KLASIFIKASI JENIS MAKANAN DARI CITRA SMARTPHONE BERDASARKAN EKSTRAKSI FITUR HARALICK DAN CIE LAB COLOR MOMENT MENGGUNAKAN LEARNING VECTOR QUANTIZATION

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Akhmad Muzanni Safi'i
NIM: 155150200111270

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI JENIS MAKANAN DARI CITRA SMARTPHONE BERDASARKAN EKSTRAKSI FITUR HARALICK DAN CIE LAB COLOR MOMENT MENGGUNAKAN LEARNING VECTOR QUANTIZATION

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Akhmad Muzanni Safi'i
NIM: 155150200111270

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Yuita Arum Sari, S.Kom, M.Kom	Sigit Adinugroho, S.Kom, M.Sc
NIK: 201609 880715 2 001	NIK: 201607 880701 1 000

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, 2 Januari 2019

Akhmad Muzanni Safi'i

NIM: 155150200111270

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT karena hanya dengan rahmat, nikmat, taufiq serta hidayah-Nya, penulis telah berhasil menyelesaikan Laporan Skripsi dengan judul "Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization".

Penulis menyadari bahwa penyusunan skripsi yang diajukan untuk memenuhi sebagian persyaratan untuk memperoleh gelar Sarjana Komputer (S.Kom) ini tidak akan terwujud tanpa adanya bantuan dan dorongan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT, Tuhan Yang Maha Esa, yang telah memberikan perlindungan dan kemudahan selama penyusunan laporan skripsi ini.
2. Bapak, Ibu, seluruh keluarga, dan seluruh teman atas segenap dukungan dan kasih sayang yang telah diberikan.
3. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika.
5. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer.
6. Ibu Yuita Arum Sari, S.Kom., M.Kom., dan Bapak Sigit Adinugroho, S.Kom., M.Sc, selaku dosen pembimbing yang telah membimbing dan memberikan masukan serta kritikan dengan sangat baik.
7. Rekan-rekan PANGLIMA (Empat Negeri Lima Menara), yang senantiasa memberikan dukungan dan masukan selama pengerjaan skripsi ini.
8. Siti Nahdiyatul Ulya yang selalu memberikan dukungan, semangat dan doa kepada penulis.
9. Segenap civitas akademik Fakultas Ilmu Komputer Universitas Brawijaya.
10. Seluruh pihak yang telah membantu kelancaran Kuliah Kerja Nyata Praktik yang tidak dapat kami sebutkan satu-persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih memiliki banyak kekurangan Oleh karena itu penulis mengharapkan kritik dan saran dari para pembaca yang bersifat membangun. Teriring doa semoga penelitian ini dapat bermanfaat dan memberikan informasi bagi kita semua.

Malang, 2 Januari 2019

Penulis
akhmadmuzannisafii@gmail.com

ABSTRAK

Pemilihan makanan menjadi sesuatu yang penting bagi para penderita penyakit tertentu. Akan tetapi, memilih sebuah makanan menjadi sebuah masalah bagi orang yang pertama kali mencicipi suatu makanan atau wisatawan yang baru pertama kali berkunjung ke suatu negara. Untuk mengatasi permasalahan tersebut, perlu dilakukan penelitian untuk mengenali/mengklasifikasikan suatu citra makanan. Fitur Haralick dan CIE Lab Color Moments terbukti dapat menghasilkan fitur yang bagus untuk kasus klasifikasi. Metode Learning Vector Quantization juga menjadi alternatif untuk melakukan proses klasifikasi.

Berdasarkan pengujian k-fold cross validation dengan k=10 dan metode evaluasi berupa akurasi, didapatkan akurasi maksimal sebesar 0.642051 dengan nilai parameter learning rate sebesar 0.2, pengali learning rate sebesar 0.8, nilai m sebesar 0.1, nilai epsilon sebesar 0.4, iterasi maksimal sebesar 10 dan learning rate minimal sebesar 0.000001. Hal tersebut menunjukkan bahwa klasifikasi citra makanan berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization menghasilkan akurasi yang cukup baik. Selain itu, penggunaan kedua fitur tekstur (Haralick) dan warna (CIE Lab Color Moments) berpengaruh terhadap hasil akurasi. Hal tersebut ditunjukkan dengan seluruh hasil pengujian yang menunjukkan bahwa hasil akurasi tertinggi dicapai menggunakan fitur tekstur dan warna.

Kata kunci: klasifikasi, *Haralick, CIE Lab Color Moments, Learning Vector Quantization, cross-validation.*

ABSTRACT

Choosing a food becomes something important for sufferers of certain diseases. However, choosing a food is a problem for people who taste a food for first time or tourists who are visiting a country for first time. To overcome these problems, research needs to be done to identify / classify a food image. The Haralick and CIE Lab Color Moments features are proven to produce good features for classification cases. The Learning Vector Quantization method is also an alternative for classification process.

Based on the k-fold cross validation with k = 10 and accuracy as evaluation method, the maximum accuracy is 0.642051 with learning rate parameter value is 0.2, the learning rate multiplier is 0.8, the m value is 0.1, the epsilon value is 0.4, maximum iteration is 10 and minimum learning rate is 0.000001. This result shows that food image classification based on Haralick feature extraction and CIE Lab Color Moment using Learning Vector Quantization produces fairly good accuracy. In addition, the use of both texture (Haralick) and color features (CIE Lab Color Moments) has an effect on the results of accuracy. This is indicated by all the test results which show that the highest accuracy results are achieved using texture and color features.

Keywords: *classification*, *Haralick, CIE Lab Color Moments, Learning Vector Quantization, cross-validation.*

DAFTAR ISI

KLASIFIKASI JENIS MAKANAN DARI CITRA SMARTPHONE BERDASARKAN EKSTRAKSI FITUR HARALICK DAN CIE LAB COLOR MOMENT MENGGUNAKAN LEARNING VECTOR QUANTIZATION i

PENGESAHAN ii

PERNYATAAN ORISINALITAS iii

KATA PENGANTAR iv

ABSTRAK vi

ABSTRACT vii

DAFTAR ISI viii

DAFTAR TABEL xi

DAFTAR GAMBAR xiii

DAFTAR LAMPIRAN xv

BAB 1 PENDAHULUAN 1

1.1 Latar belakang 1

1.2 Rumusan masalah 2

1.3 Tujuan 3

1.4 Manfaat 3

1.5 Batasan masalah 3

1.6 Sistematika pembahasan 4

BAB 2 LANDASAN KEPUSTAKAAN 5

2.1 Kajian Pustaka 5

2.2 Normalisasi 11

2.3 Preprosesing 12

2.3.1 Transformasi Warna *RGB to Grayscale* 13

2.3.2 Segmentasi 14

2.4 Ekstraksi Fitur 15

2.4.1 Haralick Features 15

2.4.2 CIE L*a*b Color Moment 18

2.5 Klasifikasi 20

2.6 Jaringan Syaraf Tiruan 20

2.6.1 Learning Vector Quantization 20

2.7 Pengujian 23
BAB 3 METODOLOGI 25
3.1 Jenis Penelitian 25
3.2 Strategi dan Rancangan Penelitian 25
3.2.1 Metode secara umum 25
3.2.2 Lokasi Penelitian 26
3.2.3 Metode Pengumpulan Data 26
3.2.4 Peralatan Pendukung 29
BAB 4 PERANCANGAN algoritme 30
4.1 Klasifikasi 30
4.1.1 Inisialisasi Data Latih dan Data Uji 30
4.1.2 Preprosesing 31
4.1.3 Ekstraksi Fitur Tekstur (Haralick) 44
4.1.4 Ekstraksi Fitur Warna (Momen Warna) 53
4.1.5 Normalisasi 56
4.1.6 Klasifikasi LVQ 59
4.2 Perancangan Antarmuka 68
4.2.1 Perancangan Antarmuka Halaman Beranda 68
4.2.2 Perancangan Antarmuka Section Klasifikasi 68
4.2.3 Perancangan Antarmuka Modal Screen Nilai Fitur 69
4.3 Skenario Pengujian 70
4.3.1 Skenario Pengujian Learning Rate 71
4.3.2 Skenario Pengujian Pengali Learning Rate 72
4.3.3 Skenario Pengujian Nilai m 72
4.3.4 Skenario Pengujian Epsilon 72
4.3.5 Skenario Pengujian Iterasi Maksimal 73
4.3.6 Skenario Pengujian Learning Rate Minimal 73
4.3.7 Skenario Pengujian Akurasi 73
BAB 5 IMPLEMENTASI 74
5.1 Implementasi Sistem 74
5.1.1 Implementasi Proses Inisialisasi Data Latih dan Data Uji 74
5.1.2 Implementasi Proses Preprosesing 75

5.1.3 Implementasi Proses Ekstraksi Fitur Tekstur 80
5.1.4 Implementasi Proses Ekstraksi Fitur Warna 85
5.1.5 Implementasi Proses Normalisasi 87
5.1.6 Implementasi Proses Klasifikasi LVQ 88
5.2 Implementasi Antarmuka 91
5.2.1 Antarmuka Halaman Beranda 92
5.2.2 Antarmuka Section Klasifikasi 92
5.2.3 Antarmuka Modal Screen Nilai Fitur 93
BAB 6 PENGUJIAN DAN ANALISIS 94
6.1 Hasil dan Analisis Pengujian Learning Rate 95
6.2 Hasil dan Analisis Pengujian Pengali Learning Rate 96
6.3 Hasil dan Analisis Pengujian Nilai m 97
6.4 Hasil dan Analisis Pengujian Epsilon 99
6.5 Hasil dan Analisis Pengujian Iterasi Maksimal 100
6.6 Hasil dan Analisis Pengujian Learning Rate Minimal 101
6.7 Hasil dan Analisis Pengujian Akurasi 102
BAB 7 PENUTUP 107
7.1 Kesimpulan 107
7.2 Saran 107
DAFTAR PUSTAKA 108
LAMPIRAN A KELAS DATA 111
A.1 Kelas Data 111
LAMPIRAN B HASIL PENGUJIAN 114
B.1 Hasil Pengujian k-Fold Cross Validation 114
B.2 Jarak antara masing-masing kelas 115

DAFTAR TABEL

Tabel 2.1 Daftar Kajian Pustaka 7
Tabel 3.1 Jenis Smartphone yang digunakan beserta jumlah citra yang diambil . 27
Tabel 3.2 Dataset berdasarkan masing-masing kelas 27
Tabel 3.3 Perangkat Keras Pendukung..... 29
Tabel 3.4 Perangkat Lunak Pendukung..... 29
Tabel 4.1 Sampel nilai intensitas citra RGB (Red/Green/Blue)..... 34
Tabel 4.2 Sampel nilai intensitas citra *grayscale* 34
Tabel 4.3 Sampel hasil normalisasi nilai intensitas citra RGB (Red/Green/Blue) . 37
Tabel 4.4 Sampel hasil konversi nilai RGB ke XYZ (X/Y/Z)..... 38
Tabel 4.5 Sampel hasil perhitungan nilai intensitas LAB (L/A/B)..... 39
Tabel 4.6 Sampel hasil denormalisasi nilai LAB 39
Tabel 4.7 Sampel nilai variabel segmen..... 43
Tabel 4.8 Sampel nilai variabel segmen2..... 43
Tabel 4.9 Contoh sebaran nilai intensitas citra..... 46
Tabel 4.10 Contoh Matriks Co-occurence..... 48
Tabel 4.11 Contoh Matriks Probabilitas..... 49
Tabel 4.12 Contoh Citra LAB 55
Tabel 4.13 Contoh Data Fitur..... 58
Tabel 4.14 Data Nilai Maksimal Per Fitur..... 58
Tabel 4.15 Data Nilai Minimal Per Fitur 58
Tabel 4.16 Data Fitur Setelah Dinormalisasi 59
Tabel 4.17 *Reference vector* beserta kelasnya..... 61
Tabel 4.18 Data Latih beserta kelasnya 62
Tabel 4.19 Jarak Euclidean antara data latih dan reference vector 62
Tabel 4.20 Reference Vector terdekat..... 63
Tabel 4.21 Bobot Hasil Pelatihan LVQ..... 66
Tabel 4.22 Data Uji Ternormalisasi 66
Tabel 4.23 Jarak Euclidean antara Data Uji dan Refence Vector..... 67
Tabel 4.24 Citra dan kelas Hasil Pengujian 67
Tabel 4.25 Kondisi Awal Parameter 71
Tabel 4.26 Skenario Pengujian Learning Rate..... 71

Tabel 4.27 Skenario Pengujian Pengali Learning Rate 72
Tabel 4.28 Skenario Pengujian Nilai m 72
Tabel 4.29 Skenario Pengujian Epsilon 72
Tabel 4.30 Skenario Pengujian Iterasi Maksimal 73
Tabel 4.31 Skenario Pengujian Learning Rate Minimal 73
Tabel 6.1 Ilustrasi Pengujian Hasil Klasifikasi 94
Tabel 6.2 Hasil Pengujian Learning Rate 95
Tabel 6.3 Hasil Pengujian Pengali Learning Rate 96
Tabel 6.4 Hasil Pengujian Nilai m 98
Tabel 6.5 Hasil Pengujian Epsilon 99
Tabel 6.6 Hasil Pengujian Iterasi Maksimal 100
Tabel 6.7 Hasil Pengujian Learning Rate Minimal 102
Tabel 6.8 Hasil Pengujian Akurasi 103
Tabel 6.9 Nilai Standar Deviasi untuk fitur tekstur 104
Tabel 6.10 Nilai Standar Deviasi untuk fitur warna 106

DAFTAR GAMBAR

Gambar 2.1 Citra dengan kontras yang berbeda: (a) citra dengan kontras rendah (b) citra dengan kontras seimbang. 12

Gambar 2.2 Citra dengan perbedaan intensitas derau: (a) citra berderau (b) citra tanpa derau. 12

Gambar 2.3 Proses deteksi tepi: (a) citra sebelum dilakukan deteksi tepi (b) citra setelah dilakukan deteksi tepi..... 13

Gambar 2.4 Proses transformasi warna: (a) citra pada ruang warna RGB (b) citra *grayscale.* 13

Gambar 2.5 Proses segmentasi: (a) citra sebelum dilakukan segmentasi (b) citra setelah dilakukan segmentasi. 14

Gambar 2.6 Orientasi fitur GLCM 16

Gambar 2.7 Arsitektur Jaringan LVQ..... 21

Gambar 2.8 Ilustrasi Metode k-Fold Cross Validation 23

Gambar 3.1 Metode Penelitian secara umum..... 26

Gambar 4.1 Tahapan Perancangan Sistem 30

Gambar 4.2 Sampel Data Latih 31

Gambar 4.3 Sampel Data Uji..... 32

Gambar 4.4 Diagram Alir Transfomasi Warna RGB to *Grayscale* 33

Gambar 4.5 Citra RGB: (a) citra dengan tiga *channel* RGB, (b) citra *channel* Red, (c) citra *channel* Green dan (d) citra *channel* Blue..... 33

Gambar 4.6 Citra *grayscale* hasil transformasi..... 35

Gambar 4.7 Diagram Alir Transformasi Warna RGB *to* LAB..... 36

Gambar 4.8 Diagram Alir Proses Segmentasi 42

Gambar 4.9 Variabel Segmen: (a) Citra segmen (b) Citra segmen2 (c) Hasil operasi bitwise OR antara segmen dan segmen2..... 43

Gambar 4.10 Hasil Akhir Segmentasi..... 44

Gambar 4.11 Diagram Alir Proses Ekstraksi Fitur Tekstur 45

Gambar 4.12 Diagram Alir Proses Perhitungan Matriks Co-occurence..... 47

Gambar 4.13 Diagram Alir Proses Perhitungan Probability Matrix 49

Gambar 4.14 Diagram Alir Proses Perhitungan Fitur Tekstur..... 50

Gambar 4.15 Diagram Alir Proses Ekstraksi Fitur Warna..... 54

Gambar 4.16 Diagram Alir Proses Normalisasi 57

Gambar 4.17 Diagram Alir Proses Pelatihan LVQ 60

Gambar 4.18 Diagram Alir Pengujian LVQ 65
Gambar 4.19 Perancangan Antarmuka Beranda 69
Gambar 4.20 Perancangan Antarmuka Section Klasifikasi untuk Proses Unggah Citra dan Segmentasi 69
Gambar 4.21 Perancangan Antarmuka Section Klasifikasi untuk Proses Ekstraksi Fitur dan Klasifikasi 70
Gambar 4.22 Perancangan Antarmuka Nilai Fitur 70
Gambar 5.1 Tampilan Halaman Beranda 92
Gambar 5.2 Tampilan Section Klasifikasi untuk Proses Unggah Citra dan Segmentasi 92
Gambar 5.3 Tampilan Section Klasifikasi untuk Proses Ekstraksi Fitur dan Klasifikasi 93
Gambar 5.4 Tampilan Modal Screen Nilai Fitur 93
Gambar 6.1 Grafik Pengujian Learning Rate 96
Gambar 6.2 Grafik Pengujian Pengali Learning Rate 97
Gambar 6.3 Grafik Pengujian Nilai m 98
Gambar 6.4 Grafik Pengujian Epsilon 100
Gambar 6.5 Grafik Pengujian Iterasi Maksimal 101
Gambar 6.6 Grafik Pengujian Learning Rate Minimal 102
Gambar 6.7 Contoh Citra (a) Kelas Mie Gepeng dan (b) Kelas *Fried Chicken* 104
Gambar 6.8 Contoh Citra (a) Kelas Pisang Kuning dan (b) Kelas *Happy Tos* 104

DAFTAR LAMPIRAN

LAMPIRAN A KELAS DATA 111
A.1 Kelas Data 111
LAMPIRAN B HASIL PENGUJIAN 114
B.1 Hasil Pengujian k-Fold Cross Validation 114
B.2 Jarak antara masing-masing kelas 115

BAB 1 PENDAHULUAN

1.1 Latar belakang

Pemilihan makanan menjadi sesuatu yang penting bagi para penderita penyakit tertentu. Sebagai contoh, mengonsumsi makanan tinggi purin dapat meningkatkan risiko penyakit *gout artritis*/asam urat hingga 4,1 kali lebih besar (Tumenggung, 2015). Para penderita Penyakit Jantung Koroner (PJK) juga diharapkan untuk memperhatikan pola makan dengan mengurangi konsumsi makanan asin dan mengingkatkan konsumsi buah dan sayur (Zuroida, 2015). Para penderita diabetes juga diharapkan untuk mengonsumsi makanan yang memiliki kadar gula terkontrol untuk menghindari ketidakstabilan gula darah (Susanti & Bistara, 2018).

Selain beberapa penyakit yang tersebut, obesitas juga menjadi masalah bagi beberapa orang. Pada tahun 2016, lebih dari 1,9 miliar orang mengalami kelebihan berat badan dan 650 juta orang diantaranya mengalami obesitas (World Health Organization, 2018). Obesitas merupakan salah satu penyakit yang dapat dicegah dengan berbagai cara. Salah satunya adalah melakukan pencatatan dan pemilihan konsumsi kalori sehari-hari serta melakukan *dietary assesment* (Liu, et al., 2016).

Memilih sebuah makanan mungkin tidak menjadi masalah bagi orang yang telah mengenali makanan tersebut. Berbeda dengan orang yang pertama kali mencicipi suatu makanan atau wisatawan yang sedang berkunjung ke suatu negara dan mendapati sebuah makanan baru yang belum diketahui bahan dan kadar gizinya. Oleh karena itu, diperlukan suatu sistem yang dapat mengenali jenis makanan dengan memanfaatkan teknologi yang ada saat ini (Joutou & Yanai, 2009).

Penelitian (Joutou & Yanai, 2009) mencoba mengenali citra lima puluh makanan yang didapatkan dari web. Metode yang digunakan adalah *Multiple Kernel Learning – Support Vector Machine* menggunakan kombinasi fitur dengan nilai bobot sub-*kernel* yang berbeda. Nilai evaluasi untuk penelitian ini sebesar 61,34% menggunakan metode evaluasi *cross validation*.

Penelitian mengenai klasifikasi makanan juga dilakukan pada penelitian (Chen, et al., 2012), yang berusaha untuk mengidentifikasi makanan Cina melalui citra ponsel dan menghitung kuantitasnya. Penelitian ini menggunakan perpaduan beberapa ekstraksi fitur seperti *Bag of Features, Local Binary Patterns, Color Histogram dan Gabor Texture*. Metode klasifikasi yang digunakan pada penelitian ini adalah *Support Vector Machine* yang dipadukan dengan algoritme *Adaboost* menghasilkan nilai akurasi sebesar 90,9% (Chen, et al., 2012).

Untuk melakukan klasifikasi citra makanan, diperlukan ekstraksi fitur yang tepat untuk mendapatkan hasil klasifikasi terbaik. Untuk mendapatkan fitur tekstur terdapat beberapa pilihan metode ekstraksi ciri tekstur, antara lain *Gabor Filter, Gaussian Filter, Haralick Features* dan lain-lain (Bagalkote & Vibhute, 2016). Sebagai contoh, penelitian (Patil, et al., 2011) mencoba untuk melakukan

klasifikasi pada citra biji-bijian menggunakan histogram CIE Lab dan 5 fitur *Haralick*. Metode klasifikasi yang digunakan adalah *K-Nearest Neighbours* dengan variasi nilai *K* sebesar 1, 3, 5 dan 7 serta menggunakan metode penghitungan jarak Canberra. Hasil yang didapatkan penelitian ini adalah perpaduan CIE Lab dan Haralick menghasilkan akurasi maksimal sebesar 94,5% dengan nilai *K* sama dengan lima. Akan tetapi penggunaan metode KNN dinilai memiliki beberapa kekurangan, antara lain komputasi yang kompleks, keterbatasan memori yang dimiliki peneliti dan mudah tertipu dengan atribut yang tidak relevan (Mutrofin, et al., 2014). Oleh karena itu, diperlukan metode klasifikasi yang lain sebagai alternatif pilihan yang lebih baik.

Beberapa metode metode Jaringan Syaraf Tiruan sering digunakan untuk menyelesaikan kasus klasifikasi. Salah satunya adalah *Learning Vector Quantization*. Penelitian (Nayef, et al., 2013) menunjukkan beberapa metode Jaringan Syaraf Tiruan seperti *Learning Vector Quantization* (LVQ), *Multi Perceptron NN* (MLP), *Radial Basis Function Networks* (RBF) dan *Self-Organize Map* (SOM). Penelitian tersebut berusaha untuk mengklasifikasikan citra *Magnetic Resonance Imaging* (MRI) otak dengan melakukan pengacakan data sebelum dilakukan pelatihan. Hasil dari penelitian tersebut menunjukkan bahwa metode LVQ cukup handal untuk digunakan pada berbagai kondisi data. Hal tersebut dibuktikan melalui Uji T dengan *p-values* 0,15 yang lebih besar dari standar signifikansi (0,005).

