Ubiquitous: Computers and its Applications Journal
Volume 2, Nomor 1, Juni 2019, 57-66

ISSN 2622-7983 online
ISSN 2622-7746 print

APLIKASI *MOBILE* PENGENALAN WAJAH SECARA *REAL-TIME* BERBASIS *PRINCIPAL COMPONENT ANALYSIS*

ADITYA DARMAWAN

Teknik Informatika, Fakultas Teknik
Universitas Maarif Hasyim Latif, Sidoarjo, Indonesia
e-mail: amarmawan@gmail.com

ABSTRAK

Manusia dapat dengan mudah mengenali wajah satu sama lain hanya dengan memandang. Penelitian ini mengimplementasikan pengenalan wajah berbasis pada Principal Component Analysis untuk mengembangkan aplikasi *mobile* secara *real-time*. Penelitian ini bertujuan untuk menghasilkan sebuah proses otentikasi melalui pengenalan wajah sebagai ganti memasukkan *username* dan *password*. *Euclidean distance* juga digunakan untuk mencari kemiripan terdekat antara citra *testing* dan citra *training*. Penelitian ini membuktikan bahwa kualitas citra dapat mempengaruhi hasil. Prapemrosesan harus dilakukan sebelum proses pengenalan untuk meminimalisir kualitas citra yang bervariasi karena semakin variatif kumpulan citra semakin tidak akurat hasilnya. Demikian juga semakin banyak citra *training* yang didapat maka semakin akurat hasil yang akan diperoleh.

Kata kunci: pengenalan wajah, *personal componetn analysis,* python *login*

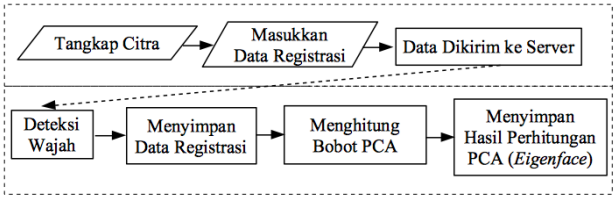
PENDAHULUAN

Teknologi komunikasi telah membersamai manusia sejak zaman prasejarah hingga zaman *now* dan akan terus membersamai. Semakin canggih teknologi komunikasi yang berkembang, ketergantungan komunikasi terhadap realitas jarak pun semakin berkurang. Hingga saat ini manusia dapat melakukan komunikasi *real-time* tanpa harus bertatap muka padahal pihak-pihak yang terlibat komunikasi dipisahkan oleh samudra dan daratan yang sangat jauh. Saat ini, dengan dikembangkannya teknologi dari teknologi sebelumnya, perangkat *mobile* seperti *smartphone* dan tablet sudah bergeser kedudukannya dari kebutuhan tersier (barang mewah) menjadi kebutuhan sekunder (kebutuhan mendesak yang bukan kebutuhan primer). Hampir semua kalangan memiliki nomor *handphone*. Tidak hanya itu, hampir semua pengguna *smartphone* memiliki nomor WhatsApp, Pin BBM, akun Facebook, serta akun-akun media sosial dan perpesanan instan lainnya.

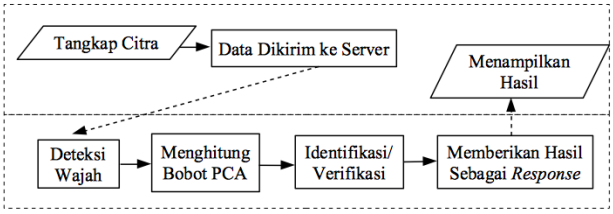
Tidak cukup sampai di situ. Telepon genggam yang di masa lalu hanya bisa digunakan untuk telepon dan sms saja kini berubah menjadi sebuah "komputer dalam genggaman". Banyak yang menjadikan *smartphone* sebagai pengganti komputer sehingga *smartphone* kini menjadi penunjang pekerjaan karena sistem operasi *mobile* saat ini telah memungkinkan pengembang membuat aplikasi yang canggih. Sistem pengenalan wajah adalah alternatif yang cepat untuk melakukan validasi pengganti *login* via *username* dan *password*. Dengan sistem pengenalan wajah, pengguna *smartphone* tidak perlu mengetik *username* dan *password* hanya untuk melakukan *login* dan menggunakan suatu aplikasi. Pengguna *smartphone* tidak perlu mengingat-ingat *password* dan tidak perlu khawatir lupa *password* karena *password*-nya adalah wajahnya sendiri.

METODE PENELITIAN

Berikut ini pada Gambar 1 dan Gambar 2 adalah diagram blok sistem pengenalan wajah secara *real-time* berbasis *Principal Component Analysis* (PCA).



Gambar 1. Diagram Blok Sistem Registrasi



Gambar 2. Diagram Blok Sistem *Login*

Sebagaimana yang diilustrasikan pada Gambar 1 dan Gambar 2, sistem yang akan dibangun dibagi menjadi dua, yakni blok registrasi dan blok

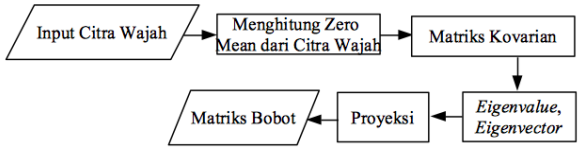
login. Blok registrasi merupakan blok pengambilan data sedangkan blok *login* merupakan blok pengenalan wajah. Proses registrasi dapat dijelaskan sebagai berikut:

1. Buka aplikasi *mobile* pada *smartphone* Android dan pilih menu registrasi, lalu akan diarahkan menuju pengambilan citra (*image capture*) melalui kamera.
2. Lakukan pengisian data pengguna, misal: nama, email, dan sebagainya.
3. Citra yang merupakan wajah dari hasil tangkapan kamera dikirim bersamaan dengan data yang dimasukkan pada langkah 2 ke *server* dengan menggunakan RESTful *web service* (dalam hal ini menggunakan metode POST).
4. Di *server* dilakukan deteksi wajah dari citra input. Sedangkan data yang dikirim selain data wajah disimpan.
5. Dari citra wajah hasil dari deteksi wajah tersebut, dilakukan proses penghitungan PCA.
6. Nilai dari hasil penghitungan PCA berupa *eigenface* disimpan di *server* sebagai acuan untuk melakukan pengenalan wajah pada tahap *login*.

