TUGAS AKHIR – KI141502

DETEKSI EKSPRESI WAJAH PADA SEKUEN CITRA MENGGUNAKAN ALGORITMA ACTIVE SHAPE MODEL DAN KLASIFIER TWOFOLD RANDOM FOREST

ARIKA SAPUTRO
NRP 5112100073

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

DETEKSI EKSPRESI WAJAH PADA SEKUEN CITRA MENGGUNAKAN ALGORITMA ACTIVE SHAPE MODEL DAN KLASIFIER TWOFOLD RANDOM FOREST

ARIKA SAPUTRO
NRP 5112100073

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

(Halman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

FACE EXPRESSION DETECTION FROM IMAGE SEQUENCES USING ACTIVE SHAPE MODEL ALGORITHM AND TWOFOLD RANDOM FOREST CLASSIFIER

ARIKA SAPUTRO
NRP 5112100073

Supervisor I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA 2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Perancangan Perangkat Lunak Deteksi Ekspresi Wajah pada Sekuen Citra Menggunakan Metode Twofold Random Forest

TUGAS AKHIR

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

Oleh:

ARIKA SAPUTRO
NRP: 5112 100 073

Disetujui oleh Pembimbing Tugas Akhir:

Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. ..... (pembimbing 1)
NIP: 19490823 197603 2 001

Dr. Eng. Nanik Suciati, S.Kom., M.Kom. ..... (pembimbing 2)
NIP: 19710428 199412 2 001

SURABAYA
JUNI, 2016

(Halaman ini sengaja dikosongkan)

DETEKSI EKSPRESI WAJAH PADA SEKUEN CITRA MENGGUNAKAN ALGORITMA ACTIVE SHAPE MODEL DAN KLASIFIER TWOFOLD RANDOM FOREST

Nama	: Arika Saputro
NRP	: 5112100073
Jurusan	: Teknik Informatika, FTIf-ITS
Dosen Pembimbing I	: Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.
Dosen Pembimbing II	: Dr. Eng. Nanik Suciati, S. Kom., M. Kom.

Abstrak

Ekspresi wajah adalah informasi nonverbal yang penting untuk melengkapi komunikasi verbal. Ekspresi wajah berpengaruh penting dalam menentukan emosi dari manusia. Deteksi ekspresi wajah secara otomatis sangat penting untuk diaplikasikan, seperti untuk image understanding, health-care, human computer interaction, video games dan animasi data-driven. akan tetapi, deteksi ekspresi wajah dengan akurasi yang tinggi masih menjadi tantangan terbuka bagi semua peneliti[1].

Kebanyakan sistem analisis pada ekspresi wajah memfokuskan pada deteksi 6 emosi dasar yang diajukan oleh Ekman, yaitu : marah, jijik, takut, bahagia, sedih, dan terkejut. Setiap ekspresi dasar tersebut dapat didekomposisikan menjadi serangkaian Action Units (AUs) yang berkaitan[1]. Misalnya, ekspresi bahagia dapat didekomposisikan menjadi pipi yang naik dan sudut bibir yang tertarik. Bahkan pada studi psikologi klasik menunjukkan bahwa manusia secara sadar memetakan AUs ke kategori emosi dasar.

Tugas akhir ini menyajikan sebuah video-based method untuk menganalisis ekspresi wajah dengan mengenali AU dan dilakukan pelacakan terhadap titik fitur wajah menggunakan Active Shape Model (ASM). Vektor perpindahan antara frame

ekspresi netral dan frame ekspresi puncak digunakan sebagai fitur gerakan dari ekspresi wajah. Fitur tersebut diidentifikasi dengan level pertama klasifier random forest untuk mendeteksi AU. AU yang terdeteksi kemudian diklasifikasikan kedalam ekspresi-ekspresi yang berbeda-beda dengan klasifier random forest level kedua. Dengan menggunakan data yang diambil dari 3 subjek uji coba, akurasi terbesar didapatkan dengan nilai parameter averaging k = 3 dan iterasi twofold random forest = 100 dengan nilai akurasi sebesar 74.45% dengan 6 kelas ekspresi dan 100% dengan 5 kelas ekspresi.

Kata kunci: expression recognition , random forest, ASM

FACE EXPRESSION DETECTION FROM IMAGE SEQUENCE USING ACTIVE SHAPE MODEL ALGORITHM AND TWOFOLD RANDOM FOREST CLASSIFIER

Name	: Arika Saputro
NRP	: 5112100073
Department	: Informatics Engineering, FTIf-ITS
Supervisor I	: Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D.
Supervisor II	: Dr. Eng. Nanik Suciati, S. Kom., M. Kom.

Abstract

Facial expressions are an important nonverbal information to supplement verbal communication. Facial expressions are important in determining the influence of human emotions. Automatic detection of facial expressions is very important to be applied, such as for image understanding, health-care, human computer interaction, video games and animation data-driven. however, the detection of facial expressions with high accuracy remains a challenge open to all researchers[1].

Most system analysis on expression face detection focus on six basic emotions proposed by Ekman, namely: anger, disgust, fear, happiness, sadness, and surprise. Each of these basic expressions can be decomposed into a series of related Action Units (AUs)[1]. For example, the happy expression can be decomposed into the cheek that rises and the lip corners are stretch. Even in a classic psychology study shows that humans consciously charted AUs to basic emotional categories.

This final project presents a video-based method for analyzing facial expressions with the AUs to recognize and to track of facial feature points using Active Shape Model (ASM). The displacement vector between the frame and the frame of neutral expression used as the ultimate expression of the movement of the

facial expressions. Such a feature is identified by the first level klasifier random forest to detect the AU. AU is detected then classified into expressions that vary with random forest klasifier second level. Using data taken from three subjects test, best accuracy is obtained with averaging parameter k = 3 and twofold random forest iteration = 100 with a value 74.45% with 6 expression class and 100% with 5 expression class.

Keywords: expression recognition , random forest, ASM

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "*Deteksi Ekspresi Wajah pada Sekuen Citra Menggunakan Algoritma Active Shape Model dan Klasifier Twofold Random Forest*" dengan tepat waktu.

Dalam pelaksanaan dan pembuatan tugas akhir ini penulis mendapatkan banyak bantuan dari berbagai pihak. Pada kesempatan ini, ucapan terima kasih penulis berikan kepada:

1. Allah SWT, karena atas limpahan rahmat-Nya, penulis diberikan kemudahan dan kelancaran dalam mengerjakan tugas akhir ini.
2. Orang tua dan keluarga penulis yang senantiasa memberikan doa dan dukungan kepada penulis untuk menyelesaikan pengerjaan tugas akhir ini.
3. Ibu Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. dan Ibu Dr. Eng. Nanik Suciati, S. Kom., M. Kom. sebagai dosen pembimbing yang telah memberikan banyak arahan dan bantuan sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Bapak Ir.FX.Arunanto,M.Sc. selaku dosen wali yang telah memberikan bimbingan dan nasihat selama masa perkuliahan.
5. Teman-teman *user* TA dan Administrator Laboraturium Komputasi Cerdas dan Visi, yang telah memberikan bantuan dalam penyelesaian tugas akhir ini.
6. Semua pihak yang telah membantu terselesaikannya tugas akhir ini.

Penulis menyadari adanya banyak kekurangan dalam pengrejaan tugas akhir baik dari segi program maupun laporan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk penyempurnaan tugas akhir ini. Akhir kata, penulis meminta maaf bila terdapat kesalahan dalam penulisan laporan tugas akhir ini. Semoga hasil dari tugas akhir ini dapat

memberikan manfaat bagi pembaca pada umumnya dan penulis pada khususnya.

Surabaya, Juni 2015

Arika Saputro.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1 Latar Belakang

Ekspresi wajah adalah informasi nonverbal yang penting untuk melengkapi komunikasi verbal. Ekspresi wajah berpengaruh penting dalam menentukan emosi dari manusia. Deteksi ekspresi wajah secara otomatis sangat penting untuk diaplikasikan, misalnya untuk *image understanding*, *health-care*, *human computer interaction*, *video games* dan animasi *data-driven*. Akan tetapi, deteksi ekspresi wajah dengan akurasi yang tinggi masih menjadi tantangan terbuka bagi semua peneliti.

Secara umum terdapat dua jenis data yang digunakan dalam deteksi ekspresi wajah menggunakan citra wajah yaitu dengan satu citra dan dengan sekuen citra. Jenis data satu citra biasanya menggunakan citra ekspresi puncak sebagai datanya. Data ini mengabaikan banyak fitur dinamis seperti pergerakan otot-otot wajah. Jenis data sekuen citra terdiri dari citra ekspresi netral dan citra ekspresi puncak pada sekuen tersebut. Data ini dapat mengekstrak informasi gerakan otot-otot wajah lebih baik karena ekspresi wajah yang natural adalah proses dinamis yang dapat mengungkapkan keadaan emosi saat itu[1].

Data citra wajah memiliki variasi bentuk wajah dan bagian wajah (mata, alis, hidung, mulut, dll) yang berbeda-beda untuk setiap orang. Dengan jumlah titik wajah sebanyak 77 titik yang tersebar pada semua bagian wajah meliputi mata, hidung, alis, bibir, dan batas wajah akan membutuhkan waktu yang cukup lama untuk mendeteksi titik wajah tersebut. Metode *Active Shape Model* (ASM) dapat mereduksi dimensi dari titik fitur wajah tersebut tanpa kehilangan banyak informasi pada setiap variasi bentuk

wajah[2]. Sehingga dengan metode ini dapat mengoptimasi waktu yang dibutuhkan untuk mendeteksi 77 titik wajah.

Setiap ekspresi wajah dapat didekomposisikan menjadi serangkaian *Action Unit* (AU) yang berkaitan. Misalnya, ekspresi bahagia dapat didekomposisikan menjadi pipi yang naik dan sudut bibir yang tertarik. Bahkan pada studi psikologi klasik menunjukkan bahwa manusia secara sadar memetakan AU ke kategori emosi dasar menggunakan aturan yg terbatas[1]. Sehingga untuk menghindari kemungkinan semua fitur hilang pada satu bagian wajah, proses seleksi fitur tidak dapat dilakukan langsung untuk matrix titik wajah, melainkan harus dikelompokkan pada tiap bagian wajah kemudian dilakukan seleksi fitur. Metode Twofold Random Forest memiliki kemampuan untuk menyeleksi fitur pada tiap bagian wajah dan mengklasifikasikan *Action Unit* dan ekspresi pada data.

Tahapan yang dilakukan dalam deteksi ekspresi wajah antara lain deteksi titik fitur wajah, ekstraksi fitur, klasifikasi *action units* dan klasifikasi ekspresi. Pada tahapan deteksi titik fitur wajah digunakan metode *active shape model,* kemudian dari titik-titik fitur wajah yang dihasilkan dihitung fitur perpindahan ekspresi netral-puncak yang kemudian dijadikan fitur untuk mengklasifikasikan *action unit*. *Action unit* yang dihasilkan kemudian digunakan sebagai fitur untuk mengklasifikasikan ekspresi. Klasifier yang digunakan untuk mengklasifikasikan *action unit* dan ekspresi wajah adalah random forest.

Berdasarkan problem pengabaian fitur dinamis pada ekspresi wajah, waktu deteksi titik fitur wajah yang lama, dan kemungkinan semua fitur hilang pada satu bagian wajah saat proses seleksi fitur, Maka dibutuhkan sistem deteksi ekspresi wajah yang dapat mengatasi problem tersebut.

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah dalam tugas akhir ini :

1. Bagaimana cara mendeteksi dan melacak titik-titik fitur wajah menggunakan *Active Shape Model* (ASM).

2. Bagaimana cara menentukan ekspresi puncak dari data sekuen citra?
3. Apakah penggunaan filter *average* dapat mempengaruhi hasil akurasi sistem?
4. Apakah penggunaan Twofold Random Forest untuk mengklasifikasikan *Action Unit* dan ekspresi menghasilkan hasil yang lebih baik daripada menggunakan Random Forest?
5. Bagaimana mengimplementasikan deteksi ekspresi wajah pada pemrograman C++?

1.3 Batasan Masalah

Berikut beberapa hal yang menjadi batasan masalah dalam pengerjaan tugas akhir ini:

1. Resolusi minimal 640x480 pixel.
2. *Input* dari aplikasi ini berupa sekuen citra dari ekspresi wajah yang menghadap ke depan.
3. Ada 6 ekspresi yang dideteksi oleh aplikasi ini.
4. Dataset yang digunakan adalah dataset CK+, yang berupa sekuen citra ekspresi wajah : http://www.consortium.ri.cmu.edu/data/ck/.

