

BAB III

DESAIN 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 baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan tugas akhir ini bisa tercapai.

3.1 Perancangan Data

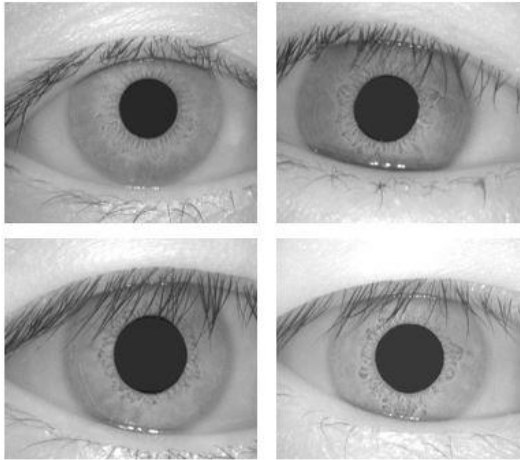
Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

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. Terakhir, 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. Pada database ini berisi 756 citra mata dengan 108 mata yang berbeda. Masing-masing mata ini dianggap sebagai satu kelas. Didalam satu kelas terdapat tujuh citra mata. Sehingga total keseluruhan citra mata pada database ini yaitu $108 \times 7 = 756$ citra mata. Semua citra mata 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 optik 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 CASIA

Database ini dibagi menjadi dua, yaitu *training data* dan *testing data* atau yang biasa dikenal dengan data latih dan data uji. Training data merupakan data yang digunakan untuk melatih sistem apakah akurasi dan keluarannya sudah sesuai dengan apa yang diharapkan. Sedangkan testing data merupakan data yang digunakan untuk menguji sistem setelah melakukan proses *training*. Dari tiap kelas didalam database akan dibagi dua bagian, yakni empat sebagai data latih dan tiga sebagai data uji.

3.1.2 Data Proses

Data proses adalah data yang digunakan selama proses berjalan. Data proses meliputi nama data, tipe data, dan keterangannya. Data proses ditunjukkan dalam Tabel 3.1.

Tabel 3.1 Data Proses

No.	Nama data	Keterangan
1.	Data citra mata	Data dari database yang akan diproses bertipe <i>bitmap</i>
2.	Data fitur	Vektor fitur citra yang telah terdeteksi dan disimpan dalam bentuk <i>.mat</i>
3.	Data citra implementasi	Citra mata yang diolah dalam pengimplementasian yang bertipe data <i>double</i>

Pemrosesan akan dilakukan dengan menggunakan data masukan yang berupa citra mata *grayscale* bertipe *bitmap*. Data citra mata ini akan dikonversi kedalam tipe data *double* untuk memudahkan pengolahan citra. Data latih digunakan sebagai data belajar klasifikasi, sedangkan juga dengan data uji sebagai masukan untuk pencocokan. Data latih dan data uji ini nanti akan diproses sampai menjadikan vektor fitur citra iris. Data vektor fitur juga disimpan dalam tipe data *double*. Pada saat pencocokan, dua data fitur dari data uji dan data latih akan dibandingkan satu sama lain.

3.1.3 Data Keluaran

Data keluaran dari sistem ini adalah hasil dari proses yang sudah dilakukan. Data masukan yang sudah diekstrasi fiturnya akan diproses dengan menggunakan klasifikasi SVM dan klasifikasi *Hamming distance* secara terpisah. Hasil dari proses klasifikasi tersebut adalah prediksi kelas pada masing-masing data uji dan akurasi.

3.2 Desain Umum Sistem

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

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

Tahapan yang pertama merupakan tahap praproses. Namun sebelum masuk ke praproses, citra mata harus didapatkan terlebih dahulu melalui tahapan akuisisi citra mata dari *disk*. Setelah mendapatkan sebuah citra mata, citra mata tersebut akan dipanggil dan diproses, sehingga tahap praproses dapat dilakukan. Tujuan tahap praproses adalah mengidentifikasi bagian iris pada citra mata. Tahap praproses ini dibagi menjadi tiga bagian. Praproses itu sendiri terdiri dari deteksi tepi, deteksi batas pupil dan iris, dan pemisahan bulu dan kelopak. Hasil dari tahap praproses adalah citra iris. Tahap ini juga memisahkan bulu mata dan kelopak mata pada lingkaran iris.

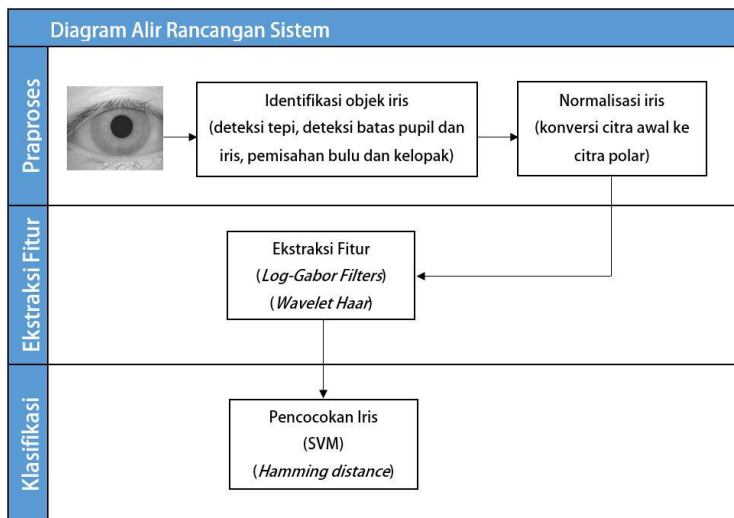
Bagian iris yang sudah teridentifikasi kemudian dilakukan normalisasi. Tujuannya adalah menciptakan dimensi yang konstan pada iris, karena iris luasan iris manusia berubah-ubah tergantung dari berbagai faktor dan juga memudahkan untuk melakukan verifikasi nantinya. Hasil dari tahap praproses diolah kembali dengan menggunakan transformasi polar pada citra iris sehingga menghasilkan sebuah citra iris yang mudah untuk diolah pada tahap selanjutnya.

Setelah melakukan normalisasi, citra iris dimasukan kedalam tahap ekstraksi fitur. Ekstraksi fitur berguna untuk mendapatkan tekstur iris yang penting. Pada tahap ekstraksi fitur ini menggunakan dua buah ekstraksi fitur yang tidak saling

berkorelasi, yaitu *wavelet Haar* dan *log-gabor filter*. Kedua metode tersebut akan diujikan pada dua metode klasifikasi yang berbeda dan tidak saling berkorelasi pula. Hasil tahap ini adalah vektor fitur yang digunakan untuk melakukan verifikasi pada pengenalan iris.

Setelah melalui tahap ekstraksi fitur, vektor-vektor fitur akan di verifikasi menggunakan metode *Support Vector Machine* atau menggunakan metode *Hamming distance* dengan menggunakan fitur yang telah didapatkan. Hasil akhir yang dikeluarkan adalah nama kelas dengan menggunakan masing-masing dua metode klasifikasi tersebut.

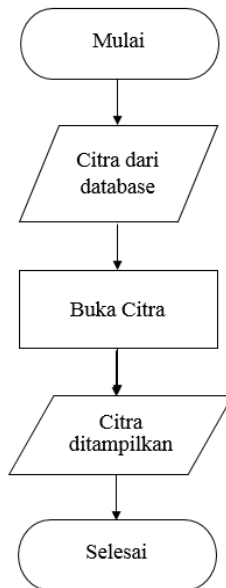
Gambar 3.2 menunjukkan tiga tahapan utama dalam pengenalan iris. Terdapat tahap Praproses, Ekstraksi Fitur, dan Klasifikasi. Tahap Praproses adalah tahap untuk mengidentifikasi iris dari citra input mata. Tahap Ekstraksi Fitur terdiri dari dua metode, *wavelet Haar* dan *log-gabor filter*. Tahap Klasifikasi juga terdiri dari dua metode, SVM dan *Hamming distance*.



Gambar 3.2 Diagram Alir Rancangan Sistem

3.3 Akuisisi Citra Mata

Seperti yang telah dijelaskan pada bagian perancangan data, citra mata akan digunakan sebagai masukan untuk keseluruhan proses pengenalan iris. Citra mata diasumsikan telah disimpan didalam *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.

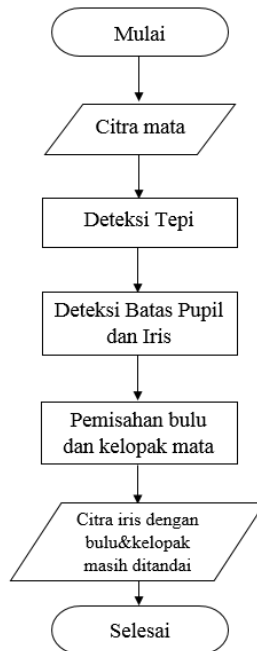


Gambar 3.3 Diagram Alir Akuisisi Citra Mata

Tahap awalnya adalah memilih sebuah citra dari *disk*, kemudian dari *disk* tersebut citra akan dipanggil kedalam aplikasi. Terakhir citra yang akan dikenali irisnya, ditampilkan didalam aplikasi. Citra mata yang ditampilkan mempunyai tipe data uint8.

3.4 Perancangan Praproses

Citra mata yang sudah diakuisisi tidak bisa langsung dilakukan verifikasi. Tahap awal yang dilakukan adalah tahap praproses atau nama lainnya *preprocessing*. Praproses bertujuan untuk memisahkan daerah iris dari citra mata dengan mendeteksi batas dalam dan batas luar iris. Praproses dilakukan melalui tiga tahap yaitu deteksi tepi, deteksi batas pupil dan iris, dan pemisahan bulu dan kelopak. Diagram alir tahap praproses ditunjukkan pada Gambar 3.4.



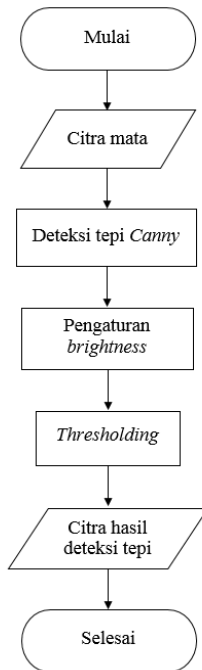
Gambar 3.4 Diagram Alir Praproses

Tahap pertama yaitu deteksi tepi pada citra mata. Kemudian mendeteksi batas antara pupil dan iris untuk mendapatkan citra iris. Setelah itu dipisahkan bagian bulu dan kelopak mata yang merupakan *noise* pada daerah iris. Hasil

akhir dari tahap ini adalah citra iris yang akan dinormalisasi. Masing-masing tahap akan dijelaskan lebih lanjut pada subbab ini.

3.4.1 Deteksi Tepi

Tahap deteksi tepi adalah tahap dimana 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 Proses Deteksi Tepi

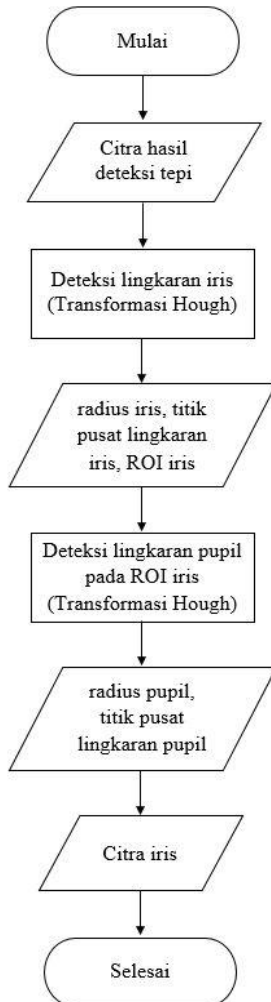
Masukan dari tahap ini adalah citra mata hasil dari akuisisi. Proses yang dilakukan dilakukan pertama kali pada data masukan tersebut adalah mendeteksi tepi. Deteksi tepi yang digunakan adalah deteksi tepi *Canny*. Didalam deteksi tepi tersebut juga sekaligus dilakukan proses pengkontrasan dan thresholding.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

3.4.2 Deteksi Batas Pupil dan Iris

Tahapan yang dilakukan disini merupakan proses untuk mendapatkan batas lingkaran iris dan pupil. Masukan dari proses ini adalah citra hasil dari deteksi tepi. Untuk mendapatkan lingkaran iris dan lingkaran pupil, dilakukan dengan metode transformasi Hough pada citra masukan. Transformasi Hough ini 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. Jika menggunakan transformasi Hough sekaligus untuk mencari kedua lingkaran, maka waktu prosesnya akan lama, maka dari itu menggunakan dua kali transformasi Hough sebagai gantinya agar waktu proses bisa direduksi.

