Sistem Pendeteksian Wajah untuk Aplikasi Penambahan Stiker Pada Wajah *Realtime*

Pemanfaatan teknik pengolahan citra untuk sistem pendeteksian wajah

Kevin Katsura Dani Sitanggang 13519216
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519216@std.stei.itb.ac.id

***Abstract*—Aplikasi penambahan stiker pada wajah diimplementasikan dengan dasar konsep pendeteksian wajah. Pendeteksian wajah dilakukan dengan menerapkan teknik Haar feature-based cascade classifier dengan melakukan konvolusi integral citra masukan dengan berbagai kombinasi fitur kernel matriks. Penggunaan teknik ini dilakukan karena akurasi dan efisiensinya dalam mengurangi waktu komputasi menjadi lebih singkat. Pada pengimplementasiannya, algoritma haar cascade classifier yang digunakan merupakan pre-trained model dari opencv. Model tersebut dapat secara langsung digunakan untuk mendeteksi wajah manusia pada gambar yang diambil dari *frame of realtime video camera*. Pengimplementasiannya berhasil dilakukan walaupun untuk beberapa stiker tidak sempurna ditempatkan pada wajah karena tidak adanya fitur pendeteksian khusus untuk lokasi hidung, maupun mata. Dengan begitu, stiker yang simetris terhadap wajah keseluruhan merupakan yang cocok untuk kasus ini.**

Keywords—konvolusi; stiker; haar cascade classifier; citra; kernel; fitur; adaboost; titik; grayscale; rgb; bgr;

I. Pendahuluan

Saat ini, sosial media seperti snapchat, instagram, dan facebook menjadi jaringan berbagi informasi yang sangat populer bagi semua kalangan, tidak terbatas oleh umur maupun status sosial. Seseorang cenderung senang berbagi berbagai hal tentang hidupnya dan menunjukkan berbagai kreativitas. Untuk mendukung kreativitas dan menunjang perkembangan sisi bisnis suatu perusahaan sosial media, tentunya dibutuhkan inovasi yang memicu tren di masyarakat. Atas dasar tersebut, fitur stiker dan filter dikembangkan di berbagai perusahaan sosial media. Kemunculan fitur ini tentunya sangat populer hingga saat ini. Berbagai macam fitur disediakan untuk mengubah penampilan seseorang menjadi berbagai sisi yang berbeda. Bahkan saat ini, fitur ini tidak hanya sekadar menempelkan stiker saja melainkan transformasi bentuk wajah secara tiga dimensi menyerupai berbagai wajah orang terkemuka.

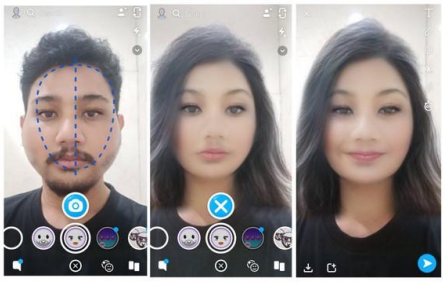

Fig1. Fitur filter wajah pada snapchat

Face filter merupakan salah satu fitur yang memiliki banyak pengguna harian. Fitur ini dijadikan fitur favorit karena dapat mengubah wujud wajah kita ke berbagai bentuk unik. Wajah mulus tanpa noda membuat beberapa orang merasa kagum melihat sisi dirinya yang menarik. Bahkan membuat orang tersebut ketergantungan dalam penggunannya. Dari sini tentunya dapat dilihat sisi negatif dari beberapa penggunaan *face filter*. Citra menawan yang ditawarkan membuat orang tidak realistis hingga berujung ke berbagai tindak kejahatan. Bukan hanya penipuan wajah di berbagai aplikasi pencarian jodoh, fitur ini juga sering digunakan oleh berbagai pelaku kejahatan dengan teknik masking tiga dimensi dimana wajah kita secara tiga dimensi menyerupai seseorang. Tindak penipuan ini membuat pelaku terdeteksi memiliki otoritas terhadap kepemilikan korban. Dengan begitu, fitur ini harus digunakan dengan hati-hati dan baik.

Pengimplementasian fitur ini dapat dilakukan dengan berbagai pendekatan. Akan tetapi, walau pendekatan yang digunakan berbeda, tujuan dasar yang perlu diperhatikan adalah bagaimana sistem dapat mengenali wajah manusia. Setelah itu, dari posisi wajah yang terdeteksi akan diterapkan filter pada posisi tersebut dengan rasio yang sesuai. Beberapa pendekatan yang dapat digunakan pada pendeteksian wajah adalah *convolution neural network* dan *haar feature-based cascade classifier*. Pada makalah ini, penulis akan menjelaskan sistem pendeteksian wajah dengan pendekatan *cascade classifier.*

II. LANDASAN TEORI

B. *Convolution*

Secara sederhana, konvolusi merupakan operasi pengolahan citra dengan mengalikan sebuah citra dengan sebuah *mask* atau kernel. Kedua citra dan kernel merupakan sebuah matriks dengan ukuran tertentu.

