TUGAS AKHIR - KI141502

SISTEM PENGENALAN WAJAH BERBASIS VIDEO UNTUK MONITORING PINTU

LUQMAN AHMAD
0511440000187

Dosen Pembimbing
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI141502

SISTEM PENGENALAN WAJAH BERBASIS VIDEO UNTUK MONITORING PINTU

LUQMAN AHMAD
05111440000187

Dosen Pembimbing I
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

VIDEO BASED FACE RECOGNITION SYSTEM FOR DOOR MONITORING

LUQMAN AHMAD
05111440000187

Supervisor I
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

Supervisor II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

SISTEM PENGENALAN WAJAH BERBASIS VIDEO UNTUK MONITORING PINTU

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
LUQMAN AHMAD
NRP: 05111440000187

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr.Eng. Nanik Suciati, S.Kom., M.Kom.
NIP. 197104281994122001 ..... (Pembimbing 1)

Dini Adni Navastara, S.Kom., M.Sc.
NIP. 198510172015042001 ..... (Pembimbing 2)

SURABAYA
JUNI, 2018

[*Halaman ini sengaja dikosongkan*]

SISTEM PENGENALAN WAJAH BERBASIS VIDEO UNTUK MONITORING PINTU

Nama Mahasiswa : Luqman Ahmad
NRP : 05111440000187
Departemen : Informatika, FTIK ITS
Dosen Pembimbing 1 : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.
Dosen Pembimbing 2 : Dini Adni Navastara, S.Kom., M.Sc.

Abstrak

Sistem pengenalan wajah merupakan suatu sistem yang berguna untuk mengidentifikasi wajah seseorang melalui citra wajah orang tersebut. Sementara itu, video merupakan sarana multimedia yang mengkombinasikan citra secara berurutan sehingga membentuk citra yang seolah – olah bergerak. Sistem pengenalan wajah berbasis video merupakan sistem pengidentifikasi wajah seseorang bukan hanya melalui satu citra melainkan juga memperhatikan beberapa citra yang diperoleh melalui video.

Tugas akhir ini mengembangkan sistem untuk mengidentifikasi seseorang ketika memasuki ruangan. Sebuah kamera diletakkan menghadap pintu masuk untuk meperoleh data citra wajah seseorang. Sistem akan memproses data tersebut dan kemudian memberikan hasil pengenalan.

Sistem ini terdiri dari tiga tahapan utama, yakni (1) tahap deteksi wajah, (2) tahap ekstraksi fitur, dan (3) tahap klasifikasi. Tahap deteksi wajah bertujuan untuk memisahkan citra wajah dari citra input keseluruhan menggunakan algoritma Seeta Face dan Kalman Filter. Tahap ekstraksi fitur bertujuan memperoleh fitur representatif citra wajah menggunakan Discrete Cosine Transform. Tahap klasifikasi adalah tahap pengambilan kesimpulan suatu citra wajah menggunakan algoritma k-Nearest Neighbour.

Uji coba meggunakan 31 video, terdiri dari 18 video untuk training dan 13 video untuk testing. Tiap video terdiri dari satu

orang yang sedang memasuki ruangan. Data video tersebut memiliki 8 kelas yang merupakan nama orang tertentu. Uji coba pengenalan wajah berbasis video menghasilkan akurasi 100%, sementara uji coba pengenalan wajah berbasis frame menghasilkan akurasi 70,73%.

Kata kunci: deteksi wajah, pengenalan wajah, seeta face detection, kalman filter, discrete cosine transform, k-nearest neighbour.

VIDEO BASED FACE RECOGNITION SYSTEM FOR DOOR MONITORING

Student Name	: Luqman Ahmad
Registration Number	: 05111440000187
Department	: Informatics Department, FTIK ITS
First Supervisor	: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Second Supervisor	: Dini Adni Navastara, S.Kom., M.Sc.

Abstract

Face recognition system is a system used for identifying face through the image of that person. Meanwhile, video is multimedia source combines sequences of image to form a moving picture. Video based face recognition system is face identifier system not only from one image but also considering sequence of image extracted from video.

This final project develop system identifying someone when he is entering a room. A camera placed in the room facing the entrance door for collection face image data. System will process the data and then give the recognition result.

This system consists of three main processes, (1) face detection process, (2) feature extraction process, and (3) classification process. Face detection process intended for separate face image from a whole input image from camera using Seeta Face Detection and Kalman Filter. Feature extraction process intended for obtaining representative feature of face image using Discrete Cosine Transform. Classification process is process of concluding face image using k-Nearest Neighbour.

Testing use 31 videos, consist of 18 videos as training set and 13 videos as testing set. Each video consists of one person entering the room. Video data have 8 classes which is name of the person. Testing of video-based face recognition system give 100% accuracy, while testing of frame-based face recognition system give 70,73% accuracy.

Keywords: face detection, face recognition, seeta face detection, kalman filter, discrete cosine transform, k-nearest neighbour.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"Sistem Pengenalan Wajah Berbasis Video untuk Monitoring Pintu"**.

Buku tugas akhir ini disusun dengan harapan dapat memberikan manfaat dalam penelitian deteksi wajah dan pengenalan wajah serta dapat berguna untuk memantau dan meningkatkan keamanan di Laboratorium Komputasi Cerdas dan Visi, Departement Informatika, ITS.

Dalam perancangan, pengerjaan dan penyusunan tugas akhir ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Penulis ingin mengucapkan terima kasih kepada:

1. Dr.Eng. Nanik Suciati, S.Kom., M.Kom. dan Dini Adni Navastara, S.Kom., M.Sc. selaku dosen pembimbing penulis yang telah memberi ide, nasihat dan arahan sehingga penulis dapat menyelesaikan tugas akhir dengan tepat waktu.
2. Orang tua penulis Bapak Yuniar Saptotri dan Ibu Amin Sholikah yang telah memberikan dukungan moral, spiritual dan material serta senantiasa memberikan doa demi kelancaran dan kemudahan penulis dalam mengerjakan tugas akhir.
3. Adik kandung (Luthfi, Fiina, dan Ais) dan seluruh keluarga besar yang telah memberikan dukungan yang besar baik secara langsung maupun secara implisit.
4. Teman – teman seperjuangan Antonius Kevin, Jeffry Nasri, Kukuh Rilo dan Hendri Febriansyah yang senantiasa berbagi pengalaman hidup, suka dan duka selama masa perkuliahan di Teknik Informatika, ITS.
5. Teman-teman Admin Laboratorium KCV dan seluruh Sahabat KCV yang telah banyak berbagi pengetahuan,

pengalaman, dan inspirasi serta bersedia menjadi dataset yang digunakan dalam Tugas Akhir ini (Mala, Galang, Chasni, Nuzul, Randi, Fadli, Ocid, Dandy).

6. Teman – teman Aku Pintar (Kukuh, Kevin, Mas Divi, Mba Ekky, Mba Rere, Mas Iyus, Mas Gilang, Mas Pebri, Mas Ali) yang senantiasa menemani, memberi dukungan, semangat dan inspirasi selama pengerjaan Tugas Akhir ini.
7. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan, kelalaian maupun kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Juni 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Aplikasi sistem pengenalan wajah dapat memberikan berbagai manfaat bagi kehidupan manusia. Misalnya, sistem pengenalan wajah dapat digunakan untuk menandai seseorang pada foto secara otomatis sehingga kita tidak perlu menandai satu per satu setiap orang yang terdapat pada sebuah foto. Sistem pengenalan wajah juga dapat digunakan untuk memudahkan kita dalam pengaturan hak akses. Sistem akan memindai wajah sesoerang kemudian memberikan hak akses kepada orang tersebut atau menolaknya. Selain itu, sistem pengenalan wajah dapat digunakan untuk mengawasi suatu lokasi dengan bantuan kamera pengawas. Sistem akan mengenali setiap orang yang melewati lokasi tersebut dan kemudian mencatatnya sebagai laporan yang dapat digunakan jika diperlukan di kemudian hari.

Dibalik berbagai manfaat yang ditawarkan, pengembangan sistem pengenalan wajah masih menjadi tantangan besar di bidang *computer vision*. Hal ini disebabkan wajah seseorang dapat memiliki banyak variasi jika direpresentasikan dalam citra digital. Tingkat pencahayaan yang berbeda dapat menyebabkan representasi citra digital yang berbeda. Pada citra digital, citra yang gelap direpresentasikan dengan nilai rendah, sedangkan citra yang terang direpresentasikan dengan nilai tinggi. Perbedaan ekspresi seseorang juga menyebabkan perbedaan komposisi data pada citra digital. Sebagai contoh, wajah seseorang yang sedang tersenyum dan wajah seseorang yang sedang tertawa akan memberikan representasi citra digital yang berbeda. Selain itu, perbedaan posisi wajah, seperti mengengok ke kiri dan kanan, menengadah, atau

menunduk, menghasilkan representasi citra digital yang sangat berbeda.

Banyaknya variasi citra wajah tersebut memerlukan pengumpulan citra wajah yang bervariasi pula agar proses klasifikasi berjalan dengan baik. Pengumpulan citra wajah yang bervariasi dapat dilakukan melalui video yang pada dasarnya merupakan citra – citra yang disusun berurutan. Dengan menggunakan video, kita dapat mengumpulkan variasi citra wajah seseorang lebih banyak dari variasi citra wajah yang bisa didapat melalui citra.

Meskipun telah memiliki citra wajah yang bervariasi, kesalahan klasifikasi masih dapat terjadi ketika melakukan klasifikasi citra wajah. Untuk mengurangi kesalahan tersebut, penggunaan video dapat berperan sebagai verificator terhadap hasil klasifikasi. Hal ini dilakukan dengan memperhatikan hasil klasifikasi dari citra sebelum atau setelahnya pada video. Apabila terdapat perbedaan klasifikasi, langkah penyesuaisan dapat dilakukan berdasarkan hasil klasifikasi citra sebelum atau setelahnya sehingga dapat mengurangi kesalahan pengenalan wajah.

Tugas akhir ini akan mengembankan sistem pengenalan wajah berbasis video untuk mengawasi pintu masuk ruangan. Pintu masuk dipilih sebagai objek yang diamati karena ketika memasuki ruangan merupakan salah satu momen terbaik untuk mendapatkan variasi wajah seseorang. Selain itu, penerapan sistem pengenalan wajah pada pintu masuk dapat dimanfaatkan untuk berbagai keperluan, seperti melakukan pencatatan orang - orang yang memasuki suatu ruangan, menghitung banyaknya kunjungan suatu ruangan, atau memberikan notifikasi ketika ruangan dimasuki oleh seseorang. Sistem ini akan menggunakan algoritma Seeta Face untuk memisahkan citra wajah dari citra utuh. Untuk pemilihan fitur, sistem ini akan menggunakan fitur ekstraksi berbasis local discrete cosine transform yang mampu menghadapi variasi citra wajah. Proses klasifikasi menggunakan algoritma *k-Nearest*

Neighbour. Penggunaan video pada sistem ini diharapkan dapat memberikan hasil pengenalan wajah yang lebih akurat.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan deteksi wajah?
2. Bagaimana melakukan tracking wajah menggunakan algoritma Kalman Filter?
3. Bagaimana melakukan ekstraksi fitur menggunakan Discrete Cosine Transform?
4. Bagaimana melakukan klasifikasi berbasis video menggunakan k-Nearest Neighbour?
5. Bagaimana melakukan uji coba performa sistem pengenalan wajah berbasis video?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Dataset diperoleh dari kamera CCTV yang diletakkan di dalam Laboratorium Komputasi Cerdas dan Visi, Departemen Informatika, ITS menghadap pintu masuk.
2. Implementasi sistem menggunakan bahasa pemrograman Python.
3. Sistem pengenalan wajah dirancang hanya untuk mendeteksi satu wajah pada pada frame video.

1.4 Tujuan Tugas Akhir

Tujuan pembuatan Tugas Akhir ini adalah mengembangkan sistem pengenalan wajah berbasis video untuk monitoring pintu menggunakan algoritma Seeta Face Detection untuk deteksi wajah, Kalman Filter untuk *tracking* wajah, Discrete Cosine Transform untuk ekstraksi fitur, dan *kNN* untuk klasifikasi.

1.5 Manfaat Tugas Akhir

Manfaat pembuatan Tugas Akhir ini adalah :

1. Menghasilkan sistem pengenalan wajah bebasis video yang mampu mengenali wajah secara akurat.
2. Menghasilkan laporan performa sistem pengenalan wajah sebagai referensi studi pengenalan wajah selanjutnya.
3. Sistem pengenalan wajah yang dihasilkan dapat digunakan untuk mencatat orang – orang yang memasuki ruangan, menghitung jumlah kunjungan ruangan, atau memberikan notifikasi ketika seseorang memasuki ruangan.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Pada tahap ini dilakukan pencarian referensi dan studi mendalam metode – metode yang akan digunakan untuk pembangunan sistem pengenalan wajah. Referensi yang digunakan dapat berupa artikel yang dipublikasikan pada jurnal atau informasi - informasi yang didapat melalui internet.
2. Analisis dan Perancangan Perangkat Lunak

   Pada tahap ini dilakukan analisis dan perancangan perangkat lunak. Tahap analisis merupakan penentuan kebutuhan dan ruang lingkup sistem yang akan dibangun. Tahap perancangan merupakan pendefinisian sistem, penentuan data yang digunakan dan proses – proses yang dilakukan sistem.
3. Implementasi Perangkat Lunak

   Pada tahap ini dilakukan implementasi sistem yang telah dirancang. Sistem pengenalan wajah menggunakan bahasa pemrograman Python dan IDE Spyder. Kamera diletakkan menghadap pintu masuk untuk mendapatkan data citra. Kemudian, sistem akan memberikan hasil pengenalan wajah dari data citra tersebut.
4. Uji Coba dan Evaluasi

Pada tahap ini dilakukan pengujian sistem yang telah dibangun. Pengujian meliputi uji coba metode dan data yang digunakan pada sistem. Pengujian metode berupa komparasi sistem berbasis video dengan tidak berbasis video. Pengujian data berupa komparasi pengaruh penggunaan skema data yang berbeda terhadap hasil klasifikasi.

1.7 Sistematika Laporan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian sebagai berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini berisi tentang analisis dan perancangan sistem yang akan dibangun.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

BAB II
DASAR TEORI

Pada bab ini diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan.