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 terdapat 70 buah radius pada bidang akumulator. Titik pusat lingkaran iris didapatkan dari nilai intensitas maksimal pada tiap bidang akumulator. Radius lingkaran 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 Proses Deteksi Batas Pupil dan Iris

Setelah melakukan transformasi pada lingkaran iris, selanjutnya dilakukan kembali transformasi yang sama terhadap lingkaran pupil dengan menggunakan hasil dari transformasi pertama. Dari hasil transformasi pertama ini bisa ditentukan ROI iris. Setelah itu transformasi Hough kedua dilakukan pada ROI iris tersebut 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 pupil 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 pupil dan iris ditunjukkan pada Gambar 3.6. Alhasil proses keseluruhan ini menghasilkan titik pusat dan radius pada masing-masing iris dan pupil yang nantinya akan diproses pada tahap berikutnya yakni normalisasi iris.

3.4.3 Pemisahan Bulu dan Kelopak Mata

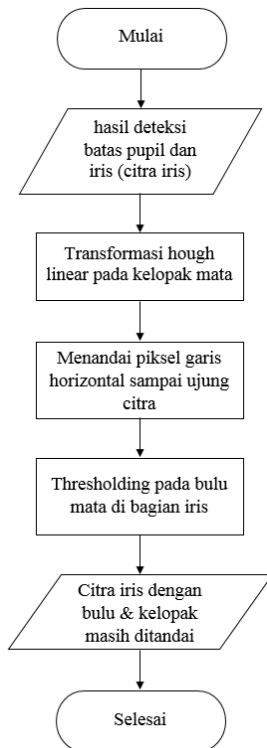
Citra mata yang telah diidentifikasi bagian irisnya masih mengandung bulu dan kelopak mata. Keduanya dianggap sebagai *noise*. Oleh karena itu perlu dilakukan penghilangan bulu dan kelopak yang menghalangi bagian iris.

Untuk memisahkan bagian kelopak mata, caranya dengan membuat sebuah garis pada bagian atas dan bagian bawah kelopak mata dengan transformasi Hough linear. Kemudian, 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 citra diganti dengan notasi NaN. Notasi ini digunakan untuk visualisasi sebagai *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 thresholding dilakukan, semua piksel yang nilainya dibawah threshold akan berubah. Perubahan piksel ini digantikan dengan notasi NaN. Nantinya piksel dengan notasi NaN ini akan diganti nilainya dengan yang lebih baik pada tahap berikutnya yakni normalisasi iris.

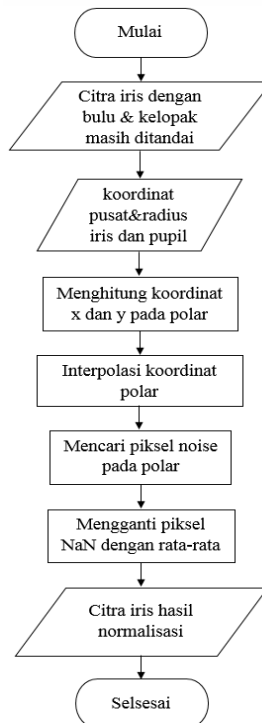
Hasil keluaran dari proses ini merupakan citra yang sudah dapat di normalisasikan dan *noise* sudah berhasil ditandai. Gambar 3.7 menunjukan diagram alir proses pemisahan bulu dan kelopak mata.



Gambar 3.7 Diagram Alir Pemisahan Bulu dan Kelopak Mata

3.5 Normalisasi Iris

Pada bab ini akan menjelaskan bagaimana citra iris yang sudah didapat dari hasil praproses dilakukan tahap normalisasi. Citra iris yang berbentuk lingkaran susah untuk dilakukan verifikasi karena mempunyai dimensi yang 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 tanpa memperhatikan penyekalaan atau pembesaran. Diagram alir tahap normalisasi ditunjukan pada Gambar 3.8.



Gambar 3.8 Diagram Alir Normalisasi Iris

Citra iris yang mempunyai radius dan titik pusat pupil dan iris yang telah ditemukan pada tahap praproses digunakan untuk tahap normalisasi. Tahap ini melakukan normalisasi daerah lingkaran iris dari koordinat cartesian ke koordinat polar. Menggunakan transformasi polar seperti pada subbab 2.6, yang didalamnya terdapat persamaan untuk mengubah bidang daerah lingkaran iris dari bentuk lingkaran ke bentuk persegi panjang.

Hasil keluaran dari tahap ini adalah citra iris yang telah dinormalisasi berbentuk persegi panjang dengan dimensi 64x512 piksel. Piksel-piksel pada citra normalisasi yang bernotasi NaN digantikan nilainya dengan rata-rata nilai piksel citra normalisasi.

3.6 Perancangan Ekstraksi Fitur

Ekstraksi fitur iris merupakan ciri khas penting pada citra iris ternormalisasi. Masukannya adalah hasil dari tahap normalisasi. Terdapat dua metode ekstraksi fitur yang dipakai. Kedua ekstraksi fitur ini tidak berkaitan satu sama lain. Metode ekstraksi fitur yang dipakai adalah *wavelet Haar* dan *log-Gabor filter*. Ekstraksi fitur yang dipakai *wavelet Haar* adalah koefisien aproksimasi dari hasil dekomposisi pada tingkat-tingkat tertentu. Sedangkan ekstraksi fitur *log-Gabor filter* menghasilkan bilangan bit biner. Masing-masing dari ekstraksi fitur tersebut menghasilkan vektor fitur yang akan digunakan untuk melakukan tahap klasifikasi.

3.6.1 Fitur Koefisien Wavelet Haar

Ekstraksi fitur dimulai dengan dengan melakukan transformasi *wavelet Haar* pada citra iris yang sudah ternormalisasi. Citra iris yang ternormalisasi akan dilakukan proses sesuai dengan penjelasan pada subbab 2.7, yang dimana citra iris akan didekomposisi. Pada sistem ini, citra iris akan didekomposisi pada tingkat tertentu. Artinya, hasil dimensi citra iris akan tereduksi menjadi lebih kecil dari dimensi awal.

Pseudocode penghitungan dekomposisi citra iris ditunjukkan pada Gambar 3.9.

Masukan	Citra iris hasil normalisasi
Keluaran	Vektor fitur koefisien aproksimasi dekomposisi tingkat 3
<pre> 1. wname ← 'haar' (nama wavelet) 2. 3. for i=1 to 3 (contoh jika direduksi lvl 3) 4. [cA,~,~,~] = dwt2(citraNormalisasi,wname) 5. citraNormalisasi ← cA 6. end 7. return cA </pre>	

Gambar 3.9 *Pseudocode* Dekomposisi Wavelet

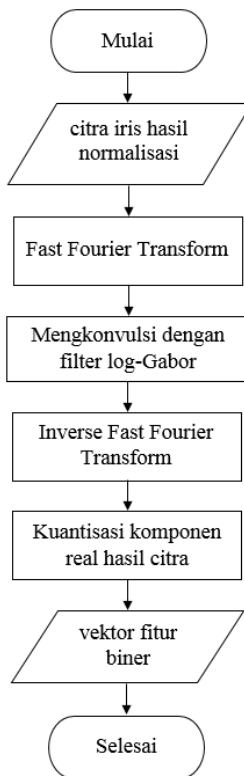
Keluaran dari tahap ini adalah vektor fitur dengan dimensi 8x64 piksel. Koefisien aproksimasi diambil karena berisi informasi yang jelas dibandingkan koefisien detail. Vektor fitur tersebut merupakan representasi dari citra iris yang dibentuk dengan transformasi *wavelet* yang akan digunakan sebagai masukan untuk tahap klasifikasi.

3.6.2 Fitur Log-Gabor Filter

Data masukan yang digunakan adalah citra normalisasi dari tahap normalisasi dan citra *noise* dari hasil proses pemisahan bulu dan kelopak mata. Pada dasarnya ekstraksi fitur ini akan mengubah fitur-fitur iris pada citra hasil normalisasi kedalam bentuk bit biner dengan dimensi yang sama, yakni 64x512. Untuk itu digunakan ekstraksi fitur *log-Gabor filter* sebagai suatu batu loncatan untuk memperoleh bilangan biner tersebut. Gambar 3.10 menunjukkan diagram alir dari ekstraksi fitur ini.

Proses ekstraksi fitur ini pada dasarnya menjadikan citra iris menjadi bilang biner seperti yang dijelaskan pada subbab 2.8. Setiap baris pada citra yang telah dinormalisasi, yaitu sebanyak 512 piksel dilakukan proses *Fast Fourier Transform* untuk mempresentasikan citra pada domain

frekuensi. Kemudian setelah itu, nilai tersebut akan dikonvulsi dengan filter log-*Gabor*. Hasil konvulsi akan dilakukan proses *Inverse Fast Fourier Transform* untuk mengembalikan representasi citra ke dalam domain spasial. Tetapi saat mengembalikan citra ke dalam domain spasial, tiap piksel pada citra memiliki komponen bilangan *real*. Selanjutnya akan dilakukan proses kuantisasi di komponen bilangan *real* pada citra.



Gambar 3.10 Diagram Alir Pembuatan Fitur Biner

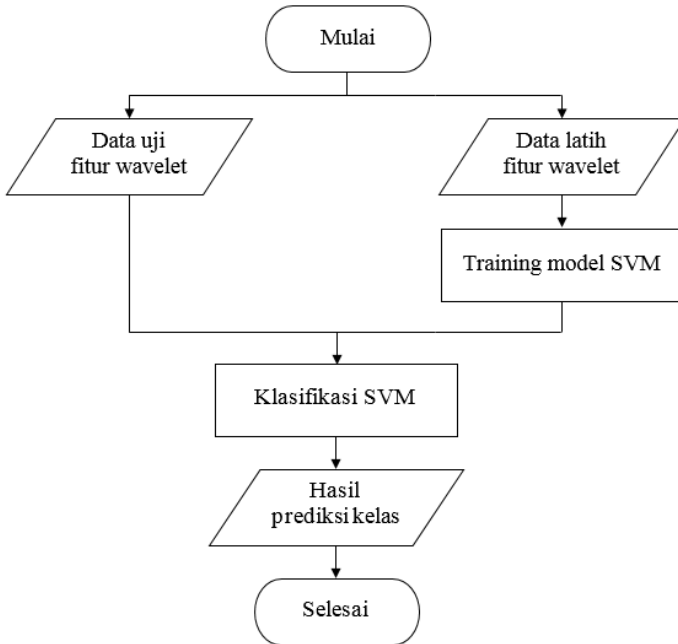
3.7 Perancangan Klasifikasi

Klasifikasi dilakukan dengan mengolah data hasil yang sudah didapatkan dari proses ekstraksi fitur. Setelah melalui tahap ekstraksi fitur, citra iris mata direpresentasikan dalam bentuk vektor fitur. Untuk setiap data latih dan data uji citra iris akan dijadikan sebagai vektor fitur. Kemudian dilakukan tahap verifikasi menggunakan metode klasifikasi *Support Vector Machine* (SVM) dan metode *Hamming distance*. Seperti halnya metode ekstraksi fitur, kedua metode klasifikasi ini tidak saling berkaitan.

Tahap ini terlebih dahulu membuat database vektor fitur seluruh citra input dan disimpan dalam sebuah file berekstensi .mat dengan menggunakan Matlab. Jika ada citra masukan yang ingin dikenali irisnya termasuk pada kelas mana, maka akan dilakukan klasifikasi menggunakan salah satu metode klasifikasi. 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.

3.7.1 Klasifikasi Support Vector Machine Fitur Wavelet

Pada subbab ini dijelaskan klasifikasi dengan menggunakan metode SVM, vektor fitur yang didapatkan adalah hasil dari ekstraksi fitur *wavelet Haar*. Sebelum masuk ke dalam proses klasifikasi, data uji dan data latih diproses terlebih dahulu menjadi vektor fitur. Dengan menggunakan SVM, data latih akan dilakukan pembuatan model pada tiap-tiap kelas. Setelah itu data uji akan diprediksi menggunakan SVM termasuk kelas manakah data uji tersebut. Gambar 3.11 menunjukkan diagram alir tahap klasifikasi dengan menggunakan SVM.