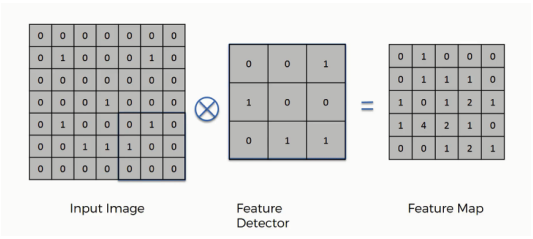

Fig2. ilustrasi operasi konvolusi

Secara teknis, citra yang dibaca pada berbagai bahasa pemrograman akan diubah menjadi matriks ataupun *array of array*. Dengan begitu, dapat dioperasikan secara langsung dengan berbagai operasi valid matriks. Secara umum, gambar digital dibentuk dengan perpaduan warna RGB sehingga memiliki tiga lapisan ke dalaman citra yang merepresentasikan *red, green,* dan *blue* untuk masing-masing lapisannya.

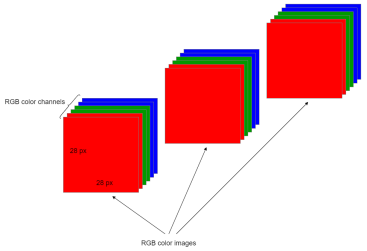

Fig3. Struktur lapisan kedalaman dari model rgb

Akan tetapi, citra *grayscale* umumnya hanya berukuran satu dimensi kedalamnnya karena hanya perlu menunjukkan satu baris rentang warna dari antara hitam dan putih. Citra skala-abu ini tidak membutuhkan kombinasi beberapa skala warna. Pada model warna HSI, keabuan dapat diperoleh dengan mengambil nilai intensitas gambar tersebut.

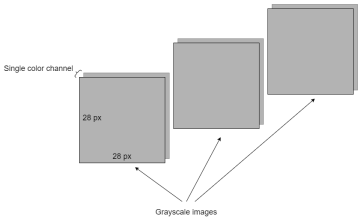

Fig4. Struktur lapisan kedalaman pada citra *grayscale*

Berbeda halnya dengan ukuran citra masukan, matriks kernel biasanya berukurang ganjil seperti 1x1, 2x2, dan seterusnya. Hal tersebut dikarenakan kernel ganjil menghasilkan matriks keluaran berukuran ganjil pula. Jika genap, maka matriks keluaran akan berukuran genap pula. Di samping padding yang dihasilkan bukan bilangan bulat, citra yang dihasilkan juga tidak memiliki titik piksel pusat karena ukurannya yang genap.

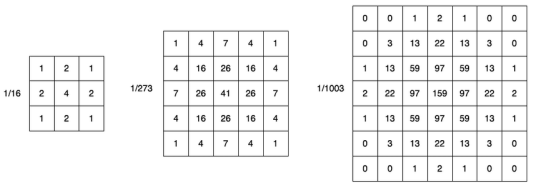

Fig5. Gaussian kernel berbagai ukuran untuk pelembutan citra

Secara teknis, proses konvolusi didefinisikan sebagai persamaan berikut.

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da$$

dimana g(x) merupakan kernel atau *mask* konvolusi dan f(x) matriks citra masukan.

1. Tahapan Operasi Konvolusi

   Operasi konvolusi dikalikan dengan melakukan operasi dot produk matriks citra masukan dengan matriks kernel. Setiap tahapannya merupakan hasil pergeseran kernel ke arah kanan-bawah dengan prioritas ke kanan. Untuk setiap tahapannya, dilakukan dot produk antara kernel dengan bidang matriks yang bersesuaian.

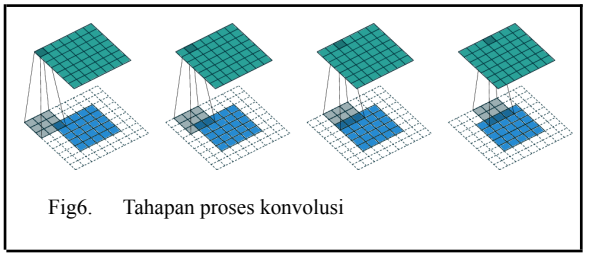

Fig6. Tahapan proses konvolusi

Setiap hasil dari dot produk tersebut merupakan nilai dari setiap elemen pada matriks keluaran. Matrisk keluaran tersebut disebut *feature map*. Ukuran dari matriks keluaran tersebut adalah sebagai berikut.

$$d * (x - y + 1)$$

dimana d merupakan ukuran dimensi misalnya 3 untuk matriks 3 dimensi, x merupakan ukuran matriks masukan, dan y merupakan ukuran dari kernel. Akan tetapi, jika diterapkan berbagai teknik padding, matriks keluaran dapat berukuran sama dengan matriks masukan.