2.1 Video Digital

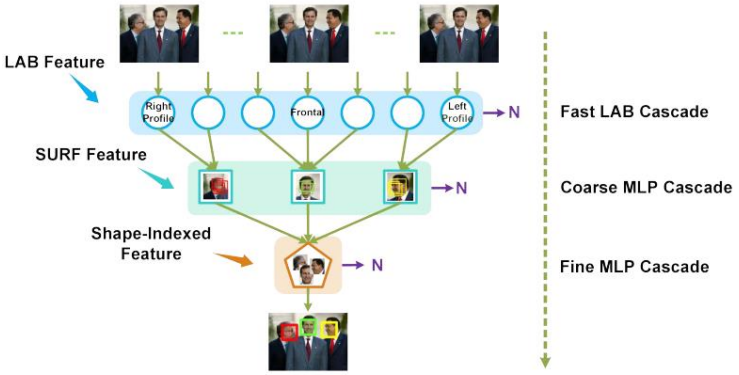
Video digital merupakan representasi elektronik dari citra visual yang bergerak [1]. Video digital terdiri dari citra - citra digital yang ditampilkan bergantian dengan kecepatan tertentu. Dalam konteks video, citra ini disebut dengan *frame*. Kecepatan *frame* yang ditampilkan diukur dengan satuan *frames per second* (fps). Sama halnya dengan citra digital, *frame* pada video memiliki satuan ukur *pixels* (*picture element*). Jika suatu *frame* memiliki lebar W *pixels* dan tinggi H *pixels*, maka ukuran *frame* tersebut adalah WxH *pixels*. Sementara, pixel merupakan suatu nilai yang merepresentasikan warna.

2.2 Seeta Face Detection

Seeta Face Detection merupakan algoritma deteksi wajah yang dirancang untuk deteksi wajah dalam berbagai sudut pandang (multi-view) pada suatu citra [4]. Pada *paper*-nya yang berjudul "*Funnel-Structured Cascade for Multi-View Face Detection with Alignment-Awareness*", Wu et al. menjelaskan bahwa seeta face detection merupakan algoritma yang dirancang khusus untuk proses deteksi wajah multi-view dengan akurasi tinggi dan cepat.

Seeta Face Detection menggunakan sebuah *funnel-structered cascade* (FuSt) *multi-view face detector* untuk melakukan proses deteksi. Perhatikan **Gambar 2.1**. Fust Detector memiliki struktur yang lebih luas pada bagian atas dan semakin sempit pada bagian bawah. Struktur ini berguna untuk menyaring kandidat window daerah wajah. Pada tahap pertama, beberapa classifier berjalan secara parallel untuk mengurangi kandidat window non-wajah. Classifier ini bekerja dengan sangat cepat

namun masih secara kasar, yang berarti beberapa kandidat window non-wajah masih mungkin lolos dari proses ini. Pada tahap berikutnya, classifier bekerja dengan waktu yang lebih banyak namun dengan kemampuan diskriminatif yang lebih tinggi untuk memverifikasi kandidat window yang tersisa. Dengan mengumpulkan kandidat window yang bertahan dari tahap sebelumnya, sebuah *unified multilayer perceptron* (MLP) *cascade* diterapkan sehingga memberikan hasil deteksi wajah.



Gambar 2.1 Funnel-Structured Cascade Multi-View Detector [4]

FuSt Detector terdiri dari tiga tahap, yakni Fast LAB Cacade Classifier, Coarse MLP Cascade Classifier, dan Fine MLP Cascade Classifier. Seiring berjalannya tahap akan semakin tinggi akurasi dan tingkat komputasinya. Sebuah citra input di-*scan* dengan window yang bergeser menyapu keseluruhan citra input, kemudian setiap window akan melalui detector tahap demi tahap. Fast LAB Cascade bertujuan menghilangkan window non-wajah dengan cepat. Coarse MLP Cascade Classifier menyeleksi kembali kandidat window dengan *cost* yang tidak terlalu tinggi. Terakhir, Unified Fine MLP Cascade Classifier menetukan wajah secara akurat.

2.2.1 Fast LAB Cascade

Untuk proses deteksi wajah yang *real-time*, perhatian utama penggunaan skema pergeseran window adalah jumlah yang besar dari kandidat window. Sebagai contoh, untuk mendeteksi wajah dengan ukuran lebih besar dari 20 x 20 piksel pada citra berukuran 640 x 480 piksel, lebih dari satu juta kandidat window yang akan diproses. Oleh karena itu, sangat perlu untuk mengusulkan sedikit kandidat window yang mungkin terdapat wajah dengan *cost* yang rendah.

Pengusulan daerah wajah dengan cepat dapat menggunakan *boosted cascade classifier* yang sangat efisien untuk proses deteksi yang digunakan oleh Viola dan Jones [2]. Sementara, penggunaan fitur LAB (Locally Assembled Binary) oleh Yan et al.[5] yang hanya memperhatikan hubungan antar fitur Haar dan penggunaan look-up table dapat meningkatkan efisiensi proses deteksi. Tahap Fast LAB Cascade merupakan penggunaan fitur LAB tesebut dengan bantuan *cascade calassifier*.

2.2.2 Coarse MLP Cascade

Setelah tahap LAB Cascade, sebagian besar window non-wajah telah dihilangkan dan window yang tersisa terlalu sulit untuk ditangani oleh LAB Cascade. Oleh karena itu, pada tahap ini, kandidat window ditangani lebih lanjut oleh classifier yang lebih canggih, yakni MLP (Multilayer Perceptron) dengan fitur SURF (Speeded-up Roboust Feature). Untuk menghindari *cost* komputasi yang terlalu tinggi, jaringan yang kecil digunakan untuk performa yang lebih baik namun masih secara kasar.

Fitur SURF lebih ekspresif dibandingkan dengan fitur LAB, namun masih efisien secara komputasi, akibat penggunaan teknik Integral Image [4]. Oleh karena itu, window wajah akan dapat dibedakan dengan window non-wajah dengan *cost* waktu yang rendah. Terlebih lagi, MLP digunakan dengan fitur SURF untuk klasifikasi window, yang dapat memodelkan variasi non-linier multi-view wajah dan membedakan non-wajah, dengan fungsi aktivasi non-linier yang dipakai.

MLP adalah sebuah tipe Neural Network yang terdiri dari input layer, output layer, dan satu atau lebih hidden layer diantara input layer dan output layer. Sebuah n-layer MLP F diformulasikan

$$F(x) = f_{n-1}(f_{n-2}(f_{n-3}(\dots f_1(x)))) \quad (2.1)$$

$$f_i(z) = \sigma(W_i z + b_i) \quad (2.2)$$

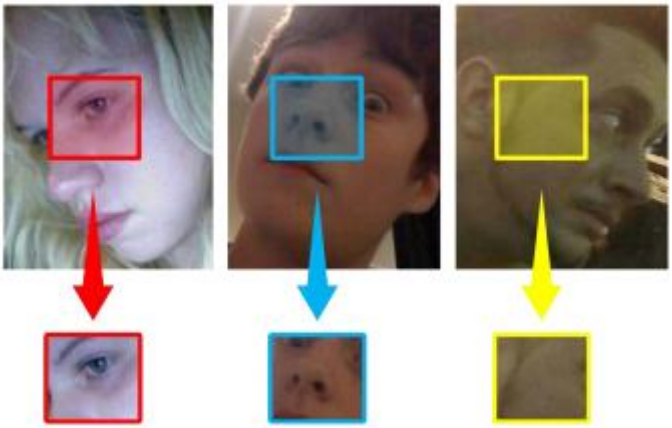
Dimana *x* adalah input, dalam hal ini merupakan fitur SURF dari kandidat window; *Wi* dan *bi* merupakan bobot dan bias pada koneksi layer *i* ke layer *i+1*. Fungsi aktivasi *σ(x)* didesain menggunakan fungsi nonlinier sigmoid $\sigma(x) = 1/1+e^{-1}$. Fungsi aktivasi tersebut digunakan pada seluruh layer; hidden, input dan output layer; agar memodelkan variasi nonlinier yang tinggi. Fungsi objektif digunakan dengan meminimalkan *mean square error* antara hasil prediksi dengan label benar seperti berikut

$$min \sum_{i=1}^{n} \|(F(x_i) - y_i)\|^2 \quad (2.3)$$

dimana x_i adalah vektor fitur ke- i dari sampel training dan y_i merupakan labelnya dengan nilai 1 atau 0 merepresentasikan wajah atau non-wajah. Problem pada fungsi objektif dapat diselesaikan dengan *back propagation framework* [6]. Multiple MLP digunakan untuk membangun struktur *cascade*, yang mana jumlah fitur dan ukuran jaringan bertambah secara bertahap. Fitur SURF yang digunakan pada tiap tahap, dipilih menggunakan *group sparse* [7]. Satu MPL Cascade dapat terkoneksi ke beberapa LAB Cascade Classifier.

2.2.3 Fine MLP Cascade with Shape-Indexed Feature

Untuk window yang bertahan dari tahap sebelumnya, kali ini akan lebih sulit membedakan wajah dan non-wajah. Penggunaan multiple model yang dijalankan secara parallel cenderung menemui kesalahan deteksi. Oleh karena itu, pada tahap ini, window yang tersisa akan diproses dengan secara cara satu kesatuan, menggunakan satu MLP cascade.



Gambar 2.2 Daerah Tengah pada Frontal Face Merupakan Hidung, Daerah Tengah pada Profile Face Merupakan Pipi [4]

Gambar 2.3 Empat Posisi Semantik yang Digunakan untuk Mendapatkan *Shape-Indexed Feature* : Mata Kiri dan Kanan, Hidung, dan Mulut [4]

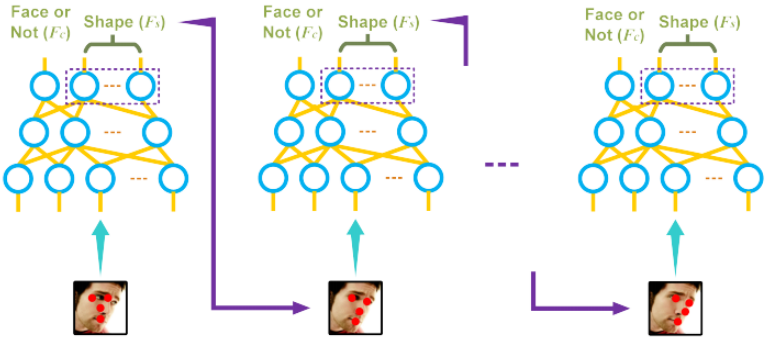
Perbedaan menonjol terdapat pada penampakan multi-view wajah, yang pada intinya disebabkan oleh fitur yang belum diluruskan. Contohnya daerah tengah pada wajah tampak depan (*frontal face*) meruapkan hidung, sementara daerah tengah wajah tampak samping (*profile face*) merupakan daerah pipi. Perhatikan **Gambar 2.2.** Untuk menangani masalah ini, tahap ini menggunakan *shape-indexed feature* yang diekstrak pada posisi – posisi yang konsisten secara semantik, yakni mata kiri dan kanan, hidung dan

mulut sebagai input Fine MLP Cascade. Perhatikan **Gambar 2.3**. Kemudian, fitur SIFT (Scale-Invariant Feature Transform) dihitung pada setiap posisi semantik tadi untuk setiap kandidat window sehingga menghasilkan fitur yang tangguh dalam menghadapi pengaruh pose dan translasi.

Berbeda dengan struktur MLP sebelumnya, MLP pada tahap ini digunakan tidak hanya untuk memprediksi wajah atau non-wajah, namun juga memprediksi *shape* secara bersamaan. Tambahan persamaan diterapkan pada fungsi fungsi objektif berikut

$$\min \sum_{i=1}^{n} \left\| F_c\left(\phi(x_i, \hat{s}_i)\right) - y_i \right\|^2 + \lambda \sum_{i=1}^{n} \left\| F_s\left(\phi(x_i, \hat{s}_i)\right) - s_i \right\|^2 \quad (2.4)$$

Dimana F_c merupakan output klasifikasi wajah dan F_s merupakan output prediksi bentuk; $\phi(x_i, \hat{s}_i)$ merupakan *shape-indexed feature* (Fitur SIFT) yang diekstrak dari sample training x_i berdasarkan *shape* yang diprediksi $\hat{s}_i$ dan s_i merupakan *groundtruth* suatu *shape*, λ merupakan pembobot untuk menjaga keseimbangan antara dua tipe error tersebut, yang merupakan *1/d* dengan *d* adalah dimensi dari *shape*. Alur proses Fine MLP Cascade ditunjukkan pada **Gambar 2.4**.



Gambar 2.4 Fine MLP Cascade

2.3 Kalman Filter

Kalman filter merupakan algoritma yang digunakan untuk memprediksi keadaan (*state*) berikutnya berdasarkan informasi keadaan (*state*) saat ini. Untuk dapat melakukan hal tersebut, kalman filter menggunakan model matematis dari sistem yang sedang diamati di tengah – tengah hasil pengukuran *realtime* yang penuh *error* [8]. Penggunaan model matematis pada kalman filter membuat metode ini menjadi sangat toleran terhadap kesalahan hasil pengukuran yang diperoleh dari alat ukur. Hal ini disebabkan oleh hasil pengukuran alat ukur bukan satu – satu nya hal yang diperhatikan oleh kalman filter untuk memberikan prediksi, melainkan juga memperhatikan kemungkinan yang dihitung dari model matematis sistem.

Kalman filter bekerja dengan 3 tahap, tahap prediksi, tahap observasi, dan tahap *update*. Pada tahap prediksi, Kalman Filter membuat memprediksi keadaan masa depan. Kemudian pada tahap observasi, kalman filter mengamati hasil pengukuran dari alat ukur. Setelah itu pada tahap *update*, kalman filter menimbang perbedaan hasil tahap prediksi dan tahap observasi, dan kemudian memperbaiki prediksi berdasarkan hasil pertimbangan tadi. Tahap – tahap diatas dilakukan berulang kali setiap akan melakukan prediksi. Perhatikan **Gambar 2.5**.

Berikut persaman tahap prediksi.

