IMPLEMENTASI ALGORITMA *MULTILEVEL THRESHOLDING* MENGGUNAKAN *OTSU* SEBAGAI *PREPROCESSING* DATA CITRA DAUN PADA PROSES IDENTIFIKASI PENYAKIT TANAMAN JERUK

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Andriansyah Yusuf Rizal
NIM: 135150201111122

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

IMPLEMENTASI ALGORITMA *MULTILEVEL THRESHOLDING* MENGGUNAKAN *OTSU* SEBAGAI *PREPROCESSING* DATA CITRA DAUN PADA PROSES IDENTIFIKASI PENYAKIT TANAMAN JERUK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Andriansyah Yusuf Rizal
NIM : 135150201111122

Skripsi ini telah diuji dan dinyatakan lulus pada 1 Februari 2017
Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Candra Dewi, S.Kom., M.Sc	Agus Wahyu Widodo, S.T., M.Cs
NIP: 19771114 200312 2 001	NIP: 19740805 200112 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, 20 Januari 2017

Andriansyah Yusuf Rizal

NIM: 135150201111122

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT, karena dengan rahmat, hidayah dan karunia-Nya, penulis dapat meneyelesaikan skripsi dengan judul "Implementasi Algoritma *Multilevel Thresholding* Menggunakan *Otsu* sebagai *Preprocessing* data Citra Daun pada Proses Identifikasi Penyakit Tanaman Jeruk" dengan baik dan tepat waktu.

Skripsi ini disusun sebagai salah satu persyaratan untuk menyelesaikan studi di Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya. Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan banyak terima kasih kepada berbagai pihak yang telah memberikan bantuan dan dukungan selama proses pengerjaan skripsi, diantaranya :

1. Kedua orang tua saya yang tidak henti-hentinya mendo'akan, memberi kasih saying serta dorongan moril dan materil dari awal sampai akhir pengerjaan skripsi ini.
2. Ibu Candra Dewi, S.Kom., M.Sc selaku dosen pembimbing I dan Bapak Agus Wahyu Widodo, S.T., M.Cs selaku dosen pembimbing II yang telah banyak memberikan bimbingan, arahan, ilmu dan saran selama penyusunan laporan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak dan Ibu dosen serta seluruh staf Fakultas Ilmu Komputer Universitas Brawijaya yang telah mendidik serta membantu penulis selama perkuliahan.
5. Bapak Dr. Ir. Anang Triwiratno, M.P selaku koordinator Laboratorium di Balai Penelitian Tanaman Jeruk dan Buah Subtropika yang telah memberikan arahan dan ilmu pertanian tanaman jeruk selama penyusunan laporan skripsi ini.
6. Mochammad Rafli Andriansyah, Dina Agustina, Amd dan Unun Triasih, S.P yang telah membantu penulis dalam proses pengumpulan data daun jeruk yang berpenyakit.
7. Teman-teman seperjuangan Informatika angkatan 2013 yang tidak bisa disebutkan satu persatu yang selalu mengajak, menemani, mendorong dan membantu penulis.
8. Keluarga EMIF Kabinet Berinovasi yang telah menemani penulis selama satu kepengurusan dengan banyak memberi ilmu, arahan, mendorong penulis untuk menyelesaikan perkuliahan dan pengerjaan skripsi ini.
9. Teman-teman BPH EMIF 2013 sebagai rekan yang saling memperjuangkan teman-teman HMIF Filkom UB yang telah mensupport, memberikan semangat, menemani dalam membantu penulisan laporan skripsi ini.

10. Rekan-rekan BCC (*Basic Computing Community*) yang telah memberikan ilmu-ilmu baru selama perkuliahan dan membantu dalam hal penulisan laporan skripsi ini.
11. Fahmi Nur Aini, Mada Eka Prayudha, Sudeni Pratama, Hasbiya Diona Arani sebagai sahabat setia penulis yang selalu memberikan dukungan, saling memotivasi dan memberi semangat.
12. Anandhi Tristiaratri, Anandita Azharunisa Sasmito, Sabrina Nurfadilla sebagai rekan belajar penulis yang telah memberi dukungan dan bantuan kepada penulis untuk menyelesaikan penulisan skripsi ini.
13. Taqwin Muzakki, S.Kom yang telah membentu memberikan arahan untuk memilih Teknik Informatika UB menjadi perkuliahan selanjutnya dan juga memberikan ilmu serta bekal baik sebelum ataupun saat perkuliahaan, dan juga menjadi inspirasi penulis saat berkuliah di Jurusan Teknik Informatika Fakultas Ilmu Komputer.
14. Rekan-rekan kontrakan sipon yang berpartisipasi dalam mengajak dan membantu dalam penyusunan skripsi serta membantu dalam mempersiapkan seminar hasil.
15. Seluruh Himpunan Mahasiswa Informatika yang telah memberikan tempat untuk mencari pengalaman, sebagai tempat curhat, tempat untuk mencari teman dan sahabat di Fakultas Ilmu Komputer Universitas Brawijaya.
16. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dan semoga penulisan laporan skripsi ini bermanfaat bai pembaca untuk pengembangan penelitian selanjutnya. Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dan tidak terlepas dari kekurangan dan kesalahan karena keterbatasan materi, kemampuan, pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, segala kritik dan saran yang bersifta membangun, penulis harapkan untuk penyempurnaan skripsi ini yang dapat disampaikan melalui email penulis yaitu andriansyahgl@gmail.com.

Malang, 1 Februari 2017

Penulis

andriansyahgl@gmail.com

ABSTRAK

Penyakit pada tanaman jeruk adalah salah satu kendala untuk meningkatkan produksi jeruk di Indonesia. Berkembangnya teknologi informasi pada saat ini dapat dilakukan pembuatan aplikasi untuk melakukan identifikasi penyakit pada tanaman jeruk melalui citra daun. Penyakit yang diidentifikasi pada penelitian ini adalah penyakit CVPD *(Citrus Vein Phloem Degeneration)*, *Mildew,* Cendawan Jelaga, Defisiensi (Mg dan Zn). Proses pengenalan penyakit tanaman jeruk diawali dengan melakukan penambahan nilai konstanta untuk menambahkan kecerahan pada citra. Selanjutnya dilakukan penerapan *multilevel thresholding menggunakan otsu* untuk mendapatkan bagian citra yang berpenyakit. Setelah didapatkan bagian berpenyakit, proses selanjutnya dilakukan ekstraksi ciri dengan mengambil nilai rata-rata *red, green, blue* (RGB). Setelah didapatkan ciri pada daun maka proses selanjutnya adalah melakukan klasifikasi dengan menggunakan metode *K-Nearest Neighbor* (KNN). Pengujian yang dilakukan pada penelitian ini adalah pengujian nilai *scale factor*, nilai *MLEVEL*, dan nilai *K*. Berdasarkan tiga pengujian tersebut didapatkan rekomendasi nilai *scale factor* yaitu 1.4, nilai *MLEVEL* yaitu 3, dan nilai *k* yaitu 2. Proses selanjutnya adalah pengujian dengan tiga tipe data uji, perbedaan data uji terletak pada penyakit defisiensi dimana masing-masing pengujian dilakukan pada defisiensi Zn, defisiensi Mg dan semua defisiensi (Mg dan Zn). Berdasarkan nilai rekomendasi tersebut didapatkan akurasi tertinggi sebesar 92% pada data defisiensi Zn, 80% pada defisiensi Mg, dan 80% pada defisiensi Mg dan Zn.

Kata Kunci : Penyakit Jeruk, Citra Digital, *Multilvel Thresholding, Otsu*, *K-Nearest Neighbor*.

ABSTRACT

Citrus disease is one of the obstacles to increase citrus production in Indonesia. With development of information technology at this moment can be making an application to identify the disease in citrus through the image of leaves. Diseases that were identified in this research are CVPD (Citrus Vein Phloem Degeneration), Mildew, Jelaga, Deficiency (Mg and Zn). The process of introduction of citrus disease begins by adding a constant value to add brightness of the image. Next is implementation of the multilevel thresholding using otsu to obtain section image of the diseased. Having obtained the disease part, feature extraction process is then performed by taking the average value of red, green, blue (RGB). Having obtained the feature of the leaves o the next step is classification using K-Nearest Neighbor (KNN). Test carried out in this research is testing the value of scale factor, MLEVEL value, and the value of K. Based on these three test were obtained recommendation scale factor value is 1.4, the value MLEVEL which is 3, and the value of k is 2. the next step is testing three types of test data, the difference lies in the test data deficiency disease in which each test performed on Zn deficiency, Mg deficiency and al deficiencies (Mg and Zn). Based on the value of the recommendations obtained the highest accuracy of 92% on the data Zn deficiency, 80% in Mg deficiency, 80% in Mg and Zn deficiency.

Keywords : Citrus Disease, Digital Image, Multilevel Thresholding, Otsu, K-Nearest Neighbor

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Salah satu komoditas hortikultura yang memiliki peranan strategis dalam pembangunan nasional adalah buah-buahan. Dilihat dari fungsi dan manfaatnya peranan buah-buahan dalam tubuh merupakan makanan bergizi yang mengandung banyak vitamin yang diperlukan oleh tubuh, selain itu dari segi pembangunan nasional juga merupakan sumber pendapatan negara, serta meningkatkan ekspor. Buah jeruk sangat digemari oleh masyarakat Indonesia karena harganya yang relatif terjangkau, dengan nilai ekonomis yang tidak besar masyarakat mendapatkan nilai gizi yang baik bagi tubuh. Jenis buah-buahan yang memiliki prospek baik terbagi menjadi tiga kelompok, yaitu : (1) mencakup mangga, rambutan, pisang, jeruk, dan sirsak; (2) mencakup durian, manggis, nanas, salak, dan nangka; (3) mencakup markisa, pepaya, duku, apel, anggur, lengkeng, dan melon (Poerwanto, 2004).

Dengan banyaknya kesadaran masyarakat akan pentingnya untuk mengkonsumsi buah yang bergizi khususnya jeruk, tentunya hal ini membuat Indonesia diharuskan untuk menyiapkan produksi jeruk yang bisa mencukupi kebutuhan masyarakat Indonesia (Hanif & Zamzami, 2012). Pada lima tahun terakhir (2011-2015), produksi jeruk di Indonesia cenderung fluktuatif (Tabel 1). Dengan banyaknya permintaan buah jeruk yang selalu meningkat dari tahun ke tahun maka Indonesia diharuskan untuk memenuhi kebutuhan jeruk dengan cara mengimpor, Indonesia menjadi negara pengimpor kedua di ASEAN setelah Malaysia. Karena banyaknya jeruk yang diimpor maka tidak heran jeruk impor lebih banyak ditemui di swalayan ataupun pedagang kaki lima, selain itu konsumen lebih banyak menyukai jeruk impor dikarenakan kualitas yang lebih baik dibandingkan jeruk lokal (Hardiyanto, 2009). Mutu jeruk yang prima ditentukan dengan beberapa faktor antara lain ukuran buah, warna buah yang rata, kemulusan buah. Selain itu dari segi nutrisi yang prima, mutu buah ditentukan pula dengan kandungan mineral yang ada di dalam buah jeruk. Kebanyakan mutu buah yang kurang prima dipengaruhi karena kekurangan nutrisi dikarenakan penyakit ataupun hama. Contoh buah yang terserang penyakit adalah CVPD (*Citrus Vein Phloem Degeneration*), Cendawan Jelaga dan *Downy Mildew* (Triwiratno, 2016).

Tabel 1.1 Produksi Jeruk Tahun 2011-2015

No	Tahun	Berat (ton)
1	2011	97.069
2	2012	113.375
3	2013	106.338
4	2014	141.288
5	2015	111.746

Sumber : BPS dan Direktorat Jendral Hortikultura 2015

Hingga saat ini banyak petani yang mengidentifikasi penyakit tanaman jeruk dengan cara menggunakan mata telanjang dan dilakukan pengujian Laboratorium. Dengan cara tersebut dirasa belum maksimal dikarenakan hanya melihat tampilan luar saja. Bahkan dengan cara lab pun diperlukan waktu yang relatif lama dan diperlukan alat yang khusus, hal itu dirasa juga kurang maksimal selain itu petani yang belum memiliki alat tersebuh dirasa juga belum merasa terbantu karena harus membeli peralatan untuk melakukan identifikasi penyakit tanamanan jeruk (Triwiratno, 2016). Oleh karena itu hasil yang tepat sangat diperlukan dan juga dapat membantu petani yang sekiranya tidak memiliki peralatan untuk melakukan identifikasi penyakit jeruk.

Dalam perkembangan teknologi saat ini, banyak ide bermunculan dalam bidang identifikasi penyakit dengan menggunakan citra. Sistem dapat mengenali penyakit yang terdapat pada sebuah tanaman melakukan proses pembelajaran dengan citra sebagai masukan. Selain melakukan pembelajaran citra, juga dilakukan pemrosesan awal yang sangat berdampak pada kualitas citra sebelum dilakukan pembelajaran data. Sehingga setelah dilakukan pembelajaran data maka aplikasi dapat mengidentifikasi citra uji.

Dalam Penelitian sebelumnya yang memiliki tema proses identifikasi tanaman dengan menggunakan metode KNN. Sebelum citra daun di proses pada KNN, pertama yang dilakukan adalah dengan menambahkan nilai *alpha* untuk di tambahkan nilai kecerahannya yang selanjutnya dicari nilai rata-rata dari RGB (*Red, Green, Blue*) dan selanjutnya dilakukan proses pembelajaran dengan menggunakan algoritma *K-Nearest Neighbor* (KNN). Dari proses tersebut didapatkan akurasi sebesar 96.67% (Pirambodo, et al., 2015). Meskipun hasilnya sudah bagus, namun masih banyak dijumpai beberapa kekurangan. Proses identifikasi data yang dilakukan dengan memasukkan citra daun kemudian, dilatih serta diuji dengan metode KNN. Pada penelitian ini penyakit yang diidentifikasi adalah CVPD, Jelaga dan juga *Mildew*. Karena proses pada saat ekstraksi RGB citra dilakukan pada semua warna baik pada luasan citra yang berpenyakit ataupun tidak maka proses ini akan memakan waktu lama karena proses perhitungan nilai RGB dilakukan pada semua bagian baik berpenyakit ataupun tidak. Selain itu pada penelitian ini juga, system belum dapat mengidentifikasi daun normal. Maka dalam penelitian ini ditambahkan proses *Preprocessing* yaitu dengan menambahkan algoritma *Multi Thresholding Otsu* yang diharapkan dapat memberikan nilai akurasi yang lebih tinggi. *Thresholding Otsu* dilakukan agar dapat membedakan bagian-bagian pada daun yang terkena penyakit dan tidak, tentunya akan berbeda dengan penelitian sebelumnya dimana tidak ada pemisahan bagian-bagian yang berpenyakit dan tidak sehingga pencarian rata-rata nilai RGB lebih difokuskan pada yang berpenyakit saja. Selain itu proses *thresholding* yang dilakukan tidak hanya satu mengingat banyaknya penyakit yang akan diidentifikasi terdapat 4 yaitu : *Downy Mildew*, Cendawan Jelaga, CVPD dan defisiensi sehingga perlu ditambahkan *multilevel thresholding.*

Dalam penelitian sebelumnya yang memiliki tema tentang proses identifikasi penyakit tanaman kedelai dengan menggunakan LVQ (Learning Vector

Quantization). Pada penelitian sebelumnya data citra daun kedelai dilatih dengan menggunakan LVQ, citra daun di *preprocessing* terlebih dahulu dengan menggunakan *Otsu* yang kemudian di dapatkan bagian yang berpenyakit saja. Citra daun kemudian dilakukan ekstraksi fitur dengan mengambil rata-rata dari nilai *red, green, blue*, minimum *red,* minimum *green*, minimum *blue,* maksimum *red*, maksimum *green*, maksimum *blue*. Kemudian proses selanjutnya fitur dari citra daun dilatih dengan menggunakan LVQ dan kemudian diuji dengan nilai *threshold, learning rate,* nilai bobot awal, nilai pengurang *learning rate*, dan jumlah data latih. Dari proses tersebut didapatkan nilai akurasi sebesar 93% (Umam, et al., 2015). Metode yang dilakukan pada penelitian sebelumnya dengan menggunakan *basic otsu* dilakukan *preprocessing* pada 2 kelas saja yaitu penyakit *downy mildew* dan c*endawan jelaga*. Pada pengujian yang dilakukan lebih dari 2 kelas sistem belum bisa mengklasifikasi dengan baik. Oleh karena itu agar dapat dilakukan *preprocessing* lebih dari 2 kelas maka dilakukan *Multilevel Thresholding* dengan menggunakan *Otsu*.

Terdapat banyak algoritma dan metode untuk melakukan pemrosesan data pada citra daun tanaman jeruk, diantaranya adalah algoritma *Thresholding Biner, Otsu, Adaptive Thresholding*. Penelitian yang dilakukan oleh Ping-Sung Liao, Tse-Sheng Chen dan Pau-Choo Chung yang meneliti tentang *multilevel thresholding* menggunakan *otsu* berjudul *A Fast Algorithm for Multilevel Thresholding* yang diterapkan pada beberapa gambar didapatkan algoritma tercepat terletak pada algoritma *Otsu Thresholding* (Liao, et al., 2001). Penelitian yang dilakukan oleh Slamet Imam Syafi'i, Rima Tri Wahyuningrum, dan Arif Muntasa tentang pencarian akurasi nilai ambang yang diuji dengan beberapa proses yaitu dengan menggunakan *Invert Image, Noise Removal* dan *Otsu*, dengan judul *Segmentasi Objek pada Citra Digital Menggunakan Metode Otsu Thresholding*, nilai akurasi yang baik dimiliki oleh metode otsu yaitu sebesar 100% (Syafi'i, et al., 2015). Berdasarkan beberapa penelitian tersebut maka peneliti memutuskan menggunakan algoritma *Multilevel Thresholding* menggunakan *Otsu* sebagai metode pemrosesan data. Melalui penelitian yang akan dibuat ini, peneliti ingin mengetahui tingkat keakurasian dari algoritma *Multilevel Thresholding* menggunakan *Otsu* yang dipadukan dengan klasifikasi menggunakan KNN (*K-Nearest Neighbor*) pada proses pembelajaran data jika digunakan dalam kasus identifikasi penyakit tanaman jeruk. Berdasarkan latarbelakang, studi kasus dan metode yang telah dijabar maka peneliti memberi judul pada penelitian ini dengan judul **"Implementasi Algoritma *Multilevel Thresholding* Menggunakan *Otsu* Sebagai *Preprocessing* Data Citra Daun pada Proses Identifikasi Penyakit Tanaman Jeruk".**

1.2 Rumusan masalah

Rumusan masalah dari penelitian ini antara lain adalah sebagai berikut :

1. Bagaimana implementasi algoritma *Multilevel Thresholding* Menggunakan *Otsu* sebagai *Preprocessing* data citra daun pada proses identifikasi penyakit tanaman jeruk?

2. Bagaimana tingkat akurasi yang didapatkan dari implementasi algoritma *Multilevel Thresholding* Menggunakan *Otsu* sebagai *Preprocessing* data citra daun pada proses identifikasi penyakit tanaman jeruk?

1.3 Tujuan dan Manfaat

Manfaat penelitian yang didapatkan antara lain sebagai berikut :

1. Bagi Balai Penelitian Tanaman Jeruk dan Buah Subtropika (Balitjestro) penelitian ini dapat membantu peneliti untuk mengidentifikasi penyakit jeruk.
2. Bagi petani tanaman jeruk, penelitian ini dapat membantu petani untuk mengetahui penyakit CVPD, Mildew , defisiensi dan Jelaga pada tanaman jeruk
3. Bagi peneliti, penelitian ini dapat digunakan sebagai refrensi untuk pengembangan penelitian baru di bidang yang sama atau di bidang yang lain..

1.4 Batasan masalah

Batasan dalam menuliskan tugas akhir ini adalah :

1. Penelitian ini dibuat hanya untuk identifikasi penyakit CVPD, Cendawan Jelaga, Downy Mildew, defisiensi (Zn dan Mg) dan daun sehat.
2. Data daun yang digunakan hanya memiliki satu penyakit daun.
3. Pengambilan data daun digunakan kamera EOS 60D ISO 100 dengan jarak 20cm secara tegak lurus.
4. Jenis jeruk yang dipakai adalah jeruk keprok batu 55
5. Aplikasi ini dibuat dengan berbasis desktop.

1.5 Sistematika pembahasan

Sistematika pembahasan dalam penelitian ini terbagi dalam enam bab, sebagai berikut :

BAB 1 : Pendahuluan

Terdiri atas Latar Belakang, Rumusan Masalah, Manfaat, Batasan Masalah dan Sistematika Pembahasan

BAB 2 : Landasan Kepustakaan

Terdiri atas Kajian Pustaka, Teori Terkait Jeruk, Penyakit Jeruk, Pengolahan Citra Digital, Otsu dan k-nearest neighbor

BAB 3 : Metodologi

Terdiri atas Studi Literatur, Pengumpulan Data, Perancangan, Implementasi, Pengujian dan Analisis, Pengambilan Kesimpulan dan Saran.

BAB 4 : Perancangan

Menjelaskan tentang semua perancangan dari Implementasi Algoritma *Multilevel Thresholding* Menggunakan *Otsu* Sebagai *Preprocessing* Data Citra Daun pada Proses Identifikasi Penyakit Tanaman Jeruk

BAB 5 : Implementasi

Memuat proses dan implementasi terhadap aplikasi yang dibuat.

BAB 6 : Pengujian dan Analisis

Terdiri atas hasil dari pengujian terhadap sisitem yang telah dibuat yang kemudian akan di analisis.

BAB 7 : Penutup

Terdiri atas kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang telah dibuat dan berisikan saran-saran untuk pengembangan salanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisikan landasan kepustakaan yang membahas tentang metode maupun teori sebagai penunjang peneliti untuk penulisan dokumen penelitian ini. Landasan kepustakaan berisikan penelitian yang sudah dilakukan, baik memiliki persamaan khusus ataupun persamaan metode. Pada landasan kepustakaan juga dibahas beberapa teori yang terkait pada penelitian ini untuk di tulis pada dokumen penelitian ini. Beberapa teori yang dimasukkan pada landasan kepustakaan ini diantaranya Tanaman Jeruk, Penyakit Jeruk, Citra Digital, Pengolahan Citra Digital, metode *Otsu*, Multilevel Otsu, *K-Nearest Neighbor*.

2.1 Landasan Kepustakaan

2.1.1 Landasan Kepustakaan Terkais Kasus

Dalam pembuatan penelitian ini diperlukan juga penelitian yang juga terkait dengan identifikasi tanaman penyakit jeruk. Seperti penelitian yang dilakukan oleh Apiladosi Priambodo (Pirambodo, et al., 2015), dengan tujuan mengidentifikasi penyakit tanaman jeruk dengan metode KNN. Dalam menentukan penyakit tanaman jeruk, penelitian ini menggunakan metode *K-Nearest Neighbor* (KNN) sebagai klasifikasinya. Pada penelitian tersebut belum digunakan algoritma ataupun metode untuk *preprosecessing* citra. Untuk megetahui tingkat akurasi pada penelitian tersebut peneliti menggunakan 30 data latih untuk masing-masing penyakit dan 30 data uji, dan dari percobaan tersebut aplikasi berhasil mengidentifikasi penyakit dan 1 kali gagal teridentifikasi. Dari penelitian tersebut nilai akurasi yang didapat sebesar 96.67%. Aplikasi yang dirancang dalam penelitian ini berbentuk *desktop apps* yang dibangun dengan menggunakan Bahasa *Java* (Pirambodo, et al., 2015).

2.1.2 Landasan Kepustakaan Terkait Metode

Berdasarkan judul penelitian diatas, beberapa penelitian yang menerapkan metode *Otsu* sangat diperlukan dalam membuat penelitian ini. Oleh karena itu berikut adalah beberapa penelitian yang menerapkan metode *thresholding otsu*.

1. Penelitian pertama dilakukan oleh Zhonghua Lin dan Hongfei Yu yang berjudul *The Pupil Location Based on the OTSU Method and Hough Transform.* Pada penelitian tersebut, peneliti menggunakan metode *otsu* yang mencari lokasi pupil pada mata. Penelitian ini melakukan *dataset* sebanyak 384 gambar pupil yang merupakan citra warna RGB. Dari penelitian ini akurasi yang didapat adalah 100% (Lin & Yu, 2011).
2. Penelitian selanjutnya adalah penelitian yang dilakukan oleh Slamet Imam Syafi'I dkk yang berjudul *Segmentasi Obyek pada Citra Digital Menggunakan Metode Otsu Thresholding*. Pada penelitian ini dibagi menjadi lima proses yaitu *input* data, *preprocessing*, segmentasi, *cleaning*, dan perhitungan akurasi. Pada penelitian ini *preprocessing* yang dilakukan adalah menggunakan metode *otsu*. Penelitian ini menggunakan data citra sebanyak

30 citra digital RGB. Pada penelitian ini akurasi yang didapat pada proses segmentasi adalah sebesar 93.33% (Syafi'i, et al., 2015).

3. Penelitian yang berjudul *Segmentasi Citra Digital Ikan Menggunakan Metode Thresholding*, yang ditulis oleh Max. R. Kumaseh dkk meneliti penggunakan metode otsu untuk melakukan proses *thresholding* pada citra digital yang digabungkan dengan proses segmentasi berupa metode deteksi tepi dengan menggunakan *Canny*, proses dilasi atau yang disebut dengan proses penebalan citra dan juga proses Ekstraksi Kontur. Pada penelitian ini didapatkan hasil yaitu pemisahan objek mata ikan dengan nilai *threshold* (T) sebesar 61 (Kumaseh, et al., 2013).
4. Penelitian selanjutnya yang berjudul *Three-level Local Thresholding Berbasis Metode Otsu untuk Segmentasi Leukosit pada Citra Leukimia Limfoblastik Akut* yang ditulis oleh Eka Prakarsa Mandyartha dan Chastine Fatichah menjelaskan bahwa dengan *Three-level local thresholding* dengan menggunakan metode *Otsu* jauh lebih baik dibandingkan dengan *global thresholding* (Mandyartha & Fatichah, 2016).
5. Penelitian selanjutnya berjudul *Pengenalan Plat Nomor Kendaraan Ganjil Genap Menggunakan Metode Support Vector Machine (SVM) Berbasis Varians Projection* yang menggunakan metode *Otsu* pada *preprocessing* menunjukkan bahwa dengan metode *Otsu* sangat membantu dalam proses segmentasi sehingga membuat akurasi menjadi tinggi dan maksimal yaitu 97.41% (Riadi & Pramunendar, 2014).
6. Penelitian selanjutnya yang berjudul *Diagnosis of Diabetus Mellitus using K Nearest Neighbor Algorithm*, yang ditulis oleh Krati Saxena dkk meneliti penggunaan metode dari KNN untuk mendiagnosis penyakit *Diabetus Mellitus* dengan menggunakan beberapa data tes diantaranya : umur, jenis kelamin, IMT (Indeks Massa Tubuh), tekanan darah, Konsentrasi Gula, *Triceps Skin Fold*, serum insulin selama 2 jam, *Diabetes pedigree function*, tingkat kolesterol, berat badan. Penelitian tersebut dilakukan dengan menggunakan 100 data latih dan menggunakan 50 data latih. Metode KNN pada penelitian tersebut diuji dengan nilai *K* sebanyak 2 yaitu k=3 dan k=5. Nilai *K* tersebut diuji dengan mencari nilai akurasi dan nilai *error* pada masing-masing data uji. Pada penelitian tersebut didapatkan nilai *k*=5 dengan akurasi terbesar pada data uji yang pertama yaitu sebesar 75%. (Saxena, et al., 2014).

2.2 Jeruk Keprok

Jeruk keprok atau yang disebut sebagai *Citrus Reticula* merupakan jeruk yang dapat tumbuh pada ketinggian 500 – 1200 meter diatas permukaan air laut. Jeruk ini memiliki buah yang besar dengan rasa yang manis keasaman. Jeruk keprok Siam merupakan jeruk keprok unggul. Kategori keprok unggul diantaranya : Kulit buah yang mudah dikupas, kulit daging buahnya mudah dipisah-pisahkan, dan daging biahnya lembut, sarinya banyak, rasanya manis dan bijinya sedikit. (Kanisius, 1994)

Gambar 2.1 Jeruk Keprok

2.3 Penyakit Jeruk

2.3.1 Cendawan Jelaga

Penyakit Cendawan Jelaga merupakan penyakit musiman yang menyerang tanaman jeruk. Penyakit ini biasanya menyerang tanaman jeruk saat musim hujan tiba, hal itu dikarenakan pada saat musim hujan biasanya cuaca menjadi lembab dan menyebabkan semakin mudahnya penyakit ini tersebar. Gejala pada penyakit cendawan jelaga adalah bercak hitam pada daun yang menempel pada daun (Semangun, 2004).

Gambar 2.2 Penyakit Cendawan Jelaga

2.3.2 Downy Mildew

Penyakit ini sering disebut sebagai penyakit Embun tepung, karena pada gejalanya penyakit ini menyerupai warna putih yang menyerupai tepung yang bewarna putih. Embun Tepung atau *Downy Mildew* ini menyerang pada bagian daun dan biasanya hal ini disebabkan oleh Jamur atau *Fungi*. Daun yang terserang *Downy Mildew* akan pucat dan juga akan mengalami kerontokan (Semangun, 2004).

Pengaruh terbersar pada penyakit ini adalah karena letak penanaman jeruk yang berada pada dataran tinggi. Penyakit ini muncul pada tempat yang cukup lembab disaat sinar matahari menerangi tanaman jeruk tersebut. Saat ini penanganan yang dilakukan pada penyakit *Downy Mildew* ini adalah berdasarkan pada tingkat keparahan yang dihitung pada luasan daun yang terkena, selanjutnya

dilakukan penyemprotan bubur California secara berkala pada daun yang terjangkit ataupun tidak sebagai upaya pencegahan (Triwiratno, 2016).

Gambar 2.3 Penyakit Downy Mildew

2.3.3 CVPD (*Citrus Vein Phloem Degeneration*)

CVPD atau *Citrus Phloem Degeneration* adalah penyakit yang dikenal secara internasional bernama *Huanglungbin (HLB)*. Penyakit CVPD ini merupakan penyakit yang paling ditakuti oleh petani jeruk di Indonesia. Penyebab dari penyakit ini adalah Kutu loncat *Diapohorina citri Kuw.*, hama ini biasanya menyerang pada hampir seluruh morfologi tanaman jeruk. Hama Kutu loncat akan memasukkan patogen kedalam jeruk yang akan menutup saluran pemberian nutrisi yang dibutuhkan oleh buah jeruk maupun morfologi lainnya. Hal ini menyebabkan CVPD menjadi penyakit yang mematikan dan sekaligus momok bagi petani jeruk di Indonesia. Ancaman CVPD ini menyebabkan menurunnya produktifitas, kualitas dan bahkan kematian pad tanaman jeruk itu sendiri. Bahkan penyakit ini bisa saja disebut sebagai penyakit *silent killer* di karenakan CVPD menyerang pada bagian jaringan *phloem* batang jeruk yang mempunyai fungsi untuk mengantarkan nutrisi kepada daun, jeruk ataupun bunga (Dwiastuti, et al., 2016).

Apa bila sebuah jeruk sudah terjangkit penyakit CVPD, akan terasa sukar untuk disembuhkan, yang dapat dilakukan adalah melakukan *monitoring* terhadap lingkungan disekitar area jeruk. Monitoring ini dilakukan dengan menggunakan perangkap *Yellow Trap* yang dipasang diantara pohon jeruk setinggi setengah tanaman dari jeruk tersebut. Selain itu pencegahan yang dilakukan adalah dengan metode penyaputan batang dengan insektisida dosis tertentu yang dilakukan secara berkala (Balitjestro, 2015).

Gambar 2.4 Penyakit *Citrus Vein Phloem Degeneration*

2.3.4 Defisiensi

Defisiensi merupakan penyakit kekurangan unsur hara atau nutrisi yang dibutuhkan oleh tanaman jeruk. Gejala yang dialami pada tanaman jeruk yang mengalami defisiensi adalah adanya perubahan warna pada daun ataupun pada buah. Jeruk yang kekurangan nutrisi atau unsur hara tembaga/*Cuprum* (Cu) contohnya akan menyebabkan perubahan warna daun menjadi hijau gelap dan berukuran besar. Selain itu tanaman jeruk yang kekurangan nutrisi Mangan (Mn) akan mengalami perubahan warna kuning atau *Bloching* yang membentuk pola pada daun jeruk (Triwiratno, 2016).

Gambar 2.5 Penyakit Defisiensi

2.4 Pengolahan Citra Digital

Pengolahan Citra Digital adalah salah satu teknologi yang menerapkan algoritma dalam komputer untuk memproses sebuah gambar digital. *Output* dalam pengolahan citra digital adalah gambar yang diolah memiliki karakteristik dari gambar original (Zhou, et al., 2010). Karakteristik yang dimaksud adalah gambar diolah menjadi memilki kualitas citra, transformasi, ataupun pemilihan ciri yang optimal untuk tujuan analisa, pengambilan informasi atau pengenalan citra (Sutoyo, et al., 2009).

2.4.1 Rescaling

Proses ini adalah melakukan penambahan tingkat kecerahan suatu citra yang menggunakan operator *Rescaling*. Proses ini dilakukan dengan mengalikan setiap warna pada piksel citra dengan nilai *scale factor* kemudian menambahnya dengan nilai *offset.* Persamaan proses ini adalah sebagai berikut (2.1) (Knudsen, 1999):

$$c = scaleFactor.c_o + offset \quad (2.1)$$

Dengan c sebagai citra setelah penyesuaian kecerahan dan c_0 sebagai citra sebelum citra ditambah kecerahannya. Nilai *scaleFactor* dan *offset* adalah nilai konstanta yang menentukan tingkat kecerahan suatu citra. Piksel warna akan dilakukan dengan nilai *scaleFactor* yang nantinya dijumlahkan dengan nilai *offset*.

