

**IMPLEMENTASI METODE *SUPPORT VECTOR MACHINE*  
UNTUK IDENTIFIKASI PENYAKIT DAUN TANAMAN  
KUBIS (STUDI KASUS: TEMU TANI MALANG)**

**SKRIPSI**

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

**Oleh:**

**AYUNDHA WULAN KURNIAWATI     NIM. 1341180153**



**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
AGUSTUS 2017**

**IMPLEMENTASI METODE *SUPPORT VECTOR MACHINE*  
UNTUK IDENTIFIKASI PENYAKIT DAUN TANAMAN  
KUBIS (STUDI KASUS: TEMU TANI MALANG)**

**SKRIPSI**

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

**Oleh:**

**AYUNDHA WULAN KURNIAWATI    NIM. 1341180153**



**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
AGUSTUS 2017**

## HALAMAN PENGESAHAN

### IMPLEMENTASI METODE *SUPPORT VECTOR MACHINE* UNTUK IDENTIFIKASI PENYAKIT DAUN TANAMAN KUBIS (STUDI KASUS: TEMU TANI MALANG)

Disusun oleh:  
**AYUNDHA WULAN KURNIAWATI      NIM. 1341180153**

**Skripsi ini telah diuji pada tanggal 29 Agustus 2017**

**Disetujui oleh:**

- |                  |   |   |       |
|------------------|---|---|-------|
| 1. Penguji I     | : | <u>Yan Watequlis Syaifudin, S.T., M.MT.</u>   | ..... |
|                  |   | NIP. 19810105 200501 1 005                    |       |
| 2. Penguji II    | : | <u>Dr.Eng. Rosa Andrie Asmara, ST, MT</u>     | ..... |
|                  |   | NIP. 19801010 200501 1 001                    |       |
| 3. Pembimbing I  | : | <u>Ariadi Retno Tri Hayati Ririd, S.Kom.,</u> | ..... |
|                  |   | <u>M.Kom.</u>                                 |       |
|                  |   | NIP. 19810810 200501 2 002                    |       |
| 4. Pembimbing II | : | <u>Yoppy Yunhasnawa, S.ST, M.Sc.</u>          | ..... |
|                  |   | NIP.  |       |

Mengetahui,

Ketua Jurusan  
Teknologi Informasi

Ketua Program Studi  
Teknik Informatika

Rudy Ariyanto, S.T., M.Cs.  
NIP. 19711110 199903 1 002

Ir. Deddy Kusbianto P., M.MKom.  
NIP. 19621128 198811 1 001

## **PERNYATAAN**

Dengan ini saya menyatakan bahwa Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Malang,   Agustus 2017

Ayundha Wulan Kurniawati

## ABSTRAK

**Kurniawati, Ayundha Wulan.** “Implementasi Metode *Support Vector Machine* Untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)”. **Pembimbing: (1) Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom. (2) Yoppy Yunhasnawa, S.ST, M.Sc.**

**Skripsi, Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2017.**

Tanaman kubis merupakan salah satu sayuran yang banyak dikonsumsi masyarakat, dalam produksi bibit tanaman kubis sering mengalami hambatan karena serangan hama. Salah satu komponen dalam keberhasilan produksi kubis adalah masa perkembangan bibit, yang dikhawatirkan banyak mendapat serangan hama. Dalam penelitian ini pengolahan citra digital digunakan untuk mengidentifikasi hama/penyakit terhadap bibit tanaman kubis.

Penelitian ini dimulai dengan pengumpulan citra daun tanaman kubis. Tahapan selanjutnya adalah *preprocessing* citra dengan menghilangkan *background* dari citra masukan kemudian dilakukan proses *grayscale* untuk mendapatkan nilai yang akan digunakan untuk proses selanjutnya. Hasil tersebut kemudian akan dihitung dengan menggunakan metode *Support Vector Machine* (SVM). Proses *training* dilakukan dengan *Sequential Training* yang kemudian dilakukan proses *testing*. Hasil klasifikasi dengan segmentasi yang menghitung nilai *foreground* dan *background* citra masukan menghasilkan akurasi sebesar 75%, sedangkan untuk klasifikasi dengan segmentasi yang hanya menghitung nilai *foreground* menghasilkan akurasi sebesar 69.44% dan penggunaan *Histogram Equalization* untuk memperbaiki citra sehingga dapat tersegmentasi dengan baik menghasilkan akurasi sebesar 80.55%.

**Kata Kunci:** *Support Vector Machine*, Pengolahan Citra, Daun Kubis, Klasifikasi

## **ABSTRACT**

**Kurniawati, Ayundha Wulan.** *“Implementation of Support Vector Machine Method for Identification of Leaf Disease of Cabbage (Case Study : Temu Tani Malang)”*. **Advisors: (1) Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom., (2) Yoppy Yunhasnawa, S.ST, M.Sc.**

**Thesis, Informatics Engineering Study Program, Department of Information Technology, State Polytechnic of Malang, 2017.**

*Cabbage is one of the vegetable that widely consumes by the people, which is the seed production were oftenly attack by disease. One of the component for the successful of cabbage production is the development of the seeds, which is easily infected by the pest attack. In this study, digital image processing apply to identify the pest attack or disease against the cabbage plant seed.*

*This research begins with gathering images of cabbage leaf. The next step is image preprocessing for image segmentation, so that can be processed on the next stage. The result of image segmentation of the cabbage leaf were classified using Support Vector Machine (SVM) methods. The training process is done by Sequential Training which carried out the testing process. The result of the classification with the segmentation that calculated the foreground and background values of the input image reaches the accuracy up to 75%, while for classification with the segmentation that only calculate the foreground value reaches the accuracy up to 69.44% and the use of Histogram Equalization to improve the image input so that can properly segmented reaches the accuracy up to 80.55%.*

**Keywords:** *Support Vector Machine, Image Processing, Cabbage Leaf, Classification*

## KATA PENGANTAR

Puji syukur kehadiran Tuhan Yang Maha Esa penulis ucapkan atas segala limpahan rahmat, taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan penyusunan Proposal Akhir yang berjudul “Implementasi Metode Support Vector Machine Untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)”.

Oleh karena itu penulis mengucapkan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa yang telah memberikan petunjuk dan hidayahnya dalam penyusunan Skripsi ini sehingga dapat berjalan dengan baik.
2. Ibu, Ayah, adik dan keluarga yang telah banyak memberikan dukungan baik secara moral maupun material sehingga dalam penyusunan skripsi ini dapat berjalan dengan lancar dan sesuai dengan harapan penulis.
3. Ibu Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom. dan Bapak Yoppy Yunhasnawa, S.ST, M.Sc. selaku dosen pembimbing skripsi, yang telah memberikan waktu, kesempatan, petunjuk dan bimbingan.
4. Dosen-dosen pengajar Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang yang telah memberikan bimbingan dan ilmunya.
5. Teman-teman dekat penulis, Gadis, Nisa, Dhike, Pipik, Elly, Dyan dan teman-teman lainnya terimakasih atas dukungannya.
6. Teman-teman Teknik Informatika angkatan 2013 dan seluruh pihak yang telah membantu dan mendukung lancarnya penyusunan dan penyelesaian Skripsi dari awal hingga akhir.

Penulis sadar bahwa hasil pengerjaan skripsi dan laporan ini masih jauh dari sempurna. Karenanya, segala kritik dan saran yang membangun sangat penulis harapkan. Terima kasih.

Malang, Agustus 2017

Penulis

## DAFTAR ISI

	Halaman
SAMPUL DEPAN .....	i
HALAMAN JUDUL .....	ii
HALAMAN PENGESAHAN .....	iii
PERNYATAAN .....	iv
ABSTRAK .....	v
<i>ABSTRACT</i> .....	vi
KATA PENGANTAR .....	vii
DAFTAR ISI .....	viii
DAFTAR GAMBAR .....	x
DAFTAR TABEL .....	xi
DAFTAR LAMPIRAN .....	xii
BAB I. PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	3
1.3. Tujuan .....	3
1.4. Batasan Masalah .....	3
1.5. Sistematika Penulisan .....	3
BAB II. LANDASAN TEORI .....	5
2.1. Tanaman Kubis .....	5
2.2.1. Jenis Kubis .....	6
2.2.2. Nilai Gizi dan Manfaat .....	8
2.2.3. Penyakit .....	9
2.2. Klasifikasi .....	10
2.3. Citra Digital .....	10
2.3.1. Citra Biner .....	11
2.3.2. Citra <i>Grayscale</i> .....	11
2.3.3. Citra Warna .....	11
2.3.4. Pengolahan Citra Digital .....	12
2.4. <i>Thresholding</i> .....	12
2.5. Normalisasi .....	13
2.6. <i>Support Vector Machine</i> (SVM) .....	14
2.6.1. Metode Sequential Training SVM .....	15
2.7. Evaluasi .....	17
BAB III. METODOLOGI PENELITIAN .....	19
3.1. Studi Literatur .....	19
3.2. Analisis Kebutuhan Sistem .....	20
3.3. Pengumpulan Data .....	20
3.4. Perancangan Sistem .....	20
3.5. Implementasi .....	23
3.6. Pengujian .....	24
BAB IV. ANALISIS DAN PERANCANGAN .....	25
4.1. Analisis Sistem .....	25
4.1.1. Kebutuhan Perangkat Lunak .....	25
4.1.2. Kebutuhan Perangkat Keras .....	25



4.2. Perancangan Sistem.....	25
4.2.1. Diagram Alir Sistem.....	26
4.2.2. Proses <i>Histogram Equalization</i> .....	27
4.2.3. Proses Segmentasi.....	27
4.2.4. Proses <i>Grayscale</i> .....	28
4.2.5. Proses Perhitungan <i>Training SVM</i> .....	30
4.2.6. Proses Perhitungan Kernel RBF.....	32
4.2.7. Proses Perhitungan Matriks Hessian.....	33
4.2.8. Proses Perhitungan <i>MaxGamma</i> .....	34
4.2.9. Proses Perhitungan <i>maxPositiveAlpha</i> dan <i>maxNegativeAlpha</i> .....	34
4.2.10. Proses Perhitungan Data <i>Testing SVM</i> .....	36
4.2.11. Proses Perhitungan Manual Kernel RBF.....	37
4.2.12. Proses Perhitungan Manual <i>Training SVM</i> .....	38
4.2.13. Proses Perhitungan Manual <i>Testing SVM</i> .....	42
4.3. Perancangan Antarmuka ( <i>Interface</i> ).....	43
4.4. Perancangan Pengujian dan Analisis.....	47
4.5. Pengambilan Keputusan.....	49
BAB V. IMPLEMENTASI.....	50
5.1. Implementasi Antarmuka.....	50
5.1.1. Implementasi antarmuka <i>Training</i> .....	50
5.1.2. Implementasi antarmuka <i>Testing</i> .....	51
5.1.3. Implementasi antarmuka <i>Help</i> .....	52
5.1.4. Implementasi antarmuka <i>About</i> .....	52
5.2. Implementasi Sistem.....	53
5.2.1. Implementasi <i>Histogram Equalization</i> .....	53
5.2.2. Implementasi Segmentasi.....	55
5.2.2. Implementasi <i>Grayscale</i> .....	55
5.2.3. Implementasi Perhitungan Kernel RBF.....	56
5.2.4. Implementasi Proses <i>Training SVM</i> .....	57
5.2.5. Implementasi Proses <i>Testing SVM</i> .....	58
5.3. Klasifikasi.....	58
BAB VI. PENGUJIAN DAN PEMBAHASAN.....	61
6.1. Pengujian.....	61
6.1.1. Pengujian <i>Blackbox</i> .....	61
6.1.2. Pengujian Akurasi.....	62
6.2. Analisa.....	70
BAB VII. KESIMPULAN.....	76
7.1. Kesimpulan.....	76
7.2. Saran.....	77
DAFTAR PUSTAKA.....	78
LAMPIRAN.....	79

## DAFTAR GAMBAR

	Halaman
Gambar 2.1 Kubis .....	6
Gambar 2.2 Kubis Bunga.....	7
Gambar 2.3 Brokoli.....	7
Gambar 2.4 Kubis Umbi .....	8
Gambar 2.5 Daun tanaman kubis terserang Bercak Daun .....	9
Gambar 2.6 Daun tanaman kubis terserang Busuk Hitam .....	10
Gambar 2.7 Citra Biner .....	11
Gambar 2.8 Citra Grayscale .....	11
Gambar 2.9 Citra Warna 4 bit .....	12
Gambar 2.10 Citra Warna 3 bit .....	12
Gambar 3.1 Diagram Blok Metodologi Penelitian .....	19
Gambar 3.2 Diagram Blok Identifikasi Penyakit Daun Tanaman Kubis .....	21
Gambar 3.3 Diagram Blok Proses Training SVM .....	22
Gambar 3.4 Diagram Blok Proses Pengujian SVM .....	23
Gambar 4.1 Diagram Alir Sistem .....	26
Gambar 4.2 Diagram Alir <i>Histogram Equalization</i> .....	27
Gambar 4.3 Diagram Alir Thresholding .....	28
Gambar 4.4 Diagram Alir Grayscale .....	29
Gambar 4.5 Diagram Alir Perhitungan Training SVM.....	31
Gambar 4.6 Diagram Alir Perhitungan Kernel RBF.....	33
Gambar 4.7 Diagram Alir Perhitungan Matriks Hessien .....	34
Gambar 4.8 Diagram Alir Perhitungan MaxGamma .....	34
Gambar 4.9 Diagram Alir Perhitungan maxPositiveAlpha .....	35
Gambar 4.10 Diagram Alir Perhitungan Data Testing SVM.....	36
Gambar 4.11 Perancangan Antarmuka Training.....	44
Gambar 4.12 Perancangan Antarmuka Testing .....	45
Gambar 4.13 Perancangan Antarmuka Help.....	46
Gambar 4.14 Perancangan Antarmuka About .....	47
Gambar 5.1 Implementasi Antarmuka Training .....	50
Gambar 5.2 Implementasi Antarmuka Testing .....	51
Gambar 5.3 Implementasi Antarmuka Help .....	52
Gambar 5.4 Implementasi Antarmuka About .....	53
Gambar 5.5 Sourcecode implementasi Histogram Equalization.....	54
Gambar 5.6 Sourcecode implementasi Threshold .....	55
Gambar 5.7 Sourcecode Implementasi Grayscale .....	56
Gambar 5.8 Sourcecode Implementasi Kernel RBF .....	57
Gambar 5.9 Sourcecode Implementasi Training SVM .....	57
Gambar 5.10 Sourcecode Implementasi Testing SVM.....	58
Gambar 5.11 Langkah <i>preprocessing</i> pada aplikasi .....	59
Gambar 5.12 Langkah <i>training</i> pada aplikasi .....	59
Gambar 5.12 Proses <i>training</i> pada aplikasi .....	60
Gambar 6.1 Hasil pengujian dengan proses segmentasi yang berbeda.....	74

## DAFTAR TABEL

	Halaman
Tabel 2.1 Kedudukan tanaman kubis dalam sistematika botani .....	5
Tabel 2.2 Confusion Matrix .....	17
Tabel 4.1 Dataset.....	37
Tabel 4.2 Hasil Perhitungan Kernel RBF .....	37
Tabel 4.3 Hasil Perhitungan Matriks Hessian.....	38
Tabel 4.4 Hasil Perhitungan Nilai Error .....	39
Tabel 4.5 Hasil Perhitungan Nilai Error .....	40
Tabel 4.6 Hasil perhitungan nilai $Kx, x +$ dan nilai $Kx, x -$ .....	41
Tabel 4.7 Hasil perhitungan $w.x^+$ dan $w.x^-$ .....	41
Tabel 4.8 Data Testing .....	42
Tabel 4.9 Hasil Perhitungan Nilai $K(x, x_i)$ dan $w$ untuk data testing .....	43
Tabel 4.10 Hasil Klasifikasi .....	43
Tabel 6.1 Pengujian Blackbox .....	61
Tabel 6.2 Hasil Klasifikasi dengan Preprocessing Grayscale 1 dan 2 .....	63
Tabel 6.3 Hasil Pengujian dengan Preprocessing Grayscale 1 .....	64
Tabel 6.4 Hasil Pengujian dengan Preprocessing Grayscale 2 .....	64
Tabel 6.5 Hasil Klasifikasi dengan Preprocessing Histogram Equalization.....	65
Tabel 6.6 Hasil Pengujian dengan Preprocessing Histogram Equalization .....	66
Tabel 6.7 Pengujian pada nilai Sigma.....	67
Tabel 6.8 Pengujian pada nilai $\Lambda$ .....	67
Tabel 6.9 Pengujian pada nilai $\Gamma$ .....	68
Tabel 6.10 Pengujian pada nilai $Complexity$ .....	69
Tabel 6.11 Pengujian pada nilai Epsilon.....	69
Tabel 6.12 Pengujian pada nilai Iterasi Maksimal .....	70
Tabel 6.13 Parameter yang digunakan .....	70
Tabel 6.14 <i>Confusion Matrix</i> hasil pengujian .....	71

## DAFTAR LAMPIRAN

- Lampiran 1. Dataset *Testing*
- Lampiran 2. Hasil Pengujian *Preprocessing Grayscale* 1 dan 2
- Lampiran 3. Hasil Pengujian *Preprocessing Histogram Equalization*
- Lampiran 4. Kuisioner
- Lampiran 5. Lembar Bimbingan Skripsi Pembimbing I
- Lampiran 6. Lembar Bimbingan Skripsi Pembimbing II
- Lampiran 7. Lembar Persetujuan Maju Ujian Skripsi
- Lampiran 8. Lembar Revisi Penguji I
- Lampiran 9. Lembar Revisi Penguji II
- Lampiran 10. Lembar Verifikasi Abstrak dan Tata Tulis

# BAB I. PENDAHULUAN

## 1.1. Latar Belakang

Bercak daun dan busuk hitam merupakan penyakit yang menyerang daun tanaman kubis dan menimbulkan kerugian sehingga produksi kubis mengalami penurunan. Identifikasi penyakit daun tanaman kubis dilakukan secara manual dengan mengamati gejala yang tampak pada daun. Penyakit bercak daun ditandai dengan munculnya gejala bercak-bercak bulat coklat dan lingkaran konsentris. Penyakit busuk hitam ditandai dengan munculnya bercak kuning di sepanjang tepi daun yang mengarah ke tengah daun.

Dalam proses pengendalian hama dan penyakit tanaman diperlukan kegiatan pemantauan untuk memantau perkembangan dari bibit tanaman kubis serta untuk memantau penyakit dan hama yang menyerang sehingga diperlukan kemampuan untuk mengidentifikasi penyakit. Proses identifikasi penyakit dilakukan secara manual dengan cara pengamatan visual secara langsung pada daun tanaman kubis.

Pengolahan citra digital dapat digunakan untuk mengidentifikasi penyakit daun tanaman kubis. Karena tidak semua petani memiliki pemahaman detail mengenai penyakit yang menyerang tanaman dan petani tidak perlu menghafal gejala penyakit yang merugikan dan tidak merugikan.

Sistem dapat mengenali penyakit daun dengan menggunakan citra digital dari daun tanaman kubis yang berpenyakit. Data citra digital daun tanaman kubis digunakan untuk melakukan proses pelatihan bagi sistem sehingga dapat mengenali penyakit, hasil dari proses pelatihan dapat digunakan untuk mengidentifikasi penyakit pada data uji.

Untuk mengidentifikasi penyakit diperlukan proses klasifikasi. Klasifikasi merupakan salah satu cara untuk pengelompokan objek terhadap sebuah kelas tertentu. Terdapat beberapa metode yang digunakan untuk klasifikasi, di antaranya adalah *Naive Bayes*, *K-Nearest Neighbor (KNN)*, *Support Vector Machine (SVM)*, *Artificial Neural Network (ANN)* [1]. Pengolahan data citra digital daun tanaman kubis diperlukan untuk proses klasifikasi.

Penelitian terhadap daun tanaman kubis pernah dilakukan oleh Faradina Vidyani (2013) dengan menggunakan Metode *Gaussian Filter* dan *Wavelet*

*Transformation*. Metode *Gaussian Filter* digunakan untuk mereduksi noise pada citra dan metode *Wavelet Transformation* digunakan untuk menghasilkan ciri setiap penyakit kubis. Kombinasi kedua metode ini mampu mengidentifikasi dengan akurasi sebesar 85%. Salah satu penyebab kesalahan klasifikasi adalah teknik pengambilan citra dan teknik pemotongan citra yang kurang tepat sehingga menghasilkan citra yang *blur* dan tidak tepat sasaran [2].

Penelitian yang dilakukan oleh Ratih Kartika Dewi, R. V. Hari Ginardi (2004) untuk mengidentifikasi penyakit pada daun tebu dengan menggunakan *Gray Level Co-occurrence Matrix* (GLCM) dan *Color Moments*. Berdasarkan fitur yang telah diekstraksi sebelumnya yang kemudian untuk proses klasifikasinya menggunakan metode SVM. Kombinasi fitur tekstur dengan GLCM dan fitur warna dengan *Color Moments* pada penelitian ini menghasilkan akurasi sebesar 97% [3].

Terdapat perbedaan warna antara daun berpenyakit busuk hitam dan bercak daun sehingga perlu adanya ekstraksi warna. *Grayscale* merupakan merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, atau nilai bagian  $Red = Green = Blue$ . Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih [4].

Tahap awal penelitian adalah pengumpulan data citra daun tanaman kubis berpenyakit. Selanjutnya adalah *preprocessing* citra masukan dengan 3 tahapan yaitu *Histogram Equalization*, segmentasi dan *grayscale*. Tahap pertama adalah *Histogram Equalization* yang digunakan memperbaiki citra masukan sehingga dapat tersegmentasi dengan baik. Tahap selanjutnya adalah citra masukan di segmentasi untuk menghilangkan *background* citra masukan. Tahap berikutnya adalah *preprocessing* citra dengan mengubah citra masukan menjadi citra *grayscale* pada objek daun tanaman kubis.

Proses identifikasi penyakit daun tanaman kubis menggunakan metode SVM dengan menggunakan *Sequential Training* dengan menghitung nilai yang didapatkan dari *preprocessing*. Proses identifikasi dilakukan berdasarkan warna yang diambil dari citra daun tanaman kubis dengan menggunakan *grayscale*, yang menghasilkan nilai keabuan dari objek/*foreground* citra daun tanaman kubis.

Berdasarkan uraian diatas, pada skripsi ini penulis mengambil judul “Implementasi Metode *Support Vector Machine* (SVM) untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)”.

### 1.2. Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka dapat dirumuskan permasalahan yaitu:

- a. Bagaimana menerapkan metode *Support Vector Machine* untuk Identifikasi Penyakit pada Daun Tanaman Kubis?
- b. Bagaimana mengklasifikasi fitur *grayscale* dari daun tanaman kubis dengan Metode *Support Vector Machine*?

### 1.3. Tujuan

Tujuan dari penelitian ini adalah untuk:

- a. Menerapkan metode *Support Vector Machine* untuk Identifikasi Penyakit pada Daun Tanaman Kubis
- b. Mengukur hasil klasifikasi dari Metode *Support Vector Machine* untuk Identifikasi Penyakit pada Citra Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang).

### 1.4. Batasan Masalah

Batasan Masalah dalam penelitian ini adalah:

- a. Data *input* berupa citra digital daun bibit tanaman kubis yang berformat \*.jpg.
- b. Pengambilan citra daun tanaman kubis dengan jarak 30 – 40 cm dari kamera pada siang hari.
- c. Pengambilan citra daun tanaman kubis dilakukan di daerah Poncokusumo, Tumpang, Kabupaten Malang, Jawa Timur.
- d. Proses normalisasi dilakukan secara manual.
- e. *Software* yang digunakan untuk mengembangkan sistem adalah Visual Studio 2012.

### 1.5. Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka peneliti akan memaparkan sistematika penulisan sebagai berikut:

**Bab I Pendahuluan**

Berisi uraian mengenai latar belakang masalah yang diteliti, rumusan masalah, tujuan, batasan masalah, serta sistematika penulisan laporan yang digunakan.

**Bab II Landasan Teori**

Berisi teori-teori yang terkait dalam penelitian seperti dan melengkapi latar belakang dari masalah yang diteliti.

**Bab III Metodologi Penelitian**

Berisi mengenai langkah-langkah yang digunakan dalam penelitian, seperti metode yang digunakan dalam penelitian serta *tools* yang akan digunakan.

**Bab IV Analisis dan Perancangan**

Berisi uraian mengenai kebutuhan sistem meliputi kebutuhan fungsional dan non-fungsional. Rancangan sistem meliputi rancangan model, arsitektur sistem, proses, prosedural, data dan antarmuka pengguna (user interface).

**Bab V Implementasi**

Berisi mengenai detail implementasi sistem sesuai dengan rancangan dan tools bahasa pemrograman yang dipakai, disertai dengan potongan kode pada proses yang dimaksud.

**Bab VI Pengujian dan Pembahasan**

Berisi mengenai hasil pengujian yaitu untuk mengetahui hasil dari sistem sesuai dengan kebutuhan sistem dan berjalan sesuai dengan lingkungan yang diinginkan. Serta pembahasan berdasarkan fakta ilmiah yang diperoleh dari hasil pengujian.

**Bab VII Kesimpulan**

Berisi penjelasan singkat dan jelas mengenai penelitian yang diperoleh sesuai dengan tujuan penelitian. Serta saran, yang digunakan untuk menyampaikan hal-hal yang dapat diperbaiki, dikembangkan atau dijadikan penelitian lebih lanjut.



## BAB II. LANDASAN TEORI

### 2.1. Tanaman Kubis

Menurut sejarahnya, kubis liar dari tipe *Brassica oleracea* var. *Sylvestris*, pertama kali dijumpai tumbuh di sepanjang pantai Laut Mediterania dan di sepanjang pantai Atlantik, Benua Eropa. Kubis telur dan kale berasal dari Eropa bagian barat, sedangkan kubis bunga dan brokoli berasal dari Mediterania.

Sebelum dibudidayakan dan digunakan sebagai bahan makanan, tanaman ini banyak dimanfaatkan sebagai obat untuk nyeri otot, diare, hilang pendengaran, dan sakit kepala. Selain itu, jus kubis dapat dimanfaatkan untuk mengobati keracunan jamur.

Di Indonesia, dari beberapa jenis kubis yang dikenal, hanya kubis putih, kubis bunga, dan brokoli (kubis bunga hijau) yang banyak diusahakan di berbagai daerah sentra industri, tetapi kubis putih adalah yang paling banyak diusahakan.

Kubis merupakan tanaman herba semusim berbatang pendek dan memiliki ruas yang merupakan tempat duduknya daun. Pada bibit, perbedaan antar jenis tanaman sulit dibedakan. Namun setelah tumbuh beberapa waktu, tanaman baru dapat dibedakan berdasarkan ciri masing-masing tanaman. Bunga kubis memiliki mahkota berwarna kuning dan tersusun dalam tandan, penyerbukannya dibantu oleh serangga. Biji yang terdapat di dalam buah yang berupa *siliqua* (polong). Biji tersebut berukuran kecil, berbentuk bulat, dan berwarna kecokelatan.

Di dalam sistematika botani tanaman kubis menempati kedudukan sebagai berikut yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 Kedudukan tanaman kubis dalam sistematika botani

Divisi	Spermatofita
Subdivisi	Angiospermae
Kelas	Dikotil
Ordo	Cruciferales
Famili	Cruciferae
Genus	<i>Brassuca</i>
Spesies	<i>Brassica oleracea</i>
Grup	Capitata

Subgrup	Alba (kubis telur putih) Rubra (kubis telur merah) Sabauda (kubis savoy)
Grup	Botrytis (kubis bunga) Italica (brokoli) Gemmifera (Brussel sprouts) Acephala (kale atau kubis daun) Gongylodes (kolrabi)

Sumber: Zulkarnaen, 2013

### 2.2.1. Jenis Kubis

Berikut ini merupakan beberapa jenis kubis yang banyak diusahakan secara komersial adalah:

#### 1. Kubis

Kubis atau dapat disebut juga kubis putih (*Brassica oleracea* L. grup Capitata) memiliki daun yang lebar dan lunak. Daun yang muncul lebih dahulu menutup daun yang muncul kemudian sehingga membentuk *krop* seperti telur dan berwarna putih. Jenis kubis telur yang dianjurkan untuk ditanam adalah KK Cross, KY Cross, Hibrid 21, dan Hibrid 31 yang semuanya berasal dari Jepang serta Hibrid 368 yang berasal dari Australia. Sementara itu, jenis kubis lokal seperti Pujo, Segon dan Yoshin memiliki krop yang lunak dan rapuh sehingga kurang populer. Tanaman Kubis ditunjukkan pada Gambar 2.1.



