UNIVERSITAS INDONESIA

IMPLEMENTASI SISTEM PENGENALAN WAJAH SEBAGAI PENGHUBUNG JEJARING SOSIAL :

APLIKASI *MOBILE* UNTUK DETEKSI WAJAH DENGAN ANDROID *FACE DETECTOR* API DAN KOMUNIKASI REST KE KOMPUTASI AWAN

SKRIPSI

Prasetyawidi Indrawan
0806459860

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JULI 2012

UNIVERSITAS INDONESIA

IMPLEMENTASI SISTEM PENGENALAN WAJAH SEBAGAI PENGHUBUNG JEJARING SOSIAL **:**

APLIKASI *MOBILE* UNTUK DETEKSI WAJAH DENGAN ANDROID *FACE DETECTOR* API DAN KOMUNIKASI REST KE KOMPUTASI AWAN

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana

Prasetyawidi Indrawan
0806459860

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	:	Prasetyawidi Indrawan
NPM	:	0806459860
Tanda Tangan	:	
Tanggal	:	2 Juli 2012

HALAMAN PENGESAHAN

Skripsi yang diajukan oleh :

Nama	:	Prasetyawidi Indrawan
NPM	:	0806459860
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Implementasi Sistem Pengenalan Wajah Sebagai Penghubung Jejaring Sosial : Aplikasi *Mobile* Untuk Deteksi Wajah dengan Android *Face Detector* API dan Komunikasi REST ke Komputasi Awan

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Prof.Dr.Ir.Riri Fitri Sari M.Sc, MM. **(** **)**

Penguji : Dr. Ir. Dodi Sudiana M.Eng. **(** **)**

Penguji : Yan Maraden ST. MSc. **(** **)**

Ditetapkan di : Depok

Tanggal : 2 Juli 2012

KATA PENGANTAR

Penulis mengucapkan terima kasih kepada :

1. Prof. Dr. Ir. Riri Fitri Sari, M.Sc.M.M. selaku pembimbing akademis dan dosen pembimbing atas segala bimbingan, ilmu, dan arahan baik dalam penulisan skripsi maupun selama masa studi di Teknik Komputer.
2. Orang tua dan keluarga di Bogor yang telah memberikan bantuan dukungan material dan moral.
3. Teman-teman di program studi Teknik Komputer dan Teknik Elektro atas segala dukungan dan kerja samanya.

sehingga penulisan skripsi ini dapat diselesaikan dengan baik. Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, 2 Juli 2012

Penulis,

Prasetyawidi Indrawan

NPM. 0806459860

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPERLUAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini :

Nama	: Prasetyawidi Indrawan
NPM	: 0806459860
Program Studi	: Teknik Komputer
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalti-Free Right***) atas karya ilmiah yang berjudul :

Implementasi Sistem Pengenalan Wajah Sebagai Penghubung Jejaring Sosial :

APLIKASI *MOBILE* UNTUK DETEKSI WAJAH DENGAN ANDROID *FACE DETECTOR* API DAN KOMUNIKASI REST KE KOMPUTASI AWAN

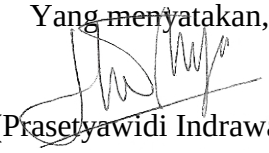
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini, Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 2 Juli 2012

Yang menyatakan,



(Prasetyawidi Indrawan)

ABSTRAK

Nama : Prasetyawidi Indrawan

Program Studi : Teknik Komputer

Judul : Implementasi Sistem Pengenalan Wajah Sebagai Penghubung Jejaring Sosial : Aplikasi *Mobile* untuk Deteksi Wajah dengan Android *Face Detector* API dan Komunikasi REST ke Komputasi Awan

Pembimbing : Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc.

Identitas diri seseorang dalam jejaring sosial menjadi hal penting terutama ketika ingin mengenali siapa sebenarnya orang tersebut. Pencarian identitas diri dapat dengan mudah dilakukan melalui pencarian dalam situs *search engine* ataupun situs jejaring sosial yang ada pada komputer atau laptop. Metode ini sepertinya bukan merupakan hal yang efektif dan praktis seiring berkembangnya perangkat *mobile* dalam masyarakat seperti *smartphone* dan *tablet*. Untuk itu, dirancang sebuah sistem pengenalan wajah pada perangkat *mobile*. Sistem ini dirancang dalam bentuk aplikasi yang dikembangkan pada perangkat *mobile* Android. Penggunaan Android *Face Detector API* akan bertindak sebagai pustaka dalam proses deteksi wajah pada perangkat *mobile* sebelum melakukan proses *offloading* ke layanan komputasi awan. Hasil implementasi berupa modul deteksi wajah pada perangkat *mobile* dan modul pengenalan wajah (*offloading*) yang memanfaatkan layanan komputasi awan dengan bantuan komunikasi *Representational State Transfer* (REST). Hasil pengujian sistem pada perangkat *mobile* menunjukkan bahwa total waktu pengenalan wajah sebesar 7,45 detik dengan waktu deteksi wajah (*onloading*) 0,45 detik dan waktu proses *offloading* 7 detik.

Kata kunci: pengenalan wajah, komputasi awan, deteksi wajah

ABSTRACT

Name : Prasetyawidi Indrawan

Study Program : Computer Engineering

Title : Implementation of Face Recognition System as a Connector to Social Network : Mobile Application for Face Detection with Android Face Detector API and REST Communication to Cloud Computing.

Supervisor : Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc.

The identity of a person in social networking becomes very important especially when we want to identify a person. Search for detailed-identity can be easily conducted through searching using the search engine sites or existing social networking website using computer or laptop. This method is not effective and practical when we consider the development of mobile device technology in the community such as smartphone and tablet. Therefore, designed a face recognition system on mobile devices. The system is designed in the form of an application developed on Android mobile devices. The use of Android Face Detector API will act as libraries in the process of face detection before performing the offloading stage. This paper describes the implementation of the facial detection module on mobile device and face recognition module (offloading) using cloud computing service with REST communication. The result of testing on mobile device indicates that total computation time for face recognition system reached 7,45 seconds with the onloading process 0,45 seconds and the offloading process 7 seconds.

Keywords: *face recognition, cloud computing, face detection*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini, jejaring sosial telah menjadi media yang sangat diminati banyak orang. Banyak informasi pribadi seseorang yang bisa didapatkan dari jejaring sosial ini. Mulai dari identitas pribadi hingga identitas pekerjaan orang tersebut. Seseorang dapat dengan mudah mengetahui identitas pribadi seseorang dengan mengakses situs jejaring sosial atau dengan melakukan pencarian melalui situs *search engine*. Akan tetapi, hal ini tampak menjadi hal yang kurang efektif dan pencarian harus dilakukan di depan komputer atau laptop.

Perkembangan teknologi *mobile* seperti *smartphone* dan *tablet* memungkinkan seseorang dapat dengan mudah menjalankan berbagai aktivitas *multitasking* termasuk aktivitas dasar telepon hingga menjalankan berbagai aplikasi seperti *email*, *multimedia*, aplikasi *office*, dan lain-lain. Hal ini menjadi pertimbangan bahwa sistem identifikasi diri seseorang bisa menjadi lebih mudah jika dapat diaplikasikan pada perangkat *mobile*. Dengan memanfaatkan sistem kamera pada perangkat *mobile*, sistem identifikasi diri seseorang bisa dilakukan dengan menggunakan sistem pengenalan wajah (*face recognition*) yang diimplementasikan pada perangkat *mobile* tersebut. Hal lain yang menjadi pertimbangan dalam penerapan fungsi sistem pengenalan wajah sebagai pencari identitas diri seseorang pada perangkat *mobile*, yaitu perangkat *mobile* hanya memiliki sumber daya yang terbatas dan dibutuhkan suatu komputasi yang efisien. Perangkat *mobile* merupakan perangkat yang menggunakan baterai sebagai penggeraknya sehingga memerlukan suatu efisiensi terutama masalah waktu pemrosesan, mengingat komputasi yang tinggi diperlukan untuk mengolah suatu citra termasuk *face recognition*.

Berawal dari [1], memberikan suatu keuntungan jika proses komputasi dari sistem pengenalan wajah ini bisa dilakukan di *server* (*cloud*). Sistem komputasi awan (*cloud computing*) telah menawarkan suatu solusi dengan membuat komputasi secara *offline* pada *cloud server*. Seperti yang telah

ditunjukkan pada [2, 3] bahwa mengirimkan proses komputasi pada perangkat lain (*server*) akan dapat memberikan keuntungan jika jumlah komputasi yang dilakukan cukup besar dengan jumlah komunikasi data yang relatif kecil termasuk pada aplikasi pengolahan citra dan video.

Seperti yang diketahui, sistem komputasi awan juga telah menawarkan suatu solusi kemudahan terhadap sumber daya komputasi sehingga pengguna komputer tidak perlu direpotkan dengan media penyimpanan ataupun *data processing*. Atas dasar itulah, penulisan skripsi ini akan membahas tentang penerapan teknologi komputasi awan pada perangkat *mobile* dalam hal pencarian identitas seseorang melalui proses pengolahan citra (*face recogniton*). Sistem yang dibuat terdiri dari modul deteksi wajah sebagai bagian pemrosesan awal sistem pengenalan wajah dan penggunaan komunikasi REST ke *cloud server* sebagai bagian lanjutan proses pengenalan wajah. Implementasi sistem yang diterapkan pada perangkat *mobile*, diharapkan dapat menjadi proses pencarian identitas diri yang efisien.

1.2 Rumusan Masalah

Pada tulisan ini, permasalahan yang menjadi fokus perhatian adalah perancangan serta implementasi bagian dari sistem pengenalan wajah pada perangkat *mobile* yang dihubungkan dengan komputasi awan. Proses perancangan dan implementasi dilakukan dengan membuat modul aplikasi sistem deteksi wajah pada perangkat *mobile* dengan *platform* Android dan pemanfaatan layanan komputasi awan untuk pengenalan wajah pada *cloud server* yang dibantu dengan komunikasi REST.

1.3 Tujuan Penulisan

Tujuan penulisan dari penelitian ini adalah untuk membuat suatu sistem pengenalan identitas diri seseorang melalui modul sistem deteksi wajah yang diimplementasikan pada perangkat *mobile* dan dihubungkan dengan komputasi awan untuk proses pengenalan wajah lebih lanjut. Sistem ini diharapkan dapat mengenali identitas diri seseorang secara efisien dan praktis. Selain itu, penggunaan komunikasi REST juga diharapkan dapat menambah efisiensi waktu terutama ketika melakukan komunikasi ke komputasi awan.

1.4 Batasan Masalah

Permasalahan dalam penulisan skripsi ini dibatasi pada perancangan dan implementasi bagian dari sistem pengenalan wajah yaitu sistem deteksi wajah pada perangkat *mobile* dengan Android *Face Detector* API serta integrasi dengan komputasi awan sebagai proses komputasi lanjutan dari sistem pengenalan wajah dengan komunikasi REST.

Bagian lain dari sistem pengenalan wajah ini, yaitu akan membahas mengenai implementasi sistem komputasi untuk fungsi pengenalan wajah yang terjadi di *server* dengan *Google App Engine* (GAE) berbasis pada *python* dan *Face.com* API [4]. Proses realita tertambah (*Augmented Reality*) sebagai penyedia antarmuka sistem pengenalan wajah pada perangkat Android akan dibahas pada [5].

1.5 Metode Penelitian

Metode penelitian yang digunakan dalam penulisan skripsi ini adalah

a. Studi literatur

   Studi literatur dilakukan untuk mendapatkan latar belakang masalah untuk merumuskan solusi dari masalah tersebut serta sebagai dasar teori dan teknis dalam implementasi modul.

b. Perancangan perangkat lunak

   Sistem yang dirancang dalam skripsi ini adalah modul aplikasi *mobile* untuk deteksi wajah yang dihubungkan ke komputasi awan melalui komunikasi REST. Perancangan sistem dilakukan dengan menggunakan kaidah rekayasa perangkat lunak, antara lain tahapan *user requirement*, *system design*, *implementation*, dan *testing*. Dalam perancangan sistem yang telah dilakukan, digunakan pula *Unified Modelling Language* (UML) sebagai bahasa standar dalam permodelan rancangan perangkat lunak [6].

c. Implementasi sistem

   Sistem ini akan diimplementasikan pada perangkat *mobile* berbasis Android sesuai dengan hasil rancangan dengan diagram UML. Sistem yang diimplementasikan pada perangkat *mobile* terdiri dari 2 bagian yaitu proses deteksi wajah (*onloading*) dan proses pengiriman hasil deteksi wajah

(*offloading*) ke komputasi awan. Proses *offloading* merupakan proses lanjutan untuk mengenali identitas wajah.

d. Pengujian dan analisa

Setelah sistem diimplementasikan pada perangkat *mobile* Android, dilakukan pengujian untuk mengetahui kinerja dari sistem yang dibangun. Parameter yang dapat dianalisa dalam pengujian ini seperti respon waktu, ukuran resolusi kamera yang dipakai, ukuran dimensi citra, dan kualitas kompresi citra.

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut :

- Bab 1 Pendahuluan

  Bab Pendahuluan ini akan membahas tentang Latar Belakang, Rumusan Penelitian, Tujuan Penulisan, Batasan Penulisan, Metode Penelitian, dan Sistematika Penulisan.

- Bab 2 Sistem Pengenalan Wajah dan Layanan Komputasi Awan.

  Bab 2 akan memberikan pengantar mengenai berbagai teori yang berkaitan dengan implementasi sistem dalam penulisan skripsi ini, yang terdiri dari pembahasan mengenai sistem pengenalan wajah (*face recognition*) dan aplikasinya, sistem kerja komputasi Awan, penggunaan Android *Face Detector* API dan sistem komunikasi REST.

- Bab 3 Perancangan Modul Sistem Deteksi Wajah pada Perangkat *Mobile* dan Komunikasi REST ke Komputasi Awan.

  Bab 3 akan membahas mengenai rancang bangun salah satu bagian dari sistem pengenalan wajah yang berbasiskan pada teknologi komputasi awan khususnya implementasi pada perangkat *mobile*. Dijelaskan pula mengenai rancangan kerja program berupa diagram alir program dan diagram-diagram *Unified Modelling Language* (UML).

- Bab 4 Implementasi Sistem Deteksi Wajah pada Perangkat *Mobile* dan Komunikasi REST ke Komputasi Awan.

  Bab 4 akan membahas mengenai implementasi sistem pengenalan wajah pada perangkat *mobile* dan komunikasi REST ke *cloud server* (*cloud computing*). Bahasan implementasi mencakup implementasi pada perangkat *mobile* Android sebagai pendeteksi wajah dengan *Android Face Detector API* dan

bantuan komunikasi REST untuk melakukan proses *offloading* ke *cloud server*.

- Bab 5 Pengujian dan Analisa Modul Deteksi Wajah Pada Perangkat *Mobile* dan Komunikasi REST ke Komputasi Awan.

  Bab 5 akan membahas analisa pengujian yang telah dilakukan pada bab 4. Pengujian dan analisis hasil dilakukan berdasarkan hasil pengujian teknis yang dilakukan terhadap implementasi modul sistem pengenalan wajah pada perangkat *mobile* yang meliputi bagian deteksi wajah dan proses *offloading*.
- Bab 6 Penutup

  Bab 6 berisi penutup dari penulisan skripsi ini.

BAB 2
SISTEM PENGENALAN WAJAH DAN LAYANAN KOMPUTASI AWAN

2.1 Pengolahan Citra untuk Sistem Pengenalan Wajah

Sistem Pengenalan wajah merupakan suatu sistem yang melakukan metode rekayasa dalam sebuah citra untuk mencari identitas atau informasi yang terkandung dalam suatu citra tersebut. Sistem pengenalan wajah secara umum dibagi menjadi dua tahap [7] yaitu sistem deteksi wajah yang merupakan tahap awal (*pre-processing*) dilanjutkan dengan tahap sistem pengenal wajah (*face recognition*).

2.1.1 Sistem Deteksi Wajah

Sistem pendeteksi wajah merupakan suatu sistem yang dirancang untuk mendeteksi suatu citra apakah memiliki aspek wajah atau tidak. Jika sistem mengenali wajah dalam suatu citra tersebut, lokasi dari citra diproses dan ditampilkan seluas wajah yang ada dalam citra tersebut. Beberapa teknik yang dapat digunakan untuk mendeteksi wajah dalam suatu citra [8], antara lain :

a. *Knowledge-based method* merupakan metode yang digunakan dengan menjabarkan pengetahuan manusia tentang keunikan dari wajah manusia. Metode ini lebih cenderung ke arah mencari hubungan keunikan wajah seperti mulut, mata, dan hidung.
b. *Feature invariant approaches* merupakan suatu algoritma yang bertujuan menemukan fitur struktural yang ada pada wajah seperti posisi wajah, sudut pandang, atau kondisi pencahayaan yang berbeda-beda yang kemudian digunakan untuk mendeteksi wajah pada citra. Contoh algoritma yang menggunakan pendekatan ini adalah *Haar-Cascade like feature* atau algoritma Viola-Jones [9].
c. *Template matching methods* merupakan metode pendeteksian wajah dengan cara menyimpan pola wajah dalam suatu *template* yang terpisah. Korelasi antara gambar input dan pola yang disimpan kemudian dihitung untuk mendapatkan sistem pendeteksi wajah.

d. *Appearance based methods* merupakan metode pendeteksian wajah dengan menggunakan pembelajaran dari satu set citra dengan menangkap variabilitas dari citra yang hendak dideteksi. Dari proses pembelajaran ini kemudian dilakukan proses pendeteksian wajah. Contoh algoritma yang menerapkan teknik ini meliputi *eigenfaces, linear discriminant analysis* (LDA), *neural network, support vector machine* dan *hidden Markov Models*.