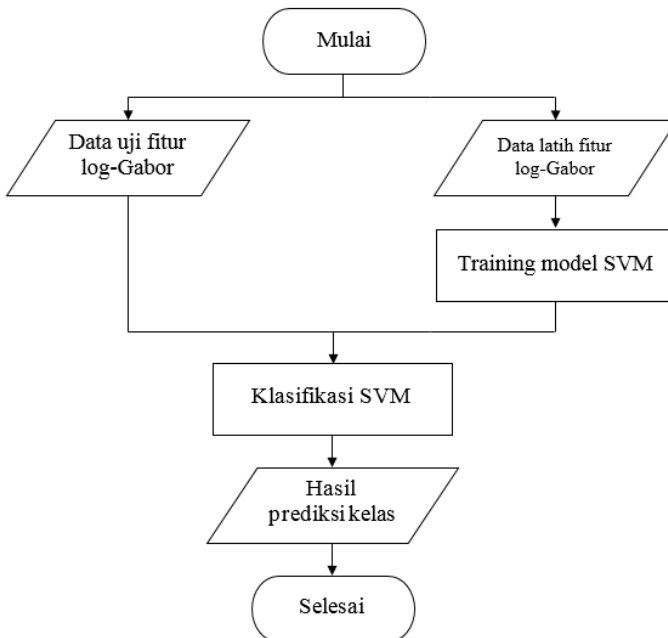
Gambar 3.11 Diagram Alir Klasifikasi SVM menggunakan Fitur *Wavelet Haar*

3.7.2 Klasifikasi Support Vector Machine Fitur Log-Gabor

Pada subbab ini dijelaskan klasifikasi dengan menggunakan metode *Support Vector Machine* dengan data masukannya adalah vektor fitur yang didapatkan adalah hasil dari ekstraksi fitur *log-Gabor filter*. Masukan dari proses ini adalah vektor fitur hasil ekstraksi fitur *log-Gabor filter*. Namun dalam hal ini, proses ekstraksi fitur *log-Gabor filter* tidak menggunakan proses kuantisasi elemen matrik menjadi bilangan biner.

Sebelum masuk ke dalam proses klasifikasi, data uji dan data latih diproses terlebih dahulu menjadi vektor fitur. Dengan menggunakan SVM, data latih akan dilakukan pembuatan model pada tiap-tiap kelas. Setelah itu data uji akan

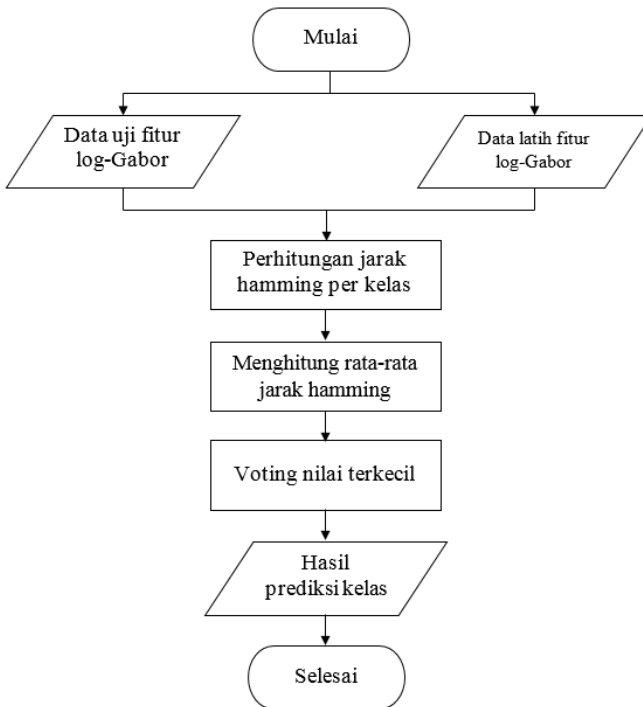
diprediksi menggunakan SVM termasuk kelas manakah data uji tersebut. Gambar 3.12 menunjukan diagram alir tahap klasifikasi ini.



Gambar 3.12 Diagram Alir Klasifikasi SVM menggunakan Fitur *Log-Gabor*

3.7.3 Klasifikasi Hamming Distance Fitur Log-Gabor

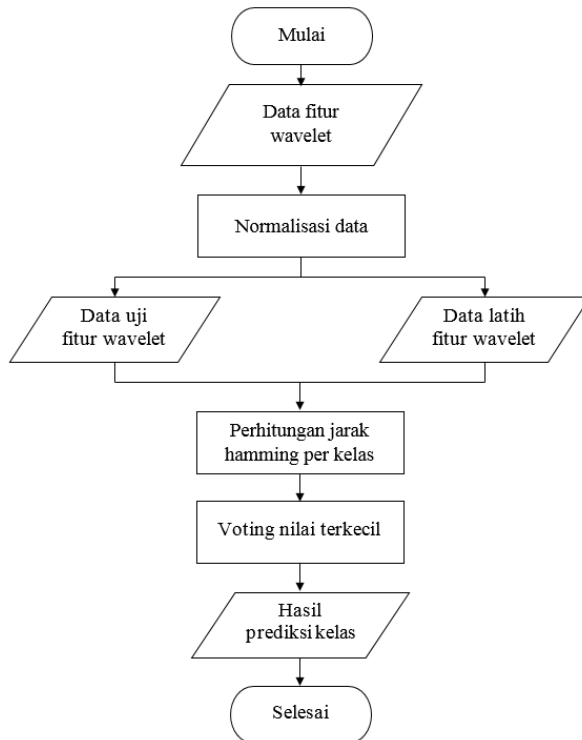
Pada klasifikasi dengan menggunakan metode *hamming distance* pada subbab ini, fitur yang didapatkan adalah hasil dari ekstraksi fitur log-Gabor filters. Sebelum masuk ke dalam proses klasifikasi, data uji dan data latih diproses terlebih dahulu menjadi fitur dalam bentuk biner. Setelah itu, dilakukan perhitungan Hamming. Diagram alir ditunjukkan pada Gambar 3.13, prediksi kelas adalah yang mempunyai nilai paling kecil.



Gambar 3.13 Diagram Alir *Hamming distance* menggunakan Fitur *Log-Gabor*

3.7.4 Klasifikasi Hamming Distance Fitur Wavelet

Pada subbab ini, klasifikasi dilakukan dengan menggunakan metode *hamming distance* dan fitur yang dipakai adalah hasil dari ekstraksi fitur *wavelet Haar*. Sebelum masuk ke dalam proses klasifikasi, dataset yang sudah dilakukan proses dekomposisi *wavelet* dilakukan normalisasi data terlebih dahulu. Kemudian dataset tersebut dibagi menjadi data uji dan data latih. Selanjutnya kedua data tersebut diproses menjadi fitur dalam bentuk biner. Setelah itu, dilakukan perhitungan Hamming. Diagram alir ditunjukkan pada Gambar 3.14, prediksi kelas adalah yang mempunyai nilai paling kecil.



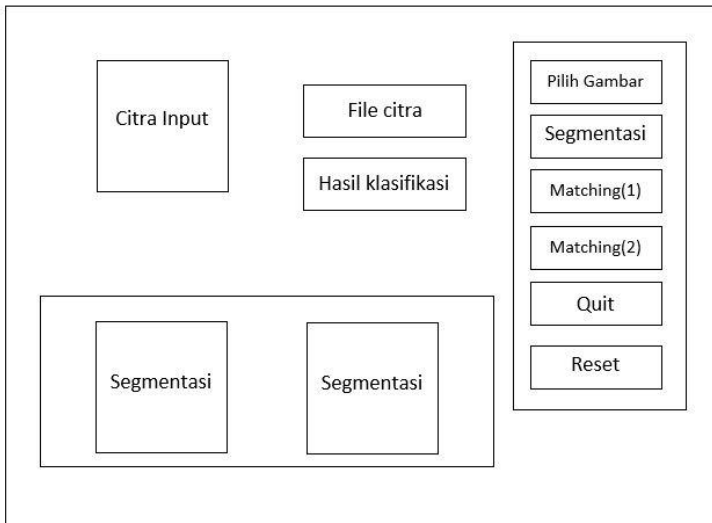
Gambar 3.14 Diagram Alir *Hamming distance* menggunakan Fitur Wavelet

3.8 Perancangan Antarmuka

Bagian ini adalah bagian sistem yang memungkinkan user untuk berinteraksi dengan sistem pengenalan iris mata ini. Antarmuka didesain semudah dan sesederhana mungkin untuk digunakan. Gambar 3.15 menunjukkan desain antarmuka sistem pengenalan iris ini. Dalam rancangan antarmuka ini terdapat satu kotak gambar untuk input citra, dua kotak gambar untuk

hasil segmentasi citra, dua kotak teks hasil inpuan, dan enam tombol pada bagian kanan sebagai menu utama.

Kotak gambar citra input akan menunjukkan gambar yang dipilih dari tombol menu *Pilih Gambar*. Nama dari gambar yang dipilih akan ditampilkan pada kotak teks *Filecitra*. Kemudian dua kotak gambar segmentasi akan menghasilkan hasil dari segmentasi mata dari tombol menu *Segmentasi*.



Gambar 3.15 Desain Antarmuka

Tombol menu *Matching(1)* merupakan tombol menu untuk melakukan pencocokan dengan menggunakan klasifikasi *Support Vector Machine*, sedangkan tombol menu *Matching(2)* untuk melakukan pencocokan menggunakan klasifikasi *Hamming distance*. Hasilnya akan ditampilkan pada kotak teks *Hasil klasifikasi*.

Tombol menu *Quit* digunakan untuk keluar dari antarmuka dan tombol menu *Reset* digunakan untuk mendefaultkan ke tampilan awal.

BAB IV IMPLEMENTASI

Pada sub bab implementasi ini menjelaskan mengenai proses pengenalan iris mata dengan menampilkan kode sumber yang digunakan. Implementasi tiap tahapan meliputi implementasi mendapatkan citra mata, praproses (deteksi tepi, deteksi batas pupil dan iris, serta pemisahan kelopak dan bulu mata), normalisasi iris, ekstraksi fitur iris (*wavelet Haar* dan *log-Gabor filter*), dan klasifikasi iris (*Support Vector Machine* dan *Hamming distance*).

4.1 Lingkungan Implementasi

Lingkungan implementasi pengenalan iris ini mencakup perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Perancangan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i5-33170U CPU @ 1.70GHz 1.70GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows 8 64-bit Perangkat Pengembang: Matlab 8.3 & <i>Image Processing Toolbox</i> Perangkat Pembantu: Microsoft Excel 2013, <i>libsvm</i>

4.2 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*. Implementasi akuisisi citra mata dilakukan dengan menggunakan fungsi yang sudah disediakan pada *Image Processing Toolbox* Matlab seperti yang ditunjukkan pada Kode Sumber 4.1.

1.	% menampilkan citra masukan
2.	[path , user_cancel] = imgetfile();
3.	if user_cancel
4.	msgbox('Silakan Pilih Gambar Kembali', ...
5.	'Cancel PopUp');
6.	im=imread(path);
7.	axes(handles.axes1);imshow(im);
8.	

Kode Sumber 4.1 Implementasi Tahap Akuisisi Citra Mata

Fungsi `imgetfile()` digunakan untuk memilih file yang berada dalam *disk*. Fungsi tersebut mengembalikan *path* dari file yang telah dipilih dan terdapat *error handling* jika user tidak jadi mengambil file. Jika tidak jadi memilih file, maka terdapat pesan yang muncul seperti yang ditunjukkan baris 4 sampai 5. Terakhir *path* file yang telah dipilih akan dibaca dan ditampilkan pada *axes* seperti yang ditunjukkan pada baris 6 sampai 7.

4.3 Implementasi Preprocessing

Setelah citra mata ditampilkan dalam aplikasi perangkat lunak, langkah berikutnya adalah melakukan tahap *preprocessing* atau praproses. Tujuan utama tahap ini adalah mengidentifikasi bagian iris pada citra mata. Tahap ini terdiri dari tiga tahap lagi, yaitu deteksi tepi, deteksi batas pupil dan iris, serta pemisahan bulu dan kelopak mata. masing-masing tahap akan dijelaskan pada subbab berikutnya.

4.3.1 Implementasi Deteksi Tepi

Tahap ini adalah langkah paling awal yang dilakukan pada tahap praproses. 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 Deteksi Tepi

Kode sumber diatas merupakan deteksi tepi dengan menggunakan metode *Canny* yang telah dimodifikasi. Pada baris 1, parameter *image* adalah citra mata yang hendak dilakukan deteksi tepi. Sedangkan *sigma* merupakan standar deviasi untuk filter Gaussian untuk dihaluskan. Parameter *scaling* digunakan untuk mengubah ukuran citra dengan tujuan untuk mempercepat proses. Kemudian dua parameter terakhir merupakan bobot untuk gradien deteksi tepi pada arah vertikal dan horizontal. Pada baris ke-3 dan ke-4 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*).

4.3.2 Implementasi Deteksi Batas Pupil dan Iris

Sub bab ini membahas lanjutan implementasi tahap praproses 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 lingkaran pupil seperti yang ditunjukkan pada Kode Sumber 4.3 dan Kode

Sumber 4.4 secara berturut-turut dengan radius yang sudah diinisialisasi pada baris ke-2 sampai ke-4.

