

**IMPLEMENTASI PENGENALAN WAJAH BERBASIS LIBRARY SVM
SCIKIT-LEARN MENGGUNAKAN PYTHON**

SKRIPSI

diajukan untuk menempuh ujian sarjana
pada Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

FAZLUR RAHMAN
140810150057



UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI S-1 TEKNIK INFORMATIKA
SUMEDANG

2019

ABSTRAK

Perkembangan teknologi informasi komunikasi sangat cepat. Banyak aktivitas sehari-hari yang dapat diotomasi untuk mempercepat suatu proses. Salah satu proses yang dapat diotomasi adalah otentifikasi kehadiran.

Sebuah sistem pengenalan wajah telah dibangun dengan menggunakan *library* SVM Scikit-learn dengan bahasa pemrograman Python. Sistem ini terdiri dari dua modul yaitu modul pendaftaran wajah dan modul verifikasi wajah.

Aplikasi pengenalan wajah ini diuji terhadap 10 mahasiswa Teknik Informatika FMIPA UNPAD. Pembangunan aplikasi ini berhasil membaca wajah dan menuliskannya pada suatu *file* csv yang berisi nama, tanggal, dan jam masuk. Dari aplikasi yang dibuat, tingkat akurasi pengenalan wajah adalah 98.94%. Kendala yang dihadapi dalam pembangunan perangkat lunak adalah masih lambatnya proses prediksi wajah dengan rata-rata waktu 0.784746 detik per *frame*.

Kata kunci: pengenalan wajah, *Support Vector Machine*, Python, sistem, *library*

ABSTRACT

The development of information communication technology is rapid. Many types of daily activities can be automated to speed up the process. One process that can be automated is attendance authentication.

A face recognition system has been built using the Scikit-learn SVM library using the Python programming language. This system consists of two modules which are face registration module and face verification module.

This facial recognition application was tested on 10 Informatics Engineering FMIPA UNPAD students. Development of this application successfully reads faces and write them in a CSV file that contains the name, date, and time of entry. The accuracy of facial recognition is 98.94%. The obstacle faced in this application is the slow processing time of face prediction with an average time of 0.784746 seconds per frame.

Keywords: face recognition, Support Vector Machine, Python, system, library

KATA PENGANTAR

Bismillahirrahmanirrahim, Dengan menyebut nama Allah yang Maha Pengasih lagi Maha Penyayang.

Puji dan syukur atas kehadiran Allah SWT atas segala berkat rahmat serta kasih sayang-Nya, penulis dapat menyelesaikan penyusunan tugas akhir yang berjudul “Implementasi Pengenalan Wajah Berbasis *Library* SVM Scikit-Learn Menggunakan Python”.

Tugas akhir ini disusun untuk memenuhi salah satu syarat dalam menempuh sarjana pada Program Studi S-1 Teknik Informatika Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Pada kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Dr. Setiawan Hadi, M.Sc. CS., selaku pembimbing utama dan Bapak Drs. R. Sudrajat, M.Kom., selaku pembimbing pendamping yang telah memberikan arahan dan bimbingan sehingga penulis dapat menyelesaikan penyusunan skripsi ini.

Tak lupa penulis ucapkan terima kasih kepada:

1. Prof. Dr. H. Sudradjat, MS., selaku Dekan FMIPA Universitas Padjadjaran.
2. Dr. Setiawan Hadi, M.Sc. CS., selaku Kepala Departemen Ilmu Komputer FMIPA Universitas Padjadjaran.
3. Dr. Juli Rejito, M.Kom, selaku Ketua Program Studi S-1 Teknik Informatika FMIPA Universitas Padjadjaran.

4. Kedua orang tua penulis, DR. Irfan Fachruddin, SH.,CN. dan Fatmiati Syam,SH.,Sp.N atas seluruh dukungan, doa, motivasi, yang menjadi sumber semangat kepada penulis.
5. Kedua saudara penulis, Egalita Irfan, S.Sos, MA. dan Karina Adla,S.A.B. yang telah memberikan dorongan dan dukungan kepada penulis.
6. Teman-teman penulis, Gilang Akbar, Bayu Roma, Yuda Aprimulyana, Umar Fadhlullah, Ragil, Adi Purnama, Fathur Ghazzani yang telah membantu dan memberi motivasi pada penulis dalam menyelesaikan skripsi ini.
7. Teman-teman seperjuangan, mahasiswa S-1 Teknik Informatika FMIPA Unpad angkatan 2015, yang telah membuat kehidupan perkuliahan menjadi lebih berwarna.
8. Seluruh pihak yang tidak dapat penulis sebutkan satu-persatu, yang telah banyak memberikan motivasi dan bantuan kepada penulis.

Akhir kata, penulis berharap semoga tugas akhir ini bermanfaat bagi pengembangan ilmu pengetahuan khususnya ilmu komputer di Indonesia .

Sumedang, Agustus 2019

Penulis

DAFTAR ISI

ABSTRAK	i
ABSTRACT	ii
KATA PENGANTAR	iii
DAFTAR ISI	v
DAFTAR GAMBAR	viii
DAFTAR TABEL	x
BAB I	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	4
BAB II	6
2.1 Citra Digital dan Pengolahan Citra	6
2.2 <i>Grayscale</i>	7
2.3 <i>Histogram of Oriented Gradient (HOG)</i>	8
2.4 <i>Face Landmark</i> pada OpenFace	9
2.5 <i>Support Vector Machines</i>	10
2.6 <i>Unified Modeling Language</i>	12
2.6.1 <i>Use Case Diagram</i>	12
2.6.2 <i>Activity Diagram</i>	13

2.6.3	<i>Class Diagram</i>	15
2.6.4	<i>Sequence Diagram</i>	15
2.6.5	<i>Component Diagram</i>	16
2.6.6	<i>Collaboration Diagram</i>	17
2.6.7	<i>State Diagram</i>	18
2.6.8	<i>Deployment Diagram</i>	19
2.7	<i>K-Fold Cross Validation</i>	20
2.8	Pembangunan aplikasi	21
2.8.1	Python	21
2.8.2	Flask	22
2.8.3	OpenCV	22
2.8.4	Scikit-learn	23
2.8.5	Dlib	23
2.8.6	HTML (Hypertext Markup Language)	24
2.9	Penelitian Sebelumnya	24
BAB III	METODOLOGI PENELITIAN	26
3.1	Tahapan Perancangan	26
3.1.1	Studi Literatur	26
3.1.2	Pengumpulan Data	27
3.1.3	Perancangan dan Pembuatan Perangkat Lunak	28
3.1.4	Pengujian Perangkat Lunak	31
3.2	Rancangan Sistem	32
3.2.1	<i>Use Case Diagram</i>	32
3.2.2	<i>Activity Diagram</i>	34
3.2.3	Rancangan Desain Antarmuka	38

3.3	Alat dan Bahan Pembangunan Perangkat Lunak	43
BAB IV	HASIL DAN PEMBAHASAN	44
4.1	Implementasi Antarmuka	44
4.1.1	Tampilan Halaman Utama	44
4.1.2	Tampilan Laporan	45
4.1.3	Tampilan Tambah Data Wajah	46
4.1.4	Tampilan <i>Train</i>	50
4.2	Pengujian Data	51
4.2.1	Pengujian Data Pelatihan	51
4.2.2	Pengujian Data dengan Video	52
4.2.3	Pengujian Data Secara <i>Real-time</i>	54
BAB V	SIMPULAN DAN SARAN	56
5.1	Simpulan	56
5.2	Saran	57
DAFTAR PUSTAKA	58
LAMPIRAN	60
RIWAYAT HIDUP	77

DAFTAR GAMBAR

Gambar 2.1 Representasi piksel.....	6
Gambar 2.2 Citra digital.....	7
Gambar 2.3 Grayscale.....	8
Gambar 2.4 Proses HOG.....	8
Gambar 2.5 68 titik acuan yang akan mengenali wajah (Boyko et al., 2018)	10
Gambar 2.6 Hyperplane SVM (Nugroho et al., 2003).....	11
Gambar 2.7 UML class	15
Gambar 2.8 Sequence diagram	16
Gambar 2.9 Component diagram	17
Gambar 2.10 Collaboration diagram.....	18
Gambar 2.11 State Diagram.....	19
Gambar 2.12 Deployment diagram	20
Gambar 2.13 Skema Fold dengan $k=10$	21
Gambar 2.14 Program sederhana pada Flask (Ronacher, 2010).....	22
Gambar 3.1 Tahapan Perancangan.....	26
Gambar 3.2 Proses Pengolahan Wajah	29
Gambar 3.3 pemotongan dengan HOG.....	29
Gambar 3.4 Ekstraksi Fitur dengan OpenFace	30
Gambar 3.5 Use Case Diagram pada modul registrasi	32
Gambar 3.6 Use case diagram pada modul otentikasi	33
Gambar 3. 7 Activity diagram upload gambar.....	35
Gambar 3.8 <i>Activity diagram</i> proses pembelajaran	36
<i>Gambar 3.9 Activity diagram</i> pada melihat video	37
<i>Gambar 3.10 Activity diagram</i> laporan.....	38
Gambar 3.11 Halaman Utama.....	39
Gambar 3.12 Halaman laporan	40
Gambar 3.13 Tampilan pilihan tambah data wajah	41
Gambar 3.14 Tampilan upload data wajah	41
Gambar 3.15 Tampilan tambah data wajah dengan kamera	42
Gambar 3.16 Halaman <i>Train Data</i>	43

Gambar 4.1 Halaman utama ketika wajah belum ditulis pada laporan.....	45
Gambar 4.2 Halaman utama ketika sudah ditulis pada laporan	45
Gambar 4.3 Tampilan laporan	46
Gambar 4.4 Halaman Opsi Tambah Data Wajah.....	47
Gambar 4.5 Page upload gambar wajah.....	47
Gambar 4.6 Page upload dengan kamera	48
Gambar 4.7 Page saat melakukan pemotretan wajah.....	48
Gambar 4.8 Tampilan berhasil tambah data	49
Gambar 4.9 Tampilan gagal tambah data	49
Gambar 4.10 Tampilan Train.....	50
Gambar 4.11 Tampilan berhasil Train	51
Gambar 4.12 Wajah yang tidak terdeteksi	53
Gambar 4.13 Pengujian real-time	55
Gambar 4.14 Tampilan pengujian real-time	55

DAFTAR TABEL

Tabel 2. 1 Notasi <i>Use Case</i>	13
Tabel 2.2 Notasi Activity Diagram	14
Tabel 3. 1 Data wajah yang digunakan	27
Tabel 4. 1 Hasil pengujian akurasi dengan K-Fold Cross Validation.....	51
Tabel 4. 2 Hasil pengujian video pada resolusi 1920x1080.....	52
Tabel 4. 3 Hasil pengujian video pada resolusi 960x540.....	53
Tabel 4. 4 Tabel Waktu Eksekusi	54

BAB I

PENDAHULUAN

Pada bab ini diuraikan latar belakang, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan

1.1 Latar Belakang

Perkembangan informasi komunikasi dan teknologi sangat cepat. Dengan perkembangan teknologi, banyak aktivitas sehari-hari yang dapat diautomasi dengan media elektronik untuk mempercepat proses. Hal ini menimbulkan kebutuhan yang tinggi dalam pemrosesan informasi secara cepat dan akurat. Salah satu proses yang dapat diautomasi adalah autentikasi kehadiran.

Proses perekaman kehadiran dan pelaporan masih banyak yang menggunakan pensil dan kertas dan memakan waktu yang cukup lama. Dengan pesatnya perkembangan teknologi, manusia dapat terbantu dengan kemudahan yang ditawarkan oleh teknologi. Untuk mempermudah pekerjaan manusia, sistem perekaman dan pelaporan dapat dilakukan secara otomatis dengan keunikan biometrik.

Biometrik merupakan karakteristik unik pada manusia yang dapat digunakan untuk proses autentikasi. Biometrik yang sering digunakan dalam autentikasi adalah sidik jari, tangan, atau wajah. Pengenalan wajah merupakan salah satu teknologi untuk autentikasi berbasis biometrik yang dapat dilakukan tanpa interaksi fisik. Perangkat keras yang diperlukan untuk pengenalan wajah

tidak sesulit pengenalan biometrik lainnya. Pengenalan wajah mengambil fitur-fitur unik dari muka, sehingga saat proses autentikasi akan sulit untuk dipalsukan.

Berdasarkan uraian di atas, penulis ingin menerapkan pengenalan wajah pada proses autentikasi kehadiran. Aplikasi yang dirancang bertujuan untuk mempercepat proses autentikasi dan pelaporan secara otomatis. Dalam perancangannya, akan digunakan *library Support Vector Machine* untuk proses klasifikasi fitur dari setiap wajah. Seluruh *library* hanya diaplikasikan dalam perancangan sistem dan tidak menyentuh pembahasan metode *library*.

1.2 Identifikasi Masalah

Dari latar belakang yang telah diuraikan, maka dalam penelitian ini akan dikembangkan sebuah sistem yang memiliki rumusan masalah:

1. Bagaimana wajah dapat digunakan sebagai elemen verifikasi dalam sistem pengenalan wajah.
2. Komponen apa saja yang diperlukan untuk merancang sebuah sistem pengenalan wajah.
3. Bagaimana cara menemukan kinerja dari sistem yang telah dibuat.

1.3 Batasan Masalah

Dari permasalahan yang telah disampaikan di atas, penulis menentukan beberapa batasan masalah yaitu:

1. Aplikasi yang dibangun menggunakan bahasa Python dan menggunakan *library* Flask dan Scikit-Learn.

2. Algoritma yang digunakan untuk klasifikasi yaitu *Support Vector Machine*.
3. Data wajah yang digunakan adalah wajah mahasiswa S1 Teknik Informatika FMIPA UNPAD.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai oleh penulis dari penelitian ini yaitu:

1. Merancang sistem pengenalan wajah sebagai sistem otentikasi.
2. Mengimplementasikan pengenalan wajah dengan menggunakan *library*.
3. Menganalisis kinerja pengenalan wajah yang dibuat.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini yaitu:

1. Dapat mengetahui kegunaan pengenalan wajah sebagai pendukung proses otentikasi otomatis.
2. Sistem pengenalan wajah dapat digunakan untuk pencatatan laporan.
3. Dapat menjadi referensi untuk pengembangan penelitian lanjutan tentang pengenalan wajah.

1.6 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah:

1. Studi literatur

Pada tahapan ini dilakukan pengumpulan bahan referensi mengenai penelitian *face recognition*, terutama pada fitur ekstraksi dan klasifikasi *Support Vector Machine*.

2. Perancangan sistem.

Dilakukan analisa kebutuhan, pengumpulan data, perancangan arsitektur, dan perancangan antar muka.

3. Pembuatan program

Dilakukan pengkodean untuk membangun aplikasi sesuai dengan perancangan yang telah dilakukan.

4. Pengujian Program

Dilakukan pengujian untuk memastikan aplikasi berjalan sesuai harapan.

5. Penyusunan Laporan

Pada tahap ini dilakukan pembuatan laporan dari penelitian yang telah dilakukan dan mengambil kesimpulan.

1.7 Sistematika Penulisan

Sistematika dari penulisan skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan secara lengkap mengenai latar belakang masalah, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian serta sistematika penelitian

BAB II TINJAUAN PUSTAKA

Bab ini akan menjelaskan tentang konsep serta teori mengenai pemrosesan citra dalam pengenalan wajah yang diklasifikasi dengan *Support Vector Machine* serta dasar dalam pembangunan aplikasi web.

BAB III METODE PENELITIAN

Bab ini menguraikan secara rinci mengenai proses penelitian yang dilakukan meliputi desain sistematika penelitian yang dilakukan serta alat dan bahan yang digunakan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai hasil dari implementasi metode yang digunakan dan membahas pengujian perangkat lunak yang dibuat

BAB V SIMPULAN DAN SARAN

Bab ini merupakan penutup dari skripsi yang berisi kesimpulan dan saran terhadap hal hal yang perlu dikembangkan pada penelitian ini.

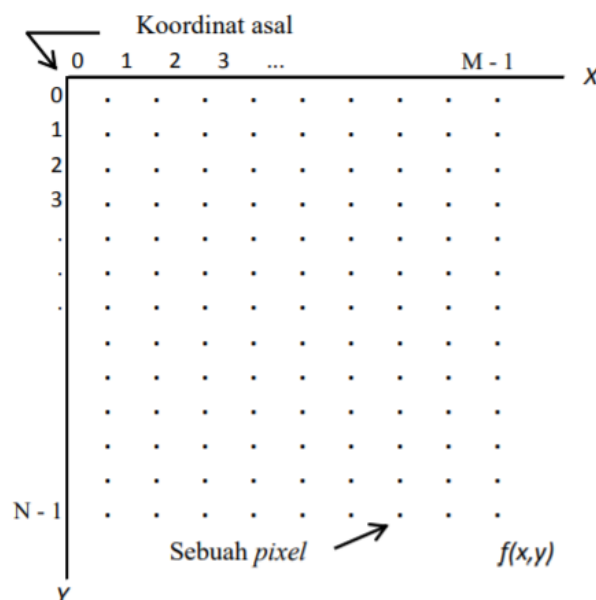
BAB II

TINJAUAN PUSTAKA

Pada bab ini akan dibahas tentang teori dasar yang berkaitan dengan pengolahan citra, pengenalan wajah dan pembangunan aplikasi web.

