LAPORAN TUGAS BESAR II
IF2123 ALJABAR LINIER DAN GEOMETRI
CONTENT-BASED IMAGE RETRIEVAL DENGAN PARAMETER WARNA DAN TEKSTUR

Disusun oleh:

Kristo Anugrah (13522024)

Adril Putra Merin (13522068)

Berto Richardo Togatorop (13522118)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

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 kalian tahu adalah Google Lens.

BAB II
LANDASAN TEORI

2.1 Dasar teori

Representasi visual dalam bentuk gambar sangat sering digunakan pada masa kini karena. Dengan adanya perkembangan teknologi yang pesat, penyimpanan dan pengiriman gambar dengan jumlah besar menjadi mungkin untuk dilakukan. Selain pencarian kata, pencarian gambar juga sangat dibutuhkan akhir-akhir ini. Content-based image retrieval atau biasa disingkat menjadi CBIR adalah salah satu cara paling efektif untuk mengakses data visual. CBIR sendiri adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan ekstraksi fitur-fitur penting dari suatu gambar, seperti warna, tekstur, dan bentuk. Fitur-fitur tersebut kemudian akan direpresentasikan menggunakan vektor atau deskripsi numerik lain yang dapat dibandingkan CBIR kemudian menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur pada gambar dalam dataset. Hasil pencocokan ini digunakan untuk mengurutkan gambar-gambar dalam *dataset* dan menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR memungkinkan pengguna untuk mengakses dan mengeksplorasi koleksi gambar dengan cara yang lebih efisien, karena tidak memerlukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai citra visual antara gambar-gambar tersebut. Pada tugas ini, akan digunakan dua tipe parameter yang paling populer digunakan, yaitu color-based CBIR dan texture-based CBIR.

CBIR dengan parameter warna adalah salah satu metode CBIR yang paling dasar dan penting karena fitur warna adalah hal yang paling intuitif dan jelas dari suatu gambar. Hal tersebut juga merupakan fitur yang penting dari persepsi gambar. Dibandingkan dengan fitur lainnya seperti tekstur atau bentuk, fitur warna sangat stabil. Disamping itu, perhitungan menggunakan fitur warna relatif lebih simpel dibandingkan fitur gambar lainnya. Dalam pembuatan CBIR dengan parameter warna, RGB dari setiap pixel akan dikonversi ke dalam bentuk HSV. Nilai HSV tersebut kemudian akan digunakan mengisi histogram warna. Histogram warna adalah frekuensi statistik untuk warna-warna berbeda pada suatu ruang warna tertentu. Pada tugas ini, suatu gambar dibagi menjadi 4x4 blok, sehingga gambar tersebut akan memiliki 16 histogram secara total. Setiap histogram atas 14 elemen dimana nilai HSV dari tiap pixel akan digunakan untuk mengisi histogram tersebut. Proses ini dilakukan terhadap semua gambar pada *dataset* dan pada image *query*. Pencocokan kemudian akan dilakukan dengan melakukan kalkulasi *cosine similarity* terhadap setiap blok yang

bersesuaian antara dua gambar. Hasil akhir merupakan rata-rata dari *cosine similarity* semua blok.

Dalam pembuatan CBIR dengan parameter tekstur, gambar dengan channel RGB pertama harus diubah menjadi grayscale. Setelah itu, dibuat Gray Level Co-occurrence Matrix dengan suatu parameter Δx dan Δy. Kemudian, untuk setiap parameter Δx dan Δy, akan dihitung 5 fitur tekstur, yaitu contrast, homogeneity, entropy, dissimilarity, dan *angular second moment (ASM)*. Lima fitur tersebut akan menjadi sebuah 'signature' dari sebuah gambar pada dataset. Proses ini akan dilakukan untuk setiap gambar pada dataset yang user masukkan.

Dalam proses image retrieval, user akan diminta untuk memasukkan sebuah gambar yang akan dijadikan query. Kemudian, gambar tersebut akan melalui proses yang telah dijelaskan pada paragraf sebelumnya, yang akan menghasilkan 5 fitur tekstur gambar input. Kelima fitur tersebut kemudian akan diukur tingkat 'kesamaannya' dengan seluruh fitur gambar pada dataset. Gambar yang memiliki tingkat 'kesamaan' di atas 60% akan ditampilkan di laman web.

2.2 Pengembangan website

Pengembangan suatu website adalah proses terstruktur yang melibatkan berbagai tahapan yang masing-masing sangat berpengaruh terhadap keberhasilan keseluruhan proyek tersebut. Hal ini dimulai dengan fase perencanaan yang cermat, dimana tujuan dan sasaran web didefinisikan dengan jelas. Memahami target pengguna juga sangat penting pada tahap ini karena hal tersebut akan digunakan untuk merancang desain, konten, dan fungsionalitas website.

Setelah proses perencanaan selesai, langkah berikutnya adalah mendapatkan nama domain dan memilih penyedia hosting yang handal untuk menyimpan dan menyajikan website. Nama domain harus unik dan mudah diingat, sedangkan penyedia hosting harus menawarkan infrastruktur yang diperlukan agar situs berfungsi tanpa masalah.