Berdasarkan permasalahan dan berbagai penelitian yang telah dipaparkan di atas, penulis tertarik untuk mengambil topik klasifikasi jenis makanan dari citra *smartphone* Berdasarkan Ekstraksi Fitur *Haralick* dan *CIE Lab Color Moment* Menggunakan *Learning Vector Quantization* dengan harapan dapat mempermudah proses klasifikasi citra makanan melalui *smartphone* di kalangan masyarakat.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas, maka dapat dirumuskan permasalahan pada penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization, antara lain:

1. Bagaimana hasil pengujian evaluasi klasifikasi jenis makanan dari citra *smartphone* berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization?
2. Bagaimana pengaruh ekstraksi fitur terhadap hasil evaluasi klasifikasi jenis makanan dari citra *smartphone* berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization?

1.3 Tujuan

Tujuan dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization, antara lain:

1. Mengetahui hasil pengujian evaluasi klasifikasi jenis makanan dari citra *smartphone* berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization.
2. Mengetahui pengaruh ekstraksi fitur terhadap hasil evaluasi klasifikasi jenis makanan dari citra *smartphone* berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization.

1.4 Manfaat

Manfaat yang diharapkan dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization, antara lain:

Bagi peneliti:

1. Sebagai sarana untuk mengaplikasikan ilmu yang telah didapatkan selama masa perkuliahan.
2. Mengetahui perkembangan dari bidang ilmu yang telah ditekuni selama masa perkuliahan.

Bagi penulis:

1. Sebagai sarana pembelajaran untuk penerapan metode yang digunakan pada studi kasus klasifikasi citra makanan.
2. Mengetahui efektivitas dan efisiensi dari metode Learning Vector Quantization.
3. Sebagai referensi dalam penelitian-penelitian di masa yang akan datang.

Bagi masyarakat:

1. Melakukan klasifikasi makanan yang belum diketahui melalui citra smartphone.

1.5 Batasan masalah

Penelitian ini dibatasi dengan beberapa hal, antara lain:

1. Data yang digunakan adalah dataset milik Tim Food Project Fakultas Ilmu Komputer Universitas Brawijaya yang diambil menggunakan beberapa jenis smartphone.
2. Dataset citra yang digunakan pada penelitian ini terbatas pada 31 kategori. Masing-masing kelas terdiri dari 76 citra.
3. Algoritme Learning Vector Quantization yang digunakan adalah algoritma LVQ3.

1.6 Sistematika pembahasan

Sistematika pembahasan pada penelitian ini tersusun sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas tentang kajian pustaka yang mendasari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 3 METODOLOGI

Bab ini membahas tentang tahapan-tahapan yang dilakukan pada penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 4 PERANCANGAN ALGORITME

Bab ini membahas tentang perancangan manualisasi perhitungan, perancangan antarmuka serta perancangan uji coba dan evaluasi dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 5 IMPLEMENTASI

Bab ini membahas tentang hal teknik mengenai implementasi dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas hasil pengujian dan analisis terhadap hasil pengujian dari penelitian Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization.

BAB 7 PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari penelitian ini yang disertai dengan saran untuk dapat dipergunakan pada penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi pembahasan mengenai kajian pustaka dan dasar teori dari objek dan metode yang berkaitan dengan penelitian ini.

2.1 Kajian Pustaka

Penelitian ini didasarkan pada studi literatur beberapa penelitian yang pernah dilakukan sebelumnya berkaitan dengan objek dan metode yang akan digunakan oleh penulis, yaitu Klasifikasi Makanan, Haralick Feature, CIE Lab Color Moments dan Learning Vector Quantization. Beberapa penelitian mengenai objek dan metode tersebut selengkapnya terangkum pada Tabel 2.1.

Penelitian pertama mencoba mengenali citra lima puluh makanan yang didapatkan dari web. Metode yang digunakan adalah *Multiple Kernel Learning – Support Vector Machine* menggunakan kombinasi fitur dengan nilai bobot sub-*kernel* yang berbeda. Nilai evaluasi untuk penelitian ini sebesar 61,34% menggunakan metode evaluasi *cross validation* (Joutou & Yanai, 2009).

Penelitian kedua mencoba untuk mengidentifikasi makanan Cina melalui citra ponsel dan menghitung kuantitasnya. Penelitian ini menggunakan perpaduan beberapa ekstraksi fitur seperti *Bag of Features, Local Binary Patterns, Color Histogram dan Gabor Texture*. Metode klasifikasi yang digunakan pada penelitian ini adalah *Support Vector Machine* yang dipadukan dengan algoritme *Adaboost* menghasilkan nilai akurasi sebesar 90,9% (Chen, et al., 2012).

Penelitian ketiga berusaha untuk melakukan klasifikasi pada dataset UECFood100 yang terbagi menjadi seratus kategori makanan. Seratus kategori makanan tersebut selanjutnya dikelompokkan kembali menjadi tiga belas grup makanan. Ekstraksi fitur yang digunakan adalah RootHOG dan Color Histogram yang dikombinasikan dengan Fisher Vector. Metode klasifikasi yang digunakan adalah Adaptive-SVM dengan Sample Selection. Hasil yang didapatkan dari penelitian ini menunjukkan bahwa metode Adaptive-SVM dan pemilihan dataset pelatihan sebelum training (Sample Selection) mendapatkan nilai tertinggi sebesar 91,36% menggunakan metode evaluasi *Precision@300* (Kawano & Yanai, 2014).

Penelitian keempat berusaha untuk melakukan klasifikasi pada citra beberapa dataset citra makanan. Metode klasifikasi yang digunakan adalah Jaringan Syaraf Tiruan dengan arsitektur GoogLeNet 22 layer. Sebelum dilakukan pelatihan, peneliti menambahkan bounding box pada dataset yang akan digunakan. Hasil pada penelitian ini menunjukkan bahwa dataset UEC-100 dapat mencapai akurasi tertinggi sebesar 94,5% menggunakan metode pengujian Top-5 Accuracy (Liu, et al., 2016).

Penelitian kelima mencoba untuk melakukan klasifikasi pada 60.000 makanan India yang terbagi menjadi sepuluh kategori. Metode klasifikasi yang digunakan adalah Convolutional Neural Network menggunakan lima convolutional layer. Fitur yang digunakan adalah masing-masing nilai intensitas piksel pada citra yang

telah dikonversi menjadi citra keabuan. Akurasi pengujian yang didapatkan pada penelitian ini mencapai 95% pada satu kali epoch (Jahan, et al., 2018).

Penelitian keenam mencoba untuk melakukan klasifikasi 10 kategori citra biji-bijian menggunakan metode ekstraksi citra lima fitur Haralick dan standar deviasi serta mean dari intensitas pada ruang warna L*a*b*, HSV, HSI dan YCbCr. Hasil dari penelitian tersebut menunjukkan bahwa menggunakan perpaduan fitur *Haralick* dan ciri statistik dari ruang warna dapat menghasilkan akurasi tertinggi sebesar 94,5% dengan proporsi data latih 12,5% dan K sama dengan lima (Patil, et al., 2011).

Penelitian ketujuh mencoba untuk melakukan klasifikasi pada citra multispektral LANDSAT dengan membandingkan metode Latent Dirichlet Allocation (LDA) pada ruang warna CIE Lab dan Normalized Difference Vegetation Index. Penelitian ini menunjukkan metode LDA-CIE Lab menghasilkan akurasi yang lebih tinggi dibandingkan NDVI dengan selisih 1,18% (Kusumaningrum & Arymurthy, 2015).

Penelitian kedelapan mencoba untuk melakukan klasifikasi enam jenis kulit pohon menggunakan 10 fitur Haralick dari citra LBP pada enam ruang warna. Metode klasifikasi yang digunakan adalan K-Nearest Neighbour dengan nilai K sam dengan 7 yang menghasilkan akurasi sebesar 85,6% (Porebski, et al., 2008).

Penelitian kesembilan berusaha untuk mengklasifikasikan citra *Magnetic Resonance Imaging* (MRI) otak dengan melakukan pengacakan data sebelum dilakukan pelatihan. Hasil dari penelitian tersebut menunjukkan bahwa metode LVQ cukup handal untuk digunakan pada berbagai kondisi data. Hal tersebut dibuktikan melalui Uji T dengan *p-values* 0,15 yang lebih besar dari standar signifikansi (0,005) (Nayef, et al., 2013).

Penelitian kesepuluh mencoba untuk melakukan klasifikasi pada 3 jenis bunga Adenium. Metode klasifikasi yang digunakan adalah LVQ dengan *learning rate* 0,01 dan *epoch* maksimal sebesar 10. Penelian ini menghasilkan akurasi maksimal sebesar 86,66% (Wulanningrum & Robby, 2016).

Berdasarkan beberapa penelitian di atas, maka penulis mengajukan penelitian dengan judul "Klasifikasi Jenis Makanan dari Citra Smartphone Berdasarkan Ekstraksi Fitur Haralick dan CIE Lab Color Moment Menggunakan Learning Vector Quantization". Sistem klasifikasi ini menggunakan Haralick Feature dan CIE Lab Color Moment sebagai metode ekstraksi fitur dan Learning Vector Quantization sebagai metode klasifikasi. Metode-metode tersebut dipilih berdasarkan kajian yang dilakukan pada beberapa pustaka penelitian dan terbukti telah menghasilkan hasil yang baik pada berbagai penelitian tersebut.

Tabel 2.1 Daftar Kajian Pustaka

No	Judul Pustaka	Input	Proses		Hasil
		Objek	Ekstraksi Fitur	Klasifikasi	
1	A Food Image Recognition System With Multiple Kernel Learning (Joutou & Yanai, 2009).	Menggunakan data image pribadi dengan 100 image untuk 50 kategori makanan yang didapatkan dari web	Menggunakan beberapa fitur, antara lain 6 vektor Bag of Feature (BoF) yang diekstrak dari Scale Invariant Feature Transform (SIFT), Color RGB Histogram dan fitur tekstur Gabor	Menggunakan Multiple Kernel Learning - Support Vector Machine (MKL-SVM) dengan menginisialisasi 49 kategori sebagai himpunan image postif dan satu kategori sebagai himpunan image negatif. Beberapa sub-kernel disiapkan untuk masing-masing fitur lalu digabungkan menjadi satu kernel kombinasi.	Akurasi yang didapatkan dengan melakukan kombinasi bobot fitur MKL adalah 61,34% dengan metode cross validation. Sebagai pengujian dilakukan pengambilan citra langsung oleh pengguna pada 166 makanan dan didapatkan 37.55%
2	Automatic Chinese Food Identification and Quantity Estimation (Chen, et al., 2012).	Menggunakan citra makanan Cina sebanyak 100 citra untuk masing - masing 50 kategori makanan. Dataset tersebut dibagi menjadi 5 bagian, satu bagian sebagai data uji dan sisanya sebagai data latih	Menggunakan perpaduan beberapa fitur, yaitu BoF SIFT, Local Binary Patterns (LBP), color histogram dan Gabor Texture.	Menggunakan metode SVM dengan algoritma Adaboost untuk melakukan pembobotan pada masing-masing SVM classifier.	Akurasi tertinggi sebesar 90.9% diperoleh menggunakan seluruh fitur yang diajukan dengan metode SVM-Multi Class Adaboost pada Top-5 accuracy. Selain itu, peneliti juga melakukan percobaan untuk melakukan beberapa pengaturan pada fitur SIFT dan LBP untuk menghasilkan akurasi yang terbaik. Sehingga didapatkan akurasi 53% untuk fitur

No	Judul Pustaka	Input	Proses		Hasil
		Objek	Ekstraksi Fitur	Klasifikasi	
					BoF Sparse coding dengan histogram warna, dan akurasi 45.9% untuk LBP Sparse Coding dengan histogram warna berdimensi 2048.
3	Automatic Expansion of a Food Image Dataset Leveraging Existing Categories with Domain Adaptation (Kawano & Yanai, 2014).	Menggunakan dataset UEC-Food100 sejumlah 14361 citra yang terbagi dalam 100 kategori makanan. Dari 100 makanan tersebut, peneliti mengelompokkan kembali jenis makanan tersebut menjadi 13 grup makanan	Menggunakan HOG 8 arah berdimensi 32 dan Color Local Patches berdimensi 24. Fitur-fitur tersebut kemudian diolah kembali menggunakan metode PCA.	Menggunakan metode Adaptive-SVM dengan Sample Selection.	Menggunakan metode evaluasi Precision@300, didapatkan nilai tertinggi sebesar 91.36% menggunakan Feature Classifier, Adaptive-SVM dan Sample Selection untuk masakan Thailand.
4	DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment (Liu, et al., 2016).	Menggunakan beberapa dataset, antara lain UEC-100(8643 image pada 100 kelas), UEC-256(28375 image pada 256 kelas) dan Food-101 (101 kelas, setiap kelas terdapat 1000 image)	Dilakukan preprocessing dengan menambahkan bounding box pada dataset UEC	Menggunakan arsitektur GoogLeNet dengan 22 layer, dengan rata-rata ukuran filter 5x5, 1x1 convolutional layer dan operasi ReLU	Akurasi tertinggi didapatkan pada dataset UEC-100 dengan top-5 accuracy sebesar 94.8%
5	Bird's Eye Review on Food Image Classification using Supervised Machine Learning (Jahan, et al., 2018).	Menggunakan dataset pribadi untuk makanan India sebanyak 60.000 image yang terbagi menjadi 10 kelas. 50000 diantaranya digunakan	Data image yang telah dilakukan preprocessing terlebih dahulu sehingga masing-masing image menjadi image grayscale	Convolutional Neural Network dengan 5 convolutional layer. Pada setiap layer terdapat 10 neuron dengan operasi ReLu dan kernel 5x5	Akurasi testing yang didapatkan adalah 95% pada satu kali epoch

No	Judul Pustaka	Input	Proses		Hasil
		Objek	Ekstraksi Fitur	Klasifikasi	
		sebagai data latih dan sisanya sebagai data uji.	dengan ukuran 280x280 pixel		
6	Color and Texture Based Identification and Classification of food Grains using different Color Models and Haralick features (Patil, et al., 2011).	Menggunakan data citra 10 kategori biji-bijian yang dibagi menjadi data latih dan data uji dengan beberapa variasi perbandingan	Menggunakan ekstraksi fitur Haralick (Homogenitas, kontras, korelasi, entropi, homogenitas lokal) dan standar deviasi serta mean pada ruang warna L*a*b*, HSV, HSI dan YCbCr.	Menggunakan K-Nearest Neighbour dengan variasi nilai k = 1,3,5 dan 7 dan canberra distance	Menggunakan ruang warna L*a*b* dapat menghasilkan akurasi maksimal sebesar 94,5% dengan proporsi data latih 12,5% dan K=5
7	CIELab Color Moments: Alternative Descriptors for LANDSAT Images Classification System (Kusumaningrum & Arymurthy, 2015).	Mengunakan citra multispectral LANDSAT sebanyak 60 citra yang terbagi menjadi 30 citra lahan vegetasi dan 30 citra lahan non-vegetasi	Menggunakan fitur Bag of Visual Words dengan ruang warna CIE Lab	Menggunakan metode Latent Dirichlet Allocation dan Normalized Difference Vegetation Index.	Metode LDA pada momen warna CIE-Lab menghasilkan akurasi yang lebih tinggi dari NDVI dengan nilai 87,43%
8	Haralick feature extraction from LBP images for color texture classification (Porebski, et al., 2008).	Menggunakan citra dataset BarkTex untuk enam kategori kulit pohon sebanyak 408 citra	Menggunakan 10 haralick feature untuk citra LBP pada enam ruang warna	Menggunakan metode KNN	Menghasilkan akurasi sebesar 85,6% menggunakan K = 7
9	Brain Imaging Classification Based On Learning Vector Quantization (Nayef, et al., 2013).	Menggunakan data citra Magnetic Resonance Imaging (MRI) otak. Preprosesing menggunakan kernel 3x3 high pass dan 3x3 median filter.	Menggunakan GLCM sebanyak 21 fitur kemudian direduksi menjadi 8 fitur menggunakan Principal Component Analysis	Menggunakan beberapa metode seperti LVQ, MLP, RBF dan SOM	penelitian ini melakukan multi-randomization pada data sebelum melakukan training. Menggunakan uji T ditemukan bahwa p-value untuk LVQ mencapai nilai 0.15.

No	Judul Pustaka	Input	Proses		Hasil
		Objek	Ekstraksi Fitur	Klasifikasi	
					Dikarenakan p-value yang lebih dari 0.05 menunjukkan bahwa tidak ada perbedaan yang signifikan ketika dilakukan pengacakan data sebelum dilakukan pelatihan
10	Learning Vector Quantization Image for Identification Adenium (Wulanningrum & Robby, 2016).	Menggunakan data 3 jenis adenium sebanyak 45 citra	Menggunakan nilai intensitas dari hasil segmentasi langsung	Menggunakan metode LVQ dengan learning rate 0,01 dan epoch maksimal 10	Menghasilkan akurasi maksimal dengan nilai 86,66%

2.2 Normalisasi

Normalisasi merupakan salah satu tahapan preprosesing yang bertujuan untuk menyesuaikan kondisi data berada pada suatu jangkauan tertentu. Normalisasi biasa digunakan pada data yang memiliki jarak yang cukup jauh. Proses normalisasi banyak digunakan pada beberapa kasus prediksi dan peramalan sehingga dibutuhkan data yang memiliki jarak relatif dekat antara satu data dengan data yang lain.

Salah satu metode normalisasi yang biasa digunakan adalah Max-Min Normalization. Max-min normalization merupakan teknik yang bertujuan untuk melakukan transformasi data secara linear pada suatu jangkauan tertentu. Pada kasus data berbentuk citra, metode *max-min normalization* dapat dilakukan menggunakan Persamaan 2.1.

$$\boldsymbol{I_{norm(i,j)} = \frac{I_{(i,j)} - min_j}{max_j - min_j}} \quad \textbf{(2.1)}$$

Keterangan:

$I_{(i,j)}$ = Nilai fitur ke-j pada citra ke-i sebelum normalisasi

$I_{norm(i,j)}$ = Nilai fitur ke-j pada citra ke-i setelah normalisasi

min_j = Nilai minimal dari fitur ke-j

max_j = Nilai maksimal dari fitur ke-j

Pada operasi transformasi warna RGB to LAB, diperlukan tahap normalisasi nilai intensitas citra RGB. Proses normalisasi tersebut dapat menggunakan metode max-min normalization dengan nilai maksimal sebesar 255 dan nilai minimal adalah 0. Persamaan 2.1 untuk proses normalisasi RGB dapat disederhanakan menjadi Persamaan 2.2.

$$\boldsymbol{I_{norm} = \frac{I}{255}} \quad \textbf{(2.2)}$$

Keterangan:

I = Intensitas warna sebelum normalisasi

I_{norm} = Intensitas warna setelah normalisasi

Proses transformasi warna RGB to LAB diakhiri dengan melakukan denormalisasi nilai intensitas LAB yang didapatkan sehingga dapat ditampilkan kembali dalam bentuk citra 8 bit. Proses denormalisasi nilai intensitas LAB dapat dilakukan menggunakan Persamaan 2.3 sampai Persamaan 2.5.

$$\boldsymbol{L_{norm} = \frac{L \times 255}{100}} \quad \textbf{(2.3)}$$

$$\boldsymbol{A_{norm} = A + 125} \quad \textbf{(2.4)}$$

$$\boldsymbol{B_{norm} = B + 125} \quad \textbf{(2.5)}$$

2.3 Preprosesing

Preprosesing merupakan langkah yang dilakukan pemrosesan data untuk mendapatkan data yang akurat dan konsisten (Kumar & Chandrasekhar, 2012). Preprosesing juga dapat diartikan pada proses menghilangkan bagian-bagian yang tidak diperlukan pada citra untuk proses selanjutnya. Dalam pengolahan citra terdapat beberapa prinsip untuk meningkatkan kualitas citra, antara lain:

1. Peningkatan kecerahan dan kontras.

   Pengambilan citra menggunakan kamera *smartphone* terkadang menghasilkan citra dengan kualitas yang kurang bagus. Hal ini dikarenakan tingkat pencahayaan yang terlalu ekstrem maupun teknik pemotretan yang kurang benar. Permasalahan ini dapat diselesaikan dengan pengolahan citra digital. Contoh dari proses peningkatan kecerahan dan kontras ditunjukkan pada Gambar 2.1.

(a) **(b)**

Gambar 2.1 Citra dengan kontras yang berbeda: (a) citra dengan kontras rendah (b) citra dengan kontras seimbang.

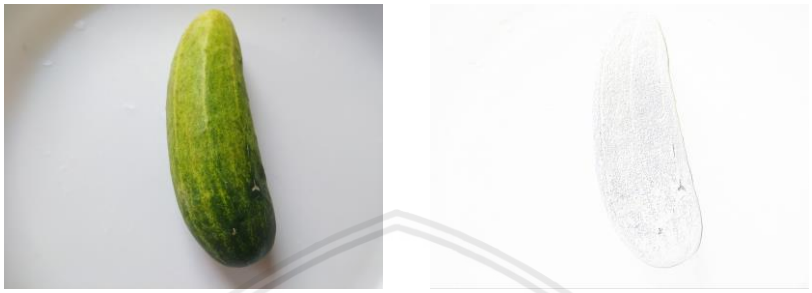
2. Penghilangan derau.

   Derau (*noise*) merupakan sinyal yang tidak diinginkan dalam suatu sistem komunikasi atau informasi yang tidak diinginkan keberadaanya (Kho, 2018). Derau yang terdapat pada citra seringkali menurunkan kualitas dari suatu citra tersebut. Sehingga diperlukan penghilangan derau untuk meningkatkan kualitas citra. Contoh dari proses penghilangan derau ditunjukkan pada Gambar 2.2.

Gambar 2.2 Citra dengan perbedaan intensitas derau: (a) citra berderau (b) citra tanpa derau.

3. Pencarian bentuk objek.

   Untuk mengenali sebuah objek, objek perlu dipisahkan terlebih dahulu dari latar belakangnya. Salah satu pendekatan yang umum dilakukan untuk keperluan ini adalah penemuan batas objek. Setelah tepi objek ditemukan, pencarian ciri terhadap sebuah citra dapat dilakukan. Contoh dari proses pencarian bentuk objek ditunjukkan pada Gambar 2.3.



