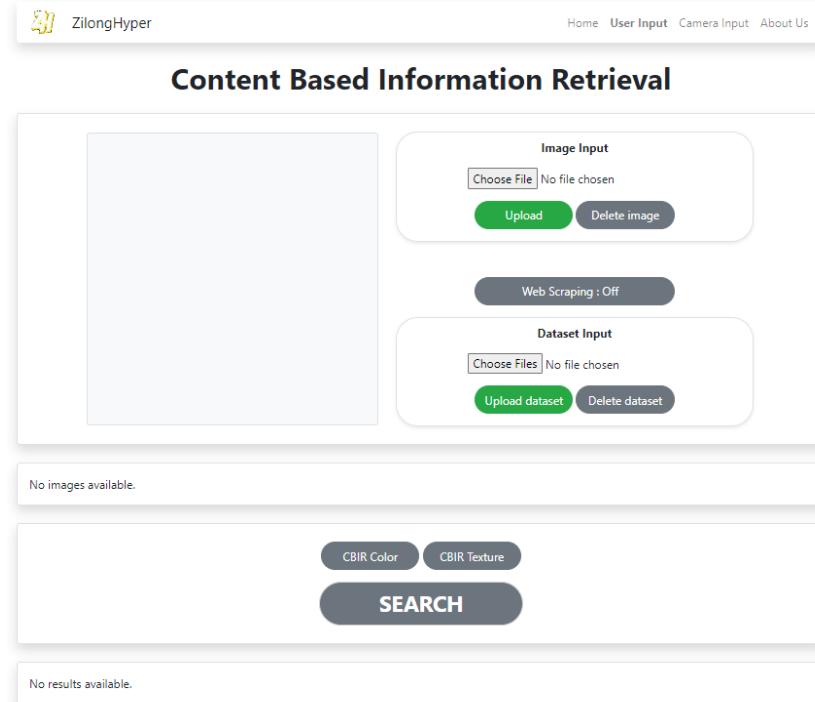
Instalasi dapat dilakukan dengan cara menjalankan command “`pip install [modul python]`” pada terminal. Selengkapnya dapat dilihat pada <https://docs.python.org/3/installing/index.html>.

Jika directory terminal sudah pada folder src dan seluruh modul python sudah terinstal, server web dapat dijalankan dengan menjalankan command “`py main.py`”. Jika berhasil akan ditampilkan alamat web <http://127.0.0.1:5000> pada terminal, silakan copy paste link tersebut pada browser Anda.



Gambar *Home Page*

Saat pertama kali membuka link tersebut, akan disajikan homepage dari web CBIR: ZilongHyper, Anda dapat menekan “START SEARCHING” atau “User Input” pada navbar untuk membuka halaman pencarian CBIR.

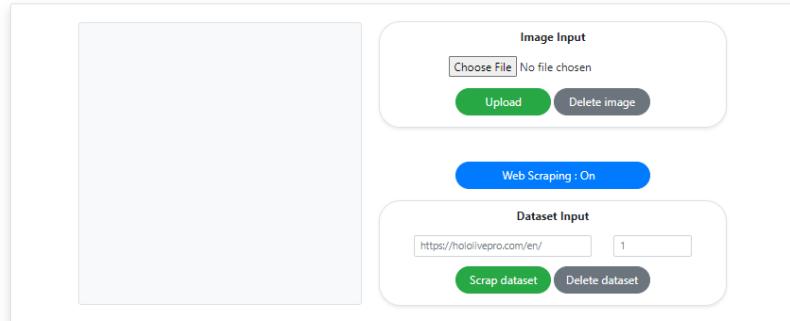


Gambar *User Input Page: Default Condition*

Pada halaman ini, anda dapat melakukan searching image yang memiliki similarity diatas 60% menggunakan metode CBIR Color maupun CBIR Texture. Dataset untuk pencarian dapat diupload dari local file atau dapat diperoleh menggunakan fitur web scraping.

Untuk mengupload image yang akan dijadikan acuan dapat diakses melalui kolom “Image Input” dengan menekan “Choose File”, lalu memilih file kemudian klik “Upload”, image tersebut kemudian akan ditampilkan pada gray box di sebelah kiri.

Untuk mengupload dataset dapat diakses melalui kolom “Dataset Input”. Web secara default akan menampilkan menu upload data set dari local file, cara mengupload data set adalah dengan cara menekan “Choose Files”, lalu memilih files kemudian klik “Upload dataset”.



