



RANCANG BANGUN SISTEM PENGENALAN WAJAH (*FACE RECOGNITION*) MENGGUNAKAN METODE *EIGENFACE* BERBASIS *ANDROID*



TUGAS AKHIR

**Program Studi
S1 Sistem Komputer**

**INSTITUT BISNIS
DAN INFORMATIKA**

stikom

SURABAYA

Oleh:

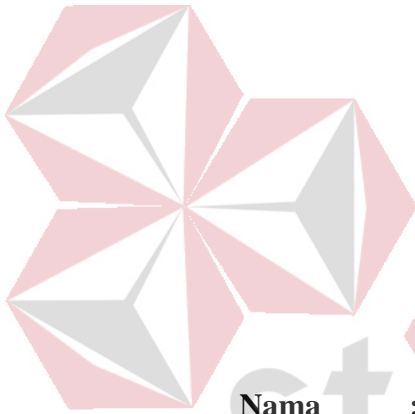
**MOCHAMMAD DIMAS PRASETIYO WIBOWO
14.41020.0016**

**FAKULTAS TEKNOLOGI DAN INFORMATIKA
INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA
2018**

RANCANG BANGUN SISTEM PENGENALAN WAJAH (*FACE RECOGNITION*) MENGGUNAKAN METODE *EIGENFACE* BERBASIS *ANDROID*

TUGAS AKHIR

**Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana Komputer**



**INSTITUT BISNIS
DAN INFORMATIKA**

Oleh :

Nama : MOCHAMMAD DIMAS PRASETIYO WIBOWO

NIM : 14.41020.0016

Program : S1 (Strata Satu)

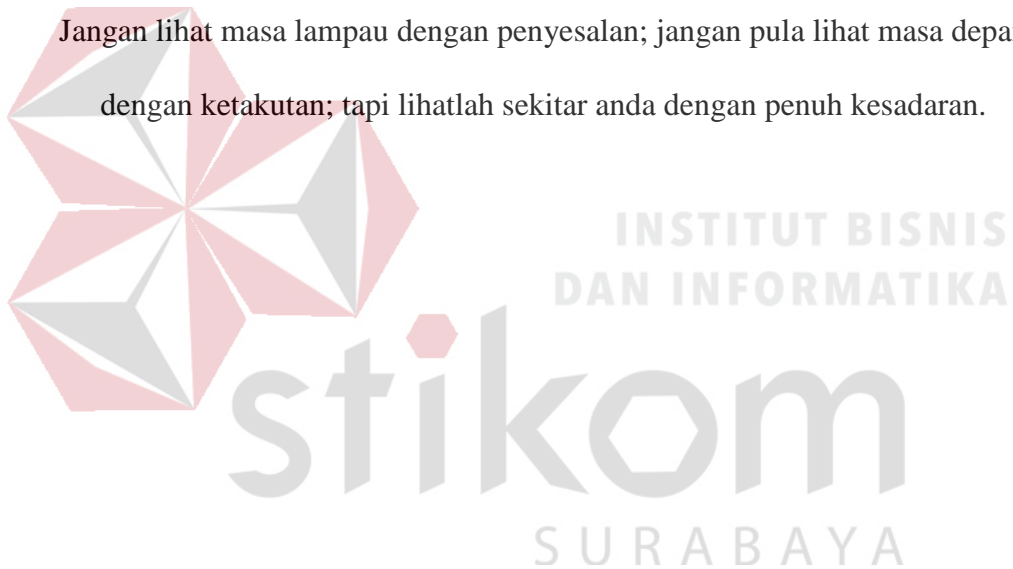
Jurusan : Sistem Komputer

**FAKULTAS TEKNOLOGI DAN INFORMATIKA
INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA**

2018

"Mereka berkata bahwa setiap orang membutuhkan tiga hal yang akan membuat mereka berbahagia di dunia ini, yaitu; sesuatu untuk dilakukan, dan sesuatu untuk diharapkan." (Tom Bodett)

Jangan lihat masa lampau dengan penyesalan; jangan pula lihat masa depan dengan ketakutan; tapi lihatlah sekitar anda dengan penuh kesadaran.



Alhamdu'lilahi robbilalamin,

Segala Puji Bagi Tuhan Yang Maha Esa,

Shalawat dan salam tidak lupa selalu tercurah kepada baginda

Rasulullah Muhammad SAW.

Tugas Akhir ini Saya Persembahkan Kepada

Orang Tua, Adik, dan Semua Keluarga atas dukungan dan doa-doanya

Terimakasih Kepada Dosen-Dosen Pembimbing

Serta Semua Rekan-Rekan Sistem Komputer dan Dikampus Institut Bisnis dan

Informatika Stikom Surabaya

Beserta Semua Orang Yang Membantu Saya.

TUGAS AKHIR
RANCANG BANGUN SISTEM PENGENALAN WAJAH (FACE
RECOGNITION) MENGGUNAKAN METODE EIGENFACE
BERBASIS ANDROID

Dipersiapkan dan disusun oleh
Mochammad Dimas Prasetyo Wibowo
NIM : 14.41020.0016

Telah diperiksa, diuji dan disetujui oleh Dewan Penguji
Pada : Agustus 2018

Susunan Dewan Penguji

Pembimbing

I. Dr. Susijanto Tri Rasmana, S.Kom., M.T.

NIDN. 0727097302

II. Ira Puspasari, S.Si., M.T.

NIDN. 0710078601

Pembahas

I. Weny Indah Kusumawati, S.Kom., M.MT.

NIDN. 0721047201

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar Sarjana



FAKULTAS TEKNOLOGI
DAN INFORMATIKA

Dr. Jusak

Dekan Fakultas Teknologi dan Informatika

FAKULTAS TEKNOLOGI DAN INFORMATIKA
INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA

**PERNYATAAN
PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH**

Sebagai mahasiswa Institut Bisnis dan Informatika Stikom Surabaya, saya :

Nama : Mochammad Dimas Prasetyo Wibowo
NIM : 14410200016
Program Studi : S1 Sistem Komputer
Fakultas : Fakultas Teknologi dan Informatika
Jenis Karya : Tugas Akhir
Judul Karya : **RANCANG BANGUN SISTEM PENGENALAN WAJAH
(FACE RECOGNITION) MENGGUNAKAN METODE
EIGENFACE BERBASIS ANDROID**

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Institut Bisnis dan Informatika Stikom Surabaya Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right*) atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data (*database*) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, Agustus 2018

Yang menyatakan



Mochammad Dimas Prasetyo Wibowo
NIM : 14410200016

ABSTRAK

Pengenalan wajah adalah teknologi biometrik yang dapat diterapkan ke berbagai bidang dalam hal identitas wajah. Namun, ada dua hal yang menjadi masalah pengenalan wajah yaitu proses ekstraksi fitur dan teknik klasifikasi yang digunakan. Pengenalan wajah menggunakan informasi di dalam *Database* untuk mengenali wajah siapakah yang ada di dalam sebuah gambar wajah. Aplikasi ini bisa terjadi karena adanya algoritma *Eigenface*, sejak pertama kali digunakan metode *Eigenface* telah mengalami banyak perkembangan hingga saat ini. Bagaimana proses mengenali wajah seseorang dengan memanfaatkan dengan metode *Eigenface*. Demikian halnya pada jenis teknologi *Smartphone*. Pengenalan wajah juga dikembangkan pada *smartphone* berbasis *Android*.

Dalam tulisan ini, ditunjukkan wajah sistem pengenalan di perangkat *Android* menggunakan metode *Eigenface*. Sistem ini dapat digunakan sebagai dasar untuk pengembangan aplikasi *Android* seperti aplikasi keamanan *mobile Android* dan beberapa arsip untuk melakukan penelitian identitas wajah.

Percobaan sistem pengenalan wajah dalam makalah ini mengambil 1~ 25 gambar wajah sebagai uji coba. Uji coba terhadap gambar wajah tersebut sebagai pelatihan pengenalan wajah diambil secara langsung dengan menggunakan kamera di perangkat *Android* dengan jarak pemotretan wajah yang berkisaran antaran 25 ~ 60 cm. Dengan tingkat keberhasilan mencapai 76,92%.

Kata kunci : Pengenalan Wajah, Metode *Eigenface*, *Face Detection*, Aplikasi *Android*

KATA PENGANTAR

Pertama-tama penulis panjatkan puji dan syukur ke haddirat Allah SWT yang telah memberikan kekuatan, kesehatan lahir dan batin sehingga penulis dapat menyelesaikan penulisan Tugas Akhir ini dengan sebaik-baiknya. Penulis mengambil judul “*RANCANG BANGUN SISTEM PENGENALAN WAJAH (FACE RECOGNITION) MENGGUNAKAN METODE EIGENFACE BERBASIS ANDROID*” ini sebagai salah satu syarat dalam menyelesaikan Tugas Akhir di Institut Bisnis dan Informatika Stikom Surabaya.

Pada kesempatan kali ini penulis juga ingin mengucapkan terima kasih kepada:

1. Bapak, Ibu tercinta yang telah memberikan dukungan dan doa selama mengerjakan Tugas Akhir ini.
2. Pimpinan Stikom Surabaya yang telah banyak memberikan motivasi serta teladan yang dapat membantu penulis selama menempuh pembelajaran hingga saat ini.
3. Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi Sistem Komputer Stikom Surabaya yang senantiasa memberikan dukungan kepada penulis sehingga penulis dapat melaksanakan Tugas Akhir ini dengan baik.
4. Bapak Dr. Susijanto Tri Rasmana, S.Kom., M.T., dan Ibu Ira Puspasari, S.Si., M.T., selaku dosen pembimbing satu dan dua yang telah membantu serta mendukung setiap kegiatan sehingga pelaksanaan Tugas Akhir ini dapat berjalan dengan baik.

5. Ibu Weny Indah Kusumawati, S.Kom., M.MT., selaku pembahas yang senantiasa memberikan dukungan kepada penulis sehingga penulis dapat melaksanakan Tugas Akhir ini dengan baik.
6. Seluruh dosen Pengajar Program Studi S1 Sistem Komputer yang telah mendidik, memberi motivasi kepada penulis selama masa kuliah di Institut Bisnis dan Informatika Stikom Surabaya.
7. Seluruh pihak yang tidak dapat penulis tuliskan satu persatu yang telah membantu penulis secara langsung maupun tidak langsung.

Banyak hal dalam laporan Tugas Akhir ini yang masih perlu diperbaiki lagi. Oleh karena itu penulis mengharapkan saran dan kritik yang dapat membangun dari semua pihak agar dapat menyempurnakan penulisan ini kedepannya. Penulis juga memohon maaf yang besar jika terdapat kata-kata yang salah serta menyinggung perasaan pembaca. Akhir kata penulis ucapkan banyak terima kasih yang besar kepada para pembaca, semoga tulisan ini dapat bermanfaat bagi para pembaca.

Surabaya, Agustus 2018

Penulis

DAFTAR ISI

	Halaman
ABSTRAK	vii
KATA PENGANTAR	viii
DAFTAR ISI.....	x
DAFTAR GAMBAR	xiv
DAFTAR TABEL.....	xvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang Masalah.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan	3
1.5 Sistematika Penulisan	3
BAB II LANDASAN TEORI.....	5
2.1 Metode Pengenalan Wajah <i>Egienface</i>	5
2.1.1 Algoritma <i>Eigenface</i>	5
2.2 <i>Smartphone</i>	8
2.2.1 Pengertian <i>Smartphone</i>	8
2.2.2 Perbedaan antara <i>Handphone</i> dengan <i>Smartphone</i>	9
2.3 <i>Android</i>	10

2.3.1 Pengertian <i>Android</i>	10
2.3.2 Bahasa Pemrograman pada <i>Android</i>	12
2.4 <i>Android Studio</i>	12
2.4.1 Mengenal <i>Android Studio</i>	12
2.4.2 Struktur Proyek <i>Android Studio</i>	13
2.4.3 Antarmuka Pengguna <i>Android Studio</i>	15
BAB III METODE PENELITIAN DAN PERANCANGAN SISTEM	17
3.1 Metode Penelitian	17
3.2 Rancangan Sistem	18
3.3 Aplikasi Sistem	20
3.3.1 Deteksi Wajah	20
3.3.2 Pengenalan Wajah.....	21
3.4 Paket Perangkat Lunak <i>Android Studio</i>	22
3.5 Parameter Keberhasilan.....	23
BAB IV HASIL DAN PEMBAHASAN	24
4.1 Pengujian Deteksi Wajah Menggunakan Metode <i>EigenFace</i>	24
4.1.1 Prosedur Pengujian Deteksi Wajah.....	24
4.1.2 Hasil Pengujian Deteksi Wajah dalam Beberapa Sudut Pandang	26

4.1.3 Hasil Pengujian Deteksi Wajah dengan Pengujian Aksesoris Tambahan	34
4.1.4 Hasil Pengujian Deteksi Wajah dengan Beberapa Sampel ...	37
4.1.5 Pengujian Pengenalan Wajah Terhadap Benda atau Tidak Ada Wajah di <i>Background</i>	42
4.2 Perhitungan dengan Metode Eigenface secara Detail atau Lengkap	44
4.2.1 Perhitungan dengan Metode <i>Eigenface</i> pada Kedua Gambar Wajah yang Sama dan <i>Data</i> Tabel juga Sama	50
4.3 Perhitungan dengan Metode <i>Eigenface</i> secara <i>Simple</i> atau Sederhana.....	53
4.3.1 Perubahan Gambar Asli atau <i>RGB</i> menjadi Gambar <i>Grayscale</i> atau Abu-abu	53
4.3.2 Perhitungan Training Image dengan Metode Flatveccor	54
4.3.3 Perhitungan Ekstrasi <i>PCA</i>	57
4.3.4 Hasil Tabel Perhitungan Proses Metode <i>EigenFace</i>	59
4.4 Menguji coba Aplikasi <i>Android</i>	61
4.4.1 Tabel Hasil Pengujian Coba Pengenalan Wajah.....	67
4.5 Pengujian Coba Pengenalan Wajah dengan Menentukan Beberapa Nilai <i>Face Threshold</i> dan Nilai <i>Distance Threshold</i>	68
4.6 Hasil Pengujian Keseluruhan	75

BAB V KESIMPULAN DAN SARAN.....	81
5.1 Kesimpulan.....	81
5.2 Saran.....	82
DAFTAR PUSTAKA	83
LAMPIRAN.....	86
BIODATA PENULIS	124



DAFTAR GAMBAR

	Halaman
Gambar 2.1. Algoritma <i>Eigenface</i>	5
Gambar 2.2. Beberapa <i>Data Training Image</i>	6
Gambar 2.3. <i>File</i> Proyek Di Tampilan <i>Android</i>	14
Gambar 2.4. Jendela Utama <i>Android Studio</i>	15
Gambar 3.1. Blok Diagram Sistem Pengenalan Wajah Berbasis <i>Android</i>	18
Gambar 3.2. Algoritma Pengenalan Wajah Pada <i>Smartphone</i>	19
Gambar 3.3 Tampilan Deteksi Wajah.....	21
Gambar 3.4 Tampilan Pengenalan Wajah.....	21
Gambar 3.5 Paket-Paket Aplikasi Pendukung Pada <i>Android Studio</i>	22
Gambar 4.1. Tampilan Layar Utama Pada <i>Android</i>	25
Gambar 4.2. Tampilan Halaman Utama Pada Aplikasi <i>Android</i>	25
Gambar 4.3. Tampilan <i>Data</i> Pada Gambar Wajah Yang Pertama Dengan Warna Atau <i>Rgb</i>	53
Gambar 4.4. Tampilan <i>Data</i> Pada Gambar Wajah Yang Pertama Dengan Abu- Abu Atau <i>Grayscale</i>	54
Gambar 4.5. Tampilan <i>Data</i> Pada Gambar Wajah Yang Pertama Dengan Abu- Abu Atau <i>Grayscale</i>	55
Gambar 4.6. Tampilan <i>Data</i> Pada Gambar Wajah Yang Kedua Dengan Abu- Abu Atau <i>Grayscale</i>	55

Gambar 4.7. Tampilan <i>Data</i> Uji Coba Pada Gambar Wajah Yang Pertama ...	58
Gambar 4.8. Tampilan <i>Data</i> Uji Coba Pada Gambar Wajah Yang Kedua.....	59
Gambar 4.9. Tampilan Layar Utama Pada <i>Android</i>	61
Gambar 4.10. Tampilan Aplikasi “ <i>Myfiles</i> ” Pada <i>Android</i>	61
Gambar 4.11. Tampilan Layar Utama Pada <i>Android</i>	62
Gambar 4.12. Tampilan Layar Utama Pada Aplikasi “Pengenalan Wajah Dengan Metode <i>Eigen Face</i> ”	62
Gambar 4.13. Tampilan Mengatur Nilai <i>Face Threshold</i> Dan <i>Distance Threshold</i> Pada Aplikasi “Pengenalan Wajah Dengan Metode <i>EigenFace</i> ”	63
Gambar 4.14. Tampilan Mengambil Gambar Wajah Yang Pertama.....	63
Gambar 4.15. Tampilan Mengisi Sebuah Nama Yang Diinginkan	64
Gambar 4.16. Tampilan Daftar Isi Pengenalan Wajah	64
Gambar 4.17. Tampilan Mendeteksi Gambar Wajah Yang Pertama.....	65
Gambar 4.18. Tampilan Mendeteksi Gambar Wajah Yang Kedua	65
Gambar 4.19. Tampilan Mengatur Nilai <i>Face Threshold</i> Dan <i>Distance Threshold</i> Pada Aplikasi “Pengenalan Wajah Dengan Metode <i>EigenFace</i> ”	66
Gambar 4.20. Mengambil Gambar Wajah Yang Tidak Memakai Kacamata ..	66
Gambar 4.21. Tampilan Mendeteksi Wajah Yang Tidak Dapat Menampilkan Nama Karena <i>Error</i>	67
Gambar 4.22. Tampilan Nilai <i>Face Threshold</i> Dan <i>Distance Threshold</i>	68

DAFTAR TABEL

Halaman

Tabel 4.1. Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Kanan	26
Tabel 4.2. Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Kiri	28
Tabel 4.3. Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Atas	30
Tabel 4.4. Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Bawah.....	32
Tabel 4.5. Tabel Pengujian Aksesoris Tambahan Dengan Kacamata	34
Tabel 4.6. Tabel Pengujian Aksesoris Tambahan Dengan Topi	35
Tabel 4.7. Tabel Pengujian Aksesoris Tambahan Dengan Kacamata Dan Topi	36
Tabel 4.8. Tabel Pengujian Beberapa Orang Yang Akan Didapatkan.....	37
Tabel 4.9. Tabel Pengujian Pengenalan Wajah Terhadap Benda Atau Tidak Ada Wajah Di Background.....	43
Tabel 4.10. Hasil Tabel Perhitungan Semua Yang Didapatkan.....	60
Tabel 4.11. Hasil Pengujian Coba Dalam Aplikasi Tersebut.....	68
Tabel 4.12. Tabel Pengajian Pengenalan Wajah Dengan Mengatur Nilai <i>Face Threshold</i> Dan Nilai <i>Distance Threshold</i>	69

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Proses pengenalan wajah yang dilakukan oleh komputer tidak semudah pengenalan wajah yang dilakukan oleh manusia. Manusia dengan mudah mengenali wajah seseorang tanpa harus berpikir, asalkan masih dalam batas-batas yang masih dapat dilihat, seperti dalam keadaan menunduk, menoleh, ataupun memakai kacamata. Sedangkan komputer selain lambat dalam melakukan pengenalan, juga sangat berpengaruh terhadap faktor dari luar, seperti cahaya, posisi wajah, maupun aksesoris yang sedang dipakai. Sudah banyak algoritma yang digunakan dalam mengatasi masalah yang dihadapi oleh komputer dalam mengenali wajah, seperti *Kohonen*, *Nearest Feature Midpoint*, *Canonical Correlation Analysis*, *Eigenface*, dan masih banyak lagi. Setiap algoritma mempunyai cara pengenalan berbeda-beda, yaitu pengenalan berdasarkan *system feature-based*, yaitu dengan mengekstraksi ciri dari komponen citra wajah seperti mata, hidung, mulut, dan lain-lain, sedangkan *system image-based* yaitu menggunakan informasi mentah dari *pixel* citra yang kemudian diekstraksi kedalam metode tertentu. Algoritma *Eigenface* dipilih penulis sebagai metode pengenalan wajah yang digunakan dalam penelitian tugas akhir ini. Pengenalan wajah dilakukan pada *smartphone* berbasis *Android* dan pembangunan aplikasi pengenalan wajah ini terdapat algoritma *Eigenface* berfungsi untuk klasifikasi identitas citra, kemudian hasil perhitungan disimpan dalam folder pola master yang nantinya akan dicocokkan dengan dengan *testface*. Penulis berharap

dapat menghasilkan *Android* dapat pengenalan wajah yang tepat guna (Adiwijaya, 2007).

Pengenalan wajah (*Face Recognition*) merupakan salah satu teknologi biometrik yang banyak diaplikasikan dalam sistem security selain pengenalan retina mata, pengenalan sidik jari dan iris mata (Pridiono, 2014). Dalam aplikasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah sebelumnya yang telah disimpan di dalam *database* tertentu. Penggunaan *face recognition* disini adalah untuk mengenali beberapa wajah yang telah dideteksi.

1.2 Rumusan Masalah

Dalam perancangan dan pembuatan alat ini, terdapat rumusan masalah, Bagaimana cara mengimplementasikan pengenalan wajah pada *smartphone* menggunakan metode *Eigenface*?

1.3 Pembatasan Masalah

Adapun permasalahan yang akan dihadapi dalam pengerjaan Tugas Akhir ini diantaranya adalah:

- a. Citra *input* berlatar ekspresi wajah, yaitu posisi menghadap kedepan tanpa tersenyum, menghadap kedepan dengan senyum tipis, menghadap kedepan dengan senyum lebar, kepala miring kekiri, kepala miring kekanan.
- b. Pencahayaan citra pada kamera *smartphone* agak terang atau memiliki *brightness* yang sedikit terang agar jelas diambil pengenalan wajah.

1.4 Tujuan

Dalam perancangan dan pembuatan alat ini, bertujuan untuk mengimplementasikan pengenalan wajah pada *smartphone* menggunakan metode *Eigenface*.

1.5 Sistematika Penulisan

Pembahasan Tugas Akhir ini secara Garis besar tersusun dari 5 (lima) bab, yaitu diuraikan sebagai berikut:

1. BAB I PENDAHULUAN

Pada Bab ini akan dibahas mengenai latar belakang masalah, batasan masalah, tujuan penulisan, dan sistematika penulisan.

2. BAB II LANDASAN TEORI

Pada Bab ini akan dibahas teori penunjang dari permasalahan, yaitu membahas mengenai metode *Eigenface*, *Smartphone*, *Android*, dan *Android Studio*.

3. BAB III METODE PENELITIAN DAN PERANCANGAN SISTEM

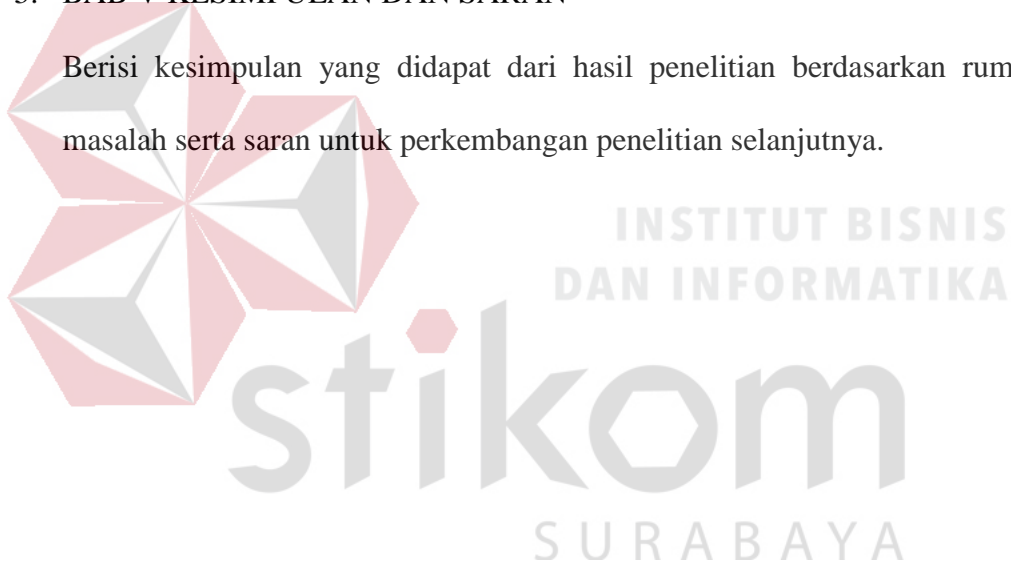
Pada bab ini akan dibahas tentang blok diagram dan desain sistematika *Android* dengan *smartphone* sistem serta metode yang digunakan dalam pembuatan rancang bangun. Perancangan dilakukan dengan melakukan perancangan yang meliputi perancangan aplikasi *Android*, membangun aplikasi *Android* dengan menggunakan *Android studio*, dan terakhir dilakukan perancangan perangkat lunak yang akan menjalankan seluruh aplikasi *Android* yang mendukung metode *Eigenface* dan *opencv* (untuk mendeteksi wajah dengan menggunakan fitur dukungan kamera).

