

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

**TUGAS BESAR II**  
**ALJABAR LINEAR DAN GEOMETRI**  
**APLIKASI ALJABAR VEKTOR DALAM SISTEM TEMU BALIK**  
**GAMBAR**

DISUSUN OLEH:

13522054 - Benjamin Sihombing

13522086 - Muhammad Atpur Raffif

13522158 - Muhammad Rasheed Qais Tandjung



**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi 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.

Di dalam Tugas Besar 2 ini, 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

#### 2.1 Gambar atau Image

Gambar atau Image adalah representasi visual dari suatu benda. Pada gambar digital, gambar merupakan representasi dari sekumpulan *pixel* (*picture element*). *Picture Element* adalah bagian terkecil dari suatu gambar yang berisi informasi nilai warna dan kecerahan. Jadi, secara sederhana, gambar bisa didefinisikan sebagai matriks *pixel-pixel* yang tiap *pixel*-nya merepresentasikan informasi warna dan kecerahan pada gambar.

#### 2.2 Cosine Similarity

*Cosine Similarity* adalah suatu konsep yang digunakan untuk mencari nilai kuantitatif kemiripan suatu vektor dengan vektor lain. *Cosine Similarity* dihitung dengan membagi dot product vektor a dan vektor b dengan perkalian norma vektor a dengan vektor b. Dot product adalah hasil nilai skalar proyeksi vektor. Norma adalah panjang dari suatu vektor. Berikut adalah rumus *Cosine Similarity*:

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

#### 2.3 Content-Based Image Retrieval

*Content-Based Image Retrieval* memiliki arti pengambilan gambar berbasis konten. Seperti artinya, *Content-Based Image Retrieval* adalah suatu teknik yang digunakan untuk mengambil/memilih gambar berdasarkan isi/konten gambar itu sendiri. Konten gambar mengacu pada fitur-fitur gambar seperti warna, tekstur, dan bentuk. Gambar akan diekstrak menjadi fitur-fitur yang ada dan direpresentasikan ke dalam bentuk elemen-elemen aljabar linier dan geometri seperti vektor dan matriks.

#### 2.4 Content-Based Image Retrieval Berdasarkan Warna

Pada *Content-Based Image Retrieval* berdasarkan warna, akan dibuat sebuah *feature vector* yang dibuat berdasarkan nilai HSV setiap pixel pada gambar yang didapat setelah mengkonversikannya dari nilai RGB. Pemilihan penggunaan HSV dikarenakan deskripsi warna tersebut lebih mudah dibedakan oleh mata manusia, terutama pada kertas. Kemudian dibuat histogram frekuensi kemunculan gambar berdasarkan ukuran bin berikut:

$$H = [0, 25], (25, 40], (40, 120], (120, 190], (190, 270], (270, 295], (295, 315], (315, 360]$$

$$S = [0, 0.2], (0.2, 0.7], (0.7, 1]$$

$$V = [0, 0.2], (0.2, 0.7], (0.7, 1]$$

Setelah melakukan pengelompokan berdasarkan histogram diatas, kemudian dibentuk *feature vector* dengan mengalikan seluruh nilai kemungkinan pada bin diatas, sehingga didapatkan *vector* dengan dimensi  $8 \times 3 \times 3 = 72$ . Nantinya *vector* ini akan dibandingkan dengan target gambar yang ingin dicari kemiripannya dengan gambar lain menggunakan *cosine similarity*.

Salah satu kekurangan dari menggunakan metode ini adalah *vector* yang dihasilkan merepresentasikan gambar secara global, atau informasi mengenai lokalitas warna hilang. Membagi gambar menjadi sebuah *block* dapat mengatasi permasalahan ini. Pada implementasi, gambar akan dibagi menjadi 16 *block*, atau secara keseluruhan, gambar memiliki bentuk  $4 \times 4$  dari *block* gambar yang lebih kecil.

## 2.5 Content-Based Image Retrieval Berdasarkan Tekstur

Pada *Content-Based Image Retrieval* berdasarkan tekstur, *feature vector* didapatkan dari parameter yang dapat berupa kontras, homogenitas, ataupun entropi. Seluruh parameter ini dihitung dari *co-occurrence matrix* yang merupakan sebuah representasi frekuensi intensitas grayscale dari *pixel* yang bersebelahan dari gambar berdasarkan jarak tertentu. Nilai dari *feature vector* ini perlu dinormalisasi menggunakan *gaussian normalization*. Sama seperti sebelumnya, nantinya *feature vector* ini akan digunakan untuk mencari kemiripan gambar menggunakan *cosine similarity*.

## 2.6 Pengembangan Website

Hasil dari melakukan komparasi diatas akan ditampilkan menggunakan website. Terdapat komponen frontend dan backend dalam pengembangannya. Bagian backend bertugas untuk

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

memanggil binary yang menjalankan komparasi antara gambar. Sedangkan untuk frontend bertugas menampilkan informasi yang dihasilkan setelah melakukan komparasi menggunakan content-based image retrieval diatas.

## BAB 3

### Analisis Pemecahan Masalah

#### 3.1 Langkah-langkah Pemecahan Masalah

- a. Terdapat gambar yang ingin dicari.
- b. Mencari dataset yang diinginkan dari banyaknya gambar di dunia.
- c. Mencari gambar-gambar yang mirip dengan gambar yang dicari menggunakan suatu metode(pada tugas besar ini menggunakan metode CBIR).
- d. Membandingkan kesamaan gambar-gambar di dataset dengan gambar yang dicari.
- e. Menggunakan fitur-fitur gambar seperti warna dan tekstur pada perbandingan gambar.
- f. Mendapatkan nilai kuantitatif perbandingan setiap gambar dengan gambar yang dicari.
- g. Mengurutkan gambar-gambar sesuai kemiripan dengan gambar yang dicari.
- h. Menampilkan gambar-gambar yang mirip dengan gambar dicari dengan tingkat kemiripan >60%.

#### 3.2 Proses Pemetaan Masalah Menjadi Elemen-Elemen pada Aljabar Geometri

- a. Membandingkan gambar-gambar dengan gambar.
- b. Mengextrak gambar menjadi bentuk elemen-elemen pada Aljabar Geometri seperti matriks dan vektor agar dapat dibandingkan.
- c. Mengolah setiap gambar berdasarkan metode tekstur atau warna.
- d. Mencari nilai kuantitatif perbandingan gambar dengan gambar yang dicari.
- e. Mengurutkan perangkingan gambar-gambar berdasarkan tingkat kemiripan dengan gambar yang dicari.

#### 3.3 Contoh Ilustrasi Kasus dan Penyelesaiannya

Budi adalah seorang *digital artist*. *Digital artist* adalah seseorang yang menghasilkan suatu karya seni seperti gambar atau video dengan bantuan teknologi digital. Budi sering membuat gambar digital dan menjualnya di suatu website. Banyak gambar telah dia ciptakan dan jual. Suatu hari, Budi sedang bermain media sosial iF. Ketika dia bermain, dia menemukan suatu postingan gambar yang menarik. Namun, dia merasa gambar itu tampak tidak asing baginya. Dia merasa gambar itu mirip dengan salah satu karya. Karena merasa gambar itu tidak original dan merupakan plagiarisme dari karyanya, Budi ingin mencari tahu apakah gambar tersebut benar merupakan hasil plagiarisme. Namun, saking banyaknya gambar yang telah Budi ciptakan, Budi tidak tahu dimana

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

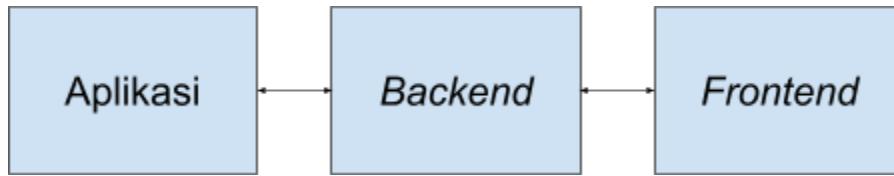
lokasi penyimpanan gambar itu berada. Nah, Untungnya, Budi mengetahui website yang bisa mencari kemiripan gambar dengan gambar lain di suatu dataset. Website itu adalah iCBIR. Budi langsung mengunggah postingan gambar iF tersebut ke dalam iCBIR dan mengunggah semua dataset hasil karyanya. Ternyata, iCBIR berhasil menunjukkan ada gambar di dataset Budi yang memiliki kemiripan dengan postingan gambar iF tersebut. Setelah mengetahui itu, Budi langsung membuat konten untuk menjatuhkan dan menghina-hina akun yang telah memposting gambar yang mirip dengan karyanya.

