APLIKASI IDENTIFIKASI *USER* PADA SISTEM OPERASI ANDROID BERDASARKAN PENGENALAN POLA CITRA OBJEK SEDERHANA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun Oleh:

AFLAHLANA SEPTIAN HABIBINA

NIM. 0610630004 - 63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2011

LEMBAR PERSETUJUAN

APLIKASI IDENTIFIKASI *USER* PADA SISTEM OPERASI ANDROID BERDASARKAN PENGENALAN POLA CITRA OBJEK SEDERHANA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun oleh:

AFLAHLANA SEPTIAN HABIBINA
NIM. 0610630004 - 63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Adharul Muttaqin, ST., MT.	**Ir. Muhammad Aswin, MT.**
NIP. 19760121 200501 1 001	**NIP. 19640626 199002 1 001**

PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi Allah SWT karena atas Taufik, Hidayah, Ridho, Inayah, Mahabbah serta KemanjaanNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "**Aplikasi Identifikasi *User* Pada Sistem Operasi Android Berdasarkan Pengenalan Pola Citra Objek Sederhana** ". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan baik bantuan moril maupun materiil selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Kedua Orang Tua penulis (Djamak Mundhoka dan Eko Juliastuti) dan seluruh keluarga yang senantiasa tiada henti-hentinya memberikan do'a dan restu demi terselesaikannya skripsi ini.
2. Bapak Sholeh Hadi Pramono, DR., Ir., MS. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak M. Azis Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
5. Bapak Waru Djuriatno, ST. MT. sebagai Ketua Kelompok Dosen Keahlian Teknik Informatika dan Komputer Jurusan Teknik Elektro Universitas Brawijaya.
6. Bapak Adharul Muttaqin ST., MT. selaku dosen pembimbing I yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.

7. Bapak Muhammad Aswin, Ir., MT. selaku dosen pembimbing II yang telah bersedia memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
8. Bapak Agung Darmawanshyah, DR., ST., MT selaku dosen pendamping akademik.
9. Bapak dan Ibu Dosen, Staff Administrasi Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
10. Semua Asisten, Ka. Lab serta Laboran dari Laboratorium Teknik Informatika dan Komputer yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan tugas akhir ini.
11. Saudara Anggakara Yudha Pradana, Andik Nur Achmad, Yulius Candra Widyanto ST., Eka Prakarsa Mandyarta atas bantuan, dukungan serta penciptaan suasana persaingan yang kompetitif selama studi di kampus ini.
12. Teman-teman angkatan 2006 (Ge-Force), teman anggota aktif RisTIE dan Workshop dan EME TEUB 2008/2009 dan 2009/2010 atas segala bantuan serta motivasinya selama menjadi mahasiswa.
13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang bisa penulis berikan, semoga Allah SWT tidak pernah melepaskan kemanjaan segala RahmatNya yang tak terhingga buat kita selama-lamanya. Amin

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 24 November 2011

Penulis

DAFTAR ISI

PENGANTAR…….. **i**
DAFTAR ISI……..... **iii**
DAFTAR TABEL……..... **vi**
DAFTAR GAMBAR……... **vii**
ASBTRAK…… **ix**
I PENDAHULUAN……... **1**
1.1 Latar Belakang…… 1
1.2 Rumusan Masalah…….. 1
1.3 Tujuan……. 2
1.4 Batasan Masalah…… 2
1.5 Manfaat……... 3
1.6 Sistematika Penulisan……. 3
II DASAR TEORI……... **5**
2.1 Citra Digital…….. 5
2.1.1 Representasi Citra……... 6
2.2 Dasar Pengolahan Citra Digital……... 7
2.3 Pengenalan Pola…….. 9
2.3.1 Komponen Sistem Pengenalan Pola……. 11
2.3.2 Fitur (*Feature*) dan Pola (*Pattern*)…..……... 13
2.3.3 Siklus Perancangan Sistem Pengenal Pola…… 14
2.4 *Gesture Recognition*……. 15
2.4.1 Asitektur Sistem *Gesture Recogition*…….. 17
2.5 Teori Vektor……..……. 19
2.5.1 Penjumlahan Dua Vektor……... 21
2.5.2 Perkalian Dua Vektor……... 21
2.6 Android…..…… 22
2.6.1 Sejarah Android…… 23
2.6.2 Struktur Platform Android…… 24
2.6.3 Struktur Aplikasi Android…… 26
2.6.4 Struktur Android *Project*…….. 26

III METODE PENELITIAN…… 28
3.1 Studi Literatur……. 28
3.2 Analisis Kebutuhan ……...……. 29
3.3 Perancangan dan Implementasi Sistem……. 29
3.4 Pengujian Sistem…….……... 31
3.5 Pengambilan Kesimpulan……... 31
IV PERANCANGAN DAN IMPLEMENTASI……... 32
4.1 Perancangan Secara Umum…… 32
4.2 Perancangan Perangkat Lunak…… 34
4.2.1 Perancangan Modul Pembuatan Pola Gambar..... 34
4.2.2 Perancangan Pengenalan Pola Gambar..... 36
4.2.2.1 *Pre-processing Unit*……..... 38
4.2.2.2 Perhitungan Jarak Vektor (*minimumCosineDistance)*..…….…….... 41
4.2.2.3 Perancangan Perhitungan Prediksi Nilai Skor 44
4.2.3 Perancangan Program Aplikasi *Pattern Test*.…..…... 47
4.2.3.1 Perancangan Modul Pembanding……... 47
4.2.4 Perancangan Program Aplikasi *Lock-Screen*.…..…..…. 49
4.3 Implementasi Sistem…….…. 51
4.3.1 Lingkungan Implementasi..... 51
4.3.2 Implementasi Modul Pembuatan Pola Gambar..... 52
4.3.2.1 Implementasi Pre-processing …… 59
4.3.2.2 Implementasi Perhitungan Jarak Vektor (*minimunCosineDistance*)……. 61
4.3.2.3 Implementasi Perhitungan Prediksi Nilai Skor 62
4.3.3 Implementasi Program Aplikasi *Pattern Test*..... 63
4.3.4 Implementasi Program *Lock-Sceen*……..... 66
4.4 Implementasi Antarmuka Program…… 71
4.4.1 Implementasi Antarmuka Program Pembuatan Pola Gambar……...…..... 71
4.4.2 Implementasi Antarmuka Program *Pattern Test*.…..... 72
4.4.3 Implementasi Antarmuka Program *Lock-Screen*..... 73

V PENGUJIAN DAN ANALISIS..... 75
5.1 Pengujian Program Aplikasi *Pattern Test*..... 75
5.1.1 Pengujian Program Aplikasi *Pattern Test* Skenario 1.. 75
5.1.2 Pengujian Program Aplikasi *Pattern Test* Skenario 2.. 78
5.1.3 Pengujian Program Aplikasi *Pattern Test* Skenario 3.. 80
5.1.4 Pengujian Program Aplikasi *Pattern Test* Skenario 4.. 83
5.1.5 Pengujian Program Aplikasi *Pattern Test* Skenario 5.. 85
5.2 Pengujian Program Aplikasi *Lock-Screen*..... 88
VI PENUTUP..... 89
6.1 Kesimpulan..... 89
6.2 Saran..... 89
DAFTAR PUSTAKA..... 91

DAFTAR TABEL

Tabel 4.1	Ilustrasi Representasi Vektor Pola Gambar…….	41
Tabel 4.2	Ilustrasi Perhitungan Jarak Sudut Dua Vektor……	44
Tabel 4.3	Ilustrasi Perhitungan Nilai Skor……...	47
Tabel 5.1	Pengujian Program Aplikasi *Pattern Test* skenario 1…….....	77
Tabel 5.2	Pengujian Program Aplikasi *Pattern Test* skenario 2…….....	80
Tabel 5.3	Pengujian Program Aplikasi *Pattern Test* skenario 3…….....	82
Tabel 5.4	Pengujian Program Aplikasi *Pattern Test* skenario 4…….....	84
Tabel 5.5	Pengujian Program Aplikasi *Pattern Test* skenario 5…….....	86
Tabel 5.6	Pengujian Program Aplikasi *Lock-Screen* …….....	88

DAFTAR GAMBAR

Gambar 2.1	Representasi Citra Digital…….	6
Gambar 2.2	Representasi *pixel*……..	7
Gambar 2.3	Kegiatan yang berkaitan dengan citra.....	8
Gambar 2.4	Komponen sistem pengenal pola.....	12
Gambar 2.5	Aspek Fitur…….....	13
Gambar 2.6	Taksonomi *hand gesture-based* didalam HCI……..	16
Gambar 2.7	Proses analisis dan pengenalan terhadap gesture ……..……	17
Gambar 2.8	Arsitektur Sistem *Gesture Recognition*.....	18
Gambar 2.9	Sebuah vektor dari A ke B.....	19
Gambar 2.10	Komponen vektor…….....	20
Gambar 2.11	Ilustrasi vektor satuan…….....	20
Gambar 2.12	Ilustrasi vektor satuan.....	22
Gambar 2.13	Konsep peranan *Open* pada ketiga aktor Android.....	23
Gambar 2.14	Struktur Platform Android.....	24
Gambar 2.15	Lingkungan aplikasi Android……	25
Gambar 3.1	Rancangan umum sistem pengenalan pola pada peralatan berbasis Android……	30
Gambar 4.1	Diagram Blok Sistem Secara Keseluruhan.....	32
Gambar 4.2	Diagram Alir Modul Program Pembuatan Pola Gambar.....	35
Gambar 4.3	Diagram Alir Pengenalan Pola gambar.....	37
Gambar 4.4	Diagram Alir *Pre-Processing Unit*.....	39
Gambar 4.5	Diagram Alir Perhitungan Jarak Dua Vektor (*minimum Cosine Distance*)…….....	43
Gambar 4.6	Diagram Alir Perhitungan Prediksi Nilai Skor……	45
Gambar 4.7	Diagram Alir Modul Pembanding…….....	48
Gambar 4.8	Diagram Alir Aplikasi *Lock-Screen*……..	50
Gambar 4.9	Deskripsi Proses Membaca File Pola Gambar.....	53
Gambar 4.10	Deskripsi Proses Pembuatan Pola Gambar….....	54
Gambar 4.11	Deskripsi Proses Menambah Data Pola Gambar….....	55
Gambar 4.12	Deskripsi Proses Penghapusan Data Pola Gambar.....	56

Gambar 4.13	Deskripsi Menu Pilihan Pergantian Nama dan Penghapusan Pola Gambar.....	57
Gambar 4.14	Deskripsi Pergantian Nama Pola Gambar.....	58
Gambar 4.15	Deskripsi Deklarasi Variabel Pre-Processing Unit……..	59
Gambar 4.16	Deskripsi Kerangka Sampel Pola Gambar…….....	60
Gambar 4.17	Deskripsi Perhitungan Jarak Vektor.....	61
Gambar 4.18	Deskripsi Perhitungan Nilai Skor…….....	63
Gambar 4.19	Deskripsi Menjalankan Aplikasi *Pattern Test*….....	64
Gambar 4.20	Deskripsi Menampilkan Informasi pada Aplikasi *Pattern Test*……..	65
Gambar 4.21	Deskripsi Setting Parameter *Lock-Screen*……....	66
Gambar 4.22	Deskripsi Setting Tampilan Layar *Lock-Screen*	67
Gambar 4.23	Deskripsi Tampilan Layar Proses Unlock….....	68
Gambar 4.24	Deskripsi Tampilan Menu Program *Lock-Screen*.....	69
Gambar 4.25	Deskripsi Logic *Lock-Screen*……....	70
Gambar 4.26	Antarmuka program pembuat pola gambar.....	72
Gambar 4.27	Antarmuka program *Pattern Test*……....	72
Gambar 4.28	Antarmuka program *Lock- Screen*…….....	74
Gambar 5.1	Karakteristik Pengujian Terhadap Pola Gambar.....	87

ABSTRAK

AFLAHLANA SEPTIAN HABIBINA. 2011: Aplikasi Identifikasi *User* pada Sistem Operasi Android Berdasarkan Pola Citra Objek Sederhana. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing: Adharul Muttaqin, ST., MT. dan Ir. Muhammad Aswin, MT.

Identifikasi merupakan proses yang penting untuk mengenali dan membedakan sesuatu hal dengan hal lainnya. Identifikasi ini dilakukan dengan mengenali ciri khas yang dimiliki oleh sesuatu hal tersebut. Untuk melakukan identifikasi ini dapat melalui sebuah mekanisme otentikasi, yaitu mekanisme yang digunakan untuk melakukan validasi terhadap identitas user yang mencoba mengakses sumber daya dalam sebuah sistem komputer atau semacamnya.

Metode otentikasi konvensional yang selama ini familiar digunakan adalah menggunakan kombinasi "username" dan "password" atau biasa juga disebut dengan metode "*single factor authentication*". Namun metode ini dapat digantikan dengan cara memasukkan suatu pola citra tertentu yang dapat dikenali oleh suatu sistem sebagai pengganti "username" dan "password" guna meningkatkan mekanisme pengamanan yang diterapkan dalam peralatan yang bersifat mobile dalam hal ini adalah handphone bersistem operasi Android.

Proses pengenalan pola gambar dilakukan dengan membandingkan pola gambar baru dengan pola gambar yang telah tersimpan pada data pustaka gambar. Hasil proses pengenalan dapat ditunjukkan melalui prediksi nilai skor yang memiliki angka tertentu. Prediksi nilai skor minimal untuk dapat mengenali pola gambar bernilai 1, semakin tinggi nilai skor yang diimplementasikan pada mekanisme otentikasi maka pola gambar yang dibandingkan memiliki tingkat kecocokan yang semakin tinggi. Dari 54 kali pengujian yang dilakukan, sistem dapat melakukan identifikasi *user* sejumlah 90,74%. Kegagalan terjadi karena sistem pengenalan pola gambar tidak bisa mengenali lebih dari satu elemen pola gambar.

Kata kunci : Identifikasi, Otentikasi, Prediksi nilai skor, Android.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Identifikasi merupakan proses yang penting untuk mengenali dan membedakan sesuatu hal dengan hal lainnya. Identifikasi ini dilakukan dengan mengenali ciri khas yang dimiliki oleh sesuatu hal tersebut. Proses identifikasi ini bisa diterapkan dalam suatu sistem perangkat lunak, tak terbatas hanya pada peralatan komputer saja namun telah merambah kepada peralatan yang bersifat *mobile* seperti *handphone*, PDA (*Personal Digital Assistant*) serta peralatan lain yang termasuk dalam kategori *small device.*

Untuk melakukan identifikasi ini dapat melalui sebuah mekanisme otentikasi, yaitu mekanisme yang digunakan untuk melakukan validasi terhadap identitas user yang mencoba mengakses sumber daya dalam sebuah sistem komputer atau semacamnya [001]. Proses otentikasi pada prinsipnya berfungsi sebagai kesepakatan pengguna dan pemberi layanan dalam proses pengaksesan *resource*. Pihak pengguna harus mampu memberikan informasi yang dibutuhkan pemberi layanan untuk berhak mendapatkan *resource*-nya. Sedang pihak pemberi layanan harus mampu menjamin bahwa pihak yang tidak berhak tidak akan dapat mengakses *resource* ini.

Metode otentikasi konvensional yang selama ini familiar digunakan adalah menggunakan kombinasi "*username*" dan "*password*" atau biasa juga disebut dengan metode "*single factor authentication*" [001]. Namun metode ini dapat digantikan dengan cara memasukkan suatu pola gambar tertentu yang dapat dikenali oleh suatu sistem sebagai pengganti "*username*" dan "*password*" guna meningkatkan mekanisme pengamanan.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut:

1. Bagaimana merancang sistem pendeteksian pola citra objek sederhana.

2. Bagaimana implementasi identifikasi pola citra objek sederhana sebagai mekanisme otentikasi untuk dapat mengakses sumber daya yang dapat dijalankan dalam sistem operasi Android.
3. Pengujian dan analisis kerja sistem identifikasi pola citra objek sederhana dalam perangkat *smartphone* berbasis Android.

1.3 Tujuan

Tujuan penulisan skripsi ini adalah merancang dan membuat aplikasi pengenalan pola citra objek sederhana sebagai proses identifikasi yang digunakan untuk mengakses kedalam suatu sumber daya yang diimplementasikan pada perangkat *smartphone* berbasis Android.

1.4 Batasan Masalah

Batasan masalah yang perlu diajukan dalam pengembangan aplikasi perangkat lunak skripsi ini, antara lain:

1. Sistem bekerja mengenali pola citra objek sederhana dan mengidentifikasi pola tersebut melalui perangkat *smartphone.*
2. Memasukkan pola citra objek sederhana melalui media layar sentuh - *smartphone.*
3. Pola citra objek sederhana yang digunakan adalah citra bangun datar dan kurva.
4. Pola citra objek sederhana yang dikenali adalah 1 citra yang terbentuk pertama kali dari awal mula jari menyentuh layar sampai dengan jari dilepaskan dari layar.
5. Platform pengembangan yang digunakan adalah Android OS (*Operating System*)
6. Implementasi dan pengujian menggunakan *smartphone* yang berbasis Android versi 2.1 (Eclair).
7. *Smartphone* menggunakan TFT *Capasitive Touchscreen* dengan resolusi ukuran layar 240 x 320 pixels.

8. IDE (*Integrated Development Environment*) yang digunakan adalah Eclipse.

1.5 Manfaat

Manfaat penelitian ini antara lain:

a. Bagi penulis

1. Menerapkan ilmu yang telah diperoleh dari Teknik Elektro Konsentrasi Teknik Informatika dan Komputer Universitas Brawijaya.
2. Mendapatkan pemahaman tentang perancangan dan pembuatan aplikasi pengenalan pola citra objek sederhana sebagai proses identifikasi yang digunakan untuk mengakses ke dalam suatu sumber daya yang diimplementasikan pada platform Android.

b. Bagi pengguna

1. Menyediakan aplikasi untuk sekuritas pada perangkat *smartphone* yang berbasis Android.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat dan sistematika penulisan.

BAB II Dasar Teori

Membahas teori dasar dan teori penunjang yang berkaitan dengan citra digital, representasi citra, dasar pengolahan citra digital, pengenalan pola, analisa pengolahan citra, serta pengenalan platform Android.

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi

perangkat lunak, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan dan Implementasi

Dalam bab ini menjelaskan langkah-langkah perancangan dan implementasi pembuatan aplikasi untuk megidentifikasi user berdasarkan pengenalan pola citra objek sederhana

BAB V Pengujian

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VI Penutup

Berisi kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran–saran untuk pengembangan lebih lanjut.

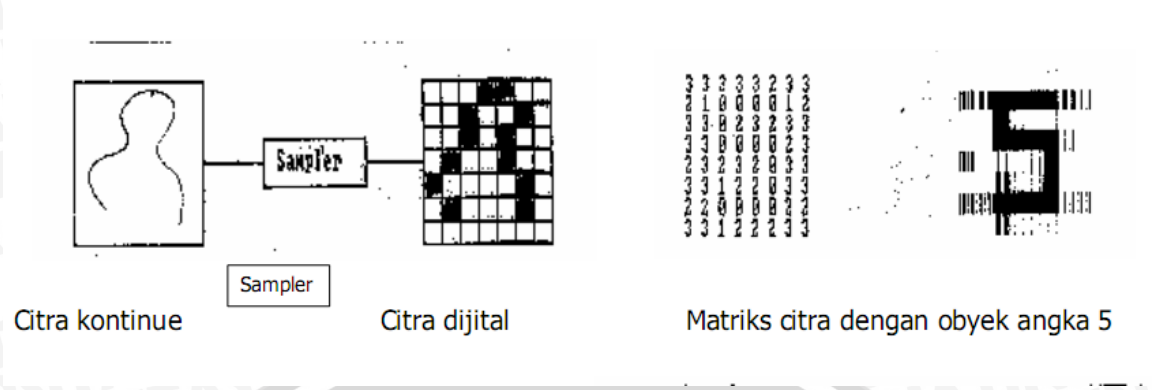
BAB II
DASAR TEORI

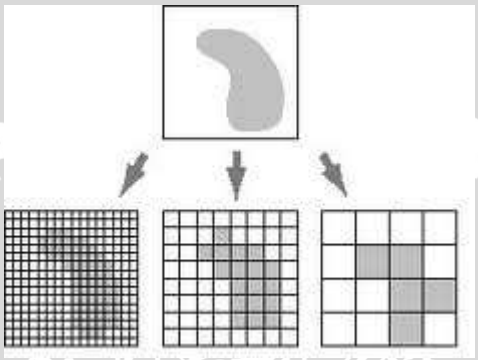
Pada bab ini dibahas mengenai dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai pembuatan aplikasi identifikasi *user* pada sistem operasi Android berdasarkan pengenalan pola citra objek sederhana. Beberapa dasar teori yang dimaksud diantaranya adalah pengetahuan berkaitan dengan citra digital, representasi citra, dasar pengolahan citra digital, pengenalan pola, analisa pengolahan citra, serta pengenalan platform Android.

2.1 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Sebuah citra diubah kedalam bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Istilah "citra" yang digunakan dalam bidang pengolahan citra dapat diartikan sebagai suatu fungsi kontinu dari intensitas cahaya dalam bidang dua dimensi. Proses mengubah citra menjadi digital dapat dilakukan dengan perangkat elektronik misalnya *scanner*, kamera digital, *handycam*, *webcam*, dan lain-lain [004].

Citra digital dapat didefinisikan sebagai fungsi dua variable f (x,y), dimana x dan y adalah koordinat spasial dan nilai I(u,v) adalah intensitas citra pada koordinat tersebut. Citra digital merupakan citra kontinu yang diubah dalam bentuk diskrit, baik koordinat ruang maupun intensitas cahayanya. Pengolahan digitalisasi terdiri dari dua proses, yaitu pencuplikan (sampling) posisi, dan kuantitas intensitas. Citra digital dapat dinyatakan dalam matriks dua dimensi f (x,y) dimana 'x' dan 'y' merupakan koordinat *pixel* dalam matriks dan 'f ' merupakan derajat intensitas *pixel* tersebut .





Gambar 2.1 Representasi Citra Digital

Sumber : http://wiqk.files.wordpress.com/2009/08/REPRESENTASI-CITRA-DIGITAL.jpg

2.1.1 Representasi Citra

Jika kita memperhatikan citra digital secara seksama, kita dapat melihat titik-titik kecil berbentuk segiempat yang membentuk citra tersebut. Titik tersebut merupakan satuan kecil dari suatu citra digital tersebut "*picture element*", "pixel", piksel, atau “pel”. Jumlah piksel per satuan panjang akan menentukan resolusi citra tersebut. Makin banyak piksel yang mewakili suatu citra, maka makin tinggi nilai resolusinya dan makin halus gambarnya. Pada sistem dengan tampilan citra digital yang dirancang dengan baik (beresolusi tinggi), titik-titik kecil tersebut tidak teramati oleh kita yang melihat secara normal.