Sistem pendeteksi wajah dengan menggunakan metode Viola-Jones merupakan metode deteksi wajah yang cukup populer. Metode ini dipublikasikan oleh Paula Viola dan Michael Jones pada tahun 2001 [9]. Metode Viola-Jones memiliki empat konsep utama dalam mendeteksi wajah dalam suatu citra, yaitu :

a. Fitur segi empat sederhana (*Haar-Like Feature*)

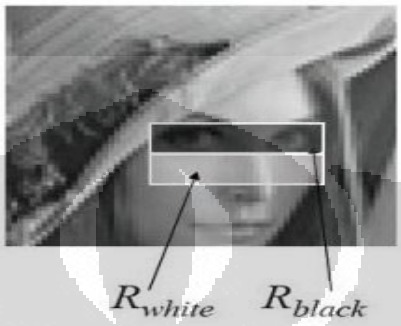
Haar feature merupakan fitur yang digunakan untuk untuk mendeteksi sebuah objek khususnya wajah. Dalam metode Viola-Jones, fitur yang digunakan berdasarkan pada *Wavelet Haar*. *Wavelet Haar* sendiri merupakan gelombang tunggal bujur sangkar yang memiliki satu interval tinggi dan satu interval rendah. Dalam suatu citra, Fitur Haar ini digambarkan dengan kondisi gelap atau terang seperti terlihat dalam Gambar 2.1.

Gambar 2.1 Fitur Haar [9]

Dalam metode Viola-Jones, *Haar feature* ini lebih dikenal dengan nama *Haar-Like feature*. Suatu *Haar-Like feature* akan memproses citra dalam wilayah kotak-kotak yang berisi beberapa piksel dari sebuah bagian citra. Piksel tersebut kemudian dijumlahkan dan dilakukan proses perhitungan sehingga didapatkan perbedaan dalam setiap wilayah kotak-kotak tersebut. Perbedaan inilah yang dapat dijadikan sebuah kode untuk menandai wilayah tersebut sehingga bagian citra yang diinginkan dapat diproses. Perbedaan nilai citra tersebut dapat dihitung dengan persamaan 2.1.

$$f(x) = \Sigma(piksel\ dalam\ area\ terang) - \Sigma(\text{piksel dalam area gelap}) \qquad (2.1)$$

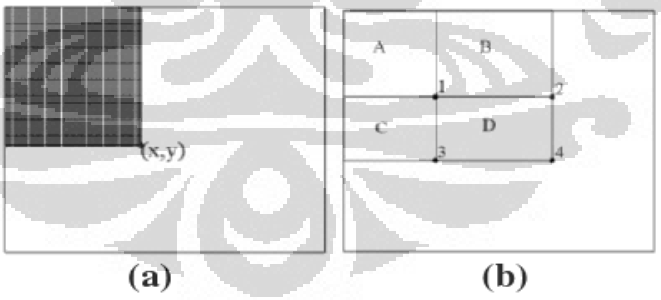
Sebagai contoh, citra wajah mata umumnya memiliki warna yang lebih gelap daripada citra di daerah pipi yang lebih terang. Dengan menggunakan *Haar-Like feature* daerah mata dan pipi dapat dideteksi dengan melakukan perhitungan perbedaan piksel dalam area gelap dan terang seperti terlihat dalam Gambar 2.2.



Gambar 2.2 Ilustrasi kerja Haar-Like Feature [9]

b. Citra integral

Citra integral merupakan metode untuk menghitung nilai *Haar-Like feature* pada metode Viola-Jones. Metode citra integral digunakan untuk melakukan perhitungan dengan cepat tanpa perlu menghitung seluruh nilai piksel yang terdapat didalam fitur. Nilai integral untuk setiap piksel didapatkan dari penjumlahan dari semua piksel di atas dan di kiri piksel tersebut.



Gambar 2.3 Citra Integral [9]

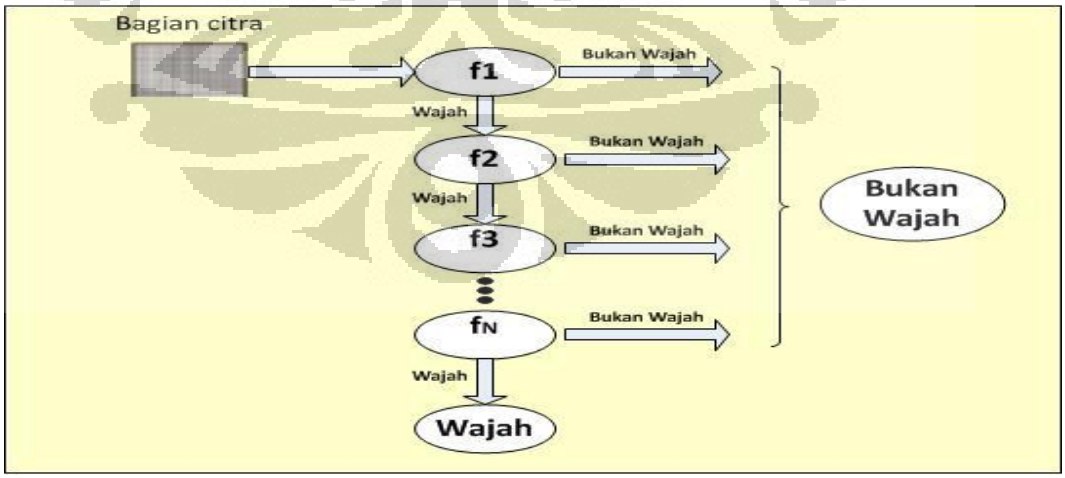
Dalam Gambar 2.3(a) dapat diketahui setelah proses integrasi, nilai lokasi piksel (x,y) berisi jumlah dari semua piksel di dalam daerah yang diarsir. Begitu pula dengan Gambar 2.3(b), untuk mengetahui nilai piksel misalnya segiempat D, perhitungan dapat dilakukan dengan cara D = (A+B+C+D) - (A+B) - (A+C) + A.

c. Metode *machine learning AdaBoost*

Adaboost merupakan metode pembelajaran dalam metode Viola-Jones untuk mencari nilai ambang (*threshold)* dalam suatu sistem deteksi wajah. Nilai *threshold* ini merupakan nilai perbedaan antara nilai fitur yang telah didapatkan. *Adaboost* akan menggabungkan banyak *classifier* lemah (*weak classifier*) untuk membuat *classifier* yang kuat. Lemah disini berarti urutan filter pada *classifier* hanya mendapatkan jawaban benar lebih sedikit. Dalam metode Viola-Jones, terjadi penggabungan beberapa *Adaboost classifier* sebagai rangkaian filter yang cukup efisien untuk menggolongkan daerah citra. Masing – masing filter adalah satu *Adaboost classifier* terpisah yang terdiri *classifier* lemah atau satu filter Haar.

d. Pengklarifikasian bertingkat (*Cascade Classifier*)

Cascade Classifier merupakan suatu langkah untuk menghubungkan banyak fitur secara efisien seperti terlihat dalam Gambar 2.4. Dalam gambar tersebut, filter pada masing-masing level dilatih untuk mengklarifikasi citra yang sebelumnya telah difilter. Selama penggunaannya, jika salah satu dari filter tersebut gagal, *image region* atau daerah pada citra diklarifikasikan sebagai bukan wajah. Saat filter berhasil melewatkan *image region, image region* tersebut kemudian masuk pada filter yang selanjutnya. *Image region* yang telah melalui semua filter akan dianggap sebagai wajah.



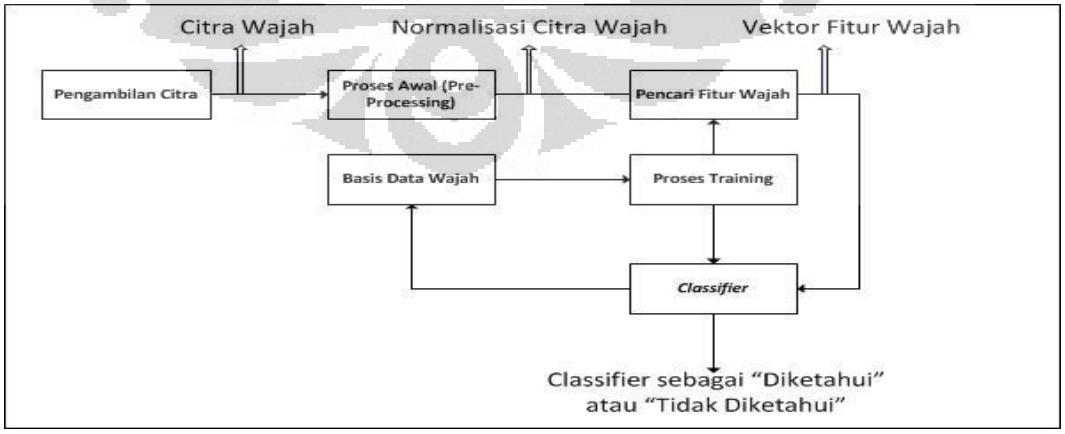
Gambar 2.4 *Cascade Classifier* [9]

2.1.2 Sistem Pengenalan Wajah

Sistem pengenalan wajah (*face recognition*) merupakan suatu sistem yang dirancang pada komputer yang bertujuan untuk mengidentifikasi wajah seseorang melalui citra atau video digital. Salah satu metode yang sering digunakan dalam sistem pengenalan wajah ini adalah dengan cara membandingkan keunikan wajah (*facial features*) dari suatu citra dengan *database* wajah yang telah diambil sebelumnya. Beberapa algoritma dalam suatu sistem identifikasi wajah didasarkan pada ekstraksi dari *landmark* maupun keunikan yang didapatkan pada citra dengan subjek wajah seperti posisi atau ukuran mata, hidung, pipi, dan rahang [10].

Beberapa pendekatan yang digunakan dalam sistem pengenalan wajah diantaranya :

a. *Geometric (Feature Based Matching)* adalah sebuah pendekatan yang mana sebuah wajah dapat dikenali dengan menggunakan *geometrical feature* seperti lebar hidung, posisi pipi, posisi mata dengan cara melakukan ekstraksi posisi relatif dari parameter pembeda pada wajah seperti mata, mulut, hidung, dan pipi [11].
b. *Template Matching* merupakan sebuah pendekatan dalam sistem pengenalan wajah yang direpresentasikan ke dalam larik dua dimensi, kemudian dibandingkan dengan matriks yang sesuai ke dalam satu *template* yang telah mempresentasikan seluruh bagian wajah [11].



Gambar 2.5 Alur Kerja Sistem Pengenalan Wajah [7]

Gambaran mengenai bagaimana alur kerja umum dari suatu sistem pengenalan wajah terdapat dalam Gambar 2.6. Secara menyeluruh sistem pengenalan wajah dapat dilakukan dengan beberapa cara [10], yaitu :

1. *Detection*

   Detection merupakan proses pengambilan data berupa citra yang dapat diperoleh dengan melakukan proses pemindaian (*scanning*) dari foto 2D atau dapat pula menggunakan citra video yang didapatkan dari *live recording* (3D).

2. *Alignment*

   Setelah wajah telah terdeteksi, sistem kemudian menentukan posisi wajah, ukuran, dan pose.

3. *Measurement*

   Sistem kemudian melakukan komputasi berupa pembentukan kurva dari wajah dan membuat suatu *template* untuk menampung hasil komputasi dari kurva tersebut.

4. *Representation*

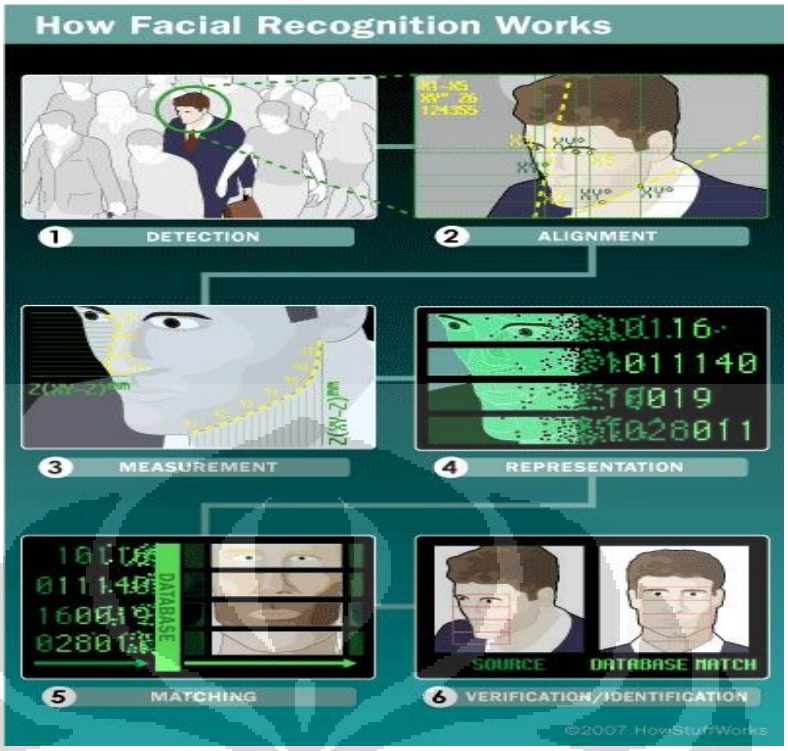
   Setelah melakukan *measurement*, sistem menerjemahkan *template* yang telah dibuat dalam bentuk kode unik (*unique code*). Pengkodean inilah yang akan memberikan representasi dari keunikan wajah dalam suatu citra.

5. *Matching*

   Citra hasil pengkodean kemudian dibandingkan dengan *database* citra yang telah dibuat sebelumnya.

6. *Verification atau Identification*

   Setelah itu, sistem dapat memberikan hasil pengenalannya terhadap citra yang ingin dikenali identitasnya.



Gambar 2.6 Proses dalam Sistem Pengenalan Wajah [10]

2.2 Teknologi Komputasi Awan

Sistem komputasi awan merupakan merupakan suatu gabungan pemanfaatan teknologi komputer sebagai pemroses komputasi dan pengembangan berbasis Internet yang mana kapabilitas terkait teknologi informasi disajikan sebagai suatu layanan (*as a service*). Adapun beberapa definisi mengenai pengertian sistem komputasi awan, antara lain :

a. Sistem komputasi awan merupakan paradigma dalam distribusi sistem dalam skala besar yang didorong oleh skala ekonomi dimana sistem komputasi bersifat abstrak dan terdapat proses virtualisasi. Komputasi energi, media penyimpanan, media pengembangan, dan pelayanan dilakukan dengan berprinsip pada permintaan yang dinginkan pelangggan (*on demand)* melalui media internet [12].
b. Sistem komputasi awan memiliki keunikan dari *grid computing*, *cluster computing*, *service computing* [13], dan *utility computing* [14].

Dari pengertian yang telah disebutkan di atas, dapat disimpulkan bahwa sistem komputasi awan merupakan sebuah paradigma baru yang mengutamakan

pelayanan. Pengguna komputer tidak perlu direpotkan terhadap hal–hal teknis seperti media penyimpanan, kerusakan perangkat keras ataupun perangkat lunak. Semua fasilitas komputasi telah disediakan oleh *cloud server* sehingga sistem komputasi menjadi lebih praktis dan efisien.

Secara umum, sistem komputasi awan [13,14] terdiri dari tiga kategori pelayanan, yaitu :

1 *Infrastructure as a Service* (IaaS)

Pada bagian ini infrastruktur menjadi hal yang dijadikan sebuah pelayanan dimana infrastruktur yang lengkap seperti teknologi *cluster*, *grid* dan virtualisasi telah ditawarkan kepada para pengguna yang ingin memanfaatkan layanan komputasi awan. Dalam hal ini, pengguna tidak perlu direpotkan lagi dengan masalah *maintenance hardware* karena infrastruktur dalam sistem komputasi ini memiliki skalabilitas besar.

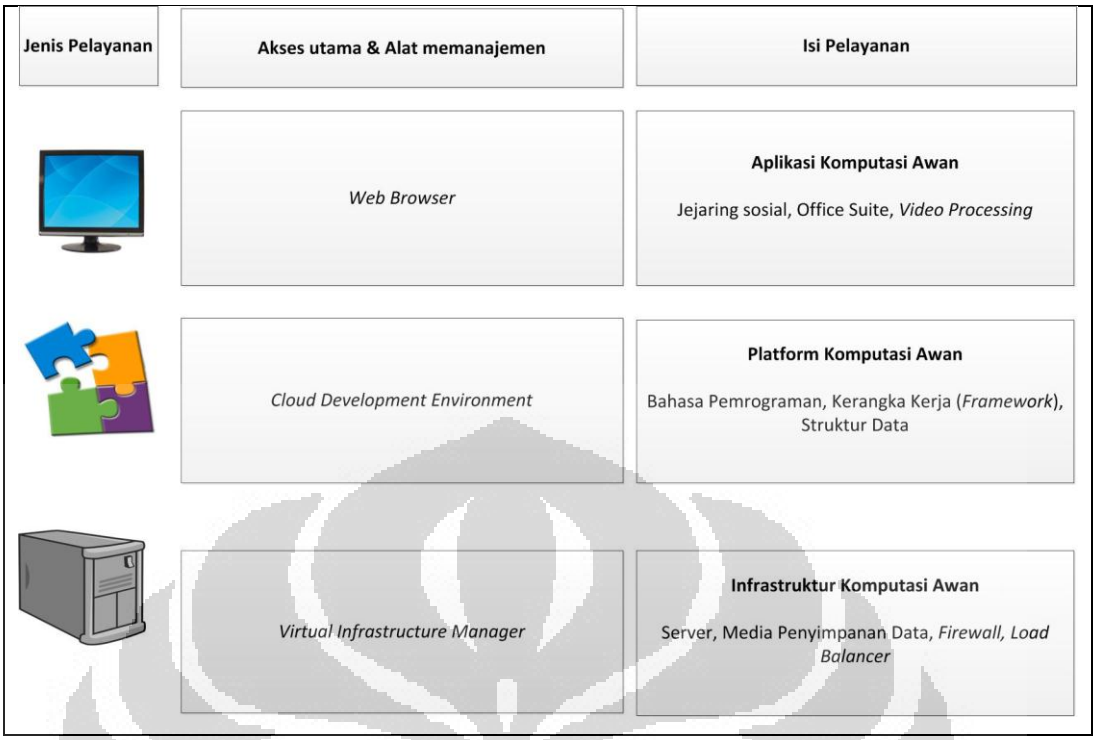
2 *Platform as a Service* (PaaS)

PaaS merupakan sebuah media yang dapat digunakan untuk membangun suatu aplikasi sendiri dalam infrastruktur sistem komputasi awan. Sebagai contoh *Google App Engine* (GAE) yang dapat memungkinkan pengguna mengembangkan aplikasi berbasis web yang terintegrasi dengan aplikasi Google.