1.4 Tujuan

Membuat sistem deteksi ekspresi wajah pada data sekuen citra dengan algoritma Active Shape Model (ASM) menggunakan klasifier Twofold Random Forest.

1.5 Metodologi

Tahap yang dilakukan untuk menyelesaikan tugas akhir ini adalah sebagai berikut:

1. **Penyusunan Proposal Tugas Akhir**

   Penulisan proposal ini merupakan tahap awal dalam pengerjaan tugas akhir. Pada proposal ini, penulis

mengajukan gagasan perancangan perangkat lunak deteksi ekspresi wajah pada sekuen citra menggunakan metode twofold random forest.

2. **Studi Literatur**

   Pada tahap ini dilakukan pencarian informasi dan studi literatur untuk dataset dan desain aplikasi yang akan dibuat. Informasi didapatkan dari buku dan literatur lain yang berhubungan dengan algoritma yang digunakan dalam pengerjaan tugas akhir ini yaitu studi literatur mengenai metode ekstraksi titik fitur wajah, klasifikasi *Action Unit* serta klasifikasi ekspresi wajah.

3. **Implementasi**

   Pada tahap ini dibangun perangkat lunak sesuai dengan rancangan yang diajukan pada proposal. Pembangunan perangkat lunak diimplementasikan sesuai dengan konsep yang telah didapatkan saat studi literatur.

4. **Pengujian dan Evaluasi**

   Tahap ini merupakan tahap uji coba aplikasi dan evaluasi data keluaran. Tahap uji coba dilakukan dengan membandingkan jumlah sekuen titik fitur wajah yang dirata-rata, dan jumlah kelas ekspresi wajah. Tahap evaluasi dilakukan dengan menghitung tingkat kebenaran pengenalan (akurasi) dan *confusion matrix* dari masing-masing kelas ekspresi.

5. **Penyusunan Buku Tugas Akhir**

   Tahap ini merupakan tahap dokumentasi dari tugas akhir. Buku tugas akhir berisi dasar teori, perancangan,

implementasi, serta hasil uji coba dan evaluasi dari aplikasi yang dibangun.

1.6 Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. Bab I. Pendahuluan

   Bab pendahuluan berisi penjelasan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan tugas akhir.

2. Bab II. Tinjauan Pustaka

   Bab tinjauan pustakan berisi penjelasan mengenai dasar teori yang mendukung pengerjaan tugas akhir.

3. Bab III. Analisis dan Perancangan

   Bab analisis dan perancangan berisi penjelasan mengenai analisis kebutuhan, perancangan sistem dan perangkat yang digunakan dalam pengerjaan tugas akhir serta urutan pelaksanaan proses.

4. Bab IV. Implementasi

   Bab implementasi berisi perancangan perangkat lunak deteksi ekspresi wajah pada sekuen citra menggunakan bahasa *C++*.

5. Bab V. Uji Coba dan Evaluasi

   Bab ini berisi hasil evaluasi aplikasi dengan menggunakan aplikasi yang dibangun. Juga disertakan analisis dari hasil evaluasi perangkat lunak.

6. Bab VI. Kesimpulan dan Saran

   Bab kesimpulan dan saran berisi kesimpulan hasil penelitian. Selain itu, bagian ini berisi saran untuk pengerjaan lebih lanjut atau permasalahan yang dialami dalam proses pengerjaan tugas akhir.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka berisi penjelasan teori yang berkaitan dengan implementasi perangkat lunak. Penjelasan tersebut bertujuan untuk memberikan gambaran mengenai sistem yang akan dibangun dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

3.1 Active Shape Model (ASM)

Active Shape Model merupakan algoritma object detection yang biasanya digunakan untuk *non-rigid shape object*. Misalnya, deteksi bentuk tangan, bentuk wajah dan posisi bagian-bagian wajah, bentuk *cartilage* pada sendi lutut, dll. Metode ini terdiri dari beberapa proses diantaranya *alignment,* Principal Component Analysis (PCA), dan mencocokkan model dengan target citra.

Pada proses *alignment*, tiap citra bentuk di *rotate, translate,* dan *scaling* terhadap satu citra bentuk(z^1) yang digunakan sebagai acuan.

$$a^j = (z^j.z^1)/(\|z^j\|)^2$$

$$b^j = \sum_{i=1}^{n} (xi^j yi^1 - xi^1 yi^j)/(\|z^j\|)^2$$

$$s^j = \sqrt{(a^j.a^j) + (b^j.b^j)}$$

$$\theta^j = \tan^{-1}\frac{b^j}{a^j}$$

$$\begin{bmatrix} \tilde{x}_j^i \\ \tilde{y}_j^i \end{bmatrix} = s^j . \begin{bmatrix} \cos\theta^j & \sin\theta^j \\ -\sin\theta^j & \cos\theta^j \end{bmatrix} . \begin{bmatrix} x_j^i \\ y_j^i \end{bmatrix}$$

Pada proses PCA, didapatkan model shape dengan beberapa parameter b yang nantinya akan digunakan untuk proses fitting dengan citra wajah.

3.2 Representasi Ekspresi Wajah

Ekspresi wajah adalah sebuah proses dinamis yang dapat dideteksi dari aktivitas otot wajah pada bagian-bagian wajah (alis, mata, hidung, mulut). Fitur dinamis dapat direpresentasikan oleh perbedaan titik wajah pada tiap frame. Enam ekspresi wajah dasar (marah, jijik, takut, bahagia, sedih, dan terkejut) dapat dideskripsikan dengan *Action Unit*(AU), dimana setiap AU berasal dari keterlibatan otot wajah tersebut.

Vektor posisi dari fitur titik ASM pada ekspresi netral frame pertama adalah :

$$S_0 = [x_{10}, y_{10}, x_{20}, y_{20}, \ldots, x_{n0}, y_{n0}] \quad (15)$$

Dan vektor posisi dari fitur titik wajah pada frame ekspresi puncak adalah :

$$S_n = [x_{1n}, y_{1n}, x_{2n}, y_{2n}, \ldots, x_{nn}, y_{nn}] \quad (16)$$

Sehingga perpindahan kedua vektor posisi (DVNP) dari fitur titik tersebut adalah :

$$S_t = [x_{1n} - x_{10}, y_{1n} - y_{10}, x_{2n} - x_{20}, y_{2n} - y_{20}, \ldots, x_{nn} - x_{n0}, y_{nn} - y_{n0}] \quad (17)$$

DVNP tersebut kemudian akan dinormalisasi dengan cara membaginya dengan keliling dari tiap bagian wajah(alis, mata, hidung, mulut).

$$\text{Fitur}[] = \frac{St[]}{\sum_{i=0}^{n-1} Dist(Point\ (i), Point\ (i+1))}$$

Fitur DVNP tersebut kemudian akan digunakan sebagai fitur gerakan ekspresi wajah pada langkah selanjutnya.

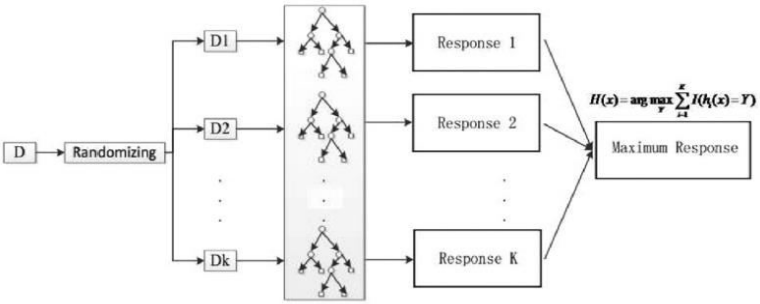
3.3 Random Forest

Random Forest (RF) adalah kombinasi dari pohon keputusan dimana setiap pohon tergantung pada nilai-nilai dari vektor acak dengan distribusi yang sama untuk setiap pohon. Dua

sumber keacakan, input acak dan fitur acak membuat RF mampu menangani ukuran fitur yang besar. Setiap pohon keputusan pada RF dibentuk secara rekursif dengan melakukan pengujian binary untuk setiap node yang bukan node daun berdasarkan contoh *training*. Pada proses klasifikasi, RF menggabungkan hasil dari pohon keputusan untuk memilih kelas yang paling populer. RF memiliki kemampuan untuk menyeimbangkan error pada populasi kelas data yang tidak seimbang. Klasifier RF dapat didefinisikan sebagai :

$$H(x) = \arg\max_{Y} \sum_{i=1}^{k} I(h_i(x) = Y) \tag{18}$$

Dimana *H(x)* adalah kombinasi akhir dari klasifier, k adalah banyaknya pohon keputusan, $h_i(x)$ merepresentasikan pohon keputusan ke-i, dan Y adalah label kelas.



Gambar 3.1 Klasifier Random Forest

(Halaman ini sengaja dikosongkan)

BAB III
DESAIN DAN PERANCANGAN

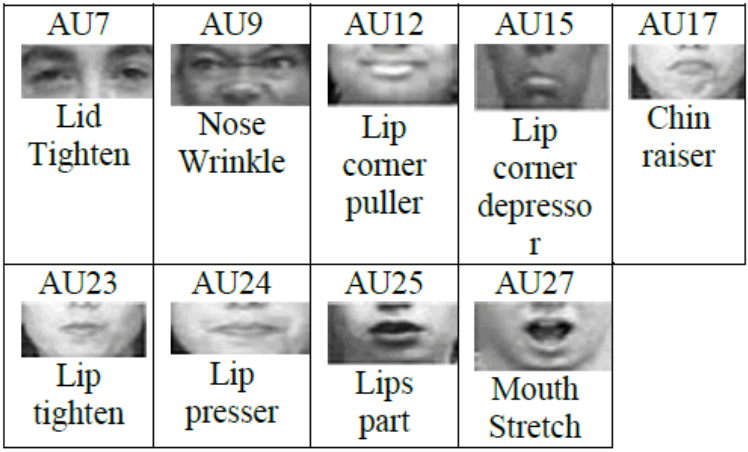
Pada Bab 3 akan dijelaskan mengenai perancangan sistem perangkat lunak untuk mencapai tujuan dari tugas akhir. Perancangan yang akan dijelaskan pada bab ini meliputi perancangan data, perancangan proses dan perancangan antar muka. Selain itu akan dijelaskan juga desain metode secara umum pada sistem.

4.1 Desain Metode Secara Umum

Pada tugas akhir ini akan dibangun suatu perangkat lunak untuk mendeteksi ekspresi wajah pada sekuen citra. Tahapan yang dilakukan untuk pembuatan perangkat lunak tersebut diantaranya adalah : deteksi titik-titik fitur wajah menggunakan algoritma Active Shape Model (ASM), deteksi ekpresi puncak pada sekuen citra, ekstraksi fitur perpindahan pada bagian-bagian wajah(alis, mata, hidung, mulut), klasifikasi *Action Unit (AU)* untuk tiap bagian-bagian wajah menggunakan random forest level pertama, dan klasifikasi ekspresi wajah menggunakan random forest level kedua.

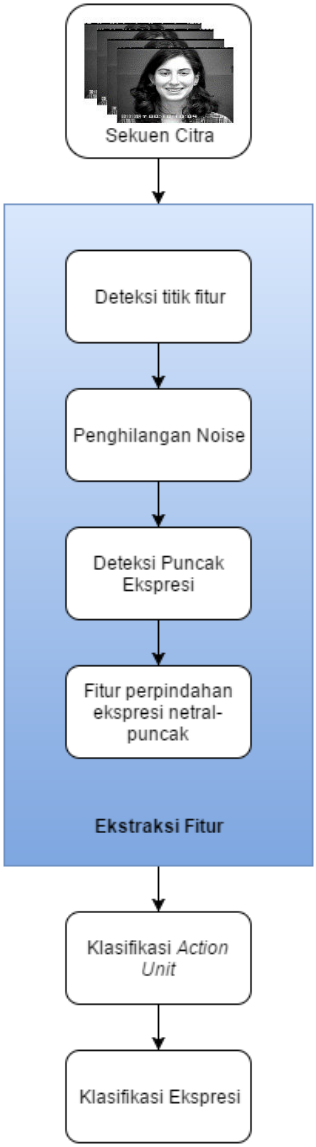
Deteksi titik-titik fitur wajah merupakan tahapan pertama yang harus dilakukan untuk medapatkan 77 titik pada bagian-bagian wajah(alis, mata, hidung, mulut). Untuk mendeteksi 77 titik pada bagian wajah tersebut digunakan algoritma Active Shape Model (ASM). Setelah titik-titik fitur wajah didapatkan pada semua sekuen citra, dilakukan deteksi ekspresi puncak pada sekuen citra dengan mencari total jarak *euclidian* dari perpindahan titik-titik fitur wajah tersebut terhadap citra pertama(ekspresi netral). Citra dengan total jarak *euclidian* terbesar adalah citra ekspresi puncak. Setelah didapatkan citra ekspresi puncak, dilakukan ekstraksi fitur perpindahan (fitur DVNP) pada bagian-bagian wajah dari ekspresi netral ke ekpresi puncak. Setelah didapatkan fitur perpindahan, dilakukan klasifikasi *Action Unit (AU)* untuk

tiap bagian-bagian wajah menggunakan random forest level pertama. *Action Unit (AU)* adalah sistem yang mewakili aktivitas otot yang menghasilkan perubahan penampilan wajah.