2.1 Citra Digital dan Pengolahan Citra

Citra digital merupakan kumpulan titik yang memiliki nilai nilai diskrit. Pada umumnya citra memiliki 2 dimensi baris dan kolom. Suatu citra dapat didefinisikan dengan fungsi $f(x,y)$ dengan ukuran N baris dan M kolom, dengan x dan y merupakan koordinat, dan f pada titik koordinat (x,y) merupakan intensitas dari citra. Titik $f(x,y)$ dapat disebut sebagai sebuah piksel. Setiap piksel dalam citra berwarna memiliki tiga elemen yang diistilahkan RGB yaitu *Red* (merah), *Green* (hijau), dan *Blue* (biru) (Putra, 2010).



Gambar 2.1 Representasi piksel

Citra digital dapat ditulis dalam bentuk sebagai berikut :

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

Gambar 2.2 Citra digital

Istilah lain dari citra digital adalah bagian dari komponen multimedia yang memiliki peran penting dalam menghasilkan informasi visual. Gambar 2.1 menunjukkan representasi piksel.

Pengolahan Citra merupakan disiplin ilmu yang membahas bagaimana teknik-teknik mengolah citra. Hal ini berkaitan dengan perbaikan kualitas gambar (peningkatan kecerahan, kontras, restorasi citra) dan transformasi gambar (translasi, rotasi, transformasi geometri). Dalam definisi lain menyebutkan pengolahan citra adalah proses dengan masukan citra dan menghasilkan keluaran berupa citra.

2.2 *Grayscale*

Sebuah piksel memiliki tiga kombinasi warna yaitu *Red*, *Green*, *Blue* (*RGB*). Warna *RGB* direpresentasikan pada tiga dimensi XYZ. Kualitas dari warna gambar tergantung pada jumlah bit.

Grayscale merupakan teknik citra digital untuk mengubah citra *RGB* menjadi citra *gray* (abu-abu) sehingga citra *gray* dapat dianggap sebagai citra dengan satu *channel* warna dengan nilai minimal 0 (hitam) sampai 255 (putih).

Grayscale menunjukkan tingkat keabuan dari piksel. *Grayscale* mengubah seluruh nilai pada RGB menjadi sama. Untuk mendapatkan nilai citra *gray*, maka dilakukan perhitungan nilai rata rata dari nilai 3 matriks *red*, *green*, *blue* dengan persamaan 2.1. (Santi, 2011).

$$X = 0.299 R + 0.587 G + 0.114B \quad (2.1)$$

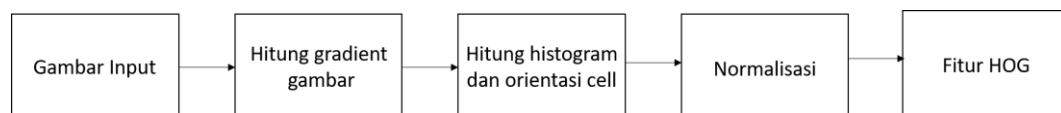
Hasil dari penerapan *grayscale* dapat dilihat pada Gambar 2.3



Gambar 2.3 *Grayscale*

2.3 *Histogram of Oriented Gradient (HOG)*

Histogram of Oriented Gradient (HOG) merupakan salah satu metode untuk mengekstraksi ciri dari suatu gambar agar dikenali sebagai suatu objek. HOG biasanya digunakan untuk membuat pengenalan objek. Gambar 2.4 menggambarkan langkah langkah dalam proses HOG.



Gambar 2.4 Proses HOG

HOG pada dasarnya mencari nilai distribusi gradien pada suatu gambar untuk memperoleh karakteristik khusus dari gambar tersebut. Karakteristik ini diperoleh

dengan cara membagi citra menjadi beberapa bagian/daerah kecil yang berisi beberapa piksel yang biasa disebut *cell*. Didalam *cell*, setiap piksel akan dihitung gradien yang didasari dari tetangga yang berdekatan. Pada akhirnya setiap piksel dalam *cell* akan berkontribusi pada saat melakukan *voting* bobot (Pranoto & Ramadhani, 2017).

2.4 *Face Landmark* pada OpenFace

Face landmark merupakan teknik yang digunakan untuk menentukan titik-titik penting yang merepresentasikan suatu wajah. *Face landmark* biasanya digunakan untuk menyelaraskan atau membuat gambar muka menjadi di tengah. Hal ini dilakukan karena pada kenyataannya muka tidak selalu menghadap ke depan, tetapi akan sedikit berputar ke kiri ataupun kanan (Boyko, Basystiuk, & Shakhovska, 2018).

Pada *library* OpenFace, titik-titik penting (*landmark*) suatu wajah akan direpresentasikan dengan 68 *landmark*. Titik-titik tersebut meliputi bagian atas dagu, garis tepi alis, tepi luar mata, bagian bawah hidung, dan bagian atas dan bawah dari bibir. Setelah mendapatkan seluruh titiknya, akan dilakukan pemposisian *landmark* tersebut dengan manipulasi standar sampai mendapatkan posisi wajah tepat di tengah. Pada Gambar 2.5 ditunjukkan 68 titik yang akan mengenali suatu wajah.



Gambar 2.5 68 titik acuan yang akan mengenali wajah

(Boyko et al., 2018)

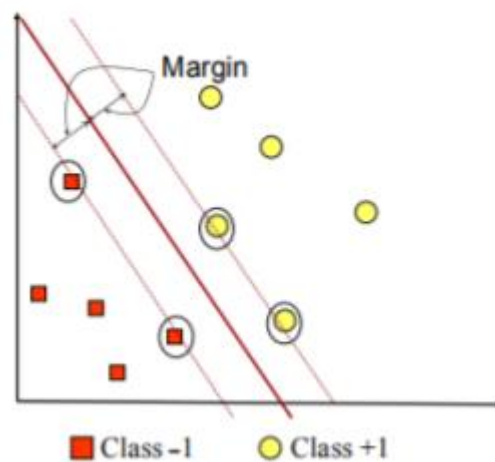
- Bagian mulut dapat diakses pada titik [48,67]
- Alis kanan pada titik [17,21]
- Alis kiri pada titik [22,26]
- Mata kanan pada titik [36,41]
- Mata kiri pada titik [42,47]
- Hidung pada titik [27,34]
- Rahang pada [0,16]

2.5 *Support Vector Machines*

Support Vector Machine (SVM) adalah teknik untuk melakukan prediksi baik dalam klasifikasi ataupun regresi. Teknik tersebut membandingkan parameter standar sekumpulan nilai diskrit yang disebut set kandidat dan mengambil salah satu yang memiliki akurasi tertinggi. Metode klasifikasi ini menerapkan *structural*

risk minimization (SRM) dengan tujuan menemukan *hyperplane* terbaik yang dapat memisahkan dua buah *class* pada *input* (Nugroho, Witarto, & Handoko, 2003).

Gambar ini memperlihatkan beberapa data yang berisikan dua macam kelas yaitu kelas : +1 dan -1. Tujuan dari klasifikasi adalah menemukan garis (*hyperplane*) yang memisahkan antara dua kelas tersebut.



Gambar 2.6 *Hyperplane SVM* (Nugroho et al., 2003)

Hyperplane pemisah antara kedua kelas dapat ditemukan dengan cara mengukur *margin hyperplane* dan mencari titik maksimal. *Margin* merupakan jarak antara *hyperplane* dengan pola terdekat dari masing masing kelas. *Support vector* merupakan sebutan untuk pola yang paling dekat dengan *hyperplane*. Proses pencarian *hyperplane* merupakan inti dari pembelajaran SVM.

SVM dasarnya merupakan *linear classifier*, dan telah dilakukan penelitian untuk dapat bekerja pada kasus *non-linear* dengan menggunakan konsep kernel pada ruang berdimensi tinggi. Berikut adalah kernel pada SVM:

1. Kernel *Linear*: $x^t x$
2. Kernel *Polynomial*: $x^t x_i + 1$

3. *Kernel Radial Basis Function*: $\exp(-\frac{1}{2\sigma^2} ||x - x_i||^2)$
4. *Kernel Tangent Hyperbolic*: $\tanh(\beta x^t x_i + \beta_i)$ dimana $\beta_1, \beta_2 \in r$; $r =$ bilangan real.

2.6 *Unified Modeling Language*


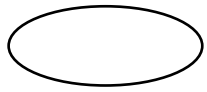

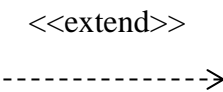
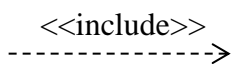
Unified Modeling Language merupakan bahasa permodelan standar dalam pengembangan suatu perangkat lunak. Untuk mendisain suatu sistem, dibutuhkan sebuah bahasa yang mendeskripsikan model. Dengan adanya *Unified Modeling Language* (UML), model yang dirancang pada sistem tidak akan salah diinterpretasi.

UML mempermudah permodelan karena notasi yang dipakai sangat jelas. Saat diperlukan, UML dapat menangani pemodelan proyek yang besar. Bahasa pemodelan ini dikontrol oleh *standards group* dengan kontribusi aktif dari grup akademik dan *vendor* dari seluruh dunia (Hamilton & Miles, 2006).

2.6.1 *Use Case Diagram*

Use case merupakan suatu penggambaran situasi dimana sistem digunakan untuk memenuhi satu atau lebih dari kebutuhan pengguna. Kebutuhan pengguna dapat divisualisasikan secara lebih mudah dengan *use case diagram*. *Use Case* hanya menggambarkan apa yang sistem seharusnya lakukan, dan apa yang sistem tidak boleh lakukan (Hamilton & Miles, 2006).

Tabel 2. 1 Notasi *Use Case*

Simbol	Nama	Keterangan
	Aktor	Pengguna yang berinteraksi dengan sistem
	<i>Use Cases</i>	Interaksi yang dapat dilakukan pada sistem
	<i>Association</i>	Hubungan aktor dengan sistem
	<i>Extend</i>	Sebuah hubungan <i>use case</i> yang mempunyai sifat dari use case lainnya (<i>base</i>)
	<i>Include</i>	Hubungan <i>use case</i> yang dapat mempengaruhi <i>use case</i>





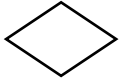
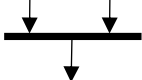
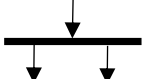
Pada Tabel 2.1 ditunjukkan notasi yang digunakan pada *use case diagram*. Simbol symbol yang sering digunakan pada proyek skala kecil adalah *actor*, *use cases*, dan *association*.

2.6.2 Activity Diagram

Activity diagram merupakan penggambaran aliran kerja atau aktivitas dari sebuah sistem. Aktivitas diagram ini lebih fokus kepada apa saja yang dapat

dilakukan oleh sistem, bukan terfokus pada apa yang dilakukan aktor. Diagram ini digunakan untuk menggambarkan alur pekerjaan sistem secara bertahap (Hamilton & Miles, 2006).

Tabel 2.2 Notasi *Activity Diagram*

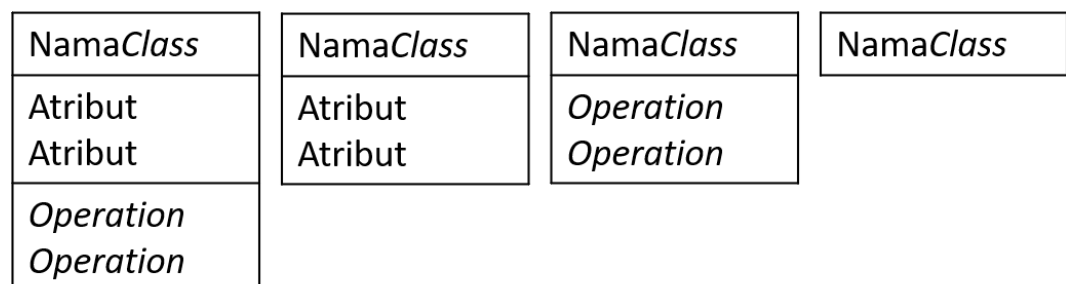
Notasi	Nama	Keterangan
	Aktivitas	Sebuah aktifitas merupakan sebuah pekerjaan yang harus diselesaikan
	Transmission (flow)	Setelah sebuah aktifitas selesai, flow akan langsung mengarahkan pada aktivitas selanjutnya
	<i>Starting Nodes</i>	Merupakan sumber dari aliran yang biasa disebut <i>Initial node</i>
	<i>Ending Node</i>	Tujuan akhir dari aliran yang biasa disebut <i>final node</i> .
	<i>Decision</i>	Sebuah pilihan yang menghasilkan yes/no
	<i>Join Node</i>	Sebuah notasi untuk mengakhiri sebuah aktifitas paralel
	<i>Fork Node</i>	Membagi aktivitas menjadi paralel yang dapat berjalan

Pada Tabel 2.2 ditunjukkan notasi-notasi yang digunakan dalam *activity diagram*. Alur pekerjaan dapat mempermudah visualisasi sistem dengan notasi notasi tersebut.

2.6.3 Class Diagram

Class merupakan inti dari pemrograman berbasis objek. Sebuah sistem terbentuk dari banyak satuan yang biasa disebut objek. *Class* menjelaskan perbedaan objek yang dimiliki oleh sistem. (Hamilton & Miles, 2006)

Berbeda dengan *Uses case diagram*, *Class* diagram menjelaskan objek apa saja yang dibutuhkan pada sebuah sistem. Sebuah *class* pada UML digambarkan dengan persegi panjang yang terbagi menjadi tiga bagian. Bagian atas merupakan nama dari *class*, bagian tengah adalah atribut atau informasi yang dimiliki oleh *class*, dan bagian terendah merupakan operasi yang merepresentasikan tingkah laku *class*. Atribut dan *operation* merupakan opsional. Gambar 2.7 menunjukkan UML *class*.

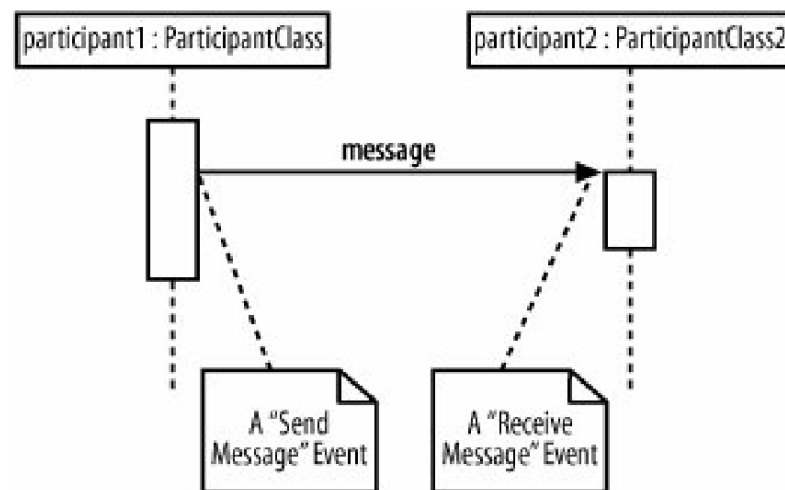


Gambar 2.7 UML *class*

2.6.4 Sequence Diagram

Sequence diagram memodelkan interaksi penting diantara bagian bagian yang ada pada sistem. *Sequence diagram* bertujuan untuk menggambarkan urutan interaksi pada sistem. Diagram ini juga menunjukkan informasi lain tentang interaksi. (Hamilton & Miles, 2006)

Sequence diagram terdiri dari kumpulan bagian sistem yang berinteraksi satu sama lain. Pada setiap diagram, partisipan selalu disusun secara horizontal dengan tidak adanya dua partisipan yang bertabrakan satu sama lain. Setiap partisipan memiliki dua tipe garis yaitu horizontal dan vertikal. Garis horizontal menunjukkan objek mana yang berinteraksi dan garis vertikal menunjukkan waktu. Bila partisipan berupa *actor*, penggambaran partisipan dapat Digambar dengan *stick figure*. Gambar 2.8 merupakan penggambaran dari *sequence diagram*.