2.4.2 Grayscaling

Proses ini dilakukan dengan mencari rata-rata dari nilai *red, green, blue* pada semua pixel yang terdapat pada sebuah citra. Setelah itu nilai rata-rata akan menggantikan nilai *red, green, blue* pada pixel citra. Citra yang dihasilkan dari proses *Grayscaling* adalah citra menjadi bewarna abu-abu. *Grascaling* ditentukan dengan persamaan sebagai berikut : (Kanan & Cottrell, 2012)

$$gray(x,y) = \frac{1}{3} * (red(x,y) + green(x,y) + blue(x,y)) \quad (2.2)$$

2.4.3 Segmentasi

Segmentasi citra adalah membagi citra kedalam sebuah wilayah yang homogen berdasarkan kriteria tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel tetangga-tetangganya (Syafi'i, et al., 2015).

2.4.4 *Thresholding*

Proses *thresholding* dilakukan untuk mendapatkan citra biner yang secara matematis dapat ditulis dalam persamaan (2.3) (Kumaseh, et al., 2013):

$$g(x,y) = \{^{1 \; if \; f(x,y) \geq T}_{0 \; if \; f(x,y) < T}\} \quad (2.3)$$

Dengan nilai *g(x, y*) adalah nilai *grayscale* dari sebuah citra, sedangkan nilai *T* adalah nilai *thresholding* yang diinputkan oleh sistem. Nilai *T* disini memiliki peran penting untuk melakukan proses thresholding pada *preprocessing*.

2.5 *Otsu*

Metode *Otsu* digunakan untuk mendapatkan nilai *threshold* optimal, untuk memisahkan bagian daun yang berpenyakit ataupun tidak. *Otsu* merupakan salah satu metode untuk segmentasi citra digital dengan nilai ambang secara otomatis, yakni dengan mengubah citra digital warna abu-abu menjadi hitam atau putih berdasarkan nilai ambang dengan nilai warna piksel citra digital (Syafi'i, et al., 2015).

2.5.1 Konsep *Otsu*

Untuk mendapatkan nilai *threshold* langkah pertama adalah dengan mengubah citra kedalam histogram. Dari histogram diketahui jumlah piksel untuk setiap tingkat keabuan. Tingkat keabuan dimulai dari *i* sampai dengan *L*. Dengan nilai *i* dimulai dari 1 yaitu piksel ke 0, dan nilai maksimum untuk *L* adalah 256 dengan nilai piksel 255 (Syafi'i, et al., 2015).

Nilai ambang yang akan dicari dari suatu citra *grayscale* dihitung dengan nilai *i*. Nilai *i* berkisar antara 0 sampai dengan *L-1*. Jadi probablititas setiap piksel dinyatakan dengan persamaan 2.4 (Syafi'i, et al., 2015) :

$$Pi = \frac{ni}{N} \quad (2.4)$$

Dimana :

Pi = probabilitas piksel ke-i

ni = jumlah piksel dengan tingkat keabuan i

N = total jumlah piksel citra

Langkah selanjutnya adalah formulasi untuk menghitung jumlah kumulatif ke 1 (2.5), rerata kumulatif (2.6) dan nilai rerata intensitas global (2.7) yang berturut-turut sebagai berikut :

$$\omega(k) = \sum_{i=0}^{k} pi \quad (2.5)$$

$$\mu(k) = \sum_{i=0}^{k} i.pi \quad (2.6)$$

$$\mu_T(k) = \sum_{i=0}^{L-i} i.pi \quad (2.7)$$

Dimana :

ω = jumlah kumulatif

μ = rerata kumulatif

μ_T = rata-rata intensitas global

Nilai ambang *k* yang menyatakan tingkat keabuan dimana setiap piksel akan dihitung. Langkah selanjutnya adalah menentukan varian antar kelas (*between class variance*) pada persamaan (2.8) yang kemudian dicari nilai maksimal yang digunakan sebagai *threshold* pada persamaan (2.9) (Syafi'i, et al., 2015) :

$$\sigma_B^2(k) = \frac{[\mu_{T\omega}(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \quad (2.8)$$

Dengan

$$\sigma_B^2 = \arg max_{1 \le x \le L} \sigma_B^2(k) \quad (2.9)$$

Nilai *k* bertujuan untuk menemukan *threshold* dari sebuah citra *grayscale*, nilai ambang atau *threshold* digunakan sebagai acuan untuk mengubah nilai ke biner (Syafi'i, et al., 2015).

2.6 *Multilevel Thresholding* Menggunakan *Otsu*

Multilevel thresholding merupakan konsep penentuan *thresholding* berdasarkan kelas-kelas atau level yang dicari secara otomatis. *Multilevel thresholding* menggunakan *otsu* merupakan proses pencarian nilai ambang batas secara maksimal dengan memanfaatkan antara kelas varians dalam gambar *grayscale* (Huang, et al., 2011).

2.6.1 Konsep *Multilevel Thresholding* Menggunakan *Otsu*

Pencarian nilai multilevel otsu dimulai dari pemberian nilai ambang pada variabel *P* dan *S* (Liao, et al., 2001). Dimana variabel *P(u, v)* yang merupakan *interval zeroth-order* dan juga variabel *S(u, v)* merupakan *interval first-order*.

$$P(u,v) = \sum_{i-1}^{v} P_i \quad (2.10)$$

Dan

$$S(u,v) = \sum_{i-1}^{v} i * P_i \quad (2.11)$$

Serta indeks u = 1, pada persamaan (2.11) dituliskan secara rekursif

$$P(1,v+1) = P(1,v) + P_{v+1} \quad (2.12)$$

$$S(1,v+1) = S(1,v) + (v+1) * P_{v+1} \quad (2.13)$$

Dimana nilai v+1 merupakan nilai histogram.

Dari persamaan tersebut didapatkan

$$P(u,v) = P(1,v) - P(1,u-1) \quad (2.14)$$

Dan

$$S(u,v) = S(1,v) - S(1,u-1) \quad (2.15)$$

Setelah didapatkan nilai P dan S baru maka dilakukan pencarian persamaan untuk didapatkan nilai *H* dengan. Nilai *H* merupakan modifikasi dari varian antar kelas pada persamaan (2.8) (Liao, et al., 2001).

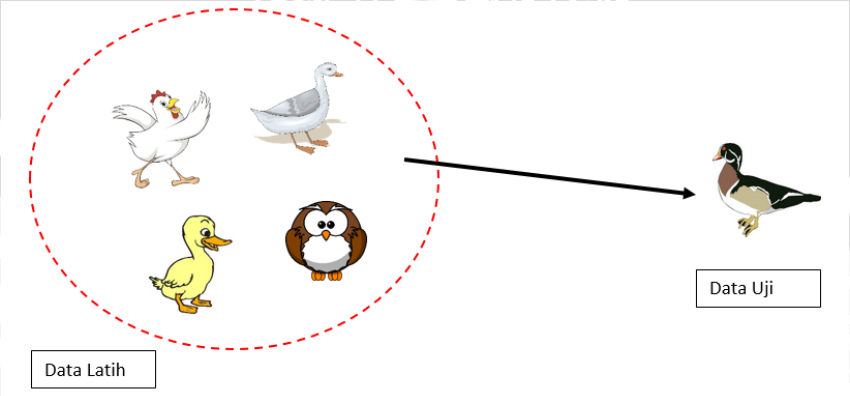
$$H(t_{i-1}+1, t_i) = \frac{S(t_{i-1}+1,t_i)^2}{P(t_{i-1}+1,t_i)} \quad (2.16)$$

Setelah didapatkan nilai *H* proses selanjutnya adalah pencarian nilai *T*, dimana nilai *T* merupakan nilai yang menentukan nilai ambang/*threshold* pada citra. nilai *T* dituliskan pada persamaan (2.17) (Liao, et al., 2001).

$$(\sigma_B{}^2)(t_1, t_2 t_i, \dots, t_{M-1}) = H(1,t_1)\text{+H}(t_1\text{+1,t2)+...+H}(t_{M-1}+1, L) \quad (2.17)$$

2.7 *K-Nearest Neighbor* (*KNN*)

Konsep dasar pada algoritma KNN adalah menentukan kelas berdasarkan kesamaan yang tinggi dari sebuah data latih (Muflikhah, et al., 2006). Nilai uji akan banyak bernilai benar apabila data latih yang dimiliki oleh KNN semakin banyak. Pada Gambar 2.1 Adalah visualisasi dari algoritma KNN.



Gambar 2.6 Ilustrasi Metode *K-Nearest Neighbor* (Muflikhah, et al., 2006)

Pencarian hasil dari data uji dilihat dari *class* yang sering muncul dilihat dari nilai *K* pada KNN. Pada kasus diatas terdapat sebuah data uji yang ingin dicari apakah termasuk dalam kategori/*class* pada sebuah data latih tersebut. Sebelum dicari

data uji termasuk dalam salah satu kategori dalam data latih, terdapat nilai variabel atau fitur yang dimiliki dalam sabuah data tersebut. Dari fitur tersebut dicarilah jarak pada masing-masing fitur yang dimiliki pada data latih, setelah dicari jarak yang ada maka di urutkan dari yang terbesar menjadi yang terkecil yang kemudian dicari berdasakan nilai *K*, kategori atau *class* yang sering muncul.

2.7.1 Algoritma *K Nearest Neighbor*

Tahap-tahap algoritma dari *K-Nearest Neighbor* adalah sebagai berikut (Wibowo & Usman, 2010)

1. Definisikan nilai *'K'*
2. Perhitungan jarak dengan *Euclidean Distance* antara data uji dengan data latih
3. Membuat kelompok jarak dan mengatur tetangga terdekat berdasarkan nilai *K*
4. Membuat grup dari nilai tetangga terdekat
5. Memilih nilai terdekat dari nilai yang sering muncul berdasarkan nilai *K*

2.7.2 Euclidean Distance

Euclidean distance adalah perhitungan untuk mencari kesamaan antara dua *vector*. Rumus dari *Euclidean distance* adalah sebagai berikut

$$d = \sqrt{\sum_{i=1}^{n}(Pi - Qi)^2} \qquad 2.18$$

Deskripsi dari persamaan 2.9 adalah sebagai berikut :

n = input uji data-i

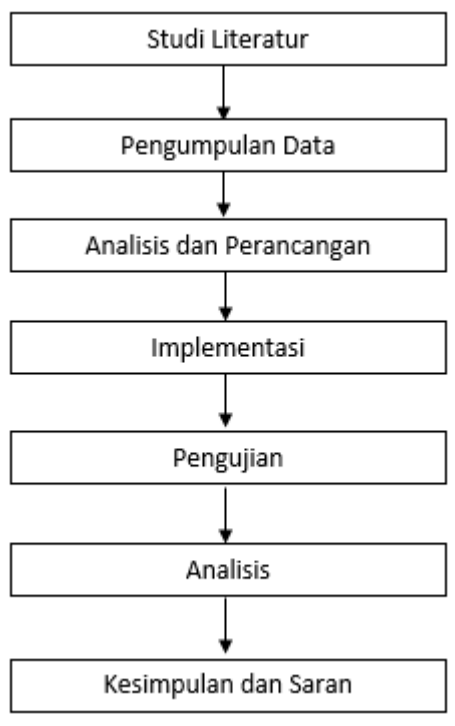
P = input latih data-i

Q = banyaknya data latih

d = Jarak *Euclidean* (Wibowo & Usman, 2010).

BAB 3 METODOLOGI

Pada bab ini akan dibahas tentang metodologi penelitian yang akan digunakan untuk implementasi metode *Multilevel Otsu* dalam kasus Identifikasi penyakit jeruk berdasarkan citra daun. Metodologi penelitian ini terdiri atas beberapa langkah yaitu studi literatur, pengumpulan data, perancangan sistem, implementasi sistem, pengujian dan pengambilan keputusan. Berikut ini adalah diagram alur dari metodologi yang akan dilakukan.



Gambar 3.1 Metodologi Penelitian

3.1 Studi Literatur

Studi literatur bertujuan sebagai bahan pembelajaran yang berhubungan dengan aplikasi yang akan dibuat, meliputi :

1. Bahasa pemrograman Java
2. Pengolahan Citra Digital
3. *Preprocessing* data dengan menggunakan *Multilevel Thresholding* Menggunakan *Otsu*.
4. Algoritma *K-Nearest Neighbor*
5. Pengakit Jeruk yang terdiri dari *Downy Mildew*, Cendawan Jelaga, *Citrus Vein Phloem Degeneration* (CVPD)

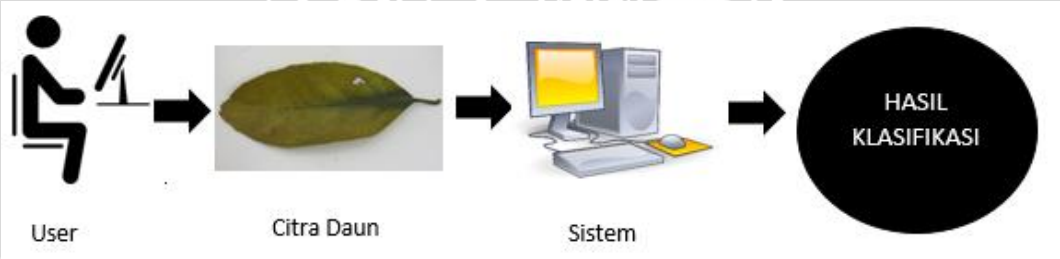
Berbagai literatur didapat dari penelitian sebelumnya, *e-book*, jurnal, buku dan juga beberapa literatur dari internet.

3.2 Pengumpulan Data

Pada penelitian ini pengambilan data di ambil pada Balai Penelitian Tanaman Jeruk dan Buah Subtropika (Balitjestro) yang berada di daerah Tlekung Kota Batu. Data yang digunakan adalah gambar yang terjangkit pengakit. Data yang diperoleh dari Balitjestro adalah daun yang terjangkit penyakit *Downy Mildew*, Cendawan Jelaga dan CVPD. Setelah itu data di foto dengan menggunakan kamera yang terdapat pada Balitjestro dengan tingkat kecerahan yang sama yaitu dengan menggunakan alat pada Laboratorium Terpadu Balitjestro pada waktu siang hari. Kamera yang digunakan adalah Canon EOS 60D dengan lensa 18-55 mm, ISO 100 yang diambil dengan jarak 20 cm secara tegak lurus. Masing-masing data diambil dengan jumlah 30 gambar setiap penyakit dan juga ditambah 30 daun yang sehat, yang nantinya data ini digunakan menjadi data latih. Sedangkan data uji yang dipakai terdapat 50 daun.

3.3 Perancangan

Aplikasi dirancang dengan memperhatikan kebutuhan-kebutuhan yang sudah dijelaskan diatas. Sistem akan menerima masukan berupa kebutuhan-kebutuhan yang sudah dijelaskan pada bab-bab sebelumnya, yang nantinya akan diolah atau diproses dengan menggunakan metode *Multilevel Thresholding* menggunakan *Otsu*, sehingga dapat menghasilkan *output* berupa kelas hasil dari klasifikasi penyakit dari citra daun jeruk.



Gambar 3.2 Proses pengenalan penyakit jeruk

3.4 Implementasi

Implementasi dibuat setelah melakukan proses perancangan selesai. Diharapkan pada proses implementasi dapat membangun aplikasi dengan menggunakan bahasa pemrograman Java dengan memperhatikan proses perancangan-perancangan yang telah dilakukan pada proses sebelumnya.

3.5 Pengujian

Tahap pengujian dibuat setelah melakukan proses perancangan dan implementasi sistem. Tahap ini bertujuan untuk mengetahui apakah aplikasi yang dibuat telah sesuai dan juga untuk mendapatkan tingkat keakurasian aplikasi setelah ditambahkannya *preprocessing* dengan metode *Multilevel Thresholding* Menggunakan *Otsu* yang nantinya diuji dengan metode *K-Nearest Neighbor*

dengan kondisi tertentu. Skenario pengujian yang dilakukan dalam penelitian ini adalah sebagai berikut :

a. Pengujian dengan nilai *Scale Factor*

   Nilai *scale factor* adalah suatu nilai konstanta yang terdapat pada algoritma *rescaling*. Pengujian ini dilakukan untuk mengetahui pengaruh nilai konstanta dari *scale factor* terhadap proses penambahan kecerahan sebelum diproses pada algoritma *Multilevel Otsu* pada data citra daun. Pengujian nilai *scale factor* dilakukan sebanyak 10 kali yaitu 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0.

b. Pengujian dengan nilai *MLEVEL*

   Nilai *MLEVEL* menentukan banyaknya segmentasi atau *threshold* terhadap citra daun dalam algoritma *Multilevel Thresholding* menggunakan *Otsu*. Pengujian ini dilakukan untuk mendapatkan besar nilai level yang nantinya digunakan dalam proses *Otsu* pada citra daun. Pengujian nilai *MLEVEL* dilakukan sebanyak yaitu level 1 level (didapatkan 2 *threshold*), 2 level (didapatkan 3 *threshold*), 3 level (didapatkan 4 *threshold*), 4 level (didapatkan 5 *threshold*), 5 level (didapatkan 6 *threshold*).

c. Pengujian dengan nilai *k*

   Pengujian ini dilakukan untuk mengetahui pengaruh dari nilai *K* saat dilakukannya proses perhitungan jarak *Euclidean* dari data uji dengan data latih. Tujuan dari pengujian ini adalah mendapatkan rekomendasi akurasi terbaik dari nilai *k*. Selain itu nantinya akan diketahui nilai apakah dengan nilai *K* terkecil dapat mendapatkan akurasi terbesar sehingga nantinya dapat meringankan pada proses komputasi sistem. Pengujian nilai *K* dilakukan sebanyak 5 kali dengan nilai 1, 2, 3, 4, 5.

3.6 Analisis

Analisis dilakukan untuk menentukan proses identifikasi penyakit tanaman jeruk berdasarkan citra daun dengan memperhatikan nilai *scale factor* pada proses rescaling, nilai *MLEVEL* pada proses *Multilevel Thresholding* menggunakan *Otsu* dan nilai *k* pada metode KNN.

3.7 Kesimpulan dan Saran

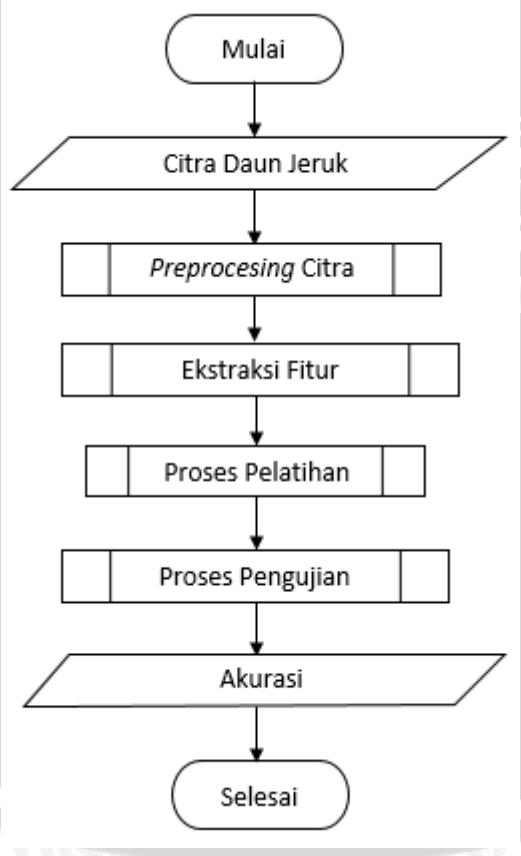
Proses pengambilan kseimpulan dilakukan setelah melakukan proses perancangan, implementasi dan juga pengujian. Kesimpulan di ambil dari hasil pengujian aplikasi dan saran juga dapat ditambahkan untuk mengembangkan penelitian yang sudah dibuat.

BAB 4 PERANCANGAN

Pada bab ini akan dibahas perancangan yang akan digunakan untuk mengimplementasikan aplikasi pengenalan penyakit daun jeruk dengan menggunakan citra dengan metode Multilevel Otsu. Perancangan pada bab ini terdiri atas perancangan Algoritma dan Perancangan Antarmuka.

4.1 Perancangan Algoritma

Pada sub bab ini akan dibahas tentang perancangan algoritma dengan menggunakan *flowchart* (diagram alir). Diagram alir sistem akan menggambarkan alur-alur sistem akan berjalan yang dikerjakan dengan sistem. Tujuan dari pembuatan diagram alir sistem adalah untuk memudahkan peneliti membuat sistem serta memudahkan untuk memahami alur proses terhadap sistem yang dibuat. Dalam sistem ini ada beberapa alur yang dibuat diantaranya Citra daun jeruk, *Preprocessing* Citra, Ekstraksi fitur, Proses pelatihan, Proses pengujian, Proses perhitungan akurasi.



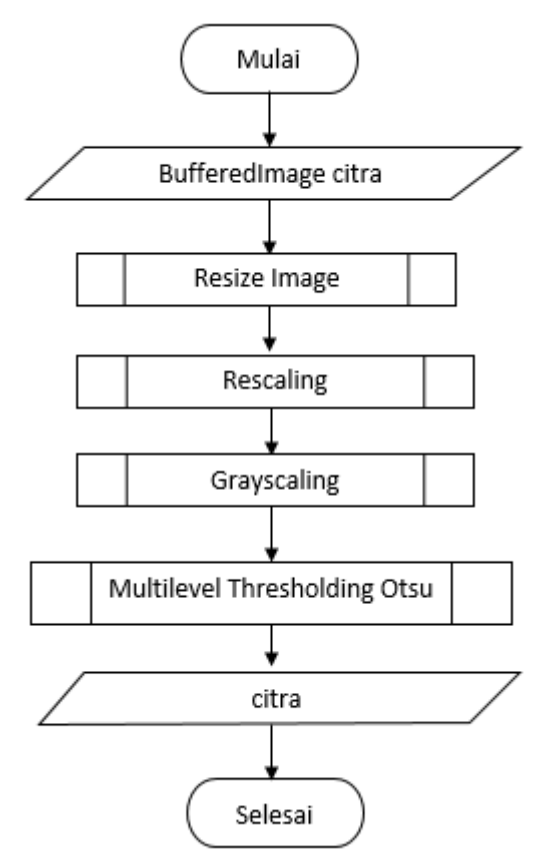
Gambar 4.1 Diagram Alir Sistem

Pada Gambar 4.1 dijelaskan alur-alur pengerjaan dimana aplikasi melakukan pembacaan Citra daun jeruk. Setelah itu sistem melakukan proses *Preprocessing* dengan menggunakan metode *Multilevel* Thresholding menggunakan Otsu untuk memisahkan bagian-bagian jeruk yang berpenyakit dan tidak. Setelah itu

dilakukan ekstraksi fitur dimana didapat nilai rata-rata dari warna merah (*Red*), Hijau (*Green*) dan Biru (*Blue*) dari citra yang berpenyakit tanpa latar belakang yang berwarna putih. Setelah itu sistem melakukan proses pelatihan dengan memasukkan rata-rata dari warna *Red, Green, Blue* untuk dimasukkan kedalam sebuah tabel sesuai dengan *class* penyakitnya. Setelah proses pelatihan selanjutnya dilakukan proses pengujian dengan melakukan perhitungan jarak *Euclidean* terhadap data latih. Pengujian dilakukan dengan menggunakan nilai *scale* factor pada *rescaling*, nilai *MLEVEL* pada *multilevel otsu,* dan nilai *k* pada *K-Nearest Neighbor.*

4.1.1 Preprocessing Citra

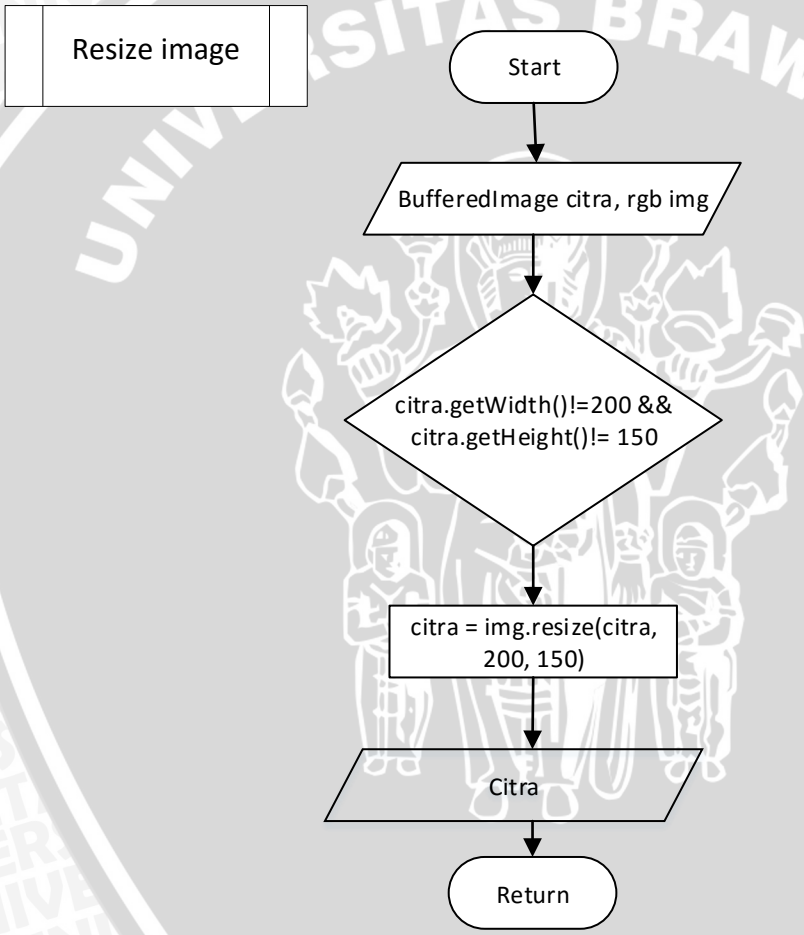
Preprocesing citra dilakukan selama 3 kali yaitu dengan proses *rescaling*, *grayscaling*, dan juga multilevel otsu. Setiap proses menghasilkan nilai citra baru yang diproses dengan melakukan ekstraksi fitur. *Rescaling* adalah proses untuk menambahkan tingkat kecerahan pada sebuah citra. *Grayscaling* adalah proses untuk mendapatkan nilai keabu-abuan pada citra dengan mencari rata-rata dari warna *red, green, blue*. Sedangkan untuk proses *Multilevel Thresholding Otsu* adalah proses untuk mendapatkan bagian dari daun yang berpenyakit untuk dilakukan proses ekstraksi fitur nilai RGB pada proses selanjutnya. *Preprocessing* citra dijelaskan pada Gambar 4.4.



Gambar 4.2 Diagram Alir *Preprocessing*

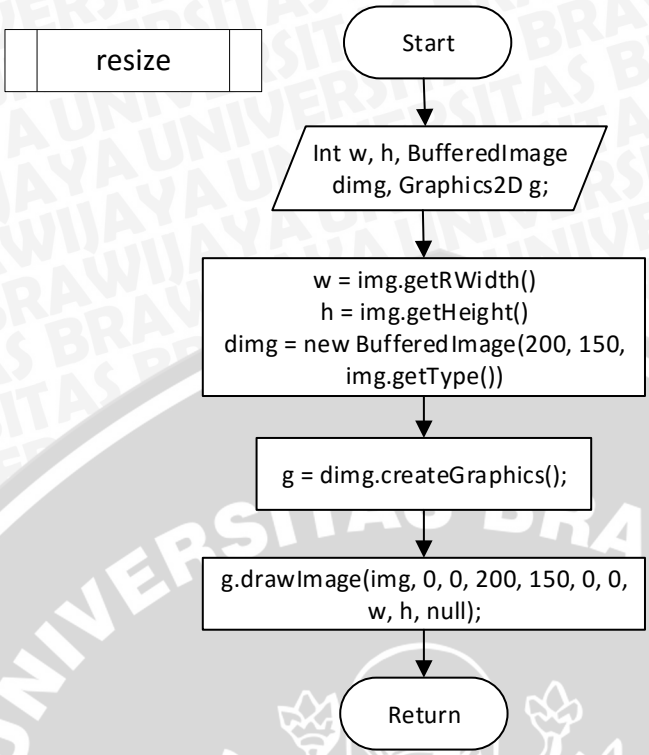
4.1.1.1 Resize Image

Pada penelitian ini, tahap proses *resize* digunakan dengan menggunakan kelas dari java yaitu *Graphics2D* dengan mengimplementasikan method *drawImage*. Dalam method *drawImage* memiliki beberapa parameter diantaranya adalah object dari kelas *BufferedImage*, nilai pixel pertama dari *image*, nilai *width* dan nilai *height* citra lama dan baru. *DrawImage* merupakan method yang memeungkinkan sebuah citra dilakukan *resize* sesuai dengan ukuran yang disesuaikan dengan inputan nilai *width* dan *height* yang baru. Hasil dari *resize image* adalah didapatkan citra baru dengan ukuran yang lebih kecil sehingga tidak membebankan memori yang banyak dalam proses perhitungan maupun tidak membutuhkan waktu yang lama dalam proses komputasi citra.



Gambar 4.3 Diagram Alir Proses Resize Image

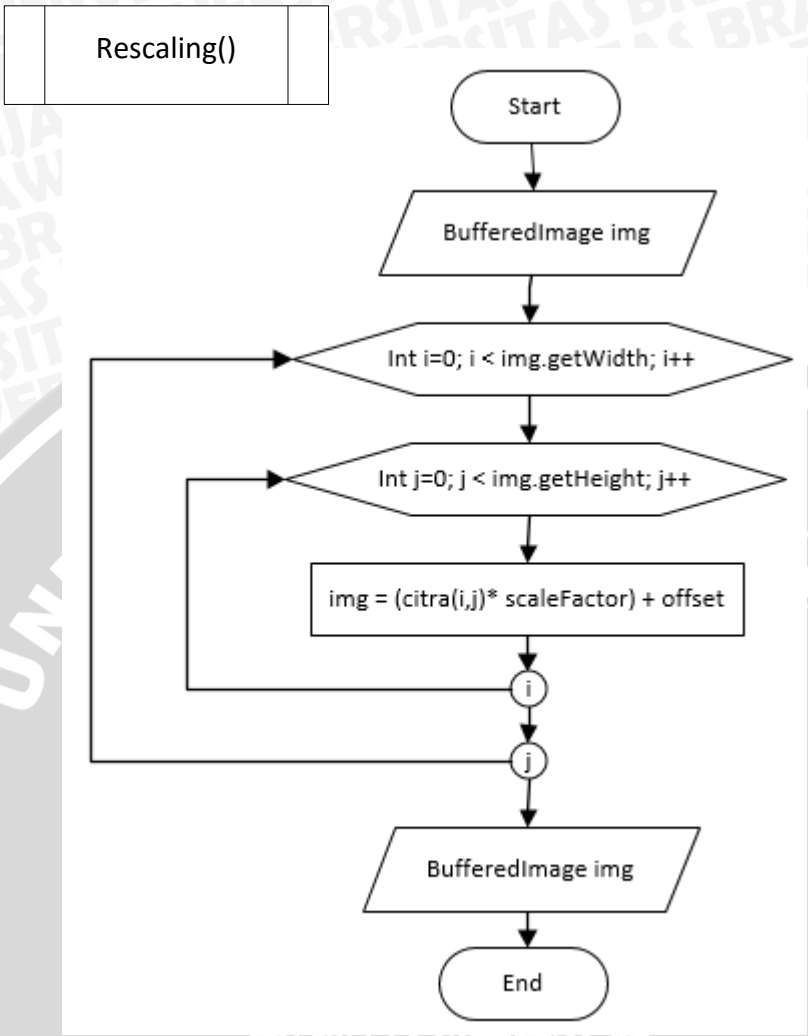
Method resize terdapat pada kelas rgb dengan mengimplementasikan method *drawImage* yang terdapat pada kelas *Graphics2D*. Proses *drawImage* dijelaskan pada gambar 4.3.



Gambar 4.4 Diagram Alir Resize Image

4.1.1.2 Rescaling

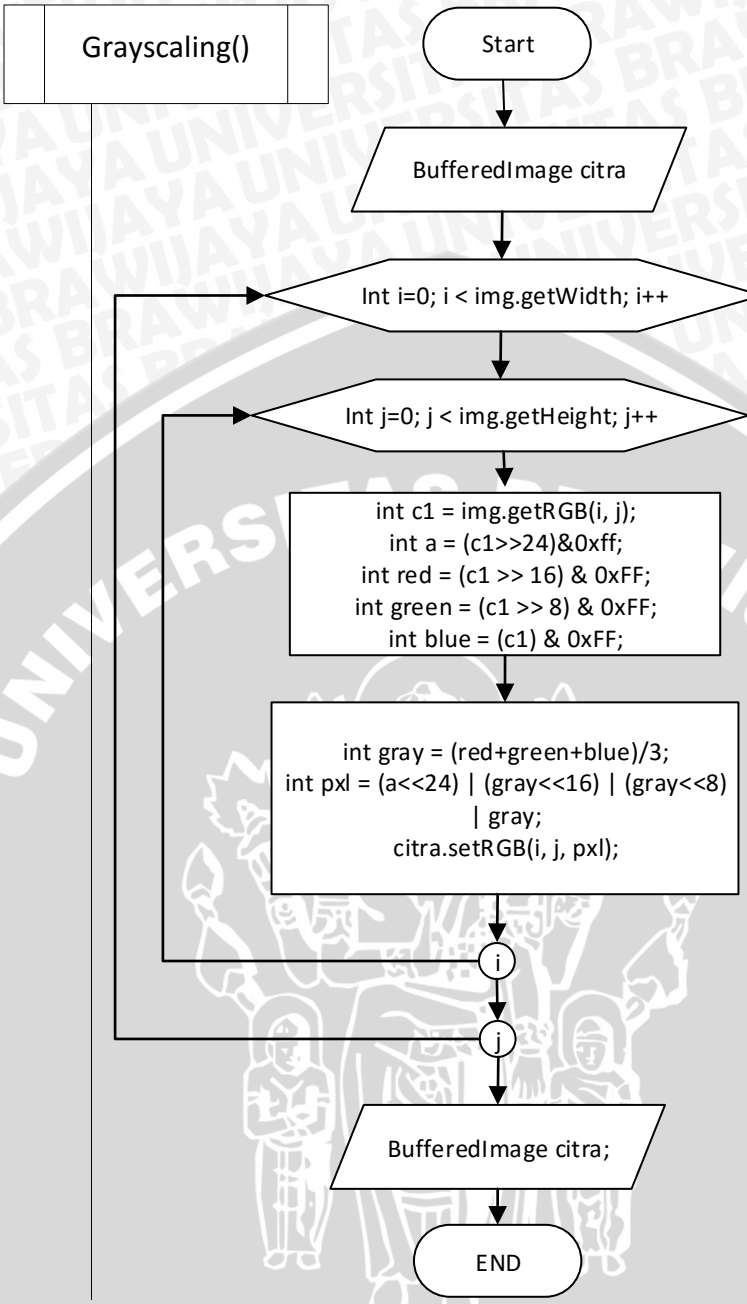
Pada proses ini citra ditambahkan dengan nilai konstanta untuk meningkatkan tingkat kecerahan sebuah gambar. Pada penelitian ini proses *rescaling* citra menggunakan kelas yang sudah disediakan pada java yaitu *RescaleOp. RescaleOp* memiliki 2 parameter yaitu nilai *scale factor* dan *offset*. Nilai *scale factor* dan *Offset* inilah yang nantinya memmpengaruhi tingkat kecerahan pada sebuah citra tersebut. Hasil output pada *rescaling* ini adalah sebuah citra dengan tingkat kecerahan yang berbeda dengan citra sebelumnya sehingga pada proses ekstraksi fitur bisa dilakukan tanpa melibatkan latar belakang daun jeruk.