Citra Digital berbentuk matriks dengan ukuran M x N akan tersusun sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M) \\ \dots & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Suatu Citra *f* (x,y) dalam fungsi matematis dapat dituliskan sebagai berikut:

$0 \le x \le M-1$

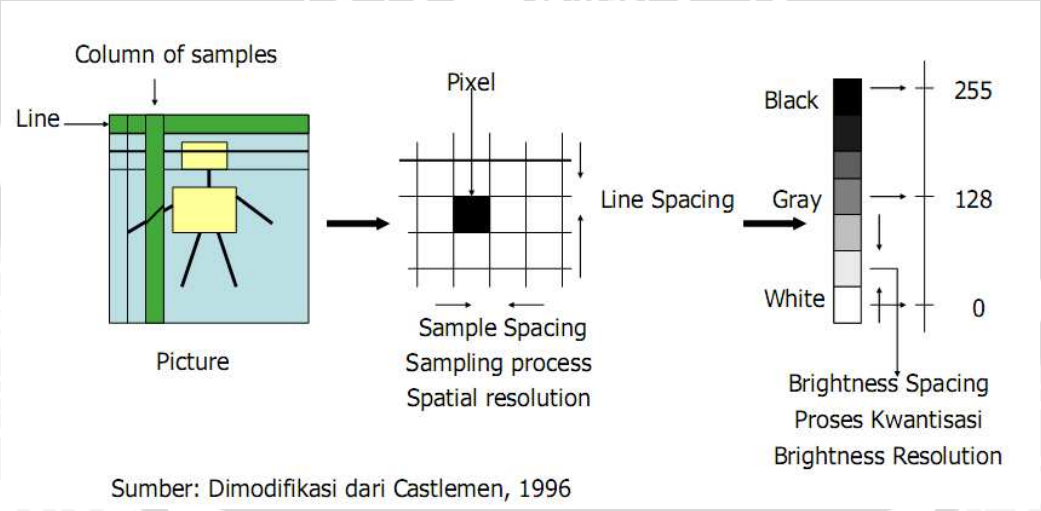
$0 \le y \le N-1$

dimana M : banyaknya baris pada array citra

N : banyaknya kolom pada array citra

Nilai f(x,y) merupakan = i(x,y) . r(x,y)

– Nilai i(x,y) adalah jumlah cahaya yg berasal dari sumbernya (illumination) $0 \le i(x,y) < \infty$
– Nilai r(x,y) adalah derajat kemampuan objek memantulkan cahaya (reflection) $0 \le i(x,y) \le \infty$, Sehingga $0 \le f(x,y) < \infty$



Gambar 2.2 Representasi *pixel*

Sumber : Dimodifikasi dari Castlemen, 1996

2.2 Dasar Pengolahan Citra Digital

Beberapa kegiatan yang berhubungan dengan citra [Idhawati Hestiningsih] :

1. Pencitraan (*imaging*)

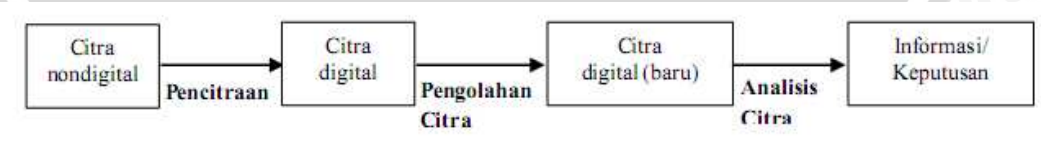
   Pencitraan merupakan kegiatan mengubah informasi dari citra tampak / citra non digital menjadi citra digital. Beberapa alat yang dapat digunakan untuk pencitraan antara lain seperti scanner, kamera digital, dan kamera sinar-x / sinar infra merah, serta peralatan lainnya.

2. Pengolahan Citra

   Pengolahan citra merupakan kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia / mesin (komputer). Masukannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik daripada citra masukan, misal suatu citra warnanya kurang tajam, kabur (*bluring*), mengandung noise (misal bintik -bintik putih) dan lain-lain sehingga perlu ada pemrosesan untuk memperbaiki citra karena citra tersebut menjadi sulit diinterpretasikan karena informasi yang disampaikan menjadi berkurang.

3. Analisis Citra

   Analisis citra merupakan kegiatan menganalisis citra sehingga menghasilkan informasi untuk menetapkan keputusan (biasanya didampingi bidang ilmu kecerdasan buatan atau Aritificial Inteligence yaitu pengenalan pola atau sering disebut *pattern recognition*).



Gambar 2.3 Kegiatan yang berkaitan dengan citra

Sumber : Sitorus, Syahriol dkk, 2006

Dalam teknik elektro dan ilmu komputer, pengolahan citra adalah setiap bentuk pengolahan sinyal dimana *input*-nya adalah gambar, seperti bingkai foto atau video. Output dari pengolahan gambar dapat berupa gambar atau seperangkat

karakteristik atau parameter yang terkait untuk gambar. Teknik pengolahan citra sebagian besar melibatkan memperlakukan gambar sebagai sinyal dua dimensi dan menerapkan teknik-pengolahan standar sinyal untuk itu.

Pengolahan citra digital (*image processing*) merupakan proses pengolahan piksel-piksel dalam citra digital untuk suatu tujuan tertentu yaitu memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin dalam hal ini komputer. Teknik-teknik pengolahan citra yaitu mentransformasi citra menjadi citra yang lain. Dalam pengolahan citra, masukannya adalah citra dan keluarannya juga citra, tetapi citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk dalam bidang ini juga adalah pemampatan citra. Proses pengolahan citra antara lain penghilangan derau (*noise*) dan penapisan (*filtering*) citra [002].

Pengolahan citra digital adalah penggunaan algoritma komputer untuk melakukan pengolahan citra pada gambar digital. Sebagai sub-kategori atau bidang pemrosesan sinyal digital, pengolahan citra digital memiliki banyak keuntungan dibandingkan pengolahan citra analog. Hal ini memungkinkan jangkauan lebih luas dari algoritma yang akan diterapkan pada data input dan dapat menghindari masalah seperti mengurangi kebisingan dan distorsi sinyal selama pemrosesan. Karena gambar didefinisikan lebih dari dua dimensi (mungkin lebih) pengolahan gambar digital dapat dimodelkan dalam bentuk *Multidimensional Systems*.

2.3 Pengenalan Pola

Pengenalan pola adalah suatu aktivitas untuk mengelompokkan data numerik dan simbolik termasuk citra secara otomatis oleh mesin dalam hal ini computer [003]. Tujuan dari pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia dapat mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek yang terdapat di alam, sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba untuk ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang diidentifikasi, memproses citra dan memberikan keluaran berupa deskripsi objek di dalam citra.

Pada pembelajaran mesin, pengenalan pola adalah penugasan semacam nilai output (atau label) untuk nilai masukan yang diberikan kepada beberapa algoritma tertentu. Sebuah contoh dari pengenalan pola adalah klasifikasi, yang mencoba untuk memberikan setiap nilai masukan ke salah satu dari satu perangkat kelas (misalnya, menentukan apakah email yang diberikan adalah "spam" atau "non-spam"). Pengenalan pola umumnya dikategorikan menurut jenis pembelajaran prosedur yang digunakan untuk menghasilkan nilai output. Pembelajaran mengasumsikan bahwa satu set data pelatihan (training set) telah disediakan, yang terdiri dari serangkaian kasus yang telah diberi label dengan output yang benar. Pengenalan pola dipelajari di berbagai bidang, termasuk psikologi, etologi, ilmu kognitif dan ilmu komputer.

Pola adalah entitas yang terdefinisi atau didefinisikan melalui ciri-cirinya (*feature*). Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola yang lainnya. Ciri yang baik adalah ciri yang memiliki daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi. Pola adalah komposit / gabungan dari ciri yang merupakan sifat dari sebuah objek . Beberapa contoh pola:

1. Huruf, memiliki ciri-ciri seperti tinggi, tebal, titik sudut, dan lengkungan garis.
2. Suara, memiliki ciri-ciri seperti amplitudo, frekuensi, nada, dan intonasi.
3. Tanda tangan, memiliki ciri-ciri seperti panjang, kerumitan, dan tekanan.
4. Sidik jari, memiliki ciri-ciri seperti lengkungan, dan jumlah garis.

Ciri-ciri pada suatu pola diperoleh dari hasil pengukuran pada titik objek uji. Khusus pada pola yang terdapat didalam citra, ciri-ciri yang dapat diperoleh berasal dari informasi :

1. Spasial, seperti intensitas piksel dan histogram.
2. Tepi, seperti arah dan kekuatan.
3. Kontur, seperti garis, ellips dan lingkaran.
4. Wilayah/bentuk, seperti keliling, luas dan pusat massa.
5. Hasil transformasi Fourier, seperti frekuensi.

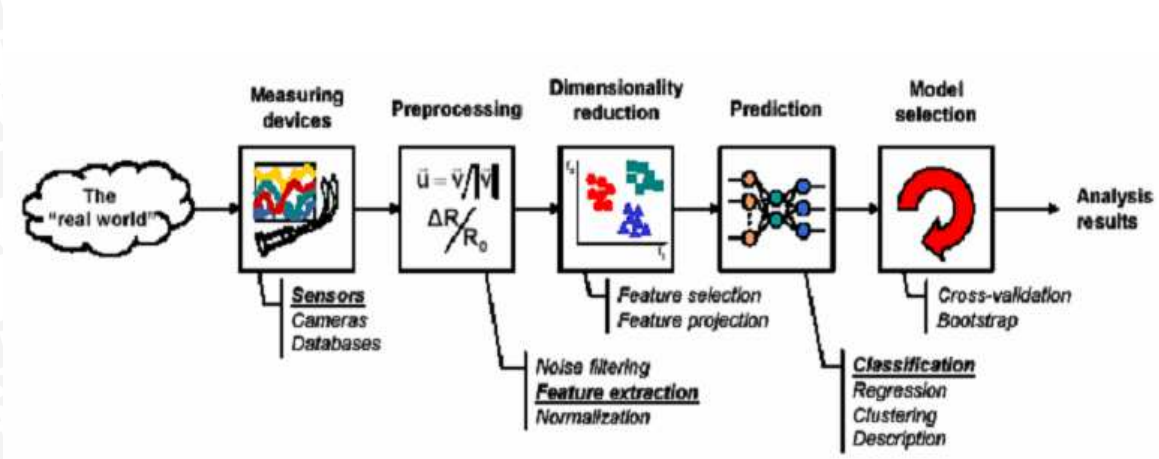
Pengenalan pola bertujuan menentukan kelompok atau kategori pola berdasarkan ciri-ciri yang dimiliki oleh pola tersebut. Dengan kata lain, pengenalan pola membedakan suatu objek dengan objek yang lain [002].

Memang banyak penulis yang bisa menjabarkan definisi tentang pengenalan pola, namun secara garis besar dapat dirangkum bahwa pengenalan pola memetakkan fitur, yang merupakan ciri utama suatu objek (yang dinyatakan dalam sekumpulan bilangan-bilangan) ke suatu kelas yang sesuai. Proses pemetaan ini menyangkut inferensi, baik secara eksplisit-statistik (misalnya menggunakan aturan Bayesian) maupun tak ekplisit dengan suatu jaringan keputusan (misalnya jaringan syaraf tiruan atau logika samar).

Pengenalan pola memainkan peran penting dalam berbagai bidang rekayasa, seperti dibidang *Machine Vision*, pengenalan pola disini dipakai untuk memeriksa hasil produksi secara visual serta pendeteksian sasaran. Dalam bidang pengenalan tulisan, pengenalan pola dapat dipakai dalam otomasi proses penyortiran surat dengan mengenali alamat. Disini suatu penyapu citra menangkap citra digital tulisan tangan dan kemudian sistem pengenalan pola menterjemahkan menjadi karakter. Selain itu dalam bidang kesehatan, pengenalan pola mulai diaplikasikan secara luas dalam dunia kedokteran untuk melakukan diagnosa dan deteksi suatu penyakit tertentu.

2.3.1 Komponen Sistem Pengenalan Pola

Secara mendasar, suatu sistem pengenalan pola terdiri dari komponen-komponen berikut : sensor, mekanisme *pre-processing*, mekanisme pencari fitur (manual / otomatis), algoritma pemilah dan sekumpulan contoh pelatihan yang telah dipilah. Diagram blok dari sistem pengenal pola dapat digambarkan sebagai berikut :



Gambar 2.4 Komponen sistem pengenal pola.

Sumber : Suksmono, Diktat Kuliah Pengenalan Pola, ITB

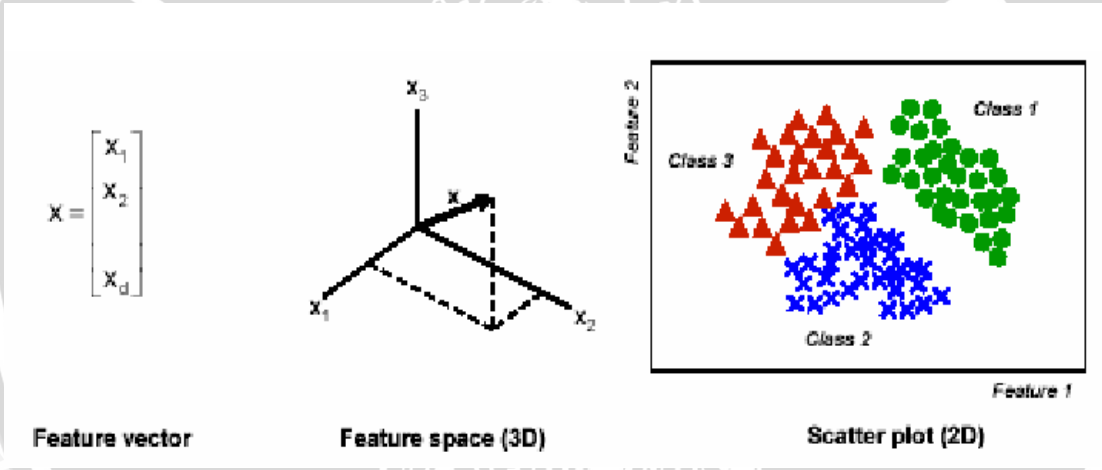
Komponen-komponen tersebut diatas memiliki peran yang berbeda, sesuai dengan fungsinya masing-masing, yaitu :

1. Sensor : berfungsi untuk menangkap objek dari dunia nyata menjadi sinyal-sinyal listrik (dan selanjutnya dalam bilangan-bilangan setelah proses digitalisasi).
2. *Pre-processing :* berfungsi untuk menonjolkan sinyal informasi dan menekan derau dalam sinyal.
3. Pencari Fitur : mengambil besaran komponen tertentu dari sinyal yang mewakili sifat utama sinyal, sekaligus mengurangi dimensi sinyal menjadi sekumpulan bilangan yang lebih sedikit tetapi representatif.
4. Algoritma Pemilah : melakukan *assignment* fitur ke dalam kelas yang sesuai.
5. Sekumpulan contoh pelatihan : digunakan untuk merepresentasikan kelas [005].

2.3.2 Fitur (*Feature*) dan Pola (Pattern)

Fitur adalah segala jenis aspek pembeda, kualitas atau karakteristik. Fitur bisa berwujud simbolik (misalnya warna) atau numeric (misalnya tinggi atau lebar). Ada beberapa definisi terkait dengan fitur sebagai berikut :

- Kombinasi dari *d*-buah fitur dinyatakan sebagai vector kolom dimensi-*d* dan disebut vektor fitur.
- Ruang dimensi-*d* yang dibentuk oleh vektor fitur sebagai ruang fitur.
- Objek dinyatakan sebagai sebuah titik didalam ruang fitur. Penggambaran demikian disebut sebagai diagram hambur (*scatter plot*).



Gambar 2.5 Aspek Fitur : (a) vektor, (b) raung fitur dan (c) diagram hambur fitur.

Pola adalah komposit / gabungan dari fitur yang merupakan sifat dari sebuah objek. Didalam klasifikasi, pola berupa sepasang varibel (**x, ω**), dimana :

- **x** adalah sekumpulan pengamatan atau fitur (vektor label).
- **ω** adalah konsep dibalik pengamatan (label).

Kualitas dari suatu fitur vektor dilihat dari kemampuannya membedakan objek yang berasal dari kelas yang berbeda-beda. Objek dalam kelas yang sama harus punya nilai fitur vektor yang sama dan objek yang berada dalam kelas yang

berbeda harus punya nilai fitur vektor yang berlainan pula. Sifat-sifat dari fitur, linier / non-linier *separability*, korelasi dan modalitas, sangat penting untuk menentukan sistem pengenal yang cocok.

2.3.3 Siklus Perancangaan Sistem Pengenal Pola

Pembuatan suatu sistem pengenal pola yang baik mengikuti beberapa tahapan, yaitu :

- Pengumpulan data.

  Merupakan bagian paling intensif dalam proyek pengenalan pola.

- Pemilihan fitur.

  Pada bagian ini merupakan bagian yang paling kritis dari keberhasilan sistem. Fitur yang baik akan memberikan hasil yang baik pula, begitu juga dengan kebalikannya.

- Pemilihan model.

  Memilih model pengenalan yang sesuai, apakah dengan pendekatan statistik, neural atau struktural. Hal yang harus diperhatikan pula adalah pengaturan parameter.

- Pelatihan.

  Diberikan kumpulan fitur dan model, untuk menentukan adaptasi sistem guna menjelaskan data.

- Pengujian.

  Mengevaluasi seberapa bagus kinerja dari model yang sudah dilatih.

Dalam pengenalan pola ini, terdapat beberapa metode yang bisa digunakan, beberapa metode penting yang perlu dikenal yaitu : metode pola statistik, jaringan syaraf tiruan dan metode sintaktik. Didalam metode statistik, pola dipilah berdasarkan model statistik dari fitur. Model statistik didefinisikan sebagai sebuah keluarga dari fungsi kerapatan peluang bersyarat kelas Pr($x|ci$), yakni peluang vektor fitur *x* jika diberikan kelas c*i*. Dalam metode jaringan syaraf tiruan, pemilihan dilakukan berdasarkan tanggapan suatu neuron jaringan

pengolah sinyal (neuron) terhadap stimulus masukan pola. Dalam metoda sintaktik atau metode struktural, pola dipilah berdasarkan keserupaan ukuran struktural.

2.4 Gesture Recognition (Pengenalan Gerak)

Pengenalan Gerak atau biasa disebut dengan *Gesture Recognition* adalah salah satu topik dalam ilmu komputer dan bahasa teknologi yang bertujuan untuk menafsirkan gerakan manusia melalui algoritma matematika. Gerakan berasal dari setiap gerak tubuh yang terdapat pada manusia. Gerakan (*gesture*) dapat didefinisikan sebagai gerakan fisik tangan, wajah, lengan atau tubuh dengan maksud menyampaikan informasi atau perintah.

Gesture Recognition terdiri dari pelacakan gerakan manusia dan interpretasi gerakan sebagai perintah semantik yang bermakna. Hal ini juga memiliki potensial sebagai suatu alat yang alami dan kuat untuk proses interaksi secara intuitif antara manusia dan komputer. *Gesture Recognition* saat ini telah muncul sebagai salah satu area penelitian paling penting dalam bidang HCI (*Human-Computer Interactions*). *Gesture* telah lama dianggap sebagai salah satu pendekatan yang menjanjikan untuk memungkinkan metode alami dan intuitif tersebut sebagai metode yang bisa meningkatkan interaksi manusia dan komputer didalam domain komputasi yang besar, domain penyelesaian tugas dan aplikasi-aplikasi tertentu. Pertama kali *gesture* diaplikasikan dalam interaksi komputer, adalah oleh Ivan Sutherland, dimana dia mendemonstrasikan SketchPad, merupakan bentuk awal dari *gesture stroke-based* yang menggunakan pena bercahaya dan digunakan untuk memanipulasi objek grafis pada layar Tablet. Kemudian *gesture* dalam bentuk tersebut dapat diterima oleh masyarakat sebagai suatu bentuk interaksi antara manusia dan komputer, yang kemudian diimplementasikan secara luas sebagai input teks pada peralatan *Personal Digital Assistant* (PDA), *mobile-computing*, dan peralatan yang berbasis pena. Sejak saat itu, gagasan menggunakan *gesture* untuk memfasilitasi gaya yang lebih ekspresif dan intuitif dalam interaksi antara manusia dan komputer telah mendapat popularitas diantara para peneliti.

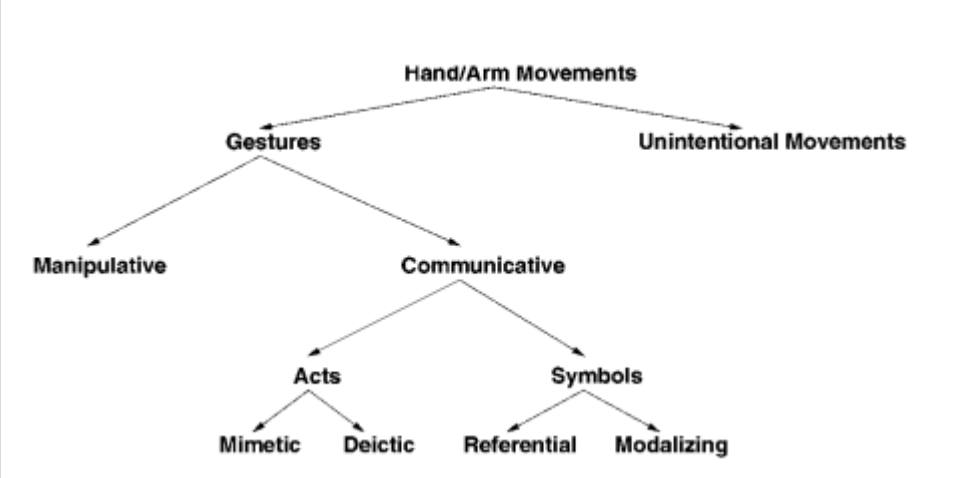
Tujuan utama penelitian tentang *Gesture Recognition* adalah untuk menciptakan sebuah sistem yang dapat mengidentifikasi gerakan manusia tertentu

dan menggunakannya untuk menyampaikan informasi atau mengirim perintah guna mengontrol suatu perangkat.