Gambar 4.1 ***Action Unit (AU)***

Setelah didapatkan *Action Unit (AU)* untuk tiap bagian-bagian wajah, dilakukan klasifikasi ekspresi menggunakan random forest level kedua. Bagian dari sistem yang dibangun ditunjukkan oleh gambar 3.2.



Gambar 4.2 Diagram alir model sistem

4.2 Perancangan Data

Perancangan data merupakan bagian yang terpenting dalam pengoperasian perangkat lunak karena diperlukan data yang tepat agar perangkat lunak dapat beroperasi dengan benar. Pada bagian ini akan dijelaskan mengenai perancangan data yang dibutuhkan untuk membangun perangkat lunak deteksi ekspresi wajah. Data yang diperlukan dalam pengoperasian perangkat lunak adalah data masukan (*input*) dan data keluaran (*output*) yang memberikan hasil proses pengoperasian perangkat lunak untuk pengguna.

4.2.1 Data Masukan

Data masukan adalah data awal yang akan diproses pada sistem pengenalan ekspresi wajah dari sekuen citra. Dataset yang digunakan adalah dataset CK+, yang berupa sekuen citra ekspresi wajah : http://www.consortium.ri.cmu.edu/data/ck/. Data yang diproses berupa sekuen citra dimana tiap sekuen citra merepresentasikan satu kelas ekspresi. Satu sekuen citra bisa terdiri dari 8 – 30 citra dimana citra pertama pada sekuen citra tersebut menggambarkan ekspresi netral.

Selain sekuen citra, terdapat data masukan nilai N yang merepresentasikan jumlah sekuen titik-titik fitur wajah yang dirata-rata, dan K yang merepresentasikan banyaknya kelas pada sistem pengenalan ekspresi wajah ini. Kedua nilai masukan ini digunakan untuk tahap evaluasi dari sistem pengenalan ekspresi wajah.

Gambar 4.3 Data masukan

4.2.2 Data Luaran

Data luaran dari sistem ini merupakan data hasil proses pengenalan ekspresi wajah. Data luaran yang didapatkan adalah hasil evaluasi akurasi pada berbagai kondisi nilai N dan K, *confusion matrix*, dan kelas ekspresi data masukan.

4.3 Perancangan Proses

Perancangan proses dilakukan untuk mengetahui alur dalam penerapan algoritma yang nantinya akan dipakai dalam tahap implementasi. Alur tersebut akan ditampilkan dalam diagram alir dari masing-masing proses.

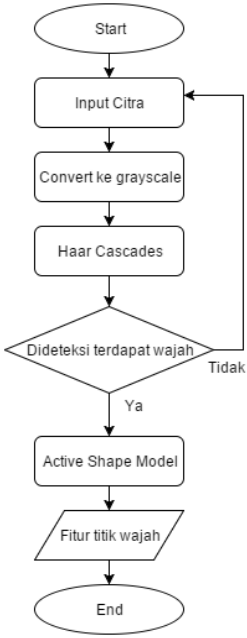
Berikut ini adalah tahapan rancangan dari sistem pengenalan ekspresi wajah dari sekuen citra:

4.3.1 Tahap Ekstraksi Fitur

Pada bagian ini, akan dijalaskan tahap-tahap yang dilakukan untuk menghasilkan fitur perpindahan titik-titik wajah dari ekspresi netral ke ekspresi puncak dari input data sekuen citra.

4.3.1.1 Deteksi Titik-Titik Fitur Wajah

Pada tahap ini, akan dilakukan deteksi letak 77 titik wajah untuk tiap citra wajah pada sekuen citra ekspresi. Pertama citra diubah menjadi citra greyscale, kemudian dilakukan deteksi wajah dengan algoritma haar cascades, setelah itu, metode Active Shape Model (ASM) digunakan untuk mendeteksi 77 titik fitur wajah tersebut. Gambar 3.4 menunjukkan diagram alir deteksi titik-titik wajah.

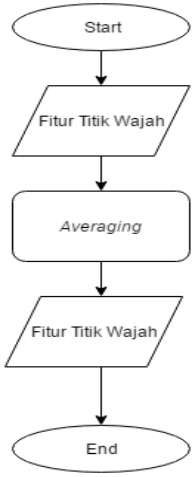


Gambar 4.4 Diagram alir deteksi titik-titik wajah

4.3.1.2 Penghilangan Noise

Hasil dari deteksi titik-titik wajah untuk tiap citra dalam satu sekuen kadang mengalami perbedaan koordinat walaupun tidak terjadi pergerakan pada wajah dan bagian-bagian wajah. Hal ini bisa disebabkan karena terdapat noise pada image. Mengingat ASM adalah *local method*, walaupun hanya terdapat sedikit noise, akan mempengaruhi *fitting* model wajah pada citra.

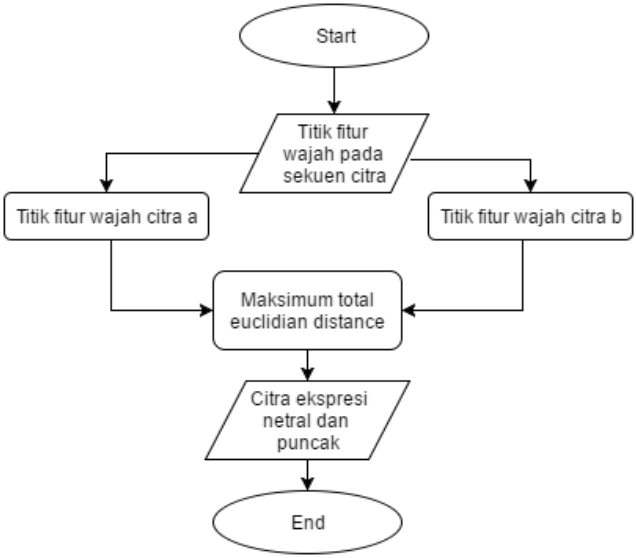
Untuk mengurangi noise pada hasil pergerakan titik wajah, digunakan metode *averaging* pada titik tersebut untuk tiap k-sekuen citra. Gambar 3.5 menunjukkan diagram alir dari proses penghilangan noise.



Gambar 4.5 Diagram alir penghilangan noise

4.3.1.3 Deteksi Ekspresi Puncak

Setelah didapatkan titik-titik fitur wajah, dilakukan deteksi ekspresi puncak pada sekuen citra. Ekspresi puncak merupakan ekspresi dengan perubahan titik-titik fitur wajah terbesar dari ekspresi netral. Sehingga untuk mendapatkan citra ekpresi puncak, digunakan jarak *euclidian* pada sekuen citra dengan citra ekspresi netral. Citra dengan jarak terbesar terhadap ekspresi netral merupakan citra ekspresi puncak. Gambar 3.7 menunjukkan diagram alir deteksi ekspresi puncak.



Gambar 4.6 Diagram alir deteksi ekspresi puncak

4.3.1.4 Fitur Perpindahan Ekspresi Netral-Puncak

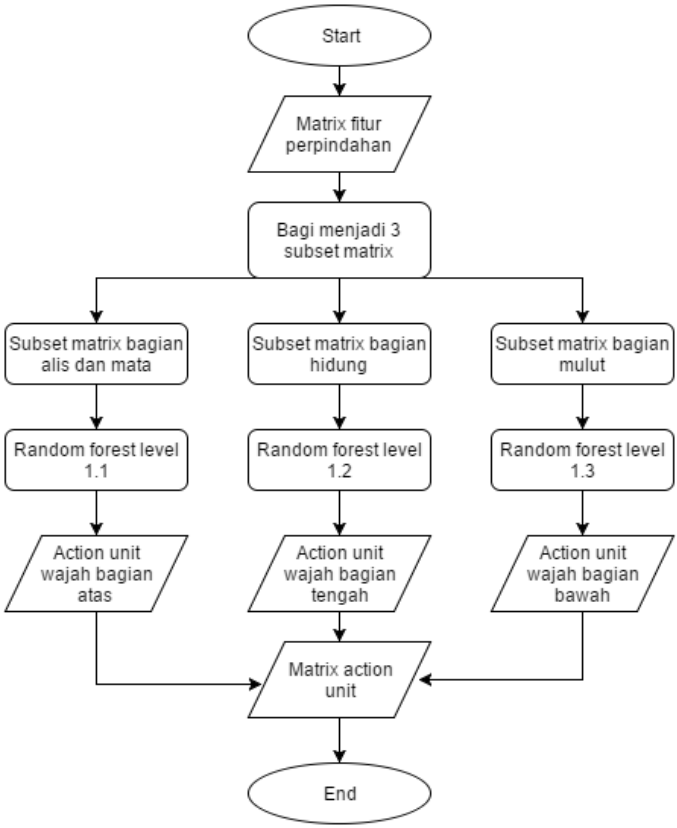
Tahap perhitungan fitur perpindahan ini merupakan tahap terakhir dari ekstraksi fitur. Setelah didapatkan 2 citra pada sekuen citra yaitu citra ekspresi netral, dan citra ekspresi puncak, dimana masing-masing citra terdiri dari 77 titik fitur wajah, dilakukan perhitungan rasio vektor perpindahan titik pada *Region of Interest* (ROI) tiap bagian wajah dengan keliling ROI tiap bagian wajah pada citra ekspresi netral. Vektor rasio tersebut berukuran 1x154 untuk satu data. Terdapat 90 data sehingga dihasilkan matrix 90x154. Matrix tersebut yang akan menjadi fitur perpindahan ekspresi netral-puncak yang akan digunakan untuk tahap pengenalan *Action Unit.* Gambar 3.8 menunjukkan proses untuk mendapatkan fitur perpindahan.

Gambar 4.7 Proses ekstraksi fitur perpindahan

4.3.2 Tahap Klasifikasi *Action Unit*

Pada bagian ini, fitur perpindahan yang didapatkan saat tahap ekstraksi fitur di-*training* menggunakan random forest level 1 untuk mengklasifikasikan *Action Unit* pada tiap bagian wajah. Matrix fitur perpindahan yang dihasilkan pada tahap ekstraksi fitur dibagi menjadi 72x154 untuk training random forest level 1 dan 18x154 untuk testing random forest level 1. *Action unit* yang dihasilkan berukuran 90x3 , dimana untuk tiap data dihasilkan 3 *action unit* yaitu *action unit* bagian atas (mata dan alis), *action unit* bagian tengah (hidung), dan *action unit* bagian bawah (mulut). *Action unit* bagian atas terdiri dari 4 nilai yaitu *AU*1*, AU*4*, AU*5*, dan AU*6. *Action unit* bagian tengah terdiri dari 2 nilai yaitu *AU*9 dan *AU*13. *Action unit* bagian bawah terdiri dari 5 nilai yaitu *AU*27*, AU*12*, AU*24*, AU*23 *dan AU*20. Gambar 3.9 menunjukkan alur

pembuatan model random forest level 1 untuk mengklasifikasikan *Action Unit.*

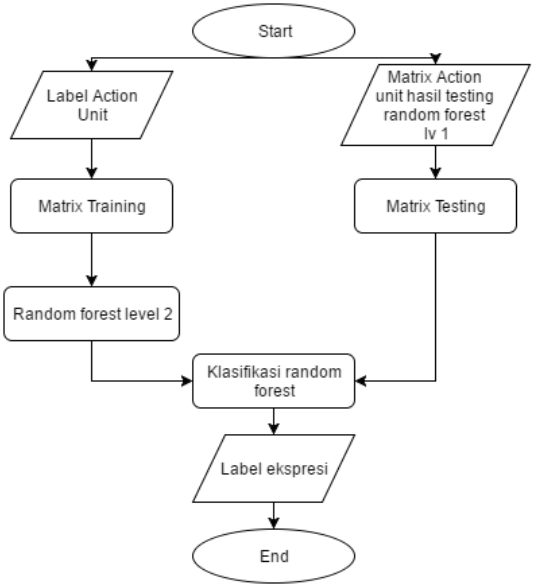


Gambar 4.8 Alur pembuatan model random forest level 1

4.3.3 Tahap Klasifikasi Ekspresi

Pada bagian ini, tiap matrix label *Action Unit* yang sudah ada pada dataset digunakan sebagai input untuk training model random forest level 2 dan hasil testing dari random forest level 1 digunakan sebagai testing random forest level 2. Label *Action Unit* yang digunakan untuk training berukuran 72x3 dan hasil testing

random forest level 1 yang digunakan untuk testing random forest level 2 berukuran 18x3. Random forest level 2 ini yang akan mengklasifikasikan 6 ekspresi dasar manusia, yaitu senang, sedih, terkejut, marah, jijik, dan takut. Gambar 3.9 menunjukkan alur pembuatan model random forest level 2.



