Tugas UAS
IF4073 Interpretasi dan Pengolahan Citra
Dokumentasi Teknis Aplikasi INSTACIT

Disusun Oleh

Stephen	13512025
Jeffrey Lingga Binangkit	13512059
Ahmad Zaky	13512076

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2015/2016

Daftar Isi

A. Pendahuluan

Aplikasi yang kami buat bernama ***Instacit*** yang memungkinkan pengguna untuk mengunggah foto. Setelah foto diunggah, akan dikenali siapa saja yang ada di foto tersebut berdasarkan data yang didapat dari anggota kelas Pengolahan Citra. Pengguna dapat pula menerapkan *outline effect* untuk foto ini berdasarkan deteksi *edge.*

Untuk mengenali foto, aplikasi ini menerapkan konsep-konsep sebagai berikut:

1. Mendeteksi lokasi yang merupakan wajah dari manusia. Pendeteksian dilakukan dengan berdasarkan.
    a. Warna kulit yang ada dalam palet.
    b. Proporsi bentuk kepala dan mencocokkan lokasi-lokasi mana yang harus berwarna gelap (mata, mulut).
2. Gaussian blur, untuk menghaluskan citra wajah.
3. Prewitt operator, untuk mendeteksi *edge* dari elemen-elemen wajah.
4. K-Means, untuk melakukan pengelompokan sebuah citra wajah menjadi elemen-elemen : mata kanan, mata kiri, hidung, mulut.
5. Algoritma Brensenham, untuk menghubungkan titik-titk dalam data Face Net.
6. Pencocokan grid untuk masing-masing elemen wajah.

B. Algoritma

Untuk memperoleh hasil rekognisi, aplikasi ini memproses gambar dengan mengikuti urutan sebagai berikut.

1. **Mempersiapkan model-model wajah peserta kuliah**

   Untuk setiap data yang ada di dalam database FaceNet, konversikan menjadi 4 *cluster*.

   a. *Cluster* mata kanan

      i. Hubungkan titik-titik alis kanan (prefix aa) menggunakan algoritma Bresenham

      ii. Hubungkan titik-titik mata kanan (prefix ma) menggunakan algoritma Bresenham

      iii. Cari x_{min}, x_{max}, y_{min}, y_{max}, dari titik-titik tersebut untuk menentukan *boundary*.

      iv. Konversikan ke koordinat lokal ($x_{min} = 0$, $y_{min} = 0$)

      v. *Resize* gambar menjadi 60x40

   b. *Cluster* mata kiri

      i. Hubungkan titik-titik alis kanan (prefix ai) menggunakan algoritma Bresenham

      ii. Hubungkan titik-titik mata kanan (prefix mi) menggunakan algoritma Bresenham

      iii. Cari x_{min}, x_{max}, y_{min}, y_{max}, dari titik-titik tersebut untuk menentukan *boundary*.

      iv. Konversikan ke koordinat lokal ($x_{min} = 0$, $y_{min} = 0$)

      v. *Resize* gambar menjadi 60x40

   c. *Cluster* hidung

      i. Hubungkan titik-titik hidung (prefix h) menggunakan algoritma Bresenham

      ii. Cari x_{min}, x_{max}, y_{min}, y_{max}, dari titik-titik tersebut untuk menentukan *boundary*.

      iii. Konversikan ke koordinat lokal ($x_{min} = 0$, $y_{min} = 0$)

      iv. *Resize* gambar menjadi 40x60

   d. *Cluster* mulut

      i. Hubungkan titik-titik bibir atas (prefix mia) menggunakan algoritma Bresenham

ii. Hubungkan titik-titik bibir bawah (prefix mb) menggunakan algoritma Bresenham

iii. Cari x_{min}, x_{max}, y_{min}, y_{max}, dari titik-titik tersebut untuk menentukan *boundary*.

iv. Konversikan ke koordinat lokal ($x_{min} = 0$, $y_{min} = 0$)

v. *Resize* gambar menjadi 60x40

Telah terbentuk model untuk seluruh mahasiswa peserta kuliah. Model-model ini akan dibandingkan dengan yang didapatkan dari foto untuk melakukan pengenalan.

2. Memasukkan gambar

Gambar diinput melalui Kamera atau melalui Galeri.

3. Mendeteksi lokasi wajah

Program telah memiliki palet warna yang berisi *range* warna kulit anggota kelas.