Secara umum, disaat membahas tentang *Gesture Recognition*, ada beberapa komponen utama yang telah kita diskusikan secara tidak langsung, dan komponen ini memiliki andil besar dalam proses tersebut, yaitu :

1. *Gesture Modeling*

   Dalam rangka pemikiran sistematis untuk membahas literatur tentang interpretasi *gesture*, merupakan hal yang terpenting untuk menentukan model *gesture* seperti apa yang digunakan, contohnya model *hand gesture*. Beberapa alternatif yang telah ditawarkan didalam literatur ketika kita telah memutuskan berkecimpug dalam model ini, gambar berikut menunjukkan taksonomi / pengelompokannya :



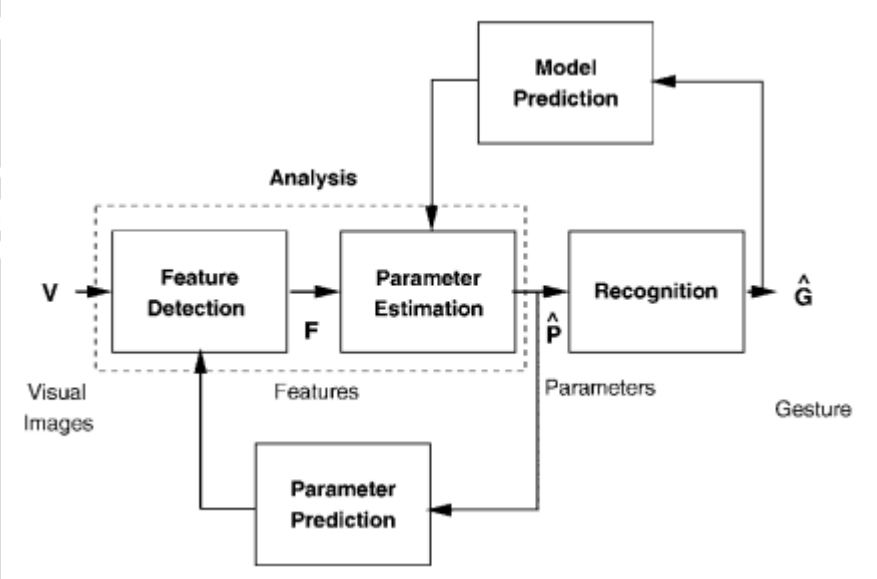
Gambar 2.6 Taksonomi *hand gesture-based* didalam HCI

Sumber : Thomas S. Huang dkk, 1997, IEEE Transaction On Pattern Analysis

Karena gerak tubuh manusia merupakan suatu proses yang dinamis, maka perlu mempertimbangkan karakteristik temporal suatu gerakan. Hal ini memungkinkan bisa membantu dalam segmentasi gerakan secara temporal, atau dengan kata lain setara dengan menentukan interval suatu gerakan. Hal tersebut biasanya disebut dengan istilah *Temporal Modeling of Gesture*.

2. *Gesture Analysis*

Bagian yang mempertimbangkan terhadap tahapan analisis, yang bertujuan untuk memperkirakan parameter dari suatu model yang menggunakan pengukuran dari objek yang telibat dalam HCI. Tugas utamanya adalah mendeteksi atau menggali informasi fitur yang relevan dari gambar mentah dan selanjutnya menggunakan fitur tersebut sebagai parameter yang dapat dikalkulasi. Ada berbagai pendekatan yang dapat digunakan untuk analisis semacam ini.



Gambar 2.7 Proses analisis dan pengenalan terhadap gesture.

Sumber : Thomas S. Huang dkk, 1997, IEEE Transaction On Pattern Analysis

3. *Gesture Recognition*

Suatu fase dimana data dianalisis dari gambar visual suatu gerakan yang diakui secara spesifik, sesuai penjelasan sebelumnya.

4. *Gesture-based System and Application.*

Fase implementasi.

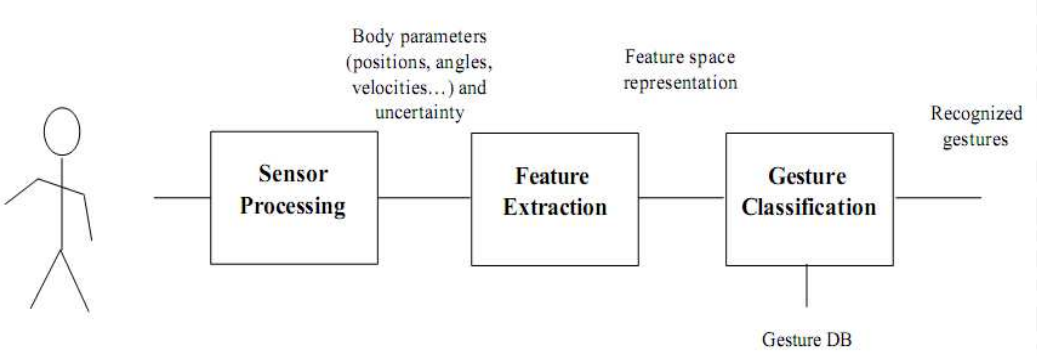
2.4.1 Arsitektur Sistem *Gesture Recognition*

Seperti halnya dengan sistem pengenalan pola , *gesture recognition* ini berisi tiga komponen utama didalamnya, yaitu : akuisisi, ekstraksi ciri (*feature extraction*) dan pengenalan (*recognition*) atau klasifikasi.

Akuisisi adalah langkah pertama dalam sistem *gesture recognition* untuk bisa mendapatkan representatif data asli. Berbagai opsi atau pilihan ditawarkan untuk mengakuisisi suatu *gesture*, misalnya dengan menggunakan sensor *vision* sebagai salah satu sumber yang paling umum digunakan. Biasanya tahap *pre-processing* juga menyertai dalam tahapan akuisisi ini, meskipun dalam skenario lain tahap *pre-processing* ini tidak digunakan.

Ekstraksi Ciri, merupakan tahapan yang mengubah data mentah atau data dari *pre-processing* menjadi bentuk yang lebih bermakna dan siap untuk diproses ke tahapan yang lebih lanjut. Ekstraksi ciri dalam cakupan *gesture recognition* terdiri dari segmentasi komponen gambar yang berkontribusi langsung terhadap pembentukan input *gesture*. Bentuk dasarnya bisa berupa data raster (misalnya : *shape signature, skin tone blobs, colored gloves*) dan informasi vektor (seperti : *joint geometry*, *facial animation parameter*).

Dan pada tahapan akhir yaitu *recognition and classification* merupakan tahap yang digunakan untuk mengklasifikasi fitur gerakan yang sedang dijalankan dengan cara membandingkan *real time* fitur dengan fitur yang telah tersimpan terdahulu dalam suatu kelas *gesture*. Banyak metode yang bisa digunakan untuk mengklasifikasi serta untuk mengenali fitur-fitur seperti ini, bisa menggunakan *Hidden Markov Model, Time Delay Neural Network, Dynamic Time Wraping, Bayesian Classifier, Genetic Algorithms, Fuzzy Interface Engine, Template Matchng* serta metode-metode lainnya (Hafiz Adnan, 2007).



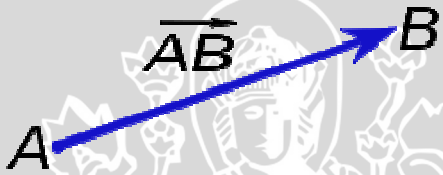
Gambar 2.8 Arsitektur Sistem *Gesture Recognition*

Sumber : Matthew Turk, 2001

2.5 Teori Vektor

Dalam tahapan ekstraksi ciri suatu *gesture recognition* yang terfokuskan pada informasi vektor, maka tak bisa dielakkan bila kita harus membahas dan paham terlebih dahulu dengan teori vektor.

Vektor dalam matematika dan fisika adalah obyek geometri yang memiliki besar dan arah. Vektor jika digambar dilambangkan dengan tanda panah (→). Besar vektor proporsional dengan panjang panah dan arahnya bertepatan dengan arah panah. Vektor dapat melambangkan perpindahan dari titik *A* ke *B*. Vektor sering ditandai sebagai $\overrightarrow{AB}$. Dan memiliki peran penting dalam fisika : posisi, kecepatan dan percepatan objek yang bergerak.



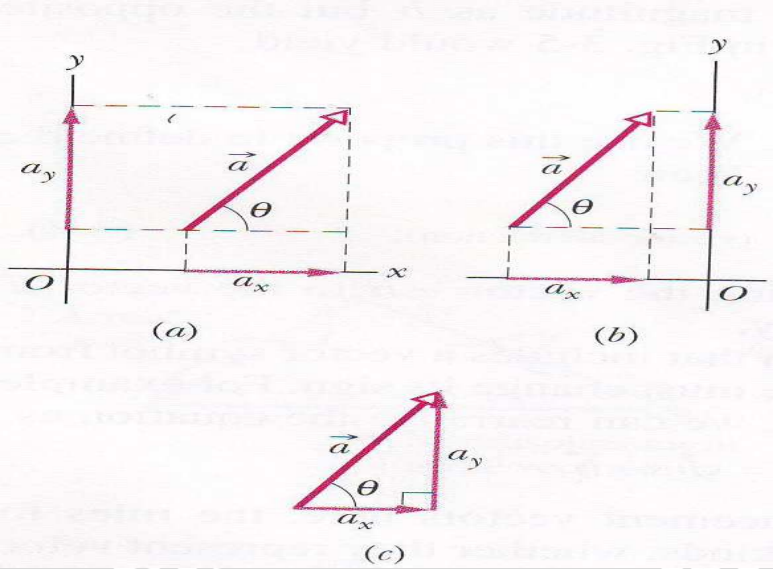
Gambar 2.9 Sebauh vektor dari A ke B

Sumber : http://id.wikipedia.org/wiki/Vektor_(spasial)

Dari gambar diatas dapat dinotasikan bahwa ruas garis berarah tersebut memiliki panjang dan arah tertentu. Dan bila **a** menyatakan ruas garis berarah dari A ke B, maka dapat ditulis dengan lambang **a** = $\overrightarrow{AB}$. Notasi **a** dapat dibaca dengan "vektor a". Penyajian vektor sebagai suatu pasangan bilangan **a** = (a,b), dimana , a : komponen mendatar, b : komponen vertikal.

Komponen vektor merupakan proyeksi vektor pada sumbu sistem koordinat.

$$a_x = a\cos\theta \quad \text{dan} \quad a_y = a\sin\theta$$

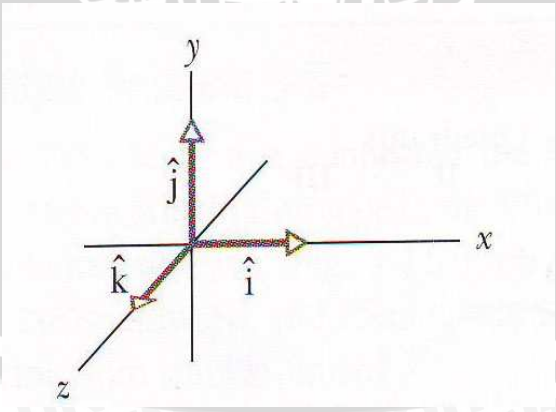


Gambar 2.10 Komponen vektor

Sumber :Marsum Najib, Diktat Matakuliah Kalkulus "Vektor", 2005, UGM

Besar vektor $\vec{a}$, $|a| = \sqrt{a_x^2 + a_y^2}$ dan $\frac{a_x}{a_y} = \tan\theta$.

Vektor satuan *(unit vektor)* merupakan suatu vektor yang telah ternormalisasi yang besarnya = 1. Vektor satuan tidak mempunyai satuan. Vektor satuan berfungsi untuk menunjukan suatu arah dalam ruang. Pada sistem koordinat kartesius (xyz) kita menggunakan vektor satuan **i** untuk menunjukkan arah sumbu x positif, vektor satuan **j** untuk menunjukkan arah sumbu y positif, vektor satuan **k** untuk menunjukkan arah sumbu y positif.



Gambar 2.11 Ilustrasi vektor satuan

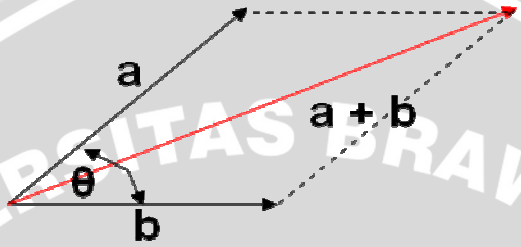
Sumber :Marsum Najib, Diktat Matakuliah Kalkulus "Vektor", 2005, UGM

Sehingga kita dapat menuliskan vektor $\vec{a}$ dan $\vec{b}$ sebagai berikut :

$$\vec{a} = a_x\hat{i} + a_y\hat{j} \qquad \text{dan} \qquad \vec{b} = b_x\hat{i} + b_y\hat{j}$$

2.5.1 Penjumlahan Dua Vektor

Misalnya terdapat dua vektor, **a** dan **b** pada sistem koordinat xy, di mana kedua vektor ini dinyatakan dalam komponen-komponennya, maka untuk menjumlahkan 2 vektor tersebut :



$$|a+b| = \sqrt{|a|^2 + |b|^2 + 2|a||b|\cos\theta}$$

2.5.2 Perkalian Dua Vektor

Perkalian vektor adalah operasi perkalian dengan dua *operand* (obyek yang dikalikan) berupa vektor. Terdapat tiga macam perkalian vektor, yaitu perkalian titik (*dot product*), perkalian silang (*cross product*) dan perkalian langsung (*direct product*).

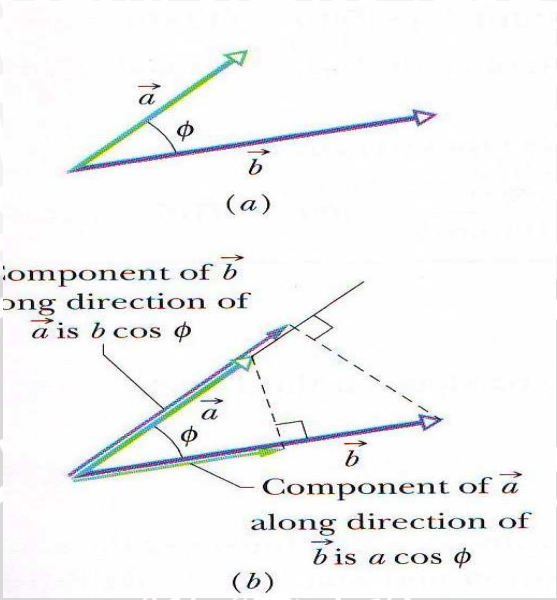
Perkalian titik dua buah vektor akan menghasilkan sebuah skalar. Jenis perkalian ini bersifat komutatif.

$$\vec{A} \cdot \vec{B} = (a_x\hat{i} + a_y\hat{j} + a_z\hat{k}) \cdot (b_x\hat{i} + b_y\hat{j} + b_z\hat{k})$$
$$= a_xb_x + a_yb_y + a_zb_z$$

Untuk vektor satuan terdapat hubungan-hubungan yang khusus dalam operasi perkalian titik, yang merupakan sifat-sifat yang digunakan dalam perkalian titik, yaitu vektor satuan **i**, **j** dan **k** saling tegak lurus satu sama lain, sehingga memudahkan kita dalam perhitungan. Sifatnya adalah **i . i** = **j . j** = **k . k** = (1)(1) cos 0 = 1 dan **i . j** = **i . k** = **j . k** = (1)(1) cos 90^o = 0. Hal ini sesuai dengan komponen vektor masing-masing.

Perkalian titik (*dot product*) **a.b** antara dua vektor a dan b merupakan perkalian antara panjang vektor dan cosinus sudut antara keduanya.

$$\vec{a}.\vec{b} = ab\cos\phi$$

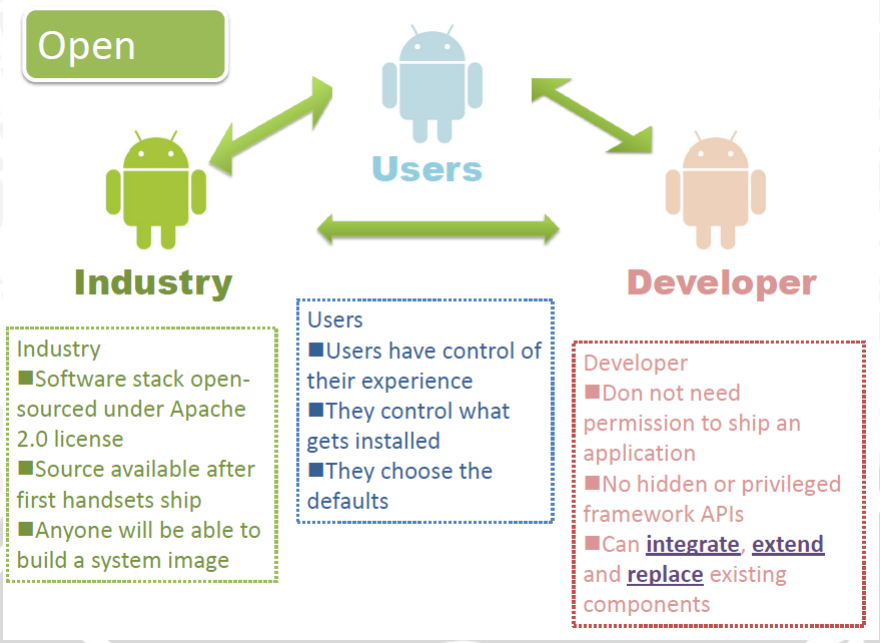


Gambar 2.12 Ilustrasi vektor satuan

Sumber :Marsum Najib, Diktat Matakuliah Kalkulus "Vektor", 2005, UGM

2.6 *Android*

Android adalah sebuah platform yang dikembangkan bersama oleh Google Inc. dan Open Handset Alliance (OHA) yang merupakan sebuah sistem operasi, perangkat perantara (*Middleware*), dan beberapa aplikasi *mobile* utama (*key mobile applications*). Konsep yang dikembangkan adalah sebuah platform yang benar – benar terbuka (*open*) bagi programmer dan pengguna [008].



Gambar 2.13 Konsep peranan *Open* pada ketiga aktor Android

Sumber : Xuguang, Haung, 2009

2.6.1 Sejarah Android

Google melakukan akuisisi pada sebuah perusahaan bernama Android Inc. pada tahun 2005 untuk memulai riset mengenai Android Platform. Beberapa pengembang yang berperan dalam Android Inc. adalah Andy Rubin, Rich Miner, Nick Sears, dan Chris White.

Pada awal 2007, beberapa perusahaan di bidang telekomunikasi membuat dukungan kepada Platform Android dan membentuk Open Handset Alliance (http://www.openhandsetalliance.com). Beberapa anggota dari OHA adalah

- Sprint Nextel
- Motorola
- Sony Ericsson
- Vodafone
- Intel
- T-Mobile
- Samsung
- Toshiba
- Google
- Texas Instruments

Tujuan utama dari persekutuan ini adalah melakukan pengembangan secara berkala dan merespon dengan cepat keinginan konsumer. Hal tersebut

sesuai dengan tujuan dasar Android yang didesain untuk memenuhi kebutuhan operator seluler, produsen perangkat keras, dan pengembang aplikasi.

2.6.2 Struktur Platform Android

Android memiliki 5 struktur dasar yaitu lapisan Linux kernel atau lapisan Driver, lapisan Libraries, Lapisan Android Runtime yang berisi Dalvik Virtual Machine, sebuah lapisan Application Framework, dan lapisan tertinggi adalah aplikasi. Struktur dasar dari platform Android seperti pada gambar 5.

Gambar 2.14 Struktur Platform Android

Sumber : Xuguang, Haung, 2009

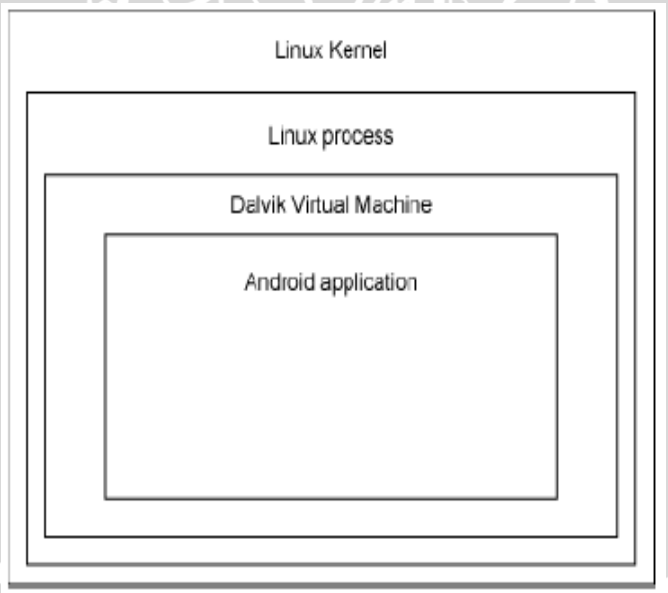
Pada lapisan Kernel, Android menggunakan kernel Linux 2.6.x sebagai inti servisnya. Lapisan ini bertanggungjawab pada driver dari perangkat keras, akses dari *resource*, dan *power management*. Driver yang disediakan meliputi Display, Camera, Keypad, WiFi, Flash Memory, Audio, dan IPC (interprocess communication).

Pada lapisan *Libraries*, terdapat berbagai macam *library*, ditulis dalam bahasa C berdasarkan BSD yang disesuaikan untuk *Embedded Linux-based*

Device. Lapisan ini bertanggungjawab pada *library* dasar yang berjalan pada kernel Linux.

Pada lapisan *Android Runtime*, merupakan kelebihan dari platform Android. *Core Libraries* yang dipanggil memungkinkan berfungsinya bahasa pemrograman Java, namun berjalan pada mesin virtual Dalvik. Mesin Virtual Dalvik adalah mesin virtual yang bertipe Register-based, beda dengan mesin virtual Java yang bertipe Stack-based. Mesin virtual Dalvik mampu menjalankan beberapa mesin virtual sekaligus. Jika sebuah aplikasi java menggunakan Library, maka setiap aplikasi akan berjalan di atas mesin virtualnya sendiri. Selain itu, sebuah mesin virtual yang bertipe *register-based* dapat memproses data yang lebih besar dan lebih cepat daripada mesin virtual *stack-based*.