Gambar 4.5 Diagram Alir Rescaling

4.1.1.3 Grayscaling

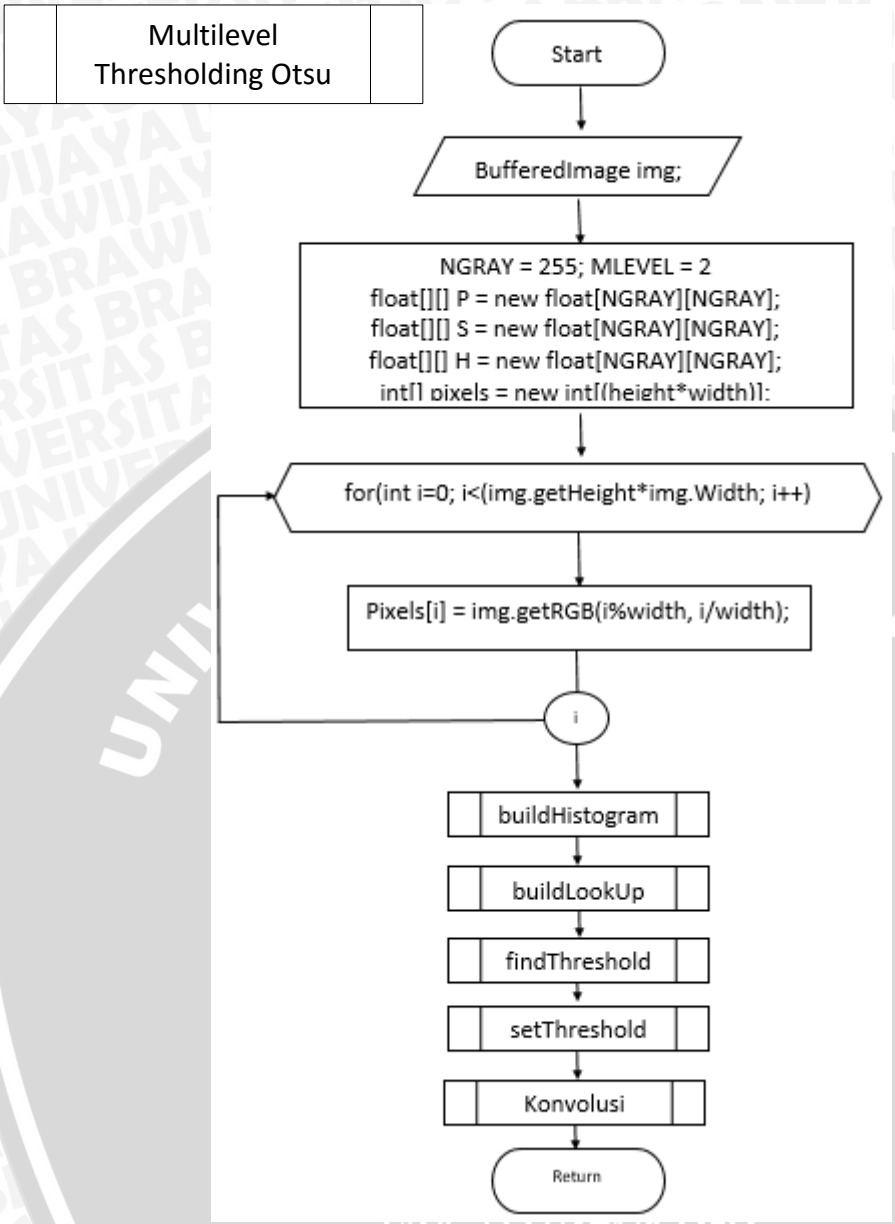
Pada proses ini citra menjadi bewarna abu-abu, hal ini dilakukan agar dapat memudahkan pada proses Segmentasi Multilevel Otsu dimana dicari nilai histogram untuk dilakukan thresholding pada setiap level yang dimasukkan. Proses ini dilakukan dengan mencari rata-rata dari nilai warna *red, green* dan *blue.* Nilai rata-rata yang didapat menjadi warna RGB baru pada citra output yang didapat dari proses *Grayscaling*.



Gambar 4.6 Diagram Alir *Grayscaling*

4.1.1.4 *Multilevel Thresholding Otsu*

Proses *Multilevel Otsu* memiliki beberapa proses diantaranya adalah proses mencari nilai *histogram*, pencarian nilai PSH, pecarian nilai *threshold* dan *konvolusi*.

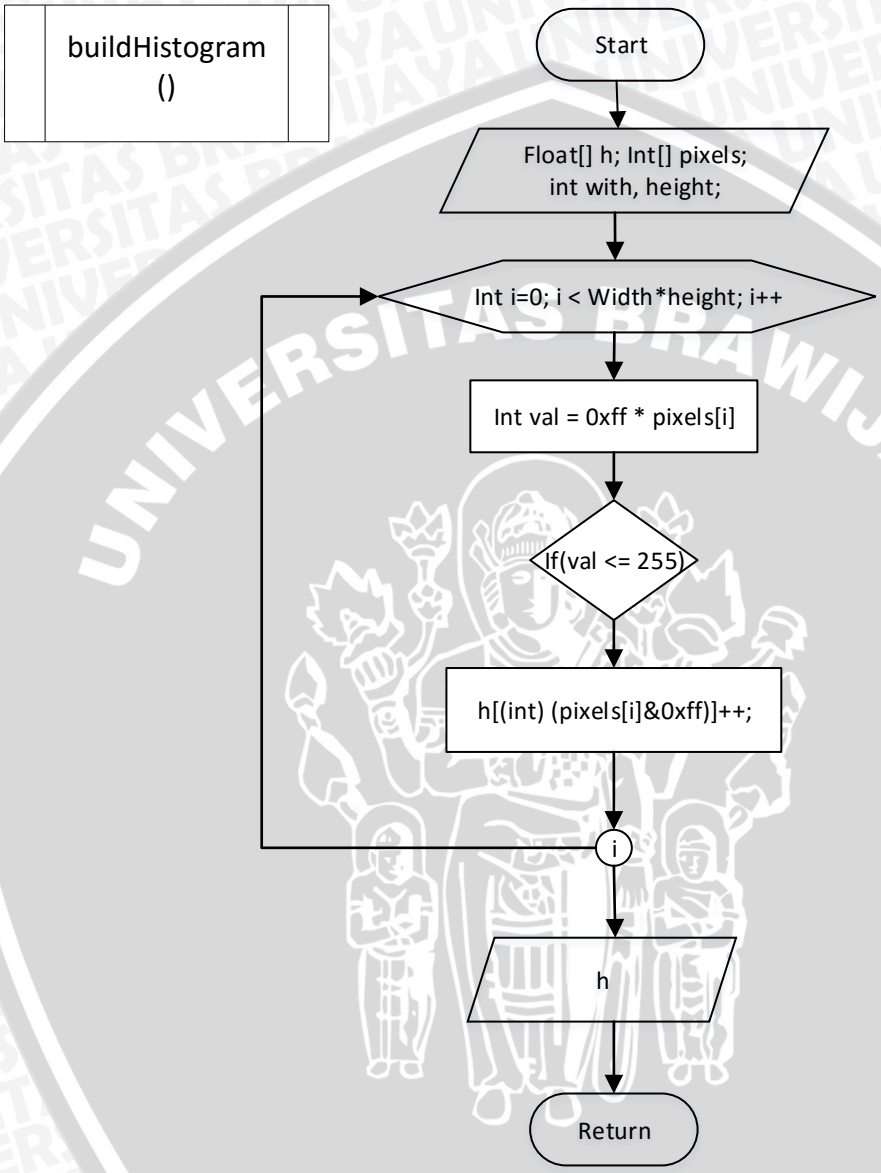


Gambar 4.7 Diagram Alir *Multilevel Otsu*

Proses *Multilevel Otsu* pertama-tama melakukan inisiasi variabel NGRAY yang digunakan sebagai banyaknya nilai gray pada sebuah citra dan juga nilai *MLEVEL* untuk mengetahui level otsu yang digunakan. Setelah itu pemberian nilai variabel array pixels dengan nilai RGB sebuah citra. Setelah itu dilakukan proses *builHistogram* yaitu untuk mencari nilai histogram dan juga mencari nilai rata-rata probabilitas pada sebuah citra. Setelah itu dilakukan proses *buildLookUp* yaitu proses pencarian nilai PSH. Selanjutnya proses *findThreshold* yaitu proses untuk mencari nilai *threshold*, dan yang terakhir adalah proses *konvolusi* yaitu proses untuk penambahan sebuah citra tersegmentasi dengan citra asli.

(a) Mencari nilai *Histogram*

Proses pertama pada *Multilevel Otsu* adalah pencarian nilai histogram, nilai histogram dicari untuk membangkitkan nilai keabuan sebuah citra RGB. Proses pencarian nilai histogram dapat ditunjukkan pada gambar 4.7.



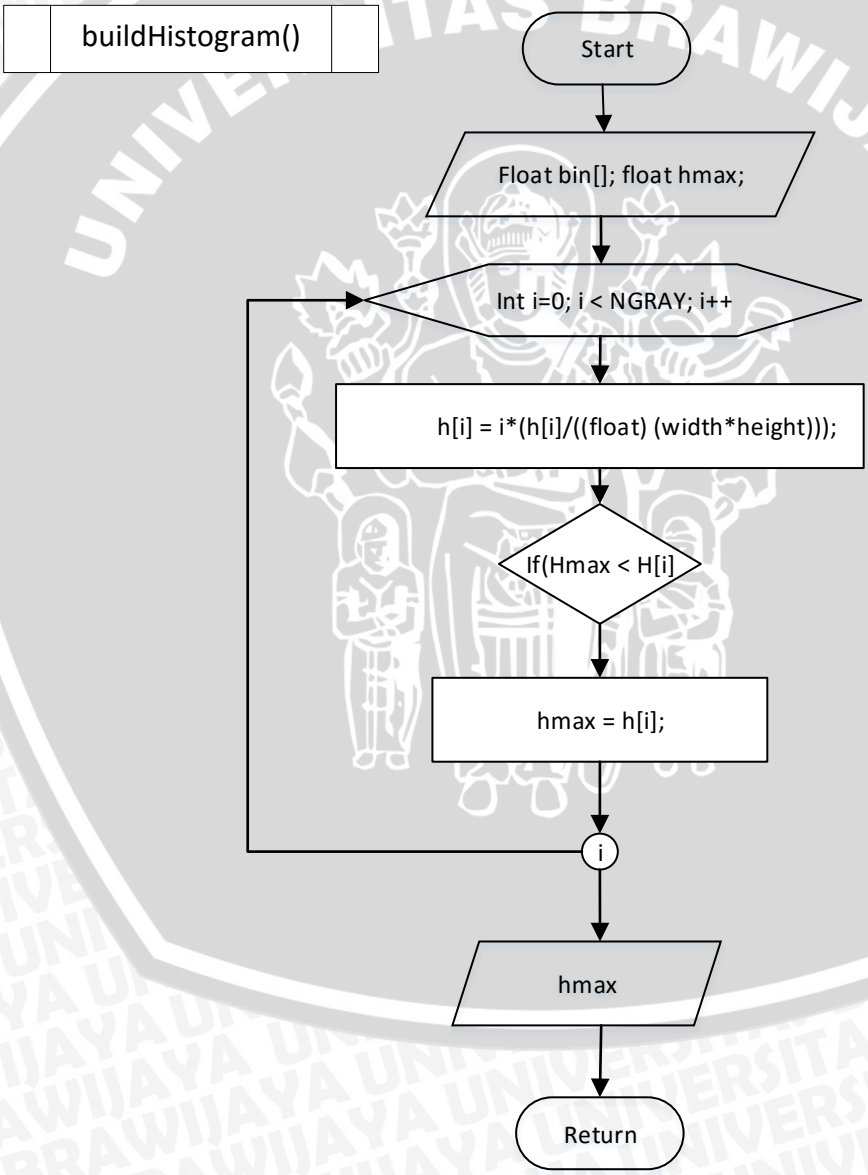
Gambar 4.8 Diagram Alir Histogram

Diagram alir pencarian histogram dibutuhkan nilai h yang bertipe float array yang digunakan penyimpanan nilai histogram setiap nilai pixels yang ada. Selain itu terdapat nilai pixels yang digunakan untuk menyimpan nilai RGB pixels yang sudah di inisiasi pada proses sebelumnya. Pada proses pencarian nilai histogram juga dibutuhkan variabel width dan height sebagai penentuan panjang dan lebar citra yang digunakan.

Proses pertama adalah melakukan perulangan dari nilai 0 hingga semua piksel citra yang ada. Perulangan yang digunakan adalah untuk mencari nilai

histogram setiap piksel yang ada. Didalam proses perulangan terdapat variabel val yang digunakan untuk membangkitkan atau mengambil nilai RGB pada sebuah piksel pada variabel pixels. Setelah itu terdapat seleksi kondisi dimana jika nilai val kurang dari atau sama dengan 255 maka proses selanjutnya adalah penambahan variabel histogram (h) sesuai dengan nilai *gray* yang sesuai. Dimisalkan nilai val adalah 50 maka proses selanjutnya adalah h[50]++. Proses *increment* atau i++ digunakan untuk menghitung nilai gray yang muncul pada sebuah piksel tertentu.

Setelah didapatkan sebuah nilai histogram, proses selanjutnya adalah mencari nilai rata-rata probabilitas histogram yang muncul sesuai dengan persamaan (2.6). Berikut ini adalah diagram alir pencarian rata-rata probabilitas histogram.

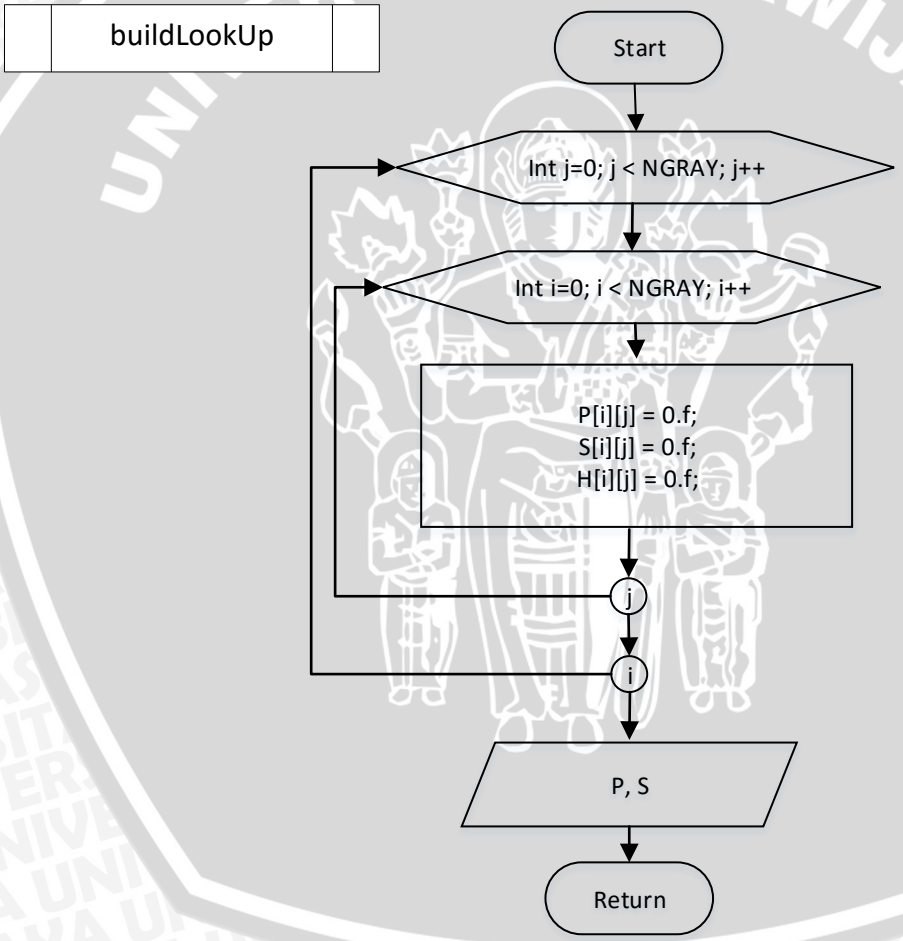


Gambar 4.9 Diagram Alir Probabilitas Nilai Gray

Pencarian nilai rata-rata probabilitas histogram yang muncul terletak dalam satu method *buildHistogram*. Proses pertama adalah melakukan perulangan dari 0 hingga banyaknya variabel NGRAY yaitu 255. Setelah itu proses pencarian nilai histogram adalah melakukan perkalian variabel perulangan i dengan nilai histogram dan di bagi dengan perkalian antara *width* dengan *height*. *Mencari nilai* P, S, H

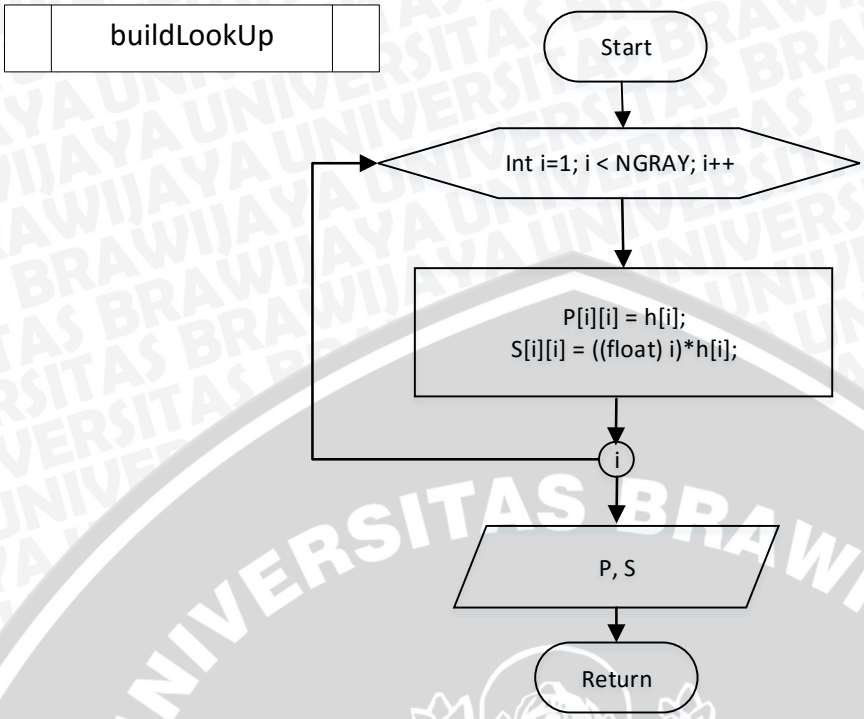
(b) Mencari nilai *P, S, H*

Proses pencarian nilai *multilevel thresholding* menggunakan otsu selanjutnya adalah pencarian nilai *P* atau yang disebut dengan *interval zeroth-order moment*, nilai *S* atau yang disebut dengan *interval zeroth moment* dan juga nilai *H* atau yang disebut dengan *modified between-class variance*. Proses pertama pada method ini adalah melakukan inisiasi pada variabel *P, S,* dan *H* dengan nilai 0 secara diagonal. Proses ini ditunjukkan pada Gambar 4.10.



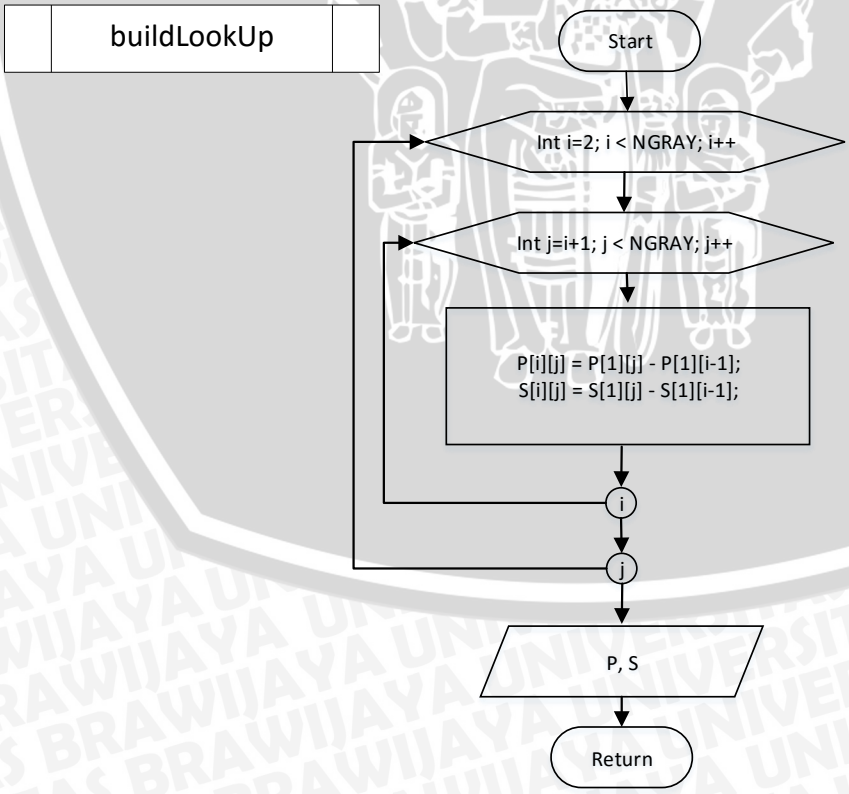
Gambar 4.10 Diagram Alir Pencarian Nilai *P* dan *S* Horizontal

Setelah dilakukan proses inisialisasi nilai *P* dan *S*, selanjutnya adalah melakukan pencarian nilai *P* dengan nilai histogram (*ni*) dan nilai *S* dengan nilai probabilitas histogram. Proses tersebut ditunjukkan pada Gambar 4.11.



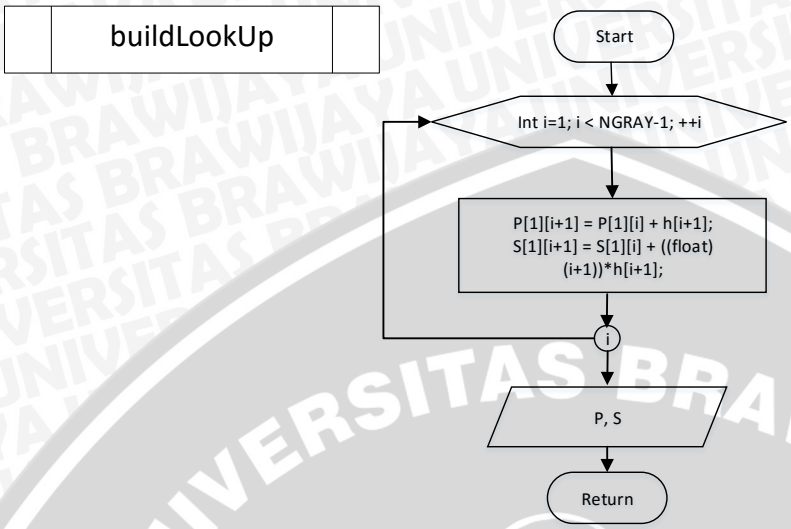
Gambar 4.11 Diagram Alir Pencarian Nilai *P* dan *S* dengan Histogram

Setelah itu proses selanjutnya adalah pencarian nilai *P* dan *S* dengan memperhatikan pada baris pertama saja. Proses ini dilakukan sesuai dengan persamaan (2.12) dan (2.13). proses tersebut ditunjukkan pada Gambar 4.12.



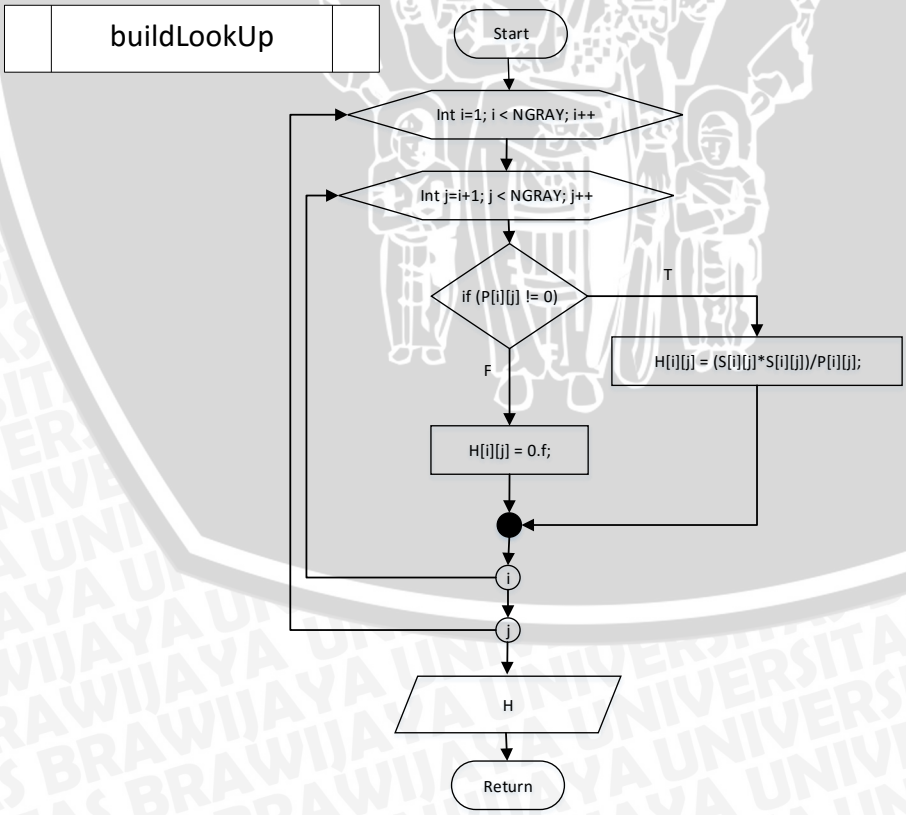
Gambar 4.12 Diagram Alir Pencarian Nilai *P* dan *S* Baris Pertama

Setelah itu diagram alir selanjutnya adalah pencarian nilai *P* dan *S* pada baris dan kolom lain selain pada proses sebelumnya. Diagram alir ini ditunjukkan pada Gambar 4.13.



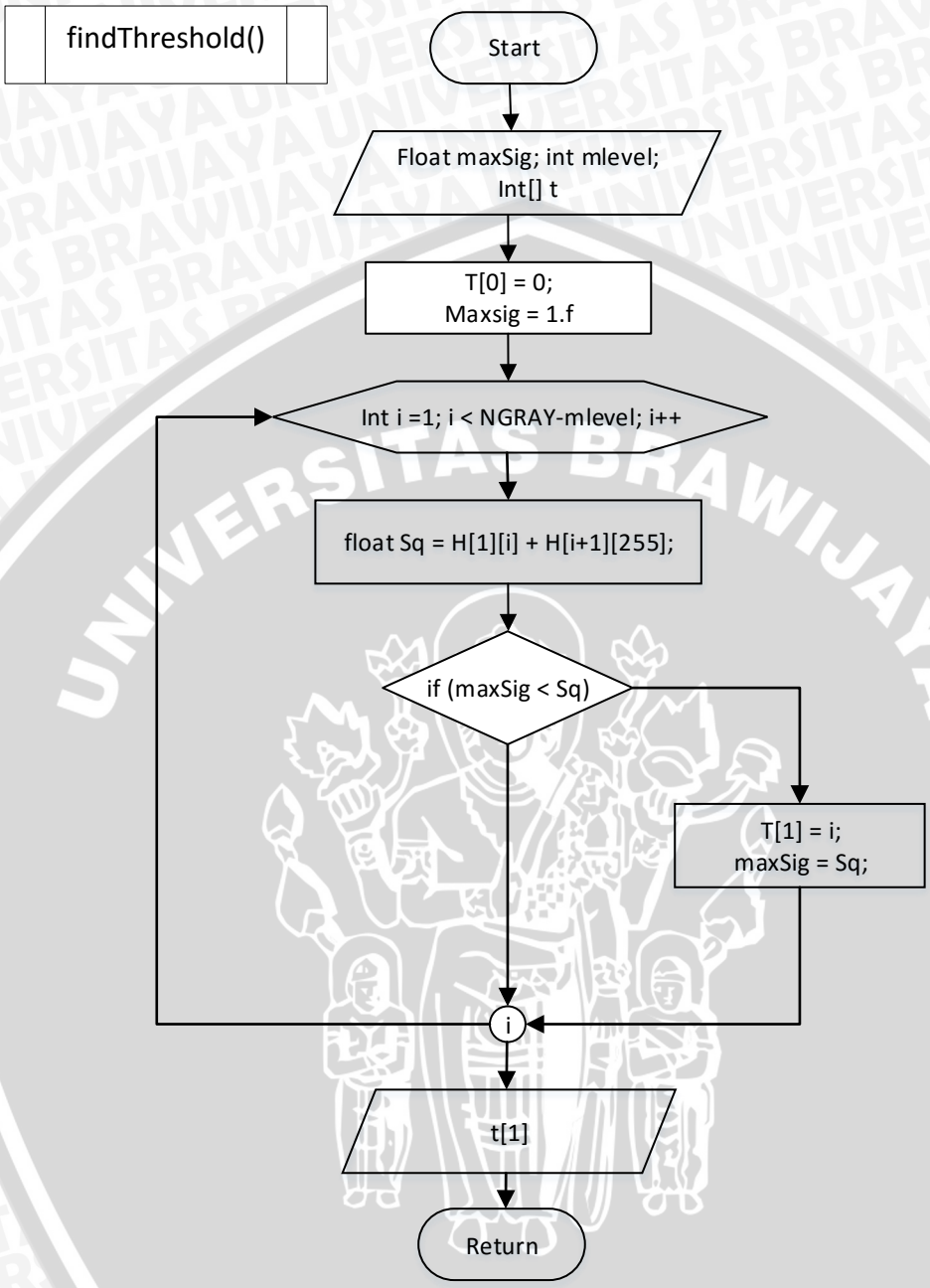
Gambar 4.13 Diagram Alir Nilai *P* dan *S*

Tahap akhir pada proses ini adalah didapatkannya nilai *H* dengan persamaan (2.16). Nilai *H* akan diseleksi jika pada nilai *P* sama dengan 0, maka nilai *H* akan menjadi bernilai 0. Diagram alir pada proses ini ditunjukkan pada Gambar 4.14.