Gambar 4.9 Alur pembuatan model random forest level 2

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab implementasi berisi pembahasan mengenai implementasi perangkat lunak berdasarkan perancangan yang telah dibuat. Tahap perancangan merupakan tahap dasar dari implementasi perangkat lunak.

5.1 Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat Keras
   a. Processor: Intel® Core™ i5-2410M CPU @ 2.30GHz
   b. Memory (RAM) : 4.00 GB
   c. Tipe Sistem: 64-bit
2. Perangkat Lunak
   a. Sistem operasi : *Window*s *Embedded* 8.1 64 bit.
   b. *C++*.
   c. Visual Studio 2013.
   d. Pustaka *stasm4.1, opencv 2.4.10.*

5.2 Implementasi Modul *Face Detector*

Modul ini adalah bagian dari sistem yang berfungsi untuk mendeteksi wajah. Modul *FaceDet* ini akan menghasilkan parameter deteksi wajah berupa titik pusat area deteksi (x,y), serta tinggi dan lebar dari area deteksi. Gambar 4.1 menujukkan kode program yang berfungsi melakukan deteksi wajah.

```
void FaceDet::OpenFaceDetector_(const char* datadir,
void*)
{
    OpenDetector(facedet_g,
"haarcascade_frontalface_alt2.xml",  datadir);
}
void DetectFaces(
```

```
    vec_DetPar&  detpars,
    const Image& img,
    int minwidth)
{
    CV_Assert(!facedet_g.empty());
    int leftborder = 0, topborder = 0;
    Image bordered_img(BORDER_FRAC == 0?
    img: EnborderImg(leftborder, topborder, img));
    Image equalized_img; cv::equalizeHist(bordered_img,
equalized_img);

    CV_Assert(minwidth >= 1 && minwidth <= 100);
    const int minpix =MAX(minwidth <= 5? 70: 100,
cvRound(img.cols * minwidth / 100.));

    static const double SCALE_FACTOR   = 1.1;
    static const int    MIN_NEIGHBORS  = 3;
    static const int    DETECTOR_FLAGS = 0;

    vec_Rect facerects = Detect(equalized_img, facedet_g,
NULL, SCALE_FACTOR, MIN_NEIGHBORS, DETECTOR_FLAGS,
minpix);

    detpars.resize(NSIZE(facerects));
    for (int i = 0; i < NSIZE(facerects); i++)
    {
        Rect* facerect = &facerects[i];
        DetPar detpar;
        detpar.x = facerect->x + facerect->width / 2.;
        detpar.y = facerect->y + facerect->height / 2.;
        detpar.x -= leftborder;
        detpar.y -= topborder;
        detpar.width  = double(facerect->width);
        detpar.height = double(facerect->height);
        detpar.yaw = 0;
        detpar.eyaw = EYAW00;
        detpars[i] = detpar;
    }
}
```

Gambar 5.1 Kode program *FaceDetector*

5.3 Implementasi Modul *GetLandmarkPoint*

Modul ini berperan untuk menghasilkan 77 titik wajah yang sudah dilakukan avaraging untuk tiap k frame. Gambar 4.2 menunjukkan potongan kode program representasi GetLandmarkPoint.

```
void GetLandmarksPoint(cv::Mat_<unsigned char> &img)
{
	stasm_force_points_into_image(landmarks, img.cols,
img.rows);

	for (int i = 0; i < stasm_NLANDMARKS; i++)
	{
		img(cvRound(landmarks[i * 2 + 1]),
cvRound(landmarks[i * 2])) = 255;
		for (int j = 0; j < FrameAveraging - 1; j++)
		{
			landmarkFaceRAW[j][i] =
landmarkFaceRAW[j + 1][i];
		}
		landmarkFaceRAW[FrameAveraging - 1][i].y =
cvRound(landmarks[i * 2 + 1]);
		landmarkFaceRAW[FrameAveraging - 1][i].x =
cvRound(landmarks[i * 2]);

		landmarkFace[i].y = 0;
		landmarkFace[i].x = 0;
		for (int j = 0; j < FrameAveraging; j++)
		{
			landmarkFace[i].y +=
landmarkFaceRAW[j][i].y;
			landmarkFace[i].x +=
landmarkFaceRAW[j][i].x;
		}

		landmarkFace[i].y /= FrameAveraging;
		landmarkFace[i].x /= FrameAveraging;
		frameCounter++;
	}
}
```

Gambar 5.2 Kode program *GetLandmarkPoint*

5.4 Implementasi Modul *GetBestPeak*

Modul ini digunakan untuk untuk mencari citra ekspresi puncak dari sekuen citra. Gambar 4.3 menunjukkan potongan kode pada *GetBestPeak.*

```
float GetBestPeak(float neutralLandmarks[], float
peakLandmarks[])
{
  float peakValue = 0;
    for (int i = 0; i < stasm_NLANDMARKS; i++)
      {
        if (i >= 16)
        {
          peakValue += sqrt(pow(neutralLandmarks[i * 2] -
peakLandmarks[i * 2], 2) + pow(neutralLandmarks[i * 2 + 1]
- peakLandmarks[i * 2 + 1], 2));
        }
      }
    return peakValue;
}
If(GetBestPeak(neutralLandmarks, peakLandmarks)>=
bestPeak)
{
  bestPeak = GetBestPeak(neutralLandmarks,
peakLandmarks);
  peakIndex = i;
}
```

Gambar 5.3 Kode program modul *GetBestPeak*.

5.5 Implementasi Modul *GetDVNPTrainingFeatures*

Modul ini digunakan untuk melakukan ekstraksi fitur perpindahan dari citra ekspresi netral ke citra ekspresi puncak untuk data training. Gambar 4.4 menunjukkan fungsi-fungsi yang digunakan pada modul *GetDVNPTrainingFeatures*.

```
void GetDeviderTrainingFeatures(float landmarks[])
{
  devTrainingFeatures[4] = { 0 };
  for (int i = 0; i < stasm_NLANDMARKS; i++)
  {
    if (i > 16 && i <= 21)
```

```
    {
      devTrainingFeatures[0] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 22 && i <= 27)
    {
      devTrainingFeatures[0] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 30 && i <= 37)
    {
      devTrainingFeatures[1] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 40 && i <= 47)
    {
       devTrainingFeatures[1] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 48 && i <= 58)
    {
       devTrainingFeatures[2] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 59 && i <= 65)
    {
       devTrainingFeatures[3] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
    else if (i > 72 && i <= 76)
    {
       devTrainingFeatures[3] += sqrt(pow(landmarks[i * 2]
- landmarks[(i - 1) * 2], 2) + pow(landmarks[i * 2 + 1] -
landmarks[(i - 1) * 2 + 1], 2));
    }
  }
```

```
  deviderTrainingFeatures[0] = devTrainingFeatures[0] / 2;
  deviderTrainingFeatures[1] = devTrainingFeatures[1] / 2;
  deviderTrainingFeatures[2] = devTrainingFeatures[2];
  deviderTrainingFeatures[3] = devTrainingFeatures[3];
}

void GetDVNPTrainingFeatures(float neutralLandmarks[],
float peakLandmarks[], int dataIndex, int kelas) // 0:
terkejut, 1:senang, 2:sedih, 3:marah, 4:jijik, 5:takut
{
  int mouthIndex = 0;
  int noseIndex = 0;
  int eyeIndex = 0;
  int eyebrowIndex = 0;
  GetDeviderTrainingFeatures(neutralLandmarks);
  for (int i = 0; i < stasm_NLANDMARKS; i++)
  {
    DVNPTrainingFeatures[i * 2][dataIndex] =
(peakLandmarks[i * 2] - neutralLandmarks[i * 2]) /
deviderTrainingFeatures[deviderIndex[i]]) * 10;
    DVNPTrainingFeatures[i * 2 + 1][dataIndex] =
(peakLandmarks[i * 2 + 1] - neutralLandmarks[i * 2 + 1]) /
deviderTrainingFeatures[deviderIndex[i]]) * 10;
    if (i >= 16 && i <= 27)
    {
      DVNPEyebrowsTrainingFeatures[eyebrowIndex][dataInde
x] = DVNPTrainingFeatures[i * 2][dataIndex] * 1000;
      eyebrowIndex++;

      DVNPEyebrowsTrainingFeatures[eyebrowIndex][dataInde
x] = DVNPTrainingFeatures[i * 2 + 1][dataIndex] * 1000;
      eyebrowIndex++;

      if (kelas == 1)
      {
        DVNPTrainingClass[3][dataIndex] = 0;
      }
      else if (kelas == 4)
      {
        DVNPTrainingClass[3][dataIndex] = 1;
      }
```

```
    }
    else if (i >= 28 && i <= 47)
    {
      DVNPEyesTrainingFeatures[eyeIndex][dataIndex] =
DVNPTrainingFeatures[i * 2][dataIndex] * 1000;
      eyeIndex++;
      DVNPEyesTrainingFeatures[eyeIndex][dataIndex] =
DVNPTrainingFeatures[i * 2 + 1][dataIndex] * 1000;
      eyeIndex++;
    }
    else if (i >= 48 && i <= 58)
    {
      DVNPNoseTrainingFeatures[noseIndex][dataIndex] =
DVNPTrainingFeatures[i * 2][dataIndex] * 1000;
      noseIndex++;
      DVNPNoseTrainingFeatures[noseIndex][dataIndex] =
DVNPTrainingFeatures[i * 2 + 1][dataIndex] * 1000;
      noseIndex++;

      if (kelas == 5)
      {
        DVNPTrainingClass[1][dataIndex] = 1;
      }
      else
      {
        DVNPTrainingClass[1][dataIndex] = 0;
      }
    }
    else if (i >= 59 && i <= 76)
    {
      DVNPMouthTrainingFeatures[mouthIndex][dataIndex] =
DVNPTrainingFeatures[i * 2][dataIndex] * 1000;
      mouthIndex++;
      DVNPMouthTrainingFeatures[mouthIndex][dataIndex] =
DVNPTrainingFeatures[i * 2 + 1][dataIndex] * 1000;
      mouthIndex++;

      if (kelas == 0)
      {
        DVNPTrainingClass[0][dataIndex] = 0;
      }
      else if (kelas == 1)
```

```
      {
         DVNPTrainingClass[0][dataIndex] = 1;
       }
       else if (kelas == 2)
       {
         DVNPTrainingClass[0][dataIndex] = 2;
       }
       else if (kelas == 3)
       {
         DVNPTrainingClass[0][dataIndex] = 3;
       }
     }
   }
}
```

Gambar 5.4 Modul *GetDVNPTrainingFeatures*

Fungsi GetDeviderTrainingFeatures dengan parameter titik landmark pada citra ekspresi netral digunakan untuk mendapatkan pembagi dari perpindahan titik landmark wajah yang berupa keliling dari ROI tiap bagian wajah (bibir, hidung, mata, dan alis). Pada fungsi GetDVNPTrainingFeatures, vektor perpindahan titik landmark wajah dibagi menjadi 4 untuk tiap bagian wajah, kemudian untuk tiap subset vektor perpindahan tersebut dibagi dengan nilai devider yang dihasilkan dari fungsi GetDeviderTrainingFeatures. Pembagian tersebut menghasilkan vektor fitur yang siap untuk ditraining. Selain itu, fungsi GetDVNPTrainingFeatures juga menyimpan kelas data untuk tiap dataIndex.