1.	% menentukan radius pupil & iris
2.	lpupilradius = 28;
2.	upupilradius = 75;
3.	lirisradius = 80;
4.	uirisradius = 150;
5.	
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.	
13.	% menentukan panjang&lebar ROI iris
14.	irl = double(round(rowd-rd));
15.	iru = double(round(rowd+rd));
16.	icl = double(round(cold-rd));
17.	icu = double(round(cold+rd));
18.	

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. Dengan menggunakan parameter *scaling* untuk mengubah ukuran citra deteksi tepi menjadi lebih kecil dengan tujuan untuk mempercepat pemrosesan transformasi Hough. Pada baris ke-8 sampai dengan baris ke-10 adalah transformasi Hough yang diimplementasikan dengan fungsi *findcircle*. Hasilnya dari transformasi Hough seperti yang terlihat pada baris ke-11. Kemudian menentukan *region of interest* iris untuk digunakan sebagai 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 dan 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 seperti yang terlihat pada baris ke-14, yakni 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.
14. % radius dan titik pusat pupil
15. 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.

4.3.3 Implementasi Pemisahan Bulu dan Kelopak Mata

Pemisahan bulu dan kelopak mata dilakukan karena kedua hal tersebut adalah *noise*. Hal yang dilakukan pertama kali adalah menghilangkan kelompok mata. Implementasinya 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 4 sampai 10. 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 sampai ujung citra ditandai dengan NaN karena dianggap sebagai *noise*.

```

1. % citra mata
2. imagewithnoise = double(eyeimage);
2.
3. % membuat garis pd bulu mata dgn hough
4. % linear
5. topeyelid = imagepupil(1:(rowp-r),:);
6. lines = findline(topeyelid);
7.
8. if size(lines,1) > 0
9.     [xl yl] =
10.    linecoords(lines,size(topeyelid));
11.    yl = double(yl) + irl-1;
12.    xl = double(xl) + icl-1;
13.
14. % menentukan batas garis
15.    yla = max(yl);
16.    y2 = 1:yla;
17.
18. % menandai kelopak(noise) dengan NaN
19.    imagewithnoise(y2, xl) = NaN;
20. end
21.

```

Kode Sumber 4.5 Implementasi Penghilangan 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 dibawah 100 dengan notasi NaN. Bulu mata yang terdapat pada citra kebanyakan memiliki nilai intensitas dibawah 100. Proses *thresholding* pada citra mata dilakukan

pada baris ke-2, kemudian 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. Dipilih NaN karena intensitas tersebut memberikan warna hitam sebagai visualisasi.

4.4 Implementasi Normalisasi Iris

Sub bab ini membahas implementasi tahap normalisasi iris. Tujuan utama melakukan tahap ini adalah untuk merepresentasikan citra iris ke dalam bentuk yang lebih baik. Dikarenakan citra mata setiap orang pada saat pengambilan citra untuk dimasukkan dalam database dan ketika dites bisa saja mengalami pergeseran ataupun perbesaran pupil. Masukan dari tahap ini adalah citra mata yang sudah melalui tahap praproses, yang dimana 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, seperti yang sudah dijelaskan pada bab sebelumnya. 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 18 sampai 23, yakni dengan mengganti nilai piksel *noise* dengan nilai rata-rata pada citra iris polar. Sehingga pada tahap selanjutnya semua nilai piksel *noise* sudah mempunyai nilai. Variabel `polar_array` merupakan variabel yang berisi hasil dari tahap normalisasi citra mata deraunya sudah diganti nilai intensitasnya, sehingga bisa 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.
17. % ganti nilai NaN dgn rata2 piksel citra
18. coords = find(isnan(polar_array));
19. polar_array2 = polar_array;
20. polar_array2(coords) = 0.5;
21. avg = sum(sum(polar_array2)) / ...
22.     (size(polar_array,1)*size(polar_array,2))
23. polar_array(coords) = avg;
24.
25.

```

Kode Sumber 4.7 Implementasi Transformasi Polar

Hasil akhir dari tahap normalisasi iris ini adalah citra yang berbentuk persegi panjang dengan ukuran 64x512 piksel. Didalamnya adalah tekstur iris yang sudah melalui tahap-tahap sebelumnya.

4.5 Implementasi Ekstraksi Fitur

Ekstraksi fitur merupakan tahap untuk pengambilan ciri dari citra iris normalisasi. Dalam tahap ini, digunakan dua buah metode ekstraksi fitur yang tidak saling berkaitan. Metode yang digunakan untuk mengekstrak citra iris normalisasi adalah *wavelet Haar* dan *log-Gabor filter*. Kedua implementasi ini akan dijelaskan pada sub bab 4.5.1 dan 4.5.2.

4.5.1 Implementasi Fitur Koefisien Wavelet Haar

Sub bab ini membahas implementasi tahap ekstraksi fitur. Ekstraksi fitur dilakukan dengan mengubah citra iris normalisasi kedalam domain *wavelet*. Pada implementasi, citra iris ternormalisasi akan dilakukan proses dekomposisi. Proses dekomposisi ini membuat dimensi citra iris ternormalisasi akan tereduksi menjadi setengahnya. Implementasi penggunaan *wavelet Haar* pada citra iris ternormalisasi dapat dilihat pada Kode Sumber 4.8.

1.	<code>% inisialisasi awal</code>
2.	<code>wname = 'haar';</code>
3.	<code>sizeEkstrak = [8 64];</code>
4.	
5.	<code>% proses wavelet haar dekomposisi lvl 3</code>
6.	<code>[C,~] = wavedec2(polar_array,level,wname);</code>
7.	<code>HasilWavelet = C(1:(sizeEkstrak(1) * ...</code>
	<code>sizeEkstrak(2)));</code>
8.	

Kode Sumber 4.8 Implementasi Ekstraksi Fitur *Wavelet Haar*

Inisialisasi penggunaan jenis *wavelet* ditunjukan pada baris 2, yang menunjukan bahwa proses dekomposisi *wavelet* ini menggunakan *wavelet Haar*. Diinisialisasikan juga ukuran untuk hasil dari dekomposisi *wavelet* pada baris 3. Kemudian fungsi `wavedec2` digunakan untuk melakukan proses dekomposisi pada `polar_array` (citra normalisasi). Fungsi ini langsung mendekomposisi citra ke tingkat tertentu sesuai dengan nilai `level`, sebagai ganti dari iterasi fungsi `dwt2`. Hasil dekomposisi yang diambil adalah nilai koefisien aproksimasi yang ditunjukan pada baris 7 pada variabel `HasilWavelet`.

4.5.2 Implementasi Fitur Log-Gabor Filter

Sub bab ini membahas implementasi tahap fitur bilangan biner. Tahap ini menggunakan ekstraksi fitur *log-Gabor filter*. Masukan dari tahap ini adalah citra iris ternormalisasi yang telah didapatkan dari tahap normalisasi iris.

Pada tahap ini, masukannya untuk ekstraksi fitur adalah citra iris ternormalisasi yang nantinya akan direpresentasikan kedalam bentuk bilangan biner. Implementasi pengubahan citra iris menjadi citra dengan karakteristik biner dapat dilihat pada Kode Sumber 4.9.

```

1.  % konvulsi citra iris dgn filter
2.  [E0] = gaborconvolve(polar_array, ...
3.      minWaveLength, sigmaOnf);
4.
5.  % kuantisasi nilai ke 0 atau 1
6.  E0 = E0{1};
7.  H1 = real(E0) > 0;
8.
9.  % rows dan cols untuk proses konversi
10. nrows = 1:size(polar_array,1);
11. length = size(polar_array,2);
12.
13. % iris template ( var store hasil konversi)
14. templateTest = zeros(size(polar_array));
15.
16. % konversi nilai piksel ke biner
17. for i=0:(length-2)
18.     ja = double((i));
19.     templateTest(nrows,ja+(2)-1)=H1(nrows,i+1);
20. end

```

Kode Sumber 4.9 Implementasi Pembuatan Fitur Biner

Kode Sumber 4.9 melakukan pengubahan citra iris ternormalisasi menjadi citra fitur biner. Hasil konvulsi dari citra iris ternormalisasi dengan filter gabor ditunjukkan pada baris 2, dengan isi fungsi `gaborconvolve` ditunjukkan pada Kode Sumber 4.10. Kemudian hasil dari konvulsi yang mengandung komponen bilangan real akan dikuantisasi untuk menghasilkan bilangan biner. Terakhir, hasil dari pengkonversian akan disimpan pada variabel `templateTest`.

1.	<code>ndata = size(polar_array,2);</code>
2.	<code>matTemp = [0:fix(ndata/2)];</code>
3.	<code>radius = matTemp/fix(ndata/2)/2;</code>
4.	
5.	<code>% pembuatan filter gabor dgn wavelength = 18</code>
6.	<code>% dan sigma = 0.5</code>
7.	<code>filterlogGabor = zeros(1, ndata);</code>
8.	<code>fo = 1.0/wavelength;</code>
9.	
10.	<code>% persamaan pembuatan filter gabor</code>
11.	<code>filterlogGabor(1:size(radius,2)) = ...</code>
12.	<code>exp((-log(radius/fo)).^2) / ...</code>
13.	<code>(2 * log(sigmaOnf)^2));</code>
14.	

Kode Sumber 4.10 Implementasi Pembuatan Filter Log-Gabor

Pada Kode Sumber 4.10, digunakan parameter `wavelength` yang mengatur frekuensi yang dipakai dan `sigmaOnf` untuk menentukan filter *bandwidth*. Semakin kecil nilai `sigmaOnf`, maka filter *bandwidth* akan semakin besar. Kemudian dengan menggunakan parameter-parameter tersebut membentuk filter gabor. Hasil filter gabor ini membentuk matriks dengan ukuran satu dimensi.

1.	<code>% konvulsi citra normalisasi dgn filter</code>
2.	<code>for r = 1:size(polar_array,1)</code>
3.	<code> signal = polar_array(r,1:ndata);</code>
4.	<code> imagefft = fft(signal);</code>
5.	<code> result(r,:) = ifft(imagefft .* filter);</code>
6.	<code>end</code>
7.	
8.	<code>EO = result;</code>
9.	

Kode Sumber 4.11 Implementasi Konvulsi Citra Iris dengan Filter Log-Gabor

Pada Kode Sumber 4.11 merupakan proses konvulsi citra iris ternormalisasi dengan filter gabor yang sudah terbentuk. Proses konvulsi ini dilakukan dengan melakukan iterasi pada tiap baris citra iris ternormalisasi dengan filter gabor. Namun filter gabor merupakan filter yang berada dalam

domain frekuensi, sehingga citra iris ternormalisasi perlu diubah pula kedalam domain frekuensi. Pengubahan ini menggunakan fungsi pada Matlab `fft` seperti pada baris 4. Hasil dari proses konversi pada domain frekuensi akan dikembalikan ke domain spasial dengan fungsi pada Matlab `ifft` seperti pada baris 5. Hasil akhirnya disimpan dalam variabel `EO`, lalu dikembalikan pada baris 2 Kode Sumber 4.9.

4.6 Implementasi Klasifikasi

Sub bab ini membahas implementasi tahap klasifikasi. Dalam tahap ini, digunakan dua buah metode klasifikasi yang tidak saling berkaitan seperti yang dijelaskan pada sub bab 3.2. Hasil dari masing-masing ekstraksi fitur akan digunakan sebagai masukan untuk klasifikasi. Hasil ekstraksi fitur *wavelet Haar* dan ekstraksi fitur *log-Gabor filter* akan digunakan sebagai masukan untuk metode klasifikasi *Support Vector Machine* dan *Hamming distance*. Implementasi keduanya dapat dilihat pada sub bab ini.

4.6.1 Implementasi Support Vector Machine Fitur Wavelet

Kode Sumber 4.12 merupakan implementasi klasifikasi *Support Vector Machine*. Sebelum masuk kedalam proses klasifikasi, data citra iris sudah dilakukan ekstraksi fitur dan dibagi menjadi data latih & data uji menggunakan *wavelet Haar* dan disimpan dalam file `dataset_wavelet_haar.mat` yang dapat dilihat pada baris 2. Sehingga dapat dengan mudah diakses isinya. Didalam file tersebut terdapat variabel `TrainingSet` yang dapat dilihat pada baris 3 dan variabel `GroupTrain` pada baris 4. Kedua variabel tersebut adalah data latih dan labelnya yang digunakan sebagai masukan tahap klasifikasi ini.

Sebelum melakukan klasifikasi, data latih dan labelnya terlebih dahulu diproses sebagai model untuk klasifikasi seperti yang terlihat pada baris 7 sampai 11. Pembuatan model ini

bertujuan sebagai representasi tiap kelas. Sehingga akhirnya akan ada model sebanyak jumlah kelas.