Gambar 4.14 Diagram Alir Pencarian Nilai *H*

(c) Mencari nilai *threshold*

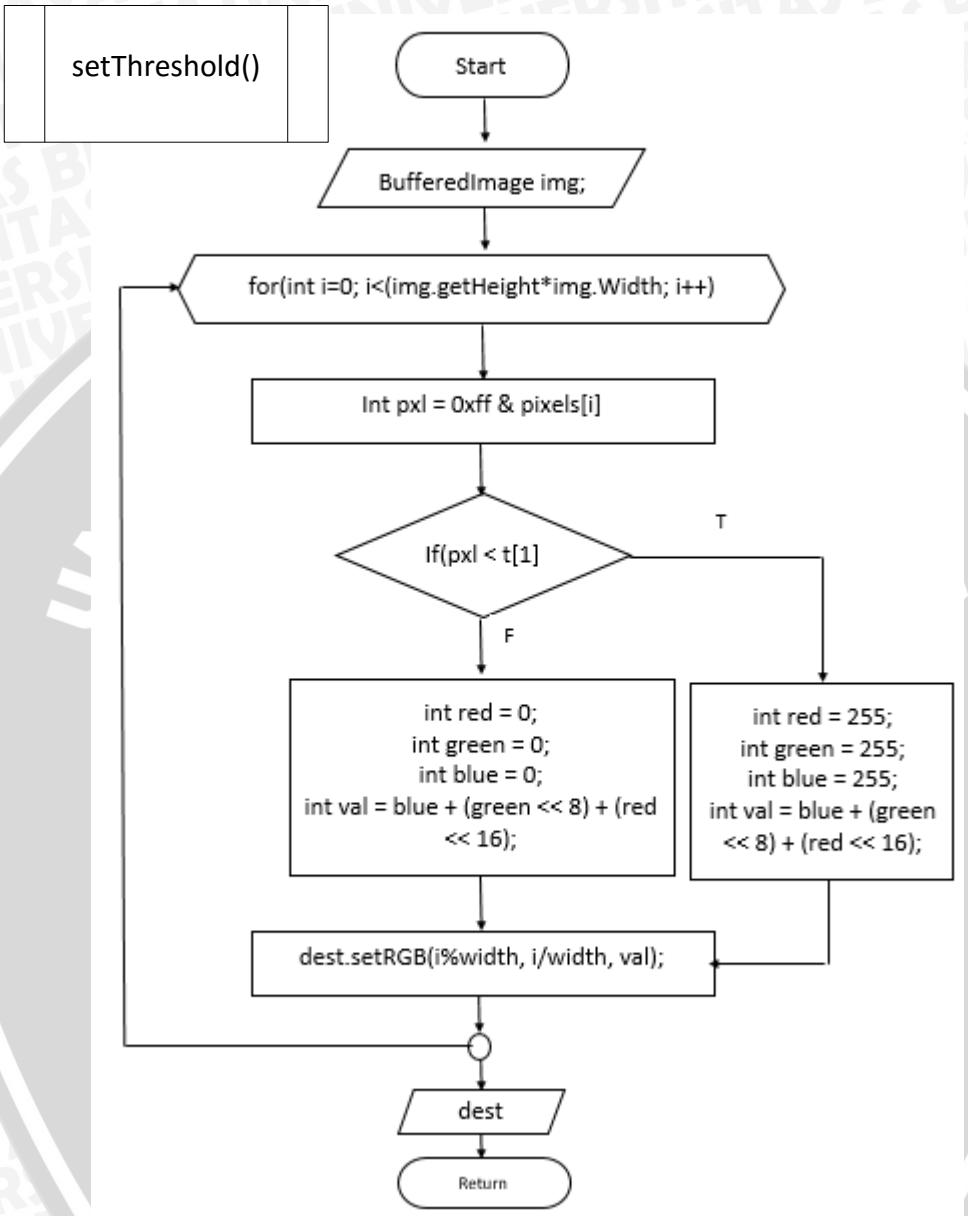


Gambar 4.15 Diagram Alir Mencari Nilai *Threshold*

Untuk melakukan pencarian nilai *threshold* diperlukan masukan nilai *MLEVEL*, nilai histogram dan nilai *threshold*. Proses pertama adalah pemberian nilai *threshold* ke-0 dengan nilai 0. Langkah selanjutnya adalah proses perulangan dimulai dari 1 hingga nilai variabel NGRAY-*MLEVEL*. Proses selanjutnya adalah pencarian nilai sq, setelah didapatkan nilai sq maka dilakukan seleksi kondisi antara variabel sq terhadap variabel maxSig. Jika proses seleksi kondisi menghasilkan nilai *true* maka dilakukan pemberian nilai *threshold* ke-1 dengan nilai i dan nilai nilai variabel maxSig

dengan nilai Sq. Output dari pencarian nilai threshold adalah variabel *threshold* ke-1.

(d) Pemberian batas nilai *threshold*



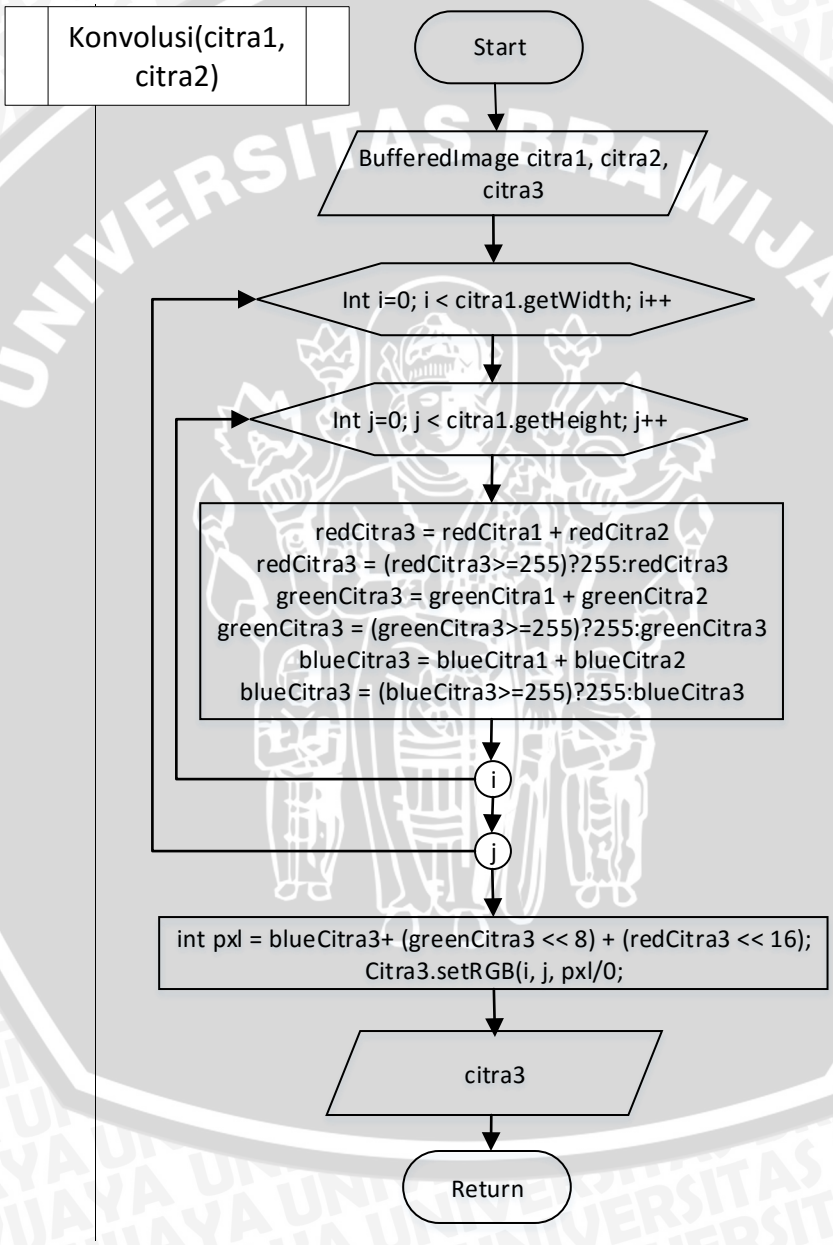
Gambar 4.16 Diagram Alir *setThreshold*

Proses pemberian nilai batas terhadap RGB pada citra yang sudah tersegmentasi dengan citra *grayscaling* merupakan proses untuk mengetahui bagian citra daun yang berpenyakit tetapi dalam format citra hitam putih. Proses pertama adalah inisiasi objek img oleh kelas *BufferdImage* setelah itu melakukan perulangan terhadap perkalian antara *variabel* panjang dan lebar citra img. Setelah itu pemberian nilai terhadap variabel pxl dengan nilai variabel *array* pixels. Selanjutnya dilakukan seleksi kondisi antara variabel pxl dengan nilai *threshold* ke-1, jika kondisi bernilai *true* maka dilakukan pemberian nilai variabel *red, green, blue* dengan nilai 255. Jika kondisi bernilai *false* maka

dilakukan pemberian variabel *red, green, blue* dengan nilai 0. Proses akhir adalah pemanggilan *method* setRGB pada objek dest. Output dari proses ini adalah objek dest yang sudah memilki RGB baru hasil dari pemberian batas nilai citra *grayscaling* terhadap nilai *threshold* ke-1.

(e) *Konvolusi*

Proses konvolusi merupakan proses penjumlahan antara RGB citra yang tersegmentasi dengan citra RGB *rescaling*. Proses ini dilakukan agar citra yang tersegmentasi dapat diketahui secara RGB.



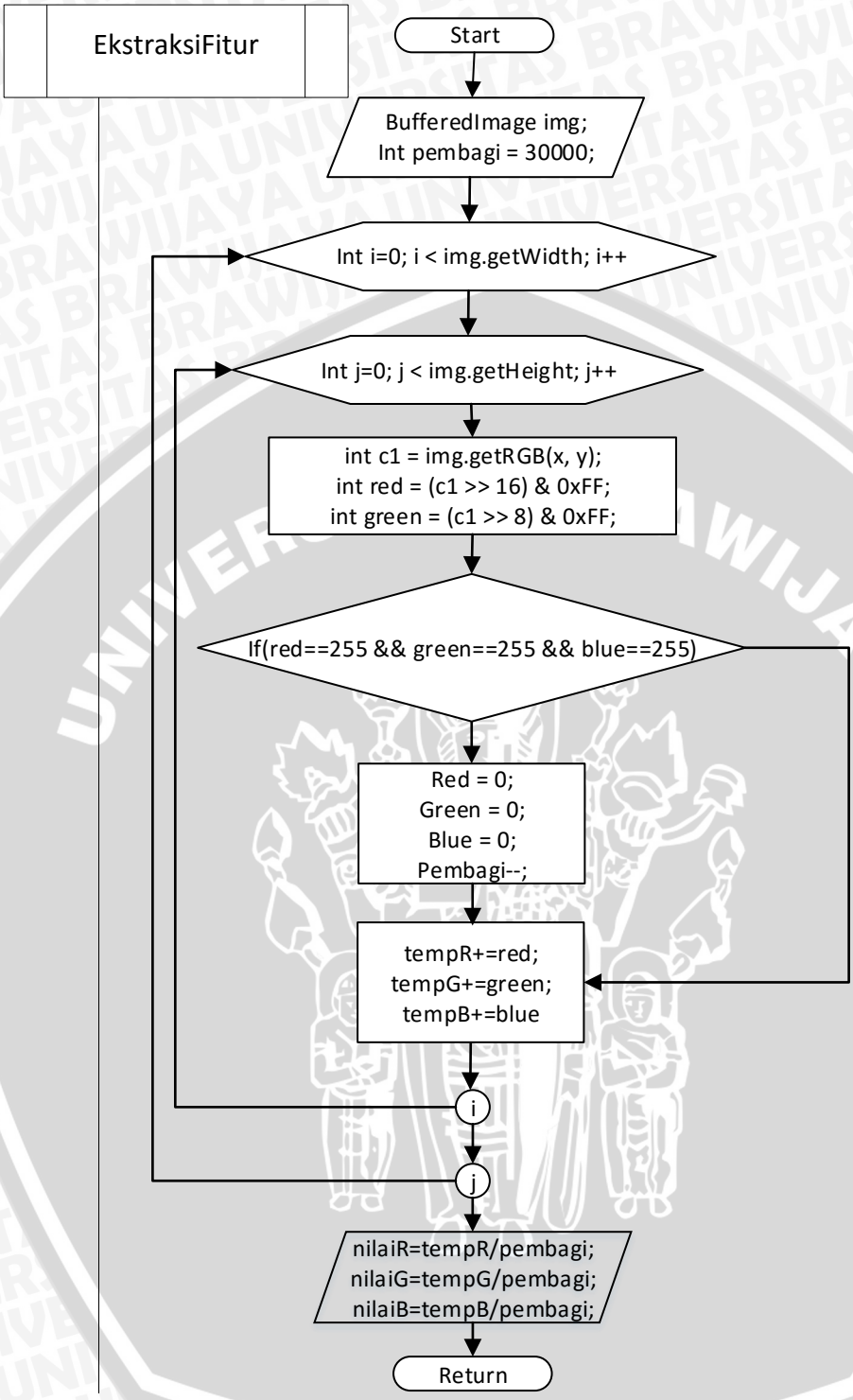
Gambar 4.17 Diagram Alir *Konvolusi*

Pertama dilakukan inisiasi terhadap objek citra1 sebagai citra citra *segmentasi*, objek citra2 sebagai citra hasil *rescaling* dan juga objek citra3

sebagai citra hasil *konvolusi*. Setelah diinisiasi, proses selanjutnya adalah menjumlahkan nilai *red, green, blue* pada setiap piksel yang ada pada citra1 dan citra2. Pada saat penjumlahan terdapat *checking* terhadap nilai penjumlahan apa bila nilai kurang dari 255 maka nilai RGB baru adalah nilai penjumlahan, nilai RGB baru menjadi 255 apabila hasil dari penjumlahan citra1 dengan citra2 lebih dari atau samadengan 255. Proses akhir dari *konvolusi* adalah hasil RGB baru yang disimpan pada citra3.

4.1.2 Ekstraksi fitur

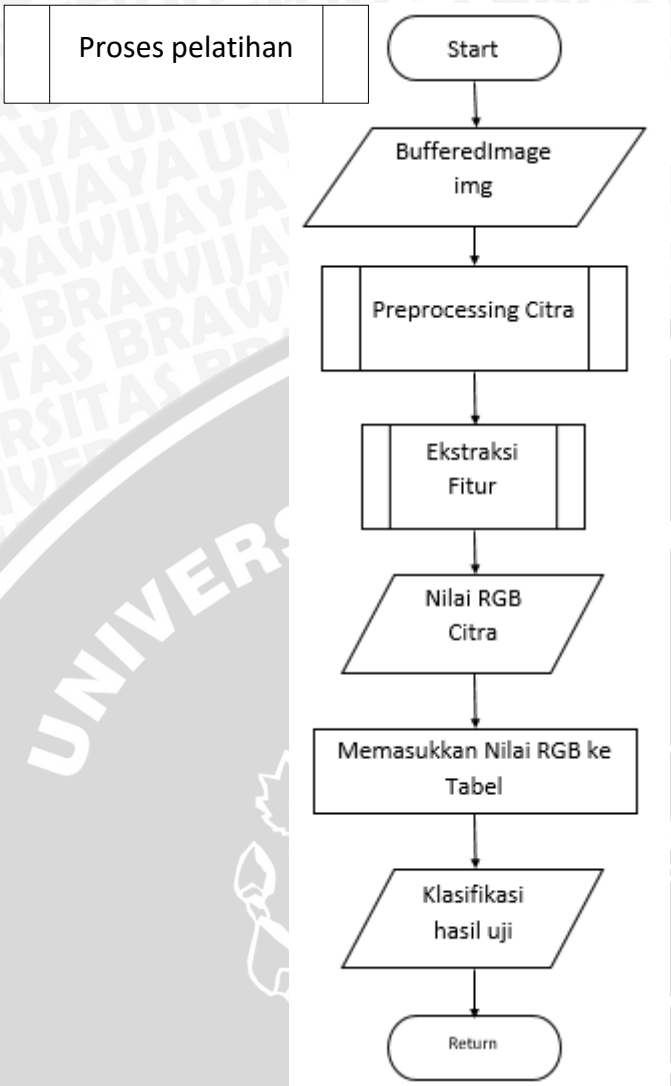
Pada *method* ini parameter yang dibutuhkan adalah citra yang sebelumnya sudah tertreshold dan juga sudah diketahui letak penyakit pada daun jeruk. Langkah pertama pada proses ini adalah inisialisasi objek citra dari kelas dan juga variabel bertipe integer dengan nama pembagi yang diberikan nilai sebesar 30000. Nilai 30000 merupakan hasil perkalian dari besar panjang citra dan juga lebar citra (*citra.getWidth()* dan *citra.getHeight()*). Setelah itu selanjutnya dilakukan *looping* bersarang dengan variabel i yang merepresentasikan *looping* terhadap *citra.getWidth()* dan juga variabel j yang merepresentasikan *looping* terhadap *citra.getHeight()*. Selanjutnya dilakukan pengambilan nilai pixel yang selanjutnya diambil nilai *red, green, blue* yang disimpan pada variabel *red*, *green*, dan *blue* pada setiap pixelnya. Setelah itu diberikan nilai 0 pada variabel *red, green* dan *blue* sebesar 0 bila nilai variabel *red, green, blue* sebesar 255 yang artinya memiliki latar belakang putih. Selain pemberian nilai 0, dilakukan juga *decrement* terhadap variabel pembagi. Hal ini dilakukan agar saat melakukan pencarian rata-rata terhadap nilai *red, green, blue* tidak rancu dan sesuai. Proses selanjutnya adalah menjumlahkan nilai red dengan variabel tempR yang disimpan pada variabel tempR, nilai green dengan variabel tempG yang disimpan pada variabel tempG, dan nilai blue dengan variabel tempB yang disimpan pada variabel tempB. Hal ini didapatkan jumlah pada setiap nilai *red, green, blue* pada setiap *pixel* tanpa latar belakang atau *background* pada citra. Proses terakhir adalah mencari rata-rata setiap *red, green, blue* yaitu dengan membagi nilai tempR, tempG, tempB dengan nilai pembagi yang disimpan pada variabel nilaiR, nilaiG, nilaiB. Diagram alir proses ekstraksi fitur ditunjukkan pada Gambar 4.18.



Gambar 4.18 Diagram Alir *Method featureExtraction*

4.1.3 Proses Pelatihan

Proses pelatihan mrupakan proses untuk memasukkan nilai *red, green, blue* (RGB) yang sudah di cari nilai rata-ratanya melalui proses ekstraksi fitur. Proses pelatihan ini merupakan tahap awal untuk memasukkan nilai RGB kedalam KNN yang nantinya nilai RGB diuji dengan data uji citra. Diagram alir proses pelatihan ditunjukkan pada Gambar 4.19.

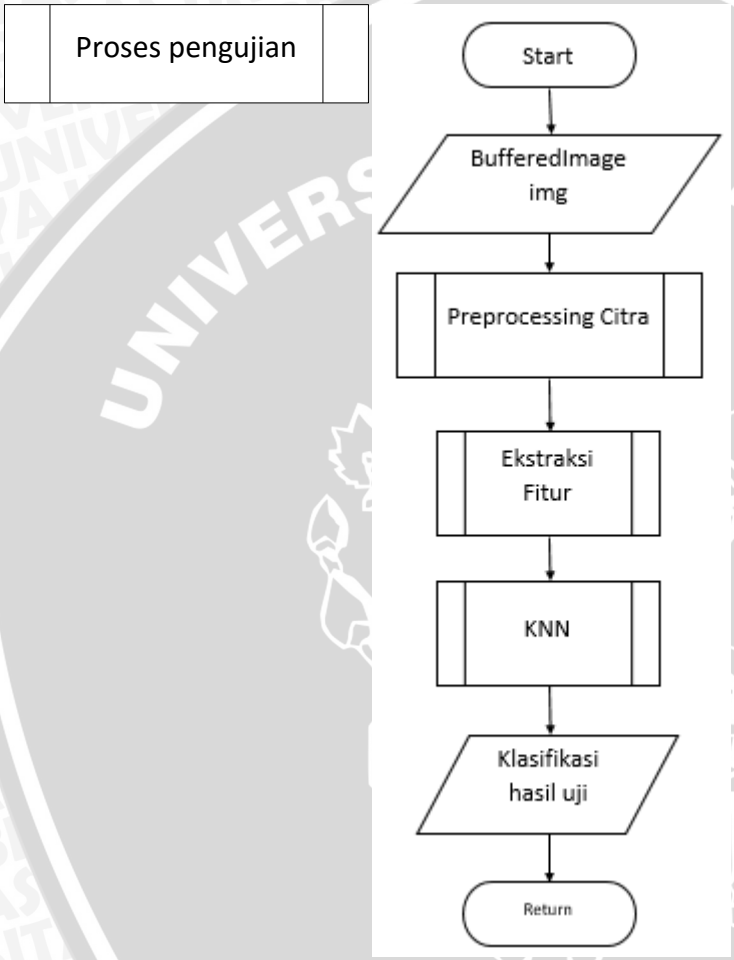


Gambar 4.19 Diagram Alir Proses Pelatihan

Proses pelatihan merupakan proses yang harus dilakukan sebelum melakukan proses pengujian citra daun untuk di uji jenis penyakit yang diidentifikasi. Proses pelatihan merupakan proses untuk memasukkan nilai RGB citra yang sudah di ekstraksi fiturnya berupa nilai rata-rata RGB untuk dimasukkan kedalam tabel yang berisikan nilai rata-rata *red, green, blue* dan kelas. Proses pertama dalam proses pelatihan adalah inisiasi objek img dari kelas BufferedImage, hal ini dilakukan untuk memudahkan proses objek citra yang memiliki nilai *red, green, blue* hasil dari ekstraksi fitur yang dimasukkan kedalam tabel. Proses selanjutnya image di *preprocessing* (*Rescaling, Grayscaling, Multilevel Otsu*) hal ini dilakukan untuk mendapatkan bagian citra daun yang berpenyakit saja. Setelah mendapatkan citra yang berpenyakit saja proses selanjutnya adalah melakukan ekstraksi fitur, yaitu mencari nilai rata-rata *red, green, blue*. Setelah didapatkan nilai rata-rata proses selanjutnya adalah mendapatkan nilai RGB yang dimasukkan kedalam tabel yang komponennya berisikan nilai rata-rata *red, green, blue*, dan kelas daun yang di latih.

4.1.4 Proses Pengujian

Proses pengujian merupakan proses untuk mengklasifikasi data uji terhadap data latih untuk dicari hasil pengujian yang didapat. Proses pengujian dilakukan apabila proses pelatihan sudah dilakukan terlebih dahulu. Proses pelatihan dilakukan untuk memberikan pelatihan data kepada sistem. Setelah aplikasi dilatih maka proses selanjutnya dilakukan pengujian untuk memvalidasi hasil uji terhadap data latih, apakah sudah sesuai dengan kelas/penyakit yang dimaksudkan. Diagram alir proses pengujian dapat dilihat pada Gambar 4.20.

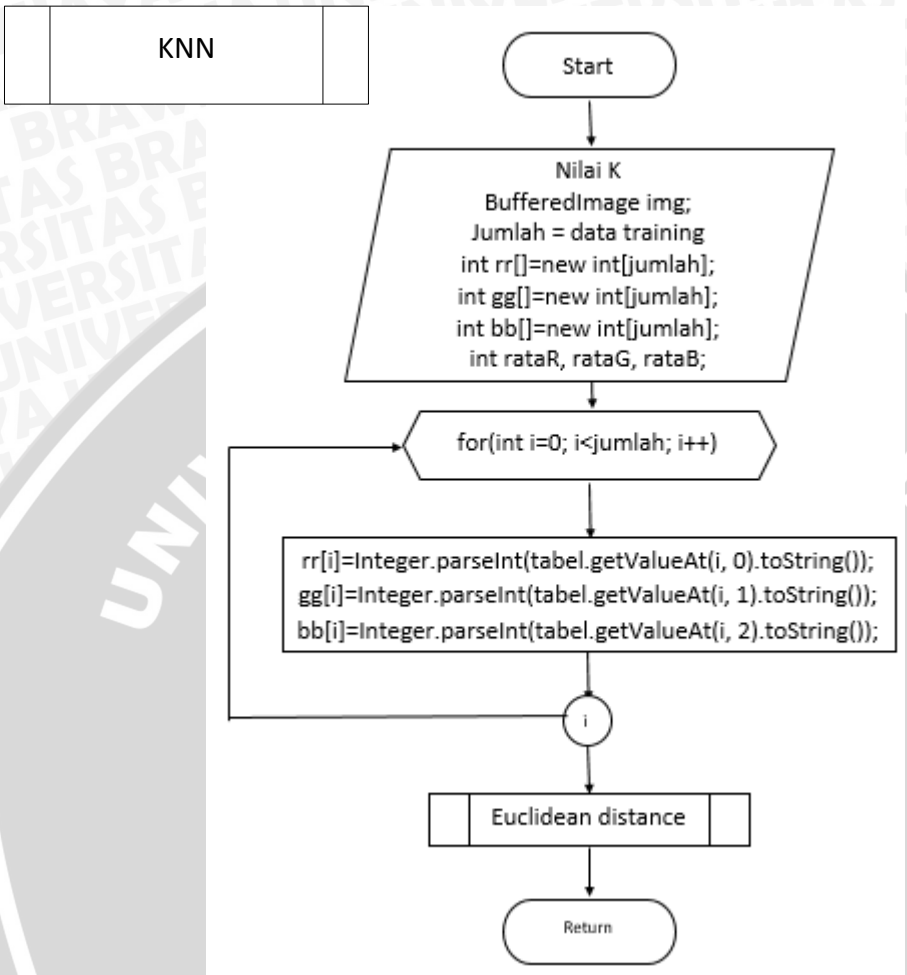


Gambar 4.20 Diagram Alir Proses Pengujian

Proses pengujian yang dilakukan pertama-tama adalah menginisiasi objek img dari *BufferedImage.* Setelah diinisiasi, dilakukan *preprocessing* (*Rescaling, Grayscaling, Multilevel Otsu*) hal ini dilakukan untuk mendapat bagian citra yang berpenyakit saja. Setelah itu dilakukan proses ekstraksi fitur untuk mendapatkan nilai rata-rata RGB setiap gambar yang pada proses selanjutnya dimasukkan kedalam KNN. Selanjutnya tahap akhir dari proses pengujian adalah mencetak hasil uji dari data uji yang dicari jarak terpendek dari data latih. Berikut dijelaskan diagram alir untuk proses *K-Nearest Neighbor* dan juga pencarian jarak terpendek dengan menggunakan *euclidean distance*.

4.1.4.1 KNN

Perhitungan KNN dilakukan untuk melatih data latih yang nantinya di klasifikasi terhadap data uji. Diagram alir KNN ditunjukkan pada Gambar 4.21.



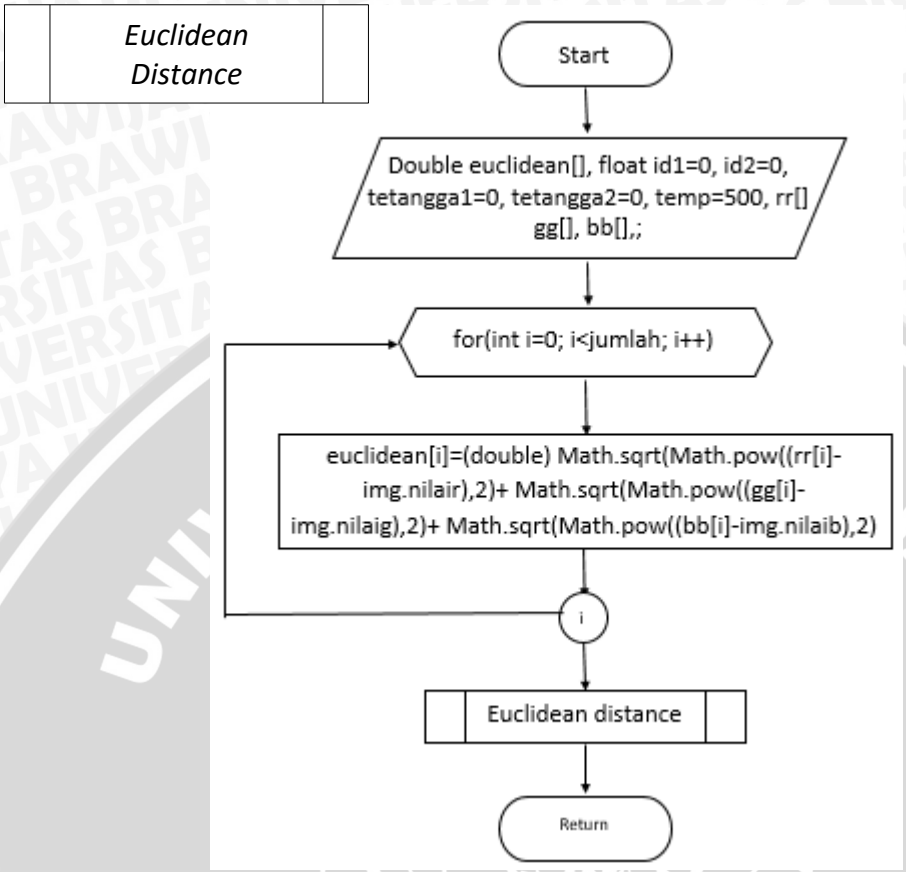
Gambar 4.21 Diagram Alir KNN

Proses KNN yang dilakukan pertama adalah menentukan nilai k pada KNN, hal ini digunakan sebagai penentuan tetangga terdakat dari data uji terhadap data latih. Setelah didapatkan nilai k proses selanjutnya adalah menginisialisasi objek citra. Objek citra nantinya digunakan sebagai citra uji yang dicari rata-rata nilai *red, green, blue*. Variabel *array* rr, gg, bb digunakan sebagai variabel yang menyimpan nilai rata-rata *red, green, blue* pada tabel pelatihan. Setelah itu variabel *array* rr, gg, dan bb diberi nilai berdasarkan pada tabel pelatihan yang sebelumnya dilakukan *looping* agar semua data latih tersimpan pada variabel rr, gg dan bb. Setelah didapatkan nilai rataR, rataG, rataB serta variabel *array* rr, gg, bb maka proses selanjutnya adalah pengujian data uji dengan melakukan pencarian tetangga terdekat dengan menggunakan *Euclidean distance*.

4.1.4.2 *Euclidean Distance*

Nilai yang sudah diproses pada KNN nantinya dicari tetangga terdekatnya dengan menggunakan *eclidean distance*. Pencarian tetangga terdekat dilakukan

sesuai dengan persamaan 2.9. Diagram alir proses *euclidean distance* ditunjukkan pada Gambar 4.22.



Gambar 4.22 Diagram Alir *Euclidean Distance*

Proses *euclidean distance* ditentukan nilai k=2, langkah pertama adalah menginisiasi jumlah *array* euclidean sebesar variabel jumlah. Setelah itu menginisiasi variabel id1, id2, tetangga1, tetangga2 dengan nilai 0 dan juga nilai temp dengan nilai 500. Setelah itu proses *looping* terhadap variabel jumlah yang nantinya diproses variabel *array* euclidean dengan persamaan 2.9 yaitu akar dari pangkat pengurangan antara variabel *array* rr dengan img.ratar, gg dengan img.ratag, dan bb dengan ratab. Setelah itu dicetak hasil klasifikasi data uji terhadap data latih.

4.2 Perhitungan Manual

Perhitungan manual memiliki fungsi sebagai gambaran perancangan perhitungan yang diterapkan pada sistem. Tujuannya adalah agar dapat diketahui benar atau tidaknya perhitungan yang nantinya dilakukan oleh sistem. Perhitungan manual yang dilakukan pada penelitian ini meliputi : Perhitungan *Preprocessing* (*Rescaling, Grayscaling, Multilevel Otsu)*, Perhitungan ekstraksi fitur, Perhitungan KNN dan perhitungan akurasi.

4.2.1 Perhitungan *Preprocessing* Citra

Proses perhitungan manual pada *Preprocessing* Citra daun dilakukan sebanyak 3 kali yaitu perhitungan *Rescaling, Grayscaling,* dan *Multilevel Otsu.* Proses *Reyscaling* yang didapatkan pada citra adalah menjadikan citra daun menjadi lebih terang atau cerah dengan mengalikan sebuah nilai konstanta pada nilai *red, green, blue* pada citra. Proses *Grayscaling* adalah upaya untuk mencari nilai rata-rata pada nilai *red, gree, blue* pada citra, yang nantinya disebut sebagai nilai keabuan citra daun. Proses terakhir dari *Preprocessing* Citra adalah penggunaan algoritma *Multilevel Thresholding* menggunakan *Otsu* untuk mencari lebih dari 1 nilai *thresholding* yang nantinya di segmentasi dengan algoritma *Otsu.*

4.2.1.1 Perhitungan *Rescaling*

Dalam perhitungan *Preprocessing* citra bagian pertama yaitu *Rescaling*, dimisalkan terdapat sebuah citra berukuran 8x8 piksel yang menjadi latar belakang sebuah citra dengan warna keabu-abuan dan didalamnya terdapat sebuah citra dengan ukuran 4x4 piksel ditengah citra dengan nilai *red, green, blue* (RGB) seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 NIlai Piksel RGB

R	G	B	R	G	B	R	G	B	R	G	B
240	240	240	240	240	240	240	240	240	240	240	240
240	240	240	240	240	240	240	240	240	240	240	240
67	93	70	80	90	87	77	87	98	91	111	80
80	120	71	90	82	23	89	89	99	89	83	84
90	80	80	45	71	78	88	87	87	100	89	90
120	90	90	112	77	85	81	82	85	87	121	99
240	240	240	240	240	240	240	240	240	240	240	240
240	240	240	240	240	240	240	240	240	240	240	240

Proses pertama didapatkan nilai *red, green, bue* (RGB) pada setiap citra daun. Setelah didapatkan maka proses selanjutnya adalah dikalikannya nilai *red, green, blue* (RGB) oleh nilai konstanta sesuai dengan persamaan (2.1). Apabila nilai RGB yang didapatkan hasil dari perkalian sebuah konstanta lebih dari 255 maka nilai RGB diset menjadi 255. Jika nilai perkalian yang didapatkan tidak melebihi dari 255 maka nilai RGB yang dituliskan sesuai dengan hasil perkalian pada persamaan (2.1). Perhitungan nilai RGB utuk piksel pertama pada Tabel 4.1 adalah sebagai berikut :

Red = red * 1.4 = 240 * 1.4 + 0 = 336 (lebih dari 255)

Green = green * 1.4 = 240 * 1.4 + 0 = 336 (lebih dari 255)

Blue = blue * 1.4 = 240 * 1.4 + 0 = 336 (lebih dari 255)

Jadi nilai RGB untuk piksel pertama pada Tabel 4.2 adalah red = 255, green = 255, blue = 255. Nilai tersebut didapatkan berdasarkan kondisi jika nilai *red, green, blue* (RGB) yang dikalikan nilai konstanta 1.4 pada piksel pertama lebih dari 255. Berikut nilai *red, green, blue* (RGB) secara keseluruhan disemua piksel yang didapatkan setelah perkalian oleh persamaan (2.1) dengan permisalan nilai konstanta *scale* factor sebesar 1.4 dengan nilai *offset* sebesar 0 pada Tabel 4.2

Tabel 4.2 Hasil Perhitungan *Rescaling*

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
94	130	98	112	126	122	108	122	137	127	155	112
112	168	99	126	115	32	125	125	139	125	116	118
126	112	112	63	99	109	123	122	122	140	125	126
168	126	126	157	108	119	113	115	119	122	169	139
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Proses pada Tabel 4.2 didapatkan citra baru dengan tingkat kecerahan pada citra yang bertambah. Proses ini juga dapat disebut sebagai *Enhanching Image Brightness* dimana menambahkan nilai konstanta untuk meningkatkan kecerahan pada citra. Hal ini dilakukan untuk memisahkan latar belakang citra dengan objek daun.