Gambar *User Input Page: Web Scraping Toggled*

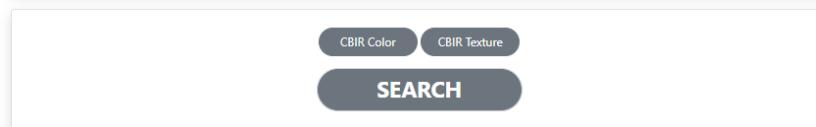
Jika menekan tombol toggle web scrap, akan ditampilkan menu perolehan dataset menggunakan metode web scrap, silakan isi link pada kolom sebelah kiri (disarankan link hasil image search google) dan isi jumlah maksimal gambar yang akan di scrap pada kolom sebelah kanan. Setelah itu klik tombol “Scrap dataset” untuk memulai proses scraping. Proses ini akan memakan waktu sesuai koneksi internet anda karena program akan mengunduh foto yang akan dijadikan dataset.

The screenshot shows the user input page with the following details:

- Image Input:** A placeholder image area and a "Choose File" button with "No file chosen". Buttons for "Upload" and "Delete image".
- Web Scraping:** A blue button labeled "Web Scraping : On".
- Dataset Input:** A "Dataset Input" section with a URL field containing "https://hololivepro.com/en/" and a count field set to "1". Buttons for "Scrap dataset" (green) and "Delete dataset" (red).
- Result Grid:** A grid titled "20 images available" displaying five images of the same blue-haired character in different poses and backgrounds. Below the grid is a pagination control with buttons labeled 1, 2, 3, and 4, where "3" is highlighted in blue.

Gambar *User Input Page: Image Available*

Jika image untuk perbandingan dan image untuk data set sudah tersedia, akan muncul tampilan seperti di atas. Image pada data set akan ditampilkan menggunakan pagination sebanyak lima gambar per page, selain itu tombol delete menjadi warna merah dan akan berfungsi sesuai namanya.



Gambar Search Box: *Default Condition*

Jika image untuk perbandingan dan image dataset belum tersedia, search box akan ditampilkan seperti di atas dan tombolnya tidak fungsional.



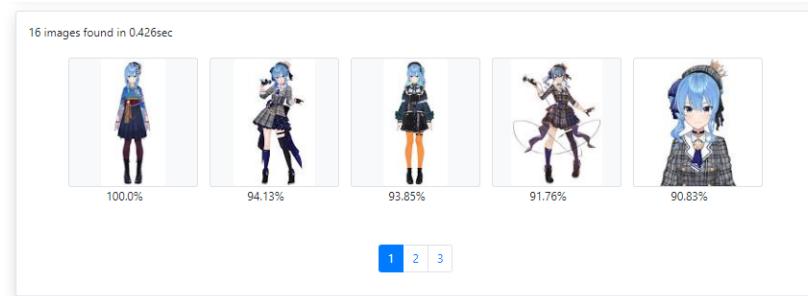
Gambar Search Box: *CBIR Color Selected*

Jika image untuk perbandingan dan image dataset sudah tersedia, button toggle mode (CBIR Color/ CBIR Texture) dapat diklik dan jika aktif warnanya akan berubah jadi biru, jika salah satu mode search sudah aktif, tombol “SEARCH” menjadi warna putih dan dapat diklik untuk memulai proses searching.



Gambar Result Box: *Empty*

Berikut adalah tampilan result box sebelum melakukan searching atau jika searching gagal (ditemukan 0 gambar).



Gambar Result Box: *Image Found*

Berikut adalah tampilan result box setelah berhasil melakukan searching, akan disajikan jumlah image yang ditemukan dan waktu pencarinya. Image akan disajikan dengan pagination sebanyak lima image per page. Image yang disajikan adalah image dengan similaritas > 60%