Gambar 2.1 Kubis  
(Sumber: Zulkarnaen, 2013)

#### 2. Kubis bunga

Kubis bunga (*Brassica oleracea* L. grup Botrytis) memiliki bakal bunga yang mengembang dan membentuk massa bunga. Bunga tersebut berbentuk kerucut

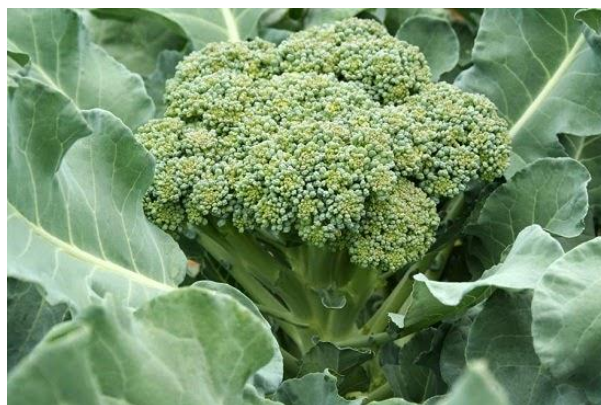
terbalik dan berwarna putih kekuningan. Beberapa kultivar yang dikenal adalah *Snowball* dan *Winter*. Agregat bunga pada kultivar *Snowball* bukanlah bunga sebenarnya, tetapi pucuk-pucuk yang tidak berdiferensiasi. Hanya kubis bunga kultivar *Winter* yang memiliki agregat bunga yang merupakan bunga sebenarnya. Tanaman Kubis Bunga ditunjukkan pada Gambar 2.2.



Gambar 2.2 Kubis Bunga  
(Sumber: Zulkarnaen, 2013)

### 3. Brokoli

Brokoli (*Brassuca oleracea* L. grup Italica) adalah jenis kubis yang mirip dengan kubis bunga. Perbedaan terletak pada massa bunga brokoli berwarna hijau, lebih kompak, dan lebih lezat dibandingkan kubis bunga. Kubis tunas atau kubis babat (*Brassica oleracea* L. grup Gemmifera) biasanya membentuk krop kecil, bahkan tunas sampingnya juga membentuk krop kecil. Oleh karena itu, satu tanaman dijumpai banyak krop berukuran kecil (panjang 5-7 cm dengan diameter 3-4 cm). Beberapa kultivar dari *Brussel sprout* yang dikenal adalah Pearl, Garmet, Jasper dan Jade Cross. Tanaman Brokoli ditunjukkan pada Gambar 2.3.



Gambar 2.3 Brokoli  
(Sumber: Zulkarnaen, 2013)

#### 4. Kubis Umbi

Kubis umbi atau kolrabi (*Brassica oleracea* L. grup *Gongylodes*). Bagian dasar batang yang berada di dalam tanah ataupun di atas tanah membesar sehingga menyerupai umbi. Salah satu kultivar kolrabi yang banyak dibudidayakan adalah *White Vienna*. Kale atau kalia (*Brassica oleracea* L. grup *Acephala*) sering disebut sebagai kubis daun kampung. Jenis kubis ini cukup terkenal dengan kultivarnya *Benten* dan *Tsoi-sim*. Tanaman Kubis Umbi ditunjukkan pada Gambar 2.4.



Gambar 2.4 Kubis Umbi  
(Sumber: Zulkarnaen, 2013)

##### 2.2.2. Nilai Gizi dan Manfaat

Semua tanaman yang termasuk ke dalam kelompok kubis kaya akan kandungan vitamin C. Kubis savoy, kale, dan brokoli kaya akan vitamin A ( $\beta$ -karoten). Kubis juga mengandung protein yang lebih tinggi daripada sayuran berpati.

Tanaman sayuran dari genus *Brassica* mengandung 3-methylcysteine sulfoxide, yaitu suatu senyawa yang dikenal sebagai "*kale anemia factor*". Dari sejumlah penelitian diketahui senyawa ini dapat menurunkan kadar kolesterol darah pada hewan percobaan. Sunaryono (2005) menambahkan bahwa air rebusan batang kubis dapat digunakan untuk menekan serangan penyakit prostat, sedangkan bunga brokoli dapat mencegah kanker dubur, kanker lambung, dan kanker usus besar (kolon). Manfaat lain dari kubis adalah menurunkan kadar kolesterol jahat (LDL) dalam tubuh, menurunkan resiko gangguan jantung, stroke dan katarak, serta mempercepat penyembuhan penyakit kulit (seperti bisul). Akan tetapi bagi penderita maag hendaknya menghindari konsumsi kubis yang berlebihan karena sayuran ini dapat menimbulkan terbentuknya gas yang berlebihan di dalam perut.

### 2.2.3. Penyakit

Berikut ini merupakan penyakit yang menyerang pada daun tanaman kubis:

#### a. Bercak Daun Alternaria

Bercak Daun Alternaria (*Alternaria brassicae* atau *Alternaria brassicicola*) merupakan penyakit yang menginfeksi tanaman muda dicirikan dengan timbulnya bercak cokelat kehitaman berdiameter kira-kira 2 mm pada keeping *kotiledon*-nya. Serangan pada tanaman dewasa dicirikan oleh timbulnya bercak-bercak bulat atau panjang berwarna cokelat sampai hitam. Jaringan daun yang terinfeksi akan mati, mengering, dan berlubang. Penyakit Bercak Daun ditunjukkan pada Gambar 2.5 [6].



Gambar 2.5 Daun tanaman kubis terserang Bercak Daun  
(Sumber: Zulkarnaen, 2013)

#### b. Busuk Hitam

Busuk Hitam (*Xanthomonas campestris* pv. *Campestris*) merupakan penyakit daun tanaman kubis yang pada tepi-tepi daun yang terserang terdapat daerah-daerah yang berwarna kuning atau pucat, yang kemudian meluas ke bagian tengah. Tulang-tulang daun berwarna cokelat tua atau hitam. Pada tingkatan yang lebih lanjut penyakit meluas terus melalui tulang-tulang daun dan masuk ke dalam batang. Jaringan helaian daun yang sakit mengering, menjadi seperti selaput, dengan tulang-tulang daun berwarna hitam. Penyakit Busuk Hitam ditunjukkan pada Gambar 2.6 [7].



Gambar 2.6 Daun tanaman kubis terserang Busuk Hitam  
(Sumber: Semangun, 2007)

## 2.2. Klasifikasi

Klasifikasi merupakan suatu proses pengelompokkan objek ke dalam suatu kelas tertentu. Klasifikasi memiliki tujuan untuk menentukan kelas suatu objek yang belum diketahui kelasnya. Dalam proses pengklasifikasian, sistem memberikan model yang dapat memberikan perhitungan sesuai dengan metode yang digunakan dan masukan yang diberikan. Proses tersebut berujung pada pemberian keputusan berupa *output* yang sudah di kategorikan sesuai kelas yang ada. Terdapat dua data yang dijadikan acuan dalam proses klasifikasi yaitu data *training* dan data *testing*. Data *training* merupakan data masukan yang berguna sebagai data latih dari model. Sedangkan data *testing* merupakan data yang digunakan untuk uji coba dari pemodelan yang masih belum memiliki kelas tertentu. Banyak metode yang terkait dengan klasifikasi, di antaranya adalah *Naive Bayes*, *K-Nearest Neighbor (KNN)*, *Support Vector Machine (SVM)*, *Artificial Neural Network (ANN)* [1].

## 2.3. Citra Digital

Citra merupakan suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan.

Citra Digital merupakan sebuah larik (array) yang berisikan nilai-nilai real maupun kompleks yang direpresentasikan dengan deretan bit tertentu. Citra dapat di definisikan sebagai fungsi  $f(x,y)$  berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitude f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut.

Nilai suatu piksel memiliki nilai dalam rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Berikut merupakan jenis citra berdasarkan nilai pikselnya.

#### 2.3.1. Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (*black and white*) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap piksel dari citra biner. Citra biner merupakan hasil dari proses pengolahan seperti segmentasi, pengambangan, morfologi ataupun dithering. Gambar 2.7 menunjukkan contoh dari citra biner.



Gambar 2.7 Citra Biner  
(Sumber: Sutoyo, 2009)

#### 2.3.2. Citra *Grayscale*

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, atau nilai bagian *Red = Green = Blue*. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Gambar 2.8 menunjukkan contoh citra *grayscale* [4].



Gambar 2.8 Citra *Grayscale*  
(Sumber: Sutoyo, 2009)

#### 2.3.3. Citra Warna

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar ( $RGB = Red\ Green\ Blue$ ). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 *byte*, yang berarti setiap warna mempunyai gradasi sebanyak

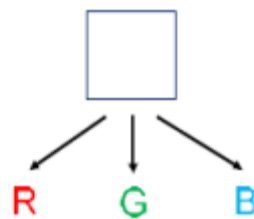


255 warna, dapat diartikan bahwa setiap piksel mempunyai kombinasi warna sebanyak  $2^8 \cdot 2^8 \cdot 2^8 = 2^{24} = 16$  juta warna lebih sehingga format citra ini dinamakan *true color* karena memiliki jumlah warna yang cukup besar sehingga bisa dikatakan hampir mencakup semua warna di alam.

Penyimpanan citra *true color* di dalam memori berbeda dengan citra *grayscale*. Setiap piksel dari citra *grayscale* 256 gradasi warna diwakili oleh 1 *byte*. Sedangkan 1 piksel citra *true color* diwakili oleh 3 *byte*, dimana masing-masing *byte* merepresentasikan warna merah (*Red*), hijau (*Green*), dan biru (*Blue*).



Gambar 2.9 Citra Warna 4 bit  
(Sumber: Putra, 2010)



Gambar 2.10 Citra Warna 3 bit  
(Sumber: Putra, 2010)

#### 2.3.4. Pengolahan Citra Digital

Pengolahan citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan citra ciri (*feature images*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data dan waktu proses data. *Input* dari pengolahan citra adalah citra sedangkan *output* berupa citra hasil pengolahan.

### 2.4. Thresholding

Proses *thresholding* merupakan proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas. Citra



hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur. Berikut ini merupakan persamaan untuk menghitung nilai *thresholding* yang ditunjukkan pada persamaan 2.1 sebagai berikut:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases} \quad (2.1)$$

Keterangan :

$g(x,y)$  : Citra biner dari ctra *grayscale*  $f(x,y)$

$f(x,y)$  : Citra *grayscale*

$T$  : Nilai *threshold*

Dengan  $g(x,y)$  adalah citra biner dari citra *grayscale*  $f(x,y)$ , dan  $T$  menyatakan nilai *threshold*. Kualitas hasil citra biner tergantung pada nilai  $T$  yang digunakan.

Metode *thresholding* secara umum dibagi menjadi dua, yaitu:

1. *Thresholding* global (*Global Thresholding*)

Seluruh piksel pada citra dikonversikan menjadi hitam atau putih dengan satu nilai  $T$ . kemungkinan besar pada pengembangan global akan banyak informasi hilang karena hanya menggunakan satu nilai  $T$  untuk keseluruhan piksel.

2. *Thresholding* local adaptif (*Locally Adaptive Thresholding*)

Suatu citra dibagi menjadi blok-blok kecil dan kemudian dilakukan *thresholding* lokal pada setiap blok dengan nilai  $T$  yang berbeda [5].

## 2.5. Normalisasi

Normalisasi diperlukan untuk memperkecil range nilai dari data agar tidak terlalu besar. Normalisasi mempunyai banyak jenis, salah satunya Normalisasi Data Sigmoid. Normalisasi ini memperkecil range nilai menjadi 0.1 hingga 0.9 [8]. Berikut ini merupakan persamaan untuk menghitung nilai normalisasi yang ditunjukkan pada persamaan 2.2 sebagai berikut:

$$z' = \frac{0,8(x - a)}{b - a} + 0,1 \quad (2.2)$$

Keterangan:

$z'$  : Nilai baru setelah dinormalisasi

$x$  : Data

$b$  : Nilai maksimum dari data

$a$  : Nilai minimum dari data

## 2.6. *Support Vector Machine (SVM)*

*Support Vector Machine (SVM)* dikembangkan oleh Boser, Guyon, dan Vapnik, pertama kali diperkenalkan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep dasar metode SVM sebenarnya merupakan gabungan atau kombinasi dari teori-teori komputasi yang telah ada pada tahun sebelumnya, seperti kernel diperkenalkan oleh Aronszajn tahun 1950, Lagrange Multiplier yang ditemukan oleh Joseph Louis Lagrange pada tahun 1766, dan demikian juga dengan konsep-konsep pendukung lain.

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari pemisah dua buah kelas pada *input space*. *Pattern* yang merupakan anggota dari dua buah kelas: +1 dan -1 dan berbagi alternatif garis pemisah (*discrimination boundaries*). Margin adalah jarak antara pemisah tersebut dengan *pattern* terdekat dari masing-masing kelas. *Pattern* yang memiliki jarak paling dekat disebut sebagai *support vector*.

Dalam menyelesaikan permasalahan linear diansumsikan terdapat data latih  $\{x_i, y_i\}$ ,  $x_i$  merupakan atribut untuk data latih  $\{x_1, \dots, x_n\}$  dan  $y_i \in \{-1, 1\}$  adalah label kelas dari data latih  $x_i$ . Bidang pemisah yang baik tidak hanya bisa memisahkan data tetapi juga memiliki margin yang besar atau maksimal, data yang berada dekat dan diatas bidang pemisah.

Apabila terdapat dua kelas yang dipisahkan oleh dua bidang pembatas secara sejajar yaitu kelas +1 dan kelas -1. Bidang pembatas dinyatakan dengan H.  $H_1$  dinyatakan sebagai bidang pembatas pada kelas +1 dan  $H_2$  dinyatakan sebagai bidang pembatas kelas -1, yang dinyatakan dengan persamaan 2.3 dan 2.4 sebagai berikut:

$$w \cdot x_i + b \geq +1 \text{ for } y_i = +1 \quad (2.3)$$

$$w \cdot x_i + b \geq -1 \text{ for } y_i = -1 \quad (2.4)$$

Keterangan :

$x_i$  : Data ke- $i$ ,  $i = 1, 2, 3, \dots, n$

$w_i$  : Bobot *support vector*

$b$  : Nilai bias

$y_i$  : Kelas dataset

Sehingga didapatkan persamaan 2.5 sebagai berikut:

$$y_i(x_i \cdot w + b) \geq 1 \quad (2.5)$$

Jika titik terpisah secara linear fungsi untuk permukaan ini ditentukan dengan persamaan 2.6 sebagai berikut:

$$f(x) = \left( \sum_{i=1}^n \alpha_i^* y_i (x_i \cdot x) + b^* \right) \quad (x_i, y_i) \in R^N \times \{-1, 1\} \quad (2.6)$$

Keterangan :

$\alpha_i^*$  : *Lagrange Multiplier*

$b^*$  : Bias

Jika kelas tidak terpisah secara linear maka fungsi untuk permukaan ditentukan oleh persamaan 2.7 sebagai berikut:

$$f(x) = \left( \sum_{i=1}^n \alpha_i^* y_i k(x, y) + b^* \right) \quad (2.7)$$

Keterangan :

$\alpha_i^*$  : *Lagrange Multiplier*

$b^*$  : Bias

$k(x, y)$  : Fungsi Kernel

Terdapat beberapa fungsi kernel yang digunakan untuk menyelesaikan masalah pada SVM non linear dapat dilihat pada persamaan 2.8 hingga 2.10 berikut ini [9].

a. Fungsi Kernel Linear

$$k(x, y) = x \cdot y \quad (2.8)$$

b. Fungsi Kernel Polinomial

$$k(x, y) = (1 + x \cdot y)^q, q = 1, 2, \dots, N \quad (2.9)$$

c. Fungsi Kernel Radial Basis

$$k(x, y) = \exp \frac{(-||x - y||^2)}{2\sigma^q} \quad (2.10)$$

#### 2.6.1. Metode Sequential Training SVM

Metode Sequential Training merupakan metode yang digunakan untuk training data agar menghasilkan hyperlane yang optimal. Metode ini digunakan juga

untuk mendapatkan nilai  $\alpha$ . Metode ini dikembangkan oleh Vijayakumar dan memiliki langkah proses seperti berikut.

- a. Melakukan perhitungan kernel dan melakukan inisialisasi parameter-parameter SVM seperti contoh nilai  $\alpha_i = 0$ ;  $\varepsilon = 0,001$ ;  $\gamma = 0,01$ ;  $\lambda = 1$ ;  $C = 1$ ; dan nilai iterasi maksimal = 10;
- b. Menghitung matriks Hessian dengan persamaan 2.11.

$$D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2) \quad (2.11)$$

- c. Melakukan iterasi untuk setiap iterasi = 1, 2, 3, ... , n dan dihitung menggunakan persamaan 2.12 hingga 2.18.

- Menghitung nilai  $E_i$  menggunakan persamaan 2.12.

$$E_i = \sum_{j=1}^n \alpha_j D_{ij} \quad (2.12)$$

- Menghitung nilai  $\gamma$  dan  $\delta\alpha_i$  menggunakan persamaan 2.13 dan 2.14.

$$\gamma = \frac{\text{konstanta}}{\max D_{ii}} = 0,008 \quad (2.13)$$

$$\delta\alpha_i = \min[\max[\gamma(1 - E_i), \alpha_i], C - \alpha_i] \quad (2.14)$$

- Memperbarui nilai  $\alpha_i$  menggunakan persamaan 2.15.

$$\alpha_i = \alpha_i + \delta\alpha \quad (2.15)$$

Keterangan:

$E_i$  : Nilai Error

$D_{ij}$  : Nilai Matriks Hessian

$\gamma$  : Gamma

$\delta\alpha_i$  : Delta alpha i

$C$  : *Complexity*

- d. Proses 1, 2, dan 3 diulangi hingga nilai  $\delta\alpha_i$  mencapai konvergen ( $|\delta\alpha| < \varepsilon$ ) dan atau ketika nilai iterasi sudah mencapai nilai maksimum maka iterasi dihentikan.
- e. Menghitung nilai  $w \cdot x^+$  dan  $w \cdot x^-$  untuk mendapatkan nilai bias menggunakan persamaan 2.16 hingga 2.18 [8].

$$w \cdot x^+ = \alpha_i \cdot y_i \cdot K(x, x^+) \quad (2.16)$$

$$w \cdot x^- = \alpha_i \cdot y_i \cdot K(x, x^-) \quad (2.17)$$

$$b = -\frac{1}{2}(w \cdot x^+ + w \cdot x^-) \quad (2.18)$$

Keterangan:

$K(x, x^+)$  : Nilai kernel data  $x$  dengan data  $x$  kelas positif yang memiliki nilai  $\alpha$  tertinggi.

$K(x, x^-)$  : Nilai kernel data  $x$  dengan data  $x$  kelas negatif yang memiliki nilai  $\alpha$  tertinggi.

$b$  : Nilai bias

## 2.7. Evaluasi

Evaluasi bertujuan untuk mengetahui tingkat akurasi dari hasil penggunaan metode *Support Vector Machine* (SVM) dengan cara menghitung jumlah data uji yang kelasnya diprediksi secara benar. Metode perhitungan tingkat akurasi sebagai pengujian dan evaluasi sistem. Teori evaluasi yang digunakan adalah dengan melakukan pendekatan statistik. Untuk mengukur akurasi dan ketepatan model dapat dilakukan dengan menghirung perbandingan jumlah ketepatan model dapat dilakukan dengan menghitung perbandingan jumlah prediksi benar terhadap seluruh record yang dapat diprediksi. Untuk formulasi jumlah testing data yang tepat dikenali dan jumlah seluruh data testing kemudian dikalikan 100%. Secara matematis dapat dirumuskan pada persamaan 2.19 sebagai berikut:

$$\text{Akurasi} = \frac{\text{Jumlah Prediksi Benar}}{\text{Total Prediksi}} \times 100\% \quad (2.19)$$

Kemudian untuk mengukur kinerja dari sistem akan dilakukan evaluasi dengan menggunakan *confusion matrix*. *Confusion Matrix* adalah matriks yang berisi data actual dan data prediksi yang telah diklasifikasikan oleh sistem. Berikut ini merupakan *confusion matrix* seperti yang ditunjukkan pada Tabel 2.3:

Tabel 2.2 *Confusion Matrix*

<i>Predictive</i>	<i>Actual</i>	
	<i>Negative</i>	<i>Positive</i>
<i>Negative</i>	<i>True Negative (TN)</i>	<i>False Positive (FP)</i>
<i>Positive</i>	<i>False Negative (FN)</i>	<i>True Positive (TP)</i>

(Sumber: Musicant, 2003)

Pada Tabel 2.3 menunjukkan data *confusion matrix* dengan dua kelas pada proses klasifikasi dimana [10]:

- *True Positive* (TP) merupakan kasus positif yang diklasifikasikan secara benar dan hasilnya positif.
- *True Negative* (TN) merupakan kasus negatif yang diklasifikasikan secara benar dan hasilnya negatif.
- *False Positive* (FP) merupakan kasus dengan kelas negatif yang diklasifikasikan pada kelas positif.
- *False Negative* (FN) merupakan kasus dengan kelas positif yang diklasifikasikan pada kelas negatif.

Berdasarkan *confusion matrix* diatas maka untuk

*Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Untuk dapat menghitung *precision* dari klasifikasi dapat menggunakan persamaan 2.20 sebagai berikut:

$$\text{Precision} = \frac{\text{TP}}{(\text{FP} + \text{TP})} \times 100\% \quad (2.20)$$

*Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Untuk dapat menghitung *recall* dari klasifikasi dapat menggunakan persamaan 2.21 sebagai berikut:

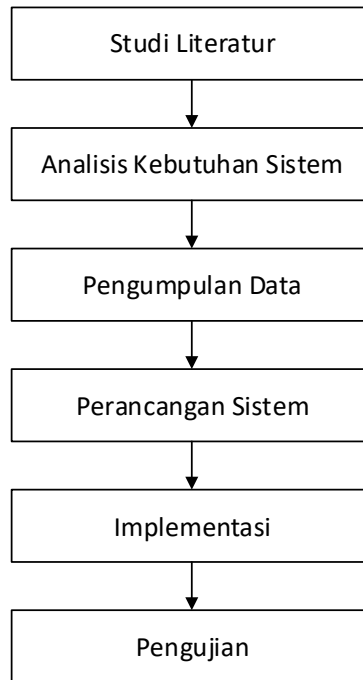
$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \times 100\% \quad (2.21)$$

Berdasarkan *confusion matrix* diatas maka untuk menghitung tingkat akurasi dari klasifikasi dapat dihitung dengan menggunakan persamaan 2.22 sebagai berikut [9]:

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\% \quad (2.22)$$

### BAB III. METODOLOGI PENELITIAN

Tahapan penelitian yang dilakukan untuk proses pembuatan perangkat lunak yang dilakukan oleh penulis yang ditunjukkan pada Gambar 3.1



Gambar 3.1 Diagram Blok Metodologi Penelitian  
(Sumber: Perancangan)

#### 3.1. Studi Literatur

Studi literatur merupakan data yang didapatkan dari literatur atau buku yang terkait dengan dasar teori yang diperlukan serta terkait dalam penelitian. Susunan studi literatur yang berkaitan dengan skripsi ini meliputi:

- a. Tanaman Kubis
- b. Klasifikasi
- c. Citra Digital
- d. *Thresholding*
- e. Normalisasi
- f. *Support Vector Machine*
- g. Evaluasi

### 3.2. Analisis Kebutuhan Sistem

Analisa kebutuhan merupakan kegiatan untuk mendapatkan kebutuhan yang akan digunakan untuk implementasi sistem. Analisa kebutuhan didapatkan dengan menentukan data objek yang akan digunakan, kebutuhan alat dan bahan yang digunakan untuk penelitian, dan lokasi dari pengambilan objek yang akan digunakan.

### 3.3. Pengumpulan Data

Pengumpulan data dilakukan dengan cara observasi langsung di daerah Tumpang, Kabupaten Malang, Jawa Timur. Objek penelitian ini adalah daun bibit tanaman kubis. Data objek berupa citra digital dari daun tanaman kubis. Data citra digital daun tanaman kubis yang digunakan dalam penelitian ini adalah sebanyak 66 citra, 30 citra digunakan sebagai data *training* dan 36 citra digunakan sebagai data *testing*.

Citra diambil dengan menggunakan kamera *handphone* dengan jarak yang digunakan adalah 30 – 40 cm. Pengambilan citra daun tanaman kubis dilakukan pada siang hari untuk mendapatkan pencahayaan maksimal (tidak berada dalam kondisi yang minim cahaya). Apabila pada citra yang diambil terdapat penurunan intensitas cahaya dapat menyebabkan hasil segmentasi tidak masimal sehingga informasi citra masukan tidak dapat terkomputasi dengan baik.

### 3.4. Perancangan Sistem

Perancangan sistem merupakan tahap merancang suatu sistem yang dapat memenuhi kebutuhan fungsional dari aplikasi dalam penelitian ini. Teori-teori dan data objek yang didapatkan dari observasi digabungkan dengan ilmu yang didapat untuk merancang dan membangun suatu aplikasi. Aplikasi tersebut merupakan hasil dari Identifikasi penyakit pada daun tanaman kubis menggunakan metode *Support Vector Machine*.

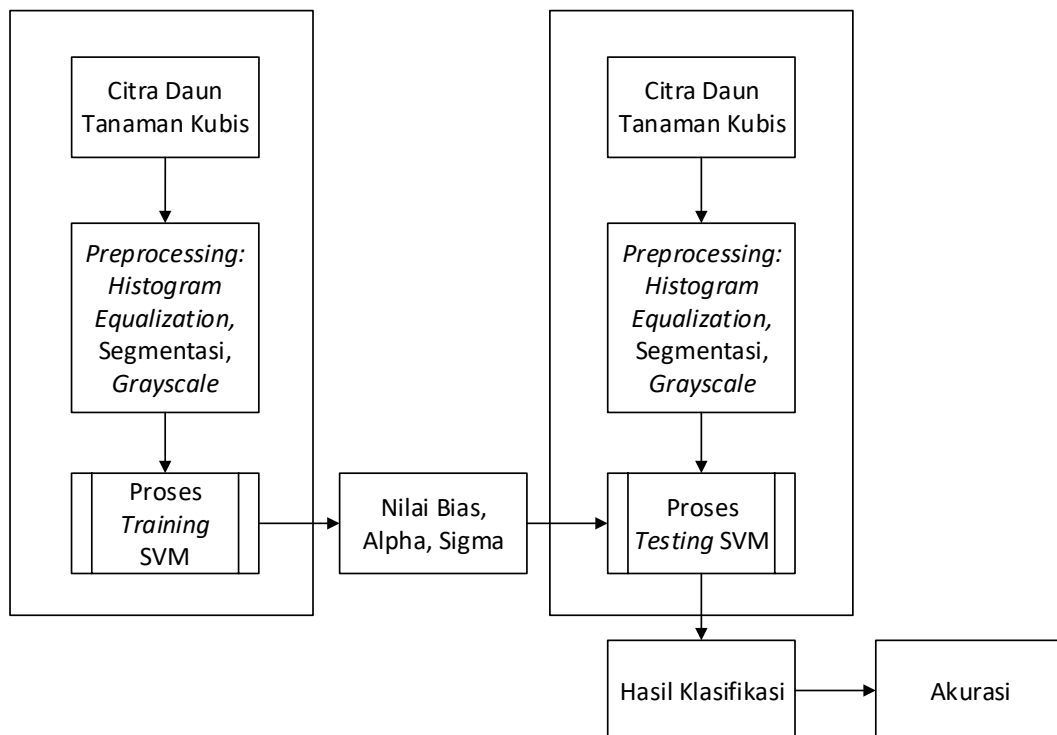
Proses akan dimulai dengan data masukan berupa citra digital daun tanaman kubis berpenyakit dengan hasil keluaran berupa hasil klasifikasi. Proses *training* dimulai dengan *input* citra digital daun tanaman kubis. Selanjutnya dilakukan proses *preprocessing* citra dimulai dengan menggunakan *Histogram Equalization* untuk memperbaiki citra masukan selanjutnya dilakukan proses segmentasi dengan



menggunakan *thresholding* untuk menghilangkan *background* citra masukan kemudian dilakukan proses *grayscale*, nilai yang didapatkan dari proses tersebut kemudian diproses dengan proses *training* SVM yang menghasilkan nilai *bias*, *alpha*, *delta alpha*, *support vector* dan *sigma*.

Proses *testing* dilakukan dengan *input* citra digital daun tanaman kubis. Selanjutnya dilakukan proses *preprocessing* citra dimulai dengan menggunakan *Histogram Equalization* untuk memperbaiki citra masukan selanjutnya dilakukan proses segmentasi dengan menggunakan *thresholding* untuk menghilangkan *background* citra masukan kemudian dilakukan proses *grayscale*. Selanjutnya nilai *bias*, *alpha* serta *sigma* yang didapatkan dari proses pelatihan digunakan untuk perhitungan proses pengujian yang kemudian akan menghasilkan klasifikasi.

