TUGAS AKHIR – KI141502

IMPLEMENTASI DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN PADA APLIKASI PENGENALAN WAJAH

KEVIN ZULKARNAIN YUSETI
5113100146

Dosen Pembimbing
RULLY SOELAIMAN, S.Kom., M.Kom.
RIZKY JANUAR AKBAR, S.Kom., M.Eng.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR – KI141502

IMPLEMENTASI DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN PADA APLIKASI PENGENALAN WAJAH

KEVIN ZULKARNAIN YUSETI
5113100077

Dosen Pembimbing I
RULLY SOELAIMAN, S.Kom., M.Kom.

Dosen Pembimbing II
RIZKY JANUAR AKBAR, S.Kom., M.Eng.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN IMPLEMENTATION FOR FACE RECOGNITION

KEVIN ZULKARNAIN YUSETI
5113100146

Supervisor I
RULLY SOELAIMAN, S.Kom., M.Kom.

Supervisor II
RIZKY JANUAR AKBAR, S.Kom., M.Eng.

DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN PADA APLIKASI PENGENALAN WAJAH

TUGAS AKHIR

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

Oleh:
KEVIN ZULKARNAIN YUSETI
NRP: 5113 100 146

Disetujui oleh Dosen Pembimbing Tugas Akhir:

RULLY SOELAIMAN, S.Kom., M.Kom.
NIP. 197002131994021001 ..... (Pembimbing 1)

RIZKY JANUAR AKBAR, S.Kom., M.Eng.
NIP. 198701032014041001 ..... (Pembimbing 2)

SURABAYA
JULI, 2017

[*Halaman ini sengaja dikosongkan*]

IMPLEMENTASI DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN PADA APLIKASI PENGENALAN WAJAH

Nama Mahasiswa : Kevin Zulkarnain Yuseti
NRP : 5113 100 146
Departemen : Teknik Informatika, FTIf ITS
Dosen Pembimbing 1 : Rully Soelaiman, S.Kom., M.Kom.
Dosen Pembimbing 2 : Rizky Januar Akbar, S.Kom., M.Eng.

Abstrak

Local Directional Pattern (LDP) adalah sebuah metode ekstraksi fitur yang digunakan untuk pengenalan pola pada gambar dan pengenalan wajah. Pada tugas akhir ini pengembangan dilakukan untuk mereduksi jumlah dimensi dari hasil ekstraksi fitur. Oleh karena itu dinamakan Dimensionality Reduced–Local Directional Pattern (DR-LDP). LDP mendapatkan deskriptor gambar dari operasi korelasi matriks antara piksel gambar dan tetangganya dengan Kirsch mask. Dari korelasi matriks didapatkan matriks baru dengan kode LDP sebagai deskriptor gambar. Reduksi dimensi pada matriks ini dilakukan dengan mengambil nilai LDP terbesar dari blok kode LDP sehingga didapati kode yang merepresentasikan satu blok kode LDP. Pengujian dilakukan pada dataset standar ORL dan Extended YALE-B. Percobaan pada dataset ORL menghasilkan akurasi 80% dan YALE-B 81% untuk kasus pencahayaan, 67% untuk kasus kemiringan kepala.

Kata kunci: Pengenalan Wajah, Local Directional Pattern, Reduksi Dimensi.

[*Halaman ini sengaja dikosongkan*]

DIMENSIONALITY REDUCED LOCAL DIRECTIONAL PATTERN IMPLEMENTATION FOR FACE RECOGNITION

Student Name	: Kevin Zulkarnain Yuseti
Registration Number	: 5113 100 146
Department	: Informatics Engineering, FTIf ITS
First Supervisor	: Rully Soelaiman, S.Kom., M.Kom.
Second Supervisor	: Rizky Januar Akbar, S.Kom., M.Eng.

Abstract

Local Directional Pattern (LDP) are feature extraction method used for pattern and face recognition. This undergraduate thesis purpose were to conduct dimensional reduction on feature extracted from LDP. Therefore called Dimensionality Reduced-Local Directional Pattern. Image descriptor from LDP came from matrix correlation between image pixel and neighboring pixels with kirsch mask. The result of matrix correlation resulting in LDP matrix as image descriptor. Dimension reduction done by pick maximum value from LDP coded block, resulting in single code representing the entire of LDP code block. Testing done by using this method on standard dataset ORL and extended YALE-B. Accuracy result for ORL dataset were 80% and YALE-B 81% for illumination variance, 67% for head tilt variance.

Keywords: Face Recognition, Local Directional Pattern, Dimensionality Reduction.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur bagi Allah SWT yang atas rahmat dan anugerah-Nya sehingga penulis bisa menyelesaikan tugas akhir yang berjudul "Implementasi Dimensionality Reduced Local Directional Pattern pada Aplikasi Pengenalan Wajah".

Pengerjaan Tugas Akhir ini memberikan banyak pelajaran kepada penulis dalam memahami dan mendalami semua yang telah penulis pelajari selama berkuliah di Teknik Informatika ITS Surabaya. Selesainya Tugas Akhir ini tak semata-mata karena usaha dari penulis. Namun, juga berkat bantuan yang tak terhitung jumlahnya oleh Allah SWT dan rekan-rekan penulis. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih dan syukur sebesar-besarnya kepada:

1. Allah SWT, yang atas kehendak dan rahmat-Nya penulis sanggup menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Bapak Ulfi Yuseti dan Ibu Nurul Akhyati. Yang berkat didikan, nasihat, dukungan moral dan materi. Sehingga penulis bisa terus hidup dan meneruskan pendidikan di Teknik Informatika ITS.
3. Bapak Rully Soelaiman S.Kom., M.Kom. dan Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku pembimbing penulis yang telah memberikan pacuan semangat, bimbingan dan nasihat selama pengerjaan Tugas Akhir.
4. Bapak Darlis Herumurti, S.Kom., M.Kom., segenap dosen dan karyawan jurusan Teknik Informatika ITS yang telah banyak membantu penulis selama menjalani 4 tahun perkuliahan.
5. Sahabat-sahabat SMAN 2 Kediri, Dili, Yogi, Zella, Rosa dan Pipit. Peh, saya bangga dengan kalian.
6. Rekan-rekan Asisten Laboratorium Pemrograman 2. John, Lusiana, Theo, Resha, Cynthia, William dan Rogo. Yang

konsisten menyemangati penulis dalam pengerjaan Tugas Akhir.

7. Rekan-rekan Sahabat KCV. Haqiqi, Ery, Ihsan, Rizok, Nela, Nida dan yang lainnya. Yang telah menjadi teman belajar dan berdiskusi sehingga meringankan beban penulis dalam menjalani perkuliahan dan pengerjaan Tugas Akhir.
8. Rekan-rekan *Admin* dan *User* setia Laboratorium Algoritma dan Pemrograman, Ridho, Anwar, Rei, Luffy, Yohana, Aldi, Dimas, Rani, Hidayatul, Rozana dan yang lainnya, Yang telah menemani penulis tiap hari dan malam selama mengerjakan tugas akhir.
9. Teman-teman C1D, TC13, Vektor, SSL, Kontrakan WC Outdoor dan banyak lainnya. Yang memberikan warna dalam keseharian penulis selama berkuliah di Surabaya.
10. Dan sumber motivasi-motivasi lain yang memicu semangat untuk segera lulus dari perkuliahan.

Penulis memohon maaf atas kekurangan yang terdapat dalam penulisan Tugas Akhir ini. Penulis menyadari bahwa tiada yang sempurna di dunia ini, Semua saran dan kritikan untuk penulis akan dijadikan pelajaran untuk saat yang akan datang, Semoga Tugas Akhir ini dapat memberikan manfaat dalam pengembangan teknologi informasi.

Surabaya, Juli 2017

Kevin Zulkarnain Yuseti

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

Daftar Gambar

Daftar Tabel

[Halaman ini sengaja dikosongkan]

Daftar Kode Sumber

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

Pengenalan wajah adalah salah satu bidang riset yang masih aktif dalam penelitiannya. Hal ini selaras dengan semakin banyaknya aplikasi yang memiliki fitur pengenalan wajah. Walaupun sudah banyak metode yang dikembangkan, bidang ini memiliki kesulitannya tersendiri. Kesulitan tersebut adalah perbedaan gambar dalam hal: pencahayaan, kemiringan kepala, ekspresi wajah. Kompleksitas algoritma yang digunakan juga berpengaruh pada kecepatan mesin dalam memproses dan mengenali wajah yang ada.

Pengenalan wajah memiliki peran yang beragam dalam kehidupan sehari-hari. Salah satu fungsi pengenalan wajah adalah untuk hal keamanan. Walau banyak metode lain yang bisa digunakan seperti pengenalan iris atau sidik jari, pengenalan wajah tetap diminati karena data wajah lebih serba guna dan mudah didapatkan tanpa peralatan khusus.

Salah satu metode yang digunakan untuk ekstraksi fitur dengan operasi kernel menggunakan *Kirsch-mask* yang menghasilkan fitur dengan nama *Local Directional Pattern*. Metode ini pertama kali digunakan untuk melakukan klasifikasi pola gambar. Metode ini bekerja dengan menentukan nilai gradasi yang paling signifikan dari hasil operasi kernel 8 arah *kirsch-mask* pada sebuah piksel dan piksel tetangganya. Dari nilai fitur tersebut, disusun sebuah histogram sebagai deskriptor gambar.

Tugas akhir ini bertujuan mengimplementasikan pengenalan wajah dengan metode *Dimensionality Reduced Local Directional Pattern (DR-LDP)*. Dari reduksi tersebut didapatkan

kode fitur yang lebih sedikit namun padat. Kode fitur yang lebih sedikit dan padat ini mampu meningkatkan kecepatan proses dari pengenalan wajah.

Tugas akhir ini diharapkan bisa meningkatkan performa dalam proses pengenalan wajah pada berbagai kasus perbedaan kondisi pemotretan wajah pada perangkat lunak pengenalan wajah.

1.2 Rumusan Masalah

Rumusan masalah yang terdapat pada tugas ini adalah:

1. Memahami konsep ekstraksi fitur dengan metode *Dimensionality Reduced Local Directional Pattern.*
2. Memahami konsep klasifikasi wajah menggunakan fitur *Dimensionality Reduced Local Directional Pattern.*
3. Merancang perangkat lunak yang dapat melakukan pra-proses, ekstraksi fitur *Dimensionality Reduced Local Directional Pattern* dan mengklasifikasikan wajah.
4. Mengimplementasikan perangkat lunak yang telah dirancang untuk pengenalan wajah.
5. Menyusun uji coba ekstraksi fitur dan melaksanakan uji coba klasifikasi wajah dengan metode *Dimensionality Reduced Local Directional Pattern.*

1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah:

1. Perhitungan tingkat kemiripan wajah menggunakan *Chi-Square Dissimilarity*.
2. Deteksi keberadaan wajah untuk pra-proses menggunakan Viola-Jones *Object Detection* dari Open CV.
3. Pengujian menggunakan *dataset* wajah dari “ORL database of Face”[1] dan “Extended Yale-B”[2].
4. Implementasi menggunakan perangkat lunak Visual Studio 2015 CE.
5. Implementasi menggunakan bahasa pemrograman C#.

1.4 Tujuan dan Manfaat Tugas Akhir

Tujuan tugas akhir ini adalah sebagai berikut:

1. Mengetahui penerapan konsep ekstraksi fitur dengan *Kirsch-mask*.
2. Mengetahui penerapan metode *Dimensionality Reduced Local Directional Pattern* untuk klasifikasi wajah.
3. Menghasilkan rancangan perangkat lunak yang dapat melakukan pengenalan wajah dengan metode *Dimensionality Reduced Local Directional Pattern*.
4. Menghasilkan perangkat lunak sesuai dengan rancangan untuk melakukan pengenalan wajah.
5. Mengevaluasi performa dari metode *Dimensionality Reduced Local Directional Pattern* dengan melakukan pengujian.

Manfaat tugas akhir ini adalah untuk mengetahui penerapan metode *Dimensionality Reduced Local Directional Pattern* pada pengenalan wajah manusia. Diharapkan penerapan metode ini dapat menghasilkan akurasi yang tinggi pada gambar wajah manusia dengan berbagai macam ekspresi wajah dan pencahayaan dengan ukuran dimensi yang lebih kecil.

1.5 Metodologi

Metodologi yang digunakan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan Proposal tugas akhir

   Tahap awal pengerjaan tugas akhir ini adalah dengan penyusunan proposal tugas akhir. Proposal yang diajukan berisi gagasan untuk melakukan implementasi metode ekstraksi fitur *Dimensionality Reduced Local Directional Pattern* pada gambar wajah manusia.
2. Studi Literatur

   Pada tahap ini, dilakukan pengumpulan informasi, pengetahuan dan literatur yang diperlukan untuk merancang perangkat lunak. Literatur dan informasi yang dikumpulkan diperlukan untuk pra-proses gambar wajah, ekstraksi fitur dan klasifikasi dari deskriptor gambar. Informasi dan literatur

yang didapatkan berasal dari jurnal ilmiah dan sumber-sumber informasi lain yang relevan.

3. Perancangan Perangkat Lunak

   Pada tahap ini berisi perancangan perangkat lunak berdasarkan metode yang sudah dipelajari pada studi literatur dan pembelajaran terhadap teknologi perangkat lunak yang akan digunakan dalam implementasi. Dalam tahap ini meliputi alur-alur jalannya perangkat lunak. Di sini juga didefinisikan tiap langkah yang perlu dijalankan oleh perangkat lunak untuk menyelesaikan permasalahan.

4. Implementasi Perangkat Lunak

   Tahap ini mengerjakan rancangan yang telah dibuat. Rancangan yang telah disusun sebelumnya direalisasikan menjadi sistem yang utuh dan dapat memenuhi kebutuhan yang telah disusun pada tahap perancangan.

5. Pengujian dan Evaluasi

   Pada tahapan ini dilakukan uji coba pada perangkat lunak yang telah dibuat. Pengujian dan evaluasi dilakukan dengan membandingkan hasil keluaran perangkat lunak yang diperoleh dari data masukan. Tahap ini dimaksudkan untuk mengevaluasi jalannya perangkat lunak, mencari kesalahan yang mungkin timbul dan menyelesaikannya.

6. Penyusunan Buku Tugas Akhir

   Akhir dari pengerjaan berupa penyusunan dokumentasi yang berisi tentang pembuatan dan hasil keluaran dari perangkat lunak yang telah dibuat.

1.6 Sistematika Pelaporan

Buku Tugas akhir ini disusun dengan sistematika laporan 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 rinci tentang dasar-dasar penunjang dan teori-teori yang digunakan untuk menyelesaikan tugas akhir ini.

BAB III. ANALISIS DAN PERANCANGAN

Bab ini berisi tentang analisis data perancangan perangkat lunak yang akan dibuat. Rancangan perangkat lunak akan direpresentasikan menggunakan bagan alur beserta penjelasan.

BAB IV. IMPLEMENTASI

Bab ini membahas tentang implementasi rancangan yang telah dibuat pada bab sebelumnya. Penjelasan akan disampaikan dengan sumber kode dari perangkat lunak yang telah dibuat.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan tentang kemampuan dari perangkat lunak yang telah dibuat dengan melakukan uji kebenaran pada keluaran sistem.

BAB VI. KESIMPULAN DAN SARAN

Bab terakhir dari buku ini berisi tentang kesimpulan dari hasil uji coba perangkat lunak ini dan saran-saran yang dapat digunakan untuk mengembangkan perangkat lunak ini menjadi lebih baik lagi.

[*Halaman ini sengaja dikosongkan*]

BAB II
DASAR TEORI

Pada bab ini akan membahas tentang teori-teori yang digunakan dalam mengerjakan tugas akhir ini. Dalam pengerjaan tugas akhir ini digunakan beberapa teori dalam tahap pengerjaan. Teori–teori yang dijelaskan pada bab ini adalah *Dimensionality Reduced Local Directional Pattern* yang menjadi metode utama dalam tugas akhir ini, *Local Directional Pattern* sebagai metode yang akan dikembangkan, algoritma deteksi objek *Viola-Jones* dan peningkatan kontras gambar dengan histogram *equalization* yang digunakan untuk pra-proses gambar, *Local Directional Pattern* untuk ekstraksi fitur dari gambar dan *K-Nearest Neighbor* untuk klasifikasi data dengan *Chi-Square Dissimilarity* sebagai penghitung tingkat kemiripan fitur.

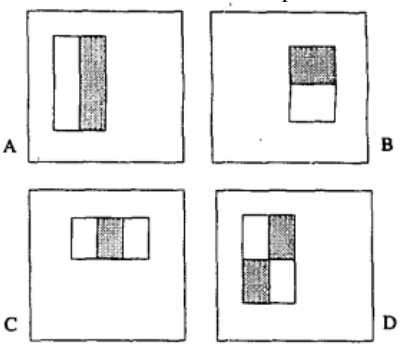
2.1 Pra-Proses Gambar Wajah

Gambar wajah yang digunakan pada pengerjaan tugas akhir ini memiliki banyak variasi pada tiap subjeknya. Variasi tersebut berupa perbedaan ekspresi pada wajah, kemiringan kepala dan sudut pencahayaan. Namun, tidak semua gambar bisa digunakan secara langsung. Terdapat informasi-informasi yang tidak diperlukan dalam gambar yang akan digunakan. Informasi-informasi yang tidak diperlukan ini disebut *noise*. Bukan hanya tidak diperlukan, *noise* ini mengganggu informasi-informasi penting yang terdapat pada data. Sehingga, keberadaan *noise* ini justru memperburuk kualitas data yang akan digunakan. Oleh karena itu diperlukan pra-proses untuk meningkatkan kualitas data yang akan digunakan.

Noise pada data yang dimaksudkan adalah adanya bagian gambar yang bukan wajah pada gambar. Piksel-piksel ini tidak memberi kontribusi pada pentingnya data. Untuk menghasilkan data yang lebih baik, *noise* ini dihapuskan dari gambar sehingga menyisakan bagian wajah dari data.

2.1.1 Viola-Jones *Object Detection*

Viola-Jones *object detection* adalah algoritma pengenalan objek yang diusulkan oleh Paul Viola dan Michael Jones pada tahun 2001. Algoritma ini menggunakan fitur nilai yang sederhana dan bukan menggunakan piksel gambar secara langsung. Pengenalan objek ini menggunakan *haar-feature* yang ada pada Gambar 2.1. *haar-feature* ini digunakan untuk deteksi pola bentuk gambar positif-negatif yang berguna untuk deteksi fitur tepi (A dan B pada gambar), garis (C pada gambar) dan fitur 4 persegi (D pada gambar) [3]. Algoritma ini efektif dalam mendeteksi wajah manusia pada gambar dengan kecepatan yang tinggi. Walaupun, hasil deteksi yang dihasilkan tidak selalu tepat.