- Menelusuri gambar
- Bila ditemukan sebuah pixel warna yang merupakan anggota palet
    - Melakukan penelusuran ke pixel-pixel lain di sekitar pixel tersebut yang warnanya juga merupakan anggota dari palet warna. Penelusuran dilakukan secara *flooding* (mirip *floodfill*).
    - Telah ditemukan kotak terduga wajah, disimpan *boundary*-nya (x_{min}, x_{max}, y_{min}, y_{max}).
- Setelah ditemukan lokasi-lokasi yang diduga wajah, untuk memastikan bagian gambar tersebut merupakan wajah, maka dilakukan pengecekan sebagai berikut
    - Pengecekan proporsi : wajah memiliki proporsi tinggi:lebar sebesar 1:1, sampai 1:1,7. Bila proporsinya tidak cocok, maka dikeluarkan dari kandidat wajah.
    - Pengecekan mata : pada sekitar 1/3 atas wajah, seharusnya ada dua bagian yang lebih gelap yaitu mata. Bila tidak ada, maka kandidat ini dikeluarkan.
    - Pengecekan mulut : pada sekitar ¼ bawah wajah, seharusnya ada sebuah bagian yang berbeda warnanya yaitu mulut. Bila tidak ada, maka dikeluarkan dari kandidat wajah.
- Telah ditemukan kandidat-kandidat wajah.

4. Memperhalus citra wajah menggunakan Gaussian Blur

Setelah didapatkan wajah-wajah yang akan dideteksi, maka dilakukan penghalusan wajah dengan menggunakan algoritma Gaussian Blur.

Algoritma Gaussian Blur adalah sebagai berikut :

- Untuk setiap pixel dalam gambar di lokasi (x,y), terapkan rumus distribusi Gauss sebagai berikut

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  - X dan Y di sini adalah koordinat di sumbu horizontal dan vertikal di bidang kartesian. σ adalah standar deviasi dari distribusi gauss.
  - Terbentuk sebuah convolution matrix. Setiap pixel diberikan nilai baru yang merupakan rata-rata bobot pixel tetangganya, dimana titik original memiliki bobot terbesar, dan tetangga-tetangganya semakin jauh dari titik original akan semakin kecil bobotnya. Convolution matrix yang perlu dihitung adalah berukuran 6σ x 6σ.

- Telah terbentuk citra-citra wajah yang telah lebih halus.

5. Mengubah gambar menjadi gambar outline menggunakan Sobel Operator

Untuk mempermudah pemrosesan, gambar diubah menjadi *outline* saja sehingga tidak terlalu terpengaruh masalah pencahayaan dan bayangan. Yang digunakan dalam aplikasi ini adalah :

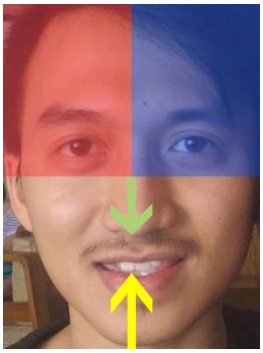
- Siapkan gambar.
- Terapkan *Horizontal Mask* yang ada di Sobel Operator. Horizontal mask dalam Sobel Operator berbentuk matriks di bawah ini:

-1	-2	-1
0	0	0
1	2	1

6. Melakukan clustering dengan K-Means

Untuk setiap wajah berbentuk outline yang didapatkan dalam tahap sebelumnya, dilakukan pengelompokan untuk mata, hidung, dan mulut. Pengelompokan ini dilakukan dengan algoritma clustering K-Means yang dimodifikasi, yang dapat dilihat sebagai berikut.

- Ubah gambar menjadi grayscale, tentukan threshold 182, kemudian ubah gambar menjadi biner sehingga kulit biasa menjadi putih, dan elemen-elemen mata, hidung, dan mulut menjadi hitam.
- Tentukan 4 buah *centroid*. Dalam algoritma K-Means biasa, *centroid* ditentukan secara acak. Namun, dalam menentukan *centroid* untuk kelompok mata kanan, kiri, hidung, dan mulut, digunakan cara heuristik.
    - Untuk mencari *centroid* mata kanan, hanya perlu menelusuri bagian merah saja, dari kanan ke kiri (sudut pandang kita) sampai ditemukan titik hitam.
    - Untuk mencari *centroid* mata kiri, hanya perlu menelusuri bagian biru saja, dari kiri ke kanan (sudut pandang kita) sampai ditemukan titik hitam.
    - Untuk mencari *centroid* hidung, dimulai dari titik tengah gambar, kemudian ditelusuri pixel-per-pixel ke arah bawah sampai ditemukan titik hitam. Dapat dilihat dengan panah warna hijau.
    - Untuk mencari *centroid* mulut, dimulai dari titik tengah-bawah gambar, kemudian ditelusuri pixel-per-pixel ke arah atas sampai ditemukan titik warna hitam. Dapat dilihat dengan panah warna cokelat.
- Lakukan langkah-langkah di bawah ini, sampai tidak ada anggota *cluster* yang berubah lagi:
    - Untuk setiap titik berwarna hitam, cari *centroid* yang terdekat darinya.
        - Masukkan titik tersebut ke *cluster* milik *centroid* tersebut.
    - Untuk setiap *cluster* mata kanan, kiri, hidung, dan mulut
        - Cari rata-rata nilai x dari seluruh titik yang ada di *cluster* tersebut.
        - Cari rata-rata nilai y dari seluruh titik yang ada di *cluster* tersebut.
        - Update koordinat *centroid* menjadi nilai (x, y) hasil rata-rata tadi.
- Setelah tidak ada perubahan, maka *cluster-cluster* mata, hidung, dan mulut sudah terbentuk dan stabil.



