IMPLEMENTASI ALGORITME *K-MEANS* SEBAGAI METODE SEGMENTASI CITRA DALAM IDENTIFIKASI PENYAKIT DAUN JERUK

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Falih Gozi Febrinanto
NIM: 145150201111040

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI ALGORITME *K-MEANS* SEBAGAI METODE SEGMENTASI CITRA DALAM IDENTIFIKASI PENYAKIT DAUN JERUK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Falih Gozi Febrinanto
NIM: 145150201111040

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Candra Dewi, S.Kom, M.Sc	Dr. Ir. Anang Tri Wiratno
NIP: 19771114 200312 2 001	NIP: 19670107 119103 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan , S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 25 Mei 2018

Falih Gozi Febrinanto

NIM: 145150201111040

KATA PENGANTAR

Puja dan puji syukur kehadirat Allah SWT atas berkah dan limpahan rahmat-Nya penulis dapat menyelesaikan skripsi yang ini. Sholawat serta salam penulis haturkan atas junjungan besar nabi Muhammad SAW.

Pengerjaan skripsi ini merupakan sebagian persyaratan untuk memperoleh gelar sarjana. Pada akhirnya, hasil ini penulis persembahkan untuk orang-orang sekitar yang berjasa besar membantu dalam prosesnya. Untuk itu penulis berterimakasih kepada:

1. Keluarga penulis, kedua orang tua tercinta (Bapak Dafik & Ibu Lis Setyaningsih) yang tidak ada hentinya mendo'akan, memberikan dukungan moril serta materil yang tak ternilai harganya. Serta adik tercinta (Mahdan Kintara Sanie) yang terus mendukung dan mendo'akan penulis sebagai bagian dari keluarganya untuk berproses menjadi pribadi yang bermanfaat.
2. Pembimbing, Ibu Candra Dewi, S.Kom, M.Sc dan Bapak Dr. Ir. Anang Tri Wiratno yang senantiasa mengarahkan penulis dalam proses pengerjaan skripsi agar tetap berada pada garis kaidah penelitian yang benar. Serta menjadi sumber ilmu baru dalam pengalaman hidup penulis.
3. Bapak Edy Santoso, S.Si, M.Kom selaku Wakil Dekan III Bidang Kemahasiswaan yang senantiasa memberikan nasehat kepada penulis untuk bisa menyeimbangkan kegiatan akademik maupun non-akademik dalam kehidupan kampus.
4. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya, yang telah meberikan ilmu serta mendidik penulis dalam proses penempaan diri untuk berfikir dan berperilaku.
5. Annisa Amalia Nur'aini yang memberikan motivasi dan ketulusan dalam membatu pengerjaan skripsi ini.
6. M. Adi Wijaya dan Istania Salma selaku rekan yang memberikan pengalaman baru pada penulis pada kegiatan diluar studinya.
7. Talitha Raisa dan Restu Widodo selaku rekan dalam kegiatan Praktek Kerja Lapang (PKL) di Balai Penelitian Buah Jeruk dan Sub Tropika (Balitjestro) yang membantu penlis dalam memunculkan ide awal pada skripsi ini.
8. Rekan-rekan DPM FILKOM 2017 yang memberikan ruang dan kesempatan pada penulis untuk lebih peka terhadap sesama dan memperjuangkan kesempatan untuk satu tujuan bersama.
9. Rekan-rekan pengurus organisasi Forum Komunikasi Mahasiswa Jember di malang (FKMJM), BEM TIIK kabinet Bersatu III, *Basic Computing Community* (BCC) FILKOM , dan DPM FILKOM 2016 yang memberikan pengalaman pada penulis dalam berorganisasi.

10. Keluarga Besar Mahasiswa Fakultas Ilmu Komputer (KBMFILKOM) yang memberikan warna tersendiri bagi penulis pada proses masa studinya.

Tak ada gading yang tak retak. Kesempurnaan sejatinya hanya milik Tuhan Yang Maha Esa. Segala kritik dan masukan penulis akan terima. Semoga, Skripsi ini dapat bermanfaat untuk pembaca dan dapat dijadikan bahan referensi untuk kebutuhan tertentu.

Malang, 27 Mei 2018

Penulis

falihgozifeb@gmail.com

ABSTRAK

Salah satu faktor yang menyebabkan rendahnya kualitas tanaman jeruk adalah penyakit yang menyerang pada daunnya. Perkembangan teknologi informasi pada bidang pengolahan citra digital memungkinkan untuk melakukan identifikasi penyakit daun jeruk secara otomatis. Pada penelitian ini, dilakukan identifikasi penyakit daun jeruk yang meliputi *Downy Mildew*, Cendawan Jelaga, dan CVPD (*Citrus Vein Phloem Degeneration*). Proses identifikasi penyakit daun jeruk diawali dengan proses *resizing* untuk menyeragamkan ukuran citra dan proses *rescaling* untuk melakukan pengaturan terhadap kecerahan citra. Selanjutnya, melakukan perubahan ruang warna dari RGB menjadi L*a*b*. Setelah melakukan perubahan ruang warna, hasil perubahan digunakan sebagai *input* pada segmentasi citra menggunakan algoritme *K-Means*. Terdapat dua bagian segmentasi yaitu segmentasi daun dan segmentasi penyakit. Setelah melakukan segmentasi, hasil dari segmentasi penyakit diklasifikasikan menggunakan algoritme *K-Nearest Neighbor* (K-NN) terhadap data latih untuk diketahui kelas penyakitnya. Pengujian yang dilakukan pada penelitian ini yaitu pengujian nilai *Scale Factor*, nilai *cluster* optimal, dan nilai K optimal. Berdasarkan tiga pengujian yang dilakukan, didapatkan rekomendasi nilai *Scale Factor* yaitu 1.1, nilai *cluster* optimal pada segmentasi daun yaitu 2, nilai *cluster* optimal pada segmentasi penyakit 9, dan nilai K optimal yaitu 4. Akurasi tertinggi yang didapatkan untuk identifikasi penyakit pada penelitian ini adalah 90.83%. Setelah dilakukan analisis lebih lanjut, hasil akurasi program dapat ditingkatkan kembali dengan menggunakan parameter batas minimal. Berdasarkan pengujian parameter batas minimal, hasil menujukkan nilai optimal yang didapat sebesar 3% dan akurasi didapatkan untuk identifikasi penyakit adalah 99.17%.

Kata kunci: Penyakit Daun, Segmentasi Citra, Identifikasi, *K-Means*, *K-Nearest Neighbor* (K-NN).

ABSTRACT

*One of the factors that causes poor quality of citrus crops is the disease which attacks the leaves. The development of information technology in digital image processing field allows to identify the citrus leaf disease automatically. This research identifies the citrus leaf disease includes Downy Mildew, Cendawan Jelaga, and CVPD (Citrus Vein Phloem Degeneration). The identification process of citrus leaf disease begins with resizing to equalize image size and rescaling to adjust the image brightness. Next, converting RGB to L*a*b* color space. After converting the color space, the results of the conversion is used as an input to image segmentation using K-Means algorithm. There are two segmentation parts, namely leaf segmentation and disease segmentation. After segmentation process, the results of disease segmentation are classified by using K-Nearest Neighbor (K-NN) algorithm on the train data to knows the class of their diseases. Tests conducted on this research are testing the value of Scale Factor, optimal cluster value, and optimal K value. Based on the three conducted tests, it recommends that the Scale Factor value is 1.1, the optimal cluster value on leaf segmentation is 2, the optimal cluster value on the segmentation of disease is 9, and the optimal K value is 4. The highest accuracy that obtained for disease identification in this research is 90.83%. After futher analysis, the accuracy of the program can be improved by using the minimum limit parameter. The minimum limit parameter test shows that the optimal value is 3% and the accuracy that obtained for disease identification is 99.17%.*

Keywords: *Leaf Disease, Image Segmentation, Identification, K-Means, K-Nearest Neighbor (K-NN)*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

IMPLEMENTASI ALGORITME *K-MEANS* SEBAGAI METODE SEGMENTASI CITRA DALAM IDENTIFIKASI PENYAKIT DAUN JERUK

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Falih Gozi Febrinanto
NIM: 145150201111040

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI ALGORITME *K-MEANS* SEBAGAI METODE SEGMENTASI CITRA DALAM IDENTIFIKASI PENYAKIT DAUN JERUK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Falih Gozi Febrinanto
NIM: 145150201111040

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Candra Dewi, S.Kom, M.Sc	Dr. Ir. Anang Tri Wiratno
NIP: 19771114 200312 2 001	NIP: 19670107 119103 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan , S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 25 Mei 2018

Falih Gozi Febrinanto

NIM: 145150201111040

KATA PENGANTAR

Puja dan puji syukur kehadirat Allah SWT atas berkah dan limpahan rahmat-Nya penulis dapat menyelesaikan skripsi yang ini. Sholawat serta salam penulis haturkan atas junjungan besar nabi Muhammad SAW.

Pengerjaan skripsi ini merupakan sebagian persyaratan untuk memperoleh gelar sarjana. Pada akhirnya, hasil ini penulis persembahkan untuk orang-orang sekitar yang berjasa besar membantu dalam prosesnya. Untuk itu penulis berterimakasih kepada:

1. Keluarga penulis, kedua orang tua tercinta (Bapak Dafik & Ibu Lis Setyaningsih) yang tidak ada hentinya mendo'akan, memberikan dukungan moril serta materil yang tak ternilai harganya. Serta adik tercinta (Mahdan Kintara Sanie) yang terus mendukung dan mendo'akan penulis sebagai bagian dari keluarganya untuk berproses menjadi pribadi yang bermanfaat.
2. Pembimbing, Ibu Candra Dewi, S.Kom, M.Sc dan Bapak Dr. Ir. Anang Tri Wiratno yang senantiasa mengarahkan penulis dalam proses pengerjaan skripsi agar tetap berada pada garis kaidah penelitian yang benar. Serta menjadi sumber ilmu baru dalam pengalaman hidup penulis.
3. Bapak Edy Santoso, S.Si, M.Kom selaku Wakil Dekan III Bidang Kemahasiswaan yang senantiasa memberikan nasehat kepada penulis untuk bisa menyeimbangkan kegiatan akademik maupun non-akademik dalam kehidupan kampus.
4. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya, yang telah meberikan ilmu serta mendidik penulis dalam proses penempaan diri untuk berfikir dan berperilaku.
5. Annisa Amalia Nur'aini yang memberikan motivasi dan ketulusan dalam membatu pengerjaan skripsi ini.
6. M. Adi Wijaya dan Istania Salma selaku rekan yang memberikan pengalaman baru pada penulis pada kegiatan diluar studinya.
7. Talitha Raisa dan Restu Widodo selaku rekan dalam kegiatan Praktek Kerja Lapang (PKL) di Balai Penelitian Buah Jeruk dan Sub Tropika (Balitjestro) yang membantu penlis dalam memunculkan ide awal pada skripsi ini.
8. Rekan-rekan DPM FILKOM 2017 yang memberikan ruang dan kesempatan pada penulis untuk lebih peka terhadap sesama dan memperjuangkan kesempatan untuk satu tujuan bersama.
9. Rekan-rekan pengurus organisasi Forum Komunikasi Mahasiswa Jember di malang (FKMJM), BEM TIIK kabinet Bersatu III, *Basic Computing Community* (BCC) FILKOM , dan DPM FILKOM 2016 yang memberikan pengalaman pada penulis dalam berorganisasi.

10. Keluarga Besar Mahasiswa Fakultas Ilmu Komputer (KBMFILKOM) yang memberikan warna tersendiri bagi penulis pada proses masa studinya.

Tak ada gading yang tak retak. Kesempurnaan sejatinya hanya milik Tuhan Yang Maha Esa. Segala kritik dan masukan penulis akan terima. Semoga, Skripsi ini dapat bermanfaat untuk pembaca dan dapat dijadikan bahan referensi untuk kebutuhan tertentu.

Malang, 27 Mei 2018

Penulis

falihgozifeb@gmail.com

ABSTRAK

Salah satu faktor yang menyebabkan rendahnya kualitas tanaman jeruk adalah penyakit yang menyerang pada daunnya. Perkembangan teknologi informasi pada bidang pengolahan citra digital memungkinkan untuk melakukan identifikasi penyakit daun jeruk secara otomatis. Pada penelitian ini, dilakukan identifikasi penyakit daun jeruk yang meliputi *Downy Mildew*, Cendawan Jelaga, dan CVPD (*Citrus Vein Phloem Degeneration*). Proses identifikasi penyakit daun jeruk diawali dengan proses *resizing* untuk menyeragamkan ukuran citra dan proses *rescaling* untuk melakukan pengaturan terhadap kecerahan citra. Selanjutnya, melakukan perubahan ruang warna dari RGB menjadi L*a*b*. Setelah melakukan perubahan ruang warna, hasil perubahan digunakan sebagai *input* pada segmentasi citra menggunakan algoritme *K-Means*. Terdapat dua bagian segmentasi yaitu segmentasi daun dan segmentasi penyakit. Setelah melakukan segmentasi, hasil dari segmentasi penyakit diklasifikasikan menggunakan algoritme *K-Nearest Neighbor* (K-NN) terhadap data latih untuk diketahui kelas penyakitnya. Pengujian yang dilakukan pada penelitian ini yaitu pengujian nilai *Scale Factor*, nilai *cluster* optimal, dan nilai K optimal. Berdasarkan tiga pengujian yang dilakukan, didapatkan rekomendasi nilai *Scale Factor* yaitu 1.1, nilai *cluster* optimal pada segmentasi daun yaitu 2, nilai *cluster* optimal pada segmentasi penyakit 9, dan nilai K optimal yaitu 4. Akurasi tertinggi yang didapatkan untuk identifikasi penyakit pada penelitian ini adalah 90.83%. Setelah dilakukan analisis lebih lanjut, hasil akurasi program dapat ditingkatkan kembali dengan menggunakan parameter batas minimal. Berdasarkan pengujian parameter batas minimal, hasil menujukkan nilai optimal yang didapat sebesar 3% dan akurasi didapatkan untuk identifikasi penyakit adalah 99.17%.

Kata kunci: Penyakit Daun, Segmentasi Citra, Identifikasi, *K-Means*, *K-Nearest Neighbor* (K-NN).

ABSTRACT

*One of the factors that causes poor quality of citrus crops is the disease which attacks the leaves. The development of information technology in digital image processing field allows to identify the citrus leaf disease automatically. This research identifies the citrus leaf disease includes Downy Mildew, Cendawan Jelaga, and CVPD (Citrus Vein Phloem Degeneration). The identification process of citrus leaf disease begins with resizing to equalize image size and rescaling to adjust the image brightness. Next, converting RGB to L*a*b* color space. After converting the color space, the results of the conversion is used as an input to image segmentation using K-Means algorithm. There are two segmentation parts, namely leaf segmentation and disease segmentation. After segmentation process, the results of disease segmentation are classified by using K-Nearest Neighbor (K-NN) algorithm on the train data to knows the class of their diseases. Tests conducted on this research are testing the value of Scale Factor, optimal cluster value, and optimal K value. Based on the three conducted tests, it recommends that the Scale Factor value is 1.1, the optimal cluster value on leaf segmentation is 2, the optimal cluster value on the segmentation of disease is 9, and the optimal K value is 4. The highest accuracy that obtained for disease identification in this research is 90.83%. After futher analysis, the accuracy of the program can be improved by using the minimum limit parameter. The minimum limit parameter test shows that the optimal value is 3% and the accuracy that obtained for disease identification is 99.17%.*

Keywords: *Leaf Disease, Image Segmentation, Identification, K-Means, K-Nearest Neighbor (K-NN)*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisikan landasan kepustakaan yang membagas tentang teori, konsep, metode atau literatur yang berkaitan dengan penelitian. Landasan kepustakaan berisi penelitian yang sudah dilakukan sebelumnya dibedakan menjadi landasan kepustakaan yang berkaitan dengan kasus dan landasan kepustakaan yang berkaitan dengan metode. Beberapa teori metode juga dibahas pada bab ini antara lain tanaman Jeruk Keprok Batu 55, penyakit daun jeruk, pengolahan citra digital, algoritme *K-Means*, konsep ruang warna, algoritme K-NN, dan perhitungan jarak menggunakan *Euclidean Distance*.

2.1 Landasan kepustakaan

Terdapat beberapa penelitian sebelumnya yang memiliki keterkaitan dengan penelitian ini. Pada bagian ini dibagi menjadi 2 landasan kepustakaan. Yang pertama adalah landasan kepustakaan yang berkaitan dengan kasus yaitu identifikasi penyakit daun jeruk yang ditunjukkan pada Tabel 2.1. Kedua adalah landasan kepustakaan yang berkaitan dengan metode yaitu segmentasi citra menggunakan algoritme *K-Means* yang ditunjukkan pada Tabel 2.2.

Tabel 2.1 Landasan Kepustakaan Berkaitan dengan Kasus

No.	Judul dan Objek Penelitian	Sumber	Metode	Hasil
1.	**Judul:** Implementasi Metode *K-Nearest Neighbor* untuk Identifikasi Penyakit Tanaman Jeruk Keprok berdasarkan Citra daun **Objek:** Membuat aplikasi pengenalan penyakit jeruk keprok melalui citra daun. Pengenalan menggunakan latar belakang putih. Mengenali tiga macam penyakit yaitu	(Priambodo, Dewi, & Triwiratno, 2015)	1. Melakukan *pre-processing* pada citra *input* daun jeruk 2. Melakukan ekstraksi ciri dengan mengambil nilai rata-rata *red, green, blue* (RGB) dari objek daun 3. Melakukan klasifikasi data uji menggunakan metode *K-Nearest Neighbor* (K-NN). Dengan diawali menentukan nilai K lalu dilakukan perhitungan jarak data latih dengan data uji	Penelitian menghasilkan akurasi mencapai 96,67% dengan dapat mengklasifikasikan 29 dari 30 data. Hasil didapatkan dengan menggunakan nilai *Scale Factor* 1.4, perhitungan jarak *Euclidean*, nilai K=2, dan dengan data latih sebanyak 30 untuk setiap kelas dengan total 90 data

	Cendawan Jelaga, *mildew*, dan CVPD		menggunakan perhitungan jarak *Euclidean*	
2.	**Judul:** Implementasi Algoritma *Multilevel Thresholding* Menggunakan *Otsu* sebagai *Preprocessing* Data Citra Daun pada Proses Identifikasi Penyakit Tanaman Jeruk **Objek:** *Input* berupa citra daun jeruk yang berpenyakit CVPD, Cendawan Jelaga, *Downy Mildew*, dan Defisiensi (Mg dan Zn). Menggunakan kamera laboratorium untuk mengambil gambar citra daunnya. Dengan rincian 30 gambar setiap penyakit daun ditambah 30 gambar daun yang sehat sebagai data latih. Dan 50 daun sebagai data uji.	(Rizal, Dewi, & Widodo, 2016)	1. Melakukan penambahan nilai konstanta dari kecerahan citra *input* 2. Melakukan penerapan *multilevel thresholding* menggunakan *Otsu* untuk mendapatkan bagian citra yang berpenyakit 3. Melakukan ekstraksi ciri dengan mengambil nilai rata-rata *red,green,blue* (RGB) 4. Melakukan klasifikasi dengan metode *K-Nearest Neighbor* (KNN)	Pengujian menggnakan nilai *Scale Factor*, nilai *MLEVEL*, dan nilai K. Berdasarkan tiga pengujian tersebut didapatkan hasil yaitu nilai *Scale Factor* 1.4, nilai *MLEVEL* yaitu 3, dan nilai k yaitu 2. Hasil penelitian dengan tiga tipe data uji menghasilkan akurasi tertinggi sebesar 92% pada data defisiensi Zn, 80% pada defisiensi Mg, dan 80% pada defisiensi Mg dan Zn

Tabel 2.2 Landasan Kepustakaan Berkaitan dengan Metode

No.	Judul dan Objek Penelitian	Sumber	Metode	Hasil
1.	**Judul:** Aplikasi Pengolahan Citra *Digital Meat Detection* Dengan	(Arsy, Nurhayati, & Martono, 2016)	1. Langkah pertama dengan pengambilan citra daging melalui	Hasil identifikasi kualitas daging sapi pada aplikasi pengolahan citra

	Metode Segmentasi *K-Means Clustering* Berbasis *OpenCV* dan *Eclipse* **Objek:** Membuat aplikasi berbasis mobile untuk menentukan mutu daging sapi. Menggunakan *input*-an citra daging sapi sebanyak 20 citra daging sapi. Dengan melakukan pencocokan terhadap tekstur dan warna terhadap sampel warna. Penentuan mutu citra daging sapi menggunakan pengukuran *marbling score* yang beracuan pada standar warna daging sapi sebanyak 9 skor dan standar warna lemak sebanyak 9 skor kemudian disesuaikan menurut kondisi tingkat mutu daging		kamera *smartphone* atau menggunakan sampel data yang sudah tersedia pada galeri 2. Melakukan segmentasi citra menggunakan *K-Means* 3. Melakukan ekstraksi ciri hasi segmentasi dengan meghitung nilai *mean* dan standar deviasi 4. Mencocokkan dengan rentan *mean* dan standar deviasi setiap mutu daging	digital dengan metode *K-Means clustering* ini menghasilkan keluaran yang baik dengan presentase 80% keberhasilan
2.	**Judul:** Segmentasi Buah Menggunakan Metode *K-Means Clustering* dan Identifikasi Kematangannya Menggunakan Metode Perbandingan Kadar Warna	(Andri, et al., 2014)	1. Langkah pertama pengujian dalam mendeteksi buah dan tingkat kematangan, pengambilan citra menggunakan *webcam*/penelusuran citra 2. Melakukan *pre-processing* terhadap citra *input* dengan	Dengan segmentasi citra menggunakan algoritme *K-Means* dihasilkan deteksi buah menggunakan sifat geometris dan deteksi kematangan menggunakan ciri warna menghasilkan keluaran yang baik

	Objek: Mendeteksi buah berdasarkan sifat geometris tingkat kebulatan dan mengidentifikasi tingkat kematangan menggunakan ciri warnanya. Buah yang dideteksi adalah buah jeruk, pisang, dan cabai. Mengidentifikasi tingkat kematangan dengan kalkulasi kadar warna RGB yang diklasifikasikan menjadi 3 tingkatan yaitu, *Ripe* (matang), *Half-Ripe* (setengah matang), dan *Un-Ripe* (tidak matang) sesuai dengan rata-rata kadar warna dan standar deviasi		penyeragaman ukuran citra 3. Melakukan segmentasi citra menggunakan algoritme *K-Means* 4. Representasi dan deskripsi. Dengan melakukan pelabelan fitur warna dan fitur geometris 5. Pengenalan dan interpretasi. Dengan melakukan proses klasifikasi dari basis pengetahuan hasil pelatihan data	dengan akurasi sebesar 93.89%
3.	**Judul:** *Potato Leaf Diseases Detection and Classification System* **Objek:** Deteksi dan klasifikasi penyakit daun kentang dengan algoritme jaringan syaraf tiruan dan segmentasi citra menggunakan algoritme *K-Means*. Menggunakan 3 fitur yaitu warna, tekstur, dan area yang nantinya digunakan	(Athanikar & Badar, 2016),	1. *Input* citra daun kentang 2. Melakukan *pre-processing* terhadap citra daun kentang. *Resize* disesuaikan dengan ukuran tetap. *filtering* untuk meningkatkan kecerahan, kontras, dan menambahkan tekstur serta menambah efek khusus pada gambar yang meningkatkan hasil segmentasi	Dari total 150 sampel daun kentang yang terdiri dari 50 daun sehat dan 100 daun berpenyakit serta menggunakan algoritme *K-Means* dengan k=3 kebanyakan gagal untuk melakukan pemisiahan area yang berpenyakit. Tetapi, dibeberapa kasus area daun yang berpenyakit dapat dipisah dengan sempurna. Contoh daum yang

	sebagai acuan dari klasifikasi penyakit daun kentang.		*3.* Segmentasi citra menggunakan algoritme *K-Means* 4. Ekstraksi fitur 5. Melakukan klasifikasi menggunakan *back Propagation Neural Network* (BPNN) untuk menentukan daun kentang sehat atau terkena penyakit.	dapat dipisah dengan sempurna menghasilkan akurasi sebesar 92% untuk tahap klasifikasinya. Dari penelitian ini diperlukan perbaikan dalam proses *pre-procesing*-nya dan batasan data *input*nya
4.	**Judul:** *White Blood Cell Segmentation by Color-Space-Based K-Means Clustering* **Objek:** Segmentasi sel darah putih untuk kepentingan sitometri (untuk memilah dan mengelompokkan sel). Bagian yang disegmentasi berupa nukleus dan sitoplasma sel darah putih itu sendiri. proses dekomposisi dan *K-Means clustering* digunakan untuk segmentasi	(Zhang, et al., 2014)	1. *Input* gambar citra asli 2. Melakukan segmentasi menggunakan *K-Means* untuk mengasilkan sel darah putih yang bercampur sel darah merah, nukleus, dan sel darah merah yang bercampur dengan nukleus. Proses segmentasi awal digunakan untuk menghasilkan daerah yang terdiri atas nukleus saja 3. Melakukan pengaturan warna. Ruang warna HSI digunakan untuk proses segmentasi guna menghasilkan bagian sel darah putih yang bercampur sel darah merah. Ruang warna CMYK digunakan untuk proses segmentasi	Sebanyak 300 gambar sel darah digunakan untuk proses pengujian penelitian ini. Dan membandingkan antara segmentasi manual dengan segmentasi otomatis menggunakan algoritme *K-Means*. Hasil dari penelitian ini menghasilkan akurasi sebesar 95,7% untuk segmentasi nukleus dan 93,1% untuk segmentasi sitoplasma

			guna menghasilkan sel darah yang bercampur dengan nukleus 4. Melakukan penghapusan daerah menggunakan *image substract*. Daerah sel darah merah yang bercampur dengan nukleus dikurangkan dengan daerah nukleus yang menghasilkan citra sel darah merah saja. Daerah sel darah putih yang bercampur dengan sel darah merah dikurangkan dengan citra hasil sel darah merah dan menghasilkan citra sel darah putih saja 5. Penghilangan *noise* untuk citra sel darah putih	
5.	**Judul:** Segmentasi Citra Berwarna dengan Menggunakan Metode *Clustering* Berbasis *Patch* untuk Identifikasi *Mycobacterium Tuberculosis*	(Rulaningtyas, et al., 2015)	4. Mencoba melakukan analisis hasil global thresholding pada citra bakteri sebagai teknik segmentasi dasar menggunakan tiga ruang warna berbeda yaitu RGB, HSV, dan CIE lab. Kemudian dipilih satu ruang warna yang paling optimal	Dari perbandingan tiga ruang warna menghasilkan ruang warna CIE lab sebagai ruang warna yang paling optimal untuk segmentasi. Dan untuk segmentasi citra bakteri menggunakan K-NN memiliki hasil yang paling baik dengan akurasi 97,90%

	Objek: Proses segmentasi citra berwarna pada citra mikroskopis bakteri TBC (*mycobacterium tuberculosis*) dari dahak pasien. Citra bakteri TBC diberi warna dengan metode pewarnaan *Ziehl – Neelsen* yang menghasilkan efek warna merah pada bakteri TB dan warna biru untuk *background*. Membandingkan tiga metode segmentasi yatu thresholding, K-NN, dan *K-Means* terhadap citra bakteri		5. Melakukan segmentasi citra bakteri dengan percobaan menggunakan algoritme *thresholding*, K-NN, dan *K-Means* 6. Membandingkan hasil segmentasi Mencoba mengkombinasikan basis patch citra dengan metode segmentasi *K-Means*	namun membutuhkan waktuk komputasi yang lama sekitar 6 menit. Oleh karena itu peneliti menggunaka metode *K-Means* dengan basis *patch* yang menghasilkan akurasi 100% dalam segementasi citra yang diduga bakteri TB
6.	**Judul :** *Infected Fruit Part Detection using K-Means Clustering Segmentation Technique* **Objek:** Segmentasi terhadap bagian buah cacat berdasarkan fitur warna menggunakan algoritme *K-Means* untuk segmentasi citra. Peneliti menggunakan buah apel untuk mendeteksi bagian cacat dari buah apel	(Dubey, et al., 2013)	1. Masukan citra *input* 2. Mengubah ruang warna RGB menjadi ruang warna L*a*b* 3. Melakukan segmentasi menggunakan algoritme *K-Means* 4. Memberikan label untuk masing-masing *pixel* hasil segmentasi *K-Means* 5. Mengelompokkan citra sesuai label warna 6. Menentukan bagian cacat pada buah	Pada segmentasi menggunakan *K-Means* dapat menghasilkan evaluasi penggunaan 3 atau 4 klaster untuk solusi optimal pada segmentasi bagian cacat buah. Hasil penelitian menunjukkan penggunaan algoritme *K-Means* akurat untuk mengelompokkan daerah cacat pada buah.

Dari landasan kepustakaan pada tabel 2.1, diperlihatkan beberapa penelitian dengan studi kasus yang sama dan dapat digunakan sebagai referensi untuk menyelesaikan kasus dengan keluaran yang lebih baik lagi. Sedangkan pada tabel 2.2, diperlihatkan beberapa contoh penelitian penggunaan metode yang sama. Hal ini digunakan sebagai referensi untuk mengoptimalkan implementasi dari metode untuk kasus yang diselesaikan.

2.2 Dasar Teori

2.2.1 Jeruk Keprok Batu 55

Jeruk Keprok Batu 55 masuk dalam kategori jeruk mandarin, masayarakat khususnya pasar di indonesia belum banyak mengetahui bahwa mandarin adalah sinonim dari jeruk keprok. Jeruk ini memiliki kualitas dan rasa sama bahkan lebih baik dibandingkan dengan buah jeruk import. Di wilayah pengembangan, jeruk keprok Batu 55 berdampak untuk peningkatan produksi dan pendapatan serta sebagai pengganti jeruk import. Usaha tani jeruk jika dikelola secara baik dan serius dengan baku teknis budidaya jeruk dipercaya dapat mensejahterakan petani jeruk Indonesia (Setiono, 2015).

Gambar 2.1 Detail Keprok Batu 55

Sumber: (Balitjestro, 2017)

2.2.2 Penyakit Daun Jeruk

2.2.2.1 CVPD (*Citrus Vein Phloem Degeneration*)

Penyakit CVPD disebabkan oleh *Liberobacter* yang merupakan bakteri gram negatif yang tergolong subdivisi *Protobacteria* (Meitayani, Adiartayasa, & Wijaya, 2014). Penyakit CVPD merupakan penyakit yang cukup berbahaya. Gejala penyakit ini digolongkan menjadi dua kelompok yaitu gejala luar dan gejala dalam. Gejala

luar memiliki ciri warna kekuning-kuningan pada daun dewasa seperti halnya kekurangan unsur Zn, Mn dan Fe. Tulang daun halus berwarna lebih hijau daripada jaringan daun lainnya. Sedangkan gejala dalam apabila diiris melintang dari ibu tulang daun/tangkai daun akan terlihat kelainan pada floemnya. Jaringan floem daun dewasa memperlihatkan gejala tingkat ketebalan yang lebih dari pada jaringan floem daun yang berwarna hijau.

CVPD memberikan dampak keberlanjutan menyebabkan kemunduran hasil seperti buah yang kecil, buah yang tidak dapat berkembang, dan keguguran buah pada tanaman jeruk sedangkan buah yang tidak gugur memiliki kualitas yang sangat rendah. CVPD mengakibatkan klorosis pada tanaman yang menghambat tanaman untuk melakukan fotosistesis sehingga daun tidak mampu memberi makanan pada seluruh bagian tanaman. Pertumbuhan menjadi sangat terhambat dan pada akhirnya tanaman menjadi kering, layu dan mati (Wahyuningsih, 2009).

Gambar 2.2 Daun Berpenyakit CVPD

2.2.2.2 Cendawan Jelaga

Penyakit Cendawan Jelaga disebabkan oleh jamur *Capnodium citri*. Penyakit ini dapat dikenal dengan nama lain penyakit embun jelaga menyerang daun, ranting, dan buah sehingga bagian yang terserang memiliki lapisan kumpulan jamur berwarna hitam. Penyebaran penyakit ini bisa lewat penyembaran langsung akibat terkelupasnya kumpulan jamur dan diterbangkan oleh angin ke tanaman yang sehat. Buah yang terserang biasanya memiliki ukuran lebih kecil dan terlambat matang. Jamur *Capnodium citri* lebih cepat berkembang dengan adanya sekresi embun madu yang dihasilkan hama kutu daun sebagai tempat pertumbuhannya. Tindakan pengendalian penyakit ini dengan cara mengendalikan populasi hama kutu-kutu daun (aphis) dan melakukan penyemprotan detergen 5% dengan waktu dua kali sebulan (Syafril, 2006).

Gambar 2.3 Daun Berpenyakit Cendawan Jelaga

2.2.2.3 *Downy Mildew*

Penyakit *Downy Mildew* dikenal dengan nama lain penyakit embun tepung diakibatkan oleh serangan jamur berbentuk tepung berwarna putih yang menyerangang permukaan daun dan dapat mengakibatkan daun mengering. Fase kritis serangan penyakit ini adalah pada fase pertunasan dan daun muda yang sedang tumbuh. Buah dari tanaman yang terserang penyakit ini menjadi mudah gugur. Penyebaran dan perkembangan penyakit terutama terjadi pada hari dan cuaca yang cukup lembab serta diikuti oleh paparan sinar matahari selama beberapa jam pada musih hujan.

Upaya yang dapat dilakukan untuk pengendalian antara lain dengan melakukan pemangkasan terhadap tunas yang terserang, melakukan penyemprotan saat akan bertunas dan diulang pada saat daun muda dengan menggunakan fungisida seperti Propineb, Siprokonozal, dan Benomil (Syafril, 2006).

Gambar 2.4 Daun Berpenyakit *Downy Mildew*

2.2.3 Pengolahan Citra Digital

Pengolahan citra digital adalah suatu teknologi yang digunakan dengan mengimplementasikan berbagai macam algoritme komputer untuk memproses citra digital. Hasil dari pemrosesan bisa berupa citra digital dengan kualitas yang tinggi ataupun menghasilkan suatu representasi karakteristik atau ciri dari citra

asli. Implementasi pengolahan citra digital dapat digunakan dibeberapa aspek seperti sistem kecerdasan buatan, citra medis, pengendalian jarak jauh, fotografi, dan forensik (Zhou, Wu, & Zhang, 2010).

2.2.3.1 *Resizing*

Resizing merupakan proses untuk melakukan perubahan luas dari citra menjadi lebih besar ataupun lebih kecil dari ukuran aslinya.dengan melakukan perubahan terhadap ukuran dapat mengakibatkan pergeseran pada nilai warna sehingga mengubah konten digital yang ada didalamnya (Wijaya & Prayudi, 2015). Proses *resizing* juga bertujan untuk menyamakan ukuran citra dalam suatu kasus pemrosesan tertentu agar mempercepat proses penyelesaian dari suatu kasus.

2.2.3.2 Segmentasi

Segmentasi citra merupakan salah satu proses pengolahan citra digital. Segmentasi citra memiliki arti membagi citra menjadi beberapa grup *pixel* dengan antara masing-masing grup memiliki kontras yang tinggi dan pada satu grup memiliki kesamaan yang tinggi (Dhanachandra, Manglem, & Chanu, 2015).

2.2.3.3 *Rescaling*

Rescaling merupakan proses menambah kecerahan pada citra yang menggunakan operator rescalling. Proses ini dilakukan dengan cara mengalikan tiap-tiap bagian ruang warna dengan nilai *Scale Factor* kemudian menambahnya dengan nilai offset. Persamaan proses *rescaling* dapat dilihat melalui persamaan berikut (Knudsen, 1999):

$$c = scaleFactor.\, c_0 + offset \tag{2.1}$$

Dengan *c* adalah sebuah citra hasil proses *rescaling* dan c_0 adalah citra sebelum dilakukan proses *rescaling*. Nilai *scaleFactor* dan *offset* merupakan nilai konstanta yang menentukan tingkat kecerahan suatu citra.

2.2.4 Ruang Warna RGB

Ruang warna *Red*, *Green*, *Blue* (RGB) adalah kombinasi warna primer dari merah, hijau dan biru yang biasa dipakai untuk komputer atau televisi. Warna yang dipakai masing-masing memiliki nilai 8 bit warna merah, 8 bit warna hijau, dan 8 bit warna biru. Campuran ketiga warna warna tersebut dengan ukuran seibang menghasilkan nuansa warna kelabu. Jika ketika warna disaturasikan penuh, akan menghasilkan warna putih (Rulaningtyas, et al., 2015).

2.2.5 Ruang Warna XYZ

Ruang warna XYZ yang juga dikenal dengan ruang warna CIE XYZ didapatkan dari transformasi ruang warna RGB memalui proses tranformasi matriks 3x3. Transformasi melibatkan nilai-nilai tristimulus, yaitu suatu konfigurasi dari tiga komponen warna cahaya linier yang memenuhi pencocokan warna CIE. Pada

ruang warna XYZ, beberapa warna direpresentasikan sebagai nilai yang selalu positif. Perhitungan untuk transformasi dari ruang warna RGB ke XYZ, dapat dilakukan melalui perhitungan matriks tranformasi dengan persamaan berikut (Rulaningtyas, et al., 2015):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.2)$$

2.2.6 Ruang Warna L*a*b*

Ruang Warna L*a*b* yang juga dikenal dengan nama ruang nama CIELAB adalah ruang warna yang paling lengkap yang ditetapkan oleh Komisi Internasional tentang iluminasi warna (French Commision Internationale de l'eclairage, dikenal juga dengan sebutan CIE). Ruang warna L*a*b* mampu menggambarkan semua warna yang dapat dilihat oleh mata manusia dan seringkali digunakan sebagai referensi ruang warna (Rulaningtyas, et al., 2015). Perhitungan konversi ruang warna L*a*B* dapat dilakukan melalui ruang warna XYZ melalui persamaan berikut ini (Pratt, 2007):

$$\mathrm{L}^* = 116\left(\frac{Y}{Y_0}\right)^{1/3} - 16 \qquad \text{for } \frac{Y}{Y_0} > 0.008856$$

$$\mathrm{L}^* = 903.3\,\frac{Y}{Y_0} \qquad \text{for } 0.0 \le \frac{Y}{Y_0} \le 0.008856$$

$$\mathrm{a}^* = 500\left[f\left\{\frac{X}{X_0}\right\} - f\left\{\frac{Y}{Y_0}\right\}\right]$$

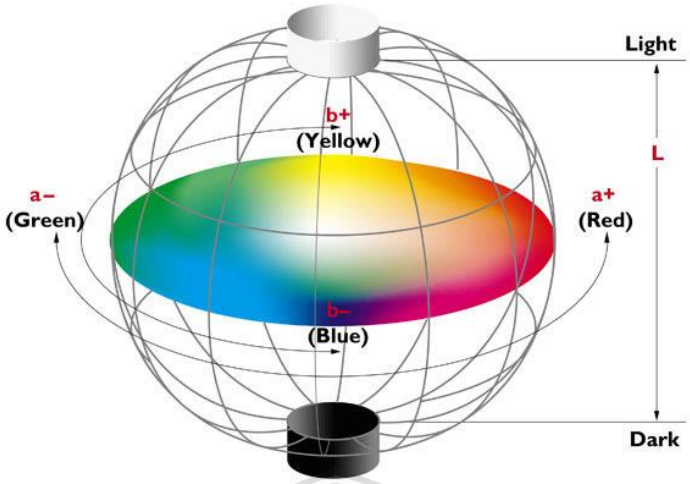
$$\mathrm{b}^* = 200\left[f\left\{\frac{Y}{Y_0}\right\} - f\left\{\frac{Z}{Z_0}\right\}\right]$$

dimana,

$$f(w) = w^{1/3} \qquad \text{for } w > 0.008856$$

$$f(w) = 7.787(w) + 0.1379 \qquad \text{for } 0.0 \le \boldsymbol{w} \le 0.008856 \quad (2.3)$$

Ruang warna L*a*b* merupakan singkatan dari *Luminance* (kecerahan) serta A dan B yang merupakan komponen berwarna. Menurut model L*a*b* A di representasikan antara hijau ke merah dan B antara biru ke kuning. Model L*a*b* dirancang untuk menjadi model yang independen atau dengan kata lain dapat menangani warna terlepas dari perangkat tertensu seperti printer, monitor, atau komputer. *Luminance* berkisar dari 0 hingga 100, komponen A berkisar dari -120 hingga +120 (hijau ke merah), dan komponen B berkisar dari -120 hingga +120 (biru ke kuning). Sistem koordinat ruang warna L*a*b* ditunjukkan pada Gambar 2.5.