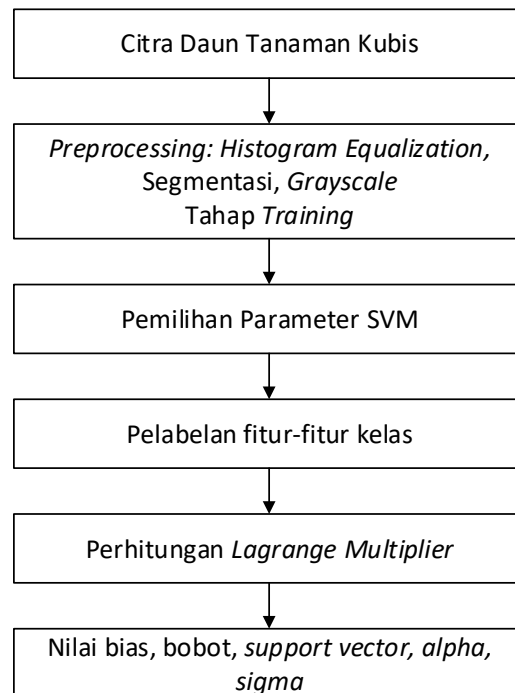
Diagram blok sistem merupakan gambaran umum cara kerja sistem pada penelitian ini. Berikut merupakan gambaran umum sistem yang dilakukan pada penelitian ini yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Diagram Blok Identifikasi Penyakit Daun Tanaman Kubis  
(Sumber: Perancangan)

Pada penelitian ini proses identifikasi terbagi menjadi dua yaitu proses *training* atau pelatihan dan proses *testing* atau pengujian. Untuk proses *training* SVM, yang pertama dilakukan setelah mendapatkan nilai dari *preprocessing* citra kemudian akan dipilih parameter parameter yang digunakan untuk menghasilkan tingkat akurasi yang optimal. Selanjutnya proses pemberian label fitur-fitur kedalam sejumlah kelas. Kemudian dilakukan perhitungan nilai *Lagrange Multiplier* untuk mencari nilai optimasi global terhadap nilai dan jumlah *support vector* dengan menggunakan *Sequential Training SVM*. Setelah itu menghitung nilai nilai *bias*, bobot dan *support vector* yang digunakan untuk proses perhitungan *testing* SVM.

Berikut ini merupakan diagram blok dari proses pelatihan yang ditunjukkan pada Gambar 3.3.

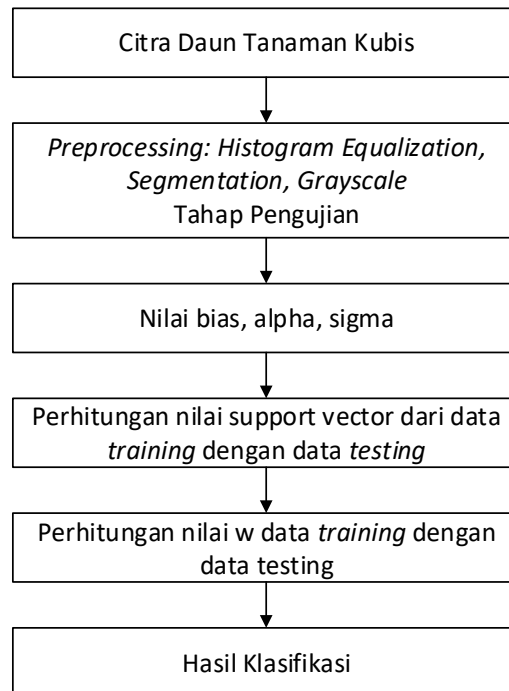


Gambar 3.3 Diagram Blok Proses *Training* SVM  
(Sumber: Perancangan)

Proses *testing* SVM dimulai dengan mendapatkan nilai dari *preprocessing* citra yang digunakan untuk pengujian selanjutnya adalah dengan memasukkan nilai *bias*, *alpha* dan *sigma* yang diperoleh dari proses *training*. Kemudian akan dihitung nilai *support vector* dari data *training* dengan data *testing*. Selanjutnya menghitung nilai *w* dan menjumlahkan semua nilai *w* untuk setiap data *training* dengan data

*testing*, dengan menggunakan nilai  $\alpha$  yang didapatkan dari proses *training*, *support vector* dari perhitungan *testing* serta kelas dari data *training*. Hasil penjumlahan nilai  $w$  tersebut kemudian akan dijumlahkan dengan nilai bias yang didapatkan dari proses *training* yang kemudian akan menjadi nilai yang akan menjadi hasil dari klasifikasi.

Berikut ini merupakan diagram blok dari proses pengujian yang ditunjukkan pada Gambar 3.4.



Gambar 3.4 Diagram Blok Proses Pengujian SVM  
(Sumber: Perancangan)

### 3.5. Implementasi

Proses implementasi atau pembuatan sistem merupakan tahap untuk penerapan yang telah diperoleh dari kegiatan sebelumnya. Dalam penelitian ini, implementasi yang dilakukan adalah implementasi dari penerapan metode *Support Vector Machine* (SVM) untuk klasifikasi penyakit pada daun tanaman kubis ke dalam program. Implementasi perangkat lunak akan menggunakan Visual Studio 2012 yang terdiri dari:

- Pembuatan antar muka.
- Memasukkan data citra daun tanaman kubis ke dalam program untuk selanjutnya agar dapat diolah.
- Menerapkan metode SVM ke dalam aplikasi.

- d. *Output* berupa hasil klasifikasi.
- e. Analisa berdasarkan hasil klasifikasi yang didapatkan.

### **3.6. Pengujian**

Pengujian dari perangkat lunak berkaitan dengan pengujian fungsional sistem, apakah sistem sudah berjalan dengan baik. Seberapa baik kemampuan sistem dalam melakukan perhitungan sesuai dengan kode yang tertulis dalam sistem.

Pengujian dan analisis sistem dilakukan dengan membandingkan data asli terhadap data hasil klasifikasi menggunakan *Support Vector Machine*. Hasil tersebut berupa nilai akurasi dari sistem dalam mengklasifikasikan penyakit daun tanaman kubis.

## BAB IV. ANALISIS DAN PERANCANGAN

### 4.1. Analisis Sistem

Analisis sistem merupakan suatu penjabaran mengenai komponen-komponen penyusun sistem dalam penelitian ini baik perangkat lunak maupun perangkat keras serta gambaran umum sistem yang akan berjalan.

Proses yang terdapat dalam aplikasi ini adalah *preprocessing* citra masukan kemudian dilakukan proses *training* dan proses *testing*. *Preprocessing* yang dilakukan adalah perbaikan citra masukan dengan *Histogram Equalization*, selanjutnya dilakukan segmentasi dengan *thresholding* untuk menghilangkan *background* dari citra masukan kemudian citra masukan diubah menjadi *grayscale* untuk mendapatkan nilai yang akan digunakan untuk proses perhitungan. Dimana proses *training* berfungsi untuk menemukan nilai *kernel*, *support vector*, *alpha*, *bias* dan *sigma*. Sedangkan proses *testing* berfungsi untuk mengidentifikasi jenis penyakit yang menyerang bibit daun tanaman kubis dengan menggunakan nilai *alpha*, *bias* serta *sigma* yang telah ditemukan dari proses *training*.

#### 4.1.1. Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak untuk melakukan perancangan dan menjalankan aplikasi yaitu diantaranya:

- a. *Operating System* : Windows 7
- b. *Software* untuk membuat aplikasi *desktop* : Microsoft Visual Studio 2012
- c. *Microsoft SQL Server 2008* untuk *database*
- d. *Microsoft Excel* untuk membuat perhitungan manual

#### 4.1.2. Kebutuhan Perangkat Keras

Spesifikasi minimum perangkat keras untuk menjalankan perangkat lunak diatas adalah:

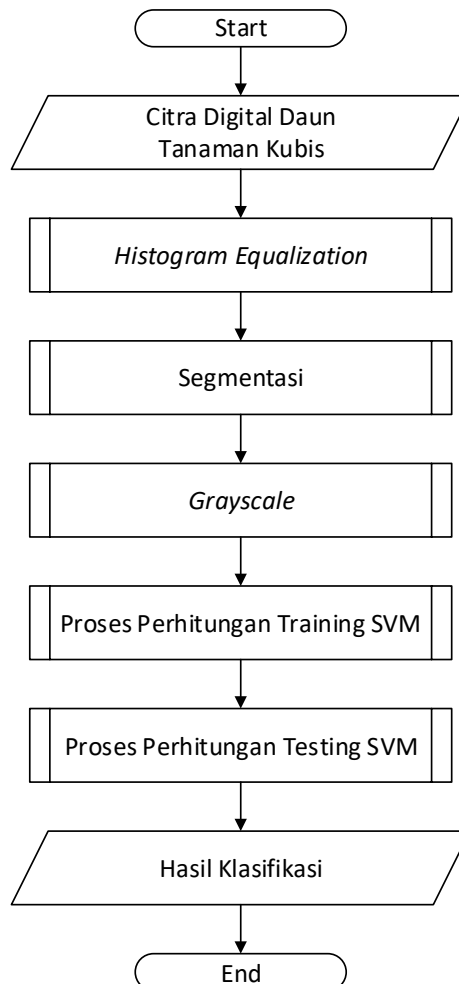
- a. Processor Intel(R) Core i3-505U CPU @ 2.00GHz 2.00GHz
- b. RAM 2.00 GB

### 4.2. Perancangan Sistem

Perancangan sistem merupakan tahap merancang suatu sistem sebagai penggambaran, perencanaan dan pembuatan sketsa dari aplikasi. Rancangan sistem akan menggambarkan proses yang disajikan ke dalam bentuk *flowchart*.

#### 4.2.1. Diagram Alir Sistem

Alur Sistem Implementasi Metode SVM untuk Identifikasi Penyakit Daun Tanaman Kubis terdapat pada Gambar 4.1.



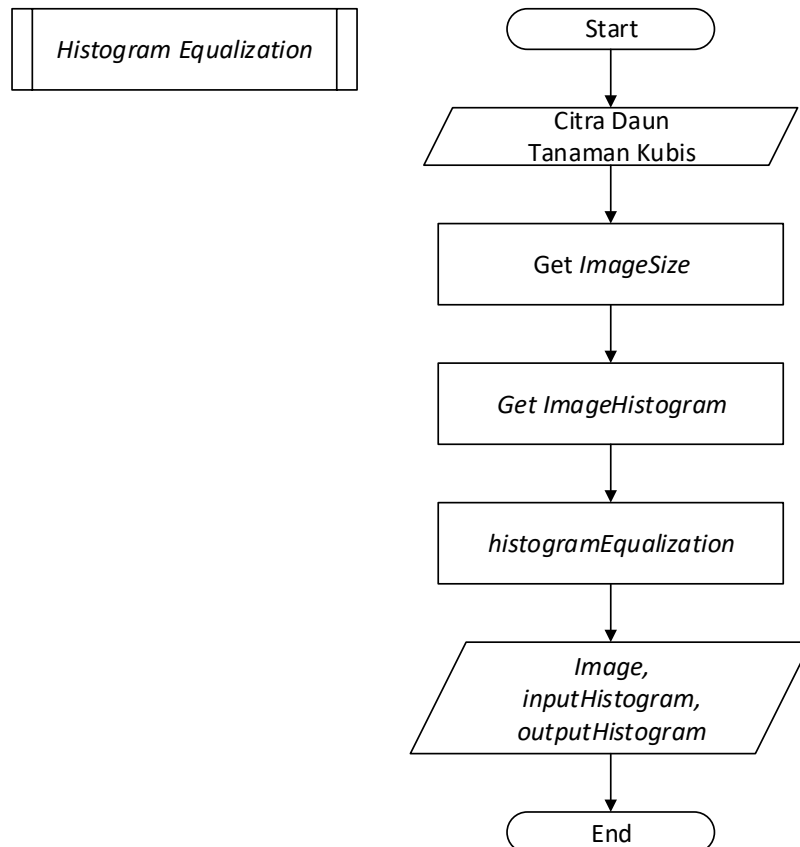
Gambar 4.1 Diagram Alir Sistem  
(Sumber: Perancangan)

Data masukan untuk penelitian ini adalah citra digital Daun Tanaman Kubis berpenyakit dengan keluaran berupa hasil klasifikasi. Proses alir sistem dimulai dengan masukan citra daun tanaman kubis yang kemudian dilakukan perbaikan kualitas citra dengan menggunakan *Histogram Equalization*, selanjutnya dilakukan proses segmentasi citra dengan menghilangkan *background* citra masukan kemudian dilakukan proses *grayscale* yang menghitung hanya nilai *foreground* citra masukan. Selanjutnya dilakukan proses normalisasi data untuk mendapatkan data dalam range yang tidak terlalu besar. Proses berikutnya perhitungan training SVM dengan menghitung seluruh nilai citra masukan yang dilakukan

*preprocessing* serta masukan parameter-parameter yang dibutuhkan. Selanjutnya pada proses *testing*, juga melakukan tahapan *preprocessing* citra sehingga nilai yang didapatkan dari *preprocessing* dapat digunakan untuk perhitungan *testing* yang akan mengambil nilai yang telah disimpan dari proses *training* sebelumnya yaitu nilai *alpha*, *bias* dan *sigma*. Hasil keluaran berupa klasifikasi citra daun tanaman kubis apakah termasuk pada penyakit bercak daun atau busuk hitam.

#### 4.2.2. Proses *Histogram Equalization*

Tahapan ini dilakukan untuk memperbaiki kualitas citra masukan dengan menggunakan *Histogram Equalization* sehingga dapat dilakukan segmentasi dengan baik, proses *Histogram Equalization* ditunjukkan pada Gambar 4.2.

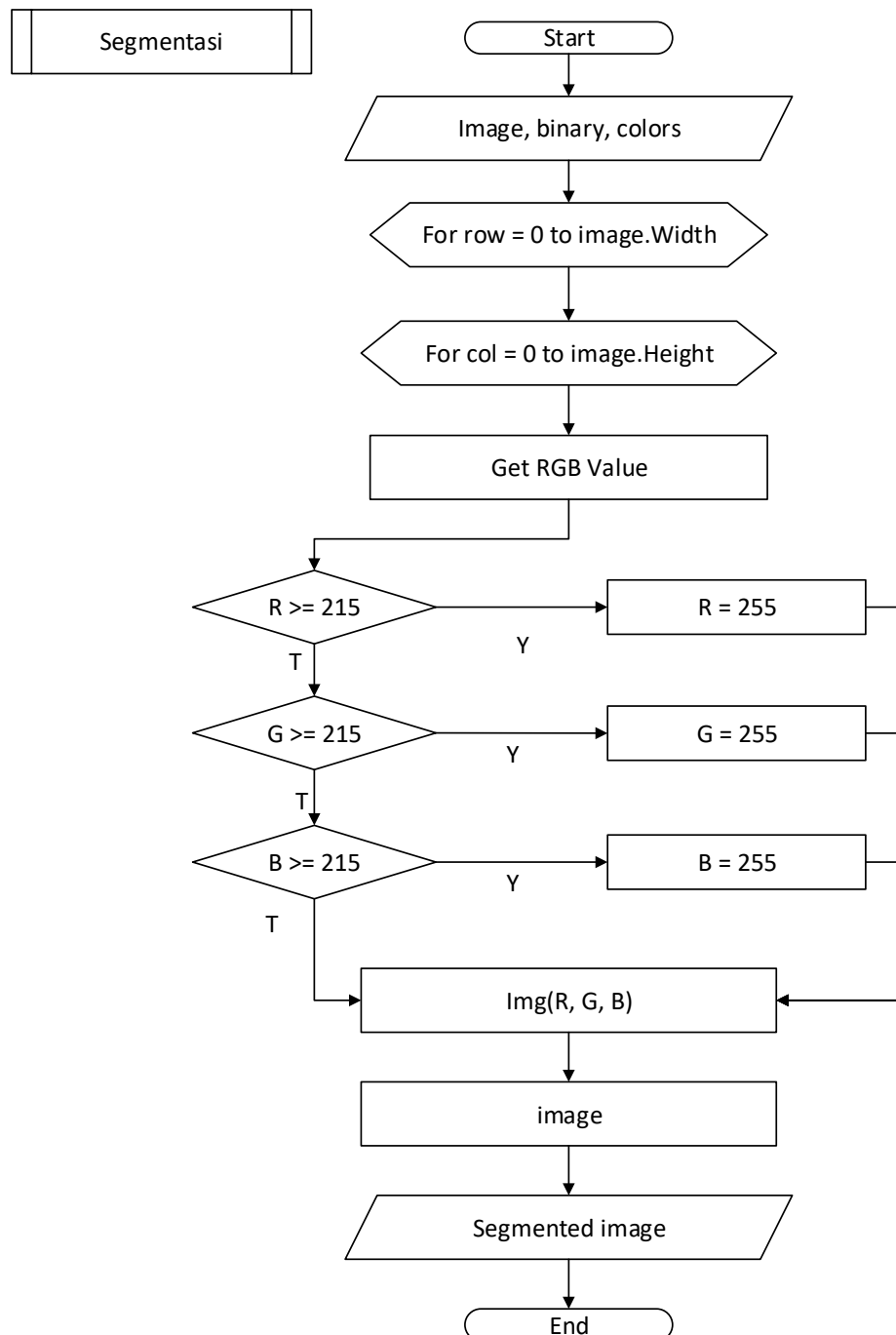


Gambar 4.2 Diagram Alir *Histogram Equalization*  
(Sumber: Perancangan)

#### 4.2.3. Proses Segmentasi

Tahapan segmentasi dilakukan setelah tahapan *Histogram Equalization* yang digunakan untuk menghilangkan *background* dari citra masukan dengan menggunakan *thresholding* sehingga nilai yang akan dihitung pada proses

selanjutnya hanya nilai *foreground* dari citra masukan. Proses segmentasi ditunjukkan pada Gambar 4.3.



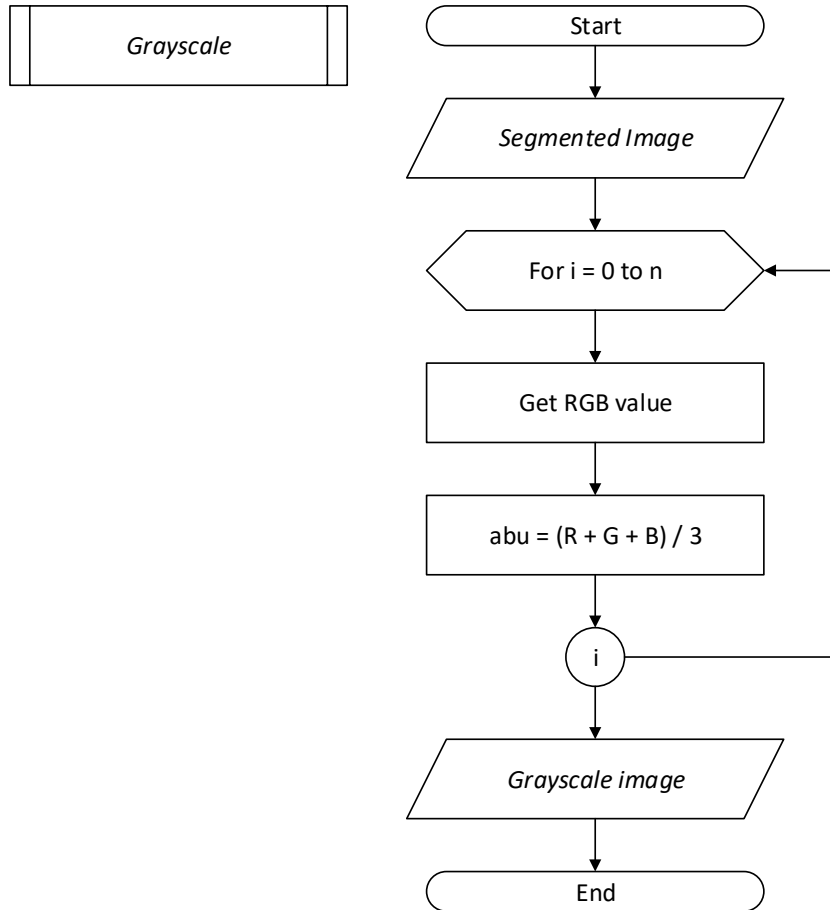
Gambar 4.3 Diagram Alir *Thresholding*  
(Sumber: Perancangan)

#### 4.2.4. Proses *Grayscale*

Tahapan *preprocessing* dilakukan untuk mempersiapkan data set citra agar siap diolah untuk proses perhitungan SVM yaitu dengan menggunakan mengubah



citra menjadi *grayscale*. Nilai yang didapatkan kemudian akan digunakan sebagai masukan dari proses *training* dan *testing* SVM. Proses *grayscale* ditunjukkan pada Gambar 4.4.

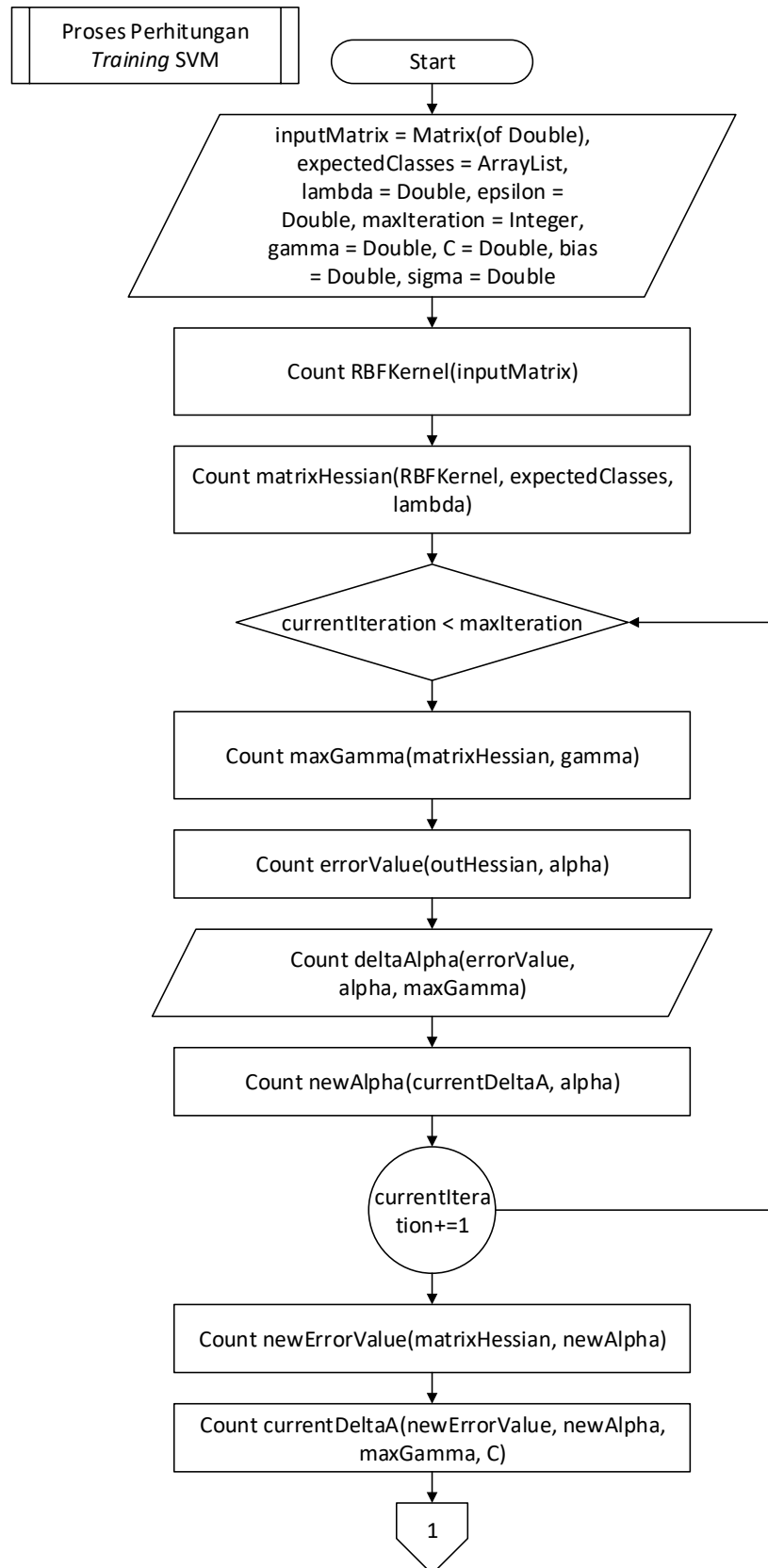


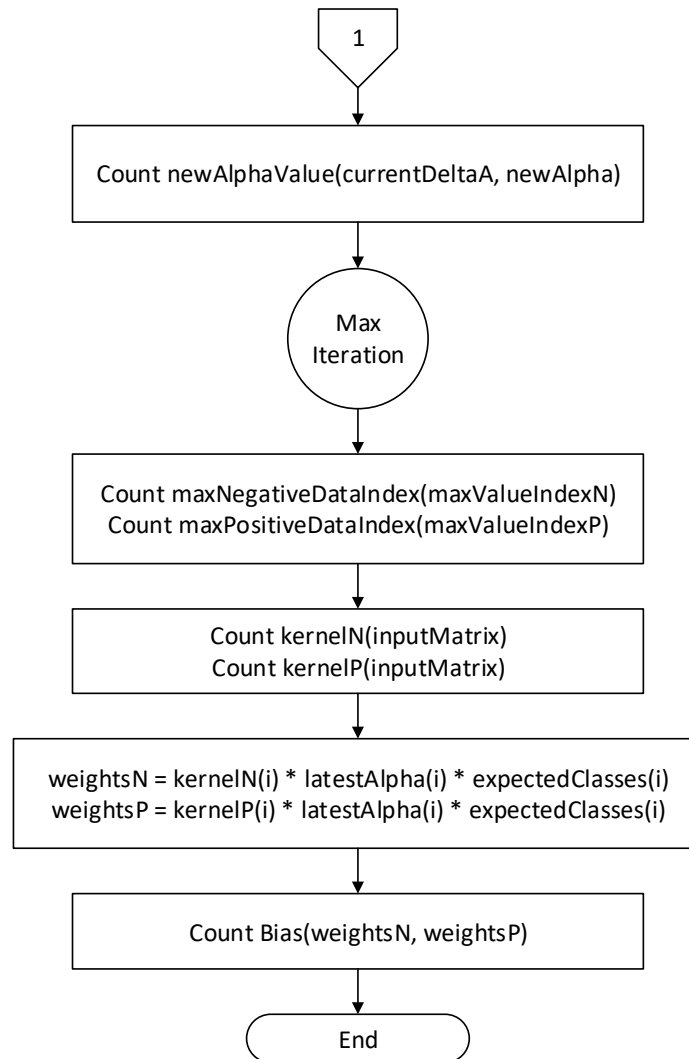
Gambar 4.4 Diagram Alir *Grayscale*  
(Sumber: Perancangan)

Data masukan berupa citra yang kemudian diambil nilai *red*, *green*, *blue* untuk kemudian dilakukan perhitungan dengan menjumlahkan ketiga nilai tersebut kemudian dibagi tiga pada setiap pikselnya, sehingga setelah seluruh nilai piksel diubah menjadi nilai keabuan maka nilai-nilai tersebut akan dijumlahkan dan kemudian akan dibagi sesuai dengan ukuran dari citra masukan yang akan menghasilkan rata-rata dari nilai keabuan seluruh citra. Hasil tersebut yang akan digunakan sebagai masukan pada proses perhitungan *training* dan *testing* SVM.

#### 4.2.5. Proses Perhitungan *Training* SVM

Proses perhitungan *training* SVM ditunjukkan pada Gambar 4.5.