Adapun proses *login* adalah sebagaimana yang digambarkan pada gambar 3.2 sebagai berikut:

1. Buka aplikasi *mobile* pada *smartphone* Android dan pilih menu *login*, lalu akan diarahkan menuju pengambilan citra (*image capture*) melalui kamera.
2. Citra yang merupakan wajah dari hasil tangkapan kamera dikirim ke *server* dengan menggunakan RESTful web service (dalam hal ini menggunakan metode POST).
3. Dari data wajah hasil deteksi wajah citra input tersebut dilakukan proses penghitungan PCA.
4. Hasil dari penghitungan PCA berupa *eigenface* dicocokan dengan data-data yang sudah ada sebelumnya, lalu dihasilkan bobot kemiripan terhadap masing-masing data.
5. Hasil dari pencocokan tersebut berupa data atau *error response* (jika data tidak ditemukan). Hasilnya kemudian dikirim ke *client* sebagai *response*.

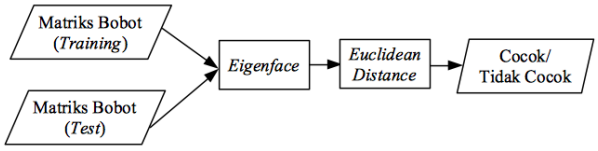
Gambar 3 adalah diagram blok pengenalan wajah (*face recognition*) berbasis *Principal Component Analysis*.



Gambar 3. Cara Kerja *Principal Component Analysis*

Gambar 3 menjelaskan cara kerja metode *Principal Component Analysis* yang dalam proses tersebut, citra wajah sebagai input diubah menjadi sebuah *vector* sedemikian rupa sehingga hasil akhirnya adalah matriks bobot (*weight*) untuk masing-masing citra, baik citra *training* (saat registrasi) maupun citra *testing* (saat *login*). Secara singkat dapat dijelaskan sebagai berikut:

1. Setiap citra wajah diubah menjadi sebuah *vector* (atau matriks 1 kolom dengan ukuran 1 x *N*) kemudian seluruh citra wajah dikumpulkan dalam sebuah matriks sehingga tercipta sebuah matriks dengan dimensi *M* x *N* (matriks *S*).
2. Menghitung mean (nilai rata-rata) dari tiap baris dari matriks *A* sehingga tercipta sebuah *vector*.
3. Menghitung selisih dari *mean* (disebut juga *zero mean*) yakni nilai asli dari matriks *M* x *N* dikurangi *vector mean* (matriks *A*).
4. Mencari matriks kovarian dengan menggunakan matriks $P = A \,.\, A^T$.
5. Mencari *eigenvalue* dan *eigenvector* dari matriks *P*.
6. Mencari nilai PCA yang merupakan proyeksi dari *zero mean* terhadap *eigenvector* (PCA = *Zero Mean* . *Eigenvector*).
7. Mencari matriks bobot (Zero Mean . PCA).



Gambar 4. Pencocokan Citra

Gambar 4 menggambarkan blok diagram proses pencocokan citra yakni membandingkan antara citra *training* dan citra *testing*. Secara rinci dapat dijelaskan sebagai berikut:

1. Matriks bobot dari data *training* didapat dari proses yang dijelaskan pada Gambar 3.
2. Matriks bobot dari data *testing* didapat dengan cara yang sama pada Gambar 3 dengan menggunakan *zero mean* dari data *training*.
3. Matriks bobot dari data *training* dan *testing* dimasukkan ke dalam sebuah koordinat kartesius (matriks bobot *training* pada satu sumbu dan matriks bobot *testing* pada sumbu lainnya) sehingga membentuk *eigenface*.
4. Dilakukan penghitungan jarak menggunakan metode *Euclidean Distance*.
5. Hasil akhirnya adalah citra *training* yang memiliki jarak terpendek dengan citra *testing* hasil dari perhitungan *Euclidean Distance* dari matriks bobot tadi. Sebuah citra dianggap cocok apabila citra *testing* dan citra *training* berada dalam satu kelas yang sama. Dengan kata lain, sama-sama merupakan citra dari satu wajah yang sama.

Citra Digital

Suatu citra dapat didefinisikan sebagai sebuah fungsi dua dimensi $f(x, y)$. Nilai x dan y adalah koordinat spasial (bidang), dan amplitudo f

pada beberapa pasangan koordinat (*x, y*) disebut intensitas atau level keabuan dari suatu citra pada titik tersebut. Jika *x*, *y*, dan nilai-nilai amplitudo dari *f* seluruhnya merupakan kuantitas yang terbatas dan diskret, itu disebut sebagai sebuah citra digital. Ruang lingkup pengolahan citra digital mengacu pada pemrosesan citra-citra digital dengan bantuan komputer digital. Catat bahwa sebuah citra digital tersusun atas sejumlah elemen yang terbatas, masing-masing memiliki lokasi tertentu dan nilai tertentu. Elemen-elemen ini disebut sebagai elemen gambar, elemen citra, pel, dan piksel. Piksel adalah istilah yang paling sering digunakan untuk menyebut elemen-elemen dari suatu citra digital [1].

Grayscale

Dalam dunia fotografi, komputer, dan pewarnaan, citra *grayscale* atau *greyscale* adalah salah satu citra yang nilai setiap pikselnya adalah satu sampel tunggal yang hanya mewakili sejumlah cahaya, yang, hanya memuat informasi tentang intensitas cahaya tersebut. Citra ini juga dikenal sebagai citra hitam putih atau monokrom, dan terdiri atas bayang-bayang abu-abu, yang bervariasi dari hitam (intensitas terlemah) sampai putih (intensitas tertinggi). Citra *grayscale* berbeda dari citra *duotone* hitam-putih yang pada konteks citra komputer adalah citra yang hanya memiliki dua warna: hitam dan putih (juga disebut citra bilevel atau biner). *Grayscale* memiliki banyak bayangan abu-abu di antara keduanya.