3 *Software as a Service* (SaaS)

SaaS merupakan sebuah sarana yang diberikan terutama oleh penyedia layanan sistem komputasi awan dalam memberikan pelayanan *software*. Layanan ini biasanya diimplementasikan dalam bentuk aplikasi berbasis web yang memungkinkan untuk diakses dimana saja dan kapan saja asalkan jaringan internet tersedia. Sebagai contoh aplikasi dari SaaS ini adalah layanan *email* dari Google (Gmail) dan *Google Docs* yang memungkinkan membuat suatu dokumen melalui internet.

Gambaran mengenai kategori pelayanan seperti terlihat pada Gambar 2.7 yang memperlihatkan metode pengaksesan dan pelayanan yang dari suatu sistem komputasi awan.



Gambar 2.7 Bagian Sistem Komputasi Awan [15]

Proses *deployment model* dalam sistem komputasi awan dapat dibedakan menjadi tiga macam [15] yaitu :

1. *Public Cloud*

   Sistem komputasi awan yang dibuat dengan tujuan dapat digunakan secara publik yang berarti ketersediaan pelayanan dari sistem komputasi awan ini tergantung dari berapa biaya layanan yang dibayarkan. Layanan publik *cloud* secara umum disediakan oleh perusahaan-perusahaan penyedia layanan untuk semua pelanggan di seluruh dunia.

2. *Private Cloud*

   Suatu sistem komputasi awan yang mana hanya berlaku dalam suatu kegiatan bisnis ataupun suatu organisasi tertentu dan *data center* terletak secara internal dalam suatu perusahaan tertentu. Sistem komputasi awan ini menerapkan teknologi yang sama dengan *public cloud* tetapi tidak dibuka secara publik.

3. *Hybrid Cloud*

   Hybrid cloud merupakan suatu gabungan antara sistem komputasi *public cloud* dengan *private cloud*. Infrastruktur sistem komputasi ini merupakan komposisi dari dua atau lebih infrastruktur *cloud* (*private* atau *public*).

Secara entitas, infrastruktur dari *hybrid cloud* tetap berdiri sendiri dan dihubungkan oleh suatu mekanisme yang memungkinkan portabilitas data dan aplikasi antar *cloud* tersebut.

2.3 Sistem Komunikasi REST

Representational State Transfer (REST) merupakan arsitektur *software* untuk distribusi atau komunikasi layanan web yang memiliki fokus pada sumber daya sistem [16]. Istilah REST sendiri diperkenalkan dan didefinisikan pada tahun 2000 oleh Roy T Fielding pada disertasi doktoralnya yang juga merupakan penggagas spesifikasi HTTP 1.0 dan HTTP 1.1. Sistem komunikasi REST disebut juga sebagai *RESTful services* dikarenakan sistem komunikasi yang digunakan menekankan pada kesederhanaan, skalabilitas, dan kegunaan.

Sistem REST akan menghubungkan pihak *client* dan *server* serta memanfaatkan fitur yang sama pada aplikasi HTTP seperti metode *GET, POST, PUT, DELETE*. Prinsip dari REST sebagai media *web service* terbagi menjadi 4 macam, yaitu

a. Menggunakan metode HTTP secara eksplisit

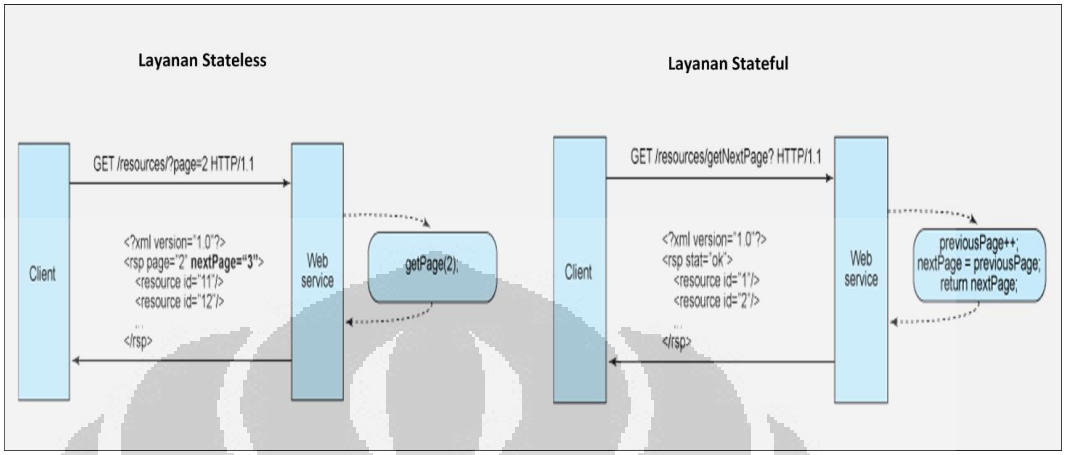
   Prinsip dasar REST memiliki pemetaan terhadap operasi *create, read, update, delete* (CRUD) dan metode dalam HTTP seperti,

   - *POST* untuk membuat *resource* di *server.*
   - *GET* untuk menerima sebuah *resource.*
   - *PUT* untuk melakukan pembaruan atau pengubahan terhadap *resource.*
   - *REMOVE* untuk menghapus *resource.*

b. Bersifat *stateless*

   Stateless web service mengindikasikan bahwa layanan akan memberikan sebuah respon yang terhubung dengan halaman *resource*. Hal ini berbeda dengan *stateful service* yang menyimpan variabel dari *request* sebelumnya dan menggunakannya kembali dengan tambahan variabel baru untuk mengakses *resource* yang diminta oleh sebuah *request* baru. Ketika *client* membuat HTTP *request*, semua informasi yang dibutuhkan *server* untuk memenuhi *request* harus dikirim sehingga *server* tidak tergantung pada

informasi yang dikirim dari *request* sebelumnya. Perbedaan mengenai gambaran komunikasi dengan layanan *stateless* dan *stateful* ini terlihat dalam Gambar 2.8.



Gambar 2.8 Ilustrasi Perbedaan Layanan *Stateless* dan *Stateful* [16]

c. Memiliki struktur direktori URI

   Metode pengaksesan layanan web REST akan menggunakan sebuah *Uniform Resource Identifiers* (URI). URI ini merupakan nama dan alamat dari sebuah *resource* yang ingin dicari.

d. Pesan yang ditransfer dalam format XML, JSON, atau keduanya.

   Dalam arsitektur REST, pesan balik akan dikirimkan kembali kepada pihak *client* dalam bentuk format *Extensible Markup Language* (XML) ataupun JSON sebagai bentuk respon dari *request* yang masuk ke *server*.

2.4 Bahasa Pemrograman Sebagai Implementasi Sistem

Bagian ini akan menjelaskan mengenai bahasa pemrograman yang akan digunakan dalam implementasi sistem pada perangkat *mobile*. Hal lain yang akan dibahas dalam bagian ini adalah format file pertukaran data (JSON), media pengembangan (*platform*) Android, dan API jejaring sosial.

2.4.1 Bahasa Pemrograman Java

Java merupakan bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk pada perangkat *mobile*. Bahasa pemrograman ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems* yang merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak

mengadopsi sintaksis yang terdapat pada bahasa C dan C++ namun dengan sintaksis model objek yang lebih sederhana sehingga mendukung suatu pemrograman yang berorientasi kepada objek. Aplikasi berbasis *java* umumnya dikompilasi ke dalam *p-code* (kode *byte*) dan dapat dijalankan pada berbagai Mesin Virtual *Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum (*general purpose*) dan secara khusus didesain untuk memanfaatkan *dependensi* implementasi seminimal mungkin. Saat ini *java* merupakan bahasa pemrograman yang paling populer digunakan dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

2.4.2 JSON

JavaScript Object Notation (JSON) adalah format pertukaran data yang ringan, mudah dibaca, dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer [17]. Format ini dibuat berdasarkan bahasa pemrograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON menggunakan format teks yang tidak tergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh pemrogram keluarga C. Oleh karena sifatnya tersebut, JSON menjadi bahasa program yang ideal sebagai bahasa pertukaran data antara *server* dan *client*.

JSON juga mempunyai format tersendiri yaitu terdiri dari 2 struktur, yaitu objek dan larik (*array*). Struktur data ini disebut struktur data *universal*. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama ataupun berlainan. Hal ini seperti terlihat dalam Gambar 2.9 yang menggambarkan struktur dari JSON. Penjelasan mengenai struktur dari JSON adalah sebagai berikut :

a. Objek

   Objek merupakan sepasang nama atau nilai yang ditak terurutkan. Objek dimulai dengan tanda { (kurung kurawal buka) dan diakhiri dengan tanda } (kurung kurawal tutup). Setiap nama diikuti dengan tanda : (titik dua) dan setiap pasangan nama atau nilai dipisahkan oleh tanda , (koma).

b. Larik

Larik (*array*) merupakan kumpulan nilai yang terurutkan. Larik dimulai dengan tanda [(kurung kotak buka) dan diakhiri dengan tanda] (kurung kotak tutup) serta setiap nilai dipisahkan oleh tanda tanda , (koma).

Gambar 2.9 Struktur Umum JSON [17]

2.4.3 API jejaring sosial

Application programming interface (API) jejaring sosial ini merupakan sebuah penghubung antara aplikasi yang dirancang dalam sistem ini dengan jejaring sosial yang ada seperti *facebook*, *twitter*, *koprol*, *foursquare*, dan *google+*. API ini akan menyediakan antarmuka untuk mempermudah proses pengembangan suatu aplikasi yang terhubung dengan jejaring sosial. Dalam proses pemrogramannya API jejaring sosial akan menjadi suatu pustaka yang akan dipanggil ketika suatu aplikasi memanggil fungsi integrasi terhadap situs jejaring sosial tertentu. Dalam penulisan skripsi ini , API jejaring sosial yang akan dipilih adalah API *facebook* [18].

2.4.4 Media Pengembangan Android

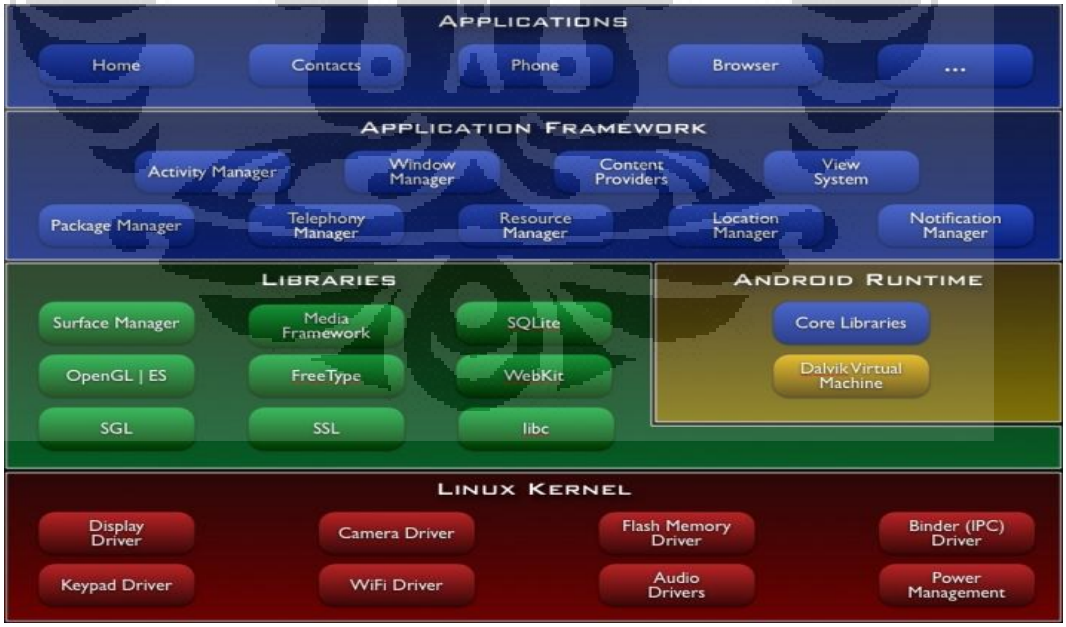
Bagian ini akan menjelaskan bagian umum dari media pengembangan aplikasi berbasis Android termasuk fitur-fitur yang ada dalam *platform* Android. Selain itu, akan dibahas juga mengenai penggunaan Android *Face Detector* API sebagai pustaka untuk melakukan deteksi wajah.

2.4.4.1 Deskripsi Android

Android merupakan suatu sistem operasi untuk perangkat *mobile* khususnya *smartphone* yang berbasis Linux. Android menyediakan *platform* terbuka (*Open Source*) bagi para pengembang yang ingin menciptakan aplikasi mereka sendiri untuk digunakan dalam berbagai perangkat *mobile*. Sistem operasi

ini dikembangkan oleh Google Inc. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Dalam pengembangan perangkat Android, dibentuk suatu perkumpulan dinamakan Open Handset Alliance, konsortium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Beberapa fitur yang tersedia pada sistem operasi Android, yaitu :

a. Kerangka aplikasi yang memungkinkan penggunaan dan penghapusan komponen yang tersedia.
b. *Dalvik virtual machine* yang merupakan mesin virtual yang dioptimalkan untuk perangkat *mobile* sehingga mendukung proses *multitasking*.
c. Grafik: grafik di 2D dan grafis 3D berdasarkan pustaka grafik OpenGL.
d. SQLite sebagai media untuk penyimpanan data.
e. Mendukung media audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
f. GSM, Bluetooth, EDGE, 3G, dan WiFi.
g. Kamera, *Global Positioning System* (GPS), kompas, dan *accelerometer*.
h. *Multitouch user-input* sebagai antarmuka dengan pengguna.



Gambar 2.10 Arsitektur Diagram *Platform* Android [19]

Dalam diagram arsitektur Android yang terdapat pada Gambar 2.10 menunjukkan bahwa Android menggunakan *Linux kernel* sebagai pemegang

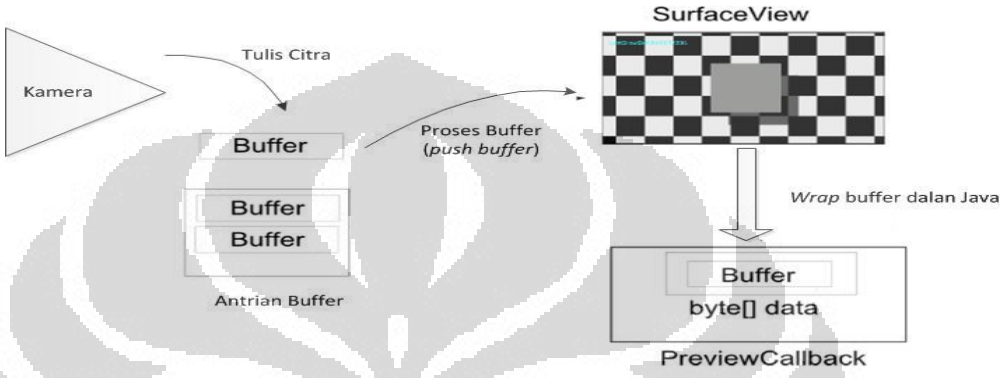
kendali atas semua perangkat keras yang ada dalam perangkat *mobile* Android. Lapisan diatasnya terdapat beberapa pustaka yang menjadi referensi Android dalam melakukan fungsi operasionalnya. Hal yang membedakan *platform* Android dengan beberapa distribusi linux adalah penggunaan *Android Runtime* yang berupa *Dalvik Virtual Machine* (Dalvik VM). Dalvik VM ini merupakan proses virtualisasi mesin tetapi memiliki perbedaan dengan *java virtual machine* (JVM) yang mana Dalvik lebih mengedepankan optimalisasi memori sehingga cocok digunakan untuk perangkat *mobile* yang mana memiliki sumber daya yang terbatas.