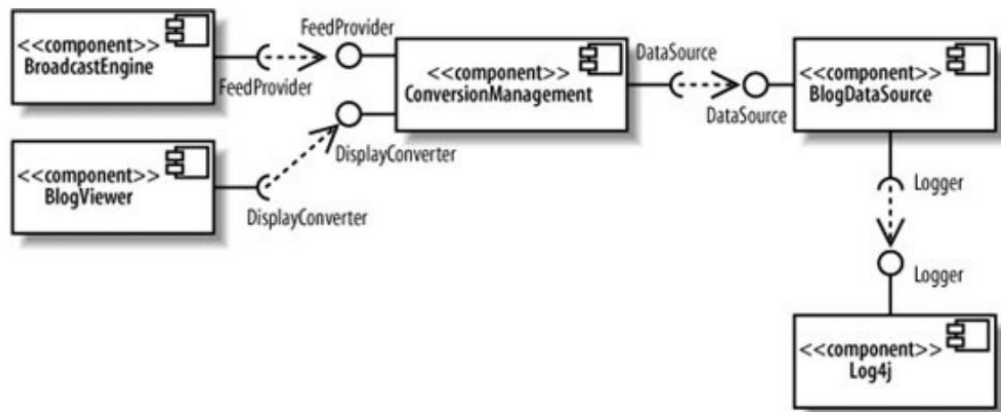


Gambar 2.8 *Sequence diagram*

2.6.5 Component Diagram

Component diagram menjelaskan tentang hubungan dari komponen fisik kode pada sistem. Diagram komponen biasanya dibuat untuk memudahkan implementasi model dan memastikan setiap aspek dalam sistem telah direncanakan. Komponen diagram memiliki beberapa notasi yaitu komponen, *interface*, *dependencies*, *port* (Hamilton & Miles, 2006).

Tujuan dari diagram komponen adalah untuk menunjukkan hubungan struktural antar komponen dari sebuah sistem. Gambar 2.9 menunjukkan salah satu contoh dari diagram komponen.

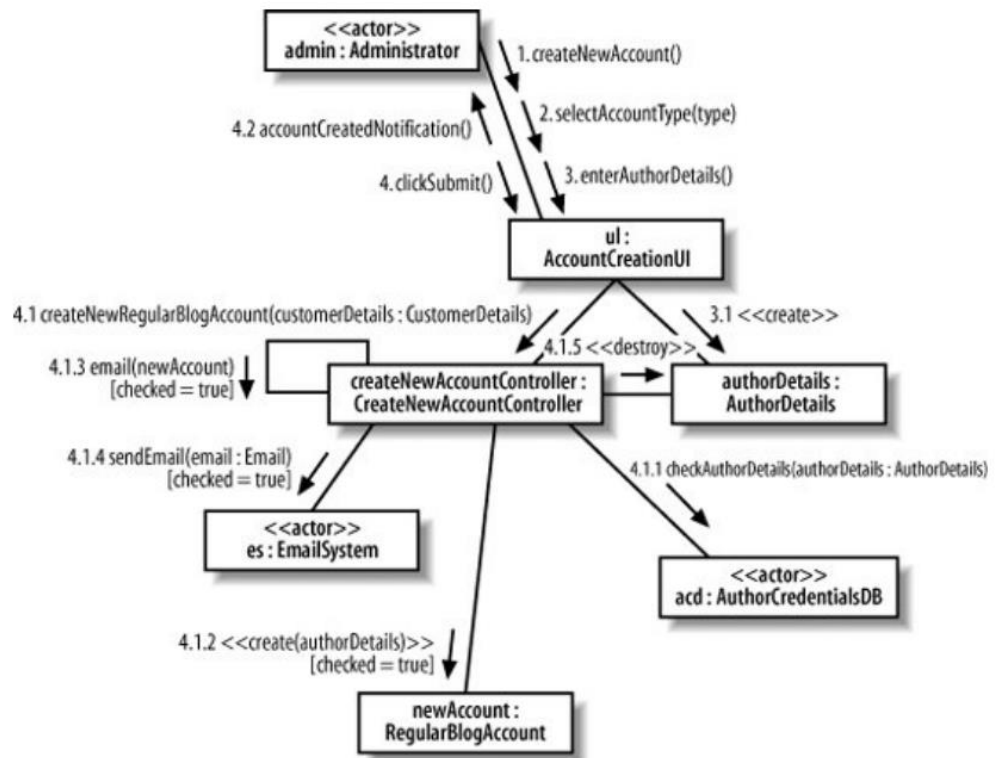


Gambar 2.9 Component diagram

2.6.6 Collaboration Diagram

Collabroration diagram digunakan untuk menunjukkan bagaimana interaksi objek untuk melakukan *use case* tertentu. *Collaboration diagram* digunakan untuk mendefinisikan peran dari suatu objek yang melakukan *event* tertentu pada *use case*. (Hamilton & Miles, 2006)

Notasi yang digunakan dalam *collaboration diagram* adalah *object*, *actors*, *link*, *message*. Pada *collaboration diagram*, *message* harus memiliki urutan yang berawal dari 1 sampai akhir. Gambar 2.10 menunjukkan contoh *collaboration diagram*.

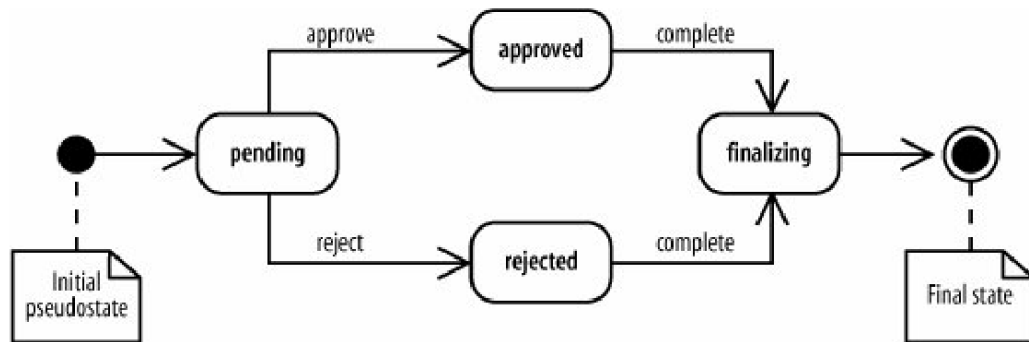


Gambar 2.10 Collaboration diagram

2.6.7 State Diagram

State diagram digunakan untuk melengkapi *activity diagram*. Terkadang *state* dari sebuah objek merupakan faktor yang penting. *State diagram* terdiri dari *states* yang digambarkan sebagai sebuah *rounded rectangle*. *Transition* merepresentasikan perubahan *state*, atau bisa dikatakan bagaimana cara berpindah dari satu *state* ke *state* lainnya (Hamilton & Miles, 2006).

State diagram biasanya memiliki *initial pseudostate* dan *final state*, yang menandakan titik awal dan titik akhir dari sebuah *state machine*. *Initial pseudostate* digambarkan dengan lingkaran padat, dan *final state* digambarkan dengan dua buah lingkaran yang didalamnya memiliki lingkaran padat seperti yang ditunjukkan pada gambar 2.11.

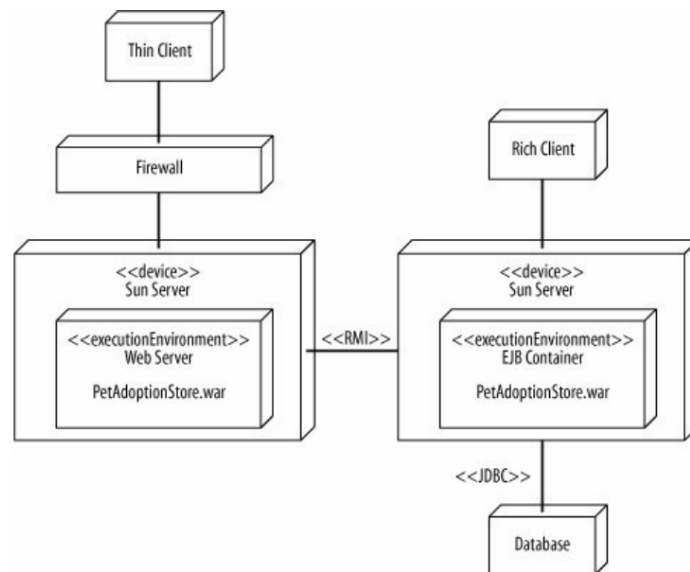


Gambar 2.11 State Diagram

2.6.8 Deployment Diagram

Deployment diagram menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak dari suatu sistem, menunjukkan hubungan komputer dengan perangkat satu sama lain. Tujuan dari diagram tersebut adalah untuk menunjukan struktur dari *run-time system* dan menunjukkan perangkat keras apa saja yang akan digunakan untuk mengimplemntasikan sistem (Hamilton & Miles, 2006).

Deployment diagram menunjukan *physical view* dari sebuah sistem. UML tersebut menunjukkan bagaimana *software* dimasukkan pada *hardware* dan bagaimana cara mereka berkomunikasi. Gambar 2.12 menunjukkan contoh dari *deployment diagram*.



Gambar 2.12 *Deployment diagram*

2.7 *K-Fold Cross Validation*

Cross validation adalah teknik model validasi untuk menentukan seberapa akurat *predictive model*. Tujuan dari *cross validation* adalah untuk mendefinisikan dataset untuk menguji model dalam *training set*. Pemilihan *cross validation* dapat didasarkan pada ukuran *dataset* (Rodríguez, Pérez, & Lozano, 2010).

Proses yang terjadi adalah mengacak *dataset* secara acak, lalu membagi dataset menjadi k grup. Pada setiap grup akan diambil test data set dan sisanya akan dijadikan *training dataset*. Dilakukan *fitting model* dan dilakukan evaluasi model. Selanjutnya akan dirata ratakan evaluasi model untuk akurasi. Pembagian data dalam *K-Fold Cross Validation* k=10 digambarkan dalam Gambar 2.13.

komputasi program. Fitur-fitur yang disediakan menjadi pemilihan yang baik untuk ilmunan dan *engineer* untuk menulis aplikasi saintifik.

2.8.2 Flask

Flask merupakan *web framework* yang menyediakan kemudahan dan fleksibilitas dengan mengimplementasikan *web server* dan *micro-framework*. Flask didesain untuk membuat untuk mempermudah mengembangkan web dari mudah sampai kompleks. Flask bergantung pada Jinja *template engine* dan Werkzeug WSGI *toolkit*. Pada perancangannya, Flask membutuhkan HTML sebagai tampilannya (Ronacher, 2010).

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Gambar 2.14 Program sederhana pada Flask (Ronacher, 2010)

Kemudahan Flask menunjukkan web sederhana yang akan menampilkan “Hello World” pada *web-service* hanya dengan menggunakan 5 baris kode dalam Python. Pada Gambar 2.14 diperlihatkan kode sederhana untuk menampilkan “Hello World” pada web. *File* tersebut harus disimpan menggunakan format “.py”.

2.8.3 OpenCV

OpenCV merupakan *library* komputer visi *open source*. *Library* tersebut ditulis menggunakan bahasa C dan C++ dan dapat berjalan di Linux, Windows, dan

Mac OS. Pada tahun 1999 Gary Bradski yang bekerja di Intel meluncurkan OpenCV dengan harapan dapat mempercepat dan mengembangkan komputer visi dan kecerdasan buatan. (Bradski & Kaehler, 2008).

Library ini didesain untuk efisiensi komputasi dan berfokus pada *real-time application*. OpenCV bertujuan untuk memudahkan komputer visi sehingga dapat membantu untuk pengembangan aplikasi secara cepat.

2.8.4 Scikit-learn

Scikit-learn memanfaatkan lingkungan Python yang dinamis untuk menyediakan banyak algoritma pembelajaran komputer sambil menjaga kemudahan *interface* dan terintegrasi dengan bahasa Python. Scikit-learn diciptakan karena tingginya kebutuhan akan analisis data statistika (Abraham, Pedregosa, Eickenberg, & Gervais, 2014).

Library ini memiliki tujuan *general* untuk *machine learning*. Pada Scikit-learn, semua objek dan algoritma menerima *input* data dengan bentuk 2 dimensi dengan besar jumlah_sampel x jumlah_fitur. Scikit-learn memiliki satu bentuk set metode yang memiliki tugas masing masing: *estimator* dapat mencocokkan model dari data, *predictor* dapat membuat prediksi pada data baru, dan *transformer* dapat mengkonversi data dari satu representasi ke representasi lainnya.

2.8.5 Dlib

Dlib merupakan *cross platform open source* yang ditulis dalam bahasa C++. *Library* ini dipengaruhi dengan kontrak dan pemrograman *component-based*.

Library ini bertujuan untuk berguna dalam dunia penelitian dan penggunaan komersial, dan telah didesain secara detil agar dapat berintegrasi dengan C++ (Boyko et al., 2018).

Dlib telah berkembang dan digunakan sebagai banyak alat. Dlib dapat menangani *graphical user interface, images, thread, XML*, dan banyak hal lainnya.

2.8.6 HTML (Hypertext Markup Language)

HTML merupakan sebuah protokol yang digunakan untuk membuat sebuah web yang dapat dibaca oleh berbagai macam *browser* (Nugraha, Tullah, & Dzulhaq, 2017). Fungsi HTML adalah untuk membangun sebuah halaman *web*.

Masih banyak orang yang menyebut HTML sebagai bahasa pemrograman, tetapi sebenarnya HTML bukan bahasa pemrograman, melainkan suatu bahasa *markup*. *Markup* (penandaan) tersebut akan dilakukan dalam sebuah dokumen teks. Tanda tersebut digunakan untuk menentukan bentuk atau format dari teks yang diberi tanda.

2.9 Penelitian Sebelumnya

Penelitian sebelumnya digunakan untuk memperkaya bahasan penelitian, analisis, dan membandingkan dengan penelitian sedang dilakukan. Berikut adalah penelitian yang berhubungan dengan konsep *face recognition*:

- 1) Jurnal dengan judul *Face Recognition – A Tool for Automated Attendance Sistem* yang diteliti oleh Naeema Mohamed Kutty dan Shelmy Mathai pada tahun 2019. Penelitian ini memaparkan tentang pengenalan wajah untuk sistem

pencatatan kehadiran menggunakan *face detection* dengan algoritma Viola Jones dan algoritma Eigenface. Penelitian ini dilatarbelakangi oleh kesulitan institusi dalam merekam dan menjaga kedatangan dari setiap siswanya. Dengan adanya implementasi pengenalan wajah, masalah ini dapat teratasi (Kutty & Mathai, 2017).

- 2) Jurnal dengan judul *Implementation of Classroom Attendance System Based on Face Recognition in Class* yang diteliti oleh Ajinkya Patil dan Mrudang Shukla pada tahun 2014. Penelitian ini memaparkan penggunaan Raspberry Pi dalam pengambilan data dan melakukan pengenalan wajah dengan menggunakan algoritma hibrida *Principal Component Analysis* dan *Linear Discriminant Analysis*. Banyak metode biometrik, namun biometrik pengenalan wajah ini dapat mempercepat proses dan lebih efisien (Patil & Shukla, 2014).

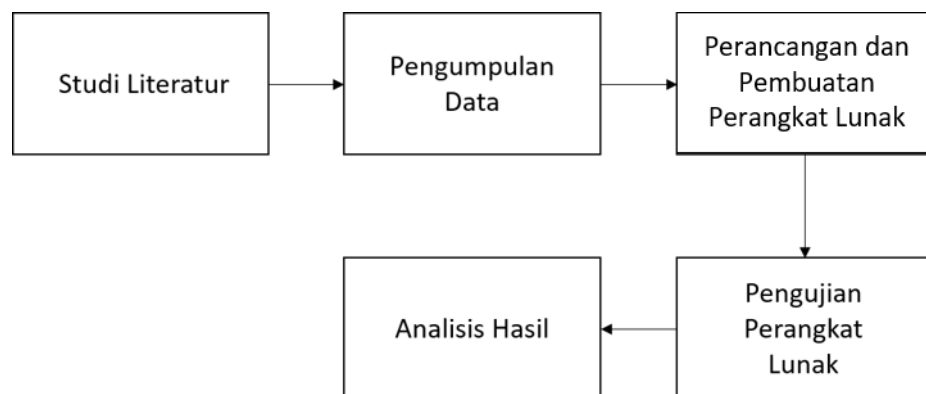
BAB III

METODOLOGI PENELITIAN

Bab ini akan menguraikan secara rinci mengenai metode, desain, serta alat dan bahan yang digunakan dalam penelitian

3.1 Tahapan Perancangan

Dalam melakukan penelitian ini dibutuhkan alur dasar pelaksanaan agar proses dan hasil penelitian dapat berjalan dengan baik. Oleh sebab itu dibuatlah suatu tahapan perancangan yang mendeskripsikan tahapan dari kegiatan penelitian yang digambarkan pada Gambar 3.1



Gambar 3.1 Tahapan Perancangan

Tahapan penelitian yang akan ditempuh dalam penelitian ini adalah: studi literatur, pengumpulan data, perancangan perangkat lunak, pengujian perangkat lunak, dan analisis hasil

3.1.1 Studi Literatur

Pada tahap ini penulis mengumpulkan referensi tentang pengenalan wajah dan pembangunan aplikasi. Referensi didapatkan dari jurnal, buku, paper, dan

sumber lainnya yang terkait dengan penelitian ini. Referensi tersebut akan digunakan untuk mendukung penelitian ini.