Gambar 2.1 Bentuk *Haar-Feature*

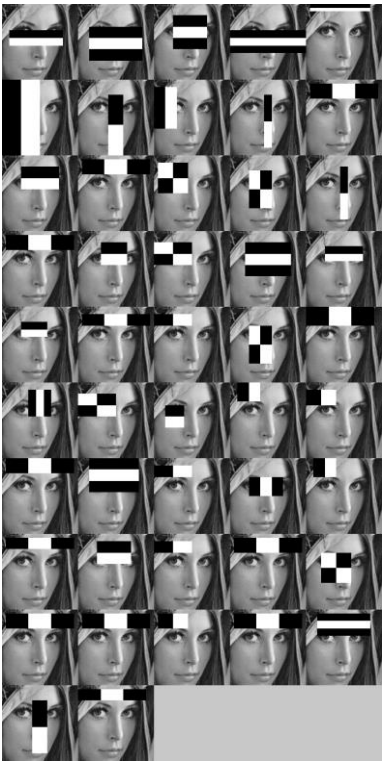
Pada deteksi wajah ditentukan dengan *haar-feature* yang sesuai dengan bentuk pola wajah manusia secara umum. Di mana secara sederhana wajah manusia bisa dilihat sebagai gambar positif-negatif seperti pada Gambar 2.2 dan Gambar 2.3.

Gambar 2.2 Bentuk Pola Wajah pada Batang Hidung

Gambar 2.3 Bentuk Pola Wajah pada Bagian Pipi Atas dan Mata

Dalam tugas akhir ini, ada dua *haar-cascade* yang digunakan untuk deteksi wajah. Dua *haar-cascade* yang digunakan dalam deteksi ini adalah "haarcascade_frontalface_default.xml" untuk deteksi bagian depan wajah dari arah depan dan "haarcascade_frontalface_alt_tree.xml" untuk deteksi wajah dari arah depan dengan *Haar-Feature* alternatif dari sudut yang sedikit berbeda[4]. Visualisasi dari kedua *haar-cascade* tersebut dapat dilihat pada Gambar 2.4 dan Gambar 2.5. Hal ini dilakukan agar kekurangan dari *haar-cascade* yang satu bisa diantisipasi oleh *haar-cascade* yang lain. Diharapkan dengan penggunaan dua set *haar-cascade* ini dapat memberikan deteksi wajah yang baik.

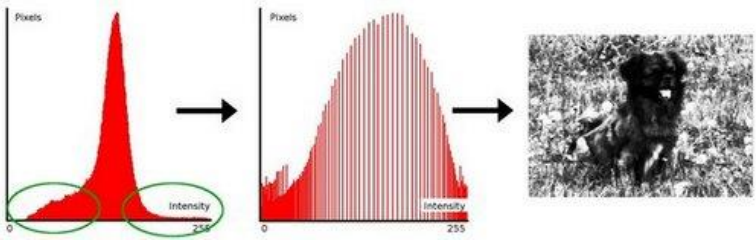
Gambar 2.4 Visualisasi *Haar-Feature* pada "haarcascade_frontalface_default"



Gambar 2.5 Visualisasi *Haar-Feature* pada "haarcascade_frontalface_alt_tree.xml"

2.1.2 Histogram *Equalization*

Histogram *equalization* merupakan teknik pemerataan distribusi warna pada sebuah citra yang bertujuan untuk meningkatkan kontras gambar. Algoritma ini bekerja dengan memetakan persebaran warna dan frekuensi masing–masing intensitas warna. Selanjutnya, warna–warna dengan frekuensi tinggi yang menempati jangkauan intensitas warna yang berdekatan akan direnggangkan ke jangkauan warna yang memiliki frekuensi warna lebih rendah[5].



Gambar 2.6 Ilustrasi Perenggangan Warna pada Histogram Warna

Algoritma ini digunakan untuk meningkatkan kontras citra agar objek pada citra lebih mudah dilihat. Algoritma ini digunakan untuk meningkatkan keakurasian algoritma Viola–Jones dalam deteksi wajah. Gambar 2.6 merupakan contoh bagaimana persebaran intensitas warna diratakan.

Gambar 2.7 Contoh Data Sebelum Dilakukan Histogram *Equalization*

Gambar 2.8 Contoh Data Sesudah Dilakukan Histogram *Equalization*

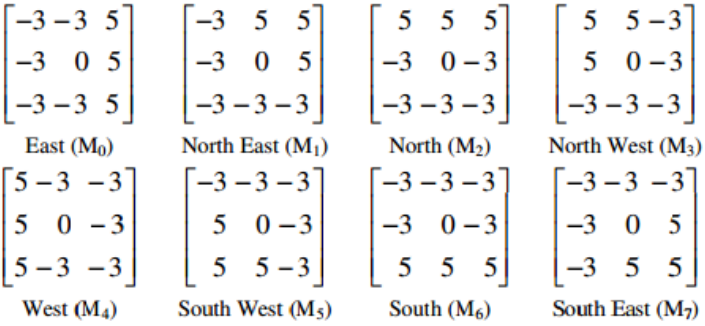
2.2 Ekstraksi Fitur

Gambar yang telah dihilangkan *noise*-nya akan diekstraksi fiturnya. Fitur yang akan diekstraksi adalah fitur *Local Directional Pattern* yang didapatkan dari operasi korelasi dari piksel dan piksel tetangganya dengan *Kirsch-mask* 8 arah yang ada pada Gambar 2.9. Operasi kernel yang dilakukan akan memberikan nilai magnitudo blok piksel tersebut pada arah kernel yang bersangkutan. Dari operasi kernel terhadap ke 8 arah tersebut akan didapati nilai yang merepresentasikan magnitudo arah pada blok tersebut. Maka dari itu metode ini disebut *Local Directional Pattern*[6].

2.2.1 *Local Directional Pattern*

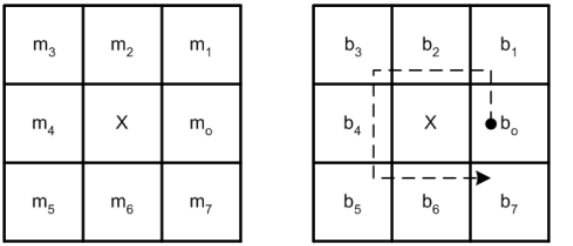
Local Directional Pattern adalah kode 8-bit yang didapat dari operasi korelasi dari tetanggaan piksel 3*3 dengan *Kirsch mask* 8 arah. Operasi ini dilakukan pada tiap piksel pada gambar

yang memiliki 8 tetangga piksel yang lengkap. *Kirsch mask* yang digunakan dapat dilihat pada Gambar 2.9.



Gambar 2.9 *Kirsch-Mask* 8 Arah

Dari operasi kernel didapatkan nilai biner dari tiap arah sebagai m_0, m_1, m_2... m_7. Perlu diketahui bahwa tidak semua nilai m_i penting dalam perhitungan. Sehingga, hanya nilai m_i sebanyak k yang paling signifikan ($|m_i|$ terbesar) yang digunakan.



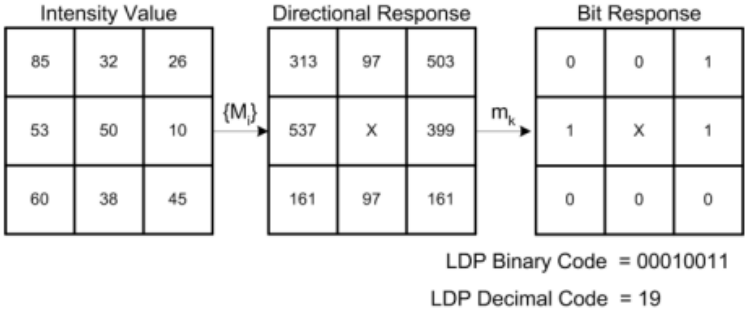
a) Eight directional edge response positions.

b) LDP binary bit positions.

Gambar 2.10 Pembentukan dan Peletakan Kode *LDP*

Nilai hasil konvolusi yang didapatkan ditempatkan ke dalam matriks sesuai dengan *mask*-nya. Penempatan hasil konvolusi dapat dilihat pada Gambar 2.10 bagian a. *Mask* dengan nomor m_0 pada sel m_0 dan seterusnya. Penyusunan kode biner LDP yang telah dipilih menyesuaikan pada Gambar 2.10 bagian b. Kode biner pertama yang menjadi *least significant bit* dibaca dari sel

matriks b_0 dan mengelilingi sel x melawan arah jarum jam hingga sel b_7 sebagai *most significant bit*[7]. Pada kode *LDP*, nilai m_i tersignifikan sebanyak k=3 diberi kode 1 dan sisanya kode 0. dengan begitu didapati sebanyak 56 kombinasi kemungkinan kode *LDP* yang dihasilkan.



Gambar 2.11 Contoh Pembentukan Kode *LDP*

Pada Gambar 2.11 merupakan contoh operasi ekstraksi nilai *LDP*. Matriks kiri pada Gambar 2.11 merupakan contoh masukan piksel 9*9 dengan *intensity value* sebagai nilai intensitas warna pada gambar. Matriks tengah pada Gambar 2.11 merupakan matriks yang berisi hasil operasi konvolusi dengan *Kirsch-mask.* Matriks kanan pada Gambar 2.11 merupakan kode biner *LDP* yang didapati setelah pengambilan nilai konvolusi tersignifikan sebanyak k. Dari matriks tersebut, didapatkan kode biner dengan cara pembacaan pada Gambar 2.10. Dari contoh ini nilai, m_i tersignifikan (m_0=399, m_1=503, m_4=537) diberikan kode 1 dan yang lainnya 0. Sehingga nilai *LDP*-nya 00010011 dalam basis biner atau 19 dalam bentuk basis desimal.

2.2.2 Operasi Korelasi Matriks

Korelasi matriks adalah sebuah operasi konvolusi matriks. Dalam korelasi matriks, *mask* yang digunakan untuk operasi tidak dibalik pada layaknya konvolusi. Hal ini dikarenakan nilai total dari *Kirsch-mask* yaitu (-3 -3 5 5 5 -3 -3 -3) adalah nol. Sehingga pembalikan nilai *mask* tidaklah diperlukan.

A	B	C
D	E	F
G	H	I

Matriks target

1	2	3
4	5	6
7	8	9

Matriks *mask*

Gambar 2.12 Contoh Operasi Kernel

Pada contoh Gambar 2.12 adalah operasi kernel yang akan dilakukan. Nilai hasil korelasi matriks yang akan didapatkan memenuhi dengan syarat pada Persamaan 2.1[5]. Nilai dari operasi kernel inilah yang akan digunakan untuk mendapatkan nilai *Local Directional Pattern*.

$$F(x) = (A * 1) + (B * 2) + (c * 3) + (D * 4) + (E * 5) + (F * 6) + (G * 7) + (H * 8) + (I * 9) \quad \textbf{(2.1)}$$

2.2.3 *Dimensionality Reduced Local Directional Pattern*

Pada *DR-LDP*, tiap piksel dibagi menjadi blok berukuran 3*3 piksel. Dari tiap blok piksel 3*3 tersebut, diambil nilai kode *LDP* terbesar. Nilai *LDP* terbesar ini digunakan untuk merepresentasikan blok piksel 3*3 tersebut. Sehingga, ukuran dimensi akhir dari proses ini akan tereduksi hingga satu per sembilan dari ukuran aslinya. Maka dari itu, metode ini disebut reduksi dimensi.

Reduksi dimensi ini diharapkan akan mengurangi waktu eksekusi dari perangkat lunak. Sebab, ukuran data yang lebih kecil akan memotong waktu yang diperlukan dalam pengolahan fitur secara keseluruhan.

16	25	160	210	66	232	67	170	114
44	27	60	125	154	48	53	51	133
148	189	88	50	44	70	8	55	154
136	114	234	131	96	177	203	84	177
108	53	148	163	250	13	71	85	16
144	22	17	155	186	105	141	167	21
7	174	45	148	131	173	183	78	34
251	207	139	40	231	158	227	254	220
172	153	213	172	110	253	197	95	118

Gambar 2.13 Contoh Matriks LDP dari Gambar masukan

Pada Gambar 2.13 terdapat matriks *LDP* berukuran 9*9. Dari matriks tersebut akan direduksi dengan mengambil nilai maksimal pada tiap blok 3*3. Sehingga dihasilkan matriks *LDP* yang tereduksi seperti pada Gambar 2.14.

189	232	170
234	250	203
251	253	254

Gambar 2.14 Contoh Hasil Reduksi Dimensi

Reduksi dimensi yang dilakukan berbeda dengan jurnal ilmiah rujukan utama[8]. Pada jurnal rujukan, reduksi dimensi dilakukan dengan melakukan operasi bit XOR pada blok matriks *LDP* 3*3.

C_4	C_3	C_2
C_5	X	C_1
C_6	C_7	C_8

Gambar 2.15 Ilustrasi Blok Matriks *LDP*

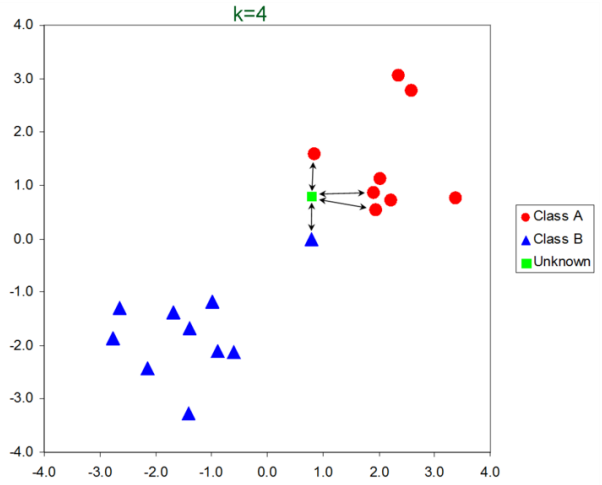
Reduksi dimensi pada blok *LDP* dilakukan dengan operasi XOR sesuai dengan Persamaan 2.2 di mana nilai C_1, C_2, C_3 hingga C_8 sesuai dengan ilustrasi blok matriks *LDP* pada Gambar 2.15 dan *Pj* menyatakan blok matriks ke-*j*.

$$Pj = \left((C_1 \oplus C_2) \oplus C_3\right) \ldots \oplus C_8) \quad \textbf{(2.2)}$$

Hasil reduksi dari operasi tersebut berkemungkinan menghasilkan kode biner 11111111 atau 00000000. Hal ini dihindari dengan cara hanya mengambil 3 *leading one* pertama dan mengubah kode 1 sisanya menjadi 0. Dengan begitu nilai yang mungkin dihasilkan ada sebanyak 56 kemungkinan kode *DR-LDP* yang dihasilkan.

2.3 Klasifikasi dengan *K-Nearest Neighbor*

K-Nearest Neighbor adalah metode klasifikasi yang berdasarkan pada jarak antara dua objek pada dimensi fitur di antara data uji dan data sampel. Pada *K-Nearest Neighbor* kelas dari data uji ditentukan oleh kelas dengan frekuensi tertinggi dalam *k* tetangga terdekat dengan data yang akan diklasifikasikan[9].



Gambar 2.16 Ilustrasi *K-NN*

Pada Gambar 2.16, sebuah objek baru diperkenalkan dalam data. Dari jumlah 4 tetangga terdekat. Kelas dengan frekuensi tertinggi adalah kelas A. maka objek baru diklasifikasikan ke dalam kelas A.

Algoritma *K-Nearest Neighbor*:

1. Hitung jarak tiap objek data sampel dengan data uji.
2. Urutkan objek sesuai dengan jaraknya. Dari terendah menuju ke tertinggi.
3. Ambil objek sebanyak *k* dengan jarak terendah dari kumpulan data.
4. Kumpulan objek tersebut diambil kelas yang paling banyak muncul dari kumpulan k objek tersebut.
5. Kelas data uji sesuai dengan kelas modus data sampel *k*.

2.4 *Dissimilarity* antar Fitur *Local Histogram*

Perhitungan *dissimilarity* antar dua fitur dilakukan pada tiap blok lokal-histogram yang dihasilkan dari setiap blok kode *DR-LDP* yang dihasilkan. Perhitungan dilakukan antar blok *LDP* ke-*i*

dari dua data yang akan dibandingkan. Nilai akhir adalah akumulasi dari hasil tiap lokal-histogram. Perhitungan diilustrasikan seperti pada Gambar 2.17.

Fitur 1 (f_1)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fitur 2 (f_2)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Gambar 2.17 Ilustrasi Pembagian Blok pada Dua Deskriptor

Pada Gambar 2.17 fitur 1 sebagai data uji dan fitur 2 sebagai data latih. Matriks kedua data tersebut dibagi menjadi blok-blok matriks yang lebih kecil. Pada contoh ini, matriks dibagi menjadi 16 blok matriks. Setiap blok matriks tersebut dihitunglah nilai histogram yang menjadi fitur pada perhitungan *Chi-Square distance*. Nilai *Dissimilarity_Distance* dari dua fitur tersebut dihitung menggunakan Persamaan 2.3 dengan *n* adalah jumlah blok yang dibentuk. Perhitungan *Chi_Square_Distance* dilakukan antara blok *f1* nomor 1 dengan blok *f2* nomor 2 dan seterusnya. Sehingga, nilai yang didapatkan adalah akumulasi dari *Chi_Square_Distance* tiap blok matriks *LDP*[8]. Perhitungan *Chi_Square_Distance* yang digunakan pada Persamaan 2.3 dapat dilihat pada Persamaan 2.4.

$$Dissimilarity\_Distance = \sum_{i=1}^{n} Chi\_Square\_Distance(f1_i, f2_i) \quad \textbf{(2.3)}$$

$$Chi\_Square\_Distance(f1_i, f2_i) = \sum_{j=0}^{L} w_i \frac{(f1_i(j) - f2_i(j))^2}{f1_i(j) + f2_i(j)} \quad \textbf{(2.4)}$$

Dari Persamaan 2.4 $f1_i$ dan $f2_i$ merupakan fitur deskriptor dari fitur 1 dan fitur 2 pada blok ke-*i* dan w_i merupakan beban dari

blok ke-*i*. Nilai beban blok ke-*i* ditentukan dengan nilai modus dari fitur yang ada pada kedua deskriptor. Nilai beban berdasarkan modus fitur sesuai dengan Tabel 2.1. *L* pada persamaan tersebut merupakan panjang dari fitur yang ada. Pada perhitungan ini, panjang fitur serupa dengan panjang histogram *DRLDP*. Sehingga nilai *L* adalah 255.

Tabel 2.1 Nilai Beban pada Masing-Masing Modus Fitur

Modus Fitur	Nilai Beban
0-63	1
64-127	2
128-191	3
192-255	4

Blok fitur f_1 nomor 1

189	232	170
234	250	203
251	253	254

Blok fitur f_2 nomor 1

210	189	234
234	122	180
255	250	254

Gambar 2.18 Contoh Perhitungan Antara Dua Blok Fitur *DRLDP*

Contoh pada Gambar 2.18 dibuatlah histogram dari kedua blok tersebut. Sehingga didapatkan histogram yang ada pada Gambar 2.19. Perlu diketahui, nilai *DRLDP* yang memiliki frekuensi 0 tidak disertakan pada gambar tersebut.

Histogram blok fitur f_1 nomor 1

Nilai *DRLDP*	Frekuensi
170	1
189	1
203	1
232	1
234	1
250	1
251	1
253	1
254	1

Histogram blok fitur f_2 nomor 1

Nilai *DRLDP*	Frekuensi
122	1
180	1
189	1
210	1
234	2
250	1
254	1
255	1

Gambar 2.19 Contoh Histogram dari Dua Blok Fitur *DRLDP*

Berdasarkan Persamaan 2.4, nilai *Chi_Square_Distance* yang didapatkan adalah 40. Nilai *Chi_Square_Distance* yang didapatkan pada iterasi ke-*i* dapat dilihat pada Tabel 2.2. Perlu diketahui, perhitungan pada nilai ke-*j* yang memiliki nilai *DRLDP* sama dengan 0 pada kedua fitur tidak disertakan karena hanya menambahkan nilai 0 pada *Chi_Square_Distance*. Nilai *w* yang digunakan pada perhitungan ini adalah 4 karena nilai modus pada blok ini adalah 234 yang mana berdasarkan Tabel 2.1.