Pengembang dapat membuat aplikasi untuk perangkat Android dengan bahasa pemrograman berbasis pada Java. Pemrograman pada Android juga dapat dilakukan dengan mengintegrasikan kode *native* seperti C/C++ sehingga memberikan kemudahan ketika hendak mengembangkan aplikasi yang mendukung pemrograman seperti OpenCV. Untuk mengembangkan aplikasi Android terlebih dahulu diperlukan suatu *Software Development Kit* (SDK) yang bisa diintegrasikan dengan aplikasi *Integrated Development Environment* (IDE) pemrograman seperti Eclipse. Pengembangan aplikasinya pun bisa dilakukan ke dalam beberapa versi sistem aplikasi pada Android. Sampai saat penulisan skripsi ini, SDK yang dapat digunakan telah mencapai *Application Programming Interface* (API) 15 yang memungkinkan pengembang untuk mengembangkan aplikasi pada sistem Android terbaru yaitu *Ice Cream Sandwich*. Pengembangan sistem pengenalan wajah dalam penulisan skripsi ini akan menggunakan versi API 9 yaitu *Android Gingerbread*.

2.4.4.2 Android *Face Detector* API

Android *Face Detector* API merupakan sebuah pustaka yang dipergunakan untuk mencari aspek wajah pada suatu *bitmap*. Android *Face Detector* API merupakan API alami (*native*) yang telah disediakan oleh Android yang mulai dirilis pada API level 1 [20]. API ini terdapat pada *android.media.FaceDetector* yang bekerja dengan mengakses metode *findFaces* pada citra yang ingin diolah. Gambar 2.11 menjelaskan bahwa Android *Face Detector* API akan memproses suatu *bitmap* yang diperoleh dari kamera android.

Bitmap sendiri memiliki pengertian sebagai bentuk representasi citra grafis yang terdiri dari susunan titik yang tersimpan di dalam memori. *Bitmap* yang didapat dari kamera android akan diproses terlebih dahulu ke dalam format *grayscale* untuk dapat dilakukan proses deteksi wajah. Format *grayscale* ini akan ditampung ke dalam *buffer* untuk dilakukan proses ekstraksi fitur wajah. Ilustrasi dari Android *Face Detector* API seperti terlihat pada Gambar 2.11.



Gambar 2.11 Ilustrasi Kerja Android Face Detector API [19]

Dalam melakukan proses deteksi wajah, Android *Face Detector* API memiliki 2 fungsi untuk mencari aspek wajah pada suatu *bitmap,* yaitu :

a. *Android.media.FaceDetector*

   Fungsi ini merupakan fungsi yang digunakan untuk mencari atau mengidentifikasi aspek wajah dalam suatu bitmap.

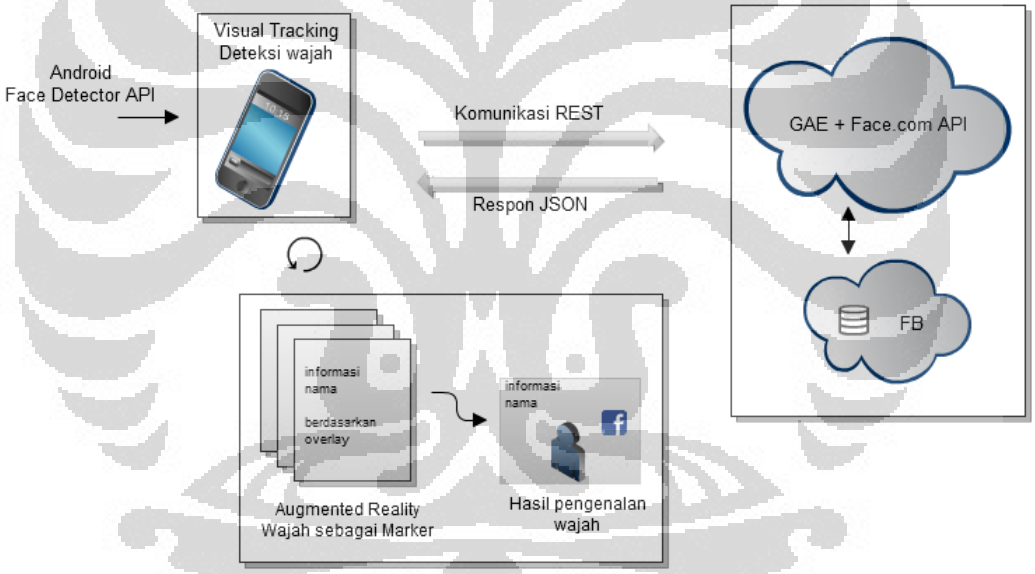
b. *Android.media.FaceDetector.Face*

   Fungsi ini akan menjelaskan informasi mengenai aspek wajah yang ada dalam suatu bitmap, misalnya pose wajah, jarak antar mata, dan tingkat kepercayaan (*confidence*).

BAB 3
PERANCANGAN MODUL SISTEM DETEKSI WAJAH DAN KOMUNIKASI REST KE KOMPUTASI AWAN

3.1 Rancangan Cara Kerja Sistem

Bagian ini akan menjelaskan mengenai rancangan modul deteksi wajah beserta cara kerja program yang dihubungkan ke *cloud server* melalui komunikasi REST. Modul deteksi wajah pada perangkat *mobile* merupakan salah satu bagian dari keseluruhan sistem pengenalan wajah seperti terlihat pada Gambar 3.1. Penjelasan lebih lanjut mengenai deskripsi sistem dan alir kerja sistem akan dibahas pada Subbab 3.1.1 dan Subbab 3.1.2.



Gambar 3.1 Gambaran Sistem Umum Secara Keseluruhan

3.1.1 Deskripsi Sistem

Sistem pengenalan wajah sebagai penghubung jejaring sosial yang akan dibangun dalam penulisan skripsi ini merupakan gabungan dari tiga sistem kerja. Ketiga sistem tersebut meliputi modul deteksi wajah pada perangkat *mobile* yang dihubungkan ke *cloud server* dengan komunikasi REST, penerapan pengenalan wajah dengan komputasi awan dalam basis web dengan *Google App Engine*

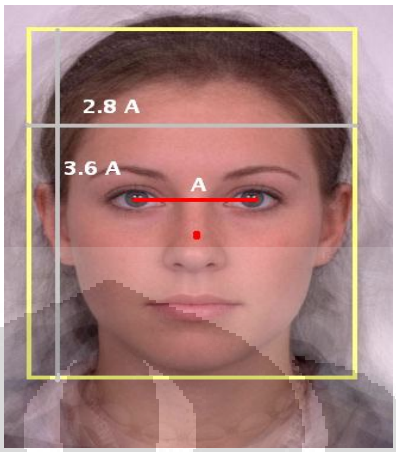
(GAE) dan *Face* API [4], dan penambahan realita tertambah (*augmented reality)* sebagai penambah antarmuka informasi hasil dari fungsi pengenalan wajah [5].

Seperti yang telah dijelaskan dalam bagian pendahuluan tulisan ini, sistem pengenalan wajah ini merupakan salah satu bagian dari sistem pengenalan wajah keseluruhan yang mana lebih memfokuskan pada penerapan dalam perangkat *mobile*. Sistem ini dirancang dengan menggunakan perangkat *mobile* Android untuk proses deteksi wajah yang berperan sebagai *client* dan perancangan komunikasi pada *cloud server* sebagai penyedia jasa layanan komputasi awan untuk melakukan proses pengenalan wajah. Kedua sistem ini kemudian diintegrasikan ke dalam perangkat *mobile* dengan melakukan suatu komunikasi *Representational State Transfer* (REST) data dari perangkat *mobile* ke *cloud server*. Dengan sistem ini diharapkan menjadi suatu sistem pengidentifikasi diri seseorang secara efektif dengan menggunakan penerapan sistem pengenalan wajah yang diintegrasikan dengan layanan komputasi awan.

Mula-mula sistem ini melakukan suatu komputasi secara *onload* yang berarti proses komputasi yang dilakukan secara individu oleh perangkat *mobile*. Proses komputasi yang dilakukan yakni suatu proses pendeteksian wajah dari suatu citra yang diambil dari kamera yang ada pada perangkat *mobile* dengan Android *Face Detector* API. Hasil dari pemotretan dari kamera ini kemudian dilakukan proses komputasi sehingga menghasilkan suatu ukuran citra dengan format JPEG yang hanya mencakup aspek wajah yang ada pada citra tersebut. Proses ini akan dilakukan oleh metode *android.media.FaceDetector.Face.* Dalam metode ini terjadi proses konversi ke dalam citra *bitmap* agar bisa dilakukan proses pemotongan area wajah. Algoritma yang digunakan dalam proses pemotongan area wajah ini adalah

1. Mendapatkan titik tengah (*mid points*) dari wajah yang terdapat dalam citra, dengan tingkat keyakinan (*confidence*) lebih besar dari 0,4 [19].
2. Menghitung jarak persegi sekitar wajah yang telah dideteksi seperti yang terlihat pada Gambar 3.2 sehingga didapatkan koordinat wajah. Dalam gambar tersebut jarak antar mata dimisalkan dengan A.

3. Setelah itu dilakukan proses pemotongan area wajah sesuai koordinat yang telah dihitung.



Gambar 3.2 Ilustrasi Proses Pemotongan Area Wajah

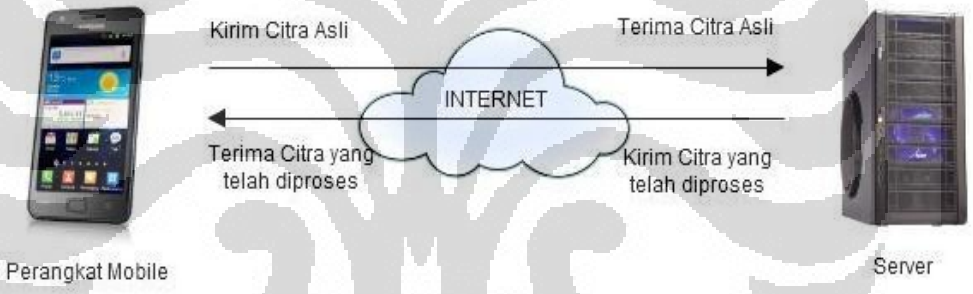
Setelah didapatkan suatu hasil citra yang telah dideteksi area wajahnya, proses komputasi kemudian dipindahkan ke *cloud* atau bisa disebut dengan proses *offloading*. Proses *offloading* yang terjadi akan melakukan proses pengenalan wajah lebih lanjut yang dilakukan di *cloud server*.

Setelah data berupa citra telah terkirim ke *cloud*, sistem pengenalan wajah selanjutnya dikomputasi oleh *server* pada *cloud*. Hasil dari komputasi ini kemudian dikirimkan kembali kepada perangkat *mobile* yang mana akan memberikan hasil identitas diri orang yang ingin diketahui tersebut. Gambaran mengenai proses ini dapat dilihat pada Gambar 3.3 yang menjelaskan tentang komunikasi yang terjadi antara *cloud server* dan perangkat *mobile*.

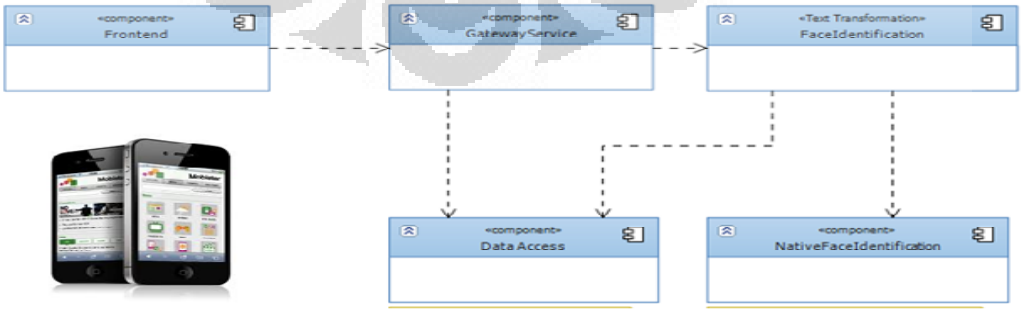
Gambar 3.3 Arsitektur Sistem Pengenalan Wajah pada Perangkat *Mobile*

Dalam sistem ini akan digunakan bahasa pemrograman Java dan *native code* C++ sebagai pengatur pendeteksian wajah yang diimplementasikan pada perangkat *mobile* Android sebagai pihak *client*. Proses deteksi wajah pada perangkat *mobile* akan menggunakan pustaka tambahan yaitu *Android Face Detector API* [19].

Proses komunikasi dalam sistem ini akan menggunakan proses komunikasi paket data yang menggunakan kanal data sebagai media komunikasi antara perangkat *mobile* dengan *cloud server*. Selain itu, penggunaan metode komunikasi *Representational State Transfer* (REST) juga akan diterapkan pada implementasi sistem pengenalan wajah ini. Gambaran lengkap mengenai proses komunikasi data dapat dilihat pada Gambar 3.4 dan Gambar 3.5.



Gambar 3.4 Sistem Arsitektur Paket Data



Gambar 3.5 Proses Komunikasi Data pada Sistem

3.1.2 Diagram Alir Kerja Program

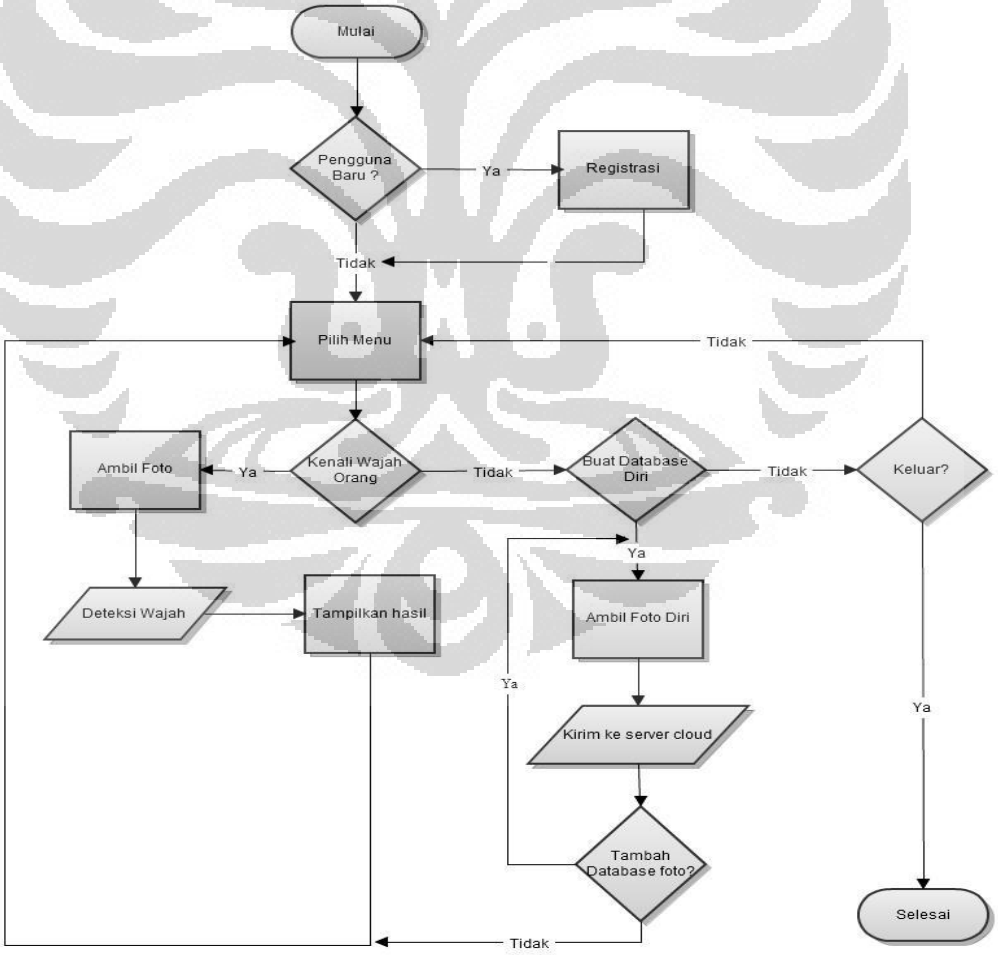
Diagram alir kerja sistem terutama proses implementasi dari sistem akan ditunjukkan oleh Gambar 3.6. Pada bagian awal dari program dalam perangkat *mobile*, pengguna diharuskan melakukan *user login*. Apabila belum terdaftar di dalam sistem, pengguna akan diberi kesempatan untuk melakukan registrasi terlebih dahulu. Setelah melakukan registrasi akun, pengguna baru akan dapat masuk ke dalam sistem. Pada bagian awal setelah melakukan *user login*, pengguna akan diberikan beberapa pilihan menu meliputi pengenalan wajah orang, pembuatan *database* diri, dan keluar dari sistem.

Pada bagian pengenalan wajah orang, pengguna akan dihadapkan dalam fungsi pengambilan foto seseorang yang ingin dikenali. Proses pengambilan foto ini dilakukan layaknya pada pemotretan foto pada umumnya. Setelah pengambilan foto dilakukan, sistem yang berada dalam perangkat *mobile* akan melakukan pemrosesan citra. Pemrosesan ini meliputi proses pendeteksian wajah, yang mana foto yang telah diambil akan dideteksi bagian wajahnya saja sehingga akan terbentuk suatu citra wajah yang berukuran resolusi kecil. Citra wajah yang telah dideteksi ini kemudian akan dikirimkan ke *cloud server* untuk dilakukan proses pengenalan wajah lebih lanjut. Setelah *server* memproses citra tersebut, hasil dari sistem pengenalan wajah pada *cloud server* akan dikirimkan kembali kepada perangkat *mobile* yang mana akan ditampilkan dalam layar perangkat *mobile*. Identitas diri dari pemilik foto tersebut akan ditampilkan ke pengguna yang menampilkan pula informasi jejaring sosial yang dimilikinya sehingga pengguna dapat mengetahui siapa sebenarnya orang yang sedang ingin dikenali.