Deteksi Wajah

Pendeteksian wajah (*face detection*) adalah salah satu tahap awal yang sangat penting sebelum dilakukan proses pengenalan wajah (*face recognition*). Bidang-bidang penelitian yang berkaitan dengan pemrosesan wajah (*face processing*) adalah [2]:

1. Pengenalan wajah (*face recognition*) yaitu membandingkan citra wajah masukan dengan suatu *database* wajah dan menemukan wajah yang paling cocok dengan citra masukan tersebut.
2. Otentikasi wajah (*face authentication*) yaitu menguji keaslian/kesamaan suatu wajah dengan data wajah yang telah diinputkan sebelumnya.
3. Lokalisasi wajah (*face localization*) yaitu pendeteksian wajah namun dengan asumsi hanya ada satu wajah di dalam citra.
4. Penjejakan wajah (*face tracking*) yaitu memperkirakan lokasi suatu wajah di dalam video secara real time.
5. Pengenalan ekspresi wajah (*facial expression recognition*) untuk mengenali kondisi emosi manusia.

Pengenalan Wajah

Pengenalan wajah adalah salah satu tugas visual yang dapat dilakukan dengan mudah oleh manusia. Namun, dalam istilah visi komputer, tugas ini tidaklah mudah [3]. Teknik-teknik pengenalan wajah yang dilakukan selama ini banyak yang menggunakan asumsi bahwa data wajah yang tersedia memiliki ukuran yang sama dan latar belakang yang seragam. Di dunia nyata, asumsi ini tidak selalu berlaku karena wajah dapat muncul dengan berbagai ukuran dan posisi di dalam citra dan dengan latar belakang yang bervariasi.

Principal Component Analysis

Metode *Principal Component Analysis* adalah sebuah metode dibuat oleh ahli statistik. Metode ini kali pertama diperkenalkan oleh Karl Pearson pada tahun 1901. Algoritma ini kemudian diperkenalkan oleh H. Hotelling pada tahun 1933 sehingga juga disebut transformasi Hotelling [4]. PCA merupakan suatu perhitungan standard modern yang digunakan untuk analisis data pada beragam *field* atau multidimensi sekumpulan data (dataset) khususnya pada bidang komputer grafik. Metode ini dinilai mudah karena tidak membutuhkan parameter khusus dalam ekstraksi informasi yang berhubungan terhadap sekumpulan data yang meragukan.

Dalam hal pengolahan citra digital, metode PCA digunakan untuk mereduksi citra yang berdimensi tinggi menjadi citra berdimensi rendah. Pengurangan ini dimaksudkan untuk mempercepat waktu komputasi dan melakukan proses pengenalan wajah. Ada tiga tahapan dalam *Principal Component Analysis* dalam hal pengenalan wajah yakni: tahap *training*, tahap reduksi dimensi, dan tahap klasifikasi. Adapun langkah-langkah PCA adalah sebagai berkut:

Tahap *Training*

1. Membuat *database* citra wajah
   Misal suatu citra Γ didefinisikan sebagai sebuah matriks berukuran *Nx* x *Ny* dikonversikan ke *vector* citra Γ dengan ukuran N x 1 dengan N = *Nx* x *Ny*. Ini adalah matriks yang dibentuk dengan menggabungkan kolom pada citra menjadi satu.

$$\Gamma = [\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_{Mt}]) \quad (1)$$

   Keterangan:
   Γ : citra *training*
   Mt : jumlah citra *training*

2. Menghitung *mean* dari citra wajah

$$\Psi = \frac{1}{Mt}\sum_{i=1}^{Mt} \Gamma_i \quad (2)$$

   Keterangan:
   Ψ : *mean* citra *training*
   Mt : jumlah citra *training*
   Γ_i : citra *training*
   Matriks *mean* ini berukuran (N x 1) piksel

3. Menghitung perbedaan citra dengan tiap citra pada database

$$\Phi = \Gamma - \Psi \quad (3)$$

Keterangan:
Φ : *zero mean*
Γ : set citra *training*
Ψ : *mean* citra *training*

$$A = [\Phi_1, \Phi_2, \Phi_{mt}] \quad (4)$$

Keterangan:
A : matriks *zero mean* (*mean subtracted image*)

Vector dari *mean subtracted image* ini memiliki matriks berukuran *N* x *Mt* piksel. Sedangkan matriks A, masing-masing memiliki ukuran (*N x Mt*) piksel.

4. Menghitung matriks kovarian

$$P = A \,.\, A^T = \sum_{i=0}^{Mt} \Phi_i \Phi_i^T \quad (5)$$

Keterangan:
P : Matriks kovarian (berukuran *N* x *N*)

Tahap Reduksi

Tahap reduksi dilakukan karena ukuran matriks *N* x *N* terlalu besar. Perhitungan dengan menggunakan matriks *N* x *N* dapat memperberat kinerja CPU dan pada beberapa kasus dapat menghentikan program. Karena itu, perhitungan matriks kovarian dilakukan melalui matriks *Mt* x *Mt*.

Tahap Klasifikasi

1. Proyeksi
   Proses klasifikasi tidak menggunakan semua *eigenface* dari citra *training* (*Mt*), melainkan hanya menggunakan *eigenface* yang signifikan (M') saja. Langkah selanjutnya citra *training* kemudian diproyeksikan pada ruang *eigenface*, dan ditentukan bobot (*weight*) dari setiap *eigenvector*-nya. Bobot ini merupakan *dot product* dari setiap citra dengan *eigenvector*-nya.

$$\omega_k = v_k^T . (\Gamma - \Psi) \quad (6)$$

   Keterangan:
   ω : bobot (*weight*)
   v : *eigenvector*
   k : 1, 2, 3, ... M'
   Γ : *citra training*
   Ψ : *zero mean*