Persamaan *state prediction* (memprediksi keadaan selanjutnya).

$$x_{predicted} = Ax_{n-1} + Bu_n \tag{2.5}$$

Persamaan *covariance prediction* (memprediksi perkiraan error).

$$P_{predicted} = AP_{n-1}A^T + Q \tag{2.6}$$

Berikut persamaan tahap observasi

Persamaan *innovation* (membandingkan hasil pengukuran dengan prediksi).

$$y = z_n - Hx_{predicted} \tag{2.7}$$

Persamaan *innovation covariance* (membandingkan perkiraan error hasil pengukuran dengan prediksi).

$$S = HP_{predicted}H^T + R \quad (2.8)$$

Berikut ini persamaan tahap *update*.

Persamaan Kalman Gain (menimbang hasil prediksi dengan hasil pengukuran).

$$K = P_{predicted}H^T S^{-1} + R \quad (2.9)$$

Persamaan *state update* (hasil perkiraan yang baru).

$$x_n = x_{predicted} + Ky \quad (2.10)$$

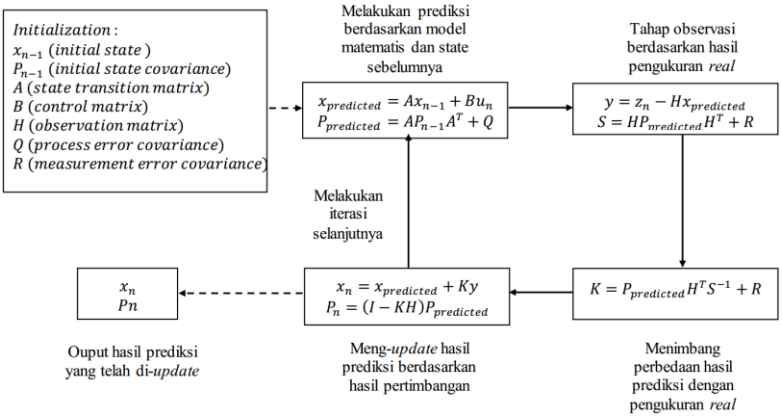
Persamaan *covariance update* (hasil perkiraan error yang baru).

$$P_n = (I - KH)P_{predicted} \quad (2.11)$$

Dimana,

- u_n : *Control vector*. Menentukan besarnya *user control* (pengaruh *user* terhadap sistem).
- z_n : *Measurement vector*. Merupakan vektor hasil pengukuran yang diperoleh setiap iterasi.
- x_n : Perkiraan *state* selanjutnya dari *state* saat ini.
- P_n : Perkiraan *error* untuk *state selanjutnya*.
- A : *State transition matrix*. Merupakan matriks transisi state saat ke ini ke state selanjutnya.
- B : *Control matrix*. Merupakan matriks yang digunakan untuk mendefinisikan persamaan linier faktor – faktor yang mempengaruhi *user control*.
- H : *Observation matrix*. Perkalian *state vector* dengan matrix ini menghasilkan *measurement vector*.
- Q : *Process error covariance*. Perkiraan *error* akibat proses (perpindahan dari *state* saat ini ke *state* selanjutnya).

R : *Measurement error covariance*. Perkiraan *error* akibat pengukuran.

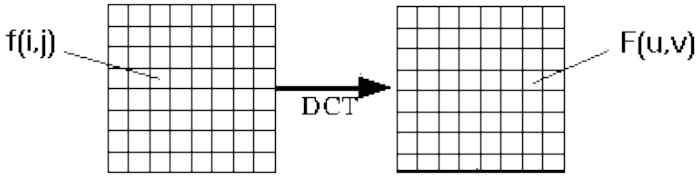


Gambar 2.5 Alur Kerja Kalman Filter

2.4 Discrete Cosine Transform

2.4.1 Pengertian Discrete Cosine Transform

Discrete cosine transform (DCT) merupakan algoritma encoding serupa dengan discrete fourier transform yang mengubah suatu sinyal atau citra dari domain spasial ke domain frekuensi [9]. Perhatikan **Gambar 2.6**.



Gambar 2.6 Discrete Cosine Transform merubah citra dari domain spatial ke domain frekuensi [9]

Persamaan umum untuk DCT 2D (citra *N x M*) didefinisikan dengan persamaan berikut.

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \alpha_i \alpha_j cos\left[\frac{\pi . u}{2.N}(2i+1)\right] cos\left[\frac{\pi . v}{2.M}(2j+1)\right] . f(i,j) \quad (2.12)$$

Dimana,

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{2}}, & p = 0 \\ 1, & otherwise \end{cases}$$

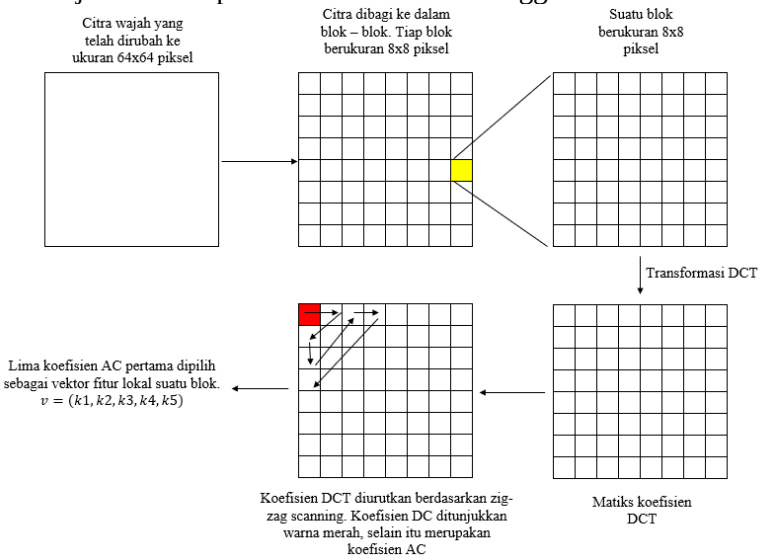
- Citra input berukuran *N x M*
- *f(i,j)* adalah intensitas pixel baris *i* kolom *j*
- *F(u,v)* merupakan hasil transformasi berupa matriks koefisien DCT

2.4.2 Discrete Cosine Transform sebagai Metode Ekstraksi Fitur

DCT dapat digunakan sebagai metode ekstraksi fitur yang memperhatikan penampakan lokal [10]. Penggunaan DCT sebagai metode ekstraksi fitur memberikan keuntungan dengan menyediakan informasi frekuensi untuk menangani penampakan wajah. Sebagai contoh, beberapa rentang frekuensi berguna untuk menangani variasi pencahayaan. Terlebih lagi, pada [10] diketahui bahwa DCT merepresentasikan fitur lokal lebih baik dari transformasi Karhunen-Loeve, Fourier, Wavelet, dan Wash-Hadamard dalam kasus pengenalan wajah.

Untuk melakukan ekstraksi fitur menggunakan DCT, pertama citra wajah diubah ke dalam ukuran 64x64 piksel. Citra wajah yang telah diubah dibagi ke dalam blok yang non-*overlap* berukuran 8x8 piksel. Lalu, DCT diterapkan pada setiap blok sehingga diperoleh matriks koefisien DCT. Koefisien DCT yang didapat diurutkan berdasarkan *zig-zag scanning*. Dari koefisien DCT yang telah diurutkan, lima koefisien *AC* pertama dipilih sebagai vektor fitur lokal untuk masing – masing blok. Koeffisien

DC diabaikan untuk normalisasi pencahayaan [11]. Koefisien DC (*direct current*) merupakan nilai yang merepresentasikan intensitas citra secara umum dan bukan merepresentasikan dalam gelombang cosinus, sementara koefisien *AC* (*alternating current*) merupakan nilai yang menentukan besarnya pengaruh suatu frekuensi gelombang cosinus [12]. Terakhir, vektor fitur lokal dari tiap blok digabungkan menjadi vektor fitur keseluruhan. **Gambar 2.7** menunjukkan alur proses ekstraksi fitur menggunakan DCT.



Gambar 2.7 Alur Proses Ekstraksi Fitur dengan DCT

2.5 K-Nearest Neighbour

Metode klasifikasi *k-Nearest Neighbour* (*kNN*) merupakan salah satu metode klasifikasi yang paling sederhana dalam *machine learning* [13]. Pada dasarnya, *kNN* melakukan klasifikasi dengan mencari data yang paling mirip pada data training dan membuat prediksi berdasarkan data yang paling mirip tersebut. Meskipun *kNN* merupakan metode yang sederhana dan mudah diimplementasikan, metode ini memiliki ranah aplikasi yang luas,

seperti sistem rekomentasi, pencarian semantik, dan deteksi anomali.

Untuk menggunakan *kNN* pertama kali, kita memerlukan sebuah vektor fitur yang merepresentasikan data kita. Vektor fitur merupakan representasi matematis dari suatu data dan karena karakteristik data yang kita miliki tidak dapat diturunkan secara langsung, kita perlu melakukan *preprocessing* dan ekstraksi fitur untuk memperoleh vektor fitur ini.

kNN merupakan metode yang bersifat *lazy learning*, yang berarti *kNN* tidak memiliki fase training sebelum proses klasifikasi. Sebagai gantinya, proses abstraksi data (training) dilakukan sebelum proses klasifikasi. Hal ini berarti kita dapat langsung melakukan proses klasifikasi ketika data training telah siap. Namun, hal ini juga menyebabkan kita harus menyimpan keseluruhann data training ke dalam memory saat akan melakukan klasifikasi. Untuk alasan ini, *kNN* akan bekerja dengan baik dataset kecil yang tidak memiliki banyak fitur.

Ketika kita telah mendapatkan tranining data-set, kita dapat memulai melakukan klasifikasi. Training data-set direpresentasikan dengan matriks berukuran *MxN* dimana *M* adalah jumlah data dan *N* merupakan panjang vektor fitur. Berikut ini meruapkan langkah – langkah proses klasifikasi *kNN* :

1. Hitung jarak antara data yang akan diklasifikasikan dengan setiap data pada training data-set
2. Ambil k data terdekat
3. Lakukan ‘major vote’ terhadap data yang telah diambil. Kelas yang paling dominan dijadikan sebagai hasil klasifikasi.

BAB III
ANALISIS DAN PERANCANGAN

Bab Analisis dan Perancangan berisi analisis kebutuhan dan perancangan sistem yang akan dibangun. Tahap analisis membahas analisis kebutuhan sistem, sementara tahap perancangan membahas rancangan sistem berdasarkan analisis yang dibuat.

3.1 Tahap Analisis

Tahap analisis mendefinisikan kebutuhan untuk membangun sistem pengenalan wajah untuk monitoring pintu.

3.1.1 Deskripsi Umum

Pada Tugas Akhir ini akan dibangun sistem pengenalan wajah untuk monitoring pintu. Data masukan merupakan video yang diperoleh dari kamera CCTV. Kamera tersebut diletakkan menghadap pintu masuk ruangan Laboratorium Komputasi Cerdas dan Visi, Departemen Informatika, ITS. Data keluaran berupa hasil pengenalan wajah seseorang yang memasuki ruangan.

Sistem ini diharapkan dapat digunakan untuk meningkatkan keamanan Laboratorium dan membantu melakukan pencatatan pengunjung Laboratorium secara otomatis.

3.1.2 Spesifikasi Kebutuhan Sistem

Berikut ini merupakan beberapa proses yang perlu dilakukan untuk menghasilkan pengenalan wajah seseorang melalui video kamera CCTV.

1. *Face Detection*

   Face Detection dilakukan untuk mendapatkan area wajah dari suatu citra. Citra yang diperoleh dari video kamera CCTV bukan hanya mengandung penampakkan wajah saja, namun juga penampakan lingkungan di dalam ruangan. Penampakan linkungan sekitar tersebut bukan merupakan objek yang ingin diamati sehingga kita perlu memisahkannya agar pengamatan dapat berfokus hanya pada area wajah.

2. *Face Tracking*
   Face Tracking merupakan proses mendapatkan posisi wajah pada citra tanpa menggunakan proses *face detection*. Ketika sedang memasuki ruangan, wajah seseorang mungkin berada pada posisi yang tidak mampu dideteksi, seperti jika wajah seseorang terhalang oleh objek atau orang tersebut memalingkan wajahnya dari kamera. Sementara, sistem perlu mengetahui posisi wajah untuk menentukan apakah orang tersebut masih berada di dalam *frame* atau tidak. Untuk menangani masalah tersebut, proses *face tracking* dilakukan untuk memperkirakan posisi wajah saat ini berdasarkan informasi posisi wajah terakhir yang berhasil terdeteksi.
3. Ekstraksi Fitur
   Ekstraksi fitur dilakukan untuk mendapatkan fitur representatif wajah seseorang. Penggunaan fitur yang representatif dapat meningkatkan performa klasifikasi. Selain itu, penggunaan fitur representatif dapat mengurangi beban komputasi karena pengolahan hanya berfokus pada fitur representatif tersebut.
4. Klasifikasi
   Klasifikasi dilakukan untuk mengidentifikasi wajah seseorang. Proses ini merupakan proses utama untuk mengenali wajah seseorang.

3.1.3 Analisis Permasalahan

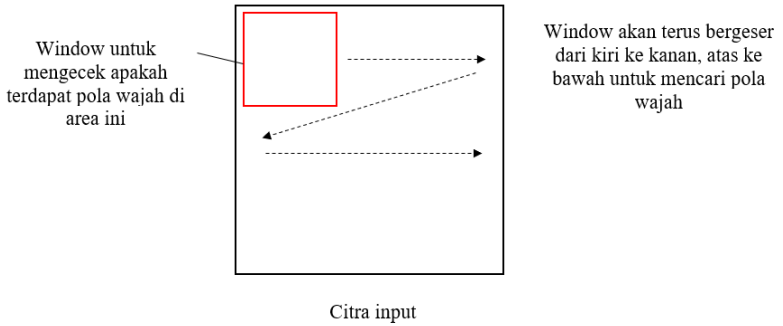
Bagian ini membahas permasalahan yang mungkin terjadi ketika melakukan implementasi berdasarkan spesifikasi kebutuhan.

3.1.3.1 Analisis Permasalahan *Face Detection*

Pada Tugas Akhir ini digunakan metode Seeta Face Detection. Metode ini bekerja dengan cara memeriksa adanya wajah di setiap area citra.

Perhatikan **Gambar 3.1**. Kotak hitam merupakan citra input yang akan dicari area wajah. Metode Seeta Face Detection menggunakan suatu *window* (ditunjukkan dengan kotak merah) untuk mengecek apakah terdapat pola wajah di daerah tersebut. *Window* tersebut akan terus bergeser dari kiri ke kanan kemudian

ke bawah untuk mencari pola wajah. *Window* tersebut juga dapat berubah ukuran, diperbesar atau diperkecil.