Desain memainkan peran kunci dalam proses pengembangan situs web. Ini melibatkan pembuatan wireframe atau tata letak dasar untuk menentukan struktur situs. Elemen desain visual seperti warna, tipografi, dan gambar kemudian dikembangkan untuk menciptakan pengalaman pengguna yang menarik dan kohesif. Desain responsif penting untuk memastikan bahwa situs web dapat diakses dan menarik secara visual di berbagai perangkat dan ukuran layar.

Pengembangan sebenarnya dari website dibagi menjadi 2, yaitu front-end dan back-end. Pengembangan front-end membangun website pada sisi klien yang umumnya menggunakan HTML, CSS dan Javascript untuk membuat antarmuka pengguna dan menangani interaksi pengguna. Pengembangan back-end berfokus pada sisi server, mengelola basis data, logika server, dan fungsionalitas back-end lainnya menggunakan bahasa seperti PHP, Python, Ruby, Node.Js, dan basis data seperti MySQL atau MongoDB.

Penciptaan konten adalah upaya paralel selama pengembangan website yang melibatkan generasi dan organisasi teks, gambar, video, dan elemen multimedia lainnya yang akan ditampilkan di situs web. Pengujian adalah tahap penting untuk memastikan fungsionalitas situs web, kompatibilitas di berbagai browser dan perangkat, serta kinerja keseluruhan.

Setelah suatu website sudah dilakukan tes secara menyeluruh, website tersebut siap untuk diluncurkan atau di-*deploy*. Hal ini dilakukan dengan membuat situs web dapat diakses secara publik dengan mengkonfigurasi domain, mengunggah file ke server hosting, dan memastikan bahwa semuanya diatur dengan benar. Pascal peluncuran, pemeliharaan berkelanjutan terhadap suatu website sangatlah penting. Hal ini melibatkan pemantauan masalah, pembaruan konten, penerapan *patch* keamanan, penerapan strategi cadangan, dan peningkatan sumber daya atau fitur sesuai kebutuhan.

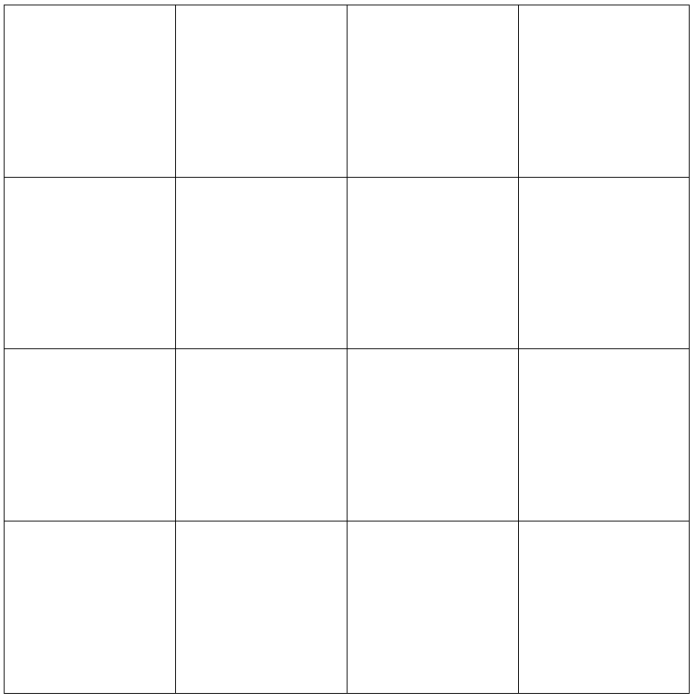
BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah-langkah pemecahan masalah

3.1.1 CBIR dengan parameter warna

Langkah-langkah pembuatan CBIR dengan parameter warna adalah sebagai berikut:

1. Bagi suatu gambar menjadi blok 4 x 4 sehingga secara total, terdapat 16 blok berbeda. Lakukan hal yang sama untuk gambar dengan ukuran panjang tidak sama dengan ukuran lebar, dimana blok pada ujung gambar sedikit lebih besar dibandingkan blok-blok lainnya. Setiap blok akan memiliki histogram warnanya sendiri.



2. Selanjutnya, kita akan mengkonversi nilai RGB dari suatu gambar ke dalam bentuk HSV. Nilai dari RGB harus dinormalisasi dengan mengubah nilai *range* [0, 255] menjadi [0, 1]

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

3. Nilai normalisasi RGB tersebut digunakan untuk mencari *Cmax, Cmin*, dan Δ sesuai dengan persamaan berikut

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

4. Selanjutnya hasil perhitungan di atas, digunakan untuk mendapatkan nilai dari HSV sesuai dengan aturan berikut.

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

5. Nilai HSV dari tiap-tiap pixel akan digunakan untuk mengisi histogram sesuai dengan aturan berikut.

$$H = \begin{cases} 0 & h \in [316, 360] \\ 1 & h \in [1, 25] \\ 2 & h \in [26, 40] \\ 3 & h \in [41, 120] \\ 4 & h \in [121, 190] \\ 5 & h \in [191, 270] \\ 6 & h \in [271, 295] \\ 7 & h \in [295, 315] \end{cases}$$
$$S = \begin{cases} 0 & s \in [0, 0.2) \\ 1 & s \in [0.2, 0.7) \\ 2 & s \in [0.7, 1] \end{cases}$$
$$V = \begin{cases} 0 & v \in [0, 0.2) \\ 1 & v \in [0.2, 0.7) \\ 2 & v \in [0.7, 1] \,. \end{cases}$$

Perhatikan bahwa, histogram tersebut dapat dianggap sebagai vektor dalam R^{14} dimana vektor H = (h0, h1, h2, h3, h4, h5, h6, h7, s0, s1, s2, v0, v1, v2)

dimana tiap elemen vektor adalah banyak kemunculan dari H, S, dan V yang bersesuaian.