## BAB 4

### Implementasi dan Uji Coba

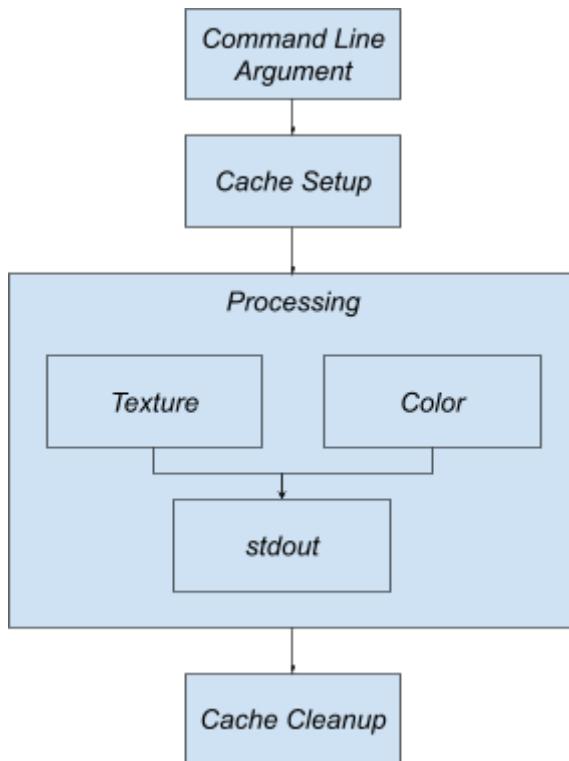
#### 4.1. Implementasi

##### a. Diagram Menyeluruh



Secara garis besar, program dibagi menjadi tiga bagian, yaitu aplikasi utama, *backend*, dan *frontend*. Pada bagian aplikasi, ditulis menggunakan bahasa *cpp*, sedangkan untuk bagian website, menggunakan *TypeScript*. Alasan dibalik pemilihan *cpp* pada program utama adalah performa.co Sedangkan untuk website dikarenakan familiaritas kami terhadap bahasa tersebut.

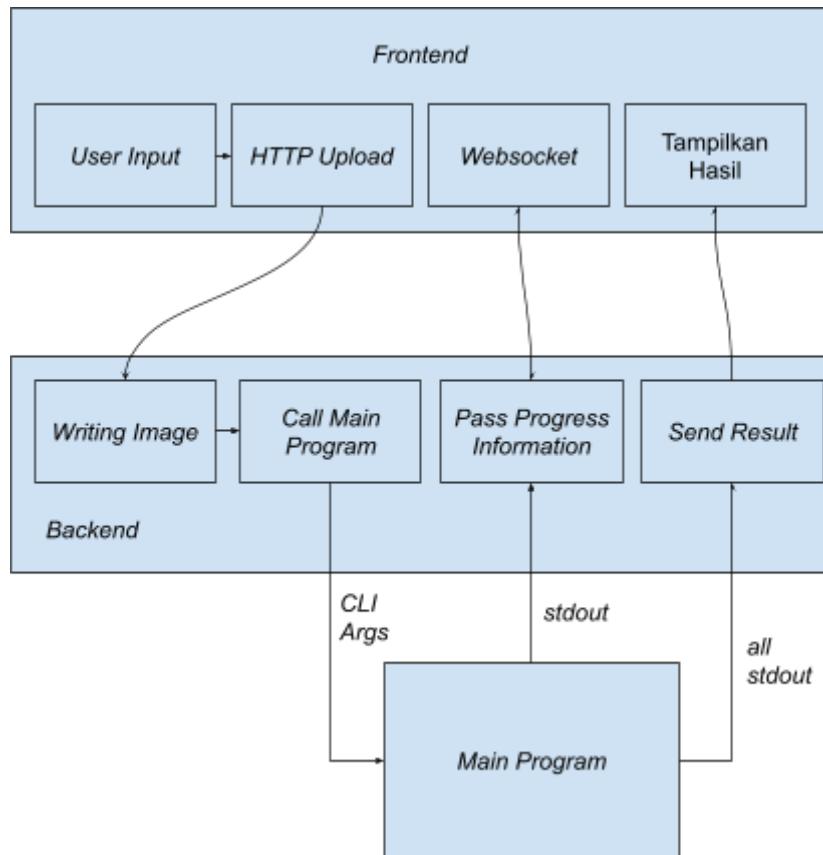
##### b. Diagram App



Pada aplikasi utama, masukan/input dilakukan melalui *Command Line Argument*. Masukan yang diminta adalah metode *CBIR* yang digunakan, *path* menuju *dataset folder*, dan *path* menuju gambar yang akan dibandingkan dengan

*dataset*. Langkah awal yang dilakukan aplikasi adalah memeriksa apakah sudah terdapat *cache*, hal ini dapat mempercepat proses berjalananya aplikasi. Kemudian masuk ke bagian *processing*, apabila *cache* tidak ditemukan, maka pada bagian ini dibentuk sebuah data yang akan disimpan pada *cache* nantinya. Struktur dari *cache* sendiri berupa *key-value pair* dengan *key* adalah nama dari file di *dataset* dan *value* yang berupa kumpulan *vector* yang dapat digunakan kembali pada *processing* selanjutnya. Seluruh hasil perbandingan yang berupa *cosine similarity* ditampilkan pada *stdout*, yang nantinya dapat diproses oleh *backend*.

c. Diagram Web



Setelah *user* melakukan *input* gambar dan *dataset*, bagian *frontend* akan mengirimkan data gambar tersebut kebagian *backend* yang bertugas menyimpan data tersebut menjadi sebuah *file*. Kemudian *backend* memeriksa apakah *cache* sudah merupakan yang terbaru, atau sudah terdapat program utama lain yang melakukan *caching* sebelumnya agar tidak terdapat dua proses *caching* secara paralel. Jika sudah ada yang berjalan, maka *backend* akan menunggu program selesai. Namun

jika *cache* sudah terbaru, maka program utama bisa dijalankan secara paralel. Lalu *backend* akan memanggil program utama dengan parameter yang sesuai. Program utama akan menghasilkan *output* yang ditulis pada *stdout*. Data ini kemudian diteruskan ke *frontend* untuk memberikan informasi mengenai proses yang berjalan.

Protokol dalam meneruskan informasi ini menggunakan *websocket*, dikarenakan keterbatasan *http* yang memiliki desain *request* dan *response* secara sekali. Sedangkan *websocket* terus membuka koneksi terhadap *client*, sehingga dapat memberikan informasi secara terus-menerus tanpa melakukan *polling* pada *client*. Setelah program utama selesai berjalan, hasilnya diteruskan pada *frontend* dan ditampilkan. Pada *frontend* sendiri menggunakan *react framework*.

