Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

PENGENALAN IRIS MATA MENGGUNAKAN EKSTRAKSI FITUR AVERAGE ABSOLUTE DEVIATION DAN GRAY-LEVEL CO-OCCURRENCE MATRIX

RAHMA DINI MAGHFIROTUL LAILY
NRP 05111440000027

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

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

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

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - KI141502

PENGENALAN IRIS MATA MENGGUNAKAN EKSTRAKSI FITUR AVERAGE ABSOLUTE DEVIATION DAN GRAY-LEVEL CO-OCCURRENCE MATRIX

RAHMA DINI MAGHFIROTUL LAILY
NRP 05111440000027

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

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

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

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

IRIS RECOGNITION USING AVERAGE ABSOLUTE DEVIATION AND GRAY-LEVEL CO-OCCURRENCE

RAHMA DINI MAGHFIROTUL LAILY
NRP 05111440000027

Supervisor I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGENALAN IRIS MATA MENGGUNAKAN EKSTRAKSI FITUR AVERAGE ABSOLUTE DEVIATION DAN GRAY-LEVEL CO-OCCURRENCE MATRIX

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
RAHMA DINI MAGHFIROTUL LAILY
NRP : 05111440000027

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. ..... (pembimbing 1)
NIP. 19751220 200112 2 002

Dini Adni Navastara, S.Kom., M.Sc. ..... (pembimbing 2)
NIP. 19851017 201504 2 001

SURABAYA
JUNI 2018

[Halaman ini sengaja dikosongkan]

PENGENALAN IRIS MATA MENGGUNAKAN EKSTRAKSI FITUR AVERAGE ABSOLUTE DEVIATION DAN GRAY-LEVEL CO-OCCURRENCE MATRIX

Nama Mahasiswa	**: RAHMA DINI MAGHFIROTUL LAILY**
NRP	**: 05111440000027**
Jurusan	**: Departemen Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr. Chastine Fatichah, S.Kom, M.Kom.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

ABSTRAK

Pengenalan iris mata adalah teknik pengenalan pola yang digunakan dalam aplikasi biometrik. Teknik biometrik ini menghasilkan pola acak yang unik secara statistik. Dengan kata lain, tekstur iris mata adalah bagian yang unik dimiliki pada masing-masing orang.

Dalam Tugas Akhir ini diimplementasikan pengenalan iris menggunakan metode ekstraksi fitur Gray-Level Cooccurrence Matrix (GLCM) dan metode ekstraksi fitur Average Absolute Deviation (AAD). Metode klasifikasi yang dipakai adalah Artificial Neural Network (ANN). Sebelum dilakukan ekstraksi fitur, terlebih dahulu dilakukan preprocessing untuk identifikasi dan normalisasi bagian iris mata.

Uji coba yang dilakukan menggunakan dataset mata CASIA versi 1.0. Hasil uji coba yang menggunakan metode ekstraksi fitur GLCM dan AAD dengan klasifikasi ANN mendapatkan akurasi terbaik sebesar 89.23%.

Kata kunci: Pengenalan iris, Average Absolute Deviation, Gray-Level Cooccurrence Matrix, Artificial Neural Network.

[Halaman ini sengaja dikosongkan]

IRIS RECOGNITION USING AVERAGE ABSOLUTE DEVIATION AND GRAY-LEVEL CO-OCCURRENCE MATRIX

Student's Name	**: RAHMA DINI MAGHFIROTUL LAILY**
Student's ID	**: 05111440000027**
Department	**: Informatics Department FTIK-ITS**
First Advisor	**: Dr. Chastine Fatichah, S.Kom, M.Kom.**
Second Advisor	**: Dini Adni Navastara, S.Kom., M.Sc.**

ABSTRACT

Iris recognition is a pattern recognition technique used in biometric applications. This biometric technique generates a unique random pattern statistically. In other words, the texture of the iris is the part that is unique to each person.

This final project implements feature extraction methods Gray-Level Cooccurrence Matrix (GLCM) and Average Absolute Deviation (AAD for iris recognition. The classification method used is Artificial Neural Netrowk (ANN). Before features extraction, first performed preprocessing for identification and normalization of the iris.

Trials performed using CASIA version 1.0 eye datasets. The test results using GLCM and AAD feature extraction methods with ANN classification obtained the best accuracy of 89.23%.

Keywords: Iris Recognition, Average Absolute Deviation, Gray-Level Cooccurrence Matrix, Artificial Neural Network.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji dan syukur bagi Allah SWT, yang telah melimpahkan rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

PENGENALAN IRIS MATA MENGGUNAKAN EKSTRAKSI FITUR AVERAGE ABSOLUTE DEVIATION DAN GRAY-LEVEL CO-OCCURRENCE MATRIX

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS.

Tugas Akhir ini selesai karena tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua dan kakak saya, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
4. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing II Tugas Akhir yang telah memberikan banyak yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS

6. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
8. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
9. Teman-teman saya, Tion, Upik, Tepe, Datin, Mala, Afiif, Sita, Lucha, dan Ade, terimakasih banyak telah menemani penulis selama masa pengerjaan Tugas Akhir.
10. Rekan-rekan Admin Laboratorium KCV 2014,2015 dan 2016 yang selalu mendukung penulis dalam menyelesaikan Tugas Akhir.

Penulis memohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Juni 2018

Rahma Dini Maghfirotul Laily

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini, gambaran tugas akhir secara umum dapat dipahami.

1.1. Latar Belakang

Iris merupakan bagian annular antara *sclera* (daerah putih) dan pupil (daerah hitam) dari mata. Iris memiliki corak yang unik untuk setiap orang, dan polanya stabil sejak kecil sampai masa dewasa. Untuk mendapatkan data iris dan mengubahnya menjadi sebuah citra bisa dilakukan dengan menggunakan kamera. Di antara berbagai teknik biometrik, seperti wajah, *palm vein, finger print,* dan tanda tangan, penggunaan citra iris memiliki tingkat keamanan yang paling tinggi. Hal ini karena sifatnya yang unik, stabil dan karakteristik iris yang non-invasif [1].

Pengenalan identitas seseorang berbasis iris semakin banyak digunakan di sektor-sektor besar seperti transportasi (iris sebagai paspor seumur hidup), kriminalitas (iris untuk melacak keberadaan seseorang), dan keamanan (iris sebagai password). Selain itu, penggunaan pengenalan berbasis iris juga bisa ditemukan di bandara dan perbatasan penyeberangan, seperti untuk kontrol imigrasi tanpa paspor atau mendapatkan akses untuk kru bandara ke area terlarang [3]. Secara luas perangkat elektronik juga memanfaatkan pengenalan berbasis iris karena tidak membutuhkan sensor sentuh dan memiliki tingkat pengenalan yang lebih tinggi daripada pengenalan berbasis sidik jari [4]. Pengenalan iris memiliki tantangan mengenai kualitas iris gambar seperti intensitas pixel, kontras, dan posisi simetris pada gambar karena bisa menghasilkan tekstur deformasi. Faktor kualitas gambar iris mempengaruhi keberhasilan proses dalam pengenalan berbasis iris.

Dalam beberapa tahun terakhir, beberapa metode ekstraksi fitur telah diusulkan untuk pengenalan iris. Sistem pengenalan iris

pertama kali dikembangkan oleh Daugman. Metode ini menggunakan 2D yang kompleks. Filter Gabor Daugman memilih filter Gabor untuk pengenalan iris karena frekuensi, orientasi dan representasi filter Gabor mirip dengan manusia sistem visual. Metode ekstraksi fitur lainnya diusulkan oleh Bole dan Boashash [5]. Mereka menggunakan representasi *zero crossing* transformasi wavelet 1D untuk mengkarakterisasi fitur pola iris yang diwakili dengan menggunakan pendekatan *fine-tocoarse*. Metode ini diaplikasikan pada sampel kecil dan sensitif terhadap perubahan nilai abu-abu. Kerangka morfologi grayscale diusulkan oleh Hayashi dan Taguchi [6] untuk mengekstrak fitur gambar iris. Kerangka morfologi gambar iris berasal dari kerangka yang diekstraksi dari pola biner gambar. Bharath dkk [7] mengusulkan *radon tranform thresholding* untuk mengekstrak fitur menonjol iris dari gambar hasil *pre-processing.* Citra *preprocessed* menggunakan gradien berbasis isolasi untuk mendapatkan tekstur iris yang menonjol. Untuk mengetahui fitur yang optimal, Optimasi swarm partikel biner diterapkan sebagai fitur algoritma seleksi.

Dalam tugas akhir ini, akan dikembangkan sistem pengenalan iris menggunakan *Canny Edge Detection* dan *Hough Transform* sebagai metode deteksi daerah iris, *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD) sebagai metode ekstraksi fitur, dan *Artificial Neural Network* (ANN) sebagai metode klasifikasi. Fokus utama Penelitian ini bertujuan untuk mengetahui kinerja ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD) sebagai metode ekstraksi fitur, dan *Artificial Neural Network* (ANN) sebagai metode klasifikasi dalam sistem pengenalan iris. Diharapkan metode ini dapat menghasilkan performa yang baik dalam pengenalan iris.

1.2. Rumusan Permasalahan

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

1. Bagaimana melakukan identifikasi ROI pada citra iris mata?

2. Bagaimana melakukan pengenalan iris menggunakan kombinasi ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD)?
3. Bagaimana melakukan pengenalan iris menggunakan metode klasifikasi *Artificial Neural Network* (ANN)?
4. Bagaimana mengevaluasi kinerja pengenalan citra iris mata menggunakan kombinasi ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD) dengan metode klasifikasi *Artificial Neural Network* (ANN)?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Dataset yang digunakan adalah dataset CASIA v1.0 yang tersedia secara terbuka di internet.
2. Metode ini diimplementasikan menggunakan MATLAB 2017a dan Python 2.7.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membangun aplikasi pengenalan iris mata menggunakan kombinasi metode ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD).

1.5. Manfaat

Pengerjaan tugas akhir ini diharapkan mampu membangun sebuah sistem pengenalan iris yang dapat meningkatkan keamanan.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. Studi literatur

   Pada tahap ini dilakukan pencarian literatur berupa jurnal atau paper yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan Tugas Akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait Deteksi Tepi, *Daugman's Rubber Sheet Model*, *Hough Transform*, *Average Absolute Deviation* (AAD), *Gray-Level Co-Occurrence Matrix* (GLCM), dan *Artificial Neural Network* (ANN). Makalah yang digunakan sebagai acuan adalah "*Feature Extraction Using Statistical Moments of Wavelet Transform for Iris Recognition*" dan"*A Comparison of Feature Extraction Techniques for Diagnosis of Lumbar Intervertebral Degenerative Disc Disease*".

3. Perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Kemudian dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan rancangan yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul pada aplikasi yang dibuat.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan Tugas Akhir

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

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.
2. Bab II Tinjauan Pustaka
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

3. Bab III Perancangan Perangkat Lunak
   Bab ini berisi penjelasan mengenai desain, perancangan, bahan, dan pemodelan proses yang digunakan dalam Tugas Akhir ini yang direpresentasikan dengan *pseudocode*.
4. Bab IV. Implementasi
   Bab ini merupakan pembangunan aplikasi dengan MATLAB dan Python sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab terakhir yang berisi kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.
7. Daftar Pustaka
   Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.
8. Lampiran
   Merupakan bab tambahan yang berisi daftar istilah atau kode-kode sumber yang penting pada sistem.

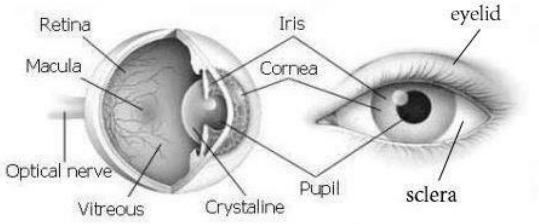
BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pembuatan aplikasi pengenalan iris mata menggunakan ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD). Penjelasan ini bertujuan untuk memberikan dasar teori yang mendasari pengembangan perangkat lunak.

2.1. Anatomi Mata

Bagian organ mata yang akan diteliti dalam tugas akhir ini adalah iris atau selaput pelangi. Bagian-bagian mata seperti yang tampak pada Gambar 2.1 dijelaskan sebagai berikut [2]:

1. Retina adalah bagian yang berfungsi untuk menangkap bayangan benda.
2. Iris adalah bagian yang mempunyai pigmen untuk memberi warna pada mata. Bagian inilah yang akan diteliti pada Tugas Akhir ini.
3. Kornea berfungsi untuk meneruskan cahaya yang masuk ke mata
4. Pupil merupakan bagian mata yang berbentuk lingkaran yang mengatur banyaknya cahaya yang masuk ke mata. Bagian ini juga akan diteliti untuk mendapatkan bagian iris.
5. *Sclera* adalah bagian pelindung mata yang berwarna putih pada mata.
6. Optical Nerve adalah saraf mata yang meneruskan atau membelokkan sinar menuju ke otak.



Gambar 2. 1 Anatomi Mata

2.2. Sistem Pengenalan Iris

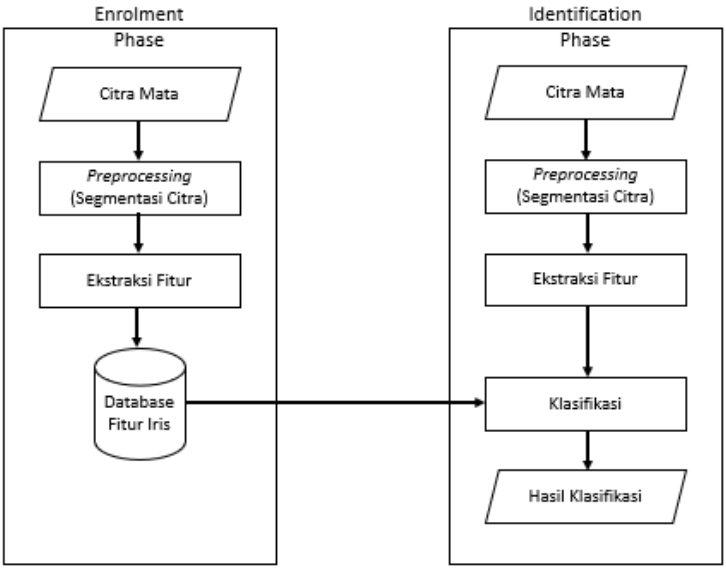
Pengenalan iris adalah suatu proses untuk mengenal seseorang dengan menganalisis pola acak dari iris. Proses ini bertujuan untuk mengenali suatu objek dengan cara mengekstraksi informasi yang terdapat di dalam citra tersebut.

Sudah banyak metode yang digunakan dalam proses pengenalan iris mata. Pada proses identifikasi iris sampai tahap normalisasi umumnya relatif sama. Pada proses ekstraksi fitur dapat menggunakan metode *Average Absolute Deviation* (AAD), *Gray-Level Co-occurrence Matrix* (GLCM) , dan berbagai metode ekstraksi fitur lainnya. Sedangkan pada proses pengenalan, ada pula klasifikasi *Artificial Neural Network* (ANN) [8] dan metode-metode yang lain.

Secara garis besar, proses-proses tersebut dikelompokkan menjadi empat proses utama, yaitu [2]:

1. Akuisisi citra
2. Praproses citra (identifikasi objek iris dan normalisasi)
3. Ekstraksi fitur
4. Pencocokkan iris

Akuisisi citra adalah proses mendapatkan citra mata yang akan dilakukan pengenalan. Citra mata yang digunakan pada tugas akhir ini yaitu citra mata CASIA v1.0 [11]. Proses selanjutnya adalah praproses citra atau melakukan pengolahan awal pada citra mata yang bertujuan untuk mengambil karakteristik tekstur iris mata dan melakukan normalisasi iris untuk mengatasi kondisi citra mata yang bervariasi pada setiap orang. Tekstur iris yang sudah terambil akan diekstraksi ciri atau fitur iris. Fitur yang didapatkan akan dibandingkan dengan fitur masukan dalam proses pencocokkan iris. Proses pencocokkan fitur masukan yang telah dibandingkan dengan fitur yang ada akan menghasilkan bobot atau tingkat kemiripan. Nilai tingkat kemiripan yang paling tinggi adalah citra yang dikenali paling mirip dengan citra masukan.



Gambar 2. 2 Sistem Pengenalan Iris secara Umum [1]

2.3. Deteksi Tepi *Canny*

Deteksi tepi merupakan langkah awal dalam pemrosesan citra digital untuk mendapatkan informasi dari sebuah citra. Untuk mendapatkan informasi berbeda dari citra dengan jenis yang berbeda memerlukan metode yang berbeda pula. Tepi merupakan bagian dari citra yang memiliki perubahan nilai intensitas tingkat keabuan yang besar yang dapat memperlihatkan rincian pada citra. Tujuan deteksi tepi adalah untuk mengekstraksi fitur penting pada suatu citra, misalnya garis, lingkaran, lengkungan, ujung. Salah satu metode deteksi tepi yang terkenal adalah deteksi tepi *Canny*.

Deteksi tepi *Canny* sangat cocok digunakan untuk pengolahan awal citra digital. Deteksi tepi ini sangat cocok digunakan oleh peneliti karena memiliki keuntungan sebagai berikut:

1. Tepi yang dihasilkan akurat
2. Banyak tepi yang dihasilkan

Pada citra iris mata, deteksi tepi ini dilakukan dengan mendeteksi citra iris mata dengan langkah-langkah berikut [12].

a. Mengurangi *noise* dengan menggunakan filter Gaussian sebesar 13 x 13. Standar deviasi iris dengan *sclera* dan iris dengan pupil sebesar 2 [13]. Citra yang dihasilkan dari filter Gaussian akan tampak sedikit buram.
b. Menentukan intensitas gradien dari citra, yaitu dengan menentukan arah gradien ke suatu arah tertentu. Proses ini dilakukan untuk mendapatkan kekuatan tepi (*edge strength*). Tepian harus ditandai pada gambar yang memiliki gradien yang besar. Digunakan operator gradien dan dilakukan pencarian secara horizontal dan vertikal.