(a) **(b)**

Gambar 2.3 Proses deteksi tepi: (a) citra sebelum dilakukan deteksi tepi (b) citra setelah dilakukan deteksi tepi.

2.3.1 Transformasi Warna *RGB to Grayscale*

Warna merupakan cahaya kromatis pada kisaran panjang gelombang tertentu yang dipantulkan oleh objek. Warna yang terdapat pada suatu citra dapat digambarkan menggunakan berbagai ruang warna. Ruang warna merupakan suatu spesifikasi sistem koordinat dengan setiap warna dinyatakan sebagai sebuah titik di dalamnya (Kadir & Susanto, 2013).

Ruang warna RGB (*Red, Green, Blue*) merupakan ruang warna yang paling dikenal pada perangkat komputer yang sesuai dengan watak manusia dalam mengangkap warna. Penggunaan citra RGB terkadang tidak terlalu diperlukan dalam proses morfologi, sehingga diperlukan transformasi ruang warna dari RGB menjadi citra biner. Citra biner merupakan citra yang terdiri dari dua kemungkinan intensitas warna, yaitu hitam dan putih. Perubahan warna yang mencolok antara hitam dan putih dapat membantu proses morfologi citra kedepannya. Contoh dari transformasi warna citra RGB to Grayscale ditunjukkan pada Gambar 2.4.

(a) **(b)**

Gambar 2.4 Proses transformasi warna: (a) citra pada ruang warna RGB (b) citra *grayscale.*

Proses transformasi warna dari ruang warna menjadi citra *grayscale* dapat menggunakan beberapa model persamaan. Salah satu persamaan tersebut ditunjukkan pada Persamaan 2.6 (Kumar & Verma, 2010).

$$\boldsymbol{I_y = (0,333 \times F_R) + (0,5 \times F_G) + (0,16666 \times F_B)} \quad \textbf{(2.6)}$$

Keterangan:

I_y = nilai intensitas citra *grayscale*

F_R = nilai intensitas citra *RGB* pada *channel Red*

F_G = nilai intensitas citra *RGB* pada *channel Red*

F_B = nilai intensitas citra *RGB* pada *channel Red*

2.3.2 Segmentasi

Segmentasi merupakan salah satu tahapan preprosesing yang dilakukan sebelum melakukan proses ekstraksi fitur. Segmentasi bertujuan untuk membuang bagian yang tidak diperlukan dari citra dan mengekstrak *mask* dari objek utama saja (Adinugroho & Sari, 2017).

Segmentasi secara umum didasarkan pada dua nilai intensitas, yaitu *discontinuity* dan *similarity*. Segmentasi dilakukan untuk memisahkan citra berdasarkan perubahan intensitas yang mendadak pada sebuah citra. Contoh dari segmentasi pemegatan adalah deteksi garis. Kesamaan dilakukan untuk memisahkan citra berdasarkan suatu daerah yang sama berdasarkan kriteria yang telah ditentukan sebelumnya. *Thresholding, region growing, region splitting* dan *region merging* merupakan beberapa contoh dari segmentasi kesamaan (Gonzales & Woods, 2008). Contoh dari penerapan proses segmentasi ditunjukkan pada Gambar 2.5.

(a) **(b)**

Gambar 2.5 Proses segmentasi: (a) citra sebelum dilakukan segmentasi (b) citra setelah dilakukan segmentasi.

Penerapan segmentasi citra berdasarkan kesamaan (*similarity*) dapat didasarkan pada nilai intensitas masing-masing piksel. Segmentasi tersebut membentuk area di mana sejumlah piksel saling terkoneksi karena memiliki kesamaan warna. Pengukuran kemiripan warna dapat menggunakan alat ukur

warna. Salah satu alat ukur kemiripan warna adalah dengan menghitung jarak antara dua piksel menggunakan rumus jarak *Euclidean*, *City Block* dan lain-lain (Madenda, 2015).

Segmentasi citra juga dapat dilakukan dengan mengubah citra pada ruang warna tertentu menjadi citra biner. Hal tersebut dapat dilakukan dengan melakukan fungsi *thresholding*. Apabila suatu nilai piksel melebihi nilai *threshold*, maka nilai piksel tersebut diubah menjadi nilai maksimal. Sebaliknya, apabila nilai piksel lebih kecil dari nilai *threshold*, maka nilai piksel tersebut akan diubah menjadi nilai minimal. Adapun nilai minimal dan maksimal bergantung pada rentang masing-masing ruang warna. Fungsi *threshold* secara umum dapat dilakukan menggunakan Persamaan 2.7.

$$\boldsymbol{I_T(\mathbf{x})} = \begin{cases} \mathbf{255}, & \mathbf{x} \geq \mathbf{t} \\ \mathbf{0}, & \mathbf{x} < \mathbf{t} \end{cases} \quad \textbf{(2.7)}$$

Keterangan:

$\boldsymbol{I_T(\mathbf{x})}$ = nilai intensitas setelah dilakukan *thresholding*

x = nilai intensitas asli

t = nilai *threshold*

2.4 Ekstraksi Fitur

Ekstraksi fitur merupakan bagian fundamental dari analisis citra. Fitur adalah karakteristik unik dari suatu objek. Fitur yang diambil sebisa mungkin memiliki jumlah yang sedikit akan tetapi masih dapat digunakan untuk membedakan antara satu objek dengan objek yang lainnya.

Fitur secara umum dapat dibagi menjadi dua, yaitu fitur level rendah dan fitur level tinggi. Fitur level rendah dapat langsung diekstraksi dari citra. Sedangkan fitur level tinggi harus diolah terlebih dahulu menggunakan data fitur level rendah. Terdapat beberapa fitur yang dapat diekstrak dari citra, antara lain fitur warna, bentuk, tekstur dan ukuran.

2.4.1 Haralick Features

Tekstur menggambarkan tingkat kekasaran dan kehalusan suatu citra dilihat dari distribusi spasial tingkat keabuan pada hubungan antara satu piksel dengan piksel yang menjadi tetangganya. Salah satu dari teknik ekstraksi fitur tekstur yang sering digunakan adalah *Gray Level Co-occurence Matrix* (GLCM).

Gray Level Co-occurence Matrix (GLCM) merupakan probabilitas sebuah tingkat keabuan i muncul di sekitar tingkat keabuan *j* dengan jarak *d*, sudut θ, dan level warna *N*. Matriks probabilitas didapatkan menggunakan Persamaan 2.8 (Madenda, 2015).

$$\mathbf{P(i.j)} = \mathbf{CM(i.j)} / \sum_{i=0}^{G-1} \boldsymbol{i} \sum_{j=0}^{G-1} \boldsymbol{j} \quad \textbf{(2.8)}$$

Keterangan:

P(i,j) = fungsi probabilitas nilai piksel i bertetangga dengan nilai piksel j

CM(i,j) = jumlah nilai intensitas pada piksel i yang bertetangga dengan nilai intensitas piksel j

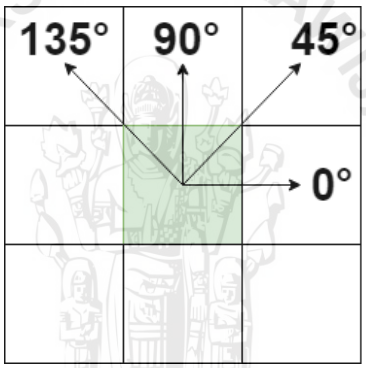
i = indeks baris

j = indeks kolom

G = panjang intensitas warna

Bentuk simetris dari GLCM didapatkan dengan dengan menjumlahkan matriks GLCM dengan transposenya dan melakukan normalisasi dengan membagi setiap elemen dengan jumlah seluruh elemen matriks GLCM.

Matriks GLCM yang sudah ternormalisasi nantinya akan digunakan untuk menghitung ciri statistik orde dua yang merepresentasikan tekstur. Penerapan ekstraksi fitur GLCM dapat menggunakan berbagai macam orientasi. Sebagai contoh, orientasi yang biasa digunakan pada ekstraksi fitur GLCM digambarkan pada Gambar 2.6.



Gambar 2.6 Orientasi fitur GLCM

Setiap orientasi matriks GLCM dapat didefinisikan menjadi empat belas fitur *Haralick*. Beberapa diantaranya adalah:

1. *Mean*

   Mean merupakan rata-rata distribusi probabilitas GLCM yang dapat dihitung berdasarkan sampel *x* dan *y*. Berdasarkan elemen pada matriks GLCM, mean berdasarkan sampel *x* dan sampel *y* dapat dihitung menggunakan Persamaan 2.9 dan 2.10.

$$\boldsymbol{\mu_x = \sum_{i=0}^{N} i \sum_{j=0}^{N} S(i,j)} \quad \textbf{(2.9)}$$

$$\boldsymbol{\mu_y = \sum_{j=0}^{N} j \sum_{i=0}^{N} S(i,j)} \quad \textbf{(2.10)}$$

Keterangan:

$\boldsymbol{\mu_x}$ = nilai mean berdasarkan sampel *x*

$\boldsymbol{\mu_y}$ = nilai mean berdasarkan sampel *y*

i = Indeks baris

j = Indeks kolom

S(i,j) = Intensitas pada baris ke-*i* dan kolom ke-*j*

2. Ragam / *Variance*

   Varians menentukan sebaran data atau simpangan data dari rata-rata data. Berdasarkan elemen pada matriks GLCM, varians berdasarkan sampel x dan sampel y dapat dihitung menggunakan Persamaan 2.11 dan 2.12.

$$\boldsymbol{\sigma_x^2 = \sum_{i=0}^{N}(i-\mu_x)^2 \sum_{j=0}^{N} S(i,j)} \quad \textbf{(2.11)}$$

$$\boldsymbol{\sigma_y^2 = \sum_{j=0}^{N}(j-\mu_y)^2 \sum_{i=0}^{N} S(i,j)} \quad \textbf{(2.12)}$$

Keterangan:

σ_x^2 = nilai varians berdasarkan sampel *x*

σ_y^2 = nilai varians berdasarkan sampel *y*

3. Kontras

   Kontras menunjukkan besar perubahan lokal yang terjadi pada citra. Berdasarkan elemen pada matriks GLCM, kontras dapat dihitung menggunakan Persamaan 2.13.

$$\boldsymbol{Kontras = \sum_{i=0}^{N}\sum_{j=0}^{N}[(i-j)^2 S(i,j)]} \quad \textbf{(2.13)}$$

4. *Dissimilarity*

   Dissimilarity merupakan ukuran yang mendefinisikan variasi tingkat intensitas pasangan piksel dalam citra. Berdasarkan elemen pada matriks GLCM, *dissimilarity* dapat dihitung menggunakan Persamaan 2.14.

$$\boldsymbol{Dissimilarity = \sum_{i=0}^{N}\sum_{j=0}^{N}[|i-j|S(i,j)]} \quad \textbf{(2.14)}$$

5. Korelasi

   Korelasi menunjukkan bagaimana suatu piksel berkorelasi dengan piksel di sekitarnya. Berdasarkan elemen pada matriks GLCM, korelasi dapat dihitung menggunakan Persamaan 2.15.

$$\boldsymbol{Korelasi = \frac{\sum_{i=0}^{N}\sum_{j=0}^{N}[i \times j \times S(i,j)] - \mu^2}{\sigma^2}} \quad \textbf{(2.15)}$$

6. Homogenitas

   Homogenitas menunjukkan kesamaan yang terjadi pada piksel yang terdapat pada citra. Berdasarkan elemen pada matriks GLCM, homogenitas dapat dihitung menggunakan Persamaan 2.16.

$$\boldsymbol{Homogenitas} = \sum_{i=0}^{N}\sum_{j=0}^{N}\frac{S(i,j)}{1+(i-j)^2} \quad \textbf{(2.16)}$$

7. Energi

   Energi dapat diartikan juga sebagai keseragaman. Nilai energi sama dengan 1 akan didapatkan pada citra yang memiliki nilai keabuan konstan. Berdasarkan elemen pada matriks GLCM, energi dapat dihitung menggunakan Persamaan 2.17.

$$\boldsymbol{Energi} = \sum_{i=0}^{N}\sum_{j=0}^{N}[S(i,j)]^2 \quad \textbf{(2.17)}$$

8. Entropi

   Entropi menunjukkan ukuran dari acaknya nilai intensitas keabuan pada citra. Berdasarkan elemen pada matriks GLCM, entropi dapat dihitung menggunakan Persamaan 2.18.

$$\boldsymbol{Entropi} = \sum_{i=0}^{N}\sum_{j=0}^{N}S(i,j)\times\log[S(i,j)+\epsilon] \quad \textbf{(2.18)}$$

2.4.2 CIE L*a*b Color Moment

Warna adalah salah satu fitur visual yang paling penting bagi manusia. Menggunakan representasi warna, kita dapat memperkirakan warna dari suatu citra secara global. Dalam merepresentasikan warna, kita dapat menggunakan berbagai macam ruang warna.

Ruang warna merupakan model untuk merepresentasikan warna dalam jangkauan nilai intensitas tertentu. Terdapat berbagai macam ruang warna, antara lain RGB, Lab, HSV, HSI, YCbCr, CMYK dan lain-lain. Masing-masing ruang warna memiliki kelebihan dan kelemahan. Pemilihan ruang warna dapat menyesuaikan penggunaan dan fungsi yang diinginkan oleh peneliti.

L*a*b* merupakan standar internasional untuk pengukuran warna, yang diadopsi oleh Commission International d'Eclairage (CIE) pada tahun 1976. Model warna ini dapat menampilkan warna yang konsisten pada suatu citra dan tidak bergantung pada perangkat yang digunakan. "L" merepresentasikan kecerahan citra. "a" (antara hijau ke merah) dan "b" (antara biru ke kuning) merupakan parameter kromatik yang memiliki jangkauan -120 hingga 120. Untuk melakukan transformasi warna dari model warna RGB ke L*a*b* dapat menggunakan persamaan-persamaan (Patil, et al., 2011):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.19)$$

$$L = 116f(Y) - 16 \quad (2.20)$$

$$a = 500\left[f\left(\frac{X}{0.982}\right) - f(Y)\right] \quad (2.21)$$

$$b = 200\left[f(Y) - f\left(\frac{Z}{1.183}\right)\right] \quad (2.22)$$

$$f(X) = \begin{cases} 7.787X + 0.138 \; untuk \; X \leq 0.008856 \\ \sqrt[3]{X} \; untuk \; X > 0.008856 \end{cases} \quad (2.23)$$

Keterangan:

R,G,B = Intensitas warna RGB pada *channel* Red, Green, Blue

X*,Y,Z* = Ruang warna tristimulus XYZ

L = Intensitas warna LAB pada channel L (*Lightness*)

a = Intensitas warna LAB pada channel *a*

b = Intensitas warna LAB pada channel b

f(X) = Fungsi *threshold*

Proses ekstraksi fitur warna dapat dilakukan dengan beberapa metode. Sebagai contoh, metode histogram warna yang digunakan pada penelitian (Stricker & Orengo, 1995) menunjukkan bahwa metode tersebut sangat efektif untuk digunakan sebagai fitur warna karena tidak terpengaruh dengan posisi dan perubahan orientasi. Solusi dari kelemahan tersebut adalah menambahkan teknik partisi citra untuk menambahkan faktor distribusi lokal. Akan tetapi metode ini membutuhkan komputasi yang cukup tinggi.

Momen warna merupakan representasi dari fitur warna untuk menghitung similaritas dari satu citra dengan citra yang lain. Momen warna mengasumsikan distribusi warna dari sebuah citra sebagai distribusi probabilitas (Halim, et al., 2013). Penelitian (Dattatherya, et al., 2013) menggunakan empat momen warna sebagai fitur warna dari sebuah citra. Keempat momen warna tersebut antara lain:

1. Rata-rata, yang dapat dihitung menggunakan Persamaan 2.24.

$$E = \frac{1}{N}\sum_{i=1}^{N} p_i \quad (2.24)$$

2. Varians, yang dapat dihitung menggunakan Persamaan 2.25.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - E)^2} \quad (2.25)$$

3. *Skewness,* yang dapat dihitung menggunakan Persamaan 2.26.

$$s = \sqrt[3]{\frac{1}{N}\sum_{i=1}^{N}(p_i - E)^3} \quad (2.26)$$

4. Kurtosis, yang dapat dihitung menggunakan Persamaan 2.27.

$$\boldsymbol{K = \sqrt[4]{\frac{1}{N}\sum_{i=1}^{N}(p_i - E)^4}} \quad \textbf{(2.27)}$$

Keterangan:

N = jumlah data

p_i = nilai intensitas pada data ke-*i*

E = nilai rata-rata data

σ = nilai varians data

s = nilai *skewness* data

K = nilai kurtosis data

2.5 Klasifikasi

Klasifikasi menurut Kamus Besar Bahasa Indonesia adalah penyusunan bersistem dalam kelompok atau golongan menurut kaidah atau standar yang ditetapkan. Sedangkan menurut (Brownlee, 2016), klasifikasi merupakan salah satu dari pembelajaran terarah (*supervised learning*) dimana output yang dihasilkan berupa kategori seperti "merah" atau "biru", "penyakit" atau "bukan-penyakit" dan seterusnya.

2.6 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) adalah sistem pemrosesan informasi yang menyerupai jaringan syaraf biologis. Jaringan syaraf tiruan merupakan model matematis dari pemahaman manusia secara umum. Beberapa asumsi yang mendukung pernyataan tersebut, antara lain (Fausett, 2004):

1. Pemrosesan informasi terjadi pada suatu elemen yang simpel. Elemen ini disebut dengan neuron.
2. Terdapat sinyal yang mengalir antara masing-masing neuron melalui suatu koneksi.
3. Terdapat bobot pada setiap koneksi antar neuron.
4. Terdapat fungsi aktivasi untuk mengolah masukan menjadi keluaran.

Jaringan syaraf tiruan memiliki beberapa karakteristik utama, antara lain (Fausett, 2004):

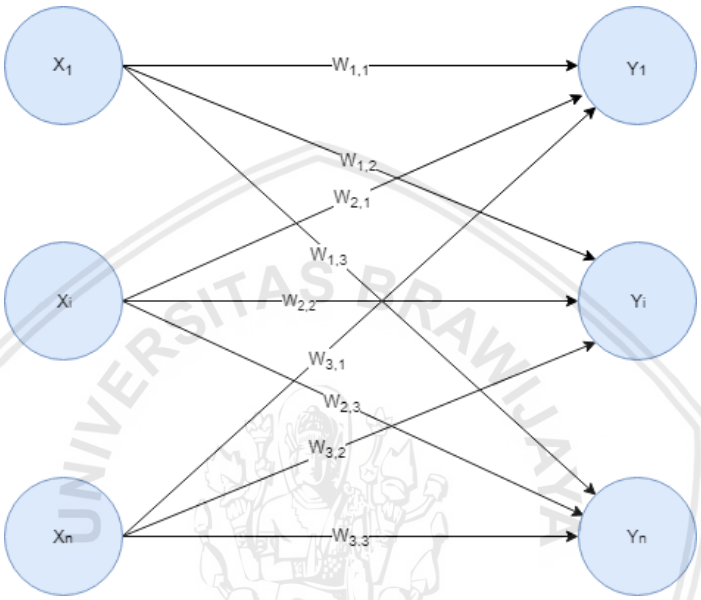
1. Pola hubungan antar neuron (arsitektur)
2. Metode untuk menentukan bobot koneksi (pelatihan)
3. Fungsi aktivasi untuk menentukan nilai keluaran.

2.6.1 Learning Vector Quantization

Learning Vector Quantization (LVQ) merupakan salah satu metode klasifikasi dari konsep jaringan syaraf tiruan. Setiap keluaran pada Learning Vector

Quantization merepresentasikan sebuah kelas atau kategori tertentu. Vektor bobot untuk suatu unit keluaran sering dinyatakan dengan sebuah vektor *referens*. Dalam proses pelatihan, unit keluaran diposisikan untuk dapat memperkirakan kelas menggunakan inisialisasi vektor *referens*. Setelah proses pelatihan, LVQ mengelompokkan vektor masukan dengan memasukkannya pada unit keluaran yang memiliki vektor bobot paling dekat dengan vektor masukan (Fausett, 2004).