Gambar 4.5 Diagram Alir Perhitungan Training SVM  
(Sumber: Perancangan)

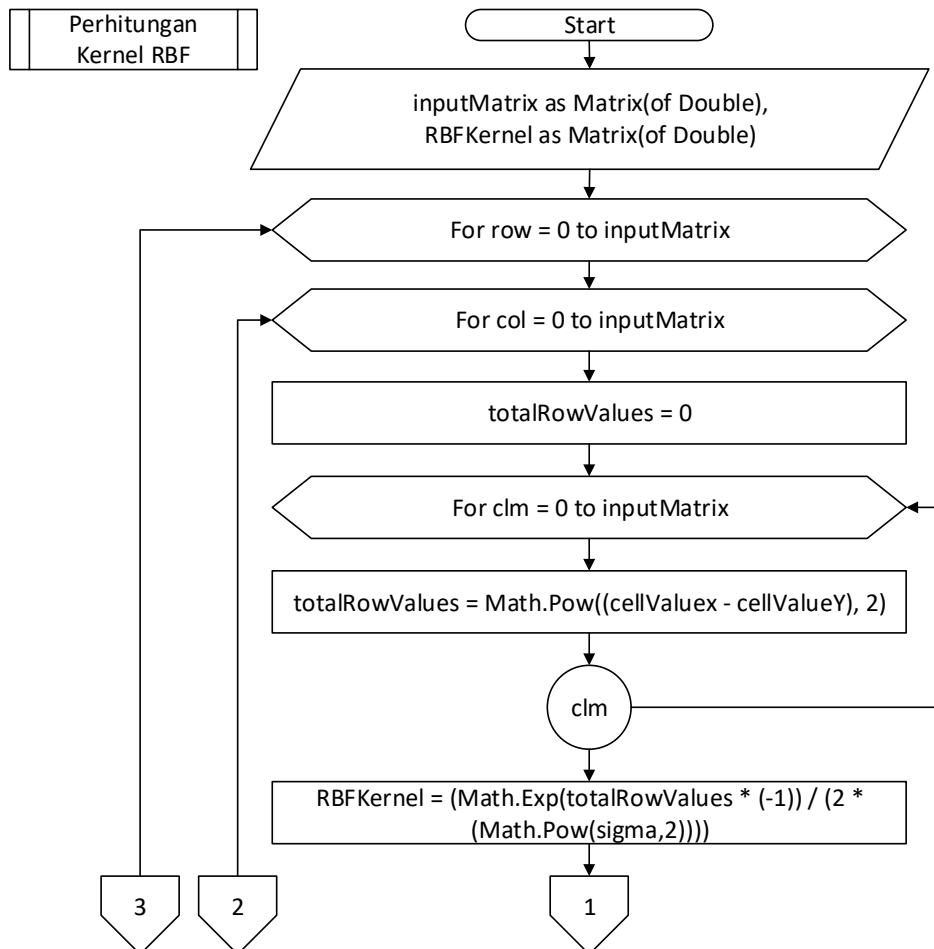
Metode *training* yang digunakan pada penelitian ini adalah metode *Sequential Training*. Metode ini mempunyai alur sebagai berikut:

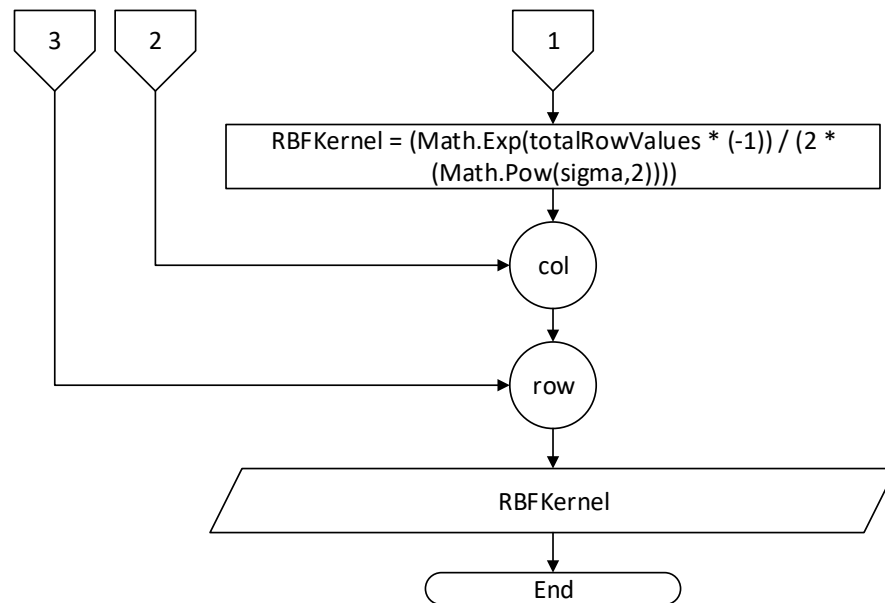
1. Data yang menjadi masukan adalah data *training* dan inisialisasi parameter-parameter SVM.
2. Menghitung kernel RBF, Matriks Hessian dan Max Gamma.
3. Melakukan iterasi untuk tiap data untuk mendapatkan nilai  $E_i$ ,  $\delta\alpha$ , dan memperbarui nilai  $\alpha_i$  menggunakan persamaan 2.12 hingga 2.14.
4. Memeriksa nilai  $|\delta\alpha| < \epsilon$ , jika benar iterasi dihentikan atau iterasi sudah mencapai maksimal iterasi dihentikan.

5. Menentukan nilai maksimal alpha kelas positif dan nilai maksimal alpha kelas negatif. Data kelas positif dan data kelas negatif memiliki nilai alpha tertinggi digunakan untuk perhitungan nilai  $w.x^+$  dan  $w.x^-$ .
6. Menghitung nilai kernel antara data *training* dengan data kelas positif yang memiliki nilai alpha tertinggi ( $K(x, x^+)$ ) dan data *training* dengan data kelas negatif yang memiliki nilai alpha tertinggi ( $K(x, x^-)$ ).
7. Menghitung nilai  $w.x^+$  dan  $w.x^-$  menggunakan persamaan 2.16 dan 2.17.
8. Menghitung nilai bias dengan menggunakan persamaan 2.18.
9. Hasil nilai keluaran adalah nilai bias.

#### 4.2.6. Proses Perhitungan Kernel RBF

Perhitungan Kernel RBF merupakan proses menentukan nilai RBF untuk data *training* yang digunakan pada proses *sequential training*. Masukan berupa hasil dari *preprocessing* citra daun tanaman kubis dari data *training*. Perhitungan Kernel RBF ditunjukkan pada Gambar 4.6.

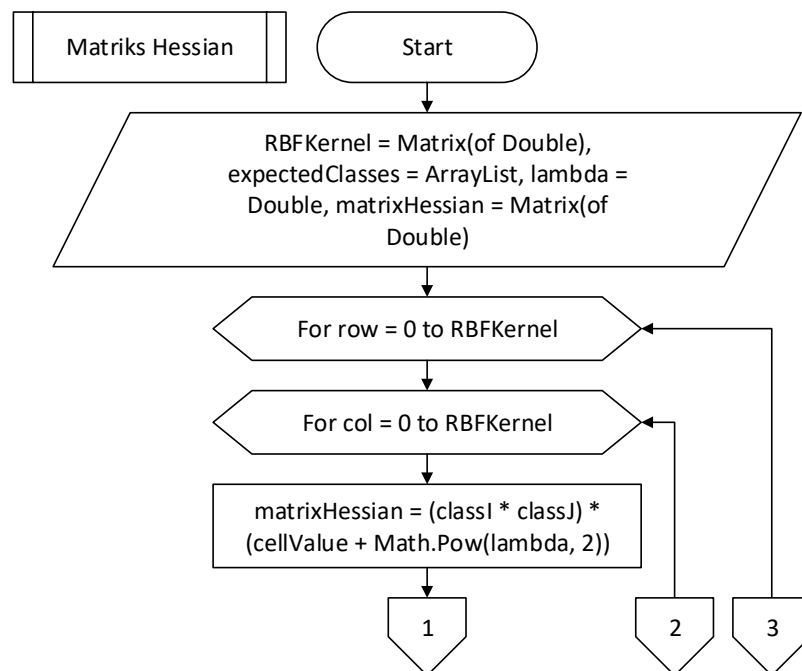


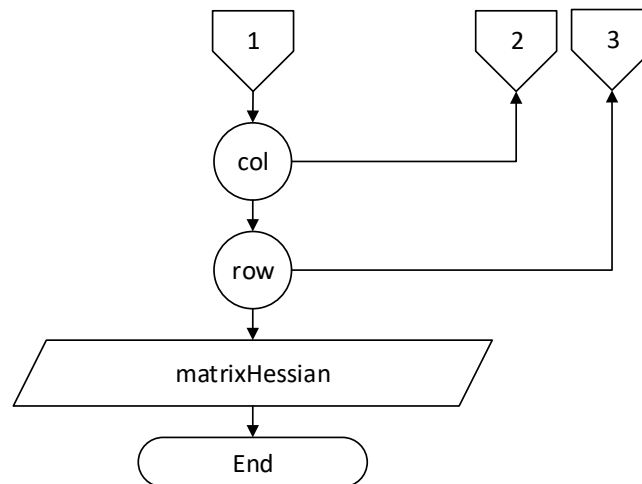


Gambar 4.6 Diagram Alir Perhitungan Kernel RBF  
(Sumber: Perancangan)

#### 4.2.7. Proses Perhitungan Matriks Hessian

Nilai masukan untuk perhitungan Matriks Hessian adalah hasil perhitungan kernel RBF, nilai kelas aktual dari tiap data training serta lambda. Hasil keluaran perhitungan berupa Matriks Hessian. Perhitungan Matriks Hessian ditunjukkan pada Gambar 4.7.

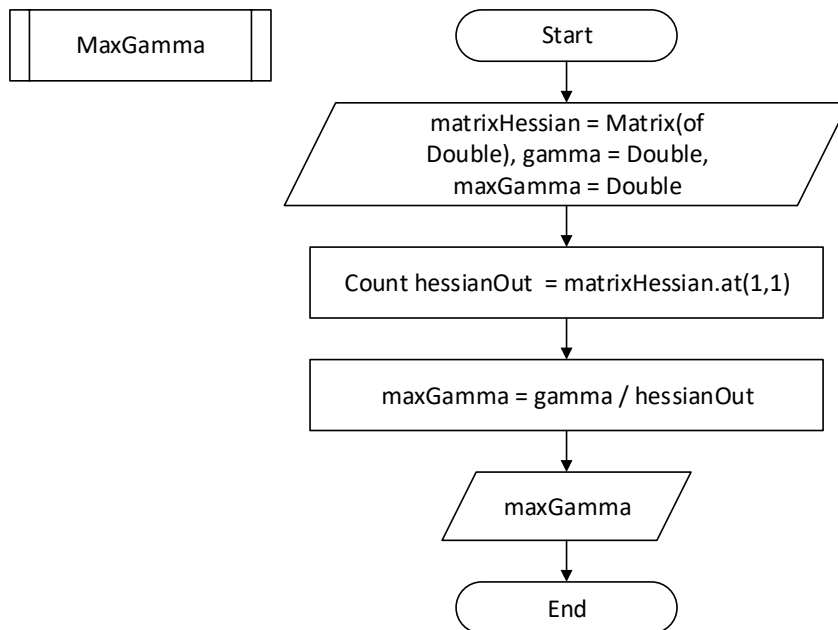




Gambar 4.7 Diagram Alir Perhitungan Matriks Hessian  
(Sumber: Perancangan)

#### 4.2.8. Proses Perhitungan *MaxGamma*

Nilai maksimal *Gamma* didapatkan dengan membagi nilai masukan dari inisialisasi nilai dari diagonal Matriks Hessian. Proses perhitungan *MaxGamma* ditunjukkan pada Gambar 4.8.

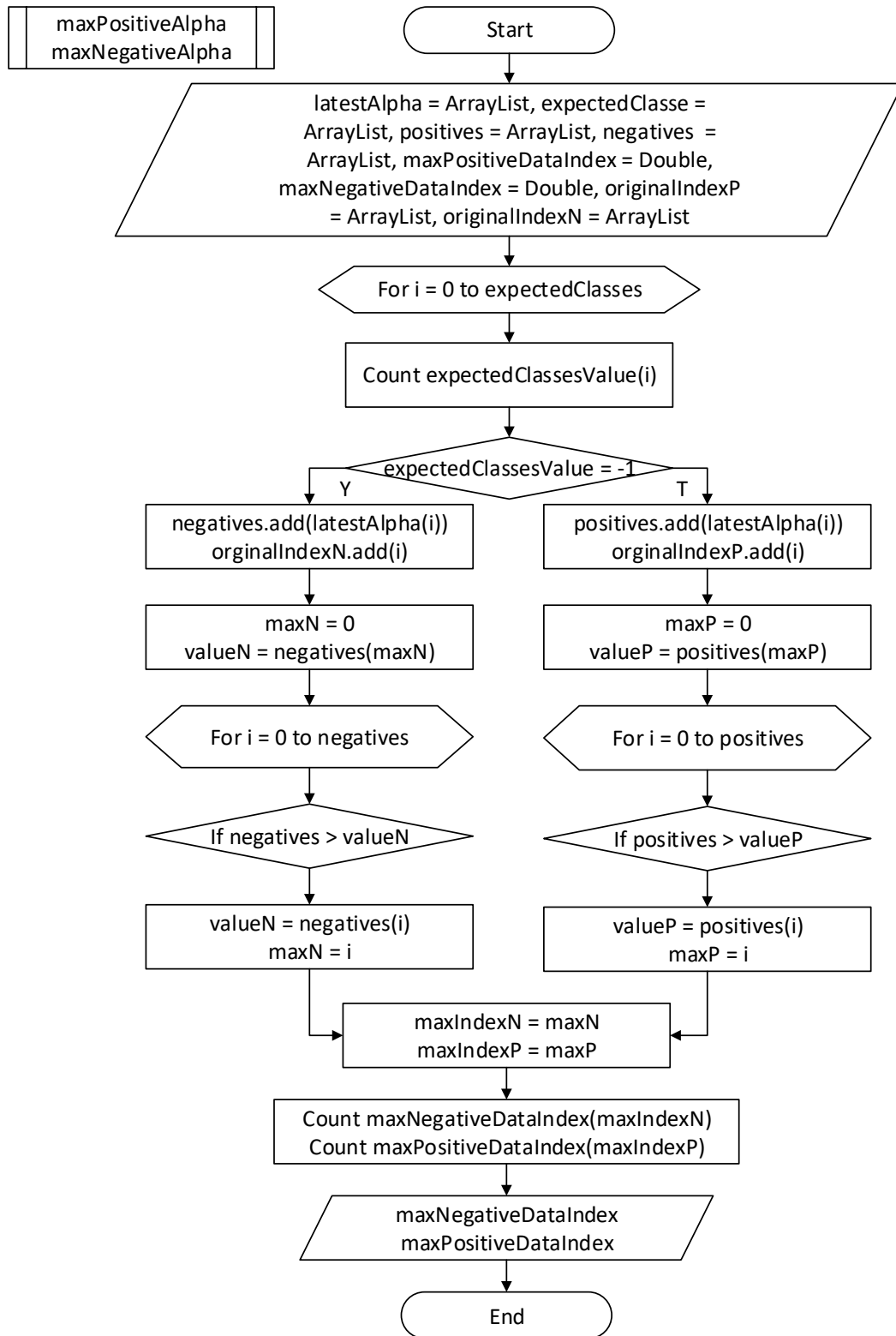


Gambar 4.8 Diagram Alir Perhitungan *MaxGamma*  
(Sumber: Perancangan)

#### 4.2.9. Proses Perhitungan *maxPositiveAlpha* dan *maxNegativeAlpha*

Perhitungan *maxPositivesAlpha* dan *maxNegativeAlpha* digunakan untuk mendapatkan nilai *alpha* tertinggi pada kelas positif dan kelas negatif. Proses

perhitungan *maxPositiveAlpha* dan *maxNegativeAlpha* ditunjukkan pada Gambar 4.9.



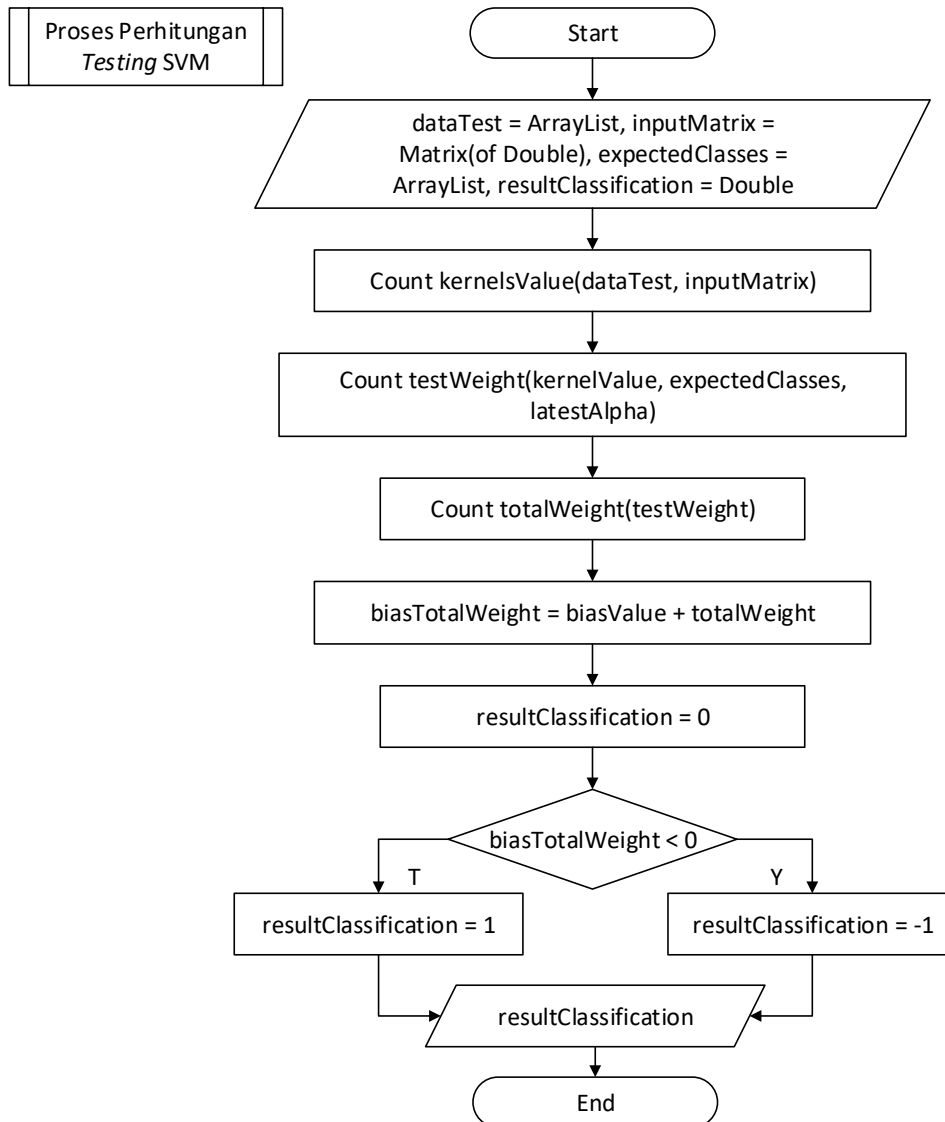
Gambar 4.9 Diagram Alir Perhitungan *maxPositiveAlpha*  
(Sumber: Perancangan)

Perhitungan *maxPositiveAlpha* digunakan untuk mendapatkan kelas positif yang mempunyai nilai *alpha* tertinggi. Masukan dalam proses ini adalah nilai *alpha* dari proses *training*, data training, kelas aktual dan jumlah data training. Hasil keluaran berupa data kelas yang memiliki nilai *alpha* tertinggi di kelas positif.

Perhitungan *maxNegativeAlpha* digunakan untuk mendapatkan kelas negatif yang mempunyai nilai *alpha* tertinggi. Masukan dalam proses ini adalah nilai *alpha* dari proses *training*, data training, kelas aktual dan jumlah data training. Hasil keluaran berupa data kelas yang memiliki nilai *alpha* tertinggi di kelas negatif.

#### 4.2.10. Proses Perhitungan Data *Testing* SVM

Proses perhitungan Data *Testing* SVM ditunjukkan pada Gambar 4.10.



Gambar 4.10 Diagram Alir Perhitungan Data Testing SVM  
(Sumber: Perancangan)



Perhitungan Data *Testing* SVM merupakan proses perhitungan untuk mendapatkan hasil klasifikasi untuk data yang diuji. Perhitungan ini mempunyai masukan berupa matriks dataset hasil normalisasi, banyak data uji banyak data training, dan kelas aktual dataset. Proses awal perhitungan adalah mencari nilai kernel dari data training dengan data testing. Kemudian menghitung nilai  $w$  dan menjumlahkan semua nilai  $w$  untuk setiap data training yang terpilih.

#### 4.2.11. Proses Perhitungan Manual Kernel RBF

Perhitungan manual kernel RBF menggunakan data berupa data *training*, berikut data *training* yang digunakan dalam perhitungan manual kernel RBF seperti ditunjukkan pada Tabel 4.1.

Tabel 4.1 Dataset

	<b>Grayscale</b>	<b>Normalisasi</b>
<b>1</b>	244	0.9000
<b>2</b>	227	0.787603
<b>3</b>	229	0.800826
<b>4</b>	235	0.840496
<b>5</b>	215	0.7083
<b>6</b>	203	0.628926
<b>7</b>	157	0.3248
<b>8</b>	186	0.5165

Sumber: Implementasi

Berikut contoh perhitungan kernel RBF untuk file D1 dengan file D2 menggunakan persamaan 2.17.

$$K(x, y) = \exp \frac{(-|| (0.9 - 0.787603) ||^2)}{2(1)^2}$$

$$K(x, y) = 0.993703$$

Data training hasil perhitungan kernel dari tiap data ditunjukkan pada Tabel 4.2.

Tabel 4.2 Hasil Perhitungan Kernel RBF

<b>K(x,y)</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>...</b>	<b>8</b>
<b>1</b>	1	0.993703	0.995094	...	0.929113
<b>2</b>	0.993703	1	0.999913	...	0.963926
<b>3</b>	0.995094	0.999913	1	...	0.960393

<b>4</b>	0.998231	0.998602	0.999213	...	0.948876
<b>5</b>	0.981787	0.996858	0.995725	...	0.981787
<b>6</b>	0.963926	0.98749	0.985334	...	0.993703
<b>7</b>	0.847528	0.898439	0.892879	...	0.981787
<b>8</b>	0.929113	0.963926	0.960393	...	1

Sumber: Implementasi

#### 4.2.12. Proses Perhitungan Manual *Training* SVM

Metode yang digunakan untuk melakukan *training* adalah *Sequential Training*. Scenario yang digunakan adalah 6:2 dengan data *training* sebanyak 6 gambar dan data *testing* sebanyak 2 gambar dari dataset sebanyak 8 gambar.

- Setelah dilakukan perhitungan kernel RBF selanjutnya adalah perhitungan *sequential training*. Langkah pertama adalah inisialisasi parameter-parameter SVM. Inisialisasi nilai  $\alpha_i = 0$ ;  $\varepsilon = 0,001$ ;  $\gamma = 0,01$ ;  $\lambda = 1$ ;  $C = 1$ ; dan nilai iterasi maksimal = 2;
- Menghitung Matriks Hessian dengan persamaan 2.11 dengan  $K(x_i, x_j)$  adalah kernel RBF, nilai  $y_i$  adalah kelas actual dari  $i$  dan  $y_j$  adalah kelas actual dari  $j$ . berikut adalah contoh dan hasil perhitungan matriks hessian pada Tabel 4.3.

$$D_{11} = 1 \times 1(0.993073 + (0,5)^2)$$

$$D_{11} = 0.79$$

Tabel 4.3 Hasil Perhitungan Matriks Hessian

<b>MH</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>...</b>	<b>8</b>
<b>1</b>	1.25	1.243703	1.245094	...	-1.17911
<b>2</b>	1.243703	1.25	1.249913	...	-1.21393
<b>3</b>	1.245094	1.249913	1.25	...	-1.21039
<b>4</b>	1.248231	1.248602	1.249213	...	-1.19888
<b>5</b>	1.231787	1.246858	1.245725	...	-1.23179
<b>6</b>	1.213926	1.23749	1.235334	...	-1.2437
<b>7</b>	-1.09753	-1.14844	-1.14288	...	1.231787
<b>8</b>	-1.17911	-1.21393	-1.21039	...	1.25

Sumber: Implementasi

- c. Menghitung nilai  $y$  baru dengan membagi nilai inisialisasi  $y$  dengan nilai maksimal dari diagonal Matriks Hessien menggunakan persamaan 2.13.

$$\gamma = \frac{\text{konstanta}}{\max D_{ii}} = 0,008$$

- d. Melakukan iterasi sebanyak inisialisasi Maksimal Iterasi yaitu 2 kali. Pada tiap iterasi melakukan perhitungan Nilai Error,  $\delta\alpha$ , dan  $\alpha$ . Nilai error didapatkan menggunakan persamaan 2.12 dan nilai  $\delta\alpha$  didapatkan menggunakan persamaan 2.14 iterasi dilakukan hingga iterasi maksimal namun jika nilai maksimal  $\delta\alpha$  kurang dari nilai epsilon maka iterasi berhenti.

- Iterasi ke-0

Menghitung nilai Error:

$$E_i = (0 \times 1.25) + (0 \times 0.79) + \dots + (0 \times (-0.82))$$

$$E_i = 0$$

Berikut hasil perhitungan nilai error pada Tabel 4.4.

Tabel 4.4 Hasil Perhitungan Nilai Error

<b>Ei</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>...</b>	<b>8</b>	<b>Ei</b>
<b>1</b>	0.00	0.00	0.00	...	0.00	0.00
<b>2</b>	0.00	0.00	0.00	...	0.00	0.00
<b>3</b>	0.00	0.00	0.00	...	0.00	0.00
<b>4</b>	0.00	0.00	0.00	...	0.00	0.00
<b>5</b>	0.00	0.00	0.00	...	0.00	0.00
<b>6</b>	0.00	0.00	0.00	...	0.00	0.00
<b>7</b>	0.00	0.00	0.00	...	0.00	0.00
<b>8</b>	0.00	0.00	0.00	...	0.00	0.00

Sumber: Implementasi

Menghitung nilai  $\delta\alpha$ :

$$\delta\alpha = \min[\text{maks}[0.008(1 - 0), -0], 1 - 0]$$

$$\delta\alpha = 0.008$$

Memperbarui nilai  $\alpha_i$ :

$$\alpha_i = \alpha_i + \delta\alpha$$

$$\alpha_i = 0 + 0.008$$

$$\alpha_i = 0.008$$

Pengecekan iterasi apabila nilai ( $|\delta\alpha| < \varepsilon$ ) maka iterasi dihentikan. Nilai maksimal  $\delta\alpha$  yang didapatkan pada iterasi pertama adalah 0.008 sehingga  $\delta\alpha$  belum memenuhi nilai konvergen iterasi dilanjutkan.

- Iterasi ke-1

Menghitung Nilai Error:

$$E_i = (0.008 \times 1.25) + (0.008 \times 1.243703) + \dots + (0.008 \times (-1.17911))$$

$$E_i = 0.01$$

Berikut hasil perhitungan nilai error pada Tabel 4.5.

Error	1	2	3	...	8
1	0.01	0.00995	0.009961	...	-0.00943
2	0.00995	0.01	0.009999	...	-0.00971
3	0.009961	0.009999	0.01	...	-0.00968
4	0.009986	0.009989	0.009994	...	-0.00959
5	0.009854	0.009975	0.009966	...	-0.00985
6	0.009711	0.0099	0.009883	...	-0.00995
7	-0.00878	-0.00919	-0.00914	...	0.009854
8	-0.00943	-0.00971	-0.00968	...	0.01

Sumber: Implementasi

Menghitung nilai  $\delta\alpha$ :

$$\delta\alpha = \min[\max[0.008(1 - 0), -0], 1 - 0]$$

$$\delta\alpha = 0.008$$

Memperbarui nilai  $\alpha$ :

$$\alpha_i = \alpha_i + \delta\alpha$$

$$\alpha_i = 0 + 0.008$$

$$\alpha_i = 0.008$$

- e. Proses selanjutnya adalah mencari nilai *weight* ( $w$ ) dan *bias* ( $b$ ) dari proses *training*. Terdapat dua kelas sehingga terdapat dua nilai  $w$  yaitu nilai  $w.x^+$  untuk kelas positif dan  $w.x^-$  untuk kelas negatif.  $x^+$  merupakan *support vector* dari kelas positif yang memiliki nilai alpha tertinggi dan  $x^-$  merupakan *support vector* dari kelas negatif yang memiliki nilai alpha tertinggi di kelasnya. Untuk mendapatkan nilai *support vector* menggunakan persamaan 2.10.

$$K(x, x^+) = \exp \frac{(-|(0.90 - 0.113233)|^2)}{2(1)^2}$$

$$K(x, x^+) = 0.733807$$

$$K(x, x^-) = \exp \frac{(-|(0.90 - 0.873554)|^2)}{2(1)^2}$$

$$K(x, x^-) = 0.99965$$

Hasil perhitungan nilai  $K(x, x^+)$  dan nilai  $K(x, x^-)$  ditunjukkan pada Tabel 4.6.

Tabel 4.6 Hasil perhitungan nilai  $K(x, x^+)$  dan nilai  $K(x, x^-)$

	<b>K(x, x+)</b>	<b>K(x, x-)</b>
<b>1</b>	0.733807	0.99965
<b>2</b>	0.796607	0.996313
<b>3</b>	0.789465	0.997359
<b>4</b>	0.767618	0.999454
<b>5</b>	0.837749	0.986433
<b>6</b>	0.875487	0.970522
<b>7</b>	0.977868	0.860218
<b>8</b>	0.921891	0.938255

Sumber: Implementasi

- f. Menghitung nilai  $w.x^+$  dan  $w.x^-$  untuk mendapatkan nilai bias menggunakan persamaan 2.16 hingga 2.17 yang ditunjukkan pada Tabel 4.7.

$$w.x^+ = 0.01591 \times 1 \times 0.733807$$

$$w.x^+ = 0.011674$$

$$w.x^- = 0.01591 \times 1 \times 0.99965$$

$$w.x^- = 0.015903$$

Tabel 4.7 Hasil perhitungan  $w.x^+$  dan  $w.x^-$

	<b>(w, x+)</b>	<b>(w, x-)</b>
<b>1</b>	0.011674	0.015903
<b>2</b>	0.012693	0.015875
<b>3</b>	0.012577	0.015889
<b>4</b>	0.012222	0.015913
<b>5</b>	0.013365	0.015737
<b>6</b>	0.013984	0.015502
<b>7</b>	-0.01559	-0.01372

**8** -0.01475 -0.01501

Sumber: Implementasi

- g. Menghitung nilai *bias* (b) menggunakan persamaan 2.18.

$$b = -\frac{1}{2}(w \cdot x^+ + w \cdot x^-)$$

$$b = -\frac{1}{2}((-0.02472) + 0.021391)$$

$$b = 0.001662$$

#### 4.2.13. Proses Perhitungan Manual *Testing* SVM

Proses selanjutnya setelah proses *training* adalah proses *testing* SVM. Data yang digunakan adalah 2 data citra. Berikut merupakan data *testing* yang ditunjukkan pada Tabel 4.8.