4.2.1.2 Perhitungan *Grayscaling*

Dalam perhitungan *preprocessing* citra bagian kedua yaitu *Grayscaling*, proses ini adalah merubah citra yang sudah di *rescaling* menjadi warna keabu-abuan. Dimisalkan terdapat citra berukuran 8x8 piksel yang menjadi latar belakang sebuah citra dengan warna putih dan didalamnya terdapat sebuah citra dengan ukuran 4x4 piksel ditengah citra yang menjadi citra daun denga nilai *red, green, blue* (RGB) seperti yang ditunjukkan pada Tabel 4.3.

Tabel 4.3 NIlai Piksel RGB

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
94	130	98	112	126	122	108	122	137	127	155	112
112	168	99	126	115	32	125	125	139	125	116	118
126	112	112	63	99	109	123	122	122	140	125	126

R	G	B	R	G	B	R	G	B	R	G	B
168	126	126	157	108	119	113	115	119	122	169	139
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Proses pertama didapatkan nilai *red, green, blue* (RGB) pada setiap citra daun. Setelah didapatkan maka proses selanjutnya adalah mencari nilai rata-rata nilai *red, green, blue* pada setiap pikselnya sesuai dengan persamaan (2.2). Setelah didapatkan nilai rata-rata proses selanjutnya adalah menggantikan nilai RGB citra sebelumnya dengan nilai rata-rata per pikselnya. Perhitungan nilai rata-rata untuk piksel pertama pada tabel 4.3 adalah sebagai berikut :

Rata-rata = (red + green + blue) / 3 = (255 + 255 + 255) / 3 = 765 / 3 = 255

Jadi nilai RGB untuk piksel pertama pada Tabel 4.2 adalah red = 255, green = 255, blue = 255. Nilai tersebut didapatkan berdasarkan nilai rata-rata yang didapatkan pada piksel pertama. Berikut nilai *red, green, blue* (RGB) secara keseluruhan disemua piksel yang didapatkan setelah pencarian rata-rata pada persamaan 2.2 yang dituliskan pada Tabel 4.4.

Tabel 4.4 Hasil Perhitungan *Grayscaling* pencarian rata-rata

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
107	107	107	120	120	120	122	122	122	132	132	132
126	126	126	91	91	91	129	129	129	119	119	119
117	117	117	91	91	91	122	122	122	130	130	130
140	140	140	128	128	128	116	116	116	143	143	143
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

4.2.1.3 Perhitungan *Multilevel Otsu*

Perhitungan *Multilevel Otsu* dimulai dari pencarian nilai histogram, pencarian nilai nilai PSH, pencarian nilai *threshold*, penentuan nilai batas terhadap nilai *threshold* dan RGB, dan terakhis adalah proses *konvolusi* atau penambahan nilai RGB. Dalam perhitungan ini meneruskan dari proses perhitungan *rescaling* dan *grayscaling*. *Output* dari perhitungan *Multilevel Thresholding* menggunakan *Otsu* adalah citra daun yang berpenyakit terpisahkan dari bagian daun yang sehat. Bagian daun yang berpenyakit dilakukan ekstraksi ciri agar dapat dimasukkan

kedalam KNN untuk dilakukan pengujian terhadap data uji. sedangkan untuk daun yang tidak berpenyakit tidak akan dimasukkan kedalam perhitungan.

Langkah pertama dari proses *Multilevel Thresholding* menggunakan *Otsu* adalah pencarian nilai *histogram* (*ni*), pencarian nilai histogram dilakukan untuk mencari nilai rata-rata setiap histogram pada setiap piksel dan juga nilai RGB 0-255. Pencarian nilai histogram dilakukan bila nilai warna citra pada sebuah piksel kurang dari atau sama dengan 255, maka dilakukan penambahan nilai pada nilai *gray* tersebut. Proses pencarian nilai histogram dilakukan sesuai dengan diagram alir pada Gambar 4.7 Proses perhitungan pencarian nilai histogram ditunjukkan pada Tabel 4.5.

Setelah mencari nilai histogram, langkah kedua adalah mencari nilai rata-rata yang berasal dari nilai probabilitas tingkat keabuan sebuah citra RGB. Dengan menggunakan persamaan (2.5) dimana nilai *Pi* dicari terlebih dahulu dengan mencari nilai probabilitas yang sesuai dengan persamaan (2.4). Berikut adalah contoh perhitungan rata-rata probabilitas berdasarkan perhitungan Tabel 4.5.

$$Pi = \frac{ni}{N}$$

$$\mu_T = \sum_{i=1}^{L} i * Pi$$

$$= \sum_{i=1}^{L} 0 * \left(\frac{0}{64}\right) = 0$$

Pada saat nilai warna ke-0 dari perhitungan diatas didapatkan nilai rata-rata probabilitas. Perhitungan pada warna ke-1 dan seterusnya dilakukan sesuai dengan rumus diatas. Untuk keseluruhan perhitungan nilai total rata-rata probabilitas berdasarkan perhitungan persamaan (2.7) yang ditunjukkan pada Tabel 4.6.

Tabel 4.5 Hasil Perhitungan Histogram

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2
i	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
i	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	
ni	0	1	1	0	1	1	0	2	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	11
i	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	
ni	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
i	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	
ni	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	253	254	255																					
ni	0	0	48																					48
																				Total Piksel Gambar				64

Tabel 4.6 Hasil Perhitungan Total Rata-rata Histogram

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0	0	0	0	0
i	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
μ_T	0	0	0	0	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	0	0.01	0.01	0	0.01	0.01
i	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
μ_T	0	0.03	0	0	0	0.01	0	0.01	0.01	0.01	0	0.01	0	0	0	0	0	0	0	0.01	0	0	0.01	0
i	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255									
μ_T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.75									

Proses perhitungan *multilevel thresholding* menggunakan *otsu* selanjutnya adalah mencari nilai *P* dengan menggunakan persamaan (2.10) dan persamaan (2.11). Pada perhitungan ini dilakukan pemberian awal nilai *P* dan *S* pada *array* diagonal dengan nilai histogram dan nilai probabilitas histogram. Perhitungan manual pada pencarian awal nilai *P* dan *S* ditunjukkan pada Tabel 4.7

Tabel 4.7 Hasil Perhitungan Nilai P dan S awal

Nilai Array P			Nilai Array S		
Nilai i	Indeks	Nilai P	Nilai i	Indeks	Nilai P
1	P[1][1]	0	1	S[1][1]	0
...	...	...	...	...	...
91	P[91][91]	0.03	91	S[91][91]	2.73
107	P[107][107]	0.01	107	S[107][107]	1.07
116	P[116][116]	0.01	116	S[116][116]	1.16
117	P[117][117]	0.01	117	S[117][117]	1.17
119	P[119][119]	0.01	119	S[119][119]	1.19
120	P[120][120]	0.01	120	S[120][120]	1.2
122	P[122][122]	0.03	122	S[122][122]	3.66
126	P[126][126]	0.01	126	S[126][126]	1.26
128	P[128][128]	0.01	128	S[128][128]	1.28
129	P[129][129]	0.01	129	S[129][129]	1.29
130	P[130][130]	0.01	130	S[130][130]	1.3
132	P[132][132]	0.01	132	S[132][132]	1.32
140	P[140][140]	0.01	140	S[140][140]	1.4
143	P[143][143]	0.01	143	S[143][143]	1.43
255	P[255][255]	0.75	255	S[255][255]	191.25

Setelah didapatkan nilai diagonal *P* dan *S* proses selanjutnya adalah pemberian nilai pada baris pertama. Dengan menggunakan persamaan (2.12) dan persamaan (2.13) nilai *P* dan *S* pada baris pertama dicari. Perhitungan secara keseluruan untuk pencarian nilai *P* dan *S* ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil Perhitungan Nilai P dan S Baris Pertama

Nilai Array P			Nilai Array S		
Nilai i	Indeks	Nilai P	Nilai i	Indeks	Nilai S
1	P[1][2]	0	1	S[1][2]	0
...	...	...	...	...	...
90	P[1][91]	0.03	90	S[1][91]	2.73
106	P[1][107]	0.04	106	S[1][107]	4.28
115	P[1][116]	0.05	115	S[1][116]	5.8
116	P[1][117]	0.06	116	S[1][117]	7.02
118	P[1][119]	0.07	118	S[1][119]	8.33
119	P[1][120]	0.08	119	S[1][120]	9.6
121	P[1][122]	0.11	121	S[1][122]	13.42
125	P[1][126]	0.12	125	S[1][126]	15.12
127	P[1][128]	0.13	127	S[1][128]	16.64
128	P[1][129]	0.14	128	S[1][129]	18.06
129	P[1][130]	0.15	129	S[1][130]	19.5
131	P[1][132]	0.16	131	P[1][132]	21.28
139	P[1][140]	0.17	139	S[1][140]	23.8
142	P[1][143]	0.18	142	S[1][143]	25.74
254	P[1][255]	0.93	254	S[1][255]	237.15

Setelah didapatkan nilai P dan S pada baris pertama, langkah selanjutnya adalah mencari indeks selain pada proses diatas. Pada proses ini digunakan persamaan (2.14) dan persamaan (2.15) untuk mencari nilai *P* dan *S* indeks yang lain. Perhitungan secara manual pada proses pencarian nilai *P* dan *S* indeks lain ditunjukkan pada Tabel 4.9.

Tabel 4.9 Hasil Perhitungan Nilai *P* dan *S*

P						S					
i = 2		...		l = 254		i = 2		...		i = 254	
Nilai J	P(i,j)	Nilai J	P(i,j)	Nilai J	P(i,j)	Nilai J	S(i,j)	Nilai J	S(i,j)	Nilai J	S(i,j)
3	0	...	...	255	0	3	0	...	...	255	0
91	0.03	...	...			91	2.73	...	...		
107	0.04	...	...			107	4.28	...	...		

P						S					
i = 2		...		I = 254		i = 2		...		i = 254	
Nilai J	P(i,j)	Nilai J	P(i,j)	Nilai J	P(i,j)	Nilai J	S(i,j)	Nilai J	S(i,j)	Nilai J	S(i,j)
116	0.05	...	...			116	5.8	...	...		
117	0.06	...	...			117	7.02	...	...		
119	0.07	...	...			119	8.33	...	...		
120	0.08	...	...			120	9.6	...	...		
122	0.11	...	...			122	13.42	...	...		
126	0.12	...	...			126	15.12	...	...		
128	0.13	...	...			128	16.64	...	...		
129	0.14	...	...			129	18.06	...	...		
130	0.15	...	...			130	19.5	...	...		
132	0.16	...	...			132	21.28	...	...		
140	0.17	...	...			140	23.8	...	...		
143	0.18	...	...			143	25.74	...	...		
255	0.93	...	...			255	237.15	...	...		

Setelah didapatkan nilai *P* dan *S* secara keseluruhan, proses selanjutnya adalah mencari nilai *H*. Pencarian nilai *H* dilakukan sesuai dengan persamaan (2.16). Berikut adalah tabel perhitungan untuk mencari nilai *H*.

Tabel 4.10 Hasil Perhitungan Nilai H

H					
i = 1		...		I = 254	
Nilai J	H(i,j)	Nilai J	H(i,j)	Nilai J	H(i,j)
2	0	...	...	255	0
91	248.43	...	...		
107	1831.84	...	...		
116	3364	...	...		
117	4928.04	...	...		
119	6938.89	...	...		
120	9216	...	...		
122	6003.213	...	...		

H					
i = 1		...		I = 254	
Nilai J	H(i,j)	Nilai J	H(i,j)	Nilai J	H(i,j)
126	22861.44	...	...		
128	27688.96	...	...		
129	32616.36	...	...		
130	38025	...	...		
132	45283.84	...	...		
140	56644	...	...		
143	66254.76	...	...		
255	74986.83	...	...		

Setelah didapatkan nilai *H*, proses selanjutnya adalah melakukan pencarian nilai maksimum pada nilai *H*. Sebelum mencari nilai maksimum dilakukan proses perhitungan sesuai dengan persamaan (2.17). Setelah didapatkan nilai dalam persamaan (2.17) proses selanjutnya adalah mencari maksimum pada nilai *H* untuk mendapatkan nilai *T2*, sedangkan nilai *T1* diberi nilai default yaitu 0. Proses pencarian nilai *T2* ditunjukkan pada Tabel 4.11.

Tabel 4.11 Hasil Perhitungan Penentuan Nilai *Threshold*

i	1	2	3	...	91	143
T2	60473.25	60473.25	60473.25	...	61306.8	125846.8

Pada tabel 4.11 didapatkan nilai terbesar terletak pada nilai 143, nilai terbesar pada proses ini akan menjadi nilai *threshold* pertama. Pada proses selanjutnya adalah melakukan proses seleksi sesuai dengan nilai *threshold* hasil dari proses ini.

Jika sudah didapatkan nilai *T2* pada citra, maka proses selanjutnya adalah memberi batas terhadap nilai *red, green, blue* terhadap nilai *T2*. Proses ini dilakukan untuk menseleksi hasil dari citra yang tersegmentasi dengan citra yang tidak tersegmentasi. Proses seleksi dilakukan dengan seleksi kondisi, bila RGB citra terletak diantara nilai *T1* dan *T2* maka nilai RGB baru disesuaikan dengan nilai *grayscale* jika tidak maka nilai RGB di beri nilai sebesar 255 (putih). Berikut adalah tabel RGB yang didapat dari *grayscale* pada Tabel 4.4 yang sudah dilakukan proses pemberian batas terhadap nilai *T1* dan *T2*.

Tabel 4.12 Proses Perhitungan Pembatasan Nilai RGB Terhadap Nilai *T1* dan *T2*

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Proses akhir dari perhitungan *Multilevel Thresholding* menggunakan *Otsu* adalah melakukan pewarnaan terhadap citra dengan menggunakan proses konvolusi citra yang telah disegmentasi dengan citra yang telah dilakukan *rescaling*. Hal ini dilakukan untuk mendapatkan citra RGB yang telah disegmentasi. Proses perhitungan ini disebut sebagai dengan proses *konvolusi* yakni menambahkan nilai RGB dari citra yang *thresholding* dengan citra asli. Hasil RGB citra hasil dari *thresholding* dapat dilihat pada Tabel 4.12 Sedangkan untuk RGB dari citra asli hasil *rescaling* dapat dilihat pada Tabel 4.2. Nilai RGB baru didapat apabila penambahan citra RGB antara citra tersegmentasi dengan citra *rescaling* tidak lebih dari 255. Nilai RGB baru menjadi 255 bila hasil dari penambahan lebih dari 255. Berikut adalah proses *konvolusi* pada piksel pertama.

$red = redCitraSegmentasi + redCitraAsli$ = 255 + 255 = 255 (nilai lebih dari 255)

$green = greenCitraSegmentasi + greenCitraAsli$ = 255 + 255 = 255 (nilai lebih dari 255)

$blue = blueCitraSegmentasi + blueCitraAsli$ = 255 + 255 = 255 (nilai lebih dari 255)

Dari hasil *konvolusi* pada piksel pertama didapatkan nilai red = 255, green = 255, blue = 255. Berikut adalah nilai *red, green, blue* secara keseluruhan terdapat pada Tabel 4.13.

Tabel 4.13 Proses Perhitungan *Konvolusi*

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
94	130	98	112	126	122	108	122	137	127	155	112

R	G	B	R	G	B	R	G	B	R	G	B
112	168	99	126	115	32	125	125	139	125	116	118
126	112	112	63	99	109	123	122	122	140	125	126
168	126	126	157	108	119	113	115	119	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

4.2.2 Perhitungan Ekstraksi fitur

Perhitungan Ekstraksi fitur dilakukan bila proses *preprocessing* telah dilakukan. Perhitungan ekstraksi fitur dilakukan untuk mendapatkan rata-rata nilai RGB secara terpisah. Langkah pertama untuk melakukan ekstraksi fitur adalah menyeleksi RGB daun yang memiliki nilai lebih dari atau sama dengan 255 agar tidak dimasukkan kedalam perhitungan, hal ini dilakukan untuk membedakan latar belakang dengan citra daun yang ada ditengah citra dan juga agar nilai RGB latar belakang tidak disertakan dalam perhitungan rata-rata atau ekstraksi fitur. Dimisalkan teradapat citra yang memiliki ukuran 8x8 piksel yang didalamnya terdapat nilai RGB daun yang berukuran 4x4 piksel. Tabel 4.14 menunjukkan hasil perubahan nilai RGB.

Tabel 4.14 Hasil Perubahan Nilai RGB

R	G	B	R	G	B	R	G	B	R	G	B
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
94	130	98	112	126	122	108	122	137	127	155	112
112	168	99	126	115	32	125	125	139	125	116	118
126	112	112	63	99	109	123	122	122	140	125	126
168	126	126	157	108	119	113	115	119	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Setelah dilakukan perubahan nilai RGB yang lebih dari atau sama dengan 255 dengan nilai 0, maka proses selanjutnya adalah dilakukan proses perhitungan rata-rata secara terpisah. Nilai RGB yang diambil adalah nilai RGB secara terpisah. Misal nilai yang diambil adalah nilai yang berada pada Tabel 4.5. Proses pencarian rata-rata pada citra dilakukan selain RGB yang bernilai 0. Maka pada Tabel 4.5 yang dilakukan proses ekstraksi fitur adalah tabel yang bewarna kuning saja. Besar nilai piksel pada Tabel 4.5 sebesar 64 piksel, nilai piksel tersebut berkurang bila nilai

RGB sama dengan 255 dan menjadi nilai pembagi saat menghitung rata-rata RGB. Dari proses tersebut didapatkan nilai pembagi pada perhitungan rata-rata adalah 15, dikarenakan nilai piksel yang ada pada tabel yang bewarna kuning sebesar 15 piksel yang menunjukkan bagian dari penyakit. Perhitungan rata-rata ekstraksi fitur dapat ditunjukkan sebagai berikut.

$$rata - rata\ R = \frac{(94 + 122 + 126 + 168 + 112 + 126 + 63 + 157 + 108 + 125 + 123 + 113 + 127 + 125)}{15}$$

$$= \frac{1469}{15} = 97.93$$

$$rata - rata\ G = \frac{(130 + 168 + 112 + 126 + 126 + 115 + 99 + 108 + 122 + 125 + 127 + 115 + 155 + 116 + 125)}{15}$$

$$= \frac{1869}{15} = 124.6$$

$$rata - rata\ B = \frac{(98 + 99 + 112 + 126 + 122 + 32 + 109 + 119 + 137 + 139 + 122 + 119 + 112 + 118 + 126)}{15}$$

$$= \frac{1690}{15} = 112.67$$

Dari perhitungan diatas, didapatkan nilai rata-rata R = 97.93 , rata-rata G = 124.6, rata-rata B = 112.67.

4.2.3 Perhitungan KNN

Perhitungan KNN dilakukan bila didapatkan nilai rata-rata RGB dari citra yang telah dilakukan pada proses ekstraksi fitur. Nilai rata-rata tersebut didapat pada data latih dan juga data uji. Perhitungan KNN yang dilakukan adalah dengan nilai *K*=2. Dimisalkan terdapat 10 buah data yang sudah didapatkan nilai ekstraksi fiturnya secara terpisah yang ditunjukkan pada tabel 4.15. Sedangkan untuk data uji yang digunakan memiliki nilai *R* = 170, *G* = 180, *B* = 200.

Tabel 4.15 Nilai Rata-rata RGB data latih

Data Latih	R	G	B	Kelas
1	120	124	128	CVPD
2	130	133	129	CVPD
3	178	185	175	Cendawan Jelaga
4	180	187	177	Cendawan Jelaga
5	243	245	242	Downy Mildew
6	221	227	218	Downy Mildew
7	180	166	150	Defisiensi

Data Latih	R	G	B	Kelas
8	166	165	178	Defisiensi
9	200	211	212	Sehat
10	220	224	230	Sehat

Setelah itu dihitung jarak *Euclidean* antara data uji dengan data latih yang sesuai dengan persamaan (2.9). Nilai data yang digunakan adalah data RGB dari data uji dan data latih. Proses perhitungan *Euclidean distance* adalah untuk mencari jarak terpendek antara datu uji dengan data latih. Berikut adalah proses perhitungan manual data latih pertama dengan *Euclidean distance.*

$$d1 = \sqrt{(170-120)^2 + (180-124)^2 + (200-128)^2}$$

$$d1 = \sqrt{(50)^2 + (56)^2 + (72)^2}$$

$$d1 = \sqrt{2500 + 3136 + 5184}$$

$$d1 = \sqrt{10820}$$

$$d1 = \sqrt{10820} = 104.0912$$

Berikut adalah perhitungan *Euclidean distance* antara data latih dengan data uji yang dituliskan pada Tabel 4.16.

Tabel 4.16 Nilai *Euclidean Distance* Antara Data Latih Dengan Data Uji

Data Latih	Hasil Perhitungan
Jarak data latih ke 1	104.0192
Jarak data latih ke 2	94.07444
Jarak data latih ke 3	26.72078
Jarak data latih ke 4	26.03843
Jarak data latih ke 5	106.3861
Jarak data latih ke 6	71.65194
Jarak data latih ke 7	52.87722
Jarak data latih ke 8	26.92582
Jarak data latih ke 9	44.77723
Jarak data latih ke 10	73.04793

Setelah dilakukan pencarian jarak antara data latih dengan data uji dengan *Euclidean distance*, proses selanjutnya adalah memilih 2 nilai jarak terpendek dari data latih. Hal itu dilakukan dikarenakan nilai K yang dimasukkan adalah 2. Tetangga atau nilai terkecil yang didapat dari tabel diatas adalah nilai 26.03843

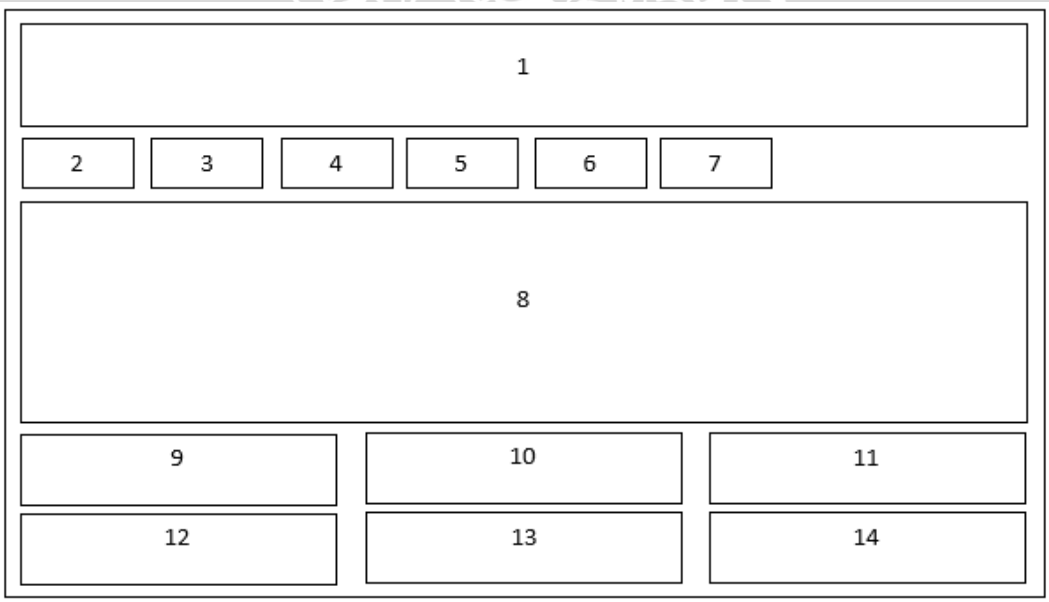
dan nilai 26.72078. Setelah itu dilakukan *voting* untuk mencari kelas yang sering muncul. Kelas yang didapatkan pada perhitungan KNN tersebut adalah Cendawan Jelaga dikarenakan kelas tersebut muncul paling banyak sesuai dengan nilai tetangga terdekat.

4.3 Perancangan Antarmuka

Perancangan antarmuka dilakukan untuk mempermudah peneliti mengimplementasikan aplikasi yang dibuat. Pada perancangan antarmuka terdapat beberapa antarmuka yang dibuat diantaranya halaman awal, halaman pengujian *Scale Factor*, halaman pengujian nilai *MLEVEL*, halaman pengujian nilai *K*, halaman pengujian semua data, halaman pengujian satu data, dan halaman *about*.

4.3.1 Halaman Awal

Antarmuka halaman awal penelitian ini berisi 6 tombol yang lainnya mengarahkan kepada halaman tertentu diantaranya adalah halaman awal, halaman pelatihan untuk mendapatkan *scale factor*, halaman pelatihan untuk mendapatkan *MLEVEL*, halaman pelatihan untuk mendapatkan nilai *K*, halaman pengujian semuda data, halaman pengujian satu data. Perancangan halaman awal ditunjukkan pada Gambar 4.17.



Gambar 4.23 Perancangan Antarmuka Halaman Awal

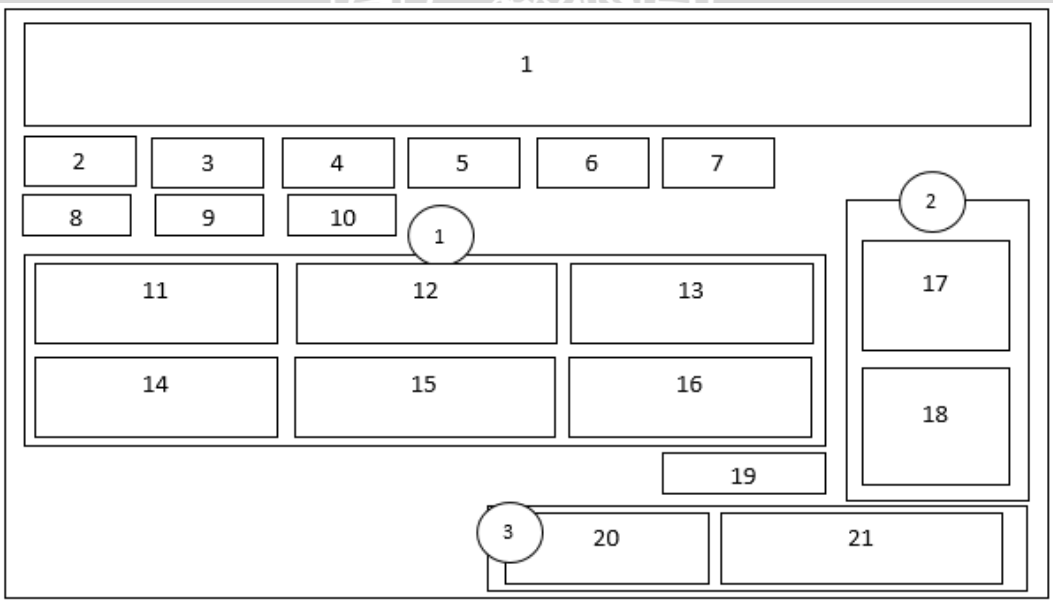
Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pengujian *Scale Factor*
4. Tombol yang mengarahkan ke halaman pengujian *MLEVEL*

5. Tombol yang mengarahkan ke halaman pengujian nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Halaman awal aplikasi yang terdiri dari informasi penyakit daun
9. Identitas Mahasiswa
10. Identitas dosen pembimbing 1
11. Identitas dosen pembimbing 2
12. Logo Universitas Brawijaya
13. Logo Fakultas Ilmu Komputer
14. Logo Balitjestro

4.3.2 Halaman Pengujian *Scale Factor*

Antarmuka halaman pelatihan untuk mendapatkan nilai *scale factor* terbaik terdiri dari 3 bagian. Bagian pertama memiliki fungsi dimana untuk menampilkan citra asli, citra hasil *rescaling*, citra *grayscaling*, citra hasil *multilevel thresholding otsu*, citra bagian penyakit dan citra tidak berpenyakit. Bagian kedua berisikan tabel, tabel pertama menampilkan RGB hasil dr pemrosesan data latih sementara tabel kedua menampilkan hasil klasifikasi sistem dan kelas asli pada citra. Bagian ketiga berisikan informasi dari citra berupa warna RGB dan juga informasi citr berupa nama file, panjang dan lebar citra. Antar muka halaman pelatihan untuk mencari nilai *scale factor* terbaik ditunjukkan pada Gambar 4.18.



Gambar 4.24 Perancangan Antarmuka Halaman Pengujian *Scale Factor*

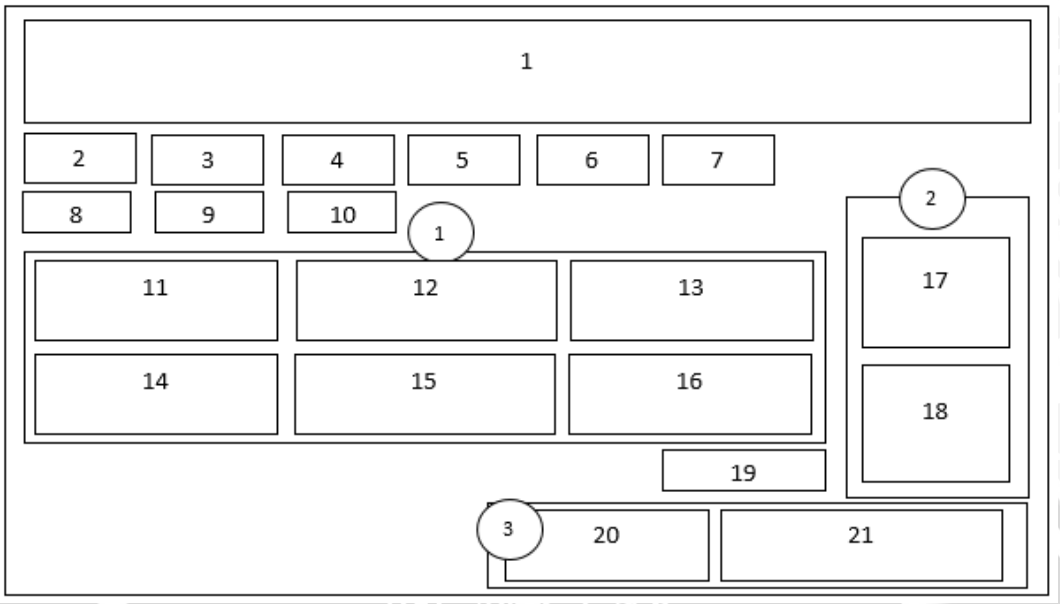
Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pengujian *Scale Factor*
4. Tombol yang mengarahkan ke halaman pengujian *MLEVEL*
5. Tombol yang mengarahkan ke halaman pengujian nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Combo box yang harus dipilih untuk ditentukan nilai konstanta *scale factor*
9. Tombol yang melakukan pemrosesan terhadap data latih
10. Tombol yang melakukan pemrosesan pelatihan
11. Panel yang menampilkan citra asli
12. Panel yang menampilkan citra hasil *rescaling*
13. Panel yang menampilkan citra asli *grayscaling*
14. Panel yang menampilkan citra asli hasil *multilevel otsu*
15. Panel yang menampilkan bagian citra yang berpenyakit
16. Panel yang menampilkan bagian citra yang tidak berpenyakit
17. Tabel yang menampilkan rata-rata *red, green, blue* pada pemrosesan data latih
18. Tabel yang menampilkan hasil klasifikasi terhadap data latih
19. Label yang menampilkan akurasi
20. Panel yang menampilkan informasi RGB suatu citra
21. Panel yang menampilkan informasi file berupa nama file, panjang dan lebar

4.3.3 Halaman Pengujian *MLEVEL*

Antarmuka halaman pelatihan untuk mendapatkan nilai *MLEVEL* terbaik terdiri dari 3 bagian. Bagian pertama memiliki fungsi dimana untuk menampilkan citra asli, citra hasil *rescaling*, citra *grayscaling*, citra hasil *multilevel thresholding otsu*, citra bagian penyakit dan citra tidak berpenyakit. Bagian kedua berisikan tabel, tabel pertama menampilkan RGB hasil dr pemrosesan data latih sementara tabel kedua menampilkan hasil klasifikasi sistem dan kelas asli pada citra. Bagian ketiga berisikan informasi dari citra berupa warna RGB dan juga informasi citr berupa

nama file, panjang dan lebar citra. Antar muka halaman pelatihan untuk mencari nilai *MLEVEL* terbaik ditunjukkan pada Gambar 4.19.