Arsitektur dari *Learning Vector Quantization*, secara umum digambarkan seperti pada Gambar 2.7 (Fausett, 2004):



Gambar 2.7 Arsitektur Jaringan LVQ

Proses pembelajaran pada LVQ merupakan penggabungan antara konsep kompetisi dan pembelajaran terarah (*supervised learning)*. Sama seperti pembelajaran terarah lainnya, LVQ memerlukan data latih yang sebelumnya telah diberi label kategori. Proses pembelajaran pada LVQ3 secara lengkap sebagai berikut (Fausett, 2004):

1. Inisialisasi bobot pada lapisan kompetisi (w_j) dengan salah satu data latih pada masing-masing kelas dan tentukan nilai α (*learning rate*).
2. Selama kondisi belum terpenuhi, lakukan langkah 3 sampai 8.
3. Untuk setiap pasangan data latih beserta targetnya (x_i : t_i), lakukan langkah 4 sampai 7.
4. Tentukan neuron pemenang pertama dan neuron pemenang kedua pada lapisan kompetisi yang memiliki jarak *Euclidean* terkecil dengan x_i. Perhitungan jarak *Euclidean* dapat menggunakan Persamaan 2.28.

$$\boldsymbol{EDist = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}} \qquad \textbf{(2.28)}$$

Keterangan:

EDist = Jarak *Euclidean*

x_i = fitur ke-i data latih

y_i = fitur ke-i *reference vector*

5. Jika jarak dari data latih ke neuron pemenang pertama dan jarak dari data latih ke neuron pemenang kedua menunjukkan nilai yang hampir sama, maka lakukan langkah 6. Kondisi tersebut dapat digambarkan dengan Persamaan 2.29.

$$\mathbf{min}\left[\frac{d_{c1}}{d_{c2}}.\frac{d_{c2}}{d_{c1}}\right] > (\mathbf{1}-\boldsymbol{\epsilon})(\mathbf{1}+\boldsymbol{\epsilon}) \quad \textbf{(2.29)}$$

dengan d_{c1} adalah jarak data latih ke neuron pemenang pertama, d_{c2} adalah jarak data latih ke neuron pemenang kedua dan ϵ adalah konstanta yang bergantung pada jumlah sampel data latih. Nilai ϵ biasa bernilai 0,2;

6. Jika dua neuron pemenang / terdekat (y_{c1} dan y_{c2}) memiliki kelas yang sama dengan data latih, maka ubah bobot kedua neuron pemenang menggunakan Persamaan 2.30.

$$\boldsymbol{y_c}(\mathbf{t}+\mathbf{1}) = \boldsymbol{y_c}(\mathbf{t}) + \boldsymbol{\beta}[\mathbf{x}(\mathbf{t}) - \boldsymbol{y_c}(\mathbf{t})] \quad \textbf{(2.30)}$$

dengan $\beta = m\alpha$ dan $0{,}1 < m < 0{,}5$

7. Jika syarat pada langkah 6 tidak terpenuhi, maka cek apakah dua neuron pemenang / terdekat memenuhi kondisi dimana salah satu neuron (y_{c1}) memiliki kelas yang sama dengan data latih dan yang lain (y_{c2}) menunjukkan kelas yang berbeda dengan target data latih. Urutan dari y_{c1} dan y_{c2} tidak diperhatikan. Jika memenuhi syarat tersebut lakukan langkah 8

8. Ubah bobot *neuron* pemenang yang memiliki kelas yang sama dengan target data latih menggunakan Persamaan 2.31.

$$\boldsymbol{y_{c1}}(\mathbf{t}+\mathbf{1}) = \boldsymbol{y_{c1}}(\mathbf{t}) + \boldsymbol{\alpha}[\mathbf{x}(\mathbf{t}) - \boldsymbol{y_{c1}}(\mathbf{t})] \quad \textbf{(2.31)}$$

Dan lakukan update bobot neuron pemenang yang memiliki kelas yang berbeda dengan target data latih menggunakan Persamaan 2.32.

$$\boldsymbol{y_{c2}}(\mathbf{t}+\mathbf{1}) = \boldsymbol{y_{c2}}(\mathbf{t}) - \boldsymbol{\alpha}[\mathbf{x}(\mathbf{t}) - \boldsymbol{y_{c2}}(\mathbf{t})] \quad \textbf{(2.32)}$$

9. Kurangi nilai α

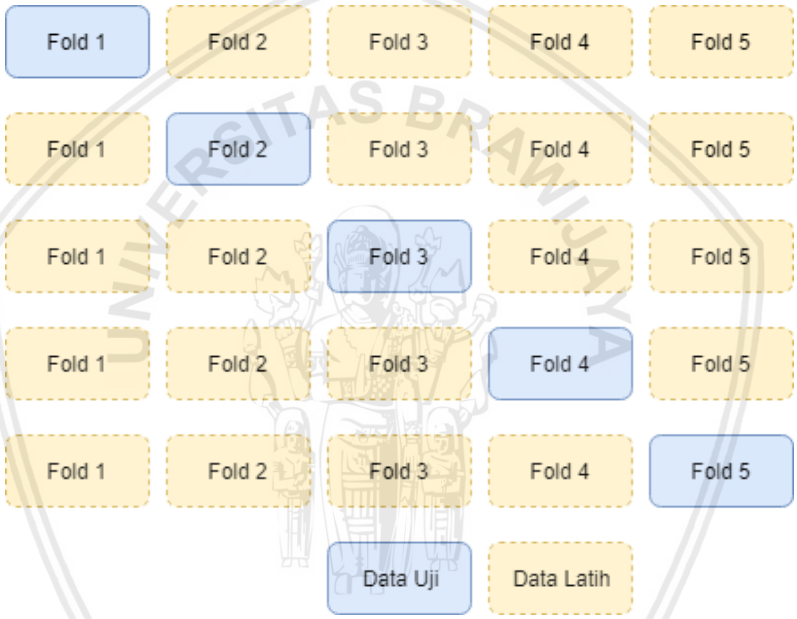
10. Lakukan pengujian kondisi berhenti. Iterasi dihentikan setelah mencapai epoch tertentu atau setelah nilai α mencapai nilai yang kecil.

Inisialisasi bobot awal biasanya menggunakan bilangan acak atau menggunakan sebagian data latih yang mewakili setiap kelas yang ada (Adinugroho & Sari, 2017).

2.7 Pengujian

Pengujian dilakukan untuk mengetahui tingkat keberhasilan sistem dalam melakukan klasifikasi. Terdapat beberapa jenis metode pengujian, antara lain *hold-out validation* dan *cross validation*. *Hold-out validation* merupakan salah satu metode pengujian yang membagi seluruh data menjadi data latih dan data uji pada proporsi tertentu. Sebagai contoh, 80% digunakan sebagai data latih dan 20% data digunakan sebagai data uji (Allibhai, 2018).

Cross-validation merupakan salah satu metode pengujian yang membagi keseluruhan data menjadi beberapa bagian untuk selanjutnya dilakukan pengujian secara bergantian. Seluruh data dibagi menjadi *k*-bagian sehingga metode ini seringkali disebut sebagai metode *k-fold cross validation*. Metode cross validation secara umum dapat digambarkan dengan Gambar 2.8.



Gambar 2.8 Ilustrasi Metode k-Fold Cross Validation

Secara umum, tahapan dari metode *k-fold cross validation* adalah sebagai berikut (Brownlee, 2018).

1. Acak keseluruhan data
2. Bagi keseluruhan data menjadi beberapa kelompok sejumlah *k*.
3. Pada setiap kelompok, lakukan langkah 4-7.
4. Jadikan kelompok tersebut sebagai data uji.
5. Jadikan kelompok yang lain sebagai data latih.
6. Lakukan evaluasi pada kelompok tersebut.
7. Simpan hasil evaluasi.
8. Jumlahkan hasil evaluasi dari masing-masing kelompok dan lakukan analisis.

Salah satu metode evaluasi yang sering digunakan adalah dengan menghitung akurasi. Akurasi didapatkan dengan membandingkan jumlah data yang sesuai dengan target dengan jumlah data keseluruhan. Akurasi direpresentasikan dalam bentuk persentase. Akurasi dapat dihitung menggunakan Persamaan 2.34.

$$\boldsymbol{Akurasi} = \frac{\boldsymbol{jumlah\ data\ yang\ relevan}}{\boldsymbol{jumlah\ data\ keseluruhan}} \times \mathbf{100}\% \qquad \textbf{(2.34)}$$

BAB 3 METODOLOGI

Bab ini berisi pembahasan mengenai jenis penelitian, strategi dan rancangan penelitian serta jadwal kegiatan penelitian.

3.1 Jenis Penelitian

Penelitian ini menggunakan jenis penelitian non-implementatif analitik. Pelaksanaan penelitian non-implementatif analitik menitikberatkan pada analisis terhadap hubungan antar fenomena yang sedang dikaji. Produk/artefak utama yang dihasilkan dari jenis penelitian ini berupa hasil analisis ilmiah. Jenis penelitian ini diharapkan dapat menjawab pertanyaan penelitian yang didefinisikan di awal baik secara kuantitatif maupun kualitatif .

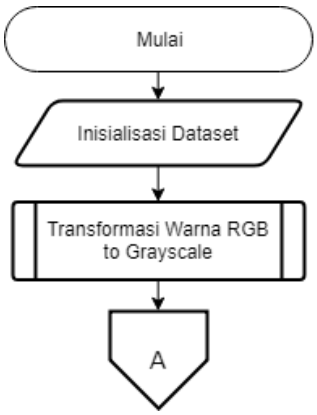
3.2 Strategi dan Rancangan Penelitian

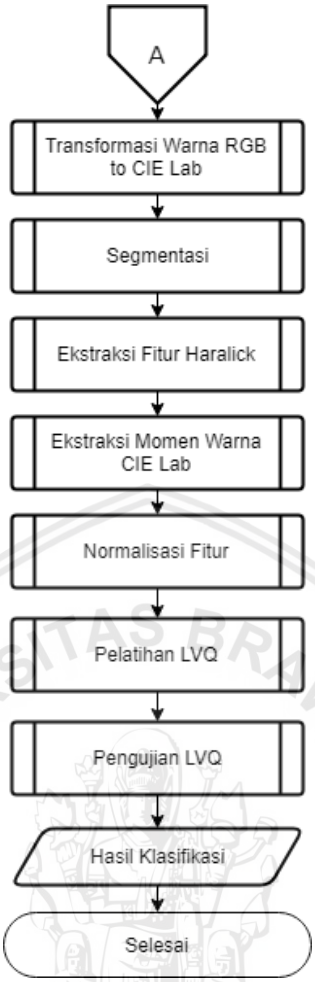
3.2.1 Metode secara umum

Metode penelitian ini secara umum digambarkan pada Gambar 3.1. Penelitian ini dimulai dengan melakukan inisialisasi data terlebih dahulu. Selanjutnya proses ekstraksi fitur tekstur dilakukan dengan melakukan transformasi warna dari ruang warna RGB menjadi *grayscale* terlebih dahulu. Kemudian dilakukan perhitungan fitur tekstur menggunakan Metode *Haralick Feature.*

Proses ekstraksi fitur warna dilakukan dengan melakukan preprosesing terlebih dahulu. Preprosesing yang dilakukan antara lain filtering, segmentasi dan transformasi warna dari ruang warna RGB menjadi ruang warna *CIE Lab.* Selanjutnya dilakukan perhitungan ciri warna menggunakan momen warna pada ruang warna *CIELab.*

Hasil ekstrasi ciri yang didapatkan dari metode *Haralick Feature* dan momen warna CIELab digunakan sebagai masukan pada metode LVQ3. Keluaran dari metode LVQ3 berupa hasil klasifikasi yang akan ditampilkan pada pengguna.





Gambar 3.1 Metode Penelitian secara umum

3.2.2 Lokasi Penelitian

Lokasi penelitian yang dipilih adalah lingkungan Fakultas Ilmu Komputer Universitas Brawijaya. Pemilihan lokasi ini didasarkan pada pertimbangan mengenai jarak antara lokasi dan domisili peneliti serta fasilitas yang terdapat pada lokasi tersebut. Pengumpulan data primer dilakukan di lantai 9 Gedung F Fakultas Ilmu Komputer Universitas Brawijaya dengan pertimbangan mengenai intensitas cahaya dan angin pada saat penelitian. Selanjutnya pengumpulan data sekunder dilakukan di Ruang Baca Fakultas Ilmu Komputer dan Perpustakaan Pusat Universitas Brawijaya. Pemilihan lokasi tersebut didasarkan pada beberapa pertimbangan seperti tempat dan suasana yang kondusif serta fasilitas jaringan internet yang mendukung.

3.2.3 Metode Pengumpulan Data

Pengumpulan data merupakan tahapan dimana penulis mengumpulkan data yang akan digunakan pada penelitian ini. Penelitian ini menggunakan dua teknik pengumpulan, yaitu pengumpulan data primer dan data sekunder.

3.2.3.1 Data Primer

Metode pengumpulan data primer yang digunakan pada penelitian ini adalah observasi. Observasi adalah teknik pengumpulan data, dimana peneliti melakukan pengamatan secara langsung pada objek data untuk mellihat dari dekat kegiatan yang dilakukan (Riduwan, 2004).

Dataset yang digunakan pada penelitian ini diperoleh melalui pengambilan citra langsung yang dilakukan bersama oleh Tim Food Project Fakultas Ilmu Komputer tahun 2018. Tim Food Project FILKOM 2018 beranggotakan 23 mahasiswa yang berasal dari Program Studi S1 Teknik Informatika Keminatan Komputasi Cerdas Angkatan 2015. Pengambilan data dilakukan pada tanggal 28 Agustus 2018. Pengambilan data dilakukan pada pukul 10.00 WIB dan berakhir pada pukul 14.00 WIB.

Dataset citra yang digunakan pada penelitian ini terbatas pada 31 kategori. Masing-masing citra merupakan objek citra penuh (bukan berupa potongan) yang diambil dengan tingkat kemiringan antara 15 sampai 90 derajat. Citra diambil menggunakan beberapa jenis smartphone sebagaimana digambarkan pada Tabel 3.1.

Tabel 3.1 Jenis Smartphone yang digunakan beserta jumlah citra yang diambil

No	Jenis Smartphone	Jumlah Citra
1	Huawei Nova 3i	319
2	Iphone 6s	54
3	Iphone 7+	350
4	Samsung J7 Prime	289
5	Samsung J7 Pro	279
6	Xiaomi Redmi 3 Pro	245
7	Xiaomi Redmi Note 4X	558
8	*Smartphone* lain	262

Keseluruhan data kemudian dihimpun dan dibagikan kepada masing-masing anggota tim menggunakan perantara penyimpanan cloud. Jumlah keseluruhan data yang digunakan pada penelitian ini adalah 2356 citra dengan perincian jumlah pada masing-masing kelas tersaji pada Tabel 3.2.

Tabel 3.2 Dataset berdasarkan masing-masing kelas

No	Jenis Makanan	Jumlah Citra
1	Donat	76
2	Roti Gandum	76
3	Roti Tawar	76
4	Indomie Goreng	76

Tabel 3.2 Dataset berdasarkan masing-masing kelas (lanjutan)

No	Jenis Makanan	Jumlah Citra
5	Mie Gepeng	76
6	Telur Ceplok	76
7	Telur Dadar	76
8	*Fried Chicken*	76
9	Rendang	76
10	Selada Air	76
11	Mentimun	76
12	Kubis	76
13	Selada	76
14	Kemangi	76
15	Tomat	76
16	Stroberi	76
17	Pisang Hijau	76
18	Pisang Kuning	76
19	Jeruk Oranye	76
20	Jeruk Ijo-Oranye	76
21	Nasi Kuning	76
22	Nasi Merah	76
23	Oreo	76
24	Beng-Beng	76
25	Soba Mie	76
26	Timtam	76
27	Happy Tos	76
28	Gerry Saluut	76
29	Biskuat	76
30	Milo Nuggets	76
31	Genji Pie	76

3.2.3.2 Data Sekunder

Data sekunder yang digunakan pada penelitian ini sebagian besar berupa *paper* internasional maupun nasional yang membahas mengenai topik dan metode pada penelitian ini. Selain itu, data sekunder yang lain berupa buku dan artikel dari situs yang dapat dipertanggungjawabkan untuk menunjang penjelasan mengenai dasar teori dari penelitian ini.

3.2.4 Peralatan Pendukung

Beberapa peralatan pendukung dibutuhkan untuk meningkatkan kualitas dari hasil penelitian yang dilakukan. Oleh karena itu, pada penelitian ini juga diperlukan beberapa peralatan pendukung yang terdiri dari perangkat keras dan perangkat lunak. Perangkat keras yang diperlukan pada penelitian ini ditunjukkan pada Tabel 3.3.

Tabel 3.3 Perangkat Keras Pendukung

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
Memori	8 GB DDR4
Harddisk	1 TB
Perangkat *mobile*	Xiaomi Redmi Note 4X

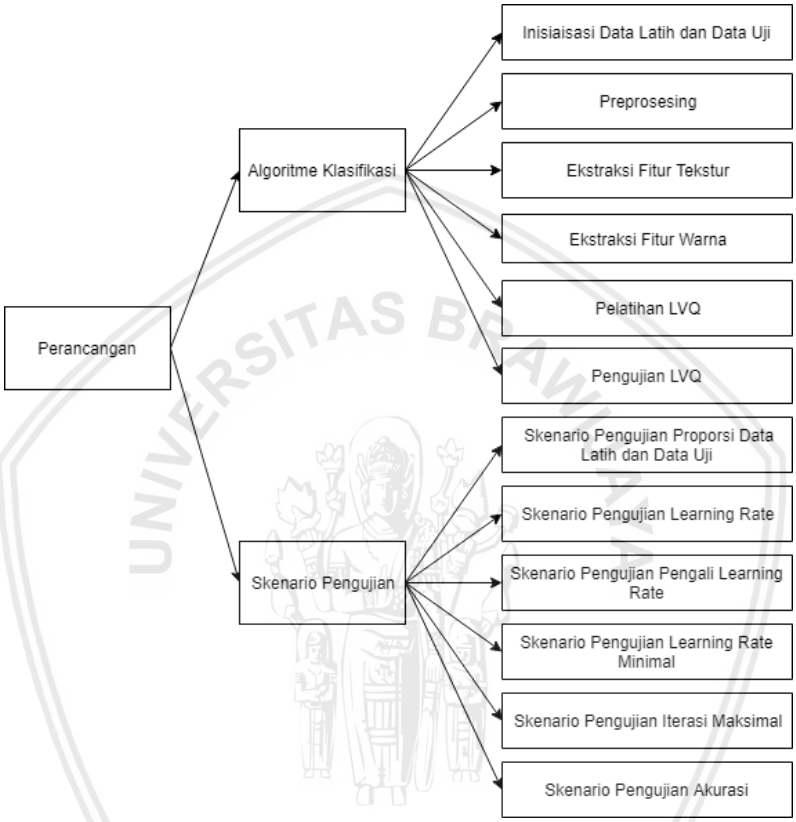
Adapun perangkat lunak yang dibutuhkan pada penelitian ini ditunjukkan pada Tabel 3.4.

Tabel 3.4 Perangkat Lunak Pendukung

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 10
Paket Pendukung Sistem	cv2, numpy, time, csv, imutils
Aplikasi Analisis Fitur	WEKA v3.8.3
Editor Pemrograman	Spyder (Python 2.7)
Editor Perancangan	*draw.io*
Editor Dokumentasi	Microsoft Office Word 2016
Browser	Google Chrome

BAB 4 PERANCANGAN ALGORITME

Bab ini menjelaskan tentang perancangan sistem klasifikasi citra makanan. Tahapan perancangan sistem klasifikasi citra makanan ini meliputi perancangan algoritme klasifikasi, perancangan antarmuka sistem dan perancangan pengujian skenario pengujian. Tahapan perancangan sistem secara umum digambarkan dengan Gambar 4.1.



Gambar 4.1 Tahapan Perancangan Sistem

4.1 Klasifikasi

Bagian ini menjelaskan tentang tahapan yang harus dilalui untuk melakukan klasifikasi, yaitu inisialisasi data latih dan data uji, preprosesing, ekstraksi fitur tekstur, ekstraksi fitur warna, normalisasi data, seleksi fitur dan algoritme LVQ3 yang meliputi tahapan pelatihan dan pengujian.

4.1.1 Inisialisasi Data Latih dan Data Uji

Dataset yang digunakan pada penelitian ini diperoleh melalui pengambilan citra langsung yang dilakukan bersama oleh Tim Food Project Fakultas Ilmu Komputer tahun 2018. Tim Food Project FILKOM 2018 beranggotakan 23 mahasiswa yang berasal dari Program Studi S1 Teknik Informatika Keminatan Komputasi Cerdas Angkatan 2015. Pengambilan data dilakukan pada tanggal 28 Agustus 2018. Pengambilan data dilakukan pada pukul 10.00 WIB dan berakhir pada pukul 14.00 WIB.

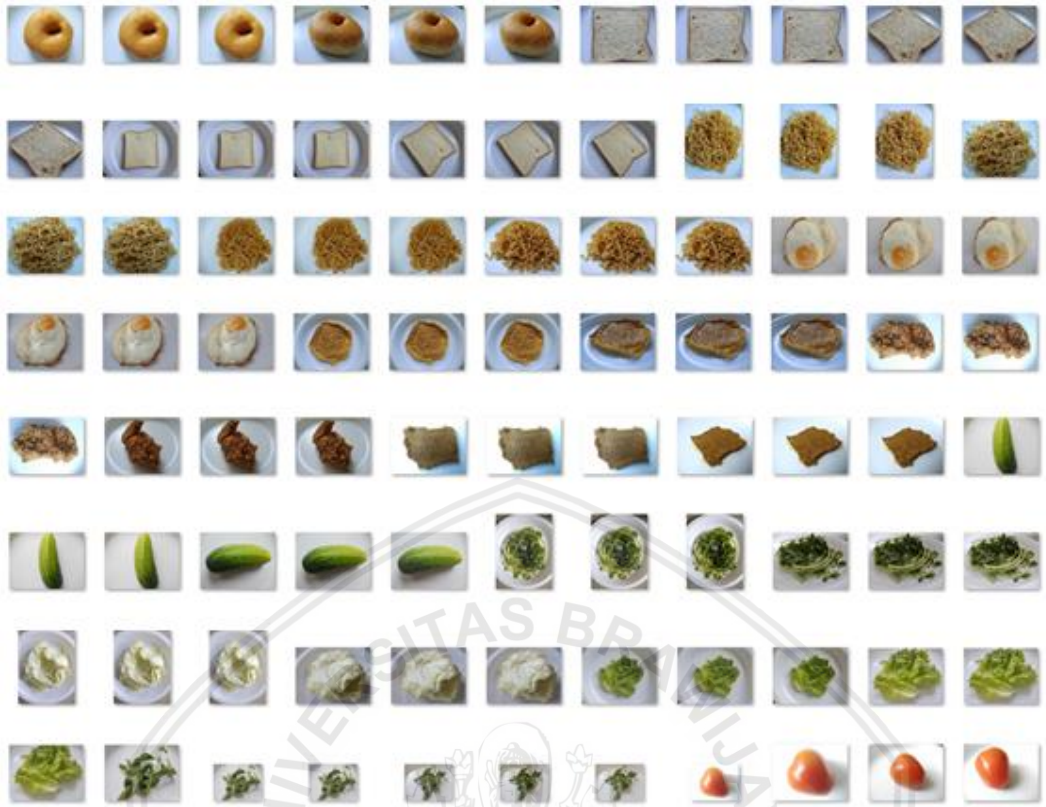
Dataset citra yang digunakan pada penelitian ini terbatas pada 31 kategori. Keseluruhan data kemudian dihimpun dan dibagikan kepada masing-masing anggota tim menggunakan perantara penyimpanan cloud. Jumlah keseluruhan data yang digunakan pada penelitian ini adalah 1350 citra dengan perincian jumlah pada masing-masing kelas tersaji pada Tabel 3.2.

Keseluruhan citra kemudian dibagi menjadi dua kelompok. Kelompok pertama adalah data yang akan digunakan pada proses pelatihan (data latih). Sedangkan kelompok kedua adalah data yang akan digunakan pada proses pengujian (data uji). Pengelompokan data pada proses awal dilakukan secara proporsional dengan perbandingan data latih dan data uji sebesar 2:1 pada setiap kelas. Contoh pembagian data latih dan data uji secara umum digambarkan pada Gambar 4.2 dan Gambar 4.3.

Gambar 4.2 Sampel Data Latih

4.1.2 Preprosesing

Preprosesing merupakan langkah yang dilakukan pemrosesan data untuk mendapatkan data yang akurat dan konsisten (Kumar & Chandrasekhar, 2012). Preprosesing juga dapat diartikan pada proses menghilangkan bagian-bagian yang tidak diperlukan pada citra untuk proses selanjutnya. Pada penelitian ini, tahapan preprosesing yang dilakukan antara lain transformasi warna dari ruang warna RGB menjadi *Grayscale*, transformasi warna dari ruang warna RGB menjadi LAB dan segmentasi citra.