Tabel 4.8 Data *Testing*

	<i>Grayscale</i>	<i>Normalisasi</i>
<b>T1</b>	234	0.833884
<b>T2</b>	190	0.542975

Sumber: Implementasi

- a. Langkah pertama dalam proses *testing* adalah menghitung nilai  $K(x, x_i)$  untuk digunakan menghitung  $w$ . Pada proses *testing* nilai yang digunakan adalah nilai data *training*, sedangkan nilai  $x_i$  adalah data *testing*. Berikut contoh perhitungan  $K(x, x_i)$  untuk data *testing* 1.

$$K(x, x^+) = \exp \frac{(-|(0.90 - 0.833884)|^2)}{2(1)^2}$$

$$K(x, x^+) = 0.997817$$

- b. Langkah selanjutnya adalah menghitung nilai  $w$  dengan nilai  $\alpha$  hasil dari perhitungan proses *training*. Nilai  $y_i$  adalah kelas dari data *training*. Berikut contoh perhitungan nilai  $w$  untuk data *testing* 1.

$$w \cdot x^+ = 0.01591 \times 1 \times 0.733807$$

$$w \cdot x^+ = 0.997817$$

Hasil perhitungan nilai  $K(x, x_i)$  dan nilai  $w$  untuk data *testing* 1 ditunjukkan pada Tabel 4.9.

Tabel 4.9 Hasil Perhitungan Nilai  $K(x, x_i)$  dan  $w$  untuk data *testing*

	$K(x_i, x)$	$w, x$
<b>1</b>	0.997817	0.015874
<b>2</b>	0.99893	0.015917
<b>3</b>	0.999454	0.015922
<b>4</b>	0.999978	0.015921
<b>5</b>	0.992141	0.015828
<b>6</b>	0.979215	0.015641
<b>7</b>	0.878458	-0.01401
<b>8</b>	0.95089	-0.01521

Sumber: Implementasi

- b. Proses selanjutnya adalah menghitung nilai  $f(x)$  dengan menggunakan persamaan  $\sum_{i=1}^m a_i y_i K(x_i, x) + b$ , kemudian menentukan kelas dari data *testing* dengan fungsi sign. Berikut contoh perhitungan untuk data *testing* 1.
- $$f(x) = 0.001662 + 0.19193$$
- $$f(x) = 0.020855$$
- c. Hasil perhitungan dari proses *testing* untuk data *testing* 1 adalah 1 yang berarti data *testing* 1 diklasifikasikan sebagai penyakit bercak daun.
- d. Melakukan proses a hingga c untuk data *testing* 2.

Tabel 4.10 Hasil Klasifikasi

	Kelas	Hasil	Penyakit	Penyakit Hasil
	Aktual	Klasifikasi	Aktual	Klasifikasi
<b>T1</b>	1	1	Bercak Daun	Bercak Daun
<b>T2</b>	-1	-1	Busuk Hitam	Busuk Hitam

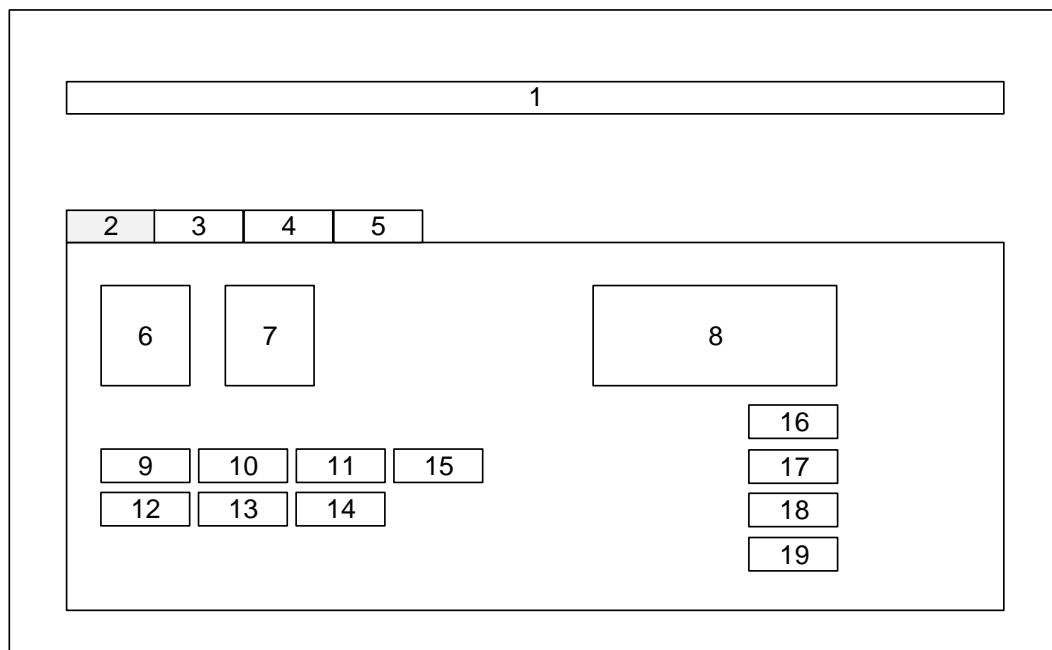
Sumber: Implementasi

#### 4.3. Perancangan Antarmuka (*Interface*)

Perancangan antarmuka identifikasi penyakit daun tanaman kubis menggunakan SVM ini terdiri dari tiga tab halaman. Tab pertama berisi tab *training*, yang berisikan *preprocessing* data yang berguna untuk proses *training*, serta input parameter yang digunakan untuk proses *training*. Tab kedua berisi tab berisikan *preprocessing* untuk data *testing*, yang kemudian akan menampilkan hasil dari klasifikasi citra daun tanaman kubis.

a. Perancangan antarmuka *Training*

Perancangan antarmuka *training* ditunjukkan pada Gambar 4.11.



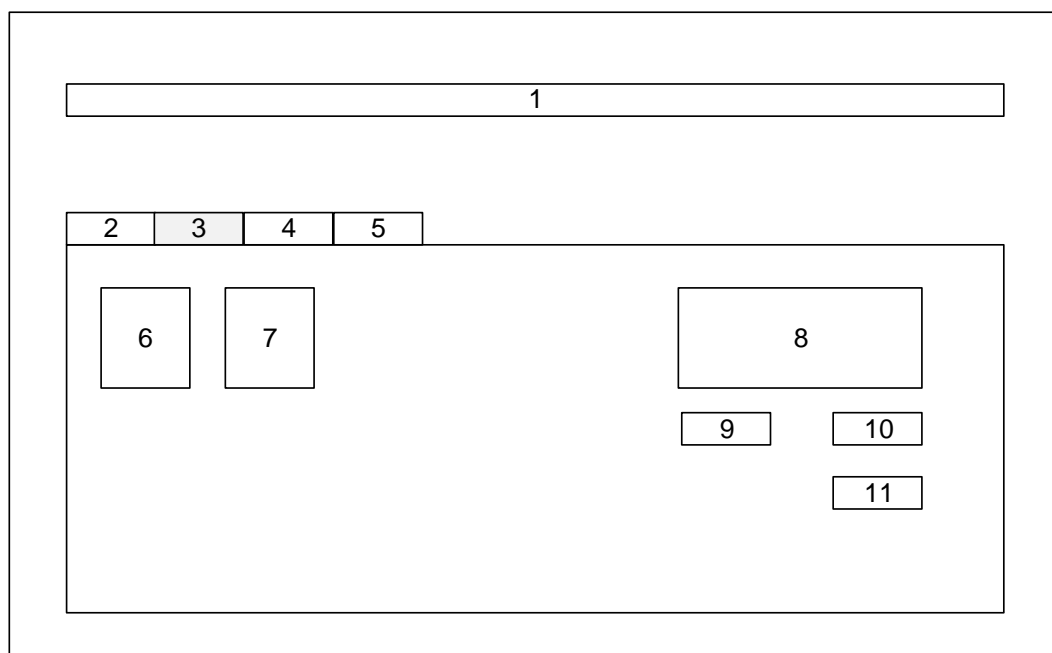
Gambar 4.11 Perancangan Antarmuka Training  
(Sumber: Perancangan)

Penjelasan dari Gambar 4.11 adalah sebagai berikut:

1. Judul dari aplikasi penelitian.
2. Tab *Training* yang berisikan proses yang dilakukan dan parameter yang dibutuhkan untuk *training*.
3. Tab *Testing* yang berisikan proses yang dilakukan dan hasil klasifikasi yang dihasilkan saat proses *testing*.
4. Tab *Help* yang berisikan pengenalan menu pada sistem.
5. Tab *About* yang berisikan informasi mengenai pembuatan aplikasi.
6. *PictureBox* untuk data citra asli.
7. *PictureBox* untuk data citra yang telah dilakukan proses *segmentation*.
8. *DataGridView* untuk data citra yang digunakan untuk proses *training*.
9. *InputBox* untuk parameter *Sigma*.
10. *InputBox* untuk parameter *Gamma*.
11. *InputBox* untuk parameter *Lambda*.
12. *InputBox* untuk parameter *Complexity*.
13. *InputBox* untuk parameter *Epsilon*.



14. *InputBox* untuk parameter *Alpha*.
  15. *InputBox* untuk parameter *Iterasi Maksimal*.
  16. *Button* untuk melakukan proses *segmentation*.
  17. *Button* untuk melakukan proses simpan nilai yang dihasilkan dari proses *segmentation* di database.
  18. *Button* untuk melakukan proses perhitungan *training*.
  19. *Button* untuk melakukan proses *reset*.
- b. Perancangan *Testing*
- Perancangan antarmuka *training* ditunjukkan pada Gambar 4.12.



Gambar 4.12 Perancangan Antarmuka *Testing*  
(Sumber: Perancangan)

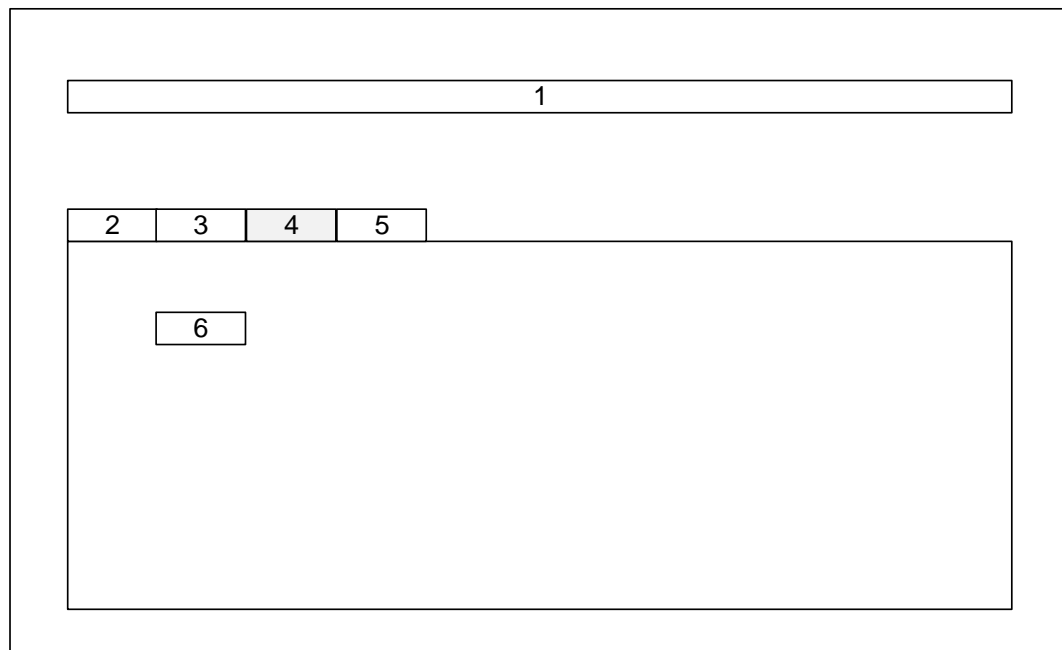
Penjelasan dari Gambar 4.12 adalah sebagai berikut :

1. Judul dari aplikasi penelitian.
2. Tab *Training* yang berisikan proses yang dilakukan dan parameter yang dibutuhkan untuk *training*.
3. Tab *Testing* yang berisikan proses yang dilakukan dan hasil klasifikasi yang dihasilkan saat proses *testing*.
4. Tab *Help* yang berisikan pengenalan menu pada sistem.
5. Tab *About* yang berisikan informasi mengenai pembuatan aplikasi.
6. *PictureBox* untuk data citra asli.

7. *PictureBox* untuk data citra yang telah diolah menjadi *segmentation*.
8. *DataGridView* data citra yang digunakan untuk proses *training*.
9. *Label* untuk menunjukkan hasil dari klasifikasi.
10. *Button* untuk melakukan proses *testing*.
11. *Button* untuk melakukan proses *reset*.

c. Perancangan *Help*

Perancangan antarmuka *Help* ditunjukkan pada Gambar 4.12.



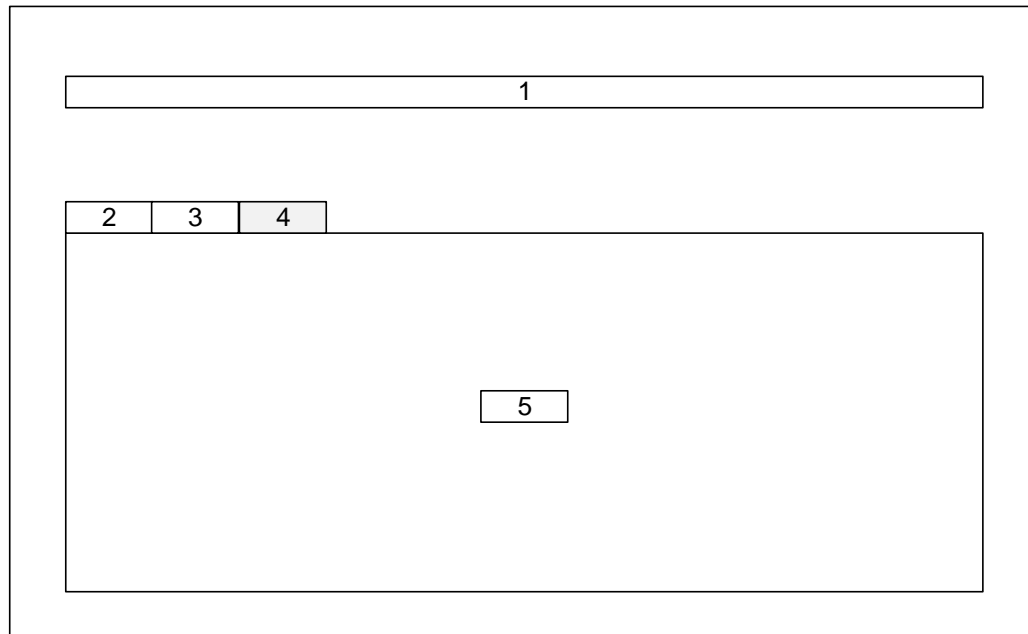
Gambar 4.13 Perancangan Antarmuka *Help*  
(Sumber: Perancangan)

Penjelasan dari Gambar 4.13 adalah sebagai berikut :

1. Judul dari aplikasi penelitian.
2. Tab *Training* yang berisikan proses yang dilakukan dan parameter yang dibutuhkan untuk *training*.
3. Tab *Testing* yang berisikan proses yang dilakukan dan hasil klasifikasi yang dihasilkan saat proses *testing*.
4. Tab *Help* yang berisikan pengenalan menu pada sistem.
5. Tab *About* yang berisikan informasi mengenai pembuatan aplikasi.
6. *Label* yang berisikan pengenalan menu pada sistem.

d. Perancangan *About*

Perancangan antarmuka *about* ditunjukkan pada Gambar 4.14.



Gambar 4.14 Perancangan Antarmuka *About*  
(Sumber: Perancangan)

Penjelasan dari Gambar 4.14 adalah sebagai berikut :

1. Judul dari aplikasi penelitian.
2. Tab *Training* yang berisikan proses yang dilakukan dan parameter yang dibutuhkan untuk *training*.
3. Tab *Testing* yang berisikan proses yang dilakukan dan hasil klasifikasi yang dihasilkan saat proses *testing*.
4. Tab *Help* yang berisikan pengenalan menu pada sistem.
5. Tab *About* yang berisikan informasi mengenai pembuatan aplikasi.
6. *Label* yang berisikan informasi pembuatan sistem.

#### 4.4. Perancangan Pengujian dan Analisis

Pengujian dari perangkat lunak ini berkaitan dengan pengujian sistem, apakah sistem sudah berjalan dengan baik. Seberapa baik kemampuan sistem dalam melakukan perhitungan sesuai dengan *source code* yang sudah tertulis dalam sistem.

Pengujian dan analisis sistem dilakukan dengan membandingkan data asli terhadap data hasil klasifikasi menggunakan *Support Vector Machine* (SVM). Hasil tersebut berbentuk nilai akurasi dari sistem dalam mengklasifikasikan penyakit

daun tanaman Kubis. Skenario pengujian yang dilakukan dalam penelitian ini adalah sebagai berikut:

Pengujian ini dilakukan untuk melihat pengaruh parameter pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai parameter yang memberi nilai akurasi terbaik.

1. Pengujian pada nilai sigma

Pengujian ini dilakukan untuk melihat pengaruh nilai *sigma* pada perhitungan Kernel RBF. Pengujian ini bertujuan untuk mendapatkan hasil terbaik pada akurasi sistem. Variasi nilai *sigma* adalah 1, 2, 3, 4, dan 5.

2. Pengujian pada parameter Lambda

Pengujian ini dilakukan untuk melihat pengaruh nilai *lambda* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Lamda* yang memberi nilai akurasi terbaik. Variasi nilai *lambda* adalah 0.1, 0.2, 0.3, 0.4, dan 0.5.

3. Pengujian pada nilai Gamma

Pengujian ini dilakukan untuk melihat pengaruh nilai *gamma* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Gamma* yang memberi nilai akurasi terbaik. Variasi nilai *gamma* adalah 0.01, 0.1, 0.2, 0.3, 0.4, dan 0.5.

4. Pengujian pada nilai Complexity

Pengujian ini dilakukan untuk melihat pengaruh nilai *complexity* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Complexity* yang memberi nilai akurasi terbaik. Variasi nilai *complexity* adalah 1, 2, 3, 4, dan 5.

5. Pengujian pada nilai Epsilon

Pengujian ini dilakukan untuk melihat pengaruh nilai *epsilon* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Epsilon* yang memberi nilai akurasi terbaik. Variasi nilai *epsilon* adalah 0.0001, 0.0002, 0.0005, 0.001, dan 0.01.

6. Pengujian pada nilai Iterasi Maksimal

Pengujian ini dilakukan untuk melihat pengaruh nilai iterasi maksimal pada *Sequential Training*. Iterasi maksimal ditentukan untuk batas nilai iterasi maksimal

pada proses training dengan variasi nilai yang digunakan adalah 2, 5, 10, 25, dan 50. Parameter-parameter SVM yang digunakan adalah nilai  $\alpha_i = 0$ ,  $\varepsilon = 0.001$ ,  $\gamma = 0.01$ ,  $\lambda = 1$ ,  $C = 1$ . Tujuan dari pengujian ini adalah melihat pengaruh Iterasi Maksimal terhadap hasil akurasi dari sistem.

#### **4.5. Pengambilan Keputusan**

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil sesuai dengan hasil pengujian dan analisis sistem yang sudah dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan memperbaiki penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

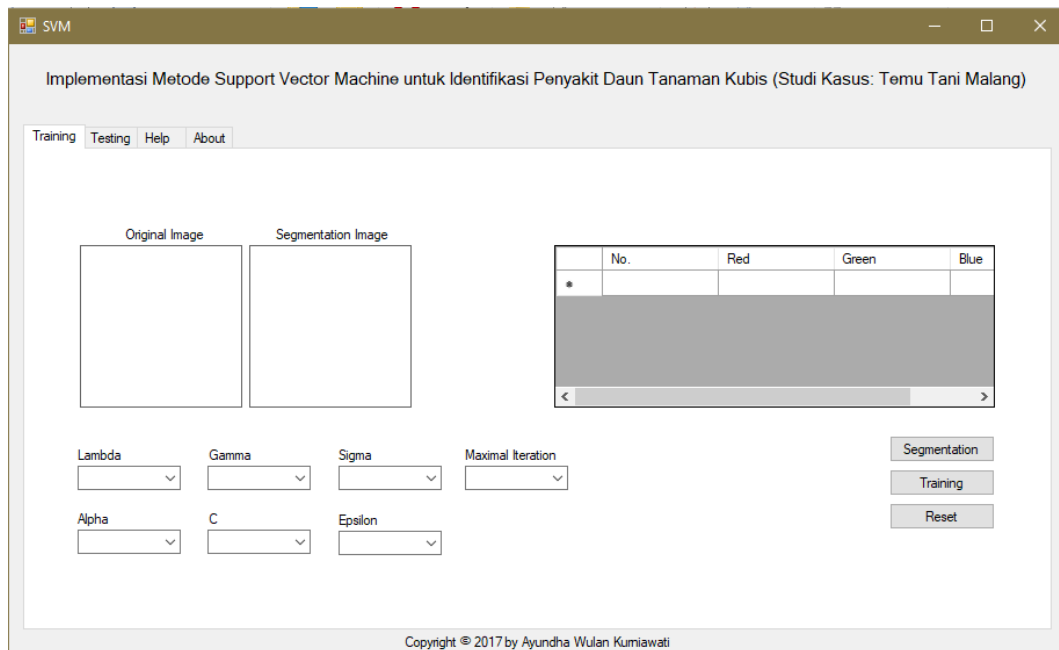
## BAB V. IMPLEMENTASI

### 5.1. Implementasi Antarmuka

Implementasi Antarmuka merupakan implementasi hasil perancangan antarmuka yang sudah dibuat di BAB IV. Pada penelitian ini terdapat 2 antarmuka yaitu, antarmuka *Training*, antarmuka *Testing*, antarmuka *About*, dan antarmuka *Help*.

#### 5.1.1. Implementasi antarmuka *Training*

Antarmuka *training* adalah antarmuka yang digunakan untuk proses *training data*. Sebelum melakukan proses *training*, dilakukan proses *preprocessing* data yaitu perbaikan kualitas citra dengan menggunakan *Histogram Equalization* kemudian dilakukan *segmentation* dengan menghilangkan *background* citra masukan kemudian dilakukan proses *grayscale* kemudian dari nilai yang didapatkan akan diproses perhitungan seperti yang ditunjukkan Gambar 5.1.



Gambar 5.1 Implementasi Antarmuka *Training*  
(Sumber: Implementasi)

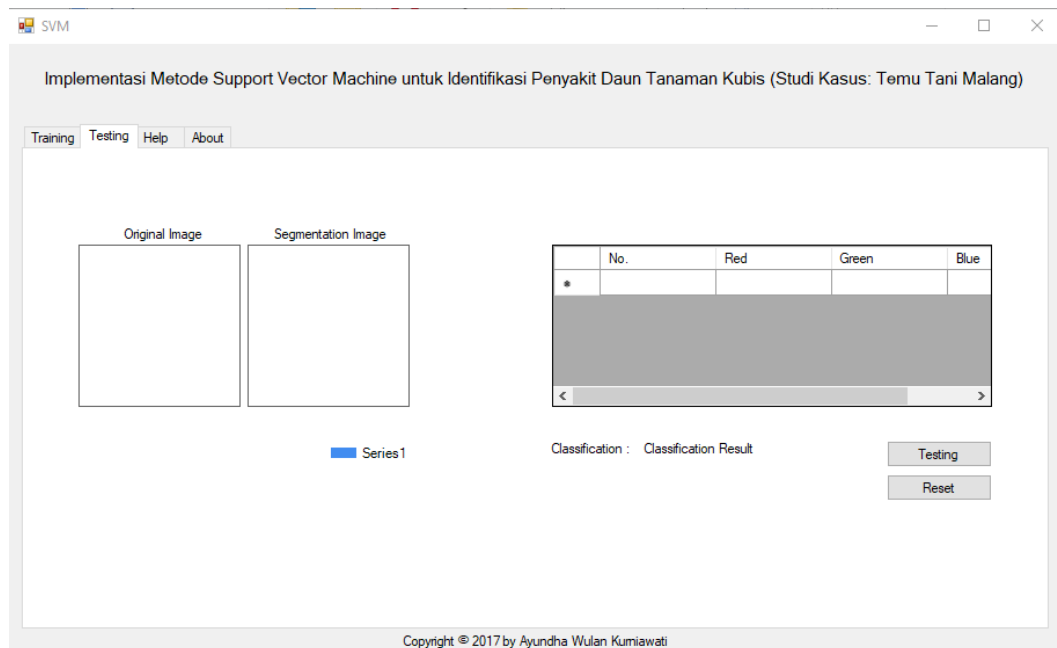
Berikut ini merupakan langkah-langkah melakukan proses *training*:

1. Klik pada *picturebox Original Image* untuk memilih citra yang akan disegmentasi.

2. Pada *picturebox Segmentation Image* akan muncul hasil citra masukan yang telah dilakukan proses perbaikan citra, segmentasi serta *grayscale*.
3. Klik *button Segmentation* untuk melakukan proses segmentasi serta untuk menyimpan nilai citra masukan yang telah diproses pada database.
4. Memasukkan nilai parameter-parameter untuk memulai proses perhitungan *training*.
5. Klik *button Training* untuk melakukan proses perhitungan *training*, setelah proses *training* selesai nilai hasil *training* seperti nilai *alpha*, *bias* dan *sigma* akan disimpan pada database.
6. Klik *button Reset* untuk membersihkan data yang telah digunakan untuk *training*.

#### 5.1.2. Implementasi antarmuka *Testing*

Antarmuka *testing* adalah antarmuka yang menyajikan data hasil testing. Sebelum melakukan proses *testing*, dilakukan proses *preprocessing* data yaitu perbaikan kualitas citra dengan menggunakan *Histogram Equalization* kemudian dilakukan *segmentation* dengan menghilangkan *background* citra masukan kemudian dilakukan proses *grayscale* kemudian dari nilai yang didapatkan akan diproses untuk perhitungan *testing* seperti yang ditunjukkan Gambar 5.2.



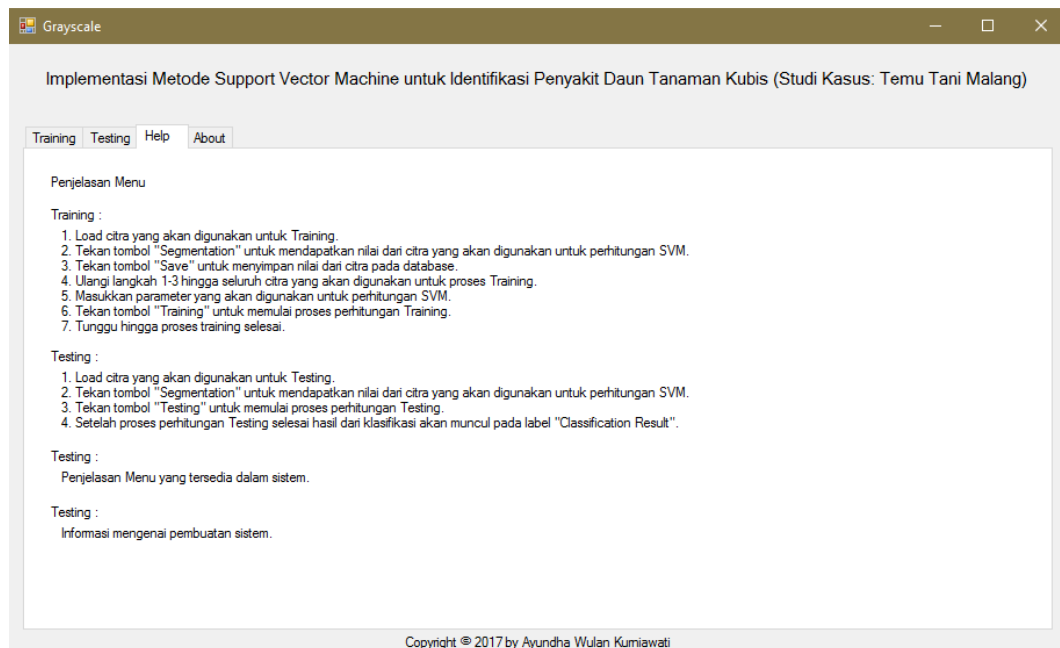
Gambar 5.2 Implementasi Antarmuka *Testing*  
(Sumber: Implementasi)

Berikut ini merupakan langkah-langkah melakukan proses *testing*:

1. Klik pada *picturebox Original Image* untuk memilih citra yang akan disegmentasi.
2. Pada *picturebox Segmentation Image* akan muncul hasil citra masukan yang telah dilakukan proses perbaikan citra, segmentasi serta *grayscale*.
3. Klik *button Testing* untuk melakukan proses perhitungan segmentasi terlebih dahulu yang kemudian nilai yang didapatkan akan digunakan untuk perhitungan *testing* dengan menggunakan nilai dari proses *training* yang telah disimpan sebelumnya.
4. Klik *button Reset* untuk membersihkan data yang telah digunakan untuk *training*.

### 5.1.3. Implementasi antarmuka *Help*

Antarmuka *Help* merupakan antarmuka yang berisikan informasi menjalankan menu yang terdapat pada sistem seperti yang ditunjukkan pada Gambar 5.3.