#### d. *Pseudocode*

##### i. image.cpp

```
{image.hpp}
Type RGB <
    red: integer,
    green: integer,
    blue: integer
>

Type HSV <
    h: float,
    s: float,
    v: float
>

Class Image <
    pixel: pointer to integer,
    width: integer,
    height: integer,
    function getRGB(x: integer, y: integer) → HSV
    function getHSV(x: integer, y: integer) → HSV
    function getGrayscale(x: integer, y: integer) → integer
    function RGBtoHSV(value: RGB) → HSV
>

Class Block <
    function getRGB(x: integer, y: integer) → HSV
    function getHSV(x: integer, y: integer) → HSV
    image: pointer to Image,
    width: integer,
```

```
        height: integer,
>

Class ImageBlocks <
    image: pointer to Image,
    blockRow: integer,
    blockCol: integer,
    function getBlock(row: integer, col: integer) -> pointer
    to Block
>

{image.cpp}
define MAX_PATH: integer = 200
define CHANNEL: integer = 3

function Image.getGrayscale(x: integer, y: integer) ->
integer
    RGB p <- getRGB(x, y)
    integer result <- (0.29 * p.red) + (0.587 * p.green) +
(0.114 * p.blue)
    clampNumber(result, 0, 255)
    return result

function Image.getRGB(x: integer, y: integer) -> RGB
    if (x < 0) then x = 0
    if (y < 0) then y = 0
    if (x > this.width) then x = this.width - 1
    if (y > this.height) then y = this.height - 1

    integer pos <- ((y * this.width) + x) * CHANNEL
    return {this.pixel[pos], this.pixel[pos+1],
this.pixel[pos+2]}

function Image.getHSV(x: integer, y: integer) -> HSV
    return Image.RGBtoHSV(Image.getRGB(x,y))

function Image.RGBtoHSV(value: RGB) -> HSV
    float r <- value.red / 255
    float g <- value.green / 255
    float b <- value.blue / 255

    float cmax <- max(max(r, g), b)
    float cmin <- min(min(r, g), b)
    float delta <- cmax - cmin

    if (delta = 0) then h <- 0
    else if (cmax = r) then h <- mod(60 * ((g - b) / delta)
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
+ 360, 360)
    else if (cmax = g) then h <- mod(60 * ((b - r) / delta)
+ 120, 360)
    else if (cmax = b) then h <- mod(60 * ((r - g) / delta)
+ 240, 360)

    if (cmax = 0) then s <- 0
    else s <- delta / cmax

    float v <- cmax

return {h, s, v}

procedure ImageBlocks.ImageBlocks(pointer to Image,
blockRow: integer, blockCol: integer)
    this.image <- img
    this.blockRow <- blockRow
    this.blockCol <- blockCol

    this.blocks <- allocate(sizeof(pointer to Block))
integer colStep <- this.image.width / blockCol
integer rowStep <- this.image.height / blockRow

    i traversal [0..blockRow]
        j traversal [0..blockCol]
            integer idx <- i * blockRow + j

            integer startW <- j * colStep
            integer endW <- (j + 1) * colStep - 1
            integer startH <- i * rowStep
            integer endH <- (i + 1) * rowStep - 1

                if (i = blockRow - 1) then endH <
this.image.height - 1
                if (j = blockCol - 1) then endW <
this.image.width - 1

            this.blocks[idx] <- new Block(img, startW, endW,
startH, endH)

function getBlock(row: integer, col: integer) → pointer to
Block
    return this.blocks[row * this.blockRow + col]

function Block.Block(img: pointer to img, startW: integer,
```

```
endW: integer, startH: integer, endH: integer){  
    this.image = img;  
    this.x ← startW;  
    this.y ← startH;  
    this.width ← endW - startW + 1;  
    this.height ← endH - startH + 1;  
}  
  
function Block.getRGB(x: integer, y: integer) → RGB  
    return this.image.getRGB(this.x + x, this.y + y)  
  
function Block.getHSV(x: integer, y: integer) → HSV  
    return Image.RGBtoHSV(this.getRGB(x, y))
```

ii. vector.cpp

```
Class Vector: <  
public:  
    dimension: integer,  
    *component: pointer to float,  
    procedure display(),  
    function norm(Vector *vector) -> float,  
    function innerProduct(Vector *a, Vector *b) -> float,  
    function angle(Vector *a, Vector *b) -> float,  
    Vector(int dimension), {konstruktor di cpp}  
    Vector(int dimension, double *components),  
    ~Vector(); {destruktur di cpp}>  
  
class Vectors: <  
public:  
    Vectors(int size), {konstruktor di cpp}  
    ~Vectors(), {destruktur di cpp}  
    size: integer,  
    vectors: pointer to pointer to vector,  
    getAngleAverage(vs1: pointer to vector, vs2: pointer to  
vector) -> float  
>  
  
function Vector.Vector(dimension: integer) -> Vector  
Kamus Lokal  
Algoritma  
    this.dimension <- dimension  
    this.component <- allocate()  
    i traversal [0..dimension-1]  
        This.component[i]] <- component[i]
```

```
function Vector.Vector(dimension: integer, components: pointer to float) -> Vector
Kamus Lokal
Algoritma
    i traversal [0..dimension-1]
        This.component[i]] <- component[i]

procedure Vector.display()
Kamus Lokal
Algoritma
    output("[")
    i traversal [0..this.dimension-1]
        output(this.components[i])
        if(i != this.dimension-1) then
            output(",")
    output("]")

function Vector.innerProduct(vectorA: pointer to Vector,
vectorB: pointer to Vector) -> float
Kamus Lokal
    result: float
Algoritma
    if(vectorA.dimension != vectorB.dimension) then
        -> NAN
    else
        result <- 0.0
        i traversal [0..vectorA.dimension-1]
            result <- (result + vectorA.component[i] *
vectorB.component[i])
        -> result

function Vector.innerProduct(vector: pointer to Vector) ->
float
Kamus Lokal
    result: float
Algoritma
    i traversal [0..vectorA.dimension-1]
        result <- (result + vectorA.component[i] *
vectorB.component[i])
    -> result

function Vector.norm(vector: pointer to Vector) -> float
Kamus Lokal
    result: float
Algoritma
    i traversal [0..vectorA.dimension-1]
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
        result <- (result + vector.component[i] *
vector.component[i])
-> sqrt(result)

function Vector.angle(vectorA: pointer to Vector, vectorB: pointer to Vector) -> float
Kamus Lokal
    c: float
Algoritma
    if(vectorA.dimension != vectorB.dimension) then
        -> NAN
    else
        c = (Vector::innerProduct(vectorA, vectorB) /
        (Vector::norm(vectorA) * Vector::norm(vectorB)))
        clampNumber(c, 0.0, 1.0)
        -> c

function Vectors.Vectors(size: integer) -> Vectors
Kamus Lokal
Algoritma
    this.size <- size
    this.vectors <- allocate()

function Vectors.getAngleAverage(vs1: pointer to Vectors,
vs2: pointer to Vectors) -> float
Kamus Lokal
    res: float
    size, i: integer
Algoritma
    res <- 0
    size <- fmin(vs1.size, vs2.size)
    i traversal [0..size-1]
        res <- (res + Vector.angle(vs1.vectors[i],
        vs2.vectors[i]))
    res = res / size
    -> res
```