```

1. % me-load data latih (hasil haar wavelet)
2. foo = load('dataset_wavelet_haar.mat');
3. TrainingData = foo.TrainingSet;
4. trainLabel = foo.GroupTrain;
5.
6. % train model
7. model = cell(108,1);
8. for k=1:108
9.     model{k} = svmtrain(double(trainLabel==k)
10.        TrainingData, '-c 1 -b 1 -q');
11. end
12.
13. % prediksi kelas
14. probKelas = zeros(1,108);
15. for k=1:108
16.     [~,~,p] = svmpredict(~, HasilWavelet, ...
17.        model{k}, '-b 1 -q');
18.     probKelas(:,k) = p(:,model{k}.Label==1);
19. end
20.
21. % penentuan kelas
22. [~,hasilKelas] = max(probKelas,[],2);
23.

```

Kode Sumber 4.12 Implementasi SVM Fitur Wavelet

Pada baris 14 sampai 23 dilakukan iterasi sebanyak jumlah model yang sudah dibentuk. Sebuah data uji hasilWavelet akan dihitung kemiripannya pada setiap model dengan menggunakan probabilitas. Sehingga pada akhirnya terdapat matriks yang berisi probabilitas pada setiap model seperti yang ditunjukkan pada baris 18. Hasil yang dikeluarkan dari proses ini adalah indeks model dengan nilai terbesar dari matriks tersebut, yang juga dapat dikatakan hasil prediksi kelasnya.

4.6.2 Implementasi Suport Vector Machin Fitur Log-Gabor

Kode Sumber 4.13 merupakan implementasi klasifikasi *Support Vector Machine*. Sebelum masuk kedalam proses klasifikasi, data citra iris sudah dilakukan ekstraksi fitur dan dibagi menjadi data latih & data uji menggunakan *log-Gabor filter* dan disimpan dalam file `dataset_logGabor.mat` yang dapat dilihat pada baris 2. Sehingga dapat dengan mudah diakses isinya. Didalam file tersebut terdapat variabel `TrainingSet` yang dapat dilihat pada baris 3 dan variabel `GroupTrain` pada baris 4. Kedua variabel tersebut adalah data latih dan labelnya yang digunakan sebagai masukan tahap klasifikasi ini.

```

1.  % me-load data latih (hasil haar wavelet)
2.  foo = load('dataset_logGabor.mat');
3.  TrainingData = foo.TrainingSet;
4.  trainLabel = foo.GroupTrain;
5.
6.  % train model
7.  model = cell(108,1);
8.  for k=1:108
9.      model{k} = svmtrain(double(trainLabel==k)
10.         TrainingData, '-c 1 -b 1 -q');
11. end
12.
13. % prediksi kelas
14. probKelas = zeros(1,108);
15. for k=1:108
16.     [~,~,p] = svmpredict(~, templateTest, ...
17.         model{k}, '-b 1 -q');
18.     probKelas(:,k) = p(:,model{k}.Label==1);
19. end
20.
21. % penentuan kelas
22. [~,hasilKelas] = max(probKelas,[],2);
23.
24.

```

Kode Sumber 4.13 Implementasi SVM Fitur Log-Gabor

Sebelum melakukan klasifikasi, data latih dan labelnya terlebih dahulu diproses sebagai model untuk klasifikasi seperti yang terlihat pada baris 7 sampai 11. Pembuatan model ini bertujuan sebagai representasi tiap kelas. Sehingga akhirnya akan ada model sebanyak jumlah kelas.

Pada baris 14 sampai 23 dilakukan iterasi sebanyak jumlah model yang sudah dibentuk. Sebuah data uji `templateTest` akan dihitung kemiripannya pada setiap model dengan menggunakan probabilitas. Sehingga pada akhirnya terdapat matriks yang berisi probabilitas pada setiap model seperti yang ditunjukkan pada baris 18. Hasil yang dikeluarkan dari proses ini adalah indeks model dengan nilai terbesar dari matriks tersebut, yang juga dapat dikatakan hasil prediksi kelasnya.

4.6.3 Implementasi Hamming Distance Fitur Log-Gabor

Kode Sumber 4.14 merupakan implementasi klasifikasi *Hamming distance*. Sebelum masuk kedalam proses perhitungan jarak, data citra iris sudah dilakukan ekstraksi fitur menggunakan *log-Gabor filter* dan disimpan dalam file `dataset_logGabor.mat` yang dapat dilihat pada baris 2. Didalam file tersebut terdapat variabel `TrainingSet` yang dapat dilihat pada baris 3, yang merupakan kumpulan data latih data citra.

Pada baris 5 terdapat variabel `templateTest` yang merupakan sebuah data uji untuk melakukan klasifikasi. Selanjutnya melakukan proses perhitungan jarak. Karena data latih berjumlah empat dalam setiap kelas, maka data uji akan dihitung jaraknya terhadap keempat data latih tersebut menggunakan fungsi `zHamming` seperti penjelasan subbab 2.10, kemudian dihitung rata-ratanya. Hal tersebut dilakukan iterasi sebanyak sejumlah kelas yang ditunjukkan pada baris 9 sampai 18. Hasil perhitungan rata-rata tersebut disimpan dalam variabel `avgDist` yang merupakan matriks. Hasil yang dikeluarkan dari proses ini adalah indeks kelas dengan nilai

terkecil dari matriks tersebut, yang juga dapat dikatakan hasil prediksi kelasnya.

```

1.  % me-load data latih (hasil log-Gabor)
2.  foo = load('dataset_logGabor.mat');
3.  TempTrainingData = foo.TrainingSet;
4.
5.  % variable masukan untuk test
6.  [templateTest] = template;
7.
8.  % perhitungan jarak
9.  for h =1:108      % iterasi sejumlah kelas
10.     distance1 = zHamming(templateTest , ...
11.        TempTrainingData{1,1} );
12.     distance2 = zHamming(templateTest , ...
13.        TempTrainingData{2,1} );
14.     distance3 = zHamming(templateTest , ...
15.        TempTrainingData{3,1} );
16.     distance4 = zHamming(templateTest , ...
17.        TempTrainingData{4,1} );
18.
19. % penyimpanan matriks avg dari tiap kelas
20. avg=distance1+distance2+distance3+distance4;
21. avgDist = [avgDist , avg];
22. end
23.
24. % penentuan kelas
25. [~,hasilKelas] = min(avgDist,[],2);

```

Kode Sumber 4.14 Implementasi Hamming Distance Fitur Log-Gabor

4.6.4 Implementasi Hamming Distance Fitur Wavelet

Subbab ini merupakan implementasi klasifikasi *Hamming distance*. Sebelum masuk kedalam proses perhitungan jarak, data citra iris sudah dilakukan ekstraksi fitur menggunakan *wavelet Haar* dan disimpan dalam file `dataset_wavelet.mat` yang ditunjukkan pada baris 2. Didalam file tersebut terdapat variabel `wavelet_all` yang dapat dilihat pada baris 3, yang merupakan kumpulan dataset seluruh citra

mata yang sudah dilakukan proses dekomposisi wavelet yang ditunjukkan pada Kode Sumber 4.15.

```

1. % normalisasi data wavelet
2. foo = load('dataset_wavelet.mat');
3. AllCitra = foo.wavelet_all;
4.
5. for i = 1:size(AllCitra,2)
6.     ALLCitra(:,i) = ,...
7.         ( AllCitra(:,i) - min(AllCitra(:,i)))./
8.         ( max(AllCitra(:,i))-min(AllCitra(:,i)))
9. );
10. end
11.
12. AllCitra(AllCitra >= 0.5) = 1;
13. AllCitra(AllCitra < 0.5) = 0;
14.

```

Kode Sumber 4.15 Implementasi Proses Normalisasi Data

Proses normalisasi tersebut dilakukan dengan tujuan untuk mengubah nilai pada seluruh elemen matrik hasil dekomposisi *wavelet* agar bernilai antara rentang 0 sampai 1. Hasil akhirnya adalah keseluruhan data yang sudah dinormalisasi dan tersimpan pada variabel `AllCitra`. Agar menjadi sebuah matrik dengan elemen bilangan biner, dilakukan pengubahan jika intensitas matriks lebih dari 0.5, maka intensitasnya menjadi 1, sebaliknya jika kurang dari 0.5, intensitasnya menjadi 0, seperti yang ditunjukkan pada baris 11 sampai 12. Selanjutnya dengan pengopeasian biasa dilakukan pembagian data uji dan data latih dan disimpan dalam sebuah file ekstensi `mat`.

Setelah dibagi dan ditentukan data uji dan data testing, kemudian melakukan pencarian jarak dengan menggunakan Hamming distance seperti yang ditunjukkan pada Kode Sumber 4.16.

Masukan pada Kode Sumber 4.16 adalah sebuah file `dataset_wavelet_norm.mat` yang berisi data latih dan data uji dari proses normalisasi data dekomposisi *wavelet Haar*. Pada

baris 2, file tersebut di-*load* untuk dibaca dan diolah isi dalamnya.

