Laporan Tugas Besar 2

IF2123 Aljabar Linier dan Geometri
Penerapan Metrik Berbasiskan Vektor di dalam Sistem Pengenalan Wajah (*Face Recognition*)
Semester 1 Tahun 2019/2020

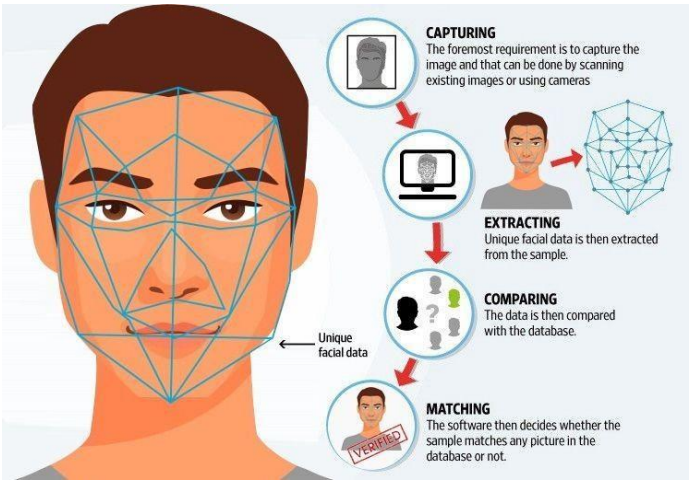
Disusun Oleh :
Chandrika Azharyanti / 13518001
Inka Anindya Riyadi / 13518038
Felicia Gillian Tekad Tuerah / 13518070

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

2019

BAB I
DESKRIPSI MASALAH

Pengenalan wajah (*face recognition*) adalah teknologi untuk mengidentifikasi atau memverifikasi wajah seseorang melalui gambar digital. Caranya adalah dengan mencocokkan fitur-fitur yang diekstraksi dari wajah yang diidentifikasi dengan data wajah yang tersimpan di dalam basisdata. Pengenalan wajah telah digunakan sebagai sebuah sistem biomettrik. Alur proses sebuah sistem pengenalan wajah diperlihatkan pada Gambar 1.



Gambar 1. Alur proses di dalam sistem pengenalan wajah
(Sumber: https://www.shadowsystem.com/page/20)

Fitur-fitur dari gambar wajah diekstraksi dengan sebuah prosedur komputasi menggunakan teknik-teknik di dalam pengolahan citra (*image processing*), namun di dalam tugas besar ini fungsi untuk ekstraksi fitur diasumsikan sudah tersedia. Sekumpulan fitur tersebut direpresentasikan sebagai vektor. Proses pencocokan antar vektor wajah yang ditanya dengan vektor-vektor wajah di dalam basisdata menggunakan metriks *similarity*. Metrik *similarity* itu mengukur seberapa dekat atau mirip antara dua buah vektor.

Gambar 2. Koleksi wajah di dalam basisdata
(Sumber: https://www.dreamstime.com/stock-photo-people-faces-collection-set-smiling-face-group-image80688279)

BAB II
TEORI SINGKAT

Diberikan dua buah vektor, $\mathbf{v} = (v_1, v_2, ..., v_n)$ dan $\mathbf{w} = (w_1, w_2, ..., w_n)$. Ada dua metriks *similarity* yang umum digunakan di dalam pencocokan data, yaitu jarak Euclidean dan *cosine similarity*.

1. **Jarak Euclidean** (*Euclidean distance*). Jarak antara vektor **v** dan **w** diukur dengan rumus

$$d = \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2 + ... + (v_n - w_n)^2} \quad (1)$$

   Nilai *d* yang kecil menunjukkan kedekatan. Jika **v** dihitung jaraknya masing- masing dengan vektor $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_m}$, yaitu $d(\mathbf{v}, \mathbf{w_i})$ untuk $i = 1, 2, ..., m$, maka nilai *d* yang paling minimum menunjukkan jarak dua vektor yang paling mirip.