Gambar 4.25 Perancangan Antarmuka Halaman Pengujian *MLEVEL*

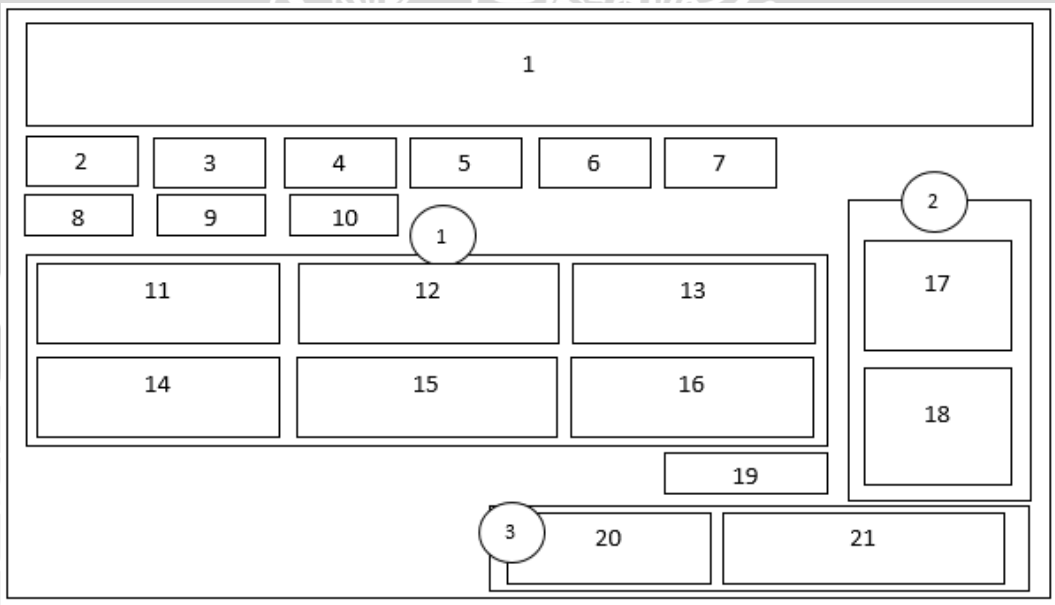
Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pengujian *Scale Factor*
4. Tombol yang mengarahkan ke halaman pengujian *MLEVEL*
5. Tombol yang mengarahkan ke halaman pengujian nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Combo box yang harus dipilih untuk ditentukan nilai level
9. Tombol yang melakukan pemrosesan terhadap data latih
10. Tombol yang melakukan pemrosesan pelatihan
11. Panel yang menampilkan citra asli
12. Panel yang menampilkan citra hasil *rescaling*
13. Panel yang menampilkan citra asli *grayscaling*
14. Panel yang menampilkan citra asli hasil *multilevel otsu*
15. Panel yang menampilkan bagian citra yang berpenyakit
16. Panel yang menampilkan bagian citra yang tidak berpenyakit

17. Tabel yang menampilkan rata-rata *red, green, blue* pada pemrosesan data latih
18. Tabel yang menampilkan hasil klasifikasi terhadap data latih
19. Label yang menampilkan akurasi
20. Panel yang menampilkan informasi RGB suatu citra
21. Panel yang menampilkan informasi file berupa nama file, panjang dan lebar

4.3.4 Halaman Pengujian Nilai *K*

Antarmuka halaman pelatihan untuk mendapatkan nilai nilai *K* terbaik terdiri dari 3 bagian. Bagian pertama memiliki fungsi dimana untuk menampilkan citra asli, citra hasil *rescaling*, citra *grayscaling*, citra hasil *multilevel thresholding otsu*, citra bagian penyakit dan citra tidak berpenyakit. Bagian kedua berisikan tabel, tabel pertama menampilkan RGB hasil dr pemrosesan data latih sementara tabel kedua menampilkan hasil klasifikasi sistem dan kelas asli pada citra. Bagian ketiga berisikan informasi dari citra berupa warna RGB dan juga informasi citr berupa nama file, panjang dan lebar citra. Antar muka halaman pelatihan untuk mencari nilai *K* terbaik ditunjukkan pada Gambar 4.20.



Gambar 4.26 Perancangan Antarmuka Halaman Pengujian Nilai *K*

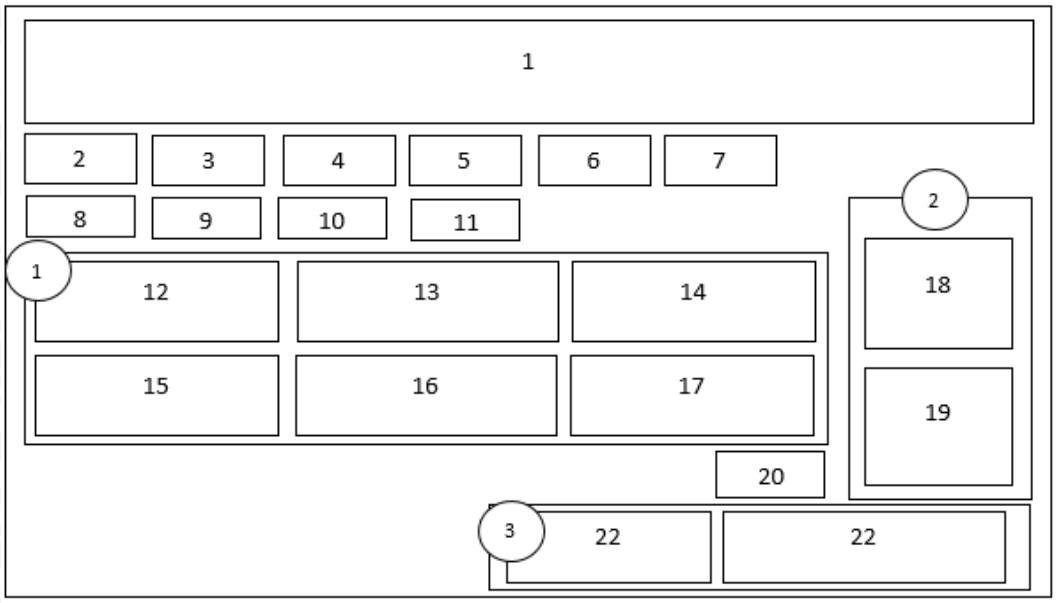
Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pengujian *Scale Factor*

4. Tombol yang mengarahkan ke halaman pengujian *MLEVEL*
5. Tombol yang mengarahkan ke halaman pengujian nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Combo box yang harus dipilih untuk ditentukan nilai *K*
9. Tombol yang melakukan pemrosesan terhadap data latih
10. Tombol yang melakukan pemrosesan pelatihan
11. Panel yang menampilkan citra asli
12. Panel yang menampilkan citra hasil *rescaling*
13. Panel yang menampilkan citra asli *grayscaling*
14. Panel yang menampilkan citra asli hasil *multilevel otsu*
15. Panel yang menampilkan bagian citra yang berpenyakit
16. Panel yang menampilkan bagian citra yang tidak berpenyakit
17. Tabel yang menampilkan rata-rata *red, green, blue* pada pemrosesan data latih
18. Tabel yang menampilkan hasil klasifikasi terhadap data latih
19. Label yang menampilkan akurasi
20. Panel yang menampilkan informasi RGB suatu citra
21. Panel yang menampilkan informasi file berupa nama file, panjang dan lebar

4.3.5 Halaman Pengujian Semua Data

Antarmuka halaman pengujian semua data terdiri dari 3 bagian. Bagian pertama memiliki fungsi dimana untuk menampilkan citra asli, citra hasil *rescaling*, citra *grayscaling*, citra hasil *multilevel thresholding otsu*, citra bagian penyakit dan citra tidak berpenyakit. Bagian kedua berisikan tabel, tabel pertama menampilkan RGB hasil dr pemrosesan data latih sementara tabel kedua menampilkan hasil klasifikasi sistem dan kelas asli pada citra. Bagian ketiga berisikan informasi dari citra berupa warna RGB dan juga informasi citra berupa nama file, panjang dan lebar citra. Antar muka halaman pengujian semua data ditunjukkan pada Gambar 4.21.



Gambar 4.27 Perancangan Antarmuka Halaman Pengujian Semua Data

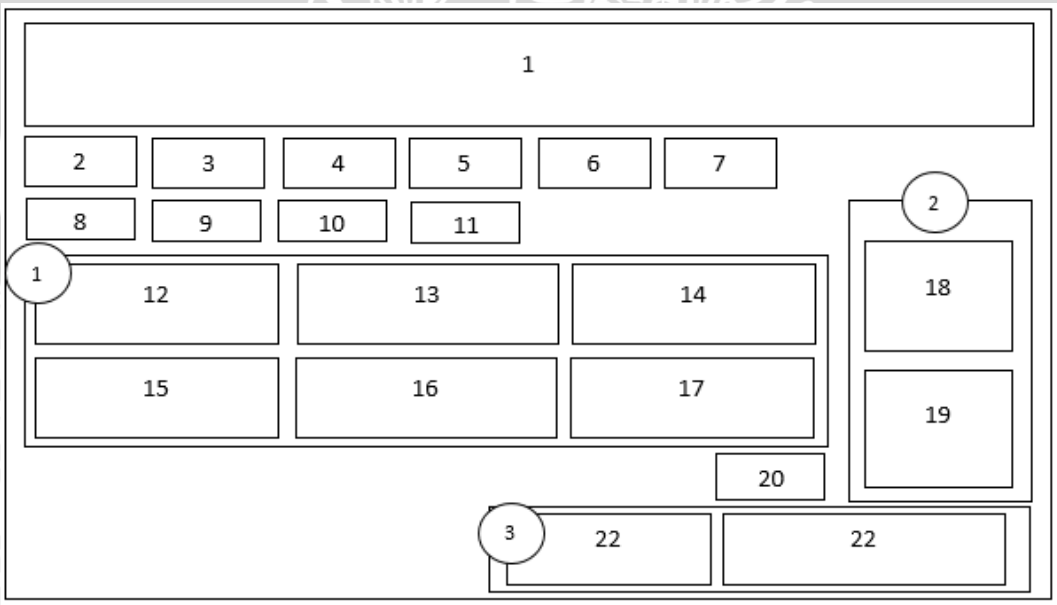
Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pelatihan *Scale Factor*
4. Tombol yang mengarahkan ke halaman pelatihan *MLEVEL*
5. Tombol yang mengarahkan ke halaman pelatihan nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Combo box yang harus dipilih untuk menentukan *path* data latih
9. Combo box yang harus dipilih untuk menentukan *path* data uji
10. Tombol yang melakukan pemrosesan terhadap data latih
11. Tombol yang melakukan pemrosesan pelatihan
12. Panel yang menampilkan citra asli
13. Panel yang menampilkan citra hasil *rescaling*
14. Panel yang menampilkan citra asli *grayscaling*
15. Panel yang menampilkan citra asli hasil *multilevel otsu*
16. Panel yang menampilkan bagian citra yang berpenyakit
17. Panel yang menampilkan bagian citra yang tidak berpenyakit

18. Tabel yang menampilkan rata-rata *red, green, blue* pada pemrosesan data latih
19. Tabel yang menampilkan hasil klasifikasi terhadap data latih
20. Label yang menampilkan akurasi
21. Panel yang menampilkan informasi RGB suatu citra
22. Panel yang menampilkan informasi file berupa nama file, panjang dan lebar

4.3.6 Halaman Pengujian Satu Data

Antarmuka halaman pengujian satu data terdiri dari 3 bagian. Bagian pertama memiliki fungsi dimana untuk menampilkan citra asli, citra hasil *rescaling*, citra *grayscaling*, citra hasil *multilevel thresholding otsu*, citra bagian penyakit dan citra tidak berpenyakit. Bagian kedua berisikan tabel, tabel pertama menampilkan RGB hasil dr pemrosesan data latih sementara tabel kedua menampilkan hasil klasifikasi sistem dan kelas asli pada citra. Bagian ketiga berisikan informasi dari citra berupa warna RGB dan juga informasi citr berupa nama file, panjang dan lebar citra. Antar muka halaman pengujian satu data ditunjukkan pada Gambar 4.22.



Gambar 4.28 Perancangan Antarmuka Halaman Pengujian Satu Data

Keterangan gambar :

1. Judul sistem
2. Tombol yang mengarahkan ke halaman awal
3. Tombol yang mengarahkan ke halaman pelatihan *Scale Factor*

4. Tombol yang mengarahkan ke halaman pelatihan *MLEVEL*
5. Tombol yang mengarahkan ke halaman pelatihan nilai *K*
6. Tombol yang mengarahkan ke halaman pengujian semua data
7. Tombol yang mengarahkan ke halaman pengujian satu data
8. Combo box yang harus dipilih untuk menentukan *path* data latih
9. Tombol yang memiliki fungsi untuk memilih *file* data uji
10. Tombol yang melakukan pemrosesan terhadap data latih
11. Tombol yang melakukan pemrosesan pelatihan
12. Panel yang menampilkan citra asli
13. Panel yang menampilkan citra hasil *rescaling*
14. Panel yang menampilkan citra asli *grayscaling*
15. Panel yang menampilkan citra asli hasil *multilevel otsu*
16. Panel yang menampilkan bagian citra yang berpenyakit
17. Panel yang menampilkan bagian citra yang tidak berpenyakit
18. Tabel yang menampilkan rata-rata *red, green, blue* pada pemrosesan data latih
19. Tabel yang menampilkan hasil klasifikasi terhadap data latih
20. Label yang menampilkan akurasi
21. Panel yang menampilkan informasi RGB suatu citra
22. Panel yang menampilkan informasi file berupa nama file, panjang dan lebar

BAB 5 IMPLEMENTASI

Bab ini adalah bab yang menjelaskan implementasi dari hasil perancagan yang telah dilakukan. Selain itu pada bab ini dijelaskan lingkungan implementasi, batasan implementasi, implementasi kode program, dan implementasi halaman antarmuka.

5.1 Lingkungan Implementasi

5.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk memenuhi kebutuhan sistem dalam penelitian ini adalah sebagai berikut :

1. *Processor* Intel Core i5-4200U @1.6GHz
2. *Memory* RAM 4GB
3. *Harddisk* 500GB
4. Monitor Laptop 14 inch
5. *Keyboard* Internal Laptop
6. *Touchpad* Internal Laptop
7. Canon EOS 60D dengan lensa 18-55 mm, ISO 100

5.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang diguakan untuk memenuhi kebutuhan sistem dalam penelitan ini adalah sebagai berikut :

1. Microsoft Office Word 2013, digunakan untuk menuliskan dokumentasi dan laporan penelitian.
2. Microsoft Office Visio 2016, digunakan untuk menuliskan *flowchart* atau diagram alir sistem.
3. Microsoft Office Power Point 2013, digunakan untuk mempresentasikan penelitian.
4. Microsoft Office Excel 2013, digunakan untuk menuliskan perhitungan manual sistem.
5. Snipping Tool, digunakan untuk mengcapture gambar.
6. Adobe Acrobat Reader DC 2015, digunakan untuk membaca refrensi *journal*, *ebook, paper.*
7. Adobe Photoshop CS4, digunakan untuk mengolah citra daun jeruk.
8. Netbeans IDE 8.0, digunakan untuk menuliskan kode program dalam membuat sistem.

5.2 Batasan Implementasi

Beberapa batasan dalam mengimplementasikan aplikasi identifikasi penyakit daun jeruk menggunakan *preprocessing* metode *Multilevel Otsu* adlaah sebagai berikut :

1. Jumlah data yang digunakan adalah sebanyak 200 citra daun yang terbagi kedalam 5 kelas. Masing masing kelas berjumlah 30 data digunakan untuk pelatihan dan 10 data digunakan untuk pengujian. Data diperoleh dari Balai Penelitian Tanaman Jeruk dan Buah Subtropika (BALITJESTRO) kota Batu, Jawa Timur.
2. Format data citra yang digunakan untuk melakukan implementasi adalah format *jpg.*
3. Ukuran data citra yang digunakan berukuran 275x150 piksel.
4. Jumlah citra daun yang digunakan berjumlah satu citra daun pada setiap melakukan implementasi.
5. Latar belakang citra memiliki warna putih.
6. Jumlah kelas yang digunakan dalam mengimplementasi aplikasi berjumlah 5 kelas yang terdiri dari :
   a. Kelas satu, daun memiliki penyakit *Downy Mildew*
   b. Kelas dua, daun memiliki penyakit Cendawan Jelaga
   c. Kelas tiga, daun memiliki penyakit *Citrus Vein Phloem Degeneration* (CVPD)
   d. Kelas empat, daun memiliki penyakit Defisiensi
   e. Kelas lima daun sehat
7. *Preprocessing* yang digunakan memakai *rescaling, grayscaling,* dan *Multilevel Otsu*
8. Jumlah ciri yang digunakan sebanyak 3 yaitu rata-rata *red,* rata-rata *green*, rata-rata *blue.*
9. Metode pengujian yang digunakan adalah *K-Nearest Neighbor* (KNN)
10. Pengujian dilakukan sebanyak 4 pengujian yaitu pengujian nilai *scale factor* (1.1, 1.2. 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0), pengujian nilai *MLEVEL* (Level 2, 3, 4, 5), pengujian nilai *K* (1, 2, 3, 4, 5).

5.3 Implementasi Kode Program

Pada sub bab ini akan dijelaskan impementasi yang dibuat sesuai dengan perancangan yang telah dibuat pada bab IV. Implementasi yang dilakukan pada sub bab ini meliputi implementasi *preprocessing* untuk proses *rescaling* dan *grayscaling*, metode *Multilevel Otsu*, ekstraksi ciri, proses pelatihan, proses pengujian (metode KNN), perhitungan akurasi.

5.3.1 Implementasi *Preprocessing*

Proses awal yang dilakukan adalah *preprocessing*, dalam *preprocessing* dilakukan 3 tahap yaitu *resizing*, *rescaling* dan *grayscaling*. *Resizing* dilakukan untuk merubah ukuran pada suatu citra dengan masukan sebuah nilai *width* dan *height* baru. *Rescaling* dilakukan untuk menambah nilai *scale factor* sebagai sebuah nilai konstanta untuk menambah tingkat kecerahan pada sebuah citra. *Grayscaling* dilakukan untuk mencari rata-rata nilai RGB untuk dilakukan mencari tingkat gray level atau tingkat keabuan pada sebuah citra. Proses *Resizing* ditunjukkan pada *Sourcecode* 5.1 dengan nama fungsi "*resize()*". Bagian kedua merupakan implementasi dari proses penambahan tingkat kecerahan tau *rescaling*. Proses *rescaling* ditunjukkan pada *Sourcecode* 5.2. Bagian terakhir merupakan implementasi dari proses pencarian tingkat keabuan citra atau yang disebut *grayscaling*, yang ditunjukkan pada *Sourcecode* 5.3 dengan nama fungsi "*setRGBtoGrayLevel()*".

```
1  public static BufferedImage resize(BufferedImage img, int
   newW, int newH) {
2        int w = img.getWidth();
3        int h = img.getHeight();
4        BufferedImage dimg = new BufferedImage(newW,
   newH,img.getType());
5        Graphics2D g = dimg.createGraphics();
6        g.drawImage(img, 0, 0, newW, newH, 0, 0, w, h, null);
7        g.dispose();
8        return dimg;
9  }
```

***Sourcecode* 5.1.** *Preprocessing* bagian pertama *resize().*

Penjelasan *sourcecode* 5.1 adalah sebagai berikut :

1. Pada baris 1 merupakan penamaan nama fungsi *resize()* bertipe *BufferedImage* dengan parameter objek img dari *BufferedImage*, variabel *newW* dan *newH* masing-masing bertipe data *Integer*.
2. Pada baris 2-3 merupakan pemberian nilai terhadap variabel *w* dan *h* dengan pengambilan nilai dari *method getWidth()* dan *getHeight* pada objek *img*.
3. Pada baris 4 merupakan inisiasi objek dari *dimg* dari *BufferedImage* dengan isi parameter adalah variabel dari *newW, newH* dan *method getType()* dari objek *img*.
4. Pada baris 5 merupakan inisiasi objek *g* dari *Graphics2D* dengan pemberian nilai dari *method createGraphics()* dari objek *dimg*.
5. Pada baris 6 pemanggilan *method drawImage* pada objek *g* dengan parameter objek *img*, nilai 0, variabel *newW* dan *newH*, nilai 0, variabel *w* dan *h* serta null.
6. Pada baris 7 merupakan pemanggilan *method dispose* pada objek *g*.
7. Pada baris 8 merupakan nilai pengembalian pada objek *dimg*.

```
1   RescaleOp rescaleOp = new RescaleOp(1.4f, 0, null);
2              BufferedImage citra = ImageIO.read(new
    File(location+""+fileEntry.getName()));
3              citra=rescaleOp.filter(citra, null);
```

***Sourcecode* 5.2.** *Preprocessing* bagian kedua *rescaling.*

Penjelasan *sourcecode* 5.2 adalah sebagai berikut :

1. Pada baris 1 merupakan inisiasi objek *rescaleOp* dari *RescaleOp* dengan parameter nilai 1.4f yang menunjukkan nilai *scalefactor,* 0 dan null.
2. Pada baris 2 merupakan inisiasi objek *citra* dengan pemanggilan letak file citra yang di *rescaling.*
3. Pada baris 3 merupakan pemanggilan *method filter* dengan parameter objek *citra* dan null, disimpan pada objek *citra*.

```
1   public BufferedImage setRGBtoGrayLevel(BufferedImage img){
2         BufferedImage citra = new
    BufferedImage(img.getWidth(),
    img.getHeight(),BufferedImage.TYPE_INT_RGB);
3         int width = img.getWidth();
4         int height = img.getHeight();
5         for (int x = 0; x < width; x++) {
6               for (int y = 0; y < height; y++) {
7                     int c1 = img.getRGB(x, y);
8                     int a = (c1>>24)&0xff;
9                     int red = (c1 >> 16) & 0xFF;
10                    int green = (c1 >> 8) & 0xFF;
11                    int blue = (c1) & 0xFF;
12                    int gray = (red+green+blue)/3;
13                    int pxl = (a<<24) | (gray<<16) | (gray<<8)
    | gray;
14                    citra.setRGB(x, y, pxl);
                }
          }
15        return citra;
    }
```

***Sourcecode* 5.3.** *Preprocessing* bagian ketiga *grayscaling.*

Penjelasan *Sourcecode* 5.3 adalah sebagai berikut :

1. Pada baris 1 merupakan pembuatan fungsi atau *method* bernama *setRGBtoGrayLevel()* dengan tipe *BufferedImage* yang memiliki parameter objek img dari *BufferedImage*.
2. Pada baris 2 merupakan inisiasi objek *citra* dari *BufferedImage* dengan parameter masukan nilai pemanggilan method *getWidth()* dan *getHeight()* dari objek *img* serta tipe citra.
3. Pada baris 3 – 4 merupakan pemberian nilai pada variabel *width* dan *height*, masing-masing diberikan pada *method getWidth()* dan *getHeight()* pada objek *img*.
4. Pada baris 5 merupakan perulangan sebanyak variabel *width* pada variabel *x*
5. Pada baris 6 merupakan perulangan sebanyak variabel *height* pada variabel *y*

6. Pada baris 7 pemberian variabel *c1* dengan nilai pemanggilan *method getRGB* berparamater variabel *x* dan *y* objek *img*.
7. Pada baris 8 – 11 merupakan pemberian nilai *a, red, green, blue* dengan nilai RGB.
8. Pada baris 12 merupakan penjumlahan variabel *red, green, blue* dan dibagi dengan nilai 3, disimpan pada variabel *gray.*
9. Pada baris 13 merupakan pemberian variabel *p* dengan nilai dari variabel *a, red, green, blue*
10. Pada baris 14 merupakan pemanggilan *method setRGB* dengan parameter *x, y* dan *pxl* pada objek *citra*.
11. Pada baris 15 merupakan nilai pengembalian objek *citra.*

5.3.2 Implementasi Metode *Multilevel Otsu*

Proses selanjutnya adalah melakukan implementasi dari metode *multilevel otsu*. Proses ini bertujuan untuk mengambil bagian daun yang berpenyakit saja dengan cara menghitamkan bagian yang tidak diolah. Proses *multilevel otsu* terdiri dari 4 fungsi yang terdiri dari *buildHistogram, buildLookUp, findThreshold, setThreholdingMultiOtsu*. Fungsi *buildHistogram* merupakan proses untuk pencarian nilai histogram dan juga pencarian nilai rata-rata peluang histogram, ***Sourcecode* 5.4**. Fungsi *buildLookUp* merupakan fungsi untuk membangkitkan nilai H dimana nantinya dicari untuk didapatkan nilai *threshold*, fungsi ini ditunjukkan pada ***Sourcecode* 5.5**. Fungsi *findThreshold* merupakan fungsi untuk mencari dan mendapatkan nilai *threshold*, fungsi ini ditunjukkan pada ***Sourcecode* 5.6**. Fungsi *setThresholdingMultiOtsu* merupakan proses untuk memasukkan hasil segmentasi citra dari *multilevel otsu* terhadap citra *grayscale* untuk didapatkan bagian penyakit dan menghitamkan pada bagian yang tidak diproses. Fungsi ini ditunjukkan pada ***Sourcecode* 5.7**.

```
1   public void buildHistogram(float [] h, int [] pixels, int
    width, int height){
2         for (int i=0; i < width*height; ++i){
3               int val = 0xff & pixels[i];
4               if(val<=255){
5                     h[(int) (pixels[i]&0xff)]++;
                }
          }
6         float [] bin = new float[NGRAY];
7         float hmax = 0.f;
8         for (int i=0; i < NGRAY; ++i){
9               bin[i] = (float) i;
10              h[i] = i*(h[i]/((float) (width*height)));
11              if (hmax < h[i])
12                    hmax = h[i];
          }
    }
```

***Sourcecode* 5.4.** *Multilevel Otsu* proses *buildHistogram.*

Penjelasan *sourcecode* 5.4 adalah sebagai berikut :

1. Pada baris 1 merupakan pembuatan fungsi bernama *buildHistogram* dengan parameter nilai *histogram, pixels, width* dan *height*.
2. Pada baris 2 merupakan perulangan sebanyak perkalian antara variabel *width* dan *height*.
3. Pada baris 3 inisialisasi variabel val dengan variabel array pixels
4. Pada baris 4 seleksi kondisi dengan menggunakan if antara variabel val dengan nilai 255 menggunakan kondisi kurangdari samadengan
5. Pada baris 5 proses untuk melakukan penambahan atau *increment* variabel array h dengan indeks dari variabel array *pixels*
6. Pada baris 6 inisialisasi variabel array *bin* bertipe data *float* dengan panjang array senilai dari variabel *NGRAY*
7. Pada baris 7 inisialisasi variabel *hmax* dengan nilai 0
8. Pada baris 8 proses perulangan sebanyak variabel *NGRAY*
9. Pada baris 9 merupakan proses inisialisasi variabel array *bin* dengan nilai variabel *i*
10. Pada baris 10 merupakan proses perhitungan variabel array *h* dibagi dengan perkalian variabel *width* dan *height*, hasil tersebut di kalikan dengan variabel *i* dan disimpan pada variabel array *h*
11. Pada baris 11 merupakan seleksi kondisi dengan menggunakan if antara variabel *hmax* dengan variabel array *h* dengan menggunakan kondisi kurang dari.
12. Pada baris 12 merupakan pemberian nilai variabel *hmax* dengan nilai dari variabel array *h.*

```
1  public void buildLookUp(float [][] P, float [][] S, float
   [][] H, float [] h){
2        for (int j=0; j < NGRAY; j++)
3              for (int i=0; i < NGRAY; ++i){
4                    P[i][j] = 0.f;
5                    S[i][j] = 0.f;
6                    H[i][j] = 0.f;
                }
7        for (int i=1; i < NGRAY; ++i){
8              P[i][i] = h[i];
9              S[i][i] = ((float) i)*h[i];
         }
10       for (int i=1; i < NGRAY-1; ++i){
11             P[1][i+1] = P[1][i] + h[i+1];
12             S[1][i+1] = S[1][i] + ((float) (i+1))*h[i+1];
         }
13       for (int i=2; i < NGRAY; i++)
14             for (int j=i+1; j < NGRAY; j++){
15                   P[i][j] = P[1][j] - P[1][i-1];
16                   S[i][j] = S[1][j] - S[1][i-1];
           }
17       for (int i=1; i < NGRAY; ++i)
18         for (int j=i+1; j < NGRAY; j++){
```

```
19              if (P[i][j] != 0)
20                  H[i][j] = (S[i][j]*S[i][j])/P[i][j];
21              else
22                  H[i][j] = 0.f;
                }
    }
```

***Sourcecode* 5.5.** *Multilevel Otsu* proses *buildLookUp*

Penjelasan *sourcecode* 5.5 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *buildLookUp* dengan parameter nilai array variabel *P, S, H,* dan juga variabel array *h*
2. Pada baris 2 – 3 merupakan proses perulangan sebanyak variabel *NGRAY*
3. Pada baris 4 – 6 merupakan proses inisialisasi terhadap variabel array *P, S, H* dengan nilai 0
4. Pada baris 7 merupakan proses perulangan sebanyak variabel *NGRAY*
5. Pada baris 8 merupakan proses pemberian nilai terhadap variabel array *P* dengan nilai variabel h
6. Pada baris 9 merupakan proses pemberian nilai terhadap variabel *S* dengan nilai perkalian antara variabel *i* dengan variabel array *h*
7. Pada baris 10 merupakan proses perulangan sebanyak variabel *NGRAY-1*
8. Pada baris 11 merupakan proses penjumlahan antara variabel array *P* dengan variabel array *h*, disimpan pada variabel array *S*
9. Pada baris 12 merupakan proses perkalian antara variable *i* dengan variabel array *h*, hasil tersebut di jumlahkan dengan variabel array *S*, disimpan pada variabel array *S*.
10. Pada baris 13 – 14 merupakan proses *nested looping* sebanyak variabel *NGRAY*
11. Pada baris 15 merupakan proses pengurangan variabel array *P* indeks 1, j dengan variabel array *P* indeks 1, i-1. Disimpan pada variabel array P
12. Pada baris 16 merupakan proses pengurangan variabel array *S* indeks 1, j dengan variabel array *S* indeks 1, i-1. Disimpan dalam variabel array *S.*
13. Pada baris 17 – 18 merupakan proses *nested looping* sebanyak variabel *NGRAY*
14. Pada baris 19 merupakan proses seleksi kondisi antara variabel array *P* dengan nilai 0 menggunakan kondisi tidak sama dengan
15. Pada baris 20 merupakan proses perhitungan, proses pertama adalah perkalian antara variabel array *S* dengan variabel array *S*, hasil perkalian tersebut dibagi dengan variabel *P*
16. Pada baris 21 merupakan seleksi kondisi *else*

17. Pada baris 22 merupakan pemberian nilai terhadap variabel array *H* sebesar 0.

```
1  public float findThreshold(int MLEVEL, float [][] H, int [] t){
2        t[0] = 0;
3        float maxSig= 0.f;
4      switch(MLEVEL){
5        case 2 :
6              for (int i= 1; i < NGRAY-MLEVEL; i++){ // t1
7                    float Sq = H[1][i] + H[i+1][255];
8                    if (maxSig < Sq){
9                          t[1] = i;
10                         maxSig = Sq;
                     }
                }
11       break;
12       case 3:
13             for (int i= 1; i < NGRAY-MLEVEL; i++) // t1
14                   for (int j = i+1; j < NGRAY-MLEVEL +1; j++){ // t2
15                         float Sq = H[1][i] + H[i+1][j] + H[j+1][255];
16                               if (maxSig < Sq){
17                                     t[1] = i;
18                                     t[2] = j;
19                                     maxSig = Sq;
                                 }
                     }
20       break;
21       case 4 :
22             for (int i= 1; i < NGRAY-MLEVEL; i++) // t1
23                   for (int j = i+1; j < NGRAY-MLEVEL +1; j++) // t2
24                         for (int k = j+1; k < NGRAY-MLEVEL + 2; k++){ // t3
25                               float Sq = H[1][i] + H[i+1][j] + H[j+1][k] + H[k+1][255];
26                                     if (maxSig < Sq){
27                                           t[1] = i;
28                                           t[2] = j;
29                                           t[3] = k;
30                                                 maxSig = Sq;
                                       }
                                 }
31 break;
32 case 5 :
33       for (int i= 1; i < NGRAY-MLEVEL; i++) // t1
34             for (int j = i+1; j < NGRAY-MLEVEL +1; j++) // t2
35                   for (int k = j+1; k < NGRAY-MLEVEL + 2; k++) // t3
36                         for (int m = k+1; m < NGRAY-MLEVEL + 3; m++){ // t4
37                               float Sq = H[1][i] + H[i+1][j] + H[j+1][k] + H[k+1][m] + H[m+1][255];
38                                     if (maxSig < Sq){
39                                           t[1] = i;
40                                           t[2] = j;
```

```
41                                                  t[3] = k;
42                                                  t[4] = m;
43                                                  maxSig = Sq;
                                                }
                                            }
        }
44      return maxSig;
    }
```

***Sourcecode* 5.6.** *Multilevel Otsu* proses *findThreshold*

Penjelasan *sourcecode* 5.6 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *findThreshold* dengan parameter *MLEVEL,* variabel array *H*, dan variabel array *t*
2. Pada baris 2 merupakan proses inisialisasi nilai 0 terhadap variabel array *t* indeks ke-0
3. Pada baris 3 merupakan proses inisialisasi nilai 0 terhadap variabel *maxSig*
4. Pada baris 4 merupakan proses seleksi dengan menggunakan *switch* dengan parameter variabel *MLEVEL*
5. Pada baris 5 merupakan proses seleksi dengan *case* bernilai 2
6. Pada baris 6 merupakan proses perulangan sebanyak variabel nilai *NGRAY-MLEVEL*
7. Pada baris 7 merupakan proses penjumlahan variabel array *H* denga indeks 1, I dengan variabel array *H* indeks i+1, 255
8. Pada baris 8 merupakan seleksi kondisi dengan menggunakan if antara variabel *maxSig* dengan variabel *Sq* menggunakan kondisi kurang dari
9. Pada baris 9 merupakan pemberian nilai terhadap variabel array *t* indeks 1 dengan nilai *i*
10. Pada baris 10 merupakan pemberian nilai variabel *maxSig* dengan nilai dari variabel *Sq*
11. Pada baris 11 operasi break
12. Pada baris 12 merupakan proses seleksi dengan *case* bernilai 3
13. Pada baris 13 - 14 merupakan proses *nested looping* sebanyak variable *NGRAY*
14. Pada baris 15 merupakan proses penjumlahan variabel array *H,* variabel array *H* indeks i+1, j dan variabel array *H* indeks j+1, 255
15. Pada baris 16 merupakan seleksi kondisi dengan menggunakan if antara variabel *maxSig* dengan variabel *Sq* menggunakan kondisi kurang dari
16. Pada baris 17 - 18 merupakan proses pemberian nilai variabel array *t* indeks 1 dan indeks 2 masing-masing bernilai variable *i* dan *j*
17. Pada baris 19 merupakan proses pemberian nilai variabel *maxSig* dengan nilai variabel *Sq*