B. Model Warna

Terdapat berbagai model warna yang populer digunakan pada gambar digital antara lain RGB, BGR, CMY, CMYK, YCbCr, HSI, XYZ, dan lainnya.

1. *Red, green, blue* model (RGB)

   Model warna ini merupakan model warna paling umum digunakan saat ini dalam proses digital.

Berbagai monitor saat ini secara umum menggunakan model RGB.

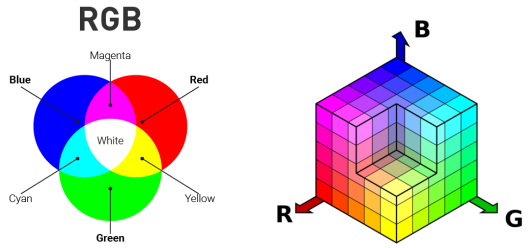

Fig7. Ilustrasi model RGB

CIE (*Commision International de I'Eclairage*) menstandarkan panjang gelombang warna-warna pokok RGB sebagai berikut.

$$R: 700\ nm$$
$$G: 546.1\ nm$$
$$B: 435.8\ nm$$

Diagonal utama pada model RGB kubus merepresentasikan tingkat keabuan dari gambar.

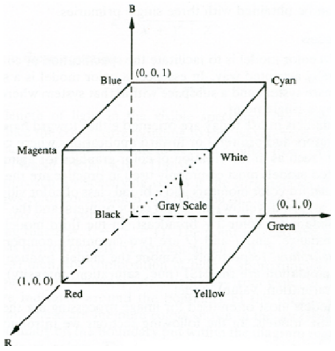

Fig8. Model RGB dengan *grayscale diagonal*

2. *Blue, green, red* model (BGR)

Model warna BGR pada dasarnya sama dengan RGB dengan urutan lapisan yang berbeda yaitu kebalikan dari RGB. Model warna ini digunakan pada sistem layar perangkat digital lama. Akan tetapi, model ini masih digunakan oleh kakas CV2 yang sangat populer pada pemrosesan gambar dengan menggunakan python. Semua citra yang ingin diproses oleh CV2 perlu dikonversi terlebih dahulu ke dalam bentuk BGR.

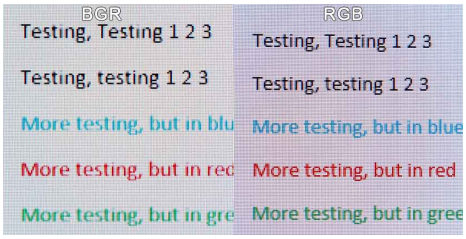

Fig9. Perbedaan ketajaman gambar RGB dan BGR

Walaupun perbedaannya hanya pada urutan lapisan saja, hasil dari model ini terlihat berbeda pada segi ketajaman gambar. Untuk mengatasinya, dapat memperbesar skala gambar karena perbedaan terlihat jelas hanya jika gambar tersebut kecil.

3. *Grayscale*

Citra dengan warna di antara hitam dan putih dengan beberapa tingkatan di antaranya. Citra ini hanya memiliki satu lapisan kedalaman yang berbeda dengan citra RGB memiliki tiga lapisan kedalaman. Pada makalah ini, terdapat konversi warna dari citra BGR ke *grayscale* melalui fitur yang disediakan pada kakas pemrosesan gambar python, OpenCV. Berikut ini definisi persamaan yang digunakan.

$$RGB[A]\ to\ Gray:$$
$$Y \leftarrow 0.299 \,.\, R + 0.587 \,.\, G + 0.114 \,.\, B$$

dimana warna hijau memiliki pengaruh yang lebih besar melalui persentasenya yang lebih tinggi. Selain itu, tersedia juga konversi dari *grayscale* ke RGB maupun BGR. Berikut ini persamaan yang digunakan.

$$Gray\ to\ RGB[A]:$$
$$R \leftarrow Y, G \leftarrow Y, B \leftarrow Y, A \leftarrow max(ChannelRange)$$

C. Haar Feature-based Cascade Classifier

Struktur wajah memiliki pola yang cukup unik sehingga dapat dikenali dan diukur secara matematis. Hal tersebut merupakan sistem pengenalan objek yang dimanfaatkan oleh teknik ini. Untuk mendeteksi objek, dilakukan pengecekan sisi gelap terang dari objek melalui beberapa pola. Pada *cascade classifier*, kernel yang digunakan juga disebut sebagai fitur. Berikut ini beberapa fitur dalam *cascade classifer* untuk mendeteksi berbagai objek.