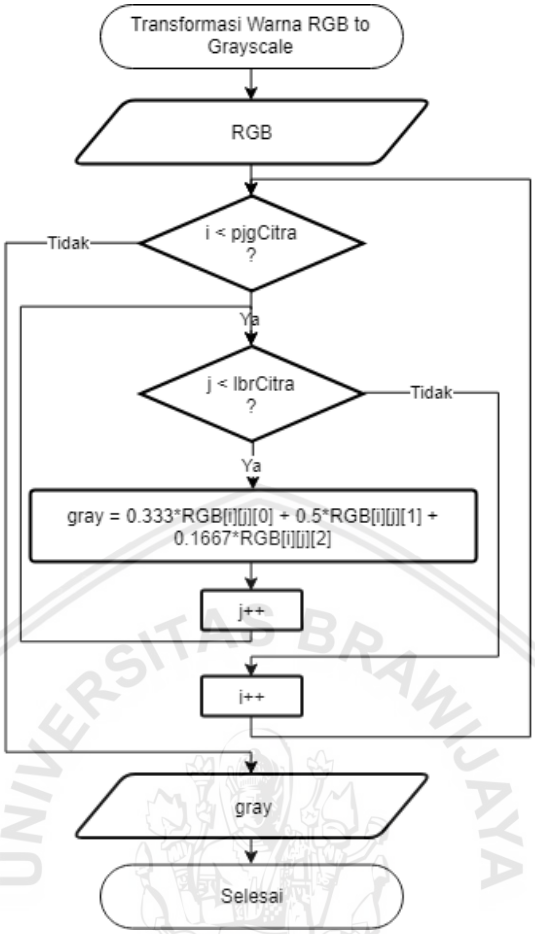
Gambar 4.3 Sampel Data Uji

4.1.2.1 Transformasi Warna RGB to Grayscale

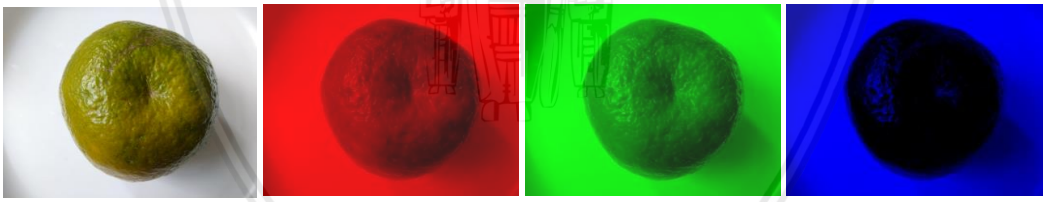
Salah satu tahapan preprosesing yang digunakan pada penelitian ini adalah transformasi warna RGB menjadi citra *grayscale*. Proses ini bertujuan untuk menghasilkan citra *grayscale* yang akan digunakan pada proses ekstraksi fitur tekstur. Proses transformasi warna RGB ke *grayscale* secara umum digambarkan dengan Gambar 4.4.

Berdasarkan Gambar 4.4 dapat dijelaskan tahapan-tahapan transformasi warna RGB ke *grayscale* adalah sebagai berikut.

1. Sistem mendapatkan masukan berupa citra dalam ruang warna RGB.
2. Sistem melakukan perulangan sebanyak jumlah piksel pada citra RGB tersebut.
3. Pada masing-masing perulangan dilakukan perhitungan nilai piksel *grayscale* menggunakan Persamaan 2.6.
4. Sistem mengembalikan keluaran berupa citra *grayscale*.



Gambar 4.4 Diagram Alir Transfomasi Warna RGB to *Grayscale*



(a) **(b)** **(c)** **(d)**

Gambar 4.5 Citra RGB: (a) citra dengan tiga *channel* RGB, (b) citra *channel* Red, (c) citra *channel* Green dan (d) citra *channel* Blue

Sebagai contoh, Gambar 4.5 menunjukkan citra RGB dengan ketiga channel RGB dan citra RGB pada masing-masing channel. Citra tersebut selanjutnya diambil nilai intensitas pada masing-masing *channel*-nya. Tabel 4.1 menunjukkan nilai intensitas 25 sampel piksel pada citra RGB tersebut.

Tahapan-tahapan perhitungan manual untuk proses transformasi warna RGB to *grayscale* adalah sebagai berikut.

Tahap 1: Identifikasi nilai intensitas pada masing-masing channel citra sebagaimana digambarkan pada Tabel 4.1.

Tabel 4.1 Sampel nilai intensitas citra RGB (Red/Green/Blue)

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	245/246/245	204/222/224	106/165/167	99/175/182	53/154/162
	91	231/237/236	139/183/185	108/171/172	42/152/154	63/159/163
	92	188/218/218	116/168/170	78/162/160	47/152/154	90/163/167
	93	141/182/183	115/174/175	47/157/153	87/160/163	86/159/163
	94	126/178/179	62/154/154	65/154/154	100/164/166	101/168/175

Keterangan

RED / GREEN / BLUE

Tahap 2: Lakukan perhitungan nilai intensitas citra *grayscale* menggunakan Persamaan 2.6. Sebagai contoh, perhitungan nilai intensitas citra *grayscale* pada posisi baris ke-90 dan kolom ke-107 adalah sebagai berikut.

$$I_{(90.107)} = (0{,}333 \times F_{R(90.107)}) + (0{,}5 \times F_{G(90.107)}) + (0{,}16666 \times F_{B(90.107)})$$
$$= (0{,}333 \times 245) + (0{,}5 \times 246) + (0{,}16666 \times 245)$$
$$= 81.585 + 123 + 40.8415$$
$$= 245.4265 \approx 245$$

Tahap 3: Simpan seluruh nilai intensitas piksel citra *grayscale*. Hasil perhitungan nilai intensitas citra grayscale dari Tabel 4.1 ditunjukkan pada Tabel 4.2.

Tabel 4.2 Sampel nilai intensitas citra *grayscale*

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	245	220	158	168	144
	91	236	178	164	140	149
	92	214	162	151	140	155
	93	177	167	143	152	151
	94	172	143	143	157	162

Hasil dari proses transformasi warna RGB ke *grayscale* adalah citra *grayscale* dengan rentang nilai antara 0 sampai dengan 255. Sebagai contoh, citra RGB pada Gambar 4.5 akan menjadi citra *grayscale* yang ditunjukkan pada Gambar 4.6.

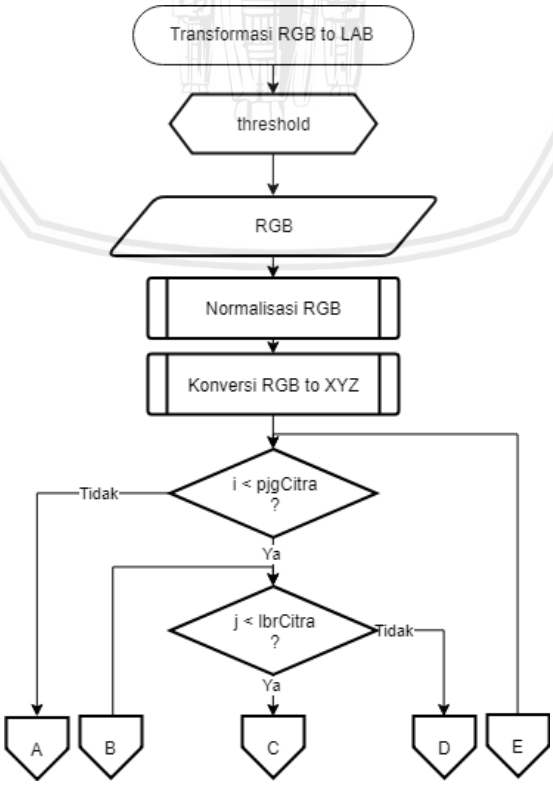
4.1.2.2 Transformasi Warna RGB to LAB

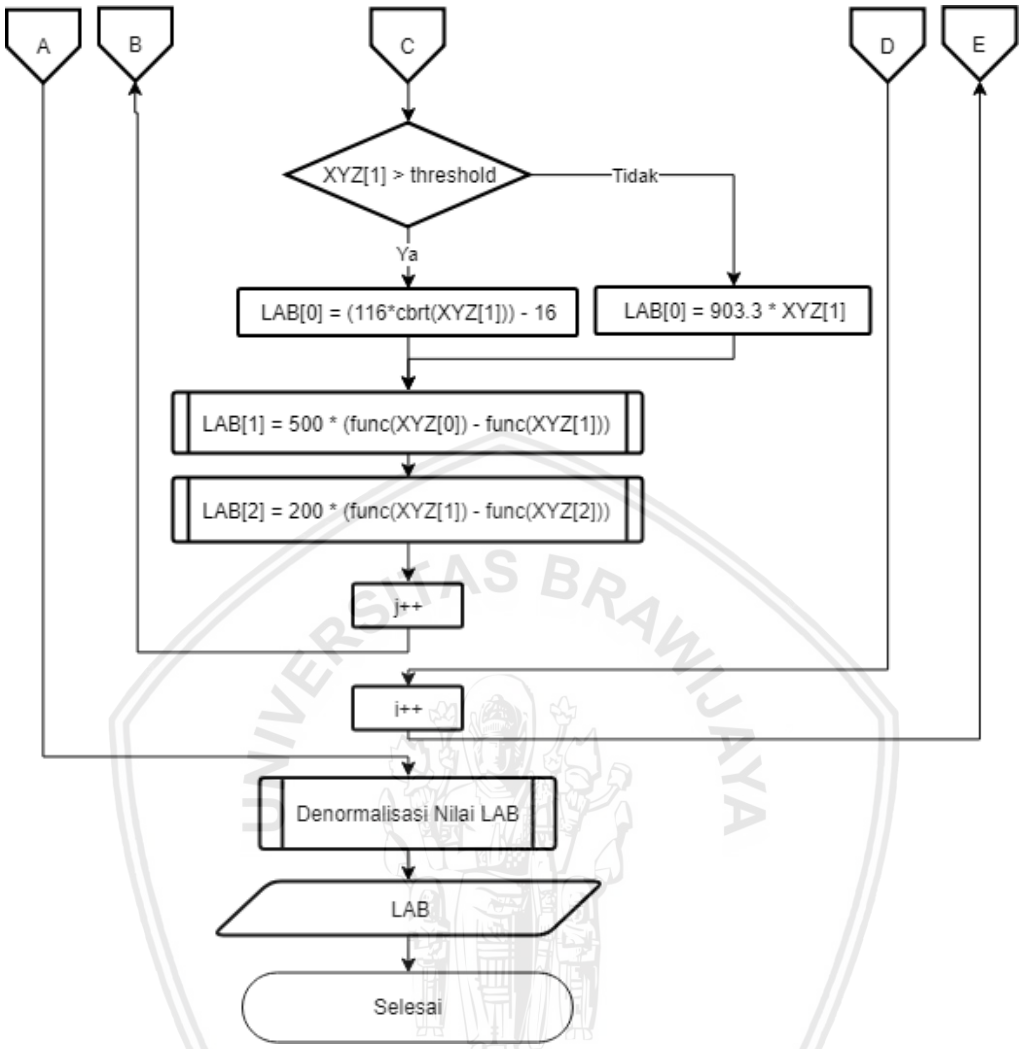
Tahapan preprosesing selanjutnya adalah transformasi warna dari ruang warna RGB ke LAB. Proses ini bertujuan untuk menghasilkan citra pada ruang warna LAB yang akan digunakan pada tahapan segmentasi dan ekstraksi fitur warna. Proses transformasi warna RGB ke *grayscale* secara umum digambarkan dengan Gambar 4.7.

Gambar 4.6 Citra *grayscale* hasil transformasi

Berdasarkan Gambar 4.7 dapat dijelaskan tahapan-tahapan transformasi warna RGB ke LAB adalah sebagai berikut:

1. Sistem mendapatkan masukan berupa citra dalam ruang warna RGB.
2. Lakukan normalisasi pada seluruh nilai intensitas piksel RGB menggunakan Persamaan 2.2.
3. Lakukan konversi nilai RGB menjadi nilai tristimulus XYZ menggunakan Persamaan 2.19.
4. Lakukan perulangan sebanyak piksel yang terdapat pada citra.
5. Lakukan perhitungan nilai L, A dan B menggunakan Persamaan 2.20 sampai Persamaan 2.23.
6. Lakukan denormalisasi pada nilai intensitas LAB menggunakan Persamaan 2.3 sampai Persamaan 2.5.
7. Sistem mengembalikan keluaran berupa citra dalam ruang warna LAB.





Gambar 4.7 Diagram Alir Transformasi Warna RGB *to* LAB

Tahapan-tahapan perhitungan manual untuk proses transformasi warna RGB ke LAB adalah sebagai berikut.

Tahap 1: Identifikasi nilai intensitas pada masing-masing channel citra sebagaimana digambarkan pada Tabel 4.1.

Tahap 2: Lakukan normalisasi pada seluruh nilai intensitas piksel RGB tersebut. Sebagai contoh, normalisasi nilai intensitas pada piksel posisi baris ke-90 dan kolom ke-107 adalah sebagai berikut.

$$I_{Rnorm(90.107)} = I_R/255 = 245/255 = 0.96078$$

$$I_{Gnorm(90.107)} = I_G/255 = 246/255 = 0.96471$$

$$I_{Bnorm(90.107)} = I_B/255 = 245/255 = 0.96078$$

Hasil dari normalisasi nilai intensitas piksel RGB pada Tabel 4.1 dapat digambarkan pada Tabel 4.3.

Tabel 4.3 Sampel hasil normalisasi nilai intensitas citra RGB (Red/Green/Blue)

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	0.960/ 0.964/ 0.960	0.905/ 0.929/0.925	0.737/ 0.854/0.854	0.552/ 0.713/0.717	0.494/ 0.698/0.701
	91	0.8/ 0.870/ 0.878	0.545/ 0.717/0.725	0.454/ 0.658/0.666	0.450/ 0.682/0.686	0.243/ 0.603/0.603
	92	0.415/ 0.647/0.654	0.423/ 0.670/0.674	0.305/ 0.635/0.627	0.184/ 0.615/0.6	0.254/ 0.603/0.603
	93	0.388/ 0.686/0.713	0.164/ 0.596/0.603	0.184/ 0.596/0.603	0.341/ 0.627/0.639	0.392/ 0.643/0.650
	94	0.207/ 0.603/0.635	0.247/ 0.623/0.639	0.352/ 0.639/0.654	0.337/ 0.623/0.639	0.396/ 0.658/0.686

Keterangan

RED / GREEN / BLUE

Tahap 3: Lakukan konversi nilai RGB menjadi nilai tristimulus XYZ. Sebagai contoh, konversi nilai RGB pada piksel posisi baris ke-90 dan kolom ke-107 adalah sebagai berikut.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} 0.960 \\ 0.964 \\ 0.960 \end{bmatrix}$$

$$= \begin{bmatrix} (0.412453 \times 0.960) + (0.357580 \times 0.964) + (0.180423 \times 0.960) \\ (0.212671 \times 0.960) + (0.715160 \times 0.964) + (0.072169 \times 0.960) \\ (0.019334 \times 0.960) + (0.119193 \times 0.964) + (0.950227 \times 0.960) \end{bmatrix}$$

$$= \begin{bmatrix} 0.913868 \\ 0.962861 \\ 1.045681 \end{bmatrix}$$

Hasil dari konversi nilai RGB ke XYZ pada Tabel 4.3 dapat digambarkan pada Tabel 4.4.

Tahap 4: Lakukan perhitungan nilai L, A dan B menggunakan Persamaan 2.20 sampai Persamaan 2.23. Sebagai contoh, perhitungan nilai LAB pada piksel posisi baris ke-90 dan kolom ke-107 adalah sebagai berikut.

- Variabel *Y* bernilai lebih dari *threshold* (0,008856), maka hitung *L* seperti ini:

$$L = 116\sqrt[3]{0.96358886} - 16$$
$$= 98.5747$$

- Hitung *f(X/0.950456), f(Y)* dan *f(Z/1.088754)*.

$$f(X/0.950456) = f(0.91458549/0.950456)$$

$$= f(0.96225968)$$
$$= 0.9872582315498232$$

$$f(Y) = f(0.96358886)$$
$$= 0.9877125922448947$$

$$f(Z/1.088754) = f\left(\frac{1.04652519}{1.088754}\right)$$
$$= f(0.96121363)$$
$$= 0.9869003598687307$$

Tabel 4.4 Sampel hasil konversi nilai RGB ke XYZ (X/Y/Z)

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	0.914/0.963/ 1.046	0.872/0.924/ 1.007	0.764/0.829/ 0.928	0.612/0.679/ 0.777	0.580/0.654/ 0.759
	91	0.799/0.856/ 0.953	0.612/0.681/ 0.785	0.543/0.616/ 0.720	0.553/0.633/ 0.742	0.425/0.527/ 0.650
	92	0.520/0.598/ 0.707	0.536/0.618/ 0.729	0.466/0.564/ 0.677	0.404/0.522/ 0.647	0.430/0.529/ 0.650
	93	0.534/0.624/ 0.767	0.390/0.504/ 0.648	0.398/0.509/ 0.648	0.480/0.567/ 0.688	0.509/0.590/ 0.702
	94	0.416/0.521/ 0.679	0.440/0.544/ 0.686	0.492/0.579/ 0.705	0.477/0.563/ 0.688	0.522/0.604/ 0.738

Keterangan

X / Y / Z

- Hitung nilai A dan B.

$$A = 500\left[f\left(\frac{X}{0.950456}\right) - f(Y)\right]$$
$$= 500[0.98726 - 0.98771]$$
$$= -0.22718$$

$$B = 200\left[f(Y) - f\left(\frac{Z}{1.088754}\right)\right]$$
$$= 200[0.98771 - 0.98690]$$
$$= 0.162446$$

Hasil perhitungan nilai LAB dari Tabel 4.4 dapat digambarkan pada Tabel 4.5.

Tahap 5: Lakukan denormalisasi pada nilai L, A dan B menggunakan Persamaan 2.3 sampai Persamaan 2.5. Sebagai contoh, perhitungan denormalisasi nilai LAB pada piksel posisi baris ke-90 dan kolom ke-107 adalah sebagai berikut.

$$L_{norm(90.107)} = \frac{L_{(90.107)} \times 255}{100}$$

$$= \frac{98.57 \times 255}{100}$$
$$= 251.3535 \approx 251$$

$$A_{norm(90.107)} = A_{(90.107)} + 128$$
$$= -0.22 + 128$$
$$= 127.78 \approx 127$$

$$B_{norm(90.107)} = B_{(90.107)} + 128$$
$$= 0.162 + 128$$
$$= 128.162 \approx 128$$

Tabel 4.5 Sampel hasil perhitungan nilai intensitas LAB (L/A/B)

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	98.57/ -0.22/0.162	97.10/ -0.63/1.275	93.72/ -2.59/7.295	87.14/ -3.83/11.66	86.21/ -5.02/15.38
	91	94.61/ -1.20/4.353	87.33/ -3.93/12.60	84.26/ -5.02/16.16	85.32/ -5.83/18.05	80.62/ -10.4/34.35
	92	83.54/ -5.84/18.91	84.66/ -6.33/19.73	82.37/ -9.60/28.97	80.80/ -13.4/42.80	80.66/ -10.0/32.88
	93	85.53/ -6.40/24.49	79.98/ -12.2/44.81	80.06/ -11.6/41.89	82.34/ -7.33/24.96	83.26/ -6.42/20.87
	94	80.85/ -9.70/39.53	81.82/ -9.79/35.48	83.01/ -7.02/24.63	82.17/ -7.14/25.17	84.26/ -5.61/21.90

Keterangan

L /
A / B

Hasil perhitungan nilai denormalisasi LAB dari Tabel 4.5 dapat digambarkan pada Tabel 4.6.

Tabel 4.6 Sampel hasil denormalisasi nilai LAB

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	251/127/128	247/127/129	239/125/135	222/124/139	219/122/143
	91	241/126/132	222/124/140	214/122/144	217/122/146	205/117/162
	92	213/122/146	215/121/147	210/118/156	206/114/170	205/117/160
	93	218/121/152	203/115/172	204/116/169	209/120/152	212/121/148
	94	206/118/167	208/118/163	211/120/152	209/120/153	214/122/149

Keterangan

L / A / B

4.1.2.3 Segmentasi

Tahapan preprosesing selanjutnya adalah segmentasi. Proses ini bertujuan untuk menghasilkan citra yang telah terpisah dari latar belakangnya (*background*). Harapan dari proses ini adalah hasil ekstraksi fitur yang dihasilkan nantinya dapat merepresentasikan objek dengan lebih tepat. Proses segmentasi secara umum digambarkan dengan Gambar 4.8.

Berdasarkan Gambar 4.8 dapat dijelaskan tahapan-tahapan proses segmentasi adalah sebagai berikut.