6. Untuk melakukan perbandingan antara *query image* dan *image* di *dataset*, lakukan perhitungan dengan menggunakan *cosine similarity* pada setiap blok yang bersesuaian sesuai dengan persamaan berikut.

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

7. Hasil akhir *similarity* dari dua gambar didapat dengan mengambil rata-rata dari *cosine similarity* pada setiap blok yang bersesuaian.

3.1.2 CBIR dengan parameter tekstur

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

1. Ubah warna gambar dari channel RGB menjadi grayscale. Hal ini dilakukan karena warna tidaklah penting dalam penentuan tekstur. Warna grayscale Y diperoleh dari suatu piksel dengan komponen merah R, hijau G, dan biru B dengan rumus berikut:

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

2. Buatlah Gray Level Co-occurrence Matrix (GLCM). Karena nilai grayscale berada di rentang 0–255, diperlukan matriks dengan ukuran 256x256. Dalam perhitungan juga diperlukan nilai offset pada sumbu x (Δx) dan sumbu y (Δy). Dalam implementasi, digunakan 5 nilai offset, yaitu $(\Delta x, \Delta y) \in \{(0,1), (-1,1), (-1,0), (-1,-1), (0,-1)\}$. Kemudian, nilai pada baris i kolom j matriks GLCM akan dihitung dengan rumus:

   $$P_{\Delta x,\Delta y}(i,j) = \sum_{p=1}^{n}\sum_{q=1}^{m}\{1,\ jika\ I(p,q) = i\ dan\ I(p+\Delta x,\ q+\Delta y) = j \mid 0\ lainnya\}$$

   dengan n sebagai tinggi gambar, m sebagai lebar gambar, P matriks GLCM, dan I(i,j) menandakan nilai piksel grayscale di baris i dan kolom j.

   Perhitungan ini akan dilakukan 5 kali untuk tiap nilai $(\Delta x, \Delta y)$.

3. Untuk tiap GLCM, akan dihitung 5 fitur tekstur, yaitu contrast, homogeneity, entropy, dissimilarity, dan *angular second moment*. Fitur-fitur tersebut dihitung dengan rumus:
    a. Contrast

    $\sum_{i,j=0}^{D-1} P_{i,j}(i-j)^2$, dengan D adalah dimensi nilai grayscale

    b. Homogeneity

    $\sum_{i,j=0}^{D-1} (P_{i,j})/(1+(i-j)^2)$

    c. Entropy

    $\sum_{i,j=0}^{D-1} P_{i,j} \times (-\ln P_{i,j})$, dengan ln menandakan logaritma natural

    d. Dissimilarity

    $\sum_{i,j=0}^{D-1} P_{i,j} \times |i-j|$

    e. Angular second moment (ASM)

    $\sum_{i,j=0}^{D-1} P_{i,j}^{\ 2}$

4. Untuk setiap gambar, ada 5 nilai offset yang masing-masing memiliki 5 fitur berbeda. Jadi, setiap gambar memiliki 25 ‘signature’ yang dapat digunakan dalam proses CBIR. Dua puluh lima fitur tersebut kemudian akan disimpan ke dalam sebuah cache .json yang akan digunakan dalam perhitungan query.

3.1.3 Justifikasi penggunaan 2 fitur tambahan dalam CBIR dengan parameter tekstur

Perlu diperhatikan bahwa dalam implementasi CBIR dengan parameter tekstur, digunakan 2 fitur tambahan yang tidak terdapat dalam spesifikasi, yaitu dissimilarity dan angular second moment. Hal ini dilakukan supaya proses CBIR dengan parameter tekstur makin akurat.

Dissimilarity adalah fitur tekstur yang mirip dengan contrast. Perbedaannya terletak pada laju peningkatan bobot pada nilai GLCM. Pada fitur contrast, bobot GLCM meningkat secara kuadratik, sedangkan pada fitur dissimilarity, bobot GLCM meningkat secara linear.

Angular second moment (ASM) merepresentasikan tingkat keteraturan sebuah gambar. Semakin teratur sebuah gambar, pasangan nilai yang ada akan muncul

semakin sering dalam matriks GLCM. Sehingga, semakin teratur gambar, semakin besar nilai ASM-nya.