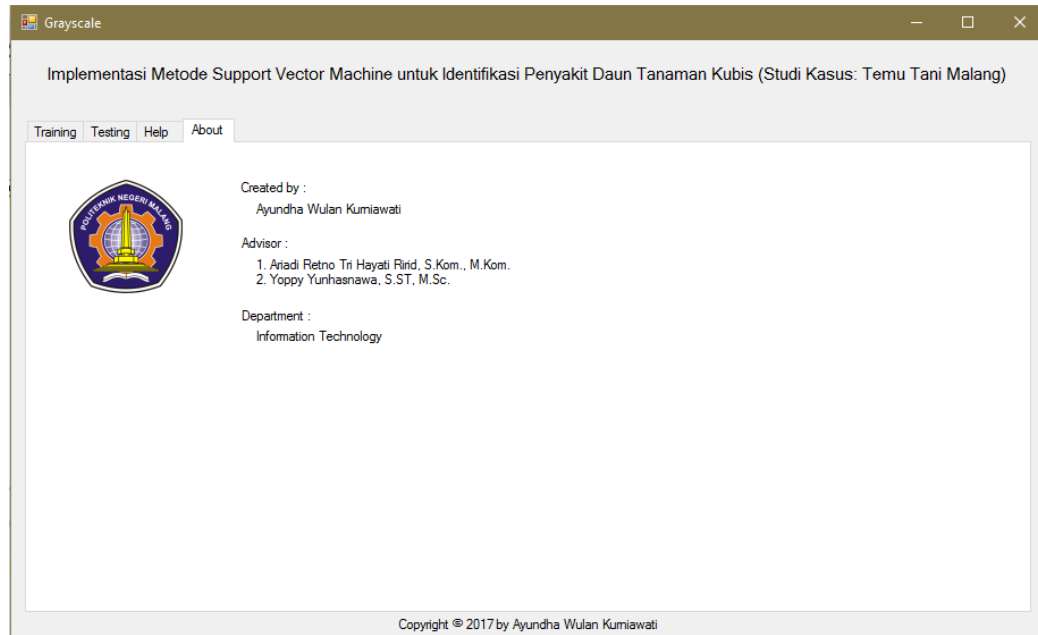


Gambar 5.3 Implementasi Antarmuka *Help*  
(Sumber: Implementasi)

### 5.1.4. Implementasi antarmuka *About*

Antarmuka *About* ini merupakan antarmuka yang berisikan informasi mengenai program seperti ditunjukkan dalam Gambar 5.4.





Gambar 5.4 Implementasi Antarmuka *About*  
(Sumber: Implementasi)

## 5.2. Implementasi Sistem

Pada Sub bab ini akan menjelaskan mengenai implementasi sistem yang sudah dirancang di bab sebelumnya. Implementasi yang dijelaskan adalah implementasi metode *grayscale*, *threshold*, proses *training* dan *testing* SVM.

### 5.2.1. Implementasi *Histogram Equalization*

Proses yang pertama dilakukan adalah *Histogram Equalization* yang digunakan untuk memperbaiki citra masukan, sehingga dapat dilakukan segmentasi dengan baik. Berikut merupakan implementasi *Histogram Equalization* yang ditunjukkan pada Gambar 5.5.

```
Private Sub histogramEqualization()
    Dim image1 As New Bitmap(pbCAsli.Image)
    pbCGrays.Image = image1
    Dim baris, kolom As Integer
    Dim merah, hijau, biru, r, g, b As Integer
    Dim nR(256), nG(256), nB(256), Sr(256), Sg(256), Sb(256)
    As Integer
    ProgressBar1.Show()

    For counter = 0 To 255
        Sr(counter) = 0
        Sg(counter) = 0
        Sb(counter) = 0
    Next
    For baris = 0 To image1.Width - 1
        For kolom = 0 To image1.Height - 1
```

```

        merah = image1.GetPixel(baris, kolom).R
        hijau = image1.GetPixel(baris, kolom).G
        biru = image1.GetPixel(baris, kolom).B

        nR(merah) += 1
        nG(hijau) += 1
        nB(biru) += 1

    Next

Next

Dim resolution As Integer
resolution = image1.Width * image1.Height
Sr(0) = CInt(255 * (nR(0) / resolution))
Sg(0) = CInt(255 * (nG(0) / resolution))
Sb(0) = CInt(255 * (nB(0) / resolution))

Dim i As Integer
For i = 1 To 256
    Dim x As Integer
    x = i - 1
    Sr(i) = CInt((255 * (nR(i) / resolution)) + Sr(x))
    Sg(i) = CInt((255 * (nG(i) / resolution)) + Sg(x))
    Sb(i) = CInt((255 * (nB(i) / resolution)) + Sb(x))
Next

For baris = 0 To image1.Width - 1
    For kolom = 0 To image1.Height - 1
        merah = image1.GetPixel(baris, kolom).R
        hijau = image1.GetPixel(baris, kolom).G
        biru = image1.GetPixel(baris, kolom).B

        r = Sr(merah)
        g = Sg(hijau)
        b = Sb(biru)

        Dim red As Integer = Truncate(r)
        Dim green As Integer = Truncate(g)
        Dim blue As Integer = Truncate(b)

        image1.SetPixel(baris, kolom,
Color.FromArgb(red, green, blue))
    Next
    Me.Text = Int(100 * baris / image1.Width).ToString &
"%"
    ProgressBar1.Value = Int(100 * baris / image1.Width)
Next
ProgressBar1.Hide()
Me.Text = "Equalization"
pbCGrays.Refresh()

End Sub

```

Gambar 5.5 Sourcecode implementasi Histogram Equalization  
(Sumber: Implementasi)

### 5.2.2. Implementasi Segmentasi

Proses selanjutnya setelah melakukan proses *Histogram Equalization* adalah proses *thresholding* untuk membedakan *background* dan *foreground*, untuk selanjutnya nilai dari proses ini akan dikonversi menjadi nilai *binary* sehingga dapat digunakan untuk proses selanjutnya. Berikut merupakan implementasi *thresholding* yang ditunjukkan pada Gambar 5.6.

```
Public Shared Function threshold()
    Dim image2 As New Bitmap(Form2.pbCGrays.Image)
    Form2.pbCThresh.Image = image2
    Dim baris, kolom As Integer
    Dim merah, hijau, biru As Integer
    Dim biner As Integer

    For baris = 0 To image2.Width - 1
        For kolom = 0 To image2.Height - 1
            merah = image2.GetPixel(baris, kolom).R
            hijau = image2.GetPixel(baris, kolom).G
            biru = image2.GetPixel(baris, kolom).B

            If (merah <= 127) Then
                biner = 1
            ElseIf (merah >= 128) Then
                biner = 255
            End If

            image2.SetPixel(baris, kolom,
Color.FromArgb(biner, biner, biner))
        Next
        Form2.ProgressBar1.Increment(1)
    Next

    Form2.ProgressBar1.Hide()
    Form2.pbCThresh.Refresh()

    Return image2
End Function
```

Gambar 5.6 *Sourcecode* implementasi *Threshold*  
(Sumber: Implementasi)

### 5.2.2. Implementasi *Grayscale*

Proses awal yang diimplementasikan untuk identifikasi penyakit daun tanaman kubis adalah proses metode *grayscale* yang nilainya akan digunakan untuk proses perhitungan SVM. Berikut implementasi *grayscale* yang ditunjukkan pada Gambar 5.7.

```
Public Shared Function grayscale()
    Dim image1 As New Bitmap(Form2.pbCAsli.Image)
    Form2.pbCGrays.Image = image1
    Dim baris, kolom As Integer
```

```

Dim merah, hijau, biru, abu As Integer
Form2.ProgressBar1.Show()

For baris = 0 To image1.Width - 1
    For kolom = 0 To image1.Height - 1
        merah = image1.GetPixel(baris, kolom).R
        hijau = image1.GetPixel(baris, kolom).G
        biru = image1.GetPixel(baris, kolom).B
        abu = CInt((merah + hijau + biru) / 3)
        image1.SetPixel(baris, kolom,
Color.FromArgb(abu, abu, abu))
    Next
    Form2.lbImageResult.Text = abu

    Form2.Text = Int(100 * baris /
image1.Width).ToString & "%"
    Form2.ProgressBar1.Value = Int(100 * baris /
image1.Width)
Next
Form2.ProgressBar1.Hide()
Form2.Text = "Grayscale"
Form2.pbCGrays.Refresh()
Return image1
End Function

```

Gambar 5.7 Sourcecode Implementasi Grayscale  
(Sumber: Implementasi)

### 5.2.3. Implementasi Perhitungan Kernel RBF

Setelah dilakukan proses *thresholding* proses berikutnya adalah perhitungan nilai Kernel RBF, yang digunakan untuk perhitungan *training* SVM. Berikut implementasi fungsi Kernel RBF ditunjukkan pada Gambar 5.8.

```

Public Function countRBFKernel(inMatrix As Matrix(Of
Double))
    Dim outputMatrix As Matrix(Of Double) = inMatrix.Clone()

    For row = 0 To inMatrix.RowCount - 1
        For col = 0 To inMatrix.ColumnCount - 1
            Dim totalRowValues As Double = 0
            For clm = 0 To inMatrix.ColumnCount - 1
                Dim cellValueX As Double = inMatrix.At(row,
clm)
                Dim cellValueY As Double = inMatrix.At(row,
clm)
                Dim calcXY As Double = cellValueX -
cellValueY
                Dim powXY As Double = Math.Pow(calcXY, 2)
                totalRowValues += powXY
            Next
            Dim cellRBF As Double = totalRowValues * (-1)
            cellRBF = cellRBF / (2 * (Math.Pow(sigma, 2)))
            cellRBF = Math.Exp(cellRBF)
            outputMatrix.At(row, col, cellRBF)
        Next
    Next
Next

```

```
Return outputMatrix
End Function
```

Gambar 5.8 *Sourcecode* Implementasi Kernel RBF  
(Sumber: Implementasi)

#### 5.2.4. Implementasi Proses *Training* SVM

Untuk dapat mengklasifikasi penyakit daun diperlukan proses pembelajaran data. Pada penelitian ini proses pembelajaran data yang digunakan adalah proses *training* SVM yaitu *Sequential Training*. Berikut implementasi fungsi *training* SVM *Sequential Training* ditunjukkan pada Gambar 5.9.

```
Public Sub learn()

    Dim outputM As Matrix(Of Double) =
kernelCal(Me.inputMatrix)
    For row = 0 To outputM.RowCount - 1
        For col = 0 To outputM.ColumnCount - 1
            Dim cellValue = outputM.At(row, col)
        Next
    Next

    Dim outputMat As Matrix(Of Double) =
countRBFKernel(Me.inputMatrix)
    For row = 0 To outputMat.RowCount - 1
        For col = 0 To outputMat.ColumnCount - 1
            Dim cellValues = outputMat.At(row, col)
        Next
    Next

    Dim outHessian As Matrix(Of Double) =
HessianMatrix(outputM, Me.expectedClasses, lambda)
    For row = 0 To outHessian.RowCount - 1
        For col = 0 To outHessian.ColumnCount - 1
            Dim cellVal = outHessian.At(row, col)
        Next
    Next

    Me.doIteration(outHessian)

    Me.countMaxAlphas()

    Me.countKernelsP(inputMatrix)
    Me.countKernelsN(inputMatrix)

    Me.countWeightsP()
    Me.countWeightsN()

    Me.countBias(Me.positiveWeights, Me.negativeWeights)
```

Gambar 5.9 *Sourcecode* Implementasi *Training* SVM  
(Sumber: Implementasi)

#### 5.2.5. Implementasi Proses *Testing* SVM

Setelah dilakukan proses *training* untuk mendapatkan hasil klasifikasi dilakukan proses terakhir yaitu proses *testing* SVM. Proses ini digunakan untuk mengklasifikasikan data testing terhadap tiga kelas penyakit yaitu busuk hitam, normal serta bercak daun. Berikut implementasi fungsi *testing* yang ditunjukkan pada Gambar 5.10.

```
Public Function test(dataTest As ArrayList)
    Me.dataTest = dataTest
    Dim kernelsValue As ArrayList =
Me.calculateTestKernel(dataTest, inputMatrix)
    Dim testWeight As ArrayList =
Me.calculateTestWeight(kernelsValue, Me.expectedClasses,
Me.latestAlpha)
    Dim totalWeight As Double = Me.sumTestWeight(testWeight)

    Dim biasTotalWeight As Double = Me.biasValue +
totalWeight

    Dim resultClassification As Double = 0

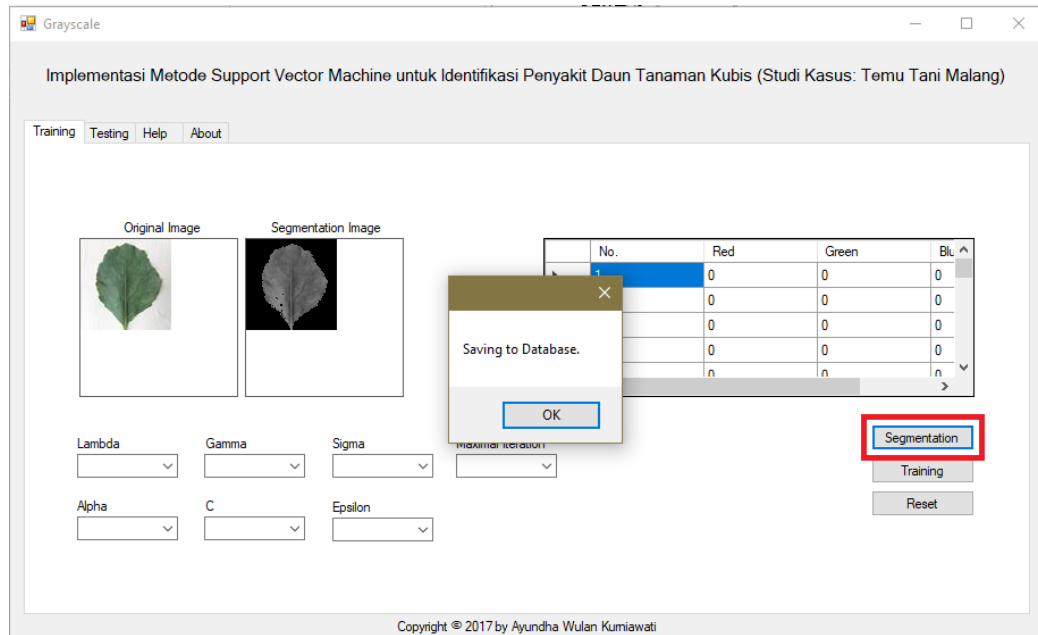
    If biasTotalWeight < 0 Then
        resultClassification = -1
    Else
        resultClassification = 1
    End If

    Return resultClassification
End Function
```

Gambar 5.10 Sourcecode Implementasi *Testing* SVM  
(Sumber: Implementasi)

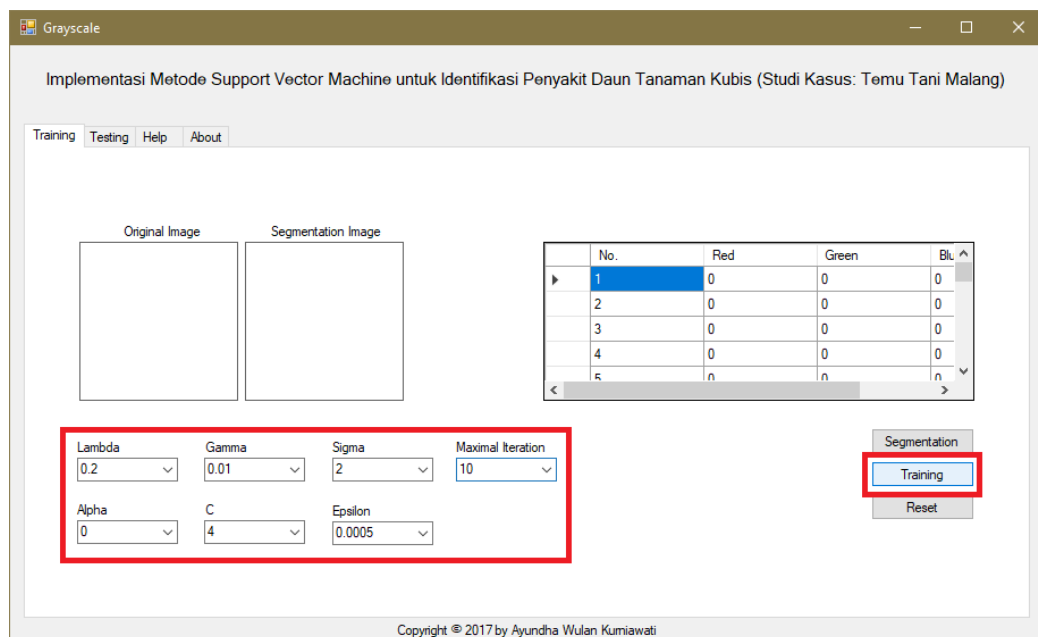
### 5.3. Klasifikasi

Klasifikasi diawali dengan mendapatkan nilai dari citra daun tanaman kubis sebagai masukan. Proses diawali *preprocessing* citra masukan yaitu dengan *Histogram Equalization*, segmentasi dengan *threshold*, dan pengubahan menjadi *grayscale* dilakukan dengan menggunakan satu button yaitu *Segmentation*, yang nilai dari *preprocessing* tersebut akan disimpan ke dalam *database*. Pada *picturebox* yang pertama menunjukkan citra masukan sebelum dilakukan *preprocessing* sedangkan pada *picturebox* yang kedua menunjukkan hasil dari *preprocessing* yang telah dilakukan, langkah tersebut diulangi hingga seluruh citra yang digunakan untuk *training* selesai dilakukan *preprocessing*. Berikut merupakan langkah *preprocessing* seperti yang ditunjukkan pada Gambar 5.11.



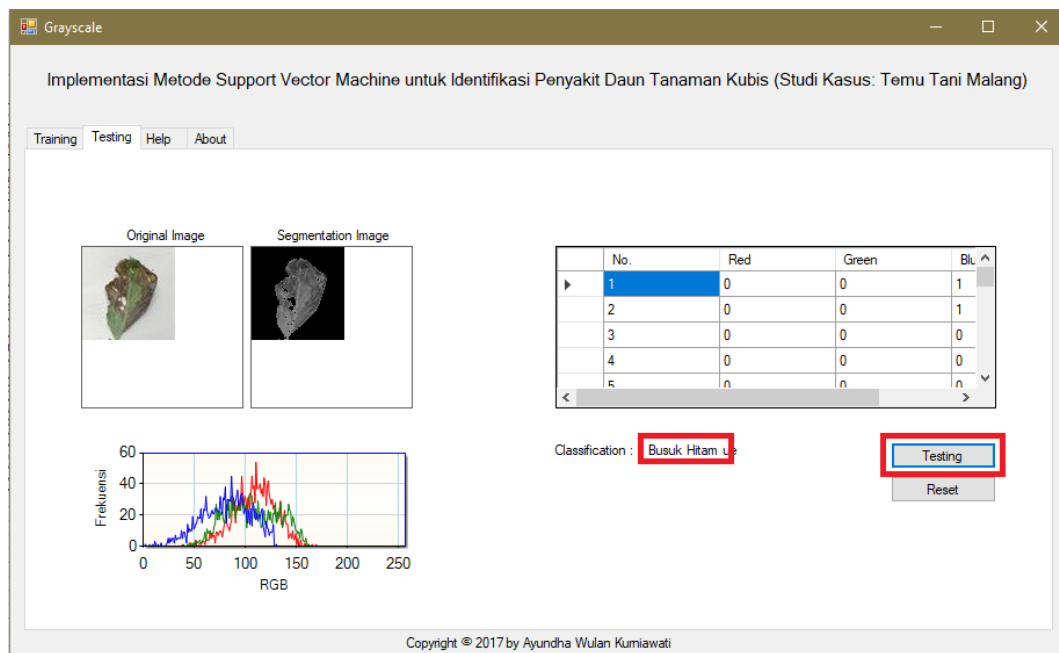
Gambar 5.11 Langkah *preprocessing* pada aplikasi  
(Sumber: Implementasi)

Langkah selanjutnya adalah memasukkan parameter yang akan digunakan untuk perhitungan *training* SVM, setelah selesai memasukkan parameter kemudian perhitungan dimulai dengan melakukan klik pada *button Training* yang akan menghasilkan nilai *alpha*, *bias* dan *sigma* yang disimpan pada *database* sehingga dapat digunakan untuk perhitungan *testing*. Berikut merupakan proses *training* seperti yang ditunjukkan pada Gambar 5.12.



Gambar 5.12 Langkah *training* pada aplikasi  
(Sumber: Implementasi)

Setelah proses *training* selesai, nilai *alpha*, *bias* dan *sigma* yang dihasilkan akan digunakan untuk perhitungan *testing* SVM. Sebelum memulai perhitungan, dilakukan *preprocessing* untuk citra masukan sebagai data *testing*. Proses yang dilakukan adalah cukup melakukan klik pada button *Testing*, karena *preprocessing* citra masukan serta proses perhitungan *testing* digabungkan pada button tersebut. Setelah proses *testing* selesai akan muncul hasil klasifikasi citra masukan tersebut, berikut merupakan hasil klasifikasi pada aplikasi yang ditunjukkan pada Gambar 5.13.



Gambar 5.13 Proses *training* pada aplikasi  
(Sumber: Implementasi)



## BAB VI. PENGUJIAN DAN PEMBAHASAN

### 6.1. Pengujian

Pengujian dari perangkat lunak ini berkaitan dengan pengujian sistem, apakah sistem sudah berjalan dengan baik. Seberapa baik kemampuan sistem dalam melakukan perhitungan sesuai dengan *source code* yang sudah tertulis dalam sistem.

Pengujian dilakukan berdasarkan spesifikasi sistem dan pengujian performansi. Pengujian spesifikasi sistem yang dilakukan meliputi pengujian kesesuaian proses, pengujian kesesuaian data dan pengujian kualitas citra. Pengujian performansi dilakukan dengan serangkaian percobaan-percobaan dalam kondisi tertentu yang dapat mempengaruhi hasil dari aplikasi ini.

#### 6.1.1. Pengujian *Blackbox*

Untuk tahap pengujian sistem menggunakan metode *blackbox*. Metode ini memungkinkan adanya pengembangan untuk melatih seluruh fungsi pada sistem. Metode ini digunakan untuk menemukan kesalahan pada saat aplikasi berjalan. Berikut *blackbox* dari pengujian sistem ditunjukkan pada Tabel 6.1.

Tabel 6.1 Pengujian Blackbox

Form	Proses	Hasil	Keterangan
Form Training	Open	Mengambil objek dan menampilkan pada ImageBox. Kemudian memproses image menjadi <i>grayscale</i> dan thresholding secara otomatis.	Berhasil
	Segmentation	Melakukan proses menghilangkan background dan mengubah citra menjadi <i>grayscale</i> .	Berhasil
	Save	Menyimpan nilai yang dihasilkan dari proses segmentation di database.	Berhasil
	Training	Melakukan proses perhitungan training dan menyimpan nilai pada database.	Berhasil

	Reset	Menghilangkan nilai parameter yang digunakan untuk proses <i>training</i> .	Berhasil
Form Testing	Open	Mengambil objek dan menampilkan pada ImageBox. Kemudian memproses image menjadi <i>grayscale</i> dan <i>thresholding</i> secara otomatis.	Berhasil
	Testing	Melakukan proses perhitungan testing dengan image yang telah di proses dan data yang telah disimpan pada database.	Berhasil
	Reset	Menghilangkan nilai parameter yang digunakan untuk proses <i>testing</i> .	Berhasil
Form Help	-	Menampilkan pengenalan menu pada sistem	Berhasil
Form About	-	Menampilkan informasi dari pembuatan sistem	Berhasil

Sumber: Pengujian







Pada Tabel 6.1 dapat dilihat hasil pengujian sistem menggunakan *blackbox*. Berdasarkan dari hasil pengujian sistem menggunakan *blackbox*, maka dapat ditarik kesimpulan bahwa aplikasi identifikasi penyakit daun tanaman kubis dengan metode *Support Vector Machine* berjalan sesuai harapan.

#### 6.1.2. Pengujian Akurasi

Pengujian ini dilakukan untuk melihat pengaruh parameter pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai parameter yang memberi nilai akurasi terbaik. Pada Tabel 6.2 menunjukkan hasil klasifikasi yang dilakukan dengan *preprocessing grayscale* 1 dan 2. Hal ini dilakukan untuk memperoleh hasil akurasi klasifikasi tertinggi. Proses segmentasi yang tidak baik mempengaruhi proses perhitungan *training* dan *testing* yang berpengaruh terhadap hasil klasifikasi.

Berikut ini merupakan hasil pengujian klasifikasi daun tanaman kubis yang ditunjukkan pada Tabel 6.2.

Tabel 6.2 Hasil Klasifikasi dengan *Preprocessing Grayscale* 1 dan 2

No.	Citra Daun	Manual	<i>Preprocessing</i> ( <i>Grayscale</i> 1)	<i>Preprocessing</i> ( <i>Grayscale</i> 2)
1.		Busuk Hitam	Bercak Daun	Busuk Hitam
2.		Bercak Daun	Bercak Daun	Bercak Daun
3.		Busuk Hitam	Busuk Hitam	Busuk Hitam
4.		Busuk Hitam	Busuk Hitam	Busuk Hitam
5.		Bercak Daun	Bercak Daun	Bercak Daun
6.		Bercak Daun	Bercak Daun	Bercak Daun

Sumber: Pengujian

Berdasarkan Tabel 6.2 yang menunjukkan hasil klasifikasi sejumlah 6 citra masukan sebagai data *testing*. Hasil klasifikasi dengan *preprocessing grayscale* 1 dan 2 untuk seluruh data *testing* dapat dilihat pada Lampiran 2.

Terdapat dua *preprocessing* yang dilakukan yaitu *Grayscale* 1 dan *Grayscale* 2. *Grayscale* 1 merupakan *preprocessing* yang dilakukan dengan segmentasi *thresholding* dan kemudian diubah menjadi *grayscale* yang menghitung *foreground* dan *background* dari citra masukan yang kemudian digunakan untuk proses perhitungan. Hasil pengujian dengan menggunakan *preprocessing Grayscale* 1 ditunjukkan pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian dengan *Preprocessing Grayscale 1*

Kategori	Jumlah	Sesuai	Tidak Sesuai	Tingkat Akurasi
Busuk Hitam	18	13	5	72.2%
Bercak Daun	18	14	4	77.7%

Sumber: Pengujian

Berdasarkan Tabel 6.3 hasil akurasi yang dihasilkan dapat dirumuskan sebagai berikut:

$$\frac{\text{Jumlah data identifikasi}}{\text{Jumlah Data Testing}} \times 100\%$$

Maka didapatkan hasil akurasi sebagai berikut:

$$\frac{27}{36} \times 100\% = 75\%$$

Tingkat akurasi yang dihasilkan dengan menggunakan *preprocessing grayscale 1* sebesar 75%.

*Grayscale 2* merupakan *preprocessing* yang dilakukan dengan segmentasi *thresholding* dan kemudian diubah menjadi *grayscale* yang menghitung nilai *foreground* dari citra masukan yang kemudian digunakan untuk proses perhitungan. Hasil pengujian dengan menggunakan *preprocessing Grayscale 2* ditunjukkan pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian dengan *Preprocessing Grayscale 2*

Kategori	Jumlah	Sesuai	Tidak Sesuai	Tingkat Akurasi
Busuk Hitam	18	12	6	66.67%
Bercak Daun	18	13	5	72.22%

Sumber: Pengujian

Berdasarkan Tabel 6.4 hasil akurasi yang dihasilkan dapat dirumuskan sebagai berikut:

$$\frac{\text{Jumlah data identifikasi}}{\text{Jumlah Data Testing}} \times 100\%$$


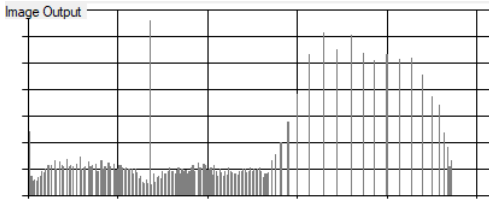

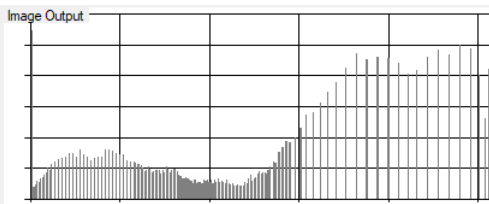

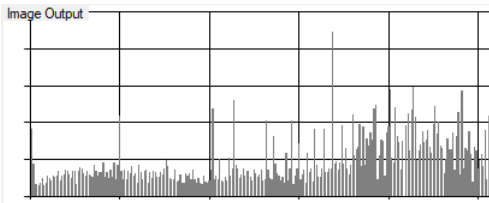

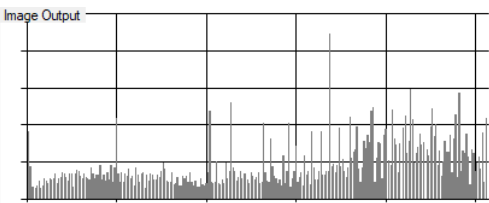
Maka didapatkan hasil akurasi sebagai berikut:


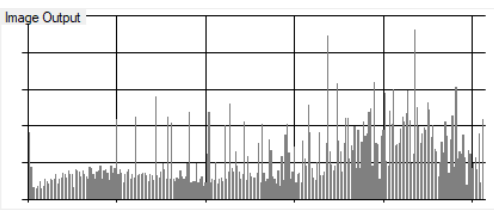

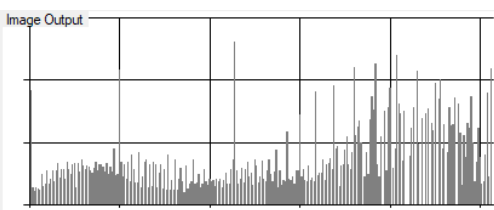
$$\frac{25}{36} \times 100\% = 69.44\%$$

Tingkat akurasi yang dihasilkan dengan menggunakan *preprocessing grayscale 2* sebesar 69.44%.