ZilongHyper

Home User Input Camera Input About Us

Content Based Information Retrieval



Image Input
Choose File No file chosen
Upload Delete image

Web Scraping : On

Dataset Input
https://hololivepro.com/en/ 1
Scrap dataset Delete dataset

20 images available



1 2 3 4

CBIR Color CBIR Texture
SEARCH

16 images found in 0.426sec



Image	Similarity (%)
1	100.0%
2	94.13%
3	93.85%
4	91.76%
5	90.83%

1 2 3

Gambar User Input Page : Finished Searching



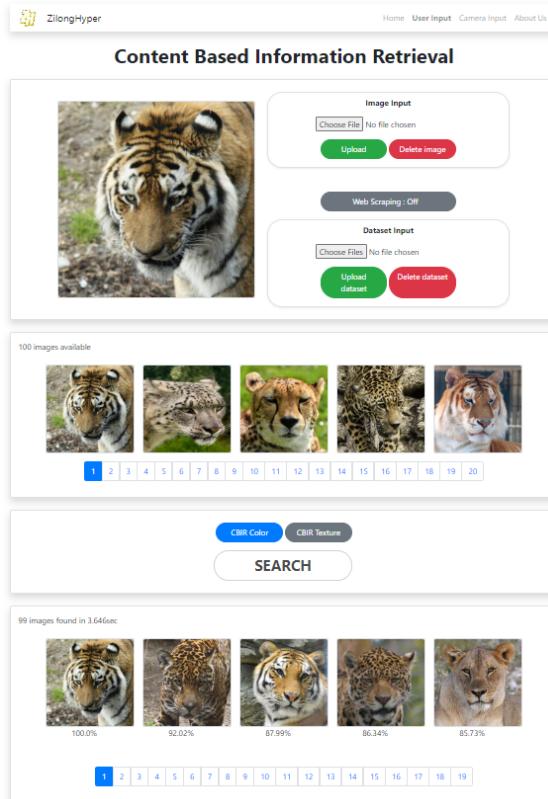
Gambar *Camera Input Page*

A screenshot of a web page titled "Content Based Information Retrieval". At the top left is a logo with the text "ZilongHyper". At the top right are navigation links: "Home", "User Input", "Camera Input", and "About Us". Below the title, the heading "Meet Our Team" is displayed. Three team members are shown in individual cards: 1. Andi Marihot Sitorus, CBIR Texture Programmer (image shows a young man with short hair). 2. Maulana Muhamad Susetyo, CBIR Color Programmer (image shows a young man with long dark hair). 3. Ahmad Rafi Maliki, Web Developer, Web Scraping Programmer (image shows a young man with glasses).

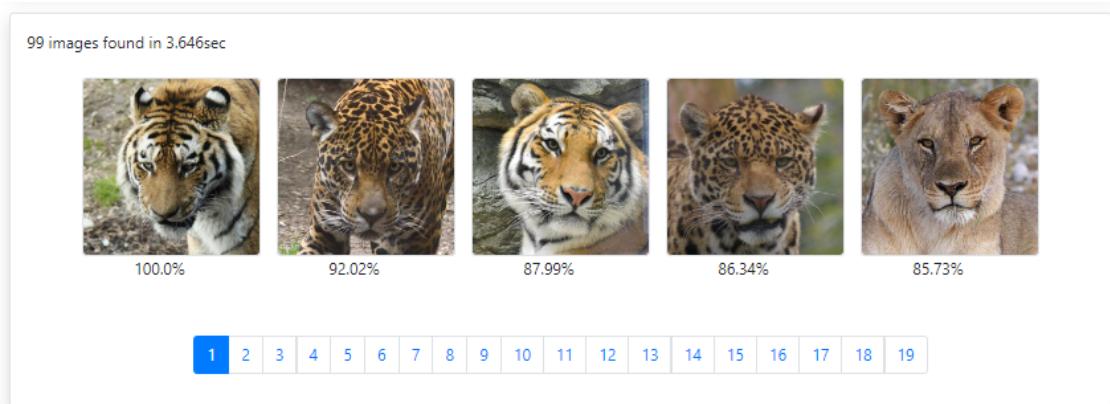
Gambar *About Us Page*

4. Hasil Pengujian

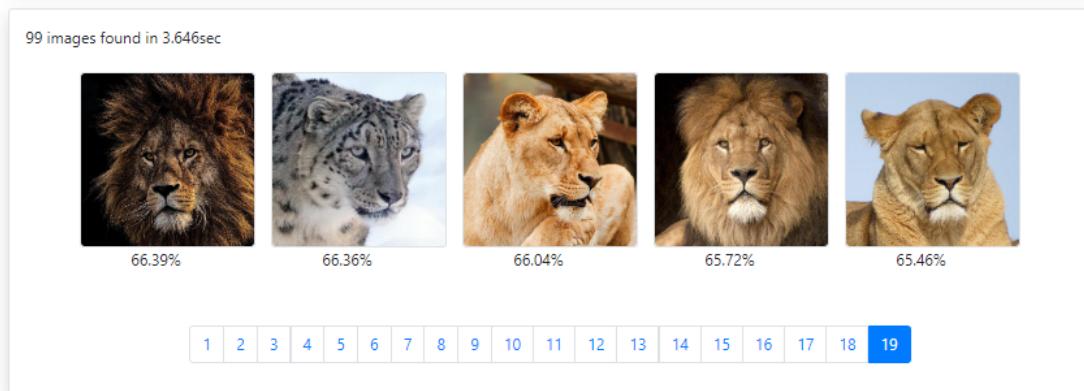
- Pengujian metode CBIR Color menggunakan dataset template (100 dataset)