G_x =

-1	0	+1
-2	0	+2
-1	0	+1

G_y =

+1	+2	+1
0	0	0
-1	-2	-1

Perhitungan intensitas gradien dilakukan dengan menggunaakan persamaan (2.1).

$$G = \sqrt{Gx^2 + Gy^2} \quad (2.1)$$

Di mana, G_x = gradien ke arah sumbu x
G_y = gradien ke arah sumbu y
G = intensitas gradien

Selanjutnya mencari arah gradien dengan persamaan (2.2).

$$\theta = \arctan\left(\frac{Gy}{Gx}\right) \quad (2.2)$$

di mana θ adalah arah gradien.

Setelah arah tepi diperoleh, maka dilakukan pengelompokkan pada arah gradien yang diperoleh dengan kondisi berikut.

- Jika arah tepi berkisar antara 0° - 22.5° serta 157.5° - 180°, arah gradien diubah menjadi 0° (horizontal).

- Jika arah tepi yang berkisar antara 22.5° - 67.5°, arah gradien diubah menjadi 45° (diagonal kanan).
- Jika arah tepi berkisar antara 67.5° - 112.5°, arah gradien diubah menjadi 90° (vertikal).
- Jika arah tepi berkisar antara 112.5° - 157.5°, arah gradien diubah menjadi 135° (diagonal kiri).

Karena bagian iris berada di antara tepi *sclera* dan tepi iris, serta tepi iris dan tepi pupil, maka dilakukan pendeteksian berturut-turut dengan menggunakan arah gradien vertikal dan horizontal.

c. Mengatur Gamma atau pengontrasan terhadap arah yang telah terdeteksi.Pengontrasan dilakukan menggunakan persamaan (2.3)

$$M = \frac{Imax - Imin}{Imax + Imin} \quad (2.3)$$

Di mana I_{max} = nilai citra intensitas maksimum
I_{min} = nilai citra intensitas minimum
M = nilai citra intensitas rata – rata

Kemudian mendapatkan hasil akhir pengontrasan dengan persamaan (2.4)

$$C = M^{1/g} \quad (2.4)$$

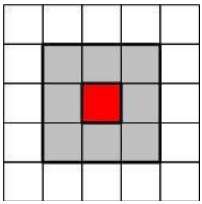
Di mana C = citra gamma
g = nilai gamma
M = nilai citra intensitas rata – rata

Citra akan semakin gelap jika nilai gamma berada diantara 0 – 1, namun semakin terang jika nilai gamma di atas 1. Pengontrasan dilakukan untuk mempermudah dalam penelusuran tepi citra iris mata.

d. Proses non-maximum suppression atau memperkecil garis tepi yang muncul sehingga menghasilkan garis tepi yang lebih ramping.Proses ini menjadikan nilai piksel yang dianggap tidak layak menjadi sebuah tepi dengan nilai nol dari citra tepi pengontrasan.

e. Langkah terakhir adalah proses *hystheresis*. Proses ini menerapkan dua buah bilangan threshold, yaitu *high threshold* (T1) dan *low threshold* (T2). Kondisi penelusuran tepi pada proses ini adalah sebagai berikut:
   - Semua piksel di atas T1 ditandai sebagai tepi.
   - Semua piksel yang berdekatan dengan titik yang telah ditandai sebagai tepi dan mempunyai nilai di atas T2 juga ditandai sebagai tepi.

   Menggunakan 8-konektivitas piksel untuk mengetahui titik yang berdekatan dengan titik yang telah ditandai sebagai tepi. Dalam gambar 2.4 memperlihatkan ilustrasi 8-konektivitas.



Gambar 2. 3 Ilustrasi 8-konektivitas

Warna merah menunjukkan titik yang telah ditandai sebagai tepi jika memenuhi syarat satu. Sedangkan warna abu merupakan koordinat pixel yang akan ditandai sebagai tepi jika memenuhi syarat dua.

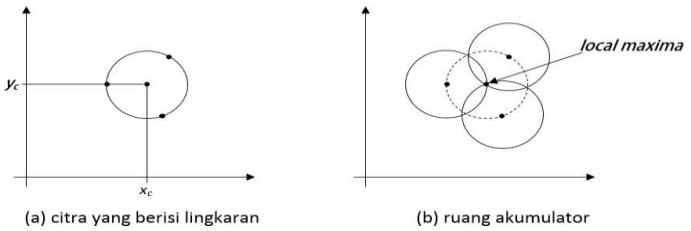
2.4. Transformasi Hough

Tranformasi Hough adalah suatu metode untuk mencari bentuk fitur tertentu seperti garis, lingkaran, atau bentuk sederhana lain dalam suatu citra. Pada umumnya, transformasi Hough diterapkan pada banyak permasalahan dalam visi komputer karena kebanyakan citra mempunyai batas tepi yang membentuk sebuah garis. Implementasi transformasi Hough menjelaskan sebuah pemetaan pada ruang akumulator. Dalam hal ini, transformasi Hough membahas pada pencarian bentuk lingkaran atau biasa disebut Transformasi Hough Lingkaran (*Circular Transform Hough*). Pencarian lingkaran ini

menggunakan persamaan (2.5) untuk mencari titik pusat dan jari-jari pada lingkaran iris dan lingkaran pupil [14].

$$(x_c - a)^2 + (y_c - b)^2 = r^2 \quad (2.5)$$

Di mana (x_c,y_c) = koordinat titik-titik lingkaran
(a,b) = koordinat pusat lingkaran
r = jari-jari lingkaran



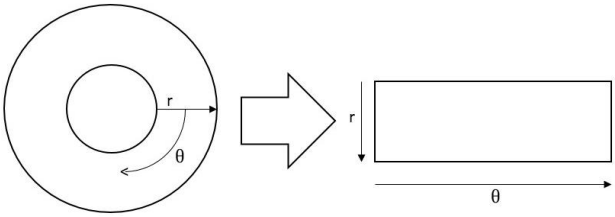
Gambar 2. 4 Ilustrasi Transformasi Hough Lingkaran

Pada proses pendeteksian iris, transformasi Hough lingkaran digunakan untuk mendeteksi area iris dengan cara mencari radius dan titik pusat lingkaran pupil dan iris. Tahap awal dari proses ini adalah mendeteksi tepi citra. Tujuan dari deteksi tepi adalah untuk menurunkan jumlah titik dalam pencarian ruang objek. Ketika titik tepi sudah ditemukan, transformasi Hough akan bekerja pada titik tersebut [12].

Circular Hough Transform membentuk lingkaran sepanjang sepanjang titik yang ditemukan dengan jari-jari sebesar r. Kemudian mencari *voting* untuk mencari titik yang sering dilewati (*local maxima*) dari lingkaran yang telah dibentuk dan titik tersebut diasumsikan sebagai titik pusat lingkaran. Pada Gambar 2.5 merupakan ilustrasi transformasi Hough lingkaran. Hasil akhirnya adalah sebuah citra iris melalui transformasi Hough.

2.5. Transformasi Polar

Setelah daerah iris teridentifikasi dari citra mata, langkah berikutnya adalah melakukan transformasi citra iris agar memiliki ukuran yang konstan sehingga memungkinkan pencocokkan. Pada dasarnya, daerah iris akan menyempit dan merenggang disebabkan oleh pelebaran pupil karena faktor pencahayaan yang beragam. Faktor lain yang mempengaruhi pelebaran pupil yaitu jarak citra, rotasi dan derajat kemiringan. Proses normalisasi akan menghasilkan daerah iris yang memiliki ukuran konstan, sehingga jika dilakukan pencocokkan antara dua daerah iris akan lebih mudah [13].



Gambar 2. 5 ***Daugman's Rubber Sheet Model***

Metode yang digunakan untuk melakukan normalisasi adalah metode *Daugman's rubber sheet model*. Metode ini bertujuan untuk mengubah citra iris yang sebelumnya berbentuk lingkaran menjadi bentuk persegi Panjang seperti yang terlihat pada Gambar 2.5.

Secara matematis, memetakan kembali setiap titik dalam iris ke dalam sepasang koordinat polar (r, θ), dimana r dalam interval [0,1] dan θ dalam interval sudut [0, 2 π]. Pemetaan tersebut dapat dilakukan dengan menggunakan persamaan (2.6).

$$I(x(r, \theta), y(r, \theta)) \rightarrow I(r, \theta) \tag{2.6}$$

Dengan

$$x(r, \theta) = (1 - r)\, x_p(\theta) + r x_i(\theta) \tag{2.7}$$

$$y(r, \theta) = (1 - r)\, y_p(\theta) + r y_i(\theta) \tag{2.8}$$

dimana $I(x, y)$ adalah citra iris

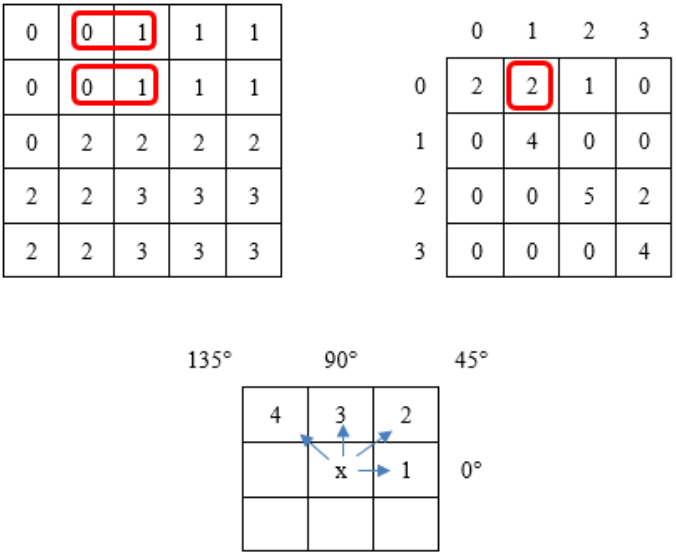
(x, y) adalah titik pada koordinat Cartesian.

(r, θ) adalah koordinat pada polar normalisasi.
x_p, y_p adalah koordinat batas tepi pupil.
x_i, y_i adalah koordinat batas tepi.

2.6. Seleksi Fitur

Seleksi fitur digunakan untuk memilih fitur-fitur yang akan digunakan dalam tahap selanjutnya. Dalam tugas akhir ini proses seleksi hanya akan dilakukan pada fitur hasil ekstraksi fitur *Gray-Level Co-Occurrence Matrix*. Seleksi fitur yang digunakan adalah *univariate feature selection* yang memilih fitur terbaik berdasarkan pengujian statistik univariat. Pemilihan fitur yang akan digunakan adalah dengan Uji F ANOVA pada tiap fitur [14].

2.7. Ekstraksi Fitur *Gray-Level Co-Occurrence Matrix*



Gambar 2. 6 Pembuatan *Matrix Cooccurrence*

Suatu proses klasifikasi citra berbasis analisis tekstur pada umumnya membutuhkan tahapan ekstraksi ciri, yang terdiri dari tiga

macam metode yaitu metode statistik, metode spaktral dan metode struktural. Pada metode statistik, fitur tekstur dihitung berdasarkan distribusi statistik dari kombinasi intensitas piksel pada posisi tertentu dalam suatu matriks citra.

Dalam hal ini, *Gray-Level Co-Occurrence Matrix* (GLCM) termasuk dalam metode statistik. Perhitungan metode ini menggunakan distribusi derajat keabuan (histogram) dengan mengukur tingkat kekontrasan, granularitas dan kekasaran suatu daerah dari hubungan ketetanggan antar piksel di dalam citra.
Ekstraksi ciri pada metode statistik GLCM dilakukan pada orde kedua, yaitu dengan matriks kookurensi. Matriks kookurensi adalah suatu matriks yang merepresentasikan hubungan ketetanggaan antar piksel dalam citra pada jarak dan orientasi tertentu. Kookurensi berarti kejadian sama, yaitu jumlah kejadian satu level nilai piksel bertetangga dengan satu level nilai piksel lain dalam jarak (d) dan orientasi sudut (θ) tertentu. Jarak dinyatakan dalam piksel (1 piksel, 2 piksel, 3 piksel dan seterusnya). Sedangkan orientasi dinyatakan dalam sudut dan dibentuk dalam empat arah sudut dengan interval sudut 45°, yaitu 0°, 45°, 90° dan 135°. Jumlah elemen kookurensi sebesar kuadrat jumlah level intensitas piksel pada citra. Setiap titik *(i, j)* pada matriks kookurensi berisi peluang kejadian piksel bernilai i bertetangga dengan piksel bernilai j pada jarak d dan orientasi θ.

Berikut ini adalah gambaran pembentukan GLCM atas citra dengan 4 tingkat keabuan (gray level) pada jarak d=1 dan arah 0°. Gambar 3 menunjukkan arah dan jarak yang dapat dipakai dalam menghitung GLCM sebuah citra. Dari pixel di tengah (x) pixel 1 menunjukkan arah 0° dengan jarak d=1; pixel 2 menunjukkan arah 45° dengan jarak d=1; pixel 3 menunjukkan arah 90° dengan jarak d=1; dan pixel 4 menunjukkan arah 135° dengan jarak d=1, seperti Gambar 2.6. Matriks kookurensi menangkap sifat tekstur tetapi tidak secara langsung dapat digunakan sebagai alat analisis, misalnya membandingkan dua tesktur [18].

Setelah didapatkan matriks kookurensi, maka selanjutnya matriks GLCM dapat dihasilkan dari penjumlahan matriks kookurensi dengan matriks *transpose* dari matriks kookurensi tersebut. Matriks

GLCM yang telah didapat, harus disarikan lagi agar mendapat angka-angka yang bisa digunakan untuk mengklasifikasikan tekstur. Ciri atau fitur statistik GLCM yang dapat diterapkan antara lain [19] :

1. Kontras
   Perhitungan kontras berkaitan dengan jumlah keberagaman intensitas keabuan dalam citra.

$$\text{Kontras} = \sum_{a,b=0}^{n-1} |a-b|^2 \; P_{a,b} \tag{2.10}$$

2. Homogenitas
   Secara matematis, homogenitas GLCM adalah *invers* dari kontras GLCM, yaitu keseragaman intensitas keabuan pada citra.

$$\text{Homogenitas} = \sum_{a,b=0}^{n-1} \frac{P_{a,b}}{1+|a-b|} \tag{2.11}$$

3. Energi
   Energi menyatakan ukuran konsentrasi pasangan dengan intensitas keabuan tertentu pada matriks.

$$\text{Energi} = \sum_{a,b=0}^{n-1} P_{a,b}^2 \tag{2.12}$$

4. Entropi
   Entropi digunakan untuk mengukur keteracakan dari distribusi intensitas.

$$\text{Entropi} = - \sum_{a,b=0}^{n-1} P_{a,b} \; log_2 \, (P_{a,b}) \tag{2.13}$$

5. Korelasi
   Menyatakan ukuran hubungan ketergantungan piksel terhadap piksel tetangga dalam citra.

$$\text{Korelasi} = \sum_{a,b=0}^{n-1} P_{a,b} \frac{(a - \mu_x)(b - \mu_y)}{\sigma_x \, \sigma_y} \tag{2.14}$$

$$\mu_x = \sum_{a=0}^{n-1} a \sum_{b=0}^{n-1} P_{a,b} \quad (2.15)$$

$$\mu_y = \sum_{b=0}^{n-1} b \sum_{a=0}^{n-1} P_{a,b} \quad (2.16)$$

$$\sigma_x = \sum_{a=0}^{n-1} (a - \mu_x)^2 \sum_{b=0}^{n-1} P_{a,b} \quad (2.17)$$

$$\sigma_x = \sum_{b=0}^{n-1} (b - \mu_y)^2 \sum_{a=0}^{n-1} P_{a,b} \quad (2.18)$$

6. *Dissimilarity*
   Mengukur ketidakmiripan suatu tekstur. Apabila teksturnya acak, maka nilai *dissimilarity* besar. Namun sebaliknya, apabila terksturnya seragam, maka nilai *dissimilarity* kecil.

$$Dissimilarity = \sum_{a,b=0}^{n-1} |a - b| P_{a,b} \quad (2.19)$$

7. *Autocorrelation*
   Digunakan untuk Mengukur correlation diantara garis diagonal utama.

$$Autocorrelation = \sum_{a,b=0}^{n-1} (ab) P_{a,b} \quad (2.20)$$

8. *Sum of Squares*
   Dikenal juga sebagai *variance*, yaitu mean dari kuadrat jarak setiap nilai intensitas terhadap nilai mean. Ini merupakan ukuran dari penyebaran distribusi mean.

$$\sigma_x^2 = \sum_{a,b=0}^{n-1} (a - \mu_x)^2 \; P_{a,b} \tag{2.21}$$

$$\sigma_y^2 = \sum_{a,b=0}^{n-1} (a - \mu_y)^2 \; P_{a,b} \tag{2.22}$$

9. *Cluster Shade*
   Ukuran kemiringan dan keseragaman matriks GLCM. *Cluster shade* yang lebih tinggi menyiratkan asimetri yang lebih besar juga.

$$\textit{Cluster Shade} = \sum_{a,b=0}^{n-1} (a + b - \mu_x - \mu_y)^3 \; P_{a,b} \tag{2.23}$$

10. *Cluster Prominence*
    Cluster prominence juga merupakan ukuran asimetri. Ketika nilai *cluster prominence* tinggi, maka citra kurang simetris. Selain itu, ketika nilainya rendah, ada puncak dalam matriks GLCM di sekitar nilai rata-rata.

$$\textit{Cluster Prominence} = \sum_{a,b=0}^{n-1} (a + b - \mu_x - \mu_y)^4 \; P_{a,b} \tag{2.24}$$

Dimana, *a* adalah nomor index baris
b adalah nomor index kolom

$P_{a,b}$ adalah nilai piksel matriks GLCM pada baris ke-*a* dan kolom ke-*b*
μ_x adalah nilai mean pada sumbu *x*
μ_y adalah nilai mean pada sumbu *y*
σ_x adalah nilai standar deviasi pada sumbu *x*
σ_y adalah nilai standar deviasi pada sumbu *y*

2.8. Ekstraksi Fitur *Average Absolute Deviation*

Ekstraksi fitur memungkinkan untuk mendapatkan perbedaan informasi dalam sebuah citra. Informasi ini dapat ditampilkan sebagai sebuah vektor fitur *Average Absolute Deviation* didefinisikan sebagai berikut [8].

$$\text{AAD} = \frac{1}{N}\sum_{N}(|f(x,y) - m|) \tag{2.9}$$

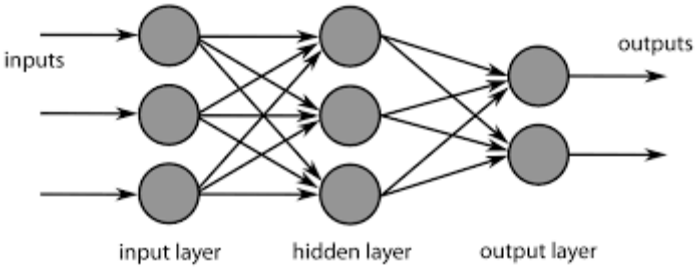
Dimana *N* adalah jumlah piksel dalam citra
f(x,y) adalah nilai piksel pada titik *(x,y)*
m adalah nilai mean dari citra
Metode AAD diterapkan pada *sub-image* dalam *image* yang didapatkan dari *pre-processing*. Setiap *sub-image* akan menghasilkan satu nilai fitur. Pada Tugas Akhir ini, setiap *sub-image* akan dicoba variasi ukuran piksel yang berdimensi dua.