Pada bagian kedua dalam sistem ini, pengguna akan diberikan pilihan untuk membuat *database* diri masing-masing pengguna. Hal ini dimaksudkan untuk membangun suatu *database* wajah yang mana akan berguna dalam proses yang terjadi dalam sistem pengenalan wajah ini. Pengguna dapat pula melakukan pembaruan *database* wajahnya pada pilihan ini sehingga *database* wajah pengguna akan selalu menjadi informasi terkini. Proses yang terjadi pada bagian kedua dari sistem ini memiliki kesamaan dengan yang terjadi pada proses bagian pertama. Perbedaannya adalah pada bagian kedua ini objek yang dideteksi adalah

wajah pengguna itu sendiri dan pengguna pun dapat melakukan penambahan informasi terkait informasi yang diperlukan layaknya sebuah kartu identitas.

Apabila pengguna ingin keluar dari program ini, pengguna dapat memilih pilihan keluar. Beberapa penjelasan diatas merupakan pembahasan yang terdapat pada perangkat *mobile* yang mana bagian ini merupakan bagian yang berinteraksi langsung dengan pengguna atau bisa disebut juga bagian *front-end*. Bagian lain pada sistem ini yakni bagian *back-end* yang terkait dengan hubungan komunikasi yang terjadi antara perangkat *mobile* dengan *cloud server*. Proses komunikasi dengan *cloud server* akan menggunakan jenis komunikasi REST. Bagian *back-end* ini akan berusaha dengan sepenuhnya menjamin proses pengenalan wajah dalam implementasi sistem dari penelitian skripsi ini dapat terwujud.



Gambar 3.6 Diagram Alir Kerja Sistem

3.2 Diagram-Diagram *Unified Modelling Language* (UML)

Pada bagian ini akan dijelaskan mengenai *Unified Modelling Language* (UML) yang merupakan sebuah metode untuk mendeskripsikan desain dari perangkat lunak kedalam notasi grafis yang telah terstandardisasi [6]. UML ini juga dapat memvisualisasikan, merancang dan mendokumentasikan arsitektur dari sistem perangkat lunak yang dibuat. Berikut adalah diagram-diagram rancangan sistem perangkat lunak dari modul yang dibuat, meliputi *use case diagram*, *sequence diagram*, *deployment diagram*, dan *class diagram*.

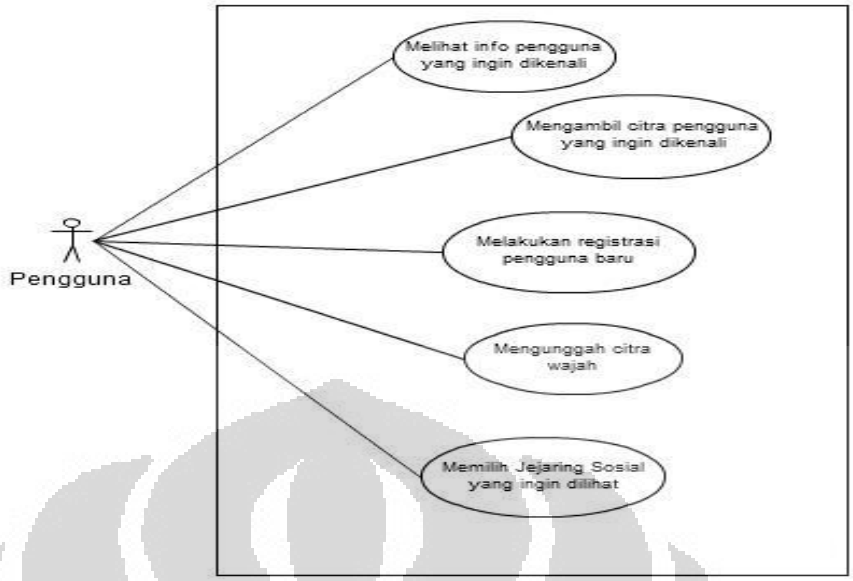
3.2.1 *Use Case Diagram*

Use Case Diagram ini akan menjelaskan fungsi yang terdapat pada sistem mengenai fungsi apa saja yang dapat dilakukan oleh pengguna [6]. *Use Case Diagram* dari sistem ini akan ditunjukkan pada Gambar 3.7. Pada diagram tersebut ditunjukkan bahwa secara umum pengguna dapat melakukan beberapa aktifitas dalam sistem pengenalan wajah berbasis komputasi awan ini. Fungsi yang terdapat dalam sistem pengenalan wajah pada perangkat *mobile* meliputi dua macam :

1. Pengguna yang ingin membuat *database* wajah supaya dikenali pengguna lainnya.

   Pada bagian ini, pengguna dapat melakukan registrasi sebagai pengguna baru sistem pengenalan wajah ini, dan pengguna dapat mengunggah citra wajah dirinya ke sistem supaya dikenali pengguna lain dalam sistem ini.
2. Pengguna yang ingin mengetahui identitas pengguna lain.

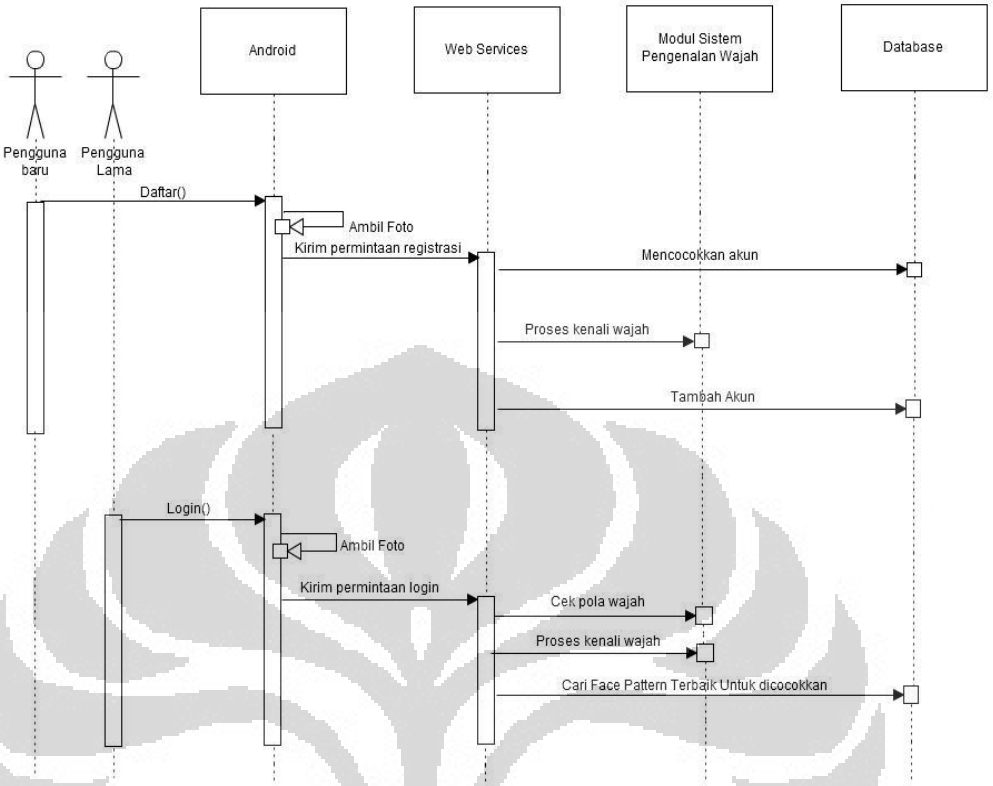
   Pada bagian ini, pengguna dapat melihat informasi diri dari pengguna lain yang ingin dikenali sekaligus memilih informasi jejaring sosial yang dimiliki oleh pengguna lain tersebut, mengambil citra pengguna yang ingin dikenali.



Gambar 3.7 *Use Case* Diagram

3.2.2 *Sequence Diagram*

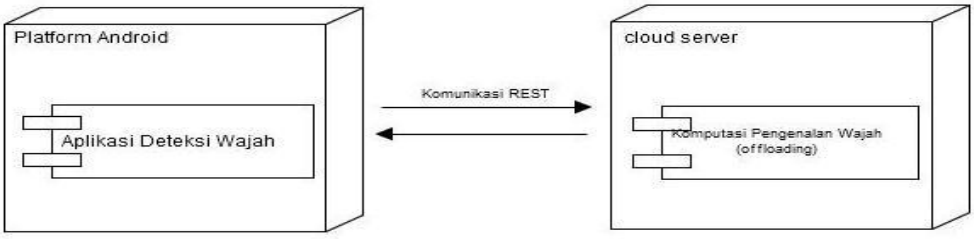
Sequence Diagram termasuk salah satu *interaction diagram* yang digunakan untuk menggambarkan perilaku dari objek-objek yang terlibat dalam sistem dan pesan-pesan antar objek yang berada dalam *use case* [6]. Dapat dilihat dalam Gambar 3.8, terdapat interaksi antara pengguna dengan sistem. Pada perangkat *mobile* terjadi proses pendeteksian wajah terlebih dahulu dengan menangkap citra wajah dari orang yang ingin dikenali. Data wajah kemudian dikirimkan melalui jaringan data ke *cloud.* Proses komputasi dilakukan di *cloud* yaitu pada bagian modul sistem pengenalan wajah dengan mencocok data wajah yang telah dikirim dengan *database* yang ada. Data hasil proses komputasi kemudian dikirimkan kembali ke perangkat *mobile* yang mana akan ditampilkan informasi mengenai orang yang ingin dikenali.



Gambar 3.8 *Sequence Diagram* Sistem

3.2.3 *Deployment Diagram*

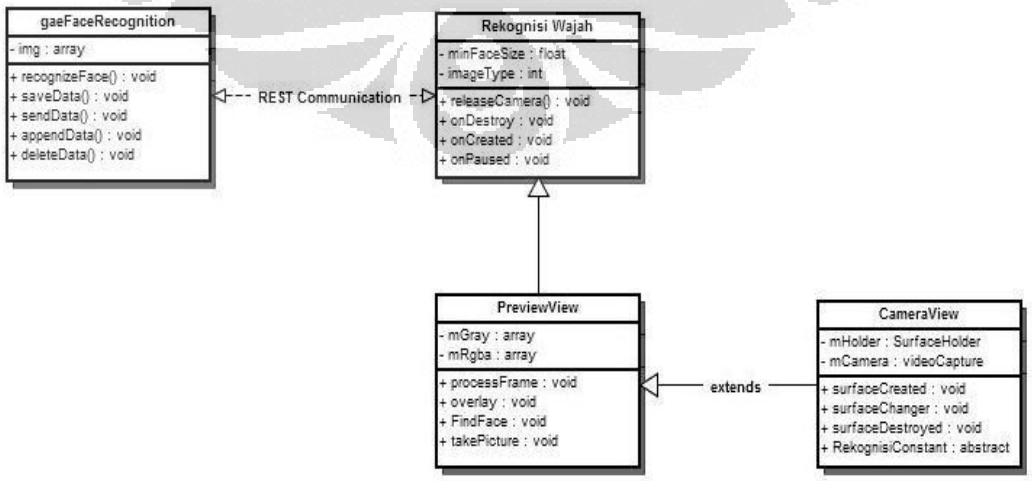
Deployment diagram digunakan untuk merepresentasikan rancangan tata letak fisik dari sistem yang menunjukkan perangkat keras dan perangkat lunak dalam sistem tersebut [6]. Dalam sistem pengenalan wajah berbasis teknologi *mobile cloud computing* ini melibatkan dua perangkat keras yang dihubungkan melalui jaringan paket data. Kedua perangkat keras tersebut adalah perangkat *mobile* dengan *platform* Android dan sitem komputasi awan yang berupa *server*. Pada perangkat *mobile* dibuat suatu aplikasi pengenalan wajah sedangkan dalam *cloud server* akan melakukan proses pengenalan citra wajah. *Deployment diagram* dari sistem ini ditunjukkan oleh Gambar 3.9.



Gambar 3.9 *Deployment* Diagram

3.2.4 *Class Diagram*

Class diagram digunakan untuk mendefinisikan hubungan objek-objek dalam sistem dan menggambarkan hubungan kelas-kelas yang terjadi dalam sistem [6]. Sistem ini secara umum menggunakan empat kelas yang mana terdapat dua bagian yaitu pada *cloud* dan perangkat *mobile*. Proses komputasi citra wajah dan pengenalan akan dilakukan oleh kelas *RekognisiWajah* yang mana akan mengirimkan kembali data hasil pemrosesan tersebut ke perangkat *mobile* melalui proses komunikasi REST. Kelas – kelas yang berada dalam perangkat *mobile* meliputi *RekognisiWajah*, *PreviewView* , dan *CameraView* akan melakukan proses deteksi wajah dimana kelas *RekognisiWajah* akan sangat tergantung pada tampilan atau akses kamera yang dilakukan oleh kelas *PreviewView* dan *CameraView*. Pengaksesan kamera pada perangkat *mobile* akan dilakukan oleh kelas abstrak *CameraView*. Penampilan deteksi wajah akan dilakukan oleh kelas *PreviewView*. Hubungan sistem dalam perangkat *mobile* ke *cloud server* juga diperlihatkan dalam Gambar 3.10 yang dihubungkan dengan komunikasi REST.



Gambar 3.10 Class Diagram Sistem

BAB 4
IMPLEMENTASI MODUL SISTEM DETEKSI WAJAH DAN KOMUNIKASI REST KE KOMPUTASI AWAN

4. Implementasi Sistem Pengenalan Wajah

Proses implementasi sistem pengenalan wajah pada bab ini akan dibagi menjadi dua bagian, yaitu perangkat *mobile* sebagai bagian antar muka (*front-end*) dan proses komunikasi dengan *cloud server* (*back-end*). Secara mendasar, bagian *front-end* akan menjalankan fungsi sebagai pemroses pendeteksian wajah secara *visual tracking* dan bagian *back-end* akan menjalankan fungsi lanjutan daripada bagian *backend* yaitu mengenali wajah seseorang dari suatu citra. Subbab berikut memperlihatkan beberapa implementasi sistem pengenalan wajah pada perangkat *mobile*.

4.1 Implementasi modul deteksi wajah pada perangkat *mobile*

Secara mendasar, proses implementasi modul deteksi wajah pada perangkat *mobile* akan dibahas menjadi dua bagian yaitu proses pemrograman dan hasil implementasi sistem deteksi wajah. Dalam proses implementasi sistem pada perangkat *mobile*, mula-mula dilakukan proses *visual tracking*. Proses *visual tracking* ini merupakan suatu proses untuk mengenali bahwa terdapat wajah seseorang. Jika wajah seseorang terdeteksi, maka perangkat *mobile* akan membuat suatu garis persegi untuk mendeteksi dimana posisi wajah orang yang ingin dikenali tersebut. Penjelasan lebih lanjut akan dijelaskan pada Subbab 4.1.1 dan Subbab 4.1.2.

4.1.1 Implementasi pemrograman pada perangkat *mobile*

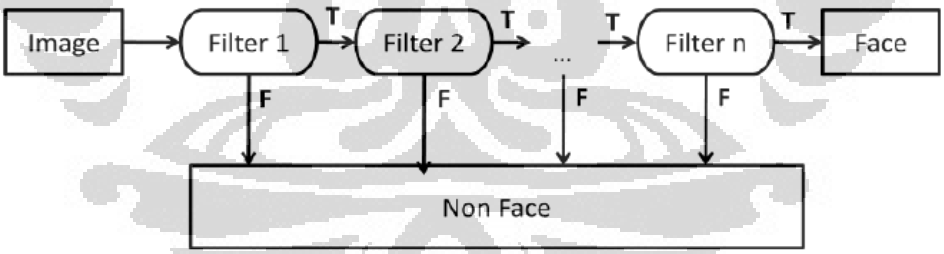
Proses pemrograman dalam sistem pendeteksian wajah di *Android client* akan diimplementasikan dengan menggunakan program *Eclipse*. Program ini merupakan suatu *Integrated Development Environment* (IDE) yang dapat juga membantu mengembangkan sistem deteksi wajah dalam bahasa pemrograman Java. Aplikasi *eclipse* juga dilengkapi dengan suatu *plugin Android Development Tools* (ADT) yang dapat diintegrasikan dengan aplikasi *Eclipse* sehingga

mempermudah pengembangan sistem dalam perangkat *mobile* Android. Dalam pengembangan modul di dalam IDE *Eclipse*, akan digunakan bantuan Android *Face Detector* API [19] untuk mendeteksi adanya wajah pada suatu citra. Kode yang digunakan untuk mencari wajah pada suatu citra seperti yang terlihat dalam Gambar 4.1.

```
FaceDetector.Face faces[] = new FaceDetector.Face[NUM_FACES];
FaceDetector arrayFaces = new FaceDetector(width, height,
NUM_FACES);
arrayFaces.findFaces(image, faces);
for(int i=0; i<faces.length && !stop; i++){
      face = faces[i];
      ....
}
```

Gambar 4.1 Kode untuk mencari wajah

Proses pendeteksian citra wajah pada perangkat *mobile* dengan metode alami android memiliki persamaan dengan metode yang ada dalam Viola-Jones [9] seperti terlihat dalam Gambar 4.2. Perbedaan metode alami android terletak pada tidak ada penggunaan *classifier* seperti pada penggunaan pustaka lain seperti JavaCV. Penggunaan *android Face Detector* API ini lebih sederhana dan hanya diperlukan metode untuk mencari lokasi wajah dalam suatu *bitmap*.