4. BAB IV HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai hasil dari pengujian mendeteksi wajah dan pengujian pengenalan wajah yang sudah terdeteksi seperti pengujian dalam beberapa sudut pandang, pengujian aksesoris tambahan, pengujian terhadap beberapa orang, dan pengujian terhadap benda atau tidak ada *background*. Kemudian akan dibahas dari hasil pengujian perancangan seluruh sistem yang nantinya dapat diperoleh hasil kondisi yang benar agar sistem dapat bekerja dengan baik sesuai dengan ide perancangan tersebut.

5. BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan yang didapat dari hasil penelitian berdasarkan rumusan masalah serta saran untuk perkembangan penelitian selanjutnya.



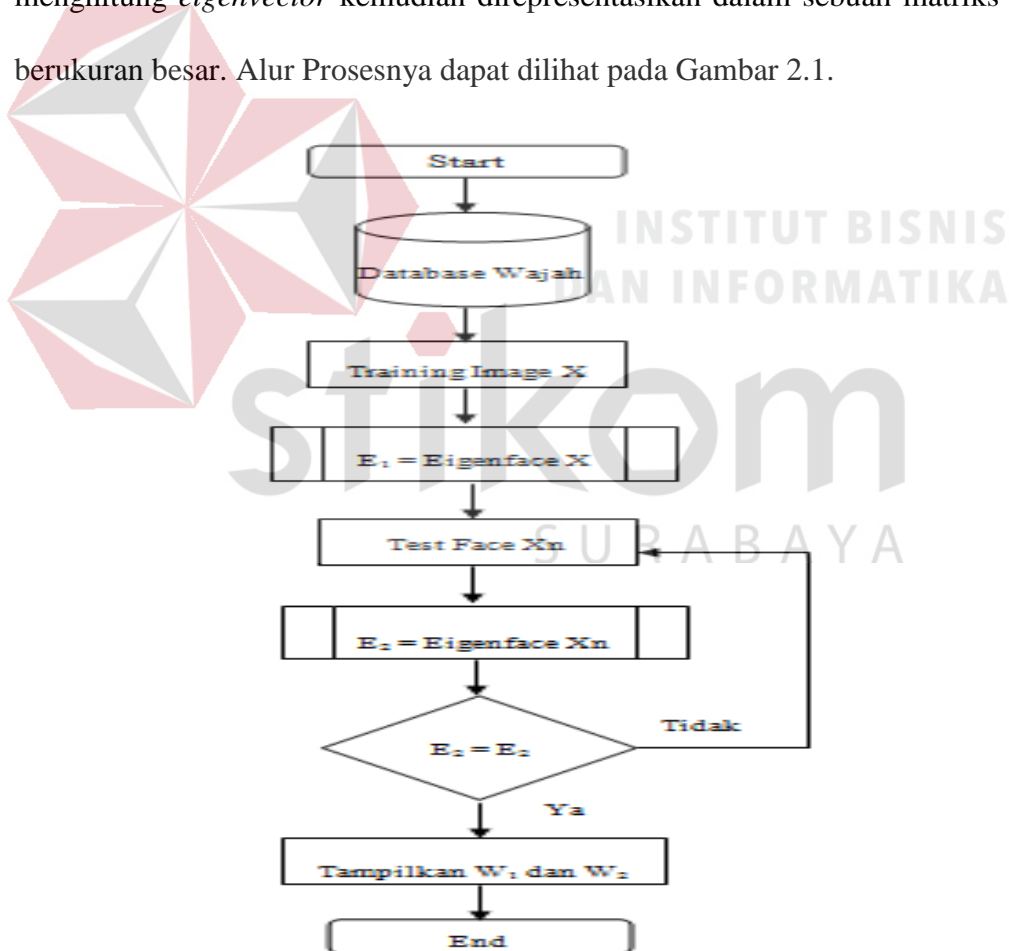
BAB II

LANDASAN TEORI

2.1 Metode Pengenalan Wajah *Eigenface*

2.2.1 Algoritma *Eigenface*

Prinsip dasar dari pengenalan wajah adalah dengan mengutip informasi unik wajah tersebut kemudian di-*encode* dan dibandingkan dengan hasil *decode* yang sebelumnya dilakukan. Dalam metode *eigenface*, *decoding* dilakukan dengan menghitung *eigenvector* kemudian direpresentasikan dalam sebuah matriks yang berukuran besar. Alur Prosesnya dapat dilihat pada Gambar 2.1.



Gambar 2.1 Algoritma *Eigenface*

Algoritma *Eigenface* secara keseluruhan cukup sederhana. *Image* Matriks (Γ) direpresentasikan ke dalam sebuah himpunan matriks ($\Gamma_1, \Gamma_2, \dots, \Gamma_M$). Cari nilai rata-rata (Ψ) dan gunakan untuk mengekstraksi *eigenvector* (v) dan *eigenvalue* (λ) dari himpunan matriks. Gunakan nilai *eigenvector* untuk mendapatkan nilai *eigenface* dari *image*. Apabila ada sebuah *image* baru atau *test face* (Γ_{new}) yang ingin dikenali, proses yang sama juga diberlakukan untuk *image* (Γ_{new}), untuk mengekstraksi *eigenvector* (v) dan *eigenvalue* (λ), kemudian cari nilai *eigenface* dari *image test face* (Γ_{new}). Setelah itu barulah *image* baru (Γ_{new}) memasuki tahapan pengenalan dengan menggunakan metode *euclidean distance*.

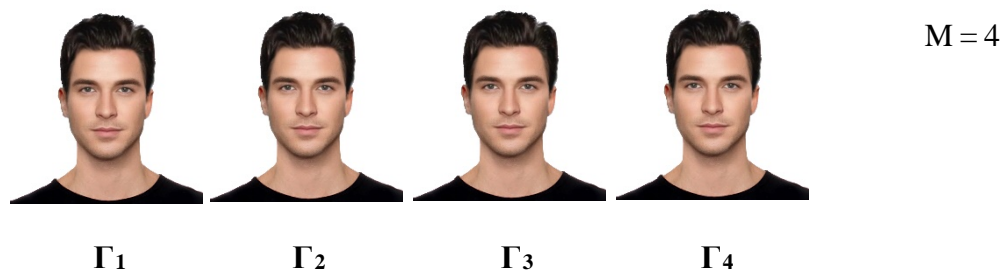
Algoritma selengkapnya adalah (Turk, Matthew dan Alex P.Pentland, 1991):

A. Tahapan Perhitungan *Eigenface*

- Langkah pertama adalah menyiapkan *data* dengan membuat suatu himpunan S yang terdiri dari seluruh *training image* ($\Gamma_1, \Gamma_2, \dots, \Gamma_M$)

$$S = (\Gamma_1, \Gamma_2, \dots, \Gamma_M) \quad (2.1)$$

>> Untuk mempersiapkan *data* dengan membuat suatu himpunan (S) yang terdiri dari seluruh *training image* seperti gambar 2.2. dibawah ini



Gambar 2.2 Beberapa *Data Training Image*

- Langkah kedua adalah ambil nilai tengah atau *mean* (Ψ)

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (2.2)$$

>> mengambil *data* nilai tengah (Ψ) untuk mencari jumlah *Image* (M)

- Langkah ketiga kemudian cari selisih (Φ) antara *training image* (Γ_i) dengan nilai tengah (Ψ)

$$\Phi_i = \Gamma_i - \Psi \quad (2.3)$$

>> mencari nilai selisih (Φ_i) antara *training image* (Γ_i) dengan nilai tengah (Ψ) apabila nilai yang sudah ditemukan nilainya dibawah nol ganti nilainya dengan nol.

- Langkah keempat adalah menghitung nilai matriks kovarian (C)

$$C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T \ll \Phi_n = AA^T \ll A = [\Phi_1, \Phi_2, \dots, \Phi_M] \quad (2.4)$$

>> Misal :: $C = \frac{1}{M} \sum_{n=1}^M [(\Phi_1 \cdot \Phi_1^T) + (\Phi_2 \cdot \Phi_2^T) + \dots + (\Phi_n \cdot \Phi_n^T)]$

$$L = A^T A \quad L = \Phi_n \Phi_m \quad (2.5)$$

>> Misal :: $L = \Phi_n^T \cdot \Phi_m$

>> menghitung nilai matriks kovarian (C) dengan mencari jumlah *image* (M) dan hasil nilai selisih (Φ_i)

- Langkah kelima menghitung *eigenvalue* (λ) dan *eigenvector* (v) dari matriks kovarian (C)

$$C x v_i = \lambda_i x v_i \quad (2.6)$$

>> Dimana v adalah *eigenvector* dari matriks M dan λ adalah *eigenvalue*.

Terdapat i buah *eigenvector* dan *eigenvalue* dalam sebuah $i \times i$ matriks.

Hubungan antara *eigenvalue* dan *eigenvector* dari suatu matriks digambarkan oleh persamaan.

6. Langkah keenam, setelah *eigenvector* (v) diperoleh, maka *eigenface* (μ) dapat dicari dengan:

$$\mu_i = \sum_{k=1}^M v_{lk} \Phi_k \quad (2.7)$$

$$l = 1, \dots, M \quad (2.8)$$

>> Setelah terdapat nilai *eigenvector* (v) yang diperoleh, maka mencari nilai *eigenface* (μ) antara hasil nilai selisih (Φ_i) dengan hasil nilai *eigenvector* (v)

B. Tahapan Pengenalan

1. Sebuah *image* wajah baru atau *test face* (Γ_{new}) akan dicoba untuk dikenali, pertama terapkan cara pada tahapan pertama perhitungan *eigenface* untuk mendapatkan nilai *eigenface* dari *image* tersebut.

$$\mu_{new} = v \cdot (\Gamma_{new} - \Psi) \quad (2.9)$$

$$\Omega = [\mu_1, \mu_2, \dots, \mu_n] \quad (2.10)$$

2. Gunakan metode *Euclidean Distance* untuk mencari jarak (*distance*) terpendek antara nilai *eigenface* dari *training image* dalam *database* dengan *eigenface* dari *image test face*.

$$\varepsilon = |\Omega - \Omega_k| \quad (2.11)$$

2.2 Smartphone

2.2.1 Pengertian Smartphone

Smartphone adalah sebuah *telephone* genggam yang memiliki fitur atau kemampuan tingkat tinggi, sering kali dalam penggunaannya menyerupai komputer, sehingga banyak orang mengartikan *smarphone* sebagai komputer genggam yang

memiliki fasilitas *telephone*. Fitur - fitur yang dapat ditemukan pada *smartphone* antara lain *telephone*, *sms*, *internet*, *ebook viewer*, *editing dokumen* dan masih banyak lagi yang lainnya. Dapat menambahkan aplikasi lain kedalam *smartphone* layaknya menginstall aplikasi pada komputer.

Sebelum *smartphone* dikenal seperti sekarang ini, mengenal adanya *telephone* seluler dan *PDA* (*Personal Digital Assistant*), ponsel berfungsi untuk *telephone* dan *sms* sementara *PDA* memiliki fungsi asisten digital pribadi dari sini munculah ide untuk menggabungkan fungsi keduanya.

2.2.2 Perbedaan antara *Handphone* dengan *Smartphone*

A. *Handphone*

Handphone memiliki fitur yang sangat terbatas, seperti *telephone*, *sms* dan untuk model terbaru dapat juga digunakan untuk memutar lagu, menonton *video* dan *internet*. Untuk *media* ketik *handphone* masih menggunakan *keypad* yang antara huruf dan angka menjadi satu ped. Sementara untuk menjalankan fitur *internet*, agak kesulitan karena harus melakukan settingan yang agak rumit. Dari sisi daya, *handphone* cenderung lebih hemat, karena proses yang dijalankan tidak sebanyak proses yang dijalankan oleh *smartphone*.

B. *Smartphone*

Dari sisi teknologi perbedaan antara *smartphone* dan *handphone* sangatlah jauh sekali, kemampuan *smartphone* mencakup semua layanan yang ada pada *handphone*. Bahkan kemampuan *smartphone* dewasa ini sebanding dengan komputer. Ini juga dapat menambah fitur-fitur pada *smartphone* dengan menginstall aplikasi-aplikasi sesuai dengan keinginan. Papan ketik yang

digunakan *Smartphone* lebih mudah untuk digunakan karena menggunakan model *QWERTY*, dan untuk masalah akses *internet* cukup menekan satu tombol untuk mengaktifkannya. Dari sisi penggunaan daya *smartphone* lebih boros, karena proses yang dijalankan lebih banyak dibandingkan *handphone* utamanya apabila mengaktifkan fitur *internet*.

2.3 *Android*

2.3.1 Pengertian *Android*

Android adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer *tablet*. *Android* awalnya dikembangkan oleh *Android, Inc.*, dengan dukungan finansial dari *Google*, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya *Open Handset Alliance*, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel *Android* pertama mulai dijual pada bulan Oktober 2008.

Antarmuka pengguna *Android* umumnya berupa manipulasi langsung, menggunakan gerakan sentuh yang serupa dengan tindakan nyata, misalnya menggeser, mengetuk, dan mencubit untuk memanipulasi objek di layar, serta papan ketik *virtual* untuk menulis teks. Selain perangkat layar sentuh, *Google* juga telah mengembangkan *Android TV* untuk televisi, *Android Auto* untuk mobil, dan *Android Wear* untuk jam tangan, masing-masingnya memiliki antarmuka pengguna yang berbeda. Varian *Android* juga digunakan pada komputer jinjing, konsol permainan, kamera *digital*, dan peralatan elektronik lainnya.

Android adalah sistem operasi dengan sumber terbuka, dan *Google* merilis kodenya di bawah Lisensi *Apache*. Kode dengan sumber terbuka dan lisensi perizinan pada *Android* memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi. Selain itu, *Android* memiliki sejumlah besar komunitas pengembang aplikasi (apps) yang memperluas fungsionalitas perangkat, umumnya ditulis dalam versi kustomisasi bahasa pemrograman *Java*. Pada bulan Oktober 2013, ada lebih dari satu juta aplikasi yang tersedia untuk *Android*, dan sekitar 50 miliar aplikasi telah diunduh dari *Google Play*, toko aplikasi utama *Android*. Sebuah survei pada bulan April-Mei 2013 menemukan bahwa *Android* adalah platform paling populer bagi para pengembang, digunakan oleh 71% pengembang aplikasi bergerak. Di *Google I/O* 2014, *Google* melaporkan terdapat lebih dari satu miliar pengguna aktif bulanan *Android*, meningkat dari 583 juta pada bulan Juni 2013.

Faktor-faktor di atas telah memberikan kontribusi terhadap perkembangan *Android*, menjadikannya sebagai sistem operasi telepon pintar yang paling banyak digunakan di dunia, mengalahkan *Symbian* pada tahun 2010. *Android* juga menjadi pilihan bagi perusahaan teknologi yang menginginkan sistem operasi berbiaya rendah, bisa dikustomisasi, dan ringan untuk perangkat berteknologi tinggi tanpa harus mengembangkannya dari awal. Sifat *Android* yang terbuka juga telah mendorong munculnya sejumlah besar komunitas pengembang aplikasi untuk menggunakan kode sumber terbuka sebagai dasar proyek pembuatan aplikasi, dengan menambahkan fitur-fitur baru bagi pengguna tingkat lanjut atau

mengoperasikan *Android* pada perangkat yang secara resmi dirilis dengan menggunakan sistem operasi lain.

Pada November 2013, *Android* menguasai pangsa pasar telepon pintar *global*, yang dipimpin oleh produk-produk *Samsung*, dengan persentase 64% pada bulan Maret 2013. Pada Juli 2013, terdapat 11.868 perangkat *Android* berbeda dengan beragam versi. Keberhasilan sistem operasi ini juga menjadikannya sebagai *target* litigasi paten "perang telepon pintar" antar perusahaan-perusahaan teknologi. Hingga bulan Mei 2013, total 900 juta perangkat *Android* telah diaktifkan di seluruh dunia, dan 48 miliar aplikasi telah dipasang dari *Google Play*.

2.3.2 Bahasa Pemrograman pada *Android*

Java

Menurut *TIOBE Index*, *Java* merupakan bahasa pemrograman paling populer pada bulan Juni 2017. Jika pembaca ingin membuat aplikasi *Android* maka *Java* adalah pilihan terbaik saat ini. *Java* memiliki komunitas yang sangat besar, baik di dunia maupun di Indonesia. Di Indonesia sendiri *Java* bahkan memiliki grup *facebook* dengan anggota terbesar setelah *PHP*. Dengan adanya komunitas yang besar kita tidak akan kesulitan untuk mencari jawaban saat menemukan kesulitan.

2.4 *Android Studio*

2.4.1 Mengenal *Android Studio*

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment (IDE)* untuk pengembangan aplikasi *Android*, berdasarkan *IntelliJ IDEA*. Selain merupakan *editor* dari kode *IntelliJ* dan alat

pengembangan yang sangat berguna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas membuat aplikasi *Android*, misalnya:

- a. Versi dari sistem yang berbasis *Gradle* yang fleksibel.
- b. Fitur *emulator* yang cepat dan kaya.
- c. Lingkungan yang menyatukan untuk pengembangan bagi semua perangkat lunak *Android*.
- d. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru.
- e. Template dan integrasi *GitHub* untuk membuat fitur aplikasi yang sama dan mengimpor beberapa contoh kode tersebut.
- f. Alat pengujian dan kerangka kerja yang ekstensif.
- g. Alat lintasan untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah yang lain.
- h. Dukungan bahasa pemrograman *C++* dan *NDK*.
- i. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah integrasi pada *Google Cloud Messaging* dan *App Engine*.

2.4.2 Struktur Proyek *Android Studio*

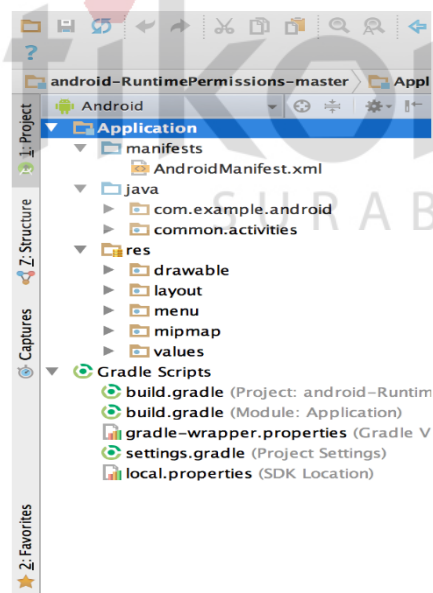
Setiap proyek di *Android Studio* berisi satu atau beberapa modul dengan sumber kode dari *file* dan *file* yang berisi sumber daya. Jenis-jenis modul mencakup:

- a. Modul aplikasi *Android*
- b. Modul Pustaka
- c. Modul *Google App Engine*

Secara *default*, *Android Studio* akan menampilkan *project file* dalam tampilan proyek *Android*, Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke *file* yang utama dari proyek tersebut, semua versi dari *file* ini terlihat di bagian atas maupun bagian bawah *Gradle Scripts* dan masing-masing modul aplikasi yang berisi *folder* berikut:

- a. Manifests ialah aplikasi yang berisi file `AndroidManifest.xml`.
- b. Java ialah berisi file sumber bahasa pemrograman Java, termasuk kode pengujian `JUnit`.
- c. Res ialah berisi semua sumber daya kode, seperti menatakan letak dengan XML, mendesain string UI, dan gambar yang menggunakan bitmap(.BMP).

Struktur proyek *Android* pada *disk* yang berbeda dari rata-rata ini. Untuk melihat struktur *file* sebenarnya dari proyek ini, pilih *Project* dari *menu* kemudian tarik turun *Project*. Dapat dilihat pada gambar 2.3. dibawah ini :

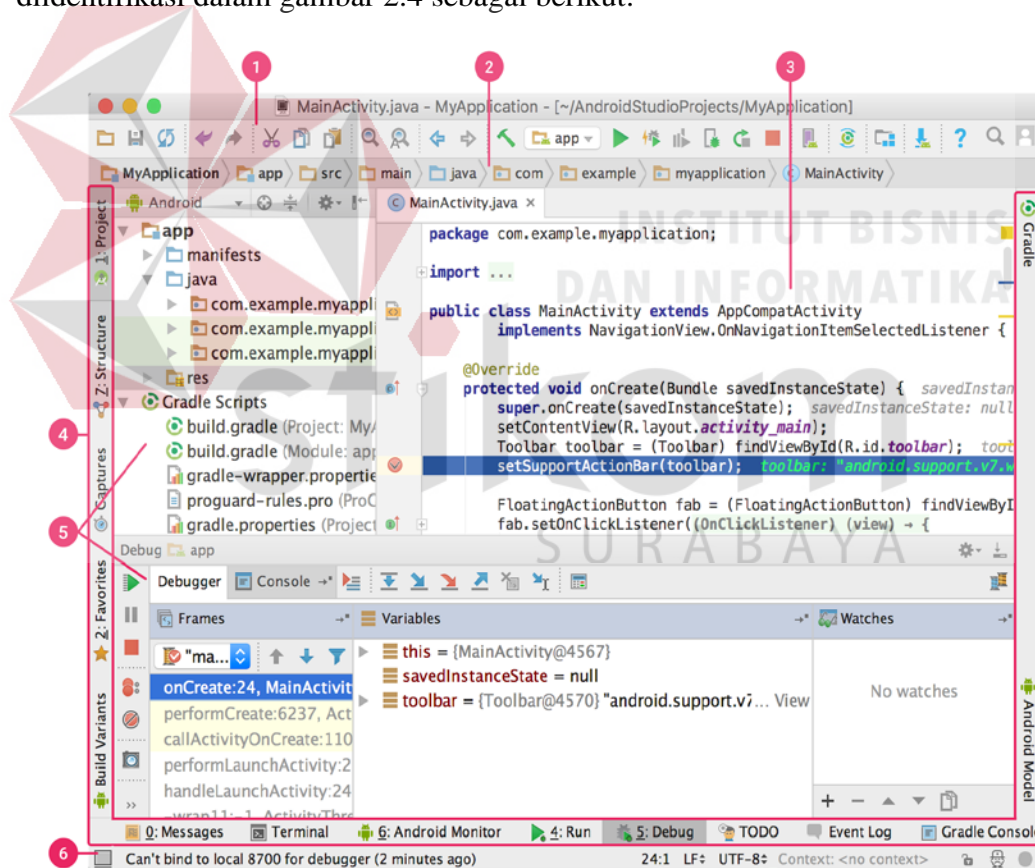


Gambar 2.3 File Proyek Di Tampilan *Android*.

Suatu struktur proyek *android* ini juga bisa menyesuaikan tampilan *file* proyek untuk melakukan pola pada aspek tertentu dari pengembangan aplikasi. Misalnya, memilih tampilan *Problems* dari tampilan proyek yang akan menampilkan tautan ke sumber *file* yang berisi kesalahan pengkodean dan *syntax* yang dikenal, misalnya *tag* penutup elemen *XML* tidak ada dalam *file* tata letak.

2.4.3 Antarmuka Pengguna *Android Studio*

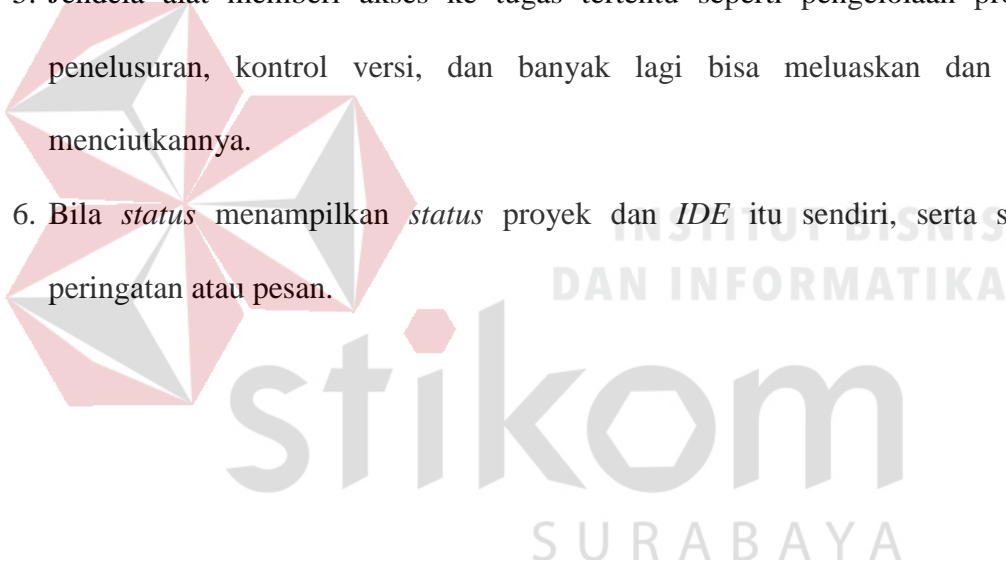
Jendela utama pada *Android Studio* terdiri dari beberapa bidang logika yang diidentifikasi dalam gambar 2.4 sebagai berikut:



Gambar 2.4 Jendela Utama *Android Studio*

1. Bila alat memungkinkan untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat *Android*.

2. Bila navigasi membantu bernavigasi di antara proyek dan membuka *file* untuk diedit. Bila ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela *Project*.
3. Jendela *editor* adalah tempat membuat dan memodifikasi kode. Bergantung pada jenis *file* saat ini, *editor* dapat berubah. Misalnya, ketika melihat *file* tata letak, *editor* menampilkan *Layout Editor*.
4. Bila jendela alat muncul di luar jendela *IDE* dan berisi tombol yang memungkinkan meluaskan atau menciutkan jendela alat *individual*.
5. Jendela alat memberi akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi bisa meluaskan dan juga menciutkannya.
6. Bila *status* menampilkan *status* proyek dan *IDE* itu sendiri, serta setiap peringatan atau pesan.



BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

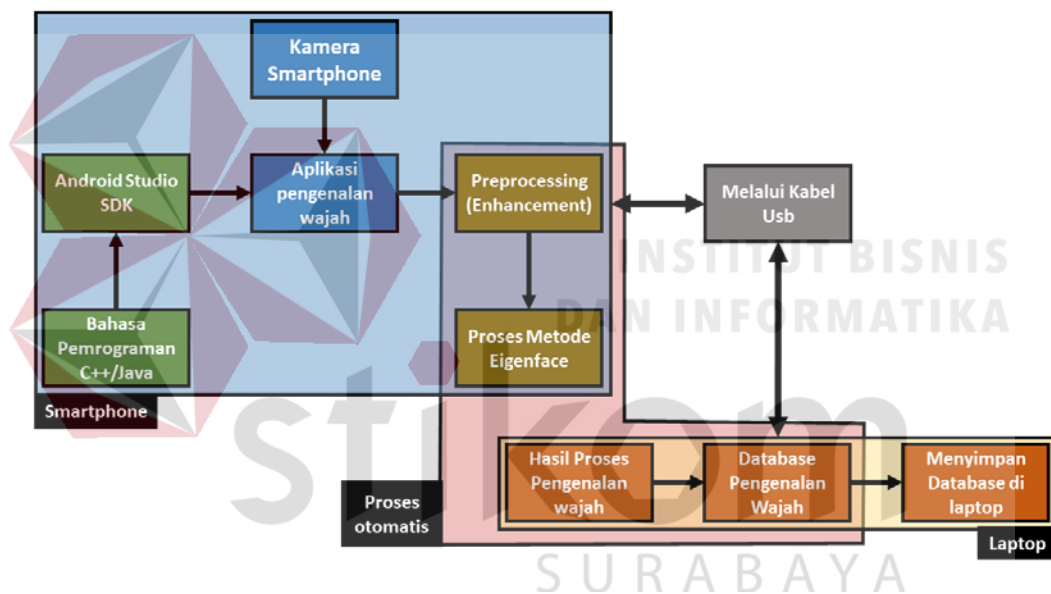
Pada sistem perancangan Tugas Akhir ini menggunakan *smartphone* yang berbasis *Android* dengan perancangan aplikasi *Android* yang bernama “pengenalan wajah dengan metode *Eigenface*”, perancangan tersebut menggunakan *Android studio* untuk membuat aplikasi *Android* pengenalan wajah dengan beberapa paket dukungan, yaitu *OpenCV*, *Face Recognition*, dan *EigenFace Library*. Perancangan sistem ini yang dimulai setelah membuat aplikasi *Android* pengenalan wajah dengan beberapa paket dukungan yang sudah diterapkan kemudian aplikasi *Android* pengenalan wajah tersebut telah membuka ke halaman utama pada aplikasi *Android*, setelah itu pada tab menu halaman pengaturan akurasi (*accurate*) ada tiga menu, yaitu mengatur nilai *Face Threshold*, nilai *Distance Threshold*, dan nilai *Maximum Image*. Tiga menu dari halaman pengaturan akurasi akan diuraikan sebagai berikut:

- Nilai *Face Threshold* untuk mengatur keakuratan wajah yang telah terdeteksi sehingga dapat menampilkan sebuah nama jika akurasi yang tinggi.
- Nilai *Distance Threshold* untuk mengatur jarak wajah yang telah terdeteksi yang berada di dekat maupun jauh terhadap kamera *smartphone* dan juga dapat menampilkan sebuah nama jika akurasi yang tinggi.

- Nilai *Maximum Image* untuk menentukan jumlah gambar wajah yang sudah terdeteksi dengan posisi yang berbeda dan juga jarak yang berbeda saat melakukan proses *training image*.

Pada metode *EigenFace* ini juga bisa bekerja dengan aplikasi *Android* pengenalan wajah yang terdeteksi dan dapat menampilkan sebuah nama dengan gambar wajah yang sudah terdeteksi.

3.2 Rancangan Sistem

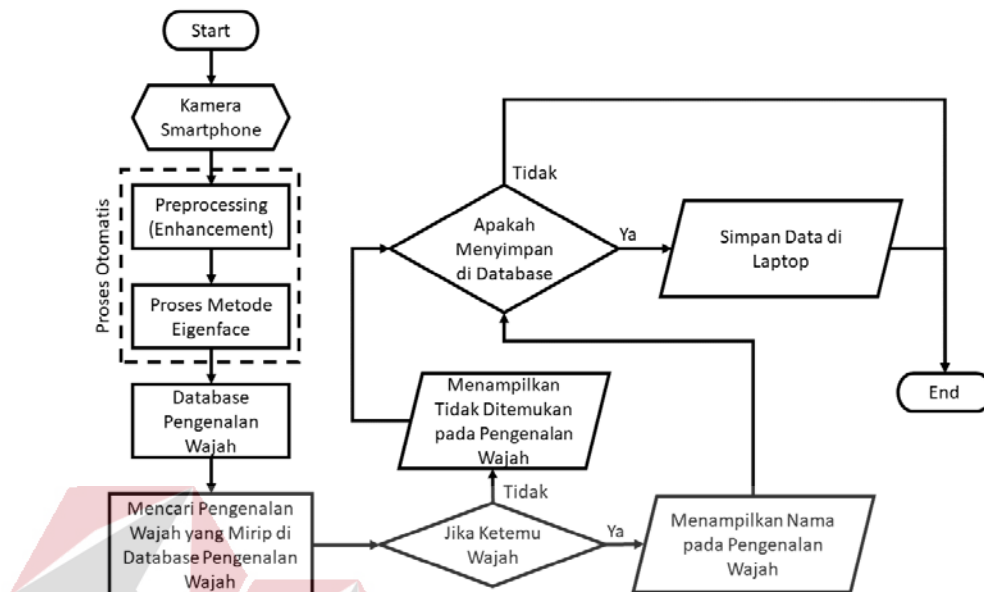


Gambar 3.1 Blok Diagram Sistem Pengenalan Wajah Berbasis *Android*

Keterangan diatas sebagai berikut:

- Apabila aplikasi pengenalan wajah yang telah dibuat dari *android studio* dengan menggunakan metode *eigenface* dan Bahasa pemrograman *java* atau Bahasa *C++*, maka menjalankan kamera *smartphone* untuk mendeteksi pengenalan wajah di aplikasi tersebut

- b. Apabila aplikasi pengenalan wajah yang sudah terdeteksi maka hasil pengenalan wajah yang akan menyimpan *file* ke simpan *data* pengenalan wajah.



Gambar 3.2 Algoritma Pengenalan Wajah Pada *Smartphone*

Hal ini menjelaskan berdasarkan keterangan diatas sebagai berikut:

- 1.) Inisialisasi *start* untuk memulai awal sebelum menjalankan aplikasi *Android*.
- 2.) Mengecek kondisi kamera pada *Smartphone* apakah fitur kamera tersebut bekerja agar bisa menjalankan aplikasi *Android*, kemudian mengambil gambar wajah dari fitur kamera pada *Smartphone* tersebut akan dilanjutkan ke *Preprocessing*.
- 3.) *Preprocessing* untuk melakukan proses gambar wajah yang tertangkap kemudian akan diproses dengan menggunakan metode *eigenface* secara otomatis.

- 4.) Apabila gambar wajah tersebut sudah diproses dengan metode *eigenface*, maka gambar wajah ini dapat menampilkan sebuah nama.
- 5.) Percocokan terhadap gambar wajah pertama dengan gambar wajah yang lain akan dihasilkan menjadi salah satu gambar wajah tersebut juga dapat menampilkan sebuah nama yang berbeda tiap gambar wajah jika terdeteksi wajah.
- 6.) Gambar wajah yang sudah terdeteksi tersebut tidak perlu menyimpan *database* di *Smartphone*, hanya bisa menyimpan *database* melalui *software Android studio* yang terhubung menggunakan kabel *USB* agar gambar wajah tersebut secara manual.
- 7.) Jika *Smartphone* dengan gambar wajah yang terdeteksi dan menampilkan sebuah nama akan otomatis menghapus dan tidak menyimpan *database* di *Smartphone* karena tidak mendukung fitur *database* di *software android studio*.

3.3 Aplikasi Sistem

3.3.1 Deteksi Wajah

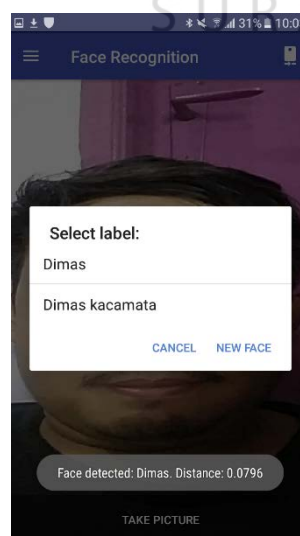
Sistem deteksi wajah dengan cara yaitu pada posisi gambar wajah yang tampak depan atau secara frontal dan jarak terhadap wajah yang agak dekat agar mudah terdeteksi wajah sebelum mengisi sebuah nama yang akan disimpan di daftar isi nama untuk mengenali sebuah nama cocok dengan gambar wajah yang terdeteksi



Gambar 3.3 Tampilan Deteksi Wajah

3.3.2 Pengenalan Wajah

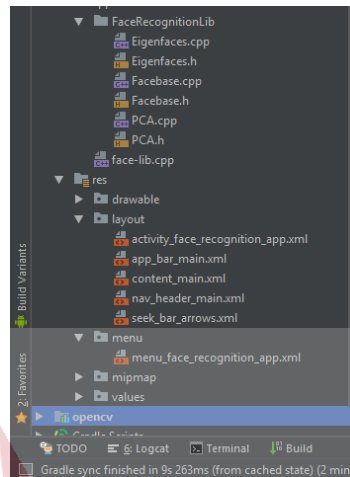
Sistem pengenalan wajah yang sudah dikenali dengan cara yaitu jika gambar wajah yang sudah terdeteksi kemudian mengisi sebuah nama yang diinginkan, setelah terisi sebuah nama setiap gambar wajah yang terdeteksi terus nama yang akan disimpan di daftar isi nama untuk mengenali sebuah nama yang cocok dengan gambar wajah yang terdeteksi.



Gambar 3.4 Tampilan Pengenalan Wajah

3.4 Paket Perangkat Lunak *Android Studio*

Ada beberapa paket perangkat lunak *Android Studio* yang dibutuhkan untuk mendukung proses aplikasi pengenalan wajah dengan metode *eigenface* sebagai berikut:



Gambar 3.5 Paket-Paket Aplikasi Pendukung Pada *Android Studio*

- 1.) *OpenCV* ialah buah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis secara *real-time*, paket *OpenCV* ini dapat mendukung fitur kamera dan fitur pendeteksi wajah.
- 2.) *Facerecognition* ialah paket ini berfungsi untuk menjalankan proses-proses mendeteksi wajah yang terdeteksi atau pengenalan wajah yang bekerja sama dengan fitur kamera dan paket *OpenCV*.
- 3.) *EigenFace library* ini terdiri dari tiga library yaitu *EigenFace*, *Facebase*, dan *PCA*. Tiga *library* ini dapat menjelaskan sebagai berikut:
 - a. *EigenFace* ini berfungsi untuk proses metode *Eigenface* terhadap gambar wajah yang telah terdeteksi dan *EigenFace* dapat bekerja sama dengan *Facebase* dan *PCA*.

- b. *Facebase* untuk mengkoleksi beberapa gambar wajah yang terdeteksi dengan beberapa posisi yang berbeda atau *background* juga beda, tetapi terus menjalankan proses *training image* sehingga hasil image yang sudah dihasilkan kemudian mendukung paket *EigenFace* untuk melanjutkan proses *training image* dari *Facebase*.
- c. *PCA library* adalah teknik yang digunakan untuk menyederhanakan suatu *data*, dengan cara mentransformasi *data* secara linier sehingga terbentuk sistem koordinat baru dengan varians maksimum, dan juga mendukung paket *EigenFace* dan *Facebase*.

3.5 Parameter Keberhasilan

Menguji pengenalan wajah pada kamera *smartphone* ini dapat mengenali wajah dan menampilkan nama dan juga dapat menyimpan *database* pengenalan wajah di laptop atau di *smartphone*. Diuji beberapa prosentase keberhasilan sistem dalam mengenali wajah yang terdeteksi atau dapat dikenali, nilai presentase keberhasilan ini telah mencapai diantara 70% sampai dengan 100% (keberhasilan dalam hasil pengujian pengenalan wajah yang mencapai 76,92 %).

BAB IV

HASIL DAN PEMBAHASAN

Hasil dan pembahasan yang dilakukan dalam penulis merupakan Hasil dan pembahasan yang dilakukan terhadap perangkat lunak (*Android*) dari sistem keseluruhan yang telah selesai dibuat untuk mengetahui program yang digunakan dalam sistem ini apakah berjalan dengan baik sesuai yang diinginkan terdapat beberapa pengujian yang dilakukan dalam perangkat lunak, antara lain:

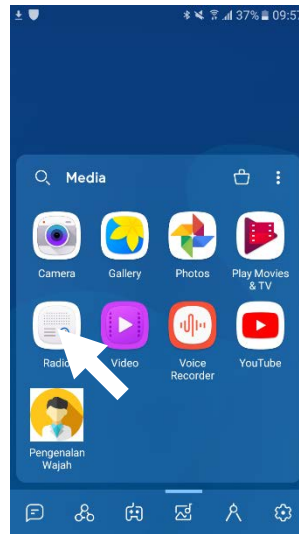
4.1 Pengujian Deteksi Wajah Menggunakan Metode *Eigenface*

Pengujian deteksi wajah menggunakan aplikasi *Android* untuk melakukan proses algoritma *principal component analysis (PCA)*. Proses ini, untuk mempresentasikan citra dalam gabungan setiap gambar *vector* yang dijadikan satu matriks tunggal sehingga akan membedakan antara citra wajah satu dengan citra yang lain.

4.1.1 Prosedur Pengujian Deteksi Wajah

Prosedur pengujian perangkat lunak (aplikasi *Android*):

1. Membuka aplikasi *Android* yang telah dibuat dan telah terinstall di *smartphone*, dapat dilihat pada gambar 4.1. sebagai berikut:



Gambar 4.1 Tampilan Layar Utama Pada *Android*

2. Halaman utama atau home pada aplikasi *Android*, dapat dilihat pada gambar 4.2. sebagai berikut:



Gambar 4.2 Tampilan Halaman Utama Pada Aplikasi *Android*

Prosedur pengujian ini terdapat empat bagian yaitu:

1. Pengujian deteksi wajah dalam beberapa sudut pandang.
2. Pengujian aksesoris tambahan ialah pengujian ini akan diuji apakah sistem dapat mendeteksi wajah apabila memakai aksesoris tambahan yaitu kacamata dan topi.
3. Pengujian pengenalan wajah terhadap beberapa orang ialah pengujian ini akan diuji berupa pengenalan wajah setiap orang yang akan dideteksi dan dapat menampilkan sebuah nama setiap orang.
4. Pengujian pengenalan wajah terhadap benda atau tidak ada wajah di *background*.

4.1.2 Hasil Pengujian Deteksi Wajah dalam Beberapa Sudut Pandang

Pengujian sudut wajah yang diarahkan ke kanan, lihat tabel 4.1 sebagai berikut ini:

Tabel 4.1 Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Kanan

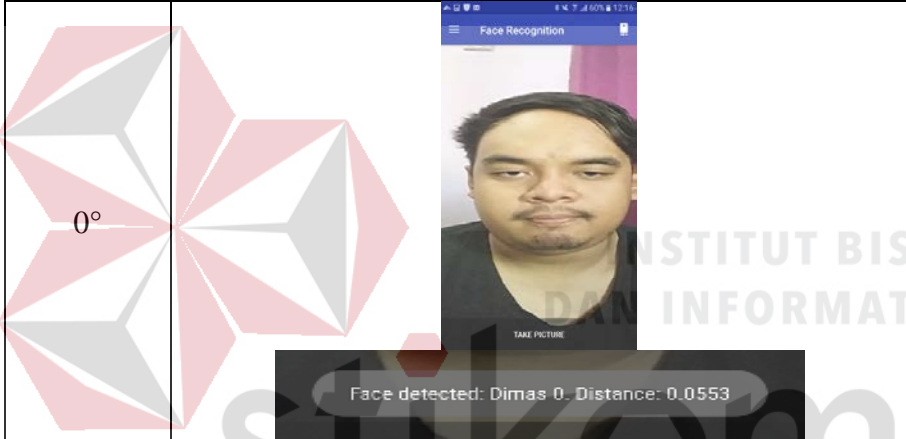
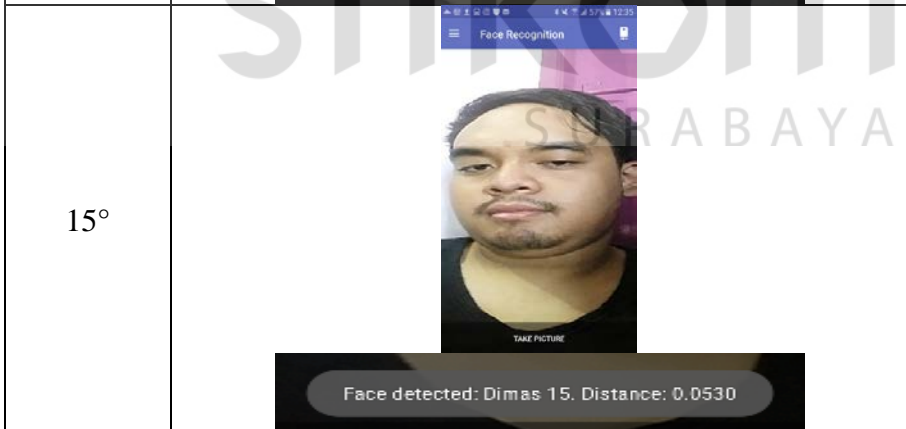
Tabel Pengujian Sudut Wajah yang Diarahkan Ke Kanan		
Sudut (°)	Deteksi Wajah	Status
0°		Berhasil

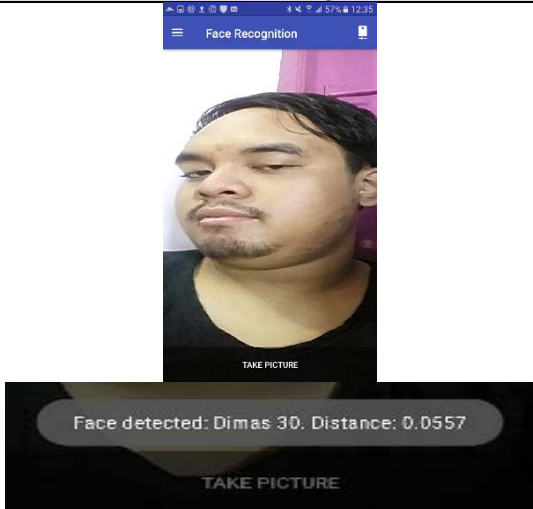
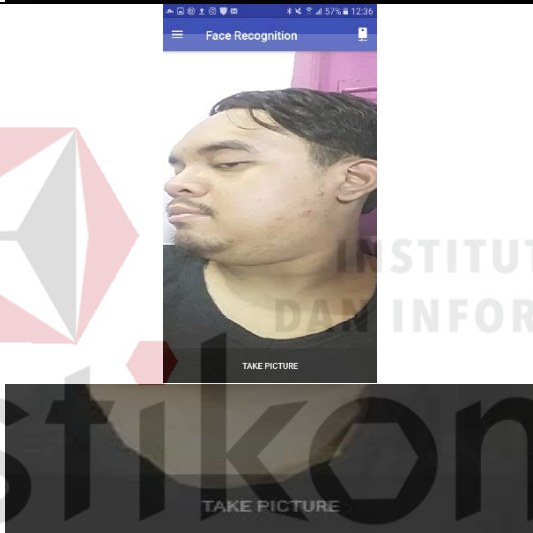
Tabel Pengujian Sudut Wajah yang Diarahkan Ke Kanan		
Sudut (°)	Deteksi Wajah	Status
15°		Berhasil
30°		Berhasil
45°		Berhasil

Hasil pengujian deteksi wajah diatas dijelaskan bahwa pengujian sudut 0° , 30° , dan 45° berhasil terdeteksi karena terdapat menampilkan sebuah nama dibawah daftar isi label pada aplikasi Android, sedangkan pengujian sudut 15° gagal karena tidak dapat menampilkan nama.

Pengujian sudut wajah yang diarahkan ke kiri, lihat tabel 4.2 sebagai berikut ini:

Tabel 4.2 Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Kiri

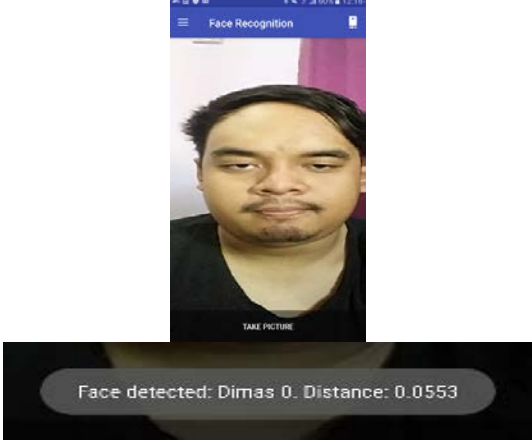

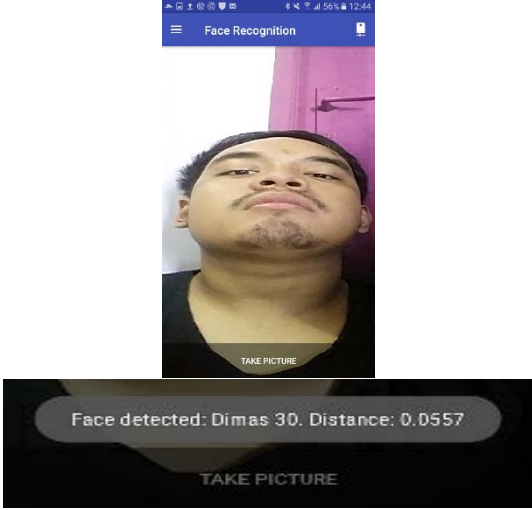
Tabel Pengujian Sudut Wajah yang Diarahkan Ke Kiri		
Sudut ($^\circ$)	Deteksi Wajah	Status
0 $^\circ$		Berhasil
15 $^\circ$		Berhasil

Tabel Pengujian Sudut Wajah yang Diarahkan Ke Kiri		
Sudut (°)	Deteksi Wajah	Status
30°		Berhasil
45°		Gagal

Hasil pengujian deteksi wajah diatas dijelaskan bahwa pengujian sudut 0°, dan 30° berhasil terdeteksi karena terdapat menampilkan sebuah nama dibawah daftar isi label pada aplikasi Android, sedangkan pengujian sudut 15°, dan 45° gagal karena tidak dapat menampilkan nama.

Pengujian sudut wajah yang diarahkan ke atas, lihat tabel 4.3 sebagai berikut ini:

Tabel 4.3 Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Atas

Tabel Pengujian Sudut Wajah yang Diarahkan Ke Atas		
Sudut (°)	Deteksi Wajah	Status
0°		Berhasil
15°		Berhasil
30°		Berhasil

Tabel Pengujian Sudut Wajah yang Diarahkan Ke Atas		
Sudut (°)	Deteksi Wajah	Status
45°		Gagal

Hasil pengujian deteksi wajah diatas dijelaskan bahwa pengujian sudut 0°, 15°, 30°, dan 45° berhasil terdeteksi karena terdapat menampilkan sebuah nama dibawah daftar isi label pada aplikasi Android, sedangkan tidak ada gagal dalam pengujian sudut.