Pada lapisan framework memuat beberapa API utama sistem yang mencakup telephony, resources, locations, UI, content providers (data), dan package managers (installation, security, dan lainnya). Dengan menggunakan API tersebut, programmer dapat membuat aplikasi yang setingkat dengan aplikasi standar lainnya.



Gambar 2.15 Lingkungan aplikasi Android

Sumber : Xuguang, Haung, 2009

2.6.3 Struktur Aplikasi Android

Arsitektur Android mendorong digunakannya kembali komponen yang memungkinkan kita untuk mempublikasikan dan berbagi aktifitas / kegiatan, layanan dan data antar aplikasi dengan batasan keamanan yang dapat kita definisikan sendiri. Hal ini memungkinkan bagi pengembang untuk memasukkan suatu komponen lain seperti telepon *dialer* atau manajemen *contact*, atau menambahkan fungsionalitas baru kedalam aplikasi yang telah dibuat. Adapun dasar-dasar didalam aplikasi yang dibangun di Android, yaitu :

- *Activity Manager* : bagian yang mengontrol siklus hidup suatu aktivitas / kegiatan. Suatu kegiatan atau aktifitas dapat dibandingkan dengan bentuk tampilan *windows* atau bentuk web yang diusung oleh suatu control (tampilan) dimana membentuk suatu antarmuka, sebuah aktifitas merepresentasikan satu layar tersendiri (*single screen*).
- *Views* : komponen UI (*User Interface*) yang membangun sebuah antarmuka.
- *Notification Manager* : suatu komponen yang menyediakan mekanisme yang selalu konsisten memberitahukan atau mengingatkan kepada pengguna.
- *Content Providers* : memungkinkan beberapa aplikasi utnuk berbagi data diantara mereka.
- *Resource Manager* : seperti konsep sumber daya yang diusung ASP.NET, aplikasi android juga memungkinkan bagi pengembang untuk menyimpan sumber daya seperti *string* atau gambar.

2.6.4 Struktur Android *Project* .

Dalam setiap membangun suatu aplikasi yang diimplementasikan di Android maka akan didapat folder-folder yang berisi :

1. *Src* : berisi semua kode sumber (*source code*) dalam file kelas (*class files*) untuk proyek tersebut.
2. *Gen* : berisi class file R.java. Dimana R.java adalah class file yang secara otomatis dihasilkan untuk memegang referensi bagi setiap sumber daya didalam aplikasi.
3. *Android 2.2* : Namanya berubah sesuai dengan versi SDK yang digunakan (2.2, disini). Berisi class file API (*Application Programming Interface*) Android yang dikemas kedalam file android.jar
4. *Assets* : Didalam folder ini kita dapat menempatkan setiap sumber daya eksternal (file teks, gambar,..) . Lebih jauh lagi dari sekedar folder *res,* folder ini lebih dari sekedar sistem file dimana kita bisa meletakkan file yan gingin kita akses sebagai byte yang baku.
5. *Drawable Folder* : Berisi gambar-gambar yang dipelukan untuk alikasi yang di buat. Didalamnya terdapat tiga buah folder, hal ini dimaksudkan utnuk menyertakan sumber gambar dengan resolusi yang berbeda, sesuai dengan layar telepon yang digunakan.
6. *Layout* : Berisi file *main.xml* yang mendefinisikan pandangan bahwa aplikasi ini membangun suatu *User Interface.*
7. *Values* : Mengandung sumber informasi lainnya yang dapat digunakan untuk aplikasi[009].

BAB III
METODE PENELITIAN

Dalam penyusunan skripsi ini, dirancang suatu pembuatan aplikasi pengenalan pola citra objek sederhana sebagai proses identifikasi yang digunakan untuk mengakses kedalam suatu sumber daya yang diimplementasikan pada perangkat *smartphone* berbasis Android. Metode penelitian yang digunakan pada penyusunan skripsi ini adalah:

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. *Image Processing*
   - Representasi Citra Digital.
   - Dasar Pengolahan Citra Digital.

b. Pengenalan Pola
   - Komponen Pengenalan Pola.
   - Mengenali Fitur dalam pola tertentu.
   - Siklus Perancangan Sistem Pengenalan Pola.

c. *Gesture Recognition*
   - Arsitektur Sistem *Gesture Recognition.*
   - Ekstraksi Ciri pada *Gesture Recognition.*
   - Metode Analisis pada sistem *Gesture Recognition.*

d. Teori Vektor
   - Mengenal Vektor Satuan.
   - Penjumlahan dua vektor.
   - Perkalian antara dua vektor.

e. Dasar Pemrograman Aplikasi pada sistem berbasis Android
   - Struktur Platform Android
   - Struktur Aplikasi Android

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Untuk dapat membuat aplikasi ini diperlukan penentuan alat yang digunakan dalam proses pembuatan dan pengujian, antara lain :

a. Perangkat Keras :

1. PC atau Laptop
2. *Smartphone Android*

b. Perangkat Lunak :

1. Sistem Operasi Windows XP / Vista.
2. Eclipse sebagai IDE (*Integrated Development Environment*).
3. SDK (*Software Development Kit*) Android.
4. ADT (*Android Developer Tools*) *plugin for Eclipse.*

3.3 Perancangan dan Implementasi Sistem

Secara garis besar desain yang dibuat untuk aplikasi identifikasi user pada sistem berbasis Android berdasarkan pengenalan pola citra objek sederhana ini terdiri dari 3 program , yaitu :

- Program aplikasi untuk membuat pola gambar pada sistem berbasis Android.

  Didalam program aplikasi ini terdapat proses bagaimana membuat pola gambar dengan memanfaatkan *class library* yang terdapat pada bahasa pemrograman *Java* didalam sistem berbasis Android. Selain itu terdapat pula proses (*recognizing*) mengenali pola gambar berdasarkan analisis vektor , proses penyimpanan data pola gambar pada media penyimpanan disebut sebagai gambar *template* serta proses memuat kembali data pola agar bisa digunakan pada tahapan proses berikutnya.
- Program aplikasi *Pattern Test*. Program ini untuk menganalisis hasil uji pelatihan yang telah dilakukan terhadap suatu pola gambar

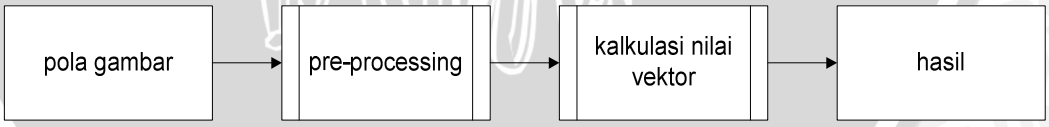
yang telah dibuat sebelumnya dibandingkan dengan pola gambar *real time* yang baru dibuat pada aplikasi ini.

Program aplikasi ini bertujuan untuk menguji unjuk kerja dari sistem *gesture recognition* yang terdapat pada peralatan berbasis Android, sehingga bisa menentukan parameter *confident test* / nilai yang bisa didapatkan sebagai acuan untuk mengimplementasikan proses identifikasi berdasarkan mekanisme otentifikasi pola gambar. Yang perlu dilakukan didalam program aplikasi ini adalah melakukan beberapa uji sampel sehingga nantinya bisa didapatkan prediksi nilai rata-rata (*score rate*), dimana nilai rata-rata inilah yang akan diterapkan pada sistem berikutnya, yaitu program aplikasi *Lock-Screen.*

- Program aplikasi *Lock-Screen*, sebagai implementasi sederhana pengenalan pola citra yang bisa diterapkan dalam pengembangan konsep pengamanan terhadap akses *resource* pada sistem yang berbasis Android.

  Pada program aplikasi inilah proses identifikasi user diterapkan, dengan memanfaatkan parameter hasil prediksi nilai pengenalan pola pada proses sebelumnya.

Berikut ini adalah perancangan secara umum dari sistem identifikasi user pada sistem berbasis Android berdasarkan pengenalan pola citra objek sederhana :



Gambar 3.1 Rancangan umum sistem pengenalan pola pada peralatan berbasis Android

Cara kerja dari rancangan umum sistem ini dapat dijelaskan sebagai berikut :

1. Pengguna membuat pola gambar dengan memanfaatkan layar sentuh yang terdapat pada peralatan *smartphone* Android.

2. Pola gambar tersebut kemudian diproses lebih lanjut dalam tahapan *pre-processing*. Karena model pengenalan pola gambar menggunakan *gesture recognition*, maka didalam tahapan ini diberlakukan langkah untuk membatasi arah pergerakan dalam pembuatan suatu pola gambar. Misalnya, arah pergerakan bisa dari kanan ke kiri atau dari kiri ke kanan, bisa juga dari atas ke bawah atau dari bawah ke atas. Dalam proses ini, parameter yang dibutuhkan adalah untuk membandingkan nilai **x** dan **y** dari setiap segmen pola gambar yang telah dibuat.
3. Dari tahapan sebelumnya telah didapat informasi mengenai vektor satuan serta bisa dihitung panjang vektor yang terjadi. Dengan demikian proses kalkulasi nilai vektor dapat dilanjutkan dengan menggunakan metode minimun *Cosinus Distance* untuk menentukan parameter nilai hasil pengenalan pola.
4. Parameter nilai hasil tersebutlah yang kemudian akan dilakukan proses pencocokan.

3.4 Pengujian Sistem

Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Serta menjelaskan langkah-langkah pengujian dari sistem yang telah dibuat dan membahas hasil pengujiannya.

3.5 Pengambilan Kesimpulan

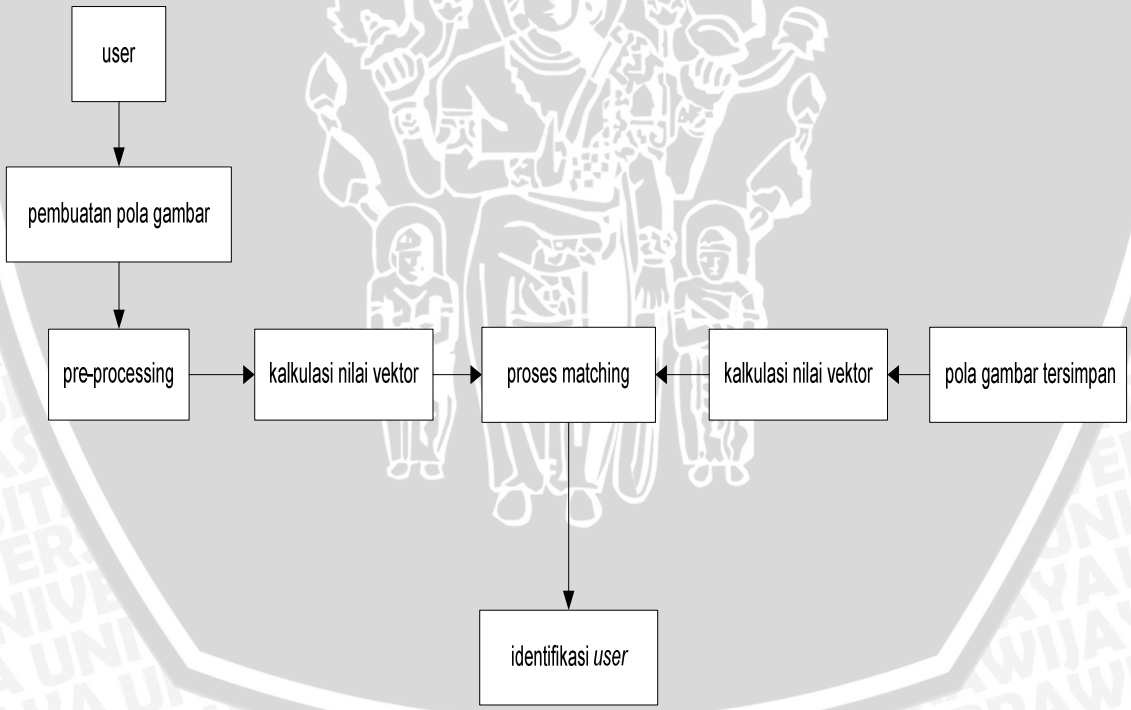
Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil data-data pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah pemberian saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas tahapan-tahapan dalam perancangan aplikasi identifikasi user pada sistem operasi Android berdasarkan pengenalan pola citra objek sederhana. Pembuatan secara bertahap dimaksudkan agar mudah dalam melakukan analisis pada setiap bagiannya maupun secara keseluruhan.

4.1 Perancangan Secara Umum

Perancangan sistem secara umum merupakan tahap awal sebagai acuan dalam perancangan aplikasi yang akan dibuat. Perancangan ini didahului dengan pendefinisian kegiatan pelaku atau user dalam menggunakan program aplikasi identifikasi berdasarkan pengenalan pola citra objek sederhana. Untuk memudahkan pemahaman, perancangan sistem secara keseluruhan dapat disajikan dalam diagram blok sistem kerja aplikasi berikut :



Gambar 4.1 Diagram Blok Sistem Secara Keseluruhan

Diagram blok diatas dapat dijelaskan menjadi langkah-langkah teknis sebagai berikut :

1. Sebuah gambar akan diperoleh melalui mekanisme *drawing* pada media layar sentuh peralatan berbasis Android. Hasil dari proses ini akan disimpan dalam media penyimpanan (*sdcard*) dari peralatan berbasis Android dalam bentuk file biner.
2. Dengan terciptanya gambar tersebut, sistem akan memulai pelatihan untuk mengenali pola dari gambar melalui tahapan *pre-processing*, yaitu untuk menentukan parameter-parameter yang akan digunakan pada tahapan selanjutnya, dengan cara menentukan sampel – sampel data titik (*points*) yang dilewati oleh gerakan tangan pada saat pembuatan pola gambar. Selanjutnya data-data titik tersebut menghasilkan suatu representasi vektor dari pola yang dibuat tersebut. Pada tahapan ini arah pergerakan saat kita menggambar menjadi faktor penentu juga dalam merepresentasikan suatu vektor. Prosedurnya dapat dilakukan dengan menterjemahkan pergerakan mencangkup penentuan *centroid* sebagai pusat asal pergerakan, kemudian melakukan rotasi pergerakan untuk menyelaraskan sudut indikatif dengan orientasi dasarnya.
3. Representasi vektor yang telah didapat, selanjutnya akan dikalkulasi sesuai hasil analisis vektor dengan menggunakan metode perhitungan *cosinus distance*. Nilai yang diperoleh pada proses ini berupa sudut yang terbentuk antara 2 representasi vektor, yaitu vektor pola gambar *template* yang terdapat didalam pustaka file biner dengan pola gambar pergerakan yang baru. Nilai tersebut menjadi acuan untuk menilai hasil pelatihan yang telah dilakukan sistem pengenalan pola ini.
4. Pencocokan dilakukan dengan membandingkan nilai hasil pelatihan dari proses sebelumnya. Tahapan ini, membandingkan nilai hasil pelatihan yang diperoleh dari pola gambar yang telah tersimpan didalam pustaka file biner dengan nilai hasil pelatihan yang diperoleh pada saat pengguna membuat pola gambar yang baru.
5. Identifikasi pengguna ditentukan dari hasil proses pencocokan pada tahap sebelumnya. Apabila hasil pencocokan mendapatkan nilai

parameter yang sesuai maka identifikasi berhasil, begitu pula dengan sebaliknya.

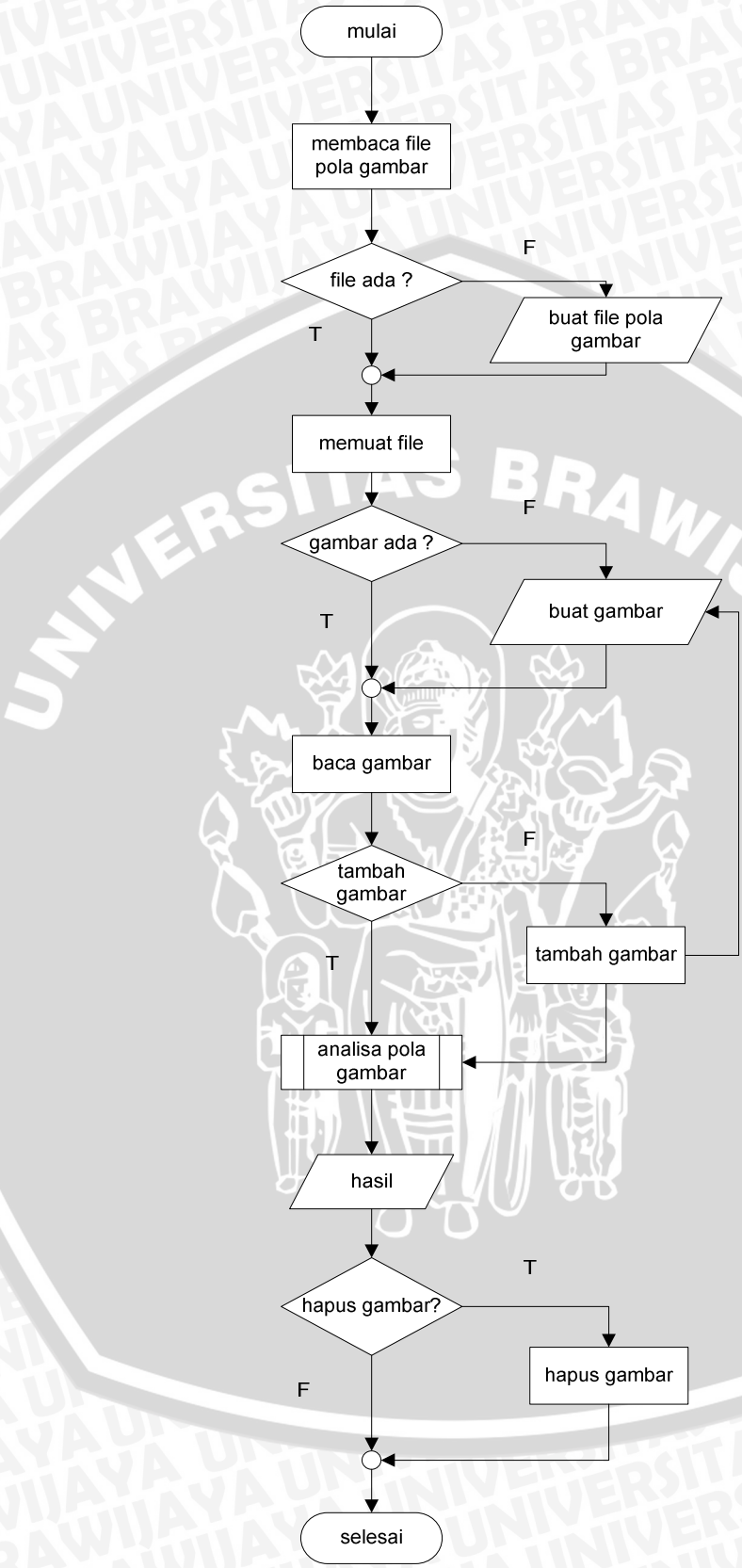
4.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak dipisahkan menjadi tiga bagian utama yaitu perancangan modul program pembuatan pola gambar, perancangan program aplikasi analisa pelatihan serta perencanaan aplikasi implementasi identifikasi. Program aplikasi yang berjalan di peralatan berbasis Android ini dibangun dengan memanfaatkan bahasa pemrograman Java.

4.2.1 Perancangan Modul Program Pembuatan Pola Gambar

Program aplikasi ini terdapat proses bagaimana membuat pola gambar didalam sistem peralatan berbasis Android. Didalamnya terdapat proses (*recognizing*) mengenali pola gambar berdasarkan analisis vektor, proses penyimpanan data pola gambar pada media penyimpanan serta proses memuat kembali data pola gambar [009]. Setiap proses yang ada ini akan dibentuk menjadi suatu program utuh agar bisa digunakan pada tahapan program berikutnya.

Untuk membahas lebih detail tentang perancangan proses-proses dasar aplikasi modul pembuatan pola gambar, diagram alir berikut akan membantu untuk menjelaskannya.



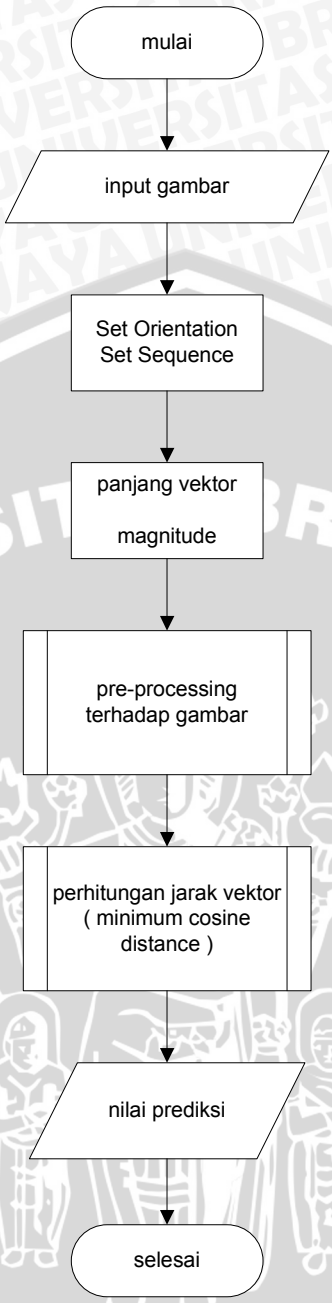
Gambar 4.2 Diagram Alir Modul Program Pembuatan Pola Gambar

Diagram alir Modul Program Pembuatan Pola Gambar dapat dijelaskan sebagai berikut :

1. Membaca pola gambar yang berbentuk file biner pada perangkat penyimpanan *smartphone.*
2. Jika file biner belum tercipta pada perangkat penyimpanan, maka program aplikasi akan membuatkan file .
3. Memuat isi file yang berisi pola gambar.
4. Apabila didalam file belum terdapat pola gambar maka program aplikasi akan melakukan proses pembuatan pola gambar, namun bila sudah terdapat pola gambar maka gambar yang telah terdapat didalam pustaka akan dibaca kembali oleh sistem dan kemudian dilakukan pengenalan (*recognizing*) terhadap pola gambar tersebut.
5. Terdapat proses penambahan pola gambar bila ingin memperbanyak data gambar yang ingin dikenali oleh sistem pengenalan pola gambar.
6. Hasil dari proses pengenalan pola gambar akan didapatkan untuk melakukan identifikasi pada program aplikasi berikutnya.