Gambar 1 Ilustrasi perbedaan gambar yang teratur dan tak teratur

3.2 Unsur aljabar geometri dalam proses pemecahan masalah

3.2.1 Unsur aljabar geometri dalam CBIR dengan parameter warna

Unsur aljabar geometri pada *color-based* CBIR adalah pada histogram warna. Histogram warna dalam hal ini adalah frekuensi kemunculan HSV pada rentang tertentu (perhitungan histogram secara lebih lengkap sudah dibahas pada bagian 3.1.1). Perhatikan bahwa, histogram warna tersebut dapat dianggap sebagai sebuah vektor dalam R^{14} karena memiliki 14 elemen. Selain itu, perkalian *dot product* antara dua vektor histogram warna pada perhitungan *cosine similarity* juga merupakan unsur aljabar geometri yang digunakan pada tugas ini.

3.2.2 Unsur aljabar geometri dalam CBIR dengan parameter tekstur

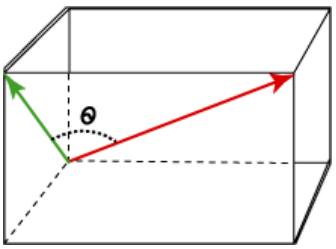
Unsur aljabar geometri muncul dalam proses kueri gambar. Seperti yang telah dijelaskan sebelumnya, 25 fitur gambar kueri akan dihitung 'kemiripannya' dengan 25 fitur pada tiap gambar dalam dataset. Dua puluh lima fitur tersebut dapat diperlakukan sebagai vektor dalam R^{25} . Hal ini dilakukan supaya kuantifikasi 'kemiripan' dapat dilakukan dengan mudah. Menghitung 'kemiripan' dua buah vektor dapat dilakukan dengan metode cosine similarity. Cosine similarity dari 2 vektor A dan B dihitung dengan rumus:

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

Dalam implementasinya, perhitungan kemiripan akan menggunakan Z-score dari tiap elemen vektor. Hal ini dilakukan supaya setiap komponen vektor memiliki

bobot yang sama dalam menentukan nilai ‘kemiripan’. Perhitungan Z-score tiap elemen vektor menggunakan rumus:

$H^{(i,j)'} = (H^{i,j} - \mu_j)/\sigma_j$, dengan σ_j menandakan standar deviasi dari komponen j, serta μ_j menandakan rata-rata seluruh nilai komponen j



Gambar 2 Ilustrasi sudut antara 2 vektor

3.3 Contoh ilustrasi kasus dan penyelesaiannya

Misalkan diberikan dua vektor dalam R^5 A dan B, dengan A = (0.5, 0.2, 295.1, 3.4, 8.75) dan B = (1.1, 2.3, 295.1, 15.0, 15.0). Maka, cosine similarity dari kedua vektor tersebut dihitung dengan:

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

Pertama-tama, akan dihitung nilai dot product dari A dan B

$A \cdot B = 0.5 * 1.1 + 0.2 * 2.3 + 295.1^2 + 15 * 3.4 + 15 * 8.75 = 87267.27$

Kemudian, akan dihitung panjang vektor A dan B

$||A|| = \sqrt{0.5^2 + 0.2^2 + 295.1^2 + 3.4^2 + 8.75^2} = 87172.4225$

$||B|| = \sqrt{1.1^2 + 2.3^2 + 295.1^2 + 15^2 + 15^2} = 87540.51$

Maka, didapat nilai $cos\theta$ adalah

$cos\theta = 87267.27/(87172.4225 * 87540.51) = 1.143 * 10^{-5}$

Karena nilai $cos\theta$ mendekati 0, maka dapat disimpulkan $\theta \approx \pi/2$. Dapat disimpulkan bahwa sudut kedua vektor mendekati 90°.

BAB IV
IMPLEMENTASI DAN UJI COBA

4.1 Implementasi dalam program utama

4.1.1 Pseudocode color.cpp

```
img_to_color_vector(file)
width <- 0
height <- 0
idxCnt <- 0
img[][] <- load_img(file, width, height)
tempRes[16][14] <- [0]
w[5] <- [0, width/4, width/2, width*3/4, width]
h[5] <- [0, height/4, height/2, height*3/4, height]
for wi ∈ [0, 4) do
     for hi ∈ [0, 4) do
          for i ∈ [w[wi], w[wi + 1]) do
               for j ∈ [h[hi], h[hi + 1]) do
                    R <- img[i][j].red/255
                    G <- img[i][j].green/255
                    B <- img[i][j].blue/255
                    cMax <- max(R, G, B)
                    cMin <- min(R, G, B)
                    delta <- cMax - cMin
                    if delta = 0 then
                         H <- 0
                    end-if
                    else if cMax = R then
                         H <- 60 * ((G - B) mod 6)/delta
                    end-if
                    else if cMax = G then
                         H <- 60 * (B - R)/delta
                    end-if
                    else
                         H <- 60 * (R - G)/delta
                    end-if
                    if cMax = 0 then
                         S <- 0
                    end-if
                    else
                         S <- delta/cMax
                    end-if
```

```
      V <- cMax
if  316 <= H and H <= 360 then
      idxH <- 0
end-if
else if 0 <= H and H < 26 then
      idxH <- 1
end-if
else if 26 <= H and H < 41 then
      idxH <- 2
end-if
else if 41 <= H and H < 121 then
      idxH <- 3
      end-if
else if 121 <= H and H < 191 then
      idxH <- 4
end-if
else if 191 <= H and H < 271 then
      idxH <- 5
end-if
else if 271 <= H and H < 296 then
      idxH <- 6
end-if
else if 296 <= H and H < 316 then
      idxH <- 7
end-if
if 0 <= S and S < 0.2 then
      idxS <- 0
end-if
else if 0.2 <= S and S < 0.7 then
      idxS <- 1
end-if
else if 0.7 <= S and S <= 1 then
      idxS <- 2
end-if
if 0 <= V and V < 0.2 then
      idxV <- 0
end-if
else if 0.2 <= V and V < 0.7 then
      idxV <- 1
end-if
else if 0.7 <= V and V <= 1 then
      idxS <- 2
```

```
                        end-if
                              tempHist[idxCnt][idxH]++
                              tempHist[idxCnt][8+idxS]++
                              tempHist[idxCnt][11+idxV]++
                        end-for
                  idxCnt++
            end-for
      end-for
end-for
return tempHist

main()
datasetPath <- input()
res[][][] <- [][][]
for file ∈ {all_files in datasetPath} do
      res.push_back(img_to_color_vector(file))
end-for
jsonObj j
j.open("color.json")
j.write(res)
j.close()
```