   Proyeksi di atas adalah proyeksi citra *training* untuk setiap *eigenvector* dengan k = 1, 2, 3, ... M'
2. Matriks Bobot
   Pada tahap ini citra dibentuk dari matriks-matriks bobot pada ruang *eigenface*. Setiap citra direpresentasikan berasal dari citra yang berukuran *Nx* x *Ny* pada ruang citra. Setelah dilakukan pemrosesan maka citra tersebut diwakili oleh suatu *vector* berukuran (*N'* x 1) pada ruang *eigenface*.

$$\Omega = [\omega_{1,}\omega_{2,}\omega_{3,}\ldots,\omega_{M'}]^T \quad (7)$$

   Keterangan:
   Ω adalah representasi citra *training* dalam ruang *eigenface* yang berukuran *M'* x 1

Proyek Citra *Testing*

Proses klasifikasi dilakukan dengan cara memproyeksikan citra baru ke dalam ruang *eigenface*. *Vector* citra *testing*, Γ_T (*N* x 1) dikurangi dengan *mean* dari citra *training*. Matriks in berukuran *N* x 1.

$$\Phi_T = \Gamma_T - \Psi \quad (8)$$

Keterangan:
Φ_T : *zero mean* citra *testing*
Γ_T : Citra *testing*
Ψ : *mean* citra *training*

Kemudian dilakukan proyeksi terhadap citra *testing*. Matriks ini adalah matriks dari selisih *vector* citra *testing* dan mean dari citra *training* yang berukuran *N* x 1.

$$\omega_{Tk} = v_k^T . \Phi_T = v_k^T . (\Gamma_T - \Psi) \quad (9)$$

Keterangan:
ω_{Tk} : bobot citra *testing*
Φ_T : *zero mean* citra *testing*
Γ_T : Citra *testing*
Ψ : *mean* citra *training*
k : 1, 2, 3, ... *M'*

Kemudian mencari matriks bobot dari citra *testing*. Matriks ini merupakan representasi dari citra *testing* pada ruang *eigenface* dengan ukuran *M'* x 1.

$$\Omega_T = [\omega_{T1}, \omega_{T2}, \omega_{T3}, \ldots, \omega_{TM'}]^T \quad (10)$$

Keterangan:
Ω_T : bobot set citra *testing*
ω_T : bobot citra *testing*

Euclidean Distance

Untuk memperoleh tingkat kesamaan antara dua citra wajah, digunakan pengukuran jarak dengan metode *Euclidean Distance*. *Euclidean Distance* antara dua titik adalah panjang sisi miring dari sebuah segitiga siku-siku [5]. Rumus *Euclidean Distance* adalah:

$$\delta_i = \|x - y\|^2 = \sum_{i=1}^{m} (x_i - y_i)^2 \quad (11)$$

Keterangan:
x : citra *training* ($x_1, x_2, x_3, ..., x_m$)
y : citra *testing* ($y_1, y_2, y_3, ..., y_m$)

Android

Android adalah sistem operasi *mobile* yang dikembangkan oleh Google, didasarkan pada versi kernel Linux yang telah dimodifikasi dan perangkat lunak *open source* lainnya dan didesain terutama untuk piranti *mobile* layar sentuh seperti *smartphone* dan *tablet*. Sejak rilis pertama tanggal 23 September 2008 hingga sekarang sistem operasi Android telah banyak mengalami update menyesuaikan dengan perkembangan teknologi piranti keras dan piranti lunak.

Android datang dengan sekelompok aplikasi inti seperti *email*, pengiriman SMS, kalender, *Internet browsing*, kontak, dan lain-lain. Aplikasi

yang dimasukkan di dalam *platform* tidak memiliki status khusus di antara aplikasi yang di-*install* oleh pengguna. Jadi aplikasi pihak ketiga dapat menjadi *default web browser* bagi pengguna, demikian juga *default* pengiriman SMS, atau bahkan *keyboard default*.

Aplikasi sistem berfungsi baik sebagai aplikasi untuk pengguna dan untuk menyediakan kemampuan utama yang dapat diakses oleh pengembang dari aplikasi mereka sendiri. Sebagai contoh, jika ingin dapat mengirim pesan SMS, tidak perlu membuatnya sendiri. Cukup dengan menjalankan aplikasi SMS yang sudah tersedia.

Python

Bahasa pemrograman Python adalah bahasa yang dibuat oleh Guido van Rossum pada awal tahun 1990 di Belanda. Bahasa Python merupakan bahasa pengganti dari bahasa pemrograman ABC. Bahasa Python saat ini dapat dikembangkan oleh siapa saja karena Python bersifat sumber terbuka (*open source*). Sintaks Python amat sederhana sehingga sangat mudah dipelajari. Selain itu Python tidak memerlukan penanda blok seperti kurung siku "{" ataupun penanda baris ";" sehingga tampilannya menjadi lebih bersih dan mudah untuk ditelusuri. Sebagai ganti dari blok, Python memberlakukan kesamaan spasi atau tab dalam setiap "blok" program. Python memiliki efisiensi tinggi, berorientasi obyek lebih sederhana dan efektif, serta dapat bekerja secara multiplatform. Tidak mengherankan jika bahasa pemrograman Python seringkali dipilih bagi mereka yang berkecimpung di bidang *data mining* dan *artificial intelligent*. Python adalah bahasa interpreter. Artinya, tidak harus meng-*compile* kode sumber untuk menjalankannya. Dengan demikian, waktu eksekusi pun bisa dihemat untuk pengembangan aplikasi. Tidak hanya itu, Python juga dapat berjalan di dalam *shell* secara interaktif sehingga bisa diketahui hasil dari eksekusi satu baris program pada saat itu juga.

RESTful API

Representational State Transfer (REST) adalah sebuah gaya arsitektur yang menjelaskan sekelompok properti dan batasan yang berhubungan dengan HTTP. Web Service yang memenuhi gaya arsitektur REST disebut *RESTful web service*. RESTful *Web Service* menyediakan operasional yang saling terkait antara sistem komputer dan Internet. *Web service* REST membolehkan sistem yang melaksanakan *request* untuk mengakses dan memanipulasi representasi tekstual dari *web resource* dengan menggunakan sekelompok operasi *stateless* yang seragam dan yang sudah didefinisikan sebelumnya. Dalam *RESTful web service*, *request* (permintaan) yang dibuat yang menjadi sumber URI akan menghasilkan sebuah *response* (tanggapan) yang berbentuk XML, HTML, JSON atau format lainnya. *Response* ini dapat menjelaskan bahwa beberapa perubahan telah dibuat di sumbernya, dan *response* dapat pula menyediakan tautan (*link*) untuk sumber-sumber yang terkait. Jika HTTP digunakan (dan ini yang paling sering), operasi yang tersedia antara lain: GET, POST, PUT, DELETE, dan beberapa metode CRUD HTTP lainnya.