2.9. *Artificial Neural Network* (ANN)

Artificial Neural Network [17,18] atau juga sering disebut Jaringan Saraf Tiruan dikembangkan melalui perseptron neuron biologis. NN merupakan program computer yang dilatih untuk berperilaku dan belajar secara analog seperti otak manusia yang mampu mengingat informasi masa lalu. Metode ini mencoba untuk meniru jaringan saraf otak biologis manusia menjadi model matematika. Model matematika yang dibuat ini adalah kumpulan dari

unit pemrosesan sederhana, saling terkait dengan bobot yang diberikan untuk koneksi.

Ada tiga jenis struktur jaringan saraf yang diusulkan dalam literatur [16]. Unit pemrosesan dalam *Multi Layer Perceptron* (MLP) disusun oleh tiga layer. Layer pertama merupakan layer input yang berisi informasi yang akan digunakan dalam membuat keputusan. Layer kedua adalah *hidden layer* yang akan membantu jaringan untuk menghitung asosiasi yang lebih rumit. Layer ketiga adalah layer output yang berisi keputusan yang dihasilkan. Struktur dasar MLP *feed-forward* ditunjukkan pada Gambar 2.8.



Gambar 2. 7 Struktur MLP-ANN

Jaringan MLP berisi satu layer input, satu atau lebih *hidden layer* dan satu layer output. Jumlah neuron pada input dan output diatur oleh jumlah input dan output dari pola yang akan dikenali. Namun, jumlah neuron pada *hidden layer* dapat dipilih sesuai dengan aplikasi. Masing – masing neuron dalam layer input dapat langsung masuk ke *hidden layer* dengan pemberian bobot terlebih dahulu. Hasil penjumlahan dari bobot dan input dihitung pada setiap *node*. Nilai hasil perhitungan langsung masuk ke neuron output melalui pembobotan. Sama seperti pada *hidden layer*, hasil penjumlahan dari bobot dan output neuron pada *hidden layer* dihitung disetiap *node* di layer output. Jika error antara nilai output dengan nilai keluaran yang diinginkan lebih dari *error ratio*, maka proses training (mengubah bobot dan menghitung ouput baru dengan menggunakan bobot baru)

dimulai. Proses training dapat berakhir jika mendapatkan tingkat error yang diinginkan untuk semua kombinasi input [19].

Di dalam jaringan MLP, terdapat fungsi aktivasi yang digunakan untuk mengaktifkan dan menonaktifkan neuron. Fungsi aktivasi merupakan operasi metematik yang dikenakan pada sinyal output. Beberapa fungsi aktivasi yang sering digunakan yaitu, *identity*, *logistic*, *sigmoid tangen* (tanh), dan relu. Penjelasan masing – masing fungsi aktivasi dijelaskan pada tabel 2.1.

Tabel 2.1 Fungsi Aktivasi

No	Fungsi Aktivasi	Penjelasan
1.	*Identity*	memiliki nilai output yang sama dengan nilai inputnya
2.	*Logistic*	fungsi aktivasi yang membawa input ke output dengan perhitungan log-sigmoid dengan nilai output antara -1 hingga 1.
3.	*Sigmoid tangen* (Tanh)	fungsi yang membawa nilai input pada output dengan menggunakan rumus *hyperbolic tangen sigmoid* dengan nilai maksimal output 1 dan minimal -1
4.	Relu	*rectified linier unit function* merupakan fungsi yang membuat pembatas pada bilangan nol. Jika elemen bernilai negatif maka nilainya diset menjadi 0, tidak ada operasi eksponensial, perkalian atau pembagian. Dengan karakteristik seperti itu, kelebihan Relu akan muncul saat berhadapan dengan jaringan yang memiliki neuron yang banyak sehingga dapat mengurangi waktu training dan testing dengan signifikan.

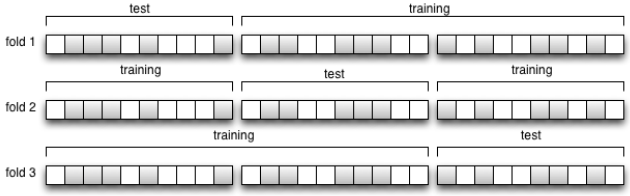
Dalam MLP juga terdapat beberapa metode optimasi, seperti *Stochastic Gradient Descent* (SGD), *Adaptive Moment Estimation*

(Adam), dan *Limited-Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) Berikut adalah penjelasan dari ketiga metode optimasi.

1. SGD
   Stochastic Gradient Descent (SGD) melakukan pembaruan parameter untuk setiap contoh pelatihan. Karena seringnya pembaruan ini, pembaruan parameter memiliki varian tinggi dan menyebabkan fungsi berfluktuasi ke intensitas yang berbeda. Ini sebenarnya hal yang baik karena membantu kami menemukan *local minima* baru dan mungkin lebih baik. Tetapi, karena seringnya pembaruan dan fluktuasi, hal itu pada akhirnya mempersulit konvergensi ke minimum yang pasti dan akan terjadi *overshooting* karena fluktuasi yang sering terjadi. Nilai *learning rate* pada algoritma optimasi ini tidak berubah sampai akhir.
2. Adam
   Adam merupakan algoritma optimisasi stokastik berdasarkan perkiraan adaptif dari momen order rendah. Algoritma Adam pertama kali diperkenalkan oleh Kingma & Ba. Metode ini dapat diimplementasikan dengan mudah, memiliki komputasi yang efisien, memiliki kebutuhan memori yang kecil, invarian terhadap penskalaan gradien dan cocok diterapkan pada data atau parameter dengan jumlah yang besar. Algoritma Adam cocok diterapkan pada permasalahan data yang sangat berderau atau gradien yang menyebar.
3. L-BFGS
   Metode optimisasi yang digunakan dalam pemrograman non-linier untuk menemukan nilai global minima. Metode ini adalah salah satu varian dari metode Quasi-Newton yang dibuat berdasarkan metode Newton yang sulit dan memiliki waktu komputasi yang lama. Karena kelemahan tersebut, dikembangkanlah L-BFGS yang memiliki keunggulan di waktu komputasi yang lebih sedikit dibandingkan metode BFGS (limited memory pada L-BFGS berarti memerlukan memori yang sedikit.

2.10. *K-Fold Cross Validation*

Adalah salah satu teknik validasi model terhadap set data independen. Secara umum caranya dengan membagi sebuah dataset menjadi K sub-dataset. Dari semua sub akan diambil satu sub sebagai data uji, dan sisanya menjadi data latih, begitu seterusnya sampai sub ke- K menjadi data uji. Kemudian performa dihitung dari rata-rata hasil *cross validation*.



Gambar 2. 5 Ilustrasi Cross Validation Menggunakan Tiga Fold

BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan diimplementasikan. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan Tugas Akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi *pre-processing,* ekstraksi fitur *Gray-Level Co-Occurrence Matrix* dan ekstraksi fitur *Average Absolute Deviation* .

3.1. Data

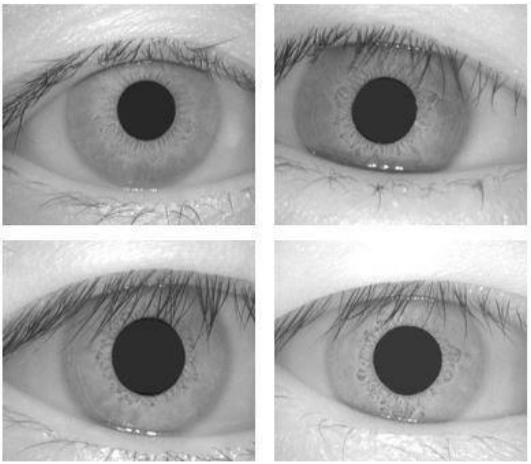
Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

Data yang digunakan pada proses pengenalan iris dibagi menjadi tiga macam yaitu data masukan, data proses dan data keluaran. Data masukan merupakan input dari pengguna perangkat lunak. Data proses adalah data ketika tahap – tahap pengenalan iris sedang dilakukan. Sedangkan data keluaran adalah data yang ditampilkan dari hasil proses pengenalan iris. Penjelasan masing – masing jenis data diberikan sebagai berikut.

3.1.1. Data Masukan

Data masukan adalah data yang digunakan sebagai masukan dari sistem. Data yang digunakan berupa citra mata yang bersumber dari *database* citra mata *Chinese Academy of Science-Institute of Automation* (CASIA) versi 1.0 seperti yang telah dibahas pada Batasan permasalahan tugas akhir. Database ini berisi 756 citra mata dari 108 mata yang berbeda. Masing – masing mata dianggap sebagai satu kelas. Di dalam satu kelas terdapat tujuh citra mata. Sehingga total keseluruhan citra mata pada database ini yaitu 108 x 7 = 756 citra mata.Semua citra pada database ini bertipe *grayscale.*

Tiap kelas pada database citra mata ini diambil dalam dua sesi dengan interval satu bulan pada sesinya. Citra mata ini diambil dengan alat optic khusus yang dikembangkan oleh *National Laboratory of Pattern Recognition*, China, dengan tujuan penelitian pengenalan iris. Pada Gambar 3.1 Citra Mata CASIA adalah contoh citra mata dari *database* CASIA.



Gambar 3. 1 Citra Mata Beberapa Individu dari Dataset CASIA v1.0

Database ini dibagi menjadi dua, yaitu data latih dan data uji. Data latih merupakan data yang digunakan untuk melatih sistem agar keluarannya sesuai dengan apa yang diharapkan. Sedangkan data uji merupakan data yang digunakan untuk menguji sistem setelah melakukan proses *training*. Pembagian data latih dan data uji akan dibagi dengan perbandingan yang sama antar kelasnya.

3.1.2. Data Proses

Data proses adalah data yang digunakan selama proses berjalan. Data proses ditunjukkan dalam table 3.1.

Tabel 3. 1 Data Proses

No.	Nama data	Keterangan
1.	Data citra mata	Data dari database yang akan digunakan pada tahap *preprocessing* untuk mendapatkan citra mata. Data ini bertipe *bitmap*
2.	Data citra iris	Data ini digunakan dapat tahap ekstraksi fitur untuk mendapatkan nilai − nilai fitur yang akan digunakan pada proses selanjutnya. Data ini bertipe *.bitmap*
3.	Data fitur	Data ini digunakan dapam proses klasifikasi. Data ini bertipe *.double*

Pemrosesan akan dilakukan dengan menggunakan data masukan yang berupa citra mata *grayscale* bertipe *bitmap*. Data citra ini selanjutnya akan diolah sehingga menghasilkan nilai – nilai fitur yang dapat digunakan untuk data latih dan dan uji dalam proses klasifikasi. Data latih digunakan sebagai data belajar pada proses klasifikasi, sedangkan data uji sebagai masukan untuk pencocokan. Data hasil *preprocessing* berupa data citra iris mata yang akan diproses sampai menjadikan vektor fitur citra iris. Data vektor fitur juga disimpan dalam tipe data double.

3.1.3. Data Keluaran

Data keluaran dari sistem ini adalah hasil dari proses – proses yang sudah dilakukan. Data masukan yang sudah diekstraksi fitur akan diproses dengan menggunakan klasifikasi *Artificial Neural Network*. Hasil dari proses klasifikasi tersebut adalah prediksi kelas pada masing – masing data uji dan nilai evaluasinya.

3.2. Desain Sistem Secara Umum

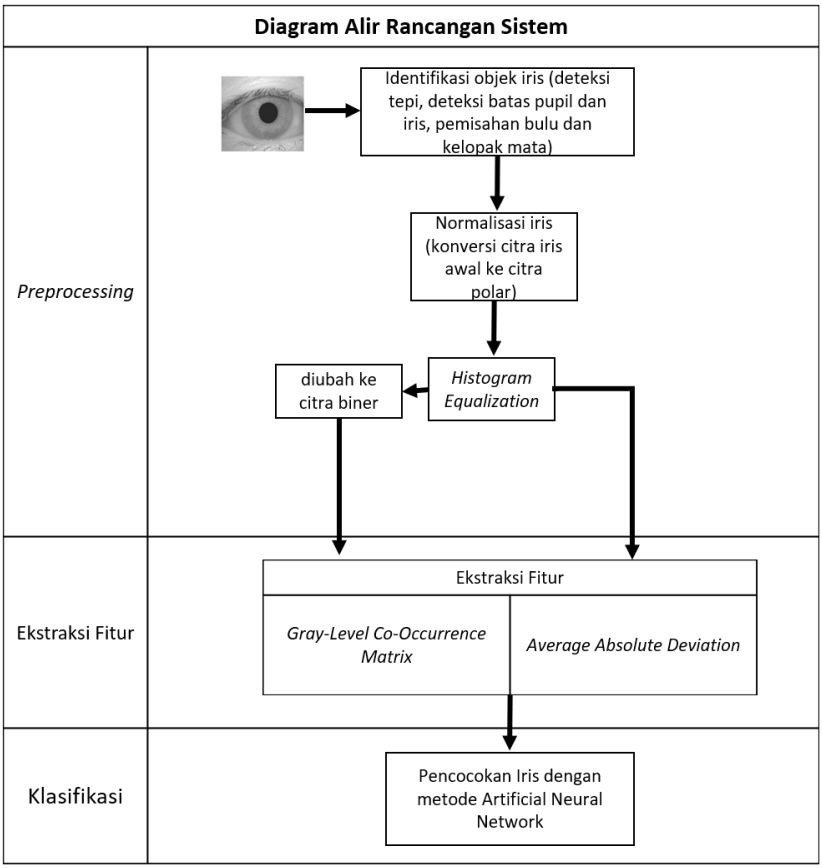
Secara garis besar ada tiga tahapan utama dalam melakukan pengenalan iris yaitu praproses, ekstraksi fitur dan klasifikasi. Namun dalam melakukan pengenalan iris ini, tiga tahap tersebut diperinci menjadi empat tahapan sebagai berikut:

1. *Preprocessing* citra mata
2. Normalisasi iris
3. Ekstraksi fitur iris
4. Klasifikasi iris

Tahapan yang pertama merupakan tahap praproses. Data yang dijadikan masukan dalam tahap *preprocessing* merupakan citra mata yang didapatkan dari tahapan akuisisi citra mata dari *disk*. Setelah mendapatkan sebuah citra mata, citra mata tersebut akan dipanggil , sehingga tahap *preprocessing* dapat dilakukan. Tujuan tahap *preprocessing* adalah untuk mengidentifikasi bagian iris pada citra mata. Tahap ini terdiri dari tiga proses, yaitu deteksi tepi, deteksi batas pupil dan iris, dan pemisahan bulu mata dan kelopak mata. Hasil dari tahap ini adalah citra iris.

Bagian iris yang sudah teridentifikasi kemudian dinormalisasi. Tujuan dari normalisasi adalah agar dimensi iris menjadi konstan. Hal ini disebabkan karena luasan iris manusia berubah – ubah tergantung dari berbagai faktor. Normalisasi iris juga memudahkan proses verifikasi. Hasil dari tahap praproses diolah kembali dengan menggunakan transformasi polar pada citra iris sehingga menghasilkan sebuah citra iris yang mudah untuk diolah pada tahap selanjutnya.

Setelah proses normalisasi, maka dilakukan tahap ekstraksi fitur terhadap citra iris. Ekstraksi fitur berguna untuk mendapatkan fitur iris. Pada tahap ini digunakan dua metode ekstraksi fitur yaitu, *Gray-Level Co-Occurrence Matrix* dan *Averange Absolute Deviation*. Kedua metode tersebut akan diujikan pada metode klasifikasi *Artificial Neural Network*. Hasil akhir yang dikeluarkan adalah nilai evaluasi yang menjadi tolok ukur keakuratan proses klasifikasi.