4.2.2 Perancangan Pengenalan Pola Gambar

Pada bab terdahulu telah disebutkan bahwa untuk mengenali pola gambar digunakan model *gesture recognition* [006]. Sehingga setiap kali gambar yang telah dibuat maka akan dikenali dari segi arah pergerakannya (*direction*). Tidak hanya itu saja akan tetapi untuk lebih mempermudah menganalisis vektornya, pembatasan jumlah goresan komponen (*stroke*) yang membentuk pola gambar pun bisa dilakukan. Tujuannya adalah untuk mendapatkan posisi nilai **x** dan **y** dari setiap segmen pola gambar yang telah dibuat.



Gambar 4.3 Diagram Alir Pengenalan Pola gambar

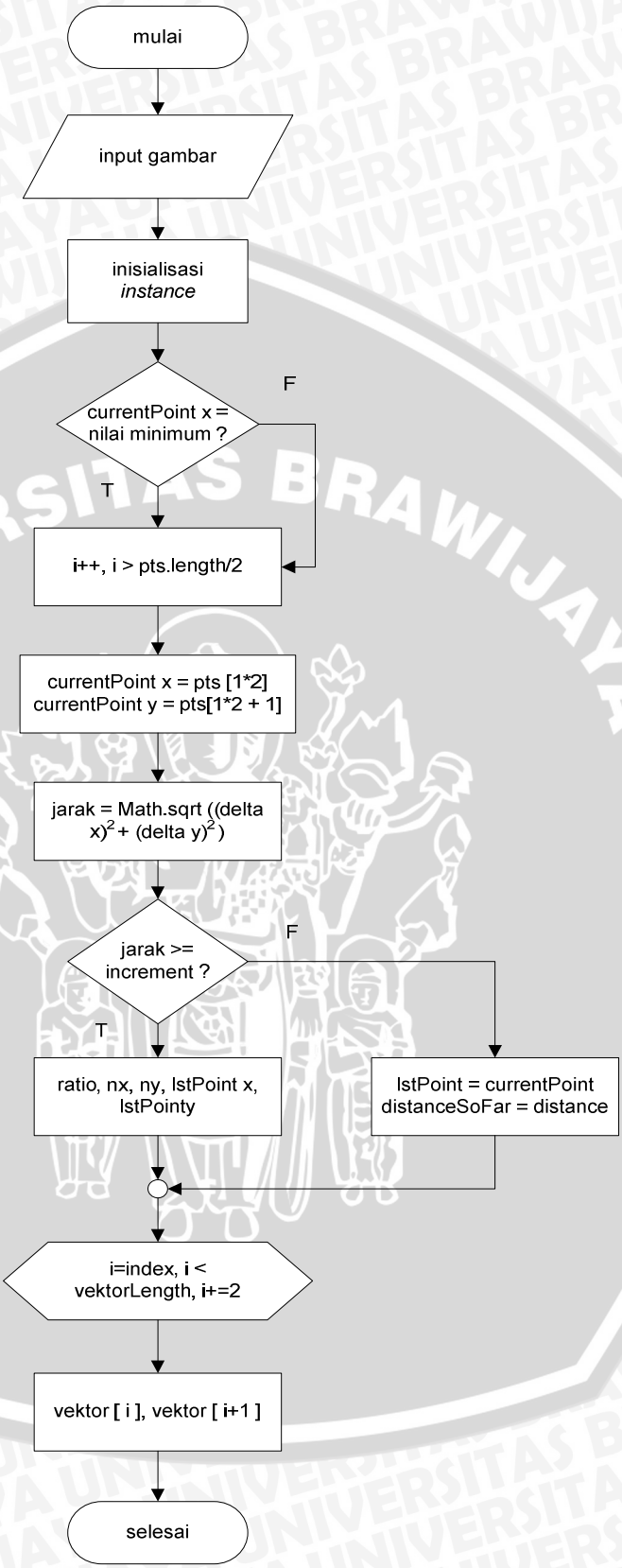
Diagram Alir Pengenalan Pola Gambar dapat dijelaskan sebagai berikut :

1. Setiap komponen gambar yang telah dibuat dan tersimpan didalam perangkat *smartphone* memiliki *label*, *id* dan nilai komponen vektor masing-masing.

2. Dalam proses ini menentukan serta menyelaraskan orientasi indikasi gerakan komponen gambar serta menentukan jumlah arah (*direction*) yang dapat dikenali oleh sistem.
3. Menentukan ukuran dan panjang vektor (*magnitude*) yang terjadi antara titik awal gambar dibuat sampai dengan titik akhirnya.
4. Setelah semua komponen pada proses terdahulu didapatkan, pada proses tahap ini pelatihan terhadap gambar dapat dilakukan. Tahapan ini dinamakan *pre-processing,* tujuannya adalah untuk mendapatkan nilai dari suatu fungsi gambar ***f (x,y)*** pada sebuah bidang diskret (*sampling*).
5. Didalam tahap pelatihan pola gambar ini, perpindahan satu atau semua titik sejauh jarak yang sama dalam satu arah (translasi) serta perbandingan antar kategori dimana masing-masing kategori memiliki bobot nilai yang berbeda (skala), masih dapat dikenali oleh sistem pengenalan pola.
6. Menjalankan proses perhitungan jarak antara 2 vektor melalui perhitungan dengan metode persamaan *minimum Cosinus Distance.*
7. Pengenalan pola ini akan diukur melalui nilai prediksi terhadap hasil pelatihan yang dilakukan terhadap pola gambar yang telah dibuat.

4.2.2.1 Pre-processing Unit (Proses Pelatihan Pada Pola Gambar)

Proses pelatihan ini dilakukan setiap kali terjadi suatu kejadian, kejadian dalam arti bila terdapat gambar yang tercipta melalui proses pembuatan pola gambar (sampling). Setiap kali pola gambar yang tercipta akan diinisalisasi berdasarkan metode *temporal sampling.*

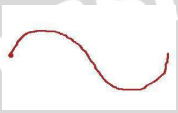


Gambar 4.4 Diagram Alir *Pre-Processing Unit*

Diagram Alir *Pre-Processing Unit* dapat dijelaskan sebagai berikut :

1. Mendeklarasikan variable yang dibutuhkan untuk proses ini.
2. Menentukan kerangka sample, satu set item atau peristiwa yang mungkin untuk diukur antara lain inisialisasi panjang vektor (*vectorLength*), fungsi ***f(x, y)*** dari setiap titik yang dilewati oleh komponen gambar (*currentPointX* dan *currentPointY*) serta menentukan kerangka nilai setiap titik yang dilewati oleh goresan komponen gambar / *stroke* (*pts*).
3. Jika fungsi X ***f(x)*** bernilai minimum dan index i kurang dari jumlah panjang titik yang terlewati, maka nilai index akan bertambah 1 (i++).
4. Mendapatkan nilai fungsi ***f(x, y)*** baru, dimana posisi *currentPointX* dan *currentPointY* telah berubah dari keadaan semula pada index terdahulunya (*lstPointX* dan *lstPointY*).
5. Menentukan jarak antar titik, dimana jarak tersebut diperoleh dari perhitungan posisi titik awal goresan yang terlewati sampai dengan titik keadaan sekarang. *DeltaX = currentPointY – lstPointX* ,dengan perlakuan yang sama dapat diperoleh nilai *DeltaY* pada fungsi pada fungsi Y.
6. Melakukan pengecekan apakah jarak yang telah diperoleh lebih besar nilainya dari parameter *increment*, dimana *increment* = panjang goresan gambar yang tercipta dibagi dengan jumlah titik – 1. Jika keadaan benar maka nilai vektor komponen X dan Y dapat diperoleh. Namun bila tidak, posisi titik sekarang (*currentPoint)* akan menjadi posisi titik dalam indeks terdahulu (*lstPoint).*

Sebagai ilustrasi, dari tahapan *pre-processing unit* inilah akan didapatkan suatu representasi vektor yang tersimpan didalam gambar yang telah terjadi. Dimana representasi tersebut menampilkan informasi nilai titik **x** dan titik **y** yang dilewati oleh komponen gambar.

No	Nama Pola	Pola Gambar	*Length*	Informasi titik
1	Pattern-Template		259	X : 34,34,35,36,36,37,37,38,39,40,40,40,41,41,42,42……dst sampai 259 data Y : 106,105,104,103,102,102,101,101,100,99,98,97,97,96,96,95……dst sampai 259 data
2	Pattern-satu		299	X : 27,28,28,28,28,28,28,29,29,30,31,31,32,33,33……dst sampai 299 data Y : 94,93,92,91,90,89,88,88,87,86,85,84,83,82,81……dst sampai 299 data
3	Pattern-dua		326	X : 313,313,313,313,311,311,310,309,309,309,308,308,308,307,306…dst sampai 326 data Y : 90,91,92,93,94,95,96,97,98,99,100,101,102,102,103…dst sampai 326 data
4	Pattern-tiga		237	X : 26,26,27,28,29,29,30,31,32,33,34,35,35,36,36…dst sampai 237 data Y : 104,103,102,101,100,99,99,99,98,97,97,97,96,96,95…dst sampai 237 data

Tabel 4.1 Ilustrasi representasi vektor pola gambar dengan informasi nilai titik yang dilewati.

4.2.2.2 Perhitungan Jarak Vektor (Minimum Cosine Distance)

Perhitungan jarak antara dua vektor dapat diperoleh berdasarkan representasi vektor dari pergerakan (*gesture*) pada proses sebelumnya. Didalam sistem aplikasi pengenalan ini akan dicari suatu gambar pada *template* yang memiliki kesamaan dengan gambar pergerakan (*gesture*) yang belum diketahui. Sistem ini menggunakan nilai *invers* dari *Cosinus Distance* antar vektor yang dimilikinya yaitu nilai vektor *template* (v_t) dan nilai vektor gambar yang belum diketahui (v_g) sebagai suatu persamaan dari nilai *score* S.

$$S(t,g)=\frac{1}{\arccos\frac{v_t \cdot v_g}{|v_t||v_g|}} \qquad \text{……..(1)}$$

Pada intinya metode *Cosine Distance* mencari sudut antara dua vektor didalam bidang dimensi *n.* Akibatnya, ukuran *gesture*, cerminan besar vektor yang dihasilkan menjadi tidak relevan dengan konsep jarak. Jadi didalam sistem ini skala invarian diperhatikan secara inheren. Jarak kosinus dari dua vektor diwakili oleh hasil kali titik (*dot product*) dari dua vektor (lihat persamaan 2) dibagi dengan perkalian besaran vektornya (lihat persamaan 3).

$$v_t \cdot v_g = \sum_{i=1}^{n}\left(x_{ti}x_{gi}+y_{ti}y_{gi}\right) \qquad \text{…….(2)}$$

$$|v_t||v_g| = \sqrt{\sum_{i=1}^{n}\left(x_{ti}^2+y_{ti}^2\right)}\sqrt{\sum_{i=1}^{n}\left(x_{gi}^2+y_{gi}^2\right)} \qquad \text{……..(3)}$$

Dengan metode seperti ini sebenarnya masih ada sebuah permasalahan namun permasalahan dapat diatasi sesuai dengan solusi yang diberikan oleh Yang Li, dalam jurnal Google Research, *A Fast And Accurate Gesture Recognizer*, Atlanta USA, 2010. Sehingga didapatkan solusi persamaan

$$\theta_{optimal} = \arctan\frac{b}{a} \qquad \text{……(4)}$$

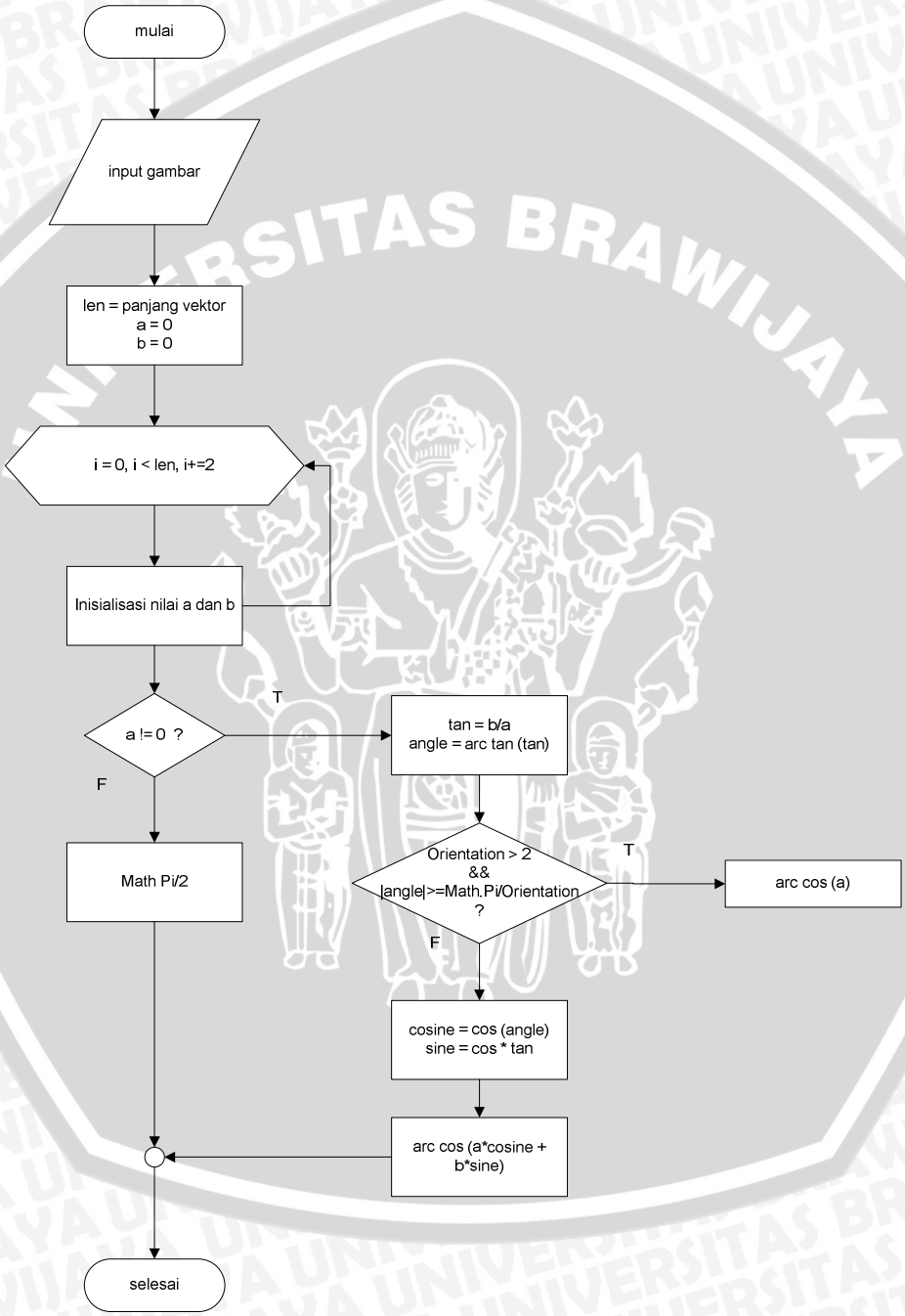
Dimana *a* merupakan hasil *dot product* dari v_t dan v_g (lihat persamaan 2) dan *b* diberikan pada persamaan berikut :

$$b=\sum_{i=1}^{n}\left(x_{ti}y_{gi}-y_{ti}x_{gi}\right) \qquad \text{……..(5)}$$

Dengan menghitung nilai $\theta_{optimal}$, dapat dengan mudah didapatkan persamaan maksimum (invers *minimum Cosine Distance*) antara dua vektor. Kemudian dengan menggunakan persamaan ini dapat menentukan skor untuk

menilai seberapa baik gerakan template *t* mampu memprediksi suatu gerakan yang belum dikenali *g*. Template gerakan yang memiliki skor tertinggi menjadi pilihan utama dalam penentuan parameter pada proses berikutnya.

Berikut kami berikan diagram alir perhitungan jarak dua vektor dengan menggunakan metode *minimum Cosine Distance*.



Gambar 4.5 Diagram Alir Perhitungan Jarak Dua Vektor (*minimum Cosine Distance*)

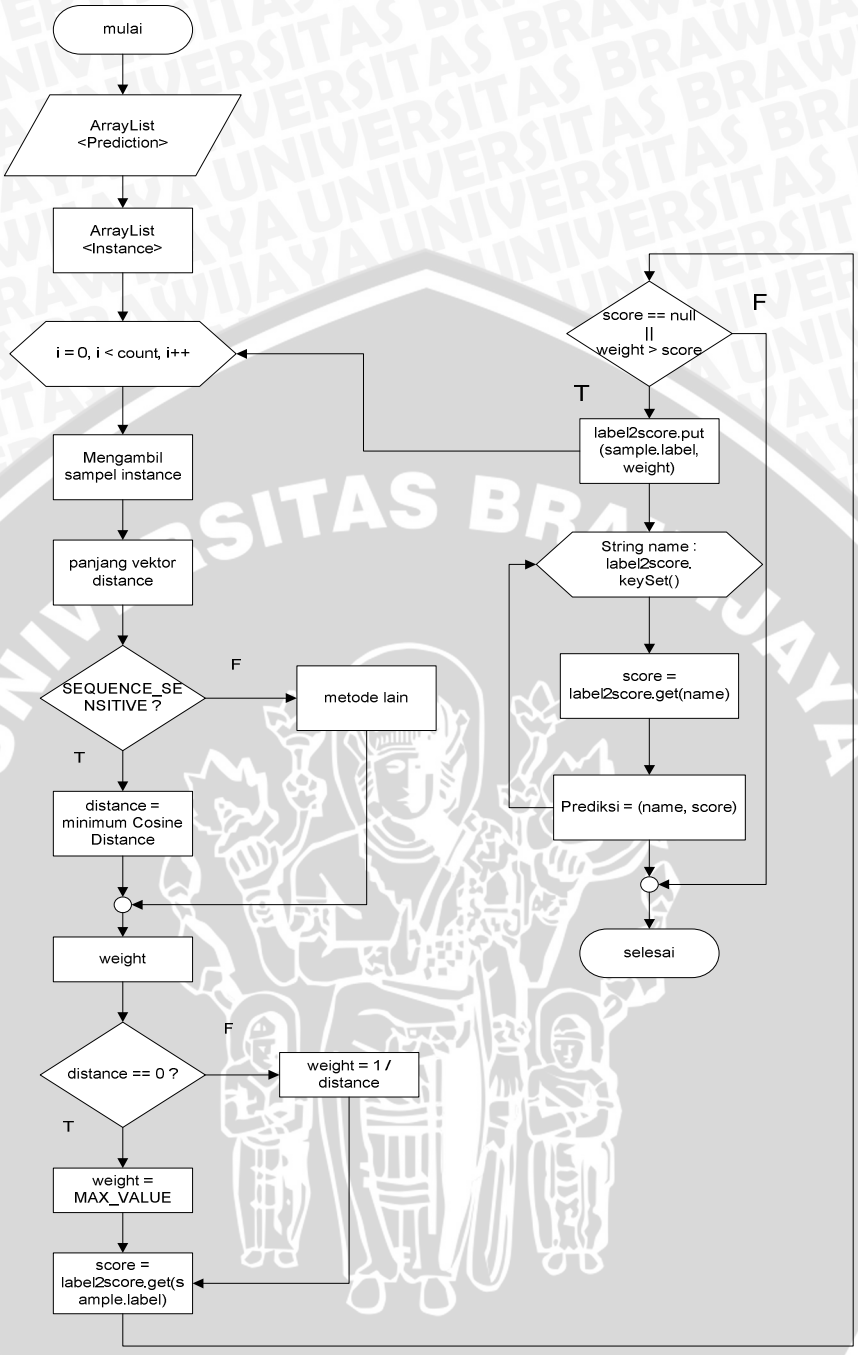
Sebagai ilustrasi perhitungan jarak sudut antara dua vektor yang terjadi antara pola gambar *template* dengan pola gambar *real time* yang belum dikenali, dapat kita tinjau kembali pada tabel 4.1. Dari representasi vektor yang telah didapatkan pada tabel 4.1 maka kita dapat menghitung jarak sudut antara dua vektor tersebut dengan memasukkan nilai informasi titik-titik **x** dan **y** yang ada kedalam rumus-rumus yang telah disediakan. Untuk lebih memahami, berikut tabel yang menjelaskan perhitungan jarak sudut antara dua vektor yang terjadi :

No	Nama Pola Gambar yang Dicari	$\sum_{i=1}^{10} Xt.Xg$	$\sum_{i=1}^{10} Yt.Yg$	$\sum_{i=1}^{10} (Xt.Xg + Yt.Yg)$ (a)	$\sum_{i=1}^{10} Xt.Yg$	$\sum_{i=1}^{10} Yt.Xg$	$\sum_{i=1}^{10} (Xt.Yg - Yt.Xg)$ (b)	$arctan\frac{b}{a} = \theta$
1	Pattern-template (*t*) dengan Pattern-satu (g)	10293	86668	96961	31120	28737	2383	0.0245
2	Pattern-template (*t*) dengan Pattern-dua (g)	113833	96614	210447	34641	318288	-283647	-0.9324
3	Pattern-template (*t*) dengan Pattern-tiga (g)	10694	102348	113042	36557	29722	6835	0.0603

Tabel 4.2 Ilustrasi perhitungan jarak sudut dua vektor.

4.2.2.3 Perancangan Perhitungan Prediksi Nilai Skor

Proses identifikasi berarti menilai seberapa kecocokan pola yang telah dibuat dengan pola baru yang belum diketahuinya. Untuk setiap pola yang telah dibuat ataupun pola baru yang belum diketahui akan dilakukan proses perhitungan prediksi nilai skor berdasarkan perhitungan analisis vektor melalui metode *cosinus distance* seperti yang telah di jabarkan pada tahapan proses sebelumnya.



Gambar 4.6 Diagram Alir Perhitungan Prediksi Nilai Skor

Diagram Alir Perhitungan Prediksi Nilai Skor dapat dijelaskan sebagai berikut :

1. Mendeklarasikan variabel yang dibutuhkan untuk menampung semua prediksi dan data kejadian pola gambar yang terbentuk, didalam ArrayList.