4.1.2 Pseudocode texture.cpp

```
img_to_vector(file)
hist[256][256] <- [][]
channel <- 256
width <- 0
height <- 0
img[][] <- load_img(file, width, height)
if img = null then
   output("error")
   exit
end-if
grayscale_img[][] <- [][]
for i ∈ [0, height) do
     for j ∈  [0, width) do
          R <- img[i][j].red
          G <- img[i][j].green
          B <- img[i][j].blue
          grayscale_img[i][j] <- 0.29*R + 0.587*G + 0.114*B
     end-for
end-for
```

```
dxs[] <- [0, -1, -1, -1, 0]
dys[] <- [1, 1, 0, -1, -1]
ds <- [1]
ans[] <- []
for d ∈ ds do
     for k ∈ [0, dxs.length) do
          dx <- d * dxs[k]
          dy <- d * dys[k]
          for c1 ∈ [0, channel) do
               for c2 ∈ [0, channel) do
                    hist[c1][c2] <- 0.0
               end-for
          end-for
          cnt <- 0.0
          for i ∈ [-dx, height) do
               for j ∈ [max(0,-dy), min(width, width-dy))
do
                    vn <- graycale_img[i][j]
                    vnx <- grayscale_img[i + dx][j + dy]
                    hist[vn][vnx] <- hist[vn][vnx] + 1.0
                    hist[vnx][vn] <- hist[vnx][vn] + 1.0
                    cnt <- cnt + 2.0
               end-for
          end-for
          for i ∈ [0, channel) do
               for j ∈ [0, channel) do
                    hist[i][j] <- hist[i][j] / cnt
               end-for
          end-for
          cont <- 0.0
          homog <- 0.0
          ent <- 0.0
          diss <- 0.0
          asm <- 0.0
          for i ∈ [0, channel) do
               for j ∈ [0, channel) do
                    cont <-cont + hist[i][j] * (i-j) * (i-j)
                    homog<-homog+hist[i][j] /(1+(i-j)*(i-j))
                    if hist[i][j] ≠ 0 then
                         ent<-ent-hist[i][j]*ln(hist[i][j])
                    end-if
                    diss<-diss+hist[i][j]*abs(i-j)
```

```
                    asm <- asm + hist[i][j] * hist[i][j]
                end-for
            end-for
            ans.push_back(cont)
            ans.push_back(homog)
            ans.push_back(ent)
            ans.push_back(diss)
            ans.push_back(asm)
      end-for
end-for
return ans

main()
datasetPath <- input()
res[][] <- [][] // dynamic array
for file ∈ {all_files in datasetPath} do
      res.push_back(img_to_vector(file))
end-for
jsonObject j // json untuk cache
j.open(“texture.json”)
j.write(res) // menulis data ke cache
j.close()
```

4.1.3 Pseudocode cmpcolor.cpp

```
cosineSim(histData, histQuery)
res <- 0
for i ∈ [0, 16) do
   sum <- 0
   aSum <- 0
   bSum <- 0
   for j ∈ [0, 14) do
      sum <- sum + histData[i] * histQuery[i]
      aSum <- aSum + histData[i] * histData[i]
      bSum <- bSum + histQuery[i] * histQuery[i]
   end-for
   res += sum / (sqrt(aSum) * sqrt(bSum)
end-for
return res/16

main
```

```
histQuery[] <- img_to_color_vector(imgQueryFile)
res[] <- []
jsonObject cacheFile
cacheFile.read("color.json")
for i ∈ [0, cacheFile.length) do
     currSim <- cosineSim(cacheFile.data[i].vector,
histQuery) * 100
     if currSim > 60 then
          res.push_back({cacheFile.data[i].name, currSim})
     end-if
end-for
sort(res) // sort result based on similarity
jsonObject resultFile
resultFile.open("colorresult.json")
resultFile.write(res)
resultFile.close()
```