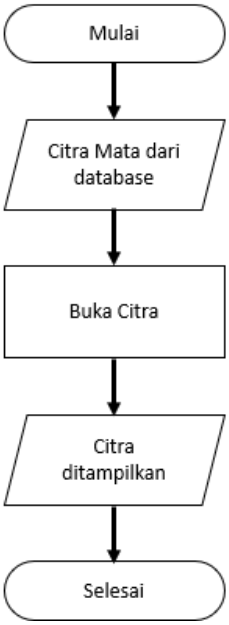


Gambar 3. 2 Diagram Alir Rancangan Sistem

Gambar 3.2 menunjukkan tiga tahapan utama dalam pengenalan iris, yaitu tahap *preprocessing*, ekstraksi fitur dan klasifikasi. Tahap *preprocessing* adalah tahap mengidentifikasi iris dari input citra mata. Tahap ekstraksi fitur terdiri dari dua metode, yaitu *Average Absolute Deviation* dan *Gray-Level Co-Occurrence Matrix*. Pada tahap klasifikasi digunakan metode *Artificial Neural Network*.

3.3. Akuisisi Citra Mata

Seperti yang telah dijelaskan pada bagian perancangan data, mata akan digunakan sebagai masukan dalam proses pengenalan iris. Citra mata diasumsikan telah disimpan di dalam *disk* komputer. Dalam tugas akhir ini, citra mata yang digunakan berasal dari *database* CASIA versi 1. Diagram alir proses akuisisi citra mata dapat dilihat pada Gambar 3.3.



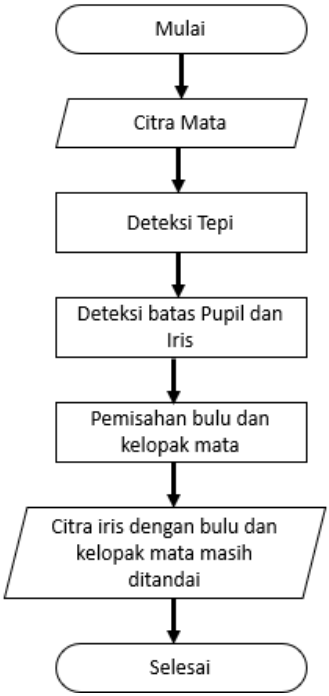
Sumber : Tugas Akhir Afdhal Basith A.

Gambar 3. 3 Diagram Alir Akuisisi Citra Mata

Tahap awal dari proses ini adalah memilih sebuah citra mata dari *disk*. Kemudian dari *disk* tersebut citra akan dipanggil ke dalam sistem.

3.4. Perancangan *Preprocessing*

Citra mata yang sudah diakuisisi tidak bisa langsung dilakukan verifikasi. Tahap awal yang dilakukan adalah *preprocessing*. Tahap ini bertujuan untuk mengidentifikasi daerah iris dari citra mata dengan mendeteksi batas luar dan batas dalam iris.



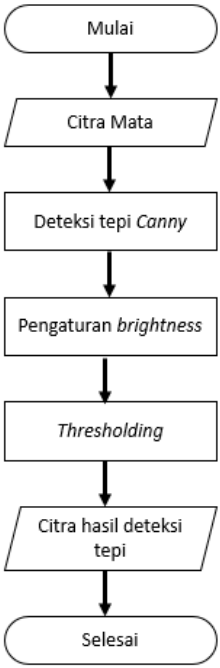
Gambar 3. 4 Diagram Alir *Preprocessing* [2]

Preprocessing dilakukan melalui tiga tahap yaitu deteksi tepi, deteksi batas pupil dan iris, dan pemisahan bulu mata dan kelopak mata. Diagram alir tahap ini ditunjukkan pada Gambar 3.4. Berdasarkan diagram alir di atas, tahap awal dari *preprocessing* adalah deteksi tepi citra mata. Kemudian dilakukan pendeteksian batas antara pupil dan iris untuk mendapatkan citra iris. Setelah itu dilakukan

pemisahan pada bagian bulu dan kelopak mata yang merupakan *noise* pada daerah iris tersebut. Hasil akhir dari tahap ini adalah citra iris yang sudah tidak ada *noise* yang akan digunakan dalam proses normalisasi pada tahap selanjutnya. Masing – masing tahap akan dijelaskan lebih lanjut pada subbab ini.

3.4.1. Deteksi Tepi

Tahap deteksi tepi adalah tahap untuk menandai tepi – tepi pada citra mata. Tepi – tepi tersebut memperlihatkan setiap bagian citra menjadi jelas. Tujuan utama tahap ini adalah memperjelas bagian lingkaran iris terlebih dahulu kemudian lingkaran pupil. Proses deteksi tepi ini menggunakan pengaturan *brightness* dan *thresholding* citra sehingga tepi lingkaran menjadi jelas. Diagram alir proses deteksi tepi ditunjukkan pada Gambar 3.5.



Gambar 3. 5 Diagram Alir Deteksi Tepi [2]

Masukan dari tahap ini adalah citra mata hasil dari akuisisi. Proses yang dilakukan pertama kali pada data masukan adalah mendeteksi tepi. Deteksi tepi yang digunakan adalah deteksi tepi *Canny*. Di dalam deteksi tepi *Canny*, juga dilakukan proses pengkontrasan dan *thresholding.* Hasil keluaran dari tahap ini adalah citra biner yang menandakan tepi – tepi pada citra masukan.

3.4.2. Deteksi Batas Pupil dan Iris

Pada tahapan ini dilakukan beberapa proses untuk mendapatkan baras lingkaran iris dan pupil. Masukan dari proses ini adalah citra hasil dari deteksi tepi. Untuk mendapatkan lingkaran iris dan lingkaran pupil, maka dilakukan dengan metode transformasi *Hough* pada citra masukan. Di tahap ini, transformasi *Hough* dilakukan dua kali. Transformasi Hough yang pertama digunakan untuk mendapatkan lingkaran iris dan menghasilkan *Region of Interest* (ROI) iris. Selanjutnya dilakukan transformasi Hough yang kedua pada ROI iris untuk mendapatkan lingkaran pupil.

Untuk mendapatkan titik pusat lingkaran iris, pada transformasi Hough membutuhkan parameter radius terlebih dahulu. Oleh karena itu dilakukan iterasi untuk radius yang tepat bagi *database* citra mata CASIA. Radius lingkaran iris minimal dan maksimal yang ditentukan yaitu 80 piksel dan 150 piksel, sehingga transformasi ini dilakukan secara iterasi dari radius 80 piksel sampai dengan 150 piksel. Jadi terdaapat 70 buah radius pada bidang akumulator. Titik pusat lingkaran iris didapatkan dari nilai intensitas maksimal pada tiap bidang akumulator. Radius yang dipilih adalah radius yang berkaitan dengan titik pusat yang ditemukan. Hasil output dari transformasi Hough yang pertama adalah titik pusat lingkaran iris dan radius iris.

Gambar 3. 6 Diagram Alir Deteksi Batas Pupil Dan Iris [2]

Setelah melakukan transformasi hough pada lingkaran iris, selanjutnya dilakukan kembali transformasi yang sama terhadap lingkaran pupil dengan menggunakan hasil dari transformasi pertama. Dari hasil transformasi pertama ini bias ditentukan ROI iris. Setelah itu transformasi Hough kedua dilakukan pada ROI iris untuk mendeteksi lingkaran pupil. Pada transformasi ini ditentukan radius untuk pupil yaitu 28 piksel sampai dengan 75 piksel sebagai batas bawah dan batas atas secara berturut–turut. Hasil akhir dari transformasi kedua ini adalah radius dan titik pusat pupil.

Proses untuk mendapatkan batas lingkaran iris dan pupil ini dilakukan dengan menggunakan metode yang sama, tetapi

menggunakan ukuran citra yang berbeda. Diagram alir deteksi batas dan iris ditunjukkan pada Gambar 3.6. Hasil dari keseluruhan proses ini menghasilkan titik pusat dan radius pada masing – masing pupil dan iris yang nantinya akan diproses pada tahap berikutnya yaitu normalisasi iris.

3.4.3. Pemisahan Bulu dan Kelopak Mata

Citra mata yang telah diidentifikasi bagian irisnya masih mengandung bulu dan kelopak mata. Keduanya dianggap sebagai *noise*. Oleh karena itu perlu dilakukan penghilangan bulu dan kelopak mata yang menghalangi bagian iris.

Untuk memisahkan bagian kelopak mata, caranya dengan membuat sebuah garis pada bagian atas dan bagian bawah kelopak mata dengan transformasi Hough linear. Selanjutnya digambar garis horizontal yang berpotongan dengan garis pertama pada tepi iris yang paling dekat dengan pupil. Garis horizontal kedua memungkinkan pemisahan maksimal dari daerah kelopak mata. Intensitas dari batas garis horizontal ini ditarik garis sampai ujung cita diganti dengan notasi NaN. Notasi ini digunakan untuk visualisasi *noise* yang ditandai.

Pada pemisahan bulu mata, cara yang digunakan adalah operasi *thresholding*. Pada citra iris terdapat bulu mata yang menutupi bagian iris. Thresholding ini dilakukan pada piksel yang mempunyai intensitas yang hampir serupa dengan bulu mata tersebut. Sehingga saat thresholsing dilakukan, semua piksel yang nilainya di bawah *threshold* akan berubah. Perubahan piksel ini digantikan dengan notasi NaN. Nantinya piksel dengan notasi NaN ini akan diganti nilainya pada tahap normalisasi iris.

Hasil keluaran proses ini merupakan citra yang sudah dapat di normalisasikan dan *noise* sudah berhasil ditandai. Gambar 3.7 menunjukkan diagram alir proses pemisahan bulu dan kelopak mata.

Gambar 3. 7 Diagram Alir Pemisahan Bulu dan Kelopak Mata [2]

3.5. Perancangan Normalisasi Iris

Citra iris yang sudah didapat dari hasil *preprocessing* dilakukan tahap normalisasi. Citra iris yang berbentuk lingkaran sulit untuk dilakukan verifikasi karena mempunyai dimensi berbeda–beda, terutama pada radiusnya. Luasan iris mata juga berubah – ubah seiring waktu diakibatkan banyak faktor, salah satunya yaitu pencahayaan yang mengakibatkan pupil mata melebar. Dengan normalisasi, citra iris direpresentasikan lebih baik. Diagram alir tahap normalisasi ditunjukkan pada Gambar 3.8.

Gambar 3. 8 Diagram Alir Normalisasi Iris Mata [2]

Citra iris yang mempunyai radius dan titik pusat pupil dan iris yang telah ditemukan pada tahap *preprocessing* digunakan untuk tahap normalisasi. Tahap normalisasi dilakukan pada daerah lingkar iris dari koordinat kartesian ke koordinat polar menggunakan transformasi polar seperti pada subbab 2.5, yang di dalamnya terdapat persamaan untuk mengubah bidang daerah lingkar iris dari bentuk lingkaran menjadi bentuk persegi Panjang.

Hasil keluaran dari tahap ini adalah citra iris yang telah dinormalisasi menjadi berbentuk persegi panjang dengan dimensi 20 x 240 piksel. Piksel – piksel pada citra normalisasi yang bernotasi NaN digantikan dengan rata–rata nilai piksel citra normalisasi.

3.6. Ekualisasi Histogram dan Binerisasi

Sebelum memasuki proses ekstraksi fitur, maka citra hasil normalisasi akan dilakukan *histogram equalization*. Tahap ini bertujuan untuk menyesuaikan kontras pada gambar. Melalui penyesuaian kontras ini, intensitas dapat terdistribusi lebih baik.

Gambar 3. 9 Diagram Alir Ekualisasi Histogram dan Binerisasi

Citra hasil *histrogram equalization* akan digunakan sebagai masukan pada ekstraksi fitur *Average Absolute Deviation*. Sedangkan masukan untuk ekstraksi fitur *Gray-Level Co-Occurrence Matrix* merupakan citra biner yang berasal dari citra hasil *histogram equalization.* Citra biner yang memiliki nilai batas derajat keabuan lebih besar dari nilai batas akan diberi nilai 1 (putih) dan sebaliknya piksel dengan derajat keabuan lebih kecil dari nilai batas akan diberi nilai 0 (hitam). Diagram alir tahap ekualisasi histogram dan binerisasi ditunjukkan pada gambar 3.9.

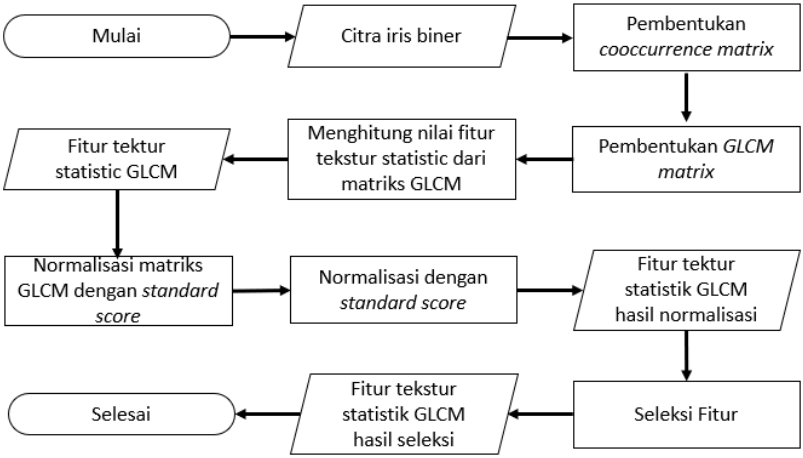
3.7. Perancangan Ekstraksi Fitur

Ekstraksi fitur iris merupakan ciri khas penting pada citra iris ternormalisasi. Masukannya adalah hasil dari tahap normalisasi yang sudah melalui tahap ekualisasi histogram dan binerisasi. Terdapat dua metode ekstraksi fitur yang akan digunakan, yaitu *Gray-Level Co-Occurrence Matrix* (GLCM) dan *Average Absolute Deviation* (AAD). Ekstraksi fitur yang dipakai AAD adalah jumlah selisih nilai piksel dengan rata – rata piksel citra dibagi dengan jumlah piksel yang diterapkan pada tiap *sub-image*. Sedangkan ekstraksi fitur yang dipakai GLCM menghasilkan nilai–nilai statistik yang telah dijelaskan pada subbab 2.8. Masing–masing dari ekstraksi fitur tersebut akan menghasilkan vektor fitur yang akan digunakan untuk melakukan tahap klasifikasi.

3.7.1. Ekstraksi Fitur Gray-Level Co-Occurrence Matrix

Data masukan yang digunakan dalam metode ekstraksi fitur *Gray-Level Co-Occurrence Matrix* (GLCM) adalah citra ternormalisasi yang sudah diekualisasi histogram dan dibinerisasi. Langkah awal dari proses ekstraksi fitur ini adalah membuat matriks kookuren. Ukuran matriks sesuai dengan rentang keabuan citra iris. Dalam hal ini ukuran matriks kookuren adalah 2 x 2 karena rentang derajat keabuan 0 – 1 akibat proses binerisasi. Kemudian matriks kookuren *(i,j)* diisi dengan jumlah piksel *i* yang mempunyai tetangga *j* pada citra iris dengan jarak *d* dan derajat θ seperti yang telah dijelaskan pada subbab 2.8. Pada tugas akhir ini ditentukan jarak yang digunakan adalah d=1 piksel dan $\theta = 0°$.

Setelah mendapatkan matriks kookuren, maka dilakukan penjumlahan antara matriks kookuren dengan matriks transposenya untuk mendapatkan matriks GLCM. Kemudian dilakukan normalisasi terhadap matriks GLCM, sehingga didapat matriks GLCM yang telah ternormalisasi. Matriks GLCM ternormalisasi inilah yang nantinya akan dihitung nilai – nilai fitur statistiknya sesuai dengan yang telah dijelaskan pada subbab 2.8. Nilai – nilai fitur yang didapat akan digunakan dalam tahap klasifikasi. Diagram alir ekstraksi fitur GLCM ditunjukkan pada Gambar 3.10. Setelah mendapatkan fiturnya, maka seleksi fiitur dilakukan untuk mengetahui fitur terbaik untuk dipilih dalam tahap pengujian.

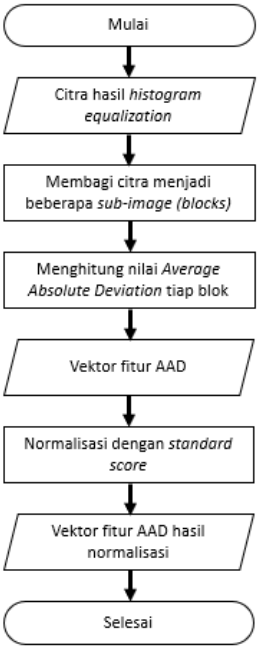


Gambar 3. 10 Diagram Alir Ekstraksi Fitur GLCM

3.7.2. Ekstraksi Fitur Average Absolute Deviation (AAD)

Pada tahap ekstraksi fitur ini, maka langkah awal yang dilakukan adalah membagi citra menjadi beberapa blok *(sub-image*). Masukan untuk ekstraksi fitur AAD adalah citra ternormalisasi berukuran 20 x 240 yang sudah dilakukan ekualisasi histogram. Perhitungan fitur AAD telah dijelaskan pada subbab 2.7. Perhitungan ini diterapkan pada setiap blok dari citra. Misalnya citra iris dibagi menjadi 12 blok, maka akan dilakukan perhitungan sebanyak 12 kali.

Dengan kata lain keluaran ekstraksi fitur AAD akan menghasilkan vektor fitur sebanyak 12 fitur yang akan digunakan untuk tahap klasifikasi. Diagram alir ekstraksi fitur AAD ditunjukkan pada Gambar 3.11.