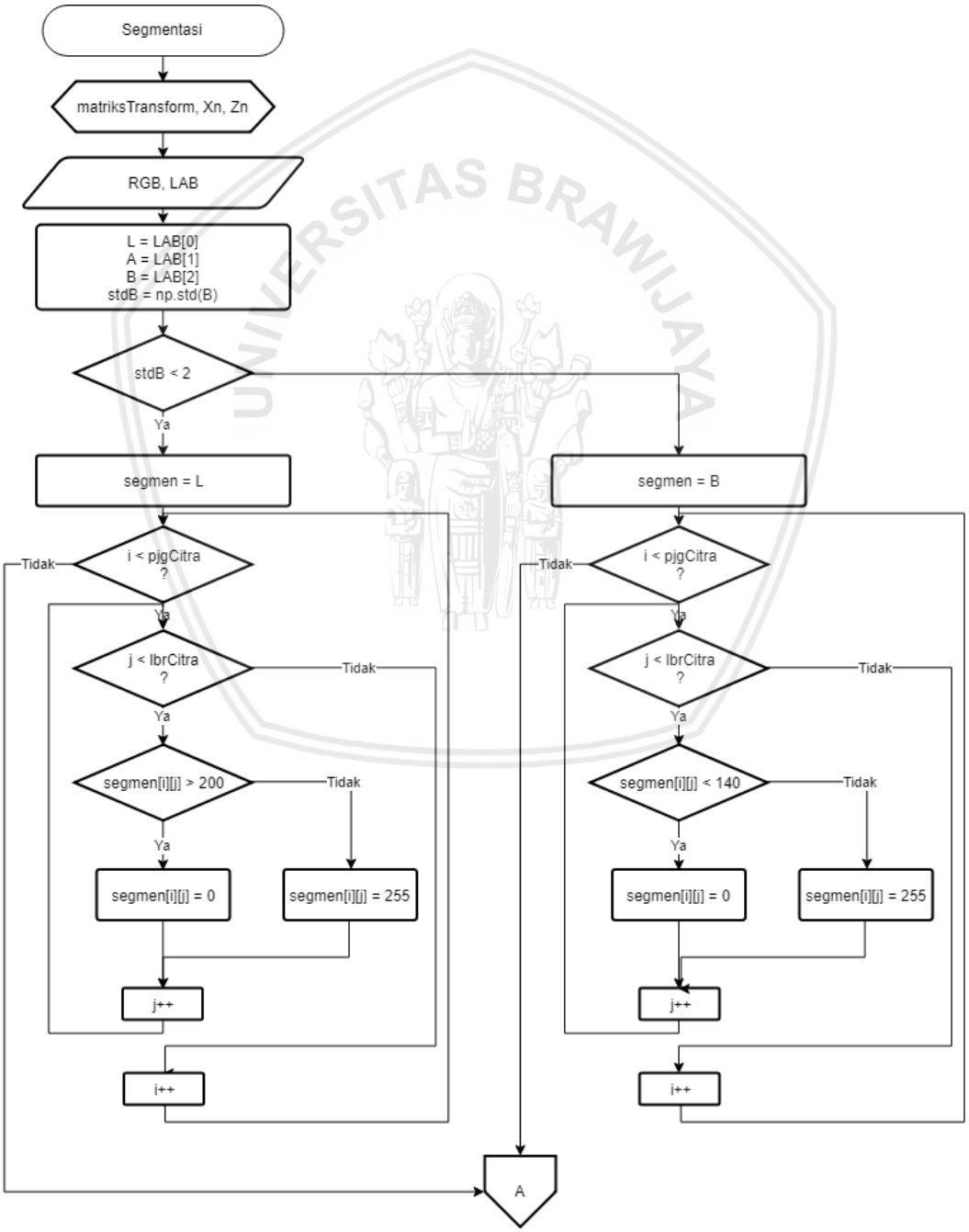
1. Sistem mendapatkan masukan berupa citra dalam ruang warna RGB.
2. Sistem melakukan transformasi warna dari ruang warna RGB ke LAB sebagaimana yang telah dijelaskan sebelumnya.
3. Sistem memisahkan masing-masing channel LAB (*L, A* dan *B*) dan menghitung nilai standar deviasi dari channel *B*.
4. Jika nilai standar deviasi dari channel *B* kurang dari 2, maka lakukan Langkah 5-7.
5. Isi variabel *segmen* dengan nilai *channel B*.
6. Jika nilai variabel *segmen* lebih besar dari 200, maka ganti nilai *segmen* menjadi 0.
7. Jika nilai variabel *segmen* lebih kecil sama dengan 200, maka ganti nilai *segmen* menjadi 255.
8. Jika nilai standar deviasi lebih dari sama dengan 2, maka lakukan Langkah 9-11.
9. Isi variabel *segmen* dengan nilai *channel L*.
10. Jika nilai variabel *segmen* lebih kecil dari 130, maka ganti nilai *segmen* menjadi 0.
11. Jika nilai variabel *segmen* lebih besar sama dengan 130, maka ganti nilai *segmen* menjadi 255.
12. Isi variabel *segmen2* dengan nilai dari *channel A*.
13. Jika nilai variabel *segmen2* lebih kecil dari 140, maka ganti nilai *segmen2* menjadi 0.
14. Jika nilai variabel *segmen2* lebih besar sama dengan 130, maka ganti nilai *segmen2* menjadi 255.
15. Lakukan operasi *bitwise* OR pada variabel *segmen* dan *segmen2,* kemudian simpan pada variabel *mask*.
16. Terapkan *masking* pada citra RGB.

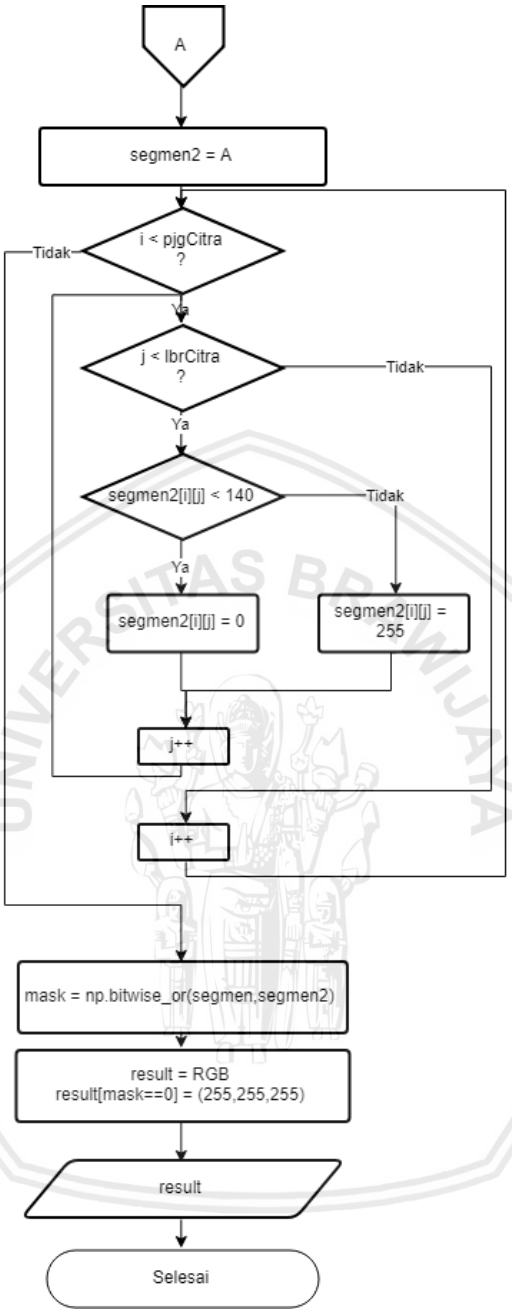
17. Sistem mengembalikan keluaran berupa citra RGB yang telah disegmentasi.

Tahapan-tahapan perhitungan manual untuk proses segmentasi adalah sebagai berikut.

Tahap 1: Identifikasi nilai intensitas pada masing-masing channel citra sebagaimana digambarkan pada Tabel 4.1.

Tahap 2: Lakukan transformasi warna RGB ke LAB sebagaimana digambarkan pada Tabel 4.6.





Gambar 4.8 Diagram Alir Proses Segmentasi

Tahap 3: Pisahkan nilai masing-masing *channel* LAB dan hitung nilai standar deviasi dari channel B. Misalkan diketahui citra pada Gambar 4.5 yang telah ditransformasi ke LAB memiliki standar deviasi channel B sebesar 5,414643814.

Tahap 4: Nilai standar deviasi citra lebih dari 2, sehingga variabel segmen berisi nilai channel B.

Tahap 5: Lakukan perulangan untuk seluruh nilai piksel segmen. Jika nilai segmen kurang dari 140, maka ubah nilai menjadi 0. Selain itu, ubah menjadi 255. Hasil dari langkah ini dapat digambarkan dengan Tabel 4.7.

Tahap 6: Isi variabel segmen2 dengan nilai channel A.

Tahap 7: Lakukan perulangan untuk seluruh nilai piksel segmen2. Jika nilai segmen2 kurang dari 140, maka ubah nilai menjadi 0. Selain itu, ubah menjadi 255. Hasil dari langkah ini dapat digambarkan dengan Tabel 4.8.

Tabel 4.7 Sampel nilai variabel segmen

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	0	0	0	0	255
	91	0	255	255	255	255
	92	255	255	255	255	255
	93	255	255	255	255	255
	94	255	255	255	255	255

Tabel 4.8 Sampel nilai variabel segmen2

Posisi		Kolom ke-				
		107	**108**	**109**	**110**	**111**
Baris ke-	**90**	0	0	0	0	255
	91	0	0	0	0	0
	92	0	0	0	0	0
	93	0	0	0	0	0
	94	0	0	0	0	0

Tahap 8: Lakukan operasi *bitwise* OR pada variabel segmen dan segmen2, dengan asumsi nilai 255 adalah TRUE dan nilai 0 adalah FALSE. Simpan hasil operasi tersebut pada variabel *mask.* Hasil dari operasi *bitwise* untuk kedua segmen dapat digambarkan dengan Gambar 4.9.

(a) (b) (c)

Gambar 4.9 Variabel Segmen: (a) Citra segmen (b) Citra segmen2 (c) Hasil operasi bitwise OR antara segmen dan segmen2

Tahap 8: Terapkan masking pada citra RGB asli. Hasil akhir segmentasi dapat digambarkan dengan Gambar 4.10.

Gambar 4.10 Hasil Akhir Segmentasi

4.1.3 Ekstraksi Fitur Tekstur (Haralick)

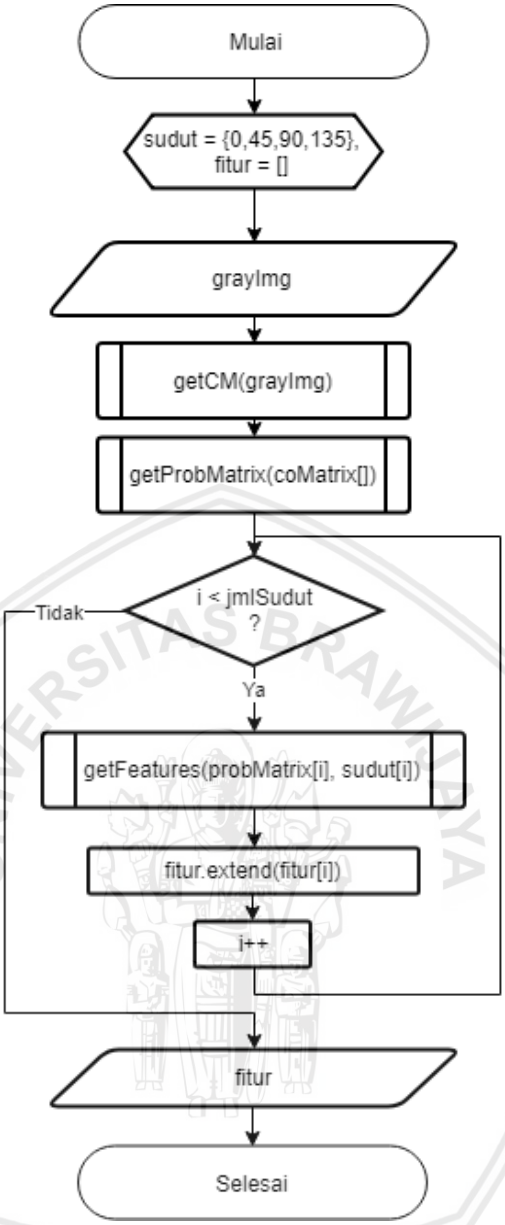
Langkah selanjutnya adalah ekstraksi fitur tekstur menggunakan *Haralick Features*. Proses ini memerlukan masukan berupa citra *grayscale* dari tahapan preprosesing sebelumnya. Keluaran dari proses ini adalah sepuluh nilai fitur Haralick untuk masing-masing citra. Proses ekstraksi fitur tekstur secara umum digambarkan dengan Gambar 4.11.

Berdasarkan Gambar 4.11 dapat dijelaskan tahapan-tahapan proses ekstraksi fitur tekstur adalah sebagai berikut.

1. Sistem mendapatkan masukan berupa citra *grayscale*.
2. Sistem melakukan perhitungan nilai matriks *co-occurence*. Tahapan penghitungan nilai matriks co-occurence akan ditunjukkan pada Gambar.
3. Sistem melakukan perhitungan nilai matriks probabilitas dari matriks *co-occurence* yang telah didapatkan sebelumnya. Tahapan perhitungan nilai matriks probabilitas akan ditunjukkan pada Gambar.
4. Sistem melakukan perulangan sejumlah sudut yang dikehendaki (0, 45, 90 dan 135) dan melakukan perhitungan fitur tekstur. Tahapan perhitungan fitur tekstur akan ditunjukkan pada Gambar.
5. Sistem mengembalikan nilai keluaran berupa sepuluh nilai fitur tekstur untuk setiap citra.

4.1.3.1 Menghitung Matriks Co-Occurence

Salah satu tahapan yang harus dilalui untuk melakukan ekstraksi fitur tekstur adalah menghitung nilai matriks *co-occurence* dari suatu citra. Proses ini bertujuan untuk mendapatkan jumlah kejadian dari dua nilai intensitas yang berada pada piksel bertetangga. Proses perhitungan nilai matriks *co-occurence* secara umum digambarkan dengan Gambar 4.12.



Gambar 4.11 Diagram Alir Proses Ekstraksi Fitur Tekstur

Berdasarkan Gambar 4.12 dapat dijelaskan tahapan-tahapan proses perhitungan matriks *co-occurence* adalah sebagai berikut.

1. Sistem mendapatkan parameter masukan berupa citra hasil segmentasi yang sudah di konversi menjadi citra *grayscale*.
2. Inisialisasi *array coMatrix* yang menampung empat *co-occurence matrix* dengan sudut 0, 45, 90 dan 135.
3. Lakukan perulangan untuk setiap piksel citra.
4. Cek apakah nilai intensitas piksel tetangga pada derajat 0 memenuhi syarat atau tidak. Syarat dari piksel tetangga pada derajat 0 adalah tidak bernilai NULL dan bukan berupa *background*. Suatu piksel dapat dikatakan sebagai piksel backgound jika intensitas piksel tersebut dan piksel tetangganya

yang sama-sama memiliki intensitas lebih dari 247 (untuk *background* citra bernilai 255).

5. Jika kondisi terpenuhi, maka lakukan penambahan nilai pada co-occurence matrix untuk nilai baris dan kolom adalah nilai intensitas piksel acuan dan piksel tetangganya.
6. Lakukan pengecekan serupa pada piksel tetangga pada derajat 45, 90 dan 135.
7. Sistem mengembalikan keluaran nilai berupa array *coMatrix*.

Tahapan-tahapan perhitungan manual untuk proses perhitungan nilai matriks *co-occurence* adalah sebagai berikut.

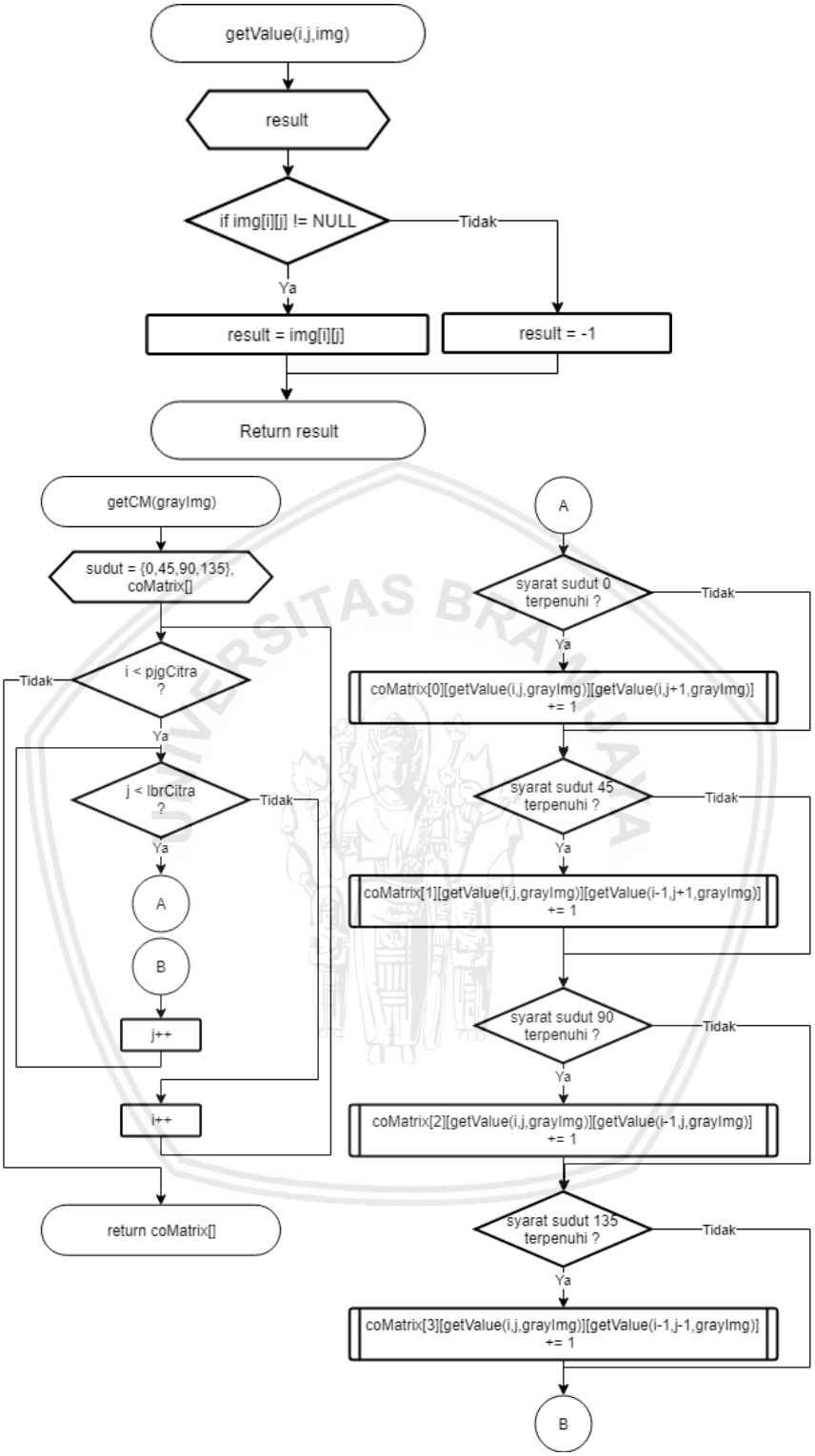
Tahap 1: Identifikasi nilai intensitas pada citra *grayscale* sebagai masukan proses ekstraksi fitur tekstur. Sebagai contoh, terdapat citra dengan sebaran nilai intensitas yang ditunjukkan pada Tabel 4.9.

Tahap 2: Lakukan perulangan untuk masing-masing piksel. Cek nilai intensitas piksel tersebut dan piksel tetangganya. Piksel tetangga yang digunakan adalah piksel yang terletak tepat di kanan, di atas, diagonal kanan atas dan diagonal kiri atas dari piksel acuan. Sebagai contoh piksel indeks baris ke-1 kolom ke-1 memiliki tetangga antara lain:

1. Piksel baris ke-1 kolom ke-2, sebagai tetangga pada derajat 0
2. Piksel baris ke-0 kolom ke-2, sebagai tetangga pada derajat 45
3. Piksel baris ke-0 kolom ke-1, sebagai tetangga pada derajat 90
4. Piksel baris ke-0 kolom ke-0, sebagai tetangga pada derajat 135

Tabel 4.9 Contoh sebaran nilai intensitas citra

Posisi		Kolom ke-							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Baris ke-	**0**	0	1	3	0	3	1	4	1
	1	4	1	2	2	1	2	3	4
	2	1	3	2	1	0	2	0	3
	3	0	2	4	2	3	2	2	3
	4	1	3	2	3	0	4	3	1
	5	3	2	2	4	3	2	3	4
	6	4	3	4	0	3	4	0	1
	7	2	4	3	1	3	2	2	2



Gambar 4.12 Diagram Alir Proses Perhitungan Matriks Co-occurence

Tahap 3: Jika piksel tetangga memenuhi syarat (tidak bernilai NULL dan bukan piksel *background*), maka lakukan penambahan nilai pada *co-occurence matrix* dengan indeks baris dan kolom sesuai dengan nilai intensitas piksel acuan dan piksel tetangganya.

Sebagai contoh, piksel indeks baris ke-1 kolom ke-1 memiliki nilai intensitas 1. Tetangga derajat 0 dari piksel tersebut telah memenuhi syarat (tidak bernilai NULL dan bukan piksel *background*) dan memiliki nilai intensitas 2. Maka lakukan penambahan nilai pada *co-occurence matrix* untuk derajat 0 pada indeks ke-1 kolom-2.

Pembuktikan ketepatan nilai *co-occurence matrix* dapat dilakukan dengan melakukan *cross-check* pada tabel co-occurence matrix. Sebagai contoh, indeks baris ke-2 kolom ke-4 co-occurence matrix sudut 0 pada Tabel 4.10 menunjukkan nilai 3. Hal tersebut menunjukkan bahwa terdapat tiga piksel acuan bernilai 2 yang memiliki tetangga derajat 0 bernilai 4. Ketiga piksel tersebut adalah piksel baris ke-7 kolom ke-0, piksel baris ke-3 kolom ke-1 dan piksel baris ke-5 kolom ke-2.