18. Pada baris 20 merupakan operasi *break*
19. Pada baris 21 merupakan proses seleksi dengan *case* bernilai 4
20. Pada baris 22 - 24 merupakan proses *nested looping* sebanyak variable *NGRAY-MLEVEL*
21. Pada baris 25 merupakan proses penjumlahan variabel array *H,* variabel array *H* indeks i+1, j, variabel array *H* indeks j+1, k dan variabel array *H* indeks k+1, 255
22. Pada baris 26 merupakan seleksi kondisi dengan menggunakan if antara variabel *maxSig* dengan variabel *Sq* menggunakan kondisi kurang dari
23. Pada baris 27 - 29 merupakan proses pemberian nilai variabel array *t* indeks 1, indeks 2, indeks 3 masing-masing bernilai variable *i*, *j, k*
24. Pada baris 30 merupakan proses pemberian nilai variabel *maxSig* dengan nilai variabel *Sq*
25. Pada baris 31 merupakan operasi *break*
26. Pada baris 32 merupakan proses seleksi dengan *case* bernilai 5
27. Pada baris 33 - 36 merupakan proses *nested looping* sebanyak variable *NGRAY-MLEVEL*
28. Pada baris 37 merupakan proses penjumlahan variabel array *H,* variabel array *H* indeks i+1, j, variabel array *H* indeks j+1, k, variabel array *H* indeks k+1, m, dan variabel array *H* indeks m+1, 255
29. Pada baris 38 merupakan seleksi kondisi dengan menggunakan if antara variabel *maxSig* dengan variabel *Sq* menggunakan kondisi kurang dari
30. Pada baris 39 - 42 merupakan proses pemberian nilai variabel array *t* indeks 1, indeks 2, indeks 3, indeks 4 masing-masing bernilai variable *i*, *j, k, m*
31. Pada baris 43 merupakan proses pemberian nilai variabel *maxSig* dengan nilai variabel *Sq*
32. Pada baris 44 merupakan proses pengembalian nilai dari variabel *maxSig*.

```
1  public BufferedImage setThresholdingMultiOtsu(BufferedImage
   citra, int MLEVEL, int [] t, int [] pixels, int width, int
   height){
2        BufferedImage dest = new BufferedImage(width, height,
   citra.TYPE_INT_RGB);
3        for(int i=0; i < width*height; i++){
4              int pxl = 0xff & pixels[i];
5              if(pxl < t[1] || pxl > t[4]){
6                    int red = 255;
7                    int green = 255;
8                    int blue = 255;
9                    int val = blue + (green << 8) + (red <<
   16);
10                   dest.setRGB(i%width, i/width, val);

               }
```

```
11              else{
12                  int val = pixels[i];
13                  int red = 0;
14                  int green = 0;
15                  int blue = 0;
16                  val = blue + (green << 8) + (red << 16);
17                  dest.setRGB(i%width, i/width, val);
                }
        }
18      return dest;
    }
```

***Sourcecode* 5.7.** *Multilevel Otsu* proses *setThresholdingMultiOtsu*

Penjelasan *sourcecode* 5.7 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *setThresholdingOtsu* dengan parameter objek *citra*, *MLEVEL,* variabel array *t,* variabel array *pixels, width, height*
2. Pada baris 2 Inisiasi objek *dest* pada class *BufferedImage*
3. Pada baris 3 merupakan proses perulangan sebanyak nilai perkalian *width* dan *height*
4. Pada baris 4 inisialisasi variabel *pxl* dengan nilai variabel array *pixels*
5. Pada baris 5 proses seleksi kondisi antara variabel *pxl* dengan variabel array *t* indeks 1 dengan kondisi kurang dari, dan juga kondisi atau antara variael *pxl* dengan variabel array *t* indeks 4
6. Pada baris 6 – 8 pemberian nilai variabel *red, green, blue* dengan nilai 255
7. Pada baris 9 merupakan penjumlahan variabel *red, green, blue* disimpan pada variabel val
8. Pada baris 10 pemanggilan method *setRGB* pada objek *dest*
9. Pada baris 11 operasi *else*
10. Pada baris 12 pemberian nilai variabel *val* dengan nilai variabel array *pixels*
11. Pada baris 13 – 15 merupakan inisialisasi nilai 0 pada variabel *red, green, blue*
12. Pada baris 16 penjumlahan variabel *red, green, blue*
13. Pada baris 17 merupakan pemanggilan method *setRGB* objek dest
14. Pada baris 18 pengembalian nilai objek *dest*

5.3.3 Implementasi Proses Konvolusi Citra

Proses konvolusi citra merupakan proses penjumlahan citra dengan menjumlahkan RGB pada masing-masing citra. Pada penelitian ini, proses konvolusi dilakukan dengan menjumlahkan citra tersegmentasi otsu dengan citra asli agar dapat diketahui bagian dari penyakit jeruk. Implementasi proses konvolusi citra ditunjukkan pada ***Sourcecode* 5.8** dengan nama fungsi *"konvolusi()"*.

```
1   public BufferedImage konvolusi(BufferedImage citra1,
    BufferedImage citra2) throws IOException{
2         BufferedImage citra3 = new
    BufferedImage(citra1.getWidth(),
    citra2.getHeight(),BufferedImage.TYPE_INT_RGB);
3         int width = citra1.getWidth();
4         int height = citra1.getHeight();
5         for (int x = 0; x < width; x++) {
6               for (int y = 0; y < height; y++) {
7                     int c1 = citra1.getRGB(x, y);
8                     int c2 = citra2.getRGB(x, y);
9                     int red1 = (c1 >> 16) & 0xFF;
10                    int green1 = (c1 >> 8) & 0xFF;
11                    int blue1 = (c1) & 0xFF;
12                    int red2 = (c2 >> 16) & 0xFF;
13                    int green2 = (c2 >> 8) & 0xFF;
14                    int blue2 = (c2) & 0xFF;
15                    int red_konv = red1 + red2;
16                    red_konv = valueCheck(red_konv);
17                    int green_konv = green1 + green2;
18                    green_konv = valueCheck(green_konv);
19                    int blue_konv = blue1 + blue2;
20                    blue_konv = valueCheck(blue_konv);
21                    int pxl = blue_konv + (green_konv << 8) +
    (red_konv << 16);
22                    citra3.setRGB(x, y, pxl);
                }
          }
23        return citra3;
    }
```

***Sourcecode* 5.8.** Implementasi Proses Konvolusi.

Penjelasan *sourcecode* 5.8 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *konvolusi* dengan parameter objek *citra1* dan *citra2*
2. Pada baris 2 inisialisasi objek *citra3* pada *class BufferedImage*
3. Pada baris 3 – 4 merupakan pemberian nilai variabel *width* dan *height* dengan pemanggilan method *getWidth* dan *getHeight* pada objek citra1
4. Pada baris 5 – 6 merupakan proses *nested loop* sebanyak *width* dan *height*
5. Pada baris 7 – 8 merupakan pemberian nilai variabel *c1* dan *c2* dengan pemanggilan method *getRGB* pada objek citra1 dan citra 2
6. Pada baris 9 – 11 merupakan pemberian nilai variabel *red1, green1, blue1* dengan nilai *c1*
7. Pada baris 12 – 14 merupakan pemanggilan variabel *red2, green2, blue2* dengan nilai c2
8. Pada baris 15 merupakan proses penjumlahan variabel *red1* dan *red2* disimpan pada variabel *red_konv*
9. Pada baris 16 pemanggilan method *valueCheck* dengan parameter variabel *red_konv*, disimpan pada variabel *red_konv*

10. Pada baris 17 merupakan proses penjumlahan variabel *green1* dan *green2* disimpan pada variabel *green_konv*
11. Pada baris 18 pemanggilan method *valueCheck* dengan parameter variabel *green_konv*, disimpan pada variabel *green_konv*
12. Pada baris 19 merupakan proses penjumlahan variabel *blue1* dan *blue2* disimpan pada variabel *blue_konv*
13. Pada baris 20 pemanggilan method *valueCheck* dengan parameter variabel *blue_konv*, disimpan pada variabel *blue_konv*
14. Pada baris 21 merupakan proses penjumlahan variabel *blue_konv, green_konv* dan *red_konv*
15. Pada baris 22 pemanggilan method *setRGB* objek *citra3*
16. Pada baris 23 pengembalian nilai objek *citra3*

5.3.4 Implementasi Ekstraksi Ciri

Implementasi ekstraksi ciri bertujuan untuk mendapatkan ciri dari daun yang dimasukkan kedalam proses pelatihan dan dilakukan pengujian untuk menguji data uji terhadap data latih. Proses pengambilan ciri pada penelitian ini menggunakan pencarian rata-rata dari warna *red, green,* dan *blue*. Implementasi proses dari ekstraksi ciri ditunjukkan pada ***Sourcecode* 5.9** dengan nama fungsi *"getFeature()"*.

```
1   public void getFeature(BufferedImage img) {
2         int tempr=0, tempg=0, tempb=0;
3         int pembagi=41250;
4               for (int i = 0; i < img.getWidth(); i++) {
5                     for (int j = 0; j < img.getHeight(); j++)
    {
6                           int pixel = img.getRGB(i, j);
7                           int red = (pixel >> 16) & 0xff;
8                           int green = (pixel >> 8) & 0xff;
9                           int blue = (pixel) & 0xff;
10                          if
    (red==255&&green==255&&blue==255){
11                                red=0;
12                                green=0;
13                                blue=0;
14                                pembagi--;
                            }
15                          tempr+=red;
16                          tempg+=green;
17                          tempb+=blue;
                      }
                }
18        nilair=tempr/pembagi;
19        nilaig=tempg/pembagi;
20        nilaib=tempb/pembagi;
    }
```

***Sourcecode* 5.9.** Implementasi Proses Ekstraksi Ciri.

Penjelasan *sourcecode* 5.9 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *getFeature* dengan parameter objek *img*
2. Pada baris 2 merupakan pemerian nilai 0 pada variabel *tempr, tempg, tempb*
3. Pada baris 3 merupakan pemberian nilai pada variabel pembagi dengan nilai 41250
4. Pada baris 4 - 5 merupakan proses perulangan sebanyak nilai dari pemanggilan method *getWidth* dan *getHeight*
5. Pada baris 6 merupakan proses pemberian nilai pada variabel *pixel* dengan nilai pemanggilan dari metod *getRGB*
6. Pada baris 7 – 9 merupakan pemberian nilai *red, green, blue* oleh nilai *pixel* dengan proses pengambilan nilai biner
7. Pada baris 10 merupakan seleksi kondisi antara variabel *red, green, blue* dengan nilai 255
8. Pada baris 11 – 13 merupakan proses pemberian nilai 0 terhadap variabel *red, green, blue*
9. Pada baris 14 merupakan proses *decrement* variabel *pembagi*
10. Pada baris 15 merupakan proses penjumlahan variabel *tempr* dengan varabel *red*, disimpan pada variabel *tempr*
11. Pada baris 16 merupakan proses penjumlahan variabel *tempg* dengan varabel *green*, disimpan pada variabel *tempg*
12. Pada baris 17 merupakan proses penjumlahan variabel *tempb* dengan varabel *blue*, disimpan pada variabel *tempb*
13. Pada baris 18 merupakan proses pembagian variabel *tempr* dengan *pembagi* dismpan pada variabel *nilair*
14. Pada baris 19 merupakan proses pembagian variabel *tempg* dengan *pembagi* dismpan pada variabel *nilaig*
15. Pada baris 20 merupakan proses pembagian variabel *tempb* dengan *pembagi* dismpan pada variabel *nilaib*

5.3.5 Implementasi Metode KNN

Implementasi metode KNN digunakan untuk melakukan pelatihan dan uji data terhadap data citra. Proses KNN dilakukan dengan mencari ketetanggaan terdekat pada data latih dan data uji. Pada KNN diperlukan proses input nilai *K* yang digunkan untuk mencari jumlah ketetanggaan yang diingankan. Pencarian ketetanggan terdekat dilakukan dengan cara melakukan perhitungan jarak *Euclidean*. Implementasi metode KNN ditunjukkan pada ***Sourcecode 6.0*** dengan nama fungsi *"hitungData()"* yang terletak pada kelas *KNN*.

```
1   public String hitungData(int k, int jumlah, JTable tabel,
    rgb img) {
2         int rr[]=new int[jumlah];
```

```
3       int gg[]=new int[jumlah];
4       int bb[]=new int[jumlah];
5       String hasil = null;
6       for (int i=0;i<jumlah;i++){
7             rr[i]=Integer.parseInt(tabel.getValueAt(i,
    1).toString());
8             gg[i]=Integer.parseInt(tabel.getValueAt(i,
    2).toString());
9             bb[i]=Integer.parseInt(tabel.getValueAt(i,
    3).toString());
        }
10      float euc[]=new float[jumlah];
11      float id1=0, id2=0, id3=0, id4=0, id5=0,tetangga1=0,
    tetangga2=0, tetangga3=0, tetangga4=0, tetangga5=0,temp=500;
12      for (int i=0;i<jumlah;i++){
13            euc[i]=(float) Math.sqrt(Math.pow((rr[i]-
    img.nilair),2)+Math.pow((gg[i]-
    img.nilaig),2)+Math.pow((bb[i]-img.nilaib),2));
14            if(k==2){
15                  if(euc[i] < temp) {
16                        temp=euc[i];
17                        tetangga2 = tetangga1;
18                        tetangga1 = euc[i];
19                        id1=i;
              }
20                  if ((euc[i] < tetangga2 || tetangga2 ==
    tetangga1) && euc[i] != tetangga1){
21                        tetangga2 = euc[i];
22                        id2=i;
                  }
          }
23            if(k==2){
24            System.out.println(tetangga1+"\t"+tetangga2);
25            System.out.println((1+id1)+"\t\t"+(1+id2));
26            double t1 = 1/Math.pow(tetangga1, 2);
27            double t2 = 1/Math.pow(tetangga2, 2);
28                  if(t1>t2){
29                hasil = tabel.getValueAt((int)id1,
    4).toString();
              }
30            else{
31                hasil = tabel.getValueAt((int)id2,
    4).toString();
              }
          }
        }
32      return hasil;
}
```

***Sourcecode* 6.0.** Implementasi Metode KNN

Penjelasan *sourcecode* 6.0 adalah sebagai berikut :

1. Pada baris 1 merupakan proses pembuatan fungsi bernama *hitungData* dengan parameter nilai variabel *k, jumlah,* objek *tabel*, objek *img*.
2. Pada baris 2-4 merupakan proses untuk menentukan jumlah panjang array pada variabel *rr, gg, bb* dengan panjang array sebesar variabel *jumlah*
3. Pada baris 5 pemberian nilai pada variabel hasil dengan nilai null

4. Pada baris 6 proses perulangan sebanyak variabel jumlah
5. Pada baris 7 merupakan pemberian variabel array *rr* dengan nilai pada *tabel* yang di ambil pada baris ke-i kolom ke-1
6. Pada baris 8 merupakan pemberian variabel array *gg* dengan nilai pada *tabel* yang di ambil pada baris ke-i kolom ke-2
7. Pada baris 9 merupakan pemberian variabel array *bb* dengan nilai pada *tabel* yang di ambil pada baris ke-i kolom ke-3
8. Pada baris 10 merupakan pemberian panjang array variabel *euc* sejumlah variabel *jumlah*
9. Pada baris 11 merupakan inisialisasi variabel
10. Pada baris 12 merupakan proses perulangan sebanyak variabel *jumlah*
11. Pada baris 13 merupakan proses perhitungan nilai *Euclidean distance*
12. Pada baris 14 seleksi kondisi dengan menggunakan if antara variabel array *k* dengan nilai 2 dengan kondisi sama dengan
13. Pada baris 15 seleksi kondisi dengan menggunakan if antara variabel array *euc* dengan variabel *temp* dengan kondisi kurang dari
14. Pada baris 16 merupakan pemberian nilai pada variabel *temp* dengan nilai variabel array *euc*
15. Pada baris 17 merupakan pemberian nilai pada variabel *tetangga2* dengan nilai *tetangga1*
16. Pada baris 18 merupakan pemberian nilai pada variabel *tetangga1* dengan nilai dari variabel array *euc*
17. Pada baris 19 pemberian nilai 1 pada variabel *id1*
18. Pada baris 20 seleksi kondisi dengan menggunakan if antara variabel array *euc* dengan *tetangga2* serta variabel *tetangga1* dan *tetangga2*
19. Pada baris 21 pemberian nilai variabel *tetangga2* dengan nilai variabel array *euc*
20. Pada baris 22 pemberian nilai pada variabel *id2* dengan nilai variabel *i*
21. Pada baris 23 seleksi kondisi dengan menggunakan if antara variabel *k* dengan nilai 2 menggunakan kondisi samadengan
22. Pada baris 24 – 25 proses mencetak kedalam layar
23. Pada baris 26 proses pembagian antara variabel 1 dengan variable *tetangga1* di kuadratkan
24. Pada baris 27 proses pembagian antara variabel 1 dengan variabel *tetangga2* di kuadratkan
25. Pada baris 28 proses seleksi kondisi dengan menggunakan if antara variabel *t1* dengan *t2* menggunakan kondisi lebih besar dari

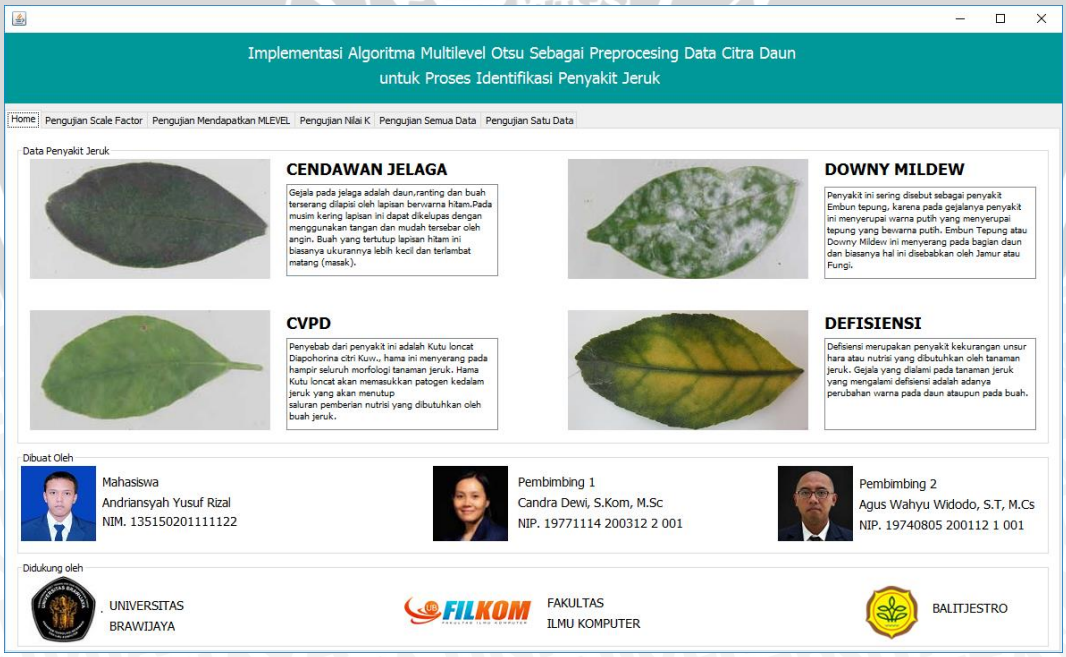
26. Pada baris 29 pemberian nilai pada variabel *hasil* dengan nilai dari pemanggilan method *getValueAt* dengan parameter dari variabel *id1* dengan 4 objek tabel
27. Pada baris 30 operasi else
28. Pada baris 31 pemberian nilai pada variabel *hasil* dengan nilai dari pemanggilan method *getValueAt* dengan parameter dari variabel *id2* dengan 4 objek tabel
29. Pada baris 32 proses pengembalian nilai dari variabel *hasil*

5.4 Implementasi Antarmuka

Implementasi antarmuka adalah menerapkan perancangan antarmuka yang telah dibuat pada sistem. Pada penelitian ini antarmuka yang diterapkan memiliki 7 halaman yang terdiri dari Halaman Awal, Halaman Pelatihan, Halaman Pengujian Data Uji, Halaman Pengujian Peubah Nilai *Scale Factor*, Halaman pengujian Peubah Nilai *MLEVEL*, Halaman pengujian Peubah Nilai *K*, Halaman About.

5.4.1 Implementasi Halaman Awal

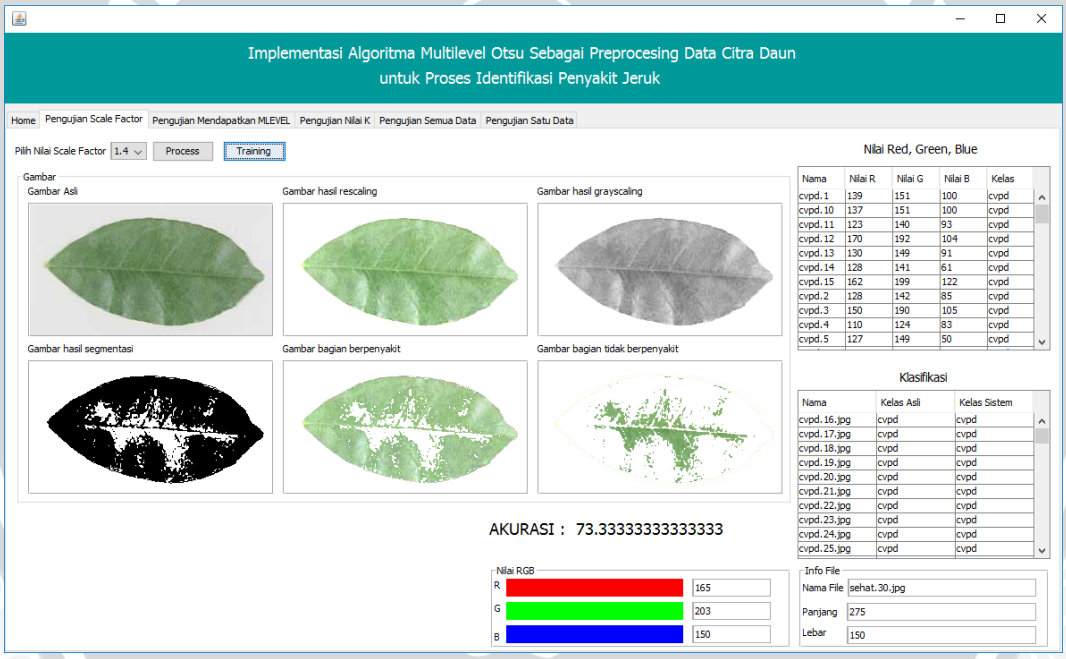
Implementasi halaman awal berisikan penjelasan penyakit yang ada pada jeruk yaitu penyakit *Downy Mildew*, Cendawan Jelaga, *Citrus Vein Phloem Degeneration* (CVPD), dan Defisiensi. Selain itu juga terdapat beberapa orang yang mengimplementasikan aplikasi dan instansi-instansi yang mendukung dalam pembuatan sistem. Implementasi Halaman Awal ditunjukkan pada Gambar 5.1.



Gambar 5.1 Implementasi Antarmuka Halaman Awal

5.4.2 Implementasi Halaman Pengujian *Scale Factor*

Halaman pelatihan *scale factor* merupakan halaman untuk mencari nilai *scale factor* dengan memperhatikan pemilihan nilai *scale factor* antara 1.1 hingga 2.0 untuk dicari nilai *scale factor* terbaik. Pada halaman ini terdapat *combo box* yang berisikan nilai *scale factor*. Pada halaman ini terdapat panel yang menampilkan citra asli, citra hasil *rescaling*, citra hasil *grayscaling*, citra hasil *multilevel otsu* . Selain itu juga terdapat tombol proses dan *training* dimana ketika tombol proses dijalankan, aplikasi memasukkan nilai *scale factor* yang dipilih dan dilakukan proses pengambilan bagian daun yang berpenyakit dan melakukan ekstraksi ciri. Tombol *training* mempunyai fungsi untuk mencari nilai akurasi dengan mdncari KNN. Semua proses dimasukkan kedalam tabel dimana terdapat 2 tabel, tabel pertama adalah tabel yang memasukkan nilai ekstraksi ciri pada daun pertama sedangkan tabel kedua adalah tabel yang berisikan hasil klasifikasi dari sistem dan juga kelas asli. *Output* dari halaman ini adalah nilai akurasi pada setiap nilai *scale factor*. Halaman pengujian untuk mendapatkan nilai *scale factor* terbaik ditunjukkan pada Gambar 5.2.

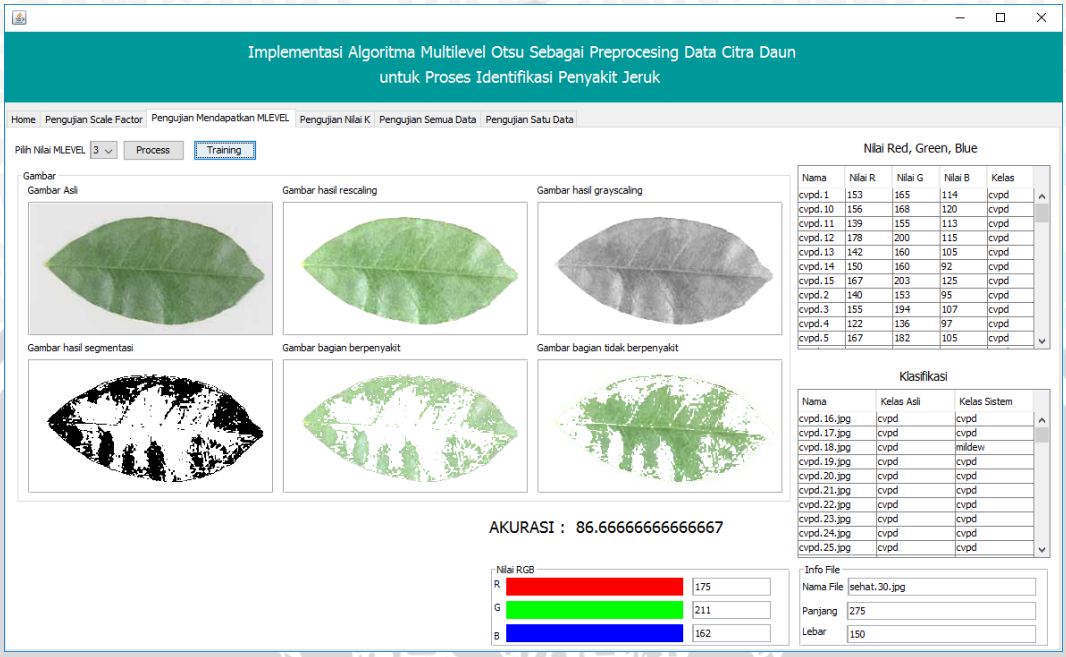


Gambar 5.2 Implementasi Antarmuka Halaman Pengujian *Scale Factor*

5.4.3 Implementasi Halaman Pengujian *MLEVEL*

Halaman pelatihan *MLEVEL* merupakan halaman untuk mencari nilai *MLEVEL* dengan memperhatikan pemilihan nilai *MLEVEL* antara 1 hingga 5 untuk dicari nilai *MLEVEL* terbaik. Pada halaman ini terdapat *combo box* yang berisikan nilai *MLEVEL*. Pada halaman ini terdapat panel yang menampilkan citra asli, citra hasil *rescaling*, citra hasil *grayscaling*, citra hasil *multilevel otsu* . Selain itu juga terdapat tombol proses dan *training* dimana ketika tombol proses dijalankan, aplikasi akan memasukkan nilai *MLEVEL* yang dipilih dan dilakukan proses pengambilan bagian daun yang berpenyakit dan melakukan ekstraksi ciri. Tombol *training* mempunyai

fungsi untuk mencari nilai akurasi dengan KNN. Semua proses akan dimasukkan kedalam tabel dimana terdapat 2 tabel, tabel pertama adalah tabel yang memasukkan nilai ekstraksi ciri pada daun pertama sedangkan tabel kedua adalah tabel yang berisikan hasil klasifikasi dari sistem dan juga kelas asli. *Output* dari halaman ini adalah nilai akurasi pada setiap nilai *MLEVEL*. Halaman pelatian untuk mendapatkan nilai *MLEVEL* terbaik ditunjukkan pada Gambar 5.3.

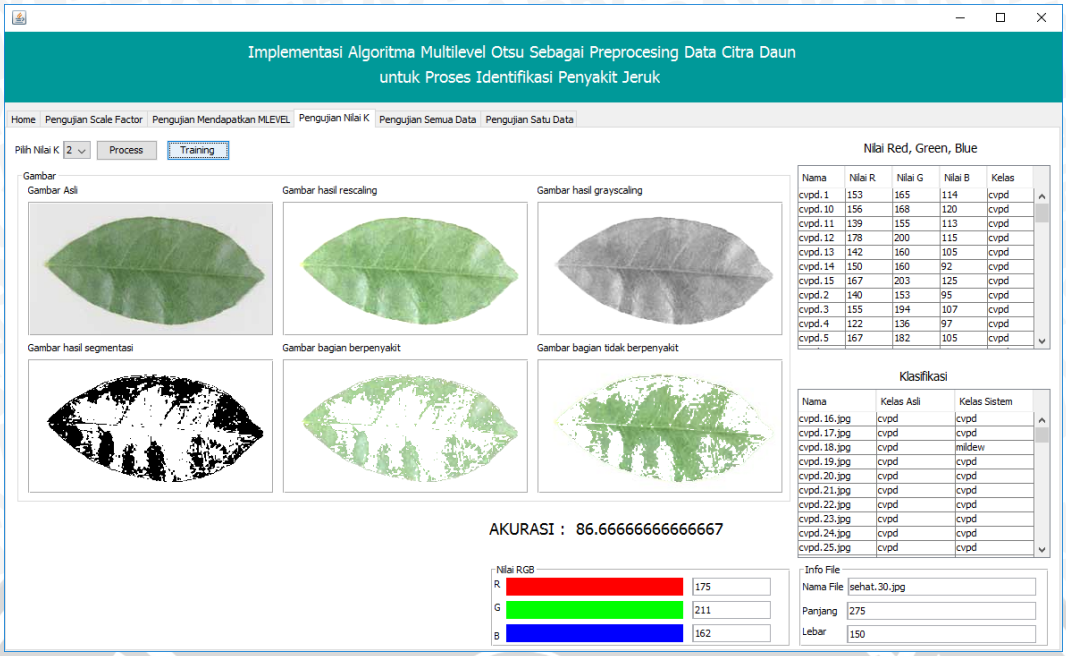


Gambar 5.3 Implementasi Antarmuka Halaman Pengujian *MLEVEL*

5.4.4 Implementasi Halaman Pengujian Nilai K

Halaman pelatihan *nilai K* merupakan halaman untuk mencari nilai *K* dengan memperhatikan pemilihan nilai *scale factor* antara 1 hingga 5 untuk dicari nilai *K* terbaik. Pada halaman ini terdapat *combo box* yang berisikan nilai *K*. Pada halaman ini terdapat panel yang menampilkan citra asli, citra hasil *rescaling*, citra hasil *grayscaling*, citra hasil *multilevel otsu* . Selain itu juga terdapat tombol proses dan *training* dimana ketika tombol proses dijalankan, aplikasi akan memasukkan nilai *K* yang dipilih dan dilakukan proses pengambilan bagian daun yang berpenyakit dan melakukan ekstraksi ciri. Tombol *training* mempunyai fungsi untuk mencari nilai akurasi dengan mencari KNN. Semua proses akan dimasukkan kedalam tabel dimana terdapat 2 tabel, tabel pertama adalah tabel yang memasukkan nilai ekstraksi ciri pada daun pertama sedangkan tabel kedua adalah tabel yang berisikan hasil klasifikasi dari sistem dan juga kelas asli. *Output* dari halaman ini adalah nilai akurasi pada setiap

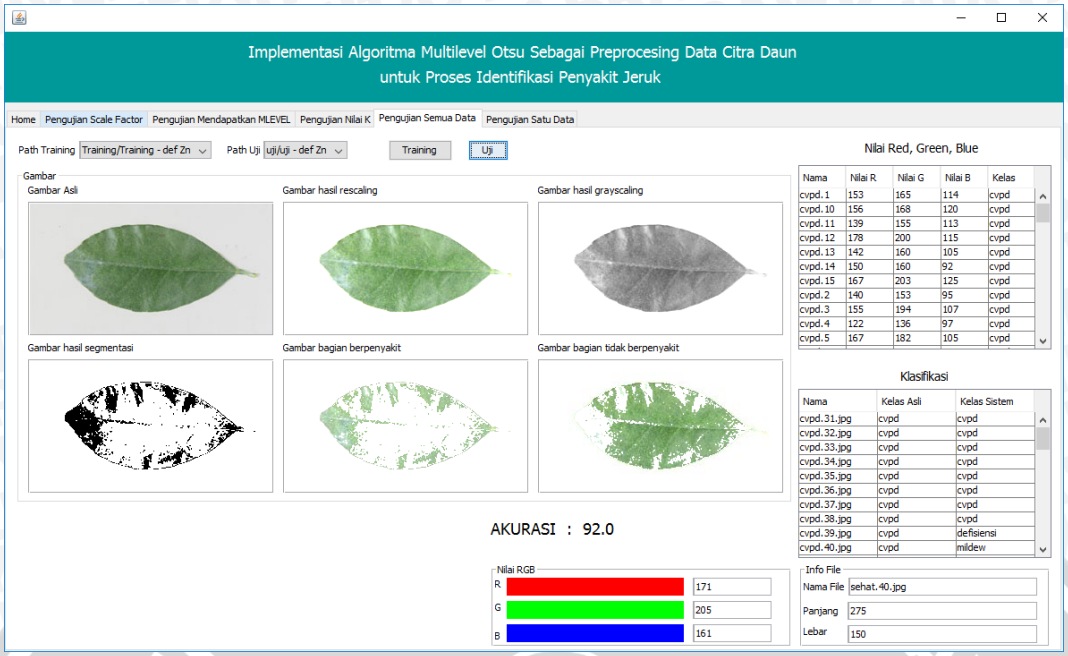
. Halaman pelatian untuk mendapatkan nilai *K* terbaik ditunjukkan pada Gambar 5.4.



Gambar 5.4 Implementasi Antarmuka Halaman Pengujian Nilai *K*

5.4.5 Implementasi Halaman Pengujian Semua Data

Halaman pengujian semua data merupakan pengujian terhadap semua data yang ada apda data uji, hal ini dilakukan untuk mencari akurasi dengan memperhatikan nilai parameter *scale factor, MLEVEL* dan nilai *K* terbaik pada proses sebelumnya. Langkah pertama yang dilakukan adalah pemilihan path untuk data latih dan data uji yang ditampilkan pada *combo box* setelah itu dilakukan proses *Training* pada tombol, langkah ini merupakan langkah memasukkan nilai ekstraksi ciri kedalam tabel. Setelah itu dilakukan proses uji dengan tombol dimana proses ini merupakan proses pencarian kelas yang dihasilkan oleh sistem dan di bandingkan dengan kelas asli untuk dicari nilai akurasinya. Semua proses pada halaman ini setiap citra hasil *rescaling, grayscaling,* segmentasi ditampilkan pada panel. Setelah proses semua di jalankan nilai akurasi disimpan dan ditampilkan pada label. Halaman pengujian semua data ditunjukkan pada Gambar 5.5.