Gambar 1. Area yang Digunakan Dalam Penentuan Centroid Seed dengan Menggunakan Cara Heuristik

7. Melakukan pengenalan per cluster menggunakan Grid

Telah didapatkan 4 cluster untuk mata kanan, mata kiri, hidung, dan mulut. Masing-masing cluster dapat dikatakan merupakan gambar pula, yaitu gambar mata, hidung dan mulut. Pengenalan dapat dilakukan dengan cara berikut:

- Lakukan resize masing-masing gambar menjadi seukuran grid di bawah ini (horizontal x vertikal):
  - Mata : 60x40
  - Hidung : 40x60
  - Mulut : 60x40
- Untuk setiap peserta kuliah,
  - Untuk setiap model *cluster* gambar yang dimiliki peserta kuliah tersebut (yaitu model mata kanan, kiri, hidung, mulut)
    - Bandingkan dengan *cluster* gambar yang bersesuaian (hidung dengan hidung, dan seterusnya)
    - Hitung kesesuaiannya dengan rumus

$$similarity(cluster) = n_{correct} - n_{incorrect}$$

$n_{correct}$ adalah jumlah titik yang sama isinya di gambar *cluster* yang diproses dengan yang ada di model (di model hitam, di *cluster* gambar juga hitam).

$n_{incorrect}$ adalah jumlah titik yang tidak sama isinya di gambar *cluster* yang diproses dengan yang ada di model (di model hitam, di *cluster* gambar berwarna putih).

- Hitung nilai total kemiripan antara *cluster* gambar yang sedang diproses dengan *cluster* gambar yang dimiliki peserta kuliah ini dengan rumus

$$score = 0.25 * similarity(right_{eye}) + 0.25 * similarity(left_{eye}) + 0.25 * similarity(nose) + 0.25 * similarity(mouth)$$

- Pilih nama mahasiswa dengan *score* kemiripan tertinggi sebagai hasil pengenalan dari wajah tersebut.

8. Menampilkan ke pengguna

Pada setiap *area* yang dideteksi wajah, digambarkan kotak untuk menandai wajah tersebut. Di bagian bawah akan diberikan detail proses pendeteksian beserta dengan hasil klasifikasinya.

C. Aplikasi Instacit

1. Lingkungan Pengembangan dan Operasi

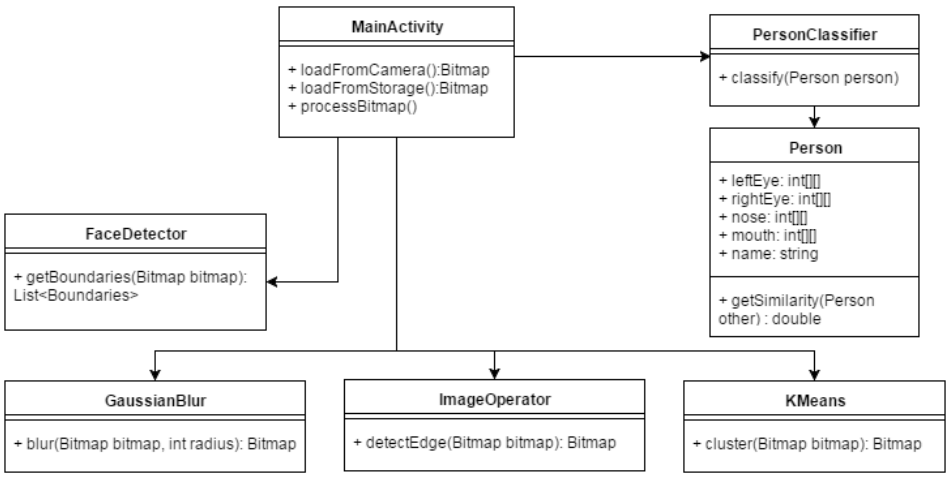
   Aplikasi ini dikembangkan menggunakan Android Studio dengan bahasa Java. Aplikasi ini dibuat untuk lingkungan operasi sebagai berikut:

   i. Platform : mobile.

   ii. Sistem operasi : Android v.4.4 (KitKat) dan versi-versi setelahnya.