2. *Cosine similarity*. Metriks *cosine similarity* dihitung dari perkalian titik (*dot product*) antara dua buah vektor. Perkalian titik antara **v** dan **w** dihitung dengan rumus

$$\mathbf{v} \cdot \mathbf{w} = \|\mathbf{v}\| \|\mathbf{w}\| \cos\theta \quad (2)$$

   Sudut antara **v** dan **w** adalah

$$\cos\theta = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\| \|\mathbf{w}\|} \quad (3)$$

   Persamaan (3) digunakan untuk mengukur *similarity* antara dua buah vektor. Dua buah vektor **v** dan **w** dikatakan sama atau berimpit jika sudut antara keduanya nol (= 0). Cosinus 0 adalah 1. Sifat ini dipakai di dalam proses pencocokan antara dua buah vektor. Nilai cosinus yang besar (maksimum 1) menunjukkan kemiripan. Jika nilai cosinus mendekati satu, maka dua vektor dikatakan hampir sama atau hampir mirip. Oleh karena itu, persamaan (3) dinamakan juga *cosine similarity*.

Di dalam tugas besar ini, anda diminta membuat sistem pengenalan wajah dengan menggunakan metrik kemiripan berbasis vektor. Input untuk sistem adalah sebuah citra wajah yang ditanyakan, sedangkan koleksi citra wajah di dalam basisdata diasumsikan sudah tersedia. Fitur-fitur dari wajah diekstraksi, begitu pula fitur-fitur wajah di dalam basisdata. Pencocokan kemiripan dilakukan antara vektor fitur citra wajah input dengan vektor-vektor fitur wajah di dalam basisdata. Hasil pencocokan di-*rangking* dari yang paling mirip hingga

yang kurang mirip (gunakan nilai ambang T untuk menampilkan hasil pencocokan, misal *T* = 10, maka akan ditampilkan 10 wajah yag paling mirip hingga yang tidak terlalu mirip).

Program dibuat dengan Bahasa Python dengan memanfaatkan sejumlah *library* di *OpenCV* (*Computer Vision*). Fungsi untuk mengekstraksi fitur dari sebuah citra wajah tidak perlu anda buat lagi, tetapi menggunakan fungsi ekstraksi yang sudah tersedia di dalam *library*.

Ada banyak algoritma untuk ekstraks fitur, yang paling poupuler adalah SURF, ORB, SIFT, BRIEF. Kebanyakan dari algoritma ini didasakan pada image gradient. Kita akan menggunakan descriptor KAZE karena sudah tersedia di dalam pustaka OpenCV (lihat Gambar 3).

Computer Vision adalah kemampuan mesin/komputer dalam melihat hingga mampu meng-ekstrak informasi dari sebuah gambar. Salah satu bidang yang berkaitan dengan *Computer Vision* adalah Pengolahan Citra atau biasa disebut *Image Processing.*

OpenCV adalah sebuah library (perpustakaan) yang digunakan untuk mengolah gambar dan video hingga kita mampu meng-ekstrak informasi didalamnya. OpenCV dapat berjalan di berbagai bahasa pemograman, seperti C, C++, Java, Python, dan juga support diberbagai platform seperti Windows, Linux, Mac OS, iOS dan Android.

BAB III
IMPLEMENTASI PROGRAM

Untuk menyelesaikan permasalahan face recognition ini, kami membagi program menjadi beberapa modularitas.