Referensi yang dikaji adalah teori dasar dari proses pengenalan wajah yang digunakan dalam penelitian dan teori dari *library* yang digunakan. *Library* yang dikaji adalah Dlib, OpenCV, OpenFace, dan Scikit-learn. Dilakukan juga studi tentang pendukung pengembangan perangkat lunak yaitu Python, Flask, HTML, *use case diagram*, dan *activity diagram*. Hasil dari tahapan ini diuraikan secara tertulis dan dijelaskan secara deskriptif pada BAB II.







3.1.2 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data berupa foto-foto wajah mahasiswa S1-Teknik Informatika FMIPA UNPAD dan pengambilan video wajah. Setelah itu, hasil video akan dilakukan proses pemotongan untuk membuang bagian yang tidak diperlukan untuk penelitian.

Pengambilan foto wajah akan dipisahkan berdasarkan nama orang. Setiap orang akan memiliki 1 *folder* tersendiri sehingga mempermudah proses pelabelan pada program. Data wajah yang diambil diuraikan dalam Tabel 3.1.

Tabel 3.1 Data wajah yang digunakan

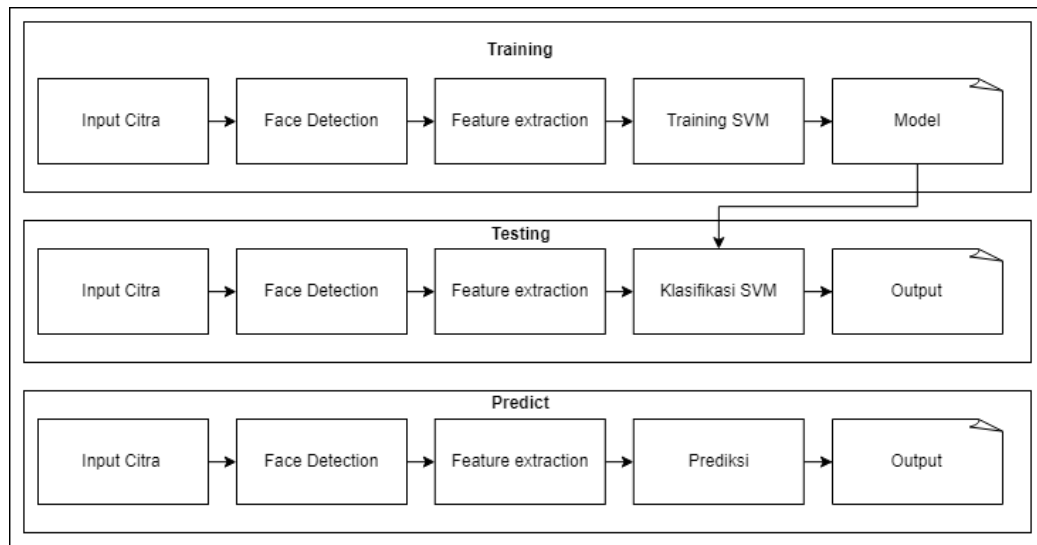
No	Foto	Nama	Jumlah Foto
1.		Bayu	10
2.		Dicky	10

3.		Fathur	10
4.		Fazlur	10
5.		Gilang	10
6.		Hilmi	10
7.		Pandji	10
8.		Qisman	10
9.		Umar	10
10.		Yudha	10

3.1.3 Perancangan dan Pembuatan Perangkat Lunak

Pada tahap ini akan dipaparkan mengenai informasi perancangan perangkat lunak. Informasi tersebut mencakup tahapan pemrosesan citra pengenalan wajah, rancangan sistem dan rancangan desain antarmuka dari aplikasi yang akan dikembangkan,

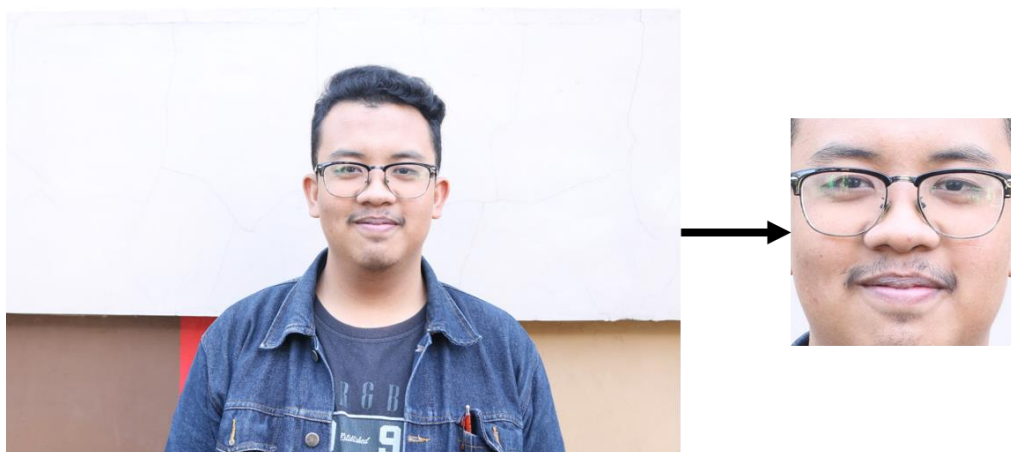
Beberapa algoritma akan diimplementasikan untuk mendeteksi wajah dalam sebuah citra. Proses ini dibagi menjadi 3 bagian yaitu *training*, *testing*, dan prediksi.



Gambar 3.2 Proses Pengolahan Wajah

A. *Face Detection*

Pada tahap ini dilakukan *face detection* dengan menggunakan *Histogram of Oriented Gradient* (HOG). Model yang digunakan adalah *object detection* berdasarkan HOG yang telah di *train* oleh *library* Dlib. Model HOG tersebut mendeteksi wajah dari depan atau biasa disebut *frontal face*. Gambar 3.3 menunjukkan hasil dari HOG *frontal face*.

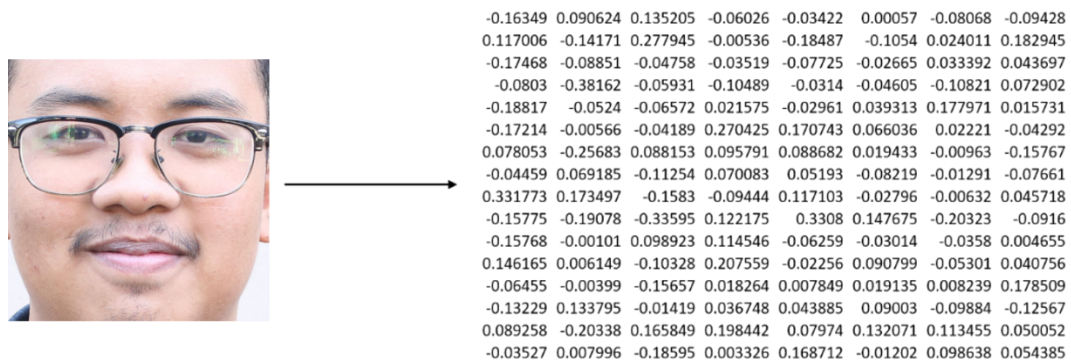


Gambar 3.3 pemotongan dengan HOG.

Pada tahapan ini, akan dilakukan pemposisian wajah sehingga wajah akan selalu ditengah dengan menggunakan *face landmark*. *Face landmark* yang digunakan adalah 68 *landmarks* dari *library* OpenFace. Setelah ditemukan *face landmark*, akan dilakukan operasi standar untuk pemposisian wajah.

B. Feature Extraction

Pada tahap ini dilakukan *feature extraction* menggunakan *library pretrained deep learning* OpenFace. Setelah dilakukan pemposisian standar, maka OpenFace akan mengekstraksi wajah menjadi 128 fitur untuk setiap wajah.



Gambar 3.4 Ekstraksi Fitur dengan OpenFace

Gambar 3.4 menunjukkan hasil dari 128 fitur yang diekstraksi dari setiap wajah. Setiap fitur mewakili karakteristik wajah yang digenerasi dengan *pretrained model*.

C. Pelatihan Data

Pelatihan data dilakukan untuk memprediksi nama wajah. Metode pelatihan data yang digunakan adalah *Support Vector Machine* (SVM). Pada proses SVM,

`data_x` merupakan 128 fitur muka yang telah diekstraksi dengan OpenFace dan `data_y` adalah label atau nama masing masing orang.

Metode SVM yang digunakan adalah *One-versus-rest* sehingga pengklasifikasian data tidak hanya 2 label, tetapi banyak label. Hasil dari pelatihan data tersebut merupakan sebuah model yang akan disimpan dengan tipe *file* `.joblib`.

D. Pengujian Data

Model hasil pelatihan akan diuji dengan menggunakan *K-fold cross validation*. Model akan menggunakan 3 kernel yaitu *radial*, *linear*, dan *sigmoid*. Hasil dari pengujian data tersebut adalah akurasi dari model setiap kernel yang sudah dilakukan pelatihan.

E. Prediksi Data

Pada tahap ini, akan diprediksi nama wajah dari sebuah gambar. Sebelum dilakukan prediksi, akan dilakukan *face detection*, *feature extraction* pada citra terlebih dahulu. Model akan diambil dari *file* `.joblib` yang sudah dilakukan *train* pada tahap pelatihan data.

3.1.4 Pengujian Perangkat Lunak

Pada tahap ini akan diuji model dari pelatihan dengan menggunakan *K-fold cross validation*. Hal ini bertujuan untuk mencari akurasi terbaik dari kernel yang terdapat pada SVM.

Pada tahap ini juga dilakukan pengujian terhadap video yang telah diambil dan secara *real-time* dengan menggunakan *webcam*. Pada proses pengujian, akan dicatat nama orang yang melewati kamera tersebut beserta tanggal dan jam terlihat

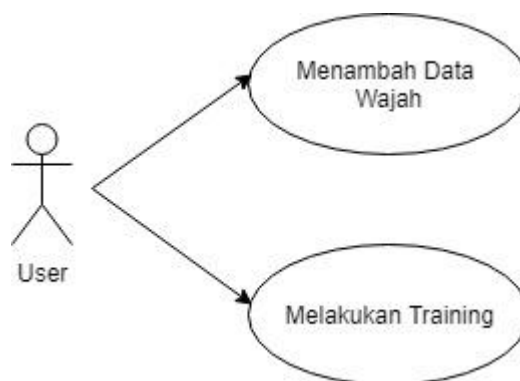
pada video. Pada tahap pengujian akan dibandingkan informasi data yang masuk pada proses prediksi dengan *groundtruth*.

3.2 Rancangan Sistem

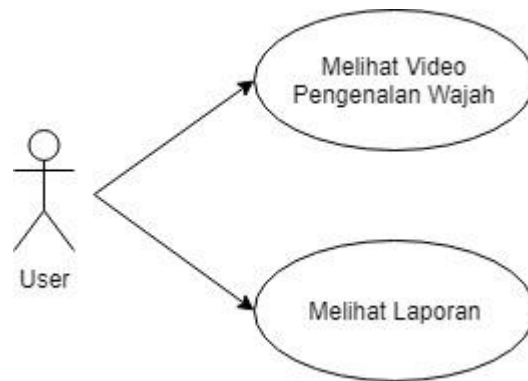
Pada tahap ini akan dijelaskan perencanaan dari pembangunan aplikasi perangkat lunak. Rancangan sistem akan memberikan penjelasan kepada pengguna mengenai cara kerja aplikasi perangkat lunak. Tahap ini memberikan penjelasan detik mengenai alur perangkat lunak.

3.2.1 Use Case Diagram

Use case diagram akan menunjukkan apa saja yang *user* dapat lakukan pada perangkat lunak. *Use case* diagram akan dibagi menjadi dua modul yaitu registrasi dan otentikasi. Pada modul registrasi, pengguna dapat mengakses fitur menambah data wajah dan melakukan *training*. Gambar 3.5 merupakan *use case* pada modul registrasi. Pada modul otentikasi, pengguna dapat mengakses fitur melihat video pengenalan wajah dan melihat laporan. Gambar 3.6 merupakan *use case* pada modul otentikasi.



Gambar 3.5 *Use Case Diagram* pada modul registrasi



Gambar 3.6 *Use case diagram* pada modul otentikasi

a) Menambah Data Wajah

Pengguna dapat menambahkan data wajah baru yang nantinya bisa dikenali oleh perangkat lunak. Pengguna dapat mengunggah satu atau lebih foto wajah dan mengisi nama orang tersebut.

b) Melakukan *Training*

Pengguna dapat melihat data apa saja yang telah diunggah pada aplikasi dan berapa banyak jumlah foto dari setiap individu. Pengguna dapat melakukan *training* ulang untuk memastikan data baru telah di-*training*.

c) Melihat Video Pengenalan Wajah

Pengguna dapat melihat video pengenalan wajah. *User* dapat mengganti video dan memutarnya di halaman tersebut.

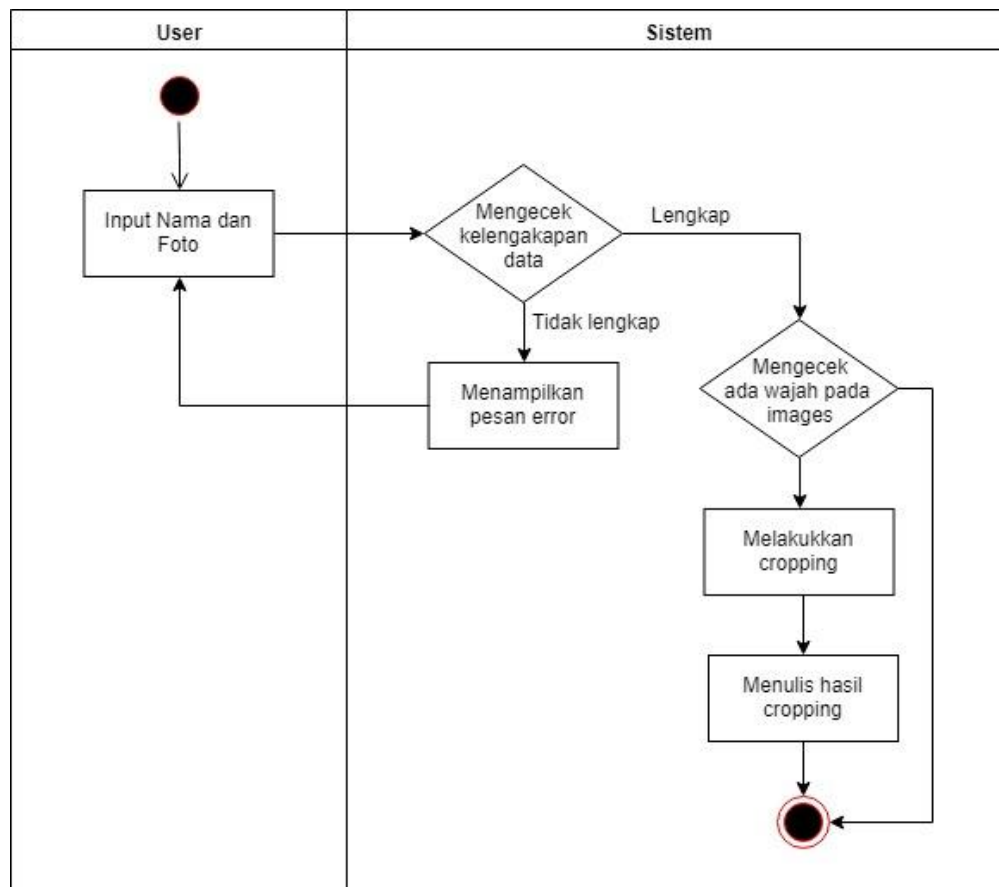
d) Melihat Laporan

Pengguna dapat melihat siapa saja yang telah tertulis di laporan. Laporan berisi 3 kolom yaitu nama, tanggal, dan jam masuk.