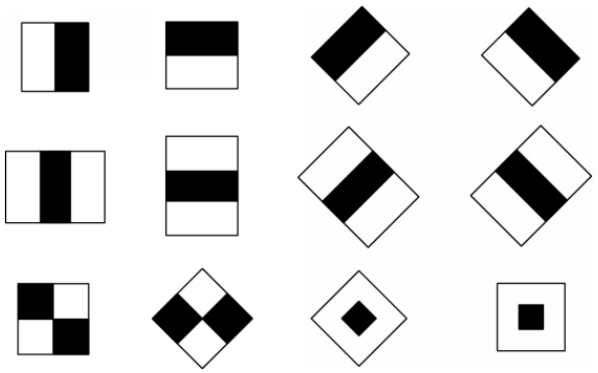

Fig10. Berbagai contoh fitur pada *cascade classifier*

Dalam representasi kernel, sisi putih merepresentasikan nilai 1 dan sisi gelap merepresentasikan -1. Dari konsep

tersebut dapat disimpulkan bahwa pendeteksian didasarkan pada besar perbedaan nilai piksel pada beberapa sisi.

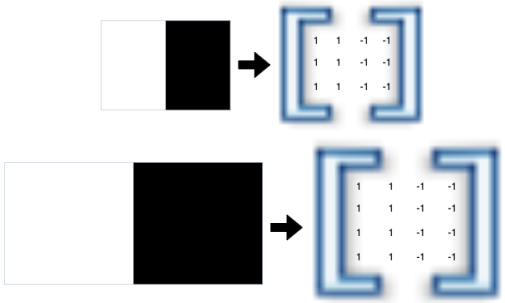

Fig11. Representasi fitu dalam bentuk kernel matriks

Setiap piksel pada gambar akan dikonvolusi dengan matriks konvolusi tersebut. Semakin besar perbedaan (nilai), maka semakin bagus hasil yang didapatkan. Jika nilai yang didapatkan mendekati 0, maka dapat disimpilkan hampir tidak terdapat perbedaan di antara kedua sisi tersebut. Dari hal tersebut, jika ternyata mendekati 0, maka proses tidak bisa dilanjutkan dan piksel tersebut diklasifikasikan sebagai negatif yang artinya tidak memenuhi kriteria berdasarkan fitur yang ditetapkan. Begitu pula sebaliknya, jika ternyata nilai yang didapatkan tinggi, maka diklasifikasikan positif dan akan dilanjutkan ke proses selanjutnya.

Untuk dapat menentukan apakah nilai yang didapatkan memenuhi atau tidak, diterapkan suatu threshold tertentu sehingga dapat diklasifikasikan sebagai positif atau negatif. Jika klasifikasi yang didapatkan adalah positif, maka dilanjutkan pengecekan fitur selanjutnya. Pengecekan fitur lain tersebut biasanya mengoperasikan matriks kernel (fitur) dengan matriks gambar pada bagian atau sekitaran operasi konvolusi sebelumnya. Dari hal tersebut dapat disimpulkan bahwa terdapat rangkaian pengecekan fitur yang dilakukan untuk mengecek apakah suatu objek terdapat pada gambar atau tidak. Pengecekan objek yang berbeda tentunya menggunakan rangkaian pengecekan yang berbeda.

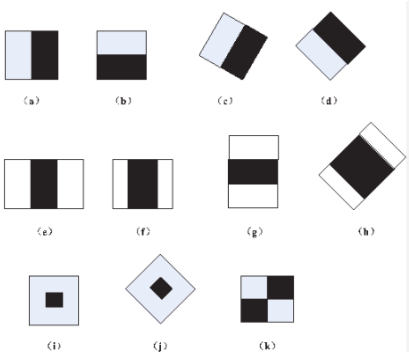

Fig12. Urutan fitur untuk pengecekan mata menatap

Fitur yang dapat kita definisikan secara keseluruhan tentunya banyak. Padahal pengecekan yang dilakukan tidak hanya terkait jenis fiturnya saja melainkan kombinasi dari lokasi pengecekan pada citra, ukuran kernel, dan juga jenis kernel itu sendiri. Jika dilakukan pengecekan terhadap semua fitur tentunya membutuhkan komputasi yang sangat besar dan tidak relevan. Oleh karena itu, perlu dipilih beberapa fitur yang tepat yang cukup dan tidak *overfit* terhadap objek yang akan kita cari.

1. Pengolahan citra masukan sebelum proses konvolusi dengan fitur

   Nilai pada citra masukan tidak secara langsung digunakan pada proses konvolusi. Nilai setiap elemen pada matriks tersebut perlu diubah menjadi penjumlahan seluruh nilai elemen matriks pada kombinasi seluruh nilai indeks sebelumnya atau disebut juga dengan integral dari gambar.

$$I(x,y) = \sum_{x' < x,\, y' < y} i(x', y')$$

Jika nilai yang akan dicari ada pada indeks (x,y), maka nilainya adalah penjumlah nilai elemen pada matriks dengan rentang indeks x adalah [0,x] dan y adalah [0,y].