5.6 Implementasi Modul *GetDVNPTestingFeatures*

Modul ini digunakan untuk melakukan ekstraksi fitur perpindahan dari citra ekspresi netral ke citra ekspresi puncak untuk data testing.

```
void GetDVNPTestingFeatures(float neutralLandmarks[],
float peakLandmarks[]) // 0: terkejut, 1:senang, 2:sedih,
3:marah, 4:jijik
{
```

```
  int mouthIndex = 0;
  int noseIndex = 0;
  int eyeIndex = 0;
  int eyebrowIndex = 0;
  GetDeviderTrainingFeatures(neutralLandmarks);
  for (int i = 0; i < stasm_NLANDMARKS; i++)
  {
    DVNPTestingFeatures[i * 2] = (peakLandmarks[i * 2] -
neutralLandmarks[i * 2])/
deviderTrainingFeatures[deviderIndex[i]]) * 10;
    DVNPTestingFeatures[i * 2 + 1] = (peakLandmarks[i * 2
+ 1] - neutralLandmarks[i * 2 + 1])/
deviderTrainingFeatures[deviderIndex[i]]) * 10;

    if (i >= 16 && i <= 27)
    {
      DVNPEyebrowsTestingFeatures[eyebrowIndex] =
DVNPTestingFeatures[i * 2] * 1000;
      eyebrowIndex++;
      DVNPEyebrowsTestingFeatures[eyebrowIndex] =
DVNPTestingFeatures[i * 2 + 1] * 1000;
      eyebrowIndex++;
    }
    else if (i >= 28 && i <= 47)
    {
      DVNPEyesTestingFeatures[eyeIndex] =
DVNPTestingFeatures[i * 2] * 1000;
      eyeIndex++;
      DVNPEyesTestingFeatures[eyeIndex] =
DVNPTestingFeatures[i * 2 + 1] * 1000;
      eyeIndex++;
    }
    else if (i >= 48 && i <= 58)
    {
      DVNPNoseTestingFeatures[noseIndex] =
DVNPTestingFeatures[i * 2] * 1000;
      noseIndex++;
      DVNPNoseTestingFeatures[noseIndex] =
DVNPTestingFeatures[i * 2 + 1] * 1000;
      noseIndex++;
    }
    else if (i >= 59 && i <= 76)
```

```
    {
      DVNPMouthTestingFeatures[mouthIndex] =
DVNPTestingFeatures[i * 2] * 1000;
      mouthIndex++;
      DVNPMouthTestingFeatures[mouthIndex] =
DVNPTestingFeatures[i * 2 + 1] * 1000;
      mouthIndex++;
    }
  }
}
```

Gambar 5.5 Kode program *GetDVNPTestingFeatures*

Fungsi GetDVNPTestingFeatures hampir sama seperti GetDVNPTrainingFeatures, hanya saja pada modul ini tidak terdapat parameter kelas untuk tiap data Index. Fungsi dari modul ini hanya menyimpan fitur perpindahan untuk tiap subset vektor fitur. Subset vektor ini yang akan ditest pada model random forest yang sudah ditraining dengan subset vektor training.

5.7 Implementasi Modul *TwoFoldRandomForestAlgorithm*

Modul ini digunakan untuk membentuk model random forest dengan parameter vektor perpindahan, nilai random fitur, dan nilai random data. Algoritma tree yang digunakan untuk tiap random tree adalah C4.5 dengan parameter pencarian *split point* menggunakan nilai gini indeks terkecil. Dalam modul ini terdapat beberapa fungsi untuk pembentukan model random forest, diantaranya *TreeNode, GiniResult, ComputeGiniIndex, ComputeBestSplit, DecisionTree, RandomForestAlgorithm.*

```
struct TreeNode
{
  //result
  bool res_isLeaf;
  int res_classResult;

  //post
  int post_leftIndex;
  int post_rightIndex;
```

```
  int post_splitPoint;
  int post_featuresIndex;
  //pre
  bool pre_isFill;
  int pre_dataSize;
  int pre_kelas[100];
  int pre_features[100][100];

  //initialisation.
  TreeNode() : pre_isFill(false), res_isLeaf(false),
res_classResult(-1), post_featuresIndex(-1) {}
};

// Inisialisasi random forest.
TreeNode RandomForest[25][100]; // 25 pohon, 200 node.

struct GiniResult
{
  int splitPoint;
  float giniIndex;
};

float ComputeGiniIndex(int variableSplit[][2], int
kelasSize)
{
  //variablesplit[kelas][splitAmount] (splitAmount = 2
(x>n dan x<n)).
  //kelasSize = juml kelas. jika ada 4 kelas => 0,1,2,3.

  float giniSplit[2] = { 0 };
  float sumPart[2] = { 0 };
  float GINI = 0;
  for (int i = 0; i < 2; i++)
  {
    for (int j = 0; j < kelasSize; j++)
      {
        sumPart[i] += variableSplit[j][i];
      }
  }

  giniSplit[0] = 1;
  giniSplit[1] = 1;
```

```
  for (int i = 0; i < 2; i++)
  {
    for (int j = 0; j < kelasSize; j++)
    {
       if (sumPart[i] != 0)
       {
         giniSplit[i] -= pow(variableSplit[j][i] /
sumPart[i], 2);
       }
    }
  }

  for (int i = 0; i < 2; i++)
  {
    GINI += giniSplit[i] * (sumPart[i] / (sumPart[0] +
sumPart[1]));
  }
  return GINI;
}

GiniResult ComputeBestSplit(int var[], int kelas[], int
dataSize, int kelasSize) // contoh kelas =0,1,2,3 mulai
dari 0, dan kelasSize = 4.
{
  //Initial result
  GiniResult newResult;

  //Get Distinct Value
  int *varDistinctTemp = new int[dataSize];
  int *varDistinct = new int[dataSize];
  int varDistinctIndex = 0;

  for (int i = 0; i < dataSize; i++)
  {
    varDistinctTemp[i] = var[i];
  }

  sort(varDistinctTemp, varDistinctTemp + dataSize);

  for (int i = 0; i < dataSize - 1; i++)
  {
    if (varDistinctTemp[i] != varDistinctTemp[i + 1])
```

```
    {
       varDistinct[varDistinctIndex] = varDistinctTemp[i];
       varDistinctIndex++;
       if (i == dataSize - 2)
       {
         varDistinct[varDistinctIndex] = varDistinctTemp[i
+ 1];
         varDistinctIndex++;
       }
     }
  }

  //Get Split Point.
  int *splitPoint = new int[varDistinctIndex + 1];
  splitPoint[0] = varDistinct[0] - 5;
  splitPoint[varDistinctIndex] =
varDistinct[varDistinctIndex] + 5;
  for (int i = 1; i < varDistinctIndex; i++)
  {
    splitPoint[i] = (varDistinct[i - 1] + varDistinct[i])
/ 2;
  }

  //Get best split & best gini.
  int bestSplit = splitPoint[0];
  float bestGINI = 1;

  for (int i = 0; i <= varDistinctIndex; i++)
  {
    int variableSplit[10][2] = { 0 };

    for (int j = 0; j < dataSize; j++)
    {
      if (var[j] <= splitPoint[i])
      {
        variableSplit[kelas[j]][0]++;
      }
      else
      {
        variableSplit[kelas[j]][1]++;
      }
    }
```

```
    float giniResult = ComputeGiniIndex(variableSplit,
kelasSize);
    if (giniResult < bestGINI)
    {
      bestGINI = giniResult;
      bestSplit = splitPoint[i];
    }
  }

  newResult.splitPoint = bestSplit;
  newResult.giniIndex = bestGINI;

  delete[] varDistinctTemp;
  delete[] varDistinct;
  delete[] splitPoint;

  return newResult;
}

void DecisionTrees(int iter,bool dataON[], bool
featuresON[], int featuresSize, int dataSize, int
kelasSize, int featuresType) // featuresON
={true,false,true...}, featuresSize = total features,
misal mulut=36,dll., featuresType ->0 = mulut, 1=
{
  int varIndex = 0; // features size after randomize.
  int dataIndex = 0;

  // Parsing features that used to var.
  for (int i = 0; i < featuresSize; i++)
  {
    //if (featuresON[i] == true)
    //{
    dataIndex = 0;
    for (int j = 0; j < dataSize; j++)
    {
      if (dataON[j] == true)
      {
        if (featuresType == 0)
       {
```

```
RandomForest[iter][0].pre_features[varIndex][dataIndex] =
(int)DVNPMouthTrainingFeatures[i][j]; // featuresType = 0
RandomForest[iter][0].pre_kelas[dataIndex] =
DVNPTrainingClass[featuresType][j];
       }
       else if (featuresType == 1)
       {
RandomForest[iter][0].pre_features[varIndex][dataIndex] =
(int)DVNPNoseTrainingFeatures[i][j]; // featuresType = 1
RandomForest[iter][0].pre_kelas[dataIndex] =
DVNPTrainingClass[featuresType][j];
       }
       else if (featuresType == 2)
       {
RandomForest[iter][0].pre_features[varIndex][dataIndex] =
(int)DVNPEyesTrainingFeatures[i][j]; // featuresType = 2
RandomForest[iter][0].pre_kelas[dataIndex] =
DVNPTrainingClass[featuresType][j];
       }
       else if (featuresType == 3)
       {
RandomForest[iter][0].pre_features[varIndex][dataIndex] =
(int)DVNPEyebrowsTrainingFeatures[i][j]; // featuresType =
3
RandomForest[iter][0].pre_kelas[dataIndex] =
DVNPTrainingClass[featuresType][j];
       }
       dataIndex++;
     }
    }
   varIndex++;
   //}
  }

  RandomForest[iter][0].pre_isFill = true;
  RandomForest[iter][0].pre_dataSize = dataIndex;

  for (int nodeIndex = 0; nodeIndex < varIndex;
nodeIndex++) // harusnya sampai maxdepth treenya. asumsi
maxdepth = maxfeatures.
  {
    if (RandomForest[iter][nodeIndex].pre_isFill == true)
```

```
    {
      // Check apakah node adalah leaf.
       bool isLeaf = true;
       int classRes = -1;
       for (int i = 0; i <
RandomForest[iter][nodeIndex].pre_dataSize - 1; i++)
  {
    if (RandomForest[iter][nodeIndex].pre_kelas[i] !=
RandomForest[iter][nodeIndex].pre_kelas[i + 1])
    {
      isLeaf = false;
    }
    else
    {
      classRes =
RandomForest[iter][nodeIndex].pre_kelas[i];
    }
  }
  if (isLeaf == true)
  {
    RandomForest[iter][nodeIndex].res_isLeaf = true;
    if (RandomForest[iter][nodeIndex].pre_dataSize == 1)
    {
      RandomForest[iter][nodeIndex].res_classResult =
RandomForest[iter][nodeIndex].pre_kelas[0];
    }
    else
    {
      RandomForest[iter][nodeIndex].res_classResult =
classRes;
    }
  }
  // If not leaf.
  if (RandomForest[iter][nodeIndex].res_isLeaf == false)
  {
    GiniResult nodeResult;
    nodeResult.giniIndex = 1;
    for (int i = 0; i < varIndex; i++)
    {
      if (featuresON[i] == true)
      {
```

```
        if
(ComputeBestSplit(RandomForest[iter][nodeIndex].pre_featur
es[i], RandomForest[iter][nodeIndex].pre_kelas,
RandomForest[iter][nodeIndex].pre_dataSize,
kelasSize).giniIndex < nodeResult.giniIndex)
        {
          nodeResult =
ComputeBestSplit(RandomForest[iter][nodeIndex].pre_feature
s[i], RandomForest[iter][nodeIndex].pre_kelas,
RandomForest[iter][nodeIndex].pre_dataSize, kelasSize);
          RandomForest[iter][nodeIndex].post_featuresIndex
= i;
          RandomForest[iter][nodeIndex].post_splitPoint =
nodeResult.splitPoint;
        }
      }
    }

    for (int i = nodeIndex + 1; i < varIndex - 1; i++)
    {
      if (RandomForest[iter][i].pre_isFill == false)
      {
        RandomForest[iter][nodeIndex].post_leftIndex = i;
        RandomForest[iter][nodeIndex].post_rightIndex=i+1;

        RandomForest[iter][i].pre_isFill = true;
        RandomForest[iter][i + 1].pre_isFill = true;

        int dataSizeLeft = 0;
        int dataSizeRight = 0;

        // Isi pre datasize, kelas, & features.
        for (int j = 0; j <
RandomForest[iter][nodeIndex].pre_dataSize; j++)
        {
          if
(RandomForest[iter][nodeIndex].pre_features[RandomForest[i
ter][nodeIndex].post_featuresIndex][j] <=
RandomForest[iter][nodeIndex].post_splitPoint)
          {
            for (int k = 0; k < varIndex; k++)
            {
```

```
RandomForest[iter][i].pre_features[k][dataSizeLeft] =
RandomForest[iter][nodeIndex].pre_features[k][j];
RandomForest[iter][i].pre_kelas[dataSizeLeft] =
RandomForest[iter][nodeIndex].pre_kelas[j];
            }
          dataSizeLeft++;
          }
          else
          {
            for (int k = 0; k < varIndex; k++)
            {
RandomForest[iter][i + 1].pre_features[k][dataSizeRight] =
RandomForest[iter][nodeIndex].pre_features[k][j];
RandomForest[iter][i + 1].pre_kelas[dataSizeRight] =
RandomForest[iter][nodeIndex].pre_kelas[j];
            }
          dataSizeRight++;
          }
        }
        RandomForest[iter][i].pre_dataSize = dataSizeLeft;
        RandomForest[iter][i + 1].pre_dataSize =
dataSizeRight;

        i = varIndex;
      }
    }
  }
}
}
}
```

```
void RFMouth()
{
  int featuresType = 0;
  int kelasSize = 4;
  int dataSize = 60;
  int iterasi = 100;
  bool *dataON = new bool[dataSize];
  bool featureON[101][50];

  for (int i = 0; i < iterasi; i++)
  {
    int randomCount = 0;
    for (int j = 0; j < 50; j++)
    {
        if (i == 0)
        {
          featureON[i][j] = true;
          randomCount++;
        }
        else
        {
          int random = rand() % 100;
          if (random <= 49)
          {
             featureON[i][j] = true;
             randomCount++;
          }
          else
          {
             featureON[i][j] = false;
          }
        }
    }
    //cout << "Random Count " << i << " : " << randomCount
<< endl;
  }

  for (int i = 0; i < dataSize; i++)
  {
    dataON[i] = true;
  }
```

```
  for (int iter = 0; iter < iterasi; iter++)
  {
    TwofoldDecisionTree(iter, dataON, featureON[iter], 36,
dataSize, kelasSize, featuresType);
  }

  delete[] dataON;
}

void RFNose()
{
  int featuresType = 1;
  int kelasSize = 2;
  int dataSize = 60;
  int iterasi = 100;
  bool *dataON = new bool[dataSize];
  bool featureON[101][50];

  for (int i = 0; i < iterasi; i++)
  {
    int randomCount = 0;
    for (int j = 0; j < 50; j++)
    {
      if (i == 0)
      {
       featureON[i][j] = true;
       randomCount++;
      }
      else
      {
       int random = rand() % 100;
       if (random <= 49)
       {
         featureON[i][j] = true;
         randomCount++;
       }
       else
       {
         featureON[i][j] = false;
       }
      }
    }
```

```
//cout << "Random Count " << i << " : " << randomCount <<
endl;
  }
  for (int i = 0; i < dataSize; i++)
  {
    dataON[i] = true;
  }

  for (int iter = 0; iter < iterasi; iter++)
  {
    TwofoldDecisionTree(iter, dataON, featureON[iter], 22,
dataSize, kelasSize, featuresType);
  }

  delete[] dataON;
}

void RFEyebrows()
{
  int featuresType = 3;
  int kelasSize = 4;
  int dataSize = 60;
  int iterasi = 100;
  bool *dataON = new bool[dataSize];
  bool featureON[101][50];

  for (int i = 0; i < iterasi; i++)
  {
    int randomCount = 0;
    for (int j = 0; j < 50; j++)
    {
      if (i == 0)
      {
       featureON[i][j] = true;
       randomCount++;
      }
      else
      {
       int random = rand() % 100;
       if (random <= 49)
       {
         featureON[i][j] = true;
```

```
          randomCount++;
        }
        else
        {
          featureON[i][j] = false;
        }
       }
    }
    //cout << "Random Count " << i << " : " << randomCount
<< endl;
  }

  for (int i = 0; i < dataSize; i++)
  {
    dataON[i] = true;
  }
  for (int iter = 0; iter < iterasi; iter++)
  {
    TwofoldDecisionTree(iter, dataON, featureON[iter], 24,
dataSize, kelasSize, featuresType);
  }
  delete[] dataON;
}

void RFExpression()
{
  int featuresType = 4;
  int kelasSize = 5;
  int dataSize = 60;
  bool *dataON = new bool[dataSize];
  bool featureON[1][10];

  for (int i = 0; i < 10; i++)
  {
    featureON[0][i] = true;
  }

  for (int i = 0; i < dataSize; i++)
  {
    dataON[i] = true;
  }
```

```
  TwofoldDecisionTree(0, dataON, featureON[0], 3,
dataSize, kelasSize, featuresType);
}
```