```

1.  % me-load data latih (hasil log-Gabor)
2.  foo = load('dataset_wavelet_norm.mat');
3.  TempTrainingData = foo.TrainingSet;
4.
5.  % variable masukan untuk test
6.  [templateTest] = templateTest;
7.
8.  % perhitungan jarak
9.  for h =1:108      % iterasi sejumlah kelas
10.     distance1 = zHamming(templateTest , ...
11.        TempTrainingData{1,1} );
12.     distance2 = zHamming(templateTest , ...
13.        TempTrainingData{2,1} );
14.     distance3 = zHamming(templateTest , ...
15.        TempTrainingData{3,1} );
16.     distance4 = zHamming(templateTest , ...
17.        TempTrainingData{4,1} );
18.
19. % penyimpanan matriks avg dari tiap kelas
20. avg=distance1+distance2+distance3+distance4;
21. avgDist = [avgDist , avg];
22. end
23.
24. % penentuan kelas
25. [~,hasilKelas] = min(avgDist,[],2);

```

Kode Sumber 4.16 Implementasi Hamming Distance Fitur Wavelet

Pada baris 6 terdapat variabel `templateTest` yang merupakan sebuah data uji untuk melakukan klasifikasi. Selanjutnya melakukan proses perhitungan jarak. Karena data latih berjumlah empat dalam setiap kelas, maka data uji akan dihitung jaraknya terhadap keempat data latih tersebut menggunakan fungsi `zHamming` seperti penjelasan subbab 2.10, kemudian dihitung rata-ratanya. Hal tersebut dilakukan iterasi sebanyak sejumlah kelas yang ditunjukkan pada baris 9 sampai 18. Hasil perhitungan rata-rata tersebut disimpan dalam

variabel `avgDist`. Hasil yang dikeluarkan dari proses ini adalah indeks kelas dengan nilai terkecil dari matriks `avgDist` tersebut, yang juga dapat dikatakan hasil prediksi kelasnya.

4.7 Implementasi Antarmuka

Bagian ini adalah bagian sistem yang memungkinkan user untuk berinteraksi dengan sistem pengenalan iris mata ini. Antarmuka didesain semudah dan sesederhana mungkin untuk digunakan. Gambar 4.1 menunjukkan desain antarmuka sistem pengenalan iris ini. Dalam rancangan antarmuka ini terdapat satu kotak gambar untuk input citra, dua kotak gambar untuk hasil segmentasi citra, dua kotak teks hasil inpuan, dan enam tombol pada bagian kanan sebagai menu utama.

Kotak gambar citra input akan menunjukkan gambar yang dipilih dari tombol menu `Pilih Gambar`. Nama dari gambar yang dipilih akan ditampilkan pada kotak teks `Filecitra`. Kemudian dua kotak gambar segmentasi akan menghasilkan hasil dari segmentasi mata dari tombol menu `Segmentasi`.



Gambar 4.1 Implementasi Antarmuka

Tombol menu *Matching (SVM)* merupakan tombol menu untuk melakukan pencocokan dengan menggunakan klasifikasi *Support Vector Machine*, sedangkan tombol menu *Matching (Hamming)* untuk melakukan pencocokan menggunakan klasifikasi *Hamming distance*. Hasilnya akan ditampilkan pada kotak teks Hasil klasifikasi.

Tombol menu *Quit* digunakan untuk keluar dari antarmuka dan tombol menu *Reset* digunakan untuk mendefaultkan ke tampilan awal.

BAB V

UJI COBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

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

Tabel 5.1 Lingkungan Uji Coba Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i5-33170U CPU @ 1.70GHz 1.70GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows 8 64-bit Perangkat Pengembang: Matlab 8.3 & <i>Image Processing Toolbox</i> Perangkat Pembantu: Microsoft Excel 2013, <i>libsvm</i>

5.2 Data Uji Coba

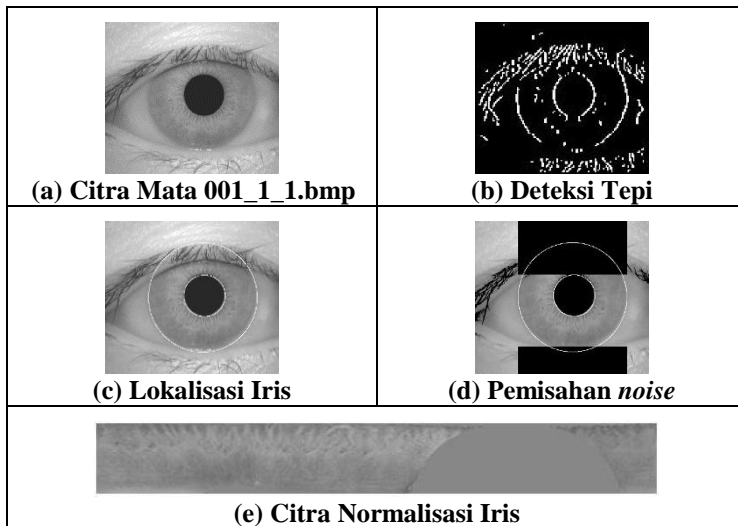
Data yang digunakan untuk uji coba implementasi pengenalan iris mata ini adalah data citra mata yang bersumber dari database citra mata *Chinese Academy of Science-Institute of Automation (CASIA)* versi 1.0. Pada database ini berisi 756 citra mata dengan 108 mata yang berbeda yang masing-masing diambil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

Contoh format nama file citra mata pada database CASIA adalah 001_1_1.bmp. Pada format file tersebut, 001 menunjukkan mata/kelas orang pertama, angka 1 dikanannya menunjukkan mata yang diambil pada sesi pertama, dan angka 1 yang terakhir menunjukkan mata yang diambil pada urutan ke-1.

Dalam melakukan uji coba, citra mata perlu dibagi ke dalam data uji dan data latih. Pengambilan citra mata pada sesi pertama akan dijadikan sebagai data uji, sedangkan pada sesi kedua akan dijadikan sebagai data latih. Sehingga jumlah data uji adalah $3 \times 108 = 324$ buah dan jumlah data latih adalah $4 \times 108 = 432$ buah.

5.3 Preprocessing

Pada tahap preprocessing, citra mata akan diolah untuk mendapatkan bagian irisnya, contoh sampel ditunjukkan pada Gambar 5.1.





Gambar 5.1 Sampel hasil praproses pada citra mata 001_1_1.bmp

Gambar 5.1 merupakan sampel gambar dari hasil *preprocessing* citra mata yang terdiri dari proses deteksi tepi, deteksi batas pupil dan iris, serta pemisahan bulu dan kelopak mata, juga proses normalisasi. Gambar (a) merupakan citra mata yang digunakan sebagai masukan, gambar (b) merupakan hasil dari proses deteksi tepi, gambar (c) adalah pengalokasian bagian iris dengan mencari batas lingkaran pupil dan iris, kemudian *noise* yang didalam bagian iris dipisahkan pada gambar (d), dan gambar (e) merupakan tahap normalisasi ukuran citra iris yang sudah tidak mempunyai *noise* di bagian iris.

5.4 Ekstraksi Fitur

Pada tahap ekstraksi fitur, citra normalisasi akan diproses menggunakan dua buah metode, yakni *wavelet Haar* dan *log-gabor filter*. Sampel dari kedua metode tersebut dapat dilihat pada Tabel 5.2.

Tabel 5.2 Sampel fitur pada citra mata 001_1_1.bmp

<i>Wavelet Haar</i>	<i>Log-Gabor Filter</i>
	
(a) Fitur Aproksimasi Haar	(b) Fitur biner

Tabel 5.2 merupakan sampel gambar dari hasil ekstraksi fitur citra iris normalisasi. Gambar (a) merupakan fitur *wavelet Haar* bidang aproksimasi pada dekomposisi level 2, sedangkan gambar (b) merupakan fitur biner hasil dari *log-Gabor filter*.

5.5 Skenario Uji Coba

Pada sub bab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan menggunakan data uji dan data latih. Melalui skenario ini, implementasi algoritma ekstraksi fitur dan klasifikasi akan diuji bagaimana performa pada masing-masing

skenario. Telah dilakukan beberapa skenario uji coba, diantaranya yaitu:

1. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *wavelet Haar* dan klasifikasi *Support Vector Machine* berdasarkan level dekomposisi *wavelet*. Level dekomposisi yang diujikan adalah 1, 2, 3, dan 4.
2. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *wavelet Haar* dan klasifikasi *Support Vector Machine* berdasarkan variasi nilai C (*penalty error term*) pada klasifikasi. Nilai C yang akan diuji yaitu 1, 10, 20, 30, dan 40.
3. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *wavelet Haar* dan klasifikasi *Support Vector Machine* berdasarkan variasi kernel yang digunakan pada klasifikasi. Kernel yang akan diuji yaitu linear, RBF, polynomial 2, dan polynomial 3.
4. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *log-Gabor filter* dan klasifikasi *Support Vector Machine* berdasarkan standar deviasi ekstraksi fitur. Besaran standar deviasi antara lain 0.2, 0.4, 0.6, dan 0.8.
5. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *log-Gabor filter* dan klasifikasi *Support Vector Machine* berdasarkan variasi nilai C (*penalty error term*) pada klasifikasi. Nilai C yang akan diuji yaitu 1, 10, 20, 30, dan 40.
6. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *log-Gabor filter* dan klasifikasi *Support Vector Machine* berdasarkan variasi kernel yang digunakan pada klasifikasi. Kernel yang akan diuji yaitu linear, RBF, polynomial 2, dan polynomial 3.
7. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *wavelet Haar* dan metode *Hamming distance* berdasarkan level dekomposisi *wavelet*. Level dekomposisi yang diujikan adalah 1, 2, 3, dan 4.

8. Perhitungan performa hasil akurasi menggunakan ekstraksi fitur *log-Gabor filter* dan metode *Hamming distance* berdasarkan standar deviasi ekstraksi fitur. Besaran standar deviasi antara lain 0.2, 0.4, 0.6, dan 0.8.

Perhitungan akurasi dilakukan dengan cara menghitung jumlah data uji yang benar sesuai dengan label kelas sesungguhnya kemudian dibagi dengan total data uji seperti yang ditunjukkan persamaan (5.1).

$$Akurasi = \frac{\sum \text{data uji benar}}{\text{jumlah data uji}} \quad (5.1)$$

5.5.1 Skenario Uji Coba 1