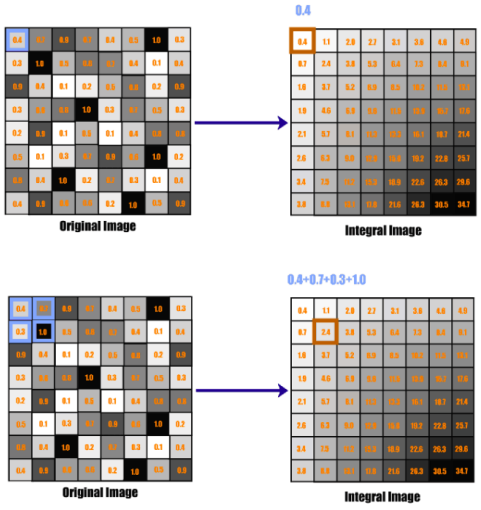

Fig13. Operasi integral pada citra masukan

Pemrosesan seperti ini membantu mengurangi waktu komputasi yang dibutuhkan pada citra. Secara konvensional, pencarian nilai rata-rata bilangan dalam suatu bidang pada citra dilakukan dengan menjumlahkan setiap nilai dan membagikannya dengan jumlah anggota yang tergabung. Untuk bidang tersebut yang mengandung 6 elemen dan jumlah komputasi dengan fitur yang sama adalah 1000, maka total komputasinya adalah 1000 * 6 (5 penjumlahan + 1 pembagian) = 6000 operasi. Akan

tetapi, dengan teknik ini operasi dasar yang dibutuhkan 4 operasi (2 penjumlahan, 1 pengurangan, 1 pembagian) sehingga total operasinya hanyalah 1000 * 4 = 4000 operasi. Hal tersebut tentunya merupakan perbedaan yang cukup signifikan sehingga tentunya menguntungkan jika menerapkan teknik pemrosesan ini.

2. Algoritma Adaboost

Proses ini digunakan untuk mendapatkan fitur-fitur yang relevan dan terbaik untuk digunakan dalam pengecekan objek yang kita miliki. Untuk mendapatkan hasil tersebut, perlu dilakukan proses pelatihan gambar untuk mendapatkan threshold terbaik untuk mengklasifikasikan apakah positif atau negatif. Proses *training* ini dilakukan hingga ditemukan sejumlah fitur yang dibutuhkan atau ditemukannya fitur dengan error rate rendah. Fitur-fitur dengan error rendah tersebut tentunya dapat mengklasifikasikan wajah atau tidak secara akurat. Dengan begitu, fitur-fitur yang didapatkan jauh lebih rendah yaitu 6000 fitur saja dibandingkan fitur awal yang lebih dari 160.000 jumlah fitur.

3. Cascade Classifier

Teknik ini bekerja untuk menentukan bagaimana pengurutan pengecekan fitur yang baik untuk dilakukan. Setelah setiap fitur didapatkan dari hasil pelatihan melalui AdaBoost, setiap fitur akan dikelompokkan terlebih dahulu dimana setiap fitur pada satu kelompok akan lebih relevan. Sebagai contoh, terdapat fitur yang cenderung pengecekannya di posisi (1,1) dan ada fitur lain yang pengecekannya di posisi (255,255). Tentunya kedua fitur tersebut tidak relevan jika disatukan karena harusnya posisinya tidak jauh antara kedua fitur jika ingin mendeteksi wajah. Selain itu, pertimbangan lain adalah bentuk kernel dari fitur. Kernel yang setiap elemen matriksnya tidak akan disatukan ke kelompok yang sama.

Setelah pengelompokan, maka akan diurutkan pengecekan setiap fitur pada satu kelompok. Ketika pengecekan dilakukan, fitur yang tidak klasifikasi negatif akan dihentikan dan dilanjutkan ke posisi selanjutnya atau kelompok fitur selanjutnya. Hal ini tentunya dapat mengurangi waktu komputasi.

III. Implementasi

Berikut ini proses implementasi yang dilakukan. Pada bagian ini akan dijelaskan bagaimana tahapan pengerjaan, penjelasan kode program, hingga hasil pengujian yang didapatkan.

B. Tahapan Pengerjaan

Berikut ini flowchart tahapan pengerjaan yang penulis lakukan.