1. KODE.PY

```
# Feature extractor
def extract_features(image_path, vector_size=32):
    image = imread(image_path, mode="RGB")
    print("Extracting ficurs from" +image_path)
    try:
        alg = cv2.KAZE_create()
        # Dinding image keypoints
        kps = alg.detect(image)
        # Getting first 32 of them.
        kps = sorted(kps, key=lambda x: -x.response)[:vector_size]
        # computing descriptors vector
        kps, dsc = alg.compute(image, kps)
        # Flatten all of them in one big vector - our feature vector
        dsc = dsc.flatten()
        # Descriptor vector size is 64
        needed_size = (vector_size * 64)
        if dsc.size < needed_size:
            # if we have less than 32 descriptors then just adding zeros at the
            # end of our feature vector
            dsc = np.concatenate([dsc, np.zeros(needed_size - dsc.size)])
    except cv2.error as e:
        print ('Error: '), e
        return None

    return dsc
```

Fungsi extract_features ini mengubah foto-foto yang ada dalam folder dataset kedalam bentuk vektor yang nantinya akan dioperasikan.

```
#prosedur untuk menyimpan nama-nama file ke nama_file.pck
def saveNamaFile():
    data = os.listdir("dataset/")
    namafile = [0 for i in range (len(data))] #berisi nama file
    for i in range (len(data)):
        namafile[i]=data[i]
    with open('nama_file.pck', 'wb') as fopen:
        pickle.dump(namafile, fopen)
    fopen.close()
```

```
#prosedur untuk menyimpan vektor ke vector.pck
def saveVector():
    data = os.listdir("dataset/")
    vec = [0 for i in range (len(data))] #berisi data vector
    for i in range (len(data)):
        vec[i]=extract_features("dataset/"+data[i])
    with open('vector.pck', 'wb') as fopen:
        pickle.dump(vec, fopen)
    fopen.close()
```

Kedua prosedur ini membuat dua file pickel yang berisikan vector hasil dari extract dan nama foto-foto dari folder dataset.

```python
#fungsi yang mengembalikan array of vector
def loadVector():
    fopen= open('vector.pck','rb')
    fload= pickle.load(fopen) #array of vector
    fopen.close()
    vector=[0 for i in range (len(fload))]
    for i in range (len(fload)):
        vector[i]=fload[i]
    return vector
```

```python
#fungsi yang mengembalikan array of nama file
def loadNamaFile():
    fopen= open('nama_file.pck','rb')
    fload= pickle.load(fopen) #array of namafile
    fopen.close()
    namafile=["" for i in range (len(fload))]
    for i in range (len(fload)):
        namafile[i]=fload[i]
    return namafile
```

Kedua fungsi diatas memindahkan isi vektor.pck dan nama_file.pck ke dalam array of vectors dan array of string.

2. VEKTOR.PY

```python
def euclidean(x,y):
    sum = 0
    for i in range(len(y)):
        sum += math.pow((x[i]-y[i]),2)
    return math.sqrt(sum)
```

Fungsi euclidian memiliki 2 parameter dimana x merupakan vektor dari foto inputan user dan y adalah array of vector dari hasil extract folder dataset dan akan menghasilkan jarak euclidean dari kedua vektor tersebut. Hasil dari perhitungan jarak euclidean ini kemudian akan dimasukkan kedalam array of floats bernama hasilje.

```
def haslowest(cek):
    minn=0
    for i in range (len(cek)):
        if(cek[i]<cek[minn]):
            minn=i
```

Fungsi haslowest menghasilkan index yang memiliki nilai terkecil dari hasilje, fungsi ini digunakan bersama fungsi euclidean dikarenakan foto paling akurat adalah foto yang memiliki jarak euclidean terkecil.

```
def norm(x):
    sum = 0
    for i in range(len(x)):
        sum += math.pow(x[i],2)
    return math.sqrt(sum)
```

```
def cosine(x,y):
    return dotproduct(x,y) / (norm(x)*norm(y))
```

Fungsi cosine merupakan cara lain dalam membandingan vektor foto inputan dengan vektor pada dataset. Untung menghitung cosine similarity sendiri kami membuat 2 fungsi pendukung yaitu dotproduct(mengalikan dua vektor dengan dot) dan norm(menghitung besaran suatu vektor). Hasil perhitungan cosine disimpan di dalam array of floats bernama hasilcs.

```
def hashighest(cek):
    maks=0
    for i in range (len(cek)):
        if(cek[i]>cek[maks]):
            maks=i
    return maks
```

Fungsi hashighest menghasilkan index yang memiliki nilai terbesar dari hasilcs, fungsi ini digunakan bersama fungsi euclidean dikarenakan foto paling akurat adalah foto yang memiliki jarak euclidean terbesar.