Gambar 4.2 Algoritma pada proses pendeteksian wajah [10]

Seperti yang dijelaskan pada [22], penggunaan android *Face Detector* API juga menunjukkan bahwa API ini lebih membutuhkan waktu proses yang lebih cepat daripada penggunaan pustaka JavaCV. Selain itu, hasil deteksi wajah juga menunjukkan hasil keakuratan yang hampir sama dengan JavaCV. Oleh karena itu, pengembangan sistem pada penelitian ini akan menggunakan Android *Face Detector* API. Terdapat tiga aktivitas terpenting untuk melakukan proses

pendeteksian wajah dan *visual tracking* wajah dalam proses implementasi sistem ini, yakni :

1. *PreviewViewActivity*

   PreviewViewActivity akan mengatur tampilan dan pemrosesan bitmap sehingga akan diketahui posisi wajah.

2. *ImageViewActivity*

   ImageViewActivity akan mengatur fungsi pencetakan wajah sehingga hasil dari proses pencetakan ini akan berupa citra wajah yang sudah di potong area wajahnya.

3. *FaceDetectActivity*

   FaceDetectActivity merupakan fungsi yang digunakan untuk mengatur tampilan dari menu utama dari sistem pendeteksian wajah termasuk didalamnya pengaturan mode pendeteksian wajah.

Proses deteksi wajah dalam sistem ini akan dilakukan secara aktual sehingga memerlukan komputasi yang cukup cepat. Dalam sistem ini, citra yang diambil merupakan hasil dari setiap *frame* yang didapat dari kamera perangkat *mobile*. Setiap *frame* itulah yang nantinya akan diproses lebih lanjut untuk mengetahui terdapat wajah atau tidak. Pemrosesan *frame* ini dapat dilakukan dengan menggunakan kelas *android.hardware.Camera.PreviewCallback* dengan menampung data tersebut ke dalam *buffer*. Potongan kode untuk dapat memproses data dalam setiap *frame* ini dapat dilakukan seperti terlihat dalam Gambar 4.3.

```
/* surfaceCreated */
      public void surfaceCreated(SurfaceHolder holder) {
            setKeepScreenOn(true);
            setupCamera();
/* onPreviewFrame */
      public void onPreviewFrame(byte[] _data, Camera _camera) {
            if(_data.length < bufflen_)
                  return;
            // run only one analysis thread at one time
            if(!isThreadWorking_){
                  isThreadWorking_ = true;
                  // copy only Y buffer
                  ByteBuffer bbuffer = ByteBuffer.wrap(_data);
                  bbuffer.get(grayBuff_, 0, bufflen_);
                  // make sure to wait for previous thread
completes
                  waitForFdetThreadComplete();
                  // start thread
```

```
                    detectThread_ = new FaceDetectThread(handler_);
                    detectThread_.setBuffer(grayBuff_);
                    detectThread_.start();
```

Gambar 4.3 Kode Pemrosesan Data *Frame* pada Kamera

Sebelum menggunakan android *Face Detector* API, data dari setiap *frame* yang telah didapatkan dalam Gambar 4.3 diubah ke dalam format *grayscale*. Pengubahan ke dalam format *grayscale* dilakukan karena data *buffer* hasil dari *preview data* hanya menghasilkan format YUV 4:2:0 yang merupakan format asli dari hasil video dari kamera perangkat *mobile* Android. Dalam Gambar 4.3 juga menjelaskan bahwa dari format *grayscale* tersebut akan dimasukkan ke dalam *graybuffer*. Data yang berada dalam *graybuffer* inilah yang akan digunakan android *Face Detector* API untuk mendeteksi ada tidaknya wajah dalam *frame* yang telah didapat.

Data dalam *graybuffer* ini juga menjadi data untuk mengisi format bitmap *red, green, blue* (RGB) ke dalam warna *grayscale* dengan mengisi tiap-tiap RGB dengan warna hitam putih. Bitmap ini nantinya digunakan untuk menandai bagian mana saja dalam citra yang memiliki area wajah. Kode untuk proses ini dapat terlihat dalam Gambar 4.5

```
private void gray8toRGB32(byte[] gray8, int width, int height,
int[] rgb_32s) {
      final int endPtr = width * height;
      int ptr = 0;
while (true) {
      if (ptr == endPtr)
      break;
      final int Y = gray8[ptr] & 0xff;
      rgb_32s[ptr] = 0xff000000 + (Y << 16) + (Y << 8) + Y;
      ptr++;
```

Gambar 4.4 Kode Program untuk Mengubah ke *Grayscale*

Setelah *bitmap* tersebut didapatkan, langkah selanjutnya adalah mengambil area wajah pada citra tersebut. Hal ini dapat dilakukan dengan cara memproses *bitmap* yang didapat dengan android *Face Detector API* seperti terlihat dalam Gambar 4.6 yang menjelaskan tentang proses deteksi wajah dan pencarian fitur dalam wajah.

```
for(int i=0;i<MAX_FACE; i++){
      FaceResult face = faces_[i]
      float eyedist = face.eyesDistance()
      if(eyedist==0.0f)
```

```
        continue
        PointF midEyes = new PointF()
        face.getMidPoint(midEyes)
```

Gambar 4.5 Kode untuk mencari fitur wajah

Proses selanjutnya adalah membuat area wajah yang terdeteksi tersebut bisa dilakukan aktifitas untuk proses pengenalan wajah selanjutnya yang akan terjadi di *cloud server*. Berikut ini merupakan potongan kode dari *method onTouchEvent* yang digunakan untuk melakukan proses pemotretan dan pemotongan area wajah sehingga area yang nantinya dihasilkan adalah hanya area wajah. Hal ini seperti terlihat dalam Gambar 4.7 dan 4.8.

```
/* CAUTION : touch point need to be same aspect to jpeg bitmap
size */
PointF lt = new PointF(midEyes.x*xRatio-
eyedist*1.5f,midEyes.y*yRatio-eyedist*1.5f);
Rect rect = new
Rect((int)(lt.x),(int)(lt.y),(int)(lt.x+eyedist*3.0f),(int)(lt.y+e
yedist*3.0f));
            if( rect.contains(touchPt.x,touchPt.y)){
            if(!isSDCardPresent_)
            Toast.makeText(context_,
R.string.SDCardNotPresentAlert, Toast.LENGTH_LONG).show();
            else{
            takingPicture_ = true;
            Toast.makeText(context_, R.string.CapturingAlert,
Toast.LENGTH_SHORT).show();
selFacePt_ = new PointF((float)touchPt.x/w,(float)touchPt.y/h);
                              }
                              break;
                          }
                                                    }
```

Gambar 4.6 Kode Program untuk Memperoleh Akses *onTouchEvent*

```
/* crop */
Bitmap facebmp =
Bitmap.createBitmap(bmp,rect.left,rect.top,rect.width(),rect.heigh
t());
FaceDetector.Face[] trackface = new FaceDetector.Face[1];
FaceDetector tracker = new FaceDetector(
facebmp.getWidth(),facebmp.getHeight(),1);
int found = tracker.findFaces(facebmp, trackface);
      if(found!=0){
      PointF ptTrack = new PointF();
      trackface[0].getMidPoint(ptTrack);
      ptTrack.x += (float)rect.left;
      ptTrack.y += (float)rect.top;
ptTrack.x *= xScale;
ptTrack.y *= yScale;
      float trkEyedist = trackface[0].eyesDistance()*xScale;
      faces_[i].setFace(ptTrack,trkEyedist);
      trackfound++;
```

Gambar 4.7 Kode Program untuk Memotong Area Wajah pada Citra

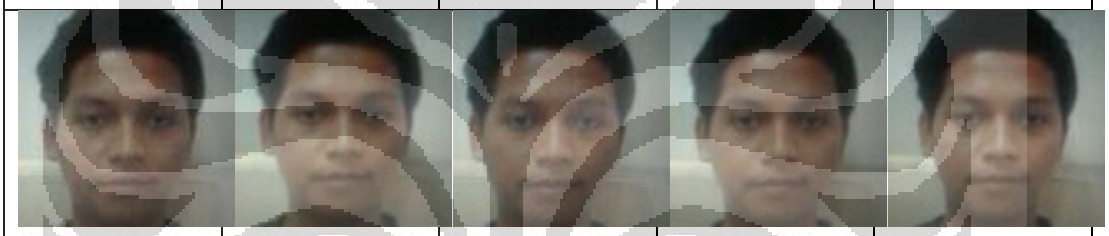
Gambar 4.8 merupakan kode yang digunakan untuk melakukan proses pemotongan area wajah. Kode ini akan dilakukan setelah method *onTouchEvent* dilakukan. Hal ini ditujukan agar proses komunikasi atau transfer data antara perangkat *mobile* dan *cloud server* tidak terlalu lama.

```
/* Save bitmap to file */
private Uri SaveBitmapToFile(Bitmap bmp) {
/* setup for storing file */
String filename = "HasilDetect";
ContentValues values = new ContentValues();
values.put(Media.DISPLAY_NAME, filename);
values.put(Media.TITLE, filename);
String absFilePath =
"/sdcard/DCIM/RekognisiWajah/"+filename+".jpg";
values.put(Media.DATA, absFilePath);
values.put(Media.MIME_TYPE, "image/jpeg");
Uri uri =
context_.getContentResolver().insert(MediaStore.Images.Media.EXTER
NAL_CONTENT_URI, values);
try {
                    /* save file */
      OutputStream outStream =
context_.getContentResolver().openOutputStream(uri);
      bmp.compress(Bitmap.CompressFormat.JPEG, 90, outStream);
      outStream.flush();
      outStream.close();
      Toast.makeText(context_,
context_.getString(R.string.SaveImageSuccessAlert)+absFilePath,
Toast.LENGTH_SHORT).show();
      return uri;
            } catch (IOException e) {
                  Toast.makeText(context_,
R.string.SaveImageFailureAlert, Toast.LENGTH_SHORT).show();
                  e.printStackTrace();
            }
            return null;
                                    }
```

Gambar 4 8 Kode Program untuk Menyimpan Wajah yang Telah Terdeteksi

Gambar 4.8 menjelaskan cara untuk menyimpan hasil dari proses deteksi wajah. Proses penyimpanan ke dalam format JPEG ini diperlukan untuk digunakan dalam proses selanjutnya yaitu pengiriman file ke *cloud server*. Gambaran mengenai area wajah yang telah dipotong dalam modul deteksi wajah seperti terlihat pada Gambar 4.9.

Gambar 4.9 Potongan Area Wajah Dalam Modul Deteksi Wajah

Kualitas Kompresi JPEG				
				
10	20	30	40	50
				
60	70	80	90	100

Gambar 4. 10 Kualitas Kompresi Area Wajah

Gambar 4.10 merupakan gambaran kualitas kompresi JPEG yang dilakukan pada hasil deteksi wajah dengan kualitas kompresi 10 hingga 100. Kompresi ini dilakukan sebagai suatu parameter untuk mempercepat proses *offloading* dengan mengurangi ukuran citra yang akan dikirim ke *cloud server*.Potongan kode Gambar 4.11 akan melakukan proses *visual tracking* pada kamera android sehingga diperoleh proses pendeteksian wajah. Proses yang dilakukan adalah menampilkan hasil wajah yang terdeteksi dalam *android canvas*. *Canvas* ini merupakan *low-level API* yang digunakan untuk melakukan operasi pemrosesan citra wajah yang terdapat dalam perangkat *mobile* android.

```
protected void onDraw(Canvas canvas) {
            float xRatio = (float)w / previewWidth_;
            float yRatio = (float)h / previewHeight_;
            for(int i=0; i<MAX_FACE; i++){
            FaceResult face = faces_[i];
            float eyedist = face.eyesDistance()*xRatio;
                        if(eyedist==0.0f)
```

```
                        continue;
                    PointF midEyes = new PointF();
                    face.getMidPoint(midEyes);
                    if(appMode_==0){
                        PointF lt = new
PointF(midEyes.x*xRatio-
eyedist*MAG_EYEDIST_FACE*0.5f,midEyes.y*yRatio-
eyedist*MAG_EYEDIST_FACE*0.5f);

      canvas.drawRect((int)(lt.x),(int)(lt.y),(int)(lt.x+eyedist*M
AG_EYEDIST_FACE),(int)(lt.y+eyedist*MAG_EYEDIST_FACE), paint_);
                    }
                    else if(overlayBitmap_!=null){
                        PointF lt = new
PointF(midEyes.x*xRatio-
eyedist*MAG_EYEDIST_FACE*0.5f,midEyes.y*yRatio-
eyedist*MAG_EYEDIST_FACE*0.5f);
                    canvas.drawBitmap(overlayBitmap_, null ,
new Rect((int)lt.x,
(int)lt.y,(int)(lt.x+eyedist*MAG_EYEDIST_FACE),(int)(lt.y+eyedist*
MAG_EYEDIST_FACE)),paint_);
```

Gambar 4.11 Kode Program untuk Menampillkan Wajah pada Kamera Android

4.1.2 Hasil Implementasi Modul Pendeteksi Wajah Pada Perangkat *Mobile*

Implementasi modul yang terdapat dalam perangkat *mobile* akan menghasilkan suatu program *client* Android seperti tampak dalam Gambar 4.12. Pada bagian awal, pengguna akan dihadapkan terhadap 3 pilihan, yaitu pilihan rekognisi wajah, pilihan *set database training*, dan pilihan keluar. Pilihan rekognisi wajah digunakan untuk melakukan proses analisa wajah secara visual (*face detection*). Bagian ini juga dilengkapi dengan penggunaan dua konfigurasi yaitu integrasi dengan *facebook* dan pilihan normal. Jika integrasi dengan *facebook* dilakukan, maka proses rekognisi wajah akan menggunakan *database* wajah dari *facebook*. *Database* wajah *facebook* ini didapatkan melalui hasil pengumpulan citra wajah (*crawling*) yang sudah diberi label dengan menggunakan bantuan *Face.com* API [20]. Proses penggunaan *database* wajah ini juga menggunakan *Facebook* API sebagai bentuk metode pengaksesan terhadap fasilitas yang telah disediakan oleh *facebook.com* seperti foto teman, data teman, status teman, dan lain-lain.

Jika mode normal dipilih, proses rekognisi wajah yang dilakukan akan menggunakan *database* wajah dari data latih yang dibuat secara manual. Data ini berisikan 10 kelas wajah dengan setiap kelas memiliki 10 citra wajah. Kelas wajah ini akan mempresentasikan kumpulan wajah orang yang akan dilakukan

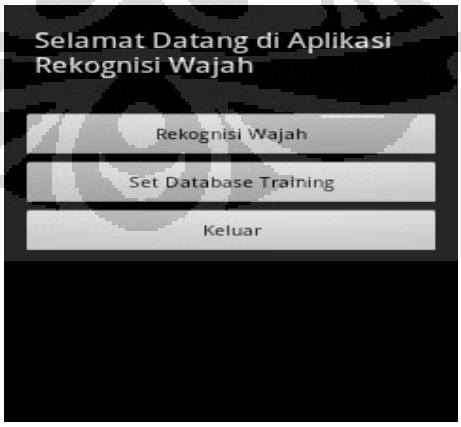
proses pengenalan wajah. Pada pilihan *set database training* digunakan untuk menambahkan data latih wajah ke dalam *database* pelatihan yang dapat digunakan untuk menambah keakuratan proses pengenalan wajah.

Sebelum memulai pilihan *set database training*, diperlukan suatu langkah untuk melakukan proses *login* atau *register* terlebih dahulu. Proses *login* akan dibagi menjadi dua bagian yang digunakan untuk proses pelatihan citra wajah yakni.

1. *Login* berdasarkan pada perangkat *mobile*

   Proses *login* ini akan memberi akses kepada pengguna untuk melakukan data *training* individu. Data *training* individu kemudian diproses lebih lanjut ke dalam *cloud server* untuk menambah keberhasilan dalam proses pengenalan wajah.
2. *Login* yang melibatkan akses jejaring sosial.

   Proses login ini akan memberikan akses kepada pengguna untuk dapat mengoleksi data wajah pada jejaring sosial yang digunakan pula dalam proses pengenalan wajah. Proses ini akan mengumpulkan dan mendeteksi wajah yang telah di lakukan proses pemberian label wajah sehingga proses pengoleksian data wajah untuk dimasukkan ke dalam *database* wajah lebih besar dan dapat menghasilkan proses keakuratan pendeteksian yang lebih tinggi.

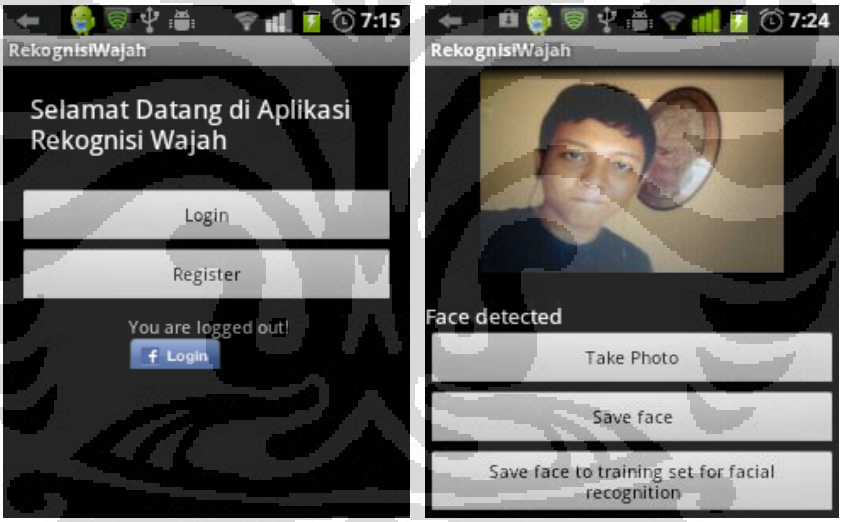


Gambar 4.12 Tampilan Menu Awal

Gambaran mengenai analisa wajah secara visual atau proses *visual tracking* seperti terlihat pada Gambar 4.13. Wajah yang telah terdeteksi akan

diberi suatu label kotak. Setelah label terlihat pada layar perangkat *mobile*, langkah selanjutnya adalah proses pengenalan wajah secara *offloading* akan dilakukan melalui komunikasi REST.