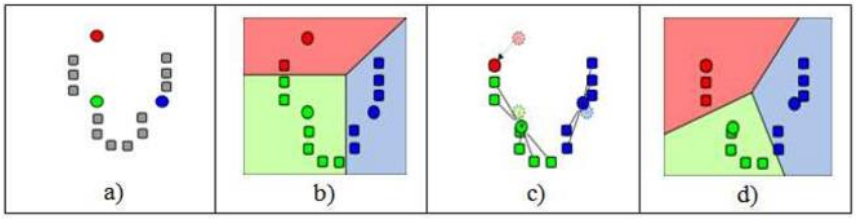


Gambar 2.5 Model Warna L*a*b* (Script Tutorials, 2014).

Ruang warna L*a*b* dirancang untuk penglihatan perkiraan manusia yang bertujuan untuk menyamakan persepsi dan komponen L yang sangat cocok untuk persepsi manusia. Hal ini digunakan untuk melakukan koreksi keseimbangan warna yang akurat dengan cara memodifikasi kurva *output* dalam komponen A dan B, atau untuk menyesuaikan kontras ringan menggunakan komponen L (Hapsari & Hidayattullah, 2013).

2.2.7 *K-Means*

K-Means merupakan salah satu metode data *clustering non-hierarchical* (non hirarki) yang bertujuan untuk mengelompokkan data yang ada ke dalam kelompok/*cluster*. Karena merupakan contoh dari konsep *data clustering* maka *K-Means* bersifat tanpa arahan (*unsupervised*). Metode ini mempartisi data ke dalam kelompok/*cluster* sehingga data yang memiliki ciri atau karakteristik yang sama dijadikan dalam satu kelompok/*cluster* yang sama sedangkan data yang memiliki ciri atau karakteristik yang berbeda dijadikan satu kelompok/*cluster* data yang lain. Tujuan dari data *clustering* adalah meminimalkan *objective function* yang diset dalam proses *clustering*, yang pada umumnya berusaha untuk meminimalisasikan variasi di dalam suatu *cluster* dan memaksimalisasikan variasi antar *cluster* (Agusta, 2007). Ilustrasi algoritme *K-means* ditunjukkan melalui Gambar 2.6.



Gambar 2.6 Ilustrasi algoritme *K-Means* (Umran & Abidin, 2009).

K-Means memilih secara acak sebanyak k *cluster* sebagai *centroid*. Kemudian menempatkan data dalam *cluster* tertekat, dihitung dari titik tengah *cluster* (*centroid*). *Centroid* baru ditetapkan bila semua data telah ditempatkan dalam *cluster* terdekat. Proses penentuan *centroid* dan penempatan data diulangi hingga nilai *centroid* kovergen (Umran & Abidin, 2009).

2.2.7.1 Konsep *K-Means*

Pengelompokan suatu data menggunakan metode *K-Means* secara umum dapat dilakukan dengan algoritme dasar sebagai berikut (Agusta, 2007):

1. Menentukan jumlah *cluster* awal
2. Menempatkan *centroid* sesuai dengan jumlah *cluster* secara *random*
3. Mengalokasikan data kedalam *cluster* sesuai dengan *centroid* menggunakan perhitungan jarak yang terdekat
4. Menghitung rata-rata dari masing-masing data *cluster* untuk menghasilkan *centroid* baru
5. Mengalokasikan data kedalam *cluster* sesuai dengan *centroid* baru menggunakan perhitungan jarak yang terdekat
6. Kembali ke *step* 4, apabila masih ada data yang berpidah *cluster* atau terjadi perubahan nilai *centroid* diatas nilai *threshold* yang ditentukan atau apabila perubahan nilai *objective function* yang digunakan diatas nilai *threshold*

2.2.7.2 Segementasi Menggunakan *K-Means*

K-Means juga dapat digunakan untuk melakukan segmentasi terhadap citra dimana dengan otomatis mengelompokkan sebanyak k *cluster* dari *pixel* citra sesuai dengan ciri atau karakteristik masing-masing. Pada proses segmentasi menggunakan *K-Means*, langkah pertama adalah menentukan jumlah *cluster* pada citra hasil *pre-processing* dan menghitung nilai *centroid* secara acak. Selanjutnya menghitung jarak *pixel* ke *centroid* dan mengelompokkan *pixel* berdasarkan jarak terdekat. Setelah nilai *pixel* gambar berkelompok sesuai jarak terdekatnya, dihitung kebali *centroid* sebagai *centroid* baru melalui perhitungan rata-rata *pixel* tiap *cluster* sebagai *centroid* baru dan dikelompokkan kembali *pixel* sesuai dengan *centroid* tersebut. Apabila masih terdapat *pixel* yang berpindah *cluster* maka dilakukan perhitungan *centroid* baru, tetapi apabila jika sudah tidak ada nilai *pixel* yang berpindah *cluster* maka proses *clustering* berakhir (Mardhiyah & Harjoko, 2011).

2.2.8 *K-Nearest Neighbour* (K-NN)

K-Nearest Neighbor (K-NN) merupakan salah satu algoritme *supervised learning* dimana *output* dari suatu data baru diklasifikasikan berdasarkan kelompok mayoritas dari k buah tetangga terdekat. Tujuan utama dari algoritme ini adalah mengelompokkan data baru berdasarkan atribut dan data *training*.

Algoritme K-NN sangatlah sederhana, bekerja berdasarkan jarak terpendek dari obyek *query* ke *training sample* untuk menentukan jumlah k tetangganya. Setelah mengumpulkan titik k tetangga, kemudian diambil mayoritas dari k

tetangga untuk dijadikan prediksi dari obyek *query*. Untuk mendapatkan nilai k yang optimal dapat digunakan optimasi parameter misal menggunakan *k-fold cross validation*. Algoritme K-NN bertujuan untuk melakukan klasifikasi terhadap obyek berdasarkan atribut dan *training sample*. *Clasifier* tidak menggunakan apapun untuk dicocokkan dan hanya berdasarkan pada memori (Syafitri, 2010).

2.2.8.1 Konsep *K-Nearest Neighbour* (K-NN)

Konsep dasar metode K-NN adalah dengan mencari jarak terdekat antara data uji dan sejumlah k tetangga terdekat dalam data latih. Berikut langkah-langkah algoritme K-NN (Priandana, Zulfikar, & Sukarman, 2016):

1. Menentukan nilai k.
2. Menghitung jarak antara data uji dengan setiap data latih.
3. Mendapatkan k data yang memiliki jarak terdekat.
4. Jumlah k data yang memiliki jarak terdekat adalah kelas yang paling banyak muncul.
5. Nilai yang paling banyak muncul ditentukan sebagai kelas dari data uji.

2.2.9 *Euclidean Distance*

Euclidean Distance digunakan untuk proses klasifikasi atau identifikasi dengan menghitung jarak antara suatu vektor *training* dan vektor *testing* pada basis data yang ada. Perhitungan *Euclidean Distance* dapat dihitung melalui persamaan berikut ini (Wibowo & Usman, 2010):

$$d = \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^2} \quad (2.4)$$

Penjelasan Persamaan 2.4 :

d = *nilai jarak*
n = *banyaknya data*
P_i = *data ke i dari testing*
Q_i = *data ke i dari training*

2.2.10 *Silhouette Coefficient*

Silhouette Coefficient adalah suatu metode yang digunakan untuk melihat kualitas serta kekuatan *cluster* dan sebera baik suatu objek ditempatkan pada suatu *cluster*. Metode ini merupakan gabungan dari metode separation dan cohesion (Anggara, Sujiani, & Nasution, 2016). hasil yang mendekati angka 1 memiliki derajat kepemilikan anggota pada *cluster* yang tinggi atau dengan kata lain kualitas *cluster* baik. Sebaliknya, jika nilai mendekati angka -1 memiliki derajat kepemilikan anggota *cluster* yang salah atau dengan kata lain kualitas *cluster* tidak

baik. Berikut rumus perhitungan dari nilai *Silhouette Coefficient* (Asana, et al., 2017):

$$s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \quad (2.5)$$

Penjelasan Persamaan 2.5 :

a(i) = nilai rata-rata jarak antara titik S_i dengan titik lain pada cluster yang sama

b(i) = nilai rata-rata jarak antara titik S_i dengan titik lain pada seluruh cluster yang berbeda

2.2.11 Akurasi

Perhitungan akurasi dilakukan untuk menguji tingkat keakuratan pada metode yang diteliti untuk membandingkan hasil pemrosesan dengan target keluaran sesungguhnya. Akurasi direpresentasikan menggunakan angka 0 hingga 100 dalam bentuk persen (%). Perhitungan akurasi didapatkan melalui persamaan berikut ini:

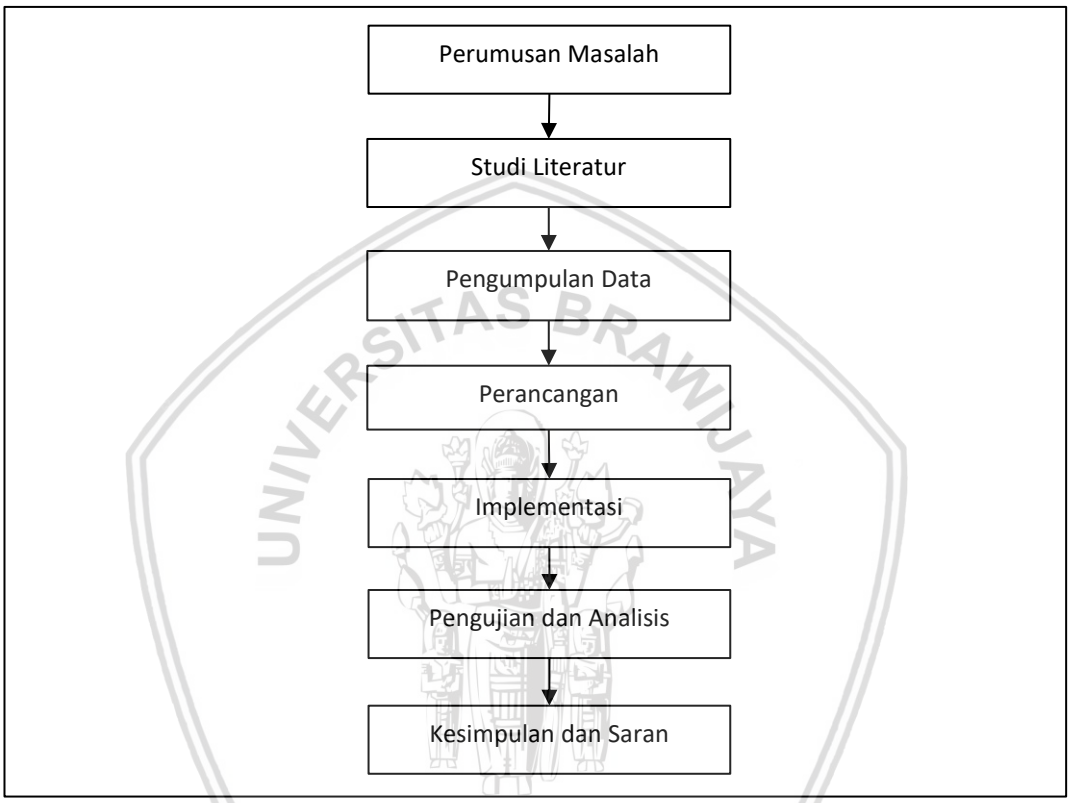
$$Akurasi = \frac{Jumlah\ keluaran\ data\ uji\ bernilai\ benar}{Jumlah\ seluruh\ data\ uji}\ x\ 100\% \quad (2.6)$$

Sesuai dengan persamaan 2.6, perhitungan akurasi didapatkan dengan membagi data uji hasil perhitungan metode yang bernilai benar dengan jumlah seluruh data uji dan dikalikan dengan 100%.

BAB 3 METODOLOGI

3.1 Metodologi

Metodologi penelitian menjelaskan tahapan yang dilakukan peneliti dalam menggunakan metode segmentasi menggunakan algoritme *K-Means* untuk proses identifikasi penyakit pada daun jeruk. Tahapan metodologi penelitian dapat dilihat pada Gambar 3.1 berikut ini:



Gambar 3.1 Alur Metodologi Penelitian

3.2 Perumusan Masalah

Tahapan ini digunakan untuk menentukan objek permasalahan yang diselesaikan. Informasi masalah yang muncul berasal dari Balai Penelitian Tanaman Jeruk dan Buah Subtropika (Balitjestro). Penggalian informasi masalah dengan melakukan observasi dan wawancara secara langsung kepada *stakeholder*. Setelah melakukan observasi dan wawancara secara langsung, kemudian melakukan analisis terhadap masalah untuk ditentukan metode penyelesaian dari masalah yang ada.

3.3 Studi Literatur

Tahapan ini digunakan untuk mendapatkan informasi tambahan yang digunakan sebagai acuan. Cara ini dilakukan dengan mempelajari literatur dari beberapa bidang ilmu yang berhubungan dengan rekomendasi dan alur kerja program meliputi :

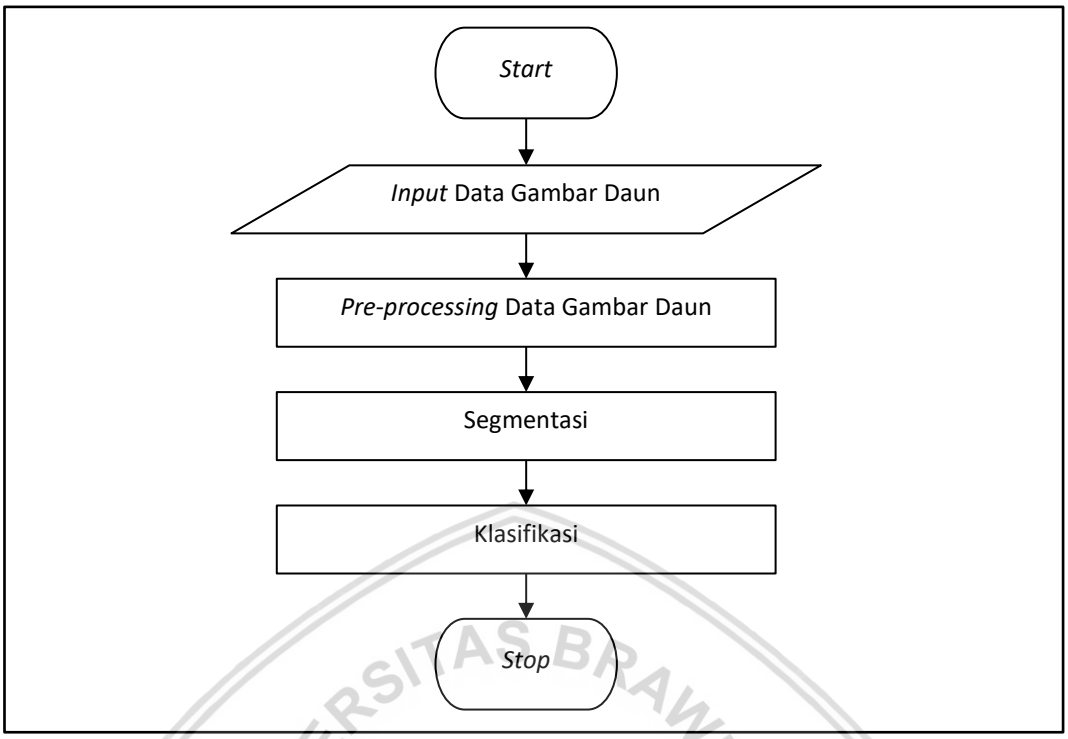
1. Penyakit daun jeruk antara lain CVPD (*Citrus Vein Phloem Degeneration*), Cendawan Jelaga, dan *Downy Mildew*
2. Pengolahan citra digital
3. Konsep algoritme *K-Means* untuk melakukan segmentasi citra
4. Ruang warna antara lain RGB, XYZ dan L*a*b*
5. Konsep algoritme *K-Nearest Neighbor* (K-NN) untuk proses identifikasi penyakit daun jeruk
6. Pemograman Java

3.4 Pengumpulan Data

Pengumpulan data untuk penelitian ini diambil pada Balai Penelitian Tanaman Jeruk dan Buah Subtropika (Balitjestro) yang berada di daerah Tlekung kota Batu. Sebanyak 136 data citra daun dikumpulkan dan dibagi menjadi 120 data uji dan 16 data latih. 120 data uji, terdiri dari masing-masing 30 jenis daun yaitu daun dengan kategori sehat, daun berpenyakit CVPD, daun berpenyakit Cendawan Jelaga, dan daun berpenyakit Downy Mildew. Kemudian, 16 data latih juga dibagi menjadi 4 untuk masing-masing kategori (Daun Sehat, CVPD, Cendawan Jelaga, Downy Mildew). Cara pengambilan gambar dengan memetik daun dan diletakkan di alat bantu *cover* berwarna merah sebagi penjepit daun. Setelah itu pengambilan gambar pada daun menggunakan kamera *smartphone Asus Zenfone Selfie* ZD551KL 13 MP dengan jarak 20 cm melalui pencahayaan yang merata. Setelah dilakukan pengambilan gambar terhadap daun lalu dilakukan *croping* sesuai dengan ukuran *background* alat bantu.

3.5 Perancangan

Perancangan diperlukan dalam penelitan ini untuk melakukan segmentasi citra menggunakan algoritme *K-Means* dan proses identifikasi terhadap penyakit pada daun jeruk agar. Tahap pertama yaitu dengan melakukan *input* data gambar daun. Tahap kedua melakukan *pre-processing* data gambar daun menggunakan metode-metode pengolahan citra digital. Tahap ketiga, melakukan segmentasi penyakit daun jeruk menggunakan algoritme *K-Means*. Dan tahap keempat melakukan klasifikasi menggunakan algoritme K-NN. Berikut diagram alur yang dapat dilihat pada Gambar 3.2 :



Gambar 3.2 *Flow Chart* Perancangan Program

3.6 Implementasi

Tahap implementasi adalah tahap pembangunan program dengan memperhatikan proses-proses perancangan yang telah dijabarkan sebelumnya. Pembangunan program menggunakan bahasa pemograman Java.

3.7 Pengujian dan Analisis

Tahap pengujian yang pertama dilakukan dengan memperhatikan nilai *Scale Factor* pada proses *rescaling*. Untuk menentukan nilai *Scale Factor* terbaik, dilakukan dengan mencari hasil akurasi terbaik dalam beberapa percobaan nilai *Scale Factor*.

Kedua, menggunakan metode *Silhouette Coefficient* untuk menentukan nilai terbaik dari jumlah *cluster* pada segmentasi menggunakan *K-Means*.

Ketiga, Melakukan pengujian untuk mendapatkan nilai K optimal dalam pemrosesan dengan menggunakan algoritme K-NN. Untuk mendapatkan nilai K optimal dilakukan pengujian dengan metode mencari hasil akurasi terbaik dalam percobaan beberapa nilai K.

Yang terakhir melakukan analisis pada nilai akurasi program dari hasil identifikasi penyakit menggunakan algoritme K-NN yang dihasilkan melalui proses segmentasi citra menggunakan algoritme *K-Means*.

3.8 Kesimpulan dan Saran

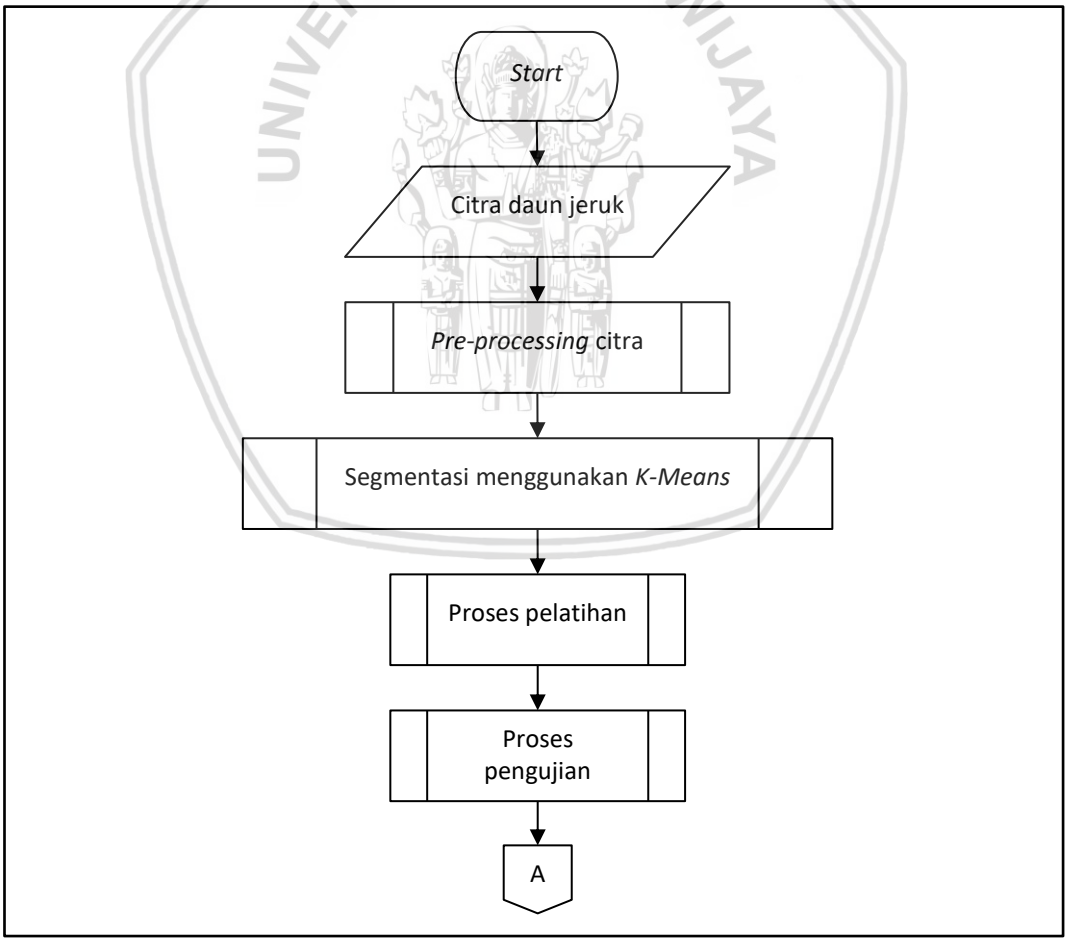
Kesimpulan diambil setelah semua tahapan selesai dilakukan dari tahapan perancangan dan hasil pengujian. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan yang terjadi dan memberikan pertimbangan untuk penelitian selanjutnya.

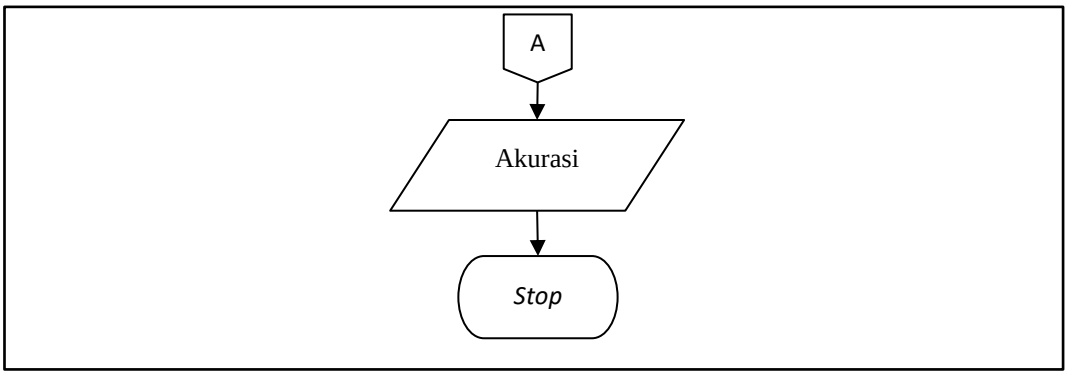
BAB 4 PERANCANGAN

Pada bab ini berisikan perancangan program yang diimplementasikan untuk segmentasi menggunakan algoritme *K-Means* pada citra daun jeruk sebagai proses Identifikasi penyakit tanaman jeruk. Pada bab perancangan ini terdiri atas perancangan algoritme, manualisasi, perancangan pengujian, dan perancangan antarmuka program.

4.1 Perancangan Algoritme

Pada sub bab perancangan algoritme perancangan program secara garis besar yang dijelaskan menggunakan *flowchart* (diagram alir). *Flowchart* menggambarkan alur proses yang dikerjakan oleh program. Tujuan dari pembuatan *flowchart* ini untuk mempermudah dalam melakukan implementasi terhadap program dan memahami lebih detail alur serta proses yang dibuat. Dalam program dilakukan beberapa proses diantaranya adalah *input* citra daun jeruk, *pre-processing* citra, segmentasi menggunakan *K-Means*, proses pelatihan, proses pengujian, dan perhitungan akurasi.



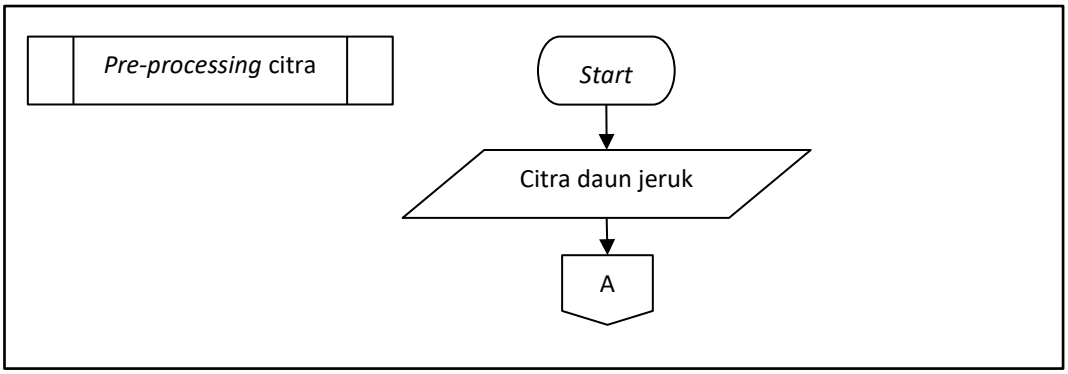


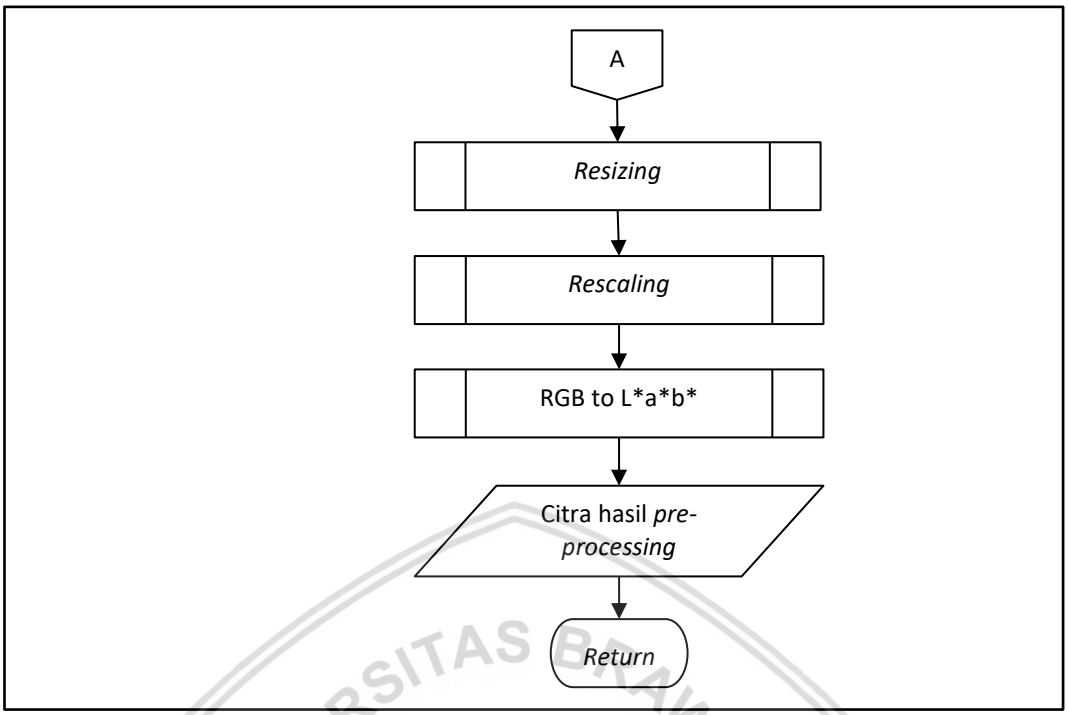
Gambar 4.1 *Flowchart* Perancangan Algoritme

Pada Gambar 4.1 menunjukkan alur-alur proses dimana program menerima *input* citra daun jeruk sebelum melakukan *pre-processing*. Tahapan *pre-processing* dengan melakukan *resizing* dan *rescaling* pada citra. Setelah itu program melakukan segmentasi menggunakan *K-Means*. Segmentasi terdapat dua tahapan yaitu segmentasi daun yang berfungsi memisah antara *cover* dengan daun dan segmentasi penyakit yang berfungsi melakukan segmentasi terhadap daun untuk ditemukan bagian berpenyakit. Selanjutnya adalah tahapan pelatihan dari hasil segmentasi penyakit data *centroid* dimasukkan pada tabel dan dilakukan pelabelan sesuai dengan jenis penyakit daun jeruk yang ada. Setelah proses pelatihan dilakukan proses pengujian untuk klasifikasi data uji terhadap data latih guna diketahui kelas penyakitnya.

4.1.1 *Pre-processing* Citra

Pada bagian *pre-processing* citra dilakukan beberapa tahapan. Proses yang pertama adalah proses *resizing* yang berfungsi untuk menyamakan ukuran citra *input* agar ukurannya seragam. Kemudian, dilakukan proses *rescaling* untuk melakukan pengaturan kecerahan pada citra *input*. Proses *pre-processing* yang terakhir adalah melakukan perubahan terhadap ruang warna RGB menjadi ruang warna L*a*b* yang dalam prosesnya ruang warna RGB diubah terlebih dahulu menjadi ruang warna XYZ kemudian diproses menjadi ruang warna L*a*b*. Selanjutnya dihasilkan citra *output* yang digunakan sebagai masukan untuk proses selanjutnya. Tahap *pre-processing* citra ditunjukkan pada Gambar 4.2.

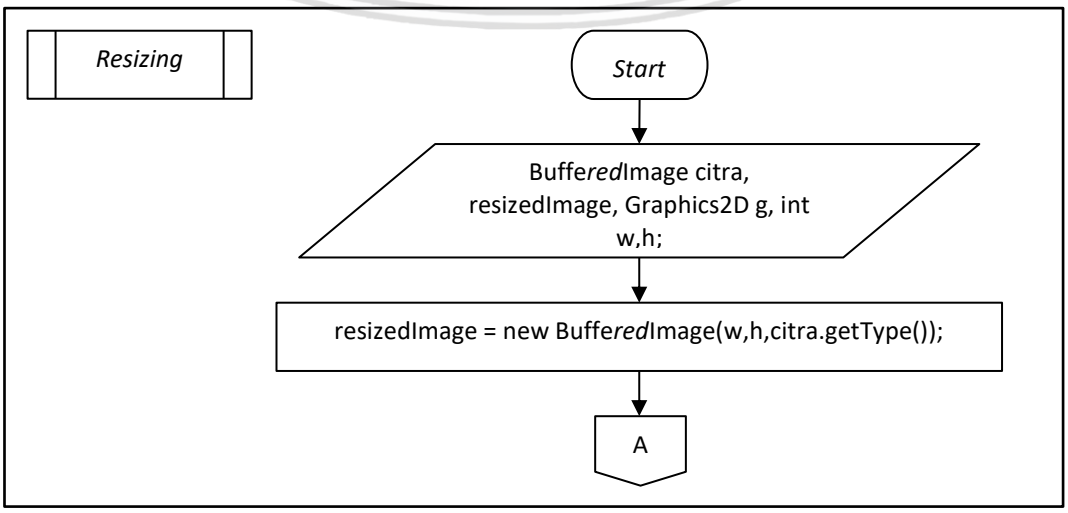


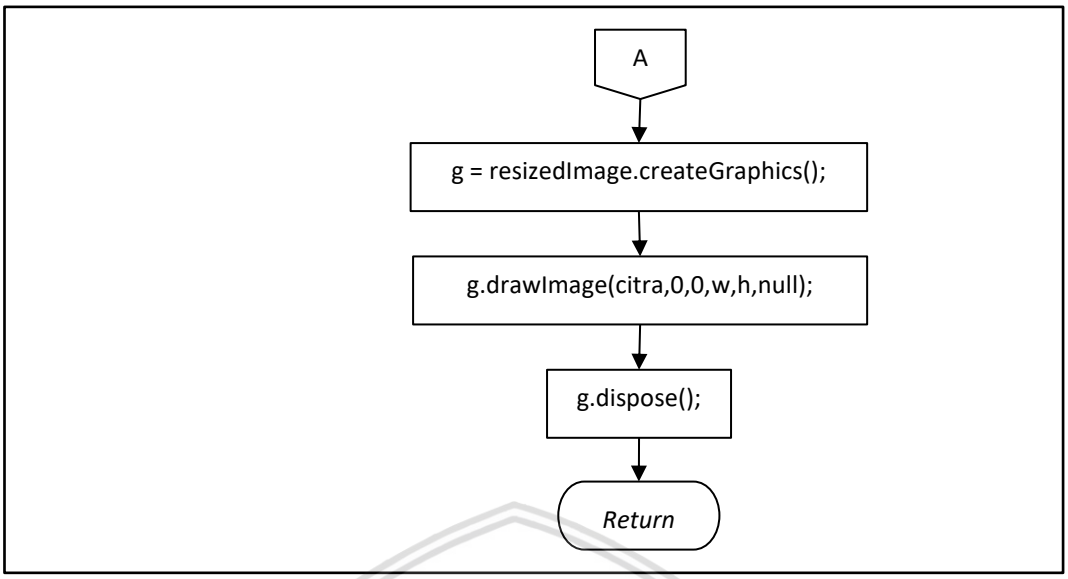


Gambar 4.2 *Flowchart Pre-processing* Citra

4.1.1.1 *Resizing*

Pada penelitian ini dilakukan proses *resizing* untuk menyeragamkan ukuran citra *input* serta mengoptimalkan proses dengan memperkecil citra sehingga beban terhadap proses perhitungan semakin kecil dan waktu eksekusi semakin cepat. Proses *resizing* dilakukan dengan menggunakan kelas dari *Java* yaitu *Graphics2D* dengan mengeksekusi *method drawImage*. Dalam *method drawImage* terdapat beberapa parameter diantaranya *BufferedImage* yang merupakan objek dari citra yang akan diproses, *index pixel* dimana citra akan dimulai untuk digambar, ukuran *width*, serta ukuran *height* sebagai parameter ukurang lebar dan tinggi citra yang baru. Proses *resizing* dijabarkan melalui *flowchart* pada Gambar 4.3.

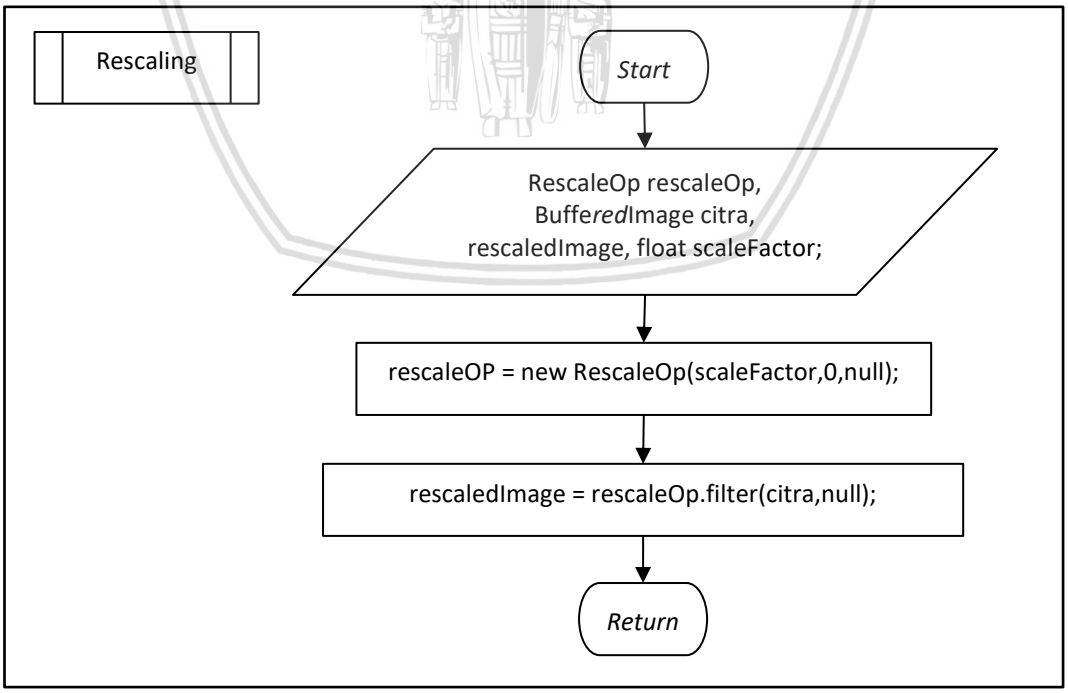




Gambar 4.3 *Flowchart Resizing*

4.1.1.2 *Rescaling*

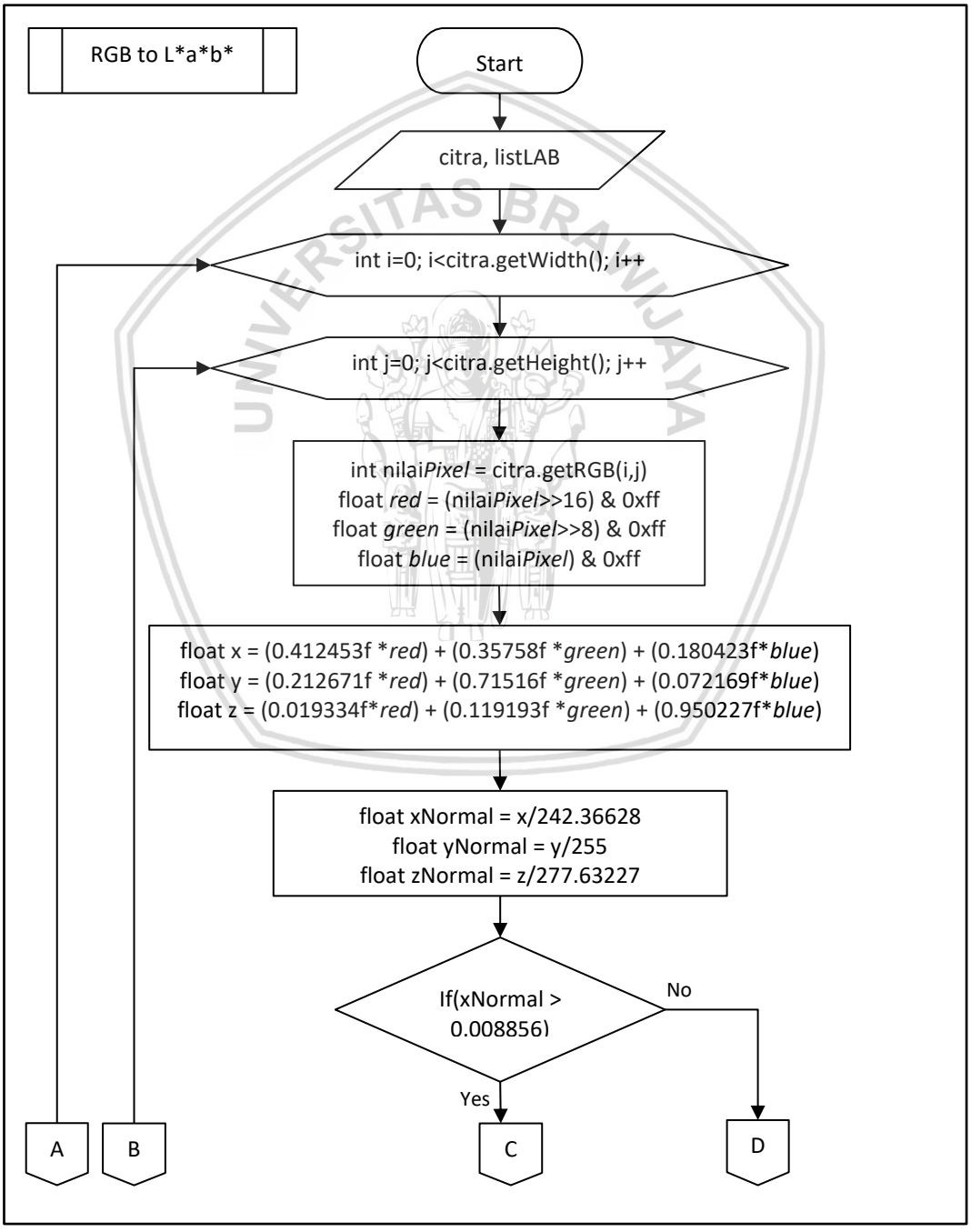
Proses *rescaling* adalah suatu proses untuk mengatur tingkat kecerahan dari suatu citra *input* dengan dipengaruhi parameter nilai *scaleFactor* dan nilai *offset*. Pada penelitian ini proses *rescaling* dilakukan dengan menggunakan kelas pada *Java* yaitu *RescaleOp* yang menggunakan dua parameter yaitu nilai *scaleFactor* dan nilai *offset*. Pemanggilan *method filter* pada kelas *RescaleOp* digunakan untuk mengimplementasikan *rescaling* menggunakan parameter objek operator *rescale* yang telah diinisialisasi. Detail proses *rescaling* dapat dilihat melalui Gambar 4.4.