Tabel 2.2 Nilai yang Didapatkan pada Tiap Iterasi (Bagian Pertama)

Nilai *j*	Nilai *Chi_Square_Distance* pada iterasi ke-*j*
122	4

Tabel 2.3 Nilai yang Didapatkan pada Tiap Iterasi (Bagian Kedua)

Nilai *j*	Nilai *Chi_Square_Distance* pada iterasi ke-*j*
170	8
180	12
189	12
203	16
210	20
232	24
234	28
250	28
251	32
253	36
254	36
255	40

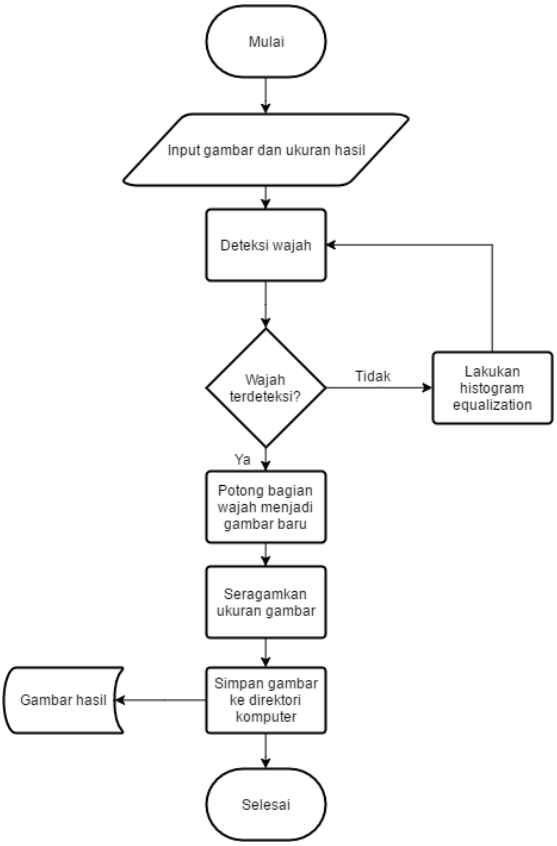
BAB III
PERANCANGAN

Bab ini berisi analisis data dan perancangan sistem. Data-data yang digunakan dalam tugas akhir ini berasal dari tiga *dataset* wajah baku yang digunakan dalam penelitian pengenalan wajah. tiga *dataset* wajah tersebut adalah Extended Yale-B dan ORL Face. Sistem yang dirancang terdiri dari beberapa tahapan dalam pengolahan data. Secara garis besar terdapat tiga tahap proses dalam sistem. Tahapan tersebut adalah pra-proses yang bertujuan untuk membersihkan data dari *noise* atau data-data yang tidak diperlukan, ekstraksi fitur yang bertujuan untuk mendapatkan informasi-informasi penting yang terdapat dalam data dan tahap klasifikasi yang bertujuan untuk menguji apakah informasi yang didapat dari data benar-benar mencerminkan data tersebut.

3.1 Pra-proses

Data yang akan digunakan perlu melalui pra-proses. Pra-proses ini bertujuan untuk menyeragamkan data dan menghilangkan informasi yang tidak perlu dan beberapa data perlu diubah formatnya agar bisa diproses oleh sistem.

Dengan seragamnya data yang akan diproses, maka hasil yang diberikan pun diharapkan juga lebih baik dari proses tanpa penyeragaman data ini. Penghilangan data yang tidak perlu juga berpengaruh besar terhadap hasil yang akan dikeluarkan oleh sistem. Secara garis besar, pra-proses dapat dipahami dari diagram alir yang terdapat pada Gambar 3.1.



Gambar 3.1 Alur Pra-Proses

Pra-proses data yang terdiri dari:

1. Konversi tipe data menjadi Bitmap.
2. Pemisahan bagian wajah dari latar belakang pada *dataset*.
    a. Deteksi wajah untuk menandai bagian wajah pada *dataset*.
    b. Peningkatan kontras gambar pada *dataset* untuk meningkatkan performa deteksi wajah.
3. Penyeragaman ukuran dimensi data.

3.1.1 Konversi Tipe Data

Konversi tipe data adalah pengubahan format data yang digunakan oleh sistem. Data yang digunakan terdiri dari 3 buah *dataset*. Dataset tersebut memiliki format data *Portable Gray Map* (.pgm) dan *Portable Pixel Map* (.ppm). Format data tersebut tidak dapat diterima oleh sistem.

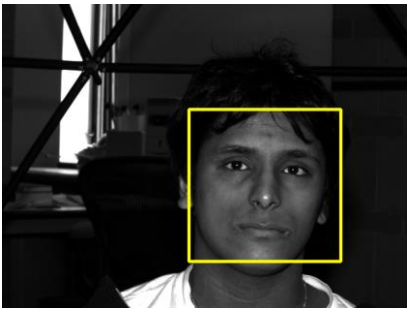
Format data yang bisa diproses oleh sistem adalah gambar dengan format Bitmap. Oleh karena itu, semua data dikonversikan ke dalam bentuk Bitmap. Berkas hasil konversi dapat disimpan dalam direktori komputer atau langsung digunakan untuk tahap selanjutnya. Setiap tipe data akan dibuatkan modul konversinya masing-masing. Modul tersebut adalah modul konverter data PGM menjadi kelas Bitmap C# dan konverter data PPM menjadi kelas Bitmap C#.

3.1.2 Pemisahan Wajah dan Latar Belakang

Tidak semua bagian dari data perlu untuk diekstrak fiturnya. Pada *dataset* terdapat beberapa bagian data yang tidak diperlukan dan justru mengganggu nilai penting yang ada di dalamnya. Data–data yang mengganggu (*noise*) perlu dihilangkan.

Noise pada data adalah bagian gambar yang bukan wajah. Seperti pada Gambar 3.2 di mana gambar latar belakang pengambilan gambar menyusun komposisi yang cukup besar dari keseluruhan gambar. Pada Gambar 3.3 merupakan bagian penting yang diperlukan dari keseluruhan proses. Dari contoh gambar tersebut, hanya bagian pada garis kuning yang akan diambil dan sisanya dibuang. Dari proses ini akan dihasilkan gambar seperti pada Gambar 3.4.

Gambar 3.2 Gambar Asli dari *Dataset*



Gambar 3.3 Bagian Wajah yang Ditandai dengan Garis Kuning

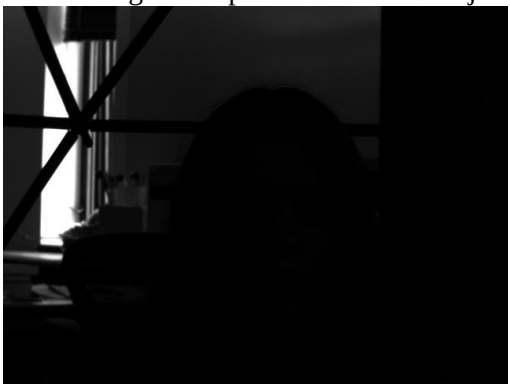
Gambar 3.4 Hasil Pembersihan Data dari *Noise*

Pemisahan wajah dilakukan dengan deteksi wajah dari *dataset* menggunakan Viola-Jones *object-detection* dari pustaka Open CV untuk mempercepat pengerjaan pra-proses sehingga mengurangi pekerjaan manual.

Dari deteksi wajah dengan Open CV akan didapati koordinat bagian gambar yang merupakan wajah. Dari koordinat tersebut dilakukan pemotongan bagian wajah dari gambar asal. Gambar hasil pemisahan tersebut dapat akan disimpan di dalam direktori komputer.

3.1.3 *Histogram Equalization*

Pada beberapa gambar, deteksi wajah Open CV kesulitan menemukan wajah dikarenakan pencahayaan yang sangat gelap sehingga deteksi wajah gagal didapatkan. Oleh karena itu, pada beberapa gambar dilakukan pra-proses ekstra dengan *histogram equalization* untuk meningkatkan performa deteksi wajah.

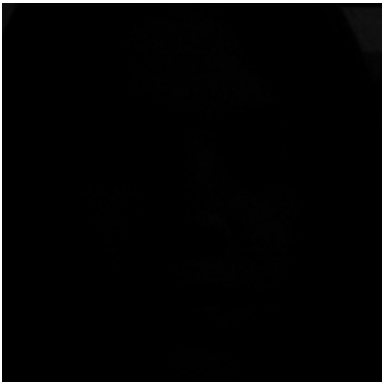


Gambar 3.5 Gambar dengan Pencahayaan Minim

Seperti pada Gambar 3.5 deteksi wajah dari Open CV tidak bisa menemukan wajah pada gambar. Sehingga, dilakukan *histogram equalization* untuk meningkatkan kontras gambar. Dengan peningkatan kontras akan didapatkan gambar seperti pada Gambar 3.6 sehingga wajah bisa dideteksi dari gambar.

Gambar 3.6 Gambar Telah Ditingkatkan Kontrasnya

Dengan peningkatan kontras gambar, wajah dari gambar tersebut bisa ditemukan seperti pada Gambar 3.6. Walaupun begitu gambar yang disimpan bukan potongan gambar yang telah ditingkatkan kontrasnya, melainkan gambar asli seperti pada Gambar 3.7.

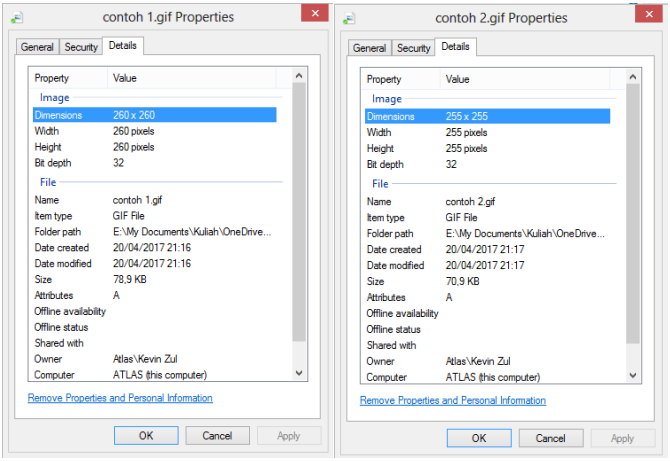


Gambar 3.7 Hasil Potongan Gambar

3.1.4 Penyeragaman Dimensi Data

Pemotongan wajah dari tiap data akan menghasilkan ukuran gambar yang berbeda-beda. Hal ini sangat wajar karena proporsi wajah pada tiap gambar juga berbeda-beda tergantung dengan jarak

kamera dan subjek, ukuran kepala subjek dan lain-lain. Pada Gambar 3.8 didapati dua contoh data yang memiliki ukuran yang berbeda terpaut 5 piksel masing-masing. Perbedaan ini harus diantisipasi dengan mengubah semua data menjadi ukuran yang sama.



Gambar 3.8 Informasi Ukuran Dimensi dari Dua Contoh Data

Penyetaraan ukuran dimensi data menggunakan Open CV. Hasil pengubahan ukuran dimensi berupa data gambar baru yang disimpan dalam direktori komputer. Data ini yang selanjutnya akan digunakan untuk ekstraksi fitur *Local Directional Pattern.*

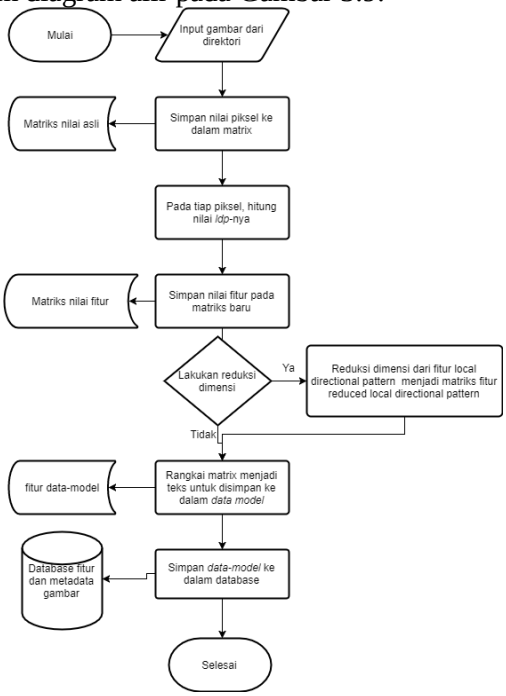
Antisipasi hilangnya informasi dari penyusutan ukuran data, dilakukan pengubahan ukuran data menjadi sepasang. Pasangan data tersebut adalah data yang ukurannya disusutkan menjadi ukuran yang serupa dengan data lain yang disusutkan dan data yang diperbesar serupa dengan data lain.

3.2 Ekstraksi Fitur

Pada bagian ini akan dijelaskan ekstraksi fitur *Local Directional Pattern* dan reduksi dimensi pada fitur tersebut sehingga didapatkan fitur *Reduced Dimensionality Local Directional Pattern.*

3.2.1 Ekstraksi Nilai *LDP*

Ekstraksi fitur dilakukan dengan mendapatkan nilai *Local Directional Pattern* dari tiap piksel gambar. Tiap piksel dan piksel tetangganya dikorelasikan dengan 8 kernel Kirsch seperti pada Gambar 2.9. Hasil pengambilan nilai *Local Directional Pattern* akan berbentuk matriks. Matriks tersebut akan dirangkai menjadi teks untuk disimpan dalam *database*. Jalannya ekstraksi fitur bisa dilihat dalam diagram alir pada Gambar 3.9.



Gambar 3.9 Diagram Alir Proses Ekstraksi Fitur Dimensionality Reduced Local Directional Pattern

Penanganan operasi kernel pada piksel tepian dilakukan dengan mengabaikannya, karena piksel-piksel ini tidak memiliki piksel tetangga yang diperlukan untuk operasi kernel. Hal ini menghasilkan ukuran gambar yang digunakan untuk ekstraksi fitur

harus dilebihkan 1 piksel pada masing-masing tepi agar ukuran hasil sesuai dengan yang diinginkan.

3.2.2 Reduksi Dimensi

Sistem akan menghasilkan 2 macam ekstraksi fitur. Selain ekstraksi nilai *LDP*, sistem juga dapat mereduksi dimensi dari nilai *LDP* tersebut. Hal ini dimaksudkan agar bisa dihasilkan perbandingan antara metode reduksi yang digunakan dengan metode tanpa reduksi dimensi. Pada Gambar 3.1 data yang hendak disimpan dalam *database* dapat ditentukan apakah direduksi atau tidak. Maka, sistem bisa menghasilkan 2 ekstraksi fitur yaitu fitur *LDP* dan *DRLDP*.

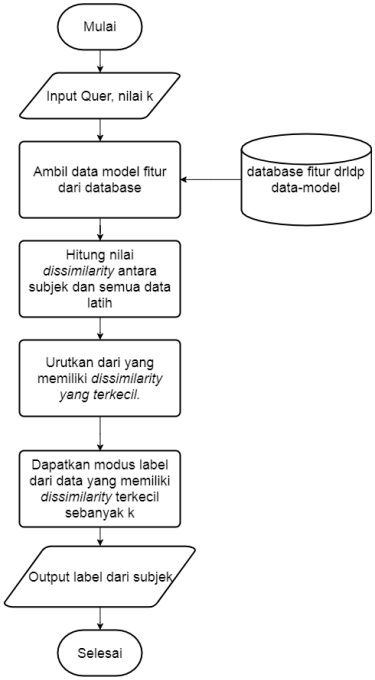
Kelas *data-model* merupakan kelas tersendiri untuk menyimpan informasi dari data yang diproses. Kelas *data-model* adalah kelas yang digunakan untuk menyimpan data dan informasi lainnya ke dalam *database*. Informasi tersebut berupa:

1. Label kelas jika kelas diketahui.
2. Ukuran dimensi matriks *Dimensionality Reduced Local Directional Pattern*.
3. Nama berkas gambar yang ada di dalam direktori.
4. Matriks data *Dimensionality Reduced Local Directional Pattern*.
5. Nama *dataset* asal data tersebut.
6. Data *Dimensionality Reduced Local Directional Pattern* yang berbentuk teks.
7. Ukuran data, ukuran data yang dimaksud adalah apakah data ini termasuk data yang disusutkan atau diperbesar ukurannya.
8. Fungsi *parse* dari data matriks ke teks dan sebaliknya.

Nilai-nilai pada matriks *DRLDP* ini nanti yang akan digunakan sebagai histogram deskriptor dari data. Kelas ini menyimpan matriks ke dalam *database* dengan mengubah formatnya menjadi teks.

3.3 Klasifikasi

Pada bagian ini akan menjelaskan pengolahan fitur yang sudah diterima untuk dilakukan klasifikasi dengan K-*Nearest Neighbor* dan *Chi-Square Dissimilarity*. Klasifikasi dari data yang telah diekstrak fiturnya dilakukan dengan penghitungan tingkat kemiripan histogram *Dimensionality Reduced Local Directional Pattern*-nya. Dari tingkat kemiripan tersebut diambil sebanyak k data dengan tingkat kemiripan tertinggi untuk diambil modus labelnya.

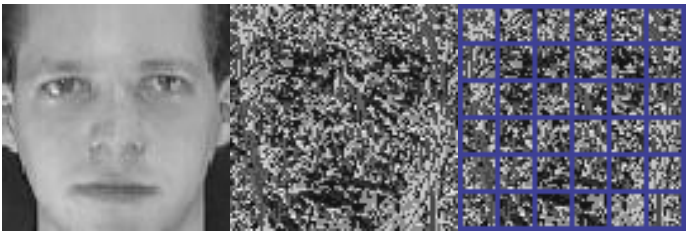


Gambar 3.10 Diagram Alir Klasifikasi

3.3.1 Fragmentasi Histogram

Dari matriks *LDP* akan disusun histogram sebagai deskriptornya. Histogram yang disusun terdiri dari beberapa blok sub-histogram. Blok – blok sub-histogram ini yang akan digunakan

untuk menghitung tingkat kemiripan data secara keseluruhan. Pada Gambar 3.11 merupakan ilustrasi bagaimana gambar hasil ekstraksi fitur dipisah menjadi beberapa blok yang berukuran lebih kecil.



Gambar 3.11 (kiri) Gambar Asli (tengah) Gambar dari Matriks *LDP* (kanan) Matriks yang Telah Dibagi Menjadi Beberapa Sub-Gambar

Perhitungan kemiripan dilakukan dengan perhitungan fitur pada tiap pasang sub-histogram di mana nilai ketidakmiripannya dijumlahkan dari tiap sub-histogram. Hal ini karena tidak semua blok gambar memiliki bobot yang sama dalam mewakili sifat yang dibawanya.

3.3.2 Perhitungan *Chi-Square Dissimilarity*

Dari lokal-histogram yang telah didapatkan. Nilai ketidakmiripan dapat dihitung dengan Persamaan 2.4 pada tiap pasang lokal-histogram ke-*i* antara data uji dan data latih yang telah ditentukan. Nilai *dissimilarity* antar data adalah sama dengan akumulasi nilai *Chi-Square Dissimilarity* dari semua lokal-histogram yang terdapat pada data.