3.2.2 Activity Diagram

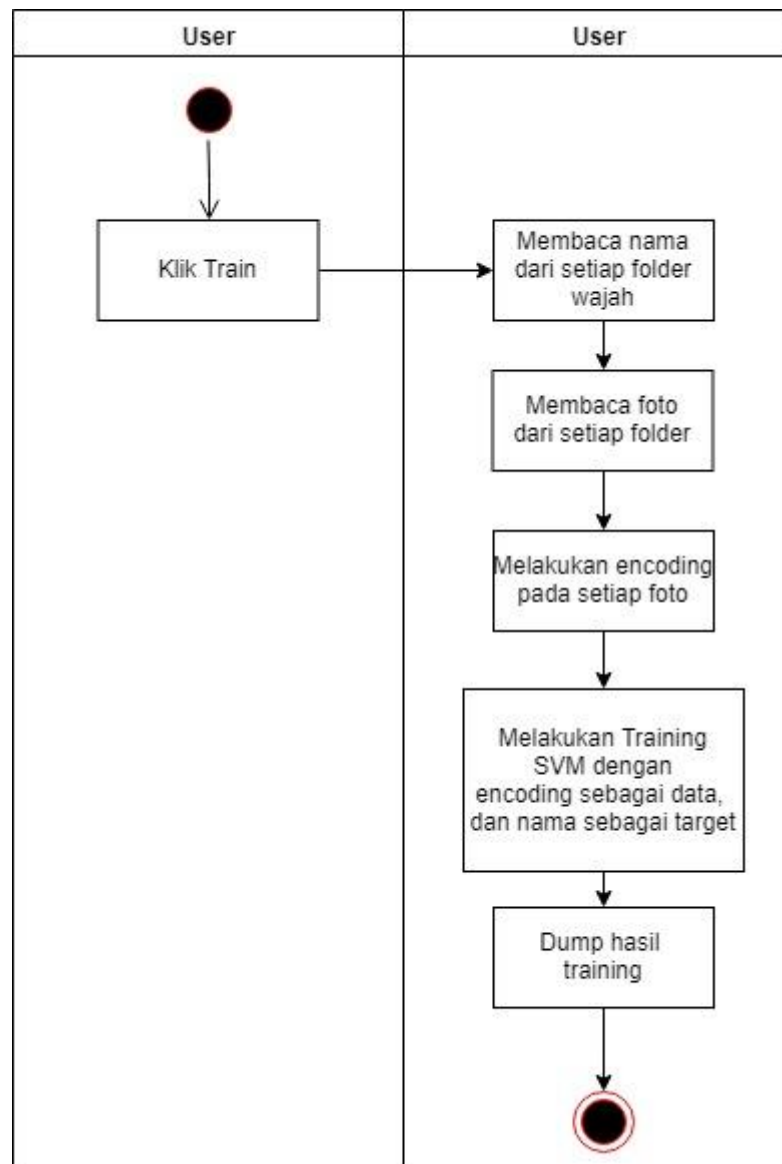
Pada tahap ini akan dijelaskan proses yang terjadi dalam suatu sistem. *Activity* diagram akan memodelkan *event* dari *use case diagram*. Hal ini dapat mempermudah pemahaman tentang bagaimana *event* bekerja.

Pada Gambar 3.7 menjelaskan tentang aktivitas pengguna dan sistem pada saat tambah data wajah. *User* akan menuliskan nama dan menunggah *file* foto orang yang bersangkutan. Bila nama atau *file* tidak diisi, akan muncul *error*. Selanjutnya sistem akan melakukan pencarian wajah pada gambar yang diunggah. Bila ada, akan dilakukan pemotongan gambar yang fokus hanya pada wajah. Bila tidak ada, sistem tidak akan menyimpan foto tersebut.



Gambar 3. 7 Activity diagram upload gambar

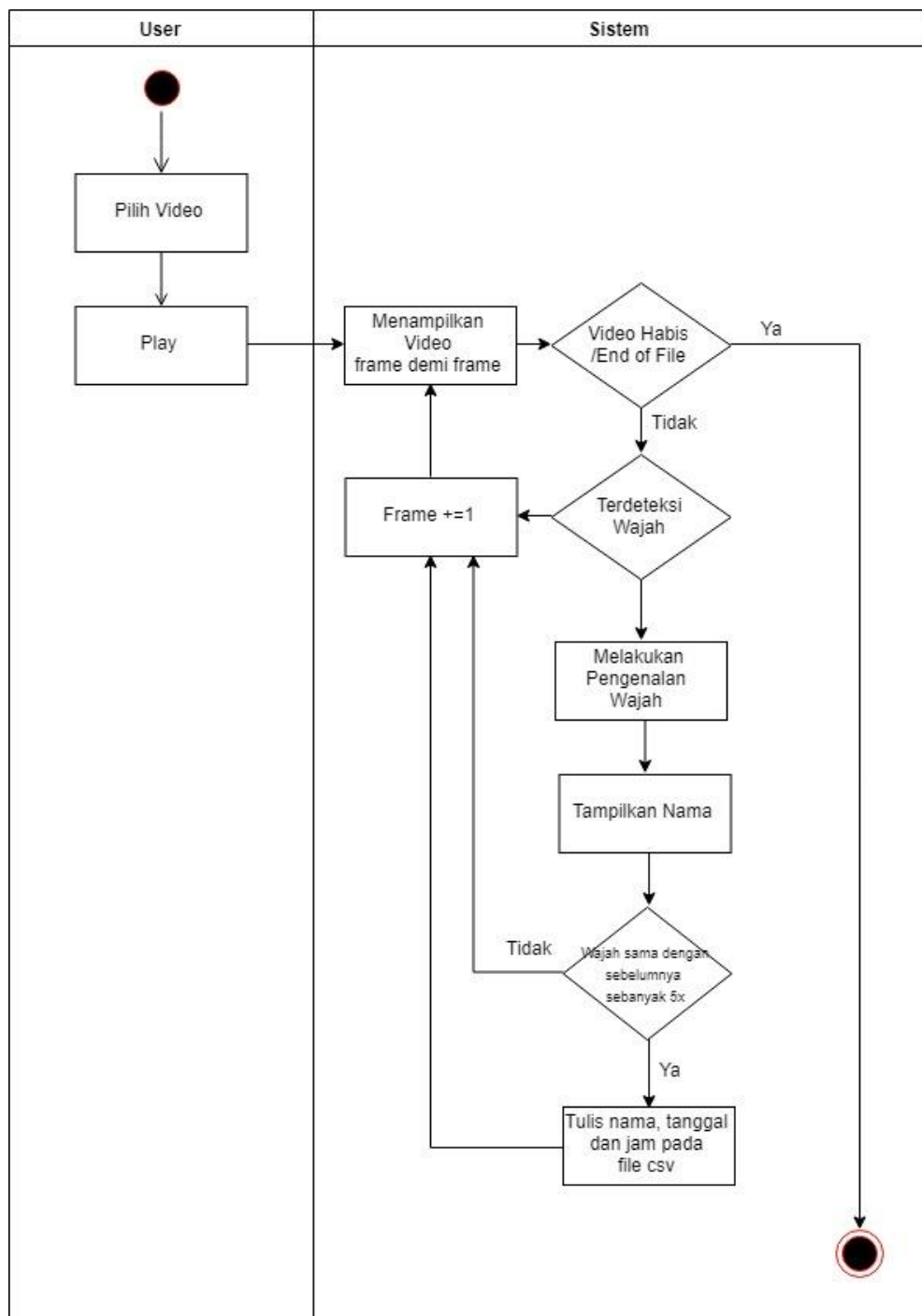
Pada Gambar 3.8 menjelaskan tentang bagaimana alur *training* pada perangkat lunak. Pada saat *user* mengklik *train*, sistem akan membaca nama nama dari setiap *folder*. Setelah itu sistem akan membaca foto dari setiap *folder* dan akan melakukan ekstraksi fitur atau biasa disebut *encoding*. Setelah data dan target sudah terkumpul, maka akan dilakukan pembelajaran SVM. Setelah selesai, sistem akan menyimpan hasil *training* dengan format *.joblib* yang dapat dibaca kembali.



Gambar 3.8 Activity diagram proses pembelajaran

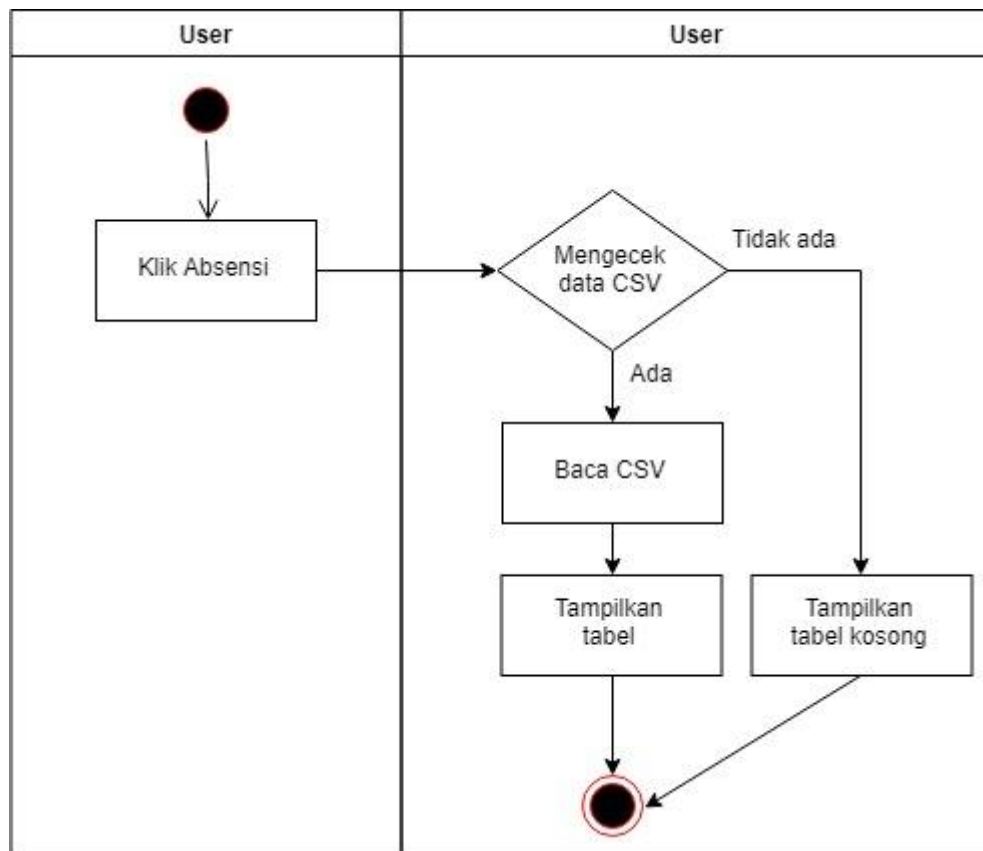
Pada Gambar 3.9 menjelaskan bagaimana aktivitas pengguna dan sistem pada fitur melihat video. Dapat dilihat pada diagram, pertama-tama akan dibaca apakah *frame* pada video sudah habis. Bila belum, maka akan dilakukan pencarian wajah. Bila tidak ditemukan akan langsung menuju ke *frame* berikutnya. Bila ditemukan, akan dilakukan pengenalan wajah atau disebut *predict*.

Sistem akan menulis nama, tanggal, dan jam bila terdeteksi wajah yang sama sebanyak 5 frame. Data tersebut akan ditulis pada *file* .csv. Tanggal dan jam merupakan waktu saat dibaca wajah tersebut.



Gambar 3.9 Activity diagram pada melihat video

Pada Gambar 3.10 menjelaskan tentang bagaimana sistem bekerja pada penampilan laporan. Saat *user* mengklik laporan, sistem akan mengecek terlebih dahulu adakah data csv yang dicari. Bila data csv ada, sistem akan membaca csv dan menampilkannya pada perangkat lunak. Sebaliknya bila tidak ditemukan csv, tabel yang ditampilkan akan kosong.



Gambar 3.10 Activity diagram laporan

3.2.3 Rancangan Desain Antarmuka

Pada tahap ini akan dilakukan perancangan aplikasi perangkat lunak berbasis web dengan menggunakan Flask. Tujuan dari desain antarmuka adalah

untuk membuat interaksi yang mudah, sederhana, dan efisien. Aplikasi yang dibuat akan terbagi menjadi dua modul yaitu otentikasi dan registrasi.

Modul otentikasi terdiri dari halaman utama dan laporan, sedangkan modul registrasi memiliki halaman tambah wajah dan *train* data. Berikut adalah rancangan desain antarmuka aplikasi pengenalan wajah:

a. Halaman Utama Otentikasi

Pada halaman ini akan ditampilkan video baik itu dari *file* ataupun melalui *streaming* dari *webcam*. Pada tampilan video akan ditampilkan *bounding box* bila terdeteksi adanya fitur wajah dan secara otomatis akan memberikan informasi nama dari wajah tersebut. Setelah wajah terbaca sebanyak 5 frame, maka akan tercatat nama, jam, dan tanggal orang tersebut masuk pada csv. Gambar 3.11 merupakan tampilan dari halaman utama.



Gambar 3.11 Halaman Utama

b. Halaman laporan

Halaman ini berfungsi untuk menampilkan laporan yang dibuat secara otomatis dari pengenalan wajah. Halaman ini menampilkan ID, tanggal, dan jam

mahasiswa tersebut terlihat di kamera/*video file*. Data pada halaman ini akan bertambah bila terdeteksi ada orang pada halaman utama. Halaman ini memiliki tombol “*download laporan*” untuk men-*download* rangkuman pada setiap tanggal. Gambar 3.12 merupakan tampilan dari halaman laporan.

Aplikasi Pengenalan Wajah

Home

Absensi

Absen


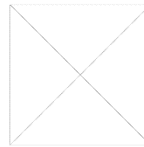
Download Laporan

Nama	Tanggal	Jam

Gambar 3.12 Halaman laporan

c. Halaman Tambah Data Wajah

Halaman ini berfungsi untuk menambahkan data wajah mahasiswa yang belum terdaftar. Pada halaman ini *user* dapat memilih metode untuk penambahan data. Metode yang ditawarkan adalah dengan mengunggah data gambar, atau memakai kamera. Gambar 3.13 merupakan rancangan tampilan pilihan tambah data wajah. Bila memilih metode *upload* maka akan muncul tampilan pada Gambar 3.14. Bila memakai kamera, maka akan muncul tampilan pada Gambar 3.15.

Aplikasi Pengenalan Wajah	
Tambah Data Wajah	Train Data
<div style="border: 1px solid black; padding: 5px; text-align: center; margin-bottom: 10px;">Pilih metode penambahan data</div> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>Upload</p> </div> <div style="text-align: center;">  <p>Kamera</p> </div> </div>	

Gambar 3.13 Tampilan pilihan tambah data wajah

Aplikasi Pengenalan Wajah	
Tambah Data Wajah	Train Data
<div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>Nama :</p> <div style="border: 1px solid black; padding: 2px; margin: 5px 0;">Masukkan nama</div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> <div style="border: 1px solid black; padding: 2px 10px;">Pilih Foto</div> <div style="font-size: 0.8em; color: gray;">Belum ada foto</div> <div style="border: 1px solid black; padding: 2px 10px;">Kirim</div> </div> </div>	

Gambar 3.14 Tampilan *upload* data wajah

Aplikasi Pengenalan Wajah	
Tambah Data Wajah	Train Data
<div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: 80%;"> <p>Nama :</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0; width: 80%;">Masukkan nama</div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0; width: 40%;">Lanjutkan Dengan Kamera</div> </div>	

Gambar 3.15 Tampilan tambah data wajah dengan kamera

Setelah *user* menekan tombol ‘Lanjutkan Dengan Kamera’, maka *user* melakukan *snapshot* wajahnya dengan *webcam*.

d. Halaman *Train Data*

Halaman ini berfungsi untuk melakukan proses pembelajaran. Pada tabel akan ditampilkan nama orang dan jumlah gambar yang sudah diunggah. Klik button “*Train Sekarang!*” untuk memulai melakukan pembelajaran. Gambar 3.16 merupakan tampilan dari halaman *train data*.

Aplikasi Pengenalan Wajah															
<div style="display: flex; justify-content: space-between;"> Tambah Data Wajah Train Data </div>															
<p style="margin-bottom: 10px;">Data yang Tersimpan</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; padding: 5px;">Nama</th> <th style="width: 50%; padding: 5px;">Jumlah Foto</th> </tr> </thead> <tbody> <tr><td style="height: 20px;"></td><td></td></tr> <tr><td style="height: 20px;"></td><td></td></tr> <tr><td style="height: 20px;"></td><td></td></tr> <tr><td style="height: 20px;"></td><td></td></tr> <tr><td style="height: 20px;"></td><td></td></tr> <tr><td style="height: 20px;"></td><td></td></tr> </tbody> </table> <div style="text-align: center; margin-top: 10px;"> <input type="button" value="Train Sekarang!"/> </div>		Nama	Jumlah Foto												
Nama	Jumlah Foto														

Gambar 3.16 Halaman *Train Data*

3.3 Alat dan Bahan Pembangunan Perangkat Lunak

Peralatan yang digunakan dalam pembangunan perangkat lunak ini terbagi menjadi dua spesifikasi, yaitu perangkat keras (*hardware*) dan perangkat lunak(*software*), yaitu sebagai berikut:

1. Spesifikasi Perangkat Keras (*Hardware*)
 - a. Laptop : Asus ZenBook 533FD.300
 - b. Processor 1.8GHz Intel Core i7-8565U
 - c. RAM 16 GB DDR3
 - d. Storage SSD 512 GB
2. Spesifikasi Perangkat Lunak (*Software*)
 - a. Windows 10 Pro 64-bit (10.0, Build 17763)
 - b. Python 3.51
 - c. Sublime *text*
 - d. Web Browser

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas hasil implementasi metode pengenalan wajah dan pengujian pada perangkat lunak yang dibuat.