Gambar Hasil Pengujian metode CBIR Color menggunakan dataset template

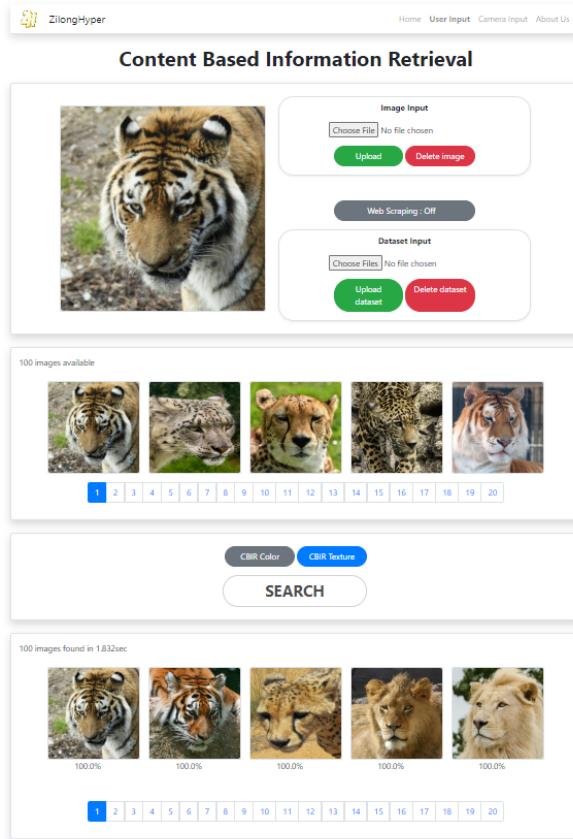


Gambar Hasil Pengujian metode CBIR Color menggunakan dataset template: page 1



Gambar Hasil Pengujian metode CBIR Color menggunakan dataset template: page 19

2) Pengujian metode CBIR Tekstur menggunakan dataset template (100 dataset)



Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset template

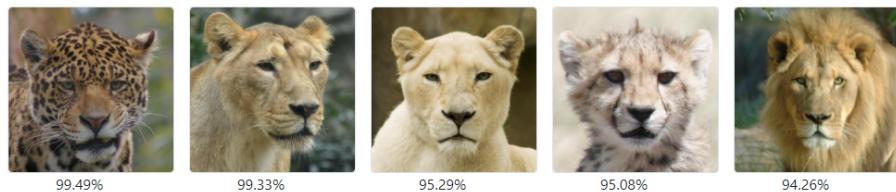
100 images found in 1.832sec



[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#)

Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset template : page 1

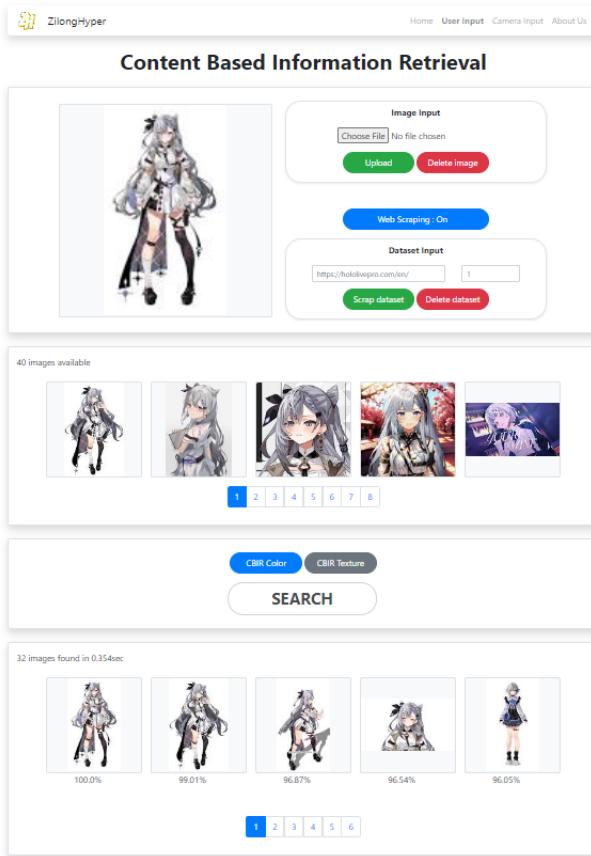
100 images found in 1.832sec



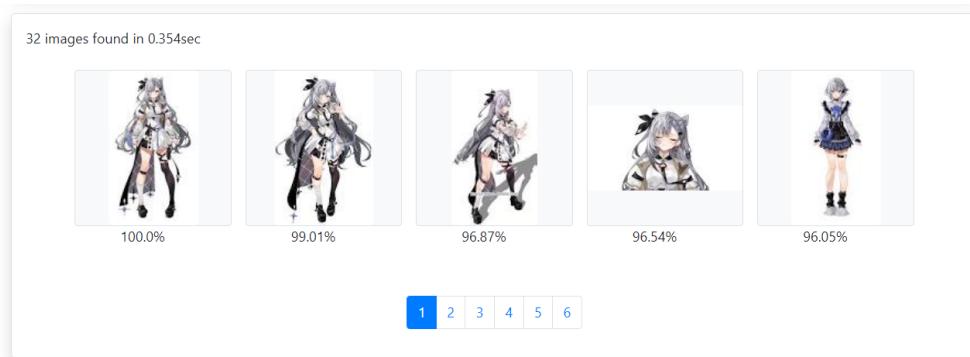
[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#)

Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset template : page 20

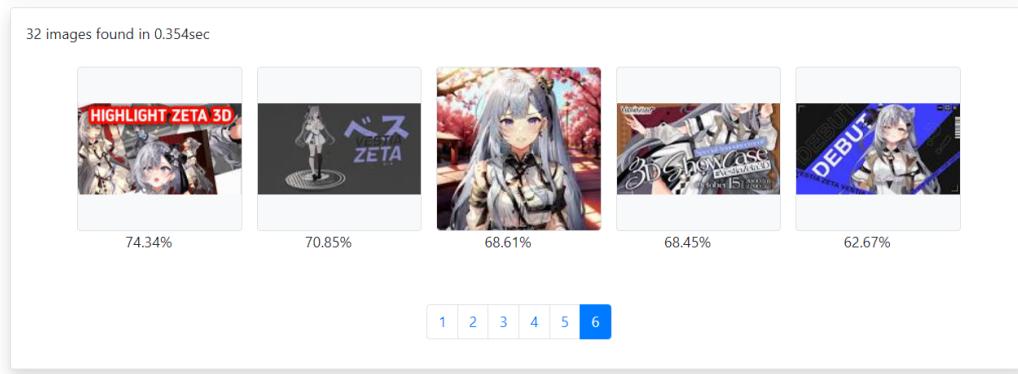
3) Pengujian metode CBIR Color menggunakan dataset scraping 1 (40 dataset)



Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1



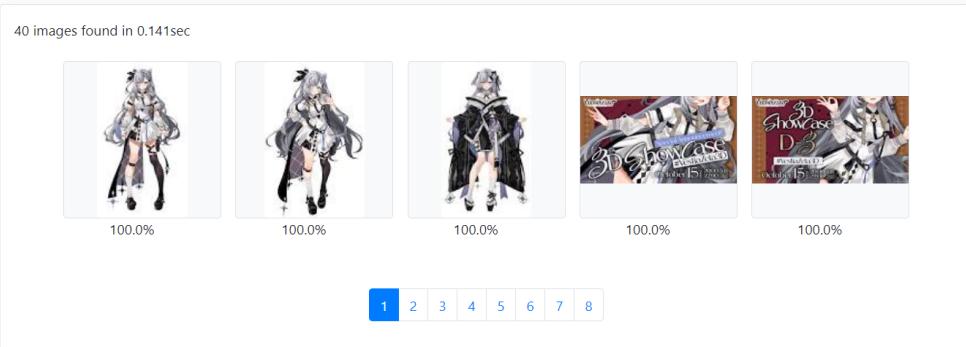
Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1 : page 1



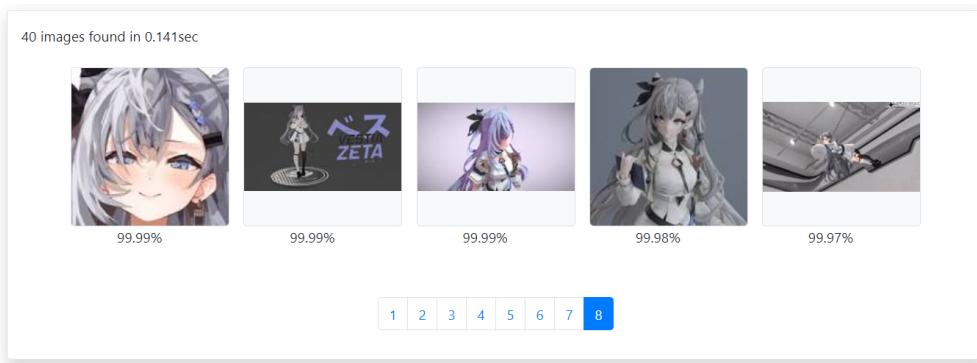
Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1 : page 6