3.3.3 Klasifikasi dengan *K-Nearest Neighbor*

Dari hasil perhitungan nilai *dissimilarity* antara subjek dan semua data latih. Diambil sebanyak k data yang memiliki nilai *dissimilarity* terendah. Dari k-data tersebut dicari nilai modusnya. Di mana label dari data subjek sesuai dengan label modus data tersebut.

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak dari rancangan metode yang telah dibahas pada Bab III meliputi kode program dalam perangkat lunak yang telah diterapkan dalam bahasa C#.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i3-4150 CPU @3.50GHz RAM: 4,00GB
Perangkat lunak	Sistem Operasi: Windows 10 64-bit Perangkat Pengembang: Microsoft Visual Studio Community Edition 2015 Perangkat *Database*: MySQL Server, SqlYog Perangkat Bantu: Notepad++, Ms Excel

4.2 Implementasi Tahap Pra-proses

Bagian ini akan menjelaskan implementasi pra-proses yang terdiri dari beberapa tahap.

4.2.1 Konversi Tipe Data

Tahap ini terdiri dari dua modul konversi gambar dari .ppm dan .pgm menjadi kelas Bitmap. Konversi tipe data akan membaca berkas asli yang berformat .ppm dan .pgm. Dari dua kode sumber pada Kode Sumber 4.1 dan Kode Sumber 4.5. Hasil bacaan berkas diubah menjadi kelas Bitmap C# dengan nilai rgb gambar yang seragam pada tiap pikselnya sehingga dihasilkan gambar keabu-abuan.

```
1.  using System.IO;
2.  using System.Drawing;
3.  class PPMReader
4.  {
5.      public static Bitmap ReadBitmapFromPPM(string file)    {
6.          var reader = new BinaryReader(new FileStream(file, FileMode.Open));
7.          if (reader.ReadChar() != 'P' || reader.ReadChar() != '6')
8.              return null;
9.          reader.ReadChar(); //Get newline
10.         string widths = "", heights = "";
11.         char temp;
12.         while ((temp = reader.ReadChar()) != ' ')
13.             widths += temp;
14.         while ((temp = reader.ReadChar()) >= '0' && temp <= '9')
15.             heights += temp;
16.         if (reader.ReadChar() != '2' || reader.ReadChar() != '5' || reader.ReadChar() != '5')
17.             return null;
18.         reader.ReadChar(); //Eat the last newline
19.         int width = int.Parse(widths),
20.             height = int.Parse(heights);
21.         Bitmap Bitmap = new Bitmap(width, height);
22.         //Read in the pixels
23.         for (int y = 0; y < height; y++)
24.             for (int x = 0; x < width; x++)
25.                 Bitmap.SetPixel(x, y, Color.FromArgb(reader.ReadByte(), reader.ReadByte(), reader.ReadByte()));
26.         return Bitmap; }}
```

Kode Sumber 4.1 Konversi Format Ppm Menjadi Bitmap

```
1.  using System;
2.  using System.IO;
3.  using System.Drawing;
4.  namespace Tugas_Akhir
5.  {
6.      class PortableGrayMap
7.      {
8.          public int width;
9.          public int heigth;
10.         public int maxVal;
11.         public byte[][] pixels;
12.         public PortableGrayMap(int width, int
    height, int maxVal, byte[][] pixels)
13.         {
14.             this.width = width;
15.             this.heigth = height;
16.             this.maxVal = maxVal;
17.             this.pixels = pixels;
18.         }
19.         public PortableGrayMap(string
    filename)
20.         {
21.             PortableGrayMap x =
    LoadImage(filename);
22.             this.width = x.width;
23.             this.heigth = x.heigth;
24.             this.maxVal = x.maxVal;
25.             this.pixels = x.pixels;
26.         }
27.         public PortableGrayMap
    LoadImage(string file)
28.         {
29.             FileStream inputFileStream = new
    FileStream(file, FileMode.Open);
30.             BinaryReader binaryReader = new
    BinaryReader(inputFileStream);
```

Kode Sumber 4.2 Konversi Pgm Menjadi Bitmap (Bagian Pertama)

```
31.             string kodeJenisFile = NextNonCommentLine(binaryReader);
32.             if (kodeJenisFile != "P5")
33.             {
34.                 throw new Exception("Non PGM file type" + kodeJenisFile);
35.             }
36.             string widthHeight = NextNonCommentLine(binaryReader);
37.             string[] tokens = widthHeight.Split(' ');
38.             int width = int.Parse(tokens[0]);
39.             int height = int.Parse(tokens[1]);
40.             string sMaxVal = NextNonCommentLine(binaryReader);
41.             int maxVal = int.Parse(sMaxVal);
42.             byte[][] pixels = new byte[height][];
43.             for(int i=0; i<height; ++i)
44.             {
45.                 pixels[i] = new byte[width];
46.             }
47.             for(int i=0; i<height; ++i)
48.             {
49.                 for(int j=0; j<width; ++j)
50.                 {
51.                     pixels[i][j] = binaryReader.ReadByte();
52.                 }
53.             }
54.             binaryReader.Close();
55.             inputFileStream.Close();
56.             PortableGrayMap hasil = new PortableGrayMap(width, height, maxVal, pixels);
57.             return hasil;
```

Kode Sumber 4.3 Konversi Pgm Menjadi Bitmap (Bagian Kedua)

```
58.         }
59.         static string NextAnyLine(BinaryReader
            br)
60.         {
61.             string s = "";
62.             byte b = 0;//dummy
63.             while(b!=10)//enter atau baris
            baru
64.             {
65.                 b = br.ReadByte();
66.                 char c = (char)b;
67.                 s += c;
68.             }
69.             return s.Trim();
70.         }
71.         static string
            NextNonCommentLine(BinaryReader br)
72.         {
73.             string s = NextAnyLine(br);
74.             while (s.StartsWith("#") || s ==
            "")
75.             {
76.                 s = NextAnyLine(br);
77.             }
78.             return s;
79.         }
80.         public Bitmap
            MakeBitmap(PortableGrayMap gambarPgm, int mag)
81.         {
82.             int width = gambarPgm.width*mag;
83.             int height = gambarPgm.heigth*mag;
84.             Bitmap result = new Bitmap(width,
            height);
85.             Graphics gr =
            Graphics.FromImage(result);
```

Kode Sumber 4.4 Konversi Pgm Menjadi Bitmap (Bagian Ketiga)

```
86.             for(int i=0; i<gambarPgm.heigth; ++i)
87.             {
88.                 for (int j=0; j<gambarPgm.width; ++j)
89.                 {
90.                     int pixelColor = gambarPgm.pixels[i][j];
91.                     Color warna = Color.FromArgb(pixelColor, pixelColor, pixelColor);
92.                     SolidBrush sb = new SolidBrush(warna);
93.                     gr.FillRectangle(sb, j * mag, i * mag, mag, mag);
94.                 }
95.             }
96.             return result;
97.         }
98.     }
99. }
```

Kode Sumber 4.5 Konversi Pgm Menjadi Bitmap (Bagian Keempat)

4.2.2 Implementasi Peningkatan Kontras Gambar

Implementasi peningkatan kontras gambar dengan pustaka OpenCV digunakan pada modul ImageCrop.cs pada baris 49 hingga 52. Peningkatan kontras dilakukan dengan memperluas distribusi warna pada histogram gambar sehingga dihasilkan gambar dengan warna yang lebih kontras.

```
49.     if (this.UseHistogramEqualiztion == true)
50.             {
51.                 grayImage._EqualizeHist();
52.             }
```

Kode Sumber 4.6. Implementasi Histogram *Equalization* pada Kelas ImageCrop

4.2.3 Implementasi Deteksi Wajah

Implementasi deteksi wajah dengan pustaka Open CV pada modul ImageCrop.cs pada baris nomor 46 hingga 54. Masukan dari deteksi wajah adalah gambar asli atau yang telah ditingkatkan kontrasnya. Dari gambar tersebut akan dicari pada bagian mana yang terdapat fitur wajah. Keluaran yang didapatkan dari sini adalah objek *rectangle* C# yang menandai bagian wajah dari gambar.

```
46.  private void getFace()
47.          {
48.              Image<Gray, byte> grayImage = new
     Image<Gray, byte>(this.Gambar);
49.              if (this.UseHistogramEqualiztion
     == true)
50.              {
51.                  grayImage._EqualizeHist();
52.              }
53.              this.CropRectangle =
     haarCascade.DetectMultiScale(grayImage, 1.01,
     4, new Size(this.Min, this.Min), new
     Size(this.Max, this.Max));
54.          }
```

Kode Sumber 4.7 Implementasi Deteksi Letak Wajah pada Gambar.

4.2.4 Implementasi Pemisahan Bagian Wajah

Proses pemisahan bagian wajah dari latar belakang diimplementasikan pada modul ImageCrop.cs yang terdiri dari deteksi wajah, peningkatan kontras dan pembuatan gambar baru pada baris 34 hingga 40 dari .

Pemotongan bagian wajah dilakukan dengan duplikasi gambar asli yang telah dijadikan masukan. Pada gambar baru, semua nilai piksel yang terdapat dalam objek *rectangle* yang dihasilkan pada deteksi wajah disalin sesuai dengan ukuran objek *rectangle*. Gambar baru ini selanjutnya disimpan pada direktori komputer.

```
1.  using System;
2.  using System.IO;
3.  using System.Drawing;
4.  using Emgu.CV;
5.  using Emgu.CV.Structure;
6.
7.  namespace Tugas_Akhir
8.  {
9.      class ImageCrop
10.     {
11.         Bitmap Gambar;
12.         int Min;
13.         int Max;
14.         bool UseHistogramEqualiztion;
15.         Rectangle[] CropRectangle;
16.         CascadeClassifier haarCascade = new
    CascadeClassifier(System.Configuration.Config
    urationManager.AppSettings["1"]);
17.         public ImageCrop(Bitmap gambarAsal,
    Rectangle[] areaCrop)
18.         {
19.             this.Gambar = gambarAsal;
20.             this.CropRectangle = areaCrop;
21.         }
22.         public ImageCrop(Bitmap gambarAsal,
    int min, int max, bool useHisteq)
23.         {
24.             this.Gambar = gambarAsal;
25.             this.Min = min;
26.             this.Max = max;
27.             this.UseHistogramEqualiztion =
    useHisteq;
28.             getFace();
29.         }
30.         public Bitmap[] getImages()
```

Kode Sumber 4.8 Implementasi Pemotongan Bagian Wajah (Bagian Pertama)

```
31.         {
32.             Bitmap[] hasil = new
Bitmap[this.CropRectangle.Length];
33.             for (int i=0;
i<this.CropRectangle.Length; i++)
34.             {
35.                 hasil[i] = new
Bitmap(this.CropRectangle[i].Width,
this.CropRectangle[i].Height);
36.                 for (int x=0;
x<this.CropRectangle[i].Width; x++)
37.                 {
38.                     for(int y=0;
y<this.CropRectangle[i].Height; y++)
39.                     {
40.                         hasil[i].SetPixel(x,
y,
this.Gambar.GetPixel(this.CropRectangle[i].X
+ x, this.CropRectangle[i].Y + y));}}}
41.             return hasil;
42.         }
43.         private void getFace()
44.         {
45.             Image<Gray, byte> grayImage = new
Image<Gray, byte>(this.Gambar);
46.             if (this.UseHistogramEqualiztion
== true)
47.                 {grayImage._EqualizeHist();}
48.             this.CropRectangle =
haarCascade.DetectMultiScale(grayImage, 1.01,
4, new Size(this.Min, this.Min), new
Size(this.Max, this.Max));
49.         }
50.     }
51. }
```

Kode Sumber 4.9 Implementasi Pemotongan Bagian Wajah (Bagian Kedua)

4.2.5 Implementasi Penyetaraan Dimensi Data

Pengubahan ukuran gambar diimplementasikan pada modul *thread* ThreadToResize.cs pada baris 25 untuk ukuran yang disusutkan dan 26 untuk ukuran yang dibesarkan.

Implementasi dilakukan pada kelas modul yang berjalan sebagai *thread* pada sistem. Hal ini dimaksudkan agar proses penyetaraan dimensi gambar lebih cepat selesai. Implementasi penyetaraan ini menggunakan fungsi dari Open CV.

```
1.  using System.IO;
2.  using Emgu.CV;
3.  using Emgu.CV.Structure;
4.
5.  namespace Tugas_Akhir
6.  {
7.      class ThreadToResize
8.      {
9.          string[] listFile;
10.         string destinationDirectory;
11.         int minSize;
12.         int maxSize;
13.         public ThreadToResize(string[]
fileName, int minSize, int maxSize, string
destinationDirectory)
14.         {
15.             this.listFile = fileName;
16.             this.minSize = minSize;
17.             this.maxSize = maxSize;
18.             this.destinationDirectory =
destinationDirectory;
19.         }
20.         public void Resize()
21.         {
22.             foreach(string file in listFile)
23.             {
```

Kode Sumber 4.10 Implementasi Pengubahan Ukuran Data (Bagian Pertama)

```
24.                 Image<Gray, byte> picture =
new Image<Gray, byte>(file);
25.                 picture.Resize(this.minSize,
this.minSize,
Emgu.CV.CvEnum.Inter.Nearest).ToBitmap().Save
(this.destinationDirectory + "/Mini " +
Path.GetFileName(file));
26.                 picture.Resize(this.maxSize,
this.maxSize,
Emgu.CV.CvEnum.Inter.Nearest).ToBitmap().Save
(this.destinationDirectory + "/Maxi " +
Path.GetFileName(file));
27.             }
28.         }
29.     }
30. }
```

Kode Sumber 4.11 Implementasi Pengubahan Ukuran Data (Bagian Kedua)

4.3 Implementasi Ekstraksi Fitur

Keseluruhan ekstraksi fitur diimplementasikan pada modul DRLocalDirectionalPattern.cs. Sehingga, penjelasan akan dipisah menjadi beberapa bagian sesuai dengan fungsi-fungsi yang terdapat di dalam modul ini. Modul DRLocalDirectionalPattern.cs dapat dilihat pada Kode Sumber 4.20.

```
1.  using System;
2.  using System.Drawing;
3.  using Accord.Statistics.Analysis;
4.  using Accord.Statistics.Kernels;
5.  namespace Tugas_Akhir
6.  {
```

Kode Sumber 4.12 Implementasi Ekstraksi Fitur (Bagian Pertama)

```
7.      class DRLocalDirectionalPattern
8.      {
9.          int[,] kirschMask;//kernel
10.         int[,] originalMatrix;//original green chanel image matrix
11.         public int[,] ldpResult;//ldp coded image matrix
12.         public Byte[,] drldpMatrix;//reduced dimension of ldpResult
13.         private void initMask()//genereating the kirsch mask
14.         {
15.             this.kirschMask = new int[8, 9] { {-3,-3, 5,-3, 0, 5,-3,-3, 5},//m0
16.             {-3, 5, 5,-3, 0, 5,-3,-3,-3},//m1
17.             { 5, 5, 5,-3, 0,-3,-3,-3,-3},//m2
18.             { 5, 5,-3, 5, 0,-3,-3,-3,-3},//m3
19.             { 5,-3,-3, 5, 0,-3, 5,-3,-3},//m4
20.             {-3,-3,-3, 5, 0,-3, 5, 5,-3},//m5
21.             {-3,-3,-3,-3, 0,-3, 5, 5, 5},//m6
22.             {-3,-3,-3,-3, 0, 5,-3, 5, 5},//m7};
23.         }
24.         public DRLocalDirectionalPattern(Bitmap inputImage)//basic constructor for this class
```

Kode Sumber 4.13 Implementasi Ekstraksi Fitur (Bagian Kedua)

```
25.         {
26.             initMask();
27.             generateInitialMatrix(inputImage);
28.         }
29.         private void generateInitialMatrix(Bitmap inputImage)//get the initial matrix of image from green chanel of the image
30.         {
31.             this.originalMatrix = new int[inputImage.Width, inputImage.Height];
32.             for(int i=0; i<inputImage.Width; i++)
33.             {
34.                 for(int j=0; j<inputImage.Height; j++)
35.                 {
36.                     this.originalMatrix[i, j] = Convert.ToInt32(inputImage.GetPixel(i, j).R.ToString());
37.                 }
38.             }
39.         }
40.         private int correlateMask(int[] mask, int[] imageMat)//used to get the correlation matrix of original image and kirsch mask
41.         {
42.             int acumulation = 0;
43.             for(int i=0; i<9; i++)
44.             {
```

Kode Sumber 4.14 Implementasi Ekstraksi Fitur (Bagian Ketiga)

```
45.                 acumulation += mask[i] * imageMat[i];
46.             }
47.             return (acumulation>0)?acumulation:-acumulation; //result converted to keep the result always positive
48.         }
49.
50.         private int getLdpCode(int[] matrixBlock)//get ldpcode for each pixel block of image
51.         {
52.             string ldpBinaryCode = "";
53.             int[] ldpMatrixSequence = new int[8];//1x9 matrix containing the result of correlation matrix
54.             for (int i=0; i<8; i++)
55.             {
56.                 int[] subMask = new int[9];//used to fetch mask from 2 dimension array to 1 dimension array
57.                 for(int j=0; j<9; j++)
58.                 {
59.                     subMask[j] = this.kirschMask[i, j];
60.                 }
61.                 ldpMatrixSequence[i] = correlateMask(subMask, matrixBlock);
62.             }
```

Kode Sumber 4.15 Implementasi Ekstraksi Fitur (Bagian Keempat)

```
63.             int[] temporal = new int[9];
64.             ldpMatrixSequence.CopyTo(temporal,0);
65.             Array.Sort(temporal);
66.             int treshold = temporal[6];
67.             for(int x=0; x<ldpMatrixSequence.Length; x++)
68.             {
69.                 if (ldpMatrixSequence[x] < treshold)
70.                 {
71.                     ldpMatrixSequence[x] = 0;
72.                 }
73.                 else
74.                 {
75.                     ldpMatrixSequence[x] = 1;
76.                 }
77.             }
78.             for(int i=7; i>=0; i--)
79.             {
80.                 ldpBinaryCode += ldpMatrixSequence[i].ToString();//concatenating binary to string
81.             }
82.             return Convert.ToInt32(ldpBinaryCode, 2);//convert from binary string to decimal integers has been tested
83.         }
```

Kode Sumber 4.16 Implementasi Ekstraksi Fitur (Bagian Kelima)

```
84.         private int[] getMax(int[] asli, int ammount)//to get most significatn bit
85.         {
86.             int[] data = new int[9];
87.             asli.CopyTo(data, 0);
88.             Array.Sort<int>(data, new Comparison<int>((a, b) => (b.CompareTo(a))));//sort descending, lambda expression has been tested
89.             int[] result = new int[3];
90.             for(int i=0; i<ammount; i++)
91.             {
92.                 result[i] = data[i];
93.             }
94.             return result;
95.         }
96.
97.         private void getLdpCodedImage()//do the ldp generating code to the entire image
98.         {
99.             this.ldpResult = new int[this.originalMatrix.GetLength(0)-2,this.originalMatrix.GetLength(1)-2];
100.            //int p = 0;
101.            for(int x=1; x<this.originalMatrix.GetLength(0)-1; x++)
102.            {
103.                for(int y=1; y<this.originalMatrix.GetLength(1)-1; y++)
104.                {
```