2. Mengambil beberapa sampel pola gambar yang ada didalam file (jika didalam file terdapat lebih dari satu pola gambar) berikut dengan informasi nilai vektor yang ada pada pola gambar tersebut. Kemudian menghitung panjang vektor yang terjadi.
3. Untuk membatasi pola gambar yang bisa dikenali dalam program ini, maka perlu dilakukan seleksi terhadap nilai parameter SEQUENCE_SENSITIVE. Bila nilai SEQUENCE_SENSITIVE bernilai sesuai dengan default yang telah diberikan, maka pola gambar bisa dideteksi dengan menggunakan metode *Cosinus Distance*. Dan bila nilai tersebut diluar nilai default maka metode untuk mengenali pola gambar bisa dilakukan dengan cara yang lain. SEQUENCE_SENSITIVE digunakan untuk membatasi hanya 1 komponen goresan (*stroke*) yang dapat dikenali dalam sistem ini.
4. Dengan menentukan batasan pola gambar sederhana yang bisa dikenali, sistem dengan mudah mengenali pola gambar tersebut dengan menentukan jarak kosinus antara dua vektor yang terjadi. Kemudian proses dilanjutkan untuk menentukan nilai *weight*, dimana nilai ini telah disebutkan didalam persamaan 1 model perancangan perhitungan jarak vektor pada subbab sebelumnya.
5. Nilai *weight* inilah yang dijadikan parameter prediksi nilai skor pada proses pelatihan terhadap pola gambar.
6. Pola gambar yang telah memiliki prediksi nilai skor dapat dikenali sesuai dengan label serta nama yang telah ditentukan sebelumnya. Dengan begitu setiap pola gambar yang telah dikenali dapat dilihat berdasarkan parameter nama dan skor.

Dari penjelasan diatas, kita sudah bisa menghitung prediksi nilai skor yang dapat diambil dari parameter nilai sudut yang telah diilustrasikan pada tabel 4.2 Didalam tahapan ini nilai skor tersebut diindikasikan sebagai *weight*, yaitu satu per nilai sudut yang terjadi. Berikut ilustrasi nilai skor yang diperoleh dari perhitungan sudut pada tabel 4.2:

No	Pola Gambar yang dinilai	$arctan\frac{b}{a} = \theta$	$weight = \lvert\frac{1}{\theta}\rvert$
1	Pattern-template (*t*) dengan Pattern-satu (g)	0.0245	40.816
2	Pattern-template (*t*) dengan Pattern-dua (g)	-0.9324	1.0725
3	Pattern-template (*t*) dengan Pattern-tiga (g)	0.0603	16.583

Tabel 4.3 Ilustrasi perhitungan nilai skor.

4.2.3 Perancangan Program Aplikasi *Pattern Test*

Program ini bertujuan untuk menguji hasil pelatihan yang telah dilakukan oleh sistem pengenalan pola pada gambar. Sama seperti program pada modul pembuatan pola gambar, didalam aplikasi ini juga terdapat fungsi untuk menggambar pola secara langsung dan setelah itu dapat menampilkan informasi hasil pelatihan yang telah dilakukan terhadap pola gambar tersebut.

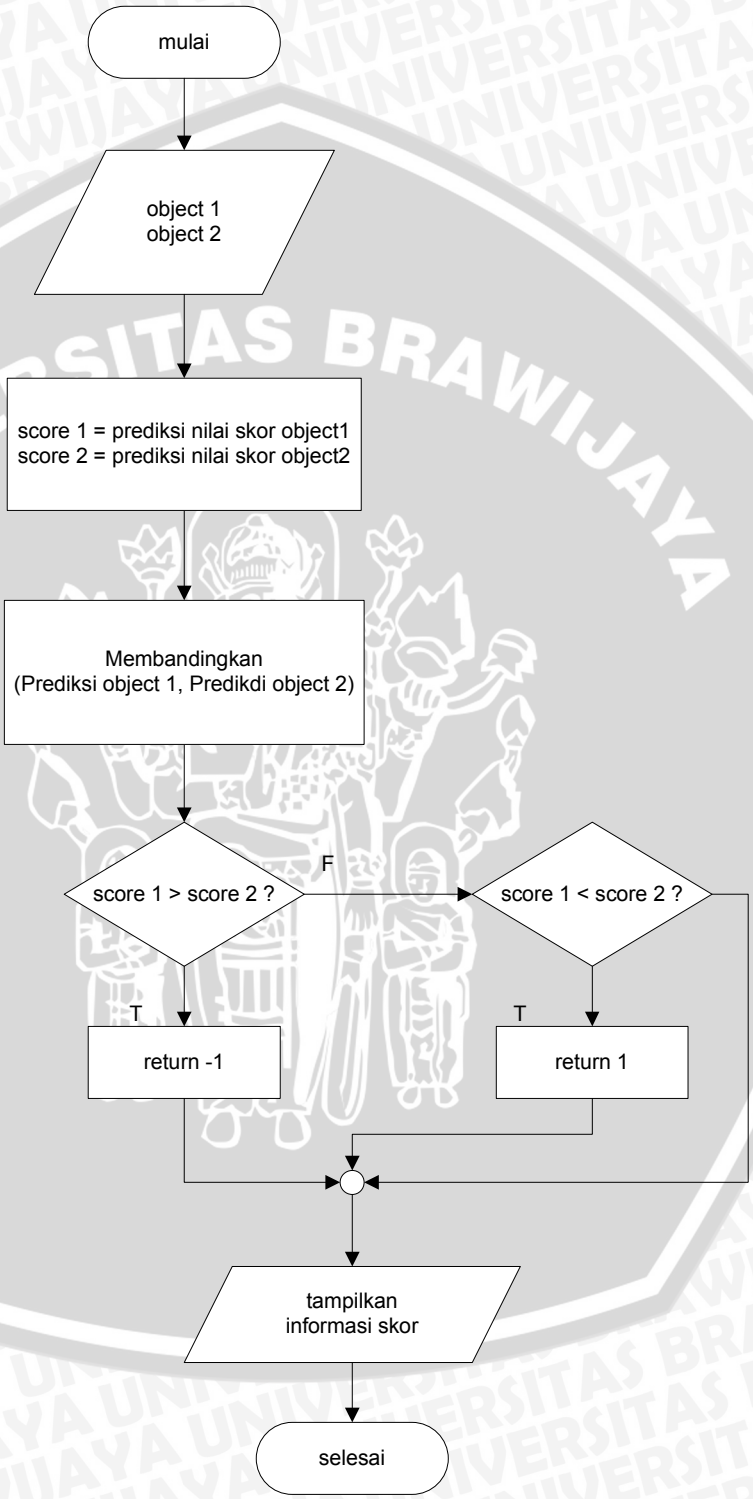
Pada dasarnya semua proses yang ada didalam program aplikasi ini, tahap-tahapan-nya sama seperti proses yang terdapat pada program aplikasi sebelumnya. Dimulai dari fungsi untuk menggambar pola sampai dengan didapatkannya prediksi nilai skor . Namun didalam aplikasi ini terdapat modul program untuk menangkap hasil pelatihan yang telah dilakukan pada proses sebelumnya, dengan tujuan untuk memberikan nilai perbandingan terhadap gambar *template* yang telah tersimpan didalam pustaka file biner dibandingkan dengan pola gambar baru yang tercipta didalam program aplikasi ini.

Dari informasi yang telah diberikan, kita dapat mengambil suatu keputusan untuk mendapatkan nilai rata-rata tertinggi hasil pelatihan terhadap pola gambar tersebut untuk dapat diaplikasikan pada rancangan program aplikasi identifikasi sederhana (*Lock Screen*).

4.2.3.1 Perancangan Modul Pembanding

Pada subbab ini akan dijelaskan mengenai proses penangkapan hasil pelatihan yang telah dilakukan pada tahapan sebelumnya. Untuk dapat menangkap hasil pelatihan diperlukan dua buah objek gambar, seperti yang telah dijelaskan

pada proses sebelumnya yaitu terdapat objek gambar *template* dan satu objek gambar gerakan yang belum diketahui, dan selanjutnya akan dilakukan proses pembandingan melalui parameter prediksi nilai skor.



Gambar 4.7 Diagram Alir Modul Pembanding

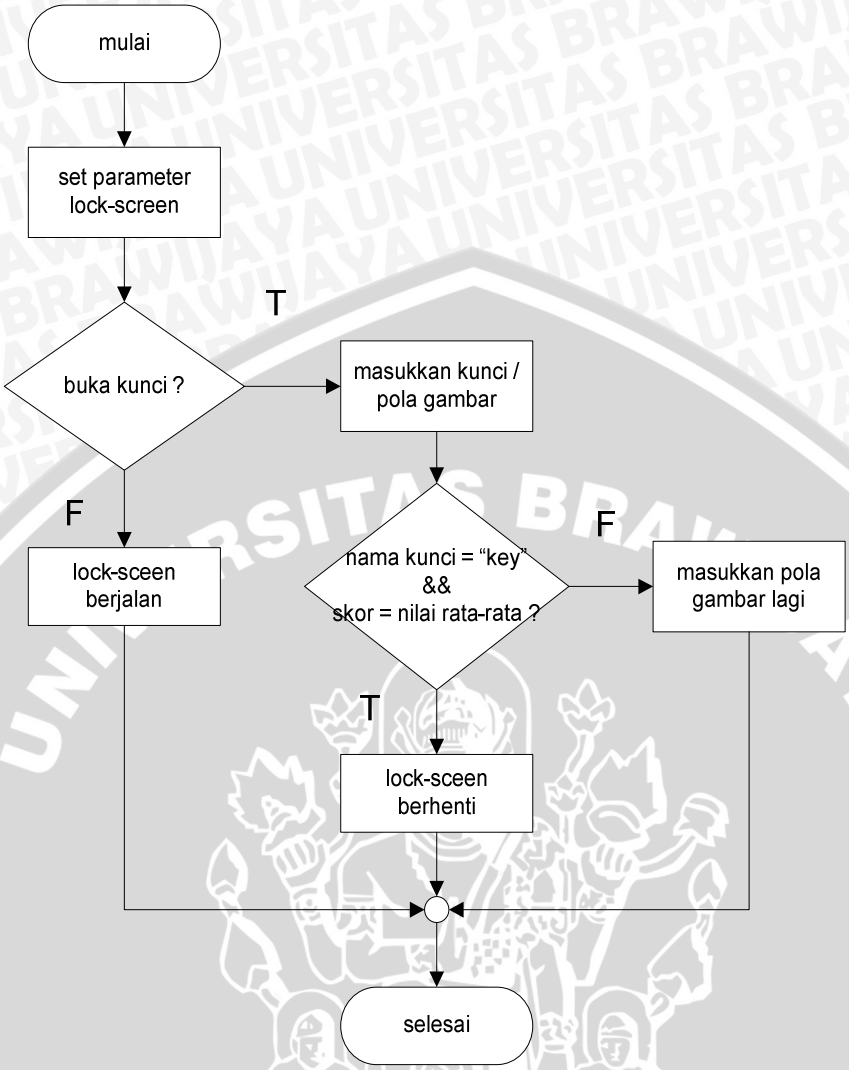
Diagram Alir Modul Pembanding dapat dijelaskan sebagai berikut :

1. Inisialisasi gambar template *t* yang terdapat pada file biner serta gambar yang belum diketahui *g* sebagai suatu objek 1 dan objek 2.
2. Kedua objek gambar tersebut telah memiliki informasi representasi vektor yang didapat pada proses sebelumnya, sehingga dengan mudah kita dapatkan nilai prediksi untuk masing-masing objek. Selanjutnya untuk mendapatkan nilai skor, yang perlu dilakukan adalah membandingkan nilai prediksi dari setiap objek itu dengan objek pembandingnya.
3. Menentukan variabel *score 1* untuk nilai skor dari objek 1, variabel *score 2* untuk nilai skor objek 2.
4. Jika nilai skor objek 1 bernilai lebih besar dari nilai skor objek 2 maka status nilai kembalian sama dengan -1, hal ini berarti tidak terjadi kondisi apapun didalam struktur data pada sistem operasi . Dan jika nilai skor objek 1 lebih kecil dari nilai skor objek 2 maka status kembalian bernilai 1, hal ini berarti proses akan berjalan pada tahapan berikutnya. Namun jika nilai kembalian bernilai 0, berarti semua program telah berjalan dengan sukses.

4.2.4 Perancangan Program Aplikasi *Lock Screen*

Program aplikasi ini merupakan program inovasi yang diciptakan dengan memanfaatkan pengenalan pola gambar yang telah direncanakan pada program aplikasi sebelumnya. Proses identifikasi dapat diterapkan didalam rancangan program ini, karena pada dasarnya program ini akan menjadikan pola gambar yang telah ditentukan sebagai sebuah kunci untuk dapat membuka sistem yang telah dilakukan proses penguncian sebelumnya. Dengan diterapkannya pola gambar sebagai salah satu alternatif identifikasi seperti yang telah direncanakan pada subbab sebelumnya, diharapkan konsep dasar sekuritas pada sistem perangkat berbasis Android dapat dikembangkan lebih luas.

Untuk memperjelas perancangan program aplikasi ini, dapat dilihat pada diagram alir berikut :



Gambar 4.8 Diagram Alir Aplikasi *Lock-Screen*

Diagram Alir Program Aplikasi *Lock-Screen* dapat dijelaskan sebagai berikut :

1. Melakukan set parameter untuk dapat mengunci aktifitas layar pada *smartphone* Android. Artinya ketika keadaan layar telah terkunci maka *user* tidak akan bisa melakukan aktifitas untuk mengakses kedalam tampilan *home* pada sistem perangkat tersebut.
2. Apabila *user* menginginkan untuk membuka sistem kunci pada layar, dapat dilakukan dengan cara memasukkan pola gambar seperti proses yang telah dijabarkan pada subbab pembuatan pola gambar.
3. Apabila pola gambar yang telah ditentukan memiliki parameter nama dengan nilai "key" serta memiliki prediksi nilai rata-rata skor yang telah ditentukan sebelumnya melalui beberapa kali hasil pelatihan,

maka sistem kunci pada layar dapat terbuka kembali, dan *user* bisa melakukan akses terhadap *resource* yang ada pada perangkat *smartphone* tersebut.

4. Bila pola gambar yang dimasukkan tidak sesuai dengan kriteria, maka program aplikasi akan meminta *user* untuk kembali memasukkan pola gambar.

4.3 Implementasi Sistem

Setelah tahap perancangan sistem tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak. Pembahasan terdiri dari lingkungan implementasi (spesifikasi perangkat lunak dan perangkat keras), implementasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

4.3.1 Lingkungan Implementasi

Aplikasi dibuat dengan menggunakan bahasa pemrograman Java. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Perangkat Keras
   - Komputer

     Spesifikasi :
     - Processor : Intel Core 2 Duo T5250 1,5 Ghz
     - Memory : 2038MB RAM
     - Display : Intel 965 Express Chipset Family
     - OS : Windows Vista Home Premium
   - *Smartphone*

     Spesifikasi :
     - Model : GT-I5503
     - CPU : 600 MHz Processor
     - Memori : 170 MB internal
     - Display : TFT, *capasitive touchscreen*, 16M colors, 240 x 320 pixels, 2.8 inches
     - OS : Android OS, v2.1 (Eclair)

- Firmware : 2.1-Update 1
- Kernel : 2.6.29-perf

2. Perangkat Lunak :
   - Sistem operasi : Windows Vista Home Premium
   - Bahasa pemrograman : Java.
   - IDE : Eclipse version 3.6.0
   - Tools Pemrograman : JDK 1.6_0.13
     android-sdk_r06
     ADT 0.9.7

4.3.2 Implementasi Modul Pembuatan Pola Gambar

Modul aplikasi untuk menggambar dan mengenali gerakan pada implementasi dari perancangan kali ini memanfaatkan *class* `GestureOverlayView` yang terdapat pada *package android.gesture* didalam struktur platform Android lapisan *Application Framework*. Tampilan ini mewarisi sifat dari *FrameLayout*, jadi dapat dimanfaatkan pada setiap tampilan layout atau bisa digunakan sebagai *parent layout*. Tampilan ini bertindak sebagai *overlay* / hamparan, dimana *user* dapat melakukan fungsi gambar didalamnya.

Dengan memanfaatkan aplikasi bawaan yang telah terdapat pada sistem berbasis Android mulai dari platform versi 2.1 keatas yang biasa disebut dengan aplikasi *Gesture Builder*, maka proses membuat pola gambar, *loading data*, menghapus data, menambahkan data, mengubah data serta proses pengenalan (*recognizing*) pola gambar dapat dilakukan dengan mudah.

```
DESCRIPTION:
1   public class GestureBuilderActivity extends ListActivity
2    {
3   private final File mStoreFile = new
4    File(Environment.getExternalStorageDirectory(),
5    "gestures");
6    public void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.gesture_list);
9           mAdapter = new GesturesAdapter(this);
10          setListAdapter(mAdapter);
11          if (sStore == null) {
12             sStore =
13   GestureLibraries.fromFile(mStoreFile);
14          }
15          mEmpty = (TextView)
16   findViewById(android.R.id.empty);
17         loadGestures();
18  ……
19
20
```

Gambar 4.9 Deskripsi Proses Membaca File Pola Gambar.

Penjelasan implementasi proses membaca file pola gambar pada Gambar 4.9 yaitu:

1. Baris 3-6, menyatakan bahwa file pola gambar yang tercipta akan disimpan didalam `ExternalStorageDirectory` perangkat bernama *'gestures' (/sdcard/gestures).*
2. Baris 7-20, mendefinisikan metode `onCreate` untuk menyatakan bahwa pertama kali program berjalan akan membaca file yang telah tercipta di direktori penyimpanan untuk dilakukan proses selanjutnya.

DESCRIPTION:

```
1   public class CreateGestureActivity extends Activity {
2    protected void onCreate(Bundle savedInstanceState) {
3            super.onCreate(savedInstanceState);
4
5            setContentView(R.layout.create_gesture);
6            mDoneButton = findViewById(R.id.done);
7
8            GestureOverlayView overlay =
9   (GestureOverlayView)
10  findViewById(R.id.gestures_overlay);
11         overlay.addOnGestureListener(new
12  GesturesProcessor());
13  ……
14
15   private class GesturesProcessor implements
16  GestureOverlayView.OnGestureListener {
17           public void onGestureStarted(GestureOverlayView
18  overlay, MotionEvent event) {
19               mDoneButton.setEnabled(false);
20               mGesture = null;
21           }
22
23           public void onGesture(GestureOverlayView
24  overlay, MotionEvent event) {
25           }
26
27           public void onGestureEnded(GestureOverlayView
28  overlay, MotionEvent event) {
29               mGesture = overlay.getGesture();
30               if (mGesture.getLength() < LENGTH_THRESHOLD)
31  {
32                   overlay.clear(false);
33               }
34               mDoneButton.setEnabled(true);
35           }
36
37
38
39
```

Gambar 4.10 Deskripsi Proses Pembuatan Pola Gambar.

Penjelasan implementasi proses pembuatan pola gambar pada Gambar 4.10 yaitu:

1. Baris 2 – 13, mendefinisikan metode `onCreate` untuk menyatakan bahwa pertama kali program berjalan akan membuat pola gambar dengan memanfaatkan *class* `GestureOverlayView` yang telah dijelaskan pada subbab diatas.
2. Baris 17 – 39, menjelaskan bahwa **`class`** `GesturesProcessor` akan memanfaatkan antarmuka `GestureOverlayView.OnGesture Listener` untuk mendapatkan informasi penanganan terhadap suatu kejadian operasi gerakan tangan pada layar.

DESCRIPTION:

```
1   public void addGesture(View v) {
2         if (mGesture != null) {
3             final TextView input = (TextView)
4   findViewById(R.id.gesture_name);
5             final CharSequence name = input.getText();
6             if (name.length() == 0) {
7
8   input.setError(getString(R.string.error_missing_name));
9                 return;
10            }
11            final GestureLibrary store =
12  GestureBuilderActivity.getStore();
13            store.addGesture(name.toString(), mGesture);
14            store.save();
15            setResult(RESULT_OK);
16            final String path = new
17  File(Environment.getExternalStorageDirectory(),
18                  "Gesture").getAbsolutePath();
19          Toast.makeText(this,
20  getString(R.string.save_success,path),
21  Toast.LENGTH_LONG).show()
22
23
24
25
```

Gambar 4.11 Deskripsi Proses Menambah Data Pola Gambar.

Penjelasan implementasi proses menambah pola gambar pada Gambar 4.11 yaitu:

1. Baris 1, mendeklarasikan metode `addGesture` untuk menambahkan pola gambar kedalam program aplikasi ini.
2. Baris 5-6, menyatakan untuk menambahkan pola gambar dengan mengisikan nama untuk setiap pola, dan harus bernilai, tidak boleh kosong.
3. Baris berikutnya menjelaskan bahwa ketika proses penambahan pola gambar telah berhasil, maka program akan menampilkan status pada layar bahwa proses penambahan telah sukses ditambahkan kedalam program.

Hal serupa terjadi untuk proses penghapusan data pola gambar didalam program aplikasi ini, berikut deskripsinya :

```
DESCRIPTION:
1     private void deleteGesture(NamedGesture gesture) {
2             sStore.removeGesture(gesture.name,
3     gesture.gesture);
4             sStore.save();
5
6
7             final GesturesAdapter adapter = mAdapter;
8             adapter.setNotifyOnChange(false);
9             adapter.remove(gesture);
10            adapter.sort(mSorter);
11            checkForEmpty();
12            adapter.notifyDataSetChanged();
13
14
15            Toast.makeText(this,
16    R.string.gestures_delete_success,
17    Toast.LENGTH_SHORT).show();
        }
```

Gambar 4.12 Deskripsi Proses Penghapusan Data Pola Gambar.

Untuk menampilkan menu pada proses pergantian nama serta penghapusan data pola gambar dapat dijelaskan melalui deskripsi berikut:

```
DESCRIPTION:
1     public void onCreateContextMenu(ContextMenu menu, View
2     v,
3             ContextMenu.ContextMenuInfo menuInfo) {
4             super.onCreateContextMenu(menu, v, menuInfo);
5             AdapterView.AdapterContextMenuInfo info =
6     (AdapterView.AdapterContextMenuInfo) menuInfo;
7             menu.setHeaderTitle(((TextView)
8     info.targetView).getText());
9             menu.add(0, MENU_ID_RENAME, 0,
10    R.string.gestures_rename);
11            menu.add(0, MENU_ID_REMOVE, 0,
12    R.string.gestures_delete);
13        }
14        @Override
15        public boolean onContextItemSelected(MenuItem item)
16    {
17            final AdapterView.AdapterContextMenuInfo
18    menuInfo = (AdapterView.AdapterContextMenuInfo)
19                    item.getMenuInfo();
20            final NamedGesture gesture = (NamedGesture)
21    menuInfo.targetView.getTag();
22            switch (item.getItemId()) {
23                case MENU_ID_RENAME:
24                    renameGesture(gesture);
25                    return true;
26                case MENU_ID_REMOVE:
27                    deleteGesture(gesture);
28                    return true;
29            }
30            return super.onContextItemSelected(item);}
31
32
33
```

Gambar 4.13 Deskripsi Menu Pilihan Pergantian Nama dan Penghapusan Pola Gambar.