4) Pengujian metode CBIR Tekstur menggunakan dataset scraping 1 (40 dataset)

Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 1

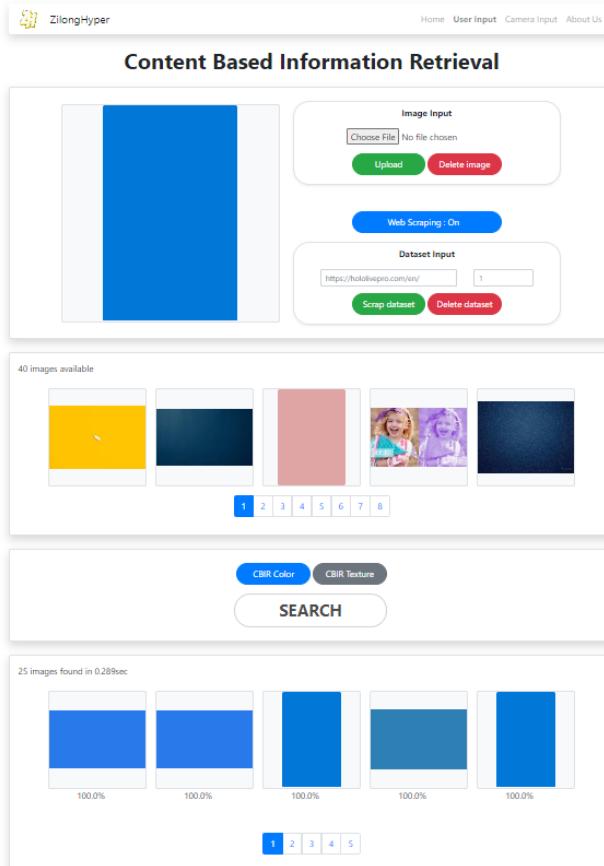


Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 1: page 1

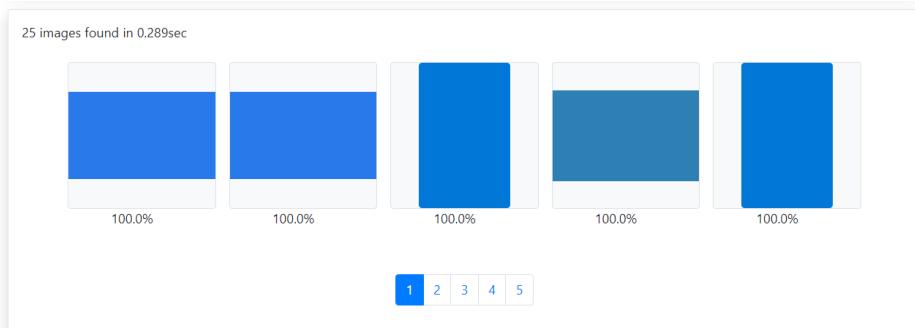


Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 1: page

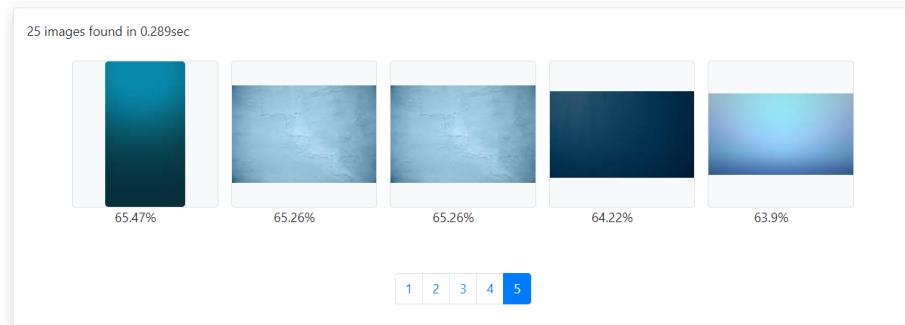
- 5) Pengujian metode CBIR Color menggunakan dataset scraping 2 (40 dataset)



Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1



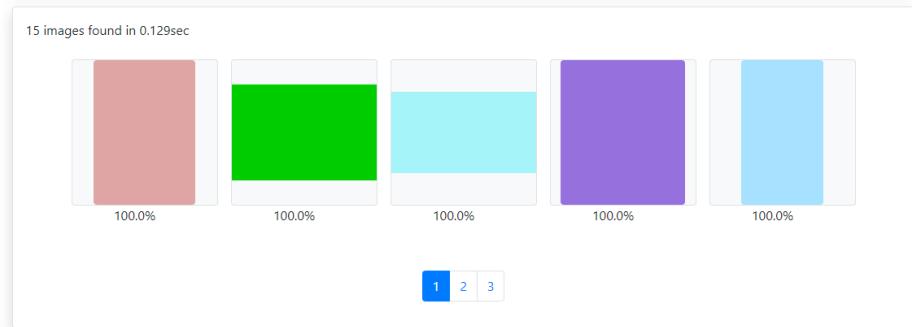
Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1: page 1