Kode Sumber 4.17 Implementasi Ekstraksi Fitur (Bagian Keenam)

```
105.                    int[] matrixChunks = new int[9];//block of image needed to be correlated with the kirsch mask
106.                    int matIndex = 0;
107.                    for(int i=-1; i<2; i++)
108.                    {
109.                        for(int j=-1; j<2; j++)
110.                        {
111.
matrixChunks[matIndex] = originalMatrix[x + i, y + j];
112.                            matIndex++;
113.                        }
114.                    }
115.                    this.ldpResult[x-1,y-1]=getLdpCode(matrixChunks);//or here
116.                }
117.            }
118.        }
119.        private                                 void dimensionReduction()//reduce dimension by xor operation
120.        {
121.            int             size            = this.ldpResult.GetLength(0);
122.            this.drldpMatrix = new Byte[size/3, size/3];
123.            for(int  i=0;       i<size;  i  += 3)//shifting block
124.            {
125.                for(int  j=0;     j<size;  j  += 3)//shifting block
```

Kode Sumber 4.18 Implementasi Ekstraksi Fitur (Bagian Ketujuh)

```
126.                 {
127.                     #region maximum this is good
128.                     int drldp = 0;
129.                     for (int x = 0; x < 3; x++)
130.                     {
131.                         for (int y = 0; y < 3; y++)
132.                         {
133.                             if (drldp < this.ldpResult[i + x, j + y])
134.                                 drldp = ldpResult[i + x, j + y];
135.                         }
136.                     }
137.                     this.drldpMatrix[i / 3, j / 3] = Convert.ToByte(drldp);
138.                     #endregion
139.                 }
140.             }
141.         }
142.         public Byte[,] getDRLDPMatrix()
143.         {
144.             getLdpCodedImage();
145.             dimensionReduction();
146.             return this.drldpMatrix;
147.         }
148.         //this part of code is finished and tested.
```

Kode Sumber 4.19 Implementasi Ekstraksi Fitur (Bagian Kedelapan)

```
149.        private     Byte     binaryChecker(Byte input)//to check that maximum '1' occurs three times in a binary string
150.        {
151.            int oneCounter = 0;
152.            char[]          result          = Convert.ToString(input, 2).ToCharArray();
153.            for (int i = 0; i < result.Length; i++)
154.            {
155.                if (result[i] == '1')
156.                {
157.                    oneCounter++;
158.                    if (oneCounter == 3)
159.                    {
160.                        for (int j = i + 1; j < result.Length; j++)
161.                        {
162.                            result[j] = '0';
163.                        }
164.                        break;
165.                    }
166.                }
167.            }
168.            return          Convert.ToByte(new string(result), 2);//return decimal integer
169.        }
170.    }
171. }
```

Kode Sumber 4.20 Implementasi Ekstraksi Fitur (Bagian Kesembilan)

4.3.1 Implementasi *Local Directional Pattern*

Implementasi *Local Directional Pattern* dapat dilihat pada Kode Sumber 4.20 baris 97 hingga 118. Pada fungsi tersebut, gambar asli dihitung nilai-nilai *ldp*-nya dengan mengoperasikan *Kirsch-mask* pada tiap piksel gambar. Pada fungsi ini memiliki tugas mencacah piksel masukan menjadi blok-blok matriks yang akan dijadikan masukan pada fungsi menghitung nilai *LDP*.

4.3.2 Implementasi Perhitungan Nilai *LDP*

Implementasi perhitungan nilai *LDP* dapat dilihat pada Kode Sumber 4.20 baris 50 hingga 83. Pada fungsi ini, dilakukan operasi korelasi terhadap masukan blok piksel dan *Kirsch-mask*. Sehingga didapatkan matriks yang menyimpan nilai magnitudo tiap arahnya. Selanjutnya, dari nilai tersebut dicari 3 nilai magnitudo tersignifikan. Dari 3 nilai tersebut didapati kode biner *LDP*. Kode biner ini selanjutnya diubah menjadi bilangan desimal sebagai keluaran yang menggambarkan kode *LDP* piksel tersebut.

4.3.3 Implementasi Korelasi Matriks dengan *Kirsch Mask*

Implementasi korelasi matriks dengan data dapat dilihat pada Kode Sumber 4.20 baris 40 hingga 48. Pada fungsi ini, nilai dari tiap matriks yang telah menjadi *array*. Dari kedua *array* tersebut, tiap nilainya dikalikan sesuai dengan Persamaan 2.4. Nilai akumulasi yang didapatkan diubah menjadi positif dan dijadikan keluaran kembali ke fungsi hitung nilai *LDP*.

4.3.4 Implementasi Reduksi Dimensi

Implementasi reduksi dimensi pada fitur *LDP* dapat dilihat pada Kode Sumber 4.20 baris 119-141. Pada fungsi ini, blok 3*3 akan digeser 3 piksel secara horizontal dan vertikal. Dan tiap iterasinya akan mengambil piksel dengan nilai *LDP* tertinggi yang dimasukkan ke dalam matriks baru sebagai fitur *DR-LDP*.

4.4 Implementasi Klasifikasi

4.4.1 Implementasi Proses *K-Nearest Neighbor*

Implementasi klasifikasi *K-Nearest Neighbor* ada pada modul KNearest.cs seperti pada Kode Sumber 4.23 Implementasi *K-Nearest* . Pada modul ini, data uji dan data-data uji digunakan untuk klasifikasi. Pada tiap data uji, dihitung nilai *Chi-Square Dissimilarity*-nya. Dari semua nilai tersebut, diambil sebanyak k data yang memiliki *dissimilarity* terkecil. Dari k data tersebut label modus dari data yang terpilih sebagai label data uji.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using Tugas_Akhir.Selfmade_DataStructure;
4.
5.  namespace Tugas_Akhir
6.  {
7.      class KNearest
8.      {
9.          byte[,] dataTest;
10.         List<byte[,]> dataTrain;
11.         List<Couple> chiDistances;
12.         int K_constanta;
13.         int fragment_constanta;
14.         public KNearest(byte[,] dataTest,
List<byte[,]> dataTrain, List<string>
labelTrain, int K_constanta, int
numberFragment)
15.         {
16.             this.fragment_constanta =
numberFragment;
17.             this.dataTest = dataTest;
18.             this.K_constanta = K_constanta;
19.             if (dataTrain.Count !=
labelTrain.Count)
```

Kode Sumber 4.21 Implementasi *K-Nearest Neighbor* (Bagian Pertama)

```
20.             {
21.                 throw new ArgumentException("Number of feature not equal to number of label");
22.             } else
23.             {
24.                 chiDistances = new List<Couple>();
25.                 this.dataTrain = dataTrain;
26.                 for (int i = 0; i < labelTrain.Count; i++)
27.                 {
28.                     chiDistances.Add(new Couple(0, labelTrain[i]));
29.                 }
30.             }
31.         }
32.         private void calculateDistances()
33.         {
34.             for (int i = 0; i < dataTrain.Count; i++)
35.             {
36.                 ChiSquareDissimilarity chiObj = new ChiSquareDissimilarity(dataTest, dataTrain[i], this.fragment_constanta);
37.
38.                 chiDistances[i].Distance = chiObj.CalculateDissimilarityValue();
39.             }
40.             chiDistances.Sort((s1, s2) => s1.Distance.CompareTo(s2.Distance));
41.         }
42.         public string getClass()
43.         {
44.             calculateDistances();
```

Kode Sumber 4.22 Implementasi *K-Nearest Neighbor* (Bagian Kedua)

```
45.             return getMostNeighbour();
46.         }
47.         private string getMostNeighbour()
48.         {
49.             List<Pair> topK = new List<Pair>();
50.             for(int i=0; i<this.K_constanta;
i++)
51.             {
52.                 topK.Add(new
Pair(this.chiDistances[i].Label));
53.             }
54.             for(int i=0; i<this.K_constanta;
i++)
55.             {
56.                 for(int j=0; j<topK.Count; j++)
57.                 {
58.                     if (topK[j].Key ==
chiDistances[i].Label)
59.                     {
60.                         topK[j].Value++;
61.                         break;
62.                     }
63.                     if (j == this.K_constanta -
1)
64.                     {
65.                         Pair pairObj = new
Pair(chiDistances[i].Label);
66.                         topK.Add(pairObj);
67.                     }
68.                 }
69.             }
70.             topK.Sort((s1, s2) =>
s1.Value.CompareTo(s2.Value));
71.             return topK[0].Key;}}}
```

Kode Sumber 4.23 Implementasi *K-Nearest Neighbor* (Bagian Kedua)

4.4.2 Implementasi Rumus *Chi-Square Dissimilarity*

Chi-Square Dissimilarity diimplementasikan pada modul ChiSquareDissimilarity.cs seperti pada Kode Sumber 4.31. Pada modul ini, nilai *dissimilarity* dihitung dengan mengakumulasikan nilai *Chi-Square Dissimilarity* yang ada pada tiap lokal histogram gambar. Sehingga gambar masukan dicacah terlebih dahulu menjadi beberapa bagian.

```
1.  using System.Collections.Generic;
2.  using System;
3.  namespace Tugas_Akhir
4.      {class ChiSquareDissimilarity
5.      {
6.          protected double Dissimilarity;
7.          protected int NumberofFragment;
8.          protected byte[,] TestFeature;
9.          protected byte[,] TrainFeature;
10.         protected byte[,] TestFragment;
11.         protected byte[,] TrainFragment;
12.         /// <summary>
13.         /// TestFeature and TrainFeature only
    can be set from this constructor
14.         /// </summary>
15.         /// <param name="TestFeature">2
    dimensional array for target data</param>
16.         /// <param name="TrainFeature">2
    dimensional array for target data</param>
17.         /// <param
    name="NumberofFragment">Number region to be
    fragmented</param>
18.         public ChiSquareDissimilarity(byte[,]
    TestFeature, byte[,] TrainFeature, int
    NumberofFragment)
19.         {
```

Kode Sumber 4.24 Implementasi *Chi-Square Dissimilarity* (Bagian Pertama)

```
20.             if (TestFeature.Length != TrainFeature.Length || TestFeature.Length % NumberofFragment != 0)
21.             {
22.                 throw new System.ArgumentException("Input of array should have same length of element", "featureDRLDP1 + featureDRLDP2 or fragment factor not multiplication of 3");
23.             }
24.             this.TestFeature = TestFeature;
25.             this.TrainFeature = TrainFeature;
26.             this.NumberofFragment = NumberofFragment;
27.             FragmentingMatrix();
28.         }
29.         private void FragmentingMatrix()
30.         {
31.             int dimension = this.TestFeature.GetLength(0);
32.             int FragmentSize = dimension / this.NumberofFragment;
33.             this.TestFragment = new byte[this.NumberofFragment * this.NumberofFragment, FragmentSize * FragmentSize];
34.             this.TrainFragment = new byte[this.NumberofFragment * this.NumberofFragment, FragmentSize * FragmentSize];
35.             int i;
36.             for (int xCriterion = 0, FragmentIndex = 0; xCriterion < dimension; xCriterion += FragmentSize)//Shift horizontal block
```

Kode Sumber 4.25 Implementasi *Chi-Square Dissimilarity* (Bagian Kedua)

```
37.             {
38.                 for (int yCriterion = 0;
    yCriterion < dimension; yCriterion +=
    FragmentSize, FragmentIndex++)//shift vertical
    block
39.                 {
40.                     i = 0;
41.                     for (int xIndex = 0; xIndex
    < FragmentSize; xIndex++)
42.                     {
43.                         for (int yIndex = 0;
    yIndex < FragmentSize; yIndex++)
44.                         {
45.                             try
46.                             {
47.
    this.TestFragment[FragmentIndex, i++] =
    this.TestFeature[xIndex + xCriterion, yIndex +
    yCriterion];
48.                             }
49.                             catch
50.                             {
51.
    System.Diagnostics.Debug.WriteLine("gagal " +
    xIndex + " " + yIndex + " " + xCriterion + " " +
    yCriterion);
52.                             }
53.                         }
54.                     }
55.                 }
56.             }
57.
58.             for (int xCriterion = 0,
    FragmentIndex = 0; xCriterion < dimension;
    xCriterion += FragmentSize)//Shift horizontal
    block
```

Kode Sumber 4.26 Implementasi *Chi-Square Dissimilarity* (Bagian Ketiga)

```
59.             {
60.                 for (int yCriterion = 0; yCriterion < dimension; yCriterion += FragmentSize, FragmentIndex++)//shift vertical block
61.                 {
62.                     i = 0;
63.                     for (int xIndex = 0; xIndex < FragmentSize; xIndex++)
64.                     {
65.                         for (int yIndex = 0; yIndex < FragmentSize; yIndex++)
66.                         {
67.                             try
68.                             {
69.                                 this.TrainFragment[FragmentIndex, i++] = this.TrainFeature[xIndex + xCriterion, yIndex + yCriterion];
70.                             }
71.                             catch
72.                             {
73.                                 System.Diagnostics.Debug.WriteLine("gagal " + xIndex + " " + yIndex + " " + xCriterion + " " + yCriterion);
74.                             }
75.                         }
76.                     }
77.                 }
78.             }
79.         }
80.         /// <summary>
81.         /// Get the Dissimilarity value for both matrix
```

Kode Sumber 4.27 Implementasi *Chi-Square Dissimilarity* (Bagian Keempat)

```
82.         /// </summary>
83.         /// <returns>return double</returns>
84.         public double
CalculateDissimilarityValue()
85.         {
86.             this.Dissimilarity = 0;
87.             for (int i = 0; i <
this.TestFragment.GetLength(0); i++)//shifting
per
88. block
89.             {
90.                 int[] histogramTest = new
int[256];
91.                 int[] histogramTrain = new
int[256];
92.                 getHistogram(ref histogramTest,
ref histogramTrain, i);
93.                 double weight =
GetWeight(getModeofRegion(this.TestFragment,
this.TrainFragment, i));
94.                 for (int j = 0; j < 256; j++)
95.                 {
96.                     this.Dissimilarity += weight
* (double)((double)Math.Pow(histogramTest[j]
97.  (double)histogramTrain[j], 2) /
(double)(((double)histogramTest[j] +
(double)histogramTrain[j] != 0) ?
(double)histogramTest[j] +
(double)histogramTrain[j] : 1));
98.                 }
99.             }
100             return this.Dissimilarity;
101         }
102
```

Kode Sumber 4.28 Implementasi *Chi-Square Dissimilarity* (Bagian Kelima)

```
103         private void getHistogram(ref int[] histogramTest, ref int[] histogramTrain, int fragmentIndex)
104         {
105             //initiate histogram
106             for (int i = 0; i < 256; i++)
107             {
108                 histogramTest[i] = 0;
109                 histogramTrain[i] = 0;
110             }
111             for (int i = 0; i < this.TestFragment.GetLength(1); i++)
112             {
113                 histogramTest[this.TestFragment[fragmentIndex, i]]++;
114                 histogramTrain[this.TrainFragment[fragmentIndex, i]]++;
115             }
116         }
117         protected double GetWeight(int modes)
118         {
119             if (modes >= 0 && modes < 64)
120             {
121                 return 1;
122             }
123             else if (modes >= 64 && modes < 128)
124             {
125                 return 2;
126             }
127             else if (modes >= 128 && modes < 192)
128             {
129                 return 3;
130             }
```

Kode Sumber 4.29 *Implementasi Chi-Square Dissimilarity* (Bagian Keenam)

```
131             else if (modes >= 192 && modes <
    256)
132             {
133                 return 4;
134             }
135             else
136             {
137                 return 0;//input must be false
    since color coded dr ldp should fall within 0255
138             }
139         }
140         protected int getModeofRegion(byte[,]
    feature1, byte[,] feature2, int index)
141         {
142             List<byte> feat = new List<byte>();
143             for (int i = 0;
    i<feature1.GetLength(1); i++){
144                 feat.Add(feature1[index, i]);
145                 feat.Add(feature2[index, i]);}
146             feat.Sort((s1, s2) =>
    s1.CompareTo(s2));
147             byte modes = 0;
148             int maxScore = 0;
149             int counter = 0;
150             for (int i=feat.Count
151 1; i>0; i
152 )
153             {
154                 if (feat[i] == feat[i
155  1])
156                 {
157                     counter++;
158                 }
159                 else
160                 {
```

Kode Sumber 4.30 Implementasi Chi-Square Dissimilarity (Bagian Ketujuh)