Penjelasan implementasi menu pilihan pergantian nama dan penghapusan pola gambar pada Gambar 4.13 yaitu:

1. Baris 1, pendeklarasian metode `onCreateContextMenu` untuk menciptakan pilihan menu pada layar sistem berbasis Android.
2. Baris 8 – 18, menyatakan isi dari menu yang akan ditampilkan, yaitu berupa pilihan untuk mengubah nama pola gambar yang telah tersimpan sebelumnya serta pilihan untuk menghapus pola gambar yang telah tercipta.
3. Baris 21 – 40, menjelaskan fungsi *listener* terhadap menu pilihan yang telah dipilih oleh user, serta menentukan apa yang harus dilakukan oleh program aplikasi.

```
 DESCRIPTION:
1       private Dialog createRenameDialog() {
2           final View layout =
3           View.inflate(this, R.layout.dialog_rename, null);
4           mInput = (EditText) layout.findViewById(R.id.name);
5           ((TextView)   layout.findViewById(R.id.label)).
6            setText(R.string.gestures_rename_label);
7              AlertDialog.Builder builder =
8                new AlertDialog.Builder(this);
9              builder.setIcon(0);
10             builder.setTitle(getString
11                       (R.string.gestures_rename_title));
12             builder.setCancelable(true);
13             builder.setOnCancelListener
14               (new Dialog.OnCancelListener() {
15                 public void onCancel(DialogInterface dialog) {
16                     cleanupRenameDialog();}});
17
18
```

Gambar 4.14 Deskripsi Pergantian Nama Pola Gambar

Deskripsi implementasi pergantian nama pola gambar pada dasarnya ketika pola menu pilihan untuk mengubah nama telah dipilih maka tampilan pada

layar perangkat akan menunjukkan suatu *interface* berupa dialog berisi kolom-kolom untuk mengisikan nama perubahan terhadap nama pola gambar yang dipilih untuk diganti.

4.3.2.1 Implementasi Pre-processing (Proses Pelatihan Pola Gambar)

Implementasi Pre-processing (Proses Pelatihan Pola Gambar) dapat diterapkan dengan memanfaatkan *class* `GestureUtils` didalam *package android.gesture.* Tujuannya adalah untuk melatih sampel gambar yang dibuat dengan metode sampel secara temporal, sehingga dengan adanya pelatihan ini bisa didapatkan representasi fungsi ***f(x, y)*** dari setiap titik yang dilewati oleh komponen gambar.

```
DESCRIPTION:
1       public static float[] temporalSampling(GestureStroke stroke
2             int numPoints) {
3             final float increment = stroke.length / (numPoints -
4               int vectorLength = numPoints * 2;
5             float[] vector = new float[vectorLength];
6             float distanceSoFar = 0;
7             float[] pts = stroke.points;
8             float lstPointX = pts[0];
9             float lstPointY = pts[1];
10            int index = 0;
11            float currentPointX = Float.MIN_VALUE;
12            float currentPointY = Float.MIN_VALUE;
13            vector[index] = lstPointX;
14            index++;
15            vector[index] = lstPointY;
16            index++;
17            int i = 0;
18            int count = pts.length / 2;
19
20
```

Gambar 4.15 Deskripsi Deklarasi Variabel Pre-Processing Unit

```
DESCRIPTION:
1       while (i < count) {
2         if (currentPointX == Float.MIN_VALUE) {
3           i++;
4           if (i >= count) {
5           break;                    }
6          currentPointX = pts[i * 2];
7          currentPointY = pts[i * 2 + 1];                    }
8         float deltaX = currentPointX - lstPointX;
9         float deltaY = currentPointY - lstPointY;
10        float distance =
11             (float) Math.sqrt(deltaX * deltaX +
12             deltaY * deltaY);
13        if (distanceSoFar + distance >= increment) {
14          float ratio = (increment - distanceSoFar) / distance;
15          float nx = lstPointX + ratio * deltaX;
16          float ny = lstPointY + ratio * deltaY;
17           vector[index] = nx;
18           index++;
19           vector[index] = ny;
20           index++;
21           lstPointX = nx;
22           lstPointY = ny;
23           distanceSoFar = 0;
24                } else {
25                 lstPointX = currentPointX;
26                 lstPointY = currentPointY;
27                 currentPointX = Float.MIN_VALUE;
28                 currentPointY = Float.MIN_VALUE;
29                 distanceSoFar += distance;
30               }              }
31            for (i = index; i < vectorLength; i += 2) {
32              vector[i] = lstPointX;
33              vector[i + 1] = lstPointY;
34                }
35           return vector;
```

Gambar 4.16 Deskripsi Kerangka Sampel Pola Gambar

Dengan adanya implementasi kerangka sampel seperti yang telah dideskripsikan , maka informasi vektor untuk setiap komponen **x** dan **y** telah kita dapatkan. Sample titik-titik tersebut direperesentasikan dalam bentuk [x1, y1, x2, y2,…., xn, yn] .

4.3.2.2 Implementasi Perhitungan Jarak Vektor (*minimum Cosine Distance*)

Implementasi perhitungan jarak vektor dapat diterapkan dengan memanfaatkan *class* `GestureUtils` didalam *package android.gesture*.

DESCRIPTION:

```
1   static float minimumCosineDistance(float[] vector1,
2       float[] vector2, int numOrientations) {
3       final int len = vector1.length;
4       float a = 0;
5       float b = 0;
6       for (int i = 0; i < len; i += 2) {
7          a += vector1[i] * vector2[i] + vector1[i + 1]
8                  * vector2[i + 1];
9          b += vector1[i] * vector2[i + 1] - vector1[i + 1]
10                 * vector2[i];     }
11         if (a != 0) {
12          final float tan = b / a;
13          final double angle = Math.atan(tan);
14           if (numOrientations > 2
15               && Math.abs(angle) >= Math.PI / numOrientations) {
16               return (float) Math.acos(a);
17               } else {
18                 final double cosine = Math.cos(angle);
19                 final double sine = cosine * tan;
20                 return (float) Math.acos(a * cosine + b * sine);
21               }
22          } else {
23            return (float) Math.PI / 2;}
24
25
26
```

Gambar 4.17 Deskripsi Perhitungan Jarak Vektor

Penjelasan implementasi perhitungan jarak vektor pada Gambar 4.17 yaitu:

1. Baris 1 – 6, mendeklarasikan variabel untuk fungsi / metode `minimumCosineDistance`.
2. Baris 8 – 11, mendefinisikan nilai `a` mewakili hasil *dot product* dari dua vektor yang terjadi. Sedangkan nilai `b` mewakili perkalian besar vektor antara dua vektor tersebut.
3. Baris 14 – 16, perhitungan sudut yang terjadi antara dua vektor.
4. Baris 17 – 27, menentukan nilai kosinus serta menentukan jumlah orientasi yang dapat dikenali oleh sistem perhitungan jarak vektor tersebut.

4.3.2.3 Implementasi Perhitungan Prediksi Nilai Skor

Implementasi untuk mendapatkan prediksi nilai skor memanfaatkan *class InstanceLearner* yang ada didalam *package Android.Gesture.*

```
 DESCRIPTION:
1   ArrayList<Prediction> classify(int sequenceType,
2      int orientationType, float[] vector) {
3   ArrayList<Prediction> predictions = new ArrayList<Prediction>()
4      ArrayList<Instance> instances = getInstances();
5      int count = instances.size();
6      TreeMap<String, Double> label2score =
7                     new TreeMap<String, Double>();
8        for (int i = 0; i < count; i++) {
9          Instance sample = instances.get(i);
10         if (sample.vector.length != vector.length) {
11             continue;
12                   }
13         double distance;
14         if (sequenceType == GestureStore.SEQUENCE_SENSITIVE) {
15            distance = GestureUtils.minimumCosineDistance(
16                  sample.vector, vector, orientationType);
17         } else {
18
19
```

```
20                distance = GestureUtils.squaredEuclideanDistance(
21                                    sample.vector, vector);
22                            }
23                double weight;
24                if (distance == 0) {
25                weight = Double.MAX_VALUE;
26               } else {
27                weight = 1 / distance;
28              }
29             Double score = label2score.get(sample.label);
30             if (score == null || weight > score) {
31                label2score.put(sample.label, weight);
32                   }
33                }
34
35              for (String name : label2score.keySet()) {
36                  double score = label2score.get(name);
37
38                  predictions.add(new Prediction(name, score));
39                  }
40
41
42                  Collections.sort(predictions, sComparator);
43
44                   return predictions;
45               }
46
47
```

Gambar 4.18 Deskripsi Perhitungan Nilai Skor.

4.3.3 Implementasi Program Aplikasi *Pattern Test*

Aplikasi *Pattern Test* ini merupakan implementasi hasil pelatihan yang dilakukan terhadap pola gambar. Maka dari itu didalam aplikasi ini, akan menampilkan informasi hasil pelatihan. Informasi yang akan ditampilkan yaitu berupa prediksi nilai skor yang terjadi selama proses pencocokan pola, hasil perhitungan nilai panjang goresan komponen pola gambar, jumlah komponen yang membentuk pola gambar, serta informasi nilai titik yang membentuk pola

gambar. Sama seperti pada aplikasi sebelumnya, dengan memanfaatkan *class* *GestureOverlayView,* tampilan ini sebagai suatu hamparan untuk melakukan proses menggambar pada tampilan antarmuka program.

```
DESCRIPTION:
1     public void onCreate(Bundle savedInstanceState)
2         {
3             super.onCreate(savedInstanceState);
4             setContentView(R.layout.main);
5             File f = new
6     File(Environment.getExternalStorageDirectory()+"/gestures"
7     );
8         patternlib = GestureLibraries.fromFile(f);
9
10        patternText =
11    (TextView)findViewById(R.id.text_pattern);
12            if (!patternlib.load())
13            {
14                Log.w("SampleTestActivity", "could not load
15    gesture library");
16
17                finish();
18                }
19        GestureOverlayView gestureview =
20    (GestureOverlayView)findViewById(R.id.overlay_view);
21        Button button =
22    (Button)findViewById(R.id.button_pattern);
23
24        gestureview.addOnGesturePerformedListener(gesturelis
25    tener);
26                button.setOnClickListener(btnClick);
27        }
28
29
30
```

Gambar 4.19 Deskripsi Menjalankan Aplikasi *Pattern Test*

Penjelasan deskripsi menjalankan aplikasi *Pattern Test* pada Gambar 4.19 yaitu:

1. Baris 1 – 17, mendefinisikan metode `onCreate` untuk menyatakan bahwa pertama kali program berjalan akan membaca file biner yang berisi informasi pola gambar yang telah tercipta dan tersimpan didalam *`ExternalStorageDirectory`*.
2. Baris 19 – 20, mendefinisikan program dengan memanfaatkan *class* `GestureOverlayView` yang telah dijelaskan pada subbab sebelumnya.
3. Baris 21 – 25, menyatakan bahwa didalam tampilan program ini ada fungsi `button` guna menampilkan informasi yang lain.

```
DESCRIPTION:
1    public void onGesturePerformed(GestureOverlayView
2    overlay, Gesture gesture)
3          {
4          result = gesture.getLength();
5          result2 = gesture.getBoundingBox() ;
6          result3 = gesture.getStrokesCount();
7          ArrayList<Prediction> predictions =
8    patternlib.recognize(gesture);
9          patternText.setText("\n\n This is the Predictions
10   \n\n");
11         if (predictions.size() > 0)
12         {
13           for(Prediction prediction: predictions){
14           patternText.append("Name: " + prediction.name +
15   ", Score: " + prediction.score + "\n");
16                         }               }
17           patternText.append("\n----Hasil Uji Coba----
18   \n");
19           patternText.append("Length :" + result);
20           patternText.append("\nRectF :" + result2);
21           patternText.append("\nHasilnya :" + result3);
22         }   };
23
24
```

Gambar 4.20 Deskripsi Menampilkan Informasi pada Aplikasi *Pattern Test*

Penjelasan deskripsi menampilkan informasi didalam aplikasi *Pattern Test* pada Gambar 4.20 yaitu:

1. Baris 1, mendefinisikan metode `onGesturePerformed` untuk menyatakan bahwa aplikasi memanfaatkan *class* `GestureOverlayView`.
2. Baris 5 – 7, mendefinisikan variabel yang akan digunakan untuk menampilkan informasi.
3. Baris 9 – 10, menyatakan bahwa data yang diambil merupakan hasil prediksi yang telah dilakukan dari hasil pengenalan pola gambar.
4. Baris 14 – 28, menampilkan informasi hasil pelatihan yang telah dilakukan.

4.3.4 Implementasi Program Lock-Screen

Program Lock-Screen merupakan implementasi sederhana identifikasi *user* berdasarkan pengenalan pola gambar objek sederhana pada perangkat berbasis Android.

Setting parameter awal agar program aplikasi dapat menjalankan sistem penguncian pada layar perangkat berbasis Android sebagai berikut :

```
DESCRIPTION:
1    //metode untuk mengunci layar, dimana fitur-fitur yang
2    dijalankan ketika layar telah terkunci
3    private void StartLock(Context context) {
4    Intent closeDialogs = new
5    Intent(Intent.ACTION_CLOSE_SYSTEM_DIALOGS);
6            context.sendBroadcast(closeDialogs);
7            Class w = LockActivity.class;
8            Intent welcome = new Intent(context, w);
9            welcome.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
10                   | Intent.FLAG_ACTIVITY_NO_USER_ACTION
11                   | Intent.FLAG_ACTIVITY_NO_ANIMATION
12                   | Intent.FLAG_ACTIVITY_NO_HISTORY);
13                   context.startActivity(welcome);
14   }
15
```

Gambar 4.21 Deskripsi Setting Parameter *Lock-Screen*

Penjelasan deskripsi setting parameter *Lock-Screen* pada Gambar 4.21 yaitu:

1. Baris 1, mendefinisikan metode `StartLock` untuk memulai menjalankan program.
2. Baris 5 - 8, menjalankan `Intent` untuk menampilkan `closeDialogs` pada tampilan layar.
3. Baris 10 , memanggil `class LockActivity`, dan menjalankannya.
4. Baris 12 – 21, menyatakan fitur-fitur yang dijalankan ketika aplikasi pengunci layar berjalan.

```
DESCRIPTION:
1   public void onCreate(Bundle savedInstanceState) {
2     super.onCreate(savedInstanceState);
3     //setting tampilan layar
4
5
6   requestWindowFeature(android.view.Window.FEATURE_NO_TITLE);
7   getWindow().
8   addFlags(WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED);
9   /*Tanda (WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED)
10  digunakan untuk mendapatkan status layar, meskipun aturan
11  tetap berlaku untuk proses LockScreen
12          */
13  updateLayout();
14
15
16  IntentFilter offfilter = new IntentFilter
17  (Intent.ACTION_SCREEN_OFF);
18      registerReceiver(screenoff, offfilter);
19      serviceHandler = new Handler();
20  }
```

Gambar 4.22 Deskripsi Setting Tampilan Layar *Lock-Screen*

```
DESCRIPTION:
1  //method yang terinisialisasi ketika aplikasi dijalankan
2  melalui fitur Menu
3      private void updateLayout()
4      {
5        LayoutInflater inflater = LayoutInflater.from(this);
6        setContentView(inflateView(inflater));
7        File f = new
8  File(Environment.getExternalStorageDirectory()+"/gestures");
9              gestureLib = GestureLibraries.fromFile(f);
10 //membaca file dari media penyimpanan.
11             if (!gestureLib.load()){
12             finish ();
13             }
14
15 //menginisialisasi overlay (hamparan canvas) bagi masukan
16 gambar pola yang di inginkan
17 GestureOverlayView gestures = (GestureOverlayView)
18 findViewById (R.id.gestures);
19 gestures.addOnGesturePerformedListener(handleGestureListener);
20       }
```

Gambar 4.23 Deskripsi Tampilan Layar Proses Unlock

Deskripsi tampilan layar proses *Unlock* pada Gambar 4.23 memanfaatkan *class* `GestureOverlayView`, sama seperti penjelasan pada implementasi aplikasi sebelumnya. Jadi didalam aplikasi ini juga menerapkan hamparan tersebut untuk menggambar sebagai media untuk memasukan kunci pola gambar.

```
DESCRIPTION:
1     // menampilkan widget menu pada layar
2           public boolean onCreateOptionsMenu(Menu menu) {
3                 menu.add(0, 1, 0, "About");
4                 menu.add(0, 2, 0, "Unlock");
5           return true;
6           }
7           public boolean onOptionsItemSelected(MenuItem
8     item) {
9                 switch (item.getItemId()) {
10                case 1:
11    // menampilkan AlertDialog ketika menu About di pilih
12                new AlertDialog.Builder(this)
13                      .setTitle("About")
14                      .setMessage("Test Purpose Only")
15                      .setNeutralButton("Close", new
16    DialogInterface.OnClickListener(){
17                            public void
18    onClick(DialogInterface dlg, int sumthin)
19          {
20            // do nothing – it will close on its own
21          }
22        })
23              .show();
24              return true;
25              case 2:
26    // kembali menjalankan proses yang ada di kelas Lock
27    Activity
28                        Intent i = new Intent(this,
29    LockActivity.class);
30                        startActivity(i);
31    return true;
32       }
33    return false;
34    }
35    }
36    }
37    }
```

Gambar 4.24 Deskripsi Tampilan Menu Program *Lock-Screen*

```
DESCRIPTION:
1    ArrayList<Prediction> predictions =
2                          gestureLib.recognize(gesture);
3          if (predictions.size()> 0)
4          {
5          Prediction prediction = predictions.get(0);
6          if (prediction.score> 7.0)
7           {
8            if (prediction.name.equals("key"))
9              {
10               Toast.makeText(LockActivity.this,
11                "Congratulation, Unlock Success" ,
12                  Toast.LENGTH_LONG).show();
13
14         /* Proses untuk kembali ke Layar Home Utama
15              ketika Unlock telah berhasil */
16         Intent home_intent = new
17   Intent(Intent.ACTION_MAIN);
18         home_intent.addCategory(Intent.CATEGORY_HOME);
19         home_intent.
20   addFlags(Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
21             startActivity(home_intent);
22               }
23               else
24         Toast.makeText(LockActivity.this,"TryAgain !!" ,
25                Toast.LENGTH_SHORT).show();
26
27
28
```

Gambar 4.25 Deskripsi Logic *Lock-Screen*

Penjelasan deskripsi logic *Lock-Screen* pada Gambar 4.25 yaitu:

1. Baris 1 - 2, menginisialisasi objek `Prediction` yang telah terdapat didalam pustaka pola gambar sebagai bagian dari `ArrayList<Prediction>` untuk dilakukan proses pengenalan dengan pola yang belum diketahui.
2. Baris 6, menyatakan nilai rata-rata skor tertinggi yang harus dicapai oleh hasil pelatihan, agar dapat diimplementasikan sebagai parameter identifikasi didalam aplikasi *Lock-Screen.* Jika nilai skor kurang dari

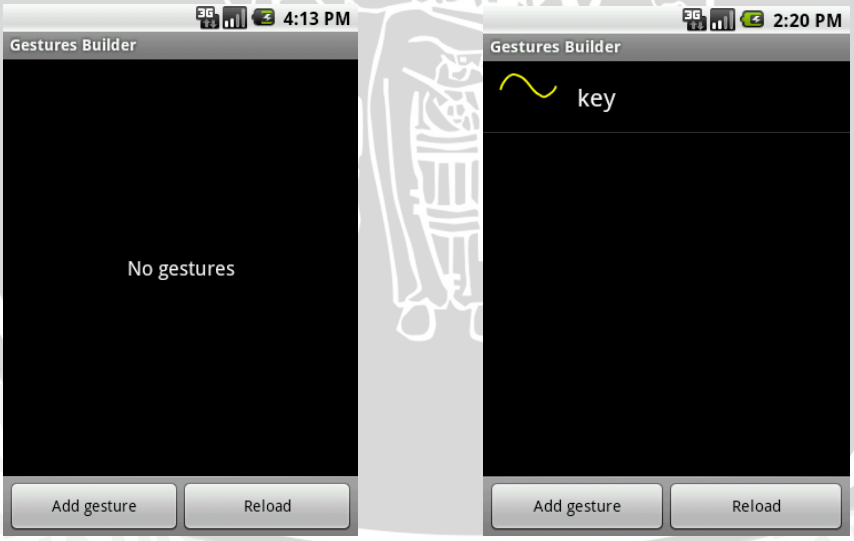
nilai yang telah ditentukan yaitu 7, maka pola gambar bukan merupakan kunci untuk membuka program aplikasi *Lock-Screen.*

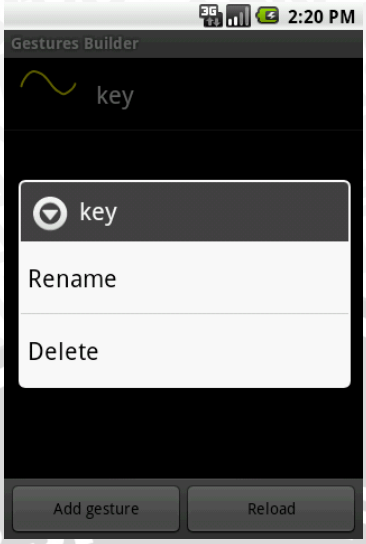
3. Baris 9, sama seperti kondisi yang telah dinyatakan pada baris 6 deskripsi *logic* aplikasi tersebut namun parameter penyaringnya adalah nama dari pola gambar yang telah dibuat. Jika pola gambar yang belum diketahui pola-nya sama dengan nama pola yang telah ditentukan yaitu bernama “key”, maka pola tersebut merupakan kunci untuk membuka sistem *Lock-Screen* .
4. Baris 17 – 24, menyatakan proses yang akan terjadi bila kondisi *logic* aplikasi telah terpenuhi semua, yaitu proses akan kembali ke tampilan *home* pada perangkat berbasis Android.

4.4 Implementasi Antarmuka Program

Pada setiap aplikasi, tampilan antarmuka dibuat dengan sesederhana mungkin agar dapat dengan mudah digunakan.