4.1 Implementasi Antarmuka

Perangkat lunak ini dibangun menggunakan Python dan Flask sehingga perangkat lunak yang dibuat berbasis web. Aplikasi ini dibagi menjadi dua modul, yaitu registrasi dan otentikasi. Modul registrasi memiliki dua navigasi yaitu tambah data wajah, dan *training*. Modul otentikasi memiliki dua navigasi yaitu menampilkan video dan melihat laporan.

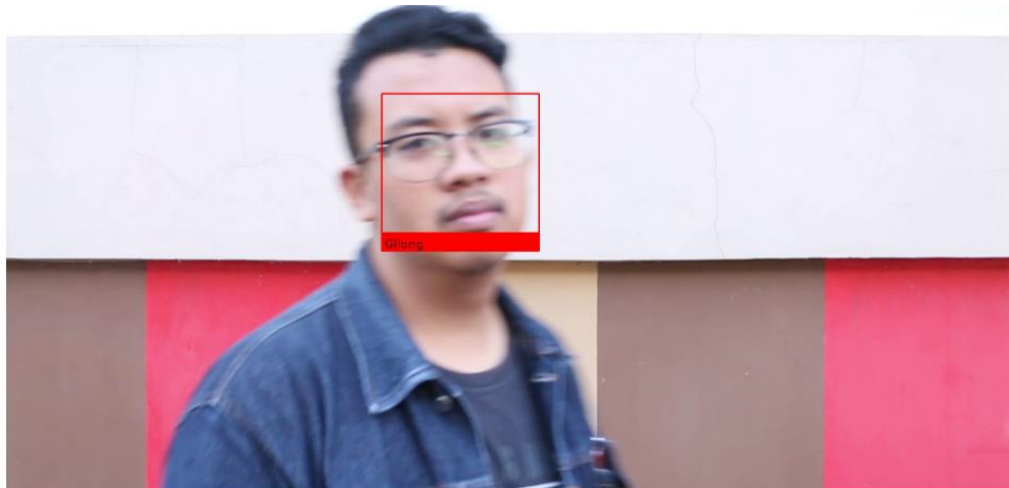
4.1.1 Tampilan Halaman Utama

Tampilan halaman utama merupakan tempat video akan ditampilkan beserta dengan identifikasi wajah. Bila terdeteksi ada wajah di dalam video, perangkat lunak secara otomatis akan menggambar kotak dengan nama didalamnya.

Pada halaman utama modul otentikasi, pengguna dapat memilih antara data video atau menggunakan *webcam*. Setelah dipilih, aplikasi akan menampilkan video sesuai dengan opsi yang dipilih. Pada saat proses berjalan, wajah orang yang terdeteksi akan diberi gambar kotak dengan nama dibawah kotak tersebut. Gambar 4.1 merupakan tampilan saat penampilan video yang wajahnya belum tercatat, sementara Gambar 4.2 merupakan tampilan saat wajah tersebut sudah tercatat.

Proses pencatatan laporan adalah ketika kotak yang awalnya berwarna merah berubah menjadi warna hijau. Proses ini terjadi ketika wajah orang yang sama

terdeteksi sama sebanyak 5 kali secara berturut turut. Data yang ditulis adalah Nama, tanggal, dan jam masuk. Data tersebut akan ditulis di *file* .csv.



Gambar 4.1 Halaman utama ketika wajah belum ditulis pada laporan.



Gambar 4.2 Halaman utama ketika sudah ditulis pada laporan

4.1.2 Tampilan Laporan

Laporan merupakan salah satu halaman yang ada pada modul otentikasi. Pada halaman ini akan ditampilkan laporan yang telah ditulis. Laporan tersebut

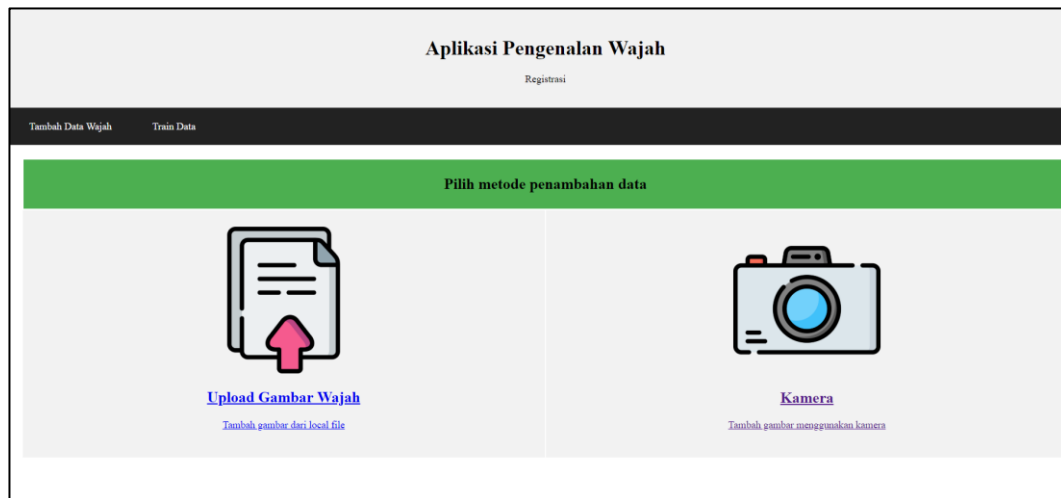
berisi nama, tanggal, dan waktu. Gambar 4.3 merupakan tampilan dari halaman laporan. Pada halaman laporan, pengguna dapat mengunduh rangkuman dari laporan berdasarkan tanggal.

Aplikasi Pengenalan Wajah		
Otentikasi		
Home	Laporan	
Laporan Download Laporan		
Nama	Tanggal	Jam Masuk
Yudha	09-August-19	08:25:59
Fazlur	09-August-19	08:25:45
Qisman	09-August-19	08:24:56
Hilmi	09-August-19	08:24:40
Gilang	09-August-19	08:24:24

Gambar 4.3 Tampilan laporan

4.1.3 Tampilan Tambah Data Wajah

Halaman tambah data wajah merupakan salah satu halaman pada modul registrasi. Pada halaman ini disediakan dua pilihan metode untuk menambah data wajah. Yang pertama adalah dengan mengunggah *file* foto dari *local file*, dan yang kedua adalah dengan menggunakan kamera. Gambar 4.4 menunjukkan tampilan pemilihan metode penambahan data.



Gambar 4.4 Halaman Opsi Tambah Data Wajah

Bila memilih ‘*Upload Gambar Wajah*’, akan ditampilkan sebuah halaman dengan form berisi *input text*, *input file*, dan *submit*. *Input file* yang diterima adalah seluruh tipe *file image*. Gambar 4.5 menunjukkan halaman ‘*Upload Gambar Wajah*’. Sedangkan bila memilih ‘*Kamera*’, akan muncul halaman dengan form berisi *input text*, dan *submit*.

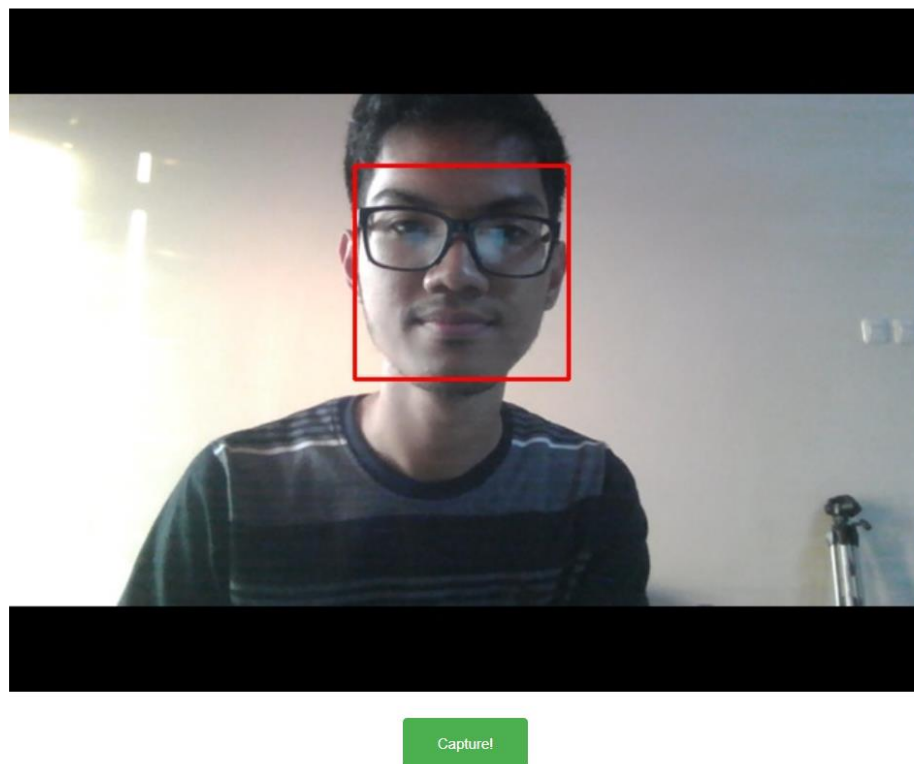
Gambar 4.5 Page upload gambar wajah

Gambar 4.6 menunjukkan halaman *upload* dengan kamera. Setelah diklik tombol ‘Lanjutkan dengan kamera’, maka pengguna akan diambil gambar wajahnya dengan menggunakan *webcam* yang terhubung.



Gambar 4.7 menunjukkan halaman saat pemotretan wajah.

Gambar 4.6 *Page upload* dengan kamera



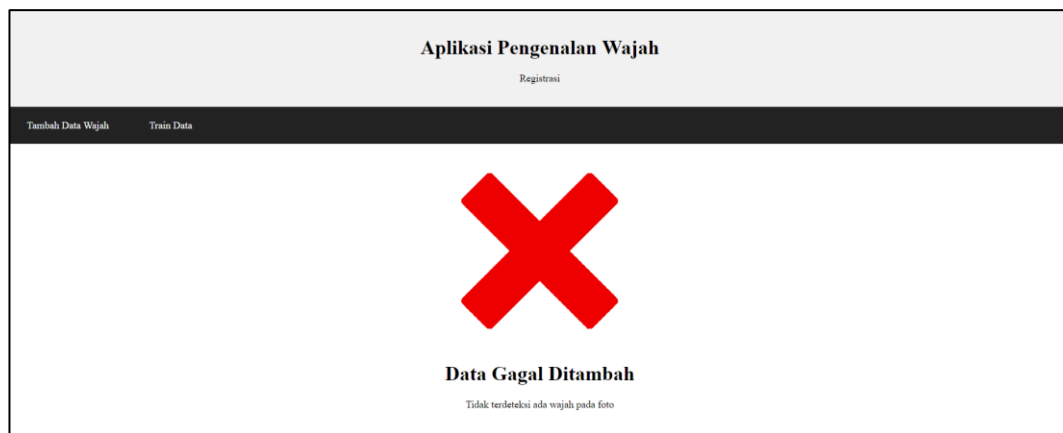
Gambar 4.7 *Page* saat melakukan pemotretan wajah

Bila telah melakukan proses *upload* dari *file* atau *upload* dari kamera, akan dimunculkan sebuah *page* untuk mengetahui apakah proses unggah berhasil. Bila foto berhasil diunggah, maka akan muncul tampilan berhasil pada Gambar 4.8.



Gambar 4.8 Tampilan berhasil tambah data

Bila gagal diunggah akan muncul tampilan gagal pada Gambar 4.9. Salah satu penyebab gambar gagal diunggah adalah saat gambar tidak memiliki elemen wajah.



Gambar 4.9 Tampilan gagal tambah data

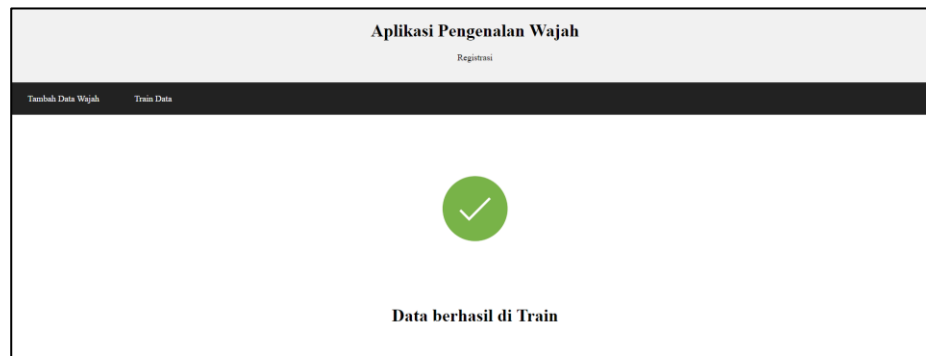
4.1.4 Tampilan *Train*

Train merupakan salah satu halaman pada modul registrasi. Pada halaman ini akan ditampilkan data apa saja yang telah diunggah pada perangkat lunak. Data yang ditampilkan adalah nama dan jumlah foto. Data ini akan ditampilkan dalam bentuk tabel yang dapat dilihat pada Gambar 4.10. Halaman ini memiliki *button* ‘*Train Sekarang*’ untuk menjalankan proses *training* pada sistem.

Aplikasi Pengenalan Wajah	
Registrasi	
Tambah Data Wajah	Train Data
Data yang Tersimpan	
Nama	Jumlah Foto
Bayu	10
Dicky	10
Fathur	10
Fazlur	10
Gilang	10
Hilmi	10
palur	10
Pandji	10
Qisman	10
Umar	10
Yudha	10
Train Sekarang!	

Gambar 4.10 Tampilan *Train*

Saat proses pelatihan berjalan, sistem akan menampilkan gambar loading. Bila *train* sudah dilakukan, maka akan muncul halaman berhasil seperti pada Gambar 4.11.

Gambar 4.11 Tampilan berhasil *Train*

4.2 Pengujian Data

Pada tahap ini akan dibahas tentang pengujian data pelatihan dalam *training* dan pengujian data dalam perangkat lunak. Akurasi pada model akan dicari menggunakan *K-fold cross validation* dengan $K=5$.

4.2.1 Pengujian Data Pelatihan

Pada bagian ini dilakukan pengujian data pelatihan dengan menggunakan *K-fold cross validation*. *K-fold* yang digunakan dalam pengujian adalah 5. Hasil pengujian ditampilkan dalam Tabel 4.1.

Tabel 4. 1 Hasil pengujian akurasi dengan *K-Fold Cross Validation*

Kernel SVM	<i>Fold-1</i>	<i>Fold-2</i>	<i>Fold-3</i>	<i>Fold-4</i>	<i>Fold-5</i>	Akurasi rata rata dari K- Fold Cross Validation
<i>Radial</i>	90%	90%	90%	90%	94.73%	90.94%
<i>Linear</i>	100%	100%	100%	100%	94.73%	98.94%
<i>Sigmoid</i>	90%	90%	90%	90%	94.73%	90.94%

Tabel 4.1 menuliskan hasil akurasi dari setiap *fold* pada *cross-validation* yang selanjutnya akan dicari rata rata dari kelima fold tersebut. Hasil dari percobaan menunjukkan bahwa *kernel radial* memiliki akurasi 90.94%, *linear* 98.94% dan *sigmoid* 90.94%. Dapat diambil kesimpulan bahwa kernel terbaik dalam permasalahan ini adalah *linear*.

4.2.2 Pengujian Data dengan Video

Pengujian akan dilakukan dengan mengambil data *groundtruth* dari video yang telah diambil. Data *groundtruth* akan dibandingkan dengan data prediksi. Data yang akan dibandingkan adalah data yang ditulis pada laporan dan *groundtruth* dari setiap video.