Pengujian sudut wajah yang diarahkan ke bawah, lihat tabel 4.4 sebagai berikut ini:

Tabel 4.4 Tabel Pengujian Sudut Wajah Yang Diarahkan Ke Bawah

Tabel Pengujian Sudut Wajah yang Diarahkan Ke Bawah		
Sudut (°)	Deteksi Wajah	Status
0°		Berhasil

Tabel Pengujian Sudut Wajah yang Diarahkan Ke Bawah		
Sudut (°)	Deteksi Wajah	Status
		
15°		Berhasil
30°		Berhasil
45°		Gagal

Hasil pengujian deteksi wajah diatas dijelaskan bahwa pengujian sudut 0°, 15°, 30°, dan 45° berhasil terdeteksi karena terdapat menampilkan sebuah nama dibawah daftar isi label pada aplikasi Android, sedangkan tidak ada gagal dalam pengujian sudut

4.1.3 Hasil Pengujian Deteksi Wajah dengan Pengujian Aksesoris Tambahan

Pengujian aksesoris tambahan pada wajah yang memakai kacamata dan tidak memakai kacamata, lihat tabel 4.5 sebagai berikut ini:

Tabel 4.5 Tabel Pengujian Aksesoris Tambahan Dengan Kacamata

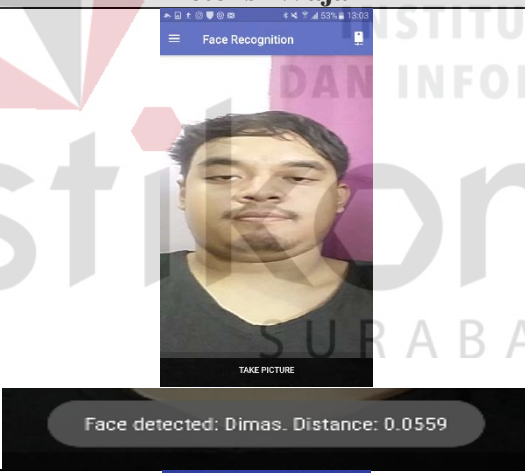
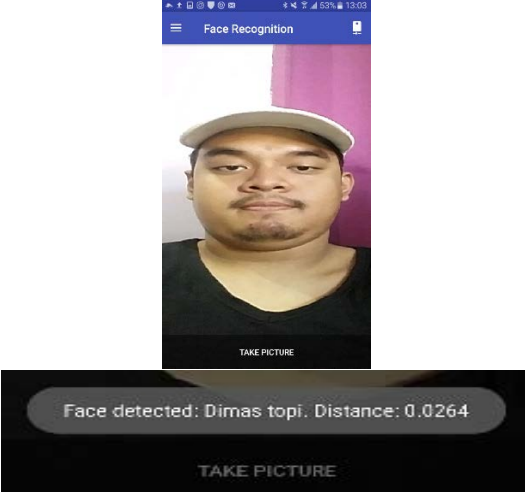
Tabel Pengujian Aksesoris Tambahan Dengan Kacamata		
Kacamata	Deteksi Wajah	Status
Tidak		Berhasil
Ya		Berhasil

Tabel Pengujian Aksesoris Tambahan Dengan Kacamata		
Kacamata	Deteksi Wajah	Status
		

Hasil pengujian aksesoris tambahan pada wajah dengan menggunakan kacamata dan tanpa menggunakan kacamata sudah berhasil terdeteksi wajah dan dapat menampilkan nama.

Pengujian aksesoris tambahan pada wajah yang memakai topi dan tidak memakai topi, lihat tabel 4.6 sebagai berikut ini:

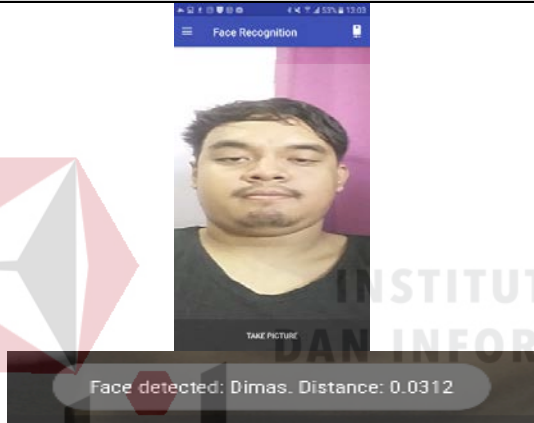

Tabel 4.6 Tabel Pengujian Aksesoris Tambahan Dengan Topi

Tabel Pengujian aksesoris tambahan dengan topi		
Topi	Deteksi Wajah	Status
Tidak		Berhasil
Ya		Berhasil

Hasil pengujian aksesoris tambahan pada wajah dengan menggunakan topi dan tanpa menggunakan topi sudah berhasil terdeteksi wajah dan dapat menampilkan nama.

Pengujian aksesoris tambahan pada wajah yang memakai topi dan kacamata dan tidak memakai topi dan kacamata, lihat tabel 4.7 sebagai berikut ini:

Tabel 4.7 Tabel Pengujian Aksesoris Tambahan Dengan Kacamata Dan Topi


Tabel Pengujian Aksesoris Tambahan Dengan Kacamata Dan Topi		
Kacamata & Topi	Deteksi Wajah	Status
Tidak		Berhasil
Ya		Berhasil

Hasil pengujian aksesoris tambahan pada wajah dengan menggunakan kacamata & topi dan tanpa menggunakan kacamata dan topi sudah berhasil terdeteksi wajah dan dapat menampilkan nama.

4.1.4 Hasil Pengujian Deteksi Wajah dengan Beberapa Sampel

Pengujian pengenalan wajah ini akan diuji terhadap beberapa sampel yang akan didapatkan, lihat tabel 4.8 sebagai berikut ini:


Tabel 4.8 Tabel Pengujian Beberapa Orang Yang Akan Didapatkan

Tabel Pengujian Beberapa Sampel Yang Didapatkan		
Nama	Deteksi Wajah	Status
Dimas		Berhasil
Herman		Berhasil

Tabel Pengujian Beberapa Sampel Yang Didapatkan		
Nama	Deteksi Wajah	Status
Amin	 <p>The screenshot shows a mobile application interface titled 'Face Recognition'. It displays a photo of a man in a green and white shirt. Below the photo, a black overlay box contains the text 'Face detected: Amin. Distance: 0.0715' and a 'TAKE PICTURE' button.</p>	Berhasil
Bagus	 <p>The screenshot shows a mobile application interface titled 'Face Recognition'. It displays a photo of a man in a green and white shirt. Below the photo, a black overlay box contains the text 'Face detected: Bagus. Distance: 0.0206' and a 'TAKE PICTURE' button.</p>	Berhasil

Tabel Pengujian Beberapa Sampel Yang Didapatkan		
Nama	Deteksi Wajah	Status
Darno		Berhasil
Riyan		Berhasil

Tabel Pengujian Beberapa Sampel Yang Didapatkan		
Nama	Deteksi Wajah	Status
Dias	 A screenshot of a mobile application interface for face recognition. The top bar is blue with the text 'Face Recognition' and a camera icon. Below the bar is a photo of a man with dark hair and a beard, wearing a red t-shirt. At the bottom of the photo is a 'TAKE PICTURE' button. Below the photo is a dark grey notification box with white text that reads 'Face detected: Dias. Distance: 0.0645'. <p>Face detected: Dias. Distance: 0.0645</p>	Berhasil
Aris	 A screenshot of a mobile application interface for face recognition. The top bar is blue with the text 'Face Recognition' and a camera icon. Below the bar is a photo of a man with dark hair, wearing a dark jacket over a green shirt. At the bottom of the photo is a 'TAKE PICTURE' button. Below the photo is a dark grey notification box with white text that reads 'Face detected: Aris. Distance: 0.0720'. <p>Face detected: Aris. Distance: 0.0720</p>	Berhasil

Tabel Pengujian Beberapa Sampel Yang Didapatkan		
Nama	Deteksi Wajah	Status
Erlan		Berhasil
Aura		Berhasil

Hasil pengujian pengenalan wajah terhadap beberapa orang yaitu Dimas, Herman, Amin, Bagus, dan Darno, semua tingkat keakuratan yang sangat tinggi dan berhasil menampilkan sebuah nama yang sudah terdeteksi wajah setiap beberapa orang tersebut.

4.1.5 Pengujian Pengenalan Wajah Terhadap Benda atau Tidak Ada Wajah di *Background*

Pengujian pengenalan wajah ini akan diuji terhadap beberapa benda atau tidak ada wajah di *background*, lihat tabel 4.9 sebagai berikut ini:

Tabel 4.9 Tabel Pengujian Pengenalan Wajah Terhadap Benda Atau Tidak Ada Wajah Di *Background*

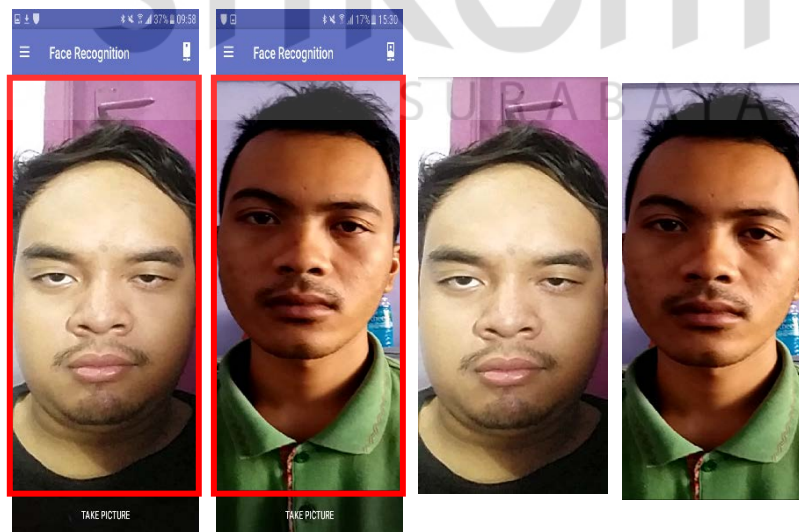
Tabel Pengujian Pengenalan Wajah terhadap Benda atau Tidak Ada Wajah di <i>Background</i>		
Nama	Hasil	Keterangan
Tempat <i>Cotton Pad</i>		Berhasil
Kunci Sepeda Motor		Berhasil

Tabel Pengujian Pengenalan Wajah terhadap Benda atau Tidak Ada Wajah di <i>Background</i>		
Nama	Hasil	Keterangan
<i>Background Pandangan</i>		Berhasil

Hasil pengujian coba terhadap beberapa benda atau tidak ada wajah di background ini sudah berhasil terdeteksi benda atau background dan juga dapat menampilkan sebuah nama tersebut.

4.2 Perhitungan dengan Metode *Eigenface* secara *Detail* atau *Lengkap*

1. Foto *Original*



2. Foto *Grayscale* / Abu-abu



3. Foto diperkecil dengan ukuran yang sama



4. Normalisasi pada kedua foto wajah



- 1.) Membuat *data* tabel yang digunakan untuk melanjutkan perhitungan metode *PCA (PRINCIPAL COMPONENT ANALYSIS)*. Dengan *random generate data table* (acak-acak angka).



9	7	3
2	1	4
5	8	9



6	9	4
2	3	5
1	9	8

- 2.) Mengubah *data* tabel menjadi matriks yang berukuran 3 x 3. Dengan nama variabel Gambar1 (foto pertama Dimas) dan Gambar2 (foto kedua Darno), sebagai berikut:

$$\text{Gambar1} = \begin{bmatrix} 9 & 7 & 3 \\ 2 & 1 & 4 \\ 5 & 8 & 9 \end{bmatrix}, \text{Gambar2} = \begin{bmatrix} 6 & 9 & 4 \\ 2 & 3 & 5 \\ 1 & 9 & 8 \end{bmatrix}$$

- 3.) Misalkan *data* pada Gambar1 dan Gambar2 merupakan sampel dari citra pelatihan, maka hal pertama yang dilakukan adalah merubah dari matriks menjadi vektor kolom yang dijadikan satu menjadi sebuah *dataset* dengan nama variable *Dataset*.

$$\text{Dataset} = \begin{bmatrix} 9 & 6 \\ 7 & 9 \\ 3 & 4 \\ 2 & 2 \\ 1 & 3 \\ 4 & 5 \\ 5 & 1 \\ 8 & 9 \\ 9 & 8 \end{bmatrix}$$

- 4.) Selanjutnya dilakukan *Zero Mean* terhadap *data* diatas, dengan terlebih dahulu mencari vektor rata-rata dari *dataset* diatas. Dengan nama variabel untuk vektor rata-rata (*mean*) = X, *Zero Mean* = Y sebagai berikut:

$$X = \begin{bmatrix} \frac{9+6}{2} = 7.5 \\ \frac{7+9}{2} = 8 \\ \frac{3+4}{2} = 3.5 \\ \frac{2+2}{2} = 2 \\ \frac{1+3}{2} = 2 \\ \frac{4+5}{2} = 4.5 \\ \frac{5+1}{2} = 3 \\ \frac{8+9}{2} = 8.5 \\ \frac{9+8}{2} = 8.5 \end{bmatrix} ; Y = \text{Dataset} - X = \begin{bmatrix} 9 - 7.5 = 1.5 & 6 - 7.5 = -1.5 \\ 7 - 8 = -1 & 9 - 8 = 1 \\ 3 - 3.5 = -0.5 & 4 - 3.5 = 0.5 \\ 2 - 2 = 0 & 2 - 2 = 0 \\ 1 - 2 = -1 & 3 - 2 = 1 \\ 4 - 4.5 = -0.5 & 5 - 4.5 = 0.5 \\ 5 - 3 = 2 & 1 - 3 = -2 \\ 8 - 8.5 = -0.5 & 9 - 8.5 = 0.5 \\ 9 - 8.5 = 0.5 & 8 - 8.5 = -0.5 \end{bmatrix}$$

5.) Setelah didapatkan matrik *Zero Mean* (Y) dilakukan perhitungan untuk mendapatkan matriks kovarian. Dengan nama variabel Kovarian = A, sebagai berikut:

$$A = Y^T * Y$$

$$A = \begin{bmatrix} 1.5 & -1 & -0.5 & 0 & -1 & -0.5 & 2 & -0.5 & 0.5 \\ -1.5 & 1 & 0.5 & 0 & 1 & 0.5 & -2 & 0.5 & -0.5 \end{bmatrix} * \begin{bmatrix} 1.5 & -1.5 \\ -1 & 1 \\ -0.5 & 0.5 \\ 0 & 0 \\ -1 & 1 \\ -0.5 & 0.5 \\ 2 & -2 \\ -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

$$A = \begin{bmatrix} 9.25 & -9.25 \\ -9.25 & 9.25 \end{bmatrix}$$

6.) Kemudian dicari nilai *eigen* (λ) dari matrik kovarian (A) diatas:

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{\text{Matriks}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\text{Vektor Eigen}} = \underbrace{k}_{\text{Nilai Eigen}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\text{Vektor Eigen}}$$

Misalkan: $A = \text{Matriks}$
 $v = \text{Vektor Eigen}$
 $\lambda = \text{Nilai Eigen}$

Maka dapat dirumuskan

$$Av = \lambda v$$

Nilai Eigen dikali dengan Matriks Identitas, maka:

$$Av = \lambda Iv$$

$$(\lambda I - A)v = 0$$

Mencari Nilai Eigen (λ)

$$A = \begin{bmatrix} 9.25 & -9.25 \\ -9.25 & 9.25 \end{bmatrix}$$

$$(\lambda I - A)v = 0 \Leftarrow \text{Persamaan Karakteristik}$$

$$\left(\lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 9.25 & -9.25 \\ -9.25 & 9.25 \end{bmatrix} \right) v = 0 \Leftarrow \text{Matriks } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 9.25 & -9.25 \\ -9.25 & 9.25 \end{bmatrix} \right) v = 0$$

$$\left(\begin{bmatrix} \lambda - 9.25 & 9.25 \\ 9.25 & \lambda - 9.25 \end{bmatrix} \right) v = 0$$

Untuk menentukan nilai λ yang skalar, berlaku :

$$\det(\lambda I - A) = 0$$

$$\det \begin{vmatrix} \lambda - 9.25 & 9.25 \\ 9.25 & \lambda - 9.25 \end{vmatrix} = 0 \Leftarrow \text{Determinan Matriks Ordo } 2 \times 2 = (ad - bc)$$

$$((\lambda - 9.25)(\lambda - 9.25) - (9.25)(9.25)) = 0$$

$$(\lambda^2 - 18.5\lambda + 85.56 - 85.56) = 0$$

$$(\lambda^2 - 18.5\lambda) = 0$$

$$(\lambda + 9.25)(\lambda - 9.25) = 0$$

$$\lambda = 9.25 \text{ atau } \lambda = -9.25 \Leftarrow \text{Nilai Eigen}$$

Dari hasil Nilai *Eigen* yang didapati adalah $\lambda = 9.25$ atau $\lambda = -9.25$

7.) Selanjutnya mencari nilai *eigen* vektor dari nilai *eigen* diatas sebagai berikut :

$$(\lambda I - A)v = 0 \Leftarrow \text{Persamaan Karakteristik}$$

$$\left(\begin{bmatrix} \lambda - 9.25 & 9.25 \\ 9.25 & \lambda - 9.25 \end{bmatrix} \right) v = 0$$

$$\begin{bmatrix} \lambda - 9.25 & 9.25 \\ 9.25 & \lambda - 9.25 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

Untuk nilai $\lambda = 9.25$, maka :

$$\begin{bmatrix} 9.25 - 9.25 & 9.25 \\ 9.25 & 9.25 - 9.25 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 9.25 \\ 9.25 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

Terbentuk sistem persamaan linear

$$9.25v_2 = 0$$

$$9.25v_1 = 0$$

Diperoleh :

$$v_1 = 9.25 \text{ dan } v_2 = 9.25$$

Pembuktian :

$$Av = \lambda v$$

$$\begin{bmatrix} 0 & 9.25 \\ 9.25 & 0 \end{bmatrix} \begin{bmatrix} 9.25 \\ 9.25 \end{bmatrix} = \begin{bmatrix} 85.56 \\ 85.56 \end{bmatrix} = 9.25 \begin{bmatrix} 9.25 \\ 9.25 \end{bmatrix} \Leftarrow \text{Terbukti}$$

Untuk nilai $\lambda = -9.25$, maka :

$$\begin{bmatrix} -9.25 - 9.25 & 9.25 \\ 9.25 & -9.25 - 9.25 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} -18.5 & 9.25 \\ 9.25 & -18.5 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

Terbentuk sistem persamaan linear

$$-18.5v_1 + 9.25v_2 = 0$$

$$9.25v_1 - 18.5v_2 = 0$$

Diperoleh :

$$v_1 = -9.25 \text{ dan } v_2 = -9.25$$

Pembuktian :

$$Av = \lambda v$$

$$\begin{bmatrix} -18.5 & 9.25 \\ 9.25 & -18.5 \end{bmatrix} \begin{bmatrix} -9.25 \\ -9.25 \end{bmatrix} = \begin{bmatrix} 85.56 \\ 85.56 \end{bmatrix} = 9.25 \begin{bmatrix} 9.25 \\ 9.25 \end{bmatrix} \Leftarrow \text{Terbukti}$$

4.2.1 Perhitungan dengan Metode *Eigenface* pada Kedua Gambar Wajah yang Sama dan *Data* Tabel juga Sama



9	7	3
2	1	4
5	8	9



9	7	3
2	1	4
5	8	9

- 1.) Mengubah *data* tabel menjadi matriks yang berukuran 3 x 3. Dengan nama variabel Gambar1 (foto pertama Dimas) dan Gambar2 (foto kedua Dimas), sebagai berikut :

$$\text{Gambar1} = \begin{bmatrix} 9 & 7 & 3 \\ 2 & 1 & 4 \\ 5 & 8 & 9 \end{bmatrix}, \text{Gambar2} = \begin{bmatrix} 9 & 7 & 3 \\ 2 & 1 & 4 \\ 5 & 8 & 9 \end{bmatrix}$$

- 2.) Misalkan *data* pada Gambar1 dan Gambar2 merupakan sampel dari citra pelatihan, maka hal pertama yang dilakukan adalah merubah dari matriks

menjadi vektor kolom yang dijadikan satu menjadi sebuah *dataset* dengan nama variabel *Dataset*.

$$Dataset = \begin{bmatrix} 9 & 9 \\ 7 & 7 \\ 3 & 3 \\ 2 & 2 \\ 1 & 1 \\ 4 & 4 \\ 5 & 5 \\ 8 & 8 \\ 9 & 9 \end{bmatrix}$$

- 3.) Selanjutnya dilakukan *Zero Mean* terhadap data diatas, dengan terlebih dahulu mencari vektor rata-rata dari dataset diatas. Dengan nama variabel untuk vektor rata-rata (*mean*) = X, *Zero Mean* = Y sebagai berikut:

$$X = \begin{bmatrix} \frac{9+9}{2} = 9 \\ \frac{7+7}{2} = 7 \\ \frac{3+3}{2} = 3 \\ \frac{2+2}{2} = 2 \\ \frac{1+1}{2} = 1 \\ \frac{4+4}{2} = 4 \\ \frac{5+5}{2} = 5 \\ \frac{8+8}{2} = 8 \\ \frac{9+9}{2} = 9 \end{bmatrix}; Y = Dataset - X = \begin{bmatrix} 9-9=0 & 9-9=0 \\ 7-7=0 & 7-7=0 \\ 3-3=0 & 3-3=0 \\ 2-2=0 & 2-2=0 \\ 1-1=0 & 1-1=0 \\ 4-4=0 & 4-4=0 \\ 5-5=0 & 5-5=0 \\ 8-8=0 & 8-8=0 \\ 9-9=0 & 9-9=0 \end{bmatrix}$$

- 4.) Setelah didapatkan matrik *Zero Mean* (Y) dilakukan perhitungan untuk mendapatkan matriks kovarian. Dengan nama variabel Kovarian = A, sebagai berikut:

$$A = Y^T * Y$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

5.) Kemudian dicari nilai *eigen* (λ) dari matrik kovarian (A) diatas:

Misalkan: A = Matriks
v = Vektor Eigen
λ = Nilai Eigen

Maka dapat dirumuskan

$$Av = \lambda v$$

Nilai Eigen dikali dengan Matriks Identitas, maka:

$$Av = \lambda v$$

$$(\lambda I - A)v = 0$$

Mencari Nilai Eigen (λ)

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$(\lambda I - A)v = 0 \Leftarrow \text{Persamaan Karakteristik}$$

$$\left(\lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)v = 0 \Leftarrow \text{Matriks } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)v = 0$$

$$\left(\begin{bmatrix} \lambda - 0 & 0 \\ 0 & \lambda - 0 \end{bmatrix}\right)v = 0$$

Untuk menentukan nilai λ yang skalar, berlaku :

$$\det(\lambda I - A) = 0$$

$$\det \begin{vmatrix} \lambda - 0 & 0 \\ 0 & \lambda - 0 \end{vmatrix} = 0 \iff \text{Determinan Matriks Ordo } 2 \times 2 = (ad - bc)$$

$$((\lambda - 0)(\lambda - 0) - (0)(0)) = 0$$

$$(\lambda^2 - 0\lambda + 0 - 0) = 0$$

$$(\lambda^2 - 0\lambda + 0) = 0$$

$$(\lambda + 0)(\lambda - 0) = 0$$

$$\lambda = 0 \text{ atau } \lambda = 0 \iff \text{Nilai Eigen}$$

Dari hasil Nilai Eigen yang didapati adalah $\lambda = 0$ atau $\lambda = 0$

Kesimpulan jika kedua foto yang berbeda (Dimas dan Darno) maka nilai *eigen* kecil, dan jika kedua foto yang sama (Dimas dan Dimas) maka nilai *eigen* nol (0).

4.3 Perhitungan dengan Metode *Eigenface* secara *Simple* atau Sederhana

4.3.1 Perubahan Gambar Asli atau *RGB* menjadi Gambar *Grayscale* atau Abu-abu

Data pengenalan wajah ini akan diekstrasi menjadi beberapa tabel *data* (misal nilai 1 ialah dapat menampilkan warna sedangkan nilai 0 ialah dapat menampilkan *grayscale* / abu-abu).