Gambar 5.6 Kode program *TwoFoldRandomForestAlgorithm*

Model random forest yang di training akan disimpan pada TreeNode RandomForest[][]. Fungsi *ComputeGiniIndex* digunakan untuk mendapatkan nilai giniIndex untuk tiap *split point*. Fungsi *ComputeBestSplit* digunakan untuk mendapatkan nilai *split point* terbaik untuk tiap variable data. Fungsi *DecisionTree* digunakan untuk mendapatkan random tree untuk tiap iterasi random forest. Fungsi *RandomForestAlgorithm* digunakan untuk melakukan desain pada model random forest, yaitu dengan merandom fitur dan merandom data untuk tiap random tree.

5.8 Implementasi Modul *Testing*

Modul ini digunakan untuk mendapatkan hasil dari random forest pada iterasi ke-iter jika dimasukkan satu data.

```
int TestingMouthAU(int iter)
{
  int nodeIndex = 0;
  int AUIndex = 0;
  while
(TwofoldRandomForest[AUIndex][iter][nodeIndex].res_isLeaf
!= true)
  {
    if
((int)DVNPMouthTestingFeatures[TwofoldRandomForest[AUIndex
][iter][nodeIndex].post_featuresIndex] <=
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_splitPo
int)
    {
      nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftInd
ex;
    }
    else
```

```
    {
      nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightIn
dex;
    }
  }
 return
TwofoldRandomForest[AUIndex][iter][nodeIndex].res_classRes
ult;
}

int TestingNoseAU(int iter)
{
  int nodeIndex = 0;
  int AUIndex = 1;
  while
(TwofoldRandomForest[AUIndex][iter][nodeIndex].res_isLeaf
!= true)
  {
    if
((int)DVNPNoseTestingFeatures[TwofoldRandomForest[AUIndex]
[iter][nodeIndex].post_featuresIndex] <=
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_splitPo
int)
    {
      nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftInd
ex;
    }
    else
    {
      nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightIn
dex;
    }
  }
  return
TwofoldRandomForest[AUIndex][iter][nodeIndex].res_classRes
ult;
}

int TestingEyebrowsAU(int iter)
```

```
{
  int nodeIndex = 0;
  int AUIndex = 3;
  while
(TwofoldRandomForest[AUIndex][iter][nodeIndex].res_isLeaf
!= true)
  {
    if
((int)DVNPEyebrowsTestingFeatures[TwofoldRandomForest[AUIn
dex][iter][nodeIndex].post_featuresIndex] <=
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_splitPo
int)
    {
      if
(TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftIn
dex == -1)
      {
       break;
      }
      else
      {
       nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftInd
ex;
      }
    }
    else
    {
      if
(TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightI
ndex == -1)
      {
       break;
      }
      else
      {
       nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightIn
dex;
      }
    }
  }
```

```
  return
TwofoldRandomForest[AUIndex][iter][nodeIndex].res_classRes
ult;
}

int TestingExpression()
{
  int nodeIndex = 0;
  int AUIndex = 4;
  int iter = 0;
  while
(TwofoldRandomForest[AUIndex][iter][nodeIndex].res_isLeaf
!= true)
  {
    if
((int)DVNPTestingFeaturesRFTwoFold[TwofoldRandomForest[AUI
ndex][iter][nodeIndex].post_featuresIndex] <=
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_splitPo
int)
    {
      if
(TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftIn
dex == -1)
      {
       break;
      }
      else
      {
       nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_leftInd
ex;
      }
    }
    else
    {
      if
(TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightI
ndex == -1)
      {
       break;
      }
      else
```

```
      {
       nodeIndex =
TwofoldRandomForest[AUIndex][iter][nodeIndex].post_rightIn
dex;
      }
    }
  }
  return
TwofoldRandomForest[AUIndex][iter][nodeIndex].res_classRes
ult;
}
```

Gambar 5.7 Kode program *Testing*

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan mengenai rangkaian uji coba perangkat lunak. Pada bab ini juga akan dibahas mengenai perhitungan akurasi dari perangkat lunak yang dibuat. Tujuan dari uji coba ini adalah untuk mengetahui apakah model yang telah dihasilkan sistem mampu mengklasifikasikan 6 ekspresi dasar manusia dari data sekuen citra dengan benar, mengetahui berapa nilai koefisien k pada parameter metode *averaging* untuk mengoptimalkan tahap ekstraksi fitur, membadingkan metode *random forest* dan *twofold random forest,* serta mencari jumlah kelas ekspresi yang paling optimal. Selain itu juga pada bab ini akan dipaparkan pembahasan yang meliputi lingkungan uji coba, data uji coba, skenario uji coba, hasil uji coba, dan evaluasi.

6.1 Lingkungan Uji Coba

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat Keras
   a. Processor: Intel® Core™ i5-2410M CPU @ 2.30GHz
   b. Memory (RAM) : 4.00 GB
   c. Tipe Sistem: 64-bit
2. Perangkat Lunak
   a. Sistem operasi : Windows Embedded 8.1 64 bit.
   b. *C++*.
   c. Visual Studio 2013.
   d. Pustaka stasm4.1, opencv 2.4.10.

6.2 Data Uji Coba

Data uji coba yang digunakan pada perangkat lunak ini adalah data sekuen citra ekspresi yang tersedia *online* (http://www.consortium.ri.cmu.edu/data/ck/) yang sudah

dijelaskan pada subbab 3.2. Dataset terdiri dari 90 sekuen citra dimana tiap sekuen terdiri dari 8-60 citra.

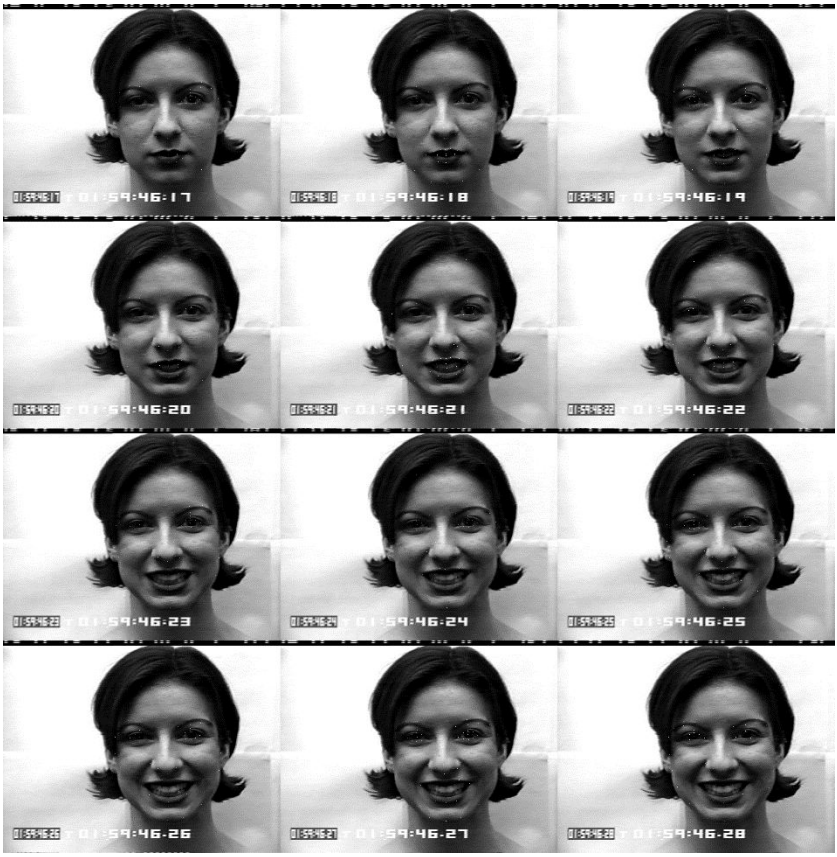
Uji coba dilakukan pada setiap subjek menggunakan skema 5 *fold cross-validation*, dimana setiap dataset semua percobaan dibagi menjadi lima bagian yang berjumlah relatif sama. Sembilan bagian digunakan untuk data latih dan satu bagian sisanya digunakan untuk data uji. Hasil uji coba pada bab ini merupakan hasil rata-rata dari lima iterasi 5 *fold cross-validation.*

6.3 Uji Coba Sistem

Pada bagian ini akan ditampilkan hasil dari masing-masing proses pada sistem pendeteksi ekspresi wajah pada sekuen citra.