```
161                         if (maxScore < counter)
162                         {
163                             maxScore = counter;
164                             modes = feat[i];
165                         }
166                         else
167                         {
168                             counter = 0;
169                         }
170                     }
171                 }
172                 return modes;
173             }
174             protected byte[,] SliceMatrix(byte[,,] target, int index)//mengambil potongan matrix
175             {
176                 int length = target.GetLength(1);
177                 byte[,] Result = new byte[length, length];
178                 for (int x=0; x<length; x++)
179                 {
180                     for (int y=0; y<length; y++)
181                     {
182                      Result[x, y] = target[index, x, y];
183                     }}
184                 return Result;
185             }}}
```

Kode Sumber 4.31 Implementasi *Chi-Square Dissimilarity* (Bagian Kedelapan)

[*Halaman ini sengaja dikosongkan*]

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

Pada bagian ini akan menjelaskan tentang perangkat yang digunakan dalam melakukan pengujian. Spesifikasi dari perangkat yang digunakan untuk pengujian dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Pengujian

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i3-4150 CPU @3.50GHz RAM: 4,00GB
Perangkat lunak	Sistem Operasi: Windows 10 64-bit Perangkat Pengembang: Microsoft Visual Studio Community Edition 2015 Perangkat *Database*: MySQL Server, SqlYog Perangkat Bantu: Notepad++, Ms Excel

5.2 Deskripsi *Dataset*

Pada bagian ini akan menjelaskan karakteristik dari tiap *dataset* yang akan berpengaruh pada perannya dalam skenario pengujian perangkat lunak.

5.2.1 Extended Yale-B

Dataset ini terdiri dari 28 subjek. Pada tiap subjek memiliki 9 arah kemiringan kepala yang berbeda dan pada tiap arah kemiringan kepala terdapat 64 sudut pencahayaan yang berbeda. Waktu pengambilan gambar pun tidak berselang lama antara satu gambar dengan gambar yang lain. Ekspresi wajah subjek juga relatif konsisten pada tiap gambar yang telah diambil. Dengan begitu, *dataset* ini terdiri dari 16.128 gambar yang berbeda.

Dari *dataset* ini, bisa digunakan untuk mengetahui akurasi dari perangkat lunak pada perbedaan sudut pencahayaan dan kemiringan kepala. Hal ini ditunjang dengan dokumentasi dan penamaan berkas gambar yang mewakili kondisi pengambilan gambar. Penamaan gambar pada *dataset* ini menggunakan format sebagai berikut: [nama subjek]_[kode kemiringan kepala][sudut pencahayaan secara horizontal][sudut pencahayaan secara vertikal]. Sehingga pengujian dengan *dataset* ini lebih mudah dilakukan.

5.2.2 ORL Face

Dataset ini terdiri dari 40 subjek. Pada tiap subjek terdapat 10 gambar. Pada *dataset* ini tiap subjek memiliki variasi perbedaan pada ekspresi wajah, rentang waktu pengambilan gambar, kemiringan kepala, sedikit variasi pencahayaan dan pada beberapa subjek terdapat perbedaan aksesoris wajah.

Pengujian pada *dataset* ini cenderung lebih sulit dilakukan. Hal ini disebabkan minimnya dokumentasi dan penamaan yang baik pada tiap gambar yang disediakan. Kondisi pada tiap gambar juga tidak direpresentasikan oleh nama berkas. Dari data ini, hal yang bisa dipelajari dengan relatif mudah adalah ekspresi wajah dari subjek.

5.3 Skenario dan Hasil Uji coba

Pada bagian ini akan dijelaskan skenario uji coba dan hasil yang diberikan pada tiap skenario. Pada skenario-skenario berikut, nilai k pada *KNN* yaitu 3 dan jumlah lokal histogram yang digunakan 36. Ukuran dimensi dari data yang dimasukkan ke dalam perangkat lunak berukuran 92*92 piksel. Dari masukan tersebut, data keluaran akan berukuran 30*30 piksel.

5.3.1 Skenario Pengujian Terhadap Pengaruh Kemiringan Kepala Subjek

Pada pengujian ini, perangkat lunak akan digunakan untuk melakukan klasifikasi pada data yang telah diseleksi untuk mengetahui seberapa akurat hasil yang diberikan dengan kondisi kemiringan kepala subjek yang berbeda-beda.

Pengujian dilakukan menggunakan *dataset* Extended Yale-B. Pada pengujian ini, variasi hanya dibatasi pada kemiringan kepala subjek. Sehingga, variasi sudut pencahayaan dihilangkan pada pengujian ini. Gambar yang digunakan hanya gambar dengan pencahayaan yang tepat dari depan kepala subjek atau dengan data yang memiliki nama dengan ketentuan [semua subjek]_[semua kemiringan kepala][+00E][+00]. Maka akan didapati 252 data.

Data tersebut dipisah menjadi data latih dan data uji. Data uji terdiri dari 4 gambar dengan kode sudut kemiringan kepala P05A hingga P08A dan data latih terdiri dari 5 gambar dengan kode kemiringan kepala P00A hingga P04A. Sehingga, rasio data uji dan data latih menjadi 4:5. Hasil yang didapatkan pada skenario ini dapat dilihat pada Tabel 5.2.

Tabel 5.2 Hasil Pengujian Terhadap Kemiringan Kepala

Metode	Akurasi	Waktu Eksekusi
LDP	72%	00:00:43
DR-LDP	67%	00:00:07

5.3.2 Skenario Pengujian Terhadap Pencahayaan

Pada pengujian ini, perangkat lunak akan digunakan untuk melakukan klasifikasi pada data yang telah diseleksi untuk mengetahui seberapa akurat hasil yang diberikan dengan kondisi sudut pencahayaan yang berbeda-beda.

Pengujian ini dilakukan menggunakan *dataset* Extended Yale-B. Pada pengujian ini, variasi hanya dibatasi pada perbedaan sudut pencahayaan saja. Sehingga, variasi kemiringan kepala pada data dihilangkan. Dari tiap subjek akan memiliki 45 gambar dengan sudut pencahayaan yang berbeda-beda. Gambar-gambar tersebut adalah gambar yang memiliki sudut pencahayaan vertikal kurang dari sama dengan 20^0 dari garis sejajar dengan wajah subjek. Maka jumlah data yang digunakan sebanyak 1.260 gambar.

Dari 45 gambar tiap subjek tersebut dipisah menjadi data latih dan *dataset* berdasarkan sudut sumber cahaya. Sudut sumber datangnya cahaya yang berasal dari sisi kanan subjek menjadi data uji dan sudut yang berasal dari depan dan sisi kiri subjek menjadi

data latih. Sehingga, didapatkan 16 data uji dan 29 data latih untuk tiap subjek. Hasil yang didapatkan dari skenario ini dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil Pengujian Terhadap Sudut Pencahayaan

Metode	Akurasi	Waktu Eksekusi
LDP	93%	00:12:08
DR-LDP	81%	00:02:42

5.3.3 Skenario Pengujian Terhadap Ekspresi Wajah

Pada pengujian ini, perangkat lunak akan digunakan untuk melakukan klasifikasi pada data yang memiliki variasi berupa perbedaan ekspresi wajah.

Pengujian ini dilakukan menggunakan *dataset* ORL Face. Pada *dataset* ini, kategori ekspresi tidak dicantumkan pada penamaan data. Sehingga, penomoran data pada tiap subjek memiliki variasi ekspresi yang acak pada nomor yang sama. Pada pengujian ini, semua data digunakan. Sehingga, 400 data yang digunakan secara keseluruhan dalam pengujian ini.

Pembagian data uji dan data latih dibagi berdasarkan nomor data. Nomor data 1, 2, 3 dan 10 menjadi data uji. Nomor data 4, 5 hingga 9 menjadi data latih. Sehingga rasio data uji dan data latih menjadi 4:6. Hasil yang didapatkan dari skenario ini dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Pengujian Terhadap Ekspresi Wajah

Metode	Akurasi	Waktu Eksekusi
LDP	87%	00:01:21
DR-LDP	80%	00:00:15

5.3.4 Pengujian dengan Parameter K dan Jumlah Lokal-Histogram Yang Berbeda

Pada bagian ini menampilkan uji coba pada ketiga skenario yang telah disebutkan dengan parameter klasifikasi yang berbeda. Parameter tersebut adalah nilai k dalam *KNN* dan jumlah lokal histogram yang digunakan. Dari pengujian ini, akan dibandingkan

pengaruh parameter tersebut pada tingkat akurasi dan waktu eksekusi yang dibutuhkan.

Jumlah lokal histogram yang digunakan adalah kuadrat dari bilangan faktorial dimensi data. Selain angka-angka tersebut, fragmentasi matriks mustahil dilakukan karena jumlah piksel tidak habis dibagi dengan ukuran blok matriks. Nilai k yang digunakan tertinggi adalah 5. Hal ini dikarenakan pada beberapa skenario, jumlah data latih yang tersedia tidak lebih dari 5 buah data tiap subjek.

Tabel 5.5 Rangkuman Uji Coba Pada Skenario Kemiringan Kepala

		Nilai k pada *KNN*					
		1		3		5	
		Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi
Jumlah Lokal-Histogram	100	00:00:16	67%	00:00:16	71%	00:00:15	69%
	36	00:00:07	68%	00:00:07	67%	00:00:08	68%
	25	00:00:06	63%	00:00:06	63%	00:00:07	61%
	9	00:00:04	50%	00:00:04	51%	00:00:04	49%
	1	00:00:03	9%	00:00:03	9%	00:00:03	10%

Tabel 5.6 Rangkuman Uji Coba Pada Skenario Pencahayaan

		Nilai k pada *KNN*					
		1		3		5	
		Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi
Jumlah Lokal-Histogram	100	00:05:39	83%	00:05:39	84%	00:05:39	85%
	36	00:02:41	81%	00:02:42	81%	00:02:41	83%
	25	00:02:12	72%	00:02:13	75%	00:02:12	76%
	9	00:01:29	50%	00:01:30	53%	00:01:30	57%
	1	00:01:14	13%	00:01:14	14%	00:01:14	12%

Tabel 5.7 Rangkuman Uji Coba Pada Skenario Ekspresi Wajah

		Nilai k pada *KNN*					
		1		3		5	
		Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi	Waktu Eksekusi	Akurasi
Jumlah Lokal-Histogram	100	00:00:38	82%	00:00:38	83%	00:00:38	83%
	36	00:00:18	81%	00:00:18	80%	00:00:19	81%
	25	00:00:15	80%	00:00:15	81%	00:00:15	79%
	9	00:00:10	58%	00:00:10	62%	00:00:10	65%
	1	00:00:08	13%	00:00:08	13%	00:00:08	15%

5.3.5 Evaluasi dengan Hasil Rujukan Utama

Perbedaan antara hasil pengujian tugas akhir ini terdapat pada reduksi dimensi. Pada jurnal rujukan, reduksi dimensi dilakukan dengan melakukan operasi bit XOR pada blok matriks *LDP*[8]. Pada tugas akhir ini, reduksi dimensi dilakukan dengan mengambil nilai *LDP* yang memiliki nilai maksimal dari dalam

blok matriks *LDP*. Perbedaan hasil klasifikasi antara jurnal rujukan dan tugas akhir ini dapat dilihat pada Tabel 5.8.

Tabel 5.8 Perbandingan Hasil Antara Jurnal Rujukan dan Hasil Implementasi *DRLDP*

Dataset	Jurnal Rujukan	Tugas Akhir
YALE-B	95%	67%-81%
ORL	97%	80%

Selisih tingkat akurasi dapat disebabkan oleh beberapa hal. Selain dikarenakan perbedaan metode reduksi dimensi, data hasil pra-proses juga berperan penting. Data hasil pra-proses pada tugas akhir ini masih memiliki kekurangan dalam menghilangkan *noise*. Sedangkan pra-proses yang dilakukan dalam jurnal rujukan tersebut memiliki hasil yang lebih baik dalam menghilangkan *noise*. Perbedaan hasil pra-proses antara jurnal rujukan dan tugas akhir ini dapat dilihat pada Gambar 5.1 dan Gambar 5.2. Hasil pra-proses pada tugas akhir ini masih terdapat bagian gambar latar yang belum terbuang.

Gambar 5.1 Sampel data subjek dari pra-proses tugas akhir

Gambar 5.2 Sampel Data Subjek Dari Jurnal Rujukan

[*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

Dari hasil uji coba yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Algoritma *Dimensionality Reduced Local Directional Pattern* dapat digunakan untuk menyusun deskriptor gambar pada wajah manusia. Algoritma ini umunya digunakan untuk mengenali pola bentuk pada gambar. Pada pengujian yang telah dilakukan dengan *dataset* ORL Face, akurasi yang dihasilkan mencapai 80% terhadap variasi ekspresi wajah. Pengujian pada Extended Yale-B memberikan hasil 67% terhadap variasi kemiringan kepala dan 81% terhadap perbedaan sudut pencahayaan. Hal ini membuktikan bahwa algoritma ini mampu mengenali wajah dengan baik meskipun terdapat perubahan ekspresi, pencahayaan dan kemiringan kepala subjek.
2. Penggunaan reduksi dimensi mengurangi akurasi paling banyak pada skenario pencahayaan dengan penurunan 12% dan paling sedikit pada skenario kemiringan kepala dengan penurunan 5%. Sedangkan waktu eksekusi yang diperlukan dalam klasifikasi berkurang 79% hingga 82%.
3. Penggunaan lokal histogram berperan penting dalam klasifikasi data. Hal ini yang membedakan antara pengenalan wajah dengan pengenalan pola pada umunya. Ukuran fragmen lokal histogram juga berpengaruh terhadap akurasi.

6.2 Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah sebagai berikut:

1. Pra-proses berperan cukup penting. Dari apa yang telah dilakukan. Pra-proses pemotongan bagian wajah dari *dataset* yang otomatis dengan bantuan algoritma Viola-Jones dan *Haar-Cascade* dari Open CV belum menghasilkan pemotongan yang sempurna. Hal ini dibuktikan dengan terdapatnya beberapa fitur subjek yang tidak diperlukan dan sebaliknya beberapa fitur yang diperlukan justru terpotong sebagian. Fitur yang tidak diperlukan tersebut adalah: rambut, leher dan beberapa bagian latar. Sedangkan, bagian yang penting yang dimaksud adalah dagu subjek yang terpotong pada beberapa hasil pra-proses.
2. Peningkatan spesifikasi yang perlu ditingkatkan. Spesifikasi sistem saat ini memberikan hasil yang sangat lama untuk berkas gambar dengan ukuran lebih dari 300 piksel. Sehingga semua data yang digunakan untuk pengujian dikompres menjadi amat kecil. Hal ini bisa berpengaruh terhadap akurasi yang disebabkan adanya informasi yang hilang dari kompresi ukuran gambar tersebut.
3. Pengujian lain masih bisa dilakukan pada *dataset* Extended Yale-B yang melibatkan kombinasi perbedaan pencahayaan dan kemiringan kepala. Namun, dengan spesifikasi perangkat keras yang lebih baik. Hal ini disebabkan karena terbatasnya kapasitas memori yang dapat digunakan jika dibandingkan dengan jumlah data yang terdapat pada *dataset* ini.
4. Modifikasi arsitektur sistem perlu dilakukan. Arsitektur sistem dimodifikasi sehingga *database* dan perangkat lunak terpisah pada dua perangkat keras yang berbeda. Hal ini berguna untuk mengurangi beban perangkat karena aplikasi dan *database* yang terletak pada sistem yang sama.

Walau waktu yang dibutuhkan untuk mengambil data akan bertambah, hal ini akan mengurangi beban CPU dan RAM perangkat karena harus menjalankan 2 perangkat lunak sekaligus.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] "The Database of Faces." [Daring]. Tersedia pada: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. [Diakses: 19-Jul-2017].

[2] "Yale Face Database B | Computer Vision Online." [Daring]. Tersedia pada: http://www.computervisiononline.com/dataset/1105138686. [Diakses: 19-Jul-2017].

[3] Gary Bradski dan Adrian Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2016.

[4] "opencv: Open Source Computer Vision Library." [Daring]. Tersedia pada: https://github.com/opencv/opencv. [Diakses: 17-Jul-2017].