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1



Gambar 4.3 Tampilan *Data* Pada Gambar Wajah Yang Pertama Dengan Warna Atau *RGB*

Kemudian gambar wajah warna asli atau *RGB* yang akan diubah menjadi *grayscale* / abu-abu yang bernilai 0.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0



Gambar 4.4 Tampilan *Data* Pada Gambar Wajah Yang Pertama Dengan Abu-Abu Atau *Grayscale*

4.3.2 Perhitungan *Training Image* dengan Metode *Flatvector*

Tahap ini adalah semua *training image* menjadi satu matriks. Misal *image* yang sudah disimpan pada *database* di laptop menggunakan *software Android studio* dengan berukuran $x \cdot y$ dan bertipe *pixel* dan jumlah a *image*, maka dari perhitungan matriks tersebut memiliki *vector* dengan ciri dimensi $a \cdot (x \cdot y)$. Jika *training image* ini terdapat 2 *image* yang memiliki ukuran 4×4 matriks yang bertipe

4.) Menghitung Rata-rata (*average*)

Dari vektor ini memiliki ciri yang telah didapatkan data tabel sehingga seluruh baris matriks yang berukuran $1 \cdot (x \cdot y)$. setelah ini melakukan pembagian terhadap matriks tersebut dengan jumlah a untuk menghasilkan nilai rata-rata *vector* seperti dibawah ini:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Kedua Matriks akan dijumlahkan sehingga menjadi

3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Kemudian hasilnya dibagi dua sehingga menjadi

1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Hasil Rata-rata *Flatvector* setiap matriks ialah bernilai 1,5

5.) Perubahan gambar wajah menjadi *facespace*

Sebuah gambar diubah menjadi *facespace* dengan melakukan operasi perkalian menggunakan metode *Eigenface*. Perubahan vektor pada gambar wajah yang dilakukan dengan perbandingan vektor yang sesuai. Menggunakan nilai rata-rata pada vektor seperti data tabel diatas tersebut akan dihitung *Eigenface* untuk matriks vektor yang telah disusun. Setiap *image* yang akan mengurangi baris-baris pada matriks vektor dengan nilai rata-rata vektor. Seperti ilustrasi pada tabel dibawah ini :

a.) Gambar wajah pertama

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5	1,5

Matriks pada gambar wajah yang pertama akan dikurangi dengan hasil rata-rata

Flatvector sehingga menjadi:

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

b.) Gambar wajah kedua

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,	1,
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Matriks pada gambar wajah yang kedua akan dikurangi dengan hasil rata-rata *Flatvector* sehingga menjadi:

0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

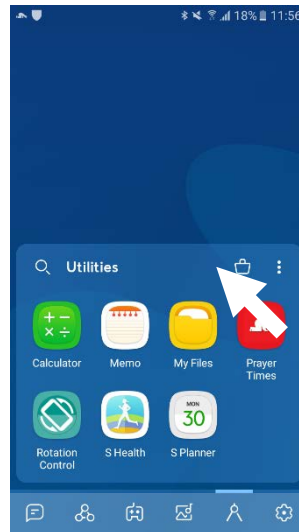
Gambar wajah yang kedua dihasilkan nilai 0,5

4.3.3 Perhitungan Ekstrasi PCA

Hasil perhitungan tersebut ini akan dieksekusi dengan perhitungan menggunakan metode *PCA* untuk mendapatkan fitur dari gambar, fitur ini penting dari sebuah *training image* yang didapatkan dari proses *training*. Tahapan ini fitur akan digunakan untuk mengklarifikasi *image* yang akan dikenali. Klasifikasi nilai *Eigenface* untuk menentukan sebuah matriks *testface*. Misalnya gambar wajah yang pertama dengan bernilai 9 untuk semua sisi sedangkan yang paling tengah adalah bernilai 1 dan gambar wajah yang kedua dengan bernilai 6 untuk semua sisi sedangkan yang paling tengah adalah bernilai 3, kemudian hasil data tabel tersebut akan dikurangkan dengan nilai rata-rata *flatvector*. Seperti dibawah ini sebagai berikut:

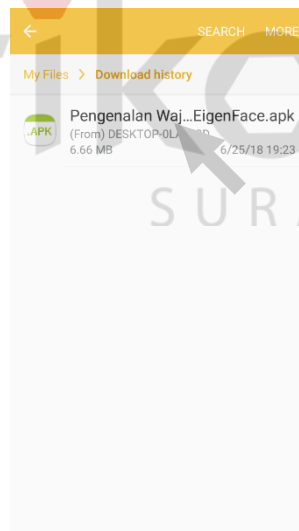
4.4 Menguji Coba Aplikasi *Android*

1.) Membuka aplikasi “*My Files*” di *Android*.



Gambar 4.9 Tampilan Layar Utama Pada *Android*

2.) Meng-klik *File* yang bernama “Pengenalan Wajah dengan Metode *Eigen Face* .apk” di dalam aplikasi “*MyFiles*”, menunggu proses instalasi sampai selesai.



Gambar 4.10 Tampilan Aplikasi “*MyFiles*” pada *Android*

3.) Membuka aplikasi “Pengenalan Wajah dengan Metode *Eigen Face*”.



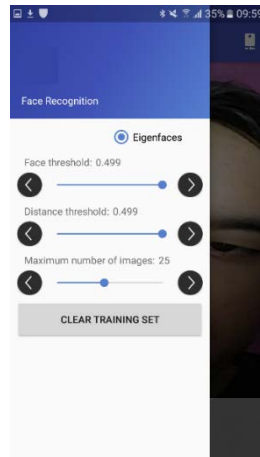
Gambar 4.11 Tampilan Layar Utama Pada *Android*

4.) Meng-klik pada *menu* untuk mengatur jarak dan wajah *threshold* pada aplikasi “Pengenalan Wajah dengan Metode *Eigen Face*”.



Gambar 4.12 Tampilan Layar Utama Pada Aplikasi “Pengenalan Wajah Dengan Metode *Eigen Face*”

- 5.) Menggeser pada *slidebar* dengan *face threshold* yang bernilai 500 (499) dan *distance threshold* yang bernilai 500 (499) agar dapat mendeteksi pengenalan wajah dengan akurasi yang tinggi.



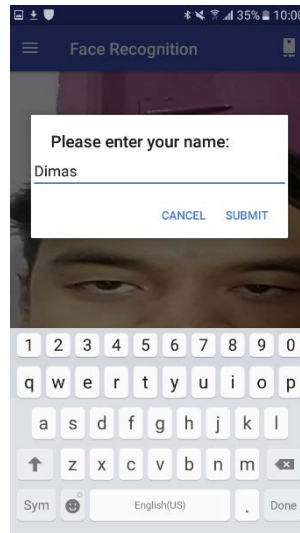
Gambar 4.13 Tampilan Mengatur Nilai *Face Threshold* Dan *Distance Threshold* Pada Aplikasi “Pengenalan Wajah Dengan Metode *Eigen Face*”

- 6.) Untuk mengambil gambar pertama, menggerakkan wajah yang tampak ke depan dan tidak memakai kacamata agar menguji coba gambar yang pertama.



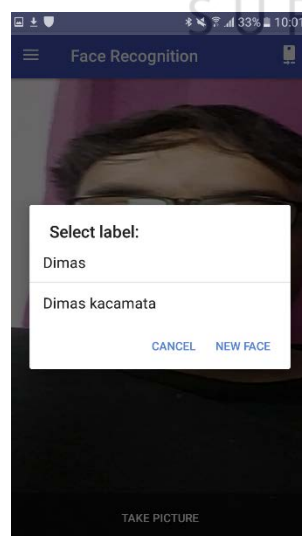
Gambar 4.14 Tampilan Mengambil Gambar Wajah Yang Pertama

- 7.) Setelah pencet tombol *“take picture”* kemudian mengisi sebuah nama *“Dimas”* yang akan disimpan di memori agar dapat mengetahui siapa gambar wajah yang dimiliki.



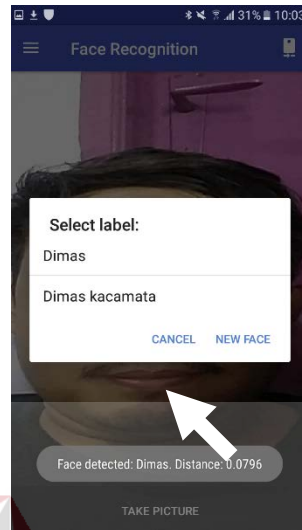
Gambar 4.15 Tampilan Mengisi Sebuah Nama Yang Diinginkan

- 8.) Setelah mendapatkan gambar wajah yang pertama kemudian klik *“New Face”* untuk mengambil gambar wajah yang kedua dengan memakai kacamata seperti gambar dibawah ini.



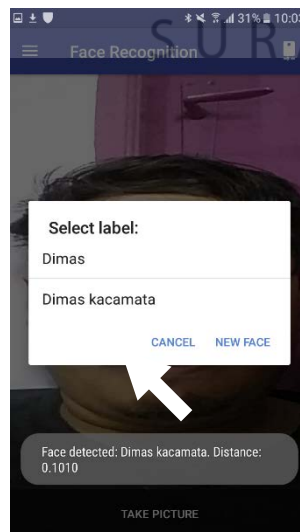
Gambar 4.16 Tampilan Daftar Isi Pengenalan Wajah

- 9.) Setelah membuat dua gambar wajah, selanjutnya menguji coba wajah yang pertama yang tidak memakai kacamata kemudian telah berhasil mendeteksi wajah dan dapat menampilkan sebuah nama “Dimas”.



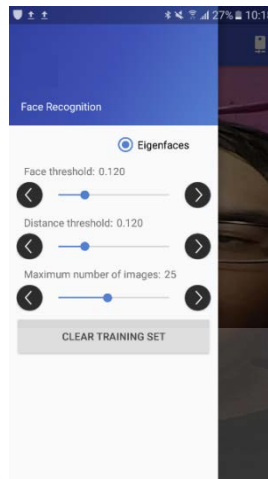
Gambar 4.17 Tampilan Mendeteksi Gambar Wajah Yang Pertama

- 10.) Menguji coba wajah yang kedua juga memakai kacamata kemudian telah berhasil mendeteksi wajah dan dapat menampilkan sebuah nama “Dimas Kacamata”.



Gambar 4.18 Tampilan Mendeteksi Gambar Wajah Yang Kedua

- 11.) Setelah menguji coba yang pertama dengan akurasi pengenalan wajah yang tinggi kemudian mengubah dan mengatur *Face Threshold* yang bernilai 120 dan *Distance Threshold* yang bernilai 120. Akurasi pengenalan wajah yang sangat rendah.



Gambar 4.19 Tampilan Mengatur Nilai *Face Threshold* Dan *Distance Threshold* Pada Aplikasi “Pengenalan Wajah Dengan Metode *Eigen Face*”

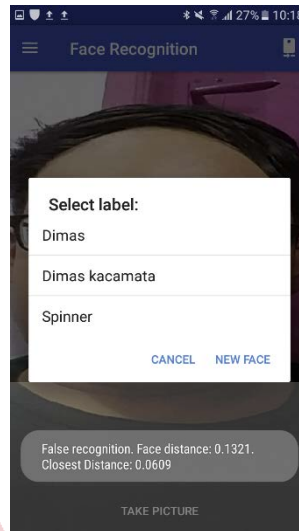
- 12.) Mengambil gambar wajah yang pertama dengan tidak memakai kacamata



Gambar 4.20 Mengambil Gambar Wajah Yang Tidak Memakai Kacamata

- 13.) Setelah mengambil wajah yang tidak memakai kacamata dan mengisi sebuah nama “Dimas” lalu menguji coba pengenalan wajah juga tidak bisa

mendeteksi wajah dan tidak dapat menampilkan sebuah nama karena kurang jelas akurasi pengenalan wajah dan sangat berbeda dengan akurasi pengenalan wajah yang tinggi



Gambar 4.21 Tampilan Mendeteksi Wajah Yang Tidak Dapat Menampilkan Nama Karena *Error*

4.4.1 Tabel Hasil Pengujian Coba Pengenalan Wajah

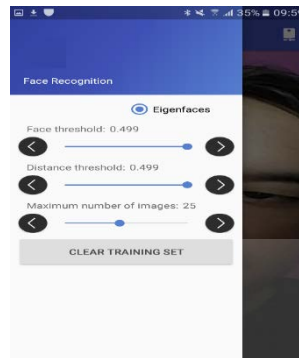
Hasil pengujian coba dalam aplikasi “pengenalan wajah dengan metode *Eigenface*“, lihat tabel 4.11 dibawah ini sebagai berikut:

Tabel 4.11 Hasil Pengujian Coba Dalam Aplikasi Tersebut

Nama	Aksurasi Pengenalan Wajah		Gambar 1 Wajah yang tidak memakai kacamata	Gambar 2 Wajah yang memakai kacamata
	<i>Face Threshold</i>	<i>Distance Threshold</i>		
Rendah	120	120	Tidak (<i>Error</i>)	Tidak (<i>Error</i>)
Tinggi	499 (500)	499 (500)	Ya	Ya

4.5 Pengajian Coba Pengenalan Wajah dengan Menentukan Beberapa Nilai *Face Threshold* dan Nilai *Distance Threshold*

Pengajian coba ini dilakukan dengan mengatur keakuratan atau akurasi pengenalan wajah menggunakan nilai *face threshold* dan *distance threshold*. Dapat dilihat pada gambar 4.22 dan dengan tabel 4.12. dibawah ini sebagai berikut :



Gambar 4.22 Tampilan Nilai *Face Threshold* dan *Distance Threshold*

Tabel 4.12 Tabel Pengajian Pengenalan Wajah Dengan Mengatur Nilai *Face Threshold* Dan Nilai *Distance Threshold*

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
No	Keakuratan atau Akurasi		Mengatur Nilai Akurasi / Keakuratan		Hasil
	<i>Face Threshold</i>	<i>Distance Threshold</i>	<i>Face Threshold</i>	<i>Distance Threshold</i>	
1	Tidak ada	Tidak ada	0	0	Gagal
2	Tidak ada	Rendah	0	50	Gagal
3	Tidak ada	Rendah	0	100	Gagal
4	Tidak ada	Rendah	0	150	Gagal
5	Tidak ada	Rendah	0	200	Gagal
6	Tidak ada	Normal	0	250	Gagal
7	Tidak ada	Normal	0	300	Gagal
8	Tidak ada	Normal	0	350	Gagal
9	Tidak ada	Tinggi	0	400	Gagal

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
No	Keakuratan atau Akurasi		Mengatur Nilai Akurasi / Keakuratan		Hasil
	Face Threshold	Distance Threshold	Face Threshold	Distance Threshold	
10	Tidak ada	Tinggi	0	450	Gagal
11	Tidak ada	Tinggi	0	500	Gagal
12	Rendah	Tidak ada	50	0	Gagal
13	Rendah	Rendah	50	50	Gagal
14	Rendah	Rendah	50	100	Gagal
15	Rendah	Rendah	50	150	Gagal
16	Rendah	Rendah	50	200	Gagal
17	Rendah	Normal	50	250	Gagal
18	Rendah	Normal	50	300	Gagal
19	Rendah	Normal	50	350	Gagal
20	Rendah	Tinggi	50	400	Gagal
21	Rendah	Tinggi	50	450	Gagal
22	Rendah	Tinggi	50	500	Gagal
23	Rendah	Tidak ada	100	0	Gagal
24	Rendah	Rendah	100	50	Gagal
25	Rendah	Rendah	100	100	Gagal
26	Rendah	Rendah	100	150	Gagal
27	Rendah	Rendah	100	200	Gagal
28	Rendah	Normal	100	250	Gagal
29	Rendah	Normal	100	300	Gagal
30	Rendah	Normal	100	350	Gagal
31	Rendah	Tinggi	100	400	Gagal
32	Rendah	Tinggi	100	450	Gagal
33	Rendah	Tinggi	100	500	Berhasil
34	Rendah	Tidak ada	150	0	Gagal
35	Rendah	Rendah	150	50	Gagal

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
No	Keakuratan atau Akurasi		Mengatur Nilai Akurasi / Keakuratan		Hasil
	Face Threshold	Distance Threshold	Face Threshold	Distance Threshold	
36	Rendah	Rendah	150	100	Gagal
37	Rendah	Rendah	150	150	Gagal
38	Rendah	Rendah	150	200	Gagal
39	Rendah	Normal	150	250	Gagal
40	Rendah	Normal	150	300	Gagal
41	Rendah	Normal	150	350	Gagal
42	Rendah	Tinggi	150	400	Gagal
43	Rendah	Tinggi	150	450	Gagal
44	Rendah	Tinggi	150	500	Berhasil
45	Rendah	Tidak ada	200	0	Gagal
46	Rendah	Rendah	200	50	Gagal
47	Rendah	Rendah	200	100	Gagal
48	Rendah	Rendah	200	150	Gagal
49	Rendah	Rendah	200	200	Gagal
50	Rendah	Normal	200	250	Gagal
51	Rendah	Normal	200	300	Gagal
52	Rendah	Normal	200	350	Gagal
53	Rendah	Tinggi	200	400	Berhasil
54	Rendah	Tinggi	200	450	Berhasil
55	Rendah	Tinggi	200	500	Berhasil
56	Normal	Tidak ada	250	0	Gagal
57	Normal	Rendah	250	50	Gagal
58	Normal	Rendah	250	100	Gagal
59	Normal	Rendah	250	150	Gagal
60	Normal	Rendah	250	200	Gagal
61	Normal	Normal	250	250	Berhasil

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
No	<i>Keakuratan atau Akurasi</i>		<i>Mengatur Nilai Akurasi / Keakuratan</i>		<i>Hasil</i>
	<i>Face Threshold</i>	<i>Distance Threshold</i>	<i>Face Threshold</i>	<i>Distance Threshold</i>	
62	Normal	Normal	250	300	Berhasil
63	Normal	Normal	250	350	Berhasil
64	Normal	Tinggi	250	400	Berhasil
65	Normal	Tinggi	250	450	Berhasil
66	Normal	Tinggi	250	500	Berhasil
67	Normal	Tidak ada	300	0	Gagal
68	Normal	Rendah	300	50	Gagal
69	Normal	Rendah	300	100	Gagal
70	Normal	Rendah	300	150	Gagal
71	Normal	Rendah	300	200	Gagal
72	Normal	Normal	300	250	Berhasil
73	Normal	Normal	300	300	Berhasil
74	Normal	Normal	300	350	Berhasil
75	Normal	Tinggi	300	400	Berhasil
76	Normal	Tinggi	300	450	Berhasil
77	Normal	Tinggi	300	500	Berhasil
78	Normal	Tidak ada	350	0	Gagal
79	Normal	Rendah	350	50	Gagal
80	Normal	Rendah	350	100	Gagal
81	Normal	Rendah	350	150	Gagal
82	Normal	Rendah	350	200	Gagal
83	Normal	Normal	350	250	Berhasil
84	Normal	Normal	350	300	Berhasil
85	Normal	Normal	350	350	Berhasil
86	Normal	Tinggi	350	400	Berhasil
87	Normal	Tinggi	350	450	Berhasil

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
<i>No</i>	<i>Keakuratan atau Akurasi</i>		<i>Mengatur Nilai Akurasi / Keakuratan</i>		<i>Hasil</i>
	<i>Face Threshold</i>	<i>Distance Threshold</i>	<i>Face Threshold</i>	<i>Distance Threshold</i>	
88	Normal	Tinggi	350	500	Berhasil
89	Tinggi	Tidak ada	400	0	Gagal
90	Tinggi	Rendah	400	50	Gagal
91	Tinggi	Rendah	400	100	Gagal
92	Tinggi	Rendah	400	150	Gagal
93	Tinggi	Rendah	400	200	Gagal
94	Tinggi	Normal	400	250	Berhasil
95	Tinggi	Normal	400	300	Berhasil
96	Tinggi	Normal	400	350	Berhasil
97	Tinggi	Tinggi	400	400	Berhasil
98	Tinggi	Tinggi	400	450	Berhasil
99	Tinggi	Tinggi	400	500	Berhasil
100	Tinggi	Tidak ada	450	0	Gagal
101	Tinggi	Rendah	450	50	Gagal
102	Tinggi	Rendah	450	100	Gagal
103	Tinggi	Rendah	450	150	Gagal
104	Tinggi	Rendah	450	200	Gagal
105	Tinggi	Normal	450	250	Berhasil
106	Tinggi	Normal	450	300	Berhasil
107	Tinggi	Normal	450	350	Berhasil
108	Tinggi	Tinggi	450	400	Berhasil
109	Tinggi	Tinggi	450	450	Berhasil
110	Tinggi	Tinggi	450	500	Berhasil
111	Tinggi	Tidak ada	500	0	Gagal
112	Tinggi	Rendah	500	50	Gagal
113	Tinggi	Rendah	500	100	Gagal

Tabel pengajian pengenalan wajah dengan mengatur nilai <i>Face Threshold</i> dan nilai <i>Distance Threshold</i>					
No	<i>Keakuratan atau Akurasi</i>		<i>Mengatur Nilai Akurasi / Keakuratan</i>		Hasil
	<i>Face Threshold</i>	<i>Distance Threshold</i>	<i>Face Threshold</i>	<i>Distance Threshold</i>	
114	Tinggi	Rendah	500	150	Gagal
115	Tinggi	Rendah	500	200	Gagal
116	Tinggi	Normal	500	250	Berhasil
117	Tinggi	Normal	500	300	Berhasil
118	Tinggi	Normal	500	350	Berhasil
119	Tinggi	Tinggi	500	400	Berhasil
120	Tinggi	Tinggi	500	450	Berhasil
121	Tinggi	Tinggi	500	500	Berhasil

Hasil Kesimpulan bahwa nilai *Face Threshold* dan *Distance Threshold* memiliki akurasi yang paling tinggi yaitu diantara nilai *Face Threshold* yang bernilai 100 sampai dengan 500 dan nilai *Distance Threshold* yang bernilai 250 sampai dengan 500 ini dapat mendeteksi wajah atau pengenalan wajah yang telah terdeteksi yang sangat jelas dan dapat menampilkan sebuah nama setiap gambar wajah. Sedangkan akurasi yang paling rendah yaitu diantara nilai *Face Threshold* yang bernilai 0 sampai dengan 100 dan nilai *Distance Threshold* yang bernilai 0 sampai dengan 250 ini tidak dapat mendeteksi wajah atau pengenalan wajah karena kurang jelas deteksi dan juga tidak dapat menampilkan sebuah nama setiap gambar wajah tersebut.

4.6 Hasil Pengujian Keseluruhan

Tabel hasil pengujian berdasarkan keseluruhan pengujian yang telah didapatkan sebagai berikut:

Tabel Hasil Pengujian Berdasarkan Keseluruhan Pengujian			
Foto	Wajah yang Terdeteksi	Dikenali	Keterangan
 <p>Dimas</p>	Terdeteksi	Dimas	Berhasil
 <p>Dimas</p>	Terdeteksi	Erlian	Gagal
 <p>Erlian</p>	Terdeteksi	Erlian	Berhasil

Tabel Hasil Pengujian Berdasarkan Keseluruhan Pengujian			
Foto	Wajah yang Terdeteksi	Dikenali	Keterangan
 <p>Erlan</p>	Terdeteksi	Dimas	Gagal
 <p>Herman</p>	terdeteksi	Herman	Berhasil
 <p>Amin</p>	Terdeteksi	Amin	Berhasil

Tabel Hasil Pengujian Berdasarkan Keseluruhan Pengujian			
Foto	Wajah yang Terdeteksi	Dikenali	Keterangan
 <p>Amin</p>	Terdeteksi	Herman	Gagal
 <p>Darno</p>	Terdeteksi	Darno	Berhasil
 <p>Riyan</p>	Terdeteksi	Riyan	Berhasil

Tabel Hasil Pengujian Berdasarkan Keseluruhan Pengujian			
Foto	Wajah yang Terdeteksi	Dikenali	Keterangan
<p>Riyan</p>  <p>Dias</p>	Terdeteksi	Dias	Berhasil
 <p>Aris</p>	Terdeteksi	Aris	Berhasil

Tabel Hasil Pengujian Berdasarkan Keseluruhan Pengujian			
Foto	Wajah yang Terdeteksi	Dikenali	Keterangan
 <p>Aura</p>	Terdeteksi	Aura	Berhasil
 <p>Bagus</p>	Terdeteksi	Bagus	Berhasil

Kesimpulan dalam pengujian keseluruhan pengenalan wajah ini terdapat 13 gambar yang didapati, dibedakan menjadi dua gambar yaitu gambar yang berhasil dan gambar yang gagal. Gambar yang berhasil mendeteksi wajah dan dapat mengenali sebuah nama yang berjumlah 10 sampel gambar tersebut, sedangkan gambar yang gagal tidak dapat mendeteksi wajah dan juga tidak bisa mencocokkan

sebuah nama yang tidak sama dengan nama yang lain ini tersapat berjumlah 3 sampel gambar.



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Ada beberapa cara mengimplementasikan pengenalan wajah pada *smartphone* menggunakan metode *eigenface* sebagai berikut:

1. Pengenalan wajah pada aplikasi *android* yang bernama “Pengenalan Wajah dengan Metode *Eigenface*”, memiliki banyak fitur yaitu fitur *Face Threshold* untuk mengatur keakuratan wajah, fitur *Distance Threshold* untuk mengatur keakuratan jarak terhadap wajah yang tepat, dan fitur jumlah gambar untuk melakukan *training image* setiap beberapa gambar yang sudah didapatkan kemudian hasil gambar wajah yang paling benar.
2. Pengenalan wajah ini mampu mendeteksi wajah dengan posisi yang tampak frontal (tampak depan) dengan memakai *background* atau tanpa *background* juga bisa menampilkan sebuah nama setiap gambar wajah yang berbeda atau memakai aksesoris yang berbeda.
3. Untuk mengatur keakuratan wajah yang sangat jelas dan tinggi yaitu dengan nilai *Face Threshold* maksimum 499 (500) dan nilai *Distance Threshold* maksimum 499 (500) dan jumlah gambar wajah maksimum 25 gambar.

Tingkat keberhasilan pengujian akhir dari pengujian keseluruhan yang telah didapatkan sejumlah 10 gambar wajah yang terdeteksi dan dapat menampilkan sebuah nama setiap gambar yang berbeda sedangkan 3 gambar wajah yang gagal karena sangat berbeda dengan gambar lain, maka jumlah keseluruhan tingkat keberhasilan ialah $\frac{10}{13} \times 100\% = 76,92\%$.

5.2 Saran

Ada beberapa saran untuk pengenalan wajah sebagai berikut:

1. Terintegrasi pada aplikasi pengenalan wajah dengan akurasi yang sangat jelas dan dapat mengenali sebuah nama setiap seorang yang tidak mengenal nama tanpa aplikasi.
2. Mempermudahkan mengatur keakuratan pengenalan wajah agar mudah mendeteksi wajah dan juga dapat menampilkan sebuah nama.
3. Pengenalan wajah ini menerapkan dengan menggunakan metode *eigenface* untuk meningkatkan keakuratan pengenalan wajah dan mengendalikan aplikasi pengenalan wajah dengan metode *eigenface* tersebut.