Gambar 3.1 Analisis Deteksi Wajah

Permasalahan yang terjadi adalah semakin besar perbandingan ukuran citra input dengan ukuran wajah, maka semakin banyak pula pergeseran *window* dan pengecekkan pola wajah. Hal ini menyebabkan beban komputasi semakin besar seiring dengan besarnya perbandingan ukuran citra input dengan ukuran wajah.

Gambar 3.2 Wajah Seseorang Terhalang Objek Ketika Memasuki Ruangan

Permasalahan berikutnya adalah proses *face detection* dapat mengalami kegagalan meskipun wajah seseorang masih terdapat pada frame. Hal ini dapat disebabkan oleh adanya penghalang pada wajah seseorang tersebut atau orang tersebut memalingkan wajah dari kamera CCTV. Padahal, posisi wajah seseorang pada frame merupakan informasi penting untuk mengetahui apakah seseorang telah selesai memasuki ruangan atau masih sedang dalam proses. Perhatikan **Gambar 3.2.** Pada gambar tersebut terlihat bahwa sesoerang sedang memasuki ruangan namun dengan tangan menutupi hampir separuh wajah. Akibatnya, *face detector* tidak mampu mendeteksi adanya wajah pada frame tersebut.

3.1.3.2 Analisis Permasalahan Ekstraksi Fitur

Citra wajah yang dihasilkan melalui kamera CCTV sangat bervariasi terhadap pencahayaan. Maka, diperlukan metode ekstraksi fitur yang mampu mendapatkan karakteristik wajah seseorang tanpa dipengaruhi kondisi pencahayaan.

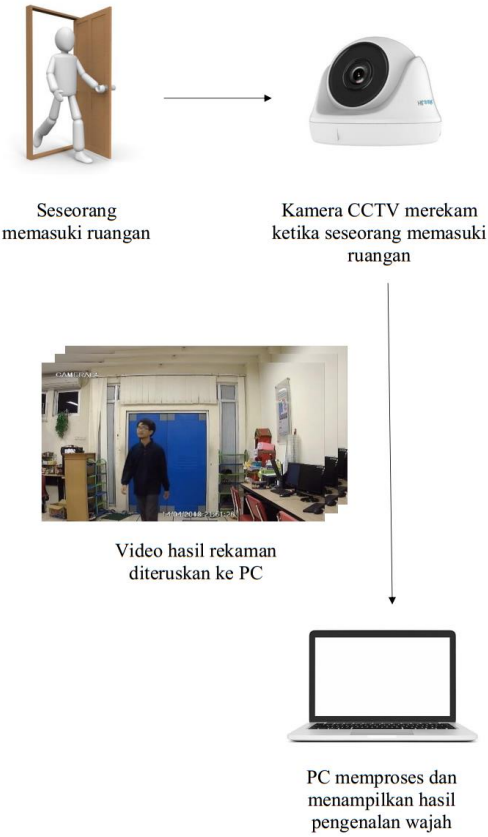
3.2 Tahap Perancangan

Tahap perancangan dilakukan untuk merancang seluruh proses berdasarkan analisis dan spesifikasi kebutuhan sistem pengenalan wajah untuk monitoring pintu.

3.2.1 Perancangan Sistem

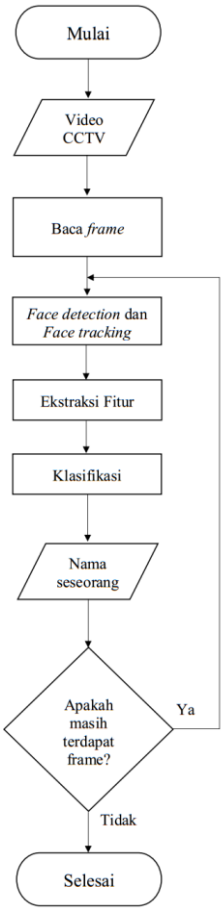
Perancangan sistem dilakukan untuk menggambarkan proses secara keseluruhan pada sistem pengenalan wajah berbasis video untuk monitoring pintu. Untuk merekam seseorang ketika memasuki ruangan, kamera CCTV diletakkan menghadap pintu masuk Laboratorium. Video hasil rekaman kemudian diteruskan ke PC untuk dilakukan pemrosesan. Hasil pengenalan wajah kemudian ditampilkan di layar. Ilustrasi sistem pengenalan wajah ditunjukkan pada **Gambar 3.3**.

Sistem pengenalan wajah memiliki tiga proses utama. Proses pertama merupakan proses *face detection* dan *face tracking*. Proses kedua merupakan ekstraksi fitur. Dan proses ketiga merupakan klasifikasi. **Gambar 3.4** menunjukkan diagram alir proses pengenalan wajah.



Gambar 3.3 Ilustrasi Sistem Pengenalan Wajah

Proses *face detection* dan *face tracking* merupakan proses yang pertama kali dilakukan setelah membaca *frame* (citra) dari video. Sistem akan mendeteksi apakah terdapat wajah seseorang pada citra menggunakan algoritma Seeta Face Detection. Apabila tidak terdeteksi adanya wajah, sistem akan memperkirakan posisi wajah menggunakan algoritma *face tracking* kalman filter. Proses ini akan mengolah citra masukan (frame) menjadi citra wajah.



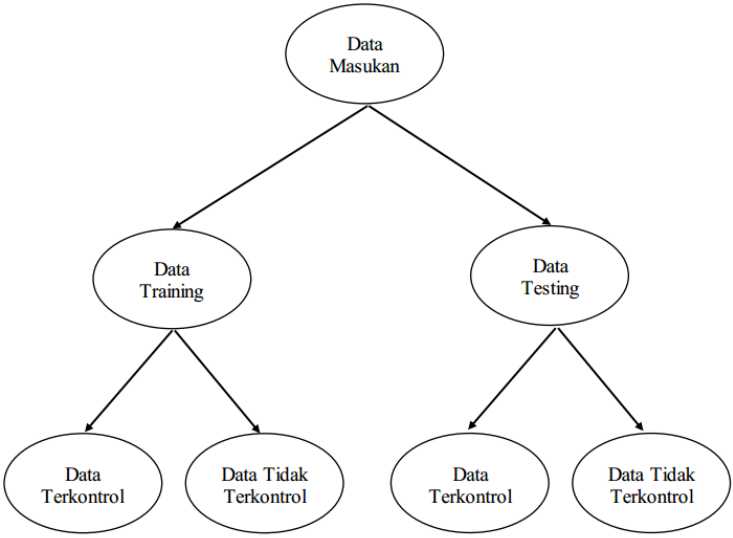
Gambar 3.4 Perancangan Sistem Pengenalan Wajah

Ekstraksi fitur adalah proses untuk mendapatkan fitur representatif wajah. Pada tugas akhir ini digunakan metode ekstraksi fitur berbasis lokal discrete cosine transform (DCT). Proses ini akan mengolah citra wajah menjadi vektor fitur yang akan digunakan pada proses klasifikasi.

Proses klasifikasi adalah proses menentukan wajah seseorang. Proses klasfikasi menggunakan algoritma *k-Nearest Neighbour* (*kNN*). Atribut *kNN* yang digunakan merupkan vektor fitur yang dihasilkan dari proses ekstraksi fitur. Vektor fitur tersebut kemudian akan diolah menjadi hasil pengenalan wajah seseorang.

3.2.2 Perancangan Data

Perancangan data dilakukan untuk menentukan data masukan dan data keluaran sesuai dengan kebutuhan sistem. Data masukan adalah data yang diperlukan sistem untuk dapat menjalankan fungsinya. Data keluaran adalah data yang dihasilkan sistem ketika atau telah selesai menjalankan fungsinya.



Gambar 3.5 Data Masukan

Data masukan adalah data yang diproses oleh sistem pengenalan wajah. Data yang digunakan sebagai data masukan

adalah video yang diperoleh dari kamera CCTV. Kemudian, citra frame diambil dari video tersebut untuk diproses sistem.

Data masukan terbagi menjadi dua jenis, yakni data *traning* dan data *testing*. Data *traning* adalah data masukan yang digunakan untuk membangun model pada klasifier, yang dalam tugas akhir ini menggunakan *kNN*. Model merupakan representasi pengetahuan yang akan digunakan klasifier dalam melakukan klasifikasi nantinya. Semakin representatif model maka semakin baik performa klasifikasi. Data *testing* adalah data masukan yang digunakan untuk melakukan pengujian sejauh mana klasifier melakukan klasifikasi dengan benar. Oleh karena itu, data *testing* tidak dapat bertindak sebagai data *training* sekaligus.

Data training dan data testing dibagi berdasarkan cara pengumpulan menjadi dua jenis, data terkontrol dan data tidak terkontrol. Perhatikan **Gambar 3.5**. Data terkontrol merupakan data yang dikumpulkan dengan memastikan sample data melakukan hal – hal tertentu. Data *training* terkontrol dikumpulkan dengan cara tiap sample direkam berdiri di depan pintu ruangan bagian dalam dan memperlihatkan wajah menghadap lurus ke kamera, sedang menengok ke kanan, sedang menengok ke kiri, sedang menengadah, dan sedang menunduk. Hal ini bertujuan agar dapat memperoleh seluruh fitur wajah. Data *testing* terkontrol dikumpulkan dengan cara tiap sample direkam ketika memasuki ruangan dan memastikan wajah melihat langsung ke arah kamera. Hal ini bertujuan agar memastikan wajah orang tersebut mampu dideteksi ketika proses *face detection*. Data tidak terkontrol merupakan data yang dikumpulkan tanpa memperhatikan hal – hal khusus. Pengumpulan data tidak terkontrol dilakukan dengan cara merekam sample memasuki ruangan pada aktivitas sehari - hari. Hal ini bertujuan untuk mendapatkan representasi data yang benar – benar terjadi di lapangan.

Data *training* dan data *testing* pada tugas akhir ini memiliki 8 kelas. Kelas pada data merupakan nama orang. Data *tranining* berjumlah 18 video. Satu video hanya berisi satu orang yang sedang memasuki ruangan. Terdiri dari 8 data terkontrol dan 10

data tidak terkontrol. Sementara, data *testing* berjumlah 13 video. Terdiri dari 8 data terkontrol dan 5 data tidak terkontrol. Perhatikan **Tabel 3.1** dan **Tabel 3.2**.

Tabel 3.1 Data *Training*

No. Video	Nama	Durasi (Detik)	Jumlah Frame	Jenis
1	Chasni	8	257	Terkontrol
2	Dandi	11	332	Terkontrol
3	Fadli	10	324	Terkontrol
4	Galang	14	437	Terkontrol
5	Mala	19	586	Terkontrol
6	Nuzul	9	270	Terkontrol
7	Ocid	16	485	Terkontrol
8	Randi	8	262	Terkontrol
9	Chasni	7	228	Tidak Terkontrol
10	Chasni	4	136	Tidak Terkontrol
11	Dandi	15	462	Tidak Terkontrol
12	Dandi	9	286	Tidak Terkontrol
13	Nuzul	9	289	Tidak Terkontrol
14	Nuzul	5	151	Tidak Terkontrol
15	Nuzul	10	316	Tidak Terkontrol
16	Randi	13	403	Tidak Terkontrol
17	Randi	9	286	Tidak Terkontrol
18	Randi	5	159	Tidak Terkontrol

Tabel 3.2 Data *Testing*

No. Video	Nama	Durasi (Detik)	Jumlah Frame	Jenis
1	Chasni	5	179	Terkontrol
2	Dandi	9	296	Terkontrol
3	Fadli	6	192	Terkontrol
4	Galang	7	233	Terkontrol

5	Mala	6	189	Terkontrol
6	Nuzul	6	189	Terkontrol
7	Ocid	9	297	Terkontrol
8	Randi	8	243	Terkontrol
9	Chasni	6	190	Tidak Terkontrol
10	Dandi	11	336	Tidak Terkontrol
11	Dandi	8	253	Tidak Terkontrol
12	Nuzul	10	307	Tidak Terkontrol
13	Randi	19	583	Tidak Terkontrol

Data keluaran sistem pengenalan wajah merupakan hasil pengenalan wajah seseorang yang ditampikan setelah seseorang keluar dari jangkauan kamera ketika memasuki ruangan.

3.2.3 Perancangan Proses

Perancangan proses dilakukan untuk memberikan gambaran mengenai setiap proses yang terdapat pada sistem pengenalan wajah.

3.2.3.1 Perancangan Proses Face Detection dan Face Tracking

Gambar 3.6 Proses *Face Detection* Hanya Dilakukan pada *Region of Interest*, yakni *Bounding Box* Berwarna Putih

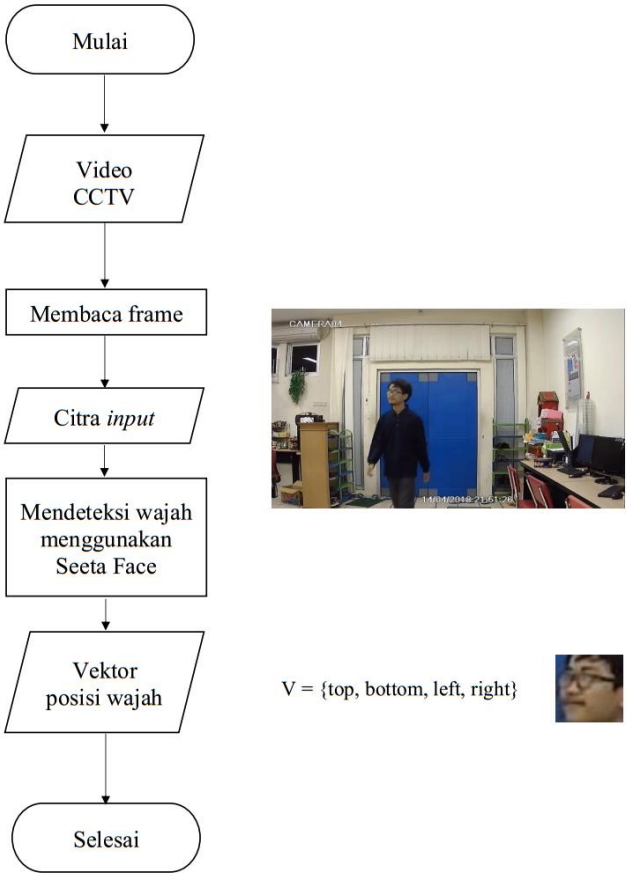
Proses *face detection* dan *face tracking* merupakan proses pertama setelah frame (citra) pada video dibaca. Pada proses ini akan dilakukan deteksi wajah menggunakan algoritma Seeta Face Detection. Untuk memperkirakan posisi wajah ketika tidak terdeksi wajah, digunakan algoritma face tracking kalman filter. Menunjukkan alur proses face detection dan gambar menunjukkan alur proses face tracking.