iii. cbir\_color.cpp

```
{cbir_color.hpp}
USE_Vector
USE_Image

constant BLOCK: integer = 4
```

```
{cbir_color.cpp}
USE_cbir_color

function getBinH(h: float) -> integer
Kamus Lokal
    r: integer
Algoritma
    r ← 1

    if (h > 25.0) then r ← 2
    if (h > 40.0) then r ← 3
    if (h > 120.0) then r ← 4
    if (h > 190.0) then r ← 5
    if (h > 270.0) then r ← 6
    if (h > 295) then r ← 7
    if (h > 315) then r ← 0

    return r

function getBinS(s: float) -> integer
Kamus Lokal
    r: integer
Algoritma
    r ← 0

    if (s > 0.2) then r ← 1
    if (s > 0.7) then r ← 2

    return r

function getBinV(v: float) -> integer
Kamus Lokal
    r: integer
Algoritma
    r ← 0

    if (v > 0.2) then r ← 1
    if (v > 0.7) then r ← 2

    return r

function getHSVFeatureVector(img: pointer to Image) ->
pointer to Vectors
    pointer to ImageBlocks blocks ← new ImageBlocks(img,
BLOCK, BLOCK);
    pointer to Vectors vs = new Vectors(BLOCK * BLOCK);
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
i traversal [0..(blocks.blockRow - 1)]
    j traversal [0..(blocks.blockCol - 1)]
        pointer to Vector v = new Vector(72);
        pointer to Block block = blocks.getBlock(i, j);

        k traversal [0..block.height]
            l traversal [0..block.width]
                HSV hsv ← block.getHSV(l, k)

                integer ih = getBinH(hsv.h)
                integer is = getBinS(hsv.s)
                integer iv = getBinV(hsv.v)

                v.component[(24 * iv) + (8 * is) + ih] ←
v.component[(24 * iv) + (8 * is) + ih] + 1

                vs.vectors[i * blocks.blockRow + j] ← v;

    delete blocks
    return vs;

function getColorAngle(vs1: pointer to Vectors, vs2: pointer
to Vectors) → float
    float res ← 0
    integer size ← vs1.size

    i traversal [0..size - 1]
        res ← res + Vector.angle(vs1.vectors[i],
vs2.vectors[i])

    res ← res / size
    return res
```

iv. cbir\_texture.cpp

```
constant QUANTIZATION_LEVEL: integer = 256
constant TEXTURE_RANGE: integer = 2

static Vectors *mean = new Vectors(3)
static Vectors *dev = new Vectors(3)

function Vector.angle(x: integer, y: integer, yScale:
integer) -> integer
Kamus Lokal
```

```
Algoritma
    -> x + y * yScale

function getGLCMVectorFeature(img: pointer to Image,
offsetX: integer, offsetY: integer) -> pointer to Vector
Kamus Lokal
    v, t: pointer to Vector
    i, j, g1, g2: integer
    tmp, sum: float
Algoritma
    v = new Vector(QUANTIZATION_LEVEL * QUANTIZATION_LEVEL)
    i traversal [0..img.height-offsetY-1]
        j traversal [0..img.width-offsetX-1]
            g1 <- img.getGrayscale(j, i)
            g2 <- img.getGrayscale(j+offsetY, i+offsetX)
            v.component[flatten(g1, g2, QUANTIZATION_LEVEL)]
            <-(v.component[flatten(g1, g2, QUANTIZATION_LEVEL
)] + 1)

    {Matrix Transpose}
    t <- new Vector(v.dimension)
    i traversal [0..QUANTIZATION_LEVEL-1]
        j traversal [0..QUANTIZATION_LEVEL-1]
            tmp <- (v.component[flatten(j, i,
QUANTIZATION_LEVEL)])
            t.component[flatten(i, j, QUANTIZATION_LEVEL)] <-
            tmp

    sum <- 0.0
    i traversal [0..v.dimension]
        v.component[i] <- v.component[i] + t.component[i]
        sum <- sum + v.component[i]
    delete t {deskripsi t}

    if (sum != 0.0) then
        i traversal [0..v.dimension]
            v.component[i] <- v.component[i] + sum

    -> v

function getTextureFeature(img: pointer to Image) -> pointer
to Vectors
Kamus Lokal
    vs: pointer to Vectors
    v: pointer to Vector
    dimension, i, j, idx: integer
```

```
    contrast,homogeneity,entropy: pointer to Vector
Algoritma
    vs <- new Vectors(3)
    int dimension <- TEXTURE_RANGE * TEXTURE_RANGE - 1
    contrast <- (vs.vectors[0] <- new Vector(dimension))
    homogeneity <- (vs.vectors[1] <- new Vector(dimension))
    entropy <- (vs.vectors[2] <- new Vector(dimension))

    i traversal [0..TEXTURE_RANGE-1]
        j traversal [0..TEXTURE_RANGE-1]
            if (i == 0 && j == 0) then
                v = getGLCMVectorFeature(img, i, j)

                idx = i * TEXTURE_RANGE + j - 1
                contrast.component[idx] <- getContrast(v)
                homogeneity.component[idx] <-
                    getHomogeneity(v)
                entropy.component[idx] <- getEntropy(v)

                delete v {destruksi v}
    -> vs

function getContrast(GLCM: pointer to Vector) -> float
Kamus Lokal
    res,diff: float
Algoritma
    res <- 0.0
    i traversal [0..QUANTIZATION_LEVEL-1]
        j traversal [0..QUANTIZATION_LEVEL-1]
            diff <- i - j
            res <- (res + GLCM.component[flatten(i, j,
                QUANTIZATION_LEVEL)] * (diff * diff))

    -> res

function getHomogeneity(GLCM: pointer to Vector) -> float
Kamus Lokal
    res,diff: float
Algoritma
    res <- 0.0
    i traversal [0..QUANTIZATION_LEVEL-1]
        j traversal [0..QUANTIZATION_LEVEL-1]
            diff <- i - j
            res <- (res + GLCM.component[flatten(i, j,
                QUANTIZATION_LEVEL)] / (1 + (diff * diff)))
    -> res
```

```
function getEntropy(GLCM: pointer to Vector) -> float
Kamus Lokal
    res, val: float
Algoritma
    res <- 0.0
    i traversal [0..QUANTIZATION_LEVEL-1]
        j traversal [0..QUANTIZATION_LEVEL-1]
            val <- (GLCM.component[flatten(i, j,
                QUANTIZATION_LEVEL)])
            if (val != 0) then
                res <- res + (val * log(val))

    -> (-1)*res

procedure normalizeWithGaussian(vs: pointer to Vectors)
Kamus Lokal
    dimension: integer
    v: float
Algoritma
    dimension <- TEXTURE_RANGE * TEXTURE_RANGE - 1
    i traversal [0..2]
        j traversal [0..dimension-1]
            v <- vs.vectors[i].component[j]
            v <- v - mean.vectors[i].component[j]
            v <- v / dev.vectors[i].component[j]
            vs.vectors[i].component[j] <- v


procedure calculateGaussianProperties(datasetPath: string,
dataset: vector<string>)
Kamus Lokal
    mean, dev: Vectors
    dimension, count, tillCount: integer
    path: string
    testVectors: pointer to Vectors
    cacheMiss: boolean
    testImage: pointer to Image
    r, m, d: float
Algoritma
    mean <- new Vectors(3)
    dev <- new Vectors(3)

    dimension <- TEXTURE_RANGE * TEXTURE_RANGE - 1
    i traversal [0..2]
        mean.vectors[i] <- new Vector(dimension)
        dev.vectors[i] <- new Vector(dimension)
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
count <- 0
int tillCount <- dataset.size()
for (const auto filename : dataset)
    path <- datasetPath + "/" + filename
    testVectors <- getCache(filename)
    cacheMiss <- testVectors = NULL

    if (cacheMiss) then
        testImage <- new Image(path)
        testVectors <- getTextureFeature(testImage)
        addCache(filename, testVectors)
        delete testImage

    i traversal [0..2]
        j traversal [0..dimension-1]

        r = testVectors.vectors[i].component[j]
        mean.vectors[i].component[j] <-
            (mean.vectors[i].component[j] + r)
        dev.vectors[i].component[j] <-
            (mean.vectors[i].component[j] + r * r)

    ++count

    output("(count/tillCount) [Texture Cache Miss]
%filename.c_str()")
    fflush(stdout)
    delete testVectors

    i traversal [0..2]
        j traversal [0..dimension-1]
        m <- mean.vectors[i].component[j]
        d <- dev.vectors[i].component[j]

        m <- m / count
        d <- d / count
        d <- d - (m * m)
        d <- sqrt(d)

        mean.vectors[i].component[j] <- m
        dev.vectors[i].component[j] <- d
```

v. main.cpp

```
function main(argc: integer, argv: pointer to pointer to
char) → integer
```