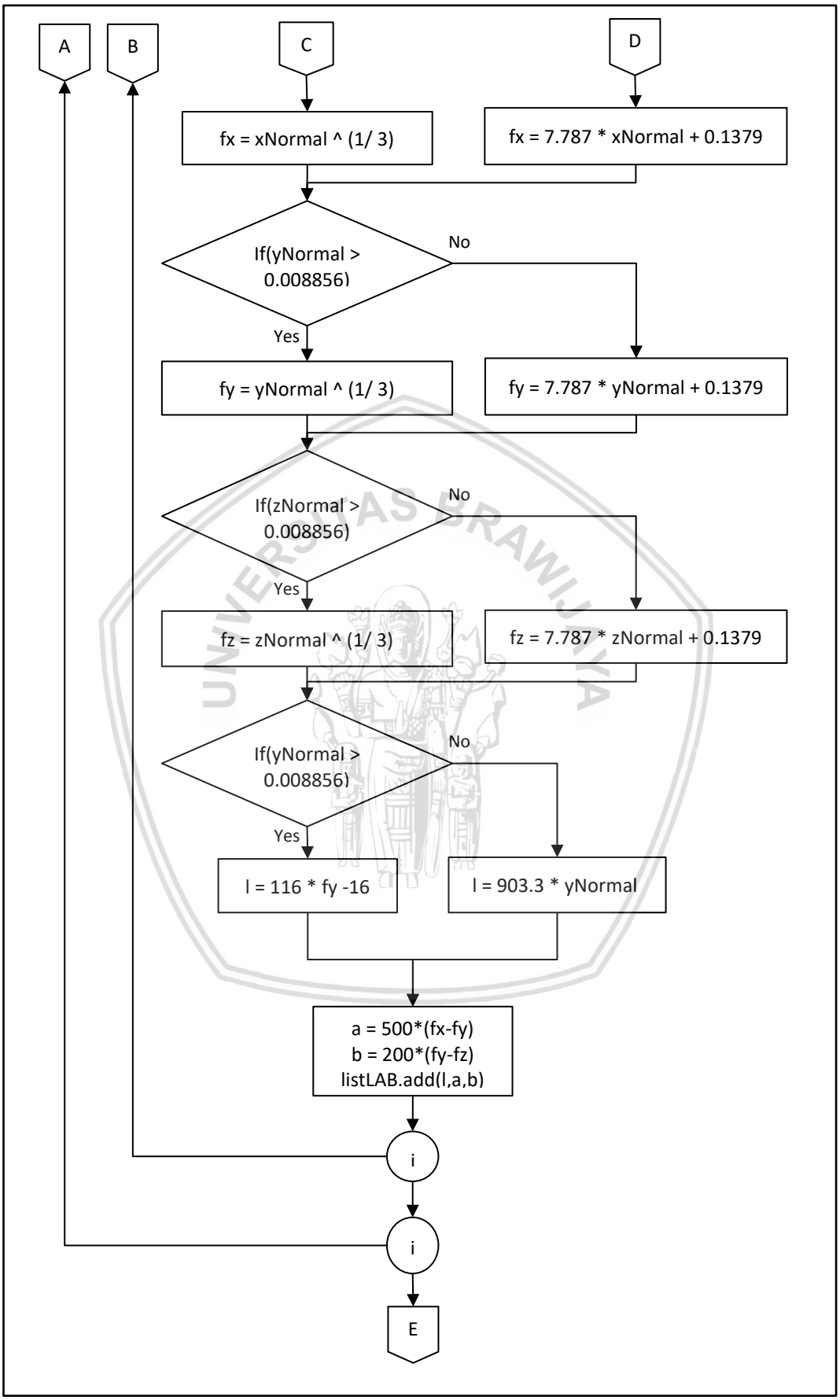


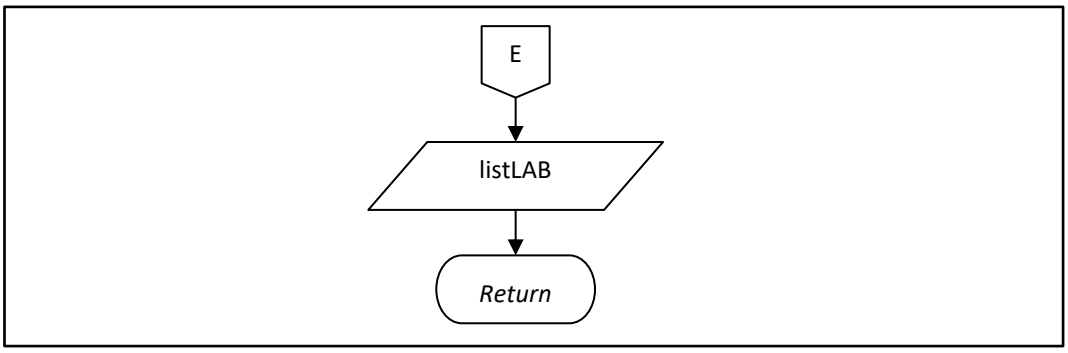
Gambar 4.4 *Flowchart Rescaling*

4.1.1.3 RGB to L*a*b*

Proses perubahan ruang warna dari ruang warna RGB menjadi ruang warna L*a*b* melalui beberapa tahapan. Pertama-tama ruang warna RGB diubah menjadi ruang warna XYZ melalui transformasi *matrix* 3x3 sesuai dengan persamaan 2.2. Setelah mendapatkan nilai X, Y, Z kemudian dilakukan perubahan ruang warna ke L*a*b* sesuai dengan rumus pada persamaan 2.3 serta menggunakan white reference untuk X_0 = 242.36628, Y_0 = 255, dan Z_0 = 277.63227. Proses perubahan ruang warna ini dilakukan pada masing-masing *pixel* pada citra sehingga diperlukan suatu proses perulangan. Langkah proses perubahan RGB menjadi L*a*b* dapat ditunjukkan pada Gambar 4.5.



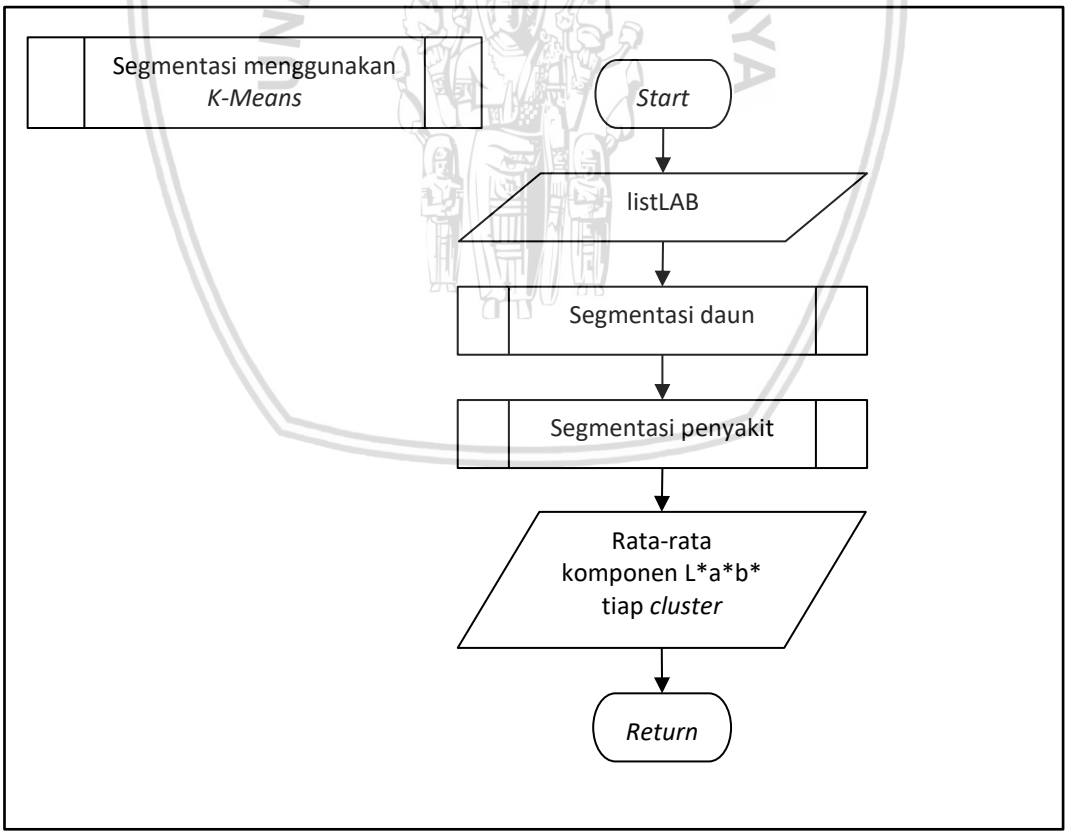

A
B
C
D
fx = xNormal ^ (1/ 3)
fx = 7.787 * xNormal + 0.1379
If(yNormal > 0.008856)
No
Yes
fy = yNormal ^ (1/ 3)
fy = 7.787 * yNormal + 0.1379
If(zNormal > 0.008856)
No
Yes
fz = zNormal ^ (1/ 3)
fz = 7.787 * zNormal + 0.1379
If(yNormal > 0.008856)
No
Yes
l = 116 * fy -16
l = 903.3 * yNormal
a = 500*(fx-fy)
b = 200*(fy-fz)
listLAB.add(l,a,b)
i
i
E



Gambar 4.5 *Flowchart* RGB *to* L*a*b*

4.1.2 Segmentasi Menggunakan *K-Means*

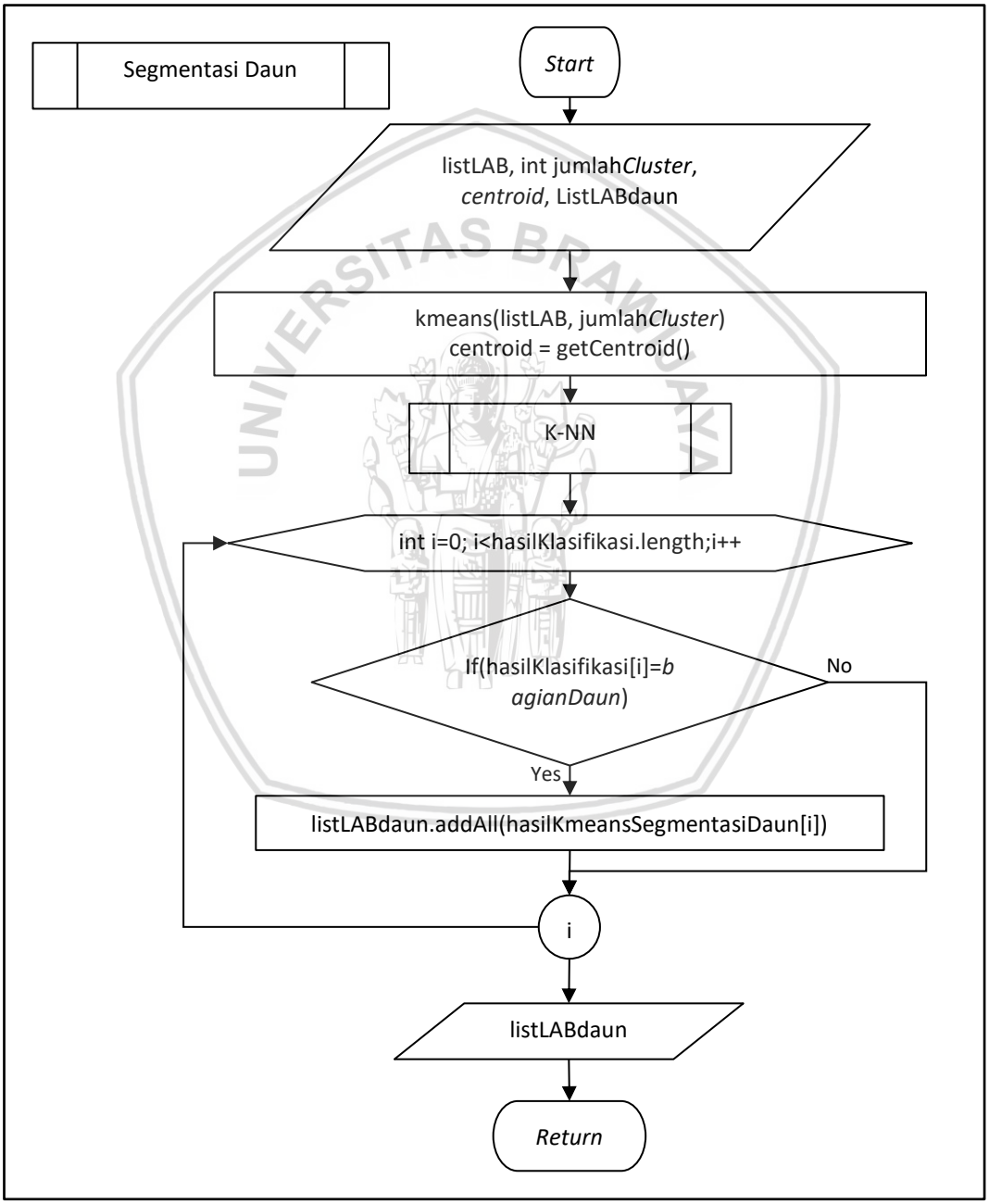
Penelitian ini memiliki fokus pada penggunaan algoritme *K-Means* untuk segmentasi citra. Pada tahapan segmentasi menggunakan *K-Means* ini, citra *input* yang sudah dilakukan *pre-processing* dan menghasilkan *list* ruang warna L*a*b* digunakan untuk data proses pada *K-Means*. Terdapat dua tahapan segmentasi yang pertama adalah segmentasi daun yang berfungsi untuk memisahkan bagian daun dengan *cover*. Yang kedua adalah segmentasi penyakit yang berfungsi untuk memisahkan antara daerah yang berpenyakit dengan bagian daun. Langkah proses segmentasi menggunakan *K-Means* dapat dilihat melalui Gambar 4.6.



Gambar 4.6 *Flowchart* Segmentasi Menggunakan *K-Means*

4.1.2.1 Segmentasi Daun

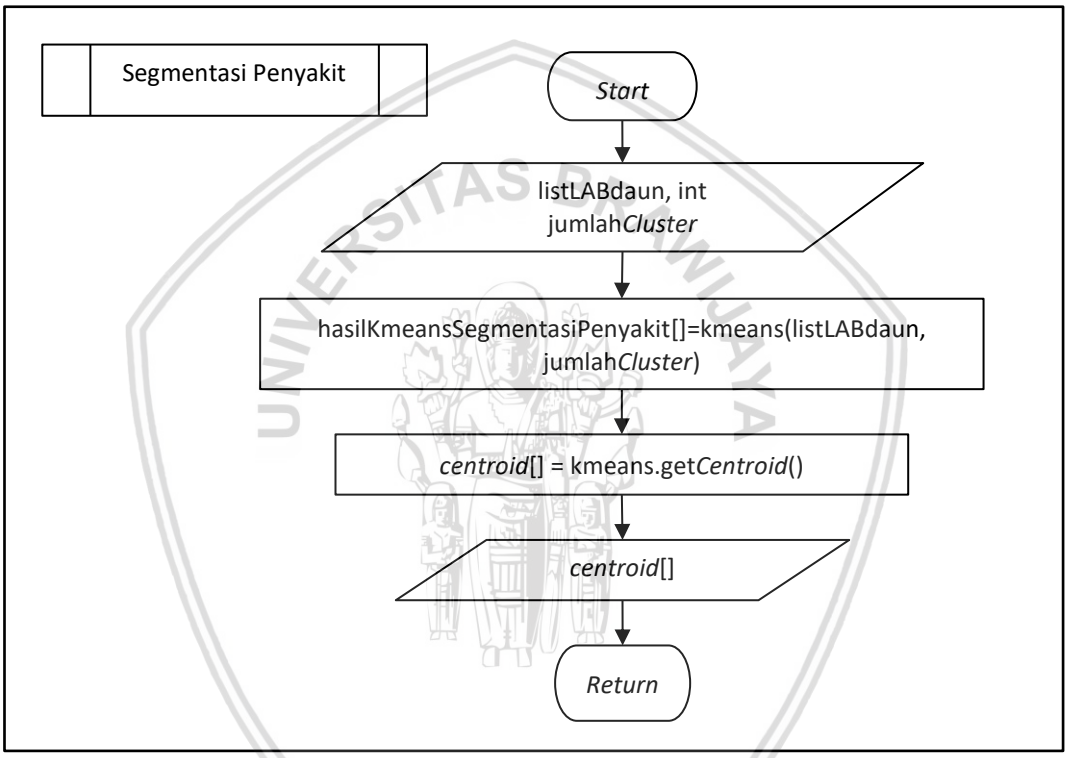
Proses segmentasi bagian daun menggunakan *input* nilai ruang warna L*a*b* dalam bentuk *list* dan jumlah *cluster* untuk acuan pada *K-Means*. Pada prosesnya melibatkan dua pemanggilan *method* yang pertama adalah *method* kmeans untuk menghasilkan perhitungan menggunakan algoritme *K-Means* yang ditampung melalui *array* berisi data ruang warna L*a*b*. *Method* kedua adalah K-NN yang digunakan untuk melakukan klasifikasi guna mendapatkan kelas daun sehingga dapat dipisahkan dengan kelas *cover*. Proses segmentasi daun dapat dilihat pada Gambar 4.7.



Gambar 4.7 *Flowchart* Segmentasi Daun

4.1.2.2 Segmentasi Penyakit

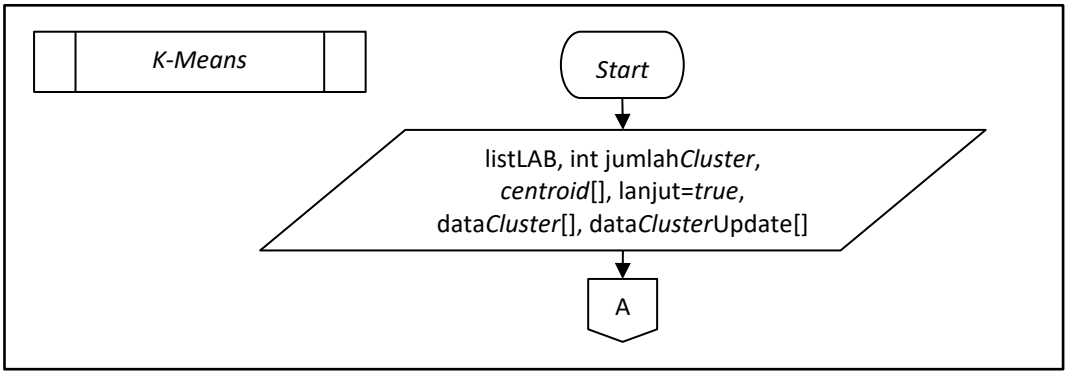
Segmentasi penyakit adalah suatu proses yang bertujuan untuk menghasilkan bagian area berpenyakit dari daun. *Input* dari proses ini yang pertama adalah nilai *listLABdaun* dari proses segmentasi daun sebelumnya. Yang kedua adalah jumlah *cluster* untuk pemrosesan menggunakan algoritme *K-Means*. Terdapat pemanggilan *method K-Means* untuk menghasilkan perhitungan menggunakan algoritme *K-Means* dan hasilnya ditampung melalui *array* berisi data ruang warna L*a*b*. Setelah melakukan proses perhitungan *K-Means*, *output* yang dihasilkan adalah *centroid* tiap *cluster* yang berisi nilai ruang warna L*a*b* yang nantinya sebagai acuan klasifikasi menggunakan algoritme K-NN. Detail dari proses segmentasi penyakit terdapat pada Gambar 4.8.

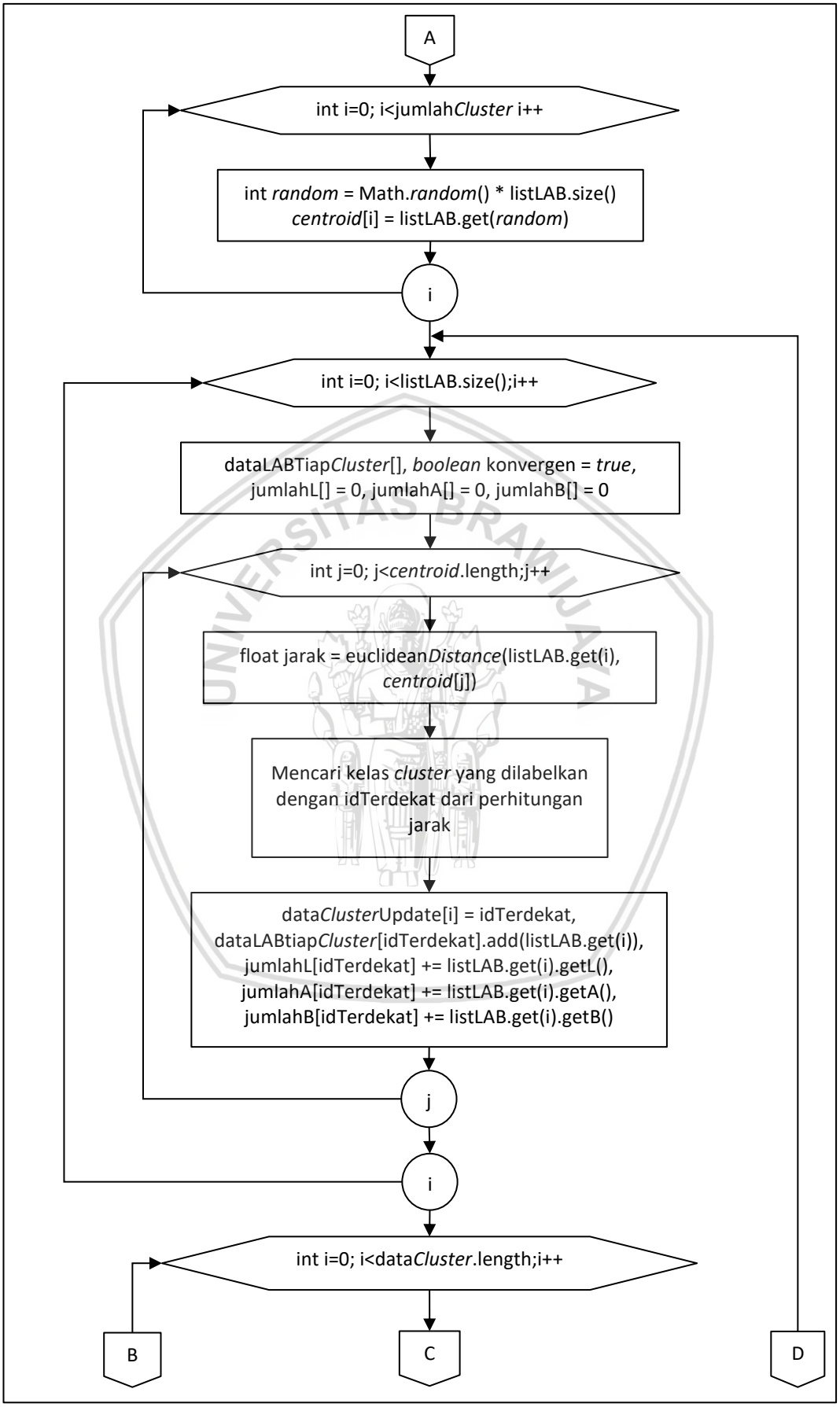


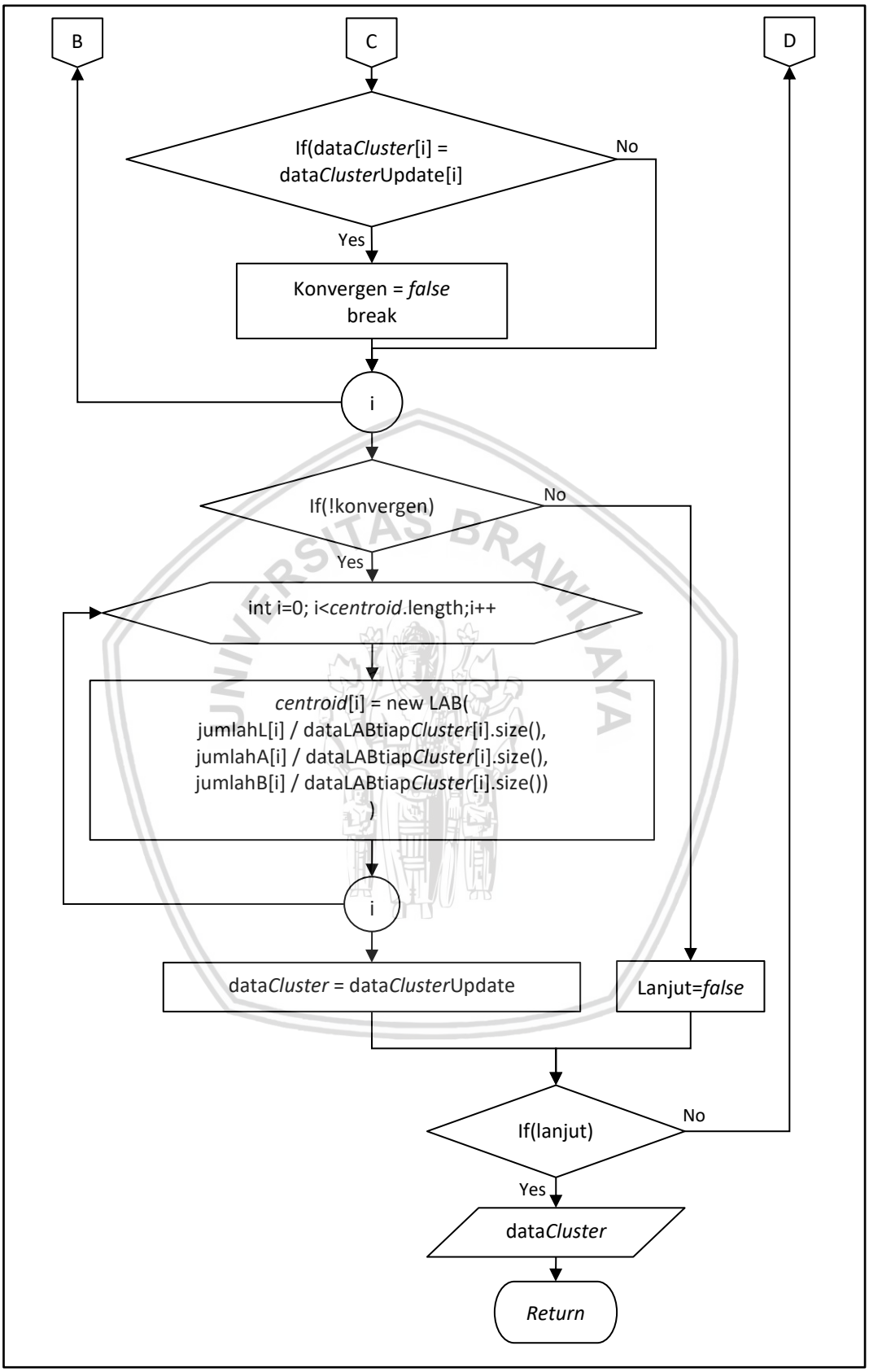
Gambar 4.8 *Flowchart* Segmentasi Penyakit

4.1.2.3 *K-Means*

Detail proses algoritme *K-Means* dapat dilihat pada Gambar 4.9.




A
int i=0; i<jumlah*Cluster* i++
int *random* = Math.*random*() * listLAB.size()
centroid[i] = listLAB.get(*random*)
i
int i=0; i<listLAB.size();i++
dataLABTiap*Cluster*[], *boolean* konvergen = *true*,
jumlahL[] = 0, jumlahA[] = 0, jumlahB[] = 0
int j=0; j<*centroid*.length;j++
float jarak = euclidean*Distance*(listLAB.get(i),
centroid[j])
Mencari kelas *cluster* yang dilabelkan
dengan idTerdekat dari perhitungan
jarak
data*Cluster*Update[i] = idTerdekat,
dataLABtiap*Cluster*[idTerdekat].add(listLAB.get(i)),
jumlahL[idTerdekat] += listLAB.get(i).getL(),
jumlahA[idTerdekat] += listLAB.get(i).getA(),
jumlahB[idTerdekat] += listLAB.get(i).getB()
j
i
int i=0; i<data*Cluster*.length;i++
B
C
D



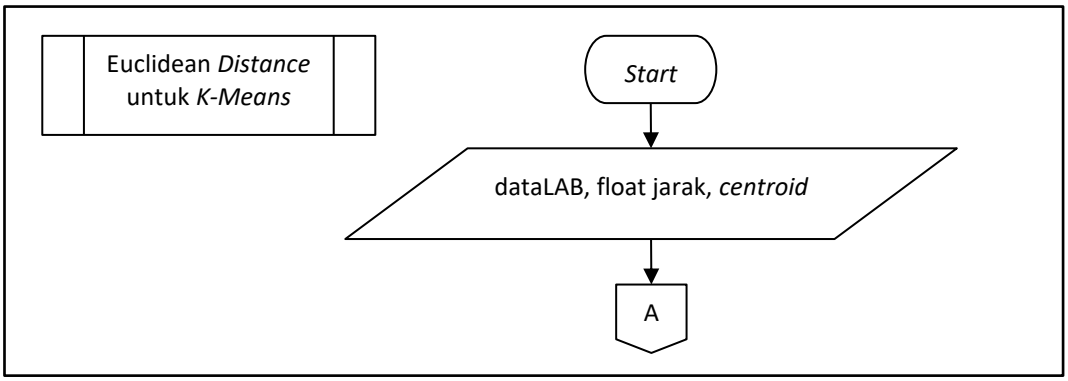
Gambar 4.9 *Flowchart K-Means*

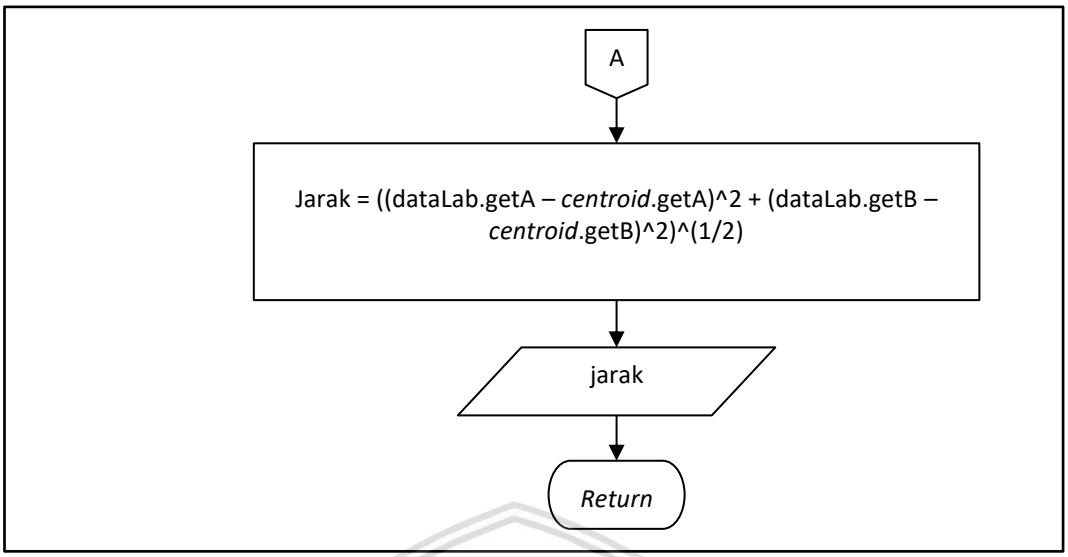
Pada penelitian ini, penggunaan algoritme *K-Means* untuk segmentasi citra menggunakan variabel a dan b pada ruang warna L*a*b* karena fokus pada tingkat kemurnian warna variabel a (representasi merah ke hijau) dan variabel b (representasi biru ke kuning) dan mengabaikan tingkat kecerahan (*luminance*). *Input* utama pada proses ini adalah *listLAB* yang berisi data ruang warna L*a*b* dan *jumlahCluster* untuk menentukan berapa *cluster* yang akan dihasilkan. Penentuan nilai dari *centroid* awal dilakukan dengan cara pengambilan data pada *listLAB* dengan *index list* yang di-*random* oleh program.

Ada beberapa bagian proses yang terdapat pada *flowchart K-Means* yang pertama adalah terdapat perhitungan jarak yang melakukan pemanggilan terhadap *method euclideanDistance*. *Method eucildeanDistance* dipanggil berfungsi untuk memberikan rekomendasi *cluster* pada masing-masing data dalam *list*. Yang kedua, terdapat suatu pengecekan terhadap data *cluster* yang lama dengan data *cluster* ter-*update*. Proses pengecekan tersebut bertujuan untuk memberikan kondisi terhadap variabel *boolean* konvergen apakah pada saat di eksekusi prosesnya tidak ada data yang berpindah ataukah terdapat data yang berpindah. Setiap data yang berpindah variabel konvergen di-*set* dengan nilai *false* dan jika tidak terjadi perpindahan data tetap bernilai *true*. Proses selanjutnya adalah melakukan update terhadap *centroid*. Jika variabel konvergen memiliki nilai *false* maka program menghitung *centroid* yang baru dengan rata-rata dari komponen data yang memiliki *cluster* sama. Pada bagian menghitung *centroid* terbaru unsur L* pada ruang warna L*a*b* juga disertakan untuk dihitung rata-rata tiap *cluster*nya yang berfungsi sebagai acuan proses klasifikasi pada tahapan berikutnya. Sebaliknya, jika konvergen bernilai *true* akan melakukan *set* variabel lanjut menjadi *false*. Kondisi yang terakhir adalah mengecek nilai dari variabel lanjut. Jika variabel lanjut sama dengan *true* program melanjutkan proses *looping* untuk iterasi berikutnya dan sebaliknya jika variabel lanjut bernilai *false* menyebabkan kondisi berhenti serta menghasilkan *output* data*Cluster* sebagai hasil akhir.

4.1.2.4 *Euclidean Distance* Untuk *K-Means*

Proses perhitungan jarak menggunakan *Euclidean distance* untuk algoritme *K-Means* menggunakan dua parameter dari ruang warna L*a*b* yaitu menggunakan nilai a* dan b*. Detail proses dari perhitungan jarak *Euclidean distance* dapat dilihat melalui Gambar 4.10.



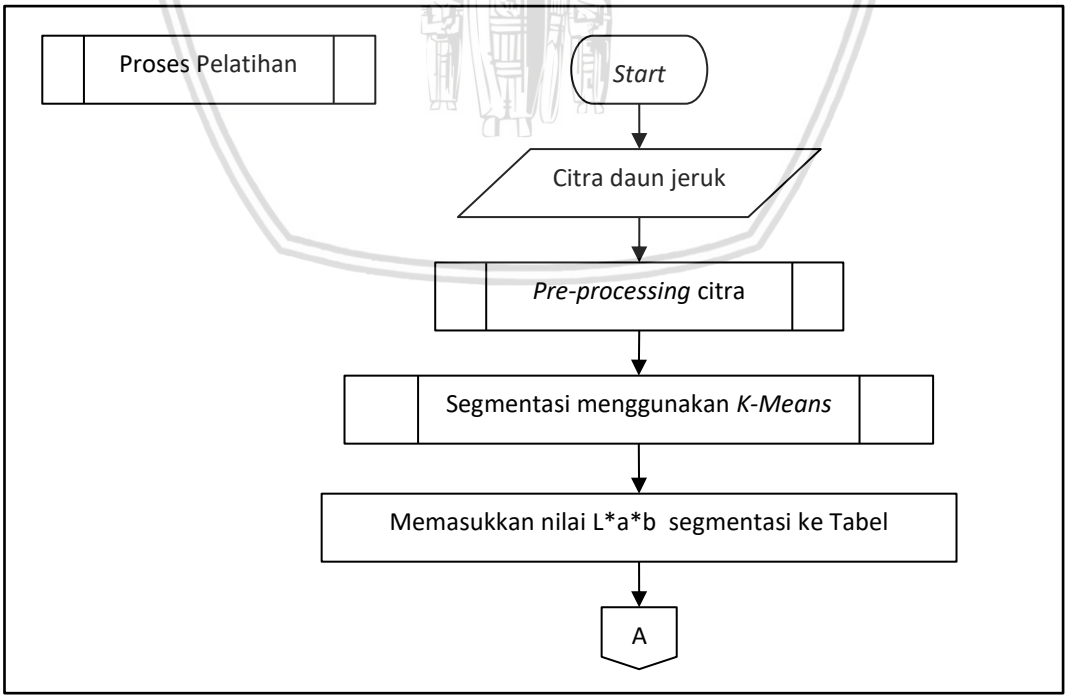


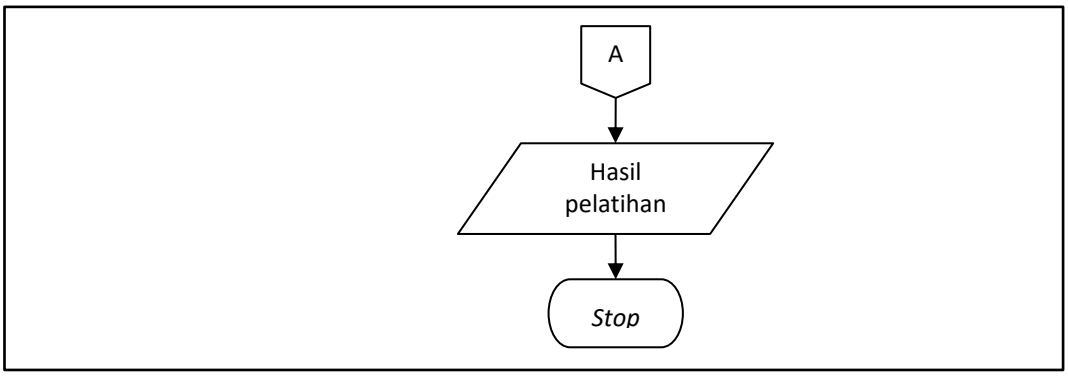
Gambar 4.10 *Flowchart Euclidean Distance* Untuk *K-Means*

Perhitungan jarak yang dilakukan menggunakan rumus dari *Euclidean distance* sesuai dengan Persamaan 2.4. *Output* yang dihasilkan berbentuk objek *float* yang dihasilkan melalui proses perhitungan jarak.

4.1.3 Proses Pelatihan

Proses pelatihan bertujuan untuk menghasilkan data latih yang memilkiki fitur ruang warna L*a*b* dan dimasukkan dalam Tabel data latih sebagai acuan dari proses klasifikasi untuk penyakit daun jeruk. Detail proses pelatihan ada pada Gambar 4.11.