Proses *face detection* diawali dengan pembacaan citra *input* pada video. Citra input merupakan citra yang diperoleh dari hasil pembacaan frame video. Kemudian, algoritma Seeta Face diterapkan pada citra tersebut. Algoritma Seeta Face menghasilkan vektor berukuran 4 satuan yang menandakan batas atas, bawah, kiri, dan kana suatu wajah pada citra *input*. Proses deteksi wajah secara detail menggunakan Seeta Face Detection dijelaskan pada **BAB II DASAR TEORI.** Hasil dari proses deteksi menggunakan Seeta Face Detection merupakan sebuah vektor berukuran 4 satuan yang menunjukan batas atas, bawah, kiri, dan kanan *bounding box* daerah wajah. Vektor ini disebut vektor posisi wajah. Vektor posisi wajah ini dapat digunakan untuk memotong citra input sehingga dihasilkan citra wajah. Perhatikan **Gambar 3.7.**

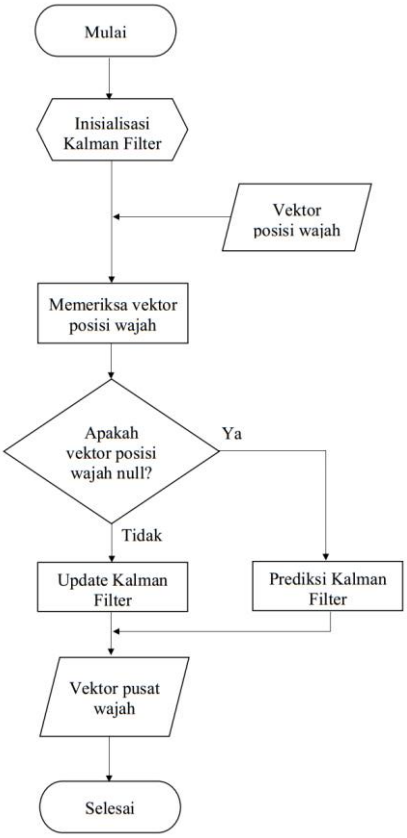
Untuk menangani permasalahan pertama pada **Analisis Permasalahan *Face Detection***, proses *face detection* tidak dilakukan terhadap keseluruhan frame video, melainkan hanya diterapkan pada *region of interest* dari citra input, dalam hal ini adalah daerah sekitar pintu masuk. Perhatikan **Gambar 3.6**. Dengan demikian, beban komputasi proses *face detection* dapat dikurangi.

Permasalahan berikutnya dari proses *face detection* adalah kemungkinan adanya kegagalan proses deteksi meskipun terdapat wajah pada frame. Untuk menangani hal ini, proses *face detection* akan selalu diiringi dengan proses *face tracking*, yang merupakan prediksi posisi wajah berdasarkan informasi posisi wajah pada frame sebelumnya. Proses ini menggunakan algoritma Kalman Filter. Jika pada suatu frame tidak terdapat wajah, maka Kalman Filter akan melakukan proses Prediksi Kalman Filter untuk

memprediksi vektor pusat wajah. Sementara, jika pada suatu frame terdapat wajah, maka Kalman Filter akan melakukan proses Update Kalman Filter untuk memperbarui vektor pusat wajah. Adanya wajah ditandai dengan vektor posisi wajah yang dihasilkan oleh proses *face detection* tidak *null*. Proses Prediksi Kalman Filter dan Update Kalman Filter secara detail dapat dilihat pada **BAB II DASAR TEORI**.



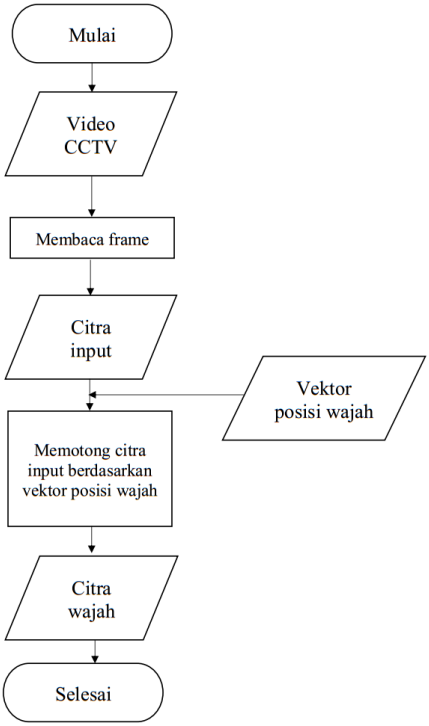
Gambar 3.7 Perancangan Proses *Face Detection*



Gambar 3.8 Perancangan Proses Face Tracking

Proses *face tracking* diawali dengan melakukan inisilisasi kalman filter. Beberapa hal yang perlu diinisialisasi adalah *initial state (x), initial covariance (P) , state transition matrix (A), observation matrix (H), process error covariance (Q)* dan *measurement error covariance (R)*. vektor state merupakan vektor berukuran 4 satuan menunjukkan posisi wajah pada sumbu x, posisi wajah pada sumbu y, kecepatan perpindahan terhadap

sumbu x, dan kecepatan perpindahan terhadap sumbu y. H merupakan matriks berukuran 2x4, R merupakan matrix berukuran 2x2, sementara P, A, Q merupakan matriks berukuran 4x4.

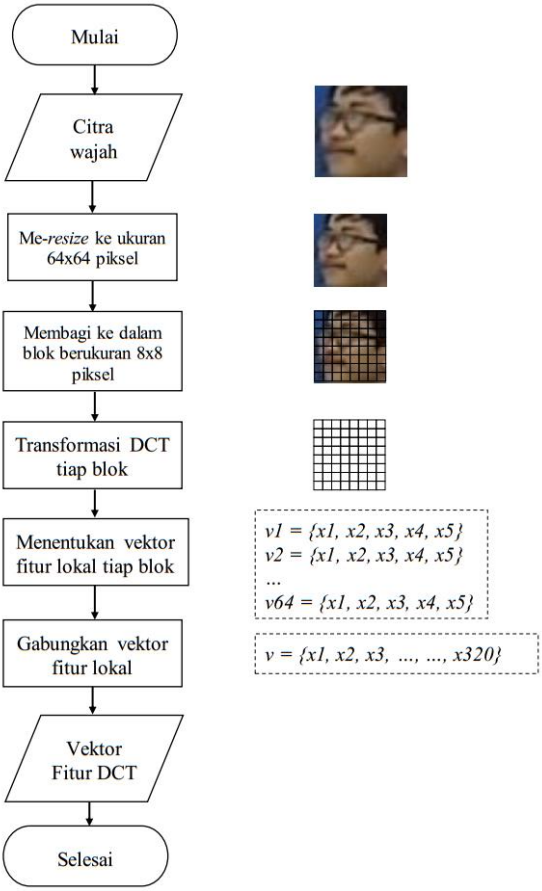


Gambar 3.9 Proses Pemotongan Citra Input Menjadi Citra Wajah

Data masukan merupakan vektor posisi wajah yang merupakan hasil dari proses *face detection*. Dengan melakukan pengecekan terhadap vektor posisi wajah, maka dapat diketahui apakah terdapat wajah pada frame. Proses Prediksi Kalman Filter dilakukan jika tidak terdeksi wajah dan Proses Update Kalman Filter dilakukan jika terdeteksi adanya wajah. Hasil proses Prediksi dan Update Kalman Filter merupakan vektor pusat wajah berukuran 2 satuan menunjukkan posisi x dan y pusat wajah. Dengan meggunakan vektor pusat wajah, kita dapat

memperkirakan posisi wajah meskipun proses *face detection* gagal mendeteksi wajah pada frame. Perhatikan **Gambar 3.8**.

3.2.3.2 Perancangan Proses Ekstraksi Fitur



Gambar 3.10 Diagram Alir Proses Ekstraksi Fitur

Untuk melakukan proses ekstraksi fitur, kita perlu mendapatkan citra yang hanya mengandung penampakan wajah saja. Citra ini disebut dengan citra wajah. Citra wajah diperoleh

dari pemotongan citra input berdasarkan vektor posisi wajah hasil proses *face detection*. Proses pemotongan citra input menjadi citra wajah ditunjukkan dengan **Gambar 3.9**.

Tugas akhir ini menggunakan metode ekstraksi fitur berbasis lokal, discrete cosine transform (DCT). Proses ini diawali dengan me-resize citra wajah ke ukuran 64x64 piksel. Citra yang telah di-resize kemudian dibagi ke dalam blok – blok non-*overlap* yang tiap bloknya berukuran 8x8 piksel. Tiap blok dilakukan transformasi DCT sehingga diperoleh matriks koefisien DCT. Matriks koefisien DCT diurutkan berdasarkan *zig-zag scanning*. Lima koefisien *AC* pertama dipilih sebagai vektor fitur lokal untuk blok tersebut, sementara koefisien DC diabaikan.

Vektor fitur keseluruhan merupakan gabungan dari seluruh vektor fitur lokal sehingga terbentuk vektor berukuran 320 satuan. **Gambar 3.10** menunjukkan diagram alir proses ekstraksi fitur.

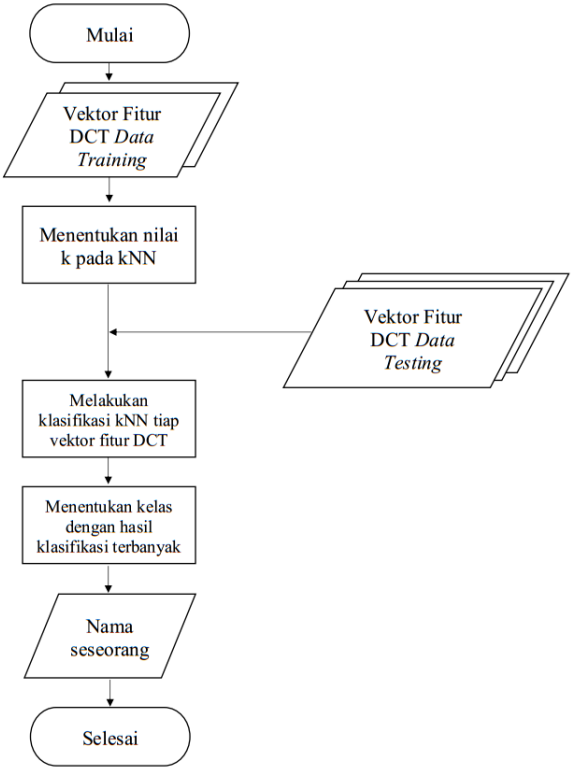
3.2.3.3 Perancangan Proses Klasifikasi

Proses klasifikasi pada tugas akhir ini menggunakan algoritma *k-Nearest Neighbour* (*kNN*). Pada proses ini dilakukan pengenalan wajah seseorang menggunakan fitur DCT yang dihasilkan oleh proses ekstraksi fitur sebagai atribut *kNN*. Sementara, kelas pada *kNN* merupakan nama seseorang. Proses ini akan menghasilkan identifikasi wajah seseorang berupa nama orang yang dikenali.

Langkah pertama pada proses ini adalah membuat model *kNN* atau disebut juga *training data*. Proses *training data* ini dilakukan dengan membaca Vektor Fitur DCT *Data Training*. Data training tersebut merupakan fitur DCT yang dihasilkan dari citra wajah yang telah melalui proses *face detection* dan ekstraksi fitur. Penentuan kelas pada data training ditentukan secara manual berdasarkan nama orang sebenarnya.

Setelah model *kNN* terbuat, maka *kNN* siap melakukan klasifikasi fitur – fitur DCT seseorang yang memasuki ruangan. Proses klasifikasi *kNN* menggunakan nilai $k = 5$ dan tanpa pembobotan. Kesimpulan akhir ditentukan merupakan kelas

dengan jumlah hasil klasifikasi terbanyak. **Gambar 3.11** menunjukkan diagram alir proses klasfikasi.



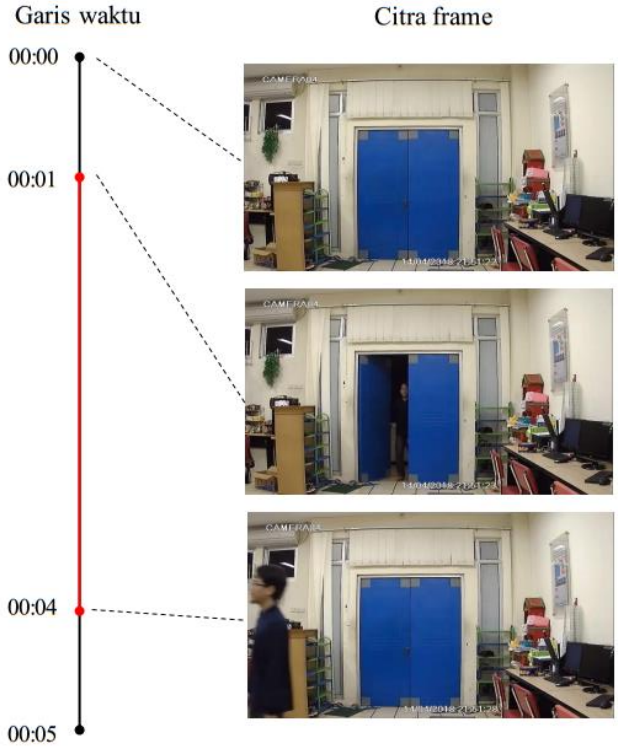
Gambar 3.11 Diagram Alir Proses Klasifikasi

3.2.3.4 Perancangan Proses Secara Keseluruhan

Perancangan proses secara keseluruhan merupakan integrasi dari ketiga proses sebelumnya. Sistem pengenalan wajah bekerja dengan membaca dan memproses video *frame by frame*. Proses membaca dan mengolah citra frame dilakukan berulang kali hingga tidak ada lagi frame tersisa pada video. Perhatikan **Gambar 3.4.** Ketika memproses suatu frame, memungkinkan melakukan proses *face detection* dan *face tracking*, ekstraksi fitur, atau klasifikasi.