Tabel 4. 2 Hasil pengujian video pada resolusi 1920x1080

Nama <i>File</i> (.mp4)	<i>Groundtruth</i>	Hasil Pembacaan
Video_1.mp4	[Gilang, Hilmi, Qisman]	[Gilang, Hilmi, Qisman]
Video_2.mp4	[Pandji, Yudha, Hilmi]	[Pandji, Yudha, Hilmi]
Video_3.mp4	[Fazlur, Yudha]	[Fazlur, Yudha]
Video_4.mp4	[Bayu, Umar, Fathur]	[Bayu, Umar, Fathur]

Tabel 4. 3 Hasil pengujian video pada resolusi 960x540

Nama File (.mp4)	Groundtruth	Hasil Pembacaan
Video_1.mp4	[Gilang, Hilmi, Qisman]	[Gilang, Hilmi, Qisman]
Video_2.mp4	[Pandji, Yudha, Hilmi]	[Pandji, Yudha, Hilmi]
Video_3.mp4	[Fazlur, Yudha]	[Fazlur, Yudha]
Video_4.mp4	[Bayu, Umar, Fathur]	[Bayu]

Pada Tabel 4.2 menunjukkan dalam seluruh video dengan resolusi 1920x1080 pencatatan nama yang tertulis semuanya benar. Namun pada Tabel 4.3 terjadi kesalahan pembacaan pada Video_4.mp4 dengan resolusi 960x540. Pada video tersebut, wajah tidak terdeteksi dikarenakan wajah teralu kecil. Gambar 4.12 menunjukkan wajah yang tidak terdeteksi pada Video_4.mp4 dengan resolusi 960x540.



Gambar 4.12 Wajah yang tidak terdeteksi

Pada pembangunan perangkat lunak dilakukan juga pengujian dari seberapa cepat waktu eksekusi dan prediksi pada saat ditemukan wajah. Waktu akan dihitung hanya ketika ada wajah terdeteksi pada video.

Tabel 4. 4 Tabel waktu eksekusi

Nama <i>File</i> (.mp4)	Rata-rata detik/frame pada resolusi 1920x1080	Rata-rata detik/frame pada resolusi 960x540
Video_1.mp4	0.937371 detik	0.754695 detik
Video_2.mp4	0.683354 detik	0.456554 detik
Video_3.mp4	0.488566 detik	0.382856 detik
Video_4.mp4	1.029693 detik	0.893803 detik

Pada Tabel 4.4 ditunjukkan bahwa Video_4.mp4 memiliki rata-rata detik/frame terbesar. Dalam tabel tersebut juga dapat diambil kesimpulan resolusi berpengaruh pada kecepatan proses. Pada resolusi yang lebih besar, waktu proses lebih lama bila dibandingkan dengan resolusi yang lebih kecil. Rata rata waktu pada resolusi 1920x1080 adalah 0.784746 detik, sedangkan pada resolusi 960x540 adalah 0.621977 detik.

4.2.3 Pengujian Data Secara *Real-time*

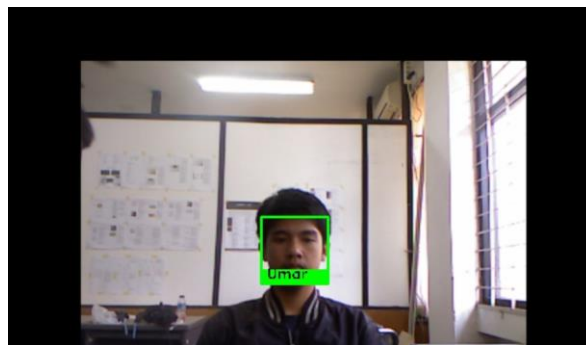
Pada bagian ini akan dilakukan pengujian dengan menggunakan *webcam*. *Webcam* yang digunakan memiliki resolusi 1280x720. Dari pengujian ini akan dilakukan perhitungan waktu pemrosesan saat terdeteksi wajah.

Didapatkan rata rata waktu pemrosesan adalah 0.663051 detik tiap *frame* yang memiliki wajah. Pada proses pengujian, wajah yang berada pada *webcam* tidak selalu terbaca dikarenakan resolusi yang terlalu kecil. Gambar 4.13 menunjukkan keadaan saat dilakukan pengujian *real-time*.



Gambar 4.13 Pengujian *real-time*

Sistem dapat mengenali wajah yang diambil secara langsung menggunakan kamera. Gambar 4.14 merupakan tampilan saat dilakukan pengujian secara *real-time*.



Gambar 4.14 Tampilan pengujian *real-time*

BAB V

SIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan serta saran terhadap pembangunan perangkat lunak yang sudah dikembangkan

5.1 Simpulan

Berdasarkan pembangunan perangkat lunak yang telah dilakukan, dapat disimpulkan bahwa:

1. *Library* Scikit-learn *Support Vector Machine* dapat digunakan dalam klasifikasi pengenalan wajah dengan bantuan *library* OpenFace.
2. Dalam pembuatan aplikasi pengenalan wajah dengan bahasa Python, dibutuhkan *library* yang dapat mendukung perangkat lunak yaitu: OpenCV sebagai pengolahan dasar citra; Dlib sebagai deteksi *frontal face* dengan menggunakan *Histogram of Oriented Gradient*; OpenFace sebagai pemposisian wajah dengan 68 *face landmark*, dan ekstraksi wajah menjadi 128 fitur; Scikit-learn SVM untuk klasifikasi data dan prediksi nama. *Library* Flask digunakan sebagai pembuatan web berbasis Python.
3. Dengan menggunakan *K-fold cross validation*, akurasi dari hasil *training* didapatkan 90.94% untuk kernel radial, 98.94% untuk kernel *linear*, dan 90.94% untuk kernel *sigmoid*. Kernel terbaik dalam kasus ini adalah kernel *linear*. Rata-rata waktu pemrosesan setiap frame jika ada wajah adalah 0.784746 detik.

5.2 Saran

Adapun saran-saran dari hasil perancangan aplikasi untuk pengembangan selanjutnya, agar aplikasi dapat berfungsi lebih baik adalah sebagai berikut:

1. Pada pengembangan perangkat lunak, proses pembelajaran memakan waktu yang cukup lama. Diharapkan proses pembelajaran dan prediksi dapat memberdayakan GPU sehingga proses dapat berjalan lebih cepat.
2. *User interface* yang dibuat masih sederhana. Maka dari itu, dapat dikembangkan tampilan yang lebih *user-friendly* sehingga pengguna lebih nyaman dan mudah menggunakan aplikasi.
3. Masih kurangnya kelengkapan pada fitur laporan. Dapat disempurnakan fitur laporan sehingga dapat menghasilkan rekapitulasi yang lebih lengkap.
4. Pengembangan aplikasi ini menggunakan *library* SVM dan OpenFace. Dapat digunakan *library* dan metode lain untuk membangun sistem pengenalan wajah.
5. Parameter yang digunakan dalam SVM masih manual. Dapat digunakan sistem hibrid untuk menentukan parameter pada SVM seperti dengan menggunakan algoritma genetika.

DAFTAR PUSTAKA

- Abraham, A., Pedregosa, F., Eickenberg, M., & Gervais, P. (2014). Machine Learning for Neuroimaging with Scikit-learn. *Frontiers in Neuroinformatics*, 8(February), 1–10. <https://doi.org/10.3389/fninf.2014.00014>
- Boyko, N., Basystiuk, O., & Shakhovska, N. (2018). Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and OpenCV Library. *Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018*, 478–482. <https://doi.org/10.1109/DSMP.2018.8478556>
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV*. O'Reilly.
- Hamilton, K., & Miles, R. (2006). *Learning UML 2.0*. O'Reilly.
- Kutty, N. M., & Mathai, S. (2017). Face Recognition - A Tool for Automated Attendance System. *International Journals of Advanced Research in Computer Science and Software Engineering ISSN: 2277-128X (Volume-7, Issue-6)*, (6), 334–336. <https://doi.org/10.23956/ijarcsse/V7I6/0268>
- Nugraha, S. P., Tullah, R., & Dzulhaq, M. I. (2017). Sistem Informasi Akademik Sekolah Berbasis Web Kurikulum 2013. *Jurnal Sisfotek Global*, 7(1), 1–5.
- Nugroho, S. A., Witarto, A. B., & Handoko, D. (2003). Support Vector Machine – Teori dan Aplikasinya dalam Bioinformatika. *Proceeding of Indonesian Scientific Meeting*.
- Patil, A., & Shukla, M. (2014). Implementation of Classroom Attendance System Based on Face Recognition in Class. *International Journal of Advances in Engineering & Technology*, 7(3), 974–979.

- Pedregosa Fabian, Michel, V., Grisel OLIVIER, Blondel, M., Prettenhofer, P., Weiss, R., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. Retrieved from <http://scikit-learn.sourceforge.net>.
- Pranoto, M. B., & Ramadhani, K. N. (2017). Face Detection System Menggunakan Metode Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) Face Detection System using Histogram of Oriented Gradients (HOG) Method amd Support Vector Machine (SVM). *Ie-Proceeding of Engineering : Vol.4, No.3 Desember 2017*, 4(3), 5038–5045.
- Putra, D. (2010). *Pengolahan Citra Digital*. Andi Offset.
- Rodríguez, J. D., Pérez, A., & Lozano, J. A. (2010). Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), 569–575.
<https://doi.org/10.1109/TPAMI.2009.187>
- Ronacher, A. (2010). Quickstart. Retrieved from <http://flask.pocoo.org/docs/0.12/quickstart/>
- Santi, N. C. (2011). Mengubah Citra Berwarna Menjadi Gray-Scale dan Citra biner. *Jurnal Teknologi Informasi DINAMIK*, 16(1), 14–19.

LAMPIRAN

Lampiran 1 Source Code Aplikasi

A. Server Flask otentikasi (Otentikasi.py)

```
import cv2
import os
import numpy as np
import pickle
import pandas as pd
import datetime
import csv
import face_recognition
import time

from datetime import date
from joblib import dump, load
from PIL import Image

from flask import Flask, render_template, request, Response,
redirect, url_for, send_from_directory
app = Flask(__name__)

APP_ROOT=os.path.dirname(os.path.abspath(__file__))

camera = cv2.VideoCapture('videos/video_1.mp4')
namaWajah = "a"
webcamPort = 0

@app.route("/")
def home():
    video_dir = os.listdir('videos')
    print(video_dir)
    names=[]
    for vid in video_dir:
        names.append(vid)

    return render_template('home.html',vid=names)

def get_frame():
    global statusVideo
    global camera
    try:
        camera
    except:
        camera=cv2.VideoCapture('videos/video_1.mp4')
    model=load('trained.joblib')

    waktuEksekusi=0
    frameCount=0
    validationCount=0
    face_temp=[]
```

```

lastWrite=[]
frameAdaWajah=0
color=(0, 0, 255) #merah
while True:
    ret, frame = camera.read()

    if ret == True and (frameCount%5)==0:
        start = time.time()
        small_frame = cv2.resize(frame, (0, 0), fx=0.25,
fy=0.25)
        rgb_small_frame = small_frame[:, :, ::-1]
        face_locations =
face_recognition.face_locations(rgb_small_frame)
        face_encodings =
face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names=[]

        for face_encoding in face_encodings:

            id = model.predict([face_encoding])
            name=id

            face_names.append(name)
            # print(face_temp,face_names)
            if(face_temp==face_names):
                validationCount+=1
                # print(validationCount)
            else:
                color=(0,0,255)
                validationCount=0
                face_temp=face_names

            for (top, right, bottom, left), name in
zip(face_locations, face_names):
                top *= 4
                right *= 4
                bottom *= 4
                left *= 4

                cv2.rectangle(frame, (left, top), (right,
bottom), color, 2)

                cv2.rectangle(frame, (left, bottom - 35),
(right, bottom), color, cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, str(name), (left + 6, bottom
- 6), font, 1.0, (0, 0, 0),1)

            # Tampilkan Gambar

            imgencode=cv2.imencode('.jpg',frame)[1]
            stringData=imgencode.tostring()

```



```

        yield (b'--frame\r\n'
               b'Content-Type:
text/plain\r\n\r\n'+stringData+b'\r\n')

        if face_names!=[]:
            print('ada wajah')
            waktuExek = float(time.time()-start)
            waktuEksekusi+=waktuExek
            frameAdaWajah+=1

        if len(face_names)>=1:
            if(validationCount==5):
                validationCount=0
                if(lastWrite!=face_names):
                    waktu = datetime.datetime.now()
                    row = [face_names[0],waktu.strftime("%d-
%B-%y"),waktu.strftime("%X")]
                    with open('absen.csv', 'a',newline='')
as csvFile:
                        writer = csv.writer(csvFile)
                        writer.writerow(row)
                        csvFile.close()

                        print('DITULIS')
                        lastWrite=face_names[0]
                        print(lastWrite)
                        color=(0,255,0)

            try:
                print('rata-rata: ',waktuEksekusi/frameAdaWajah)
            except:
                print('skip')
            frameCount=frameCount+1

            if cv2.waitKey(10) & 0xFF==ord('q'):
                break
        print('BREAK')
        del(camera)

def hapusCamera():
    global camera
    try:
        del(camera)
    except:
        pass
@app.route("/gantiVideo",methods=['POST'])
def gantiVideo():

    status=request.form['isi']
    global camera
    print(status)

    if status=="webcam":
        camera = cv2.VideoCapture(webcamPort)
    # elif status=="webcam2":
    #     camera = cv2.VideoCapture(1)
    else:

```

```

        camera = cv2.VideoCapture("videos/{}".format(status))
        return redirect(url_for('home_play'))

@app.route("/home_play")
def home_play():
    return render_template('home_play.html')

@app.route('/calc')
def calc():
    return Response(get_frame(), mimetype='multipart/x-mixed-
replace; boundary=frame')

@app.route("/absensi")
def absensi():
    hapusCamera()
    id_list = []
    dates = []
    times = []
    # absen=
    try:
        absen =
pd.read_csv("absen.csv", names=["Id", "Date", "Time"])
        id_list = absen["Id"].tolist()
        dates = absen["Date"].tolist()
        times = absen["Time"].tolist()

        id_list = id_list[::-1]
        dates = dates[::-1]
        times = times[::-1]
    except:
        with open("absen.csv", "w") as my_empty_csv:
            pass
        print('Tidak ada csv')
    pelaporan()
    return
render_template('absensi.html', ids=id_list, dates=dates, times=tim
es, dataCount=len(id_list))

def pelaporan():
    tanggalUnik=[]
    try:
        absen =
pd.read_csv("absen.csv", names=["Id", "Date", "Time"])
        print(absen)
        tanggalUnik = absen["Date"].unique().tolist()
        print(tanggalUnik)
    except:
        print('Absen Kosong')

    try:
        os.remove('laporan.csv')
    except:
        print("masih kosong")
    with open('laporan.csv', 'w', newline='') as csvFile:
        writer = csv.writer(csvFile)

```

```

        writer.writerow(["Tanggal", "Mahasiswa yang
        Hadir", "Jumlah Mahasiswa"])
        for tanggal in tanggalUnik:
            absenTanggal = absen[absen.Date==tanggal]
            orangUnik = absenTanggal["Id"].unique().tolist()
            row = [tanggal, orangUnik, len(orangUnik)]

            writer.writerow(row)
        csvFile.close()

@app.route("/get_csv")
def get_csv():
    filename = "laporan.csv"
    try:
        return send_from_directory(directory=APP_ROOT,
        filename=filename, as_attachment=True)
    except FileNotFoundError:
        abort(404)

if __name__ == '__main__':
    app.run(debug=True)

```

B. Server flask registrasi (Registrasi.py)

```

import cv2
import os
import numpy as np
import pickle
import pandas as pd
import csv
import face_recognition

from sklearn import svm
from sklearn.model_selection import cross_val_score

from joblib import dump, load
from PIL import Image

from flask import Flask, render_template, request, Response,
redirect, url_for, send_from_directory
app = Flask(__name__)

APP_ROOT=os.path.dirname(os.path.abspath(__file__))

camera = cv2.VideoCapture('videos/video_1.mp4')
namaWajah = "image"

@app.route("/")
def home():
    hapusCamera()
    return render_template('opsiTambah.html')

def get_frame_capture():
    global namaWajah

```

```

directory="images/{}".format(namaWajah)
target=os.path.join(APP_ROOT,directory)
if not os.path.isdir(target):
    os.mkdir(target)

camera = cv2.VideoCapture(0)
frameCount = 0
jumlahWajah=0
color = (0, 0, 255)
while True:
    ret, frame = camera.read()

    if ret == True and (frameCount%5)==0:
        small_frame = cv2.resize(frame, (0, 0), fx=0.25,
fy=0.25)
        gray_small_frame = small_frame[:, :, :-1]
        face_locations =
face_recognition.face_locations(gray_small_frame)
        print(face_locations)

        for (top, right, bottom, left) in face_locations:
            top *= 4
            right *= 4
            bottom *=4
            left *= 4
            cv2.rectangle(frame, (left, top), (right,
bottom), color, 2)

            try:
                face_image = frame[top:bottom, left:right]
                pil_image = Image.fromarray(face_image)
                filename = "{}
{}.jpeg".format(namaWajah,jumlahWajah)
                destination ="/".join([target, filename])
                print(filename)
                pil_image.save(destination)
                print('save')
                jumlahWajah+=1
            except:
                print('kosong')

            imgencode=cv2.imencode('.jpg',frame)[1]
            stringData=imgencode.tostring()
            yield (b'--frame\r\n'
                b'Content-Type:
text/plain\r\n\r\n'+stringData+b'\r\n')
            print(jumlahWajah)
            frameCount+=1
            if jumlahWajah>=10:
                break
        del(camera)
        gambarComplete= cv2.imread("static/Check.jpg")
        imgencode=cv2.imencode('.jpg',gambarComplete)[1]
        stringData=imgencode.tostring()
        yield (b'--frame\r\n'
            b'Content-Type: text/plain\r\n\r\n'+stringData+b'\r\n')