3. MAIN.PY

```
#je : euclidean
#cs : cosin
vector = []
fileimg = []

folder = "dataset/"

vector = kode.loadVector() #mindahin ke array
fileimg = kode.loadNamaFile() #mindahin ke array

vInit = kode.extract_features(filename) #hasil vektor yg mau di compare

hasilje = [0 for i in range(len(vector))] #nyimpen hasil jarak euclidean
hasilcs = [0 for i in range(len(vector))] #nyimpen hasil cosin

hasil = ["" for i in range(10)]
if(metode=="je"):
    for i in range (len(vector)):
        hasilje[i] = vektor.euclidean(vInit,vector[i])

    for i in range (10):
        maks = vektor.haslowest(hasilje)
        hasilje[maks]=1000
        hasil[i]=fileimg[maks]
```

```
else:
    for i in range (len(vector)):
        hasilje[i] = vektor.cosine(vInit,vector[i])

    for i in range (10):
        maks = vektor.hashighest(hasilje)
        hasilje[maks]=-1000
        hasil[i]=fileimg[maks]

img_rank1_raw = folder+hasil[0]
img_rank2_raw = folder+hasil[1]
img_rank3_raw = folder+hasil[2]
img_rank4_raw = folder+hasil[3]
img_rank5_raw = folder+hasil[4]
img_rank6_raw = folder+hasil[5]
img_rank7_raw = folder+hasil[6]
img_rank8_raw = folder+hasil[7]
img_rank9_raw = folder+hasil[8]
img_rank10_raw = folder+hasil[9]
```

Main.py berisi semua fungsi-fungsi dan procedure yang digunakan untuk membuat GUI serta program utama yang menjalankan fungsi-fungsi serta prosedur yang ada pada kedua modul yang sudah dijelaskan sebelumnya.

Pada program main di atas, terdapat array hasilje (menyimpan nilai hasil perhitungan jarak euclidean) ,hasilcs (menyimpan nilai hasil perhitungan cosine cimilarity), dan array hasil (menyimpan string nama file foto). Disediakan 10 foto yang memiliki nilai perhitungan vector yang paling sesuai, img_rank(n)_raw adalah variable yang menyimpan 10 foto terurut tersebut yang akan dipakai saat ditampilkan di GUI.

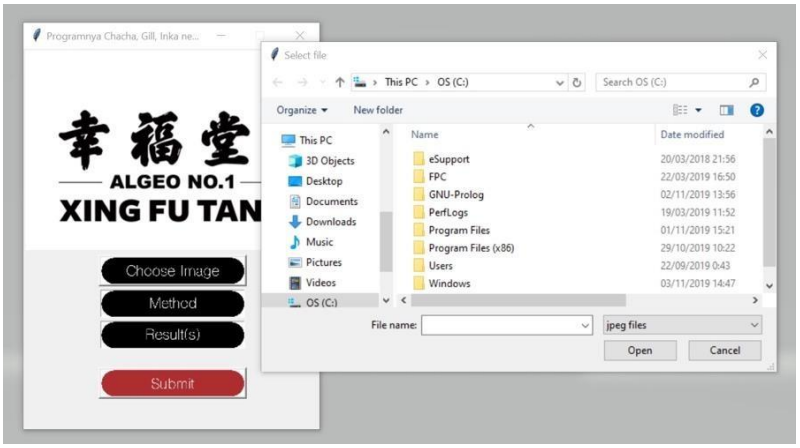
BAB IV

EKSPERIMEN

- Tampilan awal

Pada tampilan awal, terdapat 5 pilihan tombol, yaitu choose image, methode (yang dapat memilih 2 metode), results (memilih jumlah foto hasil komparasi yang ingin ditampilkan, bernilai 1-10), dan submit (di klik jika sudah memasukkan foto dan memilih metode)

- Pilih file foto yang ingin dicompare



Jika me-click choose image, user akan langsung diarahkan ke folder yang ada di laptop anda, lalu anda bisa memilih foto berformat jpg untuk dilakukan komparasi pada program utama.