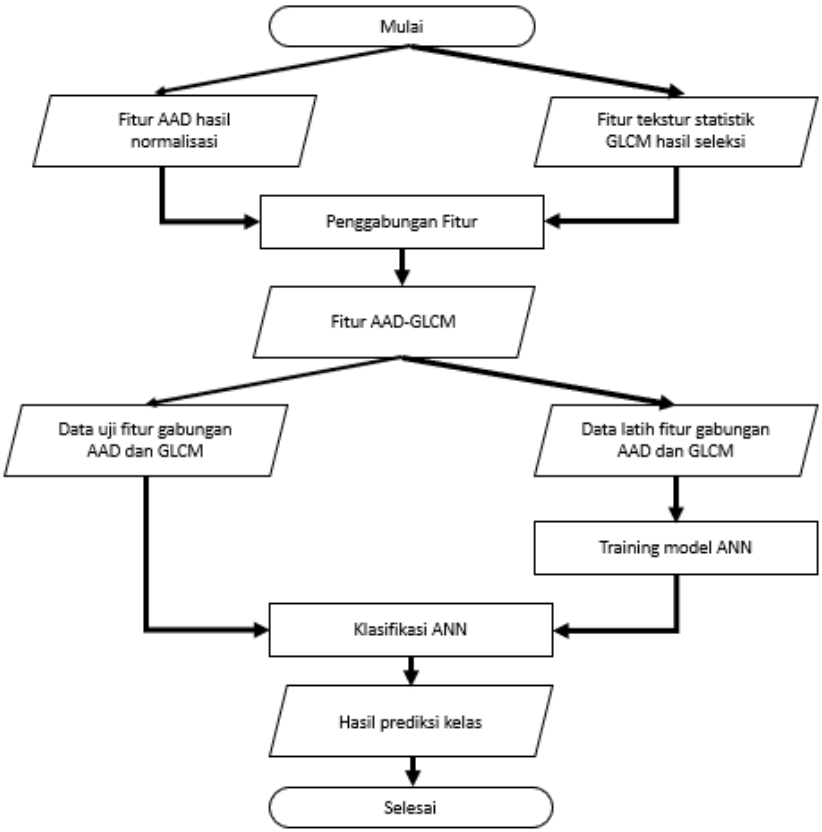
Gambar 3. 11 Diagram Alir Ekstraksi Fitur AAD

3.8. Perancangan Klasifikasi

Klasifikasi dilakukan dengan mengolah data hasil dari proses ekstraksi fitur. Setelah melalui tahap ekstraksi fitur, citra iris mata direpresentasikan dalam bentuk vektor fitur. Untuk setiap data latih dan data uji akan dijadikan sebagai vektor fitur. Kemudian dilakukan tahap verifikasi menggunakan metode klasifikasi *Artificial Neural Network* (ANN).

Tahap ini terlebih dahulu membuat database vektor fitur seluruh citra input dan disimpan dalam sebuah file berekstensi .csv dengan menggunakan Matlab. Jika ada citra masukan yang ingin dikenali

irisnya termasuk pada kelas mana, maka akan dilakukan klasifikasi menggunakan menggunakan klasifikasi ANN. Masukan dari tahap klasifikasi adalah vektor fitur citra iris yang sudah dijelaskan sebelumnya. Hasil akhir dari proses ini adalah prediksi kelas dari hasil tahap klasifikasi.



Gambar 3. 12 Diagram Alir Proses Klasifikasi

Fitur yang dipakai untuk metode klasifikasi ANN kali ini adalah gabungan hasil ekstraksi fitur *Gray-Level Co-Occurrence Matrix* dan

Average Absolute Deviation. Cara penggabungan kedua fitur adalah dengan menyimpan fitur-fitur AAD dan fitur-fitur GLCM yang telah didapatkan dalam ekstraksi fitur masing-masing dalam sebuah file berekstensi .csv. Dataset yang telah disimpan tersebut akan dibagi menjadi data uji dan data latih. Setelah itu akan dilakukan klasifikasi ANN dengan menentukan parameter–parameternya. Diagram alir proses ini ditunjukkan pada Gambar 3.12.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi ini menjelaskan mengenai proses pengenalan iris mata dengan menampilkan kode sumber yang digunakan.

4.1. Lingkungan implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan ditampilkan pada **Error! Reference source not found.**.

Tabel 4. 1 Lingkungan implementasi Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel ® Core i5-7200U CPU @ 2.50GHz (4 CPUs), ~2.7GHz Memori: 8192MB RAM
Perangkat lunak	Sistem Operasi: Windows 10 Pro 64-bit Perangkat Pengembang: Matlab, Python Perangkat Pembantu: Sublime

4.2. Implementasi

Subbab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

4.2.1. Implementasi Akuisisi Citra Mata

Tahap paling awal yang dilakukan pada proses pengenalan iris adalah akuisisi citra. Akuisisi citra merupakan proses untuk menyiapkan dan mengambil data citra mata dari *disk*. Berikut adalah implementasi akuisisi citra mata yang ditunjukkan oleh Kode Sumber 4.1.

```
1. image_folder = '../../iris108/';
2. filenames = dir(fullfile(image_folder,
   '*.bmp'));
3. total_object = numel(filenames);
4. for n = 1:total_object
5.     name = filenames(n).name;
6.     full_name= fullfile(image_folder, name);
7. % code ekstraksi fitur
8. % . . . . .
9. end
```

Kode Sumber 4. 1 Implementasi Tahap Akuisisi Citra

Baris ke-1 sampai ke- 2 membaca dataset yang terletak dalam folder iris108. Pada baris ke-3, terdapat variabel `total_object` yang digunakan untuk menyimpan jumlah dataset dalam folder tersebut. Baris ke-4 sampai ke-9 melakukan proses ekstraksi fitur pada setiap citra iris mata yang akan dijelaskan pada subbab tentang implementasi ekstraksi fitur.

4.2.2. Implementasi *Preprocessing*

Setelah citra mata dapat dibaca di dalam sistem, maka langah berikutnya adalah melakukan tahap *preprocessing*. Tujuan utama tahap ini adalah mengidentifikasi bagian iris pada citra mata. Tahap ini terdiri dari tiga tahap, yaitu deteksi tepi, deteksi batas pupil dan iris, serta pemisahan bulu dan kelopak mata. Masing – masing tahap akan dijelaskan pada subbab berikutnya.

4.2.2.1. Implementasi Deteksi Tepi

Tahap ini adalah langkah paling awal yang dilakukan pada tahap *preprocessing*. Deteksi tepi dilakukan untuk mengetahui tepi lingkaran pupil dan iris yang terdapat pada citra mata. Implementasi deteksi tepi pada citra mata ditunjukkan oleh Kode Sumber 4.2.

```
1.  [I2 or] = canny(image, sigma, scaling, ...
    vert, horz);
2. I3 = adjgamma(I2, 1.9);
3. I4 = nonmaxsup(I3, or, 1.5);
4. edgeimage = hysthresh(I4, hithres,lowthres);
```

Kode Sumber 4. 2 Implementasi Tahap Deteksi Tepi

Kode sumber di atas merupakan deteksi tepi dengan manggunakan metode *Canny* yang telah dimodifikasi. Pada baris 1, parameter `image` adalah citra mata yang hendak dilakukan deteksi tepi. Sedangkan `sigma` merupakan standar deviasi filter Gaussian untuk dihaluskan. Parameter `scalling` digunakan untuk mempercepat proses. Kemudian dua parameter terakhir merupakan bobot untuk gradien deteksi tepi pada arah vertikal dan horizontal. Pada baris kedua sampai ketiga melakukan pengaturan kontras dan perampingan pada garis deteksi tepi. Kemudian pada baris terakhir merupakan proses *threshold* pada citra dengan menggunakan dua buah *threshold* atas (`hithres`) dan *threshold* bawah (`lowthres`).

4.2.2.2. Implementasi Deteksi Batas Pupil dan Iris

Subbab ini membahas lanjutan implementasi tahap *preprocessing* setelah melakukan deteksi tepi. Tahap setelah mendapatkan garis tepi adalah melakukan Transformasi Hough. Transformasi Hough digunakan pada garis tepi lingkaran iris terlebih dulu dan selanjutnya dilakukan pada lingkungan pupil seperti yang ditunjukkan pada Kode Sumber 4.3 dan Kode 4.4 secara berturut – turut dengan radius yang sudah diinisialisasi pada baris kedua sampai keempat.

```
1.  % menentukan radius pupil & iris
2.  lpupilradius = 28;
3.  upupilradius = 75;
4.  lirisradius = 80;
5.  uirisradius = 150;
6.  % transformasi Hough pd iris
7.  scaling = 0.4;
```

```
8.  [row, col, r] = findcircle(eyeimage, ...
9.  lirisradius, uirisradius, scaling,...
10. 2, 0.20, 0.19, 1.00, 0.00);
11. circleiris = [row col r];
12. % menentukan panjang&lebar ROI iris
13. irl = double(round(rowd-rd));
14. iru = double(round(rowd+rd));
15. icl = double(round(cold-rd));
16. icu = double(round(cold+rd));
```

Kode Sumber 4. 3 Implementasi Transformasi Hough pada Iris

Kode Sumber 4.3 menunjukkan Transformasi Hough yang dilakukan pada lingkaran iris yang sudah terdeteksi tepinya. Parameter `scaling` digunakan untuk mengubah ukuran citra deteksi tepi menjadi lebih kecil agar proses Transformasi Hough lebih cepat. Pada baris ke- 9 sampai dengan baris ke -11 adalah transformasi Hough yang diimplementasikan dengan fungsi `findcircle`. Hasil dari transformasi Hough dapat terlihat pada baris ke- 12. Kemudian menentukan *region of interest* iris untuk digunakan dalam pencarian lingkaran pupil.

Pendeteksian lingkaran pupil dilakukan dengan masukannya yaitu ROI dari iris seperti yang terlihat pada Kode Sumber 4,4 pada baris ke-2. Selanjutnya dilakukan transformasi Hough pada ROI tersebut, sehingga menghasilkan radius dan titik pusat pupil. Titik pusat pupil perlu dijumlahkan dengan Panjang dan lebar ROI iris untuk mendapatkan titik pusat yang tepat. Hasil keluaran dari transformasi Hough ini terlihat pada baris ke- 15, yaitu radius dan titik pusat pupil.

```
1.  % mengambil ROI iris
2.  imagepupil = eyeimage( irl:iru,icl:icu);
3.
4.  % transformasi Hough pd pupil
5.  [rowp, colp, r] = findcircle(imagepupil, ...
6.  lpupilradius, upupilradius, scaling,...
```

```
7.  0.6, 2, 0.25, 0.25, 1.00, 1.00);
8.
9.  % penjumlahan dgn titik pusat untuk
10. % mendapatkan posisi yang tepat
11. row = round(irl + rowp);
12. col = round(icl + colp);
13. % radius dan titik pusat pupil
14. circlepupil = [row col r];
```

Kode Sumber 4. 4 Implementasi Transformasi Hough pada Pupil

Hasil akhir dari tahap deteksi batas pupil dan iris ini adalah radius iris, titik pusat iris, radius pupil dan titik pusat pupil.

4.2.2.3. Implementasi Pemisahan Bulu dan Kelopak Mata

Pemisahan bulu dan kelopak mata dilakukan karena keduanya merupakan *noise*. Langkah awal ada menghilangkan kelopak mata. Inplementasinya ditunjukkan pada Kode Sumber 4.5.

Implementasi dilakukan dengan mengkonversi citra mata ke tipe data *double* agar mudah diolah. Selanjutnya, dibuat garis pada bulu mata dengan transformasi Hough linear pada ujung bulu mata yang paling atas. Langkah ini diimplementasikan pada baris 5 sampai 11. Setelah itu pada baris ke -13 dipilih garis (y) yang maksimal dan dari batas ini ditarik garis horizontal. Dari garis horizontal yang telah didapatkan ini, akan ditandai dengan NaN samapi ujung citra karena dianggap *noise*.

```
1.  % citra mata
2.  imagewithnoise = double(eyeimage);
3.
4.  % membuat garis pd bulu mata dgn hough
5.  % linear
6.  topeyelid = imagepupil(1:(rowp-r),:);
7.  lines = findline(topeyelid);
8.  if size(lines,1) > 0
```

```
9.      [xl yl] =
    linecoords(lines,size(topeyelid));
10.     yl = double(yl) + irl-1;
11.     xl = double(xl) + icl-1;
12.     % menentukan batas garis
13.     yla = max(yl);
14.     y2 = 1:yla;
15.     % menandai kelopak(noise) dengan NaN
16.     imagewithnoise(y2, xl) = NaN;
17. end
```

Kode Sumber 4. 5 Implementasi Menghilangkan Kelopak Mata

Hal yang kedua adalah menghilangkan bulu mata dengan operasi *thresholding*. Implementasinya ditunjukkan pada Kode Sumber 4.6.

```
1.  % thresholding pd bulu mata
2.  ref = eyeimage < 100;
3.  coords = find(ref==1);
4.  imagewithnoise(coords) = NaN;
```

Kode Sumber 4. 6 Implementasi *Thresholding* Bulu Mata

Langkah ini dilakukan dengan menandai citra dengan intensitas di bawah 100 dengan notasi NaN. Bulu mata yang terdapat pada citra kebanyakan memiliki nilai intensitas di bawah 100. Proses *thresholding* pada citra mata dilakukan pada baris ke-2. Selanjutnya dicari nilai yang sesuai pada matriks citra dan dilakukan pengubahan intensitas pada baris 3 dan 4.

Hasil akhir dari tahap ini yaitu citra mata dengan bulu dan kelopak mata yang sudah digantikan intensitasnya dengan NaN karena intensitas tersebut memberikan warna hitam sebagai visualisasi.

4.2.3. Implementasi Normalisasi Iris

Subbab ini membahas implementasi tahap normalisasi iris. Tujuan utama melakukan tahap ini adalah untuk merepresentasikan

citra iris ke dalam bentuk yang lebih baik. Tahap ini perlu dilakukan karena citra mata ada yang mengalami pergeseran ataupun perbesaran pupil pada saat pengambilan gambar dari *database*. Sehingga pada saat dilakukan tes terhadap citra-citra tersebut, maka dapat mempengaruhi proses maupun hasilnya. Masukan dari tahap ini adalah citra mata hasil *preprocessing* yang daerah irisnya sudah teridentifikasi dan *noise* masih dinotasikan dengan NaN.

Implementasi tahap normalisasi iris ini menggunakan transformasi polar. Pada dasarnya transformasi polar mengubah citra iris yang berada dalam bidang kartesian ke dalam bidang polar. Implementasi transformasi polar ditunjukkan pada Kode Sumber 4.7.

Namun citra iris yang sudah diubah pada bidang polar masih memiliki *noise*, sehingga perlu dilakukan pengubahan intensitas pada piksel *noise* tersebut. Pengubahan nilai intensitas ditunjukkan pada baris ke-18 sampai ke-23, yaitu dengan mengganti niali piksel *noise* dengan nilai rata-rata pada citra iris polar. Sehingga pada tahap selanjutnya semua nilai piksel *noise* sudah mempunyai nilai. Variabel `polar_array` marupakan variable yang berisi hasil dari tahap normalisasi citra mata dengan *noise* yang sudah diganti nilai intensitasnya, sehingga bias dilanjutkan ke tahap ekstraksi fitur.

```
1.  % setting r dan theta pada citra polar
2.  [theta,r]=meshgrid(linspace(0,2*pi,512),...
3.  linspace(0,1,64));
4.
5.  % batas lingkaran pd lingkaran dalam (pupil)
6.  xp = PupilCenterX + PupilR*cos(theta);
7.  yp = PupilCenterY + PupilR*sin(theta);
8.  % batas lingkaran pada lingkaran luar (iris)
9.  xi = PupilCenterX + IrisR*cos(theta);
10. yi = PupilCenterY + IrisR*sin(theta);
11.
12. % mapping ke koordinat polar
13. x = (1-r).*xp + r.*xi;
```

```
14. y = (1-r).*yp + r.*yi;
15. polar_array = interp2(double(citra),x,y);
16. % ganti nilai NaN dgn rata2 piksel citra
17. coords = find(isnan(polar_array));
18. polar_array2 = polar_array;
19. polar_array2(coords) = 0.5;
20. avg = sum(sum(polar_array2)) / ...
21. (size(polar_array,1)*size(polar_array,2))
22. polar_array(coords) = avg;
```

Kode Sumber 4. 7 Implementasi Normalisasi Iris

Hasil akhir dari tahap normalisasi iris ini adalah citra yang berbentuk persegi Panjang dengan ukuran 20 x 240 piksel. Di dalamnya adalah tekstur iris yang sudah melalui tahap-tahap sebelumnya.

4.2.4. Implementasi Ekstraksi Fitur

Ekstraksi fitur merupakan tahap untuk pengambilan ciri dari citra iris normalisasi. Dalam tahap ini, digunakan dua buah metode ekstraksi fitur, yaitu *Average Absolute Deviation* dan *Gray-Level Co-Occurrence Matrix.*

4.2.4.1. Implementasi Ekstraksi Fitur Gray-Level Co-Occurrence Matrix

Pada subbab ini akan dijelaskan implementasi ekstraksi fitur *Gray-Level Co-occurrence Matrix*. Masukan yang digunakan dalam proses ekstraksi fitur ini adalah citra iris ternormalisasi yang sudah di *histogram equalization* dan dibinerisasi, seperti yang ditunjukkan pada baris ke-2 dan ke-4. Setelah itu, pada baris ke-7 dan ke-8 inisialisasi jarak (d) dan derajat (θ) untuk menentukan ketetanggaan antar piksel dalam citra iris. Setelah itu pembuatan *matriks cooccurrence* yang ditunjukkan pada baris ke-9 sampai ke-38. Dalam proses pembuatannya disesuaikan dengan jarak dan derajatnya. Pada baris ke-16 sampai baris ke-25 adalah pembuatan matrix cooccurrence jika derajatnya adalah 0°.