Namun, tidak pada setiap frame dilakukan ketiga proses tersebut. Sebagai contoh, ketika citra frame yang dibaca tidak terdapat wajah, maka tidak memungkinkan dilakukan proses ekstraksi fitur dan klasifikasi. **Gambar 3.4** merupakan bentuk simplifikasi untuk memudahkan pemahaman.

Perhatikan **Gambar 3.12.** Gambar tersebut merupakan ilustrasi citra frame yang dibaca dari video. Garis waktu menunjukkan waktu jalannya video. Gambar disebelahnya merupakan citra frame yang dibaca pada waktu tertentu.



Gambar 3.12 Ilustrasi Citra yang Diambil dari Frame Video

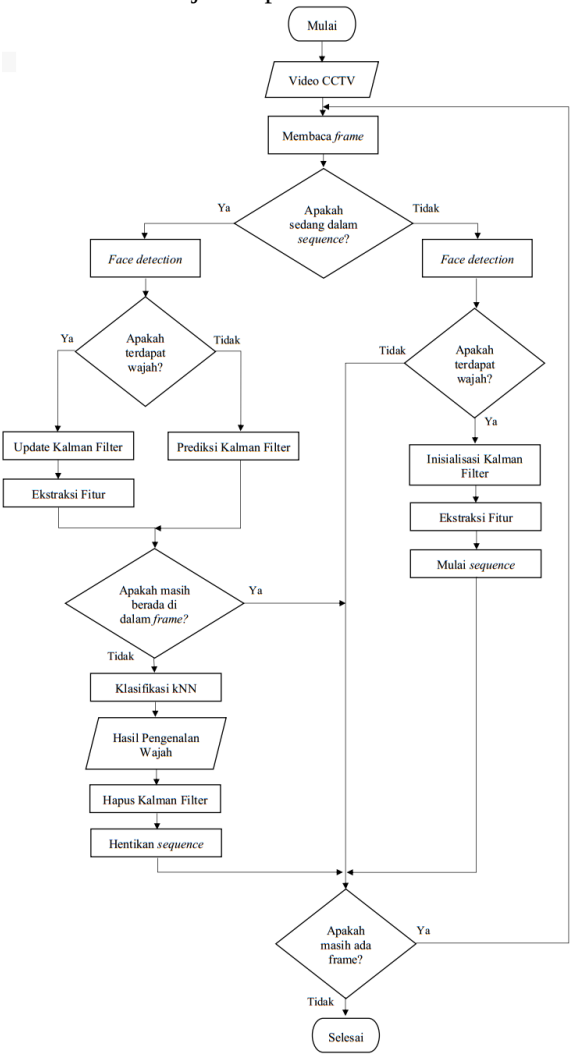
Pada waktu 00:00 sampai 00:01 merupakan waktu dimana tidak terdapat seseorang yang memasuki ruangan. Frame – frame yang dibaca pada rentang waktu ini tidak akan ditemukan adanya wajah. Oleh karena itu pemrosesan frame – frame ini hanya sampai proses *face detection* dan dilanjutkan ke iterasi frame berikutnya. Proses *face tracking* juga tidak diperlukan karena tidak terdapat wajah seseorang.

Pada waktu 00:01, merupakan saat – saat pertama seseorang memasuki ruangan. Frame yang diambil pada saat ini akan terdeksi adanya wajah pertama kali. Pada saat ini, akan dilakukan inisialisasi kalman filter untuk tracking posisi wajah jika wajah gagal terdeteksi di frame – frame selanjutnya. Kemudian, citra wajah yang terdeteksi akan melalui proses ekstraksi fitur dan vektor fitur yang dihasilkan disimpan. Pada saat ini tidak dilakukan proses klasifikasi karena proses klasifikasi dilakukan ketika seluruh frame proses-memasuki-ruangan telah didapat.

Pada waktu antara 00:01 sampai 00:04, merupakan proses orang tersebut memasuki ruangan. Frame – frame yang berada pada rentang waktu ini pasti terdapat wajah. Namun, kegagalan proses face detection masih mungkin terjadi jika wajah orang tersebut terhalang objek atau orang tersebut memalingkan wajah. Pada saat ini kalman filter berperan untuk memperkirakan posisi wajah. Posisi wajah merupakan informasi yang penting untuk mengetahui apakah orang tersebut telah selesai memasuki ruangan atau belum. Jika prediksi posisi wajah oleh kalman filter berada diluar area frame, maka saat itulah orang tersebut selesai memasuki ruangan. Pada saat ini, proses ekstraksi fitur hanya dilakukan jika proses *face detection* berhasil. Saat ini nantinya disebut sebagai '*sequence'* yang menandakan frame – frame pada saat ini adalah urutan frame seseorang dari awal memasuki ruangan sampai keluar frame.

Pada waktu 00:04, merupakan saat terakhir wajah orang tersebut tertangkap di frame. Pada saat ini, vektor fitur telah dikumpulan dari seluruh frame proses-memasuki-ruangan sehingga proses klasifikasi dapat dilakukan. Setelah proses

klasifikasi dilakukan, hasil pengenalan wajah ditampilkan ke layar dan dilanjutkan ke iterasi frame berikutnya. Diagram alir proses secara keseluruhan ditunjukan pada **Gambar 3.13.**



Gambar 3.13 Diagram Alir Proses Secara Keseluruhan

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi sistem dari rancangan metode yang telah dibahas pada Bab III meliputi kode program dalam aplikasi. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Citra yang diolah pada tugas akhir ini merupakan citra yang diperoleh dari frame video kamera CCTV. Pengolahan citra tersebut dilakukan pada perangkat – perangkat berikut.

4.1.1 Perangkat Keras

Lingkungan implementasi sistem pengenalan wajah merupakan sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah *notebook* bertipe DELL Inspiron 7447.

Spesifikias PC yang digunakan pada tugas akhir ini adalah prosesor Intel Core i7 4720HQ dengan 8 CPU yang masing – masing berkecepatan 2.60 GHz dan *Random Access Memory* (RAM) berkapasitas 8,00 GB.

4.1.2 Perangkat Lunak

Tugas akhir ini dibangun menggunakan bahasa pemrograman python 2.7 64 bit. Penggunaan Python didukung oleh modul Numpy untuk pengolahan vektor dan matriks, OpenCV untuk pengolahan citra dan Scikit-earn untuk penggunaan algoritma *machine learning*. Untuk memudahkan pencarian modul tersebut, python dipasang pada sebuah *environtment* sekaligus *package manager* Anaconda versi 4.3.29. Sementara, untuk pengolahan teks menggunakan IDE Spyder 3.4.

4.2 Implementasi *Face Detection* dan *Face Tracking*

Proses *face detection* akan melakukan deteksi wajah dari citra masukan. Pertama – tama, sebuah objek `detector` diinisialisasi. `Detector` merupakan suatu objek yang digunakan oleh modul Seeta Face untuk mendeteksi wajah. Kemudian, `detector` tersebut akan mendeteksi wajah pada citra masukan menggunakan fungsi `detect`. Hasil deteksi merupakan list objek `faces` yang menyimpan beberapa wajah yang terdeteksi dan attribute posisi wajah.

```
1  from pyseeta import Detector
2  detector = Detector()
3  faces = detector.detect(image_input)
4  face = faces.pop()
5  print face.top
6  print face.bottom
7  print face.left
8  print face.right
```

Kode Sumber 4.1 Implementasi *Face Detection*

Proses *face tracking* diawali dengan inisialisasi kalman filter. Parameter pertama merupakan jumlah dynamic parameter. Dynamic parameter merupakan nilai – nilai yang akan diprediksi menggunakan kalman filter. Jumlah dynamic parameter adalah 4, yakni x, y, v_x, dan v_y. x merupakan posisi pusat wajah terhadap sumbu x. y meruapkan posisi pusat wajah terhadap sumbu y. v_x merupakan kecepatan perubahan posisi wajah terhadap sumbu x. v_y merupakan kecepatan perubahan posisi wajah terhadap sumbu y. Parameter kedua merupakan jumlah nilai yang bisa di dapat dari proses pengukuran. Proses pengukuran merupakan proses deteksi wajah. Dari proses ini, kita bisa medapatkan nilai x dan y pada dynamic parameter sehingga nilai parameter kedua adalah 2. Parameter ketiga merupakan cotrol parameter. Pada skema deteksi wajah tidak memerlukan adanya control parameter karena tidak ada intervensi pengguna aplikasi terhadap perpindahan posisi wajah pada frame. Perhatikan baris 5 pada **Kode Sumber 4.2**.

Langkah berikutnya adalah melakukan inisialisasi matriks yang dibutuhkan, diantaranya state transition matrix (A),

observation matrix (H), proses error covariance (Q), measurement error covariance (R), process covariance (P) dan state (x). Inisialisasi matriks A ditunjukan pada baris 7, inisialisasi matriks H ditunjukan pada baris 8, inisialisasi matriks Q ditunjukan pada baris 9, inisialisasi matriks R ditunjukan pada baris 10, inisialisasi matriks P ditunjukan pada baris 11, dan inisialisai vektor x ditunjukan pada baris 12.

```
1  from pyseeta import Detector
2  import numpy as np
3  Import cv2
4
5  kalman = cv2.KalmanFilter(4,2,0)
6  state = np.array([center_x, center_y, 0, 0],
   dtype='float64')
7  kalman.transitionMatrix = np.array(
   [[1., 0., 1., 0.],
   [0., 1., 0., 1.],
   [0., 0., 1., 0.],
   [0., 0., 0., 1.]])
8  kalman.measurementMatrix = 1. * np.eye(2, 4)
9  kalman.processNoiseCov = 1e-5 * np.eye(4, 4)
10 kalman.measurementNoiseCov = 1e-3 * np.eye(2, 2)
11 kalman.errorCovPost = 1e-1 * np.eye(4, 4)
12 kalman.statePost = state
13
14 detector = Detector()
15 faces = detector.detect(image_input)
16
17 if len(faces)>0 :
18     center_x = (face.left + face.right) /2
19     center_y = (face.top + face.bottom) /2
20     prediction = kalman.predict()
21     measurement = (center_x, center_y)
22     posterior = kalman.correct(measurement)
23
24 else :
25     prediction = kalman.predict()
```

Kode Sumber 4.2 Implementasi *Face Tracking*

4.3 Implementasi Ekstraksi Fitur

Pertama – tama, tentukan ukuran blok. Pada tugas akhir ini menggunakan ukuran blok 8x8 piksel. Kemudian, tentukan ukuran citra yang di-*resize*, yakni 64x64 piksel. Tentukan juga jumlah fitur

AC yang diambil dari matriks koefisien DCT. Perhatikan baris 3 sampai 7 **Kode Sumber 4.3.**

Baris 12 sampai 19 merupkan iterasi tiap blok untuk dilakukan transformasi DCT dan diambil 5 koefisien *AC*. Vektor fitur dari tiap blok tersebut kemudian digabungkan pada `dct_feature`. Variabel `x_blok` dan `y_block` menyimpan koordinat tiap blok.

```
1  import cv2
2
3  DCT_BLOCK_SIZE = 8
4  DCT_INPUT_IMAGE = 64
5  DCT_TOTAL_BLOCK = (DCT_INPUT_IMAGE/DCT_BLOCK_SIZE)**2
6  DCT_FEATURE_LENGTH = DCT_TOTAL_BLOCK * 5
7
8  img_input = cv2.imread(join(INPUT_PATH, filename),
   cv2.IMREAD_GRAYSCALE)
9  img_input = np.array(img_input, dtype=np.float64)/255
10 img_resized = cv2.resize(img_input, (64, 64),
   interpolation=cv2.INTER_LINEAR)
11
12 dct_feature = np.zeros(DCT_FEATURE_LENGTH,
   dtype=np.float64)
13
14 for i in range(DCT_TOTAL_BLOCK):
15     x_block = i/DCT_BLOCK_SIZE
16     y_block = i%DCT_BLOCK_SIZE
17
18     img_block = img_resized[ x_block*DCT_BLOCK_SIZE :
   x_block*DCT_BLOCK_SIZE+DCT_BLOCK_SIZE ,
   y_block*DCT_BLOCK_SIZE : y_block*DCT_BLOCK_SIZE +
   DCT_BLOCK_SIZE].copy()
19     dct_block = cv2.dct(img_block)
20
21     dct_feature[i*5] = dct_block[0, 1]
22     dct_feature[i*5+1] = dct_block[1, 0]
23     dct_feature[i*5+2] = dct_block[2, 0]
24     dct_feature[i*5+3] = dct_block[1, 1]
25     dct_feature[i*5+4] = dct_block[0, 2]
26
27 print dct_feature
```

Kode Sumber 4.3 Implementasi Ekstraksi Fitur

4.4 Implementasi Klasifikasi

Proses klasifikasi diawali dengan proses traning dataset. Data training diambil dari file yang menyimpan vektor fitur DCT data training. Data dari file tersebut dibaca dan disimpan pada variabel `dataTrainingList` dan `classTrainingList`. Baris 11 sampai baris 17 merupakan proses pembacaan data training dari file. Kemudian, inisilisasi *kNN* dengan nilai k = 5. Proses training menggunakan fungsi `fit` pada objek `kNeighboursClassifier`. Proses testing menggunakan fungsi `predict` pada objek `kNeighboursClassifier`. Perhatikan baris 23 dan 26.

```
1  from sklearn.neighbors import KneighborsClassifier
2  import numpy as np
3  from os.path import join
   from os import listdir
   
4  INPUT_TRAINING_PATH = '\path\of\data-
   training\folder'
5  
6  for trainingFile in listdir(INPUT_TRAINING_PATH):
7      sample = np.genfromtxt(join(INPUT_TRAINING_PATH,
   trainingFile), delimiter=',')
8      token = trainingFile.split('.')
9      token = token[0].split('-')
10 
11     dataTrainingList.append(sample)
12     classTrainingList.append(token[0])
13 
14 dataTrainingList = np.array(dataTrainingList)
15 classTrainingList = np.array(classTrainingList)
16 
17 knn = KNeighborsClassifier(n_neighbors=5)
18 knn.fit(dataTrainingList, classTrainingList)
19 
20 dataTestingList.append(get_dct_feature(image_input))
21 knn_predictions = knn.predict(dataTestingList)
```