```

```

def hapusCamera():
    global camera
    try:
        del(camera)
    except:
        pass

@app.route('/calc2')
def calc2():
    return Response(get_frame_capture(), mimetype='multipart/x-
mixed-replace; boundary=frame')

@app.route("/upload", methods=['POST'])
def upload():
    nama = request.form['nama']
    directory='images/'+nama
    print(directory)
    target=os.path.join(APP_ROOT,directory)

    if not os.path.isdir(target):
        os.mkdir(target)

    fileUpload = request.files.getlist("file")
    print(fileUpload)
    i=0
    for file in fileUpload:
        print(file)
        face = face_recognition.load_image_file(file)
        try:
            facelocation = face_recognition.face_locations(face)
            top, right, bottom, left = facelocation[0]
            face_image = face[top:bottom, left:right]
            pil_image = Image.fromarray(face_image)

            filename = file.filename
            print(filename)
            destination = "/".join([target, filename])
            pil_image.save(destination)
            i=i+1
        except:
            print('tidak terdeteksi wajah')

    print("=====")

    if i==0:
        return render_template('failed.html')
    else:
        return render_template('completed.html', pesan1="Data
berhasil diupload", pesan2="Silahkan menuju Training untuk
melakukan pembelajaran")

```

```

@app.route("/tambahUpload")
def tambahUpload():
    return render_template('tambahUpload.html')

@app.route("/tambahKamera")
def tambahKamera():
    return render_template('tambahKamera.html')

@app.route("/tambahKameraCap", methods=['POST'])
def tambahKameraCap():
    global namaWajah
    namaWajah=request.form['nama']
    return render_template('tambahKameraCap.html')

@app.route("/train")
def train():
    hapusKamera()
    train_dir = os.listdir('images')
    names=[]
    counts=[]
    for person in train_dir:
        path = "images/" + person
        pix = os.listdir("images/" + person)
        if len(pix)==0:
            os.rmdir(path)
            continue
        names.append(person)
        counts.append(len(pix))
    return
    render_template('train.html', names=names, count=counts, dataCount=
len(names))

@app.route("/learning")
def learning():
    print('Mulai Learning')
    encodings = []
    names = []

    train_dir = os.listdir('images')
    for person in train_dir:
        pix = os.listdir("images/" + person)
        for person_img in pix:

            face = face_recognition.load_image_file("images/" +
person + "/" + person_img)
            try:
                face_enc =
face_recognition.face_encodings(face)[0]
                encodings.append(face_enc)
                names.append(person)
            except:
                print("Tidak terdapat wajah pada gambar")
                print(person)
    print(names)

    clf = svm.SVC(kernel='rbf', gamma='scale')

```

```

clf.fit(encodings,names)
akurasi=0

scores = cross_val_score(clf, encodings, names, cv=5)
print('RBF',scores)
print("Accuracy: %0.5f (+/- %0.5f)" % (scores.mean(),
scores.std() * 2))

modell = svm.SVC(kernel='linear',C=1)
modell.fit(encodings,names)
scores1 = cross_val_score(modell, encodings, names, cv=5)
print('Linear',scores1)
print("Accuracy: %0.5f (+/- %0.5f)" % (scores1.mean(),
scores1.std() * 2))

model2 = svm.SVC(kernel='sigmoid',gamma='auto')
model2.fit(encodings,names)
scores2 = cross_val_score(model2, encodings, names, cv=5)
print('Sigmoid',scores2)
print("Accuracy: %0.5f (+/- %0.5f)" % (scores2.mean(),
scores2.std() * 2))

print ('learning complete')
dump(modell,'trained.joblib')
return render_template('completed.html', pesan1="Data
berhasil di Train",pesan2="")

if __name__ == '__main__':
    app.run(debug=True)

```

Lampiran 2 Source Code *User Interface*

A. Halaman utama (home.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title></title>
    <!-- <link href="{{
url_for('static',filename='css/bootstrap.css') }}"
rel="stylesheet" /> -->
    <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">
    <script type="text/javascript">
        function validateForm() {
            var x = document.forms["pilihvideo"]["isi"].value;
            if ( x == null || x == "" || x=="Pilih Video") {
                alert("Pilih Video");
                return false;
            }
        }
    </script>
</head>

<body>

```

```

<!--    {% block content %}
    <div class="container">
        {% for message in get_flashed_messages() %}
        <div class="alert alert-warning">
            {{message}}
        </div>
        {% endfor %}
    </div>
    {% endblock %}
-->
<div class="header">
    <h1>Aplikasi Pengenalan Wajah</h1>
</div>

<ul>
    <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
    <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
    <li><a href="{{url_for('train')}}">Train Data</a></li>
    <li><a href="{{url_for('absensi')}}">Absensi</a></li>
</ul>

</div>


<form id="pilih_video" action="{{url_for('gantiVideo')}}"
method="POST" name="pilihvideo" onsubmit="return
validateForm()">
    <div align="center" style="margin: 20px;">
    <div class="select">
        <select name="isi" id="slct" align="center">
            <option selected disabled>Pilih Video</option>
            {% for video in vid %}
                <option value={{video}}>{{video}}</option>
            {%endfor%}
        <!--            <option value="0">Video 1</option>
                <option value="1">Video 2</option>
                <option value="2">Video 3</option> -->
                <option value="webcam">WebCam</option>
        </select>
    </div>
    <input type="submit" value="Go">
    </div>
</form>

</body>
</html>

```

B. Opsi tambah data (opsiTambah.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">

```



```

<title></title>
<!-- <link href="{{
url_for('static',filename='css/bootstrap.css') }}"
rel="stylesheet" /> -->
<link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">

</head>

<body>

<div class="header">
<h1>Aplikasi Pengenalan Wajah</h1>
</div>

<ul>
<li><a class="active"
href="{{url_for('home')}}">Home</a></li>
<li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
<li><a href="{{url_for('train')}}">Train Data</a></li>
<li><a href="{{url_for('absensi')}}">Absensi</a></li>
</ul>

<div width="100%" style="margin:20px">
<table>
<tr>
<td colspan="2">
<div style="padding: 5px;text-align: center;vertical-
align:middle;background-color: #4CAF50;">
<h2>Pilih metode penambahan data</h2>
</div>
</td>
</tr>
<tr>
<td>
<a href="{{url_for('tambah')}}">
<div class="pilihan">
</img>
<h2>Upload Gambar Wajah</h2>
<p>Tambah gambar dari local file</p>
</div>
</a>
</td>
<td>
<a href="{{url_for('tambahKamera')}}">
<div class="pilihan">
</img>
<h2>Kamera</h2>
<p>Tambah gambar menggunakan kamera</p>
</div>
</a>
</td>
</tr>

```

```

    </table>
  </div>
</body>

```

C. Tambah dengan *upload file* (tambahUpload.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title></title>
  <!-- <link href="{{
url_for('static',filename='css/bootstrap.css') }}"
rel="stylesheet" /> -->
<link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">

  <script type="text/javascript">
    function validateForm() {
      var x = document.forms["tambahOrang"]["nama"].value;
      var y = document.forms["tambahOrang"]["file"].value;
      if ( x == null || x == "" ) {
        alert("Mohon isi nama");
        return false;
      }
      if( y == null || y == "" ){
        alert("Mohon upload file foto")
        return false;
      }
      document.getElementById("overlay").style.display =
"block";
    }
  </script>
  <style>
  #overlay {
    position: fixed;
    display: none;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0,0,0,0.5);
    z-index: 2;
  }

  #textoverlay{
    position: absolute;
    top: 50%;
    left: 50%;
    color: white;
    transform: translate(-50%,-50%);
    -ms-transform: translate(-50%,-50%);
  }

```

```

    }

    </style>
</head>

<body>
    <div id="overlay">
        <div id="textoverlay"></div>
    </div>

    <div class="header">
        <h1>Aplikasi Pengenalan Wajah</h1>
    </div>

    <ul>
        <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
        <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
        <li><a href="{{url_for('train')}}">Train Data</a></li>
        <li><a href="{{url_for('absensi')}}">Absensi</a></li>
    </ul>

    <div style="border-radius: 5px;background-color:
#f2f2f2;padding: 100px;margin: 20px">
        <form id="upload-form" action="{{url_for('upload')}}"
method="POST" enctype="multipart/form-data" name="tambahOrang"
onsubmit="return validateForm()">
            <label for="fname">First name:</label>
            <input type="text" name="nama" id="fname"
placeholder="Your Name..">
            <input type="file" name="file" accept="image/*"
multiple="">
            <input type="submit" value="send">
        </form>
    </div>
</body>

```

D. Tambah dengan Kamera (tambahKamera.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title></title>
    <!-- <link href="{{
url_for('static',filename='css/bootstrap.css') }}"
rel="stylesheet" /> -->
<link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">

    <script type="text/javascript">
        function validateForm() {

```

```

        var x = document.forms["tambahOrang"]["nama"].value;
        if ( x == null || x == "" ) {
            alert("Mohon isi nama");
            return false;
        }
    }
</script>
</head>

<body>

    <div class="header">
        <h1>Aplikasi Pengenalan Wajah</h1>
    </div>

    <ul>
        <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
        <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
        <li><a href="{{url_for('train')}}">Train Data</a></li>
        <li><a href="{{url_for('absensi')}}">Absensi</a></li>
    </ul>

    <div style="border-radius: 5px;background-color:
#f2f2f2;padding: 100px;margin: 20px">
        <form id="upload-form"
action="{{url_for('tambahKameraCap')}}" method="POST"
enctype="multipart/form-data" name="tambahOrang"
onsubmit="return validateForm()">
            <label for="fname">First name:</label>
            <input type="text" name="nama" id="fname"
placeholder="Your Name..">
            <input type="submit" value="Lanjutkan Dengan Kamera">
        </form>

    </div>
</body>

```

E. Capture Tambah Kamera (uploadKameraCap.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">
    <script type="text/javascript">
        function validateForm() {
            var x = document.forms["pilihvideo"]["isi"].value;
            if ( x == null || x == "" || x=="Pilih Video") {
                alert("Pilih Video");
                return false;
            }
        }
    </script>

```

```

    }
    </script>
</head>

<body>

    <div class="header">
        <h1>Aplikasi Pengenalan Wajah</h1>
    </div>

    <ul>
        <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
        <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
        <li><a href="{{url_for('train')}}">Train Data</a></li>
        <li><a href="{{url_for('absensi')}}">Absensi</a></li>
    </ul>

    </div>
    

</form>
</body>
</html>

```

F. Train Data

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">

    <script type="text/javascript">
        function validateForm() {
            document.getElementById("overlay").style.display =
"block";
        }
    </script>
    <style>
    #overlay {
        position: fixed;
        display: none;
        width: 100%;
        height: 100%;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background-color: rgba(0,0,0,0.5);
```

```

        z-index: 2;
    }

    #textoverlay{
        position: absolute;
        top: 50%;
        left: 50%;
        color: white;
        transform: translate(-50%,-50%);
        -ms-transform: translate(-50%,-50%);
    }

</style>
</head>

<body>
    <div id="overlay">
        <div id="textoverlay"></div>
        </div>
        <div class="header">
            <h1>Aplikasi Pengenalan Wajah</h1>
        </div>

        <ul>
            <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
            <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
            <li><a href="{{url_for('train')}}">Train Data</a></li>
            <li><a href="{{url_for('absensi')}}">Absensi</a></li>
        </ul>

        <div style="margin:50px">
            <h1>Data yang Tersimpan</h1>
            <table>
                <tr style="background-color: #4CAF50;color: white">
                    <th class="tableRapih">Nama</th>
                    <th class="tableRapih">Jumlah Foto</th>
                </tr>
                {%for x in range(0,dataCount) %}
                <tr>
                    <th class="tableRapih">{{ names[x] }}</th>
                    <th class="tableRapih">{{ count[x] }}</th>
                </tr>
                {% endfor %}
            </table>
            <div align="center" style="margin:10px">
                <form id="pilih_video"
action="{{url_for('learning')}}" onsubmit="return
validateForm()">
                    <input type="submit" value="Train Sekarang!">
                </form>
            </div>
        </div>
    </body>

```

```
</html>
```

G. Absensi

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
    <!-- <link href="{{
url_for('static',filename='css/bootstrap.css') }}"
rel="stylesheet" /> -->
    <link rel="stylesheet" type="text/css" href="{{
url_for('static',filename='css/style.css') }}">
</head>

<body>
    <div class="header">
        <h1>Aplikasi Pengenalan Wajah</h1>
    </div>

    <ul>
        <li><a class="active"
href="{{url_for('home')}}">Home</a></li>
        <li><a href="{{url_for('opsiTambah')}}">Tambah Data
Wajah</a></li>
        <li><a href="{{url_for('train')}}">Train Data</a></li>
        <li><a href="{{url_for('absensi')}}">Absensi</a></li>
    </ul>

    <div style="margin:20px;">
        <h1>Absen</h1>
        <a href="{{url_for('get_csv')}}"><p>Download Laporan</p></a>
        <table>
            <tr style="background-color: #4CAF50;color: white">
                <th class="tableRapih">Nama</th>
                <th class="tableRapih">Tanggal</th>
                <th class="tableRapih">Jam Masuk</th>
            </tr>
            {%for x in range(0,dataCount) %}
            <tr>
                <th class="tableRapih">{{ ids[x] }}</th>
                <th class="tableRapih">{{ dates[x] }}</th>
                <th class="tableRapih">{{ times[x] }}</th>
            </tr>
            {% endfor %}
        </table>
    </div>
</body>
</html>
```

RIWAYAT HIDUP

DATA PRIBADI

Nama Lengkap : Fazlur Rahman
NPM : 140810150057
Tempat,Tanggal Lahir: Bandung, 17 Oktober 1997
Jenis Kelamin : Laki – laki
Gol. Darah : O
Agama : Islam
Alamat : Jl. Roket Raya no 42 Sangkuriang RT/RW 02/20,
Cimahi 40511
No Telepon : +62818-0693-1210
Email : fazlur97@gmail.com

RIWAYAT PENDIDIKAN

2003 s.d. 2004 : TK Dayang Sumbi Cimahi
2004 s.d. 2009 : SD Negeri Cimahi Mandiri 1
2009 s.d. 2012 : SMP Negeri 3 Cimahi
2012 s.d. 2015 : SMA Negeri 2 Cimahi
2015 s.d. sekarang : Universitas Padjadjaran, Jurusan Teknik Informatika

RIWAYAT ORGANISASI

Instansi	Jabatan	Tahun
Badan Eksekutif Himatif FMIPA UNPAD	Kepala Departemen Minat dan Bakat	2017
Badan Eksekutif Himatif FMIPA UNPAD	Staff Departemen Minat Bakat	2016

RIWAYAT KEPANITIAAN

Instansi	Nama Acara	Jabatan	Tahun
UNPAD	ASEAN Youth Initiative Conference (AYIC)	Staff Visual Communication	2018
Himatif FMIPA UNPAD	Informatics Sport Art and Games (Instagram)	Steering Committee	2017
Himatif FMIPA UNPAD	Character Building Season	Kepala Bidang II	2017
UNPAD	Internal Competition UNPAD for NUDC	Staff Dokumentasi	2017
FMIPA UNPAD	MIPA Bersatu	Staff Fasilitator	2016
Himatif FMIPA UNPAD	Informatics Festival (IFEST)	Staff Humas	2016
Himatif FMIPA UNPAD	Technopreneurship	Staff Sponsorship and Ticketing	2016
ESU UNPAD	Seminar My Cyber Personality	Koordinator Dokumentasi	2016

RIWAYAT PRESTASI

Nama Acara	Penyelenggara	Tahun	Keterangan
Regional Hult Prize Oslo	Hult Prize Foundation	2019	12 besar
Huawei in University ICT Competition	Huawei	2018	Juara 1
CIMB 3D Conquest Fintech	CIMB	2018	Semifinalist
Seeds for The Future	Huawei	2018	Delegasi Indonesia
International Monochrome Photography Award	Monochrome Award	2018	Honorable Mention in 'Amateur' People Category
Musikalisasi Puisi FORSI	UNPAD	2017	Juara 2
Universal Networking Empowerment Organization Student	UNEOS	2017	Delegasi Indonesia