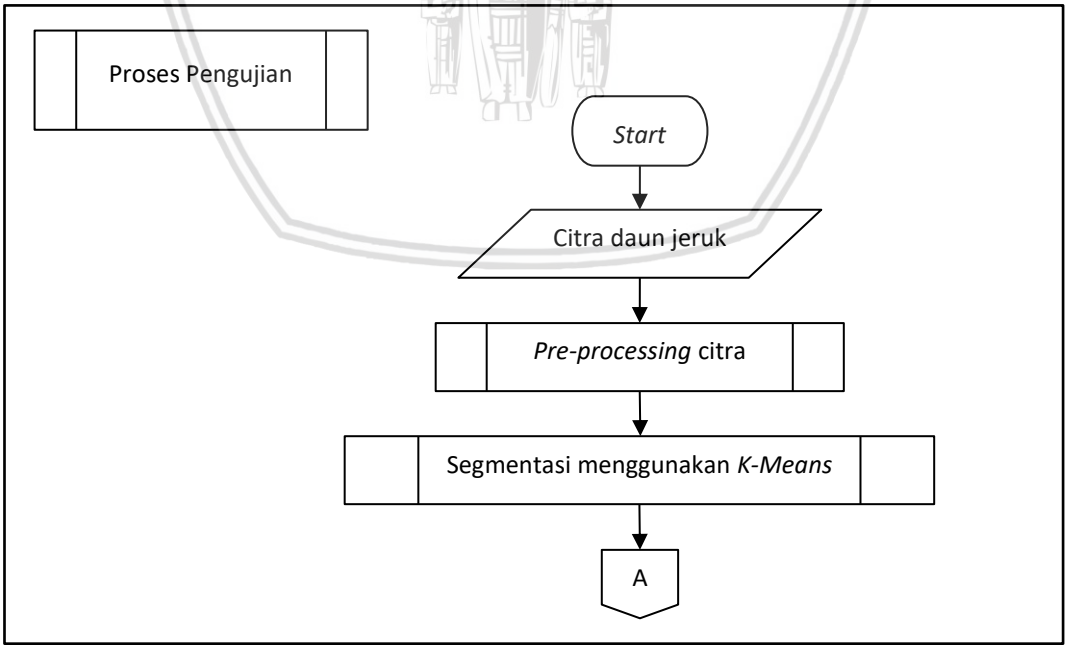


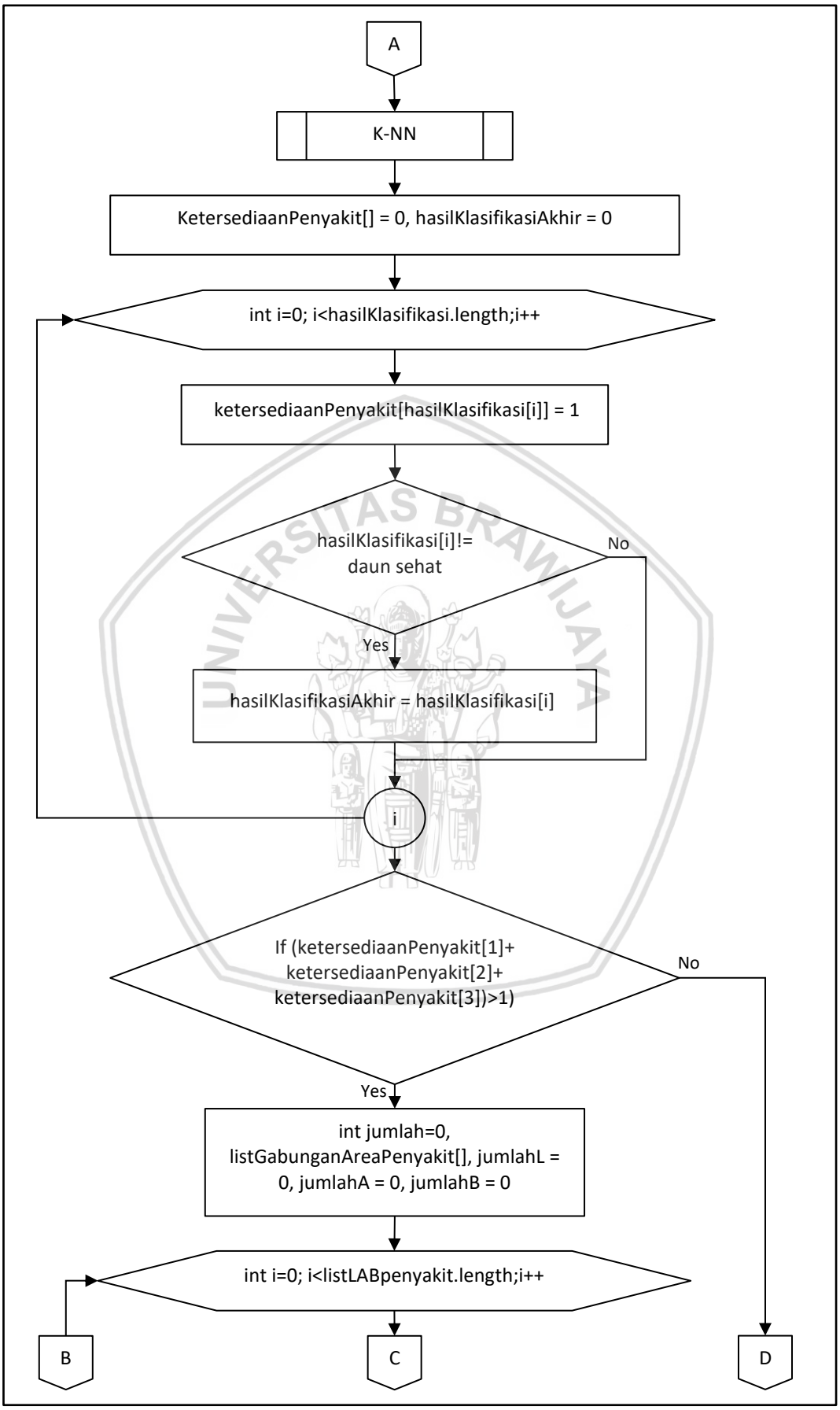
Gambar 4.11 *Flowchart* Proses Pelatihan

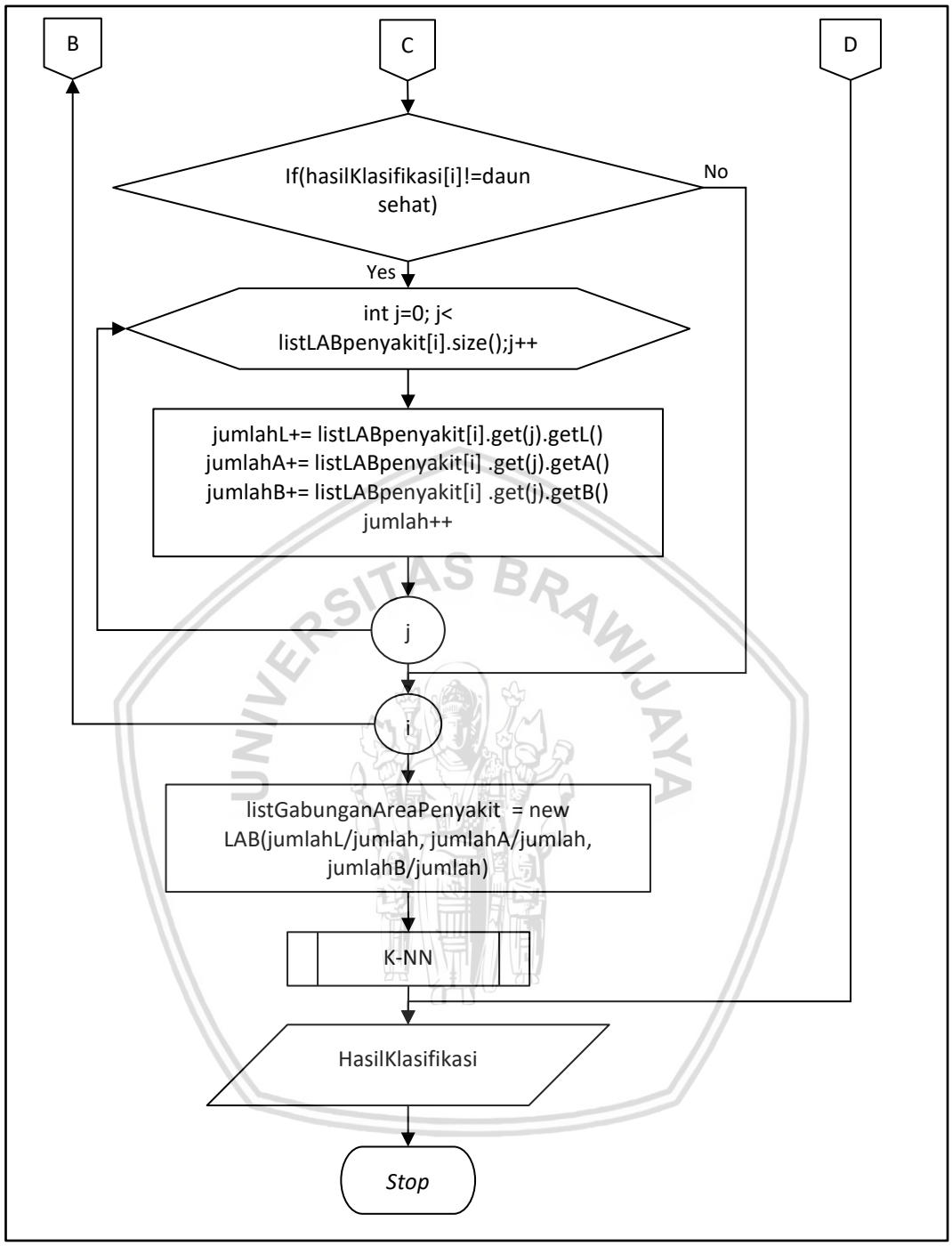
Pada proses pelatihan *input* adalah citra daun jeruk kemudian dilakukan kembali proses *pre-processing* dan segmentasi menggunakan *K-Means*. Hasil dari segmentasi penyakit dimasukkan pada Tabel dan diberi label sesuai dengan jenis penyakitnya. *Output* yang dihasilkan pada proses ini berupa hasil pelatihan yang siap digunakan untuk proses klasifikasi.

4.1.4 Proses Pengujian

Proses pengujian pada penelitian ini berfungsi untuk menguji data uji terhadap data latih yang ada. Pada akhirnya peneliti dapat menilai tingkat akurasi dari suatu program dan faktor-faktor lain yang mempengaruhi keakuratan dari program. Sama seperti proses pelatihan diperlukan untuk melakukan tahapan yaitu *pre-processing* citra dan segmentasi menggunakan *K-Means*. Pada proses pengujian dilakukan klasifikasi terhadap penyakit daun tiap-tiap *cluster* menggunakan algoritme K-NN. Detail proses pengujian ditunjukkan pada Gambar 4.13.




A
K-NN
KetersediaanPenyakit[] = 0, hasilKlasifikasiAkhir = 0
int i=0; i<hasilKlasifikasi.length;i++
ketersediaanPenyakit[hasilKlasifikasi[i]] = 1
hasilKlasifikasi[i]!=
daun sehat
No
Yes
hasilKlasifikasiAkhir = hasilKlasifikasi[i]
i
If (ketersediaanPenyakit[1]+
ketersediaanPenyakit[2]+
ketersediaanPenyakit[3])>1)
No
Yes
int jumlah=0,
listGabunganAreaPenyakit[], jumlahL =
0, jumlahA = 0, jumlahB = 0
int i=0; i<listLABpenyakit.length;i++
B
C
D

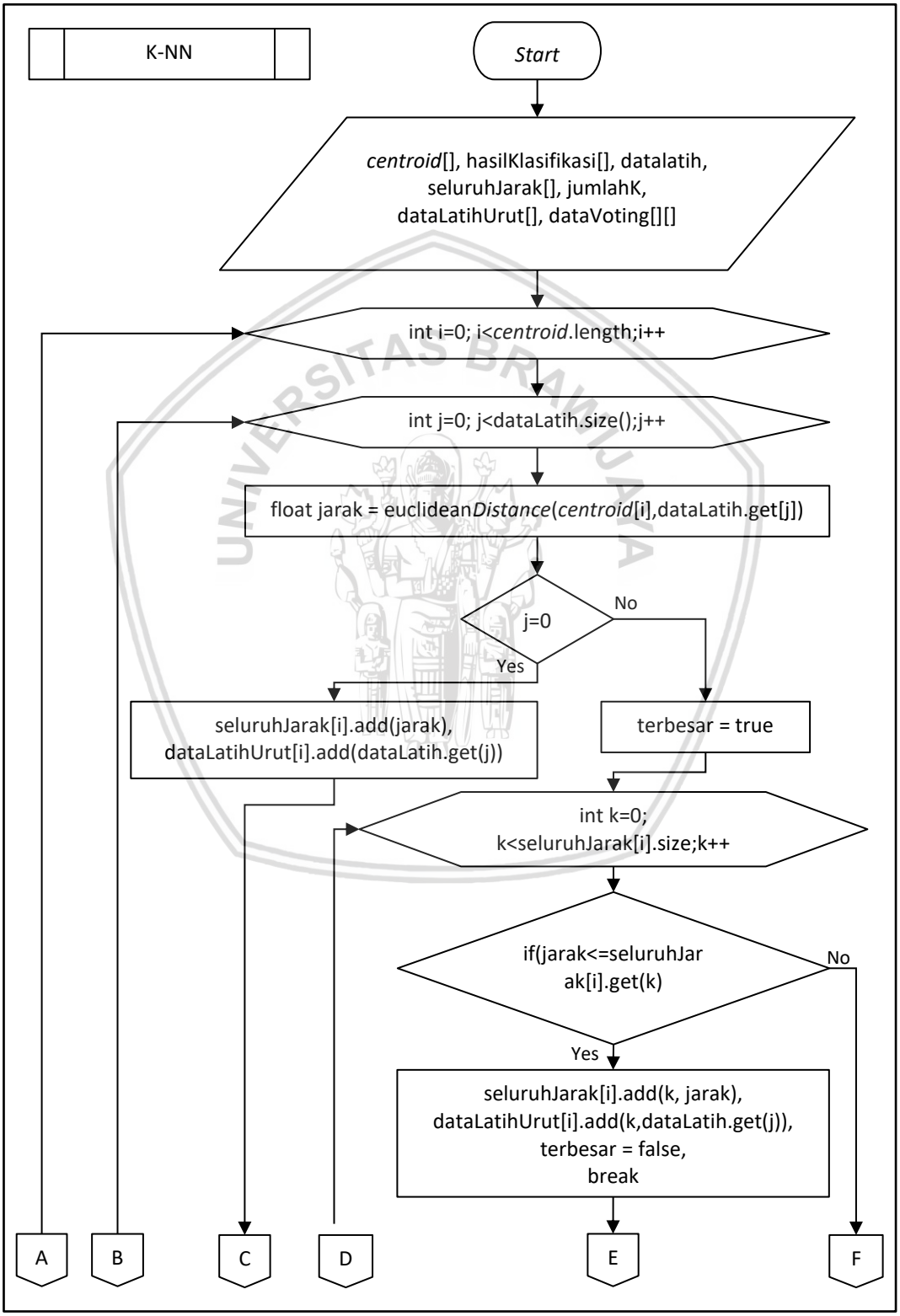


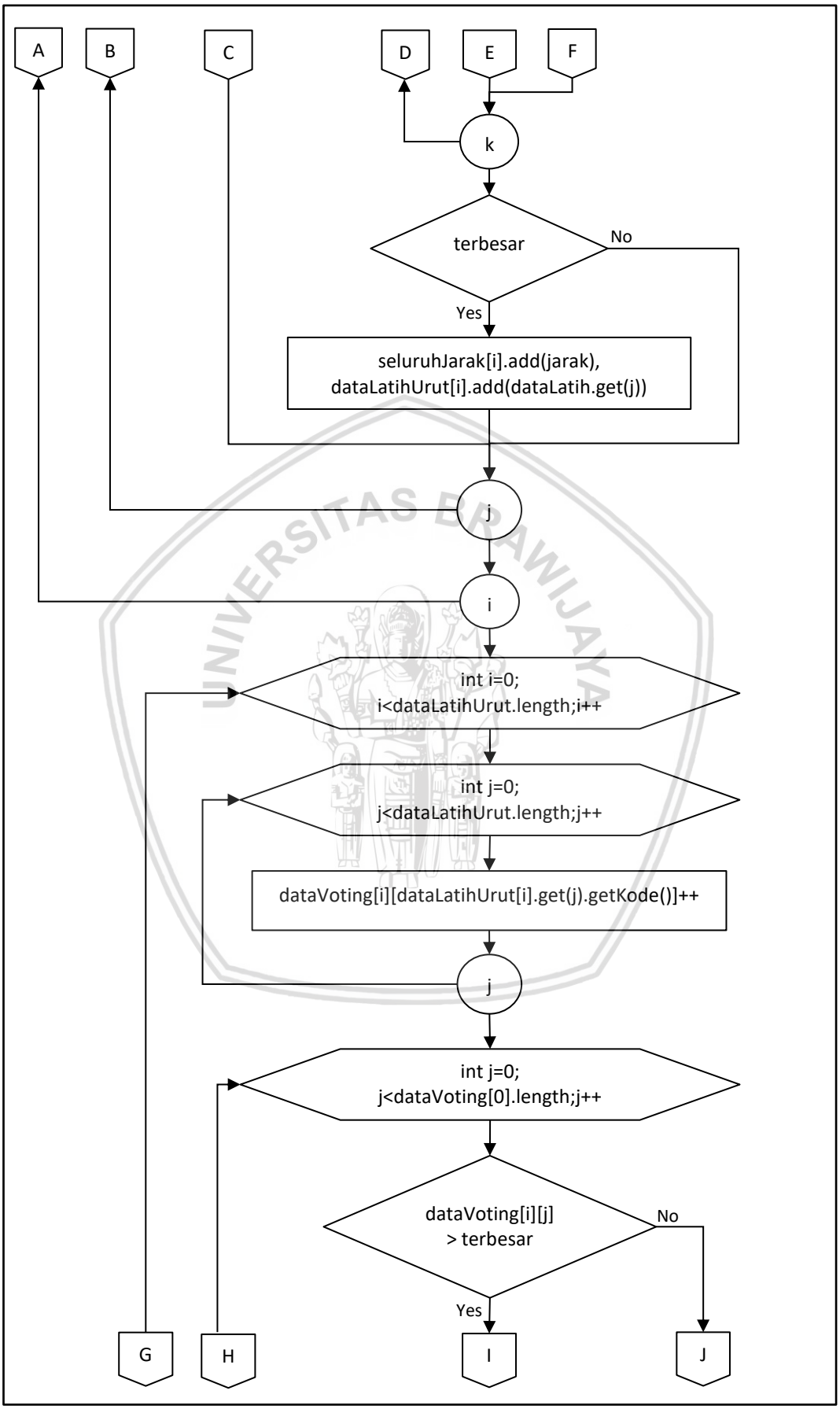
Gambar 4.12 *Flowchart* Proses Pengujian

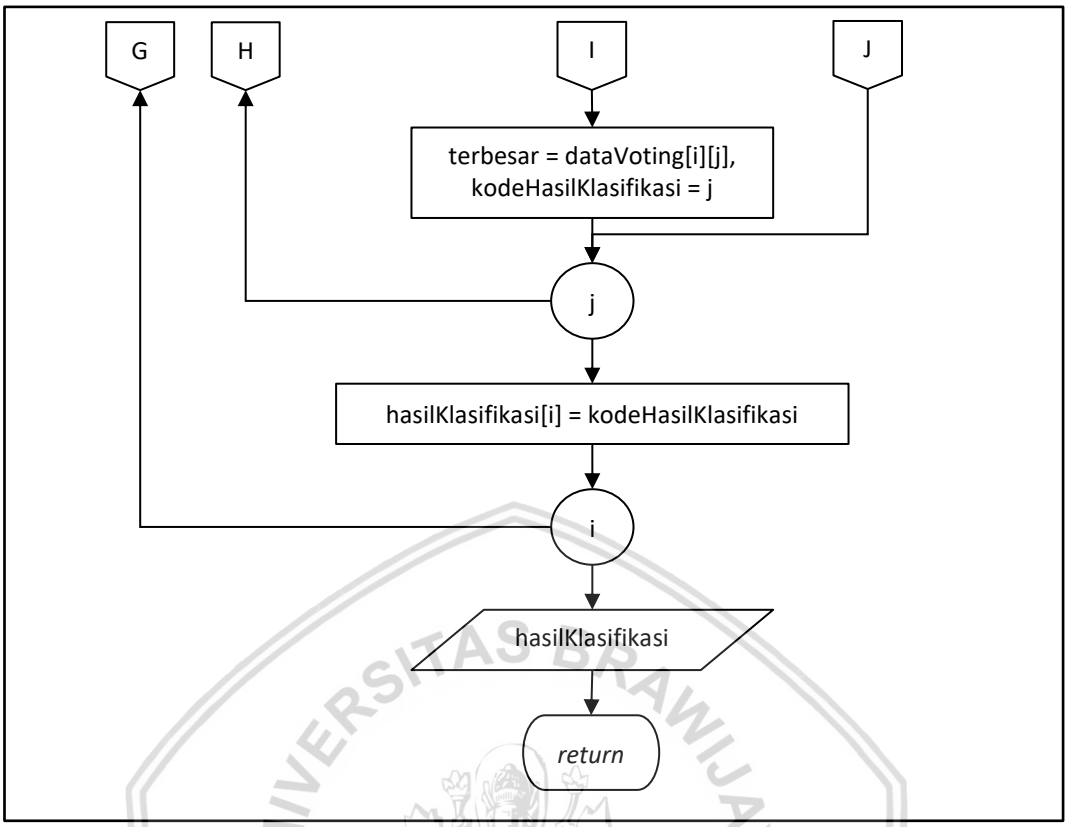
Setelah melalukan *pre-processing* hingga proses klasifikasi kemudian dilakukan cek terhadap hasil penyakit yang ada. Jika ditemukan lebih dari satu jenis penyakit dilakukan perhitungan ulang klasifikasi menggunakan algortima K-NN. *Input* terhadap algoritme K-NN yang kedua adalah dengan melakukan penggabungan seluruh data yang memiliki jenis penyakit selain berjenis daun sehat menggunakan perhitungan rata-rata untuk setiap komponen warna dari L*a*b*. Dari proses tersebut dihasilkan *output* program dengan memiliki satu jenis penyakit.

4.1.4.1 K-NN

Proses algoritme K-NN digunakan untuk melakukan klasifikasi data uji terhadap data latih penyakit daun jeruk. *Output* yang dihasilkan berupa *array* yang berisi string nama jenis penyakit. Urutan proses algoritme K-NN dapat dilihat pada Gambar 4.14.




A
B
C
D
E
F
k
terbesar
No
Yes
seluruhJarak[i].add(jarak),
dataLatihUrut[i].add(dataLatih.get(j))
j
i
int i=0;
i<dataLatihUrut.length;i++
int j=0;
j<dataLatihUrut.length;j++
dataVoting[i][dataLatihUrut[i].get(j).getKode()]++
j
int j=0;
j<dataVoting[0].length;j++
dataVoting[i][j]
> terbesar
No
Yes
G
H
I
J



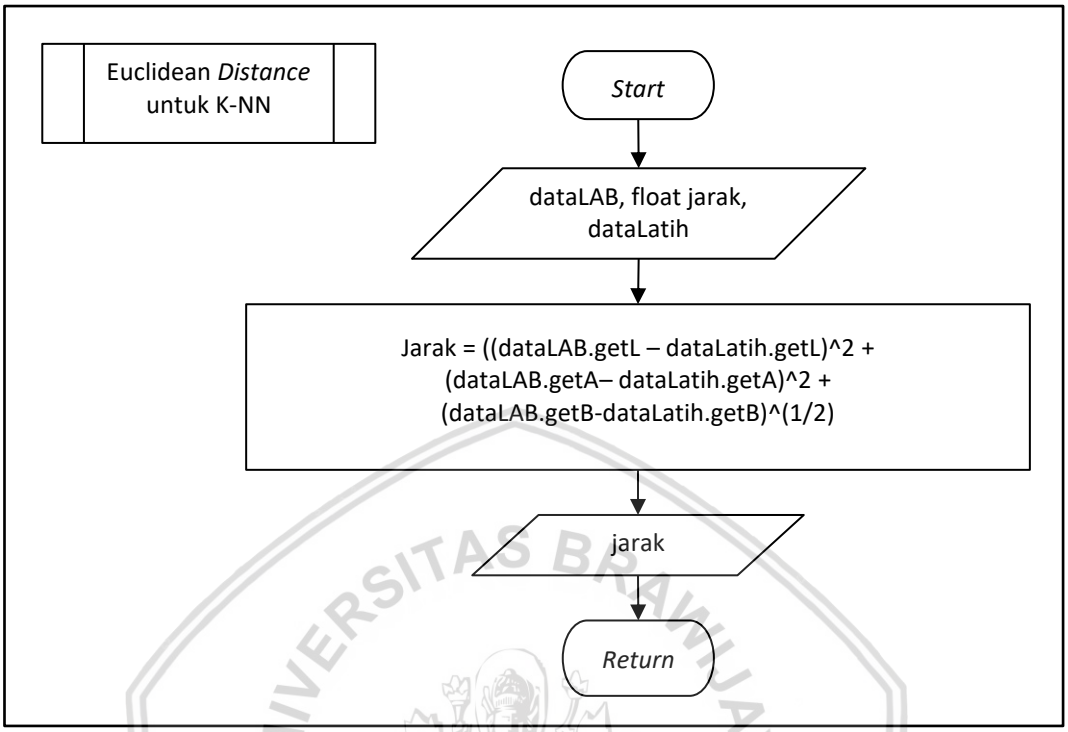
Gambar 4.13 *Flowchart* K-NN

Ada beberapa proses penting dalam algoritme K-NN yang pertama adalah melakukan perhitungan jarak menggunakan *Euclidean distance*. Pada penelitian ini, penggunaan persamaan jarak *Euclidean distance* untuk K-NN memiliki perbedaan pada perhitungan *K-Means* sebelumnya. Jika pada *K-Means* hanya menggunakan parameter a* dan b* dari ruang warna L*a*b* tetapi, jika pada K-NN menggunakan semua komponen ruang warna pada L*a*b*. Pada algoritme K-NN parameter untuk menghitung jarak adalah nilai rata-rata dari seluruh komponen ruang warna L*a*b* dengan *cluster* yang sama hasil segmentasi. Nilai rata-rata tersebut dibandingkan dengan data latih yang sudah ada untuk tiap-tiap jenis penyakit. Selanjutnya dilakukan pengurutan data dari jarak terdekat hingga terjauh tiap *cluster*-nya melalui mekanisme menambah setelah atau sebelum pada arraylist. Kemudian, program menghitung jumlah data terdekat sesuai dengan jumlah k sebagai *input*-an yang selanjutnya digunakam sebagai acuan *voting* pengambilan keputusan hasil klasifikasi. *Index* dari data penyakit yang memiliki kemunculan terbanyak sesuai nilai k disimpulkan menjadi hasil klasifikasi. Hasil klasifikasi sebanyak jumlah *cluster* yang dilakukan testing. *Output* hasil klasifikasi berbentuk *array* yang memilki Panjang sesuai dengan *cluster input*.

4.1.4.2 *Euclidean distance* Untuk K-NN

Proses *Euclidean distance* pada K-NN memiliki perbedaan penggunaan parameter dengan yang dipakai pada algoritme *K-Means*. Pada algoritme K-NN parameter yang digunakan ditambah dengan variabel L* yang terdapat pada ruang

warna L*a*b*. Detail proses *Euclidean distance* untuk K-NN dapat dilihat pada Gambar 4.15.



Gambar 4.14 *Flowchart Euclidean Distance* Untuk K-NN

Penggunaan parameter warna L*a*b* dibandingkan dengan data latih yang ada. Proses perhitungan rumus *Euclidean distance* sesuai dengan persamaan 2.4.

4.2 Perhitungan Manual

Perhitungan manual memiliki fungsi untuk menggambarkan proses matematis dari perancangan yang diterapkan pada program. Perhitungan manual juga bertujuan untuk mengetahui benar atau tidaknya program dalam mengimplementasikan proses atau algoritme yang ada sehingga dapat menjadi acuan dalam perancangan program.

Perhitungan manual yang dilakukan pada penelitian ini antara lain: perhitungan *pre-processing* (*rescaling* dan RGB *to* L*a*b*), segmentasi menggunakan *K-Means* (segmentasi daun dan segmentasi penyakit), dan klasifikasi penyakit daun jeruk. Pada penelitian ini manualisasi dimulai menggunakan contoh citra *input* berukuran *pixel* 3x4 dengan data nilai *red*, *green*, *blue* pada citra aslinya. Penulisan pada tabel dengan memperhatikan urutan penulisan dengan contoh: (*red*, *green*, *blue*). Detail dari citra *input* dapat dilihat pada Tabel 4.1.

Tabel 4.1 RGB Citra *Input*

RGB		
(249, 113, 153)	(251, 116, 156)	(254, 119, 159)
(246, 110, 150)	(50, 66, 27)	(251, 116, 156)
(53, 75, 36)	(56, 49, 33)	(51, 69, 30)
(243, 107, 147)	(52, 72, 33)	(248, 112, 152)

Masing-masing citra *input* juga sudah ditentukan target *output*-nya dengan keluaran citra *input* memiliki penyakit Jelaga. Penulisan masing-masing citra *input* berdasarkan target *output* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Target *Output*

Index	R	G	B	Target *Output*
(0,0)	249	113	153	*Cover*
(0,1)	251	116	156	*Cover*
(0,2)	254	119	159	*Cover*
(1,0)	246	110	150	*Cover*
(1,1)	50	66	27	Daun Sehat
(1,2)	251	116	156	*Cover*
(2,0)	53	75	36	Daun Sehat
(2,1)	56	49	33	Cendawan Jelaga
(2,2)	51	69	30	Daun Sehat
(3,0)	243	107	147	*Cover*
(3,1)	52	72	33	Daun Sehat
(3,2)	248	112	152	*Cover*

4.2.1 Manualisasi *Pre-processing* Citra

Proses manualisasi pada *pre-processing* citra terdiri atas dua bagian yaitu *rescaling* dan perubahan ruang warna RGB menjadi L*a*b*. *Rescaling* berfungsi untuk mengatur tingkat kecerahan pada citra yang dipengaruhi parameter nilai *Scale Factor* dan nilai *offset*. Sedangkan perubahan ruang warna RGB melalui tahap perubahan ruang warna RGB ke XYZ terlebih dahulu, barulah dari XYZ dirubah menjadi ruang warna L*a*b*.

4.2.1.1 Manualisasi *Rescaling*

Pada saat *pre-processing* langkah pertama yang dilakukan adalah proses *rescaling*. Proses *rescaling* dimulai dengan citra *input* pada Tabel 4.1 yang diperjelas dengan memisah setiap bagian variabel ruang warnanya menjadi seperti yang ditunjukkan pada Tabel 4.3.

Tabel 4.3 Citra *Input* Nilai RGB Terpisah

R			G			B		
249	251	254	113	116	119	153	156	159
246	50	251	110	66	116	150	27	156
53	56	51	75	49	69	36	33	30
243	52	248	107	72	112	147	33	152

Untuk menyelesaikan perhitungan *rescaling* dapat menggunakan Persamaan 2.1. parameter pada manualisasi kali ini menggunakan nilai *Scale Factor* = 1.1 dan nilai *offset* = 0. Berikut merupakan contoh perhitungan pada *pixel* pertama dari Tabel 4.2:

$$Red = red * scaleFactor + offset$$
$$= 249 * 1.1 + 0 = 274$$

$$Green = green * scaleFactor + offset$$
$$= 113 * 1.1 + 0 = 124$$

$$Blue = blue * scaleFactor + offset$$
$$= 153 * 1.1 + 0 = 168$$

Hasil perhitungan sesuai dengan persamaan 2.1 menghasilkan nilai *red*=274, *green*=124, dan *blue*=168. Penggunaan tipe data pada perhitungan *rescaling* menggunakan tipe data *integer* untuk setiap perhitungan dibulatkan sehingga hasilnya bertipe data *integer*. Untuk keseluruhan perhitungan *pixel* dapat dilihat melalui Tabel 4.4.

Tabel 4.4 Hasil Perhitungan *Rescaling*

R			G			B		
274	276	279	124	128	131	168	172	175
271	55	276	121	73	128	165	30	172
58	62	56	83	54	76	40	36	33
267	57	273	118	79	123	162	36	167

Ruang warna RGB memiliki rentan nilai antara 0 sampai dengan 255. Selanjutnya, jika pada hasil perhitungan *rescaling* suatu *pixel* memiliki nilai variabel lebih dari 255 akan dilakukan perubahan dengan menjadikan maksimal dengan nilai 255. Sebagai contoh nilai *red* pada *pixel* pertama memiliki nilai 274

sehingga dilakukan perubahan nilai menjadi nilai *pixel* maximal yaitu 255. Perbaikan nilai RGB ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil Perbaikan Nilai *Rescaling*

R			G			B		
255	255	255	124	128	131	168	172	175
255	55	255	121	73	128	165	30	172
58	62	56	83	54	76	40	36	33
255	57	255	118	79	123	162	36	167

Nilai yang ditunjukkan pada Tabel 4.5 juga merupakan hasil akhir dari proses *rescaling* sehingga nilai tersebut menjadi acuan pada proses selanjutnya.

4.2.1.2 Manualisasi RGB to L*a*b*

Perhitungan dalam perubahan ruang warna RGB menjadi L*a*b* melalui beberapa proses. Proses yang pertama adalah mengubah ruang warna RGB menjadi XYZ. Kemudian proses yang kedua dengan mengubah ruang warna XYZ menjadi L*a*b*. Untuk melakukan perubahan ruang warna RGB menjadi XYZ dilakukan dengan transformasi *matrix* 3x3 yang melibatkan nilai tristimulus sesuai dengan persamaan 2.2. Citra *input* untuk proses perubahan ruang warna ditunjukkan pada Tabel 4.6.

Tabel 4.6 Citra *Input* Proses Perubahan Ruang Warna

R			G			B		
255	255	255	124	128	131	168	172	175
255	55	255	121	73	128	165	30	172
58	62	56	83	54	76	40	36	33
255	57	255	118	79	123	162	36	167

berikut contoh perhitungan *pixel* pertama untuk perubahan ruang warna RGB menjadi XYZ :

$$X = (red * 0.412453) + (green * 0.357580) + (blue * 0.180423)$$
$$= (255 * 0.412453) + (124 * 0.357580) + (168 * 0.180423)$$
$$= 179.83$$

$$Y = (red * 0.212671) + (green * 0.715160) + (blue * 0.072169)$$
$$= (255 * 0.212671) + (124 * 0.715160) + (168 * 0.072169)$$
$$= 155.04$$

$$Z = (red * 0.019334) + (green * 0.119193) + (blue * 0.950227)$$
$$= (255 * 0.019334) + (124 * 0.119193) + (168 * 0.950227)$$
$$= 179.35$$

Hasil perhitungan transformasi *matrix* 3x3 sesuai dengan persamaan 2.2 pada *pixel* pertama. menghasilkan nilai X=179.83, Y=155.04, dan Z=179.35 sedangkan perhitungan pada seluruh *pixel* dapat dilihat melalui Tabel 4.7.

Tabel 4.7 Hasil Ruang Warna XYZ

X		
179,83	181,98	183,59
178,21	54,20	181,98
60,82	51,38	56,23
176,60	58,25	179,29

Y		
155,04	158,18	160,55
152,67	66,07	158,18
74,58	54,40	68,64
150,31	71,22	154,25

Z		
179,35	183,63	186,83
176,14	38,27	183,63
49,02	41,84	41,50
172,93	44,73	178,28

Setelah mendapatkan nilai ruang warna XYZ kemudian melakukan proses untuk mendapatkan nilai ruang warna L*a*b*. Komponen penyusun untuk mendapatkan ruang warna L*a*b* sesuai persamaan 2.3, diperlukan untuk membagi masing-masing nilai ruang warna XYZ dengan masing-masing *white reference*-nya. *White reference* adalah nilai maximal daripada masing-masing penyusun ruang warna XYZ disimbolkan dengan X_0, Y_0, dan Z_0. X_0 bernilai 242.36628, Y_0 bernilai 255, dan Z_0 bernilai 277,63227 sehingga rentan nilai dengan pembagian tersebut pada 0 sampai dengan 1. Berikut contoh perhitungan pembagian ruang warna dengan *white reference*-nya :

$$\frac{X}{X_0} = \frac{179.83}{242.36628} = 0.74$$

$$\frac{Y}{Y_0} = \frac{155.04}{255} = 0.61$$

$$\frac{Z}{Z_0} = \frac{179.35}{277,63227} = 0.65$$

Dihasilkan nilai 0.74 untuk X/X_0, 0.61 untuk Y/Y_0, dan 0.65 untuk Z/Z_0. Keseluruhan hasil perhitungan dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Pembagian XYZ Dengan *White Reference*

X/X_0		
0,74	0,75	0,76
0,74	0,22	0,75
0,25	0,21	0,23
0,73	0,24	0,74

Y/Y_0		
0,61	0,62	0,63
0,60	0,26	0,62
0,29	0,21	0,27
0,59	0,28	0,60

Z/Z_0		
0,65	0,66	0,67
0,63	0,14	0,66
0,18	0,15	0,15
0,62	0,16	0,64

Kemudian komponen penyusun dari persamaan 2.3 adalah dengan perhitungan fungsi yang disimbolkan dengan $f(X/X_0)$. $f(Y/Y_0)$, dan $f(Z/Z_0)$. Sesuai persamaan 2.3, terdapat dua kondisi yaitu ketika hasil pembagian XYZ dengan *white reference*-nya lebih dari 0.008856 dan ketika hasil pembagian XYZ dengan *white reference*-nya memiliki rentan nilai dari 0 hingga 0.008856. Berikut perhitungan fungsi pada *pixel* pertama untuk komponen penyusun rumus perubahan ruang warna XYZ menjadi L*a*b*:

$$f(w) = w^{1/3} \qquad \text{for } w > 0.008856$$
$$f(w) = 7.787(w) + 0.1379 \qquad \text{for } 0.0 \leq w \leq 0.008856$$

$$X/X_0 = 0.76(masuk\ kondisi\ pertama)$$
$$f(X/X_0) = (X/X_0)^{1/3}$$
$$= 0.74^{1/3} = 0.91$$

$$Y/Y_0 = 0.65(masuk\ kondisi\ pertama)$$
$$f(Y/Y_0) = (Y/Y_0)^{1/3}$$
$$= 0.61^{1/3} = 0.85$$

$$Z/Z_0 = 0.60(masuk\ kondisi\ kedua)$$
$$f(Z/Z_0) = (Z/Z_0)^{1/3}$$
$$= 0.65^{1/3} = 0.86$$

Pada perhitungan fungsi dihasilkan $f(X/X_0)$=0.91. $f(Y/Y_0)$=0.85, dan $f(Z/Z_0)$=0.86. Perhitungan fungsi pada seluruh *pixel* dapat dilihat melalui Tabel 4.9.

Tabel 4.9 Perhitungan Fungsi Terhadap XYZ

f(X/X0)			f(Y/Y0)			f(Z/Z0)		
0,91	0,91	0,91	0,85	0,85	0,86	0,86	0,87	0,88
0,90	0,61	0,91	0,84	0,64	0,85	0,86	0,52	0,87
0,63	0,60	0,61	0,66	0,60	0,65	0,56	0,53	0,53
0,90	0,62	0,90	0,84	0,65	0,85	0,85	0,54	0,86

Pada tahap akhir perhitungan untuk mengubah ruang warna XYZ menjadi L*a*b* sesuai dengan persamaan 2.3. Terdapat dua kondisi untuk perhitungan L* ketika nilai Y/Y_0 lebih dari 0.008856 atau ketika nilai Y/Y_0 berada pada rentan 0 hingga 0.008856. Berikut contoh perhitungan konversi ruang warna L*a*b*:

$$L^* = 116(\frac{Y}{Y_0})^{1/3} - 16 \qquad \text{for } \frac{Y}{Y_0} > 0.008856$$

$$L^* = 903.3\,\frac{Y}{Y_0} \qquad \text{for } 0.0 \leq \frac{Y}{Y_0} \leq 0.008856$$

$$Y/Y_0 = 0.65(masuk\ kondisi\ pertama)$$

$$L^* = 116(Y/Y_0)^{1/3} - 16$$

$$= 116(0.61)^{1/3} - 16$$

$$= 82.27$$

$$a^* = 500\,(f(X/X_0) - f(Y/Y_0))$$

$$= 500(0.91 - 0.85)$$

$$= 29.07$$

$$b^* = 200\,(f(Y/Y_0) - f(Z/Z_0))$$

$$= 200(0.85 - 0.86)$$

$$= -3.46$$

Perhitungan konversi L*a*b* menghasilkan L* = 82.27, a* = 29.07, dan b* = -3.46 untuk *pixel* pertama. Hasil perhitungan *pixel* keseluruhan terdapat pada Tabel 4.10.

Tabel 4.10 Hasil Ruang Warna L*a*b*

L*			a*			b*		
82,27	82,93	83,42	29,07	28,02	27,25	-3,46	-3,68	-3,85
81,77	57,95	82,93	29,88	-15,26	28,02	-3,29	24,19	-3,68
61,00	53,31	58,90	-16,52	-0,64	-15,61	20,55	13,07	23,00
81,26	59,82	82,10	30,70	-15,95	29,34	-3,11	21,91	-3,40

4.2.2 Manualisasi Segmentasi Citra Menggunakan *K-Means*

Perhitungan segmentasi citra dengan algoritme *K-Means* hanya menggunakan dua variabel dari ruang warna L*a*b* yaitu variabel a* dan b*. Bertujuan untuk fokus pada tingkat kemurnian warna dengan mengabaikan tingkat kecerahan (*luminance*). Pada penelitian ini segmentasi dibagi menjadi dua bagian yaitu segmentasi daun dan segmentasi penyakit. Segmentasi daun bertujuan memisahkan daun dengan *cover* sedangkan segmentasi penyakit untuk menghasilkan bagian-bagian pada daun yang digunakan pada proses klasifikasi jenis penyakit. Data *input* pada segmentasi citra menggunakan algoritme *K-Means* dapat dilihat pada Tabel 4.11.

Tabel 4.11 Data *Input K-Means*

a*			b*		
29,07	28,02	27,25	-3,46	-3,68	-3,85
29,88	-15,26	28,02	-3,29	24,19	-3,68
-16,52	-0,64	-15,61	20,55	13,07	23,00
30,70	-15,95	29,34	-3,11	21,91	-3,40

4.2.2.1 Manualisasi Segmentasi Daun

Pada proses segmentasi daun menggunakan algoritme *K-Means* dilakukan permisalan kasus menggunakan jumlah *cluster* = 2. Data pada Tabel 4.11 dijadikan satu pada setiap bagiannya dan diberi *index* untuk setiap *pixel*-nya sehingga dijadikan Tabel baru seperti yang ditunjukkan pada Tabel 4.12.