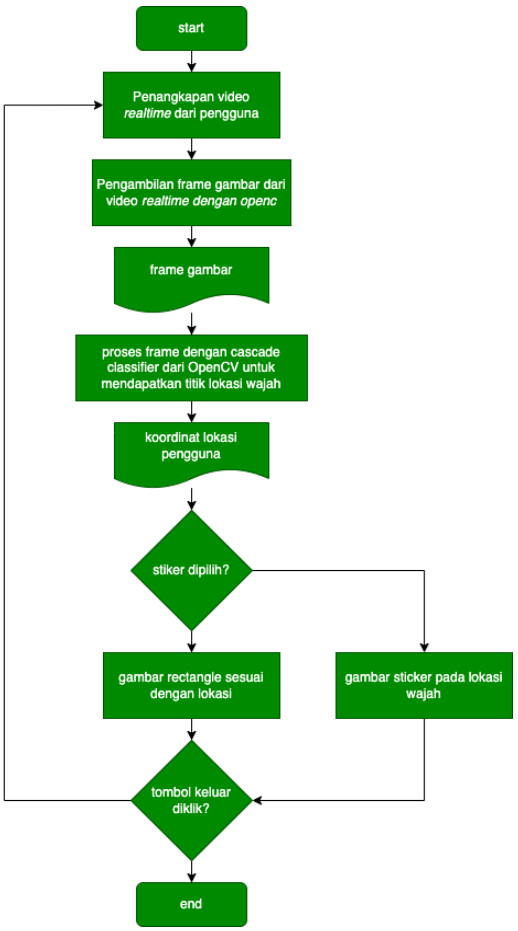

Fig14. Diagram alir dari sistem pendeteksian wajah dan penambahan stiker

Secara umum, tahapan kerja kode dibagi menjadi sistem pendeteksian wajah dan penambahan stiker. Pendeteksian wajah diawali dengan pembacaan frame gambar dari video *realtime webcam* pada aplikasi. Setelah itu, gambar tersebut dideteksi keberadaan wajahnya melalui model pembelajaran mesin *pre-trained* yang disediakan oleh OpenCV yaitu cascade classifier dengan menggunakan teknik haar feature-based cascade classifier. Model tersebut akan mengembalikan posisi wajah dalam empat koordinat. Berdasarkan kode tersebut, ditampilkan stiker atau *rectangle default* sesuai dengan pilihan pengguna yang ditimpa di area wajah tersebut.

B. Kode program

Terdapat beberapa potongan kode yang diimplementasikan ke dalam beberapa fungsi. Fungsi fungsi tersebut dijalankan dan disatukan dengan GUI yang dibangun dengan kakas bernama Tkinter pada python. Untuk membuat berbagai fitur dalam aplikasi, pertama perlu menginstansiasi objek tkinter window.

```
# VIDEO STREAM
VS = VideoStream(0).start()
```

```
time.sleep(2.0)

window = tk.Tk()
window.geometry("729x356")
window.title('Face Detection')
window.wm_protocol("WM_DELETE_WINDOW", lambda:on_close())
```

Untuk dapat menerima masukan video dari kamera dan menampilkannya pada sistem, perlu diimplementasikan *threading* sehingga tidak terjadi *blocking* atau *not responding* pada jendela aplikasi ketika sistem juga bekerja untuk membaca kamera.

```
# start a thread that constantly pools the video sensor for
# the most recently read frame
STOP_EVENT = threading.Event()
THREAD = threading.Thread(target=video_loop, args=())
THREAD.start()
window.mainloop()  # Keep the window open
```

Selama jendela dari antarmuka aplikasi tidak ditutup, kamera akan selalu berjalan dan membaca setiap *frame* dari *video camera* secara terus menerus. *Frame* tersebut diambil untuk ditampilkan di antarmuka sebagai cermin diri kita. Akan tetapi, sebelum ditampilkan, akan dilakukan pemrosesan gambar pada *frame* yaitu pendeteksian wajah ini. Secara *default,* sistem mendeteksi wajah pengguna dan menunjukkan area wajah di dalam *hollow rectangle* berwarna merah.

Pengolahan citra tahap awal yang dilakukan adalah mengubah ukuran dari gambar. Setelah itu, model warna yang dari *frame* dikonversi yang sebelumnya adalah BGR (cv2 menggunakan model warna tersebut ketika membaca kamera) menjadi RGB. Wajah tersebut kemudian ditampilkan ke dalam antarmuka dan selanjutnya dilakukan proses penempelan stiker dengan melakukan *overlapping* pada *frame* wajah yang telah ditampilkan pada antarmuka sebelumnya.

```
def video_loop():
    global PANEL
    global FRAME
    global LAST_FRAME

    try:
        while not STOP_EVENT.is_set():
            FRAME = VS.read()
            FRAME = imutils.resize(FRAME, width=300)

            image = cv2.cvtColor(FRAME, cv2.COLOR_BGR2RGB)
            image = Image.fromarray(image)
            width, height = image.size
            image = image.resize((width*2,height*2))
            image = ImageTk.PhotoImage(image)
            # if the panel is not None, we need to initialize it
            if PANEL is None:
                PANEL =
tk.Canvas(height=image.height(),width=image.width(), bg="white")
                PANEL.grid(row=1, column=1, rowspan=4,
padx=10,pady=10)
                LAST_FRAME = PANEL.create_image(0,0, image=image,
anchor=tk.NW)
                PANEL.image = image
            # otherwise, simply update the panel
            else:
                PANEL.itemconfig(LAST_FRAME, image=image)
                PANEL.image = image
            if(CURR_MODE == MODE["DEFAULT"] and len(RECTANGLE_ID) >
0):
                for id in RECTANGLE_ID:
                    PANEL.delete(id)
            show_sticker()
    except RuntimeError:
        print("[INFO] caught a RuntimeError")
```