Gambar 6.1 Sekuen citra masukan sistem

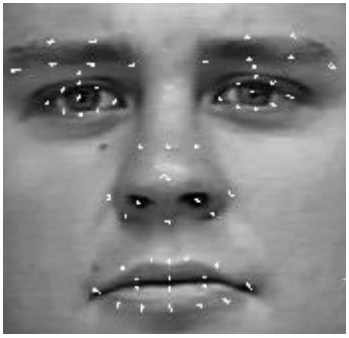
Tahap pertama yang dilalui sekuen citra masukan adalah tahap deteksi fitur titik wajah. Gambar 5.2 adalah hasil dari tahap deteksi fitur titik wajah.



Gambar 6.2 Hasil deteksi fitur titik wajah

Setelah didapatkan fitur titik wajah, dilakukan penghilangan noise pada fitur titik wajah tersebut menggunakan metode *averaging*. Gambar 5.3 dan 5.4 menunjukkan perubahan fitur titik wajah pada sekuen citra sebelum dilakukan *averaging* dan setelahnya.

Gambar 6.3 Perpindahan fitur titik wajah tanpa *averaging*

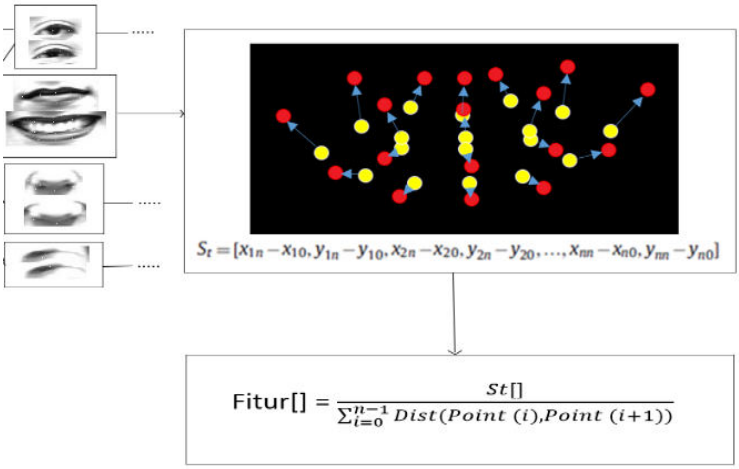


Gambar 6.4 Perpindahan fitur titik wajah dengan *averaging*

Setelah tahap ini, dilakukan deteksi puncak ekspresi. Gambar 5.5 menunjukkan citra netral dan citra puncak.

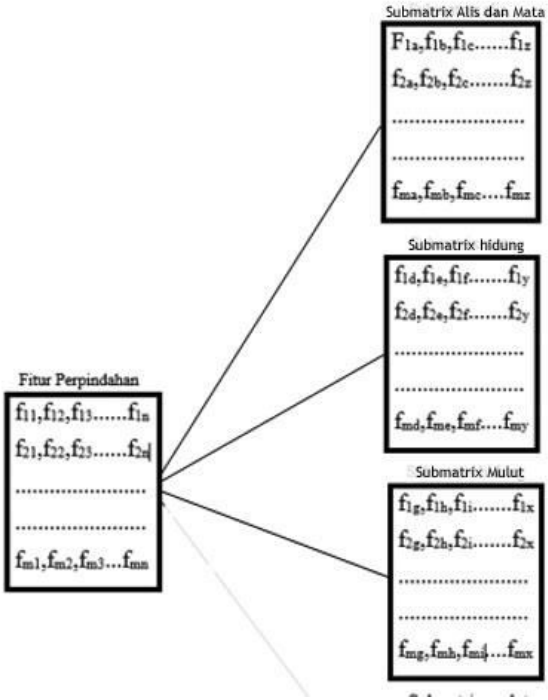
Gambar 6.5 Hasil deteksi puncak ekspresi

Setelah citra ekspresi puncak didapatkan, dicari nilai fitur perpindahan ekspresi netral-puncak untuk tiap bagian wajah (alis, mata, hidung, mulut). Gambar 5.6 menunjukkan representasi dari matrix fitur perpindahan.

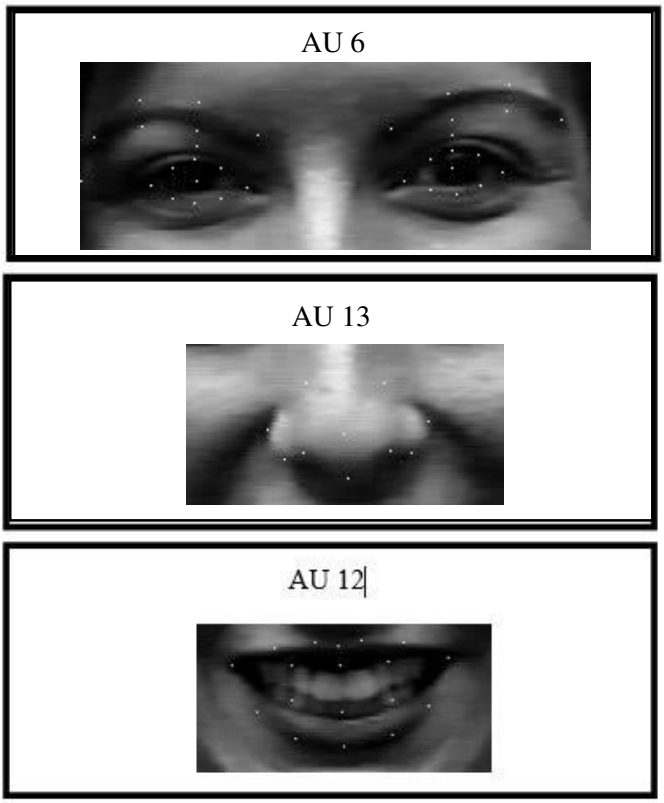


Gambar 6.6 Representasi matrix dari fitur perpindahan

Fitur perpindahan tersebut kemudian dibagi menjadi 3 submatrix sesuai bagian wajah (alis dan mata, hidung, mulut). Kemudian di-*training* menggunakna random forest level 1 untuk mengklasifikasikan *action unit*. Gambar 5.7 dan 5.8 masing-masing menunjukkan representasi submatrix bagian wajah dan hasil klasifikasi *action unit*.



Gambar 6.7 Representasi submatrix bagian wajah



Gambar 6.8 Hasil klasifikasi *action unit* pada ekspresi senang

Action unit tersebut kemudian di-*training* menggunakan random forest level 2 untuk mengklasifikasikan ekspresi.

6.4 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melaui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa masing-masing skenario, serta membandingkan skenario manakah yang memiliki performa lebih

baik. Terdapat tiga skenario uji coba pada perangkat lunak ini, antara lain:

1. Perhitungan performa dengan mengubah koefisien k pada *averaging* untuk memperbaiki fitur-fitur perpindahan. k merupakan jumlah sekuen citra yang dilakukan *averaging*. Nilai k yang akan diujikan antara lain : 1, 2, 3, dan 4.
2. Perhitungan performa dengan membadingkan klasifier random forest dengan twofold random forest. Saat menggunakan klasifier random forest, fitur-fitur perpindahan yang didapatkan sebelumnya langsung diklasifikasikan untuk tiap-tiap kelas ekspresi, sedangkan saat menggunakan klasifier twofold random forest, fitur-fitur perpindahan digunakan untuk mengklasifikasikan *Action Unit* (AU) yang terbagi menjadi 3 bagian : AU bagian atas (alis dan mata), AU bagian tengah (hidung), dan AU bagian bawah (bibir). Kemudian tiap AU digunakan untuk mengklasifikasikan ekspresi.
3. Perhitungan performa dengan mengubah jumlah kelas dan data ekspresi (marah, senang, jijik, terkejut, sedih, takut) sehingga didapatkan jumlah kelas ekspresi untuk model pengenalan ekspresi yang paling optimal.

6.4.1 Skenario Uji Coba 1

Skenario uji coba pertama adalah perhitungan performa yang meliputi akurasi dengan mengubah koefisien k pada *averaging* fitur-fitur perpindahan. k merupakan jumlah sekuen citra yang dilakukan *averaging*. Nilai k yang akan diujikan antara lain : 1, 2, 3, dan 4. Pada skenario uji coba ini, digunakan klasifier twofold random forest dengan iterasi pada random forest 50 dan 100.

Tabel 6.1 Hasil Akurasi Uji Coba Penggantian Koefisien k pada Averaging Fitur Perpindahan

Iterasi	k=1	k=2	k=3	k=4
50	62.22%	65.33%	73.33%	71.11%
100	63.33%	67.78%	74.45%	71.11%

Berdasarkan hasil uji coba yang ditunjukkan pada perhitungan akurasi di atas, saat nilai k = 3 akurasi mencapai nilai tertinggi yaitu 74.45% dengan 100 iterasi random forest dan 73.33% dengan 50 iterasi random forest.

Confusion matrix untuk sekenario uji coba 1 saat nilai k=1 dapat dilihat pada Lampiran 8.1, Lampiran 8.2, Lampiran 8.3, Lampiran 8.4, Lampiran 8.5, saat nilai k=2 dapat dilihat pada Lampiran 8.6, Lampiran 8.7, Lampiran 8.8, Lampiran 8.9, Lampiran 8.10, saat nilai k=3 dapat dilihat pada Lampiran 8.11, Lampiran 8.12, Lampiran 8.13, Lampiran 8.14, Lampiran 8.15, dan saat nilai k=4 dapat dilihat pada Lampiran 8.16, Lampiran 8.17, Lampiran 8.18, Lampiran 8.19, dan Lampiran 8.20.

6.4.2 Skenario Uji Coba 2

Skenario uji coba kedua adalah perhitungan performa yang meliputi akurasi dengan membadingkan klasifier random forest dengan twofold random forest. Pada sekenario uji coba ini, digunakan 100 iterasi serta nilai parameter *averaging* k=3.

Tabel 6.2 Hasil Akurasi Uji Coba Klasifier Random Forest dan Twofold Random Forest

Metode	Fold1	Fold2	Fold3	Fold4	Fold5	Average
Random Forest	83.33%	77.78%	77.78%	66.67%	61.11%	73.33%

Twofold Random Forest	88.89%	83.33%	77.78%	61.11%	61.11%	74.45%

Berdasarkan hasil uji coba yang ditunjukkan pada perhitungan akurasi di atas, hasil akurasi dari metode twofold random forest lebih besar dengan nilai 74.45%.

Confusion matrix untuk sekenario uji coba 2 dapat dilihat pada Lampiran 8.21, Lampiran 8.22, Lampiran 8.23, Lampiran 8.24, dan Lampiran 8.25.

6.4.3 Skenario Uji Coba 3

Skenario uji coba 3 adalah perhitungan performa dengan mengubah jumlah kelas ekspresi(marah, senang, jijik, terkejut, sedih, takut) sehingga didapatkan jumlah kelas ekspresi untuk model pengenalan ekspresi yang paling optimal. Pada skenario ini digunakan 100 iterasi, nilai k=3 ,dan ekspresi takut akan dipisahkan dari data karena akurasi untuk ekspresi takut adalah yang paling kecil.

Tabel 6.3 Hasil Akurasi Uji Coba Pemisahan Kelas Takut

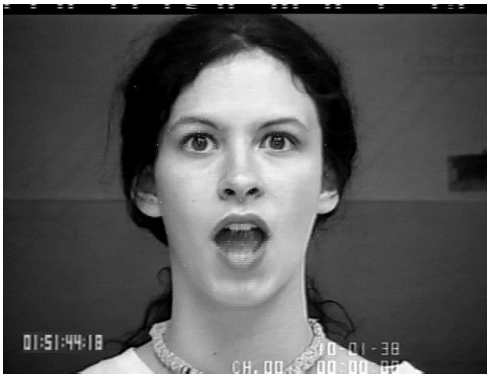
Jumlah kelas	Fold1	Fold2	Fold3	Fold4	Fold5	Average
Dengan 6 kelas (senang, sedih, marah, takut, terkejut, jijik)	88.89%	83.33%	77.78%	61.11%	61.11%	74.45%
Dengan 5 kelas (senang, sedih, marah, terkejut, jijik)	100%	100%	100%	100%	100%	100%

Berdasarkan hasil uji coba yang ditunjukkan pada perhitungan akurasi di atas, dengan pemisahan kelas ekspresi takut, dapat dihasilkan akurasi sebesar 100%.

Confusion matrix untuk sekenario uji coba 3 dapat dilihat pada Lampiran 8.26, Lampiran 8.27, Lampiran 8.28, Lampiran 8.29, dan Lampiran 8.30.

6.5 Evaluasi

Dataset sekuen citra yang digunakan memiliki rentang jumlah citra yang bervariasi yaitu antara 6 sampai 60 citra. Hal ini menyebabkan metode *averaging* yang dilakukan menghasilkan fitur titik wajah jauh dari titik sebenarnya untuk data dengan jumlah citra <20. Gambar 5.1 dan 5.2 menunjukkan hasil dari *averaging* untuk jumlah k-frame yang sama pada data dengan jumlah citra <20 dan data dengan jumlah citra >20.