Tabel 4.12 Data *Input* Segmentasi Daun

Index	a*	b*
(0,0)	29,07	-3,46
(0,1)	28,02	-3,68
(0,2)	27,25	-3,85
(1,0)	29,88	-3,29
(1,1)	-15,26	24,19
(1,2)	28,02	-3,68
(2,0)	-16,52	20,55
(2,1)	-0,64	13,07
(2,2)	-15,61	23,00
(3,0)	30,70	-3,11
(3,1)	-15,95	21,91
(3,2)	29,34	-3,40

Langkah pertama untuk memulai algoritme *K-Means* pada iterasi ke 0 adalah menentukan nilai *centroid* awal sebanyak jumlah *cluster* yang ditetapkan. Pada perhitungan manual kali ini, dengan jumlah *cluster* = 2. Ditentukan *centroid* secara acak yang dapat dilihat pada Tabel 4.13.

Tabel 4.13 *Centroid* Awal Segmentasi Daun

Centroid	a*	b*
c1	29,07	-3,46
c2	-15,26	24,19

Selanjutnya, setelah ditentukan *centroid* kemudian tiap-tiap data *pixel* dikelompokkan sesuai dengan *centroid* terdekat melalui perhitungan jarak *Euclidean distance*. Setelah jarak antara data dan dua *centroid* dibandingkan jarak

terdekat dipilih sebagai *cluster* dari data. Berikut contoh perhitungan jarak pada *pixel* dengan *index* (0,0):

$$d_{c1} = \sqrt{(29.07 - 29.07)^2 + ((-3.46) - (-3.46))^2} = 0$$

$$d_{c2} = \sqrt{(29.07 - (-15.26))^2 + ((-3.46) - 24.19)^2} = 52.25$$

Dengan hasil d_{c1}= 0 dan d_{c2}= 52.25 maka, keputusan yang diambil adalah untuk *index* (0,0) masuk kategori c1. Perhitungan jarak pada semua *index* pada iterasi ke 0 dapat dilihat pada Tabel 4.14.

Tabel 4.14 Jarak *Cluster* Segmentasi Daun Iterasi ke 0

Index	a*	b*	*distance* c1	*distance* c2	*cluster*
(0,0)	29,07	-3,46	0,00	52,25	c1
(0,1)	28,02	-3,68	1,07	51,48	c1
(0,2)	27,25	-3,85	1,86	50,92	c1
(1,0)	29,88	-3,29	0,82	52,84	c1
(1,1)	-15,26	24,19	52,25	0,00	c2
(1,2)	28,02	-3,68	1,07	51,48	c1
(2,0)	-16,52	20,55	51,53	3,85	c2
(2,1)	-0,64	13,07	34,00	18,37	c2
(2,2)	-15,61	23,00	51,93	1,24	c2
(3,0)	30,70	-3,11	1,66	53,45	c1
(3,1)	-15,95	21,91	51,68	2,38	c2
(3,2)	29,34	-3,40	0,27	52,44	c1

Dari Tabel 4.14 didapatkan jumlah *cluster* 1 sebanyak 7 dan jumlah *cluster* 2 sebanyak 5. Langkah selanjutnya adalah melakukan update *centroid* berdasarkan rata-rata dari tiap variabel pendukung sesuai *cluster* yang sama. Berikut contoh perhitungan *centroid* baru:

$$a^*_{c1} = \frac{(29.07 + 28.02 + 27.25 + 29.88 + 28.02 + 30.70 + 29.34)}{7} = 28.9$$

$$b^*_{c1} = \frac{((-3.46) + (-3.68) + (-3.85) + (-3.29) + (-3.68) + (-3.11) + (-3.40))}{7} = (-3.5)$$

$$a^*_{c2} = \frac{((-15.26) + (-16.52) + (-0.64) + (-15.61) + (-15.95))}{5} = -12.80$$

$$b^*_{c2} = \frac{(24.19 + 20.55 + 13.07 + 23.00 + 21.91)}{5} = 20.54$$

Hasil *centroid* baru dapat dilihat pada Tabel 4.15.

Tabel 4.15 *Centroid* Baru Iterasi ke 0 Segmentasi Daun

Centroid Baru	a*	b*
c1	28,90	-3,50
c2	-12,80	20,54

Perhitungan dilanjutkan pada iterasi berikutnya hingga terdapat kondisi konvergen yang mana tidak ada data berpindah dari kelas *cluster*. *Centroid* yang digunakan pada iterasi ke 1 sesuai pada Tabel 4.16 dan hasil perhitungan jarak iterasi ke 1 terdapat pada Tabel 4.17.

Tabel 4.16 *Centroid* Iterasi ke 1 Segmentasi Daun

Centroid	a*	b*
c1	28,90	-3,50
c2	-12,80	20,54

Tabel 4.17 Jarak *Cluster* Segmentasi Daun Iterasi ke 1

Index	a*	b*	*distance* c1	*distance* c2	*cluster*
(0,0)	29,07	-3,46	0,18	48,26	c1
(0,1)	28,02	-3,68	0,89	47,47	c1
(0,2)	27,25	-3,85	1,68	46,89	c1
(1,0)	29,88	-3,29	1,00	48,88	c1
(1,1)	-15,26	24,19	52,12	4,40	c2
(1,2)	28,02	-3,68	0,89	47,47	c1
(2,0)	-16,52	20,55	51,39	3,72	c2
(2,1)	-0,64	13,07	33,86	14,27	c2
(2,2)	-15,61	23,00	51,80	3,73	c2
(3,0)	30,70	-3,11	1,84	49,51	c1
(3,1)	-15,95	21,91	51,54	3,44	c2
(3,2)	29,34	-3,40	0,45	48,46	c1

Pada perhitungan jarak iterasi ke 1 yang ditampilkan pada Tabel 4.17, menunjukkan tidak ada kelas *cluster* yang berpindah dari iterasi sebelumnya sehingga kondisi ini dapat dikatakan konvergen. Hasil kesimpulan pada algoritme *K-Means* dapat dilihat pada Tabel 4.18.

Tabel 4.18 Hasil *K-Means* Pada Segmentasi Daun

Index	L*	a*	b*	*cluster*
(0,0)	82,27	29,07	-3,46	c1
(0,1)	82,93	28,02	-3,68	c1
(0,2)	83,42	27,25	-3,85	c1
(1,0)	81,77	29,88	-3,29	c1

Index	L*	a*	b*	*cluster*
(1,1)	57,95	-15,26	24,19	c2
(1,2)	82,93	28,02	-3,68	c1
(2,0)	61,00	-16,52	20,55	c2
(2,1)	53,31	-0,64	13,07	c2
(2,2)	58,90	-15,61	23,00	c2
(3,0)	81,26	30,70	-3,11	c1
(3,1)	59,82	-15,95	21,91	c2
(3,2)	82,10	29,34	-3,40	c1

Selanjutnya proses segmentasi daun dengan menentukan kelas *cover* dan memisahkannya dengan bagian daun. Cara untuk mengetahui kelas *cover* dengan menggunakan algoritme K-NN. Nilai k yang digunakan pada proses ini adalah 1. Proses pertama dari algoritme K-NN dengan menghitung jarak nilai rata-rata dari setiap komponen ruang warna L*a*b* tiap *cluster* dengan data latih yang ada. Contoh data latih pada proses ini dapat dilihat melalui Tabel 4.19.

Tabel 4.19 Data Latih Untuk Segmentasi Daun

Data Latih	L*	a*	b*	Klasifikasi
data 1	65,74	2,31	5,26	Bagian Daun
data 2	79,80	30,36	1,29	*Cover*
data 3	66,67	-18,77	16,23	Bagian Daun
data 4	77,27	33,76	-3,81	*Cover*

Data uji dihasilkan setelah proses segmentasi awal selesai. Rata-rata dari komponen ruang warna L*a*b* dapat dihasilkan oleh setiap *cluster* yang sama. Untuk nilai a* dan b* dapat menggunakan nilai *centroid* terakhir dari perhitungan *K-Means*. Sedangkan untuk nilai L* dapat mencari rata-ratanya sesuai *cluster* yang terbentuk. Berikut contoh perhitungan rata-rata nilai L* pada *cluster* 1 :

$$L^*_{c1} = \frac{(82.27 + 82.93 + 83.42 + 81.77 + 82.93 + 81.26 + 82.10)}{7} = 82.38$$

Hasil perhitungan nilai L* untuk *cluster* 1 adalah 82,38. Untuk seluruh nilai data uji ditunjukkan pada Tabel 4.20.

Tabel 4.20 Data Uji Untuk Segmentasi Daun

Rata-rata	L*	a*	b*
c1	82,38	28,90	-3,50
c2	58,20	-12,80	20,54

Perhitungan jarak menggunakan rumus *Euclidean distance*. Berikut contoh perhitungan jarak untuk data *cluster* 1 dengan data latih pertama :

$$d_{c1-data1} = \sqrt{(82.38 - 65.74)^2 + (28.90 - 2.31)^2 + ((-3.50) - 5.26)^2} = 32.57$$

Hasil perhitungan jarak antara data uji pertama dengan data latih pertama adalah 32.57. Untuk hasil perhitungan jarak *cluster* 1 dengan seluruh data latih dapat dilihat pada Tabel 4.21 sedangkan hasil perhitungan jarak *cluster* 2 dengan seluruh data latih dapat dilihat pada Tabel 4.22.

Tabel 4.21 Hasil Perhitungan Jarak Data Latih *Cluster* 1 Segmentasi Daun

Data Latih	Jarak	Penyakit
data 1	32,57	Bagian Daun
data 2	5,64	*Cover*
data 3	53,93	Bagian Daun
data 4	7,06	*Cover*

Tabel 4.22 Hasil Perhitungan Jarak Data Latih *Cluster* 2 Segmentasi Daun

Data Latih	Jarak	Penyakit
data 1	22,77	Bagian Daun
data 2	51,96	*Cover*
data 3	11,22	Bagian Daun
data 4	55,89	*Cover*

Setelah perhitungan jarak dengan seluruh data latih kemudian data latih diurutkan mulai dari jarak yang terdekat hingga jarak yang terjauh. Untuk *cluster* 1 pengurutan data dapat dilihat pada Tabel 4.23 sedangkan *cluster* 2 dapat dilihat melalui Tabel 4.24. Setelah dilakukan pengurutan lalu diambil sebanyak K nilai teratas. Karena K=1 maka, pada perhitungan kali ini diambil kesimpulan dari data teratas.

Tabel 4.23 *Voting* K-NN *Cluster* 1 Segmentasi Daun

Data Latih	Jarak	Penyakit
data 2	5,64	*Cover*
data 4	7,06	*Cover*
data 1	32,57	Bagian Daun
data 3	53,93	Bagian Daun

Tabel 4.24 *Voting* K-NN *Cluster* 2 Segmentasi Daun

Data Latih	Jarak	Penyakit
data 3	11,22	Bagian Daun
data 1	22,77	Bagian Daun
data 2	51,96	*Cover*
data 4	55,89	*Cover*

Kesimpulan pada Tabel 4.23 dan Tabel 4.24 menunjukkan bahwa *cluster* 1 diklasifikasikan sebagai area *cover* dan *cluster* 2 diklasifikasikan sebagai bagian daun. Detail hasil klasifikasi segmentasi daun ditunjukkan pada Tabel 4.25.

Tabel 4.25 Hasil Klasifikasi Segmentasi Daun

Index	L*	a*	b*	Klasifikasi
(0,0)	82,27	29,07	-3,46	*Cover*
(0,1)	82,93	28,02	-3,68	*Cover*
(0,2)	83,42	27,25	-3,85	*Cover*
(1,0)	81,77	29,88	-3,29	*Cover*
(1,1)	57,95	-15,26	24,19	Bagian Daun
(1,2)	82,93	28,02	-3,68	*Cover*
(2,0)	61,00	-16,52	20,55	Bagian Daun
(2,1)	53,31	-0,64	13,07	Bagian Daun
(2,2)	58,90	-15,61	23,00	Bagian Daun
(3,0)	81,26	30,70	-3,11	*Cover*
(3,1)	59,82	-15,95	21,91	Bagian Daun
(3,2)	82,10	29,34	-3,40	*Cover*

4.2.2.2 Manualisasi Segmentasi Penyakit

Perhitungan segmentasi penyakit menggunakan *input* nilai a* dan b* dari bagian daun sesuai Tabel 4.25 yang ditulis kembali pada Tabel 4.26. Pada segmentasi penyakit kali ini dimisalkan jumlah *cluster* adalah 2 dan *centroid* awal ditunjukkan pada Tabel 4.27. fungsi dari segmentasi penyakit untuk mendapatkan bagian-bagian dari daun yang nantinya diklasifikasikan menggunakan algoritme K-NN untuk diketahui jenis penyakitnya.

Tabel 4.26 Data *Input* Segmentasi Penyakit

Index	a*	b*
(1,1)	-15,26	24,19
(2,0)	-16,52	20,55
(2,1)	-0,64	13,07
(2,2)	-15,61	23,00
(3,1)	-15,95	21,91

Centroid awal pada iterasi ke 0 kembali dipilih secara *random* untuk nilainya. Detail nilai *centroid* awal ada pada Tabel 4.27.

Tabel 4.27 *Centroid* Awal Segmentasi Penyakit

Centroid	a*	b*
c1	-15,26	24,19
c2	-0,64	13,07

Proses selanjutnya perhitungan jarak *Euclidean distance* antara data *input* dan *centroid* untuk ditentukan kelas *cluster*-nya. Perhitungan *Euclidean distance* menggunakan Persamaan 2.4. Berikut contoh perhitungan jarak *Euclidean distance* pada data pertama:

$$d_{c1} = \sqrt{((-15.26) - (-15.26))^2 + (24.19 - 24.19)^2} = 0$$

$$d_{c2} = \sqrt{((-15.26) - (-0.64))^2 + (24.19 - 13.07)^2} = 18.37$$

Hasil jarak data pertama terhadap *cluster* 1 adalah 0 dan hasil jarak data pertama terhadap *cluster* 2 adalah 18.37 sehingga kesimpulannya data pertama masuk pada *cluster* 1. Perhitungan seluruh data ada pada Tabel 4.28.

Tabel 4.28 Jarak *Cluster* Segmentasi Penyakit Iterasi ke 0

Index	a*	b*	*distance* c1	*distance* c2	*cluster*
(1,1)	-15,26	24,19	0,00	18,37	c1
(2,0)	-16,52	20,55	3,85	17,56	c1
(2,1)	-0,64	13,07	18,37	0,00	c2
(2,2)	-15,61	23,00	1,24	17,96	c1
(3,1)	-15,95	21,91	2,38	17,68	c1

Dari Tabel 4.28 didapatkan jumlah *cluster* 1 sebanyak 4 dan jumlah *cluster* 2 sebanyak 1. Langkah selanjutnya adalah melakukan update *centroid* berdasarkan rata-rata dari tiap variabel pendukung sesuai *cluster* yang sama. Berikut contoh perhitungan *centroid* baru:

$$a^*_{c1} = \frac{((-15.26) + (-16.52) + (-15.61) + (-15.95))}{4} = -15.84$$

$$b^*_{c1} = \frac{(24.19 + 20.55 + 23.00 + 21.91)}{4} = 22.41$$

$$a^*_{c2} = \frac{(-0.64)}{1} = -0.64$$

$$b^*_{c2} = \frac{13.07}{1} = 13.07$$

Hasil *centroid* baru dapat dilihat pada Tabel 4.29.

Tabel 4.29 *Centroid* Baru Iterasi ke 0 Segmentasi Penyakit

Centroid Baru	a*	b*
c1	-15,84	22,41
c2	-0,64	13,07

Perhitungan dilanjutkan pada iterasi berikutnya hingga terdapat kondisi konvergen yang mana tidak ada data berpindah dari kelas *cluster*. *Centroid* yang digunakan pada iterasi ke 1 sesuai pada Tabel 4.30 dan hasil perhitungan jarak iterasi ke 1 terdapat pada Tabel 4.31.

Tabel 4.30 *Centroid* Iterasi ke 1 Segmentasi Penyakit

Centroid	a*	b*
c1	-15,84	22,41
c2	-0,64	13,07

Tabel 4.31 Jarak *Cluster* Segmentasi Penyakit Iterasi ke 1

Index	a*	b*	*distance* c1	*distance* c2	*cluster*
(1,1)	-15,26	24,19	0,00	18,37	c1
(2,0)	-16,52	20,55	3,85	17,56	c1
(2,1)	-0,64	13,07	18,37	0,00	c2
(2,2)	-15,61	23,00	1,24	17,96	c1
(3,1)	-15,95	21,91	2,38	17,68	c1

Pada hasil perhitungan jarak iterasi ke 1 yang ditampilkan pada Tabel 4.31, menunjukkan tidak ada kelas *cluster* yang berpindah dari iterasi sebelumnya sehingga kondisi ini dapat dikatakan konvergen. Hasil kesimpulan algoritme *K-Means* ditunjukkan pada Tabel 4.32.

Tabel 4.32 Hasil *K-Means* Pada Segmentasi Penyakit

Index	L*	a*	b*	*cluster*
(1,1)	57,95	-15,26	24,19	c1
(2,0)	61,00	-16,52	20,55	c1
(2,1)	53,31	-0,64	13,07	c2
(2,2)	58,90	-15,61	23,00	c1
(3,1)	59,82	-15,95	21,91	c1

4.2.3 Manualisasi Klasifikasi Penyakit Daun Jeruk

Perhitungan untuk melakukan klasifikasi terhadap penyakit daun jeruk menggunakan algoritme K-NN. Ada beberapa proses pada perhitungan K-NN yang pertama menghitung jarak data *input* (rata-rata nilai komponen L*a*b* tiap *cluster*) dengan setiap data latih yang ada. Kemudian data jarak tiap-tiap *cluster* diurutkan mulai dari yang terdekat hingga yang terjauh. Setelah diurutkan data jarak terdekat diambil sebanyak nilai K yang selanjutnya dilakukan proses *voting* untuk pengambilan keputusannya. Nilai K yang digunakan pada proses ini adalah 2.

Data *input* atau data yang diujikan untuk perhitungan K-NN didapat melalui rata-rata setiap komponen ruang warna L*a*b* dari tiap *cluster* hasil segmentasi

penyakit yang ditunjukkan pada Tabel 4.32. Untuk nilai variabel a* dan b* dapat diambil melalui hasil perhitungan *centroid* terakhir dari segmentasi penyakit. Sedangkan untuk nilai L* didapatkan dengan menghitung rata-rata seluruh nilai L* pada *cluster* yang sama. Berikut contoh perhitungan rata-rata nilai L* pada *cluster* 1:

$$L^*_{c1} = \frac{(57.95 + 61.00 + 58.90 + 59.82)}{4} = 59.43$$

Hasil perhitungan rata-rata nilai L* pada *cluster* 1 adalah 59.43 dan untuk seluruh nilai data uji dapat dilihat pada Tabel 4.33.

Tabel 4.33 Data Uji K-NN Untuk Klasifikasi Penyakit

Rata-rata	L*	a*	b*
c1	59,42	-15,84	22,41
c2	53,31	-0,64	13,07

Untuk data latih pada perhitungan manual ini diambil sebanyak 8 contoh data. Data berisi informasi komponen warna L*a*b* dan label dari penyakitnya. Data latih pada proses perhitungan K-NN ini dapat dilihat pada Tabel 4.34.

Tabel 4.34 Data Latih Untuk Klasifikasi Penyakit

Data Latih	L*	a*	b*	Klasifikasi
data 1	68,48	-17,59	10,63	Daun Sehat
data 2	67,33	-17,33	16,25	Daun Sehat
data 3	58,86	-0,94	9,68	Cendawan Jelaga
data 4	63,72	-0,30	4,72	Cendawan Jelaga
data 5	80,53	-34,06	54,65	CVPD
data 6	85,12	-26,56	38,32	CVPD
data 7	82,03	-14,22	-0,88	*Downy Mildew*
data 8	83,56	-9,10	-7,30	*Downy Mildew*

Selanjutnya adalah proses untuk melakukan perhitungan data uji masing-masing *cluster* dengan seluruh data latih yang ada. Proses perhitungan jarak menggunakan persamaan *Euclidean distance*. Berikut contoh dari perhitungan jarak data uji dan salah satu data latih pada *cluster* 1:

$$d_{c1-data1} = \sqrt{(59.42 - 68.48)^2 + ((-15.84) - (-17.59))^2 + (22.41 - 10.63)^2} = 14.94$$

Hasil perhitungan jarak antara *cluster* 1 dengan data 1 adalah 14.94. Untuk hasil perhitungan jarak *cluster* 1 dengan seluruh data latih dapat dilihat pada Tabel

4.35 sedangkan hasil perhitungan jarak *cluster* 2 dengan seluruh data latih dapat dilihat pada Tabel 4.36.

Tabel 4.35 Hasil Perhitungan Jarak Data Latih *Cluster* 1 Untuk Klasifikasi Penyakit

Data Latih	Jarak	Klasifikasi
data 1	14,97	Daun Sehat
data 2	10,14	Daun Sehat
data 3	19,60	Cendawan Jelaga
data 4	23,93	Cendawan Jelaga
data 5	42,63	CVPD
data 6	32,07	CVPD
data 7	32,50	*Downy Mildew*
data 8	38,87	*Downy Mildew*

Tabel 4.36 Hasil Perhitungan Jarak Data Latih *Cluster* 2 Untuk Klasifikasi Penyakit

Data Latih	Jarak	Klasifikasi
data 1	22,88	Daun Sehat
data 2	22,02	Daun Sehat
data 3	6,51	Cendawan Jelaga
data 4	13,34	Cendawan Jelaga
data 5	59,89	CVPD
data 6	48,18	CVPD
data 7	34,69	*Downy Mildew*
data 8	37,43	*Downy Mildew*

Setelah perhitungan jarak dengan seluruh data latih kemudian data latih diurutkan mulai dari jarak yang terdekat hingga jarak yang terjauh. Untuk *cluster* 1 pengurutan data dapat dilihat pada Tabel 4.37 sedangkan *cluster* 2 dapat dilihat melalui Tabel 4.38. Setelah dilakukan pengurutan lalu diambil sebanyak K nilai teratas. Karena K=2 jadi, pada perhitungan kali ini diambil sebanyak 2 data teratas.

Tabel 4.37 *Voting* K-NN *Cluster* 1 Untuk Klasifikasi Penyakit

Data Latih	Jarak	Klasifikasi
data 2	10,14	Daun Sehat
data 1	14,97	Daun Sehat
data 3	19,60	Cendawan Jelaga
data 4	23,93	Cendawan Jelaga
data 6	32,07	CVPD
data 7	32,50	*Downy Mildew*
data 8	38,87	*Downy Mildew*
data 5	42,63	CVPD

Tabel 4.38 *Voting* K-NN *Cluster* 2 Untuk Klasifikasi Penyakit

Data Latih	Jarak	Klasifikasi
data 3	6,51	Cendawan Jelaga
data 4	13,34	Cendawan Jelaga
data 2	22,02	Daun Sehat
data 1	22,88	Daun Sehat
data 7	34,69	*Downy Mildew*
data 8	37,43	*Downy Mildew*
data 6	48,18	CVPD
data 5	59,89	CVPD

Hasil pengurutan data pada Tabel 4.33 dan Tabel 4.34. Dapat disimpulkan bahwa data dua teratas yang diambil untuk data *cluster* 1 menghasilkan kesimpulan klasifikasi berupa daun sehat sedangkan untuk data *cluster* 2 menghasilkan kesimpulan klasifikasi berupa penyakit jelaga. Hasil klasifikasi penyakit daun jeruk dapat dilihat pada Tabel 4.39.

Tabel 4.39 Hasil Klasifikasi Penyakit Daun Jeruk

Index	L*	a*	b*	Klasifikasi
(1,1)	57,95	-15,26	24,19	Daun Sehat
(2,0)	61,00	-16,52	20,55	Daun Sehat
(2,1)	53,31	-0,64	13,07	Cendawan Jelaga
(2,2)	58,90	-15,61	23,00	Daun Sehat
(3,1)	59,82	-15,95	21,91	Daun Sehat

Pada akhir proses perhitungan ini memberikan kesimpulan secara garis besar bahwa daun jeruk terkena penyakit jelaga. Apabila pada saat perhitungan klasifikasi menggunakan K-NN menghasilkan dua jenis penyakit selain daun sehat, maka seluruh *cluster* yang terkena penyakit dijadikan satu untuk dilakukan perhitungan rata-rata komponen nilai L*a*b* dan perhitungan klasifikasi menggunakan algoritme K-NN kembali. Perhitungan kembali yang dilakukan bertujuan untuk menghasilkan suatu klasifikasi yang lebih akurat dengan hasil satu daun hanya memiliki satu jenis penyakit.

Pengecekan selanjunya antara hasil klasifikasi yang telah dilakukan dengan target *output*. Jika hasil klasifikasi dengan target *output* sama maka status bernilai benar dan sebaliknya jika hasil klasifikasi dengan target *output* berbeda maka status bernilai salah. Dengan menggunakan referensi hasil klasifikasi pada Tabel 4.25 dan Tabel 4.39 dapat dituliskan menjadi suatu perbandingan antara hasil klasifikasi dan target *output* yang ditunjukkan pada Tabel 4.40.

Tabel 4.40 Perbandingan Hasil Klasifikasi Dengan Target *Output*

Index	R	G	B	Hasil Klasifikasi	Target *Output*	Status
(0,0)	249	113	153	*Cover*	*Cover*	Benar
(0,1)	251	116	156	*Cover*	*Cover*	Benar
(0,2)	254	119	159	*Cover*	*Cover*	Benar
(1,0)	246	110	150	*Cover*	*Cover*	Benar
(1,1)	50	66	27	Daun Sehat	Daun Sehat	Benar
(1,2)	251	116	156	*Cover*	*Cover*	Benar
(2,0)	53	75	36	Daun Sehat	Daun Sehat	Benar
(2,1)	56	49	33	Cendawan Jelaga	Cendawan Jelaga	Benar
(2,2)	51	69	30	Daun Sehat	Daun Sehat	Benar
(3,0)	243	107	147	*Cover*	*Cover*	Benar
(3,1)	52	72	33	Daun Sehat	Daun Sehat	Benar
(3,2)	248	112	152	*Cover*	*Cover*	Benar

4.3 Perancangan Pengujian

Pada penelitian ini menggunakan pengujian guna mendapatkan parameter terbaik untuk mendukung program. Ada 3 pengujian yang dijalankan yaitu pengujian *Scale Factor*, pengujian nilai *cluster* optimal menggunakan metode *Silhouette Coefficient*, dan pengujian k optimal untuk K-NN.

4.3.1 Perancangan Skenario Pengujian *Scale Factor*

Uji coba mendapatkan perbandingan nilai *Scale Factor* digunakan untuk melihat nilai *Scale Factor* terbaik yang dibandingkan dengan nilai akurasi untuk masing-masing parameternya. Digunakan lima nilai parameter untuk *Scale Factor* yakni 1.1, 1.2, 1.3, 1.4, dan 1.5. Pada tabel pengujian ditunjukkan kolom jumlah klasifikasi yang benar, jumlah klasifikasi yang salah, dan tingkat akurasi masing-masing nilai *Scale Factor*. Tabel Skenario Pengujian *Scale Factor* dapat dilihat melalui Tabel 4.41.

Tabel 4.41 Skenario Pengujian *Scale Factor*

No	Nilai *Scale Factor*	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
1	1.1			
2	1.2			
3	1.3			
4	1.4			
5	1.5			
6	1.6			
7	1.7			
8	1.8			
9	1.9			
10	2.0			

4.3.2 Perancangan Skenario Pengujian Nilai *Cluster* Optimal

Uji Coba menggunakan metode *Silhouette Coefficient* untuk mendapatkan jumlah *cluster* terbaik pada penggunaan algoritme *K-Means*. Terdapat dua tipe pengujian nilai *cluster* optimal pada penelitian ini yaitu penggunaan algoritme *K-Means* pada segmentasi daun dan penggunaan algoritme *K-Means* pada segmentasi penyakit. Pengujian tahap ini menggunakan nilai jumlah *cluster* yaitu 2 sampai dengan 10. Masing-masing parameter nilai jumlah *cluster* dibandingkan melalui tabel perbandingan nilai *Silhouette Coefficient*. Pada tabel tersebut, pengambilan keputusan jumlah *cluster* optimal dapat dilihat melalui nilai *Silhouette Coefficient* terbesar. Tabel perbandingan nilai *Silhouette Coefficient* ditunjukkan pada Tabel 4.42.

Tabel 4.42 Perbandingan *Silhouette Coefficient* Tiap Parameter Jumlah *Cluster*

No	Jumlah *Cluster*	Nilai *Silhouette Coefficient*
1	2	
.	.	
.	.	
.	.	
9	10	

4.3.3 Perancangan Skenario Pengujian Nilai K Optimal Pada K-NN

Pada skenario pengujian nilai k optimal bertujuan untuk mendapatkan nilai k terbaik sebagai acuan algoritme K-NN. Digunakan lima nilai parameter untuk skenario pengujian nilai k optimal pada penelitian ini yakni 1, 2, 3, 4, dan 5. Pada tabel pengujian ditunjukkan kolom jumlah klasifikasi benar, jumlah klasifikasi salah, dan nilai akurasi dari tiap parameter. Nilai akurasi nantinya digunakan sebagai pembanding untuk masing-masing parameter guna ditentukan nilai k terbaiknya. Tabel skenario pengujian nilai k optimal pada K-NN ditunjukkan pada Tabel 4.43.

Tabel 4.43 Skenario Pengujian Nilai K Optimal Pada K-NN

No	Nilai K	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
1	1			
.	.			
.	.			
.	.			
21	20			

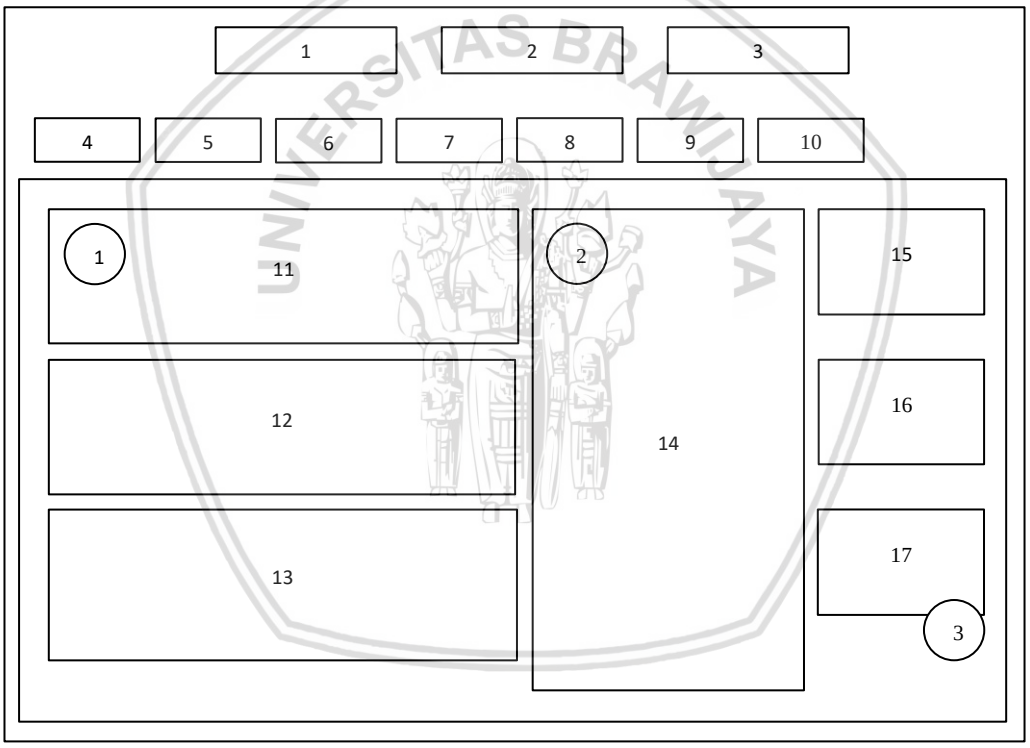
4.4 Perancangan Antarmuka

Perancangan antarmuka digunakan untuk mempermudah peneliti dalam mengimplementasikan program yang dibuat. Pada perancangan antarmuka

terdapat beberapa bagian yakni halaman awal, halaman pengujian *Scale Factor*, halaman pengujian nilai *cluster* optimal, halaman pengujian nilai k optimal pada K-NN, halaman pemrosesan data latih, halaman pemrosesan satu data uji, dan halaman pemrosesan banyak data uji.

4.4.1 Halaman Awal

Halaman awal berisi informasi tentang penyakit jeruk yang diteliti berupa deskripsi dan gambar yang berfungsi agar pengguna program mengetahui macam-macam penyakit daun jeruk yang diujikan dan juga informasi seputar program. Terdapat tiga bagian utama pada halaman awal yang pertama adalah bagian untuk menunjukkan deskripsi masing-masing penyakit daun jeruk, yang kedua adalah bagian yang menjelaskan deskripsi program, dan bagian yang terakhir berisi identitas pembimbing dan mahasiswa. Pada bagian yang berisi deskripsi masing-masing penyakit ditampilkan contoh gambarnya beserta deskripsi umumnya. Perancangan antarmuka halaman awal dapat dilihat pada Gambar 4.15.



Gambar 4.15 Perancangan Antarmuka Halaman Awal

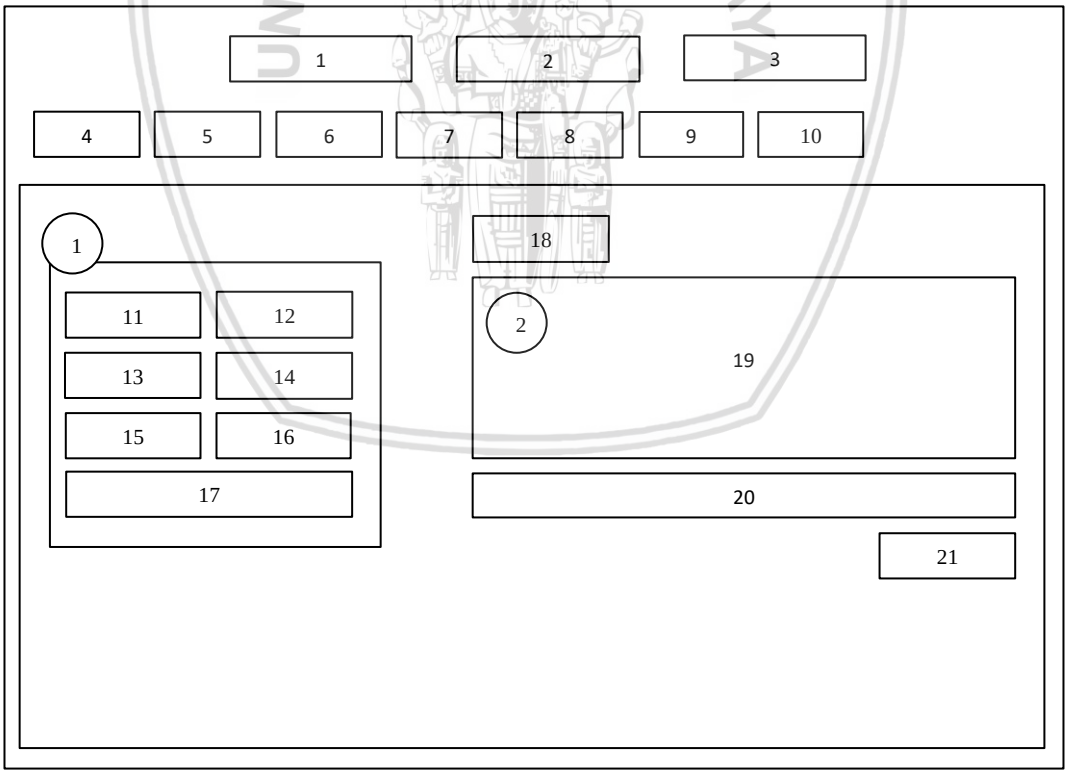
Keterangan gambar:

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*

6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Informasi Penyakit Daun CVPD
12. Informasi Penyakit Daun Cendawan Jelaga
13. Informasi Penyakit Daun *Downy Mildew*
14. Informasi Deskripsi Sistem
15. Identitas Mahasiswa
16. Identitas Pembimbing 1
17. Identitas Pembimbing 2

4.4.2 Halaman Pengujian *Scale Factor*

Perancangan antarmuka halaman pengujian *Scale Factor* dapat dilihat melalui Gambar 4.16.



Gambar 4.16 Perancangan Antarmuka Halaman Pengujian *Scale Factor*

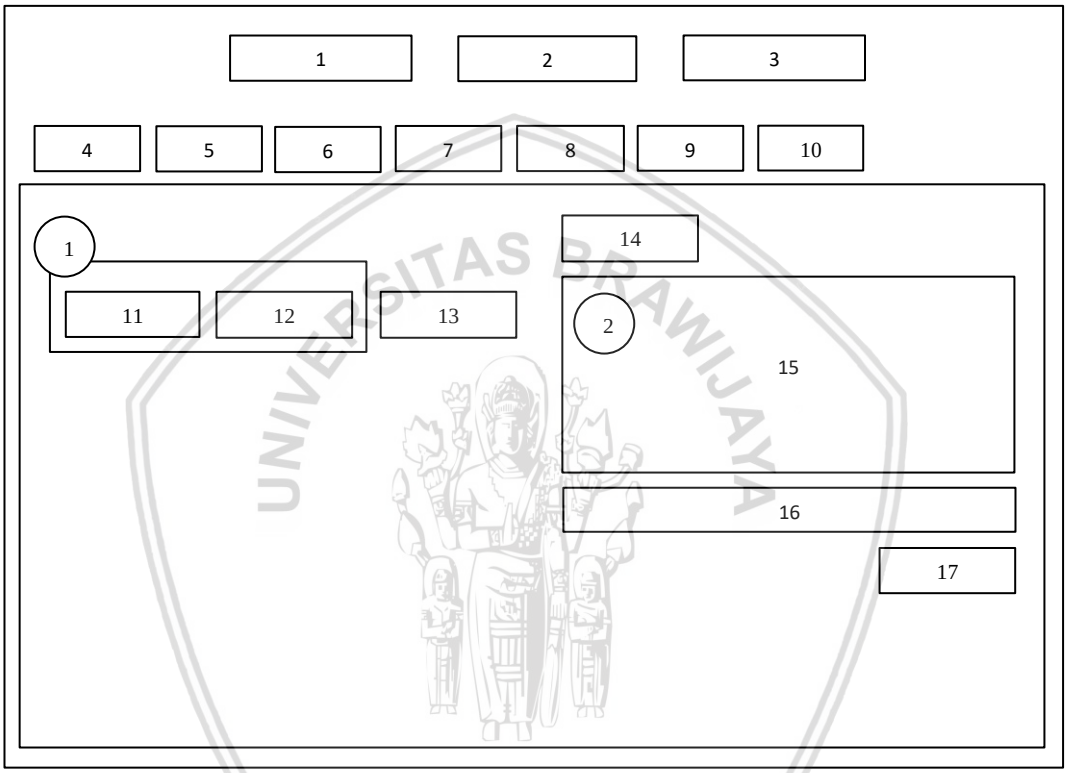
Keterangan gambar:

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jumlah *Cluster* Segmentasi Daun
12. *Textbox* Jumlah *Cluster* Segmentasi Daun
13. Label Jumlah *Cluster* Segmentasi Penyakit
14. *Textbox* Jumlah *Cluster* Segmentasi Penyakit
15. Label Jumlah K Proses K-NN
16. *Textbox* Jumlah K Proses K-NN
17. *Button* Mulai Proses
18. Label Nama Tabel Pengujian *Scale Factor*
19. Tabel Pengujian *Scale Factor*
20. *Progress Bar* Pengujian *Scale Factor*
21. Label Hasil Nilai *Scale Factor* Terbaik

Halaman pengujian *Scale Factor* menampilkan informasi nilai parameter *Scale Factor* terbaik. Nilai *Scale Factor* digunakan untuk proses *rescaling* pada citra. Pada halaman ini terdapat dua bagian utama yakni bagian *input* parameter dan bagian informasi hasil pengujian *Scale Factor*. Terdapat tiga *input*-an user yakni jumlah *cluster* untuk segmentasi daun, jumlah *cluster* untuk segmentasi penyakit, dan nilai K untuk klasifikasi menggunakan algoritme K-NN. Pada bagian informasi hasil pengujian *Scale Factor*, berisi tabel pengujian untuk membandingkan nilai parameter *Scale Factor* terbaik. Terdapat lima parameter untuk pengujian *Scale Factor* pada tabel dengan nilai *Scale Factor* 1.1, 1.2, 1.3, 1.4, dan 1.5. Pada tabel nantinya juga ditampilkan hasil akurasi untuk setiap parameter percobaan yang nantinya berfungsi sebagai acuan untuk menentukan nilai *Scale Factor* terbaik.