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

```
string datasetPath ← argv[2]
pointer to DIR datasetDir ← opendir(datasetPath.c_str())

pointer to Dirent tmpDirent
vector<string> dataset
while ((tmpDirent = readdir(datasetDir)) != NULL) do
    string filename ← tmpDirent.d_name
    string ext ← filename.substr(filename.rfind(".") + 1)
    if (exts.count(ext) == 0) then continue
    dataset.push_back(filename)

boolean hasTarget ← argc = 4
string cbirType ← argv[1]
string targetPath ← if (hasTarget) then argv[3] else ""
cacheSetup(datasetPath, cbirType)

if (cbirType = "color") then vectorsFn ← getHSVFeatureVector
else if (cbirType == "texture") then vectorsFn ←
getTextureFeature

{Calculate mean and standard deviation [Make sure to do
caching to make this faster, beside there won't be any info
about cache hit or miss in this operation]}
if (hasTarget and cbirType = "texture") then
calculateGaussianProperties(datasetPath, dataset)

pointer to Image targetImage
pointer to Vectors targetVectors
if (hasTarget) then
    targetImage ← new Image(targetPath)
    targetVectors ← vectorsFn(targetImage)

    if (cbirType = "texture") then
normalizeWithGaussian(targetVectors)

integer count ← 0
integer tillCount ← dataset.size()
for (const auto filename : dataset)
    string path ← datasetPath + "/" + filename
    pointer to Vectors testVectors ← getCache(filename)
    boolean cacheMiss ← testVectors = NULL

    if (cacheMiss) then
        pointer to Image testImage ← new Image(path)
        testVectors ← vectorsFn(testImage)
```

```
    addCache(filename, testVectors)
    delete testImage

    count ← count + 1
    if (hasTarget) then
        if (cbirType = "texture") then
            normalizeWithGaussian(testVectors)
            double angle ←
                Vectors.getAngleAverage(targetVectors, testVectors)

            string c ← ""
            if (cacheMiss) then c ← " [Cache miss] "
            else
                string msg ← "hit"
                if (cacheMiss) then msg ← "miss"

            fflush(stdout)

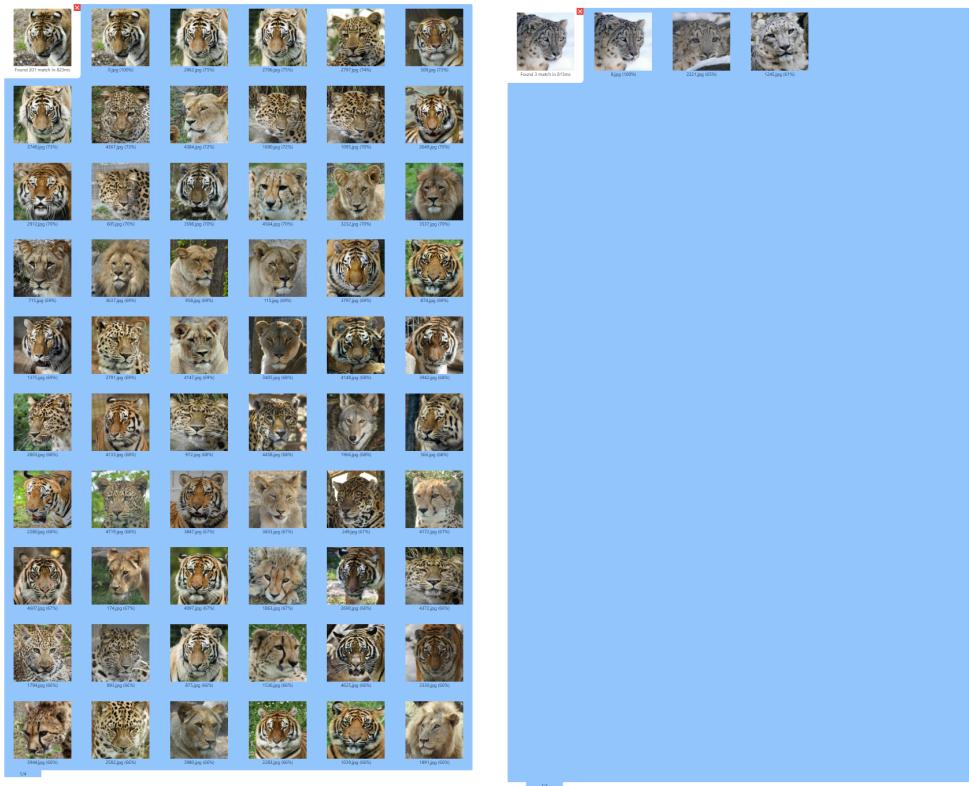
            delete testVectors

        cacheCleanup()
```

#### e. Analisis CBIR

Pada pencarian gambar menggunakan parameter warna, didapatkan hasil yang cukup memuaskan. Waktu eksekusi sebelum *caching* sekitar 25s untuk 4738 gambar. Sedangkan setelah *caching* waktu eksekusi berkisar 800ms. Kemudian untuk hasilnya sendiri, terlihat pada gambar dibawah ini, dengan gambar yang berada pada kotak biru merupakan gambar yang telah diurutkan berdasarkan kemiripan parameter warna. Terlihat pada gambar bahwa program dapat melakukan pencarian berdasarkan kemiripan warna dengan baik. Terutama ketika gambar pada *dataset* tergolong langka (seperti pada gambar di kanan). *Dataset* yang digunakan pada parameter gambar adalah [CBIR\\_dataset](#).