Dengan menggunakan protokol *stateless* dan operasi *standard*, sistem REST bertujuan untuk menciptakan performa yang cepat, dapat diandalkan, dan kemampuan untuk dapat dikembangan, dengan menggunakan kembali komponen-komponen yang dapat diatur dan disesuaikan tanpa mempengaruhi sistem secara keseluruhan, bahkan ketika sistem itu sedang berjalan. Istilah *representational state transfer* dijelaskan pada tahun 2000 oleh Roh Fielding dalam disertasi doktoralnya. Disertasi Fielding menjelaskan prinsip-prinsip REST yang dikenal sebagai "*HTTP object model*" dimulai pada tahun 1994 dan digunakan dalam mendesain HTTP 1.1 dan *standard Uniform Resource Identifiers* (URI). Istilah ini dimaksudkan untuk menciptakan sebuah gambaran bagaimana sebuah aplikasi web yang dirancang dengan baik itu bekerja: adalah sebuah jaringan sumber-sumber *Web* (mesin virtual) yang pengguna menjalankannya dengan memilih link (seperti pengguna "tom") dan operasi seperti GET atau DELETE (status transisi), menghasilkan sumber berikutnya (merepresentasikan status berikutnya dari suatu aplikasi) dengan dikirimkan ke pengguna melalui penggunaan REST.

HASIL DAN PEMBAHASAN

Perangkat Penelitian

Berikut ini adalah perangkat-perangkat yang digunakan di dalam proses uji coba sistem. Pembuatan sistem ini menggunakan perangkat keras dan perangkat lunak.

Perangkat keras yang digunakan dalam pembuatan sistem adalah sebagai berikut:

1. Laptop
   - Tipe : MacBook Pro 13 Mi-2011
   - Prosesor : Intel Core i5 2.4 Ghz
   - Memori : 12 GB 1600MHz DDR3
   - Hard Disk : SATA 120GB dan ATA 320 GB
   - Grafik : Intel HD Graphics 3000 512 MB
2. *Smartphone*
   - Tipe : Sony Xperia Z3
   - Layar : 1080 x 1921 pixel
   - OS : Android Marshmallow
   - Chipset : CPU Snapdragon 801 + GPU Adreno 330
   - RAM : 3 GB
   - Kamera : depan 2.2 MP dan belakang 20.7 MP

Laptop digunakan untuk pembuatan sistem dan juga sebagai *server*. Sedangkan handphone digunakan sebagai aplikasi *client* yang fungsinya mengakses *service* (dari *server*) yang berupa RESTful API.

Adapun perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Sistem operasi MacOS High Sierra
2. Visual Studio Code versi 1.24.1
3. Android Studio versi 3.1.3
4. OpenCV versi 3.4.1
5. Numpy versi 1.14.5
6. Django versi 2.0.7

Implementasi Sistem

Karena sistem yang dibangun adalah sistem yang menangani *login* dengan menggunakan wajah, maka sistem ini tidak mengatur hal-hal di luar ruang lingkup pengenalan wajah seperti hak akses, data pengguna secara lengkap, dan sebagainya. Sistem ini khusus untuk implementasi sistem pengenalan wajah secara *real time* untuk digunakan *login* sebagai pengganti *password*.

Aplikasi *Server*

Aplikasi *server* dibuat untuk menangani *request* (permintaan) dari *client* dalam hal: *upload* citra, *register*, dan *login*. Aplikasi *server* dibuat dengan menggunakan bahasa Python 3.7. Pemilihan Python sebagai bahasa pemrograman di sisi *server* adalah karena Python memiliki pustaka yang sangat mendukung untuk pembangunan sistem ini. Di samping itu sintaks penulisan bahasa Python juga sangat sederhana sehingga lebih mudah dibaca oleh siapa saja. Pustaka Python yang digunakan adalah cv2 yang merupakan *wrapper* dari OpenCV untuk operasi yang berhubungan dengan *image processing* seperti pembacaan citra, deteksi wajah, dan *preprocessing*. Sedangkan untuk operasi yang berhubungan dengan matriks, *vector*, serta pencarian *eigenvalue* dan *eigenvector*, penulis menggunakan *library* numpy yang sangat terkenal di kalangan *data scientist*. Lagipula, cv2 memang menggunakan numpy dalam membaca citra.

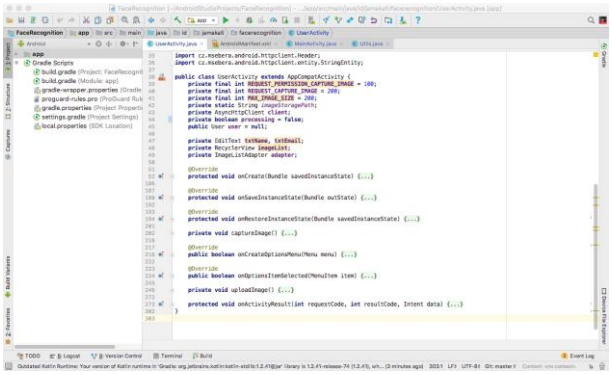
Tabel 1. Infrastruktur RESTful API Pengenalan Wajah

No.	Metode	URL	Parameter
1	POST	/register/	id = integer name = string email = string images = array of string
2	POST	/login/	image = string
3	POST	/upload-image/	filename = file
4	GET	/media/<filepath>	-

Python juga memiliki pustaka sqlite untuk operasi yang berhubungan dengan basis data yang memanfaatkan SQLite. Untuk pembuatan RESTful API, penulis menggunakan Django *framework* yang merupakan *web framework* terkenal di kalangan pengembang Python. Django memiliki fitur ORM (*Object Relational Mapper*) sehingga dalam sistem ini registrasi pengguna yang berhubungan dengan *database* menjadi lebih mudah dibuat. Karena ini adalah aplikasi *server* maka tidak ada GUI (*Graphic User Interface*) sama sekali. Yang ada adalah infrastruktur *web service* yang berupa RESTful API yang tercantum pada Tabel 1.