4.4.1 Implementasi Antarmuka Program Pembuat Pola Gambar



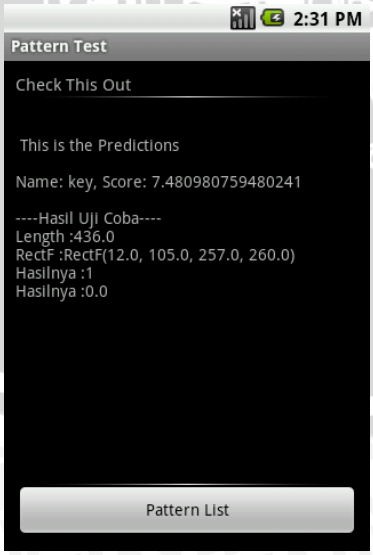


Gambar 4.26 Antarmuka program pembuat pola gambar

Antarmuka Program Pembuat Pola Gambar ini menampilkan *button* dengan fungsi masing-masing. *Add Gesture* untuk memulai membuat pola gambar baru. Sedangkan *field Name* berisikan nama *gesture* yang akan menjadi parameter pada proses berikutnya.

Terdapat tampilan *dialog*, apabila pola gambar yang telah tercipta dipilih untuk melakukan proses *delete* atau *rename* .

4.4.2 Implementasi Antarmuka Program *Pattern-Test*



Gambar 4.27 Antarmuka program *Pattern Test.*

Antarmuka program *pattern test* ini digunakan untuk menunjukkan hasil pelatihan yang telah dilakukan pada pola gambar yang belum diketahui untuk dibandingkan dengan pola gambar yang telah tercipta pada program sebelumnya. Informasi yang ditampilkan berupa prediksi nilai skor, panjang komponen gambar yang telah tercipta, serta informasi titik dari bidang yang mencakup gambar yang telah tercipta.

4.4.3 Implementasi Antarmuka Program *Lock-Screen*

Antarmuka program ini akan muncul bila program *lock-Screen* telah berjalan. Dengan berjalannya program tersebut, untuk memunculkan menu pilihan membuka kunci dengan menekan tombol menu pada perangkat. Selanjutnya akan ada tampilan berupa hamparan untuk menggambar pola masukan kunci yang telah diciptakan sebelumnya. Secara otomatis bila pola gambar yang dimasukkan sesuai dengan pola kunci yang telah dibuat, maka aplikasi akan membuka sistem pengunci layar, kemudian akan kembali ke tampilan *home* awal.

Gambar 4.28 Antarmuka program *Lock- Screen.*

BAB V
PENGUJIAN

Pada bab ini dilakukan proses pengujian dan analisis terhadap aplikasi Identifikasi User Pada Sistem Operasi Android Berdasarkan Pengenalan Pola Citra Objek Sederhana yang telah dibangun. Proses pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat sudah bekerja dengan baik dan sesuai dengan perancangan atau bahkan terjadi suatu kegagalan. Pengujian yang dilakukan pada bab ini akan dibagi menjadi beberapa hal berikut :

1. Pengujian Program Aplikasi *Pattern Test.*
2. Pengujian Program Aplikasi *Lock-Screen.*

5.1 Pengujian Program Aplikasi *Pattern Test*

Pengujian ini bertujuan untuk mengetahui prediksi nilai hasil pelatihan pola gambar bila dibandingkan dengan pola gambar yang telah terlebih dahulu tercipta dan tersimpan didalam media penyimpanan perangkat *smartphone.* Pengujian dapat dibagi menjadi beberapa skenario seperti berikut :

1. Pengujian dengan arah yang sama, pola gambar hampir sama.
2. Pengujian dengan arah yang sama, pola gambar berbeda.
3. Pengujian dengan arah yang berbeda / berkebalikan, pola yang hampir sama.
4. Pengujian dengan arah yang berbeda / berkebalikan, pola gambar berbeda.
5. Pengujian dengan pola gambar yang sama sekali berbeda.

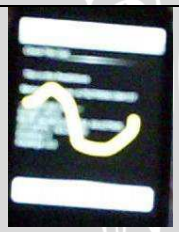
5.1.1 Pengujian Program Aplikasi *Pattern Test* Skenario 1

Pengujian aplikasi skenario 1 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan menggambarkan pola gambar yang hampir sama dengan pola gambar yang terdapat pada pustaka file biner, dengan menyamakan arah pergerakan pembuatan pola. Jadi misalnya pola gambar yang terdapat pada file biner dibangun dengan cara pola digambar dari arah kiri menuju ke kanan, maka pada program pengujian skenario 1 ini dilakukan dengan menyamakan arah, sama seperti dengan pola yang terdapat pada pustaka file biner. Selain itu panjang

komponen gambar yang membentuk akan dibuat hampir sama dengan pola gambar yang terdapat pada file biner.

Pengujian dilakukan dengan cara membuat pola gambar dalam 10 kali pelatihan didalam program aplikasi ini. Hasil pengujian dapat ditunjukkan didalam tabel berikut :

No	Pola Gambar	Hasil Pelatihan	
		Skor	Panjang
1		4,49	292
2		6,43	301
3		5,32	256
4		11,85	247
5		6,81	264

6		3,16	338
7		6,11	282
8		9,50	287
9		4,75	291
10		3,96	293
rata – rata		6.491111	285.1

Tabel 5.1 Pengujian Program Aplikasi *Pattern Test* skenario 1

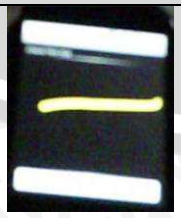
Dari hasil pengujian pada Tabel 5.1, didapatkan kesimpulan bahwa ketika melakukan pelatihan terhadap pola gambar yang dibuat dengan arah pergerakan yang sama serta pola gambar yang hampir sama pula dengan gambar yang terdapat pada pustaka file biner, maka prediksi nilai skor yang bisa didapatkan dari 10 kali percobaaan didapatkan nilai 6,49.

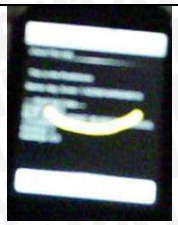
Hasil pengujian yang seperti ini-lah yang diharapkan untuk dapat melakukan proses identifikasi. Semakin tinggi nilai prediksi yang bisa diraih maka pola gambar tersebut merupakan pola gambar yang hampir mendekati pola gambar yang telah tercipta pada proses implementasi pembuatan pola gambar sebelumnya (*template*). Dengan demikian hasil pengujian ini bisa digunakan sebagai parameter utama didalam melaksanakan proses identifikasi pada program aplikasi yang membutuhkan proses identifikasi melalui persamaan pola gambar yang ada.

5.1.2 Pengujian Program Aplikasi *Pattern Test* Skenario 2

Pengujian aplikasi skenario 2 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan menyamakan arah pergerakan pembuatan pola, serta proses menggambarkan pola gambar yang sedikit berbeda dengan pola gambar yang terdapat pada pustaka file biner. Jadi misalnya pola gambar yang terdapat pada file biner dibangun dengan cara pola digambar dari arah kiri menuju ke kanan, maka pada program pengujian skenario 2 ini dilakukan dengan men-samakan arah, sama seperti dengan pola yang terdapat pada pustaka file biner. Namun yang menjadi pembeda adalah panjang komponen gambar yang membentuk. Panjang komponen gambar tersebut bisa dibuat dengan sedikit lebih pendek atau sedikit lebih panjang dari pola gambar yang telah tercipta pada file biner.

Pengujian dilakukan dengan cara membuat pola gambar dalam 10 kali pelatihan didalam program aplikasi ini. Hasil pengujian dapat ditunjukkan didalam tabel berikut :

No	Pola Gambar	Hasil Pelatihan	
		Skor	Panjang
1		1,92	196

2		1,28	169
3		2,04	184
4		1,49	634
5		2,96	459
6		2,37	447
7		1,76	371
8		1,31	243

9		1,04	894
10		1,30	562
Rata- rata		1.8675	415.9

Tabel 5.2 Pengujian Program Aplikasi *Pattern Test* skenario 2

Dari hasil pengujian pada Tabel 5.2, didapatkan kesimpulan bahwa ketika melakukan pelatihan terhadap pola gambar yang dibuat dengan arah pergerakan yang sama namun pola gambar yang dibuat panjang komponennya sedikit lebih panjang atau sedikit lebih pendek dari pola gambar yang terdapat pada pustaka file biner, maka prediksi nilai skor yang didapatkan dari 10 kali percobaaan didapatkan nilai 1,86.

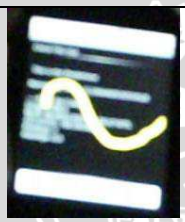
Hasil ini lebih kecil dari hasil yang didapatkan ada proses sebelumnya, hal ini menandakan untuk melakukan proses pencocokan pola, maka setiap komponen yang digunakan sebagai parameternya pun juga perlu diperhitungkan.

5.1.3 Pengujian Program Aplikasi *Pattern Test* Skenario 3

Pengujian aplikasi skenario 3 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan menggambarkan pola gambar yang hampir sama dengan pola gambar yang terdapat pada pustaka file biner, dengan membalikkan arah pergerakan pembuatan pola. Jadi misalnya pola gambar yang terdapat pada file biner dibangun dengan cara pola digambar dari arah kiri menuju ke kanan, maka pada program pengujian skenario 3 ini dilakukan dengan membalikkan arah (dari kanan kearah kiri). Selain itu panjang komponen gambar yang membentuk akan dibuat hampir sama dengan pola gambar yang terdapat pada pustaka file biner.

Pengujian dilakukan dengan cara membuat pola gambar dalam 10 kali pelatihan didalam program aplikasi ini. Hasil pengujian dapat ditunjukkan didalam tabel berikut :

No	Pola Gambar	Hasil Pelatihan	
		Skor	Panjang
1		0,69	302
2		0,32	325
3		0,33	271
4		0,36	297
5		0,34	288
6		0,32	331

7		0,33	359
8		0,33	349
9		0,33	288
10		0,33	295
Rata-rata		0,368	310,5

Tabel 5.3 Pengujian Program Aplikasi *Pattern Test* skenario 3

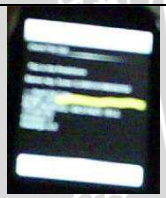
Dari hasil pengujian pada Tabel 5.3, didapatkan kesimpulan bahwa ketika melakukan pelatihan terhadap pola gambar yang dibuat dengan arah pergerakan yang berkebalikan serta pola gambar yang dibuat panjang komponennya sama dengan pola gambar yang terdapat pada pustaka file biner, maka prediksi nilai skor yang didapatkan dari 10 kali percobaaan didapatkan nilai 0,368.

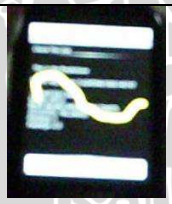
Hasil ini jauh lebih kecil dari hasil yang didapatkan ada proses sebelumnya, hal tersebut menandakan dengan digunakannya metode *gesture recognition* dengan menerapkan arah gerakan sebagai salah satu faktor utama dalam penyaringan gerakan, maka arah pergerakan pola yang dibuat pun berpengaruh terhadap hasil pencocokan pola gambar.

5.1.4 Pengujian Program Aplikasi *Pattern Test* Skenario 4

Pengujian aplikasi skenario 4 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan menggambarkan pola gambar yang berbeda dengan pola gambar yang terdapat pada pustaka file biner, dengan membalikkan arah pergerakan pembuatan pola. Jadi misalnya pola gambar yang terdapat pada file biner dibangun dengan cara pola digambar dari arah kiri menuju ke kanan, maka pada program pengujian skenario 4 ini dilakukan dengan membalikkan arah (dari kanan kearah kiri). Selain itu panjang komponen gambar tersebut bisa dibuat dengan sedikit lebih pendek atau sedikit lebih panjang dari pola gambar yang terdapat pada pustaka file biner.

Pengujian dilakukan dengan cara membuat pola gambar dalam 10 kali pelatihan didalam program aplikasi ini. Hasil pengujian dapat ditunjukkan didalam tabel berikut:

No	Pola Gambar	Hasil Pelatihan	
		Skor	Panjang
1		0,39	201
2		0,33	168
3		0,43	246
4		0,37	219

5		0,52	560
6		0,45	680
7		0,39	473
8		0,42	807
9		0,34	308
10		0,41	579
Rata – rata		0.405	424.1

Tabel 5.4 Pengujian Program Aplikasi *Pattern Test* skenario 4

Dari hasil pengujian pada Tabel 5.4, didapatkan kesimpulan bahwa ketika melakukan pelatihan terhadap pola gambar yang dibuat dengan arah pergerakan yang berkebalikan serta pola gambar yang dibuat panjang komponennya tidak sama dengan pola gambar yang terdapat pada pustaka file biner, maka prediksi nilai skor yang didapatkan dari 10 kali percobaaan didapatkan nilai 0,405.

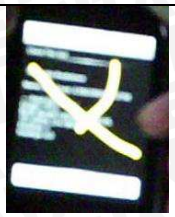
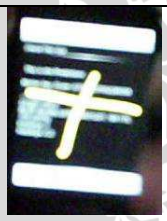
Hasil ini hampir sama dengan yang diperoleh pada pengujian skenario 3, hal tersebut menandakan dengan digunakannya metode *gesture recognition* dengan menerapkan arah gerakan serta panjang komponen yang membentuk pola sebagai faktor utama dalam penentuan hasil akhir pencocokan.

5.1.5 Pengujian Program Aplikasi *Pattern Test* Skenario 5

Pengujian aplikasi skenario 5 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan menggambarkan pola gambar yang berbeda dengan pola gambar yang terdapat pada pustaka file biner. Berbeda baik dari pola maupun arah pergerakannya serta jumlah komponen gambar yang membentuk dibuat berbeda dengan yang ada pada file biner.

Pengujian dilakukan dengan cara membuat pola gambar dalam 10 kali pelatihan didalam program aplikasi ini. Hasil pengujian dapat ditunjukkan didalam tabel berikut:

No	Pola Gambar	Hasil Pelatihan	
		Skor	Panjang
1		0,57	595
2		--	--
3		0,98	370
4		0,55	777

5		--	--
6		--	--
7		0,50	629
8		--	--
9		--	--
10		0,79	635

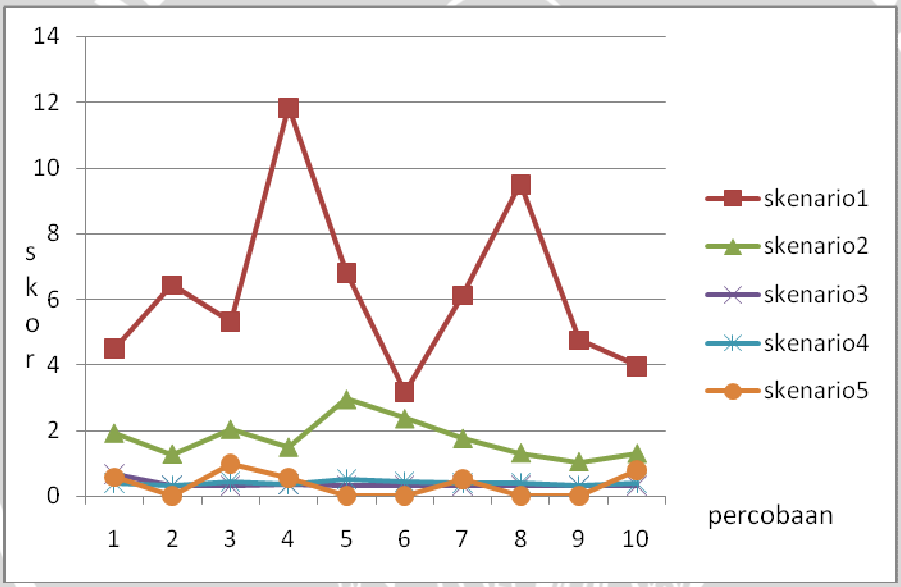
Tabel 5.5 Pengujian Program Aplikasi *Pattern Test* skenario 5

Dari hasil pengujian pada Tabel 5.5, didapatkan kesimpulan bahwa ketika melakukan pelatihan terhadap pola gambar yang dibuat dengan pola yang sama sekali berbeda dengan pola gambar yang terdapat pada pustaka file biner, maka akan ada suatu keadaan dimana tidak akan terdapat prediksi nilai hasil skor . Hal

ini dikarenakan pola gambar tersebut pola tidak dapat dikenali oleh sistem pengenalan pola yang terdapat pada program aplikasi.

Dari hasil pengujian pada Tabel 5.5, pola gambar yang tidak bisa dikenali tersebut merupakan pola gambar yang komponen penyusun gerakannya (*stroke*) melebihi nilai SEQUENCE_SENSITIVE yang telah ditentukan. Artinya apabila pola gambar komponennya melebihi komponen pola yang telah tercipta pada pustaka file biner. Didalam kasus ini komponen pola gambar adalah 2, padahal komponen yang terdapat pada pustaka file biner adalah 1.

Secara keseluruhan dari hasil pengujian yang dilakukan pada program aplikasi *Pattern Test* ini, didapatkan karakteristik prediksi nilai skor yang didapatkan dalam melatih pola gambar berdasarkan skenario-skenario yang telah ditentukan yaitu terdapat pada gambar berikut:



Gambar 5.1 Karakteristik Pengujian Terhadap Pola Gambar

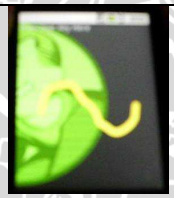
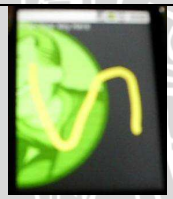
Dengan mememberikan masukan pola gambar yang hampir sama dengan pola gambar yang terdapat didalam pustaka file biner, baik itu dari arah pergerakan maupun panjang komponennya, maka akan didapatkan prediksi nilai skor yang tinggi. Prediksi nilai skor yang tinggi tersebut dapat dijadikan parameter untuk melakukan identifikasi terhadap *user* didalam pembuatan program-program yang membutuhkan tingkat sekuritas yang tinggi.

5.2 Pengujian Program Aplikasi *Lock - Screen*

Pengujian ini bertujuan untuk membuka sistem pengunci layar pada aplikasi *Lock-Screen* menggunakan kunci pembuka berdasarkan masukan pola gambar yang telah dibuat pada proses pembuatan pola pada aplikasi Modul Pembuatan Pola Gambar.

Dengan menetapkan rata-rata nilai skor tertinggi yaitu ‘7’ didalam implementasi *logic* pemrograman, serta dengan parameter nama pola gambar bernama ‘ key ‘, maka bila pola gambar kunci yang dimasukkan tidak memenuhi kriteria tersebut, aplikasi *Lock-Screen* tidak akan bisa terbuka.

Hasil pengujian dapat ditunjukkan didalam tabel berikut :

No	Pola Gambar	Hasil Pelatihan
1		salah
2		benar
3		salah
4		benar

Tabel 5.6 Pengujian Program Aplikasi *Lock-Screen*

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Aplikasi Modul Pembuatan Pola Gambar merupakan aplikasi yang digunakan untuk menciptakan, menambah, menghapus serta mengenali pola gambar yang akan tersimpan didalam pustaka file biner.
2. Aplikasi *Pattern Test* digunakan untuk melakukan proses pelatihan pola gambar, membandingkannya dengan pola gambar yang telah terdapat didalam pustaka file biner, serta mendapatkan prediksi nilai skor tertinggi yang dapat dicapai dari hasil pelatihan pola gambar baru.
3. Salah satu penerapan metode otentikasi dalam proses identifikasi *user* pada sistem operasi Android adalah program aplikasi *Lock-Screen* yang melakukan autentifikasi berdasarkan pola gambar objek sederhana yang dibuat oleh *user*.
4. Prediksi nilai skor yang dihasilkan nilainya dibawah 1 mengindikasikan bahwa hasil pencocokan pola bersifat jelek atau pola sama sekali tidak cocok dengan pola yang telah dibuat sebelumnya.
5. Semakin tinggi prediksi nilai skor yang dijadikan parameter mekanisme otentikasi, maka pola gambar baru yang dibuat sebagai kunci masukan dalam aplikasi *Lock screen* harus memiliki kesamaan pola yang sesuai dengan gambar *template.*

6.2 Saran

Saran yang dapat diberikan untuk pengembangan Aplikasi Identifikasi User ini antara lain:

1. Untuk pengembangan lebih lanjut pola gambar yang dapat dikenali bisa lebih dari 1 komponen yang membentuk pola gambar tersebut.
2. Aplikasi ini dapat dikembangkan kedalam program-program aplikasi yang lain, tidak hanya untuk membuka kunci pada layar saja, namun tergantung

inovasi dan kebutuhan para pengguna perangkat berdasarkan sekuritas data yang ingin dilindungi.

DAFTAR PUSTAKA

[001] Arief, M. Rudyanto. 2006. *Otentikasi Multi Faktor Untuk Meningkatkan Keamanan Komputer*. Yogyakarta.

[002] Sitorus, Syahriol dkk. 2006. *Pengolahan Citra Digital*. Medan. USU Press

[003] Bow, Sing – Tze. 2002. *Pattern Recognition and Image Processing.* NewYork : Marcel Pekker, Inc.

[004] Efford, Nick. 2000. *Digital Image Processing a Practical Introduction to Java*. NewYork : Addison Wesley.

[005] Suksmono. 2005. *Pendahuluan Pengenalan Pola*. Diktat Kuliah Pengenalan Pola, ITB

[006] Li, Yang. 2010. *Protactor : A Fast and Accurate Gesture Recognizer.* Atlanta, GA, USA

[007] Huang, Ying. 2009. *Begin Android Journey in Hours.*

[008] Xuguang, Heung. 2009. *An Introduction to Android.* Database Lab. Inha University.

[009] Hashimi, Sayed dkk. 2010. *Pro Android 2.* NewYork : Springer.

[010] Thomas, S. Huang dkk. 1997. *IEEE Transaction On Pattern Analysis.*

[011] Najib, Marsum. 2005. *Diktat Matakuliah Kalkulus 'vektor'*. UGM. Yogyakarta

[012] Android Source Code.

http://www.java2s.com/Open-Source/Android/android-core/platform-frameworks-base/android.gesture.htm. (diakses tanggal 9 Mei 2011)

[013] Khrisnaraj Varma.*Gesture Sample* http://code.google.com/p/krvarma-android-samples/ (diakses tanggal 4 mei 2011)

[014] Anonymous. 2011. *Handling Screen Lock and Unlock in Android* http://chandan-tech.blogspot.com/2010/10/handling-screen-lock-unlock-in-android.html. (diakses tanggal 6 Juli 2011)

[015] Anonymous. 2011. *Enabling Lock Screen Android* http://smartandroidians.blogspot.com/2010/03/enabling-and-disabling-lock-screen-in.html (diakses tanggal 10 Juli 2011)