4.4.3 Halaman Pengujian Nilai *Cluster* Optimal

Perancangan antarmuka halaman pengujian nilai *cluster* optimal menampilkan informasi untuk menentukan jumlah *cluster* terbaik untuk algoritme *K-Means*. Pada halaman ini memiliki dua bagian utama yaitu bagian *input* parameter dan bagian informasi hasil perhitungan nilai *cluster* optimal. Pada pagian *input* parameter disediakan *comboBox* untuk memilih kategori percobaan pengujian. Sedangkan pada bagian informasi hasil pengujian disediakan tabel pengujian dan informasi jumlah *cluster* terbaik. Perancangan antarmuka Halaman Pengujian Nilai *Cluster* Optimal ditunjukkan pada Gambar 4.17.



Gambar 4.17 Perancangan Antarmuka Halaman Pengujian Nilai *Cluster* Optimal

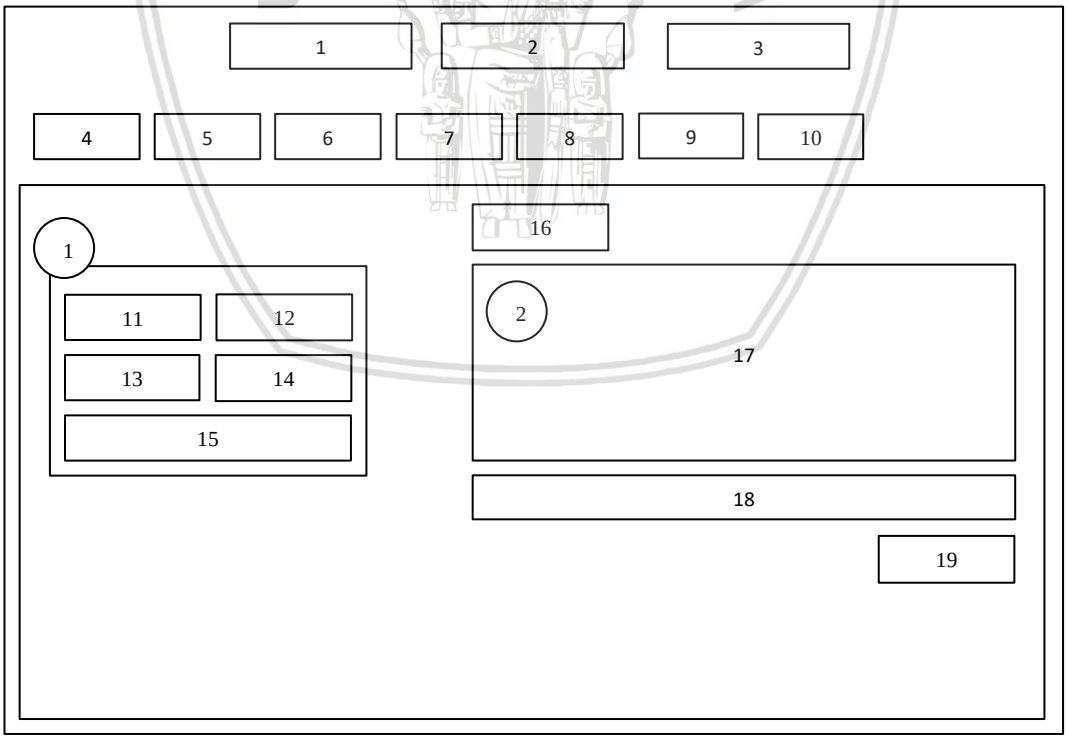
Keterangan gambar :

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih

9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jenis Pengujian Nilai *Cluster* Optimal
12. *ComboBox* Jenis Pengujian nilai *cluster* optimal
13. *Button* Mulai Proses
14. Label Nama Tabel Perbandingan *Silhouette Coefficient*
15. Tabel Perbandingan *Silhouette Coefficient*
16. *Progress Bar* Pengujian Nilai *Cluster* Optimal
17. Label Hasil Nilai *Cluster* Terbaik

4.4.4 Halaman Pengujian Nilai K Optimal Pada K-NN

Perancangan antarmuka pengujian nilai k optimal menampilkan informasi untuk mendapatkan nilai k terbaik pada algoritme K-NN. Pada bagian pertama, terdapat dua *input*-an user yakni jumlah *cluster* untuk segmentasi daun dan jumlah *cluster* untuk segmentasi penyakit. Sedangkan pada bagian kedua, disediakan tabel pengujian untuk membandingkan nilai akurasi masing-masing parameter dan informasi seputar nilai k terbaik hasil perbandingan. Perancangan antarmuka halaman pengujian nilai k optimal pada K-NN dapat dilihat melalui Gambar 4.18.



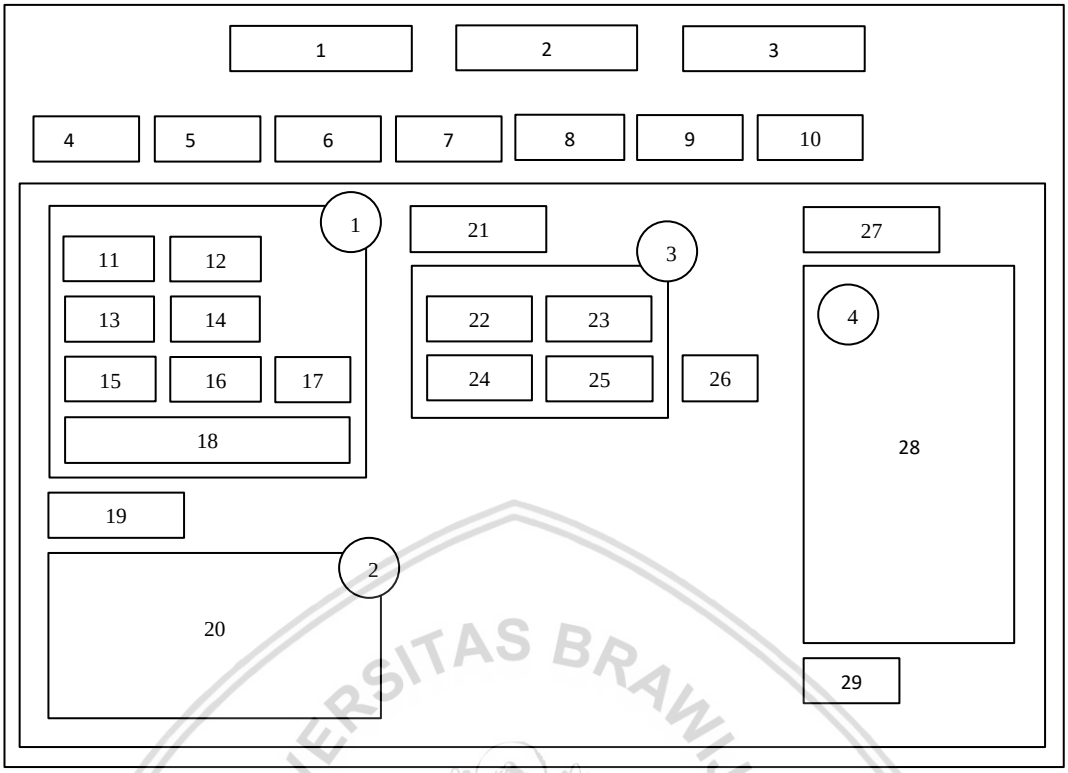
Gambar 4.18 Perancangan Antarmuka Halaman Pengujian Nilai K Optimal

Keterangan gambar :

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jumlah *Cluster* Segmentasi Daun
12. *Textbox* Jumlah *Cluster* Segmentasi Daun
13. Label Jumlah *Cluster* Segmentasi Penyakit
14. *Textbox* Jumlah *Cluster* Segmentasi Penyakit
15. *Button* Mulai Proses
16. Label Nama Tabel Pengujian K Optimal
17. Tabel Pengujian K Optimal
18. *Progress Bar* Pengujian K Optimal
19. Label Hasil Nilai K Terbaik

4.4.5 Halaman Pemrosesan Data Latih

Halaman pemrosesan data latih digunakan untuk melakukan pelatihan terhadap data yang tersedia. Terdapat empat bagian utama pada halaman pemrosesan data latih. Yang pertama bagian *input* yang berisi *input* jumlah *cluster* segmentasi segmentasi daun, jumlah *cluster* segmentasi segmentasi daun, dan citra daun. Bagian kedua berfungsi untuk menampilkan informasi fitur masing-masing *cluster* yang dituliskan melalui tabel. Bagian ketiga memiliki fungsi untuk melakukan *insert* terhadap data latih terbaru dengan memilih *cluster* yang ada serta melakukan pelabelan jenis penyakit *cluster* yang dipilih secara manual. Bagian terakhir untuk menampilkan informasi data latih yang sudah tersedia beserta *Button* untuk melakukan update terhadap tabel data latih. Perancangan antarmuka halaman pemrosesan data latih dapat dilihat melalui Gambar 4.19.



Gambar 4.19 Perancangan Antarmuka Halaman Pemrosesan Data Latih

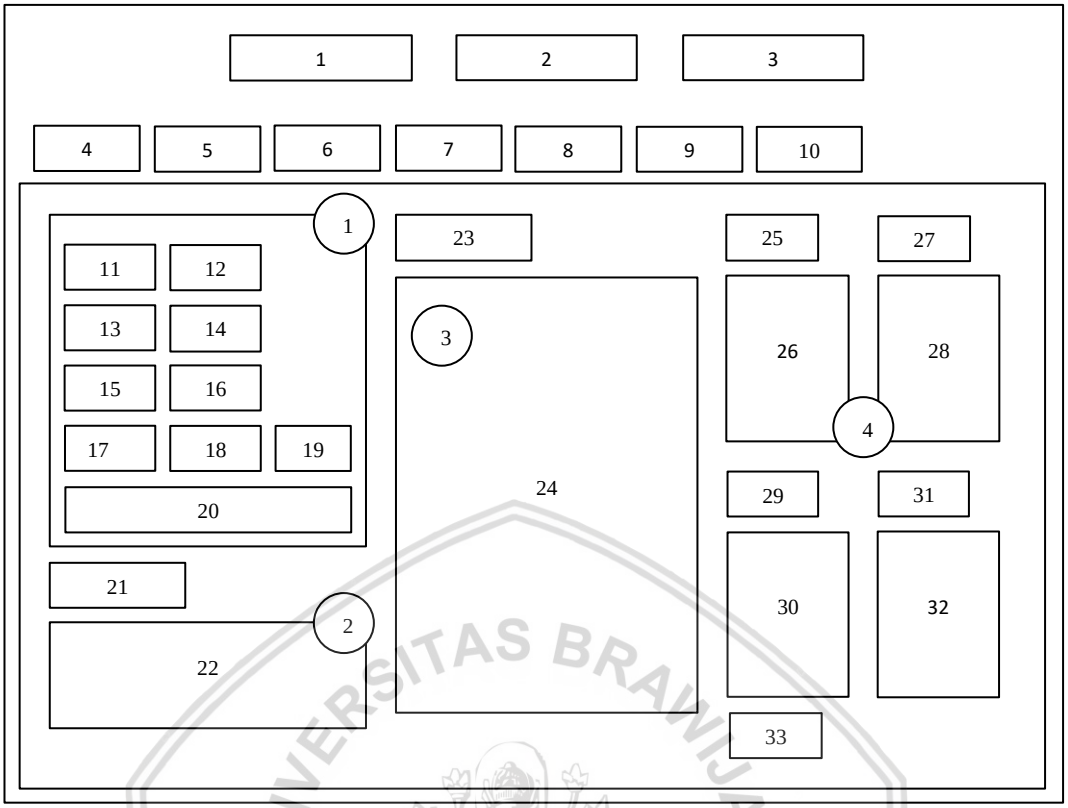
Keterangan gambar:

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jumlah *Cluster* Segmentasi Daun
12. *Textbox* Jumlah *Cluster* Segmentasi Daun
13. Label Jumlah *Cluster* Segmentasi Penyakit
14. *Textbox* Jumlah *Cluster* Segmentasi Penyakit
15. Label Pilih Citra *Input*

16. *Textbox* Lokasi Citra *Input*
17. *Button* Cari Lokasi Citra *Input*
18. *Button* Mulai Proses
19. Label Nama Tabel Hasil Segmentasi Peyakit
20. Tabel Hasil Segmentasi Peyakit
21. Label Proses Pelabelan Data Latih
22. Label Pilih *Cluster*
23. *ComboBox* Pilih *Cluster*
24. Label Pilih Jenis Penyakit
25. *ComboBox* Pilih Jenis Penyakit
26. *Button Insert* Data Latih
27. Label Nama Tabel Data Latih
28. Tabel Data Latih
29. *Button* Update Tabel Data Latih

4.4.6 Halaman Pemrosesan Satu Data Uji

Halaman pemrosesan satu data uji berfungsi untuk melakukan pengujian dari satu data uji guna diklasifikasikan jenis penyakitnya. Terdapat empat bagian utama pada halaman ini yaitu bagian *input* parameter, bagian informasi hasil klasifikasi tiap *cluster*, bagian informasi data latih, dan bagian hasil akhir pemrosesan data uji. Bagian pertama berisi empat parameter *input* yaitu *input* jumlah *cluster* untuk segmentasi daun, jumlah *cluster* segmentasi penyakit, nilai K untuk proses klasifikasi menggunakan algoritme K-NN, dan citra daun jeruk. Pada bagian kedua yang merupakan bagian informasi hasil klasifikasi tiap *cluster* digambarkan melalui tabel yang berisi fitur tiap *cluster* dan jenis penyakitnya. Pada tabel hasil klasifikasi tiap *cluster* dihasilkan melalui perhitungan pemrosesan data uji yang menampilkan beberapa fitur yaitu nama *cluster*, nilai rata-rata L*a*b* tiap *cluster*, dan klasifikasi dari jenis penyakit tiap *cluster*. Bagian ketiga menampilkan informasi data latih yang tersedia melalui pemrosesan data latih sebelumnya. Bagian yang terakhir menampilkan informasi hasil akhir pemrosesan data uji yang berisi empat gambar yaitu gambar citra *input*, gambar hasil segmentasi daun, gambar bagian daun, dan gambar bagian penyakit. Pada bagian keempat ini juga menampilkan label untuk menujukkan hasil kesimpulan akhir dari pemrosesan satu data uji. Perancangan antarmuka halaman Pemrosesan satu data uji dapat dilihat melalui Gambar 4.21.



Gambar 4.20 Peracangan Antarmuka Halaman Pemrosesan Satu Data Uji

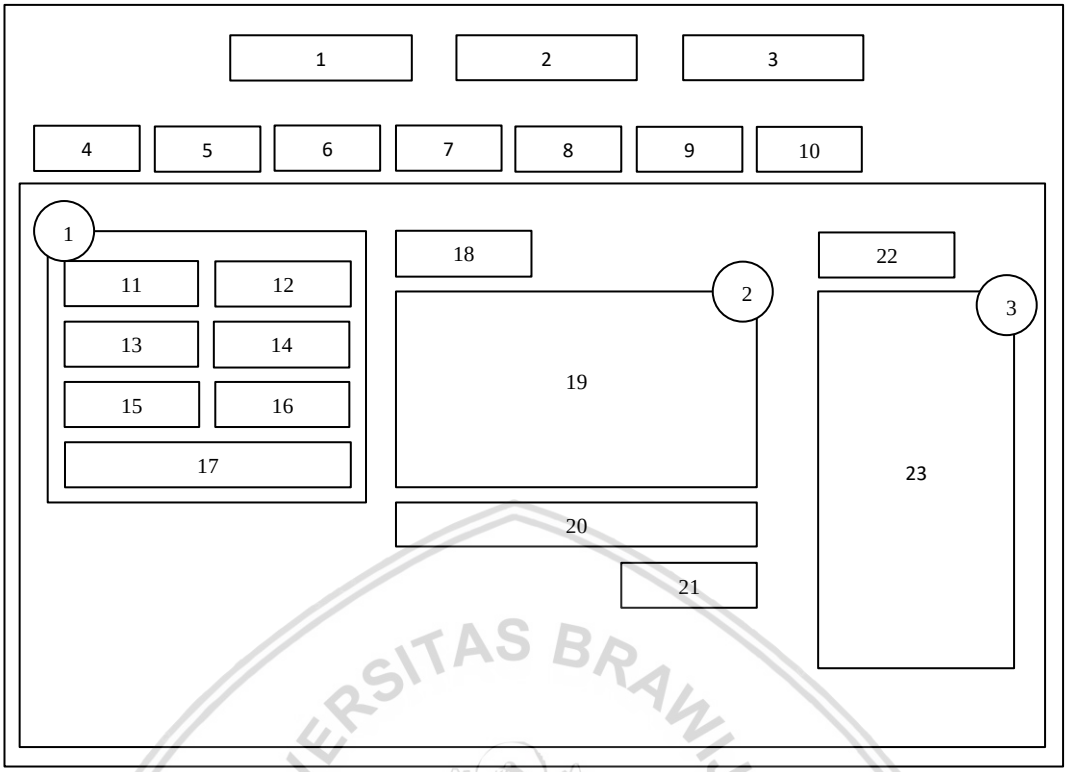
Keterangan gambar:

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jumlah *Cluster* Segmentasi Daun
12. *Textbox* Jumlah *Cluster* Segmentasi Daun
13. Label Jumlah *Cluster* Segmentasi Penyakit
14. *Textbox* Jumlah *Cluster* Segmentasi Penyakit
15. Label Jumlah K Proses K-NN

16. *Textbox* Jumlah K Proses K-NN
17. Label Pilih Citra *Input*
18. *Textbox* Lokasi Citra *Input*
19. *Button* Cari Lokasi Citra *Input*
20. *Button* Mulai Proses
21. Label Nama Tabel Hasil Klasifikasi Tiap *Cluster*
22. Tabel Hasil Klasifikasi Tiap *Cluster*
23. Label Nama Tabel Data Latih
24. Tabel Data Latih
25. Label Gambar Citra *Input*
26. Gambar Citra *Input*
27. Label Gambar Hasil Segmentasi Daun
28. Gambar Hasil Segmentasi Daun
29. Label Gambar Bagian Daun
30. Gambar Bagian Daun
31. Label Gambar Bagian Penyakit
32. Gambar Bagian Penyakit
33. Label Hasil Akhir Klasifikasi Data Uji

4.4.7 Halaman Pemrosesan Banyak Data Uji

Halaman pemrosesan banyak data uji berfungsi untuk melakukan pengujian data uji terhadap data latih dengan banyak data uji sekaligus. Pada halaman ini terdapat tiga bagian utama yaitu bagian *input* parameter, bagian informasi pemrosesan banyak data uji, dan bagian informasi data latih. Bagian yang pertama berisi tiga *input*-an parameter yaitu *input* jumlah *cluster* untuk segmentasi daun, jumlah *cluster* untuk segmentasi penyakit, dan nilai K untuk proses klasifikasi menggunakan algoritme K-NN. Bagian kedua merupakan bagian untuk menunjukkan informasi terkait pemrosesan banyak data uji. Terdapat tabel hasil klasifikasi tiap data beserta target *output*-nya sehingga dapat dibandingkan ketepatan program untuk melakukan klasifikasi terhadap penyakit daun jeruk. Pada bagian kedua juga menampilkan label yang berfungsi untuk mengetahui akurasi program dari pemrosesan beberapa data uji yang telah dilakukan. Bagian yang ketiga merupakan bagian untuk menampilkan informasi data latih. Data latih dituliskan melalui tabel yang nilainya didapatkan melalui pemrosesan data latih sebelumnya. Pada tabel terdapat beberapa fitur yaitu nama data, L*a*b* tiap data, dan jenis penyakit tiap data. Perancangan antarmuka halaman pemrosesan banyak data uji dapat dilihat pada Gambar 4.20.



Gambar 4.21 Perancangan Antarmuka Halaman Pemrosesan Banyak Data Uji

Keterangan gambar:

1. Logo Universitas Brawijaya
2. Logo Fakultas Ilmu Komputer
3. Logo Kementerian Pertanian/Balitjestro
4. *Navigation* Halaman Awal
5. *Navigation* Halaman Pengujian *Scale Factor*
6. *Navigation* Halaman Pengujian Nilai *Cluster* Optimal
7. *Navigation* Halaman Pengujian Nilai K Optimal Pada K-NN
8. *Navigation* Halaman Pemrosesan Data Latih
9. *Navigation* Halaman Pemrosesan Satu Data Uji
10. *Navigation* Halaman Pemrosesan Banyak Data Uji
11. Label Jumlah *Cluster* Segmentasi Daun
12. *Textbox* Jumlah *Cluster* Segmentasi Daun
13. Label Jumlah *Cluster* Segmentasi Penyakit
14. *Textbox* Jumlah *Cluster* Segmentasi Penyakit
15. Label Jumlah K Proses K-NN

16. *Textbox* Jumlah K Proses K-NN
17. *Button* Mulai Proses
18. Label Nama Tabel Hasil Klasifikasi dan Target *Output*
19. Tabel Hasil Klasifikasi dan Target *Output*
20. *Progress Bar* Klasifikasi
21. Label Akurasi Program
22. Label Nama Tabel Data Latih
23. Tabel Data Latih

BAB 5 IMPLEMENTASI

Pada bab ini berisikan implementasi dari perancangan yang telah disusun sebelumnya. Adapun beberapa bagian implementasi yang dibahas pada bab ini antara lain yaitu lingkungan implementasi, batasan implementasi, implementasi kode program, dan implementasi antarmuka.

5.1 Lingkungan Implementasi

Dalam membangun program dari perancangan yang telah disusun dibutuhkan bebera perangkat pendukung baik perangkat keras atau perangkat lunak. Pada bagian lingkungan implementasi kali ini terdapat dua bagian yang dibahas yaitu lingkungan implementasi perangkat keras dan lingkungan implementasi perangkat lunak.

5.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan serta untuk memenuhi kebutuhan program selama proses implementasi dari perancangan yang telah dibuat antara lain:

1. *Processor Intel Core* i7-4720HQ CPU @ 2.60 GHz
2. *Memory* RAM 8 GB
3. *Harddisk* 500 GB
4. *Monitor* Laptop 15.6 inch
5. *Keyboard Internal* Laptop
6. *Mouse/Touchpad Internal* Laptop
7. *Camera smartphone Asus Zenfone Selfie* ZD551KL 13 MP

5.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan serta untuk memenuhi kebutuhan program selama proses implementasi dari perancangan yang telah dibuat antara lain:

1. *Microsoft Office Word* 2016, digunakan untuk menulis dokumentasi atau laporan dari penelitian.
2. *Microsoft Office Excel* 2016, digunakan untuk mengitung manualisasi tiap langkah dalam penggunaan metode pada penelitian.
3. *Microsoft Office Power Point* 2016, digunakan untuk membuat media yang digunakan untuk mempresentasikan *progress* ataupun hasil penelitian.
4. Aplikasi *mobile Camera*, digunakan untuk mengambil gambar citra daun jeruk di Lapangan.
5. Aplikasi *mobile Camscanner*, digunakan untuk mengolah hasil gambar citra daun jeruk untuk dilakukan *cropping* sesuai ukuran *cover*.

6. Microsoft Photos, digunakan untuk pengolahan citra daun jeruk tahap lanjut untuk dilakukan *cropping* disesuaikan dengan ukuran perbandingan yang sama.
7. *Netbeans* IDE 8.2, sebagai *editor* bahasa pemrograman untuk menuliskan serta membangun program pada tahap implementasi penelitian.

5.2 Batasan Implementasi

Pada sub bab ini dijelaskan mengenai batasan-batasan dalam mengimplementasikan program untuk penelitian implementasi algoritme *K-Means* sebagai metode segmentasi citra pada daun jeruk untuk proses Identifikasi penyakit tanaman jeruk. Batasan implementasi pada penelitian ini antara lain:

1. Jumlah data yang digunakan sebanyak 120 data yang terbagi menjadi 4 kelas data yaitu daun sehat, daun berpenyakit *Downy Mildew*, daun berpenyakit Cendawan Jelaga, dan daun berpenyakit CVPD (*Citrus Vein Phloem Degeneration*). Data citra daun jeruk diperoleh dari diambil pada Balai Penelitian Tanaman Jeruk dan Buah Subtropika (Balitjestro) yang berada di daerah Tlekung kota Batu.
2. Pengambilan data citra daun jeruk dilakukan dengan memetik daun dari pohonnya kemudian dijepit menggunakan alat bantu *cover* berwarna merah.
3. Pengambilan citra daun jeruk dilakukan ditempat terbuka dengan pencahayaan yang cukup untuk memperlihatkan bagian daun.
4. Jumlah citra daun yang digunakan untuk pemrosesan adalah satu citra daun.
5. Parameter ukuran citra yang digunakan pada *pre-processing* tahap *resizing* adalah 375x500 *pixel*.
6. Jumlah kelas jenis kondisi daun yang digunakan adalah 4 kelas yang terdiri dari:
   a. Kelas 0 (nol), merupakan daun sehat.
   b. Kelas 1 (satu), merupakan daun berpenyakit *Downy Mildew*.
   c. Kelas 2 (dua), merupakan daun berpenyakit Cendawan Jelaga.
   d. Kelas 3 (tiga), merupakan daun berpenyakit CVPD (*Citrus Vein Phloem Degeneration*).
7. Kondisi jika pada saat pengujian suatu citra daun jeruk gagal hasil proses dilabelkan pada kelas baru yaitu kelas 4 (empat).

5.3 Implementasi Kode Program

Pada sub bab ini dijelaskan mengenai implementasi kode program sesuai dengan perancangan yang telah dibuat sebelumnya. Beberapa kode program yang dijelaskan pada sub bab ini adalah *pre-processing*, segmentasi menggunakan *K-Means*, proses pelatihan, dan proses pengujian.

5.3.1 Implementasi *Pre-processing*

Beberapa proses implementasi sesuai dengan perancangan yang telah dibuat antara lain *resizing*, *rescaling*, dan mengubah ruang warna dari RGB menjadi L*a*b*.

5.3.1.1 Implementasi *Resizing*

Proses *resizing* digunakan untuk melakukan pengaturan ulang terhadap ukuran citra *input* yang digunakan untuk tahap berikutnya. Beberapa parameter yang diperlukan untuk *method resizing* antara lain citra input yang berbentuk Object *BufferedImage*, ukuran lebar (w), dan ukuran tinggi (h). Sesuai dengan perancangan yang dibuat sebelumnya, implementasi *resizing* dijelasikan melalui kode program 5.1.

```
1  public BufferedImage resizeImage(BufferedImage originalImage, int w,
2  int h) {
3          BufferedImage resizedImage = new BufferedImage(w, h,
4          originalImage.getType());
5          Graphics2D g = resizedImage.createGraphics();
6          g.drawImage(originalImage, 0, 0, w, h, null);
7          g.dispose();
8          return resizedImage;
9      }
```

Kode Program 5.1 Implementasi *Resizing*

Penjelasan Kode Program 5.1 :

1. Pada baris 1-2 merupakan deklarasi method *resizeImage()* dengan parameter *BufferedImage originalImage* (citra *input*), w (ukuran lebar), dan h (ukuran tinggi).
2. Pada baris 2-3 inisialisasi *Object BufferedImage* dengan nama variabel *resizedImage* untuk menampung hasil akhir dari proses *resizing*.
3. Pada baris 5 digunakan untuk pembuatan grafik 2 dimensi dengan nama variabel g yang didapatkan dari method *createGraphics()* variabel *resizedImage*.
4. Pada baris 6 digunakan untuk menggambar ulang citra dengan parameter *originalImage* sebagai citra yang digambar ulang, w sebagai ukuran lebar terbaru, dan h sebagai ukuran tinggi terbaru.
5. Pada baris 7 memanggil method *dispose()* dari variabel g.
6. Baris 8 digunakan untuk mengembalikan nilai citra hasil *resize*.

5.3.1.2 Implementasi *Rescaling*

Proses *rescaling* berfungsi untuk mengatur tingkat kecerahan dari citra input. Terdapat parameter input yakni citra yang diatur tingkat kecerahannya dan nilai *Scale Factor* yang dibutuhkan. Sesuai dengan perancangan yang dibuat sebelumnya, implementasi *rescaling* dapat dilihat melalui kode program 5.2.

```
1   public BufferedImage rescaleImage(BufferedImage originalImage, float
2   scaleFactor) {
3           RescaleOp rescaleOp = new RescaleOp(scaleFactor, 0, null);
4           BufferedImage rescaleImage = rescaleOp.filter(originalImage,
5           null);
6           return rescaleImage;
7       }
```

Kode Program 5.2 Implementasi *Rescaling*

Penjelasan Kode Program 5.2:

1. Pada baris 1-2 merupakan deklarasi method *rescaleImage()* dengan parameter *BufferedImage originalImage* (citra *input*) dan nilai *Scale Factor*.
2. Pada baris 3 inisialisasi nilai operator dari proses *rescale* dengan membuat melalui object *RescaleOp*. Parameter yang digunakan *scaleFactor* sesuai dengan *input* program dan *offset* di-*set* 0.
3. Pada baris 4-5 pemangggilan method *filter()* dari kelas *RescaleOp* untuk menghasilkan citra yang telah dilakukan *rescale*. Parameter yang digunakan adalah citra *input* yang diatur tingkat kecerahannya.
4. Pada baris 6 mengembalikan citra hasil *rescale*.

5.3.1.3 Implementasi RGB *to* L*a*b*

Proses mengubah ruang warna RGB menjadi L*a*b* diperlukan untuk pemrosesan algoritme *K-Means* ditahap berikutnya. Sesuai dengan perancangan yang telah dibuat implementasi mengubah ruang warna RGB menjadi L*a*b* dijelaskan melalui kode program 5.3.

```
1   public ArrayList<LAB> rgb2lab(BufferedImage originalImage) {
2           ArrayList<LAB> listLAB = new ArrayList<>();
3           for (int i = 0; i < originalImage.getWidth(); i++) {
4               for (int j = 0; j < originalImage.getHeight(); j++) {
5                   int nilaiPixel = originalImage.getRGB(i, j);
6                   float red = (nilaiPixel >> 16) & 0xff;
7                   float green = (nilaiPixel >> 8) & 0xff;
8                   float blue = (nilaiPixel) & 0xff;
9
10                  float x = (0.412453f * red) + (0.35758f * green) +
11                  (0.180423f * blue);
12                  float y = (0.212671f * red) + (0.71516f * green) +
13                  (0.072169f * blue);
14                  float z = (0.019334f * red) + (0.119193f * green) +
15                  (0.950227f * blue);
16
17                  float xNormal = x / 242.36628f;
18                  float yNormal = y / 255f;
19                  float zNormal = z / 277.63227f;
20
```

```
21                  float fx, fy, fz, l, a, b;
22                  if (xNormal > 0.008856) {
23                      fx = (float) Math.pow(xNormal, 1 / 3.);
24                  } else {
25                      fx = (float) (7.787 * xNormal + 0.1379);
26                  }
27                  if (yNormal > 0.008856) {
28                      fy = (float) Math.pow(yNormal, 1 / 3.);
29                  } else {
30                      fy = (float) (7.787 * yNormal + 0.1379);
31                  }
32                  if (zNormal > 0.008856) {
33                      fz = (float) Math.pow(zNormal, 1 / 3.);
34                  } else {
35                      fz = (float) (7.787 * zNormal + 0.1379);
36                  }
37
38                  if (yNormal > 0.008856) {
39                      l = 116 * fy - 16;
40                  } else {
41                      l = (float) 903.3 * yNormal;
42                  }
43                  a = 500 * (fx - fy);
44                  b = 200 * (fy - fz);
45
46                  listLAB.add(new LAB(i, j, l, a, b));
47              }
48          }
49
50          return listLAB;
51      }
```

Kode Program 5.3 Implementasi RGB *to* L*a*b*

Penjelasan Kode Program 5.3:

1. Pada baris 1-2 merupakan deklarasi *method rgb2lab()* dengan parameter BufferedImage *originalImage* (citra *input*).
2. Pada baris 2 melakukan inisialisasi bertipe *ArrayList<LAB>* dengan nama variabel *listLAB* yang digunakan untuk menampung hasil dari pemrosesan perubahan ruang warna.
3. Pada baris 3-4 deklarasi perulangan yang didalamnya terdapat proses untuk mengambil informasi tiap *pixel* dari citra *input*. Baris 3 dimulai dari iterasi 0 sampai dengan acuan lebar citra *input* dan baris 4 dimulai dari iterasi 0 sampai dengan acuan tinggi citra.
4. Pada baris 5-8 digunakan untuk mengambil informasi nilai RGB dari tiap *pixel*. Proses yang dilakukan dengan cara memanggil *method getRGB()* dengan parameter iterasi yang berjalan dan ditampung melalui variabel untuk setiap nilai *red*, *green*, *blue*.
5. Pada baris 10-15 berfungsi untuk mendapatkan nilai dari ruang warna XYZ sesuai dengan perkalian matriks pada persamaan 2.2.
6. Pada baris 17-19 berfungsi untuk membagi nilai dari ruang warna XYZ dengan masing-masing *white reference*-nya.

7. Pada baris 21-44 digunakan untuk melakukan perhitungan untuk perubahan ruang warna dari XYZ menjadi L*a*b* sesuai dengan persamaan 2.3.
8. Pada baris 46 digunakan untuk melakukan penyimpanan data pada *ArrayList* dari masing-masing nilai L*a*b* hasil perhitungan.
9. Baris 50 mengembalikan nilai *listLAB*.

5.3.2 Implementasi Segmentasi Menggunakan *K-Means*

Beberapa proses pada bagian segmentasi menggunakan *K-Means* sesuai dengan perancangan yang telah dilakukan antara lain segmentasi daun dan segmentasi penyakit. Pada bagian implementasi menggunakan *K-Means* juga dijelaskan mengenai implementasi algoritme *K-Means* serta perhitungan *Euclidean distance* pada algoritme *K-Means*.

5.3.2.1 Implementasi Segmentasi Daun

Segmentasi daun berfungsi untuk memisahkan antara bagian *cover* dan bagian daun jeruk. Sesuai dengan perancangan yang telah dilakukan, segmentasi daun jeruk dapa dilihat melalui kode program 5.4.

```
1   public ArrayList<LAB> segmentasiDaun(ArrayList<LAB> listLAB, int
2   jumlahCluster) {
3           kmeansSegmentasiDaun.Kmeans(listLAB, jumlahCluster);
4           LAB centroid[] = kmeansSegmentasiDaun.getCentroid();
5           int hasilKlasifikasi[] = nearestNeighbor.knn(centroid,
6           dataLatihSegmentasiDaun, 1);
7
8           ArrayList<LAB> listLABdaun = new ArrayList();
9           for (int i = 0; i < hasilKlasifikasi.length; i++) {
10              if (hasilKlasifikasi[i] == 1) {
11                  listLABdaun.addAll(hasilKmeansSegmentasiDaun[i]);
12              }
13          }
14
15          return listLABdaun;
16      }
```

Kode Program 5.4 Implementasi Segmentasi Daun

Penjelasan Kode Program 5.4:

1. Pada baris 1-2 merupakan deklarasi *method* segmentasiDaun() dengan parameter *listLAB* (berisi nilai L*a*b* tiap *pixel*) dan jumlah *cluster*.
2. Pada baris 3 digunakan untuk perhitungan *K-Means* dengan memanggil *method K-Means* disertai parameter *listLAB* dan *jumlahCluster*.
3. Pada baris 4 berfungsi untuk mendapatkan nilai *centroid* dari perhitungan *clustering* menggunakan *K-Means* sebelumnya.
4. Pada baris 5 melakukan pemanggilan *method* K-NN untuk melakukan klasifikasi apakah bagian *cluster* masuk pada kelas *cover* ataukah masuk

pada kelas bagian daun. Pemanggilan *method* menggunakan parameter *centroid* masing-masing *cluster*, data latih segmentasi daun, dan nilai k=1.

5. Pada baris 8-13 melakukan perulangan yang bertujian untuk melakukan cek nilai hasil klasifikasi tiap *centroid*. Jika hasil klasifikasi bernilai 1 (bagian daun) maka data L*a*b* pada *cluster* terkait dialokasikan pada *ArrayList*.
6. Pada baris 15 mengembalikan nilai *listLABdaun* yang berisi nilai L*a*b* bagian daun saja.

5.3.2.2 Implementasi Segmentasi Penyakit

Segmentasi penyakit digunakan untuk membagi area pada daun sesuai dengan perhitungan algoritme *K-Means*. Implementasi segmentasi penyakit dijelaskan melalui kode program 5.5.

```
1   public LAB[] segmentasiPenyakit(ArrayList<LAB> listLAB, int
2   jumlahCluster) {
3           hasilKmeansSegmentasiPenyakit =
4           kmeansSegmentasiPenyakit.Kmeans(listLAB, jumlahCluster);
5           LAB centroid[] = kmeansSegmentasiPenyakit.getCentroid();
6           return centroid;
7       }
```

Kode Program 5.5 Implementasi Segmentasi Penyakit

Penjelasan Kode Program 5.5:

1. Pada baris 1-2 merupakan deklarasi *method* segmentasiPenyakit() dengan parameter *listLAB* (nilai L*a*b* dari masing-masing *pixel* citra daun) dan jumlah *cluster*.
2. Pada baris 3-4 digunakan untuk melakukan perhitungan algoritme *K-Means* dengan memanggil *method KMeans*() yang berisi parameter *listLAB* serta jumlah *cluster*.
3. Pada baris 5 digunakan untuk mendapatkan nilai *centroid* masing-masing *cluster* dengan memanggil *method getCentroid()*.
4. Pada baris 6 mengembalikan nilai *centroid* dari masing-masing *cluster*.

5.3.2.3 Implementasi *K-Means*

Sesuai dengan perancangan yang telah dibuat sebelumnya, implementasi algoritme *K-Means* dapat dilihat melalui kode program 5.6.

```
1   public ArrayList<LAB>[] Kmeans(ArrayList<LAB> listLAB, int
2   jumlahCluster) {
3           centroid = new LAB[jumlahCluster];
4           boolean lanjut = true;
5           int dataCluster[] = new int[listLAB.size()];
6           int dataClusterUpdate[] = new int[listLAB.size()];
7           float jumlahL[] = new float[jumlahCluster];
8           float jumlahA[] = new float[jumlahCluster];
9           float jumlahB[] = new float[jumlahCluster];
10          ArrayList<LAB> dataLABtiapCluster[] = null;
11
12          int iterasi = 0;
```

```
13          Arrays.fill(dataCluster, 0);
14
15          for (int i = 0; i < jumlahCluster; i++) {
16              int random = (int) Math.random() * listLAB.size();
17              centroid[i] = listLAB.get(random);
18          }
19
20          while (lanjut) {
21              dataLABtiapCluster = new ArrayList[jumlahCluster];
22              boolean konvergen = true;
23              for (int i = 0; i < jumlahCluster; i++) {
24                  dataLABtiapCluster[i] = new ArrayList<>();
25              }
26              Arrays.fill(jumlahL, 0);
27              Arrays.fill(jumlahA, 0);
28              Arrays.fill(jumlahB, 0);
29
30              for (int i = 0; i < listLAB.size(); i++) {
31                  float terdekat = 9999;
32                  int idTerdekat = 0;
33                  for (int j = 0; j < centroid.length; j++) {
34                      float jarak = euclideanDistance(listLAB.get(i),
35                      centroid[j]);
36                      if (jarak < terdekat) {
37                          idTerdekat = j;
38                          terdekat = jarak;
39                      }
40                  }
41                  dataClusterUpdate[i] = idTerdekat;
42                  dataLABtiapCluster[idTerdekat].add(listLAB.get(i));
43                  jumlahL[idTerdekat] += listLAB.get(i).getL();
44                  jumlahA[idTerdekat] += listLAB.get(i).getA();
45                  jumlahB[idTerdekat] += listLAB.get(i).getB();
46              }
47
48              for (int i = 0; i < dataCluster.length; i++) {
49                  if (dataClusterUpdate[i] != dataCluster[i]) {
50                      konvergen = false;
51                      break;
52                  }
53              }
54
55              if (!konvergen) {
56                  for (int i = 0; i < centroid.length; i++) {
57                      centroid[i] = new LAB(jumlahL[i] /
58                      dataLABtiapCluster[i].size(), jumlahA[i] /
59                      dataLABtiapCluster[i].size(), jumlahB[i] /
60                      dataLABtiapCluster[i].size());
61                  }
62
63                  for (int i = 0; i < dataCluster.length; i++) {
64                      dataCluster[i] = dataClusterUpdate[i];
65                  }
66
67              } else {
68                  lanjut = false;
69              }
70          }
71          return dataLABtiapCluster;
72      }
```

Kode Program 5.6 Implementasi *K-Means*

Penjelasan Kode Program 5.6:

1. Pada baris 1-2 merupakan deklarasi *method Kmeans()* dengan parameter *listLAB* dan jumlah *cluster*.
2. Pada baris 3-10 merupakan inisialisasi variabel yang dibutuhkan pada pemrosesan algoritme *K-Means*.
3. Pada baris 12-13 digunakan untuk inisialisasi variabel iterasi yang diberi nilai 0 dan isi *array* data *cluster* yang masing-masing *index*-nya diberi nilai 0.
4. Pada baris 15-18 digunakan untuk melakukan inisialisasi nilai *centroid* awal yang didapat dengan mengambil secara random nilai L*a*b* pada variabel *listLAB*.
5. Pada baris 20 deklarasi perulangan menggunakan metode *while* dengan parameter lanjut. Parameter lanjut memiliki tipe *Boolean*. Jika parameter lanjut bernilai *true* melanjutkan proses iterasi dan jika parameter lanjut bernilai *false* menghentikan proses iterasi algoritme *K-Means*.
6. Pada baris 21-28 merupakan inisialisasi variabel yang dibutuhkan didalam proses perulangan. Terdapat pemberian nilai 0 untuk masing-masing *array* dari jumlah setiap bagian ruang warna L*, a*, dan b* pada tiap *cluster*.
7. Pada baris 30-46 terdapat perulangan yang berungsi untuk menentukan kelas *cluster* dari masing-masing data pada *listLAB*. Hasil dari penentuan kelas *cluster* ditampung melalui *array dataClusterUpdate*. Variabel *dataLABtiapCluster* berfungsi untuk menampung nilai L*a*b* dan dialokasikan sesuai dengan kelas *cluster*-nya. Pada perulangan juga terdapat perhitungan jumlah nilai total tiap *cluster* dari masing-masing komponen ruang warna L*a*b* yang dialokasikan pada *array jumlahL*, *jumlahA*, dan *jumlahB*.
8. Pada baris 33-40 terdapat perulangan yang berfungsi untuk menghitung *Euclidean distance* masing-masing data *listLAB* dengan *centroid* yang ada. Pada perulangan ni juga terdapat pengecekan kondisi untuk menentukan jarak ke *centroid* terdekat.
9. Pada baris 48-53 digunakan untuk melakukan pengecekan apakah hasil dari perhitungan kelas *cluster* terbaru dan kelas *cluster* yang lama terdapat perbedaan atau tidak. Jika terdapat perbedaan maka variabel konvergen diberi nilai *false*.
10. Pada baris 55-70 terdapat pengecekan kondisi terhadap variabel konvergen. Jika kovergen bernilai *false* maka dilakukan perhitungan *centroid* terbaru dan mengalokasikan setiap nilai dari *array dataClusterUpdate* pada *array dataCluster*.
11. Pada baris 71 mengembalikan nilai *dataLABtiapCluster* yang berisi informasi ruang warna L*a*b* untuk setiap *cluster*-nya.