DAFTAR PUSTAKA

- Adiwijaya, 2007. Halaman Dokumentasi Teknologi Pengenalan <https://adiwijaya.staff.telkomuniversity.ac.id/publications/repository/>. Diakses pada tahun 2007.
- Admin Repo UGM. *Pengenalan Wajah dengan Metode Eigenface*. <https://repository.ugm.ac.id/28993/>. Diakses pada tanggal 18 Juni 2014
- Akalin, V., 2003, Face Recognition Using Eigenfaces and Neural Networks, Thesis, The Graduate School of Natural and Applied Sciences of The Middle East Technical University.
- Burga. 2013. *PCA, Eigenface and All That*. <http://bugra.github.io/work/notes/2013-07-27/PCA-EigenFace-And-All-That/>. Diakses pada tanggal 27 Juli 2013.
- Davidk. 2017. *Mengenal apa itu Android Studio?*. <http://teknologimodern.com/mengenal-apa-itu-android-studio/>. Diakses pada tanggal 4 Oktober 2017.
- Developer, Android. 2018. *Mengenal Android Studio*. <https://developer.android.com/studio/intro/?hl=id>. Diakses pada tanggal 25 April 2018.
- Eka. 2017. *Apa Itu Android?? Pengertian, Kelebihan, dan Kekurangan*. <http://hplover.com/pengertian-apa-itu-android.html>. Diakses pada tanggal 17 April 2017
- Herdi, Hafizh. 2014. *Belajar Membuat Aplikasi Android Menggunakan Android studio*. <https://www.twoh.co/2014/09/28/belajar-membuat-aplikasi-android-menggunakan-android-studio/>. Diakses pada tanggal 28 September 2014.
- Hermawati, F. A. 2013. *Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

- Indra. 2012. Sistem Pengenalan Wajah Dengan Metode Eigenface Untuk Absensi Pada PT. florindo lestari. Jakarta: Universitas Budi Luhur.
- Jubilee. 2005. Mengenal Dasar-Dasar Pemrograman. Jakarta: Gramedia.
- Jung, K. and Kim, H.J., 2002, Face Recognition Using Kernel Principal Component Analysis, IEEE Signal Processing Letters, Volume: 9 , Issue: 2, 40 – 42.
- Lyon, Douglass., Vincent. N. 2009. Interactive Embedded Face Recognition. Object Technology 8:23-25.
- Mallick, Satya. 2018. *Eigenface using OpenCV (C++/Python)*. <https://www.learnopencv.com/eigenface-using-opencv-c-python/>. Diakses pada tanggal 18 Januari 2018.
- Murinto. 2007. *Pengenalan Wajah Manusia dengan Metode Principle Component Analysis(PCA)*. https://www.researchgate.net/publication/283083488_PENGENALAN_WAJAH_MANUSIA_DENGAN_METODE_PRINCIPLE_COMPONENT_ANALYSIS_PCA. Diakses pada bulan Desember tahun 2007.
- Pratiwi. 2013. *Implementasi Pengenalan Wajah Menggunakan PCA (Principal Component Analysis)*. <https://jurnal.ugm.ac.id/ijeis/article/view/3892>. Diakses 2013.
- Pridiono. 2014. Tentang Teknologi Pengenalan Wajah. <http://www.umboh.org/2011/12/tentang-teknologi-pengenalan-wajah.html> .Diakses pada tahun 2014.
- Shabrina, Reza. 2017. *Pengertian Android Beserta Kelebihan dan Kekurangan*. <https://www.nesabamedia.com/pengertian-android-beserta-kelebihan-dan-kekurangannya/>. Diakses pada tanggal 19 Agustus 2017.
- Shervin, 2010, Introduction to Face Detection and Face Recognition, <http://www.shervinemami.info/faceRecognition.html> diakses tanggal 23 Mei 2012 jam 9.29.

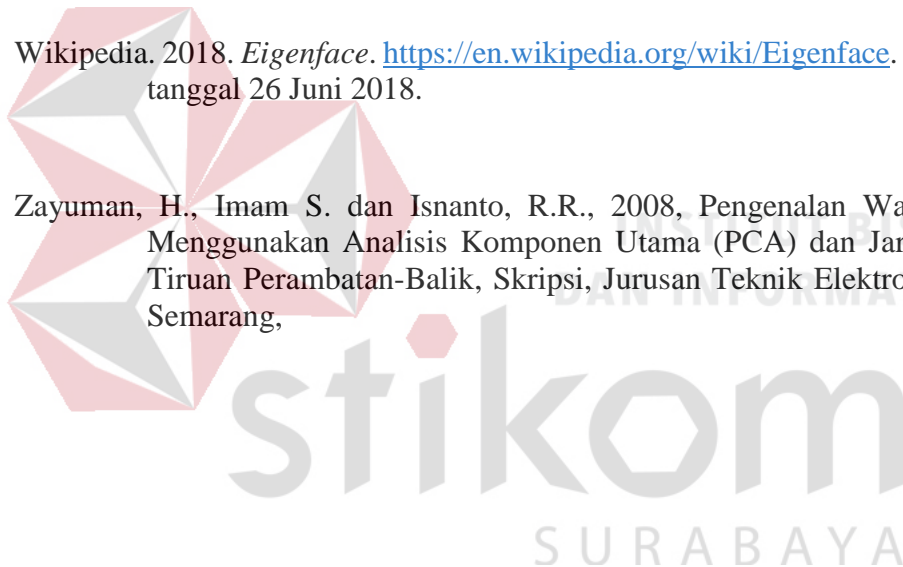
Tan, Dawud. 2010. *Pengenalan Citra Wajah dengan Menggunakan Metode Eigenface*. <https://cakur482.wordpress.com/2010/10/11/pengenalan-citra-wajah-manusia-dengan-menggunakan-metode-eigenface/>. Diakses pada tanggal 11 Oktober 2010.

Utopicomputer. 2017. *Apa itu Smartphone? Ini Pengertian dan Apa Perbedaannya dengan HP*. <https://www.utopicomputers.com/apa-itu-smartphone-ini-pengertian-dan-apa-perbedaannya-dengan-hp/>. Diakses pada tanggal 21 Juni 2017.

Wikipedia. 2018. *Android (Sistem Operasi)*. [https://id.wikipedia.org/wiki/Android_\(sistem_operasi\)](https://id.wikipedia.org/wiki/Android_(sistem_operasi)). Diakses pada tanggal 14 Juli 2018.

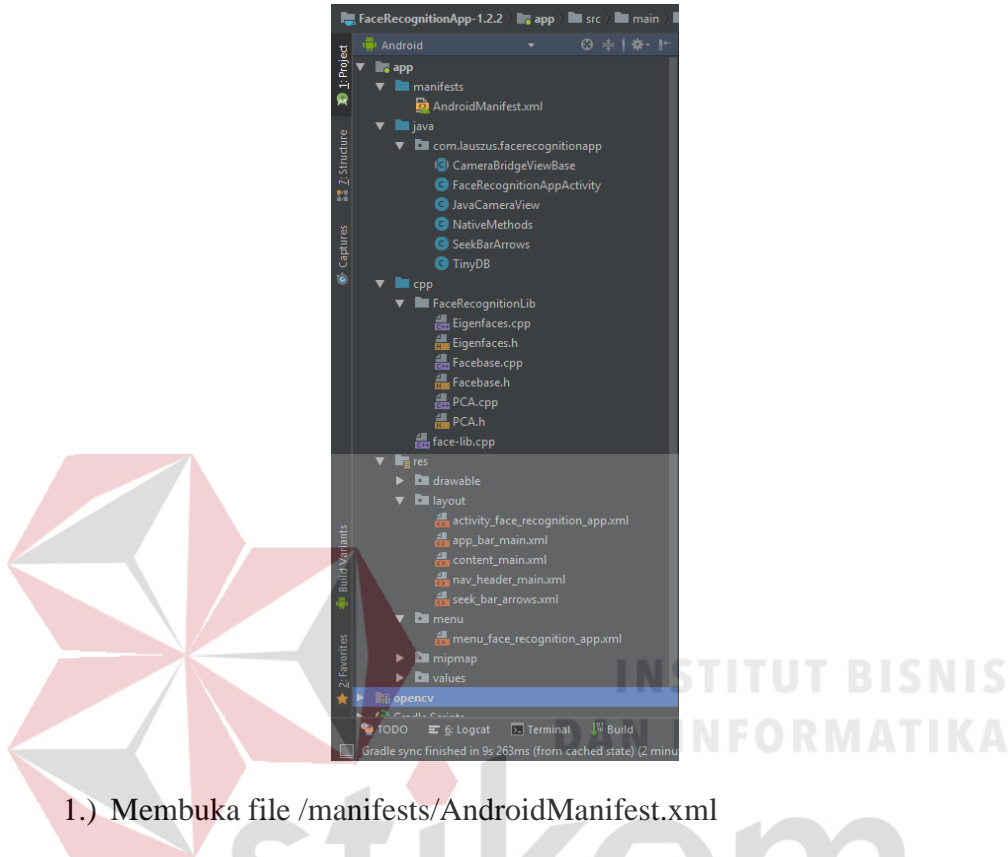
Wikipedia. 2018. *Eigenface*. <https://en.wikipedia.org/wiki/Eigenface>. Diakses pada tanggal 26 Juni 2018.

Zayuman, H., Imam S. dan Isnanto, R.R., 2008, *Pengenalan Wajah Manusia Menggunakan Analisis Komponen Utama (PCA) dan Jaringan Syaraf Tiruan Perambatan-Balik*, Skripsi, Jurusan Teknik Elektro FT UNDIP, Semarang,



LAMPIRAN

Lampiran 1 : Program Android Studio



1.) Membuka file /manifests/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lauszus.facerecognitionapp">

    <!--<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>-->
    <uses-permission android:name="android.permission.CAMERA"/>

    <uses-feature android:name="android.hardware.camera"
    android:required="false"/>
    <uses-feature android:name="android.hardware.camera.autofocus"
    android:required="false"/>
    <uses-feature android:name="android.hardware.camera.front"
    android:required="false"/>
    <uses-feature android:name="android.hardware.camera.front.autofocus"
    android:required="false"/>

    <application
        android:hardwareAccelerated="true"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="false"
        android:theme="@style/AppTheme">
        <activity android:name=".FaceRecognitionAppActivity"
            android:screenOrientation="fullSensor">
```

```

        android:label="@string/app_name"
        android:theme="@style/AppTheme.NoActionBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER"
        />
        </intent-filter>
    </activity>
</application>
</manifest>

```

2.) Membuka file /java/com.lauszus.facerecognitionapp

/CameraBridgeViewBase.java

```

package com.lauszus.facerecognitionapp;
import android.Manifest;
import android.animation.Animator;
import android.animation.ObjectAnimator;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.InputType;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceView;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;

```

```

import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.Locale;
import java.util.Set;

public class FaceRecognitionAppActivity extends AppCompatActivity
implements CameraBridgeViewBase.CvCameraViewListener2 {
    private static final String TAG =
FaceRecognitionAppActivity.class.getSimpleName();
    private static final int PERMISSIONS_REQUEST_CODE = 0;
    private ArrayList<Mat> images;
    private ArrayList<String> imagesLabels;
    private String[] uniqueLabels;
    private CameraBridgeViewBase mOpenCvCameraView;
    private Mat mRgba, mGray;
    private Toast mToast;
    private boolean useEigenfaces;
    private SeekBarArrows mThresholdFace, mThresholdDistance,
mMaximumImages;
    private float faceThreshold, distanceThreshold;
    private int maximumImages;
    private SharedPreferences prefs;
    private TinyDB tinydb;
    private Toolbar mToolbar;
    private NativeMethods.TrainFacesTask mTrainFacesTask;

    private void showToast(String message, int duration) {
        if (duration != Toast.LENGTH_SHORT && duration !=
Toast.LENGTH_LONG)
            throw new IllegalArgumentException();
        if (mToast != null && mToast.getView().isShown())
            mToast.cancel(); // Close the toast if it is already open
        mToast = Toast.makeText(this, message, duration);
        mToast.show();
    }

    private void addLabel(String string) {
        String label = string.substring(0, 1).toUpperCase(Locale.US) +
string.substring(1).trim().toLowerCase(Locale.US); // Make sure that the
name is always uppercase and rest is lowercase
        imagesLabels.add(label); // Add label to list of labels
        Log.i(TAG, "Label: " + label);

        trainFaces(); // When we have finished setting the label, then
retrain faces
    }
    /**
     * Train faces using stored images.
     * @return Returns false if the task is already running.
     */
    private boolean trainFaces() {

```

```

        if (images.isEmpty())
            return true; // The array might be empty if the method is
changed in the OnClickListener

        if (mTrainFacesTask != null && mTrainFacesTask.getStatus() !=
AsyncTask.Status.FINISHED) {
            Log.i(TAG, "mTrainFacesTask is still running");
            return false;
        }

        Mat imagesMatrix = new Mat((int) images.get(0).total(),
images.size(), images.get(0).type());
        for (int i = 0; i < images.size(); i++)
            images.get(i).copyTo(imagesMatrix.col(i)); // Create matrix
where each image is represented as a column vector

        Log.i(TAG, "Images height: " + imagesMatrix.height() + " Width: "
+ imagesMatrix.width() + " total: " + imagesMatrix.total());

        // Train the face recognition algorithms in an asynchronous task,
so we do not skip any frames
        if (useEigenfaces) {
            Log.i(TAG, "Training Eigenfaces");
            showToast("Training " +
getResources().getString(R.string.eigenfaces), Toast.LENGTH_SHORT);

            mTrainFacesTask = new
NativeMethods.TrainFacesTask(imagesMatrix, trainFacesTaskCallback);
        } else {
            Log.i(TAG, "Training Fisherfaces");
            showToast("Training " +
getResources().getString(R.string.fisherfaces), Toast.LENGTH_SHORT);

            Set<String> uniqueLabelsSet = new HashSet<>(imagesLabels); //
Get all unique labels
            uniqueLabels = uniqueLabelsSet.toArray(new
String[uniqueLabelsSet.size()]); // Convert to String array, so we can
read the values from the indices

            int[] classesNumbers = new int[uniqueLabels.length];
            for (int i = 0; i < classesNumbers.length; i++)
                classesNumbers[i] = i + 1; // Create incrementing list
for each unique label starting at 1

            int[] classes = new int[imagesLabels.size()];
            for (int i = 0; i < imagesLabels.size(); i++) {
                String label = imagesLabels.get(i);
                for (int j = 0; j < uniqueLabels.length; j++) {
                    if (label.equals(uniqueLabels[j])) {
                        classes[i] = classesNumbers[j]; // Insert
corresponding number
                        break;
                    }
                }
            }

            /*for (int i = 0; i < imagesLabels.size(); i++)
                Log.i(TAG, "Classes: " + imagesLabels.get(i) + " = " +
classes[i]);*/
            Mat vectorClasses = new Mat(classes.length, 1,
CvType.CV_32S); // CV_32S == int

```

```

        vectorClasses.put(0, 0, classes); // Copy int array into a
vector

        mTrainFacesTask = new
NativeMethods.TrainFacesTask(imagesMatrix, vectorClasses,
trainFacesTaskCallback);
    }
    mTrainFacesTask.execute();
    return true;
}

private NativeMethods.TrainFacesTask.Callback trainFacesTaskCallback
= new NativeMethods.TrainFacesTask.Callback() {
    @Override
    public void onTrainFacesComplete(boolean result) {
        if (result)
            showToast("Training complete", Toast.LENGTH_SHORT);
        else
            showToast("Training failed", Toast.LENGTH_LONG);
    }
};

private void showLabelsDialog() {
    Set<String> uniqueLabelsSet = new HashSet<>(imagesLabels); // Get
all unique labels
    if (!uniqueLabelsSet.isEmpty()) { // Make sure that there are any
labels
        // Inspired by:
http://stackoverflow.com/questions/15762905/how-can-i-display-a-list-
view-in-an-android-alert-dialog
        AlertDialog.Builder builder = new
AlertDialog.Builder(FaceRecognitionAppActivity.this);
        builder.setTitle("Select label:");
        builder.setPositiveButton("New face", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
                showEnterLabelDialog();
            }
        });
        builder.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
                images.remove(images.size() - 1); // Remove last
image
            }
        });
        builder.setCancelable(false); // Prevent the user from
closing the dialog

        String[] uniqueLabels = uniqueLabelsSet.toArray(new
String[uniqueLabelsSet.size()]); // Convert to String array for
ArrayAdapter
        Arrays.sort(uniqueLabels); // Sort labels alphabetically
        final ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<String>(FaceRecognitionAppActivity.this,
android.R.layout.simple_list_item_1, uniqueLabels) {
            @Override

```

```

        public @NonNull View getView(int position, @Nullable View
convertView, @NonNull ViewGroup parent) {
            TextView textView = (TextView)
super.getView(position, convertView, parent);
            if (getResources().getBoolean(R.bool.isTablet))
                textView.setTextSize(20); // Make text slightly
bigger on tablets compared to phones
            else
                textView.setTextSize(18); // Increase text size a
little bit
            return textView;
        }
    };
    ListView mListView = new
ListView(FaceRecognitionAppActivity.this);
    mListView.setAdapter(arrayAdapter); // Set adapter, so the
items actually show up
    builder.setView(mListView); // Set the ListView

    final AlertDialog dialog = builder.show(); // Show dialog and
store in final variable, so it can be dismissed by the ListView

    mListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
            dialog.dismiss();
            addLabel(arrayAdapter.getItem(position));
        }
    });
    } else
        showEnterLabelDialog(); // If there is no existing labels,
then ask the user for a new label
    }
    private void showEnterLabelDialog() {
        AlertDialog.Builder builder = new
AlertDialog.Builder(FaceRecognitionAppActivity.this);
        builder.setTitle("Please enter your name:");

        final EditText input = new
EditText(FaceRecognitionAppActivity.this);
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        builder.setView(input);

        builder.setPositiveButton("Submit", null); // Set up positive
button, but do not provide a listener, so we can check the string before
dismissing the dialog
        builder.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
                images.remove(images.size() - 1); // Remove last image
            }
        });
        builder.setCancelable(false); // User has to input a name
        AlertDialog dialog = builder.create();

        // Source: http://stackoverflow.com/a/7636468/2175837
        dialog.setOnShowListener(new DialogInterface.OnShowListener() {

```

```

        @Override
        public void onShow(final DialogInterface dialog) {
            Button mButton = ((AlertDialog)
dialog).getButton(AlertDialog.BUTTON_POSITIVE);
            mButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    String string =
input.getText().toString().trim();
                    if (!string.isEmpty()) { // Make sure the input
is valid
                        // If input is valid, dismiss the dialog and
add the label to the array
                            dialog.dismiss();
                            addLabel(string);
                        }
                    }
                });
            });
        }

        // Show keyboard, so the user can start typing straight away
dialog.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT
_STATE_VISIBLE);

        dialog.show();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        setContentView(R.layout.activity_face_recognition_app);
        mToolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar); // Sets the Toolbar to act as the
ActionBar for this Activity window

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
drawer, mToolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        final RadioButton mRadioButtonEigenfaces = (RadioButton)
findViewById(R.id.eigenfaces);
        final RadioButton mRadioButtonFisherfaces = (RadioButton)
findViewById(R.id.fisherfaces);

        mRadioButtonEigenfaces.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                useEigenfaces = true;
                if (!trainFaces()) {
                    useEigenfaces = false; // Set variable back

```



```

        showToast("Still training...", Toast.LENGTH_SHORT);
        mRadioButtonEigenfaces.setChecked(useEigenfaces);
        mRadioButtonFisherfaces.setChecked(!useEigenfaces);
    }
}
});
mRadioButtonFisherfaces.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        useEigenfaces = false;
        if (!trainFaces()) {
            useEigenfaces = true; // Set variable back
            showToast("Still training...", Toast.LENGTH_SHORT);
            mRadioButtonEigenfaces.setChecked(useEigenfaces);
            mRadioButtonFisherfaces.setChecked(!useEigenfaces);
        }
    }
});

// Set radio button based on value stored in shared preferences
prefs = PreferenceManager.getDefaultSharedPreferences(this);
useEigenfaces = prefs.getBoolean("useEigenfaces", false);
mRadioButtonEigenfaces.setChecked(useEigenfaces);
mRadioButtonFisherfaces.setChecked(!useEigenfaces);

tinydb = new TinyDB(this); // Used to store ArrayLists in the
shared preferences

mThresholdFace = (SeekBarArrows)
findViewById(R.id.threshold_face);
mThresholdFace.setOnSeekBarArrowsChangeListener(new
SeekBarArrows.OnSeekBarArrowsChangeListener() {
    @Override
    public void onProgressChanged(float progress) {
        Log.i(TAG, "Face threshold: " +
mThresholdFace.progressToString(progress));
        faceThreshold = progress;
    }
});
faceThreshold = mThresholdFace.getProgress(); // Get initial
value

mThresholdDistance = (SeekBarArrows)
findViewById(R.id.threshold_distance);
mThresholdDistance.setOnSeekBarArrowsChangeListener(new
SeekBarArrows.OnSeekBarArrowsChangeListener() {
    @Override
    public void onProgressChanged(float progress) {
        Log.i(TAG, "Distance threshold: " +
mThresholdDistance.progressToString(progress));
        distanceThreshold = progress;
    }
});
distanceThreshold = mThresholdDistance.getProgress(); // Get
initial value

mMaximumImages = (SeekBarArrows)
findViewById(R.id.maximum_images);
mMaximumImages.setOnSeekBarArrowsChangeListener(new
SeekBarArrows.OnSeekBarArrowsChangeListener() {

```

```

        @Override
        public void onProgressChanged(float progress) {
            Log.i(TAG, "Maximum number of images: " +
mMaximumImages.progressToString(progress));
            maximumImages = (int)progress;
            if (images != null && images.size() > maximumImages) {
                int nrRemoveImages = images.size() - maximumImages;
                Log.i(TAG, "Removed " + nrRemoveImages + " images
from the list");
                images.subList(0, nrRemoveImages).clear(); // Remove
oldest images
                imagesLabels.subList(0, nrRemoveImages).clear(); //
Remove oldest labels
                trainFaces(); // Retrain faces
            }
        }
    });
    maximumImages = (int)mMaximumImages.getProgress(); // Get initial
value

    findViewById(R.id.clear_button).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Log.i(TAG, "Cleared training set");
            images.clear(); // Clear both arrays, when new instance
is created
            imagesLabels.clear();
            showToast("Training set cleared", Toast.LENGTH_SHORT);
        }
    });

    findViewById(R.id.take_picture_button).setOnClickListener(new
View.OnClickListener() {
        NativeMethods.MeasureDistTask mMeasureDistTask;

        @Override
        public void onClick(View v) {
            if (mMeasureDistTask != null &&
mMeasureDistTask.getStatus() != AsyncTask.Status.FINISHED) {
                Log.i(TAG, "mMeasureDistTask is still running");
                showToast("Still processing old image...",
Toast.LENGTH_SHORT);
                return;
            }
            if (mTrainFacesTask != null &&
mTrainFacesTask.getStatus() != AsyncTask.Status.FINISHED) {
                Log.i(TAG, "mTrainFacesTask is still running");
                showToast("Still training...", Toast.LENGTH_SHORT);
                return;
            }

            Log.i(TAG, "Gray height: " + mGray.height() + " Width: "
+ mGray.width() + " total: " + mGray.total());
            if (mGray.total() == 0)
                return;
            Size imageSize = new Size(200, 200.0f / ((float)
mGray.width() / (float) mGray.height())); // Scale image in order to
decrease computation time
            Imgproc.resize(mGray, mGray, imageSize);

```

```

        Log.i(TAG, "Small gray height: " + mGray.height() + "
Width: " + mGray.width() + " total: " + mGray.total());
        //SaveImage(mGray);

        Mat image = mGray.reshape(0, (int) mGray.total()); //
Create column vector
        Log.i(TAG, "Vector height: " + image.height() + " Width:
" + image.width() + " total: " + image.total());
        images.add(image); // Add current image to the array

        if (images.size() > maximumImages) {
            images.remove(0); // Remove first image
            imagesLabels.remove(0); // Remove first label
            Log.i(TAG, "The number of images is limited to: " +
images.size());
        }

        // Calculate normalized Euclidean distance
        mMeasureDistTask = new
NativeMethods.MeasureDistTask(useEigenfaces, measureDistTaskCallback);
        mMeasureDistTask.execute(image);

        showLabelsDialog();
    }
});

    mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.camera_java_surface_view);
    mOpenCvCameraView.setCameraIndex(pref.get("mCameraIndex",
CameraBridgeViewBase.CAMERA_ID_FRONT));
    mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
}

    private NativeMethods.MeasureDistTask.Callback
measureDistTaskCallback = new NativeMethods.MeasureDistTask.Callback() {
        @Override
        public void onMeasureDistComplete(Bundle bundle) {
            if (bundle == null) {
                showToast("Failed to measure distance",
Toast.LENGTH_LONG);
                return;
            }

            float minDist =
bundle.getFloat(NativeMethods.MeasureDistTask.MIN_DIST_FLOAT);
            if (minDist != -1) {
                int minIndex =
bundle.getInt(NativeMethods.MeasureDistTask.MIN_DIST_INDEX_INT);
                float faceDist =
bundle.getFloat(NativeMethods.MeasureDistTask.DIST_FACE_FLOAT);
                if (imagesLabels.size() > minIndex) { // Just to be sure
                    Log.i(TAG, "dist[" + minIndex + "]: " + minDist + ",
face dist: " + faceDist + ", label: " + imagesLabels.get(minIndex));