[5] Rafael C. Gonzalez dan Richard E. Woods, *Digital Image Processing*, Third edition. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

[6] T. Jabid, M. H. Kabir, dan O. Chae, "Local Directional Pattern (LDP) – A Robust Image Descriptor for Object Recognition," 2010, hal. 482–487.

[7] D.-J. Kim, S.-H. Lee, dan M.-K. Sohn, "Face recognition via local directional pattern," *Int. J. Secur. Its Appl.*, vol. 7, no. 2, hal. 191–200, 2013.

[8] Srinivasa Perumal R. dan Chandra Mouli P.V.S.S.R., "Dimensionality reduced local directional pattern (DR-LDP) for face recognition," *Expert Syst. Appl.*, vol. 63, hal. 66–73, Nov 2016.

[9] G. TOUSSAINT, "GEOMETRIC PROXIMITY GRAPHS FOR IMPROVING NEAREST NEIGHBOR METHODS IN INSTANCE-BASED LEARNING AND DATA MINING," *Int. J. Comput. Geom. Amp Appl.*, vol. 15, no. 2, hal. 101–150, 2005.

[*Halaman ini sengaja dikosongkan*]

A. LAMPIRAN

Tabel A.1 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Pertama)

Nama Berkas	*Ground Truth Label*	*Predicted label*
10_s1	s1	s25
10_s11	s11	s11
10_s13	s13	s28
10_s15	s15	s15
10_s14	s14	s14
10_s18	s18	s18
10_s2	s2	s2
10_s19	s19	s19
10_s22	s22	s22
10_s24	s24	s24
10_s25	s25	s25
10_s26	s26	s13
10_s27	s27	s27
10_s29	s29	s29
10_s3	s3	s3
10_s30	s30	s30
10_s31	s31	s31
10_s33	s33	s27
10_s34	s34	s34
10_s35	s35	s35
10_s37	s37	s37
10_s38	s38	s38
10_s39	s39	s39
10_s6	s6	s6
10_s7	s7	s7

Tabel A.2 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Kedua)

Nama Berkas	*Ground Truth Label*	*Predicted label*
10_s9	s9	s4
10_s8	s8	s8
1_s10	s10	s10
1_s11	s11	s11
1_s12	s12	s12
1_s13	s13	s4
1_s14	s14	s14
1_s15	s15	s15
1_s17	s17	s17
1_s18	s18	s18
1_s19	s19	s19
1_s2	s2	s2
1_s20	s20	s20
1_s22	s22	s22
1_s23	s23	s23
1_s24	s24	s24
1_s26	s26	s26
1_s27	s27	s27
1_s25	s25	s25
1_s3	s3	s3
1_s29	s29	s29
1_s30	s30	s30
1_s31	s31	s31
1_s32	s32	s32
1_s34	s34	s34

Tabel A.3 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Ketiga)

Nama Berkas	*Ground Truth Label*	*Predicted label*
1_s38	s38	s38
1_s36	s36	s20
1_s37	s37	s37
1_s4	s4	s4
1_s5	s5	s5
1_s39	s39	s39
1_s40	s40	s40
1_s7	s7	s7
1_s6	s6	s6
1_s8	s8	s19
2_s10	s10	s10
2_s11	s11	s11
2_s12	s12	s12
2_s13	s13	s13
2_s14	s14	s14
2_s15	s15	s15
2_s17	s17	s17
2_s2	s2	s2
2_s20	s20	s20
2_s22	s22	s22
2_s23	s23	s23
2_s24	s24	s24
2_s25	s25	s25
2_s27	s27	s27
2_s28	s28	s28

Tabel A.4 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Keempat)

Nama Berkas	*Ground Truth Label*	*Predicted label*
2_s29	s29	s29
2_s31	s31	s31
2_s33	s33	s19
2_s32	s32	s32
2_s34	s34	s34
2_s36	s36	s36
2_s37	s37	s37
2_s35	s35	s35
2_s38	s38	s17
2_s4	s4	s4
2_s5	s5	s5
2_s6	s6	s6
2_s7	s7	s7
2_s8	s8	s19
3_s10	s10	s10
3_s11	s11	s11
3_s12	s12	s12
3_s14	s14	s14
3_s15	s15	s15
3_s17	s17	s17
3_s18	s18	s18
3_s2	s2	s2
3_s23	s23	s23
3_s22	s22	s22
3_s24	s24	s15

Tabel A.5 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Kelima)

Nama Berkas	*Ground Truth Label*	*Predicted label*
3_s25	s25	s25
3_s27	s27	s27
3_s29	s29	s29
3_s30	s30	s30
3_s31	s31	s31
3_s32	s32	s32
3_s33	s33	s33
3_s34	s34	s34
3_s37	s37	s28
3_s35	s35	s19
3_s38	s38	s38
3_s39	s39	s6
3_s5	s5	s5
3_s4	s4	s4
3_s8	s8	s8
3_s9	s9	s9
3_s7	s7	s7
4_s10	s10	s25
4_s12	s12	s12
4_s16	s16	s38
4_s20	s20	s20
4_s21	s21	s30
4_s23	s23	s23
4_s28	s28	s28
4_s32	s32	s32

Tabel A.6 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Keenam)

Nama Berkas	*Ground Truth Label*	*Predicted label*
4_s36	s36	s36
4_s4	s4	s4
4_s40	s40	s40
4_s5	s5	s5
4_s1	s1	s28
5_s16	s16	s28
5_s21	s21	s4
5_s28	s28	s28
4_s33	s33	s33
4_s35	s35	s35
4_s9	s9	s9
5_s1	s1	s20
6_s16	s16	s14
4_s18	s18	s18
4_s19	s19	s19
6_s21	s21	s21
4_s26	s26	s14
4_s3	s3	s10
4_s30	s30	s30
4_s39	s39	s19
5_s40	s40	s40
5_s9	s9	s9
7_s1	s1	s32
4_s13	s13	s3
7_s16	s16	s20

Tabel A.7 Hasil Uji Coba pada Skenario Perbedaan Ekspresi Wajah (Bagian Ketujuh)

Nama Berkas	*Ground Truth Label*	*Predicted label*
5_s19	s19	s19
5_s20	s20	s20
7_s21	s21	s29
5_s26	s26	s26
6_s28	s28	s28
5_s3	s3	s40
5_s36	s36	s11
6_s40	s40	s40
4_s6	s6	s6
Akurasi	80%	
Waktu Eksekusi	00:00:08	

Tabel A.8 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Pertama)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB11_P00A+025E+00	yaleB11	yaleB11
yaleB12_P00A+025E+00	yaleB12	yaleB12
yaleB13_P00A+025E+00	yaleB13	yaleB13
yaleB15_P00A+025E+00	yaleB15	yaleB15
yaleB18_P00A+025E+00	yaleB18	yaleB18
yaleB19_P00A+025E+00	yaleB19	yaleB18
yaleB20_P00A+025E+00	yaleB20	yaleB20
yaleB21_P00A+025E+00	yaleB21	yaleB21
yaleB22_P00A+025E+00	yaleB22	yaleB22
yaleB23_P00A+025E+00	yaleB23	yaleB23
yaleB24_P00A+025E+00	yaleB24	yaleB24
yaleB25_P00A+025E+00	yaleB25	yaleB25
yaleB26_P00A+025E+00	yaleB26	yaleB26
yaleB27_P00A+025E+00	yaleB27	yaleB27
yaleB28_P00A+025E+00	yaleB28	yaleB28
yaleB29_P00A+025E+00	yaleB29	yaleB26
yaleB30_P00A+025E+00	yaleB30	yaleB30
yaleB31_P00A+025E+00	yaleB31	yaleB31
yaleB32_P00A+025E+00	yaleB32	yaleB32
yaleB33_P00A+025E+00	yaleB33	yaleB33
yaleB34_P00A+025E+00	yaleB34	yaleB34
yaleB35_P00A+025E+00	yaleB35	yaleB35
yaleB36_P00A+025E+00	yaleB36	yaleB36
yaleB37_P00A+025E+00	yaleB37	yaleB37

Tabel A.9 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kedua)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB38_P00A+025E+00	yaleB38	yaleB38
yaleB39_P00A+025E+00	yaleB39	yaleB39
yaleB11_P00A-035E+15	yaleB11	yaleB11
yaleB12_P00A-035E+15	yaleB12	yaleB12
yaleB13_P00A-035E+15	yaleB13	yaleB13
yaleB15_P00A-035E+15	yaleB15	yaleB15
yaleB17_P00A-035E+15	yaleB17	yaleB17
yaleB20_P00A-035E+15	yaleB20	yaleB20
yaleB21_P00A-035E+15	yaleB21	yaleB21
yaleB22_P00A-035E+15	yaleB22	yaleB22
yaleB23_P00A-035E+15	yaleB23	yaleB23
yaleB24_P00A-035E+15	yaleB24	yaleB24
yaleB25_P00A-035E+15	yaleB25	yaleB25
yaleB27_P00A-035E+15	yaleB27	yaleB27
yaleB28_P00A-035E+15	yaleB28	yaleB28
yaleB29_P00A-035E+15	yaleB29	yaleB29
yaleB30_P00A-035E+15	yaleB30	yaleB30
yaleB31_P00A-035E+15	yaleB31	yaleB31
yaleB33_P00A-035E+15	yaleB33	yaleB33
yaleB34_P00A-035E+15	yaleB34	yaleB34
yaleB35_P00A-035E+15	yaleB35	yaleB35
yaleB36_P00A-035E+15	yaleB36	yaleB36
yaleB37_P00A-035E+15	yaleB37	yaleB37

Tabel A.10 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Ketiga)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB38_P00A-035E+15	yaleB38	yaleB38
yaleB39_P00A-035E+15	yaleB39	yaleB39
yaleB11_P00A+035E-20	yaleB11	yaleB11
yaleB13_P00A+035E-20	yaleB13	yaleB13
yaleB15_P00A+035E-20	yaleB15	yaleB15
yaleB17_P00A+035E-20	yaleB17	yaleB17
yaleB18_P00A+035E-20	yaleB18	yaleB15
yaleB19_P00A+035E-20	yaleB19	yaleB25
yaleB20_P00A+035E-20	yaleB20	yaleB20
yaleB21_P00A+035E-20	yaleB21	yaleB21
yaleB22_P00A+035E-20	yaleB22	yaleB22
yaleB23_P00A+035E-20	yaleB23	yaleB23
yaleB24_P00A+035E-20	yaleB24	yaleB24
yaleB25_P00A+035E-20	yaleB25	yaleB25
yaleB27_P00A+035E-20	yaleB27	yaleB27
yaleB28_P00A+035E-20	yaleB28	yaleB28
yaleB30_P00A+035E-20	yaleB30	yaleB30
yaleB31_P00A+035E-20	yaleB31	yaleB31
yaleB32_P00A+035E-20	yaleB32	yaleB32
yaleB34_P00A+035E-20	yaleB34	yaleB34
yaleB35_P00A+035E-20	yaleB35	yaleB35
yaleB36_P00A+035E-20	yaleB36	yaleB36
yaleB37_P00A+035E-20	yaleB37	yaleB37

Tabel A.11 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Keempat)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB38_P00A+035E-20	yaleB38	yaleB38
yaleB39_P00A+035E-20	yaleB39	yaleB39
yaleB11_P00A+050E+00	yaleB11	yaleB11
yaleB12_P00A+050E+00	yaleB12	yaleB12
yaleB13_P00A+050E+00	yaleB13	yaleB13
yaleB15_P00A+050E+00	yaleB15	yaleB15
yaleB17_P00A+050E+00	yaleB17	yaleB17
yaleB20_P00A+050E+00	yaleB20	yaleB20
yaleB22_P00A+050E+00	yaleB22	yaleB22
yaleB24_P00A+050E+00	yaleB24	yaleB24
yaleB25_P00A+050E+00	yaleB25	yaleB29
yaleB26_P00A+050E+00	yaleB26	yaleB26
yaleB27_P00A+050E+00	yaleB27	yaleB27
yaleB28_P00A+050E+00	yaleB28	yaleB28
yaleB30_P00A+050E+00	yaleB30	yaleB30
yaleB31_P00A+050E+00	yaleB31	yaleB31
yaleB32_P00A+050E+00	yaleB32	yaleB32
yaleB33_P00A+050E+00	yaleB33	yaleB33
yaleB34_P00A+050E+00	yaleB34	yaleB34
yaleB35_P00A+050E+00	yaleB35	yaleB35
yaleB36_P00A+050E+00	yaleB36	yaleB36
yaleB37_P00A+050E+00	yaleB37	yaleB37

Tabel A.12 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kelima)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB11_P00A+060E+20	yaleB11	yaleB11
yaleB12_P00A+060E+20	yaleB12	yaleB12
yaleB13_P00A+060E+20	yaleB13	yaleB13
yaleB15_P00A+060E+20	yaleB15	yaleB15
yaleB17_P00A+060E+20	yaleB17	yaleB17
yaleB19_P00A+060E+20	yaleB19	yaleB29
yaleB20_P00A+060E+20	yaleB20	yaleB20
yaleB22_P00A+060E+20	yaleB22	yaleB22
yaleB23_P00A+060E+20	yaleB23	yaleB23
yaleB24_P00A+060E+20	yaleB24	yaleB24
yaleB25_P00A+060E+20	yaleB25	yaleB25
yaleB26_P00A+060E+20	yaleB26	yaleB26
yaleB27_P00A+060E+20	yaleB27	yaleB27
yaleB28_P00A+060E+20	yaleB28	yaleB28
yaleB29_P00A+060E+20	yaleB29	yaleB29
yaleB30_P00A+060E+20	yaleB30	yaleB30
yaleB31_P00A+060E+20	yaleB31	yaleB23
yaleB32_P00A+060E+20	yaleB32	yaleB32
yaleB34_P00A+060E+20	yaleB34	yaleB34
yaleB35_P00A+060E+20	yaleB35	yaleB35
yaleB37_P00A+060E+20	yaleB37	yaleB37
yaleB38_P00A+060E+20	yaleB38	yaleB38
yaleB11_P00A-060E+20	yaleB11	yaleB11

Tabel A.13 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Keenam)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB12_P00A-060E+20	yaleB12	yaleB12
yaleB13_P00A-060E+20	yaleB13	yaleB13
yaleB15_P00A-060E+20	yaleB15	yaleB15
yaleB17_P00A-060E+20	yaleB17	yaleB17
yaleB19_P00A-060E+20	yaleB19	yaleB19
yaleB20_P00A-060E+20	yaleB20	yaleB20
yaleB22_P00A-060E+20	yaleB22	yaleB22
yaleB23_P00A-060E+20	yaleB23	yaleB23
yaleB24_P00A-060E+20	yaleB24	yaleB24
yaleB25_P00A-060E+20	yaleB25	yaleB25
yaleB27_P00A-060E+20	yaleB27	yaleB27
yaleB29_P00A-060E+20	yaleB29	yaleB29
yaleB31_P00A-060E+20	yaleB31	yaleB31
yaleB34_P00A-060E+20	yaleB34	yaleB34
yaleB35_P00A-060E+20	yaleB35	yaleB35
yaleB36_P00A-060E+20	yaleB36	yaleB36
yaleB37_P00A-060E+20	yaleB37	yaleB17
yaleB38_P00A-060E+20	yaleB38	yaleB27
yaleB12_P00A-070E+00	yaleB12	yaleB12
yaleB13_P00A-070E+00	yaleB13	yaleB13
yaleB15_P00A-070E+00	yaleB15	yaleB15
yaleB18_P00A-070E+00	yaleB18	yaleB18

Tabel A.14 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Ketujuh)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB19_P00A-070E+00	yaleB19	yaleB19
yaleB20_P00A-070E+00	yaleB20	yaleB20
yaleB21_P00A-070E+00	yaleB21	yaleB21
yaleB22_P00A-070E+00	yaleB22	yaleB22
yaleB23_P00A-070E+00	yaleB23	yaleB31
yaleB24_P00A-070E+00	yaleB24	yaleB24
yaleB25_P00A-070E+00	yaleB25	yaleB26
yaleB29_P00A-070E+00	yaleB29	yaleB29
yaleB34_P00A-070E+00	yaleB34	yaleB28
yaleB35_P00A-070E+00	yaleB35	yaleB35
yaleB36_P00A-070E+00	yaleB36	yaleB36
yaleB37_P00A-070E+00	yaleB37	yaleB37
yaleB38_P00A-070E+00	yaleB38	yaleB38
yaleB39_P00A-070E+00	yaleB39	yaleB39
yaleB12_P00A-085E-20	yaleB12	yaleB12
yaleB17_P00A-085E-20	yaleB17	yaleB17
yaleB20_P00A-085E-20	yaleB20	yaleB20
yaleB21_P00A-085E-20	yaleB21	yaleB21
yaleB22_P00A-085E-20	yaleB22	yaleB22
yaleB24_P00A-085E-20	yaleB24	yaleB31
yaleB25_P00A-085E-20	yaleB25	yaleB25
yaleB28_P00A-085E-20	yaleB28	yaleB28
yaleB30_P00A-085E-20	yaleB30	yaleB30

Tabel A.15 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kedelapan)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB32_P00A-085E-20	yaleB32	yaleB32
yaleB33_P00A-085E-20	yaleB33	yaleB31
yaleB34_P00A-085E-20	yaleB34	yaleB34
yaleB35_P00A-085E-20	yaleB35	yaleB35
yaleB36_P00A-085E-20	yaleB36	yaleB31
yaleB38_P00A-085E-20	yaleB38	yaleB38
yaleB39_P00A-085E-20	yaleB39	yaleB39
yaleB11_P00A+085E-20	yaleB11	yaleB11
yaleB13_P00A+085E-20	yaleB13	yaleB13
yaleB15_P00A+085E-20	yaleB15	yaleB15
yaleB19_P00A+085E-20	yaleB19	yaleB36
yaleB20_P00A+085E-20	yaleB20	yaleB20
yaleB21_P00A+085E-20	yaleB21	yaleB21
yaleB24_P00A+085E-20	yaleB24	yaleB24
yaleB25_P00A+085E-20	yaleB25	yaleB20
yaleB27_P00A+085E-20	yaleB27	yaleB27
yaleB28_P00A+085E-20	yaleB28	yaleB28
yaleB31_P00A+085E-20	yaleB31	yaleB31
yaleB32_P00A+085E-20	yaleB32	yaleB32
yaleB33_P00A+085E-20	yaleB33	yaleB33
yaleB34_P00A+085E-20	yaleB34	yaleB34
yaleB35_P00A+085E-20	yaleB35	yaleB35
yaleB36_P00A+085E-20	yaleB36	yaleB36

Tabel A.16 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kesembilan)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB37_P00A+085E-20	yaleB37	yaleB37
yaleB38_P00A+085E-20	yaleB38	yaleB38
yaleB39_P00A+085E-20	yaleB39	yaleB39
yaleB11_P00A+095E+00	yaleB11	yaleB39
yaleB12_P00A+095E+00	yaleB12	yaleB38
yaleB13_P00A+095E+00	yaleB13	yaleB13
yaleB18_P00A+095E+00	yaleB18	yaleB18
yaleB20_P00A+095E+00	yaleB20	yaleB20
yaleB21_P00A+095E+00	yaleB21	yaleB21
yaleB22_P00A+095E+00	yaleB22	yaleB22
yaleB24_P00A+095E+00	yaleB24	yaleB24
yaleB25_P00A+095E+00	yaleB25	yaleB25
yaleB29_P00A+095E+00	yaleB29	yaleB29
yaleB31_P00A+095E+00	yaleB31	yaleB31
yaleB32_P00A+095E+00	yaleB32	yaleB32
yaleB12_P00A+110E+15	yaleB12	yaleB12
yaleB13_P00A+110E+15	yaleB13	yaleB13
yaleB15_P00A+110E+15	yaleB15	yaleB15
yaleB20_P00A+110E+15	yaleB20	yaleB20
yaleB21_P00A+110E+15	yaleB21	yaleB21
yaleB22_P00A+110E+15	yaleB22	yaleB22
yaleB23_P00A+110E+15	yaleB23	yaleB23
yaleB24_P00A+110E+15	yaleB24	yaleB24

Tabel A.17 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kesepuluh)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB26_P00A+110E+15	yaleB26	yaleB26
yaleB28_P00A+110E+15	yaleB28	yaleB28
yaleB29_P00A+110E+15	yaleB29	yaleB29
yaleB32_P00A+110E+15	yaleB32	yaleB32
yaleB34_P00A+110E+15	yaleB34	yaleB34
yaleB35_P00A+110E+15	yaleB35	yaleB35
yaleB36_P00A+110E+15	yaleB36	yaleB39
yaleB37_P00A+110E+15	yaleB37	yaleB37
yaleB38_P00A+110E+15	yaleB38	yaleB38
yaleB12_P00A-110E-20	yaleB12	yaleB34
yaleB13_P00A-110E-20	yaleB13	yaleB13
yaleB19_P00A-110E-20	yaleB19	yaleB31
yaleB24_P00A-110E-20	yaleB24	yaleB15
yaleB25_P00A-110E-20	yaleB25	yaleB23
yaleB27_P00A-110E-20	yaleB27	yaleB30
yaleB28_P00A-110E-20	yaleB28	yaleB28
yaleB32_P00A-110E-20	yaleB32	yaleB32
yaleB35_P00A-110E-20	yaleB35	yaleB35
yaleB36_P00A-110E-20	yaleB36	yaleB31
yaleB37_P00A-110E-20	yaleB37	yaleB37
yaleB38_P00A-110E-20	yaleB38	yaleB38
yaleB12_P00A-120E+00	yaleB12	yaleB33
yaleB13_P00A-120E+00	yaleB13	yaleB20

Tabel A.18 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kesebelas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB18_P00A-120E+00	yaleB18	yaleB17
yaleB19_P00A-120E+00	yaleB19	yaleB19
yaleB24_P00A-120E+00	yaleB24	yaleB24
yaleB27_P00A-120E+00	yaleB27	yaleB34
yaleB28_P00A-120E+00	yaleB28	yaleB34
yaleB32_P00A-120E+00	yaleB32	yaleB32
yaleB35_P00A-120E+00	yaleB35	yaleB35
yaleB36_P00A-120E+00	yaleB36	yaleB30
yaleB37_P00A-120E+00	yaleB37	yaleB34
yaleB38_P00A-120E+00	yaleB38	yaleB34
yaleB12_P00A+120E+00	yaleB12	yaleB23
yaleB13_P00A+120E+00	yaleB13	yaleB22
yaleB17_P00A+120E+00	yaleB17	yaleB25
yaleB18_P00A+120E+00	yaleB18	yaleB24