Kode Sumber 4.4 Implementasi Klasifikasi

4.5 Implementasi Proses Secara Keseluruhan

Kode Sumber 4.5 merupakan implementasi proses secara keseluruhan berdasarkan diagram alir **Gambar 3.13.** Modul `feature_extraction_dct` merupakan implementasi ekstraksi fitur yang diletakkan pada *file* terpisah. Sementara, fungsi – fungsi tambahan yang diperlukan oleh sistem pengenalan wajah diletakkan pada modul `utils`. Pemisahan fungsi-fungsi tambahan dari program utama bertujuan memudahkan pembacaan alur program utama.

```
from os.path import join
from os import listdir
import numpy as np
import cv2
from sklearn.neighbors import KNeighborsClassifier

from pyseeta import Detector
from feature_extraction_dct import get_dct_feature
from utils import draw_cross

INPUT_TRAINING_PATH = '\path\to\training-data\folder'
INPUT_TESTING_PATH = '\path\to\testing-data\folder'

detector = Detector()

for trainingFile in listdir(INPUT_TRAINING_PATH):
  sample = np.genfromtxt(join(INPUT_TRAINING_PATH,
trainingFile), delimiter=',')
  token = trainingFile.split('.')
  token = token[0].split('-')
  dataTrainingList.append(sample)
  classTrainingList.append(token[0])

dataTrainingList = np.array(dataTrainingList)
classTrainingList = np.array(classTrainingList)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(dataTrainingList, classTrainingList)

isKalmanFilterInitialized = 0
isInSequence = 0
sequenceNumber = 0
frameNumber = 0

message_status = "-"
```

```
message_result = "-"

kalman = cv2.KalmanFilter(4,2,0)

roi_x = 300
roi_y = 100
roi_w = 600
roi_h = 400

cap = cv2.VideoCapture(join(INPUT_TESTING_PATH, 'randi-test-natural.mp4'))
while (cap.isOpened()):
  ret, img_input = cap.read()
  img_gray = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)
  img_roi = img_gray[roi_y:roi_y+roi_h, roi_x:roi_x+roi_w].copy()

  # Detect face
  faces = detector.detect(img_roi)

  if isInSequence == 1 :
    if len(faces)>0 :
      face = faces.pop()
      face.left = face.left + roi_x
      face.right = face.right + roi_x
      face.top = face.top + roi_y
      face.bottom = face.bottom + roi_y

      # Record face as data testing
      img_face = img_gray[face.top:face.bottom, face.left:face.right].copy()
      img_face = np.array(img_face, dtype=np.float64)/255
      img_resized = cv2.resize(img_face, (64, 64), interpolation=cv2.INTER_LINEAR)
      dataTestingList.append( get_dct_feature(img_resized))

      # Update kalman filter
      center_x = (face.left + face.right) /2
      center_y = (face.top + face.bottom) /2

      prediction = kalman.predict()
      measurement = (center_x, center_y)
      posterior = kalman.correct(measurement)
```

```
      draw_cross(
img_input,(np.int32(posterior[0]),np.int32(posterior[1])
), (0, 0, 255), 3)
      cv2.rectangle(img_input, (face.left, face.top),
(face.right, face.bottom), (255, 0, 0), 2)
      isNotInRoi = center_x > roi_x + roi_w or center_x
< roi_x or center_y < roi_y or center_y > roi_y + roi_h

      if isNotInRoi:
        # Stop sequence
        isInSequence = 0

        # Renew kalman filter
        kalman = cv2.KalmanFilter(4,2,0)

        # Classification
        knn_predictions = knn.predict(dataTestingList)
        dataClass, counts = np.unique(knn_predictions,
return_counts=True)
        maxIndex = counts.argmax()
        message_result = "result seq
"+str(sequenceNumber)+" : "+dataClass[maxIndex]

        dataTestingList = []

    else :
      prediction = kalman.predict()
      draw_cross(
img_input,(np.int32(prediction[0]),np.int32(prediction[1
])), (0, 255, 0), 3)

      isNotInRoi = prediction[0] > roi_x + roi_w or
prediction[0] < roi_x or prediction[1] < roi_y or
prediction[1] > roi_y + roi_h
      if isNotInRoi:
        # Stop sequence
        isInSequence = 0

        # Renew kalman filter
        kalman = cv2.KalmanFilter(4,2,0)

        # Classification
        knn_predictions = knn.predict(dataTestingList)
        dataClass, counts = np.unique(knn_predictions,
return_counts=True)
        maxIndex = counts.argmax()
        message_result = "result seq
"+str(sequenceNumber)+" : "+dataClass[maxIndex]
```

```
        dataTestingList = []

  else :
    if len(faces)>0 :
      face = faces.pop()
      face.left = face.left + roi_x
      face.right = face.right + roi_x
      face.top = face.top + roi_y
      face.bottom = face.bottom + roi_y

      center_x = (face.left + face.right) /2
      center_y = (face.top + face.bottom) /2

      # Init kalman filter
      state = np.array([center_x, center_y, 0, 0],
dtype='float64')
      kalman.transitionMatrix = np.array([[1., 0., 1.,
0.],
[0., 1., 0., 1.],
[0., 0., 1., 0.],
[0., 0., 0., 1.]])
      kalman.measurementMatrix = 1. * np.eye(2, 4)
      kalman.processNoiseCov = 1e-5 * np.eye(4, 4)
      kalman.measurementNoiseCov = 1e-3 * np.eye(2, 2)
      kalman.errorCovPost = 1e-1 * np.eye(4, 4)
      kalman.statePost = state

      isKalmanFilterInitialized = 1

      # Record face as data testing
      img_face = img_gray[face.top:face.bottom,
face.left:face.right].copy()
      img_face = np.array(img_face,
dtype=np.float64)/255
      img_resized = cv2.resize(img_face, (64, 64),
interpolation=cv2.INTER_LINEAR)
      dataTestingList.append(
get_dct_feature(img_resized))

      # Start sequence
      isInSequence = 1
      sequenceNumber = sequenceNumber + 1

      # Set message
      message_status = "Seq no. "+str(sequenceNumber)
```

```
    else :
      message_status = "Not in seq"

  # draw roi
  cv2.rectangle(img_input, (roi_x, roi_y), (roi_x +
roi_w, roi_y + roi_h), (255, 255, 255), 3)
  font = cv2.FONT_HERSHEY_SIMPLEX
  cv2.putText(img_input,message_result,(0, 600), font,
1,(0,255,0),2,cv2.LINE_AA)
  cv2.putText(img_input,message_status,(0, 500), font,
1,(0,255,0),2,cv2.LINE_AA)

  cv2.imshow('video', img_input)
  k = cv2.waitKey(30) & 0xff
  if k == 27:
    break

cap.release()
cv2.destroyAllWindows()
```

Kode Sumber 4.5 Implementasi Proses Secara Keseluruhan

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini merupakan sebuah PC dengan prosesor Intel Core i7 4720HQ dengan 8 CPU yang masing – masing berkecepatan 2.60 GHz, *Random Access Memory* (RAM) berkapasitas 8,00 GB dan sistem operasi Windows 10.

Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemrograman Python 2.7 yang dipasang pada sebuah *environtment* sekaligus *package manager* Anaconda versi 4.3.29.

5.2 Data Uji Coba

Data uji coba yang digunakan sebagai data masukan adalah video CCTV. Tiap video berisi satu orang yang sedang memasuki ruangan Laboratorium Komputasi Cerdas dan Visi, Departemen Informati, ITS. Video CCTV diberikan label sesuai dengan nama orang yang memasuki ruangan. Proses pelabelan dilakukan secara manual. Sistem pengenalan wajah kemudian memproses data video dalam bentuk citra yang diambil dari frame video tersebut. **Gambar 5.1** menunjukkan citra frame yang diambil dari video. Untuk menguji keberanan, hasil pengenalan wajah oleh sistem akan dicocokan dengan label yang diberikan pada video CCTV.

5.3 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui nilai parameter yang tepat untuk digunakan pada sistem pengenalan wajah. Penggunaan nilai parameter yang tepat dapat memberikan hasil pengenalan wajah yang lebih baik.

Skenario uji coba pada tugas akhir ini terdiri dari dua macam, yaitu:

1. Uji coba penentuan jumlah koefisien *AC* yang diambil pada ekstraksi fitur
2. Uji coba penentuan nilai *k* pada *kNN*

Gambar 5.1 Citra yang Diambil dari Frame Video

5.4 Uji Coba Penentuan Jumlah Koefisien *AC* pada Ekstraksi Fitur

Uji coba ini dilakukan untuk menentukan jumlah koefisien *AC* yang tepat pada proses ekstraksi fitur menggunakan DCT. Jumlah koefisien *AC* yang diambil akan mempengaruhi banyaknya informasi yang didapat dari suatu citra wajah. Semakin banyak jumlah koefisien, maka semakin banyak informasi yang diperoleh dari citra wajah. Banyaknya informasi yang diperoleh akan meningkatkan performa algoritma klasifikasi. Namun, Banyaknya jumlah koefisien juga menyebabkan beban komputasi yang semakin tinggi. Oleh karena itu, perlu dilakukan uji coba penentuan jumlah koefisien *AC* untuk menentukan jumlah koefisien *AC* yang optimal terhadap hasil pengenalan wajah dan beban komputasi.

Pengujian dilakukan dengan menghitung *accuracy* hasil pengenalan wajah menggunakan *confussion matrix*. Penghitungan *accuracy* dilakukan terhadap frame yang terdeteksi benar pada suatu kelas terhadap jumlah seluruh frame pada kelas tersebut. Jumlah koefisien *AC* yang diujikan adalah 5, 10, dan 15. Sementara, proses klasifikasi menggunakan nilai k tetap selama uji coba jumlah koefisien *AC*, yakni 5. Jumlah koefisien *AC* dengan nilai *accuracy* terbaik akan digunakan sebagai jumlah koefisien *AC* ekstraksi fitur pada sistem pengenalan wajah ini. *Confussion matrix* hasil uji coba dapat dilihat pada bagian lampiran tugas akhir ini.

Tabel 5.1 Hasil Uji Coba Penentuan Koefisien *AC* pada Ekstraksi Fitur

Jumlah Koefisien *AC*	**5**	10	15
Accuracy (%)	**70,73**	68,94	68,86

Hasil uji coba ditunjukkan pada **Tabel 5.1**. Dari tabel tersebut terlihat bahwa *accuracy* dengan jumlah koefisien AC 5 adalah 70,73%, *accuracy* dengan jumlah koefisien *AC* 10 adalah 68,94%, dan *accuracy* dengan jumlah koefisien *AC* 15 adalah 68,86%.

5.5 Uji Coba Penentuan Nilai *k* pada *kNN*

Uji coba ini dilakukan untuk mengetahui nilai *k* yang memberikan hasil pengenalan terbaik pada algoritma *kNN*. Nilai *k* mempengaruhi jumlah tetangga terdekat pada algoritma *kNN*. Tetangga terdekat merupakan data – data yang dipertimbangakan sebagai hasil klasifikasi. Oleh karena itu, perbedaan nilai k yang digunakan pada *kNN* akan memberikan performa pengenalan wajah yang berbeda.

Pengujian dilakukan dengan menghitung *accuracy* hasil pengenalan wajah menggunakan *confussion matrix*. Penghitungan

accuracy dilakukan terhadap frame yang terdeteksi benar pada suatu kelas terhadap jumlah seluruh frame pada kelas tersebut. Nilai k yang diujikan adalah 1, 3, 5, 7 dan 9. Sementara, proses ekstraksi fitur menggunakan jumlah koefisien *AC* dengan hasil uji coba terbaik, yakni 5. Nilai k terbaik akan digunakan sebagai nilai *k* algortima *kNN* pada sistem pengenalan wajah ini. *Confussion matrix* hasil uji coba dapat dilihat pada bagian lampiran tugas akhir ini.

Tabel 5.2 Hasil Uji Coba Penentuan Nilai *k* pada *kNN*

Nilai k	1	3	5	7	9
Accuracy (%)	63,62	63,31	70,73	70,21	69,91

Hasil uji coba ditunjukkan pada **Tabel 5.2**. Dari tabel tersebut terlihat bahwa akurasi yang dihasilkan oleh nilai k = 1 adalah 63,62%, akurasi yang dihasilkan oleh nilai *k* = 3 adalah 63,31%, , akurasi yang dihasilkan oleh nilai *k* = 5 adalah 70,73%, akurasi yang dihasilkan oleh nilai *k* = 7 adalah 70,21% dan akurasi yang dihasilkan oleh nilai *k* = 9 adalah 69,91%.

5.6 Evaluasi Uji Coba Penentuan Jumlah Koefisien *AC* pada Ekstraksi Fitur

Berdasarkan hasil uji coba penentuan jumlah koefisien *AC* pada ekstraksi fitur, akurasi terbaik dihasilkan oleh jumlah koefisien *AC* 5. Perhatikan **Tabel 5.1**. Hal ini menunjukkan banyaknya informasi yang di-ekstrak dari suatu citra wajah tidak berbanding lurus dengan performa sistem pengenalan wajah. Semakin banyak jumlah koefisien *AC* yang digunakan, semakin kecil akurasi yang dihasilkan.

Penggunaan jumlah koefisien *AC* yang semakin banyak memerlukan penggunaan memori dan beban komputasi lebih. Namun, hal tersebut tidak diiringi dengan peningkatan performa sistem pengenalan wajah.

5.7 Evaluasi Uji Coba Penentuan Nilai *k* pada *kNN*

Berdasarkan hasil uji coba penentuan nilai k pada *kNN*, akurasi terbaik dihasilkan oleh nilai $k = 5$. Hal ini menunjukkan. Untuk perubahan nilai $k = 1$ ke nilai $k = 3$ terjadi penurunan nilai akurasi namun tidak terlalu signifikan. Perubahan akurasi yang tidak terlalu signifikan ini dapat terjadi akibat kurangnya data yang digunakan selama proses uji coba. Sementara, perubahan nilai $k = 3$ ke nilai $k = 5$ menunjukkan perubahan akurasi yang sangat signifikan. Namun, penggunaan nilai $k = 7$ dan nilai $k = 9$ menunjukkan hasil akurasi yang konsisten menurun. Perubahan akurasi dengan nilai $k = 3$ ke nilai $k = 5$ menunjukkan algoritma *kNN* memerlukan jumlah tetangga yang banyak agar dapat melakukan klasifikasi dengan baik. Jumlah tetangga yang banyak dapat dijadikan sebagai referensi bagi *kNN* dalam setiap melakukan klasifikasi. Namun, jika jumlah *k* lebih dari 5 akan menyebabkan akurasi pengenalan wajah menurun.