Gambar 4.13 Analisa Wajah Secara Aktual Pada Perangkat Mobile



Gambar 4.14 Antarmuka Program Deteksi Wajah Pada Perangkat *Mobile*

Pada Gambar 4.14, setelah proses login berhasil, pengguna akan dihadapkan pada pemilihan menu. Terdapat tiga pilihan, yaitu *Take Photo* yang digunakan untuk mengambil citra wajah, *Save face* digunakan untuk mengecek apakah citra wajah berhasil terdeteksi, dan pilihan *Save face to training* merupakan pilihan untuk menyimpan citra wajah tersebut kedalam *database* di *cloud server*.

4.2 Implementasi komunikasi REST pada perangkat *mobile*

Dalam implementasi sistem pengenalan wajah, proses selanjutnya yaitu melakukan komunikasi dengan *cloud server*. Proses ini merupakan suatu proses mengirimkan hasil deteksi wajah yang telah dilakukan pada modul deteksi wajah pada perangkat *mobile*. Komunikasi yang terjadi dengan *cloud server* akan dilakukan dengan metode *representational state transfer* (REST). Metode ini juga merupakan bentuk implementasi dari proses *offloading*.

Komunikasi REST ini juga akan mengirimkan beberapa item selain citra hasil deteksi wajah. Item-item tersebut merupakan variabel tambahan yang digunakan untuk memberi deskripsi citra wajah yang akan dikirim, misalnya saja memberi label pada citra wajah tersebut. Pengiriman data dalam komunikasi REST ini akan dilakukan secara *multipart*. Proses *multipart* ini merupakan proses pengunggahan data ke *server* yang dilakukan dalam sekali transfer data. Proses ini menggunakan pustaka tambahan yaitu *httpmime-4.1.2* [23]. Pustaka ini akan mengatur bagaimana proses pengkodean beberapa variabel tambahan dan citra wajah yang akan dikirim ke *cloud server*. Hasil pengkodean yang dilakukan pada pustaka tersebut akan berbentuk kode *bytestream* yang siap untuk dikirim melalui jaringan data. Berikut ini beberapa potongan kode untuk melakukan proses unggah secara *multipart* pada perangkat *mobile* seperti terlihat pada Gambar 4.15.

```
private StringBuilder sendPhoto(File picture, HashMap<String,
String> parts, String uri) {
      MultipartEntity entity = new MultipartEntity();
      StringBuilder builder = null;
      HttpPost httpPost = new HttpPost(uri);
      HttpClient httpClient = new DefaultHttpClient();
            try {
      // Get all the properties
      for (Entry<String, String> entry : parts.entrySet()) {
      FormBodyPart formBodyPart = new FormBodyPart(entry.getKey(),
new StringBody(entry.getValue()));
      entity.addPart(formBodyPart);
                  }
      entity.addPart("image", new FileBody(picture));
      httpPost.setEntity(entity);
      Log.i(this.TAG, "Start sendPhoto( " + uri + " Length of file
in bytes: " + picture.length());
      HttpResponse response = httpClient.execute(httpPost);
      Log.i(this.TAG, "End sendPhoto( " + uri + " Length of file
in bytes: " + picture.length());
      reader = new BufferedReader(
 new InputStreamReader(response.getEntity().getContent(), "UTF-
```

```
8"));
      builder = new StringBuilder();
      for (String line = null; (line = reader.readLine()) !=
null;) {
      builder.append(line).append("\n");
                  }

            } catch (UnsupportedEncodingException e) {
                  Log.e(this.TAG, "detect()
UnsupportedEncodingException " + e.getMessage());
            } catch (ClientProtocolException e) {
                  Log.e(this.TAG, "detect()
ClientProtocolException" + e.getMessage());
            } catch (IOException e) {
                  Log.e(this.TAG, "detect() IOException" +
e.getMessage());
            }
            return builder;
      }
```

Gambar 4.15 Kode Unggah Citra Wajah Secara *Multipart*

Setelah proses pengenalan wajah atau *offloading* di *cloud server* selesai, maka proses yang dilakukan pada perangkat *mobile* adalah menerima hasil pengenalan wajah tersebut. Hasil dari proses *offloading* tersebut berupa data JSON. Data JSON ini akan memiliki deskripsi tentang identitas dari citra wajah yang diproses dan beberapa variabel lain seperti tingkat kepercayaan sistem, tingkat *error* yang terjadi. Untuk mendapatkan informasi yang diinginkan, data JSON ini kemudian dilakukan pendekodean untuk diambil informasi yang berguna saja untuk ditampilkan pada perangkat *mobile*. Bentuk kode untuk melakukan pendekodean seperti terlihat pada Gambar 4.16.

```
// Parse json response for recognise face result
try {
   if (!new
JSONObject(builder.toString()).getJSONArray("photos").getJSONObjec
t(0).getJSONArray("tags").isNull(0)) {
object = new
JSONObject(builder.toString()).getJSONArray("photos").getJSONObjec
t(0)
      .getJSONArray("tags").getJSONObject(0).getJSONArray("uids").
toString();
      // if face detected but not recognised
      if (object.equals("[]")) {
            object = "Unknown";
      } else { String object1 = new
JSONObject(builder.toString()).getJSONArray("photos").getJSONObjec
t(0)
name = new JSONArray(object).getJSONObject(0).getString("uid");
confidence = new
JSONArray(object).getJSONObject(0).getString("confidence");
```

```
label = new
JSONArray(object1).getJSONObject(0).getString("label");
                                Log.i("Object 1", object1);
```

Gambar 4.16 Kode Untuk Pendekodean JSON

Setelah informasi mengenai citra wajah dalam data JSON didapatkan, proses selanjutnya adalah menampilkan hasil informasi tersebut pada perangkat *mobile*. Hal ini dapat dilakukan dengan cara menerapkan metode realita tertambah yang akan dibahas lebih lanjut pada [5]. Metode ini akan memiliki bentuk tampilan informasi identitas yang lebih menarik dan informatif.

BAB 5
PENGUJIAN DAN ANALISA MODUL DETEKSI WAJAH DAN KOMUNIKASI REST KE KOMPUTASI AWAN

5. Pengujian Modul Deteksi Wajah Pada Aplikasi *Mobile*

Proses pengujian modul deteksi wajah dalam perangkat *mobile* akan dilakukan dengan dua tahap, yakni proses saat mendeteksi citra wajah secara aktual dan proses saat melakukan proses *offloading* untuk mengenali identitas wajah dalam citra. Pengujian modul deteksi wajah juga dilakukan ke dalam beberapa tipe perangkat *mobile* Android. Hal ini dimaksudkan untuk mengetahui efisiensi kerja dari modul aplikasi deteksi wajah pada perangkat *mobile.* Pengenalan wajah dalam *cloud server* ini merupakan suatu proses *offloading aplication* dari keseluruhan sistem pengenalan wajah. Penjelasan mengenai fungsionalitas aplikasi ini akan dibahas dalam Subbab 5.1 dan 5.2.

5.1 Pengujian Modul Untuk Pendeteksian Wajah

Modul pendeteksian wajah pada perangkat *mobile* merupakan tahap pertama dalam fungsionalitas utama aplikasi ini. Pengujian teknis yang dilakukan dalam modul ini terdiri dari pengujian terhadap variasi resolusi kamera. Pengujian terhadap variasi resolusi dilakukan dengan cara memvariasikan beberapa resolusi kamera mulai dari resolusi rendah hingga resolusi yang lebih tinggi. Hasil variasi resolusi kamera inilah yang akan menjadi bahan untuk mengetahui waktu yang diperlukan untuk melakukan proses deteksi wajah pada suatu citra. Pengujian yang dilakukan dalam modul pendeteksian wajah akan dilakukan sebanyak 10 kali untuk masing-masing kondisi resolusi kamera.

Hasil pengujian sistem deteksi wajah untuk perangkat *mobile* seperti terlihat pada Tabel 5.1. Perangkat *mobile* yang digunakan yaitu menggunakan perangkat *mobile* kelas rendah (Galaxy GT S5570) dengan spesifikasi CPU 600 MHz, 384 Mb RAM, dan kualitas kamera 3,15 Megapiksel. Dalam tabel tersebut didapatkan bahwa waktu kerja dari sistem deteksi wajah dalam perangkat *mobile*

mengalami perbedaan di setiap variasi resolusi kamera. Hal ini disebabkan adanya jumlah bidang kerja yang semakin luas yang ditandai dengan semakin besarnya resolusi. Bila menggunakan resolusi 240x160 piksel, sistem dapat mengenali ada tidaknya wajah dengan rata-rata waktu 0,46 detik. Ketika resolusi dinaikkan menjadi 320x240 piksel, sistem deteksi wajah dalam perangkat *mobile* membutuhkan waktu kerja yang lebih lama. Penggunaan resolusi yang semakin besar akan mempengaruhi lama tidaknya waktu pemrosesan dari sistem deteksi wajah tersebut.

Tabel 5.1 Waktu Deteksi Wajah Pada Resolusi Kamera yang Bervariasi dalam Perangkat *Mobile* Kelas Rendah

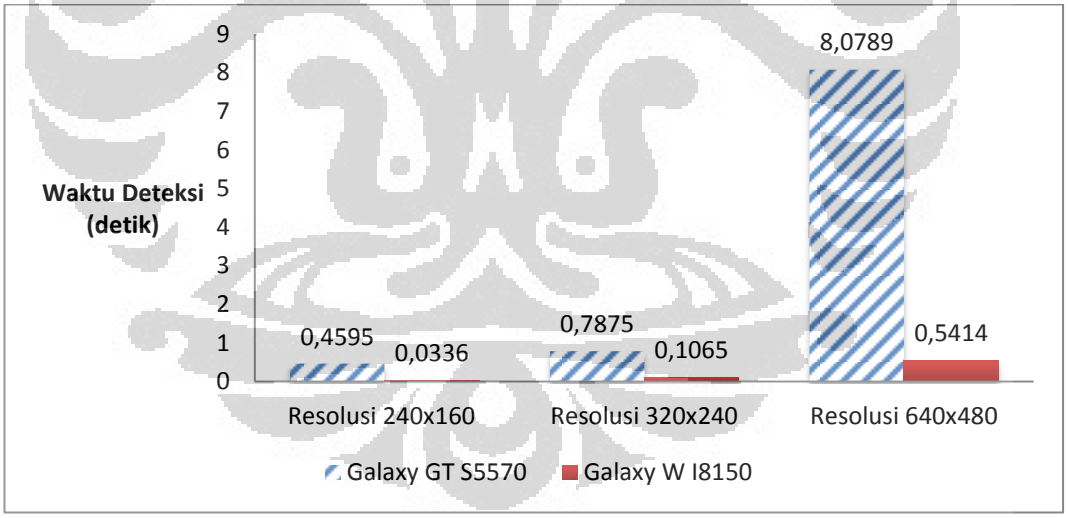
Percobaan	Waktu (detik)		
	Resolusi (240x160)	Resolusi (320x240)	Resolusi (640x480)
1	0,514	0,929	10,198
2	0,436	0,901	8,726
3	0,501	1,035	7,739
4	0,286	0,329	7,513
5	0,580	0,314	8,306
6	0,493	0,781	7,851
7	0,444	0,909	7,827
8	0,520	0,909	7,469
9	0,404	0,905	7,435
10	0,417	0,863	7,725
Rata-rata	**0,4595**	**0,7875**	**8,0789**

Perbedaan kerja dalam proses deteksi wajah pada perangkat *mobile* juga terlihat pada waktu menggunakan perangkat *mobile* yang berbeda. Seperti yang terlihat pada Tabel 5.2, pengujian terhadap resolusi kamera juga diimplementasikan pada perangkat *mobile* kelas menengah (Galaxy W I8150) yang memiliki spesifikasi CPU 1.4 GHz, 512 Mb RAM, dan kualitas kamera 5 Megapiksel. Dalam tabel tersebut, kecepatan prosesor akan mempengaruhi kerja dari sistem deteksi wajah pada perangkat *mobile*. Perbedaan ini terlihat dengan jelas bahwa rata-rata proses deteksi wajah berkisar pada perangkat *mobile* kelas menengah tidak melebihi 1 detik dalam setiap variasi resolusi kamera. Hal ini

seperti terlihat pada Gambar 5.1 yang menggambarkan perbandingan variasi resolusi kamera terhadap waktu deteksi wajah.

Tabel 5.2 Waktu Deteksi Wajah pada Resolusi Kamera yang Bervariasi dalam Perangkat *Mobile* Kelas Menengah

Percobaan	Waktu (detik)		
	Resolusi 240x160	Resolusi 320x240	Resolusi 640x480
1	0,012	0,099	0,753
2	0,010	0,101	0,542
3	0,012	0,096	0,479
4	0,042	0,101	0,538
5	0,061	0,125	0,503
6	0,039	0,129	0,532
7	0,039	0,094	0,530
8	0,042	0,096	0,478
9	0,039	0,108	0,525
10	0,040	0,116	0,534
Rata-rata	**0,0336**	**0,1065**	**0,5414**



Gambar 5.1 Sistem Deteksi Wajah Dengan Variasi Resolusi Kamera

5.2 Pengujian Modul Pada Proses Offloading

Modul dalam proses *offloading* merupakan tahap untuk mengirimkan hasil deteksi wajah ke komputasi awan. Proses *offloading* merupakan salah satu bagian sistem pengenalan wajah untuk mengenali identitas diri dari citra hasil deteksi

wajah. Pengujian ini dimaksudkan untuk mengetahui seberapa cepat waktu dari proses *offloading* ini. Hasil pengujian modul pengenalan wajah akan terlihat seperti dalam Subbab 5.2.1 dan Subbab 5.2.2.

5.2.1 Modul Proses *Offloading* Terhadap Variasi Dimensi Citra

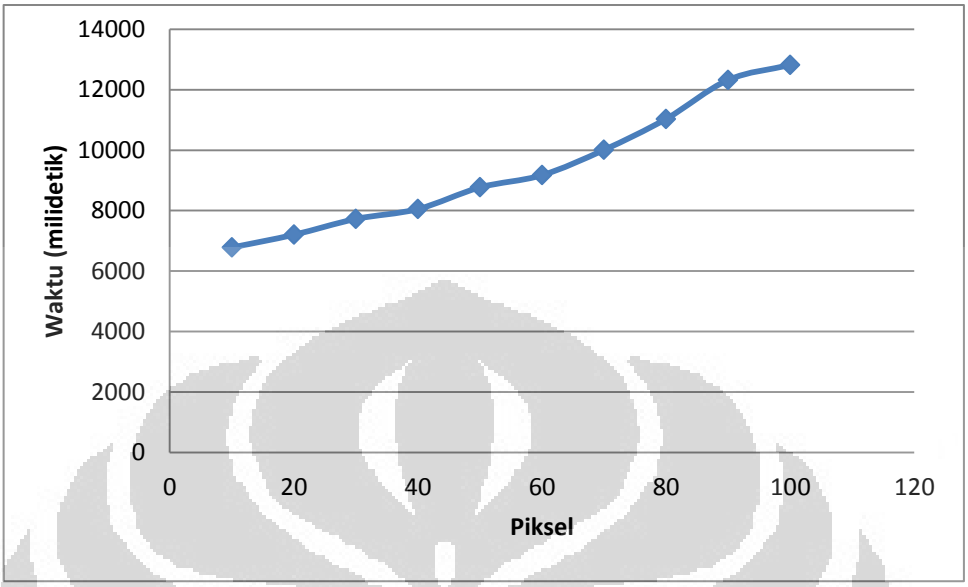
Bagian ini menjelaskan tentang pengujian proses *offloading* yang divariasikan dengan dimensi citra. Dimensi citra ini merupakan suatu parameter yang ditandai dengan besaran piksel. Pengujian yang dilakukan dengan parameter ini, divariasikan mulai dari 10x10 piksel hingga 100x100 piksel sehingga hasil modul deteksi wajah pada perangkat *mobile* akan disesuaikan sesuai variasi dimensi citra. Hal ini dimaksudkan untuk mencari efektifitas waktu dari proses *offloading* yang dilakukan. Hasil pengujian ini seperti yang terlihat pada Tabel 5.5, yang menjelaskan tentang rata-rata waktu proses *offloading* yang divariasikan dengan dimensi citra.

Tabel 5.3 Pengujian Variasi Dimensi Citra Terhadap Waktu *Offloading*

Piksel	Waktu (milidetik)	Dikenali (Ya/Tidak)
10x10	6780	Tidak
20x10	7202	Tidak
30x30	7725	Tidak
40x40	8052	Ya
50x50	8769	Ya
60x60	9173	Ya
70x70	10010	Ya
80x80	11033	Ya
90x90	12321	Ya
100x100	12819	Ya

Dalam Tabel 5.5, waktu proses *offloading* merupakan hasil rata-rata waktu dari 10 kali pengujian setiap variasi piksel. Seperti terlihat dalam tabel, proses *offloading* akan membutuhkan waktu yang lama seiring dengan semakin besarnya dimensi citra yang akan diproses oleh *cloud server*. Selain itu, besar dimensi citra juga mempengaruhi besar data wajah yang akan dikirim ke jaringan data sehingga berdampak pada lama tidaknya waktu transmisi data ke *cloud server*. Semakin besar ukuran piksel, ukuran data wajah yang dikirim juga semakin besar yang

akan berakibat pada semakin lama proses *offloading* tersebut. Hubungan ukuran piksel terhadap waktu proses *offloading* seperti terlihat pada Gambar 5.3.