*Preprocessing* yang ketiga adalah dengan penambahan *Histogram Equalization* sebelum dilakukan segmentasi dengan *thresholding* dan diubah menjadi *grayscale* yang menghitung nilai *foreground* dari citra masukan. *Histogram Equalization* digunakan untuk memperbaiki citra masukan sehingga dapat tersegmentasi dengan baik. Berikut ini merupakan hasil klasifikasi yang dilakukan dengan melakukan pengujian dengan *Histogram Equalization* yang ditunjukkan pada Lampiran 3.

Tabel 6.5 Hasil Klasifikasi dengan *Preprocessing Histogram Equalization*

No.	Citra Daun	Manual	<i>Preprocessing</i> ( <i>Histogram Equalization</i> )	Output
1.		Busuk Hitam	0.5440625 	Busuk Hitam
2.		Bercak Daun	0.52875 	Bercak Daun
3.		Busuk Hitam	0.7025 	Busuk Hitam
4.		Busuk Hitam	0.486875 	Busuk Hitam

5.		Bercak Daun	0.611875 	Bercak Daun
6.		Bercak Daun	0.53546875 	Bercak Daun

Sumber: Pengujian

Dalam pengujian ini diperlukan sebanyak 36 sampel citra daun tanaman kubis yang terdiri dari daun yang berpenyakit busuk hitam dan bercak daun. Tingkat keakurasiannya adalah sebagai berikut seperti ditunjukkan pada Tabel 6.6.

Tabel 6.6 Hasil Pengujian dengan *Preprocessing Histogram Equalization*

Kategori	Jumlah	Sesuai	Tidak Sesuai	Tingkat Akurasi
Busuk Hitam	18	16	2	88.89%
Bercak Daun	18	15	3	83.33%

Sumber: Pengujian

Untuk citra daun tanaman kubis yang diambil untuk diidentifikasi sebanyak 18 data untuk setiap kelas, dari 18 data citra daun tanaman kubis berpenyakit busuk hitam dapat teridentifikasi dengan menggunakan metode SVM sebanyak 16 data. Kemudian untuk data citra daun tanaman kubis berpenyakit bercak daun dapat teridentifikasi sebanyak 15 data.

Berdasarkan pengujian yang telah dilakukan hasil klasifikasi dipengaruhi oleh proses segmentasi citra. Segmentasi pertama yang dilakukan adalah pengubahan citra menjadi *grayscale* dengan menghitung seluruh nilai dari *foreground* dan *background* citra yang menghasilkan akurasi klasifikasi sebesar 86.11%

#### 1. Pengujian pada nilai *Sigma*

Pengujian ini dilakukan untuk melihat pengaruh nilai *sigma* pada perhitungan Kernel RBF. Pengujian ini bertujuan untuk mendapatkan hasil terbaik pada akurasi

sistem. Variasi nilai *sigma* adalah 1, 2, 3, 4, dan 5. Skenario pengujian pada nilai *sigma* ditunjukkan pada Tabel 6.7.

Tabel 6.7 Pengujian pada nilai *Sigma*

Nilai	Parameter
	Sigma
1	60.963%
2	68.519%
3	68.519%
4	72.222%
5	72.222%

Sumber: Pengujian

Berdasarkan tabel 6.7 semakin besar nilai sigma yang diberikan semakin tinggi hasil akurasi yang diperoleh, karena semakin besar nilai sigma yang diberikan maka semakin besar nilai kernel RBF. Hasil perhitungan kernel mendapatkan hasil yang kecil sehingga mempengaruhi akurasi dari sistem. Sehingga dapat disimpulkan bahwa nilai sigma mempengaruhi hasil akurasi dari sistem dan didapatkan nilai akurasi sebesar 72.222% dengan nilai 4 dan 5.

## 2. Pengujian pada parameter *Lambda*

Pengujian ini dilakukan untuk melihat pengaruh nilai *lambda* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Lamda* yang memberi nilai akurasi terbaik. Variasi nilai *lambda* adalah 0.1, 0.2, 0.3, 0.4, dan 0.5. Skenario pengujian pada nilai *lambda* ditunjukkan pada Tabel 6.8.

Tabel 6.8 Pengujian pada nilai *Lambda*

Nilai	Parameter
	Lambda
0.1	71.603%
0.2	71. 603%
0.3	71. 603%
0.4	73.333%
0.5	73.333%

Sumber: Pengujian

Berdasarkan tabel 6.8 semakin besar nilai lambda yang diberikan semakin kecil hasil akurasi yang diperoleh, karena besar kecilnya nilai lambda mempengaruhi hasil perhitungan matriks hessian. Sehingga dapat disimpulkan bahwa nilai lambda mempengaruhi hasil akurasi dari sistem dan didapatkan nilai akurasi sebesar 73.333% dengan nilai 0.4, dan 0.5.

### 3. Pengujian pada nilai *Gamma*

Pengujian ini dilakukan untuk melihat pengaruh nilai *gamma* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Gamma* yang memberi nilai akurasi terbaik. Variasi nilai *gamma* adalah 0.01, 0.1, 0.2, 0.3, 0.4, dan 0.5. Skenario pengujian pada nilai *gamma* ditunjukkan pada Tabel 6.9.

Tabel 6.9 Pengujian pada nilai *Gamma*

Nilai	Parameter
	Gamma
0.1	70.074%
0.2	72.926%
0.3	72.926%
0.4	75.185%
0.5	75.185%

Sumber: Pengujian

Berdasarkan tabel 6.9 semakin besar nilai gamma yang diberikan semakin kecil hasil akurasi yang diperoleh, karena nilai gamma mempengaruhi perhitungan nilai alpha. Sehingga dapat disimpulkan bahwa nilai gamma mempengaruhi hasil akurasi dari sistem dan didapatkan nilai akurasi sebesar 75.185% dengan nilai 0.4 dan 0.5.

### 4. Pengujian pada nilai *Complexity*

Pengujian ini dilakukan untuk melihat pengaruh nilai *complexity* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Complexity* yang memberi nilai akurasi terbaik. Variasi nilai *complexity* adalah 1, 2, 3, 4, dan 5. Skenario pengujian pada nilai *complexity* ditunjukkan pada Tabel 6.10.



Tabel 6.10 Pengujian pada nilai *Complexity*

Nilai	Parameter
	Complexity
1	76.2%
2	73.3%
3	72.8%
4	70.741%
5	78.889%

Sumber: Pengujian

Berdasarkan tabel 6.10 dapat dilihat bahwa semakin besar nilai *Complexity* (C) semakin besar pula hasil akurasi. Hal ini dikarenakan nilai *C* mempengaruhi hasil perhitungan alpha. Nilai *C* yang memberikan hasil akurasi terbesar adalah 5. Dengan hasil tersebut dapat disimpulkan bahwa nilai *C* mempengaruhi hasil akurasi dari sistem dan didapatkan nilai akurasi 78.889% dengan nilai *Complexity* 5.

#### 5. Pengujian pada nilai *Epsilon*

Pengujian ini dilakukan untuk melihat pengaruh nilai *epsilon* pada *Sequential Training* terhadap nilai akurasi sistem. Pengujian ini bertujuan untuk mendapatkan nilai *Epsilon* yang memberi nilai akurasi terbaik. Variasi nilai *epsilon* adalah 0.0001, 0.0002, 0.0005, 0.001, dan 0.01. Skenario pengujian pada nilai *epsilon* ditunjukkan pada Tabel 6.11.

Tabel 6.11 Pengujian pada nilai *Epsilon*

Nilai	Parameter
	Epsilon
0.0001	75.185%
0.0002	75.185%
0.0005	75.185%
0.001	75.185%
0.01	73.333%

Sumber: Pengujian

Berdasarkan tabel 6.11 dapat dilihat bahwa rata-rata akurasi stabil. Nilai epsilon dengan akurasi yang tertinggi adalah 0.0001, 0.0002, 0.0005 dan 0.001 dengan nilai akurasi sebesar 75.185%. Nilai epsilon mempengaruhi perhitungan

training karena digunakan untuk menghentikan jalannya iterasi. Iterasi dihentikan jika nilai maksimal dari delta alpha I sudah konvergen atau delta alpha < epsilon.

#### 6. Pengujian pada nilai Iterasi Maksimal

Pengujian ini dilakukan untuk melihat pengaruh nilai iterasi maksimal pada *Sequential Training*. Iterasi maksimal ditentukan untuk batas nilai iterasi maksimal pada proses training dengan variasi nilai yang digunakan adalah 2, 5, 10, 25, dan 100. Parameter-parameter SVM yang digunakan adalah nilai  $\alpha_i = 0$ ,  $\varepsilon = 0.001$ ,  $\gamma = 0.01$ ,  $\lambda = 1$ ,  $C = 1$ . Tujuan dari pengujian ini adalah melihat pengaruh Iterasi Maksimal terhadap hasil akurasi dari sistem, berikut hasil pengujian pada Iterasi Maksimal ditunjukkan pada Tabel 6.12.

Tabel 6.12 Pengujian pada nilai Iterasi Maksimal

Nilai	Parameter
	Iterasi Maksimal
2	75.185%
5	75.185%
10	75.185%
25	77.037%
50	75.037%

Sumber: Pengujian

Berdasarkan tabel 6.12 dapat dilihat bahwa rata-rata akurasi stabil. Nilai akurasi yang rendah dikarenakan hasil perhitungan nilai bias tidak didapatkan nilai yang baik. Nilai bias yang baik adalah nilai bias yang didapatkan saat nilai delta alpha < epsilon atau saat konvergen.

## 6.2. Analisa

Berdasarkan pengujian nilai-nilai parameter didapatkan nilai parameter yang menghasilkan nilai akurasi terbesar dari tiap-tiap pengujian yang ditunjukkan pada Tabel 6.13.

Tabel 6.13 Parameter yang digunakan

Parameter	Nilai
Lambda	0.4, 0.5
Gamma	0.4, 0.5
C	5

<b>Sigma</b>	4, 5
<b>Epsilon</b>	0.01
<b>Maximal Iteration</b>	25

Sumber: Pengujian

Tabel 6.14 merupakan *confusion matrix* yang menunjukkan perhitungan nilai *precision* dan *recall*.

Tabel 6.14 *Confusion Matrix* hasil pengujian

<b>n = 36</b>	<b>Predicted Class</b>	
<b>Actual Class</b>	<b>Class 1 (Bercak Daun)</b>	<b>Class 2 (Busuk Hitam)</b>
<b>Class 1 (Bercak Daun)</b>	TP = 14	FN = 4
<b>Class 2 (Busuk Hitam)</b>	FP = 3	TN = 15

Sumber: Pengujian

$$\begin{aligned}
 \text{Precision} &= \frac{\text{TP}}{(\text{FP} + \text{TP})} \times 100\% \\
 &= \frac{14}{(3 + 14)} \times 100\% = 82.35\%
 \end{aligned}$$

$$\begin{aligned}
 \text{Recall} &= \frac{\text{TP}}{(\text{TP} + \text{FN})} \times 100\% \\
 &= \frac{14}{(14 + 4)} \times 100\% = 77.78\%
 \end{aligned}$$

$$\begin{aligned}
 \text{Akurasi} &= \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \times 100\% \\
 &= \frac{(14 + 15)}{(14 + 15 + 3 + 4)} \times 100\% = 80.55\%
 \end{aligned}$$

Berdasarkan perhitungan *confusion matrix* diatas maka tingkat keberhasilan identifikasi penyakit daun tanaman kubis secara keseluruhan dengan menggunakan *Histogram Equalization* untuk memperbaiki citra masukan sehingga dapat tersegmentasi dengan baik adalah sebesar 80.55%.

Pada proses perhitungan *training* terdapat inputan parameter yang digunakan. Nilai-nilai parameter yang digunakan tersebut mempengaruhi proses *Sequential Training* secara keseluruhan mulai dari proses perhitungan *Matriks Hessian*, nilai *Error*, *Delta Alpha*, *Alpha*, *Gamma*, nilai  $(x, xi)$ , nilai  $w$  dan nilai *bias* yang mempengaruhi proses klasifikasi. Dengan menggunakan nilai parameter terbaik

maka sistem diharapkan mampu menghasilkan klasifikasi dengan akurasi yang lebih baik.

Berdasarkan pengujian yang telah dilakukan yaitu dengan pengujian *preprocessing* yang berbeda dan pengujian nilai-nilai parameter yang digunakan untuk perhitungan *training* menunjukkan bahwa hasil klasifikasi dipengaruhi oleh beberapa faktor seperti posisi daun, cahaya serta *background* saat pengambilan citra, *preprocessing* yang digunakan, serta parameter masukan untuk proses perhitungan SVM. Berikut ini merupakan penjelasan mengenai faktor yang berpengaruh pada klasifikasi:

1. Pengambilan Citra

- a. Posisi Daun

Letak dari daun tanaman kubis yang akan diakuisisi citra nya tidak terlalu berpengaruh dalam proses klasifikasi karena *background* dari citra tersebut akan dihilangkan sehingga yang akan dihitung dari nilai *foreground* saja.

- b. Cahaya

Pengambilan citra dilakukan di siang hari karena membutuhkan pencahayaan maksimal (tidak berada dalam kondisi yang minim cahaya). Apabila pada citra yang diambil terdapat penurunan intensitas cahaya dapat menyebabkan hasil segmentasi tidak masimal sehingga informasi citra masukan tidak dapat terkomputasi dengan baik.

- c. *Background*

Daun tanaman kubis yang akan diakuisisi diletakkan diatas *background* warna putih, pada penelitian ini yang digunakan sebagai *background* adalah kertas hvs berwarna putih. Penggunaan *background* akan mempengaruhi hasil klasifikasi karena apabila dalam proses segmentasi tidak sempurna maka *background* akan ikut terhitung menjadi *foreground*.

2. *Preprocessing*

- a. *Grayscale 1*

*Grayscale 1* pada penelitian ini adalah pengubahan citra masukan menjadi citra *grayscale* yang menghitung seluruh *background* dan *foreground* citra masukan. Perhitungan tersebut akan mempengaruhi klasifikasi karena

*background* termasuk nilai yang dihitung sehingga hasil klasifikasi menjadi kurang akurat.

b. *Grayscale 2*

*Grayscale 2* pada penelitian ini adalah pengubahan citra masukan menjadi citra *grayscale* dengan hanya menghitung nilai *foreground* dari citra masukan. Hal ini dilakukan karena yang akan diolah adalah objek daun sehingga hanya perlu mengambil nilai dari *foreground* citra masukan.

c. *Histogram Equalization*

Penggunaan *Histogram Equalization* pada penelitian ini adalah untuk memperbaiki citra masukan yang akan digunakan untuk proses perhitungan. Hal ini dikarenakan terdapat beberapa citra yang saat proses akuisisi nya tidak baik sehingga masih terdapat bayangan serta pencahayaan yang kurang maksimal. Setelah perbaikan yang dilakukan dengan *Histogram Equalization* selanjutnya citra akan dilakukan segmentasi untuk menghilangkan *background*, kemudian akan diubah menjadi citra *grayscale* yang selanjutnya nilai yang didapatkan akan digunakan sebagai *input* untuk proses perhitungan *training* dan *testing*.

3. Metode SVM

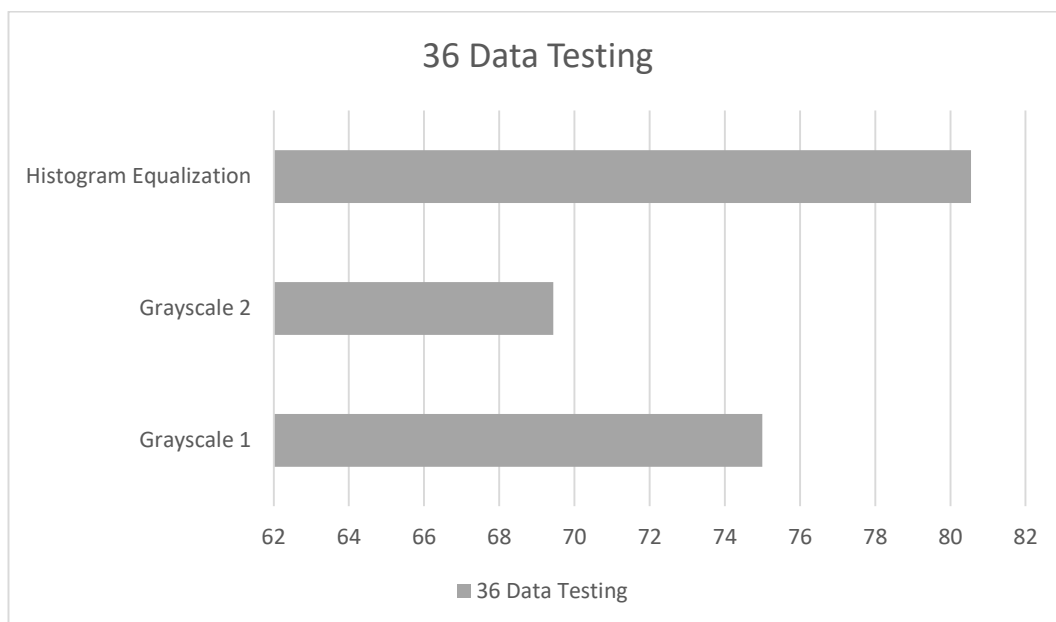
a. Parameter

Terdapat 7 parameter masukan yang digunakan untuk proses perhitungan *training* yaitu *Sigma*, *Lambda*, *Alpha*, *Complexity*, *Epsilon*, *Gamma*, dan Iterasi Maksimal.

Penggunaan nilai parameter yang terbaik menghasilkan hasil klasifikasi dengan akurasi yang lebih baik. Penentuan parameter tersebut dilakukan dengan melakukan pengujian pada nilai parameter yang berbeda, misalkan untuk parameter *Sigma*, *range* nilai yang diujikan adalah 1, 2, 3, 4, 5. Kemudian dari nilai tersebut akan diketahui mana yang menghasilkan akurasi paling tinggi, maka nilai tersebut merupakan parameter terbaik. Hal tersebut berlaku juga untuk parameter lain seperti *Lambda* dengan *range* nilai yang diujikan adalah 0.1, 0.2, 0.3, 0.4, dan 0.5. *Complexity* dengan *range* nilai yang diujikan adalah 1, 2, 3, 4, dan 5. *Epsilon* dengan *range* nilai yang diujikan adalah 0.0001, 0.0002, 0.0005, 0.001, dan 0.01. *Gamma* dengan *range* nilai

yang diujikan adalah 0.1, 0.2, 0.3, 0.4, dan 0.5. Iterasi Maksimal dengan range nilai yang diujikan adalah 2, 5, 10, 25, dan 50. Tetapi hal tersebut tidak berlaku untuk nilai *Alpha* karena nilai *Alpha* yang diinputkan diawal adalah 0 yang kemudian akan dilakukan perbaikan nilai *Alpha* pada setiap iterasi, karena selanjutnya nilai *Alpha* pada iterasi yang terakhir akan digunakan untuk proses perhitungan *testing*.

Proses pengujian dengan citra masukan sebagai data *testing* sebanyak 36 citra yang dilakukan pengujian dengan 3 proses segmentasi yang berbeda. Pada penggunaan proses segmentasi pertama yaitu *grayscale 1* yang mengubah nilai *foreground* dan *background* dari citra masukan mampu menghasilkan akurasi sebesar 75%. Penggunaan *grayscale 2* yang mengubah nilai *foreground* saja dari citra masukan mendapatkan akurasi sebesar 69.44%. Proses yang terakhir dengan menggunakan *Histogram Equalization* yang digunakan untuk memperbaiki citra masukan kemudian selanjutnya dapat dilakukan proses segmentasi untuk mendapatkan citra masukan yang akan diolah untuk perhitungan *training* dan *testing*, menghasilkan akurasi sebesar 80.55%. Berikut ini merupakan hasil yang telah dilakukan dalam penelitian ini yang ditunjukkan pada Gambar 6.1.



Gambar 6.1 Hasil pengujian dengan proses segmentasi yang berbeda

(Sumber: Pengujian)

Tingkat keberhasilan identifikasi penyakit daun tanaman kubis dengan menggunakan metode SVM dipengaruhi oleh *preprocessing* yang dilakukan dan

parameter masukan yang digunakan saat *training*. Dengan proses segmentasi yang baik akan menghasilkan klasifikasi yang lebih bagus, serta penggunaan nilai parameter terbaik juga mampu meningkatkan hasil klasifikasi.

## BAB VII. KESIMPULAN

### 7.1. Kesimpulan

Berdasarkan hasil yang didapatkan dari perancangan, implementasi dan pengujian dapat diambil beberapa kesimpulan sebagai berikut:

1. *Preprocessing* pertama yang digunakan adalah segmentasi dengan *thresholding* dan diubah menjadi *grayscale* tanpa menghilangkan *background* sehingga nilai yang digunakan adalah nilai keseluruhan dari *background* dan *foreground* citra masukan mampu menghasilkan akurasi sebesar 75%.
2. *Preprocessing* kedua yang digunakan adalah segmentasi dengan *thresholding* dan diubah menjadi *grayscale*, nilai *background* telah dihilangkan sehingga nilai yang digunakan hanya nilai *foreground* dari citra masukan mampu menghasilkan akurasi sebesar 69.44%.
3. *Preprocessing* ketiga yang digunakan adalah *Histogram Equalization* untuk memperbaiki citra sehingga citra masukan dapat tersegmentasi lebih baik, selanjutnya citra disegmentasi dengan *thresholding* dan diubah menjadi *grayscale* yang hanya menghitung nilai *foreground* citra masukan mampu menghasilkan akurasi sebesar 80.55%.
4. Identifikasi penyakit daun tanaman kubis dengan mengimplementasikan metode *Support Vector Machine* memperoleh tingkat akurasi terbesar dengan nilai parameter terbaik yaitu  $\varepsilon = 0.01$ ,  $\gamma = 0.4$  dan  $0.5$ ,  $\lambda = 0.4$  dan  $0.5$ ,  $C = 5$ , dan iterasi maksimal = 25.
5. Tingkat keberhasilan identifikasi penyakit daun tanaman kubis dengan menggunakan metode SVM dipengaruhi oleh beberapa faktor yaitu posisi daun, cahaya serta *background* saat pengambilan citra, *preprocessing* yang digunakan dan parameter masukan yang digunakan untuk proses perhitungan SVM. Dengan proses segmentasi yang baik akan menghasilkan klasifikasi yang lebih bagus, penggunaan parameter terbaik mampu meningkatkan hasil klasifikasi.



## **7.2. Saran**

Penelitian ini masih jauh dari kata sempurna, oleh karena itu berikut ini beberapa saran untuk pengembangan penelitian selanjutnya:

1. Perlu dilakukan ekstraksi fitur yang berbeda sehingga dapat membedakan bentuk dari bercak daun atau busuk hitam selain dari dari tekstur warna.
2. Perlu dilakukan penggunaan ruang warna sehingga dapat menghasilkan yang berbeda.
3. Perlu dilakukan penggunaan segmentasi yang berbeda sehingga dapat menghasilkan klasifikasi yang lebih akurat.

## DAFTAR PUSTAKA

- [1] Prasetyo, Eko. 2012. “*DATA MINING – Konsep dan Aplikasi Menggunakan MATLAB*”. Yogyakarta: Andi.
- [2] Vidyani, Faradina. 2013. “*Identifikasi Penyakit Tanaman Kubis menggunakan Gaussian Filter dan Wavelet*”. Diakses dari <http://repository.ipb.ac.id/jspui/handle/123456789/66550> pada tanggal 14 Desember 2016 pukul 15.03.
- [3] Dewi, Ratih Kartika dan Ginardi, R.V. 2014. “*Identifikasi Penyakit pada Daun Tebu dengan Gray Level Co-occurrence Matrix dan Color Moments*”. Diakses dari <http://jtiik.ub.ac.id/index.php/jtiik/article/view/114> pada tanggal 14 Desember 2016 pukul 14:30.
- [4] Sutoyo, T., dkk. 2009. “*Teori Pengolahan Citra Digital*”. Yogyakarta: Penerbit Andi.
- [5] Putra, Darma. 2010. “*Pengolahan Citra Digital*”. Yogyakarta: Andi.
- [6] Zulkarnaen, Haji. “*Budidaya Sayuran Tropis*”. 2013. Jakarta: Bumi Aksara. 62-96.
- [7] Semangun, Haryono. 2007. “*Penyakit-penyakit Tanaman Holtikultura di Indonesia*”. Yogyakarta: Gadjah Mada University Press. 171-194.
- [8] Arbawa, Yoke Kusuma. 2016. “*Implementasi Metode Support Vector Machine (SVM) untuk Identifikasi Penyakit pada Citra Daun Tanaman Kacang Tanah Menggunakan Gray Level Cooccurrence Matrix (GLCM)*”. Malang. Universitas Brawijaya
- [9] Nugroho, Anto S., *et al.* 2003. “*Support Vector Machine : Teori dan Aplikasinya dalam Bioinformatika*”. Ilmukomputer.
- [10] Musicant, D.R., Kumar, V., dan Ozgur, A. 2003. “*Optimizing F-Measure with Support Vector Machines*”. Florida: FLAIRS (The Florida Artificial Intelligence Research Society. 356-360.