Setelah matriks kookuren dibuat, maka langkah selanjutnya adalah membuat matrik GLCM, yaitu dengan menjumlahan matriks kookuren dengan matriks *transpose*-nya. Proses penjumlahan ini ditunjukkan pada baris ke-39. Langkah terakhir agar matriks GLCM dapat diekstrak fiturnya adalah melakukan normalisasi seperti pada baris ke-41 sampai ke-45.

```
1. %histogram equalization
2. citrairis = histeq(citrairis);
3.  %mengubah ke citra biner
4.  citrairis = imbinarize(citrairis);
5.  [r,c] = size(citrairis);
6.  max_citrairis = (max(max(citrairis)));
7.  d = 1;
8.  tetha = 0;
9. matrix_cooccurrence =
   zeros(max_citrairis+1,max_citrairis+1);
10.   for i=1:max_citrairis+1
11.     for j=1:max_citrairis+1
12.         [row, column] = find(citrairis==i-1);
13.         long = length(column);
14.         temp = 0;
15.         for k=1:long
16.           if (tetha == 0) && (column(k) <= c)
17.             index_row = row(k);
18.             index_column = column(k) + d;
19.             if(index_row <= r && index_column
   <= c && index_row > 0 && index_column > 0)
20.                value=
                 citrairis(index_row,index_column);
21.                if value == j-1
22.                   temp = temp + 1;
23.                end
24.             end
```

```
25.          end
26.          if (tetha == 45) && (column(k) <= c)
27.          ...
28.          end
29.          if (tetha == 90) && (column(k) <= c)
30.          ...
31.          end
32.          if (tetha == 135) && (column(k) <= c)
33.          ...
34.          end
35.    matrix_cooccurrence(i,j) = temp;
36.       end
37.    end
38. end
39. glcm = matrix_cooccurrence +
    matrix_cooccurrence';
40. %normalisasi glcm
41. for p=1:h
42.       for q=1:w
43.           glcm (p, q) = glcm(p, q)/sum_glcm;
44.       end
45. end
```

Kode Sumber 4. 8 Implementasi Ekstraksi Fitur GLCM

Setelah mendapatkan matriks GLCM yang sudah dinormalisasi, maka dilakukan perhitungan statistik untuk mendapatkan fiturnya. Perhitungan dilakukan pada setiap *cell* dalam matriks GLCM seperti yang terlihat pada baris ke-1 sampai baris ke-19. Hasil perhitungan ini disimpan dalam matriks sesuai dengan nama fitur masing-masing. Kemudian, matriks yang berisi hasil perhitungan masing-masing *cell* dijumlahkan sehingga menghasilkan satu nilai fitur untuk setiap matriksnya. Proses ini ditunjukkan pada baris ke-20 sampai ke-30. Hasil nilai – nilai fitur ini akan dinormalisasi seperti yang ditunjukkan dalam baris ke-31 sampai ke-35.

```
1. for p=1:h
2.     for q=1:w
3.       matrix_kontras(p,q) = abs(p-
   q)^2*glcm(p,q);
4.       matrix_homogenitas(p,q) = glcm(p,q)/(1+
   abs(p-q));
5.       matrix_energi(p,q) = glcm(p,q).^2;
6.       if (glcm(p,q)~= 0)
7.         matrix_entropi(p,q) =
   glcm(p,q)*log2(glcm(p,q));
8.       else
9.         matrix_entropi(p,q) = 0;
10.      end
11.   matrix_autokorelasi(p,q) = p*q*glcm(p,q);
12.   matrix_diss(p,q) = glcm(p,q)*(abs(p-q));
13.   matrix_variance_x(p,q) = ((p -
   mean2(glcm))^2)*glcm(p,q);
14.   matrix_variance_y(p,q) = ((q -
   mean2(glcm))^2)*glcm(p,q);
15.   matrix_korelasi(p,q)=glcm(p,q)*(((p-ux)*(q-
   uy))/(ox*oy));
16.   matrix_prom(p,q) = ((p + q - ux -
   uy)^4)*glcm(p,q);
17.   matrix_shade(p,q) = ((p + q - ux -
   uy)^3)*glcm(p,q);
18.    end
19.   end
20.   kontras = sum(sum(matrix_kontras));
21.   homogenitas = sum(sum(matrix_homogenitas));
22.   energi =  sum(sum(matrix_energi));
23.   entropi =  -(sum(sum(matrix_entropi)));
24.   autokorelasi = sum(sum(matrix_autokorelasi));
25.   diss = sum(sum(matrix_diss));
```

```
26.    variance_x = sum(sum(matrix_variance_x));
27.    variance_y = sum(sum(matrix_variance_y));
28.    korelasi =  sum(sum(matrix_korelasi));
29.    prom = sum(sum(matrix_prom));
30.    shade = sum(sum(matrix_shade));
31. for j=1: length(fitur(1,:))
32.      for i=1:length(fitur(:,j))
33.          fitur(i,j) = (fitur(i,j) -
   mean2(fitur(:, j)))/ std2(fitur(:,j));
34.      end
35. end
```

Kode Sumber 4. 9 Implementasi Perhitungan dan Normalisasi Fitur GLCM

Sebelum memasuki tahap klasifikasi, dilakukan seleksi fitur. Hasil ekestraksi fitur sebelumnya disimpan di file .csv. Kemudian file .csv tersebut dibaca pada python. Setelah itu, pada baris ke-6 sampai ke-9 merupakan proses seleksi fitur dengan menggunakan SelectKBest yang tersedia pada *scikit-learn python*. Variabel selection akan merupakan inisialisas parameter, seperti fungsi *score* (`score_func`) yang digunakan dalam pemberian skor tiap fitur dan jumlah fitur terbaik yang akan diseleksi (`k`). Dalam hal ini fungsi skor yang digunakan adalah `f_classif` dan jumlah fitur yang diseleksi sebanyak 9 fitur.

```
1. data= pd.read_csv("aad_20_block-
   glcm12_d_1_0.csv", header=None).values
2.
3. X=data[1:,0:24]
4. y=data[1:,-1]
5.
6. selection = SelectKBest(score_func=f_classif,
   k=9)
7. fit = selection.fit(X,y)
```

```
8. np.set_printoptions(precision=3)
9. features = fit.transform(X)
```

Kode Sumber 4. 10 Implementasi Seleksi Fitur GLCM

4.2.4.2. Implementasi Ekstraksi Fitur *Average Absolute Deviation*

Masukan yang digunakan untuk melakukan ekstraksi fitur adalah citra iris mata ternormalisasi setelah dilakukan *histogram equalization* yang ditunjukkan pada baris ke-2. Setelah itu melakukan pembagian image menjadi beberapa blok atau *sub-image* yang ditunjukkan pada baris ke-5 sampai baris ke-14. Kemudian dilakukan perhitungan AAD pada setiap blok. Perhitungan ini ditunjukkan pada baris ke-18 sampai baris ke-22.

```
1.  %histogram equalization
2.  citrairis = histeq(citrairis);
3.
4.  %membagi image ke dalam beberapa sub-image
5.  Blocks =
    cell(row/size_block,column/size_block);
6.  counti = 0;
7.  for i = 1:size_block : row -(size_block -1)
8.      counti = counti + 1;
9.      countj = 0;
10.     for j = 1:size_block : column -(size_block
    -1)
11.         countj = countj + 1;
12.         Blocks{counti,countj}=
    citrairis(i:i+(size_block -1),j:j+(size_block
    -1));
13.     end
14. end
15. %menghitung AAD
16. sum_blocks=length(Blocks)*row/size_block;
17. num_pixels = size_block*size_block;
18. for k = 1:sum_blocks
```

```
19.     temp = abs(Blocks{k}-mean2(Blocks{k}));
20. sum_block = sum(sum(temp));
21. features(n+1,k)=num2cell(1/num_pixels*
    sum_block);
22. end
```

Kode Sumber 4. 11 Implementasi Ekstraksi Fitur AAD

Setelah mendapatkan fitur AAD pada masing-masing blok *image*, maka pada baris ke-24 sampai ke-28 dilakukan normalisasi pada setiap nilai fitur. Setiap Nilai fitur dikurangi dengan nilai mean lalu dibagi dengan strandar deviasi yang ditunjukkan pada baris ke-26.

```
23. %normalisasi hasil ekstraksi fitur AAD
24. for j=1: length(fitur(1,:))
25.       for i=1:length(fitur(:,j))
26. fitur(i,j) = (fitur(i,j) - mean2(fitur(:,
    j)))/ std2(fitur(:,j));
27.       end
28. end
```

Kode Sumber 4. 12 Implementasi Normalisasi Fitur AAD

4.2.5. Implementasi Klasifikasi *Artificial Neural Network*

Tahap ini merupakan implementasi tahap klasifikasi dengan menggunakan metode *Artificial Neural Network*. Sebelumnya, dilakukan penggabungan fitur AAD dengan fitur GLCM yang sudah diseleksi fiturnya. Tahap ini ditunjukkan pada Kode Sumber 4.13 pada baris ke-1 dan ke-2.

```
1. gabungan =
   {[features(:,1:jumlah_fitur_aad),fitur_glcm]};
2. gabungan = gabungan{1,1};
```

Kode Sumber 4. 13 Implementasi Penggabungan Fitur AAD dan Fitur GLCM

Setelah melakukan penggabungan fitur, maka pada baris ke-1 dilakukan pembagian data latih dan data uji menggunakan

`StratifiedKFold` sesuai yang ditunjukkan pada Kode Sumber 4.14. Pada parameter `n_split` tertulis angka 7, artinya akan dilakukan pengujian sebanyak tujuh kali dengan data uji sebanyak tujuh per jumlah data tiap kelas, sedangkan data sisanya menjadi data latih. Selanjutnya masuk ke tahap implementasi klasifikasi. `MLPClassifier` merupakan fungsi klasifikasi ANN *Multilayer Perceptron* beserta parameter – parameternya. Pada klasifikasi ini, akan dihitung nilai akurasi, presisi dan f1.

```
1. skf = StratifiedKFold(n_splits=7, shuffle=True,
   random_state = None)
2. clf = MLPClassifier(solver='adam', alpha=0.02,
   hidden_layer_sizes=(216,108),
   activation='tanh', random_state=None,
   batch_size=100, learning_rate_init=0.005)
3. for train_indices, test_indices in
   skf.split(X,y):
4.      clf.fit(X[train_indices], y[train_indices])
5.      Y_pred = clf.predict(X[test_indices])
6.      presisi +=
   precision_score(y[test_indices],Y_pred,average=
   'macro')
7.      f1 += f1_score(y[test_indices], Y_pred,
   average='macro')
8.      akurasi += accuracy_score(y[test_indices],
   Y_pred)
```

Kode Sumber 4. 14 Implementasi Klasifikasi ANN

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

5.1. Lingkungan Pengujian

Lingkungan uji coba yang akan digunakan untuk pengenalan iris mata mencakup perangkat keras dan perangkat lunak. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 5.1.

Tabel 5. 1 Lingkungan Uji Coba Perangkat Keras dan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel ® Core i5-7200U CPU @ 2.50GHz (4 CPUs), ~2.7GHz Memori: 8192MB RAM
Perangkat lunak	Sistem Operasi: Windows 10 Pro 64-bit Perangkat Pengembang: Matlab, Python Perangkat Pembantu: Sublime

5.2. Data Uji Coba

Data yang digunakan dalam uji coba pengenalan iris mata bersumber dari database citra mata *Chinese Academy of Science-Institute of Automation* (CASIA) versi 1.0. Database ini berisi 756 citra mata dari 108 mata yang berbeda. Masing-masing diambil dua sesi dengan interval satu bulan antar sesinya. Sesi pertama diambil sebanyak 3 citra mata dan sesi kedua sebanyak 4 citra mata.

Contoh format nama file citra mata pada database CASIA adalah 001_1_1.bmp. Pada format file tersebut, 001 menunjukkan mata atau kelas orang pertama, angka 1 dikanannya menunjukkan citra

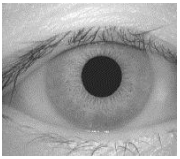
mata diambil pada sesi pertama, dan angka 1 yang terakhir menunjukkan mata yang diambil pada urutan ke-1.

Dalam melakukan pengujian di tahap klasifikasi, fitur vector citra mata dibagi menjadi data latih dan data uji. Pembagian data ini menggunakan Cross Validation yang dibagi secara rata setiap kelasnya atau sering disebut *Stratified.* Data di-*split* menjadi 7 bagian tiap kelasnya. Setiap kelas terdapat 7 data. Jumlah kelas pada *database* ini sebanyak 108 kelas/iris mata. Sehingga didapatkan setiap kelas diambil satu data sebagai data uji sedangkan sisanya sebagai data latih. Karena *split*-nya 7, maka pengujian dilakukan sebanyak 7 kali dengan data uji berjumlah 1 x 108 = 108 buah dan data latih berjumlah 6 x 108 = 648 buah.

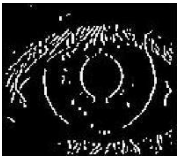
5.3. Hasil Uji Coba Setiap Tahapan

5.3.1. *Preprocessing*

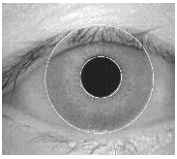
Pada tahap *preprocessing*, citra mata berukuran 320 x 280 piksel akan diolah untuk mendapatkan bagian irisnya. Contoh sampel langkah–langkah untuk mendapatkan iris mata ditunjukkan pada Gambar 5.1.



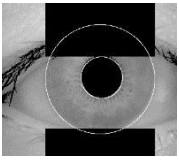
(a) Citra Mata 001_1_1.bmp



(b) Deteksi Tepi



(c) Lokalisasi Iris



(d) Pemisahan *noise*

(e) Citra Normalisasi Iris

Gambar 5. 1 Sampel hasil Preprocessing pada Citra Mata 001_1_1.bmp

Gambar 5.1 merupakan sampel citra dari tahap *preprocessing* yang terdiri dari proses deteksi tepi, deteksi batas iris dan pupil, dan proses pemisahan bulu dan kelopak mata, serta proses normalisasi iris. Gambar (a) merupakan citra mata yang digunakan sebagai masukan dalam tahap ini. Gambar (b) merupakan hasil dari proses deteksi tepi. Gambar (c) merupakan gambar citra mata yang sudah ditandai batas iris dan pupilnya. Gambar (d) adalah *noise* pada bagian dalam iris yang sudah diberi nilai NaN. Gambar (e) adalah gambar iris yang sudah dinormalisasi dan tidak mempunyai *noise* lagi.

5.3.2. Ekstraksi Fitur

Ekstraksi fitur yang dilakukan dalam uji coba ini ada 2 metode, yaitu *Average Absolute Deviation* dan *Gray-Level Co-occurrence Matrix*. Langkah – langkah untuk melakukan ekstraksi fitur pada kedua metode tersebut telah dijelaskan pada subbab 3.7. Hasil Ekstraksi fitur ini akan disimpan di dalam file .csv yang nantinya akan digunakan sebagai dataset pada proses klasifikasi.

Pada ekstraksi fitur *Average Absolute Deviation*, citra iris dibagi ke dalam beberapa blok dengan ukuran piksel tertentu sehingga menghasilkan jumlah fitur yang berbeda. Berikut merupakan tabel 5.2 hasil pembagian citra iris menjadi beberapa blok.

Tabel 5. 2 Jumlah fitur AAD beberapa ukuran blok

Ukuran Blok (piksel)	Jumlah Fitur
20 x 20	12
10 x 10	48
5 x 5	192
4 x 5	300
2 x 2	1200

Sedangkan untuk ekstraksi fitur *Gray-Level Cooccurrence Matrix*, digunakan 13 fitur statistik, yaitu kontras, homogenitas, energi, entropi, korelasi, autokorelasi, *dissimilarity*, *cluster prominence*, *cluster shade*, *variance* pada sumbu x, *variance* pada sumbu y, *standar deviasi* dan *mean.* Sebelumnya, ditentukan jarak *(d)* = 1 dan orientasi (θ) = 0° dalam pembentukan matriks GLCM

Selanjutnya semua fitur tersebut akan dilakukan seleksi fitur terbaik yang berguna dalam skenario uji coba. Berdasarkan seleksi fitur terbaik, maka pada tabel 5.3 akan ditunjukkan bobot nilai hasil seleksi fitur. Semakin tinggi bobot nilai, maka semakin bagus fitur tersebut.

Fitur – fitur yang dihasilkan oleh kedua metode ekstraksi fitur dinormalisasi menggunakan *standard score*, yaitu dengan melakukan pengurangan nilai fitur terhadap nilai mean dan dibagi dengan standar deviasi. Nilai fitur hasil normalisasi berkisar dari -1 sampai 1.

Tabel 5. 3 Bobot Nilai Fitur GLCM

No	Nama Fitur	Bobot Nilai
1	Korelasi	164.622
2	*Cluster Shade*	163.509
3	Homogenitas	147.776
4	*Cluster Prominence*	94.099
5	Autokorelasi	55.909
6	Entropi	50.969
7	*Variance* sumbu *x*	30.956
8	*Variance* sumbu *y*	30.956
9	Standar deviasi	11.942
10	*Mean*	11.55
11	Energi	10.405
12	Kontras	9.181
13	*Dissimilarity*	9.181

5.4. Skenario Uji Coba

Pada subbab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan menggunakan data latih dan data uji. Melalui

skenario ini, implementasi algoritma ekstraksi fitur pada klasifikasi ANN akan diuji bagaimana performanya pada masing-masing skenario. Telah dilakukan beberapa skenario uji coba, diantaranya yaitu :

1. Perhitungan performa hasil akurasi dan waktu eksekusi dari ekstraksi fitur GLCM menggunakan metode klasifikasi ANN dengan variasi jumlah fitur yang dipilih melalui seleksi fitur terlebih dahulu. Variasi jumlah fitur yang akan diuji yaitu, 5, 6, 7, 8, 9, 10, 11, 12, 13.
2. Perhitungan performa hasil akurasi dan waktu eksekusi dari ekstraksi fitur AAD menggunakan metode klasifikasi ANN dengan variasi ukuran blok/*subimage*. Variasi ukuran blok yang akan diuji yaitu, 20 piksel, 10 piksel, 5 piksel, 4 piksel dan 2 piksel.
3. Perhitungan performa hasil akurasi dan waktu eksekusi dari gabungan ekstraksi fitur GLCM dan ekstraksi fitur AAD yang didapatkan dari performa terbaik pada skenario 1 dan skenario 2 dengan menggunakan variasi jumlah ukuran blok, yaitu 20 piksel, 10 piksel, 5 piksel, 4 piksel dan 2 piksel.
4. Perhitungan performa hasil akurasi dan waktu eksekusi dari gabungan ekstraksi fitur GLCM dan ekstraksi fitur AAD yang didapatkan dari performa terbaik pada skenario 1 dan skenario 2 dengan menggunakan variasi *activation function*, yaitu *tanh, relu*, *identity*, dan *logistic*.
5. Perhitungan performa hasil akurasi dan waktu eksekusi dari gabungan ekstraksi fitur GLCM dan ekstraksi fitur AAD yang didapatkan dari performa terbaik pada skenario 1 dan skenario 2 menggunakan variasi optimizer, yaitu adam, lbfgs, dan sgd.
6. Perhitungan performa hasil akurasi dan waktu eksekusi dari gabungan ekstraksi fitur GLCM dan ekstraksi fitur AAD yang didapatkan dari performa terbaik pada skenario 1 dan skenario 2 menggunakan variasi hidden layer dan jumlah neuron dengan dua *hidden layer* yaitu (54,54), (54, 108), (108,54), (108,108).