yaleB19_P00A+120E+00	yaleB19	yaleB19
yaleB27_P00A+120E+00	yaleB27	yaleB27
yaleB28_P00A+120E+00	yaleB28	yaleB28
yaleB30_P00A+120E+00	yaleB30	yaleB30
yaleB32_P00A+120E+00	yaleB32	yaleB34
yaleB33_P00A+120E+00	yaleB33	yaleB24
yaleB35_P00A+120E+00	yaleB35	yaleB35
yaleB36_P00A+120E+00	yaleB36	yaleB31
yaleB37_P00A+120E+00	yaleB37	yaleB37

Tabel A.19 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kedua Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB38_P00A+120E+00	yaleB38	yaleB38
yaleB39_P00A+120E+00	yaleB39	yaleB30
yaleB12_P00A-130E+20	yaleB12	yaleB25
yaleB13_P00A-130E+20	yaleB13	yaleB22
yaleB27_P00A-130E+20	yaleB27	yaleB33
yaleB28_P00A-130E+20	yaleB28	yaleB34
yaleB32_P00A-130E+20	yaleB32	yaleB34
yaleB36_P00A-130E+20	yaleB36	yaleB33
yaleB37_P00A-130E+20	yaleB37	yaleB34
yaleB11_P00A+130E+20	yaleB11	yaleB39
yaleB12_P00A+130E+20	yaleB12	yaleB19
yaleB13_P00A+130E+20	yaleB13	yaleB23
yaleB24_P00A+130E+20	yaleB24	yaleB24
yaleB26_P00A+130E+20	yaleB26	yaleB39
yaleB27_P00A+130E+20	yaleB27	yaleB23
yaleB28_P00A+130E+20	yaleB28	yaleB28
yaleB32_P00A+130E+20	yaleB32	yaleB34
yaleB33_P00A+130E+20	yaleB33	yaleB25
yaleB34_P00A+130E+20	yaleB34	yaleB34
yaleB35_P00A+130E+20	yaleB35	yaleB20
yaleB36_P00A+130E+20	yaleB36	yaleB23
yaleB17_P00A+000E+00	yaleB17	yaleB17
yaleB18_P00A+000E+00	yaleB18	yaleB36

Tabel A.20 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Ketiga Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB19_P00A+000E+00	yaleB19	yaleB36
yaleB26_P00A+000E+00	yaleB26	yaleB26
yaleB32_P00A+000E+00	yaleB32	yaleB32
yaleB12_P00A+000E+00	yaleB12	yaleB12
yaleB26_P00A+000E+20	yaleB26	yaleB26
yaleB29_P00A+000E+00	yaleB29	yaleB19
yaleB33_P00A+000E+00	yaleB33	yaleB33
yaleB18_P00A+000E+20	yaleB18	yaleB18
yaleB19_P00A+000E+20	yaleB19	yaleB19
yaleB21_P00A+000E+00	yaleB21	yaleB21
yaleB23_P00A+000E+00	yaleB23	yaleB23
yaleB29_P00A+000E+20	yaleB29	yaleB29
yaleB38_P00A+000E+00	yaleB38	yaleB38
yaleB18_P00A+000E-20	yaleB18	yaleB36
yaleB21_P00A+000E+20	yaleB21	yaleB21
yaleB33_P00A+000E+20	yaleB33	yaleB33
yaleB36_P00A+000E+00	yaleB36	yaleB36
yaleB39_P00A+000E+00	yaleB39	yaleB36
yaleB18_P00A-005E-10	yaleB18	yaleB15
yaleB21_P00A+000E-20	yaleB21	yaleB21
yaleB26_P00A+000E-20	yaleB26	yaleB26
yaleB28_P00A+000E+00	yaleB28	yaleB28
yaleB30_P00A+000E+00	yaleB30	yaleB30

Tabel A.21 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Keempat Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB32_P00A+000E+20	yaleB32	yaleB32
yaleB33_P00A+000E-20	yaleB33	yaleB31
yaleB39_P00A+000E+20	yaleB39	yaleB39
yaleB11_P00A+000E+00	yaleB11	yaleB11
yaleB17_P00A+000E+20	yaleB17	yaleB17
yaleB26_P00A-005E-10	yaleB26	yaleB26
yaleB27_P00A+000E+00	yaleB27	yaleB27
yaleB28_P00A+000E+20	yaleB28	yaleB28
yaleB30_P00A+000E+20	yaleB30	yaleB30
yaleB31_P00A+000E+00	yaleB31	yaleB31
yaleB32_P00A+000E-20	yaleB32	yaleB32
yaleB33_P00A-005E-10	yaleB33	yaleB33
yaleB11_P00A+000E+20	yaleB11	yaleB11
yaleB13_P00A+000E+00	yaleB13	yaleB13
yaleB15_P00A+000E+00	yaleB15	yaleB15
yaleB18_P00A-005E+10	yaleB18	yaleB18
yaleB19_P00A+000E-20	yaleB19	yaleB19
yaleB23_P00A+000E+20	yaleB23	yaleB23
yaleB26_P00A-005E+10	yaleB26	yaleB26
yaleB27_P00A+000E+20	yaleB27	yaleB27
yaleB29_P00A+000E-20	yaleB29	yaleB29
yaleB31_P00A+000E+20	yaleB31	yaleB31
yaleB37_P00A+000E+00	yaleB37	yaleB37

Tabel A.22 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kelima Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB12_P00A+000E+20	yaleB12	yaleB12
yaleB17_P00A+000E-20	yaleB17	yaleB17
yaleB18_P00A+005E+10	yaleB18	yaleB36
yaleB22_P00A+000E+00	yaleB22	yaleB22
yaleB23_P00A+000E-20	yaleB23	yaleB23
yaleB26_P00A+005E+10	yaleB26	yaleB26
yaleB29_P00A-005E-10	yaleB29	yaleB29
yaleB30_P00A+000E-20	yaleB30	yaleB30
yaleB15_P00A+000E+20	yaleB15	yaleB15
yaleB17_P00A-005E-10	yaleB17	yaleB17
yaleB19_P00A-005E-10	yaleB19	yaleB19
yaleB23_P00A-005E-10	yaleB23	yaleB23
yaleB26_P00A+005E-10	yaleB26	yaleB36
yaleB27_P00A+000E-20	yaleB27	yaleB27
yaleB28_P00A+000E-20	yaleB28	yaleB28
yaleB30_P00A-005E-10	yaleB30	yaleB30
yaleB33_P00A-005E+10	yaleB33	yaleB33
yaleB34_P00A+000E+00	yaleB34	yaleB34
yaleB36_P00A+000E+20	yaleB36	yaleB36
yaleB38_P00A+000E+20	yaleB38	yaleB38
yaleB39_P00A+000E-20	yaleB39	yaleB39
yaleB11_P00A+000E-20	yaleB11	yaleB11
yaleB17_P00A-005E+10	yaleB17	yaleB17

Tabel A.23 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Keenam Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB18_P00A+005E-10	yaleB18	yaleB19
yaleB19_P00A-005E+10	yaleB19	yaleB19
yaleB25_P00A+000E+00	yaleB25	yaleB25
yaleB27_P00A-005E-10	yaleB27	yaleB27
yaleB30_P00A-005E+10	yaleB30	yaleB30
yaleB31_P00A+000E-20	yaleB31	yaleB31
yaleB33_P00A+005E+10	yaleB33	yaleB33
yaleB39_P00A-005E-10	yaleB39	yaleB39
yaleB11_P00A-005E-10	yaleB11	yaleB11
yaleB15_P00A+000E-20	yaleB15	yaleB15
yaleB17_P00A+005E+10	yaleB17	yaleB17
yaleB18_P00A+010E+00	yaleB18	yaleB36
yaleB20_P00A+000E+00	yaleB20	yaleB20
yaleB21_P00A-005E-10	yaleB21	yaleB21
yaleB22_P00A+000E+20	yaleB22	yaleB22
yaleB23_P00A-005E+10	yaleB23	yaleB23
yaleB26_P00A-010E+00	yaleB26	yaleB26
yaleB29_P00A-005E+10	yaleB29	yaleB21
yaleB30_P00A+005E+10	yaleB30	yaleB30
yaleB31_P00A-005E-10	yaleB31	yaleB31
yaleB33_P00A+005E-10	yaleB33	yaleB33
yaleB34_P00A+000E+20	yaleB34	yaleB34
yaleB39_P00A-005E+10	yaleB39	yaleB39

Tabel A.24 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Ketujuh Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB11_P00A-005E+10	yaleB11	yaleB11
yaleB15_P00A-005E-10	yaleB15	yaleB15
yaleB17_P00A+005E-10	yaleB17	yaleB36
yaleB20_P00A+000E+20	yaleB20	yaleB20
yaleB21_P00A-005E+10	yaleB21	yaleB21
yaleB22_P00A+000E-20	yaleB22	yaleB22
yaleB23_P00A+005E+10	yaleB23	yaleB23
yaleB25_P00A+000E+20	yaleB25	yaleB25
yaleB26_P00A+010E+00	yaleB26	yaleB36
yaleB29_P00A+005E+10	yaleB29	yaleB36
yaleB30_P00A+005E-10	yaleB30	yaleB30
yaleB31_P00A-005E+10	yaleB31	yaleB31
yaleB33_P00A-010E+00	yaleB33	yaleB33
yaleB34_P00A+000E-20	yaleB34	yaleB34
yaleB39_P00A+005E+10	yaleB39	yaleB39
yaleB11_P00A+005E+10	yaleB11	yaleB11
yaleB15_P00A-005E+10	yaleB15	yaleB15
yaleB20_P00A+000E-20	yaleB20	yaleB20
yaleB21_P00A+005E+10	yaleB21	yaleB21
yaleB22_P00A-005E-10	yaleB22	yaleB22
yaleB23_P00A+005E-10	yaleB23	yaleB23
yaleB24_P00A+000E+00	yaleB24	yaleB24
yaleB25_P00A+000E-20	yaleB25	yaleB25

Tabel A.25 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kedelapan Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB26_P00A+010E-20	yaleB26	yaleB11
yaleB29_P00A+005E-10	yaleB29	yaleB29
yaleB31_P00A+005E+10	yaleB31	yaleB31
yaleB34_P00A-005E-10	yaleB34	yaleB34
yaleB11_P00A+005E-10	yaleB11	yaleB11
yaleB15_P00A+005E+10	yaleB15	yaleB15
yaleB17_P00A+010E+00	yaleB17	yaleB13
yaleB18_P00A+010E-20	yaleB18	yaleB18
yaleB19_P00A+005E+10	yaleB19	yaleB19
yaleB20_P00A-005E-10	yaleB20	yaleB20
yaleB21_P00A+005E-10	yaleB21	yaleB21
yaleB22_P00A-005E+10	yaleB22	yaleB22
yaleB23_P00A-010E+00	yaleB23	yaleB23
yaleB24_P00A+000E+20	yaleB24	yaleB24
yaleB25_P00A-005E-10	yaleB25	yaleB25
yaleB26_P00A-010E-20	yaleB26	yaleB26
yaleB29_P00A-010E+00	yaleB29	yaleB19
yaleB30_P00A-010E+00	yaleB30	yaleB30
yaleB31_P00A+005E-10	yaleB31	yaleB31
yaleB33_P00A+010E+00	yaleB33	yaleB33
yaleB34_P00A-005E+10	yaleB34	yaleB34
yaleB35_P00A+000E+00	yaleB35	yaleB35
yaleB38_P00A+000E-20	yaleB38	yaleB38

Tabel A.26 Hasil Uji Coba pada Skenario Terhadap Pencahayaan (Bagian Kesembilan Belas)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB39_P00A+005E-10	yaleB39	yaleB39
yaleB15_P00A+005E-10	yaleB15	yaleB15
yaleB17_P00A+010E-20	yaleB17	yaleB20
yaleB18_P00A-010E-20	yaleB18	yaleB15
yaleB19_P00A+005E-10	yaleB19	yaleB19
yaleB20_P00A-005E+10	yaleB20	yaleB20
yaleB21_P00A-010E+00	yaleB21	yaleB21
yaleB22_P00A+005E+10	yaleB22	yaleB22
yaleB23_P00A+010E+00	yaleB23	yaleB23
yaleB25_P00A-005E+10	yaleB25	yaleB25
yaleB29_P00A+010E+00	yaleB29	yaleB21
yaleB30_P00A+010E+00	yaleB30	yaleB30
yaleB31_P00A-010E+00	yaleB31	yaleB31
yaleB37_P00A+000E+20	yaleB37	yaleB37
yaleB38_P00A-005E-10	yaleB38	yaleB27
yaleB39_P00A-010E+00	yaleB39	yaleB39
Akurasi	81%	
Waktu eksekusi	00:02:42	

Tabel A.27 Hasil Uji Coba pada Skenario Perbedaan Kemiringan Kepala Subjek (Bagian Pertama)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB11_P05A+000E+00	yaleB11	yaleB11
yaleB11_P06A+000E+00	yaleB11	yaleB11
yaleB11_P07A+000E+00	yaleB11	yaleB11
yaleB11_P08A+000E+00	yaleB11	yaleB13
yaleB12_P05A+000E+00	yaleB12	yaleB12
yaleB12_P06A+000E+00	yaleB12	yaleB13
yaleB12_P07A+000E+00	yaleB12	yaleB12
yaleB12_P08A+000E+00	yaleB12	yaleB12
yaleB13_P05A+000E+00	yaleB13	yaleB16
yaleB13_P06A+000E+00	yaleB13	yaleB36
yaleB13_P07A+000E+00	yaleB13	yaleB13
yaleB13_P08A+000E+00	yaleB13	yaleB13
yaleB15_P05A+000E+00	yaleB15	yaleB15
yaleB15_P06A+000E+00	yaleB15	yaleB23
yaleB15_P07A+000E+00	yaleB15	yaleB17
yaleB15_P08A+000E+00	yaleB15	yaleB26
yaleB17_P05A+000E+00	yaleB17	yaleB21
yaleB17_P06A+000E+00	yaleB17	yaleB18
yaleB17_P07A+000E+00	yaleB17	yaleB17
yaleB17_P08A+000E+00	yaleB17	yaleB26
yaleB18_P05A+000E+00	yaleB18	yaleB26
yaleB18_P06A+000E+00	yaleB18	yaleB18
yaleB18_P07A+000E+00	yaleB18	yaleB18
yaleB18_P08A+000E+00	yaleB18	yaleB18
yaleB19_P05A+000E+00	yaleB19	yaleB18
yaleB19_P06A+000E+00	yaleB19	yaleB29

Tabel A.28 Hasil Uji Coba pada Skenario Perbedaan Kemiringan Kepala Subjek (Bagian Kedua)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB19_P07A+000E+00	yaleB19	yaleB30
yaleB19_P08A+000E+00	yaleB19	yaleB34
yaleB20_P05A+000E+00	yaleB20	yaleB30
yaleB20_P06A+000E+00	yaleB20	yaleB20
yaleB20_P07A+000E+00	yaleB20	yaleB20
yaleB20_P08A+000E+00	yaleB20	yaleB20
yaleB21_P05A+000E+00	yaleB21	yaleB21
yaleB21_P06A+000E+00	yaleB21	yaleB21
yaleB21_P07A+000E+00	yaleB21	yaleB21
yaleB21_P08A+000E+00	yaleB21	yaleB21
yaleB22_P05A+000E+00	yaleB22	yaleB22
yaleB22_P06A+000E+00	yaleB22	yaleB22
yaleB22_P07A+000E+00	yaleB22	yaleB22
yaleB22_P08A+000E+00	yaleB22	yaleB22
yaleB23_P05A+000E+00	yaleB23	yaleB31
yaleB23_P06A+000E+00	yaleB23	yaleB23
yaleB23_P07A+000E+00	yaleB23	yaleB23
yaleB23_P08A+000E+00	yaleB23	yaleB23
yaleB24_P05A+000E+00	yaleB24	yaleB24
yaleB24_P06A+000E+00	yaleB24	yaleB24
yaleB24_P07A+000E+00	yaleB24	yaleB24
yaleB24_P08A+000E+00	yaleB24	yaleB24
yaleB25_P05A+000E+00	yaleB25	yaleB25
yaleB25_P06A+000E+00	yaleB25	yaleB26
yaleB25_P07A+000E+00	yaleB25	yaleB25

Tabel A.29 Hasil Uji Coba pada Skenario Perbedaan Kemiringan Kepala Subjek (Bagian Ketiga)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB25_P08A+000E+00	yaleB25	yaleB25
yaleB26_P05A+000E+00	yaleB26	yaleB26
yaleB26_P06A+000E+00	yaleB26	yaleB26
yaleB26_P07A+000E+00	yaleB26	yaleB17
yaleB26_P08A+000E+00	yaleB26	yaleB26
yaleB27_P05A+000E+00	yaleB27	yaleB27
yaleB27_P06A+000E+00	yaleB27	yaleB27
yaleB27_P07A+000E+00	yaleB27	yaleB27
yaleB27_P08A+000E+00	yaleB27	yaleB27
yaleB28_P05A+000E+00	yaleB28	yaleB28
yaleB28_P06A+000E+00	yaleB28	yaleB28
yaleB28_P07A+000E+00	yaleB28	yaleB28
yaleB28_P08A+000E+00	yaleB28	yaleB28
yaleB29_P05A+000E+00	yaleB29	yaleB21
yaleB29_P06A+000E+00	yaleB29	yaleB21
yaleB29_P07A+000E+00	yaleB29	yaleB29
yaleB29_P08A+000E+00	yaleB29	yaleB29
yaleB30_P05A+000E+00	yaleB30	yaleB21
yaleB30_P06A+000E+00	yaleB30	yaleB30
yaleB30_P07A+000E+00	yaleB30	yaleB30
yaleB30_P08A+000E+00	yaleB30	yaleB30
yaleB31_P05A+000E+00	yaleB31	yaleB31
yaleB31_P06A+000E+00	yaleB31	yaleB31
yaleB31_P07A+000E+00	yaleB31	yaleB23
yaleB31_P08A+000E+00	yaleB31	yaleB36

Tabel A.30 Hasil Uji Coba pada Skenario Perbedaan Kemiringan Kepala Subjek (Bagian Keempat)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB32_P05A+000E+00	yaleB32	yaleB32
yaleB32_P06A+000E+00	yaleB32	yaleB32
yaleB32_P07A+000E+00	yaleB32	yaleB32
yaleB32_P08A+000E+00	yaleB32	yaleB34
yaleB33_P05A+000E+00	yaleB33	yaleB33
yaleB33_P06A+000E+00	yaleB33	yaleB33
yaleB33_P07A+000E+00	yaleB33	yaleB30
yaleB33_P08A+000E+00	yaleB33	yaleB33
yaleB34_P05A+000E+00	yaleB34	yaleB34
yaleB34_P06A+000E+00	yaleB34	yaleB34
yaleB34_P07A+000E+00	yaleB34	yaleB34
yaleB34_P08A+000E+00	yaleB34	yaleB34
yaleB35_P05A+000E+00	yaleB35	yaleB35
yaleB35_P06A+000E+00	yaleB35	yaleB35
yaleB35_P07A+000E+00	yaleB35	yaleB34
yaleB35_P08A+000E+00	yaleB35	yaleB35
yaleB36_P05A+000E+00	yaleB36	yaleB27
yaleB36_P06A+000E+00	yaleB36	yaleB19
yaleB36_P07A+000E+00	yaleB36	yaleB36
yaleB36_P08A+000E+00	yaleB36	yaleB36
yaleB37_P05A+000E+00	yaleB37	yaleB27
yaleB37_P06A+000E+00	yaleB37	yaleB37
yaleB37_P07A+000E+00	yaleB37	yaleB37
yaleB37_P08A+000E+00	yaleB37	yaleB37
yaleB38_P05A+000E+00	yaleB38	yaleB38

Tabel A.31 Hasil Uji Coba pada Skenario Perbedaan Kemiringan Kepala Subjek (Bagian Kelima)

Nama Berkas	*Ground Truth Label*	*Predicted Label*
yaleB38_P06A+000E+00	yaleB38	yaleB16
yaleB38_P07A+000E+00	yaleB38	yaleB25
yaleB38_P08A+000E+00	yaleB38	yaleB38
yaleB39_P05A+000E+00	yaleB39	yaleB18
yaleB39_P06A+000E+00	yaleB39	yaleB39
yaleB39_P07A+000E+00	yaleB39	yaleB39
yaleB39_P08A+000E+00	yaleB39	yaleB39
yaleB16_P06A+000E+00	yaleB16	yaleB31
yaleB16_P07A+000E+00	yaleB16	yaleB26
yaleB16_P08A+000E+00	yaleB16	yaleB26
yaleB16_P09A+000E+00	yaleB16	yaleB27
Akurasi	67%	
Waktu eksekusi	00:00:07	

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Kevin Zulkarnain Yuseti lahir di kota Kediri, 26 Februari 1995. Dalam riwayat pendidikan, penulis menempuh pendidikan tingkat dasar di Sekolah Dasar Islam Al-Huda Kediri dan lulus pada tahun 2007, melanjutkan ke tingkat Sekolah Menengah Pertama di SMPN 1 Kediri dan lulus pada tahun 2010. Sekolah menengah tingkat akhir dilanjutkan di SMAN 2 Kediri yang lulus pada tahun 2013. Penulis melanjutkan studi ke Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada Jurusan Teknik Informatika Fakultas Teknologi Informasi.

Selama menjalani studi, penulis beberapa kali mengikuti kegiatan keorganisasian. Saat masih SMP, penulis menjadi Pengurus Harian OSIS SMPN 1 Kediri dan Anggota Takmir Masjid Al-Ikhlas SMPN 1 Kediri. Pada saat SMA penulis menjadi pengurus PMR SMAN 2 Kediri. Ketika menempuh pendidikan di ITS, penulis sempat menjadi staf departemen Pengembangan Sumber Daya Mahasiswa Himpunan Mahasiswa Teknik Computer Informatika ITS, anggota tim pengkaji independensi Himpunan Mahasiswa Teknik Computer Informatika ITS, Tim Ad-Hoc Himpunan Mahasiswa Teknik Computer Informatika ITS. Penulis juga berkesempatan menjadi staf dan staf ahli National Logic Competition Schematics HMTC 2014 dan 2015. Penulis juga berkesempatan menjadi asisten pengajar mata kuliah Struktur Data pada tahun 2015 dan 2016. Saat ini penulis sedang diamanahi sebagai asisten Laboratorium Pemrograman 2 Jurusan Teknik Informatika ITS.