                    String minDistString = String.format(Locale.US,
"%0.4f", minDist);
                    String faceDistString = String.format(Locale.US,
"%0.4f", faceDist);

```

```

        if (faceDist < faceThreshold && minDist <
distanceThreshold) // 1. Near face space and near a face class
            showToast("Face detected: " +
imagesLabels.get(minIndex) + ". Distance: " + minDistString,
Toast.LENGTH_LONG);
        else if (faceDist < faceThreshold) // 2. Near face
space but not near a known face class
            showToast("Unknown face. Face distance: " +
faceDistString + ". Closest Distance: " + minDistString,
Toast.LENGTH_LONG);
        else if (minDist < distanceThreshold) // 3. Distant
from face space and near a face class
            showToast("False recognition. Face distance: " +
faceDistString + ". Closest Distance: " + minDistString,
Toast.LENGTH_LONG);
        else // 4. Distant from face space and not near a
known face class.
            showToast("Image is not a face. Face distance: "
+ faceDistString + ". Closest Distance: " + minDistString,
Toast.LENGTH_LONG);
    }
} else {
    Log.w(TAG, "Array is null");
    if (useEigenfaces || uniqueLabels == null ||
uniqueLabels.length > 1)
        showToast("Keep training...", Toast.LENGTH_SHORT);
    else
        showToast("Fisherfaces needs two different faces",
Toast.LENGTH_SHORT);
}
};
}
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull
String permissions[], @NonNull int[] grantResults) {
    switch (requestCode) {
        case PERMISSIONS_REQUEST_CODE:
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                loadOpenCV();
            } else {
                showToast("Permission required!", Toast.LENGTH_LONG);
                finish();
            }
        }
    }
}

@Override
public void onPause() {
    super.onPause();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}

@Override
public void onStart() {
    super.onStart();
    // Read threshold values
    float progress = prefs.getFloat("faceThreshold", -1);

```

```

        if (progress != -1)
            mThresholdFace.setProgress(progress);
        progress = prefs.getFloat("distanceThreshold", -1);
        if (progress != -1)
            mThresholdDistance.setProgress(progress);
        mMaximumImages.setProgress(prefs.getInt("maximumImages", 25)); //
Use 25 images by default
    }

    @Override
    public void onStop() {
        super.onStop();
        // Store threshold values
        Editor editor = prefs.edit();
        editor.putFloat("faceThreshold", faceThreshold);
        editor.putFloat("distanceThreshold", distanceThreshold);
        editor.putInt("maximumImages", maximumImages);
        editor.putBoolean("useEigenfaces", useEigenfaces);
        editor.putInt("mCameraIndex", mOpenCvCameraView.mCameraIndex);
        editor.apply();

        // Store ArrayLists containing the images and labels
        if (images != null && imagesLabels != null) {
            tinydb.putListMat("images", images);
            tinydb.putListString("imagesLabels", imagesLabels);
        }
    }

    @Override
    public void onResume() {
        super.onResume();

        // Request permission if needed
        if (ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED/* ||
ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED*/)
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA/*,
Manifest.permission.WRITE_EXTERNAL_STORAGE*/}, PERMISSIONS_REQUEST_CODE);
        else
            loadOpenCV();
    }

    private BaseLoaderCallback mLoaderCallback = new
BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status) {
                case LoaderCallbackInterface.SUCCESS:
                    NativeMethods.loadNativeLibraries(); // Load native
libraries after(!) OpenCV initialization
                    Log.i(TAG, "OpenCV loaded successfully");
                    mOpenCvCameraView.enableView();

                    // Read images and labels from shared preferences
                    images = tinydb.getListMat("images");
                    imagesLabels = tinydb.getListString("imagesLabels");

```

```

        Log.i(TAG, "Number of images: " + images.size() + ".
Number of labels: " + imagesLabels.size());
        if (!images.isEmpty()) {
            trainFaces(); // Train images after they are
loaded
            Log.i(TAG, "Images height: " +
images.get(0).height() + " Width: " + images.get(0).width() + " total: "
+ images.get(0).total());
        }
        Log.i(TAG, "Labels: " + imagesLabels);

        break;
    default:
        super.onManagerConnected(status);
        break;
    }
}
};

private void loadOpenCV() {
    if (!OpenCVLoader.initDebug(true)) {
        Log.d(TAG, "Internal OpenCV library not found. Using OpenCV
Manager for initialization");
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0,
this, mLoaderCallback);
    } else {
        Log.d(TAG, "OpenCV library found inside package. Using it!");
    }
    mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}

public void onCameraViewStarted(int width, int height) {
    mGray = new Mat();
    mRgba = new Mat();
}

public void onCameraViewStopped() {
    mGray.release();
    mRgba.release();
}

public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
    Mat mGrayTmp = inputFrame.gray();
    Mat mRgbaTmp = inputFrame.rgba();

    // Flip image to get mirror effect
    int orientation = mOpenCvCameraView.getScreenOrientation();
    if (mOpenCvCameraView.isEmulator()) // Treat emulators as a
special case
        Core.flip(mRgbaTmp, mRgbaTmp, 1); // Flip along y-axis
    else {
        switch (orientation) { // RGB image

```

```

        case ActivityInfo.SCREEN_ORIENTATION_PORTRAIT:
        case ActivityInfo.SCREEN_ORIENTATION_REVERSE_PORTRAIT:
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_FRONT)
                Core.flip(mRgbaTmp, mRgbaTmp, 0); // Flip along
x-axis
            else
                Core.flip(mRgbaTmp, mRgbaTmp, -1); // Flip along
both axis
            break;
        case ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE:
        case ActivityInfo.SCREEN_ORIENTATION_REVERSE_LANDSCAPE:
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_FRONT)
                Core.flip(mRgbaTmp, mRgbaTmp, 1); // Flip along
y-axis
            break;
    }
    switch (orientation) { // Grayscale image
        case ActivityInfo.SCREEN_ORIENTATION_PORTRAIT:
            Core.transpose(mGrayTmp, mGrayTmp); // Rotate image
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_FRONT)
                Core.flip(mGrayTmp, mGrayTmp, -1); // Flip along
both axis
            else
                Core.flip(mGrayTmp, mGrayTmp, 1); // Flip along
y-axis
            break;
        case ActivityInfo.SCREEN_ORIENTATION_REVERSE_PORTRAIT:
            Core.transpose(mGrayTmp, mGrayTmp); // Rotate image
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_BACK)
                Core.flip(mGrayTmp, mGrayTmp, 0); // Flip along
x-axis
            break;
        case ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE:
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_FRONT)
                Core.flip(mGrayTmp, mGrayTmp, 1); // Flip along
y-axis
            break;
        case ActivityInfo.SCREEN_ORIENTATION_REVERSE_LANDSCAPE:
            Core.flip(mGrayTmp, mGrayTmp, 0); // Flip along x-
axis
            if (mOpenCvCameraView.mCameraIndex ==
CameraBridgeViewBase.CAMERA_ID_BACK)
                Core.flip(mGrayTmp, mGrayTmp, 1); // Flip along
y-axis
            break;
    }
}

mGray = mGrayTmp;
mRgba = mRgbaTmp;

return mRgba;
}

@SuppressWarnings("ResultOfMethodCallIgnored")
public void SaveImage(Mat mat) {

```

```

        Mat mIntermediateMat = new Mat();

        if (mat.channels() == 1) // Grayscale image
            Imgproc.cvtColor(mat, mIntermediateMat,
                Imgproc.COLOR_GRAY2BGR);
        else
            Imgproc.cvtColor(mat, mIntermediateMat,
                Imgproc.COLOR_RGBA2BGR);

        File path = new
        File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_
        PICTURES), TAG); // Save pictures in Pictures directory
        path.mkdir(); // Create directory if needed
        String fileName = "IMG_" + new
        SimpleDateFormat("yyyyMMdd_HHmss_SSS", Locale.US).format(new Date()) +
        ".png";
        File file = new File(path, fileName);

        boolean bool = Imgcodecs.imwrite(file.toString(),
        mIntermediateMat);

        if (bool)
            Log.i(TAG, "SUCCESS writing image to external storage");
        else
            Log.e(TAG, "Failed writing image to external storage");
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
        findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START))
            drawer.closeDrawer(GravityCompat.START);
        else
            super.onBackPressed();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_face_recognition_app,
        menu);
        // Show rear camera icon if front camera is currently used and
        front camera icon if back camera is used
        MenuItem menuItem = menu.findItem(R.id.flip_camera);
        if (mOpenCvCameraView.mCameraIndex ==
        CameraBridgeViewBase.CAMERA_ID_FRONT)
            menuItem.setIcon(R.drawable.ic_camera_rear_white_24dp);
        else
            menuItem.setIcon(R.drawable.ic_camera_front_white_24dp);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.flip_camera:
                mOpenCvCameraView.flipCamera();

                // Do flip camera animation
                View v = mToolbar.findViewById(R.id.flip_camera);

```



```

        ObjectAnimator animator = ObjectAnimator.ofFloat(v,
"rotationY", v.getRotationY() + 180.0f);
        animator.setDuration(500);
        animator.addListener(new Animator.AnimatorListener() {
            @Override
            public void onAnimationStart(Animator animation) {

            }

            @Override
            public void onAnimationEnd(Animator animation) {
                supportInvalidateOptionsMenu(); // This will call
onCreateOptionsMenu()
            }
            @Override
            public void onAnimationCancel(Animator animation) {

            }
            @Override
            public void onAnimationRepeat(Animator animation) {

            }
        });
        animator.start();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
}

```

3.) Membuka file/java/com.lauszus.facerecognitionapp/JavaCameraView.java

```

// Based on:
https://github.com/opencv/opencv/blob/master/modules/java/generator/android/java/org/opencv/android/JavaCameraView.java

package com.lauszus.facerecognitionapp;
import android.annotation.SuppressLint;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.PreviewCallback;
import android.os.Build;
import android.util.AttributeSet;
import android.util.Log;
import android.view.ViewGroup.LayoutParams;
import org.opencv.BuildConfig;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import java.util.List;

public class JavaCameraView extends CameraBridgeViewBase implements
PreviewCallback {
    private static final int MAGIC_TEXTURE_ID = 10;
    private static final String TAG = "JavaCameraView";
    private byte mBuffer[];

```

```

private Mat[] mFrameChain;
private int mChainIdx = 0;
private Thread mThread;
private boolean mStopThread;
protected Camera mCamera;
protected JavaCameraFrame[] mCameraFrame;
private int mPreviewFormat = ImageFormat.NV21;

@SuppressWarnings("FieldCanBeLocal") // This slows down the frame
rate significantly
private SurfaceTexture mSurfaceTexture;

public static class JavaCameraSizeAccessor implements
ListItemAccessor {

    @Override
    public int getWidth(Object obj) {
        Camera.Size size = (Camera.Size) obj;
        return size.width;
    }

    @Override
    public int getHeight(Object obj) {
        Camera.Size size = (Camera.Size) obj;
        return size.height;
    }
}

@SuppressWarnings("unused")
public JavaCameraView(Context context, int cameraId) {
    super(context, cameraId);
}

public JavaCameraView(Context context, AttributeSet attrs) {
    super(context, attrs);
}

@SuppressLint("ObsoleteSdkInt")
protected boolean initializeCamera(int width, int height) {
    Log.d(TAG, "Initialize java camera");
    boolean result = true;
    synchronized (this) {
        mCamera = null;

        if (mCameraIndex == CAMERA_ID_ANY || isEmulator()) { // Just
open any camera on emulators
            Log.d(TAG, "Trying to open camera with old open()");
            try {
                mCamera = Camera.open();
            }
            catch (Exception e){
                Log.e(TAG, "Camera is not available (in use or does
not exist): " + e.getLocalizedMessage());
            }

            if(mCamera == null && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.GINGERBREAD) {
                boolean connected = false;
                for (int camIdx = 0; camIdx <
Camera.getNumberOfCameras(); ++camIdx) {

```

```

        Log.d(TAG, "Trying to open camera with new open("
+ camIdx + ")");
        try {
            mCamera = Camera.open(camIdx);
            connected = true;
        } catch (RuntimeException e) {
            Log.e(TAG, "Camera #" + camIdx + "failed to
open: " + e.getLocalizedMessage());
        }
        if (connected) break;
    }
} else {
    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.GINGERBREAD) {
        int localCameraIndex = mCameraIndex;
        if (mCameraIndex == CAMERA_ID_BACK) {
            Log.i(TAG, "Trying to open back camera");
            Camera.CameraInfo cameraInfo = new
Camera.CameraInfo();
            for (int camIdx = 0; camIdx <
Camera.getNumberOfCameras(); ++camIdx) {
                Camera.getCameraInfo( camIdx, cameraInfo );
                if (cameraInfo.facing ==
Camera.CameraInfo.CAMERA_FACING_BACK) {
                    localCameraIndex = camIdx;
                    break;
                }
            }
        } else if (mCameraIndex == CAMERA_ID_FRONT) {
            Log.i(TAG, "Trying to open front camera");
            Camera.CameraInfo cameraInfo = new
Camera.CameraInfo();
            for (int camIdx = 0; camIdx <
Camera.getNumberOfCameras(); ++camIdx) {
                Camera.getCameraInfo( camIdx, cameraInfo );
                if (cameraInfo.facing ==
Camera.CameraInfo.CAMERA_FACING_FRONT) {
                    localCameraIndex = camIdx;
                    break;
                }
            }
        }
        if (localCameraIndex == CAMERA_ID_BACK) {
            Log.e(TAG, "Back camera not found!");
        } else if (localCameraIndex == CAMERA_ID_FRONT) {
            Log.e(TAG, "Front camera not found!");
        } else {
            Log.d(TAG, "Trying to open camera with new open("
+ localCameraIndex + ")");
            try {
                mCamera = Camera.open(localCameraIndex);
            } catch (RuntimeException e) {
                Log.e(TAG, "Camera #" + localCameraIndex +
"failed to open: " + e.getLocalizedMessage());
            }
        }
    }
}

if (mCamera == null)

```

```

        return false;

        /* Now set camera parameters */
        try {
            Camera.Parameters params = mCamera.getParameters();
            Log.d(TAG, "getSupportedPreviewSizes()");
            List<android.hardware.Camera.Size> sizes =
params.getSupportedPreviewSizes();

            if (sizes != null) {
                /* Select the size that fits surface considering
maximum size allowed */
                Size frameSize = calculateCameraFrameSize(sizes, new
JavaCameraSizeAccessor(), width, height);

                /* Image format NV21 causes issues in the Android
emulators */
                if (isEmulator())
                    params.setPreviewFormat(ImageFormat.YV12);
                else
                    params.setPreviewFormat(ImageFormat.NV21);

                Log.d(TAG, "Set preview size to " + frameSize.width +
"x" + frameSize.height);
                params.setPreviewSize((int)frameSize.width,
(int)frameSize.height);

                if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.ICE_CREAM_SANDWICH &&
!android.os.Build.MODEL.equals("GT-I9100"))
                    params.setRecordingHint(true);

                List<String> FocusModes =
params.getSupportedFocusModes();
                if (FocusModes != null &&
FocusModes.contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO))
                {
                    params.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO);
                }
                mCamera.setParameters(params);
                params = mCamera.getParameters();
                mPreviewFormat = params.getPreviewFormat();
                mFrameWidth = params.getPreviewSize().width;
                mFrameHeight = params.getPreviewSize().height;

                if ((getLayoutParams().width ==
LayoutParams.MATCH_PARENT) && (getLayoutParams().height ==
LayoutParams.MATCH_PARENT))
                    mScale = Math.min(((float)height)/mFrameHeight,
((float)width)/mFrameWidth);
                else
                    mScale = 0;
                if (mFpsMeter != null) {
                    mFpsMeter.setResolution(mFrameWidth,
mFrameHeight);
                }
                int size = mFrameWidth * mFrameHeight;
                size = size *
ImageFormat.getBitsPerPixel(params.getPreviewFormat()) / 8;
                mBuffer = new byte[size];

```

```

        mCamera.addCallbackBuffer(mBuffer);
        mCamera.setPreviewCallbackWithBuffer(this);
        mFrameChain = new Mat[2];
        mFrameChain[0] = new Mat(mFrameHeight +
(mFrameHeight/2), mFrameWidth, CvType.CV_8UC1);
        mFrameChain[1] = new Mat(mFrameHeight +
(mFrameHeight/2), mFrameWidth, CvType.CV_8UC1);
        AllocateCache();
        mCameraFrame = new JavaCameraFrame[2];
        mCameraFrame[0] = new JavaCameraFrame(mFrameChain[0],
mFrameWidth, mFrameHeight);
        mCameraFrame[1] = new JavaCameraFrame(mFrameChain[1],
mFrameWidth, mFrameHeight);
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB) {
            mSurfaceTexture = new
SurfaceTexture(MAGIC_TEXTURE_ID);
            mCamera.setPreviewTexture(mSurfaceTexture);
        } else
            mCamera.setPreviewDisplay(null);

        /* Finally we are ready to start the preview */
        Log.d(TAG, "startPreview");
        mCamera.startPreview();
    }
    else
        result = false;
    } catch (Exception e) {
        result = false;
        e.printStackTrace();
    }
    }
    return result;
}
}

protected void releaseCamera() {
    synchronized (this) {
        if (mCamera != null) {
            mCamera.stopPreview();
            mCamera.setPreviewCallback(null);

            mCamera.release();
        }
        mCamera = null;
        if (mFrameChain != null) {
            mFrameChain[0].release();
            mFrameChain[1].release();
        }
        if (mCameraFrame != null) {
            mCameraFrame[0].release();
            mCameraFrame[1].release();
        }
    }
}

private boolean mCameraFrameReady = false;
@Override
protected boolean connectCamera(int width, int height) {

    /* 1. We need to instantiate camera
    * 2. We need to start thread which will be getting frames
    */

```

```

    /* First step - initialize camera connection */
    Log.d(TAG, "Connecting to camera");
    if (!initializeCamera(width, height))
        return false;
    mCameraFrameReady = false;
    /* now we can start update thread */
    Log.d(TAG, "Starting processing thread");
    mStopThread = false;
    mThread = new Thread(new CameraWorker());
    mThread.start();
    return true;
}

@Override
protected void disconnectCamera() {
    /* 1. We need to stop thread which updating the frames
    * 2. Stop camera and release it
    */
    Log.d(TAG, "Disconnecting from camera");
    try {
        mStopThread = true;
        Log.d(TAG, "Notify thread");
        synchronized (this) {
            this.notify();
        }
        Log.d(TAG, "Waiting for thread");
        if (mThread != null)
            mThread.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        mThread = null;
    }
    /* Now release camera */
    releaseCamera();
    mCameraFrameReady = false;
}

@Override
public void onPreviewFrame(byte[] frame, Camera arg1) {
    if (BuildConfig.DEBUG)
        Log.d(TAG, "Preview Frame received. Frame size: " +
frame.length);
    synchronized (this) {
        mFrameChain[mChainIdx].put(0, 0, frame);
        mCameraFrameReady = true;
        this.notify();
    }
    if (mCamera != null)
        mCamera.addCallbackBuffer(mBuffer);
}

private class JavaCameraFrame implements CvCameraViewFrame {
    @Override
    public Mat gray() {
        return mYuvFrameData.submat(0, mHeight, 0, mWidth);
    }

    @Override
    public Mat rgba() {
        if (mPreviewFormat == ImageFormat.NV21)

```

```

        Imgproc.cvtColor(mYuvFrameData, mRgba,
        Imgproc.COLOR_YUV2RGBA_NV21, 4);
        else if (mPreviewFormat == ImageFormat.YV12)
            Imgproc.cvtColor(mYuvFrameData, mRgba,
        Imgproc.COLOR_YUV2RGB_I420, 4); // COLOR_YUV2RGBA_YV12 produces inverted
        colors
        else
            throw new IllegalArgumentException("Preview Format can be
        NV21 or YV12");

        return mRgba;
    }

    @SuppressWarnings("WeakerAccess")
    public JavaCameraFrame(Mat Yuv420sp, int width, int height) {
        super();
        mWidth = width;
        mHeight = height;
        mYuvFrameData = Yuv420sp;
        mRgba = new Mat();
    }
    public void release() {
        mRgba.release();
    }
    private Mat mYuvFrameData;
    private Mat mRgba;
    private int mWidth;
    private int mHeight;
}
private class CameraWorker implements Runnable {
    @Override
    public void run() {
        do {
            boolean hasFrame = false;
            synchronized (JavaCameraView.this) {
                try {
                    while (!mCameraFrameReady && !mStopThread) {
                        JavaCameraView.this.wait();
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (mCameraFrameReady)
                {
                    mChainIdx = 1 - mChainIdx;
                    mCameraFrameReady = false;
                    hasFrame = true;
                }
            }

            if (!mStopThread && hasFrame) {
                if (!mFrameChain[1 - mChainIdx].empty())
                    deliverAndDrawFrame(mCameraFrame[1 - mChainIdx]);
            }
        } while (!mStopThread);
        Log.d(TAG, "Finish processing thread");
    }
}
}
}

```

4.) Membuka file /java/com.lauszus.facerecognitionapp/NativeMethods.java

```

package com.lauszus.facerecognitionapp;

import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;

import org.opencv.core.Mat;

// All computations is done in an asynchronous task, so we do not skip
any frames
class NativeMethods {
    private static final String TAG =
FaceRecognitionAppActivity.class.getSimpleName() + "/" +
NativeMethods.class.getSimpleName();

    static void loadNativeLibraries() {
        System.loadLibrary("face-lib");
    }

    static class TrainFacesTask extends AsyncTask<Void, Void, Boolean> {
        private final Mat images, classes;
        private final Callback callback;
        private Exception error;

        interface Callback {
            void onTrainFacesComplete(boolean result);
        }

        /**
         * Constructor used for Eigenfaces.
         * @param images Matrix containing all images as column vectors.
         */
        TrainFacesTask(Mat images, Callback callback) {
            this(images, null, callback);
        }

        /**
         * Constructor used for Fisherfaces.
         * @param images Matrix containing all images as column vectors.
         * @param classes Vector containing classes for each image.
         */
        TrainFacesTask(Mat images, Mat classes, Callback callback) {
            this.images = images;
            this.classes = classes;
            this.callback = callback;
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            try {
                if (classes == null)
                    TrainFaces(images.getNativeObjAddr(), 0); // Train
Eigenfaces
                else
                    TrainFaces(images.getNativeObjAddr(),
classes.getNativeObjAddr()); // Train Fisherfaces

```



```

        return true;
    } catch (Exception e) {
        error = e;
        return false;
    }
}

@Override
protected void onPostExecute(Boolean result) {
    callback.onTrainFacesComplete(result);
    if (result)
        Log.i(TAG, "Done training images");
    else
        Log.e(TAG, error.getMessage());
}
}

static class MeasureDistTask extends AsyncTask<Mat, Void, Bundle> {
    static final String MIN_DIST_FLOAT = "minDist";
    static final String MIN_DIST_INDEX_INT = "minDistIndex";
    static final String DIST_FACE_FLOAT = "distFace";

    private final Callback callback;
    private final boolean useEigenfaces;
    private Exception error;

    interface Callback {
        void onMeasureDistComplete(Bundle bundle);
    }

    MeasureDistTask(boolean useEigenfaces, Callback callback) {
        this.useEigenfaces = useEigenfaces;
        this.callback = callback;
    }

    @Override
    protected Bundle doInBackground(Mat... mat) {
        float[] minDist = new float[] { -1 };
        int[] minDistIndex = new int[1];
        float[] faceDist = new float[1];
        try {
            MeasureDist(mat[0].getNativeObjAddr(), minDist,
minDistIndex, faceDist, useEigenfaces);
        } catch (Exception e) {
            error = e;
            return null;
        }
        Bundle bundle = new Bundle();
        bundle.putFloat(MIN_DIST_FLOAT, minDist[0]);
        bundle.putInt(MIN_DIST_INDEX_INT, minDistIndex[0]);
        bundle.putFloat(DIST_FACE_FLOAT, faceDist[0]);
        return bundle;
    }

    @Override
    protected void onPostExecute(Bundle bundle) {
        callback.onMeasureDistComplete(bundle);
        if (bundle != null)
            Log.i(TAG, "Done measuring distance");
        else
            Log.e(TAG, error.getMessage());
    }
}

```

```

    }
}

/**
 * Train faces recognition.
 * @param addrImages Address for matrix containing all images as
column vectors.
 * @param addrClasses Address for vector containing classes for
each image.
 *
 * This must be a incrementing list starting at
1.
 *
 * If set to NULL, then Eigenfaces will be used.
 *
 * If this is set, then Fisherfaces will be
used.
 */
private static native void TrainFaces(long addrImages, long
addrClasses);

/**
 * Measure euclidean distance between the weight of the image
compared to all weights.
 * @param addrImage Vector containing the image.
 * @param minDist Returns a list of sorted distances to images
 * @param minDistIndex Returns the index of the closest distance
 * @param faceDist Returns the distance to facespace
 * @param useEigenfaces Set to true if Eigenfaces are used. If set to
false,
 *
 * then Fisherfaces will be used.
 */
private static native void MeasureDist(long addrImage, float[]
minDist, int[] minDistIndex, float[] faceDist, boolean useEigenfaces);
}

```

5.) Membuka file /java/com.lauszus.facerecognitionapp/SeekBarArrows.java