Skenario uji coba 1 adalah perhitungan akurasi dan waktu eksekusi. Uji coba ini dicoba dengan menggunakan beberapa level dekomposisi *wavetlet Haar* pada proses ekstraksi fitur dan SVM sebagai *classifier*. Level dekomposisi *wavelet* yang di uji coba yaitu 1, 2, 3 dan 4. Bidang aproksimasi akan digunakan sebagai fitur karena memuat energi atau informasi yang lebih banyak dari bidang horizontal, vertikal, ataupun diagonal.

Tabel 5.3 Persentase Akurasi dan Waktu Eksekusi masing-masing level dekomposisi *wavelet Haar* (SVM)

Level dekomposisi	Akurasi (%)	Waktu Eksekusi (s)
1	91.98	165.36
2	91.98	39.82
3	89.81	8.60
4	85.19	1.84

Setiap level akan menghasilkan jumlah fitur yang berbeda. Jumlah fitur pada level 1 menghasilkan 8192 fitur,

level 2 akan menghasilkan 2048 fitur, level 3 menghasilkan 512 fitur dan level 4 menghasilkan 128 fitur. Pada skenario 1, jenis *wavelet* yang digunakan adalah *Haar*, sedangkan parameter nilai C adalah 1 menggunakan kernel RBF. Hasil performa dan waktu eksekusi masing-masing dekomposisi dapat dilihat pada Tabel 5.3.

Berdasarkan hasil performa pada Tabel 5.3., dekomposisi *wavelet Haar* pada level 1 dan level 2 mempunyai akurasi tertinggi yaitu 91.98%, dibandingkan level 3 dan level 4 yang mencapai 89.81% dan 85.19% secara berturut-turut. Namun pada level dekomposisi level 1 dan level 2 memiliki hasil akurasi yang sama, tetapi waktu akurasi yang berbeda. Hal ini dikarenakan dimensi citra tereduksi setiap level dekomposisi yang mengakibatkan pula waktu komputasi menjadi lebih cepat saat masuk ke tahap klasifikasi. Dari uji coba tersebut didapatkan dekomposisi *wavelet Haar* level 2 sebagai yang terbaik. Hasil setiap data uji yang pada dekomposisi level 2 terdapat pada lampiran.

5.5.2 Skenario Uji Coba 2

Skenario uji coba 2 bertujuan menghitung nilai akurasi pada variasi nilai C pada klasifikasi, dimana nilai C adalah nilai *penalty error term*. Nilai C yang diuji yaitu 1, 10, 20, dan 40. Fitur yang digunakan adalah hasil dekomposisi *wavelet Haar* level 2 dan menggunakan kernel RBF sebagai klasifikasi.

Tabel 5.4 Persentase Akurasi fitur *wavelet* menggunakan variasi nilai C (SVM)

Level wavelet	C	Akurasi (%)	Waktu Eksekusi (s)
2	1	91.98	39.82
	10	91.98	38.24
	20	91.98	39.02
	30	92.28	39.66
	40	91.98	39.71

Pada Tabel 5.4 Hasil uji coba 2 hasil terbaik didapatkan ketika nilai C adalah 30, dimana memiliki akurasi yang lebih tinggi dari yang lainnya. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.5.3 Skenario Uji Coba 3

Pada skenario uji 3 merupakan uji coba menggunakan variasi kernel SVM untuk melihat persentase akurasi. Variasi kernel yang digunakan yaitu linear, *polynomial 2*, *polynomial 3*, dan RBF. Fitur yang digunakan adalah hasil dekomposisi *wavelet Haar* level 2 dan nilai C diberikan adalah 30.

Tabel 5.5 Persentase Akurasi fitur *wavelet* menggunakan variasi kernel SVM

Level wavelet	C	Kernel	Akurasi (%)	Waktu Eksekusi (s)
2	30	Linear	91.98	38.56
		Polynomial 2	91.98	39.89
		Polynomial 3	91.67	40.23
		RBF	92.28	39.66

Berdasarkan Tabel 5.5, hasil performa yang diperlihatkan diatas, akurasi tertinggi didapatkan menggunakan kernel RBF yaitu 92.28%. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.5.4 Skenario Uji Coba 4

Pada skenario uji coba 4 akan dilakukan percobaan untuk menghitung akurasi dan waktu eksekusi menggunakan fitur *log-Gabor filter* dan klasifikasi SVM. Pada uji coba ini akan dilakukan percobaan pada variasi nilai standar deviasi yang terdapat pada ekstraksi fitur *log-Gabor filter*. Standar deviasi yang akan digunakan adalah 0.2, 0.4, 0.6, dan 0.8. Hasil perhitungan performa tiap nilai standar deviasi dapat dilihat

pada Tabel 5.6. Pada skenario 1, parameter nilai C yang digunakan adalah 1 dan menggunakan kernel RBF.

Tabel 5.6 Persentase Akurasi dan Waktu Eksekusi fitur biner menggunakan variasi standar deviasi (SVM)

Standar deviasi	Akurasi (%)	Waktu Eksekusi (s)
0.2	89.51	45.96
0.4	81.48	46.77
0.6	73.15	45.90
0.8	55.86	47.45

Berdasarkan Tabel 5.6, hasil yang ditunjukkan oleh Tabel 5.6 nilai simpangan 0.2 memiliki hasil akurasi paling tinggi yaitu 89.51%. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.5.5 Skenario Uji Coba 5

Skenario uji coba 5 dilakukan dengan menghitung nilai akurasi jika menggunakan variasi nilai C pada klasifikasi SVM. Nilai C yang diuji yaitu 1, 10, 20, dan 40. Fitur yang digunakan adalah hasil *log-Gabor filter* pada saat standar deviasi 0.2 dan menggunakan kernel RBF sebagai klasifikasi.

Tabel 5.7 Persentase Akurasi fitur biner menggunakan variasi nilai C (SVM)

Standar deviasi	C	Akurasi (%)	Waktu Eksekusi (s)
0.2	1	89.51	45.96
	10	88.58	45.12
	20	88.27	46.31
	30	88.27	47.95
	40	88.27	45.34

Berdasarkan hasil yang ditunjukkan oleh Tabel 5.7, akurasi tertinggi didapat ketika nilai C adalah 1 yaitu 89.51%.

5.5.6 Skenario Uji Coba 6

Uji coba 6 adalah percobaan menggunakan variasi kernel SVM untuk menghitung persentase akurasi. Variasi kernel yang digunakan yaitu linear, *polynomial 2*, *polynomial 3*, dan RBF. Fitur yang digunakan adalah hasil *log-Gabor filter* dengan standar deviasi 0.2 dan nilai C diberikan adalah 1. Hasil perhitungan akurasi variasi kernel dapat dilihat pada Tabel 5.8.

Berdasarkan hasil yang ditunjukkan, kernel RBF memiliki hasil akurasi paling tinggi yaitu 89.51%.

Tabel 5.8 Persentase Akurasi fitur biner menggunakan variasi kernel SVM

Standar deviasi	C	Kernel	Akurasi (%)	Waktu Eksekusi (s)
0.2	1	Linear	87.96	47.12
		Polynomial 2	88.89	46.61
		Polynomial 3	89.20	45.89
		RBF	89.51	45.96

5.5.7 Skenario Uji Coba 7

Skenario uji coba 7 adalah perhitungan akurasi dan waktu eksekusi. Uji coba ini dicoba dengan menggunakan beberapa level dekomposisi *wavetlet Haar* pada proses ekstraksi fitur dan *Hamming distance* sebagai metode untuk pengklasifikasiannya. Level dekomposisi *wavelet* yang di uji coba yaitu 1, 2, 3 dan 4. Bidang aproksimasi akan digunakan sebagai fitur karena memuat energi atau informasi yang lebih banyak dari bidang horizontal, vertikal, ataupun diagonal. Hasil percobaan ditunjukkan pada Tabel 5.9.

Tabel 5.9 Persentase Akurasi dan Waktu Eksekusi masing-masing level dekomposisi *wavelet Haar* (*Hamming distance*)

Level dekomposisi	Akurasi (%)	Waktu Eksekusi (s)
1	83.02	311.15
2	78.09	76.74
3	75.00	17.88
4	64.81	2.52

Dari uji coba 7, hasil terbaik didapatkan pada dekomposisi *wavelet Haar* level 1, dimana memiliki akurasi yang lebih tinggi dari yang lainnya. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.5.8 Skenario Uji Coba 8

Pada uji skenario yang terakhir akan dilakukan percobaan untuk melihat akurasi dan waktu eksekusi menggunakan fitur *log-Gabor filter* dan metode *Hamming distance*. Pada uji coba ini akan dilakukan percobaan pada variasi nilai standar deviasi yang terdapat pada ekstraksi fitur *log-Gabor filter*. Standar deviasi yang akan digunakan adalah 0.2, 0.4, 0.6, dan 0.8. Hasil perhitungan performa tiap nilai standar deviasi dapat dilihat pada Tabel 5.10.

Tabel 5.10 Persentase Akurasi dan Waktu Eksekusi fitur biner menggunakan variasi standar deviasi (*Hamming distance*)

Standar deviasi	Akurasi (%)	Waktu Eksekusi (s)
0.2	91.67	279.52
0.4	86.73	280.32
0.6	82.41	278.93
0.8	76.23	281.84

Berdasarkan hasil yang ditunjukkan oleh Tabel 5.10 nilai simpangan 0.2 memiliki hasil akurasi paling tinggi yaitu 91.67%. Hasil setiap data uji dengan parameter terbaik tersebut terdapat pada lampiran.

5.6 Analisis Hasil Uji Coba

Berdasarkan skenario-skenario uji coba yang telah dilakukan, dapat diketahui bahwa tingkat ketepatan untuk mengenali iris mata sudah sangat baik. Uji coba 1 sampai uji coba 8 merupakan kombinasi uji coba untuk melihat performa algoritma ekstraksi fitur (*wavelet Haar* dan *log-Gabor filter*) dan algoritma klasifikasi (*support vector machine* dan *Hamming distance*) untuk melakukan pengenalan iris mata.

Dari hasil skenario uji coba yang telah dilakukan, beberapa parameter memberikan pengaruh terhadap hasil akurasi. Pada ekstraksi fitur *wavelet Haar*, parameter yang berpengaruh adalah tingkatan level dekomposisinya, sedangkan pada ekstraksi fitur *log-Gabor filter* parameter yang berpengaruh adalah standar deviasinya. Kemudian pada klasifikasi SVM, parameter yang berpengaruh adalah nilai C . Kemudian uji coba dilakukan dengan membandingkan akurasi.

Pada skenario ke-1 didapat bahwa level dekomposisi yang menghasilkan akurasi paling baik yaitu *wavelet Haar* level 2. Selanjutnya level dekomposisi tersebut akan diuji pada skenario ke-2 yang menggunakan variasi nilai C pada SVM untuk pengujiannya dan hasil akurasi menjadi naik saat nilai C bernilai 30. Skenario ke-3 dengan menggunakan level dekomposisi *wavelet Haar* level 2 dan nilai $C=30$, diuji coba menggunakan variasi kernel dan mendapatkan akurasi yang sama seperti skenario sebelumnya yaitu 92.28%.