2. Arsitektur Aplikasi

   Berikut adalah arsitektur aplikasi dalam bentuk diagram kelas.

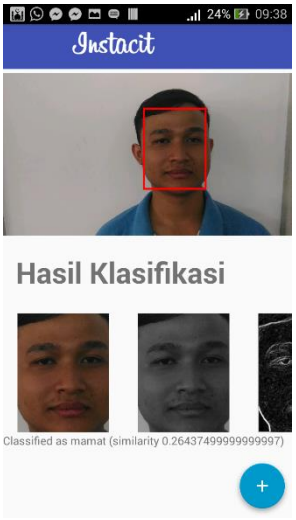


   Kelas utama adalah MainActivity yang mempunyai fungsi processBitmap(). Di dalamnya, gambar akan diproses sesuai urutan: FaceDetector, GaussianBlur, ImageOperator, Kmeans, dan PersonClassifier.

3. *Testcase Screenshot*

   Berikut adalah hasil klasifikasi beberapa citra.

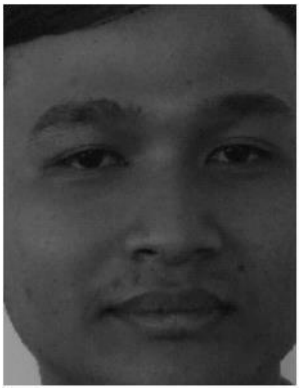
   Klasifikasi 1 wajah



Berikut adalah langkah-langkah yang ditempuh:

1. Hasil crop wajah

2. Konversi ke grayscale dan blurring dengan Gaussian Blur

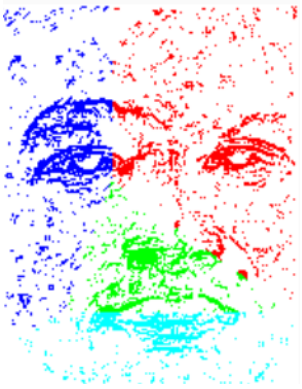


3. Deteksi edge dengan operator Prewitt

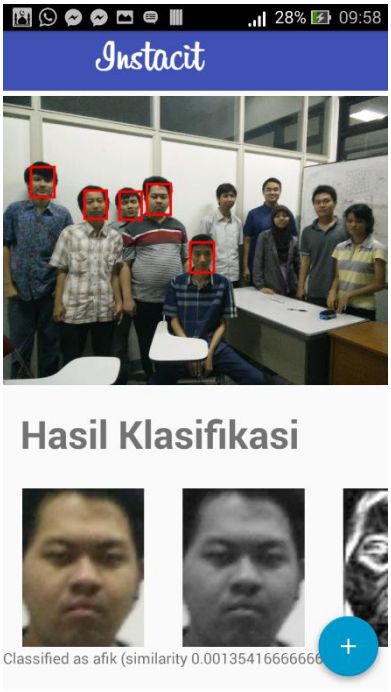
4. Hasil edge setelah diberikan treshold

5. Face boundary removal

6. Clustering komponen-komponen wajah



Klasifikasi banyak wajah



Pada kasus uji di atas, terdapat 5 wajah yang terdeteksi dari total 10 wajah. Meskipun demikian, masih belum semua wajah terklasifikasi dengan benar.

D. Kesimpulan

Metode mendeteksi wajah dengan menggunakan peta warna kulit sudah cukup baik. Meskipun demikian, tetap ada beberapa wajah yang belum

terdeteksi. Hal ini dapat diperbaiki dengan menambah atau memperkaya data warna kulit. Tapi *drawback*-nya adalah semakin banyak objek non-muka pula yang bisa jadi terdeteksi. Tapi hal itu dapat diatasi dengan mendeteksi area mata pada area dengan warna kulit.

Metode klasifikasi wajah dengan menggunakan Edge detection dan Grid masih belum baik untuk wajah yang berukuran kecil (contohnya wajah pada foto grup), karena hasil edge yang dihasilkan kurang akurat sehingga kurang menggambarkan kondisi wajah secara utuh. Selain itu, Grid kurang bisa menangani orientasi wajah yang berbeda2, contohnya saat mata dimiringkan sedikit, maka akan dianggap sangat berbeda.