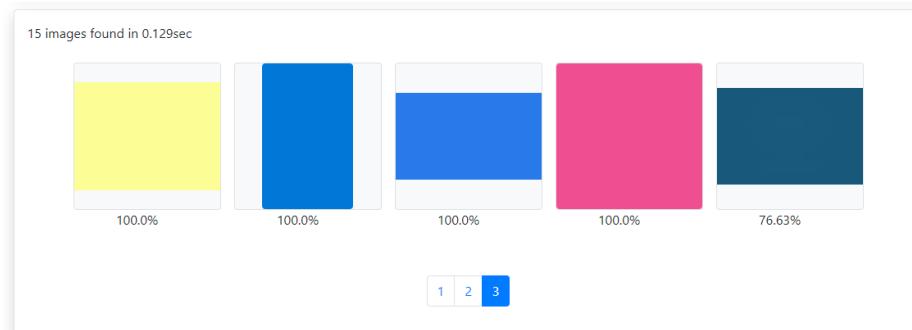
Gambar Hasil Pengujian metode CBIR Color menggunakan dataset scraping 1: page 6

- 6) Pengujian metode CBIR Tekstur menggunakan dataset scraping 2 (40 dataset)

Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 2



Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 2: page 1



Gambar Hasil Pengujian metode CBIR Tekstur menggunakan dataset scraping 2: page 3

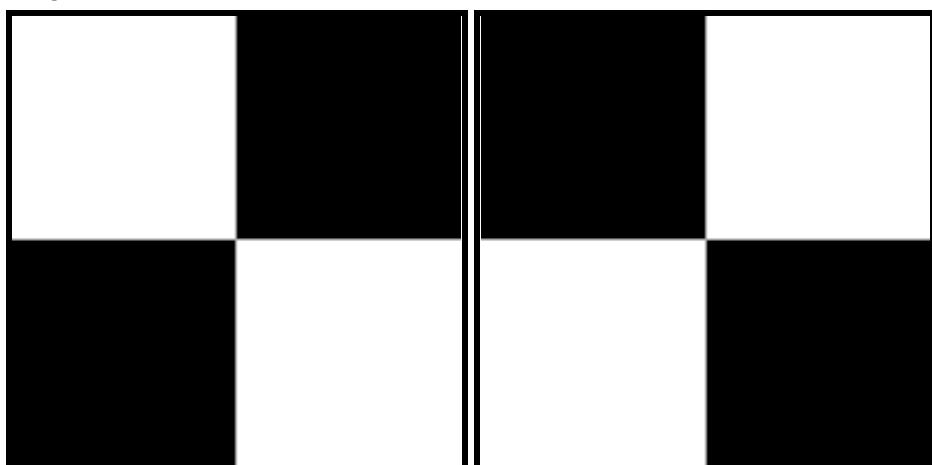
5. Analisis dari desain algoritma pencarian

Pada Hasil pengujian 1 diatas (untuk CBIR dengan parameter warna) pada halaman pertama dapat dilihat hasil yang didapat adalah gambar dengan komposisi warna (jingga, hitam, putih) yang berdekatan dengan gambar orisinil. Pada Halaman terakhir adalah gambar dengan *lighting* yang berbeda dengan gambar orisinil sehingga walaupun hewan ataupun warnanya mirip tetapi karena persentase valuenya rendah, maka gambar dianggap tidak terlalu mirip oleh algoritma.

Pada Hasil pengujian 2 (CBIR dengan parameter tekstur), hasil pada halaman pertama hingga halaman terakhir menunjukkan kemiripan yang sangat tinggi. Hasil itu didapatkan karena dataset berisi hewan-hewan yang mirip kucing atau *felidae* sehingga terdapat kemiripan yang tinggi pada tekstur setiap gambar dari hewan di dataset.

Pada Hasil pengujian 3 (untuk CBIR dengan parameter warna), hasil pada halaman pertama adalah hasil yang sangat mirip dengan gambar orisinil secara komposisi warna, dimana terdapat latar belakang putih dengan karakter yang berwarna mayoritas abu-abu dan putih. pada halaman terakhir. walaupun foreground dari gambar masih mayoritas abu-abu, tetapi karena background yang kaa warna, nilai kemiripan menjadi turun. Dapat juga dilihat dari gambar pada halaman terakhir proporsi gambarnya berbeda dengan gambar awal (tidak ditunjukkan pada gambar 1 karena proporsi gambar pada dataset identik) karena saat preprocessing gambar dibagi menjadi 16 subgambar. ini membantu untuk membedakan gambar yang secara keseluruhan memiliki komposisi warna yang identik tetapi letaknya tidak sesuai.