4.1.4 Pseudocode cmptexture.cpp

```
img_to_vector(file)
hist[256][256] <- [][]
channel <- 256
width <- 0
height <- 0
img[][] <- load_img(file, width, height)
if img = null then
     output("error")
     exit
end-if
grayscale_img[][] <- [][]
for i ∈ [0, height) do
     for j ∈  [0, width) do
          R <- img[i][j].red
          G <- img[i][j].green
          B <- img[i][j].blue
          grayscale_img[i][j] <- 0.29*R + 0.587*G + 0.114*B
     end-for
end-for
dxs[] <- [0, -1, -1, -1, 0]
dys[] <- [1, 1, 0, -1, -1]
ds <- [1]
ans[] <- []
for d ∈ ds do
```

```
     for k ∈ [0, dxs.length) do
          dx <- d * dxs[k]
          dy <- d * dys[k]
          for c1 ∈ [0, channel) do
               for c2 ∈ [0, channel) do
                    hist[c1][c2] <- 0.0
               end-for
          end-for
          cnt <- 0.0
          for i ∈ [-dx, height) do
               for j ∈ [max(0,-dy), min(width, width-dy))
do
                    vn <- graycale_img[i][j]
                    vnx <- grayscale_img[i + dx][j + dy]
                    hist[vn][vnx] <- hist[vn][vnx] + 1.0
                    hist[vnx][vn] <- hist[vnx][vn] + 1.0
                    cnt <- cnt + 2.0
               end-for
          end-for
          for i ∈ [0, channel) do
               for j ∈ [0, channel) do
                    hist[i][j] <- hist[i][j] / cnt
               end-for
          end-for
          cont <- 0.0
          homog <- 0.0
          ent <- 0.0
          diss <- 0.0
          asm <- 0.0
          for i ∈ [0, channel) do
               for j ∈ [0, channel) do
                    cont <-cont + hist[i][j] * (i-j) * (i-j)
                    homog<-homog+hist[i][j] /(1+(i-j)*(i-j))
                    if hist[i][j] ≠ 0 then
                         ent<-ent-hist[i][j]*ln(hist[i][j])
                    end-if
                    diss<-diss+hist[i][j]*abs(i-j)
                    asm <- asm + hist[i][j] * hist[i][j]
               end-for
          end-for
          ans.push_back(cont)
          ans.push_back(homog)
```

```
            ans.push_back(ent)
            ans.push_back(diss)
            ans.push_back(asm)
      end-for
end-for
return ans

cmpGreater(pa, pb)
return pa.second > pb.second

vectorlength(x[])
sum <- 0.0
for elem in x do
      sum <- sum + elem * elem
end-for
ans <- sqrt(sum)
return ans

cossinesim(xa[], xb[], mean[], sigma[])
sum <- 0.0
a <- []
b <- []
for i in [0, xa.length) do
      a[i] <- (xa[i] - mean[i]) / sigma[i]
      b[i] <- (xb[i] - mean[i]) / sigma[i]
end-for
for i in [0, a.length) do
      sum <- sum + a[i] * b[i]
end-for
lena <- vectorlength(a)
lenb <- vectorlength(b)
ans <- sum / (lena * lenb)
return ans
main()
var[] <- []
sum[] <- []
mean[] <- []
sigma[] <- []
indexedImage[] <- []
for file ∈ {all_files in imagePath} do
      indexedImage <- img_to_vector(file)
end-for
```

```
jsonObject file
j.read("texture.json")
cossim[] <- []
for i ∈ [0, j.length) do
     done[] <- from_json(j[i]) // ubah jsonObject jd array
     for k ∈ [0, done.length) do
          sum[k] <- sum[k] + done[k]
     end-for
     nameFile[i] <- getFileName(j[i])
end-for
for k ∈ [0, 25) do
     mean[k] <- sum[k] / j.length
end-for
for i ∈ [0, j.length) do
     done[] <- from_json(j[i])
     for k ∈ [0, done.length) do
          var[k]<-var[k]+(done[k]-mean[k])*(done[k]-mean[k])
     end-for
end-for
for k ∈ [0, 25) do
     sigma[k] <- sqrt(var[k] / (j.length - 1))
end-for
for i ∈ [0, j.length) do
     done <- from_json(j[i])
     val <- cossinesim(done, indexedImage, mean, sigma)
     cossim.push_back(make_pair(nameFile[i], val))
end-for
sort(cossim, key=cmpGreater)
jsonObject jres
jres.open("textureresult.json")
jres.write(cossim)
jres.close()
```