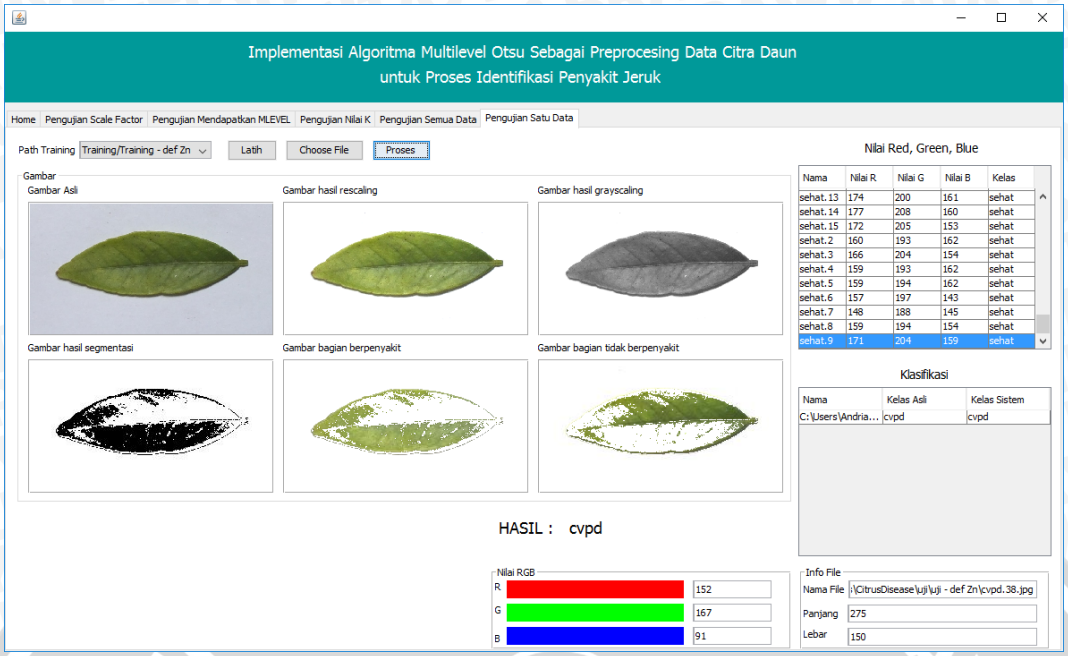


Gambar 5.5 Implementasi Antarmuka Halaman Pengujian Semua Data

5.4.6 Implementasi Halaman Pengujian Satu Data

Halaman pengujian satu data uji merupakan halaman yang digunakan untuk menguji data uji. Dalam halaman pengujian data uji data dilatih terlebih dahulu dengan mimilih path untuk melakukan pelatihan, parameter pada pelatihan dan pengujian ini merupakan parameter terbaik yang telah dicari pada peroses pencarian nilai *scale factor*, *MLEVEL, nilai K* terbaik. Pemilihan path pelatihan di tunjukkan pada *combo box.* Setelah itu terdapat tombol latih, dimana dilakukan ketika sudah memilih path pelatihan. Setelah melakukan pelatihan maka terdapat tombol *"Choose File"* yang berfungsi sebagai untuk memanggil kotak dialog *explorer* untuk mencari citra yang akan diuji. Setelah melakukan pemilihan *file* maka proses selanjutnya adalah melakukan pemrosesan dari data yang dipilih dengan data latih.

Setelah didapatkan citra hasil pemilihan file uji, proses selanjutnya adalah menampilkan citra asli pada panel dan juga memberikan informasi *file* berupa panjang, lebar dan nama *file*, selain itu juga ditampilkan informasi RGB pada file yang dipilih. Setelah itu citra akan di *rescaling* dan *grayscaling* dan ditampilkan pada panel. Setelah itu dilakukan *preprocessing* dengan *Multilevel Otsu* dan ditampilkan pada panel. Setelah itu dicari nilai rata-rata *red, green, blue* dan di identifikasi terhadap data latih. Hasil dari identifikasi ditampilkan pada *textbox* hasil identifikasi. Halaman pengujian satu data uji ditunjukkan pada Gambar 5.6.



Gambar 5.6 Implementasi Antarmuka Halaman Pengujian Satu Data

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini merupakan pembahasan tentang pengujian dan analisis sistem yang telah dibuat. Pengujian dilakukan untuk mencari nilai akurasi tertinggi dari sistem yang telah dibuat. Pada proses pengujian dilakukan beberapa skenario pada saat proses pelatihan dan uji data menggunakan metode KNN. Proses analisis dilakukan dengan menganalisa hasil pengujian yang telah dilakukan.

6.1 Skenario Pengujian dan Analisis

Sesuai dengan skenario pengujian yang telah dirancang pada bab 3, terdapat tiga skenario pengujian pada penelitian ini :

1. Pengujian pada nilai *scale actor*
2. Pengujian pada nilai *MLEVEL*
3. Pengujian pada nilai *K*

Disetiap pengujian dilakukan nilai yang berbeda-beda, pada pengujian nilai *scale factor* digunakan 10 nilai yaitu 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0. Pengujian dengan nilai *MLEVEL* dan nilai *K* digunakan nilai yang sama sebanyak 5 nilai yaitu 1, 2, 3, 4, 5. Pengujian pertama dilakukan pada pencarian nilai *scale factor* terbaik dengan nilai *MLEVEL* dan nilai *K* default yaitu 5 dan 2. Pada skenario pengujian kedua dilakukan pencarian nilai *MLEVEL* dengan cara mencari kombinasi nilai *MLEVEL*, nilai *scale factor* pada pengujian ketiga ini menggunakan hasil terbaik pada skenario pengujian sebelumnya. Skenario ketiga dilakukan pencarian nilai terbaik dari nilai *K* pada KNN. Pada skenario ini nilai *scale factor* dan nilai *MLEVEL* merupakan nilai terbaik pada proses sebelumnya. Proses pengujian untuk mencari nilai *Scale Factor*, *MLEVEL*, dan Nilai *K* terbaik akan menghasilkan masing-masing nilai terbaik yang nantinya akan diuji dengan data uji. Proses pengujian akhir dilakukan sebanyak 3 kali, Perbedaan pengujian akhir terletak pada tipe defisiensi yang berbeda dimana pada pengujian pertama digunakan defisiensi Zn dengan seluruh data, pengujian kedua defisien Mg dengan seluruh data dan pengujian ketiga defisiensi Mg dan Zn dengan seluruh penyakit yang ada.

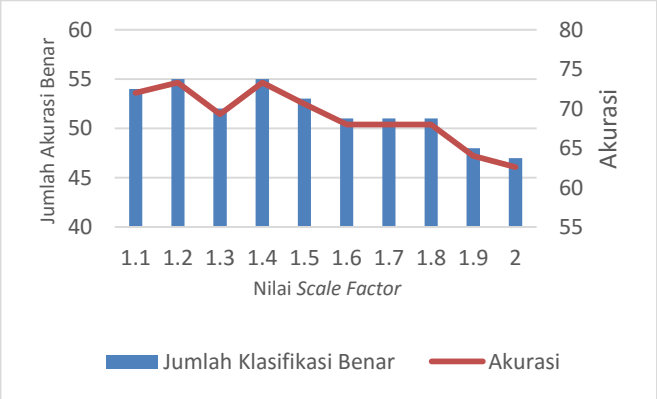
6.1.2 Pengujian dan Analisis Peubah Nilai *Scale Factor*

Pengujian terhadap nilai *scale factor* bertujuan untuk mendapatkan nilai *scale terbaik* untuk proses *preprocessing* awal. Nilai yang digunakan pada pengujian peubah nilai *scale factor* antara lain 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0. Pada pengujian ini nilai *MLEVEL* dan nilai *K* pada KNN masing-masing bernilai 5 dan 2. Data latih yang digunakan pada pengujian ini adalah 15 setiap kelas dengan total keseluruhan data latih 75 data, sedangkan jumlah data uji yang digunakan sama dengan data latih yaitu sebanyak 75 data. Perhitungan jarak dilakukan dengan menggunakan perhitungan *Euclidean Distance*. Hasil pengujian dari nilai *scale factor* ditunjukkan pada Tabel 6.1.

Tabel 6.1 Hasil Uji Coba Akurasi Terhadap Nilai *Scale Factor*

Nilai Scale Factor	Jumlah Klasifikasi Benar	Akurasi
1.1	54	72%
1.2	55	73.3%
1.3	52	69.3%
1.4	55	73.3%
1.5	53	70.6%
1.6	51	68%
1.7	51	68%
1.8	51	68%
1.9	48	64%
2.0	47	62.6%

Untuk memudahkan pembaca dalam mengetahui hasil analisa pada nilai peubah *scale factor*, maka dibuatlah representasi dalam bentuk grafik. Representasi grafik akurasi pengujian nilai *scale factor* ditunjukkan pada Gambar 6.1.



Gambar 6.1 Grafik Akurasi Pengujian Nilai *Scale Factor*

Pada tabel dan grafik akurasi tersebut diketahui bahwa nilai *Scale Factor* tertinggi didapatkan sebesar 73.3 dimana nilai *scale factor* yang didapat adalah 1.4 dan 1.2. Akurasi didapatkan berdasarkan jumlah data uji benar dibandingkan dengan total data uji yaitu 75, lalu dikalikan 100%. Maka dapat disimpulkan bahwa nilai *scale factor* 1.4 dan 1.2 adalah rekomendasi data *scale factor* terbaik. Dan data tersebut digunakan untuk kombinasi skenario pengujian selanjutnya.

Semakin tinggi nilai *scale factor* yang digunakan maka semakin cerah suatu citra yang digunakan. Sedangkan apabila nilai *scale factor* semakin rendah maka tingkar kecerahan suatu citra menjadi berkurang dan hal tersebut berpengaruh pada latar belakang citra. Pada pengujian tersebut pada nilai *scale factor* ke 1.2 dan 1.4 dapat

mengenali 55 citra. Jadi, dapat disimpulkan bahwa tingkat kecerahan suatu citra dapat mempengaruhi proses ekstraksi ciri.

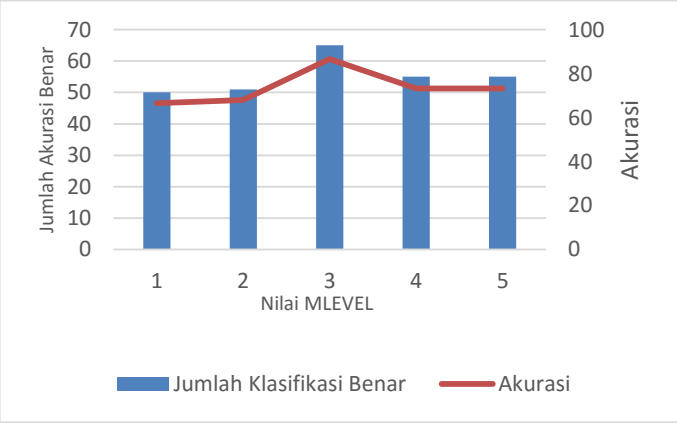
6.1.3 Pengujian dan Analisis Peubah Nilai *MLEVEL*

Pengujian terhadap nilai *MLEVEL* bertujuan untuk mendapatkan nilai *MLEVEL* terbaik pada saat proses *preprocessing* pencarian bagian penyakit daun. Nilai yang digunakan untuk pengujian diantaranya 1, 2, 3, 4, 5. Pada pengujian ini nilai *scale factor* yang digunakan merupakan perolehan nilai terbaik pada pengujian sebelumnya yaitu nilai 1.4 dan 1.2 dan nilai *K* yang digunakan merupakan nilai default yaitu 2. Pada pengujian ini di uji sebanyak 2 nilai *scale factor* yaitu 1.2 dan 1.4. Hal itu dilakukan untuk mencari kombinasi nilai *MLEVEL* dan nilai *scale factor* terbaik untuk menjadi acuan dari proses pengujian berikutnya. Hasil pengujian nilai *MLEVEL* dengan nilai *scale factor* 1.2 ditunjukkan pada Tabel 6.2.

Tabel 6.2 Hasil Uji Coba Akurasi Terhadap Nilai *MLEVEL* dan *Scale Factor* 1.2

Nilai *MLEVEL*	Jumlah Klasifikasi Benar	Akurasi
1	45	60%
2	52	69.3%
3	59	78.6%
4	55	73.3%
5	55	73.3%

MLEVEL dengan nilai *scale* factor 1.2 maka penulis membuat grafik perhitungan akurasi. Grafik hasil akurasi pengujian terhadap nilai *MLEVEL* ditunjukkan pada Gambar 6.2.



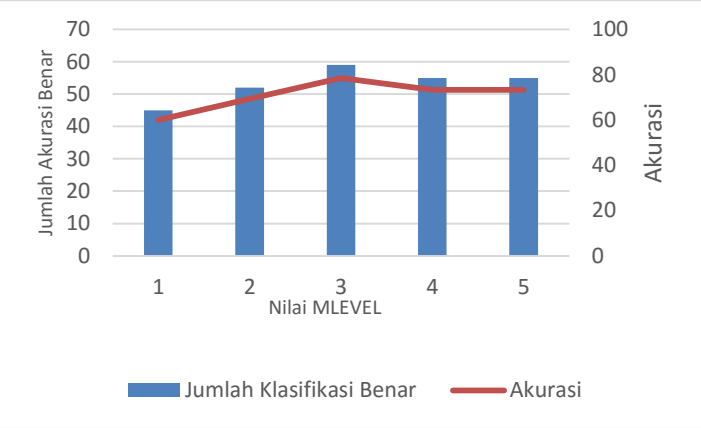
Gambar 6.2 Grafik Akurasi Pengujian Nilai *MLEVEL* dan *Scale Factor* 1.2.

Proses pengujian *MLEVEL* yang kedua adalah dengan menggunakan nilai *scala factor* sebesar 1.4 Hasil pengujian nilai *MLEVEL* dengan nilai *scale factor* 1.4 ditunjukkan pada Tabel 6.3. Untuk memudahkan pembaca untuk membaca analisa pengujian terhadap nilai

Tabel 6.3 Hasil Uji Coba Akurasi Terhadap Nilai *MLEVEL* dan *Scale Factor* 1.4

Nilai *MLEVEL*	Jumlah Klasifikasi Benar	Akurasi
1	50	66.6%
2	51	68%
3	65	86.6%
4	55	73.3%
5	55	73.3%

MLEVEL dengan nilai *scale* factor 1.4 maka penulis membuat grafik perhitungan akurasi. Grafik hasil akurasi pengujian terhadap nilai *MLEVEL* ditunjukkan pada Gambar 6.3.



Gambar 6.3 Grafik Akurasi Pengujian Nilai *MLEVEL* dan *Scale Factor* 1.4.

Pada tabel dan grafik akurasi tersebut di ketahui bahwa akurasi terbesar nilai *MLEVEL* pada tabel 6.2 tertinggi adalah 3 dengan nilai *scale factor* sebesar 1.2 sedangkan pada tabel 6.3 didapatkan nilai *MLEVEL* tertinggi adalah 3 dengan nilai *scale factor* sebesar 1.4. Akurasi yang dimiliki pada nilai *MLEVEL* = 3 dan nilai *scale factor* = 1.4 memiliki nilai terbesar yaitu 86.6% sehingga dapat disimpulkan bahwa nilai rekomendasi nilai *MLEVEL* dan *scale factor* yang menjadi pengujian selanjutnya adalah sebesar 3 dan 1.4.

MLEVEL mempengaruhi terhadap bagian citra daun yang nantinya melewati proses ekstraksi ciri. Nilai *MLEVEL* yang kecil berpengaruh pada proses pengenalan bagian penyakit pada daun. Hal itu dikarenakan nilai *MLEVEL* merupakan penentu banyaknya *threshold* yang akan digunakan. Jika nilai *MLEVEL* kecil maka proses penentuan bagian berpenyakit pada citra daun menjadi kurang sempurna. Pada pengujian tersebut didapatkan bahwa nilai *scale factor* 1.4 yang berpengaruh besar terhadap akurasi penggunaan nilai *MLEVEL*. Pada nilai *MLEVEL* 3 dan nilai *scale* factor sebesar 1.4 dapat mengenali 65 daun. Maka dapat disimpulkan bahwa nilai *MLEVEL* berpengaruh pada proses pengenalan penyakit jeruk pada tahap *preprocessing*.

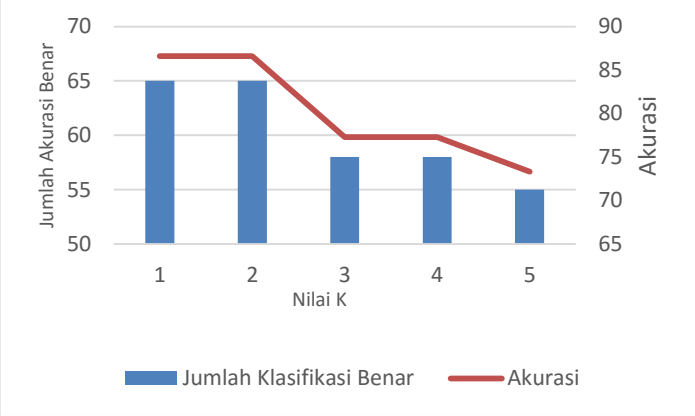
6.1.4 Pengujian dan Analisis Peubah Nilai *K*

Pengujian terhadap nilai *K* bertujuan untuk mendapatkan nilai pencarian tetangga terbaik pada proses klasifikasi menggunakan KNN. Nilai *K* yang digunakan pada pengujian ini adalah 1, 2, 3, 4, 5. Pada pengujian nilai *K* digunakan nilai *scale factor* dan nilai *MLEVEL* terbaik pada pengujian sebelumnya yaitu 1.4 dan 5. Hasil pengujian nilai *K* ditunjukkan pada Tabel 6.4.

Tabel 6.4 Hasil Uji Coba Akurasi Terhadap Nilai *K*

Nilai *MLEVEL*	Jumlah Klasifikasi Benar	Akurasi
1	65	86.6
2	65	86.6
3	58	77.3
4	58	77.3
5	55	73.3

Untuk memudahkan pembaca dalam menganalisa maka penulis membuat grafik akurasi pengujian terhadap nilai *K*. Grafik hasil akurasi pengujian terhadap nilai *K* ditunjukkan pada Gambar 6.4.



Gambar 6.4 Grafik Akurasi Pengujian Nilai *K*.

Dari pengujian diatas maka dilihat bahwa akurasi tertinggi yang didapatkan dari pengujian nilai *K* adalah sebesar 86.6% yaitu pada nilai *K* sebesar 1 dan 2. Pengujian nilai *K* digunakan nilai rekomendasi terbaik pada nilai *scale factor* dan nilai *MLEVEL*.

Pengujian terakhir didapatkan nilai *K* sebesar 2 dimana bila kita melihat grafik pada Gambar 6.4 nilai *K* yang semakin besar mengakibatkan akurasi yang didapatkan menjadi semakin kecil, hal itu terjadi karena pengambilan tetangga menjadi lebih banyak dan lebih variatif. Maka dapat disimpulkan bahwa semakin banyak nilai *K* maka semakin banyak klasifikasi yang dihasilkan.

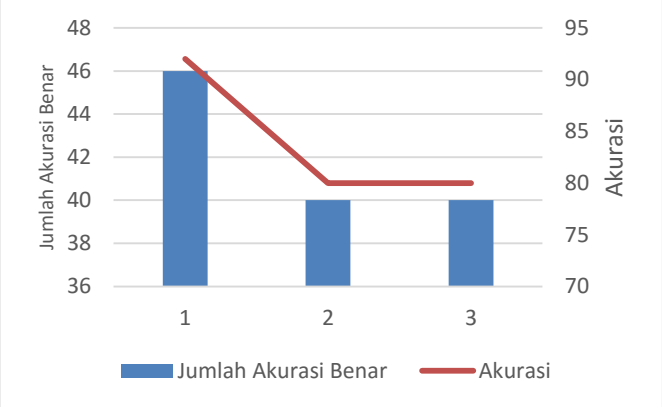
6.1.5 Pengujian dan Analisis Terhadap Seluruh Data

Pengujian terakhir adalah pengujian terhadap seluruh data uji yang ada. Data uji yang digunakan sebanyak 3 tipe, tipe yang membedakan antar data uji terletak pada daun yang memiliki penyakit defisiensi. Defisiensi yang dipakai sebanyak 2 yaitu defisiensi Zn dan Mg. Jumlah data uji yang digunakan adalah sebanyak 10 data pada setiap kelas, sehingga jumlah data uji adalah sebesar 50 data. Pada pengujian akhir ini nilai *scale factor*, *MLEVEL* dan nilai *K* yang digunakan merupakan nilai rekomendasi terbaik. Berikut hasil pengujian akhir dapat ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian Akhir

Data	Jumlah Klasifikasi Benar	Akurasi
Defisiensi Zn	46	92%
Defisiensi Mg	40	80%
Defisiensi Zn dan Mg	40	80%

Untuk memudahkan pembaca dalam menganalisa maka penulis membuat grafik pengujian akhir. Angka 1 pada grafik merepresentasikan pengujian dengan defisiensi Zn, CVPD, Jelaga, mildew, sehat sedangkan angka 2 defisiensi Mg, CVPD, jelaga, Mildew, sehat dan angka 3 merepresentasikan defisiensi Mg dan Zn, CVPD, Jelaga, Mildew, sehat. Grafik hasil akurasi pengujian akhir ditunjukkan pada Gambar 6.5.



Gambar 6.5 Grafik Akurasi Pengujian Akhir.

Pada grafik diatas ditunjukkan bahwa data latih dan data uji defisiensi Zn, CVPD, Jelaga, Mildew, sehat memiliki grafik tertinggi di bandingkan yang lain yaitu sebesar 92% atau berhasil mengenali 46 penyakit dari 50 penyakit yang ada.

BAB 7 PENUTUP

Pada bab ini diapaparkan kesimpulan dan saran yang dapat diambil berdasarkan pengujian yang telah dilakukan terhadap implementasi algoritma multilevel otsu sebagai *preprocessing* data citra daun untuk proses identifikasi penyakit jeruk.

7.1 Kesimpulan

Berikut kesimpulan yang dapat diambil dari hasil yang telah didapatkan dari perancangan, implementasi dan pengujian yang telah dilakukan.

1. Metode *multilevel thresholding* menggunakan *otsu* sebagai *preprocessing* untuk mengidentifikasi penyakit jeruk melewati beberapa tahap yaitu *rescaling, grayscaling*, pencarian nilai *threshold* dan proses konvolusi. Tahap *rescaling* merupakan tahap untuk mencapatkan citra menjadi cerah dan memisahkan bagian daun dengan latar belakang. Proses pencarian rata-rata RGB dicari sebagai tahap *grayscaling*. Pencarian nilai *threshold* merupakan proses penentuan nilai threshold yang dihasilkan berdasarkan nilai level pada proses *Multilevel Thresholding* yang dimasukkan. Tahap akhir yang dilakukan adalah penjumlahan citra antara citra segmentasi dengan citra asli untuk mendapatkan bagian penyakit yang dicari.
2. Berdasarkan pengujian yang telah dilakukan dilakukan yaitu pada peubah nilai konstanta pada proses rescaling, nilai level pada proses *Multilevel Thresholding*, dan nilai ketetanggan pada KNN. Sistem mendapatkan akurasi tertinggi yaitu 86.6%. Untuk nilai *scale factor* didapatkan rekomendasi nilai 1.4. Untuk nilai level pada proses *Multilevel Thresholding* didapatkan rekomendasi nilai 3 dan Untuk nilai ketetanggan pada KNN didapatkan rekomendasi nilai 1 dan 2. Pada tahap akhir pengujian didapatkan didapatkan akurasi trebaik pada pengenalan defisiensi Zn, CVPD, Jelaga, Mildew dan daun sehat sebesar 92%.

7.2 Saran

Peneliti menganggap bahwa penelitian ini masih jauh dari sempurna, oleh karena itu berikut beberapa saran yang dapat peneliti berikan untuk pengembangan penelitian selanjutnya.

1. Perlu dilakukan penelitian lebih lanjut penambahan kelas penyakit selain *Downy Mildew*, Cendawan Jealga, CVPD, defisiensi seperti kanker daun jeruk, kudis, *Woody Gall.*
2. Perlu dilakukan penelitian lebih lanjut untuk penjabaran identifikasi penyakit defisiensi diantaranya defisiensi Mg, Zn, K, C, dll.
3. Perlu dilakukan penelitian lebih lanjut untuk ekstraksi fitur, misal ditambahkan dengan fitur *texture* seperti *mean, variance, skewness, kurtosis, entropy* agar dapat membedakan tekstur tiap daun.

4. Perlu dilakukan penelitian lebih lanjut untuk penentuan latar belakang citra selain bewarna putih seperti pewarnaan yang random, agar petani yang ingin melakukan identifikasi penyakit tidak perlu menyiapkan latar belakang bewarna putih.
5. Perlu dilakukan penelitian lebih lanjut pada sistem agar dapat melakukan identifikasi pada lebih dari satu daun dalam satu *frame* citra.
6. Perlu dilakukan penelitian lebih lanjut pada sistem agar dapat melakukan identifikasi lebih dari satu penyakit jika pada daun memiliki lebih dari satu macam penyakit.

DAFTAR PUSTAKA

Balitjestro, 2015. *Mengenal Penyakit CVPD (Huanglongbing).* [Online] Available at: http://balitjestro.litbang.pertanian.go.id/penyakit-cvpd-huanglongbing-pada-tanaman-jeruk/ [Diakses 5 November 2016].

Dwiastuti, M. E., Agustina, D. & Unun, T., 2016. *Teknik Identifikasi Penyakit CVPD dan Defisiensi Unsur Hara Secara Visual, Cepat dan Sederhana.* Batu: s.n.

Hanif, Z. & Zamzami, L., 2012. *Trend Jeruk Impor dan Posisi Indonesia sebagai Produsen Jeruk Dunia.* [Online] Available at: https://balitjestro.litbang.pertanian.go.id/trend-jeruk-impor-dan-posisi-indonesia-sebagai-produsen-jeruk-dunia/ [Diakses 1 Februari 2017].

Hardiyanto, R., 2009. *Ayo Bangkitlah Buah Indonesia dalam Hortikultura.* [Online] Available at: https://balitjestro.litbang.pertanian.go.id/mampukah-jeruk-keprok-nasional-kita-menggeser-jeruk-impor/ [Diakses 10 September 2016].

Huang, D.-Y., Lin, T.-W. & Hu, W.-C., 2011. Automatic Multilevel Thresholding Based on Two-Stage Otsu's Method With Cluster Determination By Valley Estimation. *International Journal of Innovative Computing, Information and Control.*

Kanan, C. & Cottrell, G. W., 2012. Color-to-Grayscale: Does the Method Matter in Image Recognition?. *PLOS ONE,* 7(e29740).

Kanisius, A. A., 1994. *Google Books.* [Online] Available at: https://books.google.co.id/books?id=c2s9IMQSrlgC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false [Diakses 7 Februari 1].

Knudsen, J., 1999. *Java 2D Graphics.* California: O'Reilly Media.

Kumaseh, M. R., Latumakulita, L. & Nainggolan, N., 2013. Segmentasi Citra Digital Ikan Menggunakn Metode Thresholding. *Jurnal Ilmiah Sains,* Volume 13, pp. 75-79.

Liao, P.-S., Chen, T.-S. & Chung, P.-C., 2001. A Fast Algorithm for Multilevel Thresholding. *Journal of Information Science Engineering,* Volume 10, pp. 713-727.

Lin, Z. & Yu, H., 2011. The Pupil Location Based on the OTSU Method and Hough Transform. *Procedia Environmental Sciences,* Volume 8, pp. 352-356.

Mandyartha, E. P. & Fatichah, C., 2016. Three-level Local Thresholding Berbasis Metode Otsu untuk Segmentasi Leukosit pada Citra Leukimia Limfoblastik Akut. *Jurnal Buana Informatika,* Volume 7, pp. 43-54.

Muflikhah, L., Rahmawati, D. E. & Putri, R. R. M., 2006. *Data Mining.* Malang: UB Press.

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

Poerwanto, R., 2004. *Ayo Bangkitlah Buah Indonesia dalam Hortikultura.* s.l.:s.n.

Riadi, A. & Pramunendar, A., 2014. Pengenalan Plat Nomor Kendaraan Ganjil Genap Menggunakan Metode Support Vector Machine (SVM) Berbasis Varians Projection. *Jurnal Teknologi Informasi,* Volume 10, pp. 168-185.

Saxena, K., Khan, Z. & Singh, S., 2014. Diagnosis of Diabetes Mellitus using K Nearest Neighbor Algorithm. *International Journal of Computer Science Trends and Technologi (IJST),* 2(4), pp. 36-43.

Semangun, H., 2004. *Penyakit-penyakit Tanaman Hortikultura di Indonesia.* Yogyakarta: Gadjah Mada University Press.

Sutoyo, T. et al., 2009. *Teori Pengolahan Citra Digital.* Yogyakata: Penerbit Andi.

Syafi'i, S. I., Wahyuningrum, R. T. & Muntasa, A., 2015. Segmentasi Obyek pada Citra Digital Menggunakan Metode Otsu Thresholding. *Jurnal Informatika,* Volume 13, pp. 1-8.

Triwiratno, A., 2016. *Penyakit Jeruk dan Cara Mengidentifikasi* [Wawancara] (12 Juli 2016).

Umam, M. S., Dewi, C. & Cholissodin, I., 2015. *Implementasi Metode Learning Vector Quantizaion (LVQ) untuk Identifikasi Penyakit Pada Citra Daun Tanaman Kedelai,* Malang: s.n.

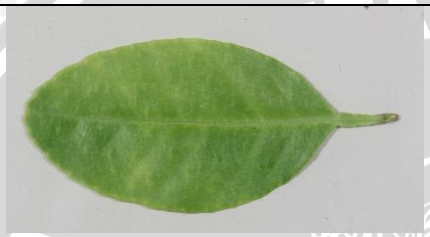
Wibowo, S. A. & Usman, K., 2010. Voice Activity Detection G729B Improvement Technique Using K-Nearest Neigbor Method. *International Conference on Distributed Frameworks For Multimedia Applications (DFmA).*

Zhou, H., Wu, J. & Zhang, J., 2010. *Digital Image Processing : Part I.* s.l.:Ventus Publishing Aps.

LAMPIRAN

A.1 Daftar Citra Daun Untuk Data Uji

No	Citra Daun Jeruk	Nama File	Penyakit Daun/Sehat
1.		Cvpd.31.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
2.		Cvpd.32.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
3.		Cvpd.33.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
4.		Cvpd.34.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
5.		Cvpd.35.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
6.		Cvpd.36.jpg	CVPD *(Citrus Vein Phloem Degeneration)*

7.		Cvpd.37.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
8.		Cvpd.38.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
9.		Cvpd.39.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
10.		Cvpd.40.jpg	CVPD *(Citrus Vein Phloem Degeneration)*
11.		jelaga.31,jpg	Cendawan Jelaga
12.		jelaga.32,jpg	Cendawan Jelaga
13.		jelaga.33,jpg	Cendawan Jelaga

14.		jelaga.34,jpg	Cendawan Jelaga
15.		jelaga.35,jpg	Cendawan Jelaga
16.		jelaga.36,jpg	Cendawan Jelaga
17.		jelaga.37,jpg	Cendawan Jelaga
18.		jelaga.38,jpg	Cendawan Jelaga
19.		jelaga.39,jpg	Cendawan Jelaga
20.		jelaga.40,jpg	Cendawan Jelaga

21.		mildew.31.jpg	*Mildew*
22.		mildew.32.jpg	*Mildew*
23.		mildew.33.jpg	*Mildew*
24.		mildew.34.jpg	*Mildew*
25.		mildew.35.jpg	*Mildew*
26.		mildew.36.jpg	*Mildew*
27.		mildew.37.jpg	*Mildew*

28.		mildew.38.jpg	*Mildew*
29.		mildew.39.jpg	*Mildew*
30.		mildew.40.jpg	*Mildew*
31.		sehat.31.jpg	Sehat
32.		sehat.32.jpg	Sehat
33.		sehat.33.jpg	Sehat
34.		sehat.34.jpg	Sehat

35.		sehat.35.jpg	Sehat
36.		sehat.36.jpg	Sehat
37.		sehat.37.jpg	Sehat
38.		sehat.38.jpg	Sehat
39.		sehat.39.jpg	Sehat
40.		sehat.40.jpg	Sehat
41.		defisiensi.31.jpg	Defisiensi Zn

42.		defisiensi.32.jpg	Defisiensi Zn
43.		defisiensi.33.jpg	Defisiensi Zn
44.		defisiensi.34.jpg	Defisiensi Zn
45.		defisiensi.35.jpg	Defisiensi Zn
46.		defisiensi.36.jpg	Defisiensi Zn
47.		defisiensi.37.jpg	Defisiensi Zn
48.		defisiensi.38.jpg	Defisiensi Zn

49.		defisiensi.39.jpg	Defisiensi Zn
50.		defisiensi.40.jpg	Defisiensi Zn
51.		defisiensi.31.jpg	Defisiensi Mg
52.		defisiensi.32.jpg	Defisiensi Mg
53.		defisiensi.33.jpg	Defisiensi Mg
54.		defisiensi.34.jpg	Defisiensi Mg
55.		defisiensi.35.jpg	Defisiensi Mg

56.		defisiensi.36.jpg	Defisiensi Mg
57.		defisiensi.37.jpg	Defisiensi Mg
58.		defisiensi.38.jpg	Defisiensi Mg
59.		defisiensi.39.jpg	Defisiensi Mg
60.		defisiensi.40.jpg	Defisiensi Mg