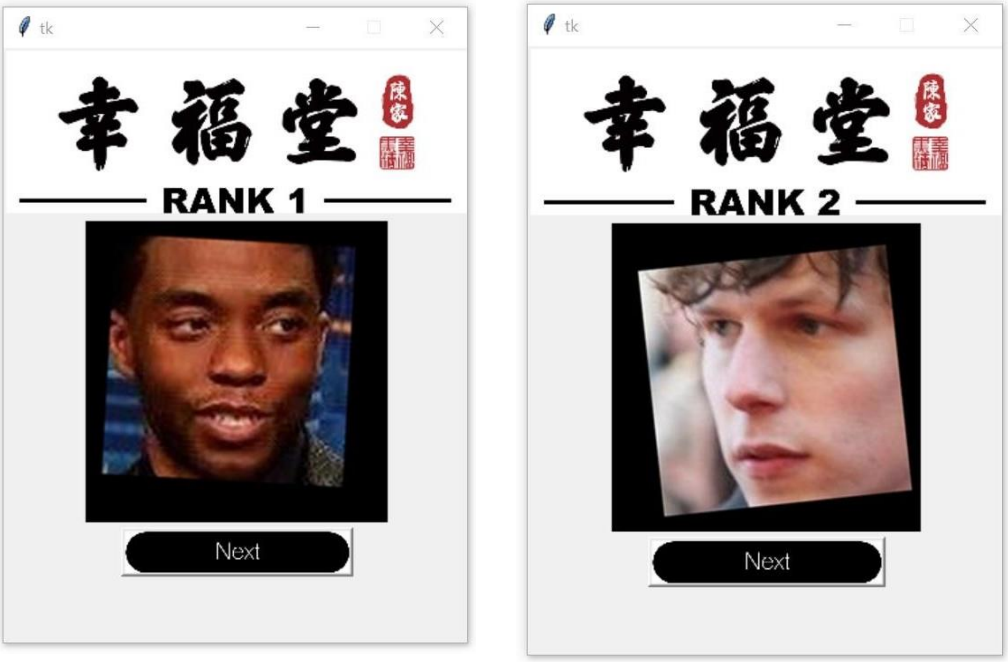
- Pilih metode face recognition

Terdapat 2 pilihan cara face recognition, yaitu Euclidean distance dan cosine similarity. Anda dipersilahkan untuk memilih salah satu cara, penjelasan cara kerja metode terdapat pada teori singkat bab 2.

- Pilih jumlah hasil yang ingin ditampilkan

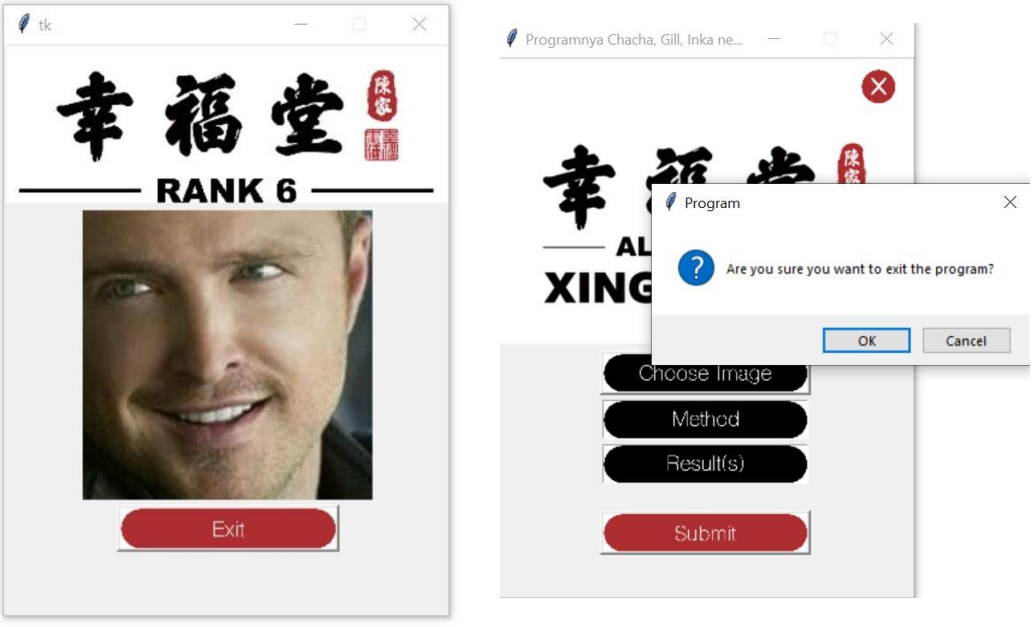
Disini kami membatasakan hasil foto untuk ditampilkan kelayar sebanyak 1 – 10, dan user dipersilahkan memilih jumlah yang diinginkan.