5.3.2.4 Implementasi *Euclidean Distance* Untuk *K-Means*

Euclidean distance berfungsi untuk menghitung jarak dari dua data yaitu dataLAB dan *centroid*. Untuk penggunaan pada algoritme *K-Means* pada proses perhitungan *Euclidean distance* hanya menggunakan nilai dari a* dan b*. implementasi *Euclidean distance* untuk *K-Means* dijelaskan pada kode program 5.7.

```
1   public float euclideanDistance(LAB dataLAB, LAB centroid) {
2           float jarak;
3           jarak = (float) Math.sqrt(Math.pow(dataLAB.getA() -
4           centroid.getA(), 2) + Math.pow(dataLAB.getB() -
5           centroid.getB(), 2));
6           return jarak;
7       }
```

Kode Program 5.7 Implementasi *Euclidean Distance* Untuk *K-Means*

Penjelasan Kode Program 5.7:

1. Pada baris 1 merupakan deklarasi *method euclideanDistance()* dengan parameter *dataLAB* dan nilai *centroid*.
2. Pada baris 2-5 merupakan perhitungan jarak *Euclidean distance* sesuai dengan persamaan 2.4. Nilai parameter yang digunakan pada proses perhitungan adalah nilai a* dan b* untuk masing-masing nilai *dataLAB* dan *centroid*.
3. Pada baris 5 mengembalikan nilai jarak yang memiliki tipe *float*.

5.3.3 Implementasi Proses Pelatihan

Sesuai dengan perancangan yang telah dibuat sebelumnya, proses pelatihan dijelaskan melalui kode program 5.8.

```
1   public void prosesDataLatih() throws BiffException {
2           Logika logikaProsesDataLatih = new Logika();
3           Kmeans kmeansDataLatih = new Kmeans();
4           BufferedImage citraInput = null;
5
6           try {
7               citraInput = ImageIO.read(fileCitraInput);
8           } catch (IOException ex) {
9
10              Logger.getLogger(DeteksiPenyakitDaunJeruk.class.getName()
11              ).log(Level.SEVERE, null, ex);
12          }
13
14          citraInput = logikaProsesDataLatih.resizeImage(citraInput,
15          400, 500);
16
17          citraInput = logikaProsesDataLatih.rescaleImage(citraInput,
18          1.1f);
19
20          ArrayList<LAB> listLAB =
21          logikaProsesDataLatih.rgb2lab(citraInput);
22          ArrayList<LAB> listLABdaun =
23          logikaProsesDataLatih.segmentasiDaun(listLAB,
24          Integer.valueOf(spInputJumlahCluster1DataLatih.getValue()
25          .toString()));
```

```
26
27          ArrayList<LAB> dataLABtiapCluster[] =
28          kmeansDataLatih.Kmeans(listLABdaun,
29          Integer.valueOf(spInputJumlahCluster2DataLatih.getValue()
30          .toString()));
31
32          centroidDataLatih = kmeansDataLatih.getCentroid();
33  }
```

Kode Program 5.8 Implementasi Proses Pelatihan

Penjelasan Kode Program 5.8:

1. Pada baris 1 merupakan deklarasi *method prosesDataLatih()*.
2. Pada baris 2-12 berfungsi untuk melakukan inisialisasi variabel yang diperlukan. Terdapat inisialisasi variabel *citraInput* yang berfungsi sebagai penampung citra sesuai lokasi yang di-*input*-kan pada program.
3. Pada baris 14-21 berfungsi untuk melakukan *pre-processing* terhadap citra *input*. Beberapa *method* yang dipanggil antara lain *resizeImage()*, *rescaleImage()*, dan *rgb2lab()*.
4. Pada baris 22-30 berfungsi untuk melakukan segmentasi menggunakan algoritme *K-Means*. Segmentasi yang pertama adalah segmentasi daun untuk memisahkan bagian daun dan *cover*. Segmentasi yang kedua adalah segmentasi penyakit untuk memisahkan tiap-tiap bagian dari daun jeruk sesuai dengan kesamaan fiturnya. Setiap jumlah *cluster* untuk segmentasi penyakit didapatkan dari *input*-an terhadap program.
5. Pada baris 32 berfungsi untuk mendapatkan nilai *centroid* dari masing-masing *cluster* dengan memanggil *method getCentroid()*. Proses pelatihan selanjutnya dengan memasukkan tiap *cluster* pilihan dan melabelkan secara manual tiap jenis penyakitnya lalu dimasukkan pada tabel data latih.

5.3.4 Implementasi Proses Pengujian

Sesuai dengan perancangan yang telah dibuat sebelumnya, proses pengujian dijelaskan melalui kode program 5.9.

```
1   public void prosesDataUji() throws BiffException {
2           BufferedImage citraInput = null;
3           Logika logikaProsesDataUji = new Logika();
4
5           try {
6               citraInput = ImageIO.read(fileCitraInput);
7           } catch (IOException ex) {
8               Logger.getLogger(DeteksiPenyakitDaunJeruk.class.getName()
9               ).log(Level.SEVERE, null, ex);
10          }
11
12          citraInput = logikaProsesDataUji.resizeImage(citraInput, 400,
13          500);
14
15          citraInput = logikaProsesDataUji.rescaleImage(citraInput,
16          1.1f);
17
18          ArrayList<LAB> listLAB =
```

```
19          logikaProsesDataUji.rgb2lab(citraInput);
20
21          ArrayList<LAB> listLABdaun =
22          logikaProsesDataUji.segmentasiDaun(listLAB,
23          Integer.valueOf(spInputJumlahCluster1SatuDataUji.getValue()
24          .toString()));
25
26          LAB centroidHasilSegmentasiDaun[] =
27          logikaProsesDataUji.segmentasiPenyakit(listLABdaun,
28          Integer.valueOf(spInputJumlahCluster2SatuDataUji.getValue()
29          .toString()));
30
31          int hasilKlasifikasi[] = logikaProsesDataUji.klasifikasiKNN(
32          centroidHasilSegmentasiDaun, Integer.valueOf(spInputJumlahK.
33          getValue().toString()));
34
35          ArrayList<LAB> listLABpenyakit[] =
36          logikaProsesDataUji.getHasilKmeansSegmentasiPenyakit();
37
38          int ketersediaanPenyakit[] = new int[4];
39          int hasilKlasifikasiAkhir = 0;
40
41          for (int i = 0; i < hasilKlasifikasi.length; i++) {
42              ketersediaanPenyakit[hasilKlasifikasi[i]] = 1;
43              if (hasilKlasifikasi[i] != 0) {
44                  hasilKlasifikasiAkhir = hasilKlasifikasi[i];
45              }
46          }
47          if (ketersediaanPenyakit[1] + ketersediaanPenyakit[2] +
48          ketersediaanPenyakit[3] > 1) {
49
50              LAB listGabunganAreaPenyakit[] = new LAB[1];
51              float jumlahL = 0, jumlahA = 0, jumlahB = 0;
52              int jumlah = 0;
53              for (int i = 0; i < listLABpenyakit.length; i++) {
54                  if (hasilKlasifikasi[i] != 0) {
55                      for (int j = 0; j < listLABpenyakit[i].size();
56                      j++) {
57
58                          jumlahL += listLABpenyakit[i].get(j).getL();
59                          jumlahA += listLABpenyakit[i].get(j).getA();
60                          jumlahB += listLABpenyakit[i].get(j).getB();
61
62                          jumlah++;
63                      }
64                  }
65              }
66              listGabunganAreaPenyakit[0] = new LAB(jumlahL / jumlah,
67              jumlahA / jumlah, jumlahB / jumlah);
68
69              int hasilKlasifikasiKNNulang[] =
70              logikaProsesDataUji.klasifikasiKNN(
71              listGabunganAreaPenyakit,Integer.valueOf(spInputJumlahK
72              .getValue().toString()));
73
74              hasilKlasifikasiAkhir = hasilKlasifikasiKNNulang[0];
75          }
76      }
```

Kode Program 5.9 Implementasi Proses Pengujian

Penjelasan Kode Program 5.9:

1. Pada baris 1 merupakan deklarasi *method prosesDataUji()*.

2. Pada baris 2-10 berfungsi untuk melakukan inisialisasi variabel yang diperlukan. Terdapat inisialisasi variabel *citraInput* yang berfungsi sebagai penampung citra sesuai lokasi yang di-*input*-kan pada program.
3. Pada baris 12-19 berfungsi untuk melakukan *pre-processing* terhadap citra *input*. Beberapa *method* yang dipanggil antara lain *resizeImage()*, *rescaleImage()*, dan *rgb2lab()*.
4. Pada baris 21-29 berfungsi untuk melakukan segmentasi menggunakan algoritme *K-Means*. Segmentasi yang pertama adalah segmentasi daun untuk memisahkan bagian daun dan *cover*. Segmentasi yang kedua adalah segmentasi penyakit untuk memisahkan tiap-tiap bagian dari daun jeruk sesuai dengan kesamaan fiturnya. Hasil dari segmentasi penyakit berupa nilai *centroid* tiap *cluster* dan dialokasikan pada *array centroidHasilSegmentasiDaun*. Setiap jumlah *cluster* untuk segmentasi penyakit didapatkan dari *input*-an terhadap program.
5. Pada baris 31-33 adalah proses untuk menghitung algoritme K-NN dengan memanggil *method klasifikasiKNN()*. Parameter yang dibutuhkan pada baris ini adalah *centroid* tiap *cluster* dan nilai k berdasarkan *input*-an program.
6. Pada baris 35-36 berfungsi untuk mendapatkan nilai hasil perhitungan *K-Means* pada segmentasi penyakit. Hasil perhitungan *K-Means* ditampung melalui *listLABpenyakit*.
7. 38-39 digunakan untuk inisialisasi variabel yang digunakan pada tahap selanjutnya.
8. Pada baris 41-46 terdapat perulangan untuk memberikan penanda ketersediaan tiap jenis penyakit yang dialokasikan pada *array* ketersediaanPenyakit dengan *index* sesuai dengan hasil klasifikasi tiap *cluster*-nya. Setiap ketersediaan penyakit diberi nilai 1. Pada perulangan ini juga terdapat pegecekan ketika hasil klasifikasi tidak sama dengan daun sehat, maka dilakukan penggantian nilai *hasilKlasifikasiAkhir* menjadi *hasilKlasifikasi cluster* pada iterasi yang berjalan.
9. Pada baris 47-48 merupakan deklarasi pengecekan kondisi jika terdapat jenis penyakit lebih dari 1. Sesuai dengan batasan implementasi kelas 1 merepresentasikan penyakit *Downy Mildew*, kelas 2 merepresentasikan penyakit Cendawan Jelaga, dan kelas 3 merepresentasikan penyakit CVPD maka ketika ketersediaan masing-masing penyakit dijumlahkan hasilnya lebih dari 1 dianggap duplikat.
10. Pada baris 50-52 digunakan untuk inisialisasi variabel yang digunakan pada tahap selanjutnya.
11. Pada baris 53-65 berfungsi untuk mendapatkan jumlah masing-masing komponen ruang warna L*a*b* selain yang memiliki hasil klasifikasi daun sehat. Masing-masing komponen warna ditampung melalui beberapa variabel yakni *jumlahL*, *jumlahA, jumlahB*.

12. Pada baris 66-67 digunakan untuk mendapatkan nilai L*a*b* dari gabungan nilai L*a*b* terkena penyakit dengan cara menghitung rata-rata tiap komponennya.
13. Pada baris 69-72 berfungsi untuk menghitung ulang algoritme K-NN dengan parameter hasil nilai L*a*b* gabungan dan nilai K. Hasil klasifikasi ditampung melalui *array hasilKlasifikasiKNNulang*.
14. Pada baris 74 adalah alokasi nilai terbaru hasil klasifikasi ulang.

5.3.4.2 Implementasi K-NN

Pada pengenalan jenis penyakit daun jeruk pada tiap *cluster* hasil segmentasi penyakit menggunakan algoritme K-NN. Sesuai dengan perancangan yang telah dibuat sebelumnya, Implementasi K-NN dapat dilihat melalui kode program 5.10.

```
1  public int[] knn(LAB centroid[], ArrayList<DataLatih> dataLatih, int
2  jumlahK) {
3
4          ArrayList<Float> seluruhJarak[] = new
5          ArrayList[centroid.length];
6
7          ArrayList<DataLatih> dataLatihUrut[] = new
8          ArrayList[centroid.length];
9
10         int hasilKlasifikasi[] = new int[centroid.length];
11         int dataVoting[][] = new int[centroid.length][4];
12
13         for (int i = 0; i < centroid.length; i++) {
14             Arrays.fill(dataVoting[i], 0);
15         }
16
17         for (int i = 0; i < centroid.length; i++) {
18             seluruhJarak[i] = new ArrayList<>();
19             dataLatihUrut[i] = new ArrayList<>();
20             for (int j = 0; j < dataLatih.size(); j++) {
21                 float jarak = euclideanDistance(centroid[i],
22                 dataLatih.get(j));
23
24                 if (j == 0) {
25                     seluruhJarak[i].add(jarak);
26                     dataLatihUrut[i].add(dataLatih.get(j));
27                 } else {
28                     boolean terbesar = true;
29                     for (int k = 0; k < seluruhJarak[i].size(); k++)
30                     {
31                         if (jarak < seluruhJarak[i].get(k)) {
32                             seluruhJarak[i].add(k, jarak);
33                             dataLatihUrut[i].add(k,
34                             dataLatih.get(j));
35
36                             terbesar = false;
37                             break;
38                         }
39                     }
40                     if (terbesar) {
41                         seluruhJarak[i].add(jarak);
42                         dataLatihUrut[i].add(dataLatih.get(j));
43                     }
44                 }
45             }
46         }
```

```
47
48          for (int i = 0; i < dataLatihUrut.length; i++) {
49              for (int j = 0; j < jumlahK; j++) {
50                  dataVoting[i][dataLatihUrut[i].get(j).getKode()]++;
51              }
52
53              int terbesar = -1;
54              int kodeHasilKlasifikasi = -1;
55              for (int j = 0; j < dataVoting[0].length; j++) {
56                  if (dataVoting[i][j] > terbesar) {
57                      terbesar = dataVoting[i][j];
58                      kodeHasilKlasifikasi = j;
59                  }
60              }
61              hasilKlasifikasi[i] = kodeHasilKlasifikasi;
62          }
63
64          return hasilKlasifikasi;
65      }
```

Kode Program 5.10 Implementasi K-NN

Penjelasan Kode Program 5.10:

1. Pada baris 1-2 merupakan deklarasi *method knn()* dengan parameter *centroid*, data latih, dan nilai k.
2. Pada baris 4-15 berfungsi untuk melakukan inisialisasi pada variabel yang dibutuhkan pada pemrosesan tahap berikutnya. Terdapat perulangan untuk memberi nilai dari *array dataVoting* dengan nilai 0 tiap *index*-nya.
3. Pada baris 17 deklarasi perulangan dengan batas iterasi sebanyak jumlah *array* pada variabel *centroid*.
4. 18-19 berfungsi untuk melakukan inisialisasi pada variabel yang dibutuhkan pada pemrosesan tahap berikutnya.
5. Pada baris 20-45 terdiri dari perulangan yang berfungsi untuk mengurutkan jarak data uji dengan seluruh data latih dari yang memiliki jarak terkecil hingga terbesar. Perhitungan jarak menggunakan pemanggilan pada *method euclideanDistance()*. Terdapat beberapa kondisi yang menentukan penempatan data jarak agar urut yaitu dengan skenario sisip di Depan, sisip di Tengah, dan sisip di Akhir.
6. Pada baris 48-51 terdapat perulangan bersusun yang berfungsi untuk melakukan perhitungan jumlah kemunculan sesuai dengan nilai K. Untuk setiap data jarak *cluster* dilakukan perhitungan jumlah kemunculan yang disesuaikan kelas peyakitnya. Perhitungan jumlah kemunculan dialokasikan pada *array dataVoting*.
7. Pada baris 53-60 digunakan untuk melakukan *voting* berdasarkan jumlah kemunculan dari tiap jenis penyakit. Pada tiap *cluster* dilakukan pengecekan kelas penyakit mana yang memiliki jumlah kemunculan tertinggi. Hasil *voting* dialokasikan pada variabel *kodeHasilKlasifikasi*.

8. Pada baris 61 berfungsi untuk mengganti nilai dari tiap *index hasilKlasifikasi* dengan nilai *kodeHasilKlasifikasi*.
9. Pada baris 64 berfungsi untuk mengembalikan nilai *hasilKlasifikasi* (nilai hasil klasifikasi tiap *cluster*).

5.3.4.3 Implementasi *Euclidean Distance* Untuk K-NN

Euclidean distance berfungsi untuk menghitung jarak dari dua data yaitu *centroid* dan *dataLatih*. Untuk penggunaan pada algoritme K-NN pada proses perhitungan *Euclidean distance* hanya menggunakan seluruh komponen ruang warna L*a*b*. implementasi *Euclidean distance* untuk *K-Means* dijelaskan pada kode program 5.11.

```
1  public float euclideanDistance(LAB centroid, DataLatih dataLatih) {
2          float jarak;
3          jarak = (float) Math.sqrt(Math.pow(centroid.getL() -
4  dataLatih.getL(), 2) + Math.pow(centroid.getA() - dataLatih.getA(),
5  2) + Math.pow(centroid.getB() - dataLatih.getB(), 2));
6          return jarak;
7      }
```

Kode Program 5.11 Implementasi *Euclidean Distance* Untuk K-NN

Penjelasan Kode Program 5.11:

1. Pada baris 1 merupakan deklarasi *method euclideanDistance()* dengan parameter *centroid* dan nilai *dataLatih*.
2. Pada baris 2-5 merupakan perhitungan jarak *Euclidean distance* sesuai dengan persamaan 2.4. Nilai parameter yang digunakan pada proses perhitungan adalah seluruh komponen ruang warna L*a*b* pada setiap data *centroid* dan data latih.
3. Pada baris 6 mengembalikan nilai jarak yang memiliki tipe *float*.

5.4 Implementasi Antarmuka

Implementasi antarmuka menerapkan perancangan dari antarmuka yang telah dibuat sebelumnya. Terdapat 7 perancangan antarmuka yang diimplementasikan yakni halaman awal, halaman pengujian *Scale Factor*, Halaman Pengujian Nilai *Cluster* Optimal, halaman pengujian K optimal, halaman pemrosesan data latih, halaman pemrosesan satu data uji, dan halaman pemrosesan banyak data uji.

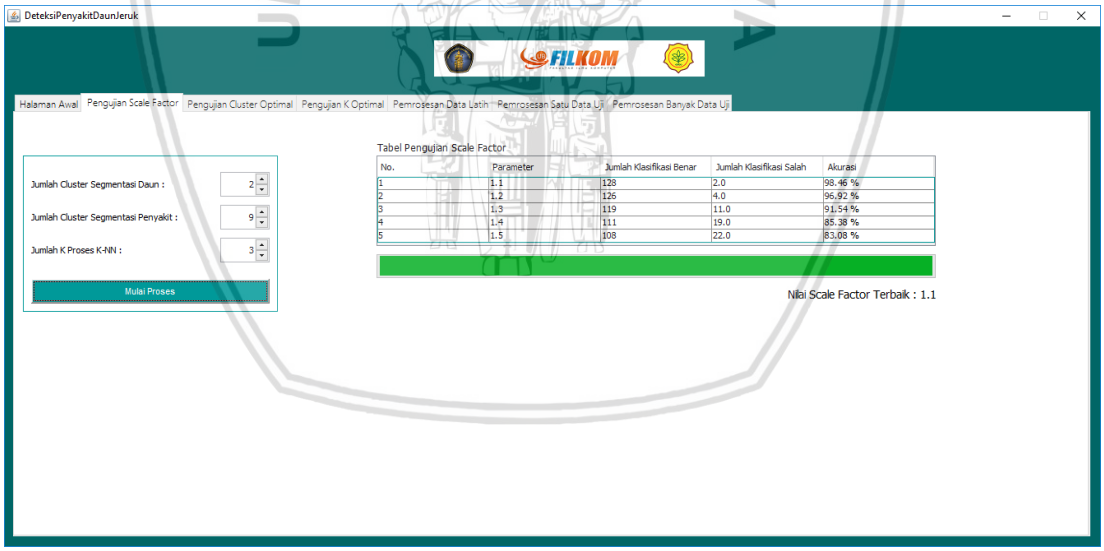
5.4.1 Implementasi Halaman Awal

Halaman awal berfungsi untuk menampilkan deskripsi masing-masing penyakit daun jeruk (CVPD, Cendawan Jelaga, *Downy Mildew*), deskripsi program, dan identitas pembimbing serta mahasiswa. Sesuai dengan perancangan antarmuka yang dibuat sebelumnya, implementasi antarmuka halaman awal dapat dilihat melalui Gambar 5.1.

Gambar 5.1 Implementasi Halaman Awal

5.4.2 Implementasi Halaman Pengujian *Scale Factor*

Halaman pengujian *Scale Factor* berfungsi untuk menampilkan perbandingan nilai *Scale Factor* agar ditemukan nilai yang terbaik untuk *pre-processing* citra. Sesuai dengan perancangan antarmuka yang telah dibuat, implementasi halaman pengujian *Scale Factor* dapat dilihat pada Gambar 5.2.

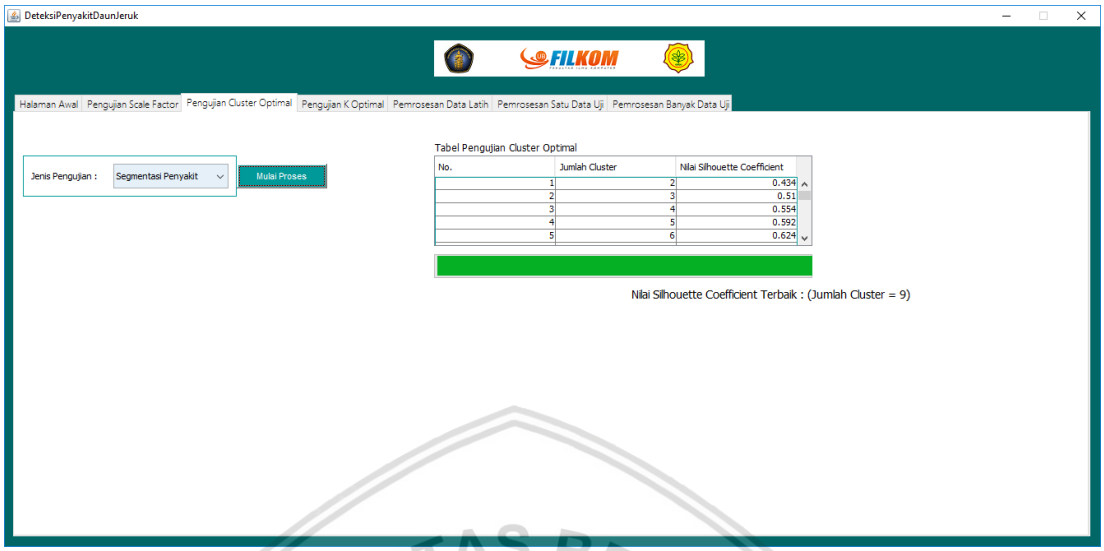


Gambar 5.2 Implementasi Halaman Pengujian *Scale Factor*

5.4.3 Implementasi Halaman Pengujian Nilai *Cluster* Optimal

Halaman pengujian nilai *cluster* optimal menampilkan tabel dari perhitungan menggunakan metode *Silhouette Coefficient* untuk membandingkan dan mendapatkan jumlah *cluster* terbaik pada algoritme *clustering*. Pada halaman pengujian nilai *cluster* optimal, terdapat *comboBox* untuk memilih jenis pengujian dengan memilih salah satu dari segmentasi daun atau segmentasi penyakit. Sesuai

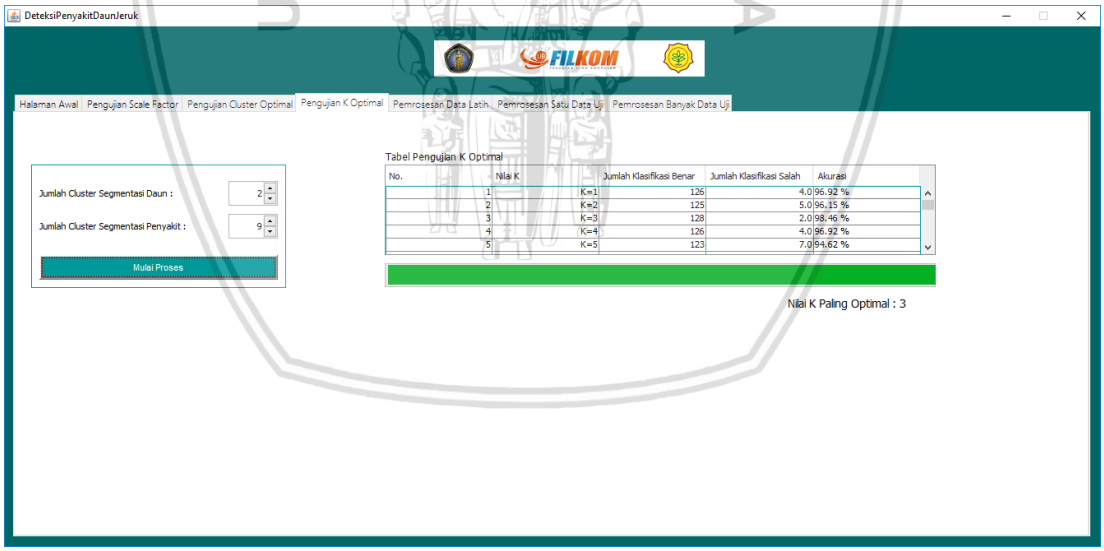
perancangan antarmuka yang dibuat pada tahap sebelumnya, implementasi halaman pengujian nilai *cluster* optimal dapat dilihat melalui Gambar 5.3.



Gambar 5.3 Implementasi Halaman Pengujian Nilai *Cluster* Optimal

5.4.4 Implementasi Halaman Pengujian Nilai K Optimal Pada K-NN

Implementasi halaman pengujian nilai k optimal dapat dilihat melalui Gambar 5.4.



Gambar 5.4 Implementasi Halaman Pengujian K Optimal

Halaman pengujian nilai k optimal berfungsi untuk menampilkan referensi perbandingan masing-masing nilai k pada pemrosesan menggunakan algoritme K-NN.

5.4.5 Implementasi Halaman Pemrosesan Data Latih

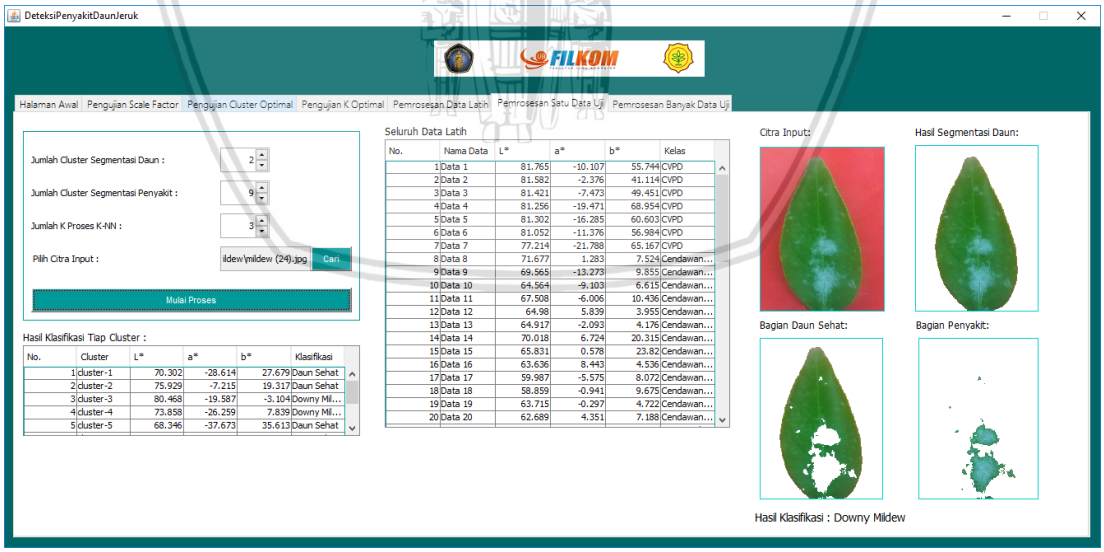
Halaman pemrosesan data latih dibuat untuk memudahkan dalam proses pelatihan data. Masing-masing fitur data ditampilkan melalui tabel hasil

segmentasi penyakit kemudian dapat dipilih setiap *cluster*-nya untuk dilabelkan jenis penyakitnya. Setelah melalui proses pelabelan terdapat *button insert* yang berfungsi untuk memasukkan kedalam tabel data latih. Implementasi halaman pemrosesan data latih sesuai dengan perancangan sebelumnya dapat dilihat melalui Gambar 5.5.

Gambar 5.5 Implementasi Halaman Pemrosesan Data Latih

5.4.6 Implementasi Halaman Pemrosesan Satu Data Uji

Sesuai dengan perancangan antarmuka yang dilakukan sebelumnya, implementasi halaman pemrosesan satu data uji dapat dilihat pada Gambar 5.6.



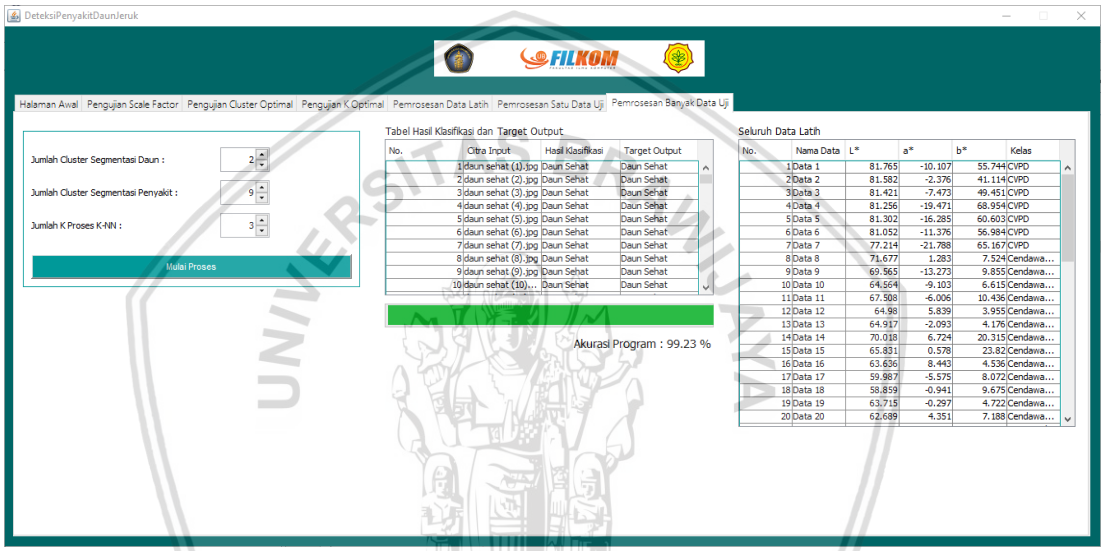
Gambar 5.6 Implementasi Halaman Pemrosesan Satu Data Uji

Implementasi halaman pemrosesan satu data uji berfungsi untuk melakukan pemrosesan identifikasi penyakit daun jeruk dengan satu data uji. Pada halaman pemrosesan satu data uji terdapat hasil identifikasi berupa gambar yakni gambar citra *input*, gambar hasil segmentasi daun, gambar bagian daun (gabungan *cluster*

yang memiliki hasil klasifikasi daun sehat), dan gambar bagian penyakit (gabungan *cluster* yang memiliki hasil klasifikasi selain daun sehat). Selain hasil yang ditampilkan dalam bentuk gambar, terdapat label yang memberikan informasi tentang hasil akhir klasifikasi penyakit daun jeruk.

5.4.7 Implementasi Halaman Pemrosesan Banyak Data Uji

Implementasi halaman pemrosesan satu data uji berfungsi untuk melakukan pemrosesan identifikasi penyakit daun jeruk dengan beberapa data uji. Pada halaman ini ditampilkan tabel hasil klasifikasi dan target *output*-nya serta ditampilkan hasil akurasi dari program. Sesuai perancangan antarmuka yang telah dibuat sebelumnya, implementasi halaman pemrosesan banyak data uji dapat dilihat melalui Gambar 5.7.



Gambar 5.7 Implementasi Halaman Pemrosesan Banyak Data Uji

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini dibahas mengenai pengujian dan analisis dari program yang telah dibuat. Proses pengujian dilakukan untuk menghasilkan *output* akurasi terbaik dari program yang telah dibuat. Terdapat dua bahasan pokok pada bab ini yaitu skenario pengujian dan analisis hasil pengujian seluruh data uji.

6.1 Skenario Pengujian

Sesuai dengan skenario pengujian yang telah ditentukan pada bab 3, skenario pengujian yang dibahas antara lain:

1. Pengujian nilai *Scale Factor*
2. Pengujian nilai *cluster* optimal
3. Pengujian nilai K optimal Pada K-NN

6.1.2 Pengujian Nilai *Scale Factor*

Pengujian nilai *Scale Factor* berfungsi untuk mendapatkan nilai *Scale Factor* terbaik yang digunakan pada tahap *pre-processing* citra. Tahap pada *pre-processing* yang dimaksud adalah pada bagian *rescaling* yaitu untuk melakukan pengaturan pada kecerahan citra. Semakin tinggi nilai *Scale Factor* semakin tinggi pula kecerahan pada citra. Diperlukan pengujian nilai *Scale Factor* guna mendapatkan pengaturan tingkat kecerahan terbaik dalam pemrosesan data uji.

Penggunaan metode pengujian kali ini dengan membandingkan hasil *output* akurasi terbaik untuk setiap parameter nilai *Scale Factor* yang diujikan. Parameter *Scale Factor* yang diujikan dengan nilai 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, dan 2.0. Diperlukan parameter lain untuk mendukung proses pengujian nilai *Scale Factor* yang telah ditentukan antara lain:

1. Ukuran citra proses *resize* : 375 x 500
2. Jumlah *cluster* pada segmentasi daun : 2
3. Jumlah *cluster* pada segmentasi penyakit : 9
4. Jumlah nilai K pada proses klasifikasi penyakit : 3

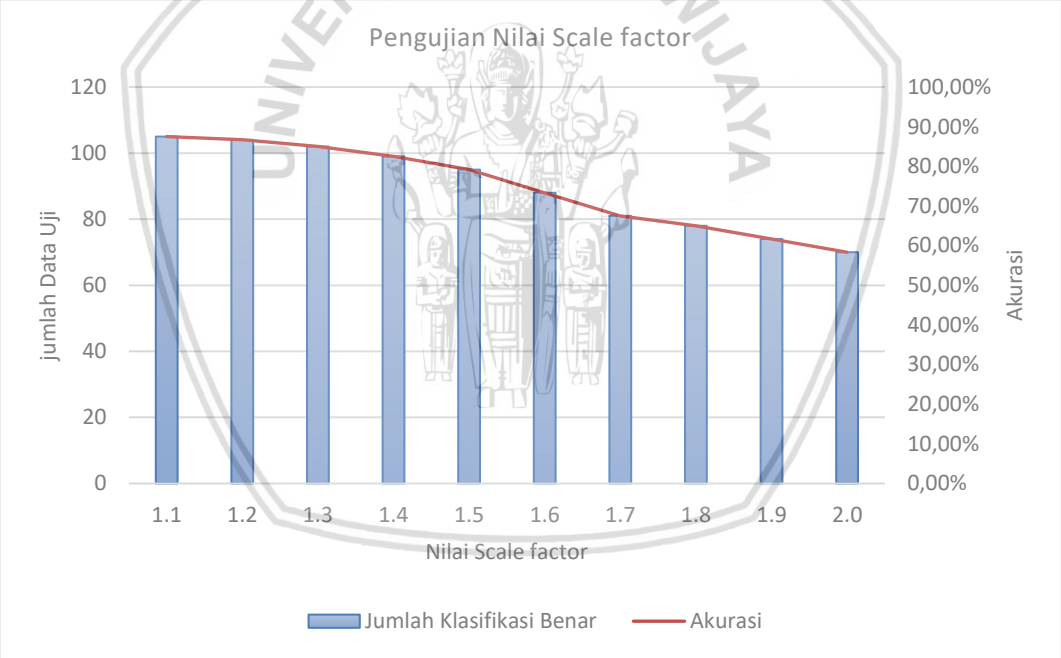
Sesuai perancangan skenario yang telah dilakukan sebelumnya hasil perhitungan akurasi untuk setiap parameter nilai *Scale Factor* dapat dilihat melalui Tabel 6.1.

Tabel 6.1 Hasil Pengujian Nilai *Scale Factor*

No	Nilai *Scale Factor*	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
1	1.1	105	15	87,5 %
2	1.2	104	16	86,67 %
3	1.3	102	18	85%
4	1.4	99	21	82,5 %

No	Nilai *Scale Factor*	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
5	1.5	95	25	79,17 %
6	1.6	88	32	73,33 %
7	1.7	81	39	67,5 %
8	1.8	78	42	65%
9	1.9	74	46	61,67 %
10	2.0	70	50	58,33 %

Berdasarkan Tabel 6.1 memperlihatkan perbandingan hasil akurasi untuk setiap parameter nilai *Scale Factor*. Setelah pengujian terhadap nilai *Scale Factor* 1.1, hasil akurasi terus menurun. Dari total 120 data yang terdiri 30 daun sehat, 30 daun berpenyakit *Downy Mildew*, 30 daun berpenyakit Cendawan Jelaga, dan 30 daun berpenyakit CVPD menghasilkan nilai *Scale Factor* yang paling optimal dengan nilai 1.1. Nilai *Scale Factor* 1.1 memiliki jumlah hasil klasifikasi benar sebanyak 105 dan jumlah hasil klasifikasi salah sebanyak 15 sehingga memiliki persentase akurasi yakni 87.5%. Grafik pengujian nilai *Scale Factor* ditunjukkan pada Gambar 6.1.



Gambar 6.1 Diagram Hasil Pengujian Nilai *Scale Factor*

6.1.3 Pengujian Nilai *Cluster* Optimal

Nilai *cluster* optimal dibutuhkan untuk pemrosesan algoritme *K-Means*. Pengujian nilai *cluster* optimal pada penelitian ini menggunakan metode *Silhouette Coefficient*. Sesuai dengan persamaan 2.5, perhitungan nilai *Silhouette Coefficient* dilakukan untuk seluruh data dan diambil rata-rata pada hasil keseluruhan perhitungan *Silhouette Coefficient*. Terdapat dua pengujian nilai *cluster* optimal yang dilakukan yaitu pada segmentasi daun dan pada segmentasi penyakit.