Pada masing – masing skenario, pembagian data uji dan data latih menggunakan *cross validation* dengan 7 *fold*, artinya data dari

setiap kelas akan diambil 1 sebagai data uji sedangkan enam lainnya menjadi data latih. Sehingga, untuk keseluruhan data uji berjumlah 108 data dan data latih berjumlah 648 data. Setiap *fold* dilakukan perulangan sebanyak 20 kali.

Data yang digunakan adalah data hasil ekstraksi fitur GLCM dan AAD yang telah dinormalisasi sehingga nilai fiturnya berkisar dari -1 sampai 1.

5.4.1 Skenario Uji Coba 1

Skenario uji coba 1 merupakan perhitungan akurasi dan waktu eksekusi pada metode ekstraksi fitur GLCM dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi jumlah fitur, yaitu 5, 6, 7, 8, 9, 10, 11, 12, dan 13. Parameter *default* yang digunakan dalam skenario ini adalah optimizer (adam), *hidden layer* (108,108), alpha ($\alpha = 0.02$), *activation function* (*tanh*), *batch_size* (100), dan *learning_rate_init* (0.005).

Tabel 5. 4 Persentase Akurasi dan Waktu Eksekusi Masing-Masing Jumlah Fitur pada Ekstraksi Fitur GLCM

Jumlah Fitur	Akurasi (%)	Waktu Eksekusi (s)
5	62.13	184.15
6	**64.93**	**188.50**
7	64.65	165.77
8	64.16	161.40
9	64.50	162.87
10	64.21	162.87
11	63.92	153.82
12	63.27	162.86
13	62.39	164.25

Setiap jumlah fitur dipilih dengan menggunakan seleksi fitur terbaik. Hasil performa dan waktu eksekusi dapat dilihat pada Tabel 5.3. Hasil akurasi yang dihasilkan tidak terlihat cukup signifikan perbedaannya. Namun, berdasarkan hasil performa pada tabel tersebut, maka fitur yang berjumlah 6 menghasilkan nilai akurasi

tertinggi, yaitu 64.93% dengan waktu eksekusi selama 188.50 detik. Waktu eksekusi pada tiap – tiap data uji juga tidak tampak signifikan, karena waktu yang diperlukan untuk proses komputasi rata-rata hampir sama.

5.4.2. Skenario Uji Coba 2

Skenario uji coba 2 adalah perhitungan akurasi dan waktu eksekusi pada metode ekstraksi fitur AAD dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi ukuran blok, yaitu 20 piksel, 10 piksel, 5 piksel, 4 piksel, dan 2 piksel. Parameter *default* yang digunakan dalam skenario ini adalah optimizer (adam), hidden *layer* (108,108), alpha ($\alpha = 0.02$), *activation function* (*tanh*), *batch_size* (100), dan *learning_rate_init* (0.005). Hasil setiap data uji skenario ini dengan jumlah fitur 6 terdapat pada lampiran.

Tabel 5. 4 Persentase Akurasi dan Waktu Eksekusi Masing-Maasing Ukuran Blok pada Ekstraksi Fitur AAD

Ukuran Blok	Akurasi (%)	Waktu Eksekusi (s)
20 x 20	67.26	386.29
10 x 10	80.01	494.31
5 x 5	87.55	543.12
4 x 4	**88.82**	**690.78**
2 x 2	86.36	1118.40

Setiap Ukuran blok akan menghasilkan fitur yang berbeda. Jumlah fitur pada blok ukuran 20 piksel menghasilkan 12 fitur, blok ukuran 10 piksel menghasilkan 48 fitur, blok ukuran 5 piksel menghasilkan 192 fitur, blok ukuran 4 piksel menghasilkan 300 fitur dan blok ukuran 2 piksel menghasilkan 1200 fitur. Hasil performa dan waktu eksekusi masing – masing ukuran blok dapat dilihat pada Tabel 5.2.

Berdasarkan hasil performa pada table 5.2, ekstraksi fitur AAD yang menggunakan metode klasifikasi ANN dengan ukuran blok 4 piksel menghasilkan nilai akurasi tertinggi yaitu 88.82% dengan waktu eksekusi selama 690.78 detik. Selain itu, dapat diamati juga

bahwa semakin kecil ukuran blok maka semakin lama waktu eksekusinya. Hal ini terjadi karena semakin kecil ukuran blok, maka semakin banyak jumlah fiturnya, sehingga waktu eksekusinya menjadi lebih lama saat masuk ke tahap klasifikasi.

5.4.3. Skenario Uji Coba 3

Skenario uji coba 3 adalah perhitungan akurasi dan waktu eksekusi pada gabungan hasil ekstraksi fitur GLCM terbaik yang didapatkan dari skenario uji coba 1 dan hasil ekstraksi fitur AAD dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi ukuran blok pada ekstraksi fitur AAD, yaitu 20 piksel, 10 piksel, 5 piksel, 4 piksel dan 2 piksel. Parameter *default* yang digunakan dalam skenario ini adalah *activation function* (tanh) optimizer (adam), hidden *layer* (108,108), alpha ($\alpha = 0.02$), *batch_size* (100), dan *learning_rate_init* (0.005).

Tabel 5. 5 Persentase Akurasi dan Waktu Eksekusi Gabungan Fitur GLCM dan AAD dengan Variasi Ukuran Blok

Ukuran Blok	Akurasi (%)	Waktu Eksekusi (s)
20 x 20	73.96	300.62
10 x 10	81.95	445.22
5 x 5	88.20	539.50
4 x 4	**89.23**	**590.55**
2 x 2	86.42	1148.11

Berdasarkan tabel 5.5, gabungan hasil ekstraksi fitur GLCM terbaik dan hasil ekstraksi fitur AAD yang menggunakan metode klasifikasi ANN dengan ukuran blok 4 piksel menghasilkan nilai akurasi tertinggi yaitu 89.23%. Selain itu, dapat diamati juga bahwa semakin kecil ukuran blok, maka semakin lama waktu eksekusinya. Hal ini terjadi karena semakin kecil ukuran blok, maka semakin banyak jumlah fiturnya, sehingga waktu komputasinya menjadi lebih lama saat masuk ke tahap klasifikasi. Fitur hasil gabungan 6 fitur GLCM dan fitur AAD dengan ukuran blok 4 piksel akan digunakan dalam skenario uji coba 4 sampai 6. Hasil setiap data uji dengan performa terbaik terdapat pada lampiran.

5.4.4. Skenario Uji Coba 4

Skenario uji coba 4 adalah perhitungan akurasi dan waktu eksekusi pada gabungan metode ekstraksi fitur AAD dan ekstraksi fitur GLCM dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi *activation function*, yaitu *tanh, relu, identity, logistic*. Parameter *default* yang digunakan dalam skenario ini adalah *optimizer* (adam), hidden *layer* (108,108), alpha ($\alpha = 0.02$), *batch_size* (100), dan *learning_rate_init* (0.005).

Tabel 5. 6 Persentase Akurasi dan Waktu Eksekusi Gabungan Fitur AAD dan GLCM dengan Variasi *Activation Function*

Activation Function	Akurasi (%)	Waktu Eksekusi (s)
tanh	**89.23**	**590.55**
relu	78.48	414.71
identity	81.69	114.77
logistic	82.28	490.30

Berdasarkan Tabel 5.6, hasil performa yang diperlihatkan di atas, akurasi tertinggi didapatkan ketika menggunakan *activation function* tanh. Nilai akurasi tertinggi yang didapatkan yaitu sebesar 89.23 % dengan waktu eksekusi selama 590.78 detik. Sedangkan waktu eksekusi tercepat didapatkan ketika menggunakan *activation function identity* yaitu 114.77 detik. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.4.5. Skenario Uji Coba 5

Skenario uji coba 5 adalah perhitungan akurasi dan waktu eksekusi pada gabungan metode ekstraksi fitur AAD dan ekstraksi fitur GLCM dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi *optimizer*, yaitu *adam, lbfgs, sgd*. Parameter *default* yang digunakan dalam skenario ini adalah *activation function* (*tanh*), hidden *layer* (108,108), alpha ($\alpha = 0.02$), *batch_size* (100), dan *learning_rate_init* (0.005).

Tabel 5. 7 Persentase Akurasi dan Waktu Eksekusi Gabungan Fitur AAD dan GLCM dengan Variasi *Optimizer*

Optimizer	Akurasi (%)	Waktu Eksekusi (s)
adam	**89.23**	**590.55**
lbfgs	80.31	110.19
sgd	83.12	612.14

Berdasarkan Tabel 5.7, Nilai akurasi tertinggi didapatkan ketika menggunakan *optimizer* Adam dengan nilai akurasi yaitu 89.23% dan waktu eksekusi selama 590.55 detik. Waktu eksekusi tercepat didapatkan ketika menggunakan *optimizer lbfgs*. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran

5.4.6. Skenario Uji Coba 6

Skenario uji coba 6 adalah perhitungan akurasi dan waktu eksekusi pada gabungan metode ekstraksi fitur AAD dan ekstraksi fitur GLCM dengan klasifikasi ANN. Skenario dicoba dengan menggunakan beberapa variasi jumlah neuron dengan dua *hidden layer*, yaitu (54,54), (54,108), (108,54), dan (108,108). Parameter *default* yang digunakan dalam skenario ini adalah *activation function* (*tanh*), *optimizer* (adam), alpha ($\alpha = 0.02$), *batch_size* (100), dan *learning_rate_init* (0.005).

Tabel 5. 8 Persentase Akurasi Waktu Eksekusi Fitur AAD dan GLCM dengan Variasi Jumlah Neuron dengan 2 *hidden layer*

Jumlah Neuron	Akurasi (%)	Waktu Eksekusi (s)
(54,54)	82.65	379.34
(54,108)	86.24	448.20
(108,54)	86.71	484.99
(108,108)	**89.23**	**590.55**

Pada tabel 5.8, hasil performa terbaik pada uji coba 6 didapatkan ketika jumlah neuron pada 2 *hidden layer* adalah (108,108) dengan nilai akurasi sebesar 89.23%. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.5. Analisis Hasil Uji Coba

Pada subbab ini akan dilakukan analisis terhadap skenario-skenario uji coba yang dihasilkan. Skenario uji coba 1 dilakukan pada fitur hasil metode ekstraksi fitur *Gray-Level Cooccurrence Matrix* dengan menggunakan klasifikasi *Artificial Neural Network* dan Skenario uji coba 2 dilakukan pada fitur hasil metode ekstraksi fitur *Average Absolute Deviation* dengan menggunakan klasifikasi yang sama. Pada skenario uji coba 3 dilakukan penggabungan 6 fitur GLCM yang menghasilkan performa terbaik pada skenario uji coba 1 dengan fitur AAD yang didapat dari variasi ukuran blok piksel sesuai dengan skenario uji coba 2. Sedangkan uji coba 4 sampai 5 menggunakan fitur hasil performa terbaik pada skenario uji coba 3.

Pada skenario uji coba ke-1, performa yang dihasilkan tidak terlalu baik. Hasil performa yang didapatkan sebenarnya juga tidak memiliki perbedaan yang signifikan, namun hasil akurasi tertinggi terdapat pada fitur yang berjumlah 6, yaitu sebesar 64.93% dengan waktu eksekusi 188.50 detik. Uji coba GLCM dengan 6 fitur selanjutnya akan digunakan dalam skenario 3.

Selanjutnya pada skenario uji coba 2, performa terbaik dihasilkan ketika ukuran blok 4 piksel dengan nilai akurasi 88.82% dengan wkatu eksekusi 690.78 detik. Selain itu, dapat dilihat berturut-turut dari ukuran blok 20 piksel, 10 piksel, 5 piksel dan 4 piksel memberikan nilai akurasi yang semakin tinggi. Namun pada saat ukuran blok menjadi 2 piksel, nilai akurasi menjadi turun. Hal ini dapat saja terjadi karena jumlah kelas yang ada 108 dengan 7 data setiap kelasnya. Sehingga jika fitur semakin banyak, maka fitur-fitur tersebut dapat membantu memperbaiki hasil klasifikasi. Namun, jika fitur tersebur terlalu banyak, bisa jadi terdapat fitur-fitur tidak berguna atau bahkan dapat memperburuk hasil akurasi. Dapat diamati pula bahwa semakin banyak fitur, maka waktu eksekusi juga semakin lama.

Skenario uji coba ke-3 dilakukan pengujian terhadap 6 fitur GLCM dengan fitur – fitur AAD yang divariasikan pada ukuran blok piksel, yaitu 20 piksel, 10 piksel, 5 piksel, 4 piksel, dan 2 piksel. Berdasarkan hasil uji coba tersebut terbukti bahwa fitur AAD dapat meningkatkan hasil akurasi GLCM. Hasil performa terbaik terjadi

ketika GLCM 6 fitur digabungkan dengan fitur AAD ukuran blok 4 piksel. Nilai akurasi yang didapatkan yaitu sebesar 89.23% dengan waktu eksekusi selama 590.55 detik. Fitur yang mendapatkan hasil performa terbaik akan digunakan dalam skenario uji coba 4 sampai 6.

Pada skenario ke-4, uji coba dilakukan dengan menggunakan variasi *activation function*. Berdasarkan performa yang dihasilkan, fungsi aktivasi tanh memberikan hasil terbaik dengan nilai akurasi sebesar 89.23% dengan waktu eksekusi selama 590.55 detik. Namun, meskipun *activation function* tanh memberikan performa terbaik, waktu eksekusi yang dibutuhkan lebih lama dari fungsi aktivasi yang lain. Fungsi aktivasi yang membutuhkan waktu eksekusi tercepat adalah *identity*, yaitu 114.77 detik. Hal ini terjadi karena output yang dihasilkan sama dengan inputnya, sehingga tidak memerlukan waktu komputasi yang lama.

Kemudian pada skenario 5 menggunakan variasi *optimizer* atau metode optimasi. Berdasarkan subbab skenario uji coba 5, maka hasil akurasi tertinggi terjadi ketika menggunakan *optimizer* Adam, yaitu sebesar 89.23% dengan waktu eksekusi 590.55 detik. Sedangkan untuk *optimizer* yang membutuhkan waktu eksekusi tercepat adalah lbfgs, yaitu 110.19 detik.

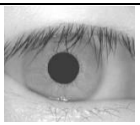
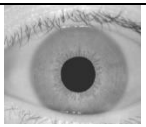
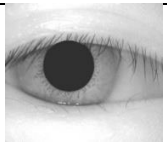
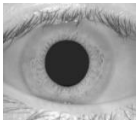
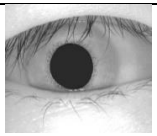
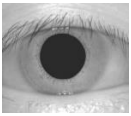
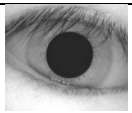
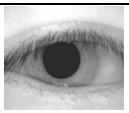
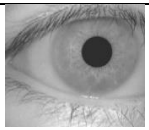
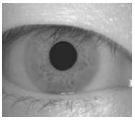
Terakhir, skenario uji coba 6 menggunakan variasi jumlah neuron dengan 2 *hidden layer*. Dapat dilihat pada tabel 5.7, performa terbaik adalah ketika jumlah neuron pada dua *hidden layer* (108,108). Akurasi yang dihasilkan adalah sebesar 89.23%. Berdasarkan uji coba juga, dapat diamati bahwa semakin banyak jumlah neuron, maka waktu eksekusi yang dibutuhkan semakin lama. Sehingga uji coba dengan jumlah neuron (108,108) memiliki waktu eksekusi terlama, yaitu 590.55 detik.

5.6. Analisis Hasil Prediksi Kelas yang Salah

Dari hasil uji coba, terdapat beberapa kelas/mata yang sering mengalami kesalahan prediksi. Uji coba dilakukan sebanyak 15 kali iterasi. Kelas yang mengalami kesalahan prediksi yaitu berdasarkan jumlah kelas yang mengalami kesalahan ≥ 7 kali iterasi. Kelas-kelas tersebut adalah kelas 042, kelas 067, kelas 070, kelas 091 dan kelas 104. Kelas 042 mengalami kesalahan sebanyak 10 kali, kelas 067

sebanyak 7 kali, kelas 070 sebanyak 9 kali, kelas 091 sebanyak 11 kali dan kelas 104 sebanyak 11 kali. Dari beberapa contoh kelas tersebut, dapat dianalisis bahwa kesalahan prediksi dapat terjadi karena faktor gambar mata yang memiliki banyak *noise*, seperti bulu mata panjang dan kelopak mata yang menutupi area iris mata. Hal ini akan mempengaruhi nilai fitur pada area yang memiliki *noise* tersebut. Contoh sampel hasil prediksi kelas/mata yang salah ditunjukkan pada Tabel 5.9.