- Tampilan urutan foto hasil face recognition



Berikut contoh hasil tampilan foto sesuai dengan urutan sesuai hasil komparasi pada program utama, urutan dapat dilihat pada angka setelah tulisan RANK, untuk melihat urutan selanjutnya user dapat me-click tombol next pada bagian bawah.

- Exit face recognition



Setelah foto hasil face recognition yang ditampilkan sebanyak jumlah yang anda pilih, pada rank terakhir akan muncul tombol exit pada bagian bawah foto, jika anda click exit akan muncul kembali pilihan “are you sure you want to exit the program?”, click ok untuk keluar dan tampilan akan hilang dari layar, jika pilih cancel akan kembali pada tampilan awal.

BAB V
PENUTUP

Kesimpulan

Kesimpulan yang kelompok kami dapatkan adalah hasil dari face recognition dengan menggunakan metode jarak euclidean dengan cosine similarity adalah sama. Foto hasil dari face recognition identitasnya dapat berbeda dengan foto inputan user, hal ini dapat terjadi karena vektor hasil extract foto kurang akurat, sehingga jika wajah di dalam foto sedang menengok / menggunakan kacamata hitam / warna gelap akan menghasilkan vektor yang berbeda walaupun identitasnya sama. Serta hasil perhitungan dari foto dengan identitas yang berbeda dapat bernilai sama dengan foto yang beridentitas sama, sehingga urutan foto akan membuat foto yang beridentitas beda akan tercetak terlebih dahulu.

Refleksi

Kami seluruh anggota kelompok XINGFUTANG mengucapkan terimakasih karena tugas besar aljabar geometri ini membuat kami belajar banyak hal baru mulai dari openCV hingga GUI, walaupun implementasi dari pelajaran aljabar geometri yang diajarkan di kelas kurang teraplikasikan di program ini namun cukup untuk membuat kita paham bahwa vektor berperan besar dalam perkembangan teknologi.

Saran

Saran pengembangan untuk program pada tugas ini adalah kita lebih diberi penjelasan tentang bagaimana cara kerja dari ekstrak foto menjadi vektor, agar kita dapat membuat program face recognition yang lebih akurat.

REFERENSI

Dreamstime. https://www.dreamstime.com/stock-photo-people-faces-collection-set-smiling-face-group-image80688279 (Diakses pada 7 November 2019 pukul 9.00)

GeeksforGeeks. https://www.geeksforgeeks.org/understanding-python-pickling-example/ (Diakses pada 7 November 2019 pukul 10.00)

Medium. https://medium.com/machine-learning-world/feature-extraction-and-similar-image- search-with-opencv-for-newbies-3c59796bf774 (Diakses pada 7 November 2019 pukul 10.40)

Munir, Rinaldi(2019). http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2019-2020/algeo19-20.htm (Diakses pada 7 November 2019 pukul 15.10)