Tabel 4.10 Contoh Matriks Co-occurence

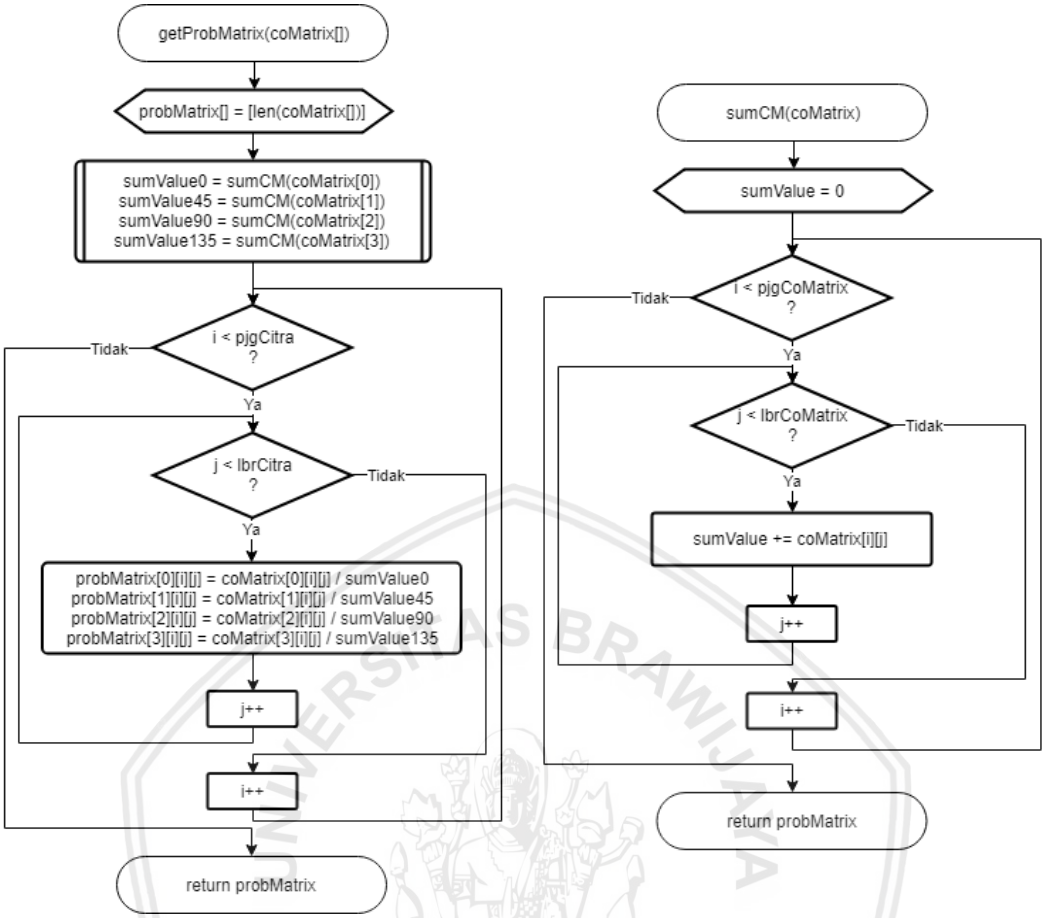
Nilai Intensitas		Intensitas Piksel Tetangga Derajat 0				
		0	**1**	**2**	**3**	**4**
Intesitas Piksel Acuan	**0**	0	2	2	3	1
	1	1	0	2	4	1
	2	1	2	5	5	3
	3	2	3	6	0	4
	4	2	2	1	4	0

4.1.3.2 Menghitung Probability Matrix

Tahapan selanjutnya untuk melakukan ekstraksi fitur tekstur adalah menghitung nilai matriks probabilitas dari matriks *co-occurence* yang telah didapatkan dari proses sebelumnya. Proses ini bertujuan untuk mendapatkan nilai probabilitas dari matriks *co-occurence*. Proses perhitungan nilai *probability matrix* secara umum digambarkan dengan Gambar 4.13.

Berdasarkan Gambar 4.13 dapat dijelaskan tahapan-tahapan proses perhitungan *probability matrix* adalah sebagai berikut.

1. Sistem mendapatkan parameter masukan berupa *array coMatrix* berisi empat matriks co-occurence pada sudut 0, 45, 90 dan 135.
2. Inisialisasi *array probMatrix* untuk menyimpan empat matriks probabilitas.
3. Hitung jumlah dari seluruh nilai *co-occurence matrix* pada masing-masing sudut menggunakan fungsi *sumCM*.
4. Lakukan pembagian antara nilai *co-occurence matrix* pada indeks tertentu dengan jumlah nilai *co-occurence matrix* sesuai sudut *co-occurence matrix*.
5. Sistem mengembalikan nilai keluaran berupa *array probMatrix*.



Gambar 4.13 Diagram Alir Proses Perhitungan Probability Matrix

Tahapan-tahapan perhitungan manual untuk proses perhitungan *probability matrix* adalah sebagai berikut.

Tahap 1: Lakukan perulangan untuk semua indeks co-occurence matrix dan hitung jumlah seluruh nilai. Sebagai contoh untuk matriks co-occurence pada Tabel 4.10 menghasilkan nilai sumCM sebesar 55 (didapatkan dari 7+8+16+15+9).

Tabel 4.11 Contoh Matriks Probabilitas

Probabilitas		**Kolom ke-**				
		0	**1**	**2**	**3**	**4**
Baris ke-	**0**	0	0.04	0.04	0.05	0.02
	1	0.02	0	0.04	0.07	0.02
	2	0.02	0.04	0.09	0.09	0.05
	3	0.04	0.05	0.11	0	0.07
	4	0.04	0.04	0.02	0.07	0

Tahap 2: Isi nilai *probability matrix* dengan hasil pembagian antara *co-occurence matrix* pada indeks tertentu dengan jumlah seluruh nilai *co-occurence matrix* pada sudut tertentu. Sebagai contoh, nilai *probability matrix* indeks baris ke-2 kolom ke-1 adalah 2/55 = 0,04. Lakukan pada seluruh indeks *probability matrix* pada seluruh

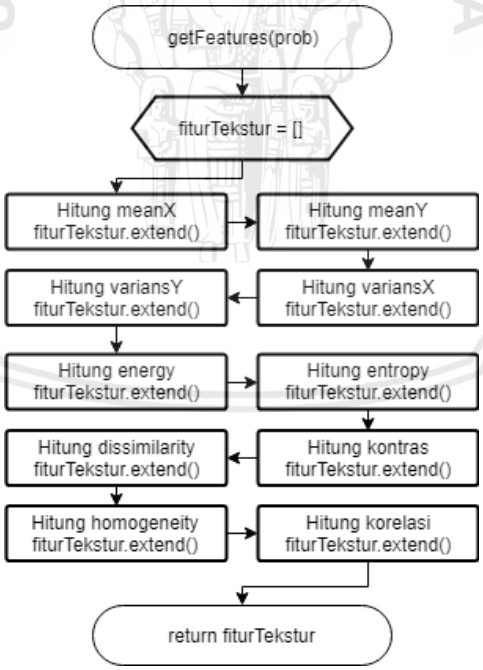
sudut sehingga nilai *probability matrix* sudut 0 dapat ditunjukkan dengan Tabel 4.11.

4.1.3.3 Menghitung Fitur Tekstur

Tahapan selanjutnya untuk melakukan ekstraksi fitur tekstur adalah menghitung fitur tekstur berdasarkan matriks probabilitas yang telah didapatkan dari proses sebelumnya. Proses ini bertujuan untuk mendapatkan nilai fitur tekstur dari sebuah citra. Proses perhitungan nilai fitur tekstur secara umum digambarkan dengan Gambar 4.11.

Berdasarkan Gambar 4.13 dapat dijelaskan tahapan-tahapan proses perhitungan *probability matrix* adalah sebagai berikut.

1. Sistem mendapatkan parameter masukan berupa *array probMatrix* berisi empat matriks probabilitas pada sudut 0, 45, 90 dan 135.
2. Inisialisasi *array* kosong dengan nama *fiturTekstur* untuk menampung hasil perhitungan fitur.
3. Lakukan perhitungan untuk masing-masing fitur dan tambahkan hasil perhitungan ke *array fiturTekstur*.
4. Sistem mengembalikan nilai berupa *array fiturTekstur*.



Gambar 4.14 Diagram Alir Proses Perhitungan Fitur Tekstur

Tahapan-tahapan perhitungan manual untuk proses perhitungan fitur tekstur adalah sebagai berikut.

Tahap 1: Inisialisasi *array fiturTekstur*.

Tahap 2: Lakukan perhitungan nilai fitur tekstur menggunakan Persamaan 2.9 sampai Persamaan 2.18. Sebagai contoh, perhitungan nilai fitur tekstur untuk matriks probabilitas pada Tabel 4.11 adalah sebagai berikut.

1. Mean X, dapat dihitung menggunakan Persamaan 2.9.

$$\mu_x = 0(0.15) + 1(0.15) + 2(0.29) + 3(0.27) + 4(0.16)$$
$$= 0 + 0.15 + 0.58 + 0{,}82 + 0.62$$
$$= 2.2$$

2. Mean Y, dapat dihitung menggunakan Persamaan 2.10.

$$\mu_y = 0(0.11) + 1(0.16) + 2(0.29) + 3(0.29) + 4(0.16)$$
$$= 0 + 0.16 + 0.58 + 0{,}87 + 0.64$$
$$= 2.27$$

3. Varians X, dapat dihitung menggunakan Persamaan 2.11.

$$\sigma_x^2 = (0 - 2.2)^2(0.15) + (1 - 2.2)^2(0.15) + (2 - 2.2)^2(0.29) + (3 - 2.2)^2(0.27) + (4 - 2.2)^2(0.16)$$
$$= 4.84(0.15) + 1.44(0.15) + 0.04(0.29) + 0.64(0.27) + 3.24(0.16)$$
$$= 0.7 + 0.21 + 0.01 + 0.17 + 0.53$$
$$= 1.63$$

4. Varians Y, dapat dihitung menggunakan Persamaan 2.12.

$$\sigma_y^2 = (0 - 2.27)^2(0.11) + (1 - 2.27)^2(0.16) + (2 - 2.27)^2(0.29) + (3 - 2.27)^2(0.29) + (4 - 2.27)^2(0.16)$$
$$= 5.17(0.11) + 1.62(0.16) + 0.07(0.29) + 0.53(0.29) + 2.98(0.16)$$
$$= 0.56 + 0.27 + 0.02 + 0.15 + 0.49$$
$$= 1.49$$

5. Kontras, dapat dihitung menggunakan Persamaan 2.13.

$$Kontras = [0(0) + 1(0.04) + 4(0.04) + 9(0.05) + 16(0.02)] + [1(0.02) + 0(0) + 1(0.04) + 4(0.07) + 9(0.02)] + [4(0.02) + 1(0.04) + 0(0.09) + 1(0.09) + 4(0.05)] + [9(0.04) + 4(0.05) + 1(0.11) + 0(0) + 1(0.07)] + [16(0.04) + 9(0.04) + 4(0.02) + 1(0.07) + 0(0)]$$
$$= [0 + 0.04 + 0.15 + 0.49 + 0.29] + [0.02 + 0 + 0.04 + 0.29 + 0.16] + [0.07 + 0.04 + 0 + 0.09 + 0.22] + [0.33 + 0.22 + 0.11 + 0 + 0.07] + [0.58 + 0.33 + 0.07 + 0.07 + 0]$$
$$= 0.96 + 0.51 + 0.42 + 0.73 + 1.05$$
$$= 3.67$$

6. Dissimilarity, dapat dihitung menggunakan Persamaan 2.14.

$$
\begin{aligned}
Disimilarity = {} & [0(0) + 1(0.04) + 2(0.04) + 3(0.05) + 4(0.02)] \\
& + [1(0.02) + 0(0) + 1(0.04) + 2(0.07) + 3(0.02)] \\
& + [2(0.02) + 1(0.04) + 0(0.09) + 1(0.09) + 2(0.05)] \\
& + [3(0.04) + 2(0.05) + 1(0.11) + 0(0) + 1(0.07)] \\
& + [4(0.04) + 3(0.04) + 2(0.02) + 1(0.07) + 0(0)] \\
= {} & [0 + 0.04 + 0.07 + 0.16 + 0.07] \\
& + [0.02 + 0 + 0.04 + 0.15 + 0.05] \\
& + [0.04 + 0.04 + 0 + 0.09 + 0.11] \\
& + [0.11 + 0.11 + 0.11 + 0 + 0.07] \\
& + [0.15 + 0.11 + 0.04 + 0.07 + 0] \\
= {} & 0.35 + 0.25 + 0.27 + 0.4 + 0.36 \\
= {} & 1.64
\end{aligned}
$$

7. Korelasi, dapat dihitung menggunakan Persamaan 2.15.

$$
\begin{aligned}
Korelasi = {} & \big[-2.2[(-2.3)(0) + (-1.3)(0.04) + (-0.3)(0.04) + (0.73)(0.05) \\
& + (1.73)(0.02)] \\
& + (-1.2)[(-2.3)(0.02) + (-1.3)(0) + (-0.3)(0.04) \\
& + (0.73)(0.07) + (1.73)(0.02)] \\
& + (-0.2)[(-2.3)(0.02) + (-1.3)(0.04) + (-0.3)(0.09) \\
& + (0.73)(0.09) + (1.73)(0.05)] \\
& + (0.8)[(-2.3)(0.04) + (-1.3)(0.05) + (0.3)(0.11) \\
& + (0.73)(0) + (1.73)(0.07)] \\
& + (1.8)[(-2.3)(0.04) + (-1.3)(0.04) + (0.3)(0.02) \\
& + (0.73)(0.07) + (1.73)(0)]\big]/(1.63 \times 1.49) \\
= {} & (-2.2)(0.0149) + (-1.2)(0.03306) + (-0.2)(0.0479) \\
& + (0.8)(-0.0562) + (1.8)(-0.08099) \\
= {} & 0.0257 + 0.0571 + 0.0828 - 0.0971 - 0.1399 \\
= {} & -0.071374906
\end{aligned}
$$

8. Homogenitas, dapat dihitung menggunakan Persamaan 2.16.

$$
\begin{aligned}
Homogenity = {} & [(0/1) + (0.04/2) + (0.04/5) + (0.05/10) + (0.02/17)] \\
& + [(0.02/2) + (0/1) + (0.04/2) + (0.07/5) + (0.02/10)] \\
& + [(0.02/5) + (0.04/2) + (0.09/1) + (0.09/2) + (0.05/5)] \\
& + [(0.04/10) + (0.05/5) + (0.11/2) + (0/1) + (0.07/2)] \\
& + [(0.04/16) + (0.04/10) + (0.02/5) + (0.07/2) + (0/1)] \\
= {} & [0 + 0.0182 + 0.0073 + 0.054 + 0.0011] \\
& + [0.0091 + 0 + 0.0182 + 0.146 + 0.0018] \\
& + [0.0036 + 0.0182 + 0.0909 + 0.0455 + 0.0109] \\
& + [0.0039 + 0.0109 + 0.0546 + 0 + 0.0364] \\
& + [0.0021 + 0.0036 + 0.0036 + 0.0364 + 0] \\
= {} & 0.032 + 0.0436 + 0.1691 - 0.1055 - 0.0458 \\
= {} & 0.395935829
\end{aligned}
$$

9. Energi, dapat dihitung menggunakan Persamaan 2.17.

$$
\begin{aligned}
Energi = {} & [(0)^2 + (0.04)^2 + (0.04)^2 + (0.05)^2 + (0.02)^2] \\
& + [(0.02)^2 + (0)^2 + (0.04)^2 + (0.07)^2 + (0.02)^2] \\
& + [(0.02)^2 + (0.04)^2 + (0.09)^2 + (0.09)^2 + (0.05)^2] \\
& + [(0.04)^2 + (0.05)^2 + (0.11)^2 + (0)^2 + (0.07)^2] \\
& + [(0.04)^2 + (0.04)^2 + (0.02)^2 + (0.07)^2 + (0)^2] \\
= {} & [0 + 0.0013 + 0.0013 + 0.003 + 0.0003] \\
& + [0.0003 + 0 + 0.0013 + 0.0053 + 0.0003] \\
& + [0.0003 + 0.0013 + 0 + 0.0083 + 0.003] \\
& + [0.0013 + 0.003 + 0.0119 + 0 + 0.0053] \\
& + [0.0013 + 0.0013 + 0.0003 + 0.0053 + 0] \\
= {} & 0.006 + 0.007 + 0.0212 + 0.0215 + 0.0083 \\
= {} & 0.064132
\end{aligned}
$$

10. Entropi, dapat dihitung menggunakan Persamaan 2.18.

$$
\begin{aligned}
Entropi = {} & -[(0)(-7) + (0.04)(-1.4) + (0.04)(-1.4) + (0.05)(-1.3) \\
& + (0.02)(-1.7)] \\
& + [(0.02)(-1.7) + (0)(-7) + (0.04)(-1.4) + (0.07)(-1.1) \\
& + (0.02)(-1.7)] \\
& + [(0.02)(-1.7) + (0.04)(-1.4) + (0.09)(-1) + (0.09)(-1) \\
& + (0.05)(-1.3)] \\
& + [(0.04)(-1.4) + (0.05)(-1.3) + (0.11)(-1) + (0)(-7) \\
& + (0.07)(-1.1)] \\
& + [(0.04)(-1.4) + (0.04)(-1.4) + (0.02)(-1.7) \\
& + (0.07)(-1.1) + (0)(-7)] \\
= {} & -[-0.2052 - 0.1984 - 0.3422 - 0.309 - 0.2191] \\
= {} & 1.273971305
\end{aligned}
$$

Tahap 3: Pada setiap perhitungan fitur, tambahkan hasil perhitungan ke dalam *array fiturTekstur*. Sehingga nilai dari *array fiturTekstur* adalah sebagai berikut.

$$
fiturTekstur = [2{,}2\ .\ 2.27\ .\ 1{,}63\ .\ 1{,}49\ .\ 3{,}67\ .\ 1{,}64\ .\ -0{,}071374906\ .\ 0{,}395935829\ .\ \ 0{,}064132\ .\ 1{,}273971305]
$$

Tahap 4: Sistem mengembalikan nilai keluaran berupa *array fiturTekstur*.

4.1.4 Ekstraksi Fitur Warna (Momen Warna)

Langkah selanjutnya adalah melakukan ekstraksi fitur warna menggunakan empat momen warna. Proses perhitungan nilai fitur warna secara umum digambarkan dengan Gambar 4.15.

Berdasarkan Gambar 4.15 dapat dijelaskan tahapan-tahapan proses perhitungan *probability matrix* adalah sebagai berikut.

1. Inisialisasi array *fiturWarna* untuk menampung seluruh fitur warna.
2. Sistem mendapatkan masukan berupa citra yang sudah tersegmentasi dalam ruang warna LAB. Sebagai contoh, citra LAB yang sudah tersegmentasi ditunjukkan pada Tabel 4.12.

3. Lakukan perhitungan fitur warna untuk masing-masing channel (L, A dan B) pada citra LAB mengunakan Persamaan 2.20 sampai 2.23 dan tambahkan hasil perhitungan ke *array fiturWarna*.
4. Sistem mengembalikan nilai berupa *array fiturWarna*.

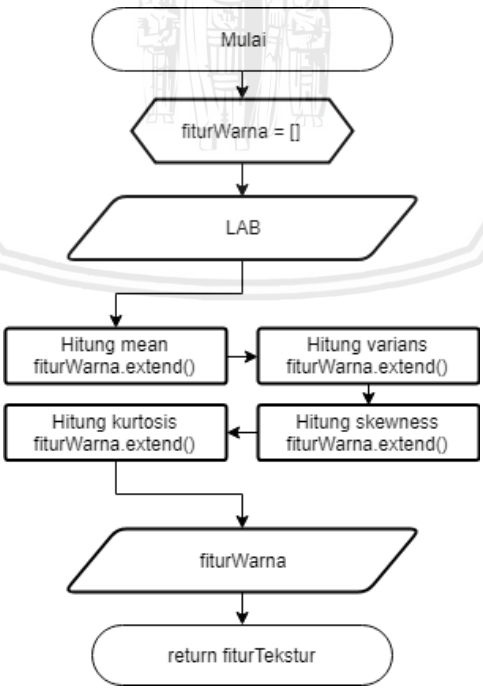
Tahapan-tahapan perhitungan manual untuk proses perhitungan fitur warna adalah sebagai berikut.

Tahap 1: Inisialisasi *array fiturWarna*.

Tahap 2: Lakukan perhitungan fitur warna untuk masing-masing channel (L, A dan B) pada citra LAB mengunakan Persamaan 2.19 sampai 2.22. Sebagai contoh, perhitungan mean, varians, skewness dan kurtosis untuk channel L pada Tabel 4.12 adalah sebagai berikut.

1. Mean

$$E = \frac{1}{25}[98.57 + 97.10 + 93.72 + 87.14 + 86.21 + 94.61 + 87.33 + 84.26 + 85.32 + 80.62 + 83.54 + 84.66 + 82.37 + 80.80 + 80.66 + 85.53 + 79.98 + 80.06 + 82.34 + 83.26 + 80.85 + 81.82 + 83.01 + 82.17 + 84.26]$$
$$= \frac{1}{25} 2130.08$$
$$= 85.212308$$



Gambar 4.15 Diagram Alir Proses Ekstraksi Fitur Warna

Tabel 4.12 Contoh Citra LAB

Posisi		Kolom ke-				
		0	**1**	**2**	**3**	**4**
Baris ke-	**0**	98.57/ -0.22/0.162	97.10/ -0.63/1.275	93.72/ -2.59/7.295	87.14/ -3.83/11.66	86.21/ -5.02/15.38
	1	94.61/ -1.20/4.353	87.33/ -3.93/12.60	84.26/ -5.02/16.16	85.32/ -5.83/18.05	80.62/ -10.4/34.35
	2	83.54/ -5.84/18.91	84.66/ -6.33/19.73	82.37/ -9.60/28.97	80.80/ -13.4/42.80	80.66/ -10.0/32.88
	3	85.53/ -6.40/24.49	79.98/ -12.2/44.81	80.06/ -11.6/41.89	82.34/ -7.33/24.96	83.26/ -6.42/20.87
	4	80.85/ -9.70/39.53	81.82/ -9.79/35.48	83.01/ -7.02/24.63	82.17/ -7.14/25.17	84.26/ -5.61/21.90

2. Varians

$$\sigma = \left[\frac{1}{25}[13.36^2 + 11.89^2 + 8.516^2 + 1.929^2 + 1.002^2 + 9.405^2 + 2.12^2 - 0.947^2 + 0.114^2 - 4.591^2 - 1.663^2 - 0.544^2 - 2.842^2 - 4.406^2 - 4.544^2 + 0.322^2 - 5.224^2 - 5.144^2 - 2.872^2 - 1.944^2 - 4.362^2 - 3.391^2 - 2.198^2 - 3.038^2 - 0.951^2]\right]^{1/2}$$

$$= \left[\frac{1}{25}[178.553 + 141.336 + 72.522 + 3.722 + 1.003 + 88.461 + 4.494 + 0.896 + 0.013 + 21.08 + 2.766 + 0.296 + 8.074 + 19.411 + 20.648 + 0.104 + 27.285 + 26.457 + 8.246 + 3.78 + 19.026 + 11.496 + 4.833 + 9.229 + 0.904]\right]^{1/2}$$

$$= \frac{1}{25} 674.6389 = 26.9855575623$$

3. Skewness

$$s = \left[\frac{1}{25}[13.36^3 + 11.89^3 + 8.516^3 + 1.929^3 + 1.002^3 + 9.405^3 + 2.12^3 - 0.947^3 + 0.114^3 - 4.591^3 - 1.663^3 - 0.544^3 - 2.842^3 - 4.406^3 - 4.544^3 + 0.322^3 - 5.224^3 - 5.144^3 - 2.872^3 - 1.944^3 - 4.362^3 - 3.391^3 - 2.198^3 - 3.038^3 - 0.951^3]\right]^{1/3}$$

$$= \left[\frac{1}{25}[2385.902 + 1680.275 + 617.598 + 7.181 + 1.005 + 832.014 + 9.528 - 0.849 + 0.001 - 96.783 - 4.602 - 0.161 - 22.943 - 85.522 - 93.825 + 0.033 - 142.524 - 139.083 - 23.68 - 7.35 - 82.99 - 38.979 - 10.625 - 28.039 - 0.859]\right]^{1/3}$$

$$= \frac{1}{25} 4747.722$$

$$= 190.30888616855$$

4. Kurtosis

$$K = \left[\frac{1}{25}[13.36^4 + 11.89^4 + 8.516^4 + 1.929^4 + 1.002^4 + 9.405^4 + 2.12^4 - 0.947^4 + 0.114^4 - 4.591^4 - 1.663^4 - 0.544^4 - 2.842^4 - 4.406^4 - 4.544^4 + 0.322^4 - 5.224^4 - 5.144^4 - 2.872^4 - 1.944^4 - 4.362^4 - 3.391^4 - 2.198^4 - 3.038^4 - 0.951^4]\right]^{1/4}$$

$$= \left[\frac{1}{25}[31881.359 + 19975.933 + 5259.458 + 13.855 + 1.006 + 7825.419 + 20.199 + 0.804 + 0.00017 + 444.371 + 7.654 + 0.0875 + 65.192 + 376.793 + 426.34 + 0.0108 + 744.473 + 699.957 + 67.999 + 14.291 + 361.998 + 132.163 + 23.358 + 85.183 + 0.8169]\right]^{1/4}$$

$$= \frac{1}{25} 68428.7212428353$$

$$= 2737.1488497134$$

Tahap 3: Pada setiap perhitungan fitur, tambahkan hasil perhitungan ke dalam *array fituWarnar*. Sehingga nilai dari *array fiturWarna* adalah sebagai berikut.

$$fiturWarna = [85{,}212\ .\ 26{,}986\ .\ 190{,}3089\ .\ 2737{,}1489]$$

Tahap 4: Sistem mengembalikan nilai keluaran berupa *array fiturWarna*.