Skenario ke-4 adalah uji coba dengan menggunakan parameter standar deviasi pada ekstraksi fitur *log-Gabor filter* dan menggunakan klasifikasi *Hamming distance*. Hasilnya baik dan mendapatkan akurasi sebesar 91.67%.

Skenario ke-5 merupakan uji coba dengan menggunakan variasi level dekomposisi *wavelet Haar* dan metode *Hamming distance* sebagai klasifikasi. Hasil terbaik yang didapatkan adalah menggunakan level dekomposisi pada level 1 yaitu 83.02%.

Para skenario ke-6 sampai dengan ke-8 merupakan uji coba menggunakan variasi standar deviasi pada ekstraksi *log-Gabor filter*, variasi nilai C , dan variasi kernel SVM. Hasil akurasi terbaik yang didapatkan adalah 89.51% dengan standar deviasi adalah 0.2, nilai C adalah 1, dan kernel RBF.

Perbandingan akurasi dua metode klasifikasi sudah terlihat cukup efektif untuk dapat melakukan pengenalan orang melalui iris mata. Perbandingan dua metode klasifikasi ini dapat dilihat pada Tabel 5.11.

Tabel 5.11 Perbandingan Akurasi Metode Klasifikasi

Ekstraksi Fitur	Klasifikasi	Akurasi (%)
Wavelet Haar	SVM	92.28
Wavelet Haar	Hamming distance	83.02
Log-Gabor Filter	SVM	89.51
Log-Gabor Filter	Hamming distance	91.67

Berdasarkan Tabel 5.11, klasifikasi *Support Vector Machine* dengan menggunakan *Wavelet Haar* dan pencocokan *Hamming distance* dengan menggunakan *log-Gabor filter* memberikan hasil yang cukup baik untuk mengenali iris.

LAMPIRAN A

TABEL UJI COBA

Tabel A.1 Hasil Uji Coba Data Uji Citra Iris Fitur *Wavelet Haar* Dekomposisi level 2 dan Klasifikasi SVM dengan $C = 1$ dan kernel RBF

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	2
3	3	3
4	3	3
5	3	3
6	3	2
7	3	3
8	3	3
9	3	3
10	3	3
11	3	2
12	3	3
13	3	3
14	3	3
15	3	3
16	3	3
17	3	3
18	3	3
19	3	1
20	3	3
21	3	3
22	3	3

23	3	3
24	3	3
25	3	3
26	3	3
27	3	3
28	3	3
29	3	3
30	3	3
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	3
39	3	3
40	3	3
41	3	2
42	3	3
43	3	3
44	3	3
45	3	2
46	3	2
47	3	3
48	3	2
49	3	3
50	3	3
51	3	3
52	3	3

53	3	3
54	3	3
55	3	3
56	3	2
57	3	3
58	3	1
59	3	3
60	3	3
61	3	3
62	3	3
63	3	3
64	3	3
65	3	3
66	3	2
67	3	2
68	3	3
69	3	3
70	3	3
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	3
79	3	2
80	3	3
81	3	2
82	3	3

83	3	3
84	3	3
85	3	3
86	3	2
87	3	3
88	3	3
89	3	3
90	3	3
91	3	2
92	3	2
93	3	3
94	3	2
95	3	3
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	3
102	3	3
103	3	3
104	3	0
105	3	2
106	3	2
107	3	3
108	3	3
JUMLAH	324	298
PERSENTASE		91.98%

Tabel A.2 Hasil Uji Coba Data Uji Citra Iris Fitur *Wavelet Haar* Dekomposisi level 2 dan Klasifikasi SVM di $C = 30$ dan kernel RBF

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	2
3	3	3
4	3	3
5	3	3
6	3	2
7	3	3
8	3	3
9	3	3
10	3	3
11	3	2
12	3	3
13	3	3
14	3	3
15	3	3
16	3	3
17	3	3
18	3	3
19	3	1
20	3	3
21	3	3
22	3	3
23	3	3
24	3	3

25	3	3
26	3	3
27	3	3
28	3	3
29	3	3
30	3	3
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	3
39	3	3
40	3	3
41	3	2
42	3	3
43	3	3
44	3	3
45	3	2
46	3	2
47	3	3
48	3	2
49	3	3
50	3	3
51	3	3
52	3	3
53	3	3
54	3	3

55	3	3
56	3	2
57	3	3
58	3	1
59	3	3
60	3	3
61	3	3
62	3	3
63	3	3
64	3	3
65	3	3
66	3	2
67	3	2
68	3	3
69	3	3
70	3	3
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	3
79	3	2
80	3	3
81	3	2
82	3	3
83	3	3
84	3	3

85	3	3
86	3	3
87	3	3
88	3	3
89	3	3
90	3	3
91	3	2
92	3	2
93	3	3
94	3	2
95	3	3
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	3
102	3	3
103	3	3
104	3	0
105	3	2
106	3	2
107	3	3
108	3	3
JUMLAH	324	299
PERSentase		92.28%

Tabel A.3 Hasil Uji Coba Data Uji Citra Iris Fitur *Wavelet Haar* Dekomposisi level 2 dan Klasifikasi SVM pada $C = 30$ dan kernel RBF

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	2
3	3	3
4	3	3
5	3	3
6	3	2
7	3	3
8	3	3
9	3	3
10	3	3
11	3	2
12	3	3
13	3	3
14	3	3
15	3	3
16	3	3
17	3	3
18	3	3
19	3	1
20	3	3
21	3	3
22	3	3
23	3	3
24	3	3
25	3	3

26	3	3
27	3	3
28	3	3
29	3	3
30	3	3
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	3
39	3	3
40	3	3
41	3	2
42	3	3
43	3	3
44	3	3
45	3	2
46	3	2
47	3	3
48	3	2
49	3	3
50	3	3
51	3	3
52	3	3
53	3	3
54	3	3
55	3	3

56	3	2
57	3	3
58	3	1
59	3	3
60	3	3
61	3	3
62	3	3
63	3	3
64	3	3
65	3	3
66	3	2
67	3	2
68	3	3
69	3	3
70	3	3
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	3
79	3	2
80	3	3
81	3	2
82	3	3
83	3	3
84	3	3
85	3	3

86	3	3
87	3	3
88	3	3
89	3	3
90	3	3
91	3	2
92	3	2
93	3	3
94	3	2
95	3	3
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	3
102	3	3
103	3	3
104	3	0
105	3	2
106	3	2
107	3	3
108	3	3
JUMLAH	324	299
PERSENTASE		92.28%

Tabel A.4 Hasil Uji Coba Data Uji Citra Iris Fitur *Log-Gabor Filter* standar deviasi 0.2 dan Pencocokan *Hamming distance*

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	3
3	3	3
4	3	3
5	3	3
6	3	3
7	3	3
8	3	3
9	3	2
10	3	3
11	3	3
12	3	3
13	3	3
14	3	3
15	3	3
16	3	3
17	3	3
18	3	2
19	3	3
20	3	3
21	3	3
22	3	3
23	3	3
24	3	0
25	3	2
26	3	3

27	3	3
28	3	3
29	3	3
30	3	2
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	3
39	3	3
40	3	3
41	3	3
42	3	3
43	3	2
44	3	3
45	3	2
46	3	2
47	3	3
48	3	3
49	3	3
50	3	2
51	3	3
52	3	3
53	3	3
54	3	3
55	3	3
56	3	3

57	3	3
58	3	1
59	3	3
60	3	3
61	3	3
62	3	3
63	3	3
64	3	3
65	3	3
66	3	3
67	3	2
68	3	3
69	3	3
70	3	3
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	3
79	3	2
80	3	3
81	3	2
82	3	3
83	3	3
84	3	3
85	3	2
86	3	3

87	3	3
88	3	2
89	3	3
90	3	3
91	3	2
92	3	2
93	3	3
94	3	2
95	3	3
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	3
102	3	2
103	3	3
104	3	2
105	3	1
106	3	2
107	3	3
108	3	3
JUMLAH	324	297
PERSENTASE		91.67%

Tabel A.5 Hasil Uji Coba Data Uji Citra Iris Fitur *Wavelet Haar* Dekomposisi level 1 dan Pencocokan *Hamming distance*

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	2
3	3	3
4	3	3
5	3	3
6	3	3
7	3	2
8	3	3
9	3	3
10	3	3
11	3	3
12	3	3
13	3	2
14	3	3
15	3	2
16	3	3
17	3	3
18	3	2
19	3	1
20	3	3
21	3	3
22	3	3
23	3	3
24	3	3
25	3	3
26	3	3

27	3	3
28	3	3
29	3	3
30	3	2
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	3
39	3	3
40	3	3
41	3	2
42	3	2
43	3	3
44	3	3
45	3	2
46	3	2
47	3	3
48	3	1
49	3	1
50	3	3
51	3	3
52	3	1
53	3	3
54	3	3
55	3	0
56	3	3

57	3	3
58	3	1
59	3	3
60	3	3
61	3	2
62	3	3
63	3	3
64	3	3
65	3	3
66	3	2
67	3	2
68	3	0
69	3	3
70	3	0
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	0
79	3	1
80	3	3
81	3	2
82	3	3
83	3	3
84	3	3
85	3	3
86	3	2

87	3	3
88	3	2
89	3	3
90	3	3
91	3	1
92	3	2
93	3	3
94	3	3
95	3	2
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	2
102	3	3
103	3	2
104	3	0
105	3	2
106	3	2
107	3	0
108	3	3
JUMLAH	324	269
PERSENTASE		83.02%

Tabel A.6 Hasil Uji Coba Data Uji Citra Iris Fitur *Log-Gabor Filter* standar deviasi 0.2 dan Klasifikasi SVM pada $C = 1$ dan kernel RBF

No Subyek	UJI CASIA IRIS DATABASE V.1	
	Data Uji	Data Uji Benar
1	3	3
2	3	2
3	3	3
4	3	3
5	3	3
6	3	2
7	3	3
8	3	3
9	3	3
10	3	3
11	3	3
12	3	3
13	3	3
14	3	3
15	3	3
16	3	2
17	3	3
18	3	2
19	3	1
20	3	3
21	3	3
22	3	3
23	3	3
24	3	2
25	3	1

26	3	3
27	3	3
28	3	3
29	3	3
30	3	3
31	3	3
32	3	3
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3
38	3	2
39	3	3
40	3	3
41	3	2
42	3	3
43	3	3
44	3	3
45	3	2
46	3	3
47	3	3
48	3	3
49	3	3
50	3	2
51	3	3
52	3	3
53	3	3
54	3	3
55	3	3

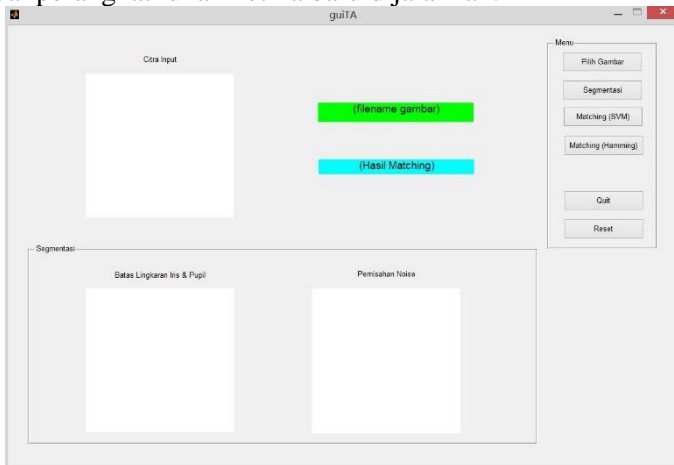
56	3	3
57	3	3
58	3	1
59	3	3
60	3	3
61	3	3
62	3	3
63	3	3
64	3	3
65	3	3
66	3	3
67	3	2
68	3	3
69	3	3
70	3	3
71	3	3
72	3	3
73	3	2
74	3	3
75	3	3
76	3	3
77	3	3
78	3	3
79	3	2
80	3	3
81	3	2
82	3	3
83	3	3
84	3	3
85	3	2

86	3	3
87	3	3
88	3	2
89	3	3
90	3	2
91	3	0
92	3	2
93	3	3
94	3	2
95	3	3
96	3	3
97	3	3
98	3	3
99	3	3
100	3	3
101	3	0
102	3	2
103	3	3
104	3	3
105	3	1
106	3	2
107	3	3
108	3	3
JUMLAH	324	290
PERSENTASE		89.51%

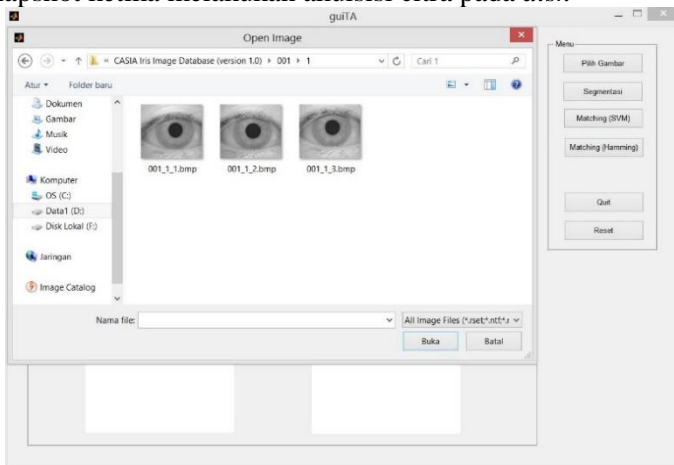
LAMPIRAN B

SNAPSHOT PERANGKAT LUNAK

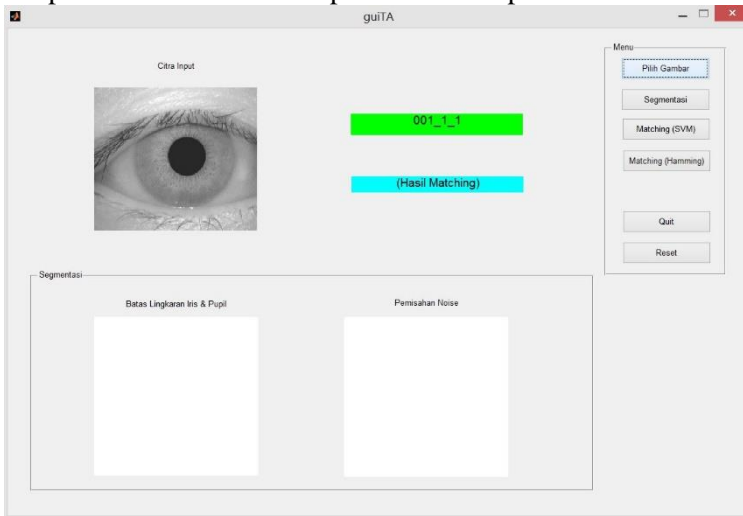
Pada lampiran ini akan ditampilkan snapshot perangkat lunak yang sudah diimplementasikan. Berikut adalah snapshot awal perangkat lunak ketika baru dijalankan.



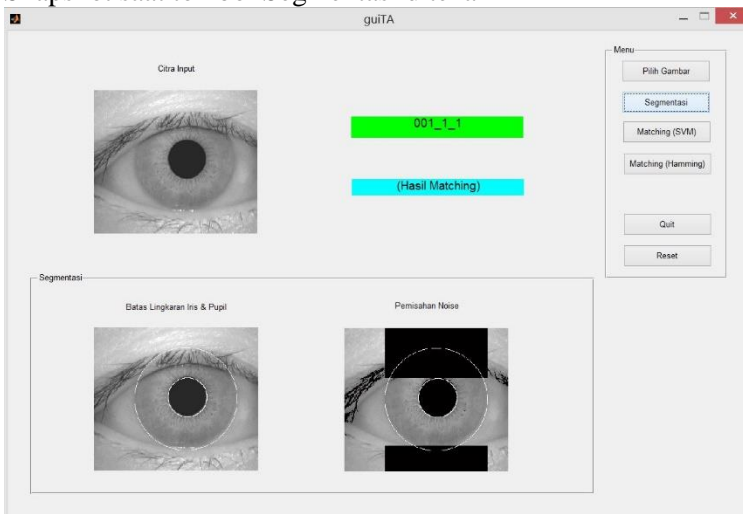
Snapshot ketika melakukan akuisisi citra pada *disk*



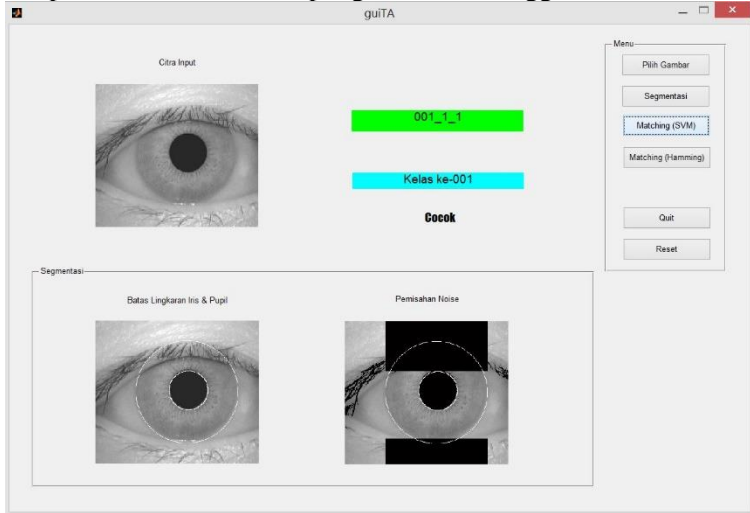
Snapshot saat citra sudah dipilih dan ditampilkan



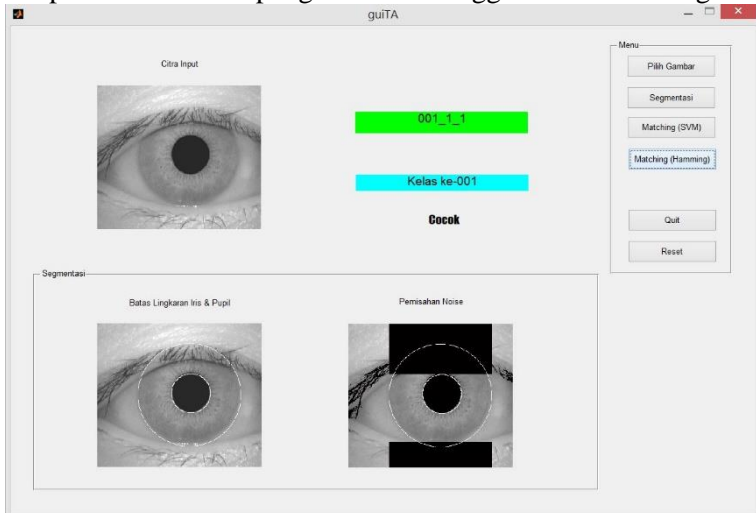
Snapshot saat tombol Segmentasi ditekan



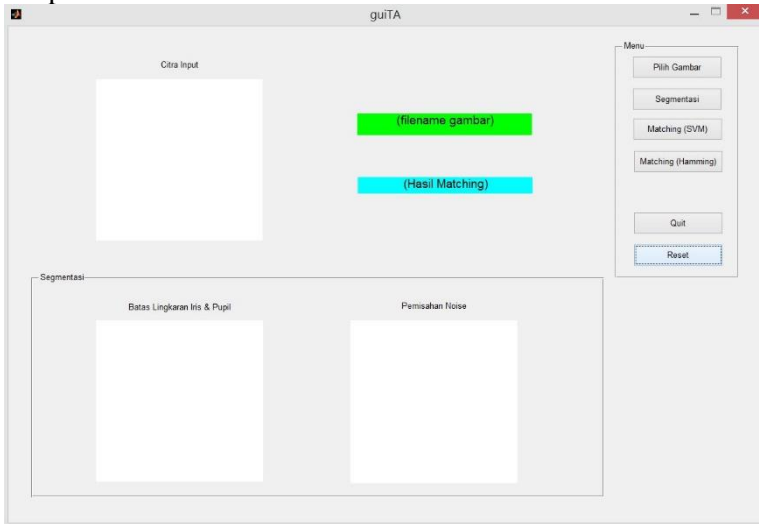
Snapshot saat melakukan pengklasifian menggunakan SVM



Snapshot melakukan pengklasifian menggunakan Hamming



Snapshot ketika menekan tombol Reset



Snapshot saat hendak menekan tombol Quit dan keluar dari antarmuka