4.1.5 Pseudocode webscraper.py

```
// global variabel
URL <- input()
pathToFolder <- input()
numOfThread <- int(input()) // jumlah thread, minimal 1
images <- []
iterator <- 1
```

```
default <- ".jpg"
aliases <- ["src", "data-src"]
allowedFormats <- [".jpg", ".png", ".jpeg"] // format yang
dibolehin

find_image_format(path):
list_dir <- path.strip().split('/') // split string menjadi
array
last_path <- list_dir.back // elemen terakhir dari array
i <- last_path.length - 1
found <- false
while i >= 0 and not found do
     if last_path[i] = '.' then
          found <- True
          break
     end-if
     i <- i - 1
end-while
ans <- “”
while found and i < len(last_path) do
     if last_path[i] = '?' or last_path[i] = '#' then
           break
     end-if
     ans <- concat(ans, last_path[i])
     i <- i + 1
end-while
return ans

main()
try: // try-except
     options <- webdriver.FirefoxOptions()
     options.add_argument("--headless")
     driver <-  webdriver.Firefox(options=options)
     driver.get(URL)
     scroll(driver) // scroll website ke akhir halaman
     images <- driver.find_elements(By.TAG_NAME, 'img')
     curLen <- len(images)

     for ind ∈ [0, curLen) do
          try:
               img <- images[ind]
               source <- null
```

```
                for alias ∈ aliases do
                    if img.get_attribute(alias) ≠ null then
                        source <- img.get_attribute(alias)
                        break
                    end-if
                end-for
                if source = null or "http" not in source then
                    continue
                end-if
                img_data <- requests.get(source).content
                img_format <- find_image_format(source)
                if img_format = "":
                    img_format <- default
                end-if

                file_name <- concat(str(iterator),img_format)
                File file
                real_path <- concat(pathToFolder, file_name)
                handler<-file.open(real_path)
                handler.write(img_data)
                iterator <- iterator + 1
            except Exception as e:
                continue // pass error
        end-for

    except Exception as e:
            pass // ERROR DIBIARIN
```

4.2 Penjelasan struktur program

4.2.1 Arsitektur kode

Dalam implementasi, proses pembuatan cache dilakukan dengan banyak thread dengan bantuan modul multithreading. Hal ini dilakukan supaya proses CBIR berjalan seefisien mungkin, mempercepat waktu eksekusi program.

Selain itu, kode yang dibuat telah dipastikan memory-safe dan thread-safe. Hal ini dimungkinkan dengan atribut mutex pada library threading C++. Selain itu, operasi dealokasi memori juga meningkatkan reliability program, mengingat C++ tidak memiliki *automatic garbage collector*.

Untuk proses web scraping, kode python telah dipastikan dapat memuat dan mengunduh seluruh gambar pada sebuah halaman, bahkan gambar yang dimuat

dengan teknik ‘lazy-loading’. Kode python tersebut juga telah didesain untuk mengunduh seluruh gambar yang memiliki format .jpg, .png, serta .jpeg. Dalam implementasinya proses web scraping juga menggunakan banyak thread dengan bantuan modul multithreading. Akses memori juga dipastikan thread-safe dengan atribut Lock pada modul threading di Python. Demi kenyamanan pengguna, web driver telah dikonfigurasi dengan opsi “--headless”, yang menjadikan browser tidak terbuka di layar pengguna.

Perlu diperhatikan juga bahwa detail implementasi seperti konversi json ke array, pengelolaan memori, serta penulisan file json ke direktori tidak dimasukkan ke dalam pseudocode. Hal ini dikarenakan proses tersebut tidak berpengaruh pada proses utama secara keseluruhan. Untuk detail lengkapnya, pembaca disarankan untuk membaca file yang bersangkutan pada github release yang telah dilampirkan di daftar pustaka.

4.3 Tata cara penggunaan program

Setelah pengguna masuk ke halaman web, maka pengguna akan diarahkan untuk memencet tombol “Try Now” pada halaman Home. Setelah itu, pengguna akan diarahkan untuk memilih opsi upload dataset. Pengguna dapat meng-upload dataset secara manual, atau memasukkan link sebuah webpage berisi gambar untuk memilih opsi web-scraping. Jika pengguna memilih opsi web-scraping, maka seluruh elemen HTML dengan tag <img> akan dimasukkan ke dalam dataset.

Setelah itu, pengguna akan diminta untuk memilih parameter CBIR. Terdapat dua parameter CBIR yang diberikan, yaitu parameter tekstur dan warna. Kemudian, pengguna akan diminta untuk memasukkan gambar kueri. Gambar kueri ini akan dicocokkan dengan seluruh gambar pada dataset dengan metode CBIR dengan parameter yang telah dipilih. Seluruh gambar yang memiliki tingkat kemiripan di atas 60% akan ditampilkan pada web dengan pagination yang sesuai. Pada setiap gambar juga akan disertakan tingkat kemiripan dengan gambar kueri (0-100%).

4.4 Hasil pengujian

Akan diuji image yang ada pada dataset.

Hasilnya adalah sebagai berikut:

4.5 Analisis desain solusi algoritma

Dari hasil pengujian di atas, dapat dilihat bahwa parameter warna memiliki performa lebih baik jika fokus gambar ada pada objek yang memiliki satu warna dominan, seperti api, lautan (body of water), dan pepohonan. Sedangkan parameter tekstur memiliki performa lebih baik jika fokus gambar ada pada objek yang memiliki tekstur unik, seperti tanah kering, close-up tembok, serta close-up lukisan.