Gambar 5.2 Hubungan Ukuran Dimensi Citra dengan Waktu *Offloading*

Begitu pula dengan ukuran dimensi yang terlalu kecil juga akan mempengaruhi hasil pengenalan wajah dari *cloud server*. Ukuran piksel yang terlalu kecil membuat data wajah yang dikirim menjadi tidak dikenali sistem. Sesuai dengan Tabel 5.5, ukuran piksel yang terbaik yang didapatkan adalah berukuran 40x40 piksel yang menunjukkan waktu proses *offloading* untuk mengenali wajah adalah sebesar 8 detik.

5.2.2 Modul Proses *Offloading* Terhadap Kualitas Kompresi Citra

Pengujian pada bagian ini akan membahas mengenai pengaruh kualitas kompresi data yang dilakukan pada hasil deteksi area wajah. Pengujian ini merupakan lanjutan pada Subbab 5.2.2 yang juga membahas mengenai efisiensi waktu dalam proses *offloading*. Seperti yang telah dijelaskan pada Subbab sebelumnya, waktu proses *offloading* terbaik diperoleh dengan menggunakan parameter ukuran dimensi citra sebesar 40x40 piksel dengan waktu 8 detik. Hal ini dapat dilakukan suatu peningkatan efisiensi waktu dengan cara mengurangi ukuran citra wajah yang dikirim ke *cloud server*, salah satunya dengan melakukan kompresi citra. Kompresi citra diharapkan dapat mempercepat proses *offloading*

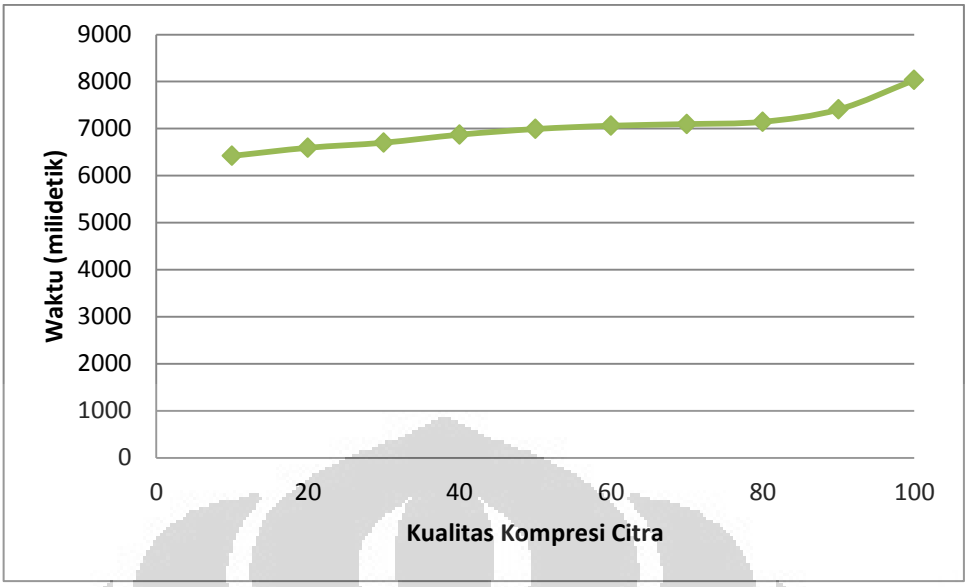
yang dilakukan. Oleh karena itu, Subbab ini akan membahas pengaruh kompresi citra terhadap waktu proses *offloading*.

Seperti yang telah dijelaskan pada Bab 3, hasil deteksi wajah dalam perangkat *mobile* akan memiliki format data JPEG. Citra wajah dalam format JPEG ini yang akan dikirimkan ke *cloud server* untuk dikenali identitasnya. Oleh karena itu, pengujian dilakukan dengan memvariasikan tingkat kompresi JPEG dengan kualitas 10 sampai 100. Pengujian kualitas kompresi citra ini seperti terlihat pada Tabel 5.6

Tabel 5.4 Pengujian Kualitas Kompresi Citra Dengan Waktu Proses *Offloading*

Kualitas Kompresi	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Waktu Proses *Offloading* (milidetik)	6925	5624	6531	6590	6090	6686	8422	7250	8242	8387
	6910	7297	6369	7076	7443	7592	7831	7214	7963	8217
	6413	6849	7546	6756	7556	7486	6073	7918	9194	7359
	7387	5928	7630	6748	7246	6018	7573	8602	7916	8443
	6472	6868	7263	6276	7124	6905	6585	7920	7527	8116
	6015	6701	6555	7536	6343	7248	7022	7717	8422	8720
	6224	7246	7032	7814	6819	7883	6955	7475	8909	7926
	6815	6855	6955	7434	6853	7109	6819	6526	7394	7753
	6686	7580	6437	6736	7442	7109	7345	7139	7783	7812
	6078	6063	6392	6946	7712	6926	6831	6308	6980	7792
Rata-rata (milidetik)	**6592**	**6701**	**6871**	**6991**	**7062**	**7096**	**7145**	**7406**	**8033**	**8052**
Dikenali	Tidak	Tidak	Tidak	Tidak	Ya	Ya	Ya	Ya	Ya	Ya

Grafik mengenai hubungan kualitas kompresi ini seperti terlihat pada Gambar 5.4. Seperti yang terlihat pada Tabel 5.6, kualitas kompresi citra akan mempengaruhi proses *offloading* dan hasil pengenalan wajah. Dalam data percobaan diperoleh tingkat kompresi yang terbaik adalah menggunakan kualitas kompresi 50. Hal ini ditunjukkan pula dengan waktu proses *offloading* sebesar 7 detik. Waktu proses *offloading* ini mengalami peningkatan dibandingkan citra hasil deteksi wajah tanpa kompresi. Dengan demikian, penggunaan kompresi citra wajah dapat meningkatkan efisiensi waktu dalam proses *offloading* dengan batas ambang kualitas kompresi sebesar 50.



Gambar 5.3 Hubungan Kualitas Kompresi Citra Terhadap Waktu *Offloading*

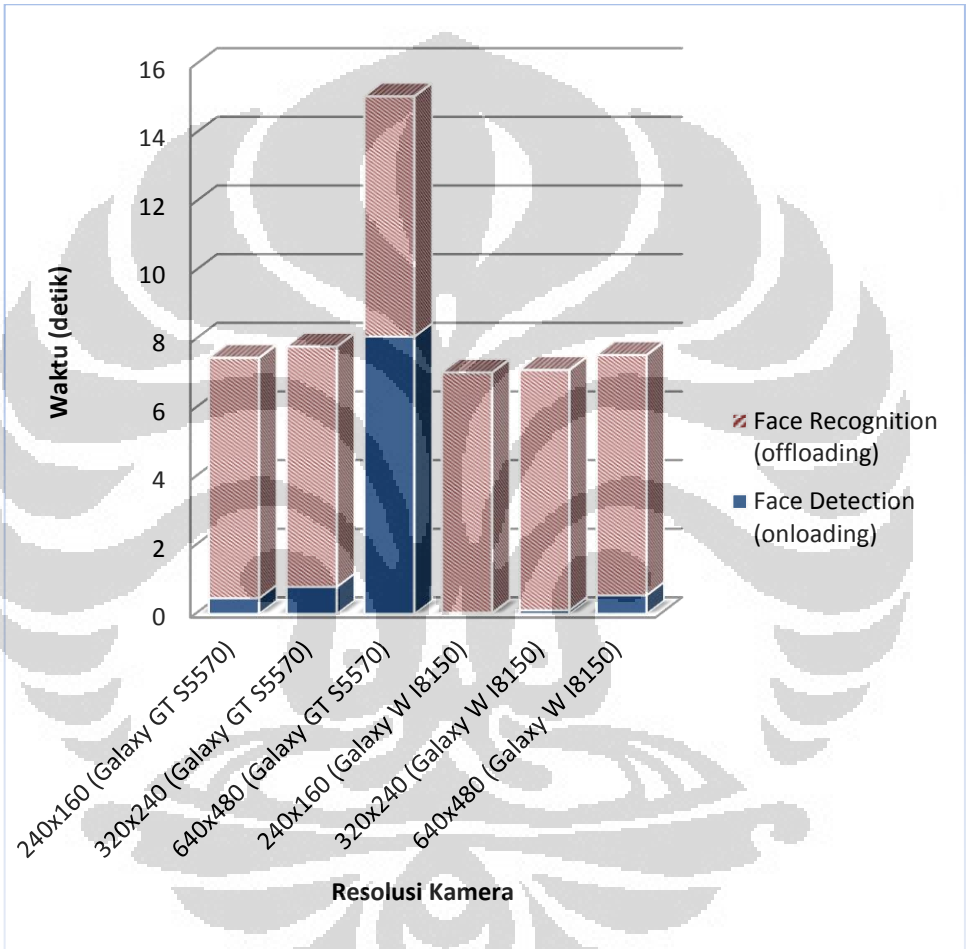
5.3 Pengujian Seluruh Modul Pengenalan Wajah Pada Perangkat *Mobile*

Pada bagian ini, dijelaskan mengenai pengujian dari keseluruhan modul yang telah diimplementasikan pada perangkat *mobile*. Pengujian dilakukan dengan menggabungkan hasil uji yang telah dibahas dalam Subbab sebelumnya, yaitu Subbab 5.1 dan Subbab 5.2. Parameter yang akan dibahas pada bagian ini adalah mengenai total proses komputasi yang telah dilakukan baik pada modul pendeteksian wajah maupun pada modul pengenalan wajah yang dilakukan melalui proses *offloading* ke *cloud server*.

Proses komputasi yang terjadi dalam sistem pengenalan wajah pada perangkat *mobile* merupakan hal yang penting. Hal ini ditujukan untuk mengetahui seberapa besar efektifitas komputasi pada perangkat *mobile* terutama ketika diintegrasikan dengan teknologi komputasi awan. Waktu proses *offloading* diperoleh dari penggunaan parameter terbaik yang didapat pada pengujian Subbab 5.2.1 dan Subbab 5.2.2. Pengujian ini akan menggunakan dimensi citra 40x40 piksel dan kualitas kompresi sebesar 50 yang merupakan hasil dari pengujian di Subbab sebelumnya. Berikut ini merupakan hasil keseluruhan dari proses komputasi sistem pengenalan wajah pada perangkat *mobile* seperti terlihat pada Tabel 5.5.

Tabel 5.5 Deskripsi Waktu Komputasi Sistem Pengenalan Wajah pada Perangkat *Mobile*

Tipe Perangkat Mobile	GTS5570			W I8150		
Resolusi	240x160	320x240	640x480	240x160	320x240	640x480
Pendeteksian wajah (*onloading*)	0,4595 s	0,7875 s	8,0789 s	0,0336 s	0,1065 s	0,5414 s
Proses Pengenalan Wajah (*offloading*)	7 s	7 s	7 s	7 s	7 s	7 s
Total Waktu Komputasi	**7,4595 s**	**7,7875 s**	**15,0789 s**	**7,0336 s**	**7,1065 s**	**7,5414 s**



Gambar 5.4 Perbandingan Komputasi Pengenalan Wajah Secara Menyeluruh

Pada Gambar 5.4 menjelaskan bahwa komputasi untuk melakukan proses pendeteksian wajah (*onloading*) memerlukan waktu yang lebih cepat pada semua tipe perangkat *mobile* yang diujikan. Hal ini berbeda dengan proses *offloading* yang memerlukan waktu komputasi yang lebih lama. Hal ini disebabkan proses *offloading* ke *cloud server* memerlukan komputasi yang lebih berat dan waktu pengiriman hasil pengenalan wajah juga sangat tergantung dari konektivitas

jaringan data. Namun, secara keseluruhan proses pengenalan wajah yang diimplementasikan pada perangkat *mobile* menunjukkan hasil yang tidak begitu buruk. Proses *visual tracking* atau proses deteksi wajah yang berlangsung juga cukup menunjukkan sistem yang bekerja secara aktual (*real-time*). Sementara, proses *offloading* masih terlalu lama untuk menjadi sistem yang aktual dan sangat tergantung dari konektivitas jaringan data.

BAB 6
PENUTUP

6.1 Kesimpulan

Berikut adalah kesimpulan dari penulisan skripsi ini :

1. Implementasi sistem pengenalan wajah pada perangkat *mobile* terdiri dari dua modul yaitu modul pendeteksian wajah secara *onloading* dan modul pengenalan wajah secara *offloading* yang dihubungkan dengan komunikasi REST.
2. Total waktu pemrosesan sistem pengenalan wajah yaitu sebesar 7,45 detik (GT S5570) dan 7,03 detik (W I8150) dengan resolusi kamera 240x160 piksel.
3. Dimensi citra akan mempengaruhi waktu proses *offloading* dengan semakin besar dimensi maka semakin lama waktu proses *offloading*. Begitu juga dengan kualitas kompresi yang akan mempercepat waktu proses *offloading* karena ukuran data yang dikirim ke *cloud server* semakin kecil.
4. Waktu proses *offloading* tercepat diperoleh pada kondisi penggunaan dimensi citra 40x40 piksel dan dengan tingkat kualitas kompresi sebesar 50.

6.2 Pengembangan ke depan

Sistem pengenalan wajah yang telah diterapkan dengan menggunakan teknologi komputasi awan (proses *offloading*) yaitu menggunakan komunikasi REST ke *cloud server* cukup memberikan hasil yang memuaskan dengan hasil yang cukup akurat. Namun, sistem secara keseluruhan belum merepresentasikan sistem yang *real time*. Untuk pengembangan ke depan, penerapan teknologi komputasi awan bisa menjadi alternatif untuk menghemat komputasi dalam perangkat *mobile* seiring dengan perkembangannya yang cukup luas. Hal ini bisa menjadi suatu panduan di masa depan bahwa sistem ini seharusnya mampu menjadi sistem pengenalan wajah yang *real time* seiring perkembangan jaringan komunikasi yang semakin menyediakan akses lebih cepat dan teknologi komputasi awan yang menjanjikan.

DAFTAR REFERENSI

[1] A.P. Miettinen and J.K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud 2nd USENIX Workshop on Hot Topics in Cloud Computing*, 2010.

[2] A.Gember and A. Akella, "*Mobile device offloading using enterprise network and cloud resources*," Tech. Rep., University of Wisconsin Madison, 2010.

[3] K.Kumar and Y.H. Lu, "*Cloud computing for mobile users: Can offloading computation save energy?*," *Computer*, vol. 43, no. 4, 2010.

[4] Ridho, NM. (2012). *Implementasi Sistem Pengenalan Wajah Sebagai Penghubung Jejaring Sosial : Layanan Komputasi Awan untuk Fungsi Pengenalan Wajah dengan Google App Engine berbasis Python dan API Face.com*. Depok : Universitas Indonesia.

[5] Budiyatno, Slamet. (2012). *Implementasi Sistem Pengenalan Wajah Sebagai Penghubung Jejaring Sosial : Penerapan Augmentasi Reality sebagai Penampil Informasi Hasil Pengenalan Wajah Pada Perangkat Android*. Depok : Universitas Indonesia.

[6] Fowler, Martin. (2003). *UML Distilled: A Brief Guide to the Standard Object Modelling Language, Third Edition*. USA: Addison Wesley.

[7] http://www.shervinemami.co.cc/faceRecognition.html diakses pada 18 Desember 2011.

[8] D. Kriegman Ming-Hsuan Yang and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, January 2002.

[9] P.Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR'01.

[10]http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial recognition.htm diakses pada 18 November 2011

[11] O. Rezq and A. El-Sayed, "Geometrical Approach of Face Detection and Recognition,"

[12] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu , “Cloud Computing and Grid Computing 360-Degree Compared”, *Grid Computing Environments Workshop*,2008,GCE’08,12-16 Nov.2008

[13] Lijun Mei, W.K. Chan, T.H. Tse, "A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues," apscc, pp.464-469, *2008 IEEE Asia-Pacific Services Computing Conference*, 2008.

[14] Vincenzo D. Cunsolo, Salvatore Distefano, Antonio Puliafito, Marco Scarpa, "Volunteer Computing and Desktop Cloud: The Cloud@Home Paradigm," nca, pp.134-139, *2009 Eighth IEEE International Symposium on Network Computing and Applications*, 2009.

[15] Rajkumar Buyya, James Broberg, Andrzej Goscinski, “Cloud Computing Principle and Paradigm,” Wiley, 2011

[16]REST,http://www.ibm.com/developerworks/webservices/library/ws-restful/ diakses pada 31 Mei 2012

[17] JSON, http://www.json.org/index.html diakses pada 28 Mei 2012

[18] Facebook, https://github.com/facebook/facebook-android-sdk diunduh pada 1 Mei 2012

[19]*Developer Android*, http://developer.android.com/ diakses pada 18 Desember 2011

[20] *Face.com API, http://developers.face.com* diakses pada 1 April 2012.

[21]*Speedtest*,http://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest diunduh pada 31 Mei 2012

[22]*FaceDetectionAndroid*,http://www.smallscreendesign.com/2011/01/25/about-face-detection-on-android-part-1/ diakses pada 31 Juni 2012

[23]*HttpMultipart*,http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.httpcomponents/httpmime/4.1.2 diunduh pada 1 April 2012