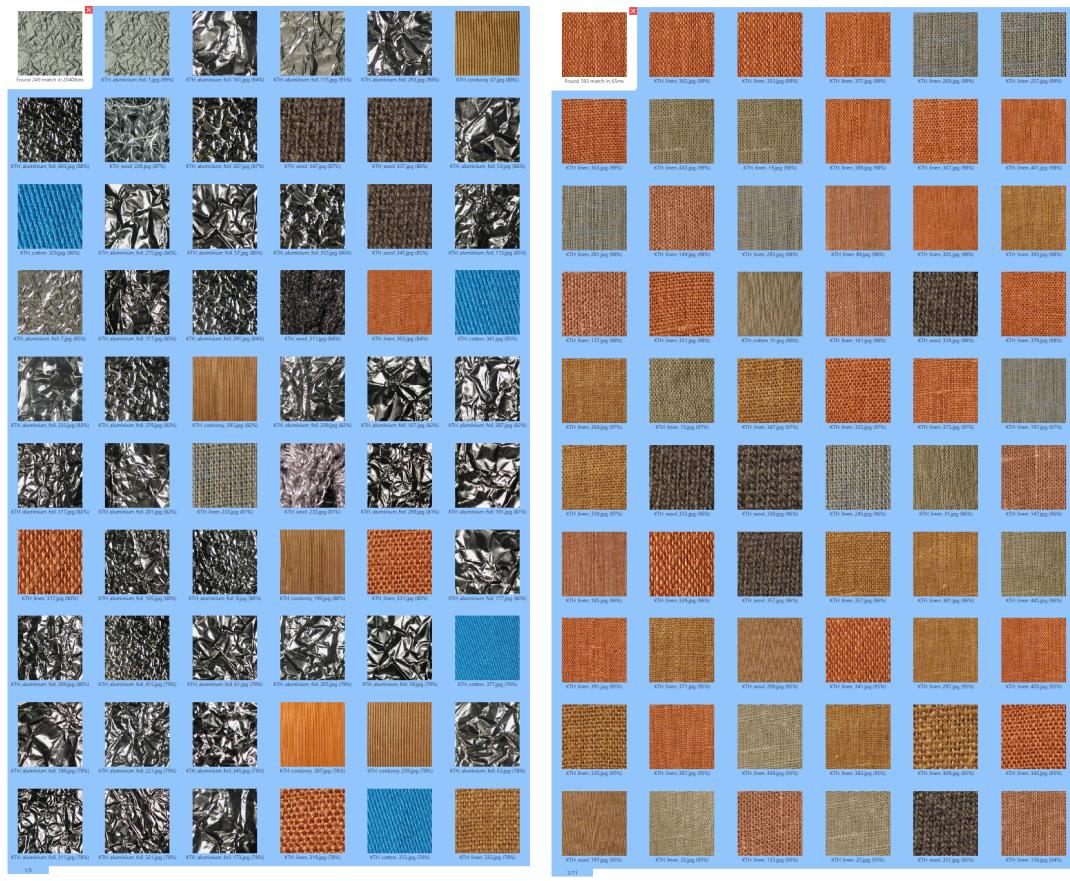
Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Kemudian untuk pencarian menggunakan parameter tekstur, tidak memiliki konsistensi yang sama seperti parameter warna. Hal ini terlihat pada gambar bagian kiri dibawah yang menampilkan tekstur selain *alumunium foil* pada gambar teratas. Namun pada lain kesempatan, gambar yang ditemukan memiliki tekstur yang diinginkan. Pada tes ini menggunakan [Textures-Dataset](#). Hasil eksekusi tanpa *cache* berkisar 20s, sedangkan setelah menggunakan *cache* eksekusi menjadi sekitar 100ms. Seluruh waktu eksekusi dihitung menggunakan Apple Silicon M1 Max sebagai server untuk aplikasi utama dan *backend*.

# Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri

## Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

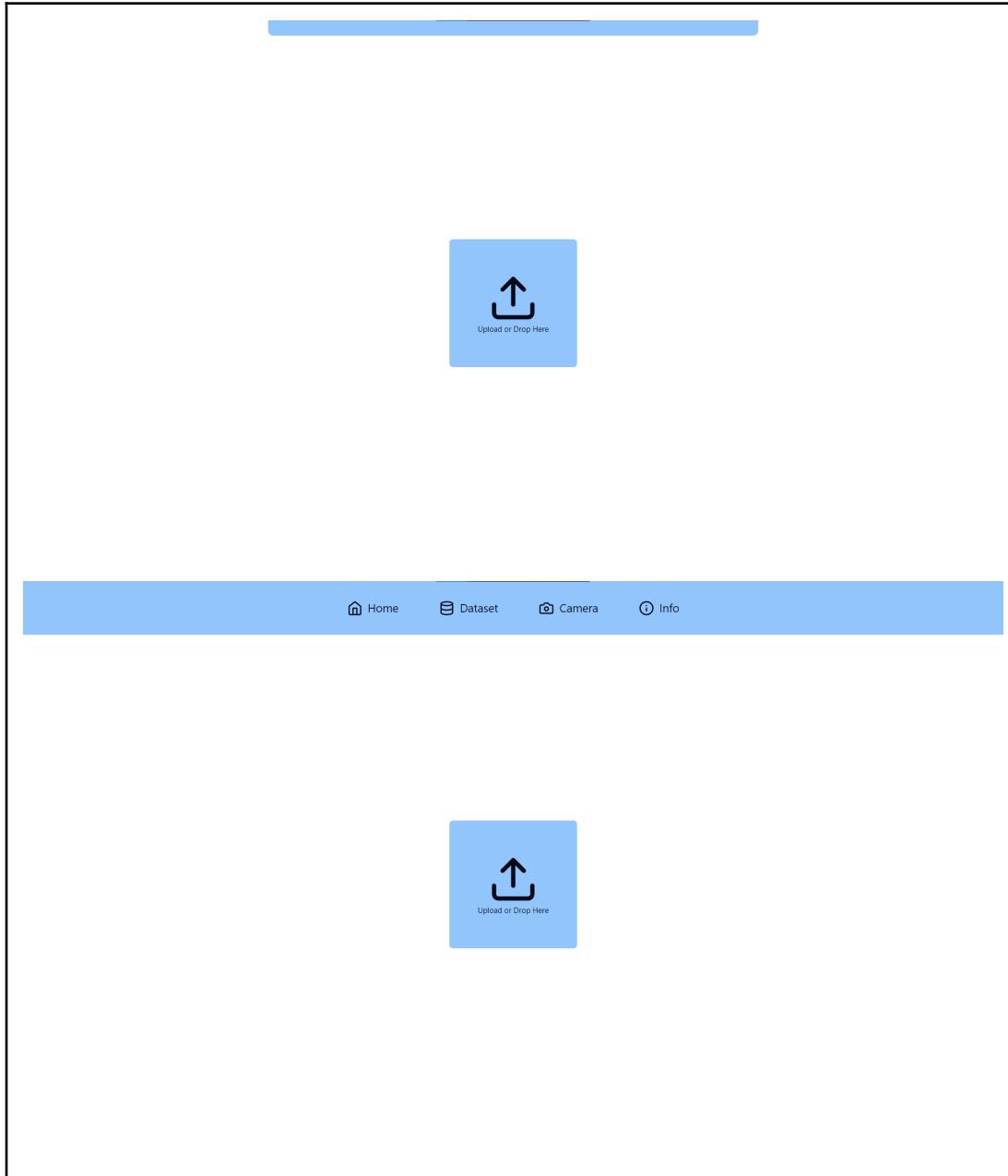


## 4.2. Uji Coba

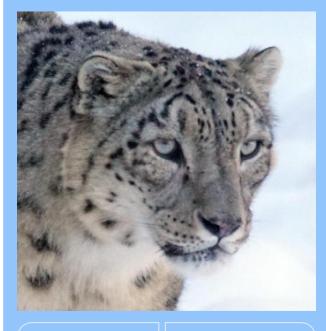
### a. Tampilan

*Tampilan Home*

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Color      |      Texture

**Tampilan Dataset**

Add New (Folder)

Delete All

Clear Cache


1/106

**Tampilan Camera**

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



A screenshot of a video call interface. At the top, there is a blue horizontal bar with a color calibration icon labeled "Color". Below it is a video frame showing a person from the chest up, wearing a light green shirt. The background shows an indoor setting with warm lighting.