Hasil ini tentunya logis dan menunjukkan bahwa implementasi algoritma telah berjalan dengan tepat. CBIR dengan parameter warna menggunakan histogram kemunculan warna untuk menghitung kemiripan 2 gambar. Jika objek utama gambar memiliki channel warna dominan (red, green, ataupun blue), algoritma akan mencari gambar pada dataset yang memiliki warna channel dominan yang sesuai. Sedangkan, CBIR dengan parameter tekstur menggunakan 5 karakteristik permukaan dalam menghitung kemiripan 2 gambar. Dua tekstur yang serupa akan memiliki 5 karakteristik yang serupa pula, menjadikan performa CBIR dengan parameter tekstur baik digunakan untuk gambar yang memiliki tekstur unik.

BAB V
PENUTUP

5.1 Kesimpulan

Content-Based Image Retrieval dengan parameter warna dan tekstur dapat digunakan dalam mencari citra gambar yang serupa pada sebuah dataset. Proses pencarian diawali dengan pengguna mengunggah sebuah dataset gambar. Kemudian, pengguna akan diminta untuk mengunggah gambar yang akan dijadikan kueri terhadap dataset. Terakhir, pengguna memilih antara dua parameter CBIR: warna dan tekstur. Program lalu akan menjalankan algoritma CBIR sesuai parameter yang telah dipilih, dan kemudian menampilkan seluruh gambar dengan kemiripan di atas 60%.

Algoritma CBIR dengan parameter warna baik digunakan pada gambar yang memiliki warna dominan, seperti gambar api, laut, dan pepohonan. Hal ini dikarenakan pada histogram warna, akan ada selisih yang besar antara suatu warna dengan warna lain. Selisih yang besar membuat algoritma makin sensitif terhadap *false-positives*.

Algoritma CBIR dengan parameter tekstur baik digunakan pada gambar yang memiliki tekstur unik dan jelas, seperti tanah kering, close-up tembok, serta close-up lukisan. Pada algoritma CBIR dengan parameter tekstur, sebuah gambar akan "dikompresi" menjadi 5 tekstur utama, yaitu contrast, homogeneity, entropy, dissimilarity, dan angular second moment. Gambar yang memiliki tekstur yang jelas cenderung akan memiliki fitur tekstur yang jelas pula, menjadikan algoritma CBIR dengan parameter tekstur cocok digunakan.

5.2 Saran

Saran yang ingin penulis sampaikan dari tugas besar yang telah dilakukan adalah sebagai berikut:

1. Untuk asisten IF2123 Aljabar Linier dan Geometri, supaya tugas besar yang diberikan berikutnya lebih relevan dengan materi IF2123 Aljabar Linier dan Geometri yang diajarkan di kelas.
2. Untuk pembaca yang ingin mengimplementasikan CBIR, supaya berhati-hati dalam melakukan floating-point calculation, mengingat jumlah memori yang terbatas. Baca juga makalah serta riset yang terkait dengan CBIR supaya implementasi makin akurat.

5.3 Komentar atau Tanggapan

Content-Based Image Retrieval adalah algoritma yang cukup efisien dalam mencari dan melakukan kueri terhadap dataset. Algoritma CBIR secara fundamental mengambil fitur-fitur utama dari sebuah gambar, dan kemudian mencocokannya dengan fitur dari gambar kueri. Pengambilan fitur inilah yang membuat algoritma ini berjalan dengan cepat, karena program tidak perlu mengecek nilai setiap pixel dari kedua gambar.

5.4 Refleksi terhadap tugas besar

Hal paling utama yang dapat kami pelajari dari tugas besar ini adalah pentingnya mencicil tugas jauh sebelum deadline. Tugas besar ini juga telah mengajari penulis pentingnya komunikasi. Tugas besar ini telah memperluas wawasan penulis mengenai algoritma CBIR dengan parameter tekstur dan warna. Tugas besar ini juga telah membuat penulis semakin mahir dalam pengembangan website.

5.5 Ruang perbaikan atau pengembangan

Penulis menyadari bahwa hasil pengerjaan tugas besar ini masih belum maksimal. Hal-hal yang dapat dikembangkan dari pengerjaan tugas besar ini yaitu:

1. Pembuatan website menjadi lebih responsif yang akan menjadikan website semakin user-friendly dan mudah digunakan
2. Penggunaan multiprocessing sebagai ganti multithreading, baik pada pembuatan cache maupun web-scraping, yang memungkinkan komputasi secara paralel.
3. Pengadaan fitur bagi pengguna untuk memilih campuran parameter warna dan tekstur dalam melakukan CBIR. Hal ini tentunya akan menjadikan algoritma CBIR makin akurat.
4. Pembuatan fitur tambahan pada website yang memperkaya pengalaman pengguna, seperti fitur foto dengan webcam

DAFTAR PUSTAKA

Link repository : https://github.com/ninoaddict/Algeo02-22024

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

https://prism.ucalgary.ca/server/api/core/bitstreams/8f9de234-cc94-401d-b701-f08ceee6cfdf/content

https://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Hayes_GreyScaleCoOccurrenceMatrix.pdf

https://arxiv.org/ftp/arxiv/papers/1205/1205.4831.pdf

https://www.browserstack.com/guide/web-scraping-using-selenium-python

https://medium.com/codex/c-multithreading-the-simple-way-95aa1f7304a2

https://docs.python.org/3/library/threading.html

https://www.geeksforgeeks.org/multithreading-python-set-1/