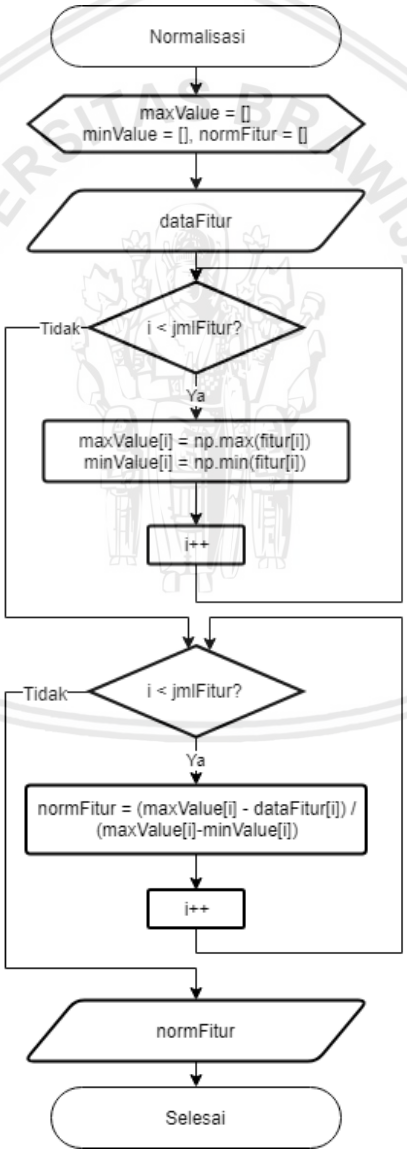
4.1.5 Normalisasi

Langkah selanjutnya adalah melakukan normalisasi untuk seluruh fitur yang telah didapatkan pada proses ekstraksi fitur tekstur dan warna. Proses normalisasi bertujuan untuk menyesuaikan kondisi data sehingga jarak antara satu fitur dengan fitur yang lain tidak terlalu jauh. Normalisasi diharapkan dapat meningkatkan hasil klasifikasi yang akan dilakukan.

Metode normalisasi yang digunakan pada penelitian ini adalah *min-max normalization*. Normalisasi tersebut diterapkan pada setiap fitur. Proses normalisasi secara umum digambarkan dengan Gambar 4.16.

Berdasarkan Gambar 4.16 dapat dijelaskan tahapan-tahapan proses normalisasi adalah sebagai berikut.

1. Inisialisasi array *max* dan *min* untuk menampung nilai maksimal dan minimal untuk masing-masing fitur.
2. Sistem mendapatkan masukan berupa seluruh nilai fitur tekstur dan warna.
3. Lakukan perhitungan nilai maksimal dan minimal untuk setiap fitur dan simpan kedua nilai tersebut pada *array max* dan *min*.
4. Lakukan normalisasi pada seluruh data menggunakan Persamaan 2.1
5. Sistem mengembalikan nilai keluaran berupa seluruh data fitur yang telah dinormalisasi.



Gambar 4.16 Diagram Alir Proses Normalisasi

Tahapan-tahapan perhitungan manual untuk proses normalisasi fitur adalah sebagai berikut.

Tahap 1: Inisialisasi *array max* dan *min* untuk menyimpan nilai maksimal dan minimal dari masing-masing fitur.

Tahap 2: Sistem menerima masukan berupa data seluruh fitur tekstur dan warna dari seluruh citra. Sebagai contoh, terdapat sebuah data fitur yang digambarkan pada Tabel 4.13.

Tahap 3: Lakukan perhitungan nilai maksimal dan minimal untuk masing-masing fitur. Nilai maksimal dan minimal dari data fitur pada Tabel 4.13 ditunjukkan pada Tabel 4.14 dan Tabel 4.15.

Tahap 4: Lakukan perhitungan nilai normalisasi menggunakan Persamaan 2.1. Sebagai contoh, perhitungan normalisasi untuk data citra ke-1 fitur ke-1 adalah sebagai berikut.

$$I_{norm(1,1)} = \frac{I_{(1.1)} - min_1}{max_1 - min_1}$$
$$= \frac{0.000325 - 0.000172}{0.000481 - 0.000172}$$
$$= \frac{0.000153}{0.000309}$$
$$= 0.495145631$$

Tabel 4.13 Contoh Data Fitur

Posisi		Fitur ke-				
		0	1	2	3	4
Citra ke-	0	1653.218	0.000237	0.000237	493.8574	11.6108
	1	3654.851	0.000325	0.000325	8076.66	65.60464
	2	2812.236	0.000481	0.000481	4696.133	47.47972
	3	2573.254	0.000172	0.000172	2269.72	25.7576
	4	1173.441	0.000335	0.000335	827.2181	13.44725

Tabel 4.14 Data Nilai Maksimal Per Fitur

Fitur ke-				
0	1	2	3	4
3654.851	0.000481	0.000481	8076.66	65.60464

Tabel 4.15 Data Nilai Minimal Per Fitur

Fitur ke-				
0	1	2	3	4
1173.441	0.000172	0.000172	493.8574	11.6108

Hasil dari perhitungan normalisasi untuk data fitur pada Tabel 4.13 dapat ditunjukkan pada Tabel 4.16.

Tahap 5: Sistem mengembalikan nilai berupa data fitur yang telah dinormalisasi beserta *array max* dan *min*

Tabel 4.16 Data Fitur Setelah Dinormalisasi

Posisi		Fitur ke-				
		0	**1**	**2**	**3**	**4**
Citra ke-	**0**	0.193349	0.209911	0.209911	0	0
	1	1	0.494489	0.494489	1	1
	2	0.660429	1	1	0.554185	0.664315
	3	0.56412	0	0	0.234196	0.262008
	4	0	0.529541	0.529541	0.043963	0.034012

4.1.6 Klasifikasi LVQ3

Proses klasifikasi LVQ3 merupakan proses inti dari penelitian ini. Proses ini membutuhkan masukan berupa seluruh data fitur yang telah melewati proses seleksi fitur sehingga siap untuk digunakan. Proses LVQ3 secara umum terbagi menjadi dua, yaitu tahap pelatihan dan tahap pengujian.

4.1.6.1 Pelatihan LVQ3

Proses pelatihan LVQ3 bertujuan untuk mengetahui bobot dari masing-masing kelas yang terdapat pada data latih. Masukan dari proses ini adalah data latih yang berupa fitur dari masing-masing citra telah diketahui kelasnya. Keluaran dari proses ini adalah bobot masing-masing kelas yang terdapat pada data latih.

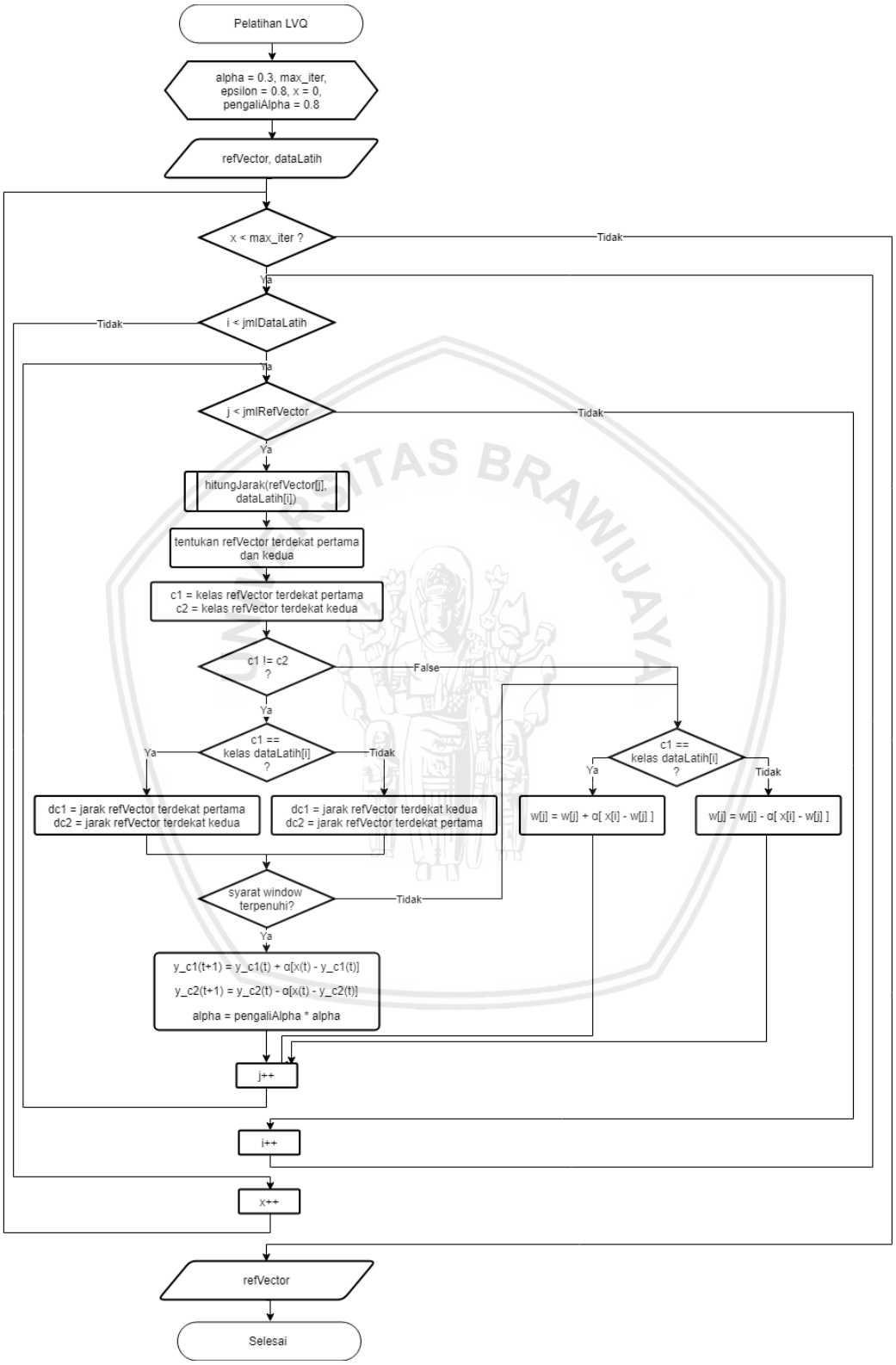
Metode LVQ yang digunakan pada penelitian ini adalah LVQ3. Proses pelatihan LVQ3 secara umum digambarkan dengan Gambar 4.17.

Berdasarkan Gambar 4.17 dapat dijelaskan tahapan-tahapan proses pelatihan LVQ3.0 adalah sebagai berikut.

1. Inisialisasi bobot pada lapisan kompetisi (w_j) dengan salah satu data latih pada masing-masing kelas dan tentukan nilai α (*learning rate*).
2. Selama kondisi belum terpenuhi, lakukan langkah 3 sampai 9
3. Untuk setiap pasangan data latih beserta targetnya ($x_i : t_i$), lakukan langkah 4 sampai 7.
4. Tentukan neuron pemenang pertama dan neuron pemenang kedua pada lapisan kompetisi yang memiliki jarak *Euclidean* terkecil dengan x_i.

5. Jika jarak dari data latih ke neuron pemenang pertama dan jarak dari data latih ke neuron pemenang kedua menunjukkan nilai yang hampir

sama, maka lakukan langkah 6. Kondisi tersebut dapat digambarkan dengan Persamaan 2.29.



Gambar 4.17 Diagram Alir Proses Pelatihan LVQ3

6. Jika dua neuron pemenang / terdekat (y_{c1} dan y_{c2}) memiliki kelas yang sama dengan data latih, maka ubah bobot kedua neuron pemenang menggunakan Persamaan 2.30.
7. Jika syarat pada langkah 6 tidak terpenuhi, maka cek apakah dua neuron pemenang / terdekat memenuhi kondisi dimana salah satu neuron (y_{c1}) memiliki kelas yang sama dengan data latih dan yang lain (y_{c2}) menunjukkan kelas yang berbeda dengan target data latih. Urutan dari y_{c1} dan y_{c2} tidak diperhatikan. Jika memenuhi syarat tersebut lakukan langkah 8
8. Ubah bobot *neuron* pemenang yang memiliki kelas yang sama dengan target data latih menggunakan Persamaan 2.31. Kemudian lakukan update bobot neuron pemenang yang memiliki kelas yang berbeda dengan target data latih menggunakan Persamaan 2.32.
9. Jika syarat pada langkah 7 tidak terpenuhi, maka ubah bobot neuron pemenang (w_j) sebagai berikut.
    a. Jika $c_j \neq t_j$ maka ubah bobot menggunakan Persamaan 2.33.
    b. Jika $c_j = t_j$ maka maka ubah bobot menggunakan Persamaan 2.34.
10. Kurangi nilai α
11. Lakukan pengujian kondisi berhenti. Iterasi dihentikan setelah mencapai epoch tertentu atau setelah nilai α mencapai nilai yang kecil.

Tahapan-tahapan perhitungan manual untuk proses pelatihan LVQ3.0 adalah sebagai berikut.

Tahap 1: Inisialisasi nilai dari *reference vector* menggunakan salah satu data latih pertama pada masing-masing kelas. Jumlah dan kelas dari *reference vector* juga menyesuaikan dengan data latih pertama pada masing-masing kelas. Sebagai contoh, nilai *reference vector* beserta kelasnya dapat ditunjukkan pada Tabel 4.17.

Tahap 2: Inisialisasi data latih beserta kelasnya. Data latih yang digunakan merupakan data latih yang tidak memuat *reference vector* di dalamnya. Sebagai contoh data latih dapat ditunjukkan pada Tabel 4.18.

Tabel 4.17 *Reference vector* beserta kelasnya

No	Id Gambar	Nilai Fitur				Kelas
		Fitur 1	Fitur 2	Fitur 3	Fitur 4	
1	001_0001	0.090651	0.00087	0.00087	0.005599	1
2	002_0001	0.233624	0.001605	0.001605	0.137817	2
3	003_0001	0.156368	0.000327	0.000327	0.036564	3
4	004_0001	0.056382	0.001695	0.001695	0.011412	4

Tabel 4.18 Data Latih beserta kelasnya

No	Id Gambar	Nilai Fitur				Kelas
		Fitur 1	Fitur 2	Fitur 3	Fitur 4	
1	001_0002	0.087566	0.001022	0.001022	0.004534	1
2	001_0003	0.088482	0.001114	0.001114	0.004572	1
3	002_0002	0.210601	0.001481	0.001481	0.118048	2
4	002_0003	0.237756	0.001845	0.001845	0.144137	2
5	003_0002	0.159305	0.000336	0.000336	0.037876	3
6	003_0003	0.156209	0.000312	0.000312	0.035699	3
7	004_0002	0.052965	0.001667	0.001667	0.009161	4
8	004_0003	0.056274	0.00169	0.00169	0.011186	4

Tahap 3: Lakukan Tahap 3 sampai Tahap 7 dengan iterasi sejumlah data latih. Lakukan perhitungan jarak *euclidean* antara data latih dengan masing-masing *reference vector* menggunakan Persamaan 2.28. Sebagai contoh, perhitungan jarak antara data latih dengan *ID 001_0003* dengan reference vector *001_0001* adalah sebagai berikut.

$$EDist = \sqrt{\begin{matrix}(0.088482 - 0.090651)^2 + (0.001114 - 0.00087)^2 + \\ (0.001114 - 0.00087)^2 + (0.004572 - 0.005599)^2\end{matrix}}$$

$$= \sqrt{\begin{matrix}(-8.612 \times 10^{-3})^2 + \left(-4.9 \times 10^{-5}\right)^2 + \\ \left(-4.9 \times 10^{-5}\right)^2 + (-1.44 \times 10^{-3})^2\end{matrix}}$$

$$= \sqrt{\begin{matrix}4.7047 \times 10^{-6} + 5.98 \times 10^{-8} + \\ 5.98 \times 10^{-8} + 1.08 \times 10^{-6}\end{matrix}}$$

$$= \sqrt{5.87866 \times 10^{-6}}$$

$$= 0.002424596$$

Hasil perhitungan jarak *euclidean* antara data latih pada Tabel 4.18 dan *reference vector* pada Tabel 4.17 ditunjukkan pada Tabel 4.19.

Tabel 4.19 Jarak Euclidean antara data latih dan reference vector

No	Data Latih	Reference Vector			
		001_0001	**002_0001**	**003_0001**	**004_0001**
1	001_0003	0.002425	0.19703	0.075054	0.032831

Tahap 4: Tentukan *reference vector* yang memiliki jarak terdekat pertama dan kedua dari masing-masing data latih. Sebagai contoh, reference vector terdekat untuk data latih 001_003 dari Tabel 4.17 ditunjukkan dengan Tabel 4.20.

Tabel 4.20 Reference Vector terdekat

No	Data Latih	Reference Vector Terdekat	
		Pertama	Kedua
1	001_0003	(001_0001) 0.002425	(004_0001) 0.032831

Tahap 5: Lakukan pengecekan terhadap nilai *window* menggunakan Persamaan 2.29. Nilai ϵ yang digunakan adalah 0,2. Sebagai contoh, perhitungan nilai window untuk data latih 001_003 adalah sebagai berikut.

$$d_{c1} = 0.002425$$

$$d_{c2} = 0.032831$$

$$\min\left[\frac{0.002425}{0.032831}.\frac{0.032831}{0.002425}\right] > (1-0.2)(1+0.2)$$

$$\min[0.073863117.\,13.5385567] > 0.8 \times 1.2$$

$$0.073863117 > 0.96$$

Dikarenakan 0,184565818 > 0.96 dapat disimpulkan bahwa jarak antara *reference vector* terdekat pertama dan kedua tidak memenuhi syarat *window*.

Tahap 6: Jika kondisi window pada Tahap 5 terpenuhi, maka lakukan perubahan nilai bobot menggunakan Persamaan 2.30. Sebagai contoh, perhitungan bobot fitur pertama untuk *reference vector* 001_0001 untuk data latih 003_0001 dari Tabel 4.18 adalah sebagai berikut (dengan asumsi kondisi window terpenuhi).

$$\begin{aligned}\beta &= 0{,}4\alpha \\ &= 0{,}4 \times 0{,}1 \\ &= 0{,}04\end{aligned}$$

$$\begin{aligned}y_c(\mathrm{t}+1) &= y_c(\mathrm{t}) + \beta[\mathrm{x}(\mathrm{t}) - y_c(\mathrm{t})] \\ &= 0.090651 + 0.04[0.088482 - 0.090651] \\ &= 0.090651 + 0.04(-0.002169) \\ &= 0.090651 - 0.00008676 \\ &= 0.09056424\end{aligned}$$

Tahap 7: Jika kondisi pada Tahap 6 tidak terpenuhi, maka lakukan perubahan bobot menggunakan Persamaan 2.31 dan Persamaan 2.32. Nilai α yang digunakan adalah 0,01. Sebagai contoh, perhitungan bobot fitur pertama untuk *reference vector* 001_0001 dan 002_0001 untuk data latih 001_003 dari Tabel 4.18 adalah sebagai berikut.

$$\boldsymbol{y_{c1}(t) = 0.090651}$$

$$\boldsymbol{y_{c2}(t) = 0.233624}$$

$$\boldsymbol{x(t) = 0.088482}$$

$$\begin{aligned} \boldsymbol{y_{c1}(t+1)} &= \boldsymbol{y_{c1}(t) + \alpha[x(t) - y_{c1}(t)]} \\ &= \boldsymbol{0.090651 + 0.01[0.088482 - 0.090651]} \\ &= \boldsymbol{0.090651 + 0.01(-0.002169)} \\ &= \boldsymbol{0.090651 - 0.00002169} \\ &= \boldsymbol{0.09062931} \end{aligned}$$

$$\begin{aligned} y_{c2}(t+1) &= y_{c2}(t) - \alpha[x(t) - y_{c2}(t)] \\ &= 0.233624 - 0.01[0.088482 - 0.145142] \\ &= 0.233624 - 0.01(-0.145142) \\ &= 0.233624 - (-0.00145142) \\ &= 0.23364569 \end{aligned}$$

Tahap 8: Lakukan perubahan nilai alpha dengan mengalikan nilai alpha dengan pengali *learning rate* yang telah ditentukan sebelumnya. Sebagai contoh, perhitungan nilai alpha yang baru adalah sebagai berikut.

$$\begin{aligned} \boldsymbol{\alpha(t)} &= \boldsymbol{0.8\alpha(t)} \\ &= \boldsymbol{0.8 \times 0.01} \\ &= \boldsymbol{0.008} \end{aligned}$$