Proses selanjutnya merupakan penambahan stiker maupun *rectangle*. Untuk menambahkan stiker pada wajah, perlu diketahui dimana lokasi wajah pada gambar. Mencapai hal tersebut, dilakukan instansiasi model pembelajaran mesin yang dengan memanfaatkan teknik *haar cascade classifier*. Pada implementasi saat ini, model yang digunakan didapatkan dari *pre-trained* model OpenCV untuk mendeteksi tampilan depan wajah. Beberapa *hyperparameter* yang digunakan adalah skala 1.1, minimum neighbors adalah 5, minimum size adalah (30,30). Untuk gambar sendiri, perlu diubah dari model warna BGR (model warna *default* CV2 mendeteksi kamera video) menjadi *grayscale.* Dari proses pendeteksian wajah yang dilakukan, akan didapatkan posisi koordinat wajah sebagai kembalian dari model pembelajaran. Dari setiap titik yang didapatkan, digambarkan stiker maupun *rectangle* sesuai dengan mode yang dipilih pengguna dan melakukan penskalaan terhadap setiap stiker dan *rectangle* yang akan ditampilkan sehingga cocok dengan ukuran lebar dan tinggi dari wajah.

```
def show_sticker():
    global FRAME
    global STICKER_IMAGE
    global STICKER_PHOTO_IMAGE
    global STICKER_WIDTH
    global STICKER_HEIGHT
    global RECTANGLE_ID
    global STICKER_ID

    cascPath = "data/haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(cascPath)

    FRAME_TO_DETECT = imutils.resize(FRAME, width=300*2, height=
168*2)

    gray = cv2.cvtColor(FRAME_TO_DETECT, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )
```

```
    if(CURR_MODE == MODE["STICKER"] and STICKER_IMAGE is None):
        STICKER_IMAGE = Image.open('.' + STICKER_PATH + '/' +
STICKER_NAME)
        rgba = STICKER_IMAGE.convert("RGBA")
        data = rgba.getdata()

        new_data = []
        for item in data:
            if(item[0] == 255 and item[1] == 255 and item[2] ==
255):
                new_data.append((255,255,255,0))
            else:
                new_data.append(item)

        rgba.putdata(new_data)

        STICKER_IMAGE = rgba

    for (x, y, w, h) in faces:
        if(CURR_MODE == MODE["DEFAULT"]):

RECTANGLE_ID.append(PANEL.create_line(x,y,x+w,y,x+w,y+h,x,y+h,x,y,f
ill="red"))
        else:
            if(not (STICKER_WIDTH == w and STICKER_HEIGHT == h)):
                STICKER_WIDTH = w
                STICKER_HEIGHT = h
                sticker_resized = STICKER_IMAGE.resize((w,h))
                STICKER_PHOTO_IMAGE =
ImageTk.PhotoImage(sticker_resized)

            STICKER_ID.append(PANEL.create_image(x,y,
image=STICKER_PHOTO_IMAGE, anchor=tk.NW))
```

Untuk beberapa fitur tambahan dari sistem, pengguna dapat memilih tiga stiker yang disediakan. Setiap pemilihan stiker dan pembatalan penggunaan stiker, terdapat fungsi yang dijalankan untuk mengubah status *state* sistem dari mode *default* menjadi *sticker* ataupun sebaliknya. Pada kondisi ini, beberapa global variabel juga akan direset nilainya.

```
def change_sticker_status(sticker_name):
    global CURR_MODE
    global SHOW_STICKER
    global STICKER_NAME
    global STICKER_ID

    # TO FLUSH
    global STICKER_IMAGE
    global STICKER_PHOTO_IMAGE
    global STICKER_WIDTH
    global STICKER_HEIGHT

    STICKER_IMAGE = None
    STICKER_PHOTO_IMAGE = None
    STICKER_HEIGHT = None
    STICKER_WIDTH = None

    for id in STICKER_ID:
        PANEL.delete(id)

    for id in RECTANGLE_ID:
        PANEL.delete(id)

    CURR_MODE = not CURR_MODE

    SHOW_STICKER = not SHOW_STICKER
    if(STICKER_NAME is None):
        STICKER_NAME = sticker_name
    else:
        STICKER_NAME = None
```

Selain memilih stiker yang telah ada, sistem juga mendukung pencarian gambar dari masukan pengguna dari tombol *browse* yang disediakan di sisi kanan-bawah antarmuka.

C. Hasil Pengujian

Berikut ini tampilan hasil pengujian dari sistem yang telah dikembangkan.