**Tampilan Info**



iCBIR - Image Retriever  
by Frontend: Prolog / Backend: Haskell

Kiri ke kanan: Alif, Ben, Qais

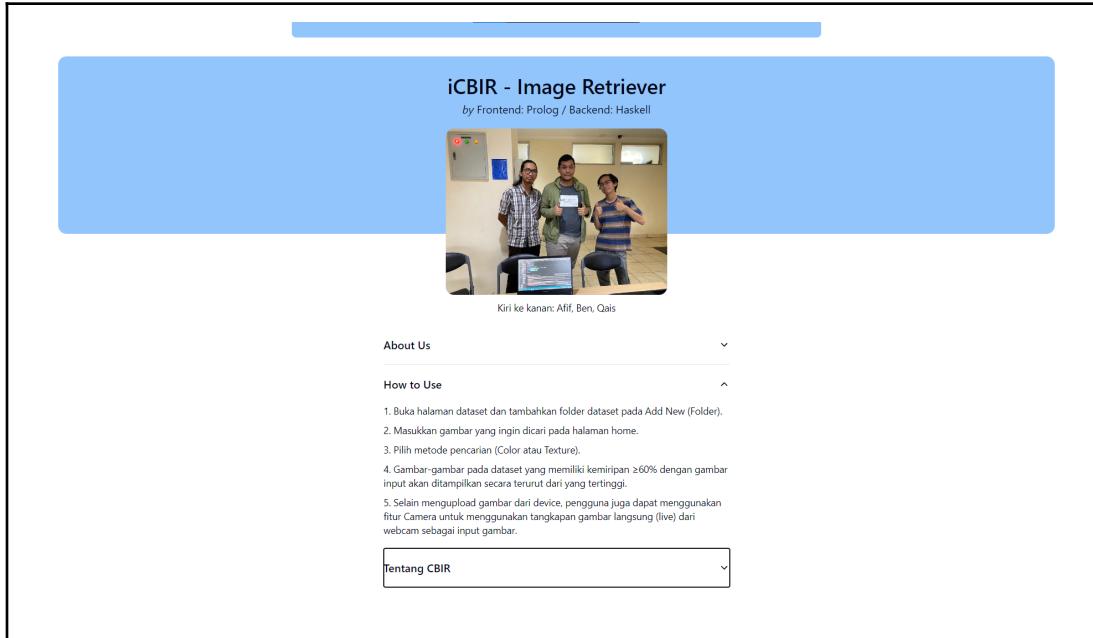
About Us

How to Use

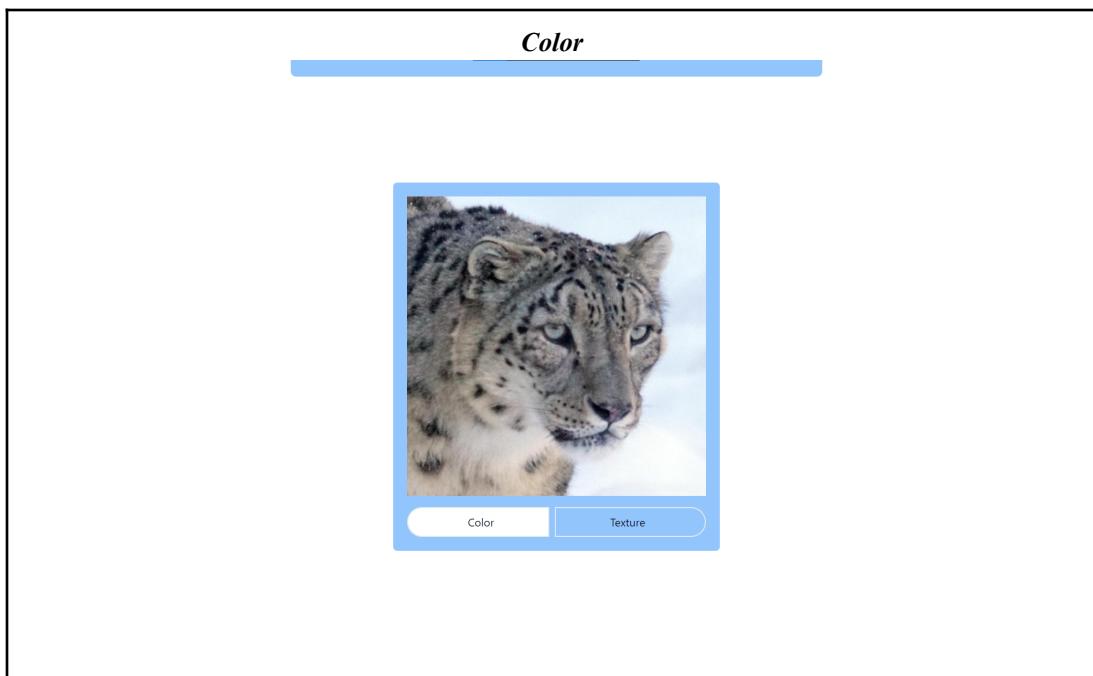
Tentang CBIR

The screenshot shows a blue header bar with the text "Tampilan Info". Below it is a blue rectangular area containing the title "iCBIR - Image Retriever" and the text "by Frontend: Prolog / Backend: Haskell". Inside this area is a small image of three people standing together, with the caption "Kiri ke kanan: Alif, Ben, Qais". Below this are three dropdown menus with the labels "About Us", "How to Use", and "Tentang CBIR".

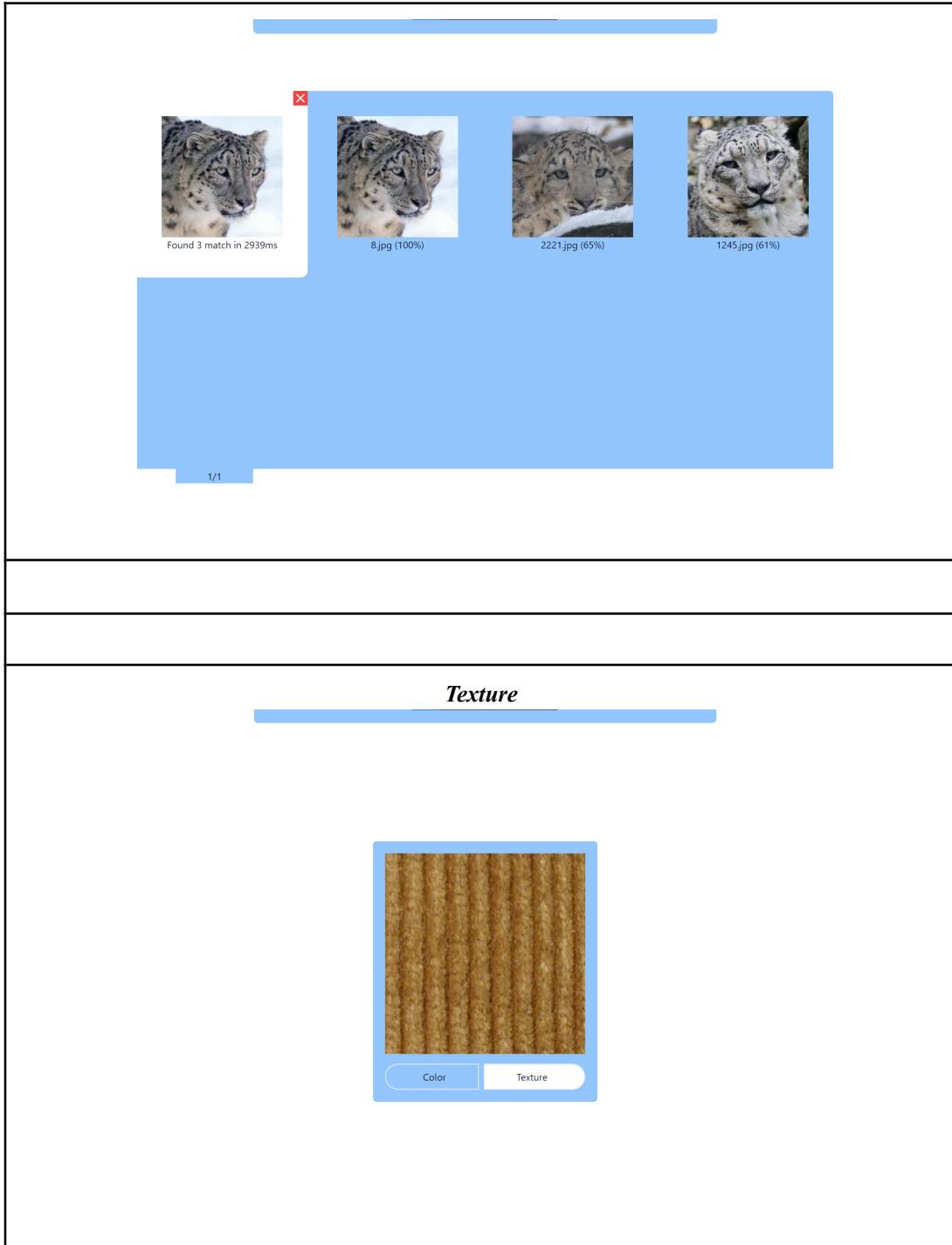
Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