Aplikasi *Client*

Aplikasi *client* adalah aplikasi yang dipakai untuk melakukan *request* kepada *server* dan melakukan aksi setelah mendapatkan *response* (tanggapan) dari *server*. Dalam sistem ini, aplikasi *client* dibuat dengan menggunakan Android Studio 3.1.3. Untuk mengakses *server* penulis menggunakan pustaka AsyncHttp dari loopj. Aplikasi client dibuat secara *native* berbasis Java dan layout menggunakan XML. Proses pembuatan aplikasi *client* dengan menggunakan Android Studio dapat dilihat pada Gambar 5.



Gambar 5. Tampilan Android Studio

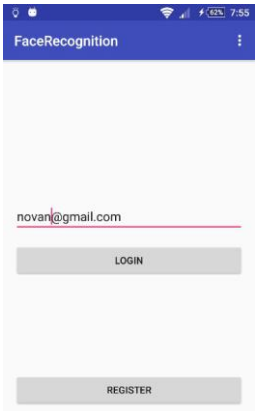
Secara garis besar ada dua tahap yang dilakukan oleh aplikasi *client*: *Register* dan *Login*. Android memiliki istilah Activity untuk menyebut sebuah "halaman" tampilan. Perpindahan antara satu tampilan ke tampilan lainnya artinya adalah perpindahan antara satu Activity ke Activity lainnya. Untuk itu, sebagai ganti menyebut halaman (seperti halnya pada pemrograman web), di sini akan dipakai istilah Activity.

Form *Login*

Main Activity menempati posisi pertama saat aplikasi *client* dijalankan. Pada halaman ini, pengguna diminta untuk menuliskan email yang dimasukkan pada saat pendaftaran. Dalam penerapan yang sesungguhnya, pengguna hanya perlu menuliskan email sekali saja setelah aplikasi diinstall dan pengguna telah *login*.

Tombol *Login* adalah salah satu elemen yang terdapat pada MainActivity. Ketika tombol ini diklik, maka akan muncul sebuah Activity bawaan dari Android untuk melakukan penangkapan gambar melalui kamera. Secara berurutan proses *login* adalah sebagai berikut:

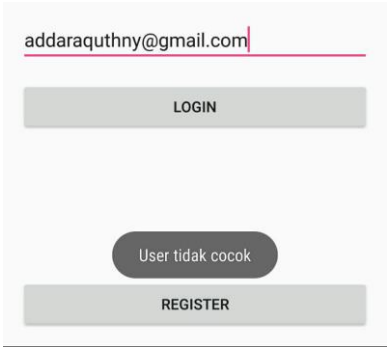
1. Ambil gambar dari kamera (melalui Activity bawaan Android)
2. Mengirim gambar hasil tangkapan ke *server*
3. *Server* memberikan hasil apakah gambar tersebut dikenali atau tidak



Gambar 6. Tampilan MainActivity (*Login*)

Gambar 7. Pengambilan Gambar dari Kamera

Jika tidak dikenali, maka aplikasi *client* akan menampilkan pesan bahwa gambar yang dikirim tidak cocok.

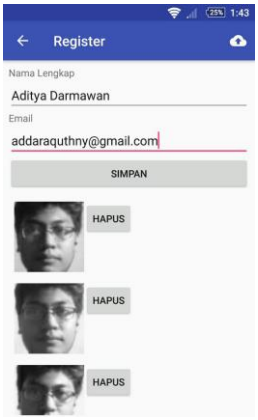


Gambar 8. Tampilan Saat Wajah Tidak Cocok

Adapun jika gambar berhasil dikenali maka pengguna akan diarahkan ke halaman lain yang dapat diakses oleh pengguna yang *authenticated* (memiliki hak masuk). Dalam sistem ini pengguna akan diarahkan ke UserActivity yang isinya sama persis saat pengguna melakukan registrasi. Atau dengan kata lain pengguna diarahkan ke sebuah halaman "*profile*". Penggunaan layout yang sama persis antara halaman *register* dan *profile* untuk memenuhi prinsip DRY (*Don't repeat yourself*).

Form *Register*

Form *register* digunakan untuk memasukkan data pengguna sekaligus upload citra wajah. Karena sistem ini berfokus pada pengenalan wajah maka data yang dimasukkan hanya nama dan *email* saja (sebagai data unik). Pada penerapan di lapangan tentu saja sebuah pendaftaran memerlukan lebih banyak data. Namun pada sistem ini dicukupkan pada *email* saja.



Gambar 9. Tampilan UserActivity (*Register*)

Tombol *upload* pada Gambar 9 diletakkan pada ActionBar untuk memudahkan dalam melakukan penangkapan citra. Hasil tangkapan diletakkan di bawah tombol simpan dan ditampilkan dengan menggunakan fitur RecyclerView.

Uji Coba Sistem

Uji coba sistem dibagi menjadi dua tahap. Tahap pertama adalah pengujian terhadap kode sumber dan tahap kedua adalah pengujian terhadap data input yang didapat dari hasil tangkapan kamera *smartphone* secara *real-time*.

Uji Coba Kode Sumber

Uji coba kode sumber dilakukan untuk mengetahui apakah kode sumber sudah sesuai mengimplementasikan pengenalan wajah berbasis *Principal Component Analysis*. Data input yang digunakan adalah data set publik wajah AT&T yang berjumlah 40 wajah dengan 10 pose tiap wajah dengan kualitas citra yang sama. Penulis melakukan beberapa skenario percobaan sehingga didapat hasil seperti yang tercantum pada Tabel 2.