Gambar 6.9 *Averaging* pada data dengan jumlah citra <20

Gambar 6.10 *Averaging* pada data dengan jumlah citra >20

Pada perbandingan penggunaan klasifier random forest dan twofold random forest, walaupun akurasi pada twofold random forest lebih besar, masih tidak dapat disimpulkan klasifier mana yang lebih baik untuk mengklasifikasikan data sekuen citra tersebut karena hanya terdapat selisih akurasi sebesar 1.11%.

Terdapat beberapa data yang salah dikenali, dan sebagian besar pada data ekspresi takut. Hal ini disebabkan karena beberapa ekspresi takut mirip dengan ekspresi lainnya, seperti senang, terkejut, dan sedih. Gambar 5.3, Gambar 5.4, dan Gambar 5.5 menunjukkan contoh ekspresi takut yang mirip ekspresi lainnya.

Gambar 6.11 Ekspresi takut yang mirip ekspresi senang

Gambar 6.12 Ekspresi takut yang mirip ekspresi sedih

Gambar 6.13 Ekspresi takut yang mirip ekspresi terkejut

Dari skenario uji coba yang telah dilakukan, nilai akurasi terbaik didapatkan saat menggunakan twofold random forest dengan parameter *averaging* k =3, dan dengan 5 kelas ekspresi (senang, sedih, terkejut, marah, jijik) dengan akurasi sebesar 100%.

LAMPIRAN

Pada bagian lampiran, akan dilampirkan *confusion matrix* untuk masing-masing skenario.

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	1	0
C	0	0	3	0	0	0
D	0	0	1	2	0	0
E	0	1	0	2	0	0
F	0	0	1	0	0	2

Lampiran 8.1 *Confusion matrix* uji coba skenario 1 Parameter k =1, iterasi = 100, fold = 1

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	1	0
C	0	0	1	0	0	2
D	0	0	1	2	0	0
E	0	1	0	1	0	1
F	0	0	0	0	0	3

Lampiran 8.2 *Confusion matrix* uji coba skenario 1 Parameter k =1, iterasi = 100, fold = 2

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	1	1	1	0
D	0	0	0	3	0	0
E	1	0	0	1	0	1
F	0	0	0	1	0	2

Lampiran 8.3 *Confusion matrix* uji coba skenario 1 Parameter k =1, iterasi = 100, fold = 3

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	1	0	2	0
D	0	0	0	1	1	1
E	0	0	0	0	1	2
F	0	0	1	0	0	2

Lampiran 8.4 *Confusion matrix* uji coba skenario 1 Parameter k =1, iterasi = 100, fold = 4

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	1	0
C	0	0	2	0	1	0
D	0	0	1	2	0	0
E	0	1	2	0	0	0
F	0	0	1	0	0	2

Lampiran 8.5 *Confusion matrix* uji coba skenario 1 Parameter k =1, iterasi = 100, fold = 5

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	1	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	1	0	2	0	0
F	0	0	0	0	0	3

Lampiran 8.6 *Confusion matrix* uji coba skenario 1 Parameter k =2, iterasi = 100, fold = 1

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	0	1
C	0	0	1	0	0	2
D	0	0	0	3	0	0
E	0	1	0	0	0	2
F	0	0	0	0	0	3

Lampiran 8.7 *Confusion matrix* uji coba skenario 1 Parameter k =2, iterasi = 100, fold = 2

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	1	1	0	1
D	0	0	0	3	0	0
E	1	0	0	0	0	2
F	0	0	1	0	0	2

Lampiran 8.8 *Confusion matrix* uji coba skenario 1 Parameter k =2, iterasi = 100, fold = 3

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	1	0	1	1
D	0	0	0	2	1	0
E	0	1	0	0	0	2
F	0	0	1	0	0	2

Lampiran 8.9 *Confusion matrix* uji coba skenario 1 Parameter k =2, iterasi = 100, fold = 4

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	2	0	1	0
D	0	0	1	2	0	0
E	0	1	2	0	0	0
F	0	0	1	0	0	2

Lampiran 8.10 *Confusion matrix* uji coba skenario 1 Parameter k =2, iterasi = 100, fold = 5

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	0	0	2	1	0
F	0	0	0	0	0	3

Lampiran 8.11 *Confusion matrix* uji coba skenario 1 Parameter k =3, iterasi = 100, fold = 1

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	0	1	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	0	0	0	1	2
F	0	0	0	0	0	3

Lampiran 8.12 *Confusion matrix* uji coba skenario 1 Parameter k =3, iterasi = 100, fold = 2

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	1	0	1	0	1
F	0	0	1	0	0	2

Lampiran 8.13 *Confusion matrix* uji coba skenario 1 Parameter k =3, iterasi = 100, fold = 3

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	1	0	0
C	0	0	2	0	0	1
D	0	0	0	2	1	0
E	0	1	0	1	0	1
F	0	0	1	0	0	2

Lampiran 8.14 *Confusion matrix* uji coba skenario 1 Parameter k =3, iterasi = 100, fold = 4

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	2	0	1	0
D	0	0	1	2	0	0
E	0	1	2	0	0	0
F	0	0	1	1	0	1

Lampiran 8.15 *Confusion matrix* uji coba skenario 1 Parameter k =3, iterasi = 100, fold = 5

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	0	0	1	2	0
F	0	0	1	0	0	2

Lampiran 8.16 *Confusion matrix* uji coba skenario 1 Parameter k =4, iterasi = 100, fold = 1

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	2	1	0	0
D	0	0	0	3	0	0
E	0	1	0	0	1	1
F	0	0	1	0	0	2

Lampiran 8.17 *Confusion matrix* uji coba skenario 1 Parameter k =4, iterasi = 100, fold = 2

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	3	0	0	0	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	2	0	0	0	1
F	0	0	1	0	0	2

Lampiran 8.18 *Confusion matrix* uji coba skenario 1 Parameter k =4, iterasi = 100, fold = 3

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	1	0	0
C	0	0	2	0	0	1
D	0	0	0	1	2	0
E	0	1	0	1	0	1
F	0	0	1	0	0	2

Lampiran 8.19 *Confusion matrix* uji coba skenario 1 Parameter k =4, iterasi = 100, fold = 4

	A	B	C	D	E	F
A	3	0	0	0	0	0
B	0	2	0	1	0	0
C	1	0	1	1	0	0
D	0	0	1	2	0	0
E	1	0	1	0	1	0
F	0	0	1	1	0	1

Lampiran 8.20 *Confusion matrix* uji coba skenario 1 Parameter k =4, iterasi = 100, fold = 5

	A	B	C	D	E	F
A	1	0	0	0	2	0
B	0	3	0	0	0	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	1	0	0	2	0
F	0	0	0	0	0	3

Lampiran 8.21 *Confusion matrix* uji coba skenario 2 Parameter k =3, iterasi = 100, fold = 1

	A	B	C	D	E	F
A	1	0	0	0	2	0
B	0	2	0	0	1	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	1	0	0	2	0
F	0	0	0	0	0	3

Lampiran 8.22 *Confusion matrix* uji coba skenario 2 Parameter k =3, iterasi = 100, fold = 2

	A	B	C	D	E	F
A	0	0	0	0	3	0
B	0	2	0	0	1	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	0	0	0	3	0
F	0	0	0	0	0	3

Lampiran 8.23 *Confusion matrix* uji coba skenario 2 Parameter k =3, iterasi = 100, fold = 3

	A	B	C	D	E	F
A	2	0	0	0	1	0
B	0	1	1	0	1	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	0	2	0	0	1	0
F	0	0	1	0	0	2

Lampiran 8.24 *Confusion matrix* uji coba skenario 2 Parameter k =3, iterasi = 100, fold = 4

	A	B	C	D	E	F
A	0	0	0	0	3	0
B	0	1	0	0	2	0
C	0	0	3	0	0	0
D	0	0	0	3	0	0
E	1	1	0	0	1	0
F	0	0	0	0	0	3

Lampiran 8.25 *Confusion matrix* uji coba skenario 2 Parameter k =3, iterasi = 100, fold = 5

	A	B	C	D	E
A	3	0	0	0	0
B	0	3	0	0	0
C	0	0	3	0	0
D	0	0	0	3	0
E	0	0	0	0	3

Lampiran 8.26 *Confusion matrix* uji coba skenario 3 Parameter k =3, iterasi = 100, jumlah kelas = 5, fold =1

	A	B	C	D	E
A	3	0	0	0	0
B	0	3	0	0	0
C	0	0	3	0	0
D	0	0	0	3	0
E	0	0	0	0	3

Lampiran 8.27 *Confusion matrix* uji coba skenario 3 Parameter k =3, iterasi = 100, jumlah kelas = 5, fold =2

	A	B	C	D	E
A	3	0	0	0	0
B	0	3	0	0	0
C	0	0	3	0	0
D	0	0	0	3	0
E	0	0	0	0	3

Lampiran 8.28 *Confusion matrix* uji coba skenario 3 Parameter k =3, iterasi = 100, jumlah kelas = 5, fold =3

	A	B	C	D	E
A	3	0	0	0	0
B	0	3	0	0	0
C	0	0	3	0	0
D	0	0	0	3	0
E	0	0	0	0	3

Lampiran 8.29 *Confusion matrix* uji coba skenario 3 Parameter k =3, iterasi = 100, jumlah kelas = 5, fold =4

	A	B	C	D	E
A	3	0	0	0	0
B	0	3	0	0	0
C	0	0	3	0	0
D	0	0	0	3	0
E	0	0	0	0	3

Lampiran 8.30 *Confusion matrix* uji coba skenario 3
Parameter k =3, iterasi = 100, jumlah kelas = 5, fold =5

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

7.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan uji coba dan evaluasi yang telah dilakukan antara lain :

1. Sistem pengenalan ekspresi wajah dengan menggunakan metode ASM dan klasifier twofold random forest berhasil mengenali ekspresi wajah dengan akurasi 74.45%.
2. Koefisien k (jumlah citra yang dilakukan *averaging*) bernilai 3 memberikan nilai akurasi paling tinggi yaitu sebesar 74.45%.
3. Penggunaan twofold random forest pada klasifikasi ekspresi memeberikan hasil yang lebih baik daripada menggunakan metode random forest satu level dengan selisih akurasi sebesar 1.11%.
4. Uji coba data sekuen citra tanpa menggunakan ekpresi takut menaikkan akurasi sebesar 25.5%.

7.2 Saran

Beberapa saran yang disampaikan terkait pengerjaan tugas akhir ini adalah :

1. Penggunaan 5 ekspresi (senang, marah, sedih, terkejut, dan jijik) untuk sistem pengenalan ekspresi sehingga dihasilkan akurasi yang lebih baik.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] K. X. a. L. a. Z. XiaorongPu a, "Facial expression recognition from image sequences using twofold," *Neurocomputing,* vol. 168, pp. 1173-1180, 2015.

[2] P. Lucey1, J. F. C. 2 dan T. K. J. S. Z. A. 2, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit," dalam *Disney Research*, Pittsburgh, 2015.

[3] T. Cootes, An Introduction to Active Shape Models, Oxford University Press,, 2000.

[4] Z. Zhang, "Feature-Based Facial Expression Recognition: Sensitivity," *Pattern Recognition and Artificial Intelligence,* vol. 6, no. 13, pp. 893-911, 1999.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Arika Saputro, penulis dari buku tugas akhir ini lahir di kota Magelang, 13 Mei 1993. Penulis telah menempuh pendidikan di SD Negeri Paremono 4 (1999-2005), SMP Negeri 1 Mungkid (2005-2008), SMA Negeri 3 Magelang (2008-2011) dan Teknik Informatika ITS Surabaya (2012-2016). Selama masa perkuliahan, penulis menghasilkan beberapa prestasi yang membanggakan, salah satunya adalah juara dua dalam pengembangan aplikasi permainan nasional di GemasTIK 8 yang bertempat di UGM dan finalis Microsoft ImagineCup 2016 Indonesia. Penulis pernah menjadi asisten pada mata kuliah Perancangan dan Analisis Algoritma 1, Perancangan dan Analisis Algoritma 2, dan Teori Pengembangan Game. Penulis juga pernah menjadi koordinator laboratorium Komputasi Cerdas dan Visi (2014/2015). Penulis memilih bidang minat Komputasi Cerdas dan Visi (KCV) dan tertarik pada topik pengenalan pola dan kecerdasan buatan. Penulis dapat dihubungi melalui email arika.saputro12@gmail.com.