Penggunaan metode pengujian dengan membandingkan hasil nilai *Silhouette Coefficient* untuk setiap parameter nilai *cluster* yang diujikan. Parameter nilai *cluster* yang diujikan memiliki rentang nilai 2 sampai dengan 10. Pada pengujian ini dibutuhkan satu data uji citra daun dengan cara memilih salah satu dari keseluruhan data uji.

6.1.3.1 Pengujian Pada Segmentasi Daun

Pengujian pada segmentasi daun berfokus pada pencarian jumlah *cluster* optimal algoritme *K-Means* saat pemrosesan segmentasi daun. Segmentasi daun sendiri berfungsi untuk memisahkan area daun jeruk dan area *cover*. Berikut beberapa parameter pendukung yang dibutuhkan dalam proses pengujian:

1. Ukuran citra proses *resize* : 375 x 500
2. Nilai *Scale Factor* proses *rescale* : 1.1
3. Centroid awal menggunakan nilai ruang warna L*a*b* citra input pada data *n* pertama. Data *n* disesuaikan dengan jumlah parameter jumlah *cluster* yang diuji. Hal ini dilakukan agar hasil dari perhitungan *Silhouette Coefficient* tidak berubah-ubah.

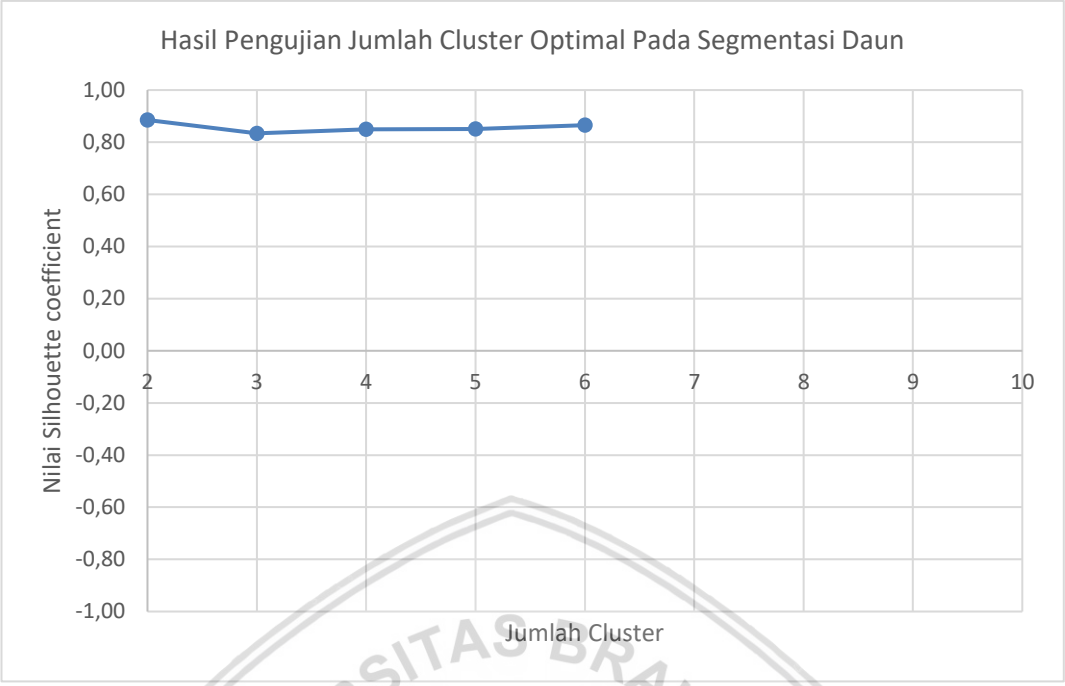
Sesuai perancangan skenario yang telah dilakukan sebelumnya hasil pengujian nilai *cluster* optimal pada segmentasi daun dapat dilihat melalui Tabel 6.2.

Tabel 6.2 Hasil Pengujian Nilai *Cluster* Optimal Pada Segmentasi Daun

No	Jumlah *Cluster*	Nilai *Silhouette Coefficient*
1	2	0,89
2	3	0,83
3	4	0,85
4	5	0,85
5	6	0,87
6	7	#N/A
7	8	#N/A
8	9	#N/A
9	10	#N/A

hasil pengujian mencapai puncak disaat jumlah *cluster* adalah 2 dengan nilai *Silhouette Coefficient* sebesar 0,89. Pada saat pengujian jumlah *cluster* = 7 dan seterusnya tidak mengalami pembentukan *cluster* yang sempurna sehingga nilai *Silhouette Coefficient* menjadi #N/A (tidak terdefinisi). Maka, Dengan ini disimpulkan bahwa jumlah *cluster* = 2 yang menjadi jumlah *cluster* paling optimal pada segmentasi daun.

Kemudian, hasil pengujian pada Tabel 6.2 digambarkan melalui diagram garis yang dapat dilihat pada Gambar 6.2. Sumbu x menjelaskan parameter jumlah *cluster* dengan nilai 2 sampai dengan 10 sedangkan sumbu y menjelaskan nilai *Silhouette Coefficient* dari nilai terendah yaitu -1 hingga tertinggi 1.



Gambar 6.2 Diagram Hasil Pengujian Jumlah *Cluster* Optimal Pada Segmentasi Daun

6.1.3.2 Pengujian Pada Segmentasi Penyakit

Pengujian pada segmentasi penyakit berfokus pada pencarian jumlah *cluster* optimal algoritme *K-Means* saat pemrosesan segmentasi penyakit. Segmentasi penyakit sendiri berfungsi untuk memisahkan area pada daun sesuai dengan kesamaan fiturnya untuk dilakukan proses identifikasi penyakit pada tahap berikutnya. Untuk mendukung proses pengujian, diperlukan beberapa parameter tambahan antara lain:

1. Ukuran citra proses *resize* : 375 x 500
2. Nilai *Scale Factor* proses *rescale* : 1.1
3. Jumlah *cluster* pada segmentasi daun : 2
4. Centroid awal menggunakan nilai ruang warna L*a*b* citra input pada data *n* pertama. Data *n* disesuaikan dengan jumlah parameter jumlah *cluster* yang diuji. Cara ini dilakukan agar hasil dari perhitungan *Silhouette Coefficient* tidak berubah-ubah.

Sesuai perancangan skenario yang telah dilakukan sebelumnya hasil pengujian nilai *cluster* optimal pada segmentasi penyakit dapat dilihat melalui Tabel 6.3.

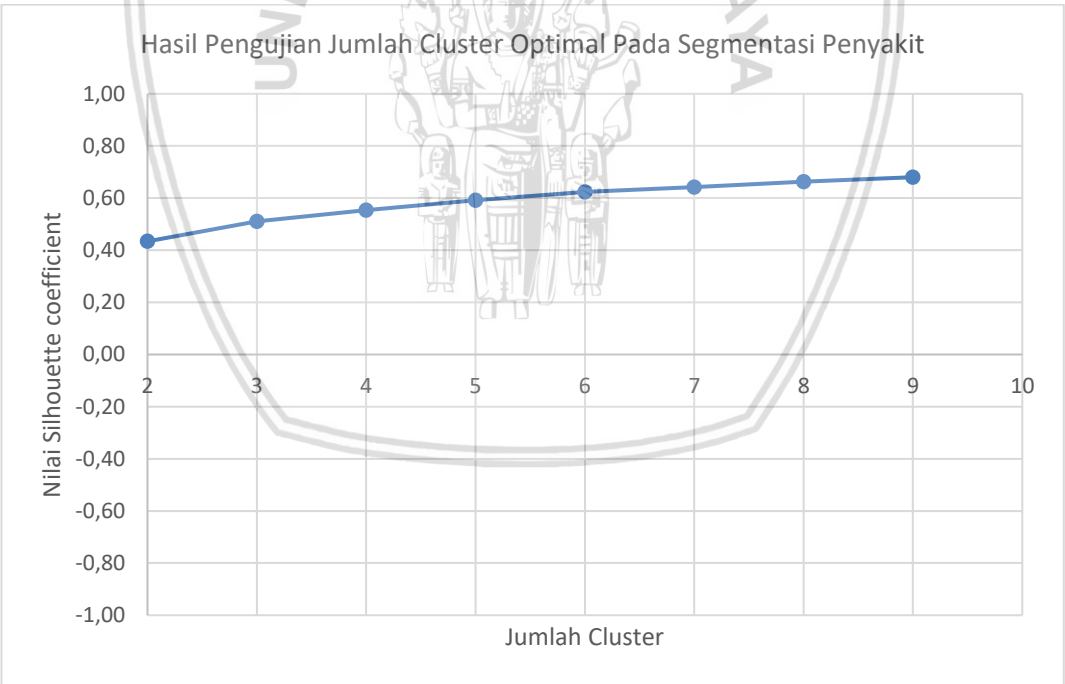
Tabel 6.3 Hasil Pengujian Nilai *Cluster* Optimal Pada Segmentasi Daun

No	Jumlah *Cluster*	Nilai *Silhouette Coefficient*
1	2	0,43
2	3	0,51
3	4	0,55

No	Jumlah *Cluster*	Nilai *Silhouette Coefficient*
4	5	0,59
5	6	0,62
6	7	0,64
7	8	0,66
8	9	0,68
9	10	#N/A

Pada Tabel 6.3 jumlah *cluster* dengan nilai *Silhouette Coefficient* terbesar adalah saat jumlah *cluster* = 9. Nilai *Silhouette Coefficient* yang dihasilkan saat jumlah *cluster* = 9 sebesar 0,68. Pada saat pengujian jumlah *cluster* = 10 dan seterusnya tidak mengalami pembentukan *cluster* yang sempurna sehingga nilai *Silhouette Coefficient* menjadi #N/A (tidak terdefinisi). Sehingga, Dengan ini disimpulkan bahwa jumlah *cluster* = 9 yang menjadi jumlah *cluster* paling optimal pada segmentasi penyakit menggunakan algoritme *K-Means*.

Hasil pengujian pada Tabel 6.3 digambarkan melalui diagram garis yang dapat dilihat pada Gambar 6.3. Sumbu x menjelaskan parameter jumlah *cluster* dengan nilai 2 sampai dengan 10 sedangkan sumbu y menjelaskan nilai *Silhouette Coefficient* dari nilai terendah yaitu -1 hingga tertinggi 1.



Gambar 6.3 Diagram Hasil Pengujian Jumlah *Cluster* Optimal Pada Segmentasi Penyakit

6.1.4 Pengujian Nilai K Optimal Pada K-NN

Nilai K salah satu parameter untuk pemrosesan menggunakan algoritme K-NN. Fungsi mencari nilai K optimal difokuskan pada klasifikasi penyakit daun jeruk. Pengujian nilai K optimal pada penelitian ini menggunakan metode perhitungan

akurasi dengan perbandingan jumlah klasifikasi bendar dan jumlah klasifikasi salah.

Terdapat 120 data uji yang disertakan dalam proses pengujian nilai K optimal yang terdiri dari 30 citra daun dengan kategori sehat, 30 daun berpenyakit *Downy Mildew*, 30 daun berpenyakit Cendawan Jelaga, dan 30 daun berpenyakit CVPD. Parameter nilai K yang diujikan dengan rentang nilai 1 hingga 20. Untuk mendukung proses pengujian, diperlukan beberapa parameter tambahan antara lain:

1. Ukuran citra proses *resize* : 375 x 500
2. Nilai *Scale Factor* proses *rescale* : 1.1
3. Jumlah *cluster* pada segmentasi daun : 2
4. Jumlah *cluster* pada segmentasi penyakit : 9

Sesuai perancangan skenario yang telah dilakukan sebelumnya, hasil pengujian nilai K optimal pada algoritme K-NN dapat dilihat melalui Tabel 6.4.

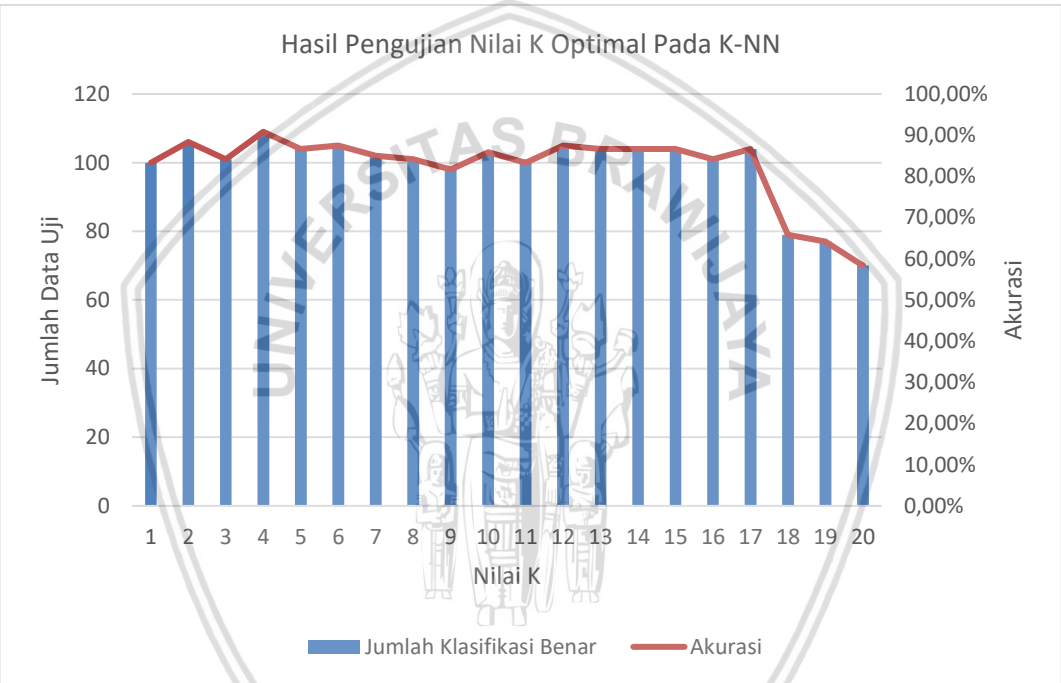
Tabel 6.4 Hasil Pengujian Nilai K Optimal Pada K-NN

No	Nilai K	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
1	K=1	100	20	83,33%
2	K=2	106	14	88,33%
3	K=3	101	19	84,17%
4	K=4	109	11	90,83%
5	K=5	104	16	86,67%
6	K=6	105	15	87,50%
7	K=7	102	18	85%
8	K=8	101	19	84,17%
9	K=9	98	22	81,67%
10	K=10	103	17	85,83%
11	K=11	100	20	83,33%
12	K=12	105	15	87,50%
13	K=13	104	16	86,67%
14	K=14	104	16	86,67%
15	K=15	104	16	86,67%
16	K=16	101	19	84,17%
17	K=17	104	16	86,67%
18	K=18	79	41	65,83%
19	K=19	77	43	64,17%
20	K=20	70	50	58,33%

Pada Tabel 6.4 nilai K yang paling optimal pada saat K=4. Ketika parameter K=4 hasil menunjukkan jumlah klasifikasi bernilai benar adalah 109 dan jumlah klasifikasi salah adalah 11 dengan ini program memiliki akurasi 90,83%. Hasil

akurasi yang dihasilkan untuk setiap parameter K beragam dan cenderung konstan hingga percobaan K=18 dan setelahnya akurasi mulai menurun sehingga tidak memungkinkan untuk menaikkan parameter percobaan pada nilai yang lebih besar. Sehingga, Dengan ini disimpulkan bahwa jumlah *cluster* = 9 yang menjadi jumlah *cluster* paling optimal pada segmentasi penyakit menggunakan algoritme *K-Means*.

Hasil pengujian pada Tabel 6.4 digambarkan melalui diagram yang dapat dilihat pada Gambar 6.4. Ada tiga macam sumbu pada diagram hasil pengujian K optimal. Pada bagian sumbu vertikal dengan rentang nilai 0 hingga 120 menunjukkan jumlah data uji daun jeruk dan nilai presentase dengan rentang 0% hingga 100% merupakan representasi dari nilai akurasi dari program. Pada bagian horizontal dengan rentang nilai 1 hingga 20 menujukkan parameter nilai K yang diujikan.



Gambar 6.4 Diagram Hasil Pengujian Nilai K Optimal Pada K-NN

6.2 Analisis Hasil Pengujian Seluruh Data Uji

Analisis hasil pengujian seluruh data uji menjelaskan mengenai analisa hasil pengujian yang berpengaruh pada hasil akurasi program secara keseluruhan. Analisa meliputi faktor lain selain parameter yang telah dilakukan pada skenario pengujian sebelumnya. Sesuai hasil pada skenario pengujian yang dilakukan antara lain pengujian nilai *Scale Factor*, pengujian nilai *cluster* optimal, dan pengujian nilai K pada K-NN menghasilkan nilai parameter terbaik sebagai berikut:

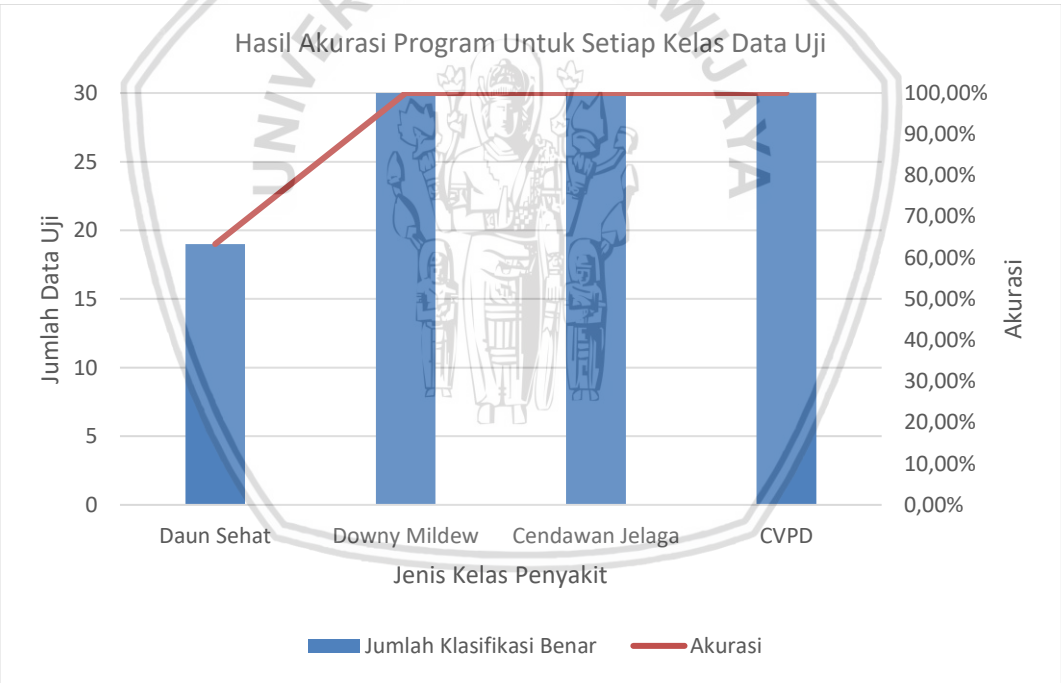
1. Nilai *Scale Factor* : 1.1
2. Jumlah *cluster* pada segmentasi daun : 2
3. Jumlah *cluster* pada segmentasi penyakit : 9
4. Nilai K optimal pada K-NN : 4

Dengan menggunakan parameter terbaik dari hasil pengujian, dilakukan pengujian terhadap 120 data uji dengan rincian 30 data citra daun sehat, 30 daun berpenyakit *Downy Mildew*, 30 daun berpenyakit Cendawan Jelaga, dan 30 daun berpenyakt CVPD menghasilkan akurasi terbaik program sebesar 90.83%. Setelah mendapatkan nilai akurasi keseluruhan, jenis citra daun dipisah dan dibandingkan akurasi untuk tiap-tiap jenisnya. Berbandingan untuk setiap jenis daun dan akurasinya ditunjukkan pada tabel 6.5.

Tabel 6.5 Hasil Akurasi Program Untuk Setiap Kelas Data Uji

Kelas Daun	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
Daun Sehat	19	11	63,33 %
Downy Mildew	30	0	100 %
Cendawan Jelaga	30	0	100 %
CVPD	30	0	100 %

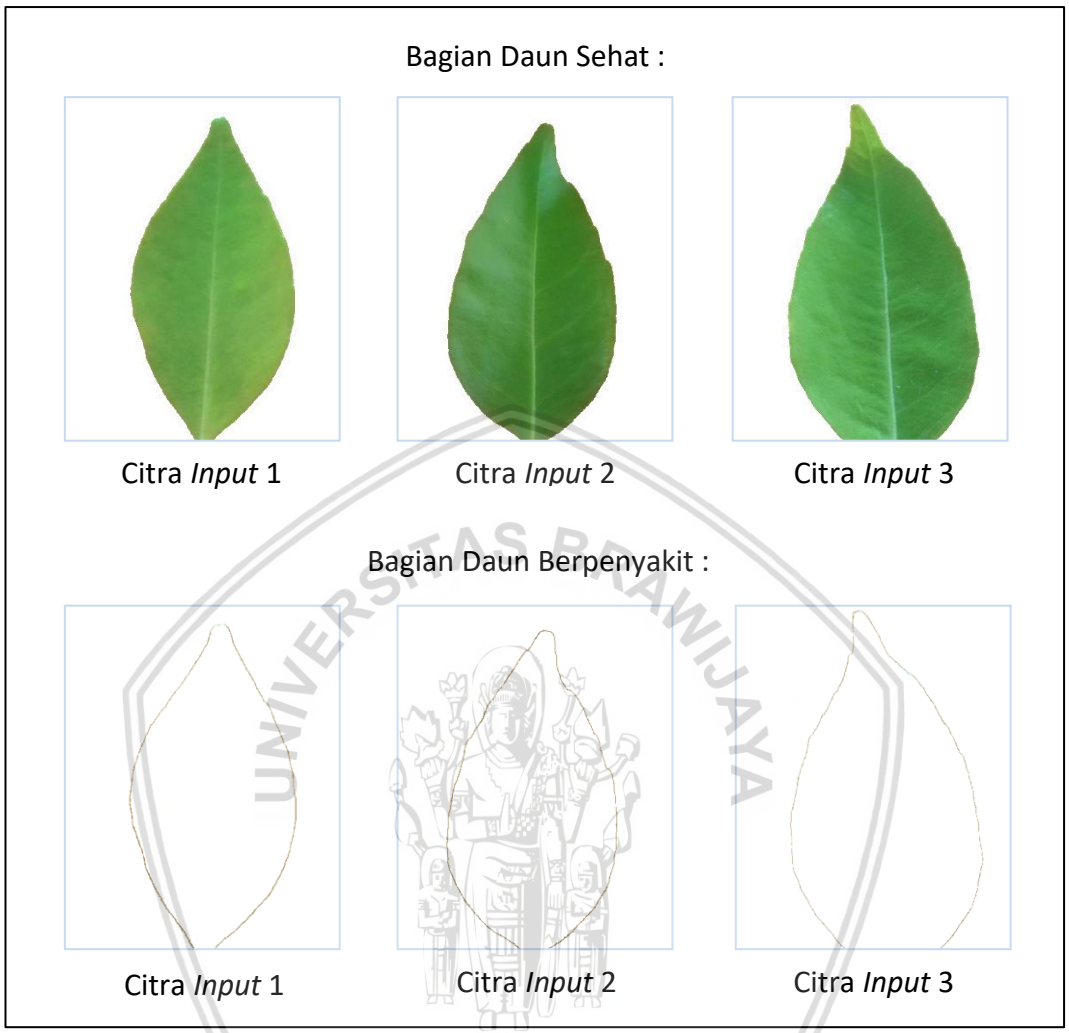
Sesuai dengan Tabel 6.5 hasil akurasi program untuk setiap kelas data uji digambarkan melalui diagram yang dapat ditunjukkan pada Gambar 6.5.



Gambar 6.5 Diagram Hasil Akurasi Program Untuk Setiap Kelas Data Uji

Pada Gambar 6.5 menunjukkan beberapa hasil akurasi dari tiap kelas daun. Untuk daun berpenyakit (*Downy Mildew*, Cendawan Jelaga, dan CVPD), program sudah dapat melakukan klasifikasi dengan baik dan memiliki akurasi masing-masing sebesar 100%. Tetapi kondisi berbeda ditunjukkan pada kelas daun sehat program hanya dapat mencapai akurasi 63.33% untuk mengenali daun tersebut dengan baik. Analisis berikutnya dengan melakukan pengujian satu-persatu kelas daun yang tidak dapat diklasifikasikan dengan baik. Dengan menggunakan 3

contoh daun sehat yang tidak dapat diklasifikasikan dengan baik, hasil pemrosesan dapat dilihat melalui Gambar 6.6.



Gambar 6.6 Hasil Pengujian Daun Sehat

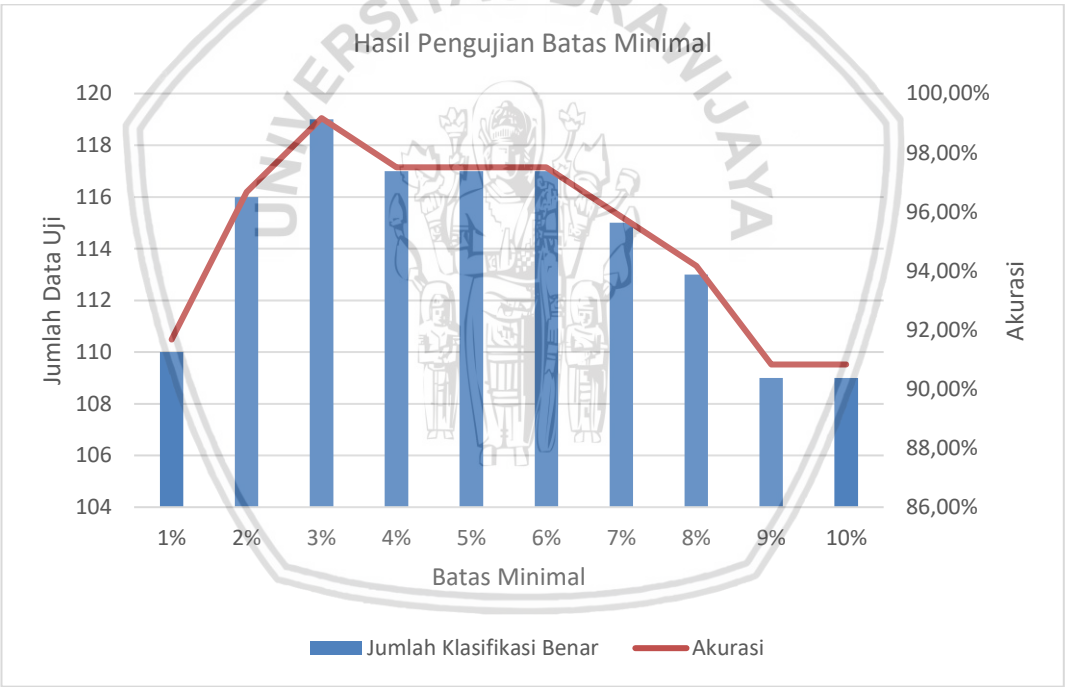
Dari hasil proses identifikasi, citra *input* 1 menghasilkan klasifikasi berpenyakit Cendawan Jelagam citra *input* 2 menghasilkan klasifikasi berpenyakit Cendawan Jelaga, dan citra 3 menghasilkan klasifikasi berpenyakit *Downy Mildew*. Sesuai bagian berpenyakit pada Gambar 6.6 untuk masing-masing citra *input* menghasilkan sedikit daerah berpenyakit yaitu hanya pada daerah tepinya. Sehingga, 3 citra *input* yang seharusnya memiliki taget *output* daun sehat menjadi tidak bisa diklasifikasikan dengan baik. Sesuai dengan analisis yang ada, diperlukan suatu batas minimal dari daerah berpenyakit.

Pada pengujian selanjutnya, untuk memaksimalkan hasil *output* diperlukan langkah untuk mengabaikan daerah berpenyakit sesuai batas minimal tertentu sehingga dianggap sebagai bagian daun sehat. Penggunaan batas minimal berdasarkan persentase area daun dengan beberapa parameter yaitu 1%, 2%, 3%, 4%, 5%, 6%, 7%, 8% ,9%, dan 10%. Hasil pengujian batas minimal untuk seluruh data uji (120 data uji) ditunjukkan melalui Tabel 6.6.

Tabel 6.6 Hasil Pengujian Batas Minimal

No	Batas Minimal	Jumlah Klasifikasi Benar	Jumlah Klasifikasi Salah	Akurasi
1	1 %	110	10	91,67 %
2	2 %	116	4	96,67 %
3	3 %	119	1	99,17 %
4	4 %	117	3	97,5 %
5	5 %	117	3	97,5 %
6	6 %	117	3	97,5 %
7	7 %	115	5	95,83 %
8	8 %	113	7	94,17 %
9	9 %	109	11	90,83 %
10	10 %	109	11	90,83 %

Sesuai Tabel 6.6 hasil pengujian batas minimal digambarkan melalui diagram yang ditunjukkan pada Gambar 6.7.



Gambar 6.7 Diagram Hasil Pengujian Batas Minimal

Pada Gambar 6.7 yang merupakan diagram hasil pengujian batas minimal mencapai kondisi puncak pada saat batas minimal sebesar 3%. Pada kondisi tersebut program dapat mencapai akurasi sebesar 99,17% dengan rincian 119 jumlah klasifikasi benar dan 1 klasifikasi salah. Setelah pengujian parameter pada batas minimal 3 % akurasi program mengalami penurunan secara signifikan. Maka, analisis akhir yang didapat bahwa program dapat mencapai akurasi tertinggi jika diberi batas minimal sebesar 3%. Hal berarti, untuk bagian berpenyakit yang memiliki bagian sebesar 3% atau kurang pada area daun dianggap sebagai bagian daun sehat.

BAB 7 PENUTUP

Pada bab ini berisi mengenai kesimpulan dan saran pada penetian implementasi algoritme *K-Means* sebagai metode segmentasi citra dalam proses identifikasi penyakit daun jeruk.

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian pada penelitian implementasi algoritme *K-Means* sebagai metode segmentasi citra dalam proses identifikasi penyakit daun jeruk dihasilkan beberapa kesimpulan, yaitu:

1. Untuk melakukan implementasi algoritme *K-Means* sebagai metode segmentasi citra dalam proses identifikasi penyakit daun jeruk, melalui beberapa tahapan yaitu *pre-processing* citra *input*, segmentasi menggunakan *K-Means*, dan klasifikasi menggunakan algoritme K-NN. Untuk proses *pre-processing* tersendiri memiliki beberapa tahapan yakni, *resizing* yang berfungsi untuk menyeragamkan ukuran citra *input*, *rescaling* yang berfungsi mengatur kecerahan citra *input*, dan pengubahan ruang warna RGB menjadi L*a*b* yang berfungsi sebagai parameter pada pemrosesan berikutnya. Pada segmentasi menggunakan *K-Means* dibagi menjadi dua proses yakni segmentasi daun yang berfungsi untuk memisahkan bagian daun dan bagian *cover* serta segmentasi penyakit yang membagi daerah pada daun menjadi beberapa bagian yang diproses jenis kelasnya pada tahap selanjutnya. Tahap terakhir adalah identifikasi yang menggunakan metode klasifikasi K-NN. Setelah mendapatkan masing-masing bagian pada daun sesuai hasil segmentasi penyakit masing-masing bagian pada daun dilakukan klasifikasi terhadap data latih menggunakan algoritme K-NN.
2. Berdasarkan pengujian yang telah dilakukan dengan beberapa pengujian antara lain pengujian nilai *Scale Factor*, pengujian nilai *cluster* optimal, dan pengujian K optimal pada K-NN, sistem mendapatkan akurasi tertinggi yaitu 90.83%. Untuk nilai *Scale Factor* didapatkan rekomendasi nilai dengan akurasi tertinggi yaitu 1.1. pengujian nilai *cluster* optimal dilakukan pada dua tahap yakni segmentasi daun dan segmentasi penyakit. Untuk nilai *cluster* optimal yang didapatkan pada proses segmentasi daun adalah 2 dan untuk nilai *cluster* optimal yang didapatkan pada proses segmentasi penyakit adalah 9. Pengujian K optimal pada K-NN merekomendasikan hasil nilai K paling optimal dengan nilai 4. Sesuai dengan analisis berikutnya hasil akurasi program dapat ditingkatkan kembali dengan menggunakan parameter batas minimal. Sesuai hasil pegujian batas minimal, hasil rekomendasi dengan nilai akurasi tertinggi dengan penggunaan batas minimal sebesar 3%. Pada penggunaan seluruh parameter hasil pengujian, hasil akurasi program mencapai 99.17%.

7.2 Saran

Pada pengerjaan penelitian ini, peneliti menganggap bahwasannnya penelitian ini masih jauh dari kata sempurna. Beberapa saran yang bisa diberikan peneliti untuk pengembangan penelitian selanjutnya sebagai berikut:

1. Diperlukan penelitian lebih lanjut pada proses pengambilan informasi pada citra. Bila perlu ditambahkan metode untuk mendapatkan informasi seputar tekstur dari citra.
2. Diperlukan penelitian lebih lanjut untuk melakukan penambahan kelas penyakit selain yang diujikan (*Downy Mildew*, Cendawan Jelaga, CVPD) pada penelitian ini.
3. Perlu dilakukan penelitan lebih lanjut untuk mengidentifikasi penyakit daun jeruk tanpa memetik daun dari pohonnya.
4. Diperlukan penelitian lebih lanjut untuk mengidentifikasi penyakit daun jeruk dengan lebih dari satu daun dalam satu *frame* citra.

DAFTAR PUSTAKA

Agusta, Y., 2007. K-Means – Penerapan, Permasalahan dan Metode Terkait. *Jurnal Sistem dan Informatika,* Volume 3, pp. 47-60.

Andri, Paulus, Wong, N. P. & Gunawan, T., 2014. Segmentasi Buah Menggunakan Metode K-Means Clustering dan Identifikasi Kematangannya Menggunakan Metode Perbandingan Kadar Warna. *JSM STMIK Mikroskil,* Volume 15, pp. 91-100.

Anggara, M., Sujiani, H. & Nasution, H., 2016. Pemilihan Distance Measure Pada K-Means Clustering Untuk Pengelompokkan Member Di Alvaro Fitness. *Jurnal Sistem dan Teknologi Informasi (JUSTIN),* 1(1), pp. 1-6.

Arsy, L., Nurhayati, O. D. & Martono, K. T., 2016. Aplikasi Pengolahan Citra Digital Meat Detection dengan Metode Segmentasi K-Means Clustering Berbasis OpenCV dan Eclipse. *Jurnal Teknologi dan Sistem Komputer,* IV(2), pp. 322-332.

Asana, I. M. D. P., Widyantara, I. M. O., Wirastuti, N. & Adnyana, I. B. P., 2017. Metode Contrast Stretching untuk Perbaikan Kualitas Citra pada Proses Segmentasi Video. *Teknologi Elektro,* 16(02), pp. 1-6.

Athanikar, G. & Badar, P., 2016. Potato Leaf Diseases Detection and Classification System. *A Monthly Journal of Computer Science and Information Technology ,* V(2), pp. 76-88.

Balitjestro, 2015. *Laporan Akuntabilitas Kerja,* Malang: Balai Penelitian Tanaman Jeruk dan Buah Subtropika.

Balitjestro, 2017. *Balitjestro - Kementerian Pertanian.* [Online] Available at: http://balitjestro.litbang.pertanian.go.id/wp-content/uploads/2017/08/batu-55.jpg [Diakses 16 Januari 2018].

Dhanachandra, N., Manglem, K. & Chanu, Y. J., 2015. Image Segmentation using K-means Clustering Algorithm and Subtractive Clustering Algorithm. *Procedia Computer Science,* pp. 764-771.

Dubey, S. R., Dixit, P., Singh, N. & Gupta, J. P., 2013. Infected Fruit Part Detection using K-Means Clustering Segmentation Technique. *International Journal of Artificial Intelligence and Interactive Multimedia,* 2(2), pp. 65-72.

Hapsari, Y. & Hidayattullah, M. F., 2013. *Deteksi Wajah Dari Berbagai Ras Manusia Menggunakan Warna Kulit Berbasis Ruang Warna L*a*b.* Semarang: s.n.

Kementan, 2017. *Statistik Pertanian 2017,* s.l.: Pusat Data dan Sistem Informasi Pertanian Kementerian Pertanian Republik Indonesia.

Knudsen, J., 1999. *Java 2D Graphics.* Sebastopol: O'Reilly & Associates.

Mardhiyah, A. & Harjoko, A., 2011. Metode Segmentasi Paru-paru dan Jantung Pada Citra XRay Thorax. *IJEIS,* 1(2), pp. 35-44.

Meitayani, N. P. S., Adiartayasa, W. & Wijaya, I. N., 2014. Deteksi Penyakit Citrus Vein Phloem Degeneration (CVPD) dengan Teknik Polymerase Chain Reaction (PCR) pada Tanaman Jeruk di Bali. *E-Jurnal Agroekoteknologi Tropika,* 3(2), pp. 70-79.

Pratt, W. K., 2007. *Digital Image Processing.* 4th ed. Los Altos, California: A John Wiley & Sons, Inc..

Priambodo, A., Dewi, C. & Triwiratno, A., 2015. *Implementasi Metode K-Nearest Neighbor untuk Identifikasi Penyakit Tanaman Jeruk Keprok berdasarkan Citra Daun.* Malang: s.n.

Priandana, K., Zulfikar, A. & Sukarman, 2016. Android-Based Mobile Munsell Soil Color Chart by Using HVC Color Model Histogram with KNN Classification. *Jurnal Ilmu Komputer Agri-Informatika,* 3(2), pp. 93-101.

Rizal, A. Y., Dewi, C. & Widodo, A. W., 2016. *Implementasi Algoritma Multilevel Thresholding Menggunakan Otsu sebagai Preprocessing Data Citra Daun pada Proses Identifikasi Penyakit Tanaman Jeruk.* Malang: s.n.

Rizal, P., Bachrian, W. & Retno, 2011. *Budidaya Jeruk Bebas Penyakit,* Kalimantan Timur: Balai Pengkajian Teknologi Pertanian.

Rulaningtyas, R., Suksmono, A. B., Mengko, T. L. R. & Saptawati, G. A. P., 2015. Segmentasi Citra Berwarna dengan Menggunakan Metode Clustering Berbasis Patch untuk Identifikasi Mycobacterium Tuberculosis. *Biosains,* 17(1).

Script Tutorials, 2014. *What is a color model?.* [Online] Available at: https://www.script-tutorials.com/what-is-a-color-model/ [Diakses 18 Januari 2018].

Setiono, 2015. *Inovasi Jeruk Keprok Batu 55.* Jakarta: IAARD Press.

Syafitri, N., 2010. Perbandingan Metode K-Nearest Neighbor (KNN) dan Metode Nearest Cluster Classifier (NCC) dalam Pengklasifikasian Kualitas Batik Tulis. *Jurnal Teknologi Informasi & Pendidikan,* 2(1), pp. 43-53.

Syafril, 2006. *Jenis Hama dan Penyakit Penting Menyerang Jeruk Koto Tinggi,* Padang: Balai Pengkajian Teknologi Pertanian Sumatera Barat.

Tatiraju, S. & Mehta, A., 2008. Image Segmentation using K-Means Clustering, EM and Normalized Cuts. *Department of EECS, University of California,* pp. 1-7.

Triwiratno, A., 2018. *Macam dan Cara Mengindentifikasi Penyakit Daun Jeruk* [Wawancara] (10 Januari 2018).

Umran, M. & Abidin, T. F., 2009. Pengelompokkan Dokumen Menggunakan K-Means dan Singular Value Decomposition: Studi Kasus Menggunakan Data Blog. *SESINDO - Jurusan Sistem Informasi ITS,* pp. 1-5.

Wahyuningsih, E., 2009. CVPD pada Jeruk (Citrus spp) dan Upaya Pengendaliannya. *Vis Vitalis,* 02(2), pp. 65-73.

Wibowo, S. A. & Usman, K., 2010. Voice Activity Detection G729B Improvement Technique Using K-Nearest Neighbor Method. *International Conference on Distributed Frameworks for Multimedia Applications (DFmA) .*

Wijaya, E. S. & Prayudi, Y., 2015. Integrasi Metode Steganografi DCS Pada Image Dengan Kriptografi Blowfish Sebagai Model Anti Forensik Untuk Keamanan Ganda Konten Digital. *Seminar Nasional Aplikasi Teknologi Informasi (SNATi),* pp. D11-D17.

Zhang, C. et al., 2014. White Blood Cell Segmentation by Color-Space-Based K-Means Clustering. *Sensors,* Issue 14, pp. 16128-16147.

Zhou, H., Wu, J. & Zhang, J., 2010. *Digital Image Processing : Part I.* 1st ed. Frederiksberg: Ventus Publishing Aps.