Tabel 5. 9 Contoh Gambar hasil prediksi kelas yang salah

No	Aktual	Prediksi
1	 Kelas 042	 Kelas 017
2	 Kelas 067	 Kelas 086
3	 Kelas 070	 Kelas 063
4	 Kelas 091	 Kelas 101
5	 Kelas 104	 Kelas 095

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1. Kesimpulan

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

1. Pada skenario 1, hasil akurasi maupun waktu eksekusi tidak memiliki perbedaan yang signifikan, namun hasil terbaik terjadi ketika fitur berjumlah 6 dengan nilai akurasi 64.93% dan waktu eksekusi 188.50 detik.
2. Berdasarkan hasil pengujian, maka dapat dikatakan bahwa performa terbaik ketika ukuran blok pada skenario 2 adalah 4 piksel yaitu menghasilkan akurasi sebesar 88.82% dengan waktu eksekusi selama 690.78 detik.
3. Fitur GLCM yang berjumlah 6 akan menghasilkan nilai akurasi terbaik ketika digabungkan dengan fitur AAD ukuran blok 4 piksel. Nilai akurasi yang dihasilkan yaitu sebesar 89.23% dengan waktu eksekusi selama 590.55 detik.
4. Pada skenario uji coba 4 sampai 6 menghasilkan performa terbaik, yaitu sebesar 89.23%. dengan waktu eksekusi selama 590.55 detik dengan menggunakan parameter fungsi aktivasi Tanh, *Optimizer* Adam dan jumlah neuron dengan 2 *hidden layer* (108, 108).
5. Berdasarkan hasil uji coba keseluruhan, Fitur Gabungan GLCM (6 fitur) dengan AAD (blok ukuran 4 piksel) menghasilkan akurasi tertinggi yaitu sebesar 89.23% dan waktu eksekusi 590.55 detik.

6.2. Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Peningkatan kemampuan perangkat keras untuk melakukan komputasi dapat mempercepat proses *learning*, serta memungkinkan penelitian pada banyak parameter.
2. Melakukan perhitungan klasifikasi atau pencocokan dengan metode yang lain.
3. Mencoba sistem ini dengan menggunakan dataset lain

Demikian saran yang mungkin menjadi masukan dalam pengembangan selanjutnya.

DAFTAR PUSTAKA

[1] N.Suciati, A.B.Anugrah, C.Fatichah, H.Tjandrasa, A.Z.Arifin, D.Purwitasari, D.A.Navastara, “Feature Extraction Using Statistical Moments Wavelet Transform for Iris recognition”, Information & Communication Technology and Systems (ICTS), 2016 International Conference on, IEEE, 2017.

[2] A.B.Anugrah, "Implementasi Pengenalan Iris Mata Menggunakan Metode Support Vector Machines dan Hamming Distance", Surabaya: Buku Tugas Akhir Fakultas Teknologi Informasi dan Komunikasi ITS , 2017.

[3] J.Daugman, S.Z.Li, A.K. Jain, “Iris Recognition at Airports and Border Crossing”, Encyclopedia of Biometrics hal 998-1004, Springer-Boston, 2015.

[4] K.Dongik, J.Yujin, T.Kar-Ann, S.Byungjun, K.Jaihie, “An empirical study on iris recognition in a mobile phone”, Expert Systems with Applications Vol.54 hal 328-339, 2016.

[5] Boles, Wageeh W., and Boualem Boashash, "A human identification technique using images of the iris and wavelet transform," IEEE transactions on signal processing 46, no. 4 (1998): 1185-1188.

[6] Ma, Li, Tieniu Tan, Yunhong Wang, and Dexin Zhang, "Efficient iris recognition by characterizing key local variations," IEEE Transactions on image processing 13, no. 6 (2004): 739-750.

[7] Bharath, B. V., A. S. Vilas, K. Manikantan, and S. Ramachandran, "Iris recognition using radon transform thresholding based feature extraction with Gradient-based Isolation as a pre-processing technique," In 2014 9th International Conference on Industrial and Information Systems (ICIIS), pp. 1-8. IEEE, 2014.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[8] Y. Unal, H.E. Kocer, H.E. Akkurt, "A Comparison of Feature Extraction Techniques for Diagnosis of Lumbar Intervertebral Degenerative Disc Disease", 2011 International Symposium on Innovations in Intelligent Systems and Applications, IEEE, Istanbul, Turkey, 2011.

[9] R. P. Moreno and A. Gonzaga, "Features Vector For Personal Identification Based On Iris Texture," *Departamento de Engenharia Elétrica - EESC – USP*, 2009.

[10] S. Koompairojn, K. Hua, K. A. Hua, and J. Srisomboon, "ComputerAided Diagnosis of Lumbar Stenosis Conditions", in Proc. of ComputerAided Diagnosis Conference, part of SPIE Symposium on Medical Imaging, San Diego, February 13-18, 2010.

[11] S. K. Agarwal, S. Shah and R. Kumar, "Classification ofmental tasks from EEG data using backtracking search optimization based neural classifier," *Neurocomputing*, pp. 1-7, 2015.

[12] D. Coyle, T. M. McGinnity and G. Prasad, "Improving the separability of multiple EEG features for a BCI by neural-time-series-prediction-preprocessing," *Biomedical Signal Processing and Control*, vol. 5, p. 196–204, 2010.

[13] J. M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial," *Proceedings of The IEEE*, vol. 83, no. 3, pp. 345 - 377, 1995.

[14] Scikit Learn. sklearn.feature_selection.SelectKBest [Internet]. Memilih fitur dengan skor tertinggi [dikutip 05 Juli 2018]. Tersedia dari: https://padamu.net/keterampilan-menulis-paragraf/

[15] M. R. Faundra, Aplikasi Filter Log-Gabor pada Sistem Pengenalan Iris Mata, Surabaya: Buku Tugas Akhir Fakultas Matematika dan Ilmu Pengetahuan Alam ITS , 2011.

[16] L. Masek, Recognition of Human Iris Patterns for Biometric Identification, The University of Western Australia, 2003.

[17] L. Ma, Y. Wang and T. Tan., "Iris recognition based on Multichannel Gabor Filtering", The 5th Asian Conference on Computer Vision, Australia, 2002.

[18] S. Kul, "Diagnosis of lumbar disc hernia from images using artificial neural network", MS thesis Fatih University Graduate Institute of Sciences and Engineering,2008.

[19] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features of Image Classification", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-3, no. 6, Nov. 1973.

[20] C. Lau (Ed.), Neural Networks: Theoretical Foundations and Analysis, IEEE Press, New York, 1992.

[22] M. Caudill, C. Butler, Understanding Neural Networks: Computer Explorations, MIT Press, Cambridge, MA, 1992.

[23] Y. Özbay, B. Karlık., "A Recognition of ECG Arrhythmia Using Artificial Neural Networks", 23rd Annual International Conference IEEE Engineering in Medicine and Biology Society, Istanbul, Turkey 2001.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Lampiran 7. 1 Hasil testing skenario uji coba GLCM 6 fitur dengan AAD ukuran blok 4 piksel pada Fold Ke-7

No	Nama Data uji	Aktual	Prediksi
1	001_1_1.bmp	1	1
2	002_2_2.bmp	2	2
3	003_1_2.bmp	3	3
4	004_2_4.bmp	4	4
5	005_2_2.bmp	5	5
6	006_1_2.bmp	6	6
7	007_1_1.bmp	7	7
8	008_1_3.bmp	8	8
9	009_2_3.bmp	9	9
10	010_2_2.bmp	10	10
11	011_1_3.bmp	11	11
12	012_2_1.bmp	12	12
13	013_1_1.bmp	13	13
14	014_1_1.bmp	14	14
15	015_1_1.bmp	15	15
16	016_2_2.bmp	16	16
17	017_1_1.bmp	17	17
18	018_2_1.bmp	18	18
19	019_2_3.bmp	19	19
20	020_2_4.bmp	20	20
21	021_1_1.bmp	21	21
22	022_1_1.bmp	22	22
23	023_1_1.bmp	23	23
24	024_1_2.bmp	24	24
25	025_1_3.bmp	25	25

26	026_1_1.bmp	26	22
27	027_2_1.bmp	27	86
28	028_1_3.bmp	28	28
29	029_1_1.bmp	29	29
30	030_1_1.bmp	30	30
31	031_1_1.bmp	31	31
32	032_1_1.bmp	32	32
33	033_1_1.bmp	33	33
34	034_1_1.bmp	34	34
35	035_1_1.bmp	35	35
36	036_2_3.bmp	36	36
37	037_2_1.bmp	37	81
38	038_1_1.bmp	38	38
39	039_1_1.bmp	39	39
40	040_1_1.bmp	40	40
41	041_2_2.bmp	41	41
42	042_2_4.bmp	42	42
43	043_1_1.bmp	43	43
44	044_1_1.bmp	44	44
45	045_2_2.bmp	45	45
46	046_1_3.bmp	46	46
47	047_2_1.bmp	47	47
48	048_2_1.bmp	48	48
49	049_1_2.bmp	49	49
50	050_2_2.bmp	50	50
51	051_1_2.bmp	51	51
52	052_1_3.bmp	52	48
53	053_1_1.bmp	53	53
54	054_1_1.bmp	54	54

55	055_1_1.bmp	55	55
56	056_1_1.bmp	56	56
57	057_1_1.bmp	57	57
58	058_1_3.bmp	58	58
59	059_1_1.bmp	59	59
60	060_1_1.bmp	60	60
61	061_1_3.bmp	61	61
62	062_1_1.bmp	62	62
63	063_1_1.bmp	63	63
64	064_1_1.bmp	64	64
65	065_1_1.bmp	65	65
66	066_2_2.bmp	66	79
67	067_1_1.bmp	67	67
68	068_1_1.bmp	68	68
69	069_1_1.bmp	69	69
70	070_2_1.bmp	70	73
71	071_1_1.bmp	71	71
72	072_1_1.bmp	72	72
73	073_1_1.bmp	73	62
74	074_1_1.bmp	74	74
75	075_2_2.bmp	75	75
76	076_1_1.bmp	76	76
77	077_1_1.bmp	77	81
78	078_1_1.bmp	78	78
79	079_1_1.bmp	79	79
80	080_1_1.bmp	80	80
81	081_2_2.bmp	81	81
82	082_1_1.bmp	82	82
83	083_2_4.bmp	83	83

84	084_1_1.bmp	84	84
85	085_1_1.bmp	85	85
86	086_2_3.bmp	86	86
87	087_1_1.bmp	87	87
88	088_1_1.bmp	88	88
89	089_2_4.bmp	89	80
90	090_1_1.bmp	90	90
91	091_2_1.bmp	91	91
92	092_2_3.bmp	92	85
93	093_1_1.bmp	93	93
94	094_1_1.bmp	94	94
95	095_1_2.bmp	95	95
96	096_1_1.bmp	96	96
97	097_1_1.bmp	97	97
98	098_1_1.bmp	98	98
99	099_1_1.bmp	99	99
100	100_1_1.bmp	100	100
101	101_1_2.bmp	101	101
102	102_2_3.bmp	102	102
103	103_2_1.bmp	103	103
104	104_1_2.bmp	104	97
105	105_1_3.bmp	105	105
106	106_1_1.bmp	106	106
107	107_1_2.bmp	107	101
108	108_2_3.bmp	108	108

Lampiran 7. 2 Hasil *Running* skenario Uji coba GLCM 6 fitur, AAD ukuran blok 4 piksel dan Gabungan GLCM 6 fitur dengan AAD ukuran blok 4 piksel

fold	Perulangan ke-	GLCM	AAD	GLCM-AAD
1	1	0,69	0,91	0,9
	2	0,66	0,89	0,9
	3	0,69	0,88	0,86
	4	0,71	0,89	0,9
	5	0,67	0,93	0,91
	6	0,59	0,9	0,89
	7	0,66	0,91	0,91
	8	0,64	0,89	0,91
	9	0,69	0,91	0,89
	10	0,7	0,88	0,86
	11	0,75	0,91	0,91
	12	0,7	0,93	0,88
	13	0,7	0,86	0,92
	14	0,69	0,94	0,91
	15	0,64	0,92	0,91
	16	0,73	0,91	0,86
	17	0,7	0,9	0,87
	18	0,68	0,93	0,89
	19	0,69	0,92	0,87
	20	0,71	0,93	0,89
	1	0,69	0,92	0,87
	2	0,69	0,87	0,89
	3	0,69	0,88	0,89
	4	0,67	0,89	0,9
	5	0,67	0,87	0,88

2	6	0,69	0,89	0,91
	7	0,71	0,85	0,87
	8	0,77	0,87	0,88
	9	0,7	0,88	0,9
	10	0,67	0,86	0,91
	11	0,69	0,88	0,89
	12	0,69	0,84	0,9
	13	0,67	0,86	0,88
	14	0,63	0,86	0,9
	15	0,67	0,83	0,9
	16	0,7	0,86	0,88
	17	0,68	0,89	0,88
	18	0,68	0,9	0,9
	19	0,63	0,87	0,87
	20	0,69	0,84	0,89
3	1	0,75	0,91	0,89
	2	0,73	0,9	0,9
	3	0,71	0,92	0,84
	4	0,74	0,85	0,85
	5	0,74	0,91	0,83
	6	0,72	0,89	0,88
	7	0,69	0,94	0,87
	8	0,76	0,91	0,87
	9	0,71	0,9	0,89
	10	0,73	0,89	0,87
	11	0,73	0,9	0,88
	12	0,7	0,89	0,93
	13	0,7	0,87	0,88
	14	0,73	0,86	0,87

	15	0,75	0,88	0,86
	16	0,77	0,91	0,9
	17	0,74	0,93	0,89
	18	0,73	0,9	0,84
	19	0,66	0,93	0,9
	20	0,69	0,89	0,89
4	1	0,75	0,86	0,92
	2	0,78	0,88	0,93
	3	0,73	0,83	0,88
	4	0,69	0,88	0,91
	5	0,72	0,85	0,94
	6	0,68	0,85	0,92
	7	0,76	0,86	0,91
	8	0,77	0,81	0,9
	9	0,68	0,82	0,9
	10	0,71	0,86	0,91
	11	0,73	0,86	0,92
	12	0,75	0,88	0,89
	13	0,75	0,87	0,9
	14	0,71	0,86	0,93
	15	0,76	0,8	0,89
	16	0,75	0,88	0,94
	17	0,77	0,86	0,89
	18	0,75	0,85	0,91
	19	0,67	0,9	0,92
	20	0,76	0,87	0,9
	1	0,7	0,91	0,89
	2	0,68	0,92	0,9
	3	0,64	0,92	0,88

5	4	0,62	0,94	0,91
	5	0,65	0,91	0,9
	6	0,64	0,92	0,91
	7	0,69	0,93	0,9
	8	0,64	0,92	0,91
	9	0,64	0,92	0,86
	10	0,67	0,9	0,9
	11	0,7	0,9	0,88
	12	0,7	0,93	0,89
	13	0,69	0,9	0,89
	14	0,66	0,88	0,89
	15	0,63	0,95	0,86
	16	0,63	0,9	0,88
	17	0,66	0,91	0,9
	18	0,7	0,88	0,9
	19	0,67	0,88	0,93
	20	0,68	0,91	0,91
6	1	0,8	0,87	0,9
	2	0,74	0,9	0,94
	3	0,78	0,89	0,88
	4	0,79	0,87	0,9
	5	0,74	0,9	0,9
	6	0,79	0,87	0,89
	7	0,79	0,89	0,92
	8	0,81	0,91	0,9
	9	0,76	0,91	0,91
	10	0,76	0,91	0,91
	11	0,79	0,91	0,87
	12	0,81	0,87	0,88

	13	0,73	0,87	0,91
	14	0,79	0,91	0,92
	15	0,73	0,92	0,9
	16	0,77	0,88	0,9
	17	0,79	0,89	0,89
	18	0,79	0,92	0,92
	19	0,79	0,89	0,9
	20	0,77	0,92	0,91
7	1	0,71	0,89	0,9
	2	0,71	0,88	0,88
	3	0,71	0,89	0,91
	4	0,71	0,88	0,9
	5	0,73	0,9	0,87
	6	0,66	0,87	0,9
	7	0,71	0,9	0,88
	8	0,69	0,86	0,88
	9	0,73	0,89	0,88
	10	0,76	0,9	0,89
	11	0,74	0,89	0,9
	12	0,69	0,92	0,9
	13	0,75	0,89	0,88
	14	0,7	0,87	0,93
	15	0,76	0,86	0,9
	16	0,71	0,87	0,89
	17	0,68	0,85	0,86
	18	0,73	0,91	0,87
	19	0,72	0,9	0,87
	20	0,7	0,89	0,89

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Rahma Dini Maghfirotul Laily atau biasa dipanggil Dini dilahirkan di Lumajang pada tanggal 05 Agustus 1995 dan dibesarkan di Lumajang. Penulis adalah anak terakhir dari dua bersaudara.

Penulis menempuh pendidikan di SDN Tempeh Lor 01 (2002-2008), SMP N 1 Tempeh (2008-2011), dan SMAN 2 Lumajang (2011-2014). Setelah lulus SMA penulis melanjutkan ke jenjang perkuliahan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Bidang Studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Komputasi Cerdas dan Visi.

Selama menempuh kuliah penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer (HMTC) ITS pada departemen Kesejahteraan Mahasiswa. Selain itu penulis juga aktif di organisasi Lembaga Dakwah Jurusan (LDJ) Keluarga Muslim Informatika (KMI) ITS sebagai staff di Departemen Annisa. Selain itu penulis juga aktif menjadi administrator Lab Komputasi Cerdas dan Visi (KCV) Departemen Informatika ITS dari tahun 2016-2018. Penulis dapat dihubungi melalui alamat *email* rahmadinilaily123@gmail.com.