Dari uji coba pada Tabel 2 bisa dilihat bahwa pada skenario 1 data *training* hanya berjumlah 4 pose untuk tiap-tiap wajah. Sementara itu data *testing* berjumlah 6 pose. Dengan skenario data *testing* lebih banyak daripada data *training* didapat hasil akurasi 87,5% yakni dari 240 percobaan (6 pose x 40 wajah) terjadi 210 wajah yang sukses dikenali dengan baik. Sementara itu 30 wajah gagal terdeteksi. Dengan demikian akurasi yang didapat adalah 87,5%. Kemudian pada skenario 5 dilakukan proses *training* lebih banyak menjadi 9 pose

sementara data *testing* hanya 1 pose. Dari 40 percobaan (1 pose x 40 wajah) terdapat 38 wajah dikenali dengan baik sedangkan ada 2 wajah yang gagal dikenali dengan baik. Dengan demikian akurasinya menjadi 95%.

Tabel 2. Skenario Percobaan Menggunakan Citra AT&T

Skenario	*Training*	*Testing*	Sukses	Gagal	Akurasi
1	2 5 7 8	1 3 4 6 9 10	210	30	87,5%
2	1 4 7 8 9	2 3 5 6 10	191	9	95,5%
3	1 2 3 6 9 10	4 5 7 8	154	6	96,25%
4	2 4 6 7 8 9 10	1 3 5	114	6	95%
5	2 3 4 5 6 7 8 9 10	1	38	2	95%

Disimpulkan bahwa semakin banyak data *training* yang dimasukkan maka akan semakin besar presentasi keberhasilan pengenalan wajah. Kegagalan terjadi ketika sebuah wajah hasil proses pengenalan wajah bukan merupakan satu kelas yang sama dengan data wajah *testing*. Dapat disimpulkan pula bahwa dengan menggunakan data publik yang stabil, kode sumber berhasil mengimplementasikan pengenalan wajah berbasis *Principal Component Analysis*.

Uji Coba Menggunakan Kamera *Smartphone*

Uji coba ini memanfaatkan aplikasi *client* untuk menangkap citra *input* dan mengirimkannya ke *server*. Pada awalnya, aplikasi *server* diupload ke sebuah *virtual private server* dengan spesifikasi 1 GB RAM, 1 Core CPU, dan 25 GB SSD. *Server* selalu menghasilkan response *Gateway Timeout* (504) setiap kali melakukan proses pengenalan wajah. Ini karena proses pengolahan wajah melibatkan komputasi yang tidak sederhana sehingga diperlukan spesifikasi *server* yang lebih tinggi dari spesifikasi tersebut. Penulis kemudian menggunakan laptop yang digunakan untuk pengembangan sistem sebagai *server* dan hasilnya sering terjadi t*imeout* (waktu habis). Hal ini wajar karena komputer yang digunakan bukan komputer berspesifikasi *server* sehingga *request* yang masuk seringkali terganggu karena koneksi yang tidak stabil. Meski demikian uji coba sistem tetap dapat dilakukan hingga mendapat *response* dari *server*.

Pada uji coba ini dikumpulkan 15 wajah dengan masing-masing wajah memiliki 4 gambar dengan pose bervariasi sebagai data *training* dan 1 gambar lainnya sebagai data *testing*. Gambar diambil dengan mengunakan *smartphone* dengan bantuan tripod untuk menjaga stabilitas pengambilan serta memiliki latar belakang yang seragam.

Dari citra input pada Tabel 2 dilakukan 5 skenario percobaan sebagaimana yang tercantum pada Tabel 3.

Skenario 1 adalah skenario *real-time* yakni data *training* diambil sebanyak empat kali dan hasil yang didapat adalah hasil dari perhitungan secara *real-time* pada saat proses *login*. Sedangkan skenario selanjutnya adalah menggunakan kembali data input yang sudah didapat dari skenario 1 untuk kemudian diteliti kembali untuk mengetahui seberapa banyak kemungkinan pengenalan wajah yang terjadi. Pada tabel 4.4 dapat dilihat bahwa skenario 1 memiliki akurasi lebih besar sebanyak 66,7% didapat dari 15 kali percobaan dengan 4 pose untuk data *training* dan 1 pose untuk data *testing*. Pada pengujian *real-time* juga dilakukan pengujian *login* menggunakan akun email yang sudah teregistrasi namun dengan wajah yang belum teregistrasi. Didapat hasil bahwa wajah tersebut tidak dikenali.

Tabel 3. Skenario Percobaan Menggunakan Kamera *Smartphone*

Skenario	*Training*	*Testing*	Sukses	Gagal	Akurasi
1	2 3 4 5	1	10	5	66,67%
2	2 3 4	1 5	17	13	56,67%
3	2 5	1 3 4	19	26	42,22%
4	3	1 2 4 5	18	42	30%
5	5	1 2 3 4	22	38	36,67%

Pada skenario 5 data *training* hanya 1 pose dan data *testing* ada 4 pose didapat hasil akurasi 36,67% jauh lebih kecil dibandingkan dengan skenario 1. Jika diperhatikan skenario 4 ternyata jumlah kegagalannya lebih banyak daripada skenario 5 dengan jumlah data *training* dan *testing* yang sama namun dengan data yang berbeda. Ini menunjukkan bahwa perbedaan pose juga berpengaruh terhadap hasil pengenalan wajah. Namun demikian, sebagaimana yang telah dibahas pada bagian sebelumnya, bahwa data secara *real-time* tidak seseragam data publik. Data *real-time* cenderung beragam karena kecerahan dan latar belakang gambar tidak bisa dikontrol. Kualitas kamera pada *smartphone* juga berpengaruh terhadap hasil akhirnya nanti. Dapat disimpulkan bahwa pengenalan wajah secara *real-time* menggunakan metode *Principal Component Analysis* sukses dilakukan meskipun terdapat beberapa kali kegagalan. Banyaknya citra *training* berpengaruh terhadap hasil. Semakin banyak jumlah citra *training* dengan bermacam pose yang dimiliki oleh sebuah kelas wajah maka semakin baik sistem mengenali wajah. Jika dilihat data set wajah dari AT&T semua wajah memiliki kualitas yang sama dan konsisten serta jumlah datanya pun dua kali lebih banyak sehingga penggunaan metode *Principal Component Analysis* menjadi lebih akurat.