b. Eksperimen



Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

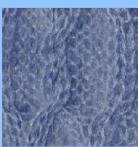
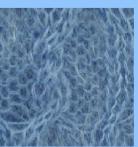


Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

Camera

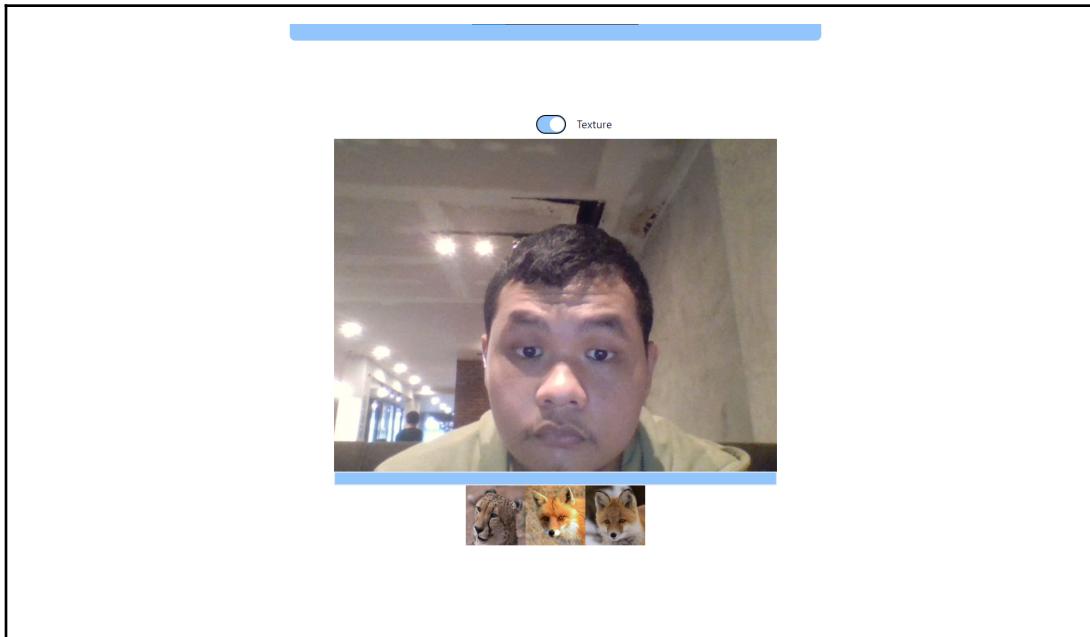
Color 

Found 1301 match in 424ms X

	KTH_corduroy_21.jpg (99%)
	KTH_corduroy_39.jpg (99%)
	KTH_wool_341.jpg (99%)
	KTH_corduroy_33.jpg (99%)
	KTH_wool_81.jpg (99%)
	KTH_corduroy_249.jpg (99%)
	KTH_wool_73.jpg (99%)

1/186

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



## BAB 5

### Kesimpulan, Saran, Komentar, dan Refleksi

#### 5.1 Kesimpulan

Dari penggerjaan tugas ini, kami berhasil menerapkan pelajaran Aljabar Linear pada suatu program website. Kami berhasil mengimplementasikan baik struktur data vektor dan properti-propertinya serta mengimplementasikan fungsi-fungsi untuk CBIR. Program ini berhasil mencari gambar-gambar dari dataset yang mirip dengan gambar input berdasarkan warna atau tekstur dengan menggunakan CBIR. Gambar-gambar yang memiliki kemiripan >60% dengan gambar input akan berhasil ditampilkan ke website secara terurut dari yang terbesar.

#### 5.2 Saran

Dari hasil penggerjaan tugas besar ini, terbukti bahwa gambar-gambar *digital* dapat diubah menjadi bentuk vektor dan diolah secara aljabar. Topik *image processing* merupakan suatu masalah yang sangat sering ditemukan di bidang ilmu komputer, contohnya meliputi (tapi tidak terbatas pada) *Image Recognition*, *Face Recognition*, dan *Content-Based Image Retrieval* yang merupakan masalah utama pada tugas besar ini.

Maka saran dari kelompok kami terkait tugas besar ini adalah agar para pembaca dan pelajar lain yang sedang menempuh ilmu di bidang Ilmu Komputer agar memahami konsep-konsep yang diajarkan di mata kuliah Aljabar Linear dan Geometri ini, dan dapat menerapkan konsep-konsep tersebut di permasalahan lain yang ditemukan di kehidupan sehari-hari.

#### 5.3 Komentar

Walaupun eksplorasi terkait hal-hal baru merupakan konsep yang penting dalam proses pembelajaran, kami rasa konsep-konsep yang belum diajarkan dan perlu dipelajari secara mandiri pada tugas besar ini terlalu banyak dan terlalu kompleks untuk dipelajari dan langsung diterapkan pada kurun waktu kurang dari 3 minggu. Saran dari kami untuk tugas-tugas besar berikutnya lebih menyesuaikan pada konsep-konsep yang diajarkan dan rentang waktu yang diberikan

#### 5.4 Refleksi

Untuk refleksi terkait tugas besar ini, seluruh anggota kelompok kami berasal dari kelas yang berbeda-beda, sehingga cukup sulit untuk mencari waktu berkumpul dan melakukan progress. Kami rasa untuk penggerjaan tugas-tugas berikutnya perlu dicari cara untuk berkumpul dan mengerjakan tugas secara lebih efektif dan efisien.

#### 5.5 Ruangan Perbaikan dan Pengembangan

Hasil dari penggerjaan tugas besar ini adalah kami melihat secara langsung atas sulitnya proses pengembangan sebuah *website*, sehingga setelah pengumpulan tugas besar ini kami perlu mempelajari dan mengasah lebih lanjut kemampuan mengembangkan *website*.

## **DAFTAR REFERENSI**

“RGB to HSV color conversion”

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

“Content-based image retrieval using color and texture fused features”

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

“Feature Extraction : Gray Level Co-occurrence Matrix (GLCM)”

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

Tugas Besar 2 IF 2123 Aljabar Linear Dan Geometri  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar

## REPOSITORY

Link repository dari Tugas Besar 2 IF 2123 Aljabar Linear dan Geometri kelompok

Frontend: Prolog / Backend: Haskell adalah <https://github.com/atpur-rafir/Algeo02-22054>

Link release terakhir: <https://github.com/atpur-rafir/Algeo02-22054/releases/tag/v1.0>