5.8 Evaluasi Hasil Pengenalan Wajah Berbasis Video

Evaluasi hasil pengenalan wajah berbasis video dilakukan dengan membandingkan pengenalan wajah berbasis video dengan pengenalan wajah berbasis frame. Pengenalan wajah berbasis video merupakan pengenalan wajah yang meperhatikan beberapa frame pada video, sementara pengenalan wajah berbasis frame merupakan pengenalan wajah yang hanya memperhatikan satu frame saja.

Tabel 5.3 Hasil Pengenalan Wajah

	Berbasis Video	Berbasis Frame
Accuracy (%)	**100**	70,73

Evaluasi ini dilakukan pada 13 video uji coba dengan total frame 1336. Tiap video terdiri dari satu orang yang sedang memasuki ruangan. Pada tahap ini digunakan parameter yang didapat pada uji coba sebelumnya. Parameter tersebut adalah jumlah koefisien $AC = 5$ pada ekstraksi fitur dan nilai $k = 5$ pada *kNN*. Hasil pengenalan wajah berbasis video dibandingkan dengan

pengenalan wajah berbasis frame. **Tabel 5.3** menunjukkan akurasi hasil pengenalan wajah.

Hasil pengenalan wajah berbasis video mampu mengenali 13 video dengan benar dari 13 video uji coba. Seluruh orang pada video uji coba mampu dikenali seseuai dengan nama orang sesungguhnya. Sementara, hasil pengenalan wajah berbasis frame mampu mengenali 945 frame dengan benar dari 1336 frame yang terdapat wajah. *Confussion matrix* dapat dilihat pada bagian lampiran dengan nilai k = 5 dan jumlah koefisien *AC* = 5.

Peningkatan akurasi untuk pengenalan wajah berbasis video dapat terjadi karena proses penentuan hasil pengenalan tidak hanya memperhatikan wajah pada frame tertentu, namun juga memperhatikan seluruh frame ketika seseorang sedang memasuki ruangan sehingga kesalahan proses pengenalan pada suatu frame dapat ditutupi dengan hasil pengenalan oleh frame lainnya.

Evaluasi pengenalan wajah berbasis video juga dilakukan terhadap waktu eksekusi. Hal ini dilakukan untuk mengetahui apakah sistem pengenalan wajah yang dibangun dapat menjalankan fungsinya secara *real-time*. Proses uji coba ini dilakukan terhadap seluruh *data testing* pada **Tabel 3.2**. Proses uji coba ini dilakukan dengan membandingkan hal berikut.

1. Waktu rata-rata pemrosesan frame selama seseorang memasuki ruangan sampai sistem menampilkan hasil pengenalan.
2. Waktu video dalam menampilkan frame. Pada umumnya video menampilkan frame dengan kecepatan 30 fps sehingga diperoleh waktu video dalam menampilkan satu frame adalah 0,033 sekon.

Tabel 5.4 menunjukkan hasil uji coba waktu eksekusi. Waktu eksekusi rata – rata tiap frame pada 13 video data uji coba berkisar antara 0,060 sekon sampai 0,076 sekon. Hasil ini menunjukkan bahwa performa sistem pengenalan wajah masih belum dapat bekerja secara *real-time* karena proses video menampilkan satu frame memakan waktu 0,033 sekon (30 fps). Hasil tersebut

menunjukkan bahwa sistem pengenalan wajah akan memiliki *delay* waktu sekitar dua kali lipat dibandingkan dengan proses video menampilkan frame.

Tabel 5.4 Evaluasi Waktu Eksekusi

No. Video	Kelas	Jumlah Frame Memasuki Ruangan	Waktu Total (Sekon)	Waktu Rata – Rata Pemrosesan Tiap Frame (Sekon)
1	Chasni	114	7,641	0,067
2	Dandi	158	10,065	0,064
3	Fadli	119	7,586	0,064
4	Galang	148	9,378	0,063
5	Mala	105	6,653	0,063
6	Nuzul	57	3,497	0,061
7	Ocid	102	6,503	0.064
8	Randi	107	8,136	0,076
9	Chasni	117	8,841	0,075
10	Dandi	170	10,667	0,063
11	Dandi	140	8,734	0,062
12	Nuzul	180	10,714	0,060
13	Randi	376	23,570	0,063

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi adalah sebagai berikut:

1. Proses ekstraksi fitur dapat dilakukan menggunakan algoritma Discrete Cosine Transform (DCT) dengan jumlah koefisien *AC* yang diambil adalah 5 untuk hasil akurasi pengenalan wajah terbaik.
2. Proses klasifikasi dapat dilakukan menggunakan algoritma *kNN* dengan nilai $k = 5$ untuk hasil akurasi pengenalan wajah terbaik.
3. Uji coba dilakukan dengan membandingkan performa pengenalan wajah berbasis video dengan performa pengenalan wajah berbasis frame. Pengenalan wajah berbasis video menghasilkan akurasi 100%, sementara pengenalan wajah berbasis frame menghasilkan akurasi 70,73%.
4. Sistem pengenalan wajah memerlukan waktu pemrosesan rata – rata tiap frame sekitar dua kali lipat dari waktu video dalam menampilkan frame. Hal ini menunjukkan sistem pengenalan wajah belum dapat bekerja secara *real-time.*

6.2 Saran

Saran yang dapat diberikan dalam pengujian sistem pengenalan wajah adalah sebagai berikut:

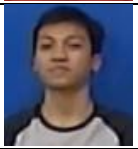
1. Penambahan jumlah data uji coba video dengan berbagai variasi kondisi untuk mengetahui sejauh mana sistem pengenalan wajah mampu mengenali wajah berbasis video.

2. Penambahan skenario uji coba pada proses klasifikasi menggunakan *kNN* dengan meningkatkan nilai k yang digunakan atau mengubah cara klasifikasi dari tanpa pembobotan menjadi dengan pembobotan (contoh pembobotan dengan jarak) untuk mengetahui parameter *kNN* yang lebih optimal untuk pengenalan wajah.

LAMPIRAN

A. *Confussion Matrix* Uji Coba Penentuan Jumlah Koefisien *AC* pada Ekstraksi Fitur

No.	Nama Kelas	Akronim	Foto
1.	chasni	c	
2.	dandi	d	
3.	fadli	f	
4.	galang	g	
5.	mala	m	
6.	nuzul	n	

7.	ocid	o	
8.	randi	r	

***Confussion Matrix* Jumlah Koefisien *AC* = 5**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	156	21	2	1	0	0	0	13
	d	26	270	10	4	0	3	0	40
	f	7	31	72	0	0	0	0	2
	g	7	41	0	91	0	0	0	5
	m	12	17	0	0	53	0	0	7
	n	5	25	0	3	0	53	0	22
	o	24	2	0	0	0	0	46	19
	r	19	20	0	1	0	1	0	204

***Confusion Matrix* Jumlah Koefisien *AC* = 10**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	154	21	2	1	0	0	0	15
	d	23	266	7	8	0	4	0	45
	f	8	31	71	0	0	0	0	3
	g	6	43	0	83	0	0	0	12
	m	11	17	0	0	53	0	0	8
	n	10	20	0	1	0	50	0	27
	o	27	2	0	0	0	0	40	22
	r	18	21	0	1	0	1	0	204

Confussion Matrix Jumlah Koefisien *AC* = 15

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	154	21	2	1	0	0	0	15
	d	25	263	5	10	0	4	0	46
	f	8	31	71	0	0	0	0	3
	g	8	41	0	84	0	0	0	11
	m	11	15	0	0	53	0	0	10

	n	10	21	0	1	0	51	0	25
	o	29	2	0	0	0	0	39	21
	r	18	20	0	1	0	1	0	205

B. Confussion Matrix Uji Coba Penentuan Nilai *k* pada *kNN*

***Confussion Matrix* Nilai k = 1**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	150	26	5	0	0	0	0	12
	d	19	279	8	10	3	4	0	30
	f	3	30	77	1	0	0	0	2
	g	4	33	0	96	0	0	0	11
	m	4	18	0	0	55	0	0	12
	n	2	20	0	2	0	66	0	18
	o	21	1	0	0	0	0	48	21
	r	30	26	2	1	2	0	0	79

***Confussion Matrix* Nilai k = 3**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	150	28	3	0	0	0	0	12
	d	23	265	8	11	0	4	0	42
	f	10	28	73	0	0	0	0	2
	g	7	41	0	88	0	0	0	8
	m	7	19	0	0	53	0	0	10
	n	6	20	0	2	0	60	0	20
	o	23	2	0	0	0	0	48	18
	r	32	22	0	1	1	0	0	189

***Confussion Matrix* Nilai k = 5**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	156	21	2	1	0	0	0	13
	d	26	270	10	4	0	3	0	40
	f	7	31	72	0	0	0	0	2
	g	7	41	0	91	0	0	0	5
	m	12	17	0	0	53	0	0	7

	n	5	25	0	3	0	53	0	22
	o	24	2	0	0	0	0	46	19
	r	19	20	0	1	0	1	0	204

***Confussion Matrix* Nilai k = 7**

		Predicted Class							
		c	d	f	g	m	n	o	r
Actual Class	c	152	20	2	1	0	0	0	18
	d	26	270	11	7	3	1	0	35
	f	6	31	73	0	0	0	0	3
	g	6	43	0	88	0	0	0	7
	m	12	16	0	0	53	0	0	8
	n	9	19	0	0	0	52	0	28
	o	25	2	0	0	0	0	45	19
	r	12	27	0	1	0	0	0	205

***Confussion Matrix* Nilai k = 9**

		Predicted Class							
		c	d	f	g	m	n	o	r
	c	152	20	0	1	0	0	0	20

Actual Class	d	22	270	12	5	0	1	0	42
	f	8	30	72	0	0	0	0	3
	g	7	47	0	86	0	0	0	4
	m	6	18	0	0	53	0	0	12
	n	10	23	0	0	0	50	0	25
	o	22	4	0	0	0	0	43	22
	r	10	25	0	1	1	0	0	208

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] "The DV, DVCAM, & DVCPRO Formats -- tech details, FAQ, and links." [Daring]. Tersedia pada: http://www.adamwilt.com/DV.html. [Diakses: 23-Jun-2018].

[2] R. G. Golla, "Viola Jones face detection and tracking explained - YouTube." [Daring]. Tersedia pada: https://www.youtube.com/watch?v=WfdYYNamHZ8. [Diakses: 23-Mei-2018].

[3] Anonim, "OpenCV: Face Detection using Haar Cascades." [Daring]. Tersedia pada: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html. [Diakses: 23-Mei-2018].

[4] S. Wu, M. Kan, Z. He, S. Shan, dan X. Chen, "Funnel-Structured Cascade for Multi-View Face Detection with Alignment-Awareness," *Neurocomputing*, vol. 221, hlm. 138–145, Jan 2017.

[5] S. Yan, S. Shan, X. Chen, dan W. Gao, "Locally Assembled Binary (LAB) feature with feature-centric cascade for fast and accurate face detection," dalam *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, hlm. 1–7.

[6] "minFunc - unconstrained differentiable multivariate optimization in Matlab." [Daring]. Tersedia pada: https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html. [Diakses: 27-Jun-2018].

[7] Y. C. Eldar, P. Kuppinger, dan H. Bolcskei, "Block-Sparse Signals: Uncertainty Relations and Efficient Recovery," *IEEE Trans. Signal Process.*, vol. 58, no. 6, hlm. 3042–3054, Jun 2010.

[8] G. Czerniak, "Kalman Filters for Undergrads." [Daring]. Tersedia pada: http://greg.czerniak.info/guides/kalman1/. [Diakses: 23-Mei-2018].

[9] D. Marshall, "The Discrete Cosine Transform (DCT)," 04-Okt-2001. [Daring]. Tersedia pada:

https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html. [Diakses: 22-Mei-2018].

[10] H. K. Ekenel, J. Stallkamp, dan R. Stiefelhagen, "A video-based door monitoring system using local appearance-based face models," *Comput. Vis. Image Underst.*, vol. 114, no. 5, hlm. 596–608, Mei 2010.

[11] H. K. Ekenel dan R. Stiefelhagen, "Local appearance based face recognition using discrete cosine transform," dalam *2005 13th European Signal Processing Conference*, 2005, hlm. 1–5.

[12] "(3) JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile - YouTube." [Daring]. Tersedia pada: https://www.youtube.com/watch?v=Q2aEzeMDHMA&t=462s. [Diakses: 27-Jun-2018].

[13] "Introduction to k-Nearest Neighbors." [Daring]. Tersedia pada: https://www.kdnuggets.com/2018/03/introduction-k-nearest-neighbors.html. [Diakses: 06-Jun-2018].

BIODATA PENULIS

Luqman Ahmad lahir di Blitar pada tanggal 1 November 1996. Penulis telah menempuh pendidikan dari SDIT Daarul Fataa (2002 – 2008), SMPN 2 Cibinong (2008 – 2011), dan SMAN 1 Cibinong (2011 – 2014) hingga terakhir Institut Teknologi Sepuluh Nopember Surabaya (2014 – 2018) di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi angkatan tahun 2014.

Selama belajar di kampus Informatika, penulis aktif di berbagai kegiatan keorganisasian, diantaranya sebagai Staf Departemen Riset dan Teknologi Himpunan Mahasiswa Teknik Computer-Informatika HMTC (2015 - 2016), Staf Keluarga Muslim Informatika KMI (2015 - 2016), Ketua Umum Keluarga Muslim Informatika KMI (2016 - 2017), dan Administrator Laboratorium Komputasi Cerdas dan Visi (2015 – 2017). Selain itu, penulis pernah terlibat pada proyek sistem pengadaan kontrak Badak LNG (2017) yang dikelola oleh Departemen Informatika.

Penulis memiliki bidang minat Komputasi Cerdas dan Visi (KCV) dengan fokus studi pada bidang *image processing*, dan *data mining*. Selain itu, penulis juga memliki ketertarikan di bidang perancangan dan pengembangan perangkat lunak. Komunikasi dengan penulis dapat dilakukan melalui email : **luqman_ahmads@ymail.com.**