Pengenalan wajah secara *real-time* berbasis *Principal Component Analysis* dalam aplikasi *mobile* bisa diterapkan dengan baik untuk keperluan *login* pengguna sehingga pengguna tidak perlu memasukkan *username* dan *password* setiap kali melakukan *login*. Hal yang perlu diperhatikan ketika menggunakan pengenalan wajah sebagai otentikasi

pengguna adalah bahwa tidak ada batasan yang baku untuk menentukan apakah wajah terdeteksi dengan benar atau tidak karena yang dihasilkan adalah yang memiliki jarak terdekat (paling mirip). Tidak ada rumus baku pemberian ambang batas (*threshold*) untuk menentukan apakah jarak yang dihasilkan dari perhitungan *Euclidean distance* dapat dipakai atau tidak. Oleh karena itu penggunaan pengenalan wajah berbasis *Principal Component Analysis* sebagai pengganti *login* hanya optimal dilakukan untuk otentikasi pengguna 1:1 (*verification*) bukan 1:N (*identification*). Meski demikian bukan berarti penggunaan sistem pengenalan wajah ini terbatas pada sistem keamanan saja.

PENUTUP

Pada uji coba kode sumber menggunakan data AT & T sebagai citra input didapat hasil akurasi 87,5% dengan 4 pose sebagai citra *training* dan 6 pose sebagai citra *testing* dan 95% dengan 9 pose sebagai citra *training* dan 1 pose sebagai citra *testing*. Sedangkan pada uji coba menggunakan kamera *smartphone* secara *real-time* didapat hasil akurasi 66,67% dengan 4 pose sebagai citra *training* dan 1 pose sebagai citra *testing*.

Berdasarkan hasil pembahasan dan uji coba sistem dapat disimpulkan bahwa sistem dapat mengenali wajah menggunakan metode *Principal Component Analysis* dengan baik sehingga metode ini layak untuk dikembangkan lebih lanjut. Penggunaan *smartphone* untuk menangkap gambar sangat berpengaruh terhadap hasil yang akan diperoleh dengan menggunakan metode *Principal Component Analysis*. Pengenalan wajah berbasis *Principal Component Analysis* dapat digunakan untuk otentikasi pengguna secara 1:1 (*verification*) bukan 1-N (*identification*). Namun untuk penggunaan di luar otentikasi tergantung pada kebutuhan program tersebut.

Penggunaan metode *Principal Component Analysis* sangat dipengaruhi oleh citra hasil tangkapan. Oleh karena itu dibutuhkan hasil citra yang seragam untuk memperkecil variasi kualitas citra. Perlu diadakan prapemrosesan citra (*image preprocessing*) karena penangkapan citra menggunakan media kamera *smartphone* sangat rentan terhadap perbedaan kualitas, terlebih dalam intensitas cahaya yang kurang dan pengambilan gambar yang kurang *steady* (bergoyang-goyang). Perlu adanya spesifikasi *server* yang sedikit lebih tinggi (lebih dari 1 GB RAM) dan *dedicated* agar pengolahan citra digital dapat dilakukan dengan baik di sisi *server*. Dengan memanfatkan *server* sebagai pengolah citra digital, *resource* dari komputer *client* dapat dihemat sehingga aplikasi *client* dapat berjalan lebih cepat.

DAFTAR PUSTAKA

[1] R. Gonzales and R. Woods, "Digital image processing.[Sl]," *Addison-Wesley Publ. Co.*, 1992.

[2] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[3] E. Hjelmås and B. K. Low, "Face detection: A survey," *Comput. Vis. image Underst.*, vol. 83, no. 3, pp. 236–274, 2001.

[4] M. H. Purnomo and A. Muntasa, "Konsep pengolahan citra digital dan ekstraksi fitur," *Yogyakarta Graha Ilmu*, 2010.

[5] D. A. N. Rahmah, "Teknik Pengenalan Wajah Dengan Algoritma PCA Berbasis Seleksi Eigenvector," in *Proceedings Seminar Tugas Akhir Jurusan Teknik Elektro*, 2011.

[6] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Soc. Artif. Intell.*, vol. 14, no. 771–780, p. 1612, 1999.

[7] M. Oliveira and V. Santos, "Automatic detection of cars in real roads using haar-like features," *Dep. Mech. Eng. Univ. Aveiro*, vol. 3810, 2008.

[8] R. Munir, "Pengolahan citra digital dengan pendekatan algoritmik," *Inform. Bandung*, 2004.

[9] M. Hatta, I. G. Susrama, I. K. E. Purnama, and M. Hariadi, "Cacah Spermatozoa Menggunakan Background Segmentation dan Boundary Detection," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 11, no. 1, pp. 67–74, 2016.

[10] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings. international conference on image processing*, 2002, vol. 1, pp. I–I.

[11] D. Suprianto and R. N. Hasanah, "Sistem Pengenalan Wajah Secara Real-Time dengan Adaboost, Eigenface PCA & MySQL," *J. EECCIS*, vol. 7, no. 2, pp. 179–184, 2014.

[12] B. C. Putra and Y. N. Afifah, "GAUSSIAN MIXTURE MODEL UNTUK PENGHITUNGAN TINGKAT KEBERSIHAN SUNGAI BERBASIS PENGOLAHAN CITRA," *Tek. Eng. Sains J.*, vol. 2, no. 1, pp. 53–58, 2018.

[13] K.-K. Sung, "Learning and example selection for object and pattern detection," 1996.

[14] A. Santoso, I. Arif, and M. Hatta, "Pembelajaran Supervised SVM Untuk Identifikasi Obyek Pisau Pada Mesin X-Ray Bandara Juanda," *Nusant. J. Comput. its Appl.*, vol. 1, no. 1, 2017.

[15] D. A. Prasetya and I. Nurviyanto, "Deteksi Wajah Metode Viola Jones pada Opencv Menggunakan Pemrograman Python." [Online]. Available: https://publikasiilmiah.ums.ac.id/xmlui/handle/11617/3936. [Accessed: 18-Oct-2018].