Sebagai Contoh:



Jika tidak dibuat subgambar, kedua gambar ini akan mengembalikan hasil yang identik.

Pada Hasil pengujian 4 (CBIR dengan parameter tekstur), gambar di dataset berisi satu karakter fiktif sehingga setiap gambar akan menghasilkan tekstur yang relatif sama. Semua gambar berisi warna yang cukup serupa dan pastinya teksturnya juga akan seperti itu. Oleh karena itu dihasilkan persentase kemiripan yang sangat tinggi untuk semua gambar di dataset.

Pada Hasil pengujian 5 (untuk CBIR dengan parameter warna), karena gambar orisinil adalah blok berwarna biru, gambar yang dikembalikan adalah blok berwarna biru. Pada gambar keempat halaman pertama terdapat hasil yang menarik, yaitu warna biru yang tidak seratus persen sama dengan gambar orisinil. Hal ini dikarenakan pada pembuatan bins range valuenya hanya dibagi menjadi 3. sehingga ada kasus dimana perbedaan yang cukup signifikan masuk ke dalam rentang yang sama

The image shows two separate color conversion interfaces side-by-side. Both interfaces allow input via hex code or RGB values, and both show identical output fields for Hue (H), Saturation (S), and Value (V), along with a color preview box.

Color Hex / RGB	Enter RGB hex code (#):	Enter red color (R):	Enter green color (G):	Enter blue color (B):	Hue (H):	Saturation (S):	Value (V):	Color preview:
2e7fb3	2e7fb3	46	127	179	203	74.3	70.2	Blue
0177d7	0177d7	1	119	215	207	99.5	84.3	Blue

dapat dilihat pada gambar diatas dimana warna kanan adalah warna blok orisinil dan warna kiri adalah warna blok pada kolom keempat halaman pertama. Karena untuk saturation >70% dan value>70% masuk dalam dimensi vektor yang sama, kedua gambar tersebut dianggap sama oleh algoritma, padahal ada perbedaan saturation sebesar 25.2 persen.

Pada Hasil pengujian 6 (CBIR dengan parameter tekstur), gambar input merupakan blok berwarna biru. Gambar dengan kemiripan tinggi yang dihasilkan memiliki warna yang cukup berbeda dengan gambar input. Hal itu disebabkan kemiripan tekstur tidak dapat dinilai hanya dengan warna saja. Warna-warna yang terlihat mirip, bisa saja memiliki tekstur yang cukup berbeda di grayscale image. Nilai similarity yang diberikan berdasarkan tekstur gambar pada grayscale level sehingga banyak warna berbeda yang memiliki kemiripan tinggi dengan blok biru tersebut.

Bab 5

Penutup

1. Kesimpulan

Kedua metode searching baik CBIR Color maupun CBIR Tekstur bekerja dengan baik. CBIR Color dapat bekerja secara maksimal jika warna pada dataset variatif dan yang dicari adalah gambar dengan kemiripan warna, metode ini dapat mengenali gambar dengan warna yang mirip. Sedangkan, CBIR Tekstur dapat bekerja secara maksimal jika dataset adalah gambar yang tekstur (pola) bervariasi, metode ini dapat mengenali gambar mana yang polanya bervariasi atau mana yang monoton.

2. Saran

Tambahkan fitur bonus seperti *camera input*, *caching result*, dan *export result*.

3. Komentar atau Tanggapan

Tugas besar ini sangat bagus karena mendorong kami mahasiswa untuk mempelajari sebagian kecil dari *computer graphics* dan *web development*.

Tugas ini membuat saya mengerti numpy lebih dalam dan mengapresiasi kecepatan suatu algoritma.

4. Refleksi

Telah dipelajari algoritma *content based information retrieval* menggunakan metode *color* dan *texture* serta *web development* sederhana menggunakan Flask Python dan Bootstrap5 HTML CSS.

5. Ruang Perbaikan atau Pengembangan

Untuk tugas berikutnya dapat memanfaatkan Gantt chart untuk membantu manajemen waktu sehingga tugas dapat selesai lebih awal dan lebih banyak bonus dapat terselesaikan.

Daftar Pustaka

<https://brainstation.io/career-guides/what-is-web-development>

<https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm>

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

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

<https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask>

Link Repository GitHub

<https://github.com/rafimaliki/Algeo02-22127>