```

package com.lauszus.facerecognitionapp;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Handler;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.SeekBar;
import android.widget.TextView;
import java.util.Locale;

public class SeekBarArrows extends LinearLayout implements
SeekBar.OnSeekBarChangeListener {
    private static final String TAG =
FaceRecognitionAppActivity.class.getSimpleName() + "/" +
SeekBarArrows.class.getSimpleName();
    private SeekBar mSeekBar;
    private TextView mSeekBarValue;
    private float multiplier;
    private int nValues;
    private int min;

```

```

public SeekBarArrows(Context context, AttributeSet attrs) {
    super(context, attrs);

    inflate(context, R.layout.seek_bar_arrows, this); // Use custom
layout

    TypedArray styledAttrs =
getContext().obtainStyledAttributes(attrs, R.styleable.SeekBarArrows); //
Read all attributes from xml

    String mSeekBarText =
styledAttrs.getString(R.styleable.SeekBarArrows_text);
    min = styledAttrs.getInt(R.styleable.SeekBarArrows_min, 0);
    float max = styledAttrs.getFloat(R.styleable.SeekBarArrows_max,
0);
    nValues = styledAttrs.getInt(R.styleable.SeekBarArrows_n_values,
0);

    mSeekBar = (SeekBar) findViewById(R.id.seekBar);
    ((TextView) findViewById(R.id.text)).setText(mSeekBarText);
    mSeekBarValue = (TextView) findViewById(R.id.value);

    setMax(max); // Set maximum value
    mSeekBar.setOnSeekBarChangeListener(this); // Set listener
    mSeekBar.setProgress(mSeekBar.getMax() / 2); // Now center the
SeekBar

    // Use custom OnArrowListener class to handle button click,
button long click and if the button is held down
    new OnArrowListener(findViewById(R.id.rightArrow), mSeekBar,
true);
    new OnArrowListener(findViewById(R.id.leftArrow), mSeekBar,
false);

    styledAttrs.recycle();
}

interface OnSeekBarArrowsChangeListener {
    void onProgressChanged(float progress);
}

private OnSeekBarArrowsChangeListener mOnSeekBarArrowsChangeListener;

public void
setOnSeekBarArrowsChangeListener(OnSeekBarArrowsChangeListener l) {
    mOnSeekBarArrowsChangeListener = l;
}

public float getProgress() {
    return mSeekBar.getProgress() * multiplier + min;
}

public void setProgress(float value) {
    mSeekBar.setProgress((int) (value / multiplier) - min);
}

public int getMin() {
    return min;
}

public void setMin(int min) {

```

```

        this.min = min;
        setMax(getMax());
    }

    public float getMax() {
        return mSeekBar.getMax() * multiplier;
    }

    public void setMax(float max) {
        multiplier = max / (float)nValues;
        mSeekBar.setMax(nValues - min);
        Log.i(TAG, "Max: " + max + " Raw: " + mSeekBar.getMax() + "
Multiplier: " + multiplier);
    }

    private String getFormat() {
        return multiplier <= 0.00001f ? "%.5f" : multiplier <= 0.0001f ?
("%.4f" : multiplier <= 0.001f ? "%.3f" : multiplier <= 0.01f ? "%.2f" :
multiplier <= 0.1f ? "%.1f" : "%.0f";
    }

    public String progressToString(float value) {
        String format = getFormat(); // Set decimal places according to
multiplier
        return String.format(Locale.US, format, value);
    }

    public String progressToString(int value) {
        String format = getFormat(); // Set decimal places according to
multiplier
        return String.format(Locale.US, format, (float)value *
multiplier); // SeekBar can only handle integers, so format it to a float
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        mSeekBarValue.setText(progressToString(progress + min));
        if (mOnSeekBarArrowsChangeListener != null)
mOnSeekBarArrowsChangeListener.onProgressChanged(((float)progress *
multiplier + min);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }

    private class OnArrowListener implements View.OnClickListener,
View.OnLongClickListener, View.OnTouchListener {
        private Handler handler = new Handler();
        private static final int repeatInterval = 300; // Repeat interval
is 300 ms
        private SeekBar mSeekBar;
        private boolean positive;

        OnArrowListener(View v, SeekBar mSeekBar, boolean positive) {

```

```

        Button mButton = (Button) v;
        this.mSeekBar = mSeekBar;
        this.positive = positive;

        mButton.setOnClickListener(this);
        mButton.setOnLongClickListener(this);
        mButton.setOnTouchListener(this);
    }

    private int round10(int n) {
        return Math.round((float)n / 10.0f) * 10;
    }

    private void longClick() {
        mSeekBar.setProgress(round10(mSeekBar.getProgress() +
        (positive ? 10 : -10)) - min); // Increase/decrease with 10 and round to
        nearest multiple of 10
    }

    private Runnable runnable = new Runnable() {
        @Override
        public void run() {
            longClick();
            handler.postDelayed(this, repeatInterval); // Repeat long
            click if button is held down
        }
    };

    @Override
    public void onClick(View v) {
        mSeekBar.setProgress(mSeekBar.getProgress() + (positive ? 1 :
        -1)); // Increase/decrease with 1
    }

    @Override
    public boolean onLongClick(View v) {
        longClick();
        handler.postDelayed(runnable, repeatInterval); // Repeat
        again in 300 ms
        return true;
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                handler.removeCallbacks(runnable); // Remove callback
                if button is released
            }
            return false;
        }
    }
}

```

6.) Membuka file /cpp/FaceRecognitionLib/Eigenfaces.cpp

```
#include <iostream>
```

```

#include <Eigen/Dense> // http://eigen.tuxfamily.org

#include "Eigenfaces.h"

using namespace std;
using namespace Eigen;

void Eigenfaces::train(const MatrixXi &images) {
    this->n_pixels = images.rows();

    // Copy values from PCA
    this->numComponents = PCA::compute(images);
    this->V = PCA::U; // This contains all Eigenfaces

#ifdef NDEBUG
    cout << "Calculate weights for all images" << endl;
#endif
    this->W_all = project(images); // Calculate weights
#ifdef NDEBUG
    cout << "W_all: " << W_all.rows() << " x " << W_all.cols() << endl;
#endif
}

```

7.) Membuka file /cpp/FaceRecognitionLib/Eigenfaces.h

```

#ifdef __eigenfaces_h__
#define __eigenfaces_h__
#include <Eigen/Dense> // http://eigen.tuxfamily.org
#include "Facebase.h"
#include "PCA.h"

using namespace Eigen;

class Eigenfaces : public Facebase {
public:
    /**
     * Train Eigenfaces.
     * @param images Each images represented as a column vector.
     */
    void train(const MatrixXi &images);
};

#endif

```

8.) Membuka file /cpp/FaceRecognitionLib/Facebase.cpp

```

#include <Eigen/Dense> // http://eigen.tuxfamily.org
#include "Facebase.h"

using namespace Eigen;

MatrixXf Facebase::project(const MatrixXi &X) {
    return V.transpose()*(X.cast<float>().colwise() - mu); // Project X
    onto subspace
}

VectorXf Facebase::euclideanDist(const VectorXf &W) {
    return ((W_all.colwise() -
W)/n_pixels).colwise().norm()/sqrt(numComponents); // Measure euclidean
distance between weights
}

VectorXf Facebase::reconstructFace(const VectorXf &W) {
    return V*W;
}

float Facebase::euclideanDistFace(const VectorXi &X, const VectorXf
&face) {
    return (((X.cast<float>() - mu) -
face)/n_pixels).colwise().norm()/sqrt(numComponents)).value(); // Measure
euclidean distance between weights
}

```

9.) Membuka file /cpp/FaceRecognitionLib/Facebase.h

```

#ifndef __facebase_h__
#define __facebase_h__

```

```

#include <Eigen/Dense> // http://eigen.tuxfamily.org
#include "PCA.h"

using namespace Eigen;

class Facebase : public PCA {
public:
    /**
     * Project X onto subspace.
     * @param X Input image.
     * @return Returns the weight matrix.
     */
    MatrixXf project(const MatrixXi &X);

    /**
     * Calculate distance between weight and the weights for the current
    method used.
     * @param W Weights calculated by projecting images onto subspace.
     * @return Return a vector containing all distances.
     */
    VectorXf euclideanDist(const VectorXf &W);

    /**
     * Reconstruct a face from a weight.
     * @param W Weight calculated by projecting image onto subspace.
     * @return Returns the face vector.
     */
    VectorXf reconstructFace(const VectorXf &W);

    /**
     * Calculate the distance to the face subspace.
     * @param X Input image.
     * @param face Face vector.
     * @return Returns the distance to the face subspace.
     */
    float euclideanDistFace(const VectorXi &X, const VectorXf &face);

    MatrixXf V; // Eigenvector
    int32_t numComponents; // Number of components

protected:
    MatrixXf W_all; // Total weights
    size_t n_pixels;
};

#endif

```

10.) Membuka file /cpp/FaceRecognitionLib/PCA.cpp

```

#include <iostream>

#include <Eigen/Dense> // http://eigen.tuxfamily.org
#include <RedSVD/RedSVD-h> // https://github.com/ntessore/redsvd-h

#include "PCA.h"

using namespace std;
using namespace Eigen;

```



```

// See: http://eigen.tuxfamily.org/dox/structEigen_1_1IOFormat.html
static IOFormat OctaveFmt(StreamPrecision, 0, " ", " ", ";\n", " ", " ", "[",
" ]");

int32_t PCA::compute(const MatrixXi &images, int32_t numComponents /*= -
1*/) {
#ifdef NDEBUG
    cout << "Computing PCA" << endl;
#endif // NDEBUG

    const size_t n_pixels = images.rows();
    const size_t n_images = images.cols();
    int32_t K = numComponents;

    mu = images.cast<float>().rowwise().mean(); // Calculate the mean
    along each row

    MatrixXf images_mu = images.cast<float>().colwise() - mu; // Subtract
    means from all columns before doing SVD

#ifdef NDEBUG
    cout << "Calculating the covariance matrix" << endl;
#endif // NDEBUG
    if (n_pixels < n_images) {
        if (K == -1) {
#ifdef NDEBUG
            cout << "Please specify the number of singular values used"
<< endl;
#endif // NDEBUG
            assert(K != -1);
        }
        MatrixXf cov_matrix = images_mu*images_mu.transpose();
#ifdef NDEBUG
        cout << "cov_matrix: " << cov_matrix.rows() << " x " <<
cov_matrix.cols() << endl;
#endif // NDEBUG

        cout << "Calculating the SVD" << endl;
        RedSVD::RedSVD<MatrixXf> svd(cov_matrix, K); // Calculate K
largest singular values, using the JacobiSVD function with this size of
covariance matrix is extremely slow, so beware!
        //cout << svd.singularValues().format(OctaveFmt) << endl;
        U = svd.matrixU();
    } else { // Method based on "Eigenfaces for recognition" by M. Turk
and A. Pentland
        MatrixXf cov_matrix = images_mu.transpose()*images_mu;
#ifdef NDEBUG
        cout << "cov_matrix: " << cov_matrix.rows() << " x " <<
cov_matrix.cols() << endl;
#endif // NDEBUG

        cout << "Calculating the SVD" << endl;
#ifdef NDEBUG
        //JacobiSVD<MatrixXf> svd(cov_matrix, ComputeThinV); // Calculate
singular values
        BDCSVD<MatrixXf> svd(cov_matrix, ComputeThinV); // Calculate
singular values

        if (K == -1) { // Calculate K based on cumulative energy instead
of using hardcoded value - see:
https://en.wikipedia.org/wiki/Principal\_component\_analysis#Compute\_the\_cumulative\_energy\_content\_for\_each\_eigenvector

```

```

    VectorXf S = svd.singularValues(); // Get singular values
    //cout << S.format(OctaveFmt) << endl;
    VectorXf cumulativeEnergy(S.size());
    cumulativeEnergy(0) = S(0);
    for (int i = 1; i < S.size(); i++)
        cumulativeEnergy(i) = cumulativeEnergy(i - 1) + S(i); //
Calculate the cumulative sum of the singular values
    //cout << cumulativeEnergy.format(OctaveFmt) << endl;
    K = 1; // Make sure that we have at least two Eigenfaces -
note that we add one to this value below
    for (; K < cumulativeEnergy.size(); K++) {
        float energy = cumulativeEnergy(K) /
cumulativeEnergy(cumulativeEnergy.size() - 1); // Convert cumulative
energy into percentage
        //cout << energy << endl;
        if (energy >= cumulativeEnergyThreshold) {
            K++; // Since indices start at 0 we need to add one
to the K value
            break;
        }
    }
#ifdef NDEBUG
    cout << "Extracting " << K << " Eigenfaces. Containing " <<
cumulativeEnergyThreshold << " % of the energy" << endl;
#endif // NDEBUG
    if (K > (int32_t)(n_images - 1)) { // Make sure that K is
never equal to n_images
        K = n_images - 1; // K can never be larger than n_images
- 1
#ifdef NDEBUG
        cout << "K was limited to:" << K << endl;
#endif // NDEBUG
    }
#ifdef NDEBUG
    else
        cout << "Using " << K << " Eigenfaces" << endl;
#endif // NDEBUG

    // MatrixXf D = S.block(0, 0, n_images, K); // Extract K largest
values
    // D = D*D / n_pixels; // Calculate eigenvalues
    MatrixXf V = svd.matrixV().block(0, 0, n_images, K); // Extract K
largest values

#ifdef NDEBUG
    cout << "V: " << V.rows() << " x " << V.cols() << " norm: " <<
V.norm() << endl;
#endif // NDEBUG
    U = images_mu*V; // Calculate the actual Eigenvectors of the true
covariance matrix
    U.colwise().normalize(); // Normalize Eigenvectors
}
#ifdef NDEBUG
    cout << "U: " << U.rows() << " x " << U.cols() << " norm: " <<
U.norm() << endl;
#endif // NDEBUG

    return K;
}

```

11.) Membuka file /cpp/FaceRecognitionLib/PCA.h

```

#ifndef __pca_h__
#define __pca_h__
#include <Eigen/Dense> // http://eigen.tuxfamily.org

using namespace Eigen;

class PCA {
public:
    /**
     * Computes the Eigenvectors of the images using PCA.
     * @param images      Each images is represented as a column
vector.
     * @param numComponents Number of singular values used. If this is
set to -1, a cumulative energy threshold of 90 % is used.
     * @return            Returns the number of components used.
     */
    int32_t compute(const MatrixXi &images, int32_t numComponents = -1);

protected:
    MatrixXf U; // Eigenvectors
    VectorXf mu; // Mean along each row

private:
    const float cumulativeEnergyThreshold = .9f; // Determine the number
of principal components required to model 90 % of data variance
};

#endif

```

12.) Membuka file /cpp/FaceLib.cpp

```

#include <jni.h>
#include <Eigen/Dense> // http://eigen.tuxfamily.org
#include <opencv2/core.hpp>
#include <opencv2/core/eigen.hpp>
#include <FaceRecognitionLib/Eigenfaces.h>
#include <FaceRecognitionLib/Fisherfaces.h>
#include <FaceRecognitionLib/Tools.h>
#include <android/log.h>

#ifdef NDEBUG
#define LOGD(...) ((void)0)
#define LOGI(...) ((void)0)
#define LOGE(...) ((void)0)
#define LOG_ASSERT(condition, ...) ((void)0)
#else
#define LOG_TAG "FaceRecognitionAppActivity/Native"
#define LOGD(...) ((void)__android_log_print(ANDROID_LOG_DEBUG, LOG_TAG,
__VA_ARGS__))
#define LOGI(...) ((void)__android_log_print(ANDROID_LOG_INFO, LOG_TAG,
__VA_ARGS__))
#define LOGE(...) ((void)__android_log_print(ANDROID_LOG_ERROR, LOG_TAG,
__VA_ARGS__))
#define LOG_ASSERT(condition, ...) if (!(condition))
__android_log_assert(#condition, LOG_TAG, __VA_ARGS__)
#endif

Eigenfaces eigenfaces;
Fisherfaces fisherfaces;

using namespace std;

```

```

using namespace cv;
using namespace Eigen;

#ifdef __cplusplus
extern "C" {
#endif

JNIEXPORT void JNICALL
Java_com_lauszus_facerecognitionapp_NativeMethods_TrainFaces(JNIEnv,
 jobject, jlong addrImages, jlong addrClasses) {
    Mat *pImages = (Mat *) addrImages; // Each images is represented as a
    column vector
    Mat *pClasses = (Mat *) addrClasses; // Classes are represented as a
    vector

    LOG_ASSERT(pImages->type() == CV_8U, "Images must be an 8-bit
    matrix");
    MatrixXi images;
    cv2eigen(*pImages, images); // Copy from OpenCV Mat to Eigen matrix

    //Facebase *pFacebase;
    if (pClasses == NULL) { // If classes are NULL, then train Eigenfaces
        eigenfaces.train(images); // Train Eigenfaces
        LOGI("Eigenfaccs numComponents: %d", eigenfaces.numComponents);
        //pFacebase = &eigenfaces;
    } else {
        LOG_ASSERT(pClasses->type() == CV_32S && pClasses->cols == 1,
        "Classes must be a signed 32-bit vector");
        VectorXi classes;
        cv2eigen(*pClasses, classes); // Copy from OpenCV Mat to Eigen
        vector
        LOG_ASSERT(classes.minCoeff() == 1, "Minimum value in the list
        must be 1");
        fisherfaces.train(images, classes); // Train Fisherfaces
        LOGI("Fisherfaces numComponents: %d", fisherfaces.numComponents);
        //pFacebase = &fisherfaces;
    }

    /*
    if (!pFacebase->V.hasNaN()) {
        for (int i = 0; i < pFacebase->numComponents; i++) { // Loop
        through eigenvectors
            for (int j = 0; j < 10; j++) // Print first 10 values
                LOGI("Eigenvector[%d]: %f", i, pFacebase->V(j, i));
        }
    } else
        LOGE("Eigenvectors are not valid!");
    */
}

JNIEXPORT void JNICALL
Java_com_lauszus_facerecognitionapp_NativeMethods_MeasureDist(JNIEnv
 *env, jobject, jlong addrImage, jfloatArray minDist, jintArray
 minDistIndex, jfloatArray faceDist, jboolean useEigenfaces) {
    Facebase *pFacebase;
    if (useEigenfaces) {
        LOGI("Using Eigenfaces");
        pFacebase = &eigenfaces;
    } else {
        LOGI("Using Fisherfaces");
        pFacebase = &fisherfaces;
    }

    if (pFacebase->V.any()) { // Make sure that the eigenvector has been
    calculated

```

```

        Mat *pImage = (Mat *) addrImage; // Image is represented as a
column vector

        VectorXi image;
cv2eigen(*pImage, image); // Convert from OpenCV Mat to Eigen
matrix

        LOGI("Project faces");
        VectorXf W = pFacebase->project(image); // Project onto subspace
        LOGI("Reconstructing faces");
        VectorXf face = pFacebase->reconstructFace(W);

        LOGI("Calculate normalized Euclidean distance");
        jfloat dist_face = pFacebase->euclideanDistFace(image, face);
        LOGI("Face distance: %f", dist_face);
        env->SetFloatArrayRegion(faceDist, 0, 1, &dist_face);

        VectorXf dist = pFacebase->euclideanDist(W);

        vector<size_t> sortedIdx = sortIndexes(dist);
        for (auto idx : sortedIdx)
            LOGI("dist[%zu]: %f", idx, dist(idx));

        int minIndex = (int) sortedIdx[0];
        env->SetFloatArrayRegion(minDist, 0, 1, &dist(minIndex));
        env->SetIntArrayRegion(minDistIndex, 0, 1, &minIndex);
    }
}
/*
static inline void convertYUVToRGBA(uint8_t y, uint8_t u, uint8_t v,
uint8_t *buf) __attribute__((always_inline));

static void convertYUVImageToRGBA(const Mat *pYUV, Mat *pRGB) {
    const Size size = pRGB->size();
    const int width = size.width;
    const int height = size.height;
    const int n_pixels = width * height;
    const int rgba_channels = pRGB->channels();

    // See:
https://android.googlesource.com/platform/frameworks/av/+/master/media/li
bstagefright/yuv/YUVImage.cpp,
    // https://wiki.videolan.org/YUV/#Semi-planar
    // and
https://en.wikipedia.org/wiki/YUV#Y.E2.80.B2UV420p\_.28and\_Y.E2.80.B2V12\_o
r\_YV12.29\_to\_RGB888\_conversion
    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
            // U and V channels are interleaved as VUVUVU.
            // So V data starts at the end of Y channel and
            // U data starts right after V's start.
            const int yIndex = x + y * width;
            const uint8_t y_val = pYUV->data[yIndex];

            // Since U and V channels are interleaved, offsets need to be
doubled.
            const int uvOffset = (y >> 1) * (width >> 1) + (x >> 1);
            const int vIndex = n_pixels + 2*uvOffset;
            const int uIndex = vIndex + 1;
            const uint8_t v_val = pYUV->data[vIndex];
            const uint8_t u_val = pYUV->data[uIndex];
            convertYUVToRGBA(y_val, u_val, v_val, &pRGB-
>data[rgba_channels * yIndex]);
        }
    }
}

```

```

}

JNIEXPORT void JNICALL
Java_com_lauszus_facerecognitionapp_NativeMethods_YUV2RGB(JNIEnv,
 jobject, jlong addrYuv, jlong addrRgba) {
    Mat *pYUV = (Mat *) addrYuv; // YUV 4:2:0 planar image, with 8 bit Y
    samples, followed by interleaved V/U plane with 8bit 2x2 sub-sampled
    chroma samples
    Mat *pRGB = (Mat *) addrRgba; // RGBA image

    convertYUVImageToRGBA(pYUV, pRGB);
}

JNIEXPORT void JNICALL
Java_com_lauszus_facerecognitionapp_NativeMethods_HistEQ(JNIEnv, jobject,
 jlong addrYuv, jlong addrRgba) {
    Mat *pYUV = (Mat *) addrYuv; // YUV 4:2:0 planar image, with 8 bit Y
    samples, followed by interleaved V/U plane with 8bit 2x2 sub-sampled
    chroma samples
    Mat *pRGB = (Mat *) addrRgba; // RGBA image

    const Size size = pRGB->size();
    const int width = size.width;
    const int height = size.height;

    // 1. Step: Compute histogram of Y channel
    uint32_t histogram[256];
    memset(histogram, 0, sizeof(histogram));

    for (int y = 0; y < height; y++) {
        for (int x = width / 2; x < width; x++) { // Only look at half
the image
            const int yIndex = x + y * width;
            const uint8_t y_val = pYUV->data[yIndex];
            histogram[y_val]++;
        }
    }

    // Step 2: Compute CDF of histogram
    uint32_t histogram_cdf[256];

    histogram_cdf[0] = histogram[0];
    for (int i = 1; i < 256; i++)
        histogram_cdf[i] = histogram_cdf[i - 1] + histogram[i]; //
Calculate CDF

    for (int i = 0; i < 256; i++)
        histogram_cdf[i] /= histogram_cdf[255] / 255; // Normalize CDF

    // Step 3: Apply histogram equalization
    for (int y = 0; y < height; y++) {
        for (int x = width / 2; x < width; x++) { // Image is flipped
after this function
            const int yIndex = x + y * width;
            const uint8_t y_val = pYUV->data[yIndex];
            pYUV->data[yIndex] = (uint8_t) histogram_cdf[y_val];
        }
    }

    // Step 4: Convert from YUV to RGB
    convertYUVImageToRGBA(pYUV, pRGB);
}

#define clamp(amt,low,high)
((amt)<(low)?(low):((amt)>(high)?(high):(amt)))

```

```
static inline void convertYUVToRGBA(uint8_t y, uint8_t u, uint8_t v,
uint8_t *buf) {
    const int rTmp = y + (int)(1.370705f * (v - 128));
    const int gTmp = y - (int)(0.698001f * (v - 128)) - (int)(0.337633f *
(u - 128));
    const int bTmp = y + (int)(1.732446f * (u - 128));

    buf[0] = (uint8_t) clamp(rTmp, 0, 255);
    buf[1] = (uint8_t) clamp(gTmp, 0, 255);
    buf[2] = (uint8_t) clamp(bTmp, 0, 255);
    buf[3] = 255; // Alpha channel
}
*/
#ifdef __cplusplus
}
#endif
```



BIODATA PENULIS



Nama : Mochammad Dimas Prasetiyo Wibowo
Alamat : Jl. Medokan Asri Barat MA.IH no 33,
Surabaya
Tempat Lahir : Sidoarjo
Tanggal Lahir : 31 Mei 1995
Email : moch.dimas.2015@gmail.com
Nomor Telepon : 083831409377

RIWAYAT PENDIDIKAN :

- SD Laboratorium UNESA, Ketintang Surabaya (2002 – 2008)
- SMP Negeri 29, Prof. Dr. Moestopo Surabaya (2008 – 2011)
- SMA Negeri 10, Jemursari Surabaya (2011 – 2014)
- S1 Sistem Komputer Institut Bisnis dan
Informatika Stikom Surabaya (2014 – Sekarang)

stikom
SURABAYA