## LAMPIRAN










### Lampiran 1. Dataset *Testing*










	<i>Grayscale</i>	Normalisasi
1	244	0.9000
2	227	0.787603
3	229	0.800826
4	235	0.840496
5	215	0.7083
6	203	0.628926
7	157	0.3248
8	186	0.5165
9	170	0.410744
10	181	0.483471
11	199	0.602479
12	219	0.734711
13	225	0.77438
14	233	0.827273
15	238	0.860331
16	190	0.542975
17	195	0.576033
18	175	0.443802
19	168	0.397521
20	128	0.1331
21	206	0.64876
22	160	0.344628
23	240	0.873554
24	123	0.1
25	211	0.681818
26	208	0.661983
27	154	0.304959










28	139	0.205785
29	217	0.721488
30	125	0.113223
31	132	0.159504
32	136	0.18595
33	166	0.384298
34	145	0.245455
35	149	0.271901
36	151	0.285124

**Lampiran 2. Hasil Pengujian *Preprocessing* Grayscale 1 dan 2**


No.	Citra Daun	Manual	<i>Preprocessing</i> (Grayscale 1)	<i>Preprocessing</i> (Grayscale 2)
1.		Bercak Daun	Bercak Daun	Bercak Daun
2.		Bercak Daun	Bercak Daun	Bercak Daun
3.		Bercak Daun	Bercak Daun	Bercak Daun
4.		Bercak Daun	Bercak Daun	Bercak Daun
5.		Busuk Hitam	Bercak Daun	Busuk Hitam
6.		Bercak Daun	Bercak Daun	Bercak Daun
7.		Busuk Hitam	Busuk Hitam	Busuk Hitam
8.		Busuk Hitam	Busuk Hitam	Busuk Hitam

9.		Busuk Hitam	Busuk Hitam	Busuk Hitam
10		Busuk Hitam	Busuk Hitam	Busuk Hitam
11		Busuk Hitam	Busuk Hitam	Busuk Hitam
12		Bercak Daun	Busuk Hitam	Bercak Daun
13.		Bercak Daun	Bercak Daun	Bercak Daun
14.		Bercak Daun	Bercak Daun	Bercak Daun
15.		Bercak Daun	Bercak Daun	Bercak Daun
16.		Bercak Daun	Bercak Daun	Bercak Daun
17.		Bercak Daun	Bercak Daun	Bercak Daun


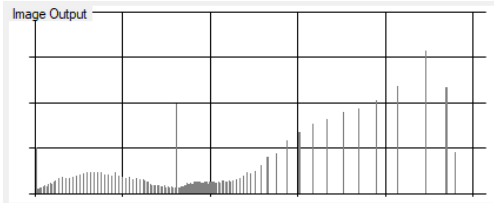

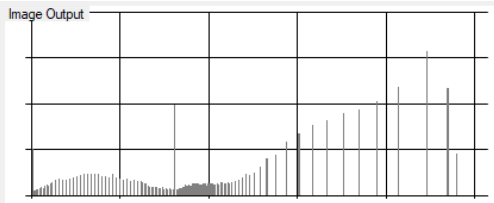

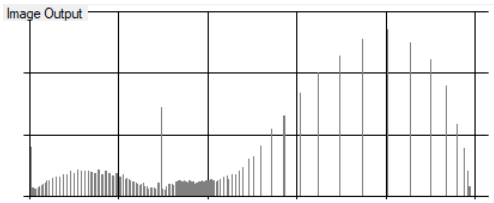

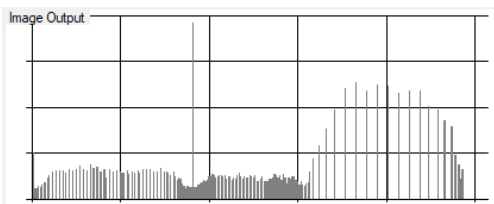

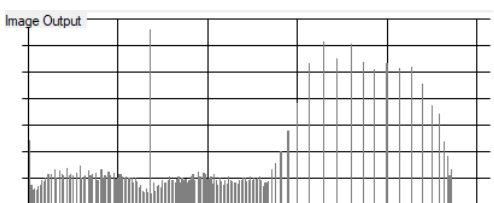

18.		Bercak Daun	Bercak Daun	Bercak Daun
19.		Busuk Hitam	Busuk Hitam	Busuk Hitam
20.		Busuk Hitam	Busuk Hitam	Busuk Hitam
21.		Busuk Hitam	Busuk Hitam	Busuk Hitam
22.		Busuk Hitam	Busuk Hitam	Busuk Hitam
23.		Busuk Hitam	Busuk Hitam	Busuk Hitam
24.		Busuk Hitam	Busuk Hitam	Busuk Hitam
25		Bercak Daun	Bercak Daun	Bercak Daun
26		Bercak Daun	Bercak Daun	Bercak Daun

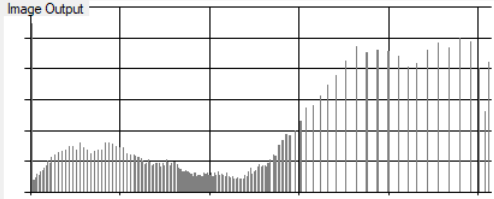

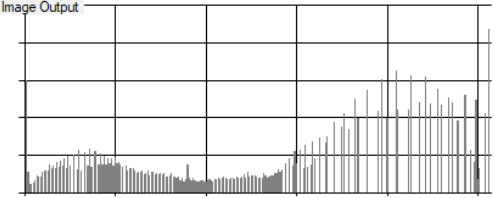

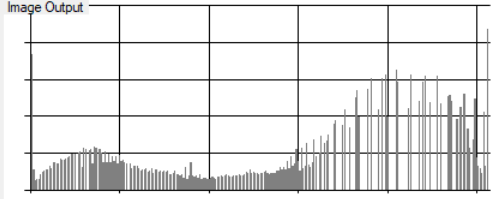

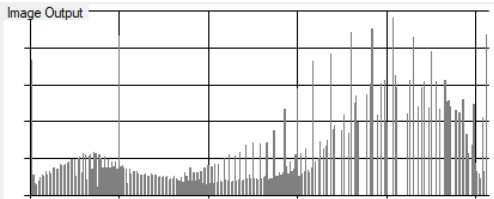

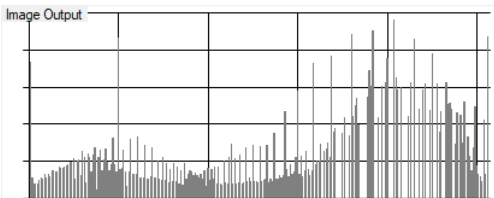

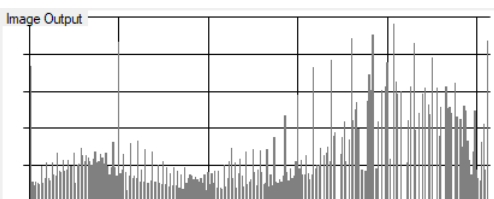
27.		Busuk Hitam	Bercak Daun	Bercak Daun
28.		Busuk Hitam	Bercak Daun	Bercak Daun
29.		Bercak Daun	Bercak Daun	Bercak Daun
30.		Busuk Hitam	Bercak Daun	Bercak Daun
31.		Busuk Hitam	Busuk Hitam	Busuk Hitam
32.		Busuk Hitam	Busuk Hitam	Busuk Hitam
33.		Bercak Daun	Busuk Hitam	Busuk Hitam
34.		Bercak Daun	Busuk Hitam	Busuk Hitam
35.		Busuk Hitam	Busuk Hitam	Busuk Hitam


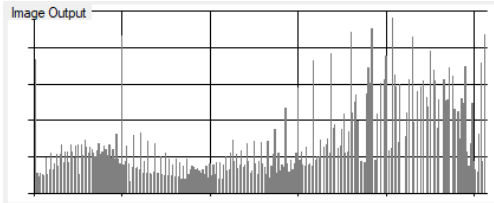

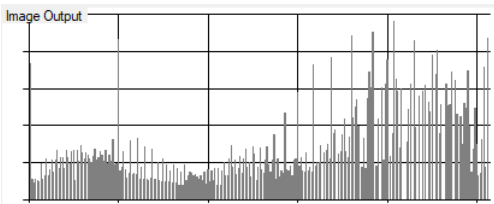

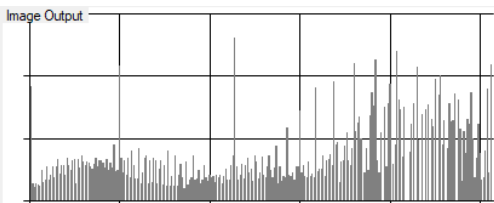

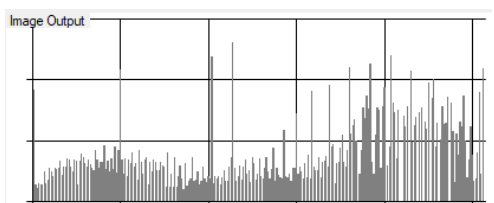

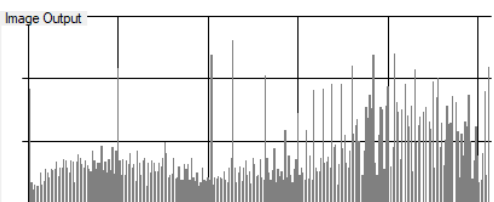

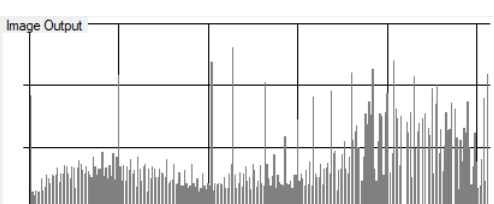



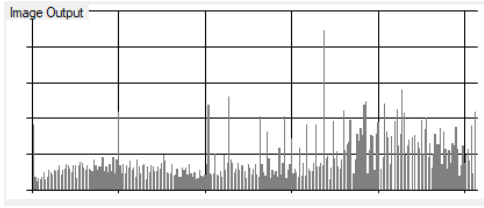

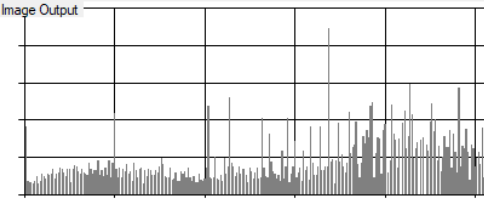

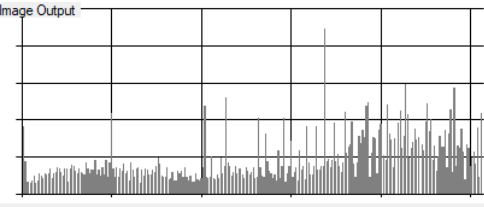

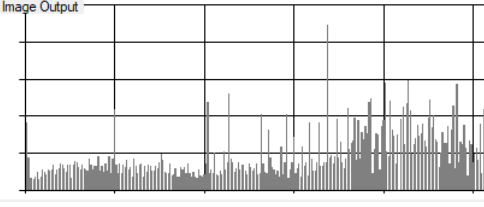

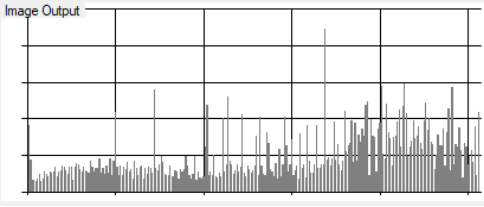

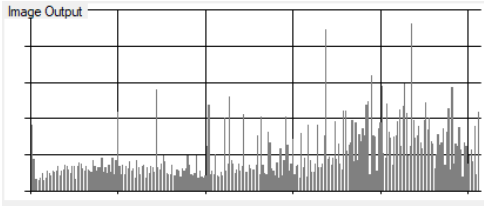
36.		Busuk Hitam	Busuk Hitam	Busuk Hitam
-----	---	-------------	-------------	-------------


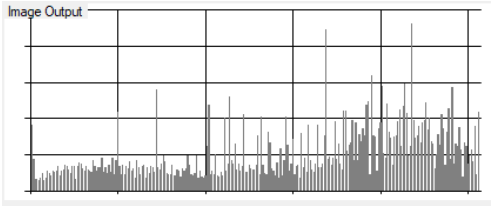

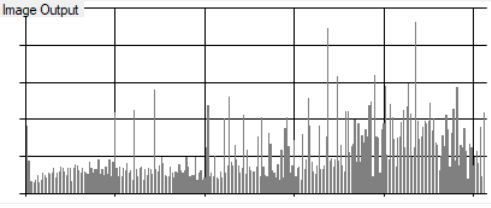

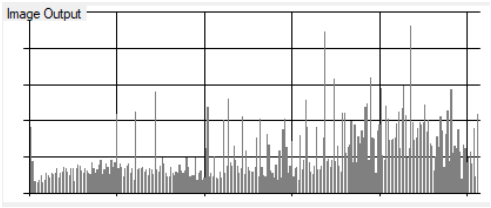

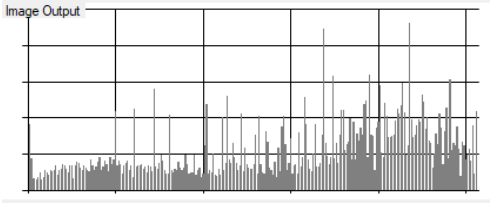

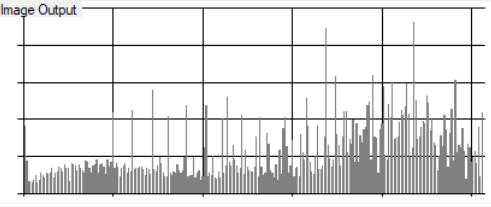

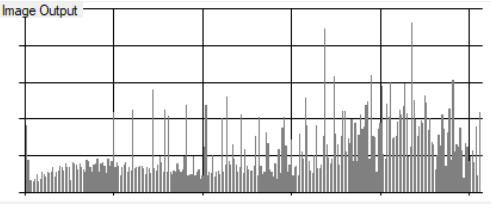
**Lampiran 3. Hasil Pengujian *Preprocessing Histogram Equalization***


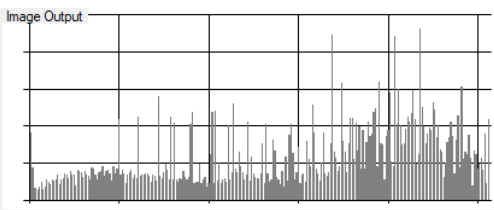

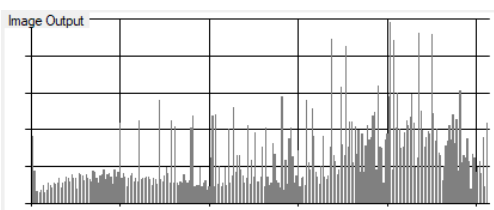

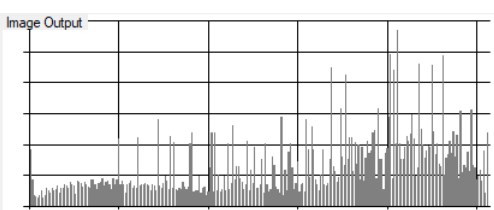

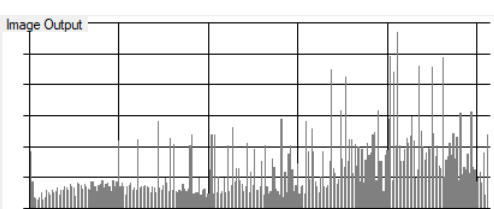

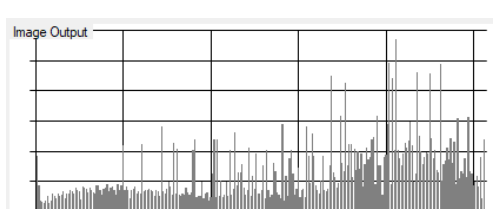

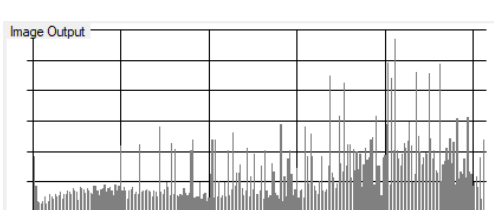
No.	Citra Daun	Manual	<i>Preprocessing (Histogram Equalization)</i>	<i>Output</i>
1.		Bercak Daun	0.538125 	Bercak Daun
2.		Bercak Daun	0.566875 	Busuk Hitam
3.		Bercak Daun	0.3978125 	Bercak Daun
4.		Bercak Daun	0.4521875 	Bercak Daun
5.		Busuk Hitam	0.5440625 	Busuk Hitam
6.		Bercak Daun	0.52875	Bercak Daun


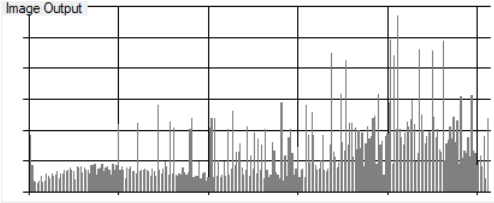
				
7.		Busuk Hitam	0.59171875 	Busuk Hitam
8.		Busuk Hitam	0.54109375 	Busuk Hitam
9.		Busuk Hitam	0.5421875 	Busuk Hitam
10		Busuk Hitam	0.62 	Busuk Hitam
11		Busuk Hitam	0.7009375 	Busuk Hitam

12		Bercak Daun	0.55546875 	Bercak Daun
13.		Bercak Daun	0.60265625 	Bercak Daun
14.		Bercak Daun	0.53546875 	Bercak Daun
15.		Bercak Daun	0.710625 	Busuk Hitam
16.		Bercak Daun	0.73203125 	Bercak Daun
17.		Bercak Daun	0.75265625 	Busuk Hitam

18.		Bercak Daun	0.67046875 	Busuk Hitam
19.		Busuk Hitam	0.66875 	Busuk Hitam
20.		Busuk Hitam	0.486875 	Busuk Hitam
21.		Busuk Hitam	0.7025 	Busuk Hitam
22.		Busuk Hitam	0.71921875 	Busuk Hitam
23.		Busuk Hitam	0.61875 	Bercak Daun

24.		Busuk Hitam	0.73515625 	Busuk Hitam
25		Bercak Daun	0.5678125 	Bercak Daun
26		Bercak Daun	0.67921875 	Bercak Daun
27.		Busuk Hitam	0.5625 	Bercak Daun
28.		Busuk Hitam	0.6875 	Bercak Daun
29.		Bercak Daun	0.611875 	Bercak Daun

30.		Busuk Hitam	0.620625 	Bercak Daun
31.		Busuk Hitam	0.7971875 	Busuk Hitam
32.		Busuk Hitam	0.68171875 	Busuk Hitam
33.		Bercak Daun	0.61984375 	Bercak Daun
34.		Bercak Daun	0.61875 	Bercak Daun
35.		Busuk Hitam	0.53921875 	Busuk Hitam

36.		Busuk Hitam	<div>0.60421875</div> <div>Image Output</div> 	Busuk Hitam
-----	---	----------------	--	----------------



## Lampiran 4. Kuisisioner

KUISISIONER IMPLEMENTASI METODE SUPPORT VECTOR MACHINE  
UNTUK IDENTIFIKASI PENYAKIT DAUN TANAMAN KUBIS  
(STUDI KASUS : TEMU TANI MALANG)

Nama : .....

Lingkari pada penilaian yang sesuai

No.	Pengujian	K	C	B
1.	Aplikasi mudah digunakan	1	②	3
2.	Aplikasi memiliki tampilan yang menarik	1	②	3
3.	Aplikasi menyediakan petunjuk dan instruksi yang mudah dimengerti	1	②	3
4.	Aplikasi mampu menampilkan hasil identifikasi yang sesuai	1	②	3
5.	Aplikasi menyediakan informasi yang dapat mempermudah identifikasi penyakit daun tanaman kubis	1	②	3

Keterangan :

B : Baik  
C : Cukup  
K : Kurang

Malang, 10 Agustus 2017  
Ina Setiyani  
0341 7777 164  
081 2334 57555  
BELUNG - TUMPANG - MALANG

## Lampiran 5. Lembar Bimbingan Skripsi Pembimbing I



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



NO SKRIPSI: 101

### LEMBAR BIMBINGAN SKRIPSI 2016/2017

JUDUL : Implementasi Metode *Support Vector Machine* untuk Identifikasi Penyakit  
Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)

Nama : Ayundha Wulan Kurniawati

NIM : 1341180153

No.	Tanggal	Materi Bimbingan	Tanda Tangan	
			Mahasiswa	Dosen
1.	3/03/17	Konsultasi Metode + jadwal	Anu.	
2.	24/03/17	Metode <del>svm</del> SVM	Anu.	
3.	30/03/17	Konsultasi kernel	Anu.	
4.	07/04/17	Perhitungan excel SVM + nilai error	Anu.	
5.	20/04/17	Pembahasan RBF	Anu.	
6.	28/04/17	Iterasi SVM, Nilai error	Anu.	
7.	12/05/17	Perbaikan iterasi SVM, nilai error	Anu.	
8.	19/05/17	Proses testing	Anu.	
9.	26/05/17	Proses testing + array image	Anu.	
10.	16/06/17	Proses ekstraksi image	Anu.	
11.	19/06/17	Proses ekstraksi image	Anu.	
12.	21/06/17	Konsultasi ekstraksi + multiclass	Anu.	
13.	07/07/17	vector image	Anu.	
14.	20/07/17	serialize image	Anu.	
15.	21/07/17	Finalisasi metode SVM	Anu.	
16.	24/07/17	Perbaikan input vector	Anu.	
17.	28/07/17	Konsultasi laporan	Anu.	
18.	31/08/17	ACC	Anu.	
19.				

Malang, 03 Agustus 2017  
Dosen Pembimbing Skripsi,

Ariadi Retno Tri Hayati Ririd, S.Kom.,  
M.Kom  
NIP. 19810810 200501 2 002

## Lampiran 6. Lembar Bimbingan Skripsi Pembimbing II



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



NO SKRIPSI: 101

### LEMBAR BIMBINGAN SKRIPSI 2016/2017

**JUDUL :** Implementasi Metode *Support Vector Machine* untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)

**Nama :** Ayundha Wulan Kurniawati

**NIM :** 1341180153

No.	Tanggal	Materi Bimbingan	Tanda Tangan	
			Mahasiswa	Dosen
1.	10/03/17	Pemahaman SVM + Core Program	Ank.	Ank.
2.	22/03/17	Dasar ubinet - funct, perhitungan SVM	Ank.	Ank.
3.	31/03/17	Kernel + Hessian Matrix	Ank.	Ank.
4.	05/04/17	Nilai error, gamma, delta alpha, newalpha	Ank.	Ank.
5.	26/04/17	Iterasi SVM	Ank.	Ank.
6.	03/05/17	Implementasi class, min/max alpha, k, w	Ank.	Ank.
7.	10/05/17	perbaikan max alpha +/-, k, findmax	Ank.	Ank.
8.	17/05/17	Perbaikan fungsi kernel, weight, bias	Ank.	Ank.
9.	24/05/17	Finalisasi fungsi testing pd core SVM	Ank.	Ank.
10.	12/06/17	Multiclass testing	Ank.	Ank.
11.	22/06/17	Perbaikan multiclass testing	Ank.	Ank.
12.	05/07/17	Perbaikan return input & db	Ank.	Ank.
13.	07/07/17	convert image ke threshold	Ank.	Ank.
14.	12/07/17	Finalisasi konversi image	Ank.	Ank.
15.	17/07/17	Konsultasi Laporan	Ank.	Ank.
16.	25/07/17	Project demo	Ank.	Ank.
17.	28/07/17	Finalisasi Project + Laporan	Ank.	Ank.
18.	04/08/17	Acc	Ank.	Ank.
19.				

Malang, 4 Agustus 2017  
Dosen Pembimbing Skripsi,

Yoppy Yughashawa, S.ST, M.Sc  
NIP.

## Lampiran 7. Lembar Persetujuan Maju Ujian Skripsi



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



NO SKRIPSI: 101

### LEMBAR PERSETUJUAN MENGIKUTI UJIAN SKRIPSI 2016/2017 PROGRAM STUDI TEKNIK INFORMATIKA

**N A M A** : Ayundha Wulan Kurniawati **NIM / K E L A S** : 1341180153 / TI - 4D  
**JUDUL SKRIPSI** : Implementasi Metode *Support Vector Machine* (SVM) untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)  
**PEMBIMBING** : 1. Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom NIP : 19810810 200501 2 002  
2. Yoppy Yunhasnawa, S.ST, M.Sc NIP :

No.	Uraian / Bab	Diselesaikan	Tanda Tangan	
			Pembimbing 1	Pembimbing 2
1.	PENDAHULUAN	✓		
2.	LANDASAN TEORI	✓		
3.	METODOLOGI PENELITIAN	✓		
4.	ANALISIS DAN PERANCANGAN	✓		
5.	IMPLEMENTASI	✓		
6.	PENGUJIAN DAN PEMBAHASAN	✓		
7.	KESIMPULAN DAN SARAN	✓		
8.	<b>BAGIAN AKHIR</b> - Daftar Pustaka - Lampiran ( <i>Isi lampiran disesuaikan dengan judul laporan akhir</i> ) - Profile Penulis ( <i>Riwayat Penulis</i> )	✓		
9.	<b>Hardware/Software</b> - Didemokan di depan pembimbing	✓		
10.	Draft Makalah	✓		

Malang, Agustus 2017  
Ketua Pelaksana LA & SKRIPSI 2016/2017  
Program Studi Teknik Informatika

**Arief Prasetyo, S.Kom., M.Kom.**  
NIP. 19790313 200812 1 002

FRM.RTI.01.49.04

DISETUJUI UNTUK DAPAT MAJU UJIAN SETELAH HASIL KARYA DINILAI LAYAK SERTA HASIL UJI SESUAI DENGAN SPESIFIKASI YANG DIRENCANAKAN

**Pembimbing I**

**Pembimbing II**

Ariadi Retno Tri Hayati Ririd,  
S.Kom., M.Kom  
NIP. 19810810 200501 2 002

Yoppy Yunhasnawa, S.ST, M.Sc  
NIP.

## Lampiran 8. Lembar Revisi Penguji I



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



### FORM REVISI SKRIPSI

No. Skripsi : 101

Nama Mahasiswa : Ayundha Wulan Kurniawati NIM : 1341180153  
Tanggal Ujian : 9 Agustus 2017  
Judul : Implementasi Metode *Support Vector Machine* (SVM) untuk  
Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus:  
Temu Tani Malang)

NO	SARAN PERBAIKAN	PARAF
1	Jelaskan arah dan penelitian ini	} 4
2	Tata tulis : source code + tabel (spasi)	
3	Buat aplikasi untuk end-user	

Malang, Agustus 2017

Dosen Penguji,

*[Signature]*  
(.....)

### FORM VERIFIKASI:

Laporan Akhir telah diperbaiki sesuai dengan saran perbaikan dari dosen penguji.

PENGUJI/PEMBIMBING	NAMA	TTD	TANGGAL
Penguji	Yan Wategulis S., ST., MMT	<i>[Signature]</i>	18-08-2017
Pembimbing 1	Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom	<i>[Signature]</i>	22-8-2017
Pembimbing 2	Yoppy Yunhasnawa, S.ST, M.Sc	<i>[Signature]</i>	22-08-2017





KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



### FORM REVISI SKRIPSI

No. Skripsi : 101

Nama Mahasiswa : Ayundha Wulan Kurniawati NIM : 1341180153  
Tanggal Ujian : 29 Agustus 2017  
Judul : Implementasi Metode *Support Vector Machine* (SVM) untuk  
Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus:  
Temu Tani Malang)

NO	SARAN PERBAIKAN	PARAF
1	Tambahkan primary key pada database	uf
2	Buat analisis hasil pengujian <del>disertai</del> disertai parameter <sup>2</sup> nya	uf
3	Perbaiki kesimpulan	uf

Malang, Agustus 2017

Dosen Penguji,

(.....w.s.....)

### FORM VERIFIKASI:

Laporan Akhir telah diperbaiki sesuai dengan saran perbaikan dari dosen penguji.

PENGUJI/PEMBIMBING	NAMA	TTD	TANGGAL
Penguji	Yan Wageulis S., ST., MMT	uf	6-9-2017
Pembimbing 1	Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom	uf	6-9-2017
Pembimbing 2	Yoppy Yunhasnawa, S.ST, M.Sc	uf	6-9-2017

## Lampiran 9. Lembar Revisi Penguji II



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



### FORM REVISI SKRIPSI

No. Skripsi : 101

Nama Mahasiswa : Ayundha Wulan Kurniawati NIM : 1341180153  
Tanggal Ujian : 9 Agustus 2017  
Judul : Implementasi Metode *Support Vector Machine* (SVM) untuk Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)

NO	SARAN PERBAIKAN	PARAF
1.	Penjelasan gambar & tabel.	Ace
2.	Sitasi <del>standarkan</del> sesuai prosedur	
3.	Perhitungan SVM error / iterasi / hyperplane?	
4.	Aplikasi labelling class otomatis	
5.	Precision & Recall?	
6.	Coba di praktekkan	
7.	80x80? → citra asli	
8.	Block diagram fungsi hyperplane?	

9. Baku Database?  
10. fitur grayscale referensinya?  
11. Segmentasi

Malang, 9 Agustus 2017  
Dosen Penguji,  
(Rosa Andria A.)

### FORM VERIFIKASI:

Laporan Akhir telah diperbaiki sesuai dengan saran perbaikan dari dosen penguji.

PENGUJI/PEMBIMBING	NAMA	TTD	TANGGAL
Penguji	Rosa Andria A.		22-8-2017
Pembimbing 1	Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom		22-8-2017
Pembimbing 2	Yoppy Yunhasnawa, S.ST, M.Sc		22-8-2017

12. Diujikan di Temu Tani Malang!

FRM.RTI.01.35.03



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
Jl. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



### FORM REVISI SKRIPSI

No. Skripsi : 101

Nama Mahasiswa : Ayundha Wulan Kurniawati NIM : 1341180153  
Tanggal Ujian : 29 Agustus 2017  
Judul : Implementasi Metode *Support Vector Machine* (SVM) untuk  
Identifikasi Penyakit Daun Tanaman Kubis (Studi Kasus:  
Temu Tani Malang)

NO	SARAN PERBAIKAN	PARAF
1.	Tabel 4.11 — 4.16 dihilangkan. &	Ace
2.	Update. &	
3.	Histeq / Otsu threshold digunakan. lalu masukkan ke laporan, kesimpulan, dan saran &	

Malang, 29 Agustus 2017

Dosen Penguji

Rosa Andrie A  
(.....)

### FORM VERIFIKASI:

Laporan Akhir telah diperbaiki sesuai dengan saran perbaikan dari dosen penguji.

PENGUJI/PEMBIMBING	NAMA	TTD	TANGGAL
Penguji	Rosa Andrie A		5-9-2017
Pembimbing 1	Ariadi Retno Tri Hayati Ririd, S.Kom., M.Kom		6-9-2017
Pembimbing 2	Yoppy Yunhasnawa, S.ST, M.Sc		6-9-2017



## Lampiran 10. Lembar Verifikasi Abstrak dan Tata Tulis



KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI  
POLITEKNIK NEGERI MALANG  
JURUSAN TEKNOLOGI INFORMASI  
PROGRAM STUDI TEKNIK INFORMATIKA  
JL. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122



No. Skripsi : 101

### FORM VERIFIKASI

#### ABSTRAK BAHASA INGGRIS DAN TATA TULIS BUKU SKRIPSI

**Nama Mahasiswa 1** : Ayundha Wulan Kurniawati      **NIM** : 1341180153  
**Tanggal Ujian** : 23 Agustus 2017  
**Judul** : Implementasi Metode *Support Vector Machine* (SVM) untuk Identifikasi  
Penyakit Daun Tanaman Kubis (Studi Kasus: Temu Tani Malang)

NO	BAGIAN YANG DIVERIFIKASI	NAMA VERIFIKATOR	TANGGAL VERIFIKASI	TTD
1	Abstrak Berbahasa Inggris	Satrio Binusa S., SS.	12 September 2017	
2	Tata Tulis Buku Skripsi	Yoppy Yunhasnawa, S.ST, M.Sc	11 SEPTEMBER 2017	

## **PROFIL PENULIS**



### **DATA PRIBADI**

Nama : Ayundha Wulan Kurniawati  
Tempat, Tanggal Lahir : Blitar, 21 Mei 1995  
Alamat : Jl. Diponegoro No. 17 Dsn Krajan, Pagerwojo,  
Kesamben, Blitar  
Email : ayundhawulan@gmail.com

### **DATA PENDIDIKAN**

2001 – 2007 : SDN Pagerwojo 1  
2007 – 2010 : SMPN 1 Kesamben  
2010 – 2013 : SMAN 1 Kesamben  
2013 – 2017 : Politeknik Negeri Malang