1. Tampilan *default*

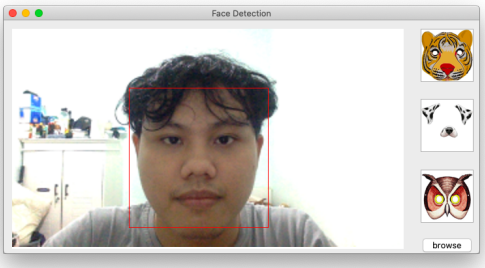

Fig15. Tampilan *default* dari antarmuka aplikasi

Pada kondisi *default,* ditampilkan *frame* wajah pengguna dan adanya segi empat berwarna merah di area sekitar wajah untuk menunjukkan wajah. Selain itu, terdapat tiga stiker dasar di sisi kanan antarmuka dan tombol *browse* di sisi kanan-bawah untuk mencari stiker khusus sesuai keinginan pengguna.

2. Tampilan menggunakan tiga stiker dasar

Fig16. Tampilan dengan stiker harimau

Fig17. Tampilan dengan stiker anjing

Fig18. Tampilan dengan stiker burung hantu

3. Tampilan akses stiker masukan pengguna

Berikut ini tampilan ketika ingin mencari stiker secara *custom*.

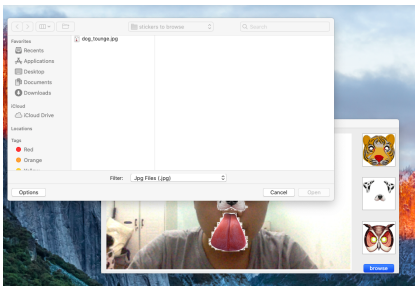

Fig19. Tampilan ketika mencari stiker pada *filesystem*

Fig20. Tampilan ketika mencari stiker pada *filesystem*

IV. Kesimpulan

Pengolahan citra merupakan proses yang penting di dunia digital saat ini. Di balik operasi sederhana pengolahan citra seperti konvolusi, konversi warna, maupun integral gambar, diciptakan berbagai sistem canggih yang sangat bermanfaat saat ini hingga seterusnya. Sistem deteksi wajah merupakan salah satu produk digital yang paling banyak digunakan saat ini. Selain penerapannya dalam penambahan *filter* wajah yang indah, sistem pendeteksian wajah juga digunakan sebagai basis sistem pendeteksi anti penipuan, deteksi penggunaan masker, dan sebagainya. Dari sisi yang lebih umum, teknik *haar cascade classifier* yang digunakan pada kasus ini juga dapat digunakan untuk mendeteksi objek lain selain wajah melalui berbagai kombinasi fitur hasil pelatihan model pembelajaran mesin.

Pada sistem penambahan stiker pada wajah yang dibangun ini, pengguna telah dimungkinkan untuk membubuhkan stiker pada tampilan wajah dengan cukup sempurna pada kasus stiker yang ukurannya sebesar proporsi wajah. Akan tetapi, jika stiker yang dimasukkan merupakan stiker khusus hidung atau mata, stiker tidak dapat diletakkan dengan tepat karena sistem tidak menerapkan sistem pendeteksian mata dan hidung secara spesifik, melainkan keseluruhan bidang wajah.

Pranala Terkait

Kode dari pengimplementasian program dapat diakses pada *repository* di sini.

Video penjelasan konsep, pengimplementasian, dan pengujian dari aplikasi yang dibuat dapat diakses di sini.

Ucapan Terima Kasih

Penulis mengucapkan puji dan syukur atas berkat dan rahmat dari Tuhan Yang Maha Esa karena dapat menyelesaikan makalah dengan judul "Sistem Pendeteksian Wajah untuk Aplikasi Penambahan Stiker Pada Wajah *Realtime*" dengan baik untuk memenuhi tugas mata kuliah IF4073, Interpretasi dan Pengolahan Citra di tahun ajaran 2022/2023. Penulis juga mengucapkan terima kasih kepda bapak Dr. Rinaldi Munir, S.T., M.T. selaku dosen pengampu mata kuliah ini yang telah banyak memberikan bimbingan dan ilmu terkait pengolahan citra. Tidak lupa juga penulis mengucapkan terima kasih kepada keluarga dan teman-teman yang selalu mendukung dan memberikan semangat kepada penulis selama menyusun makalah ini.

Referensi

[1] Munir, Rinaldi. "IF4073 Interpretasi dan Pengolahan Citra - Semester I Tahun 2022/2023" https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/citra22-23.htm diakses pada 18 Desember 2022.

[2] https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/ diakses pada 18 Desember 2022.

[3] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html diakses pada 18 Desember 2022.

[4] https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray diakses pada 18 Desember 2022.

[5] https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77 diakes pada 18 Desember 2022.

[6] https://medium.com/swlh/haar-cascade-classifiers-in-opencv-explained-visually-f608086fc42c diakses pada 18 Desember 2022

[7] https://github.com/opencv/opencv/tree/4.x/data/haarcascades diakses pada 18 Desember 2022.

[8] https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77 diakses pada 18 Desember 2022.

[9] https://github.com/shantnu/FaceDetect diakses pada 17 Desember 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2022

Kevin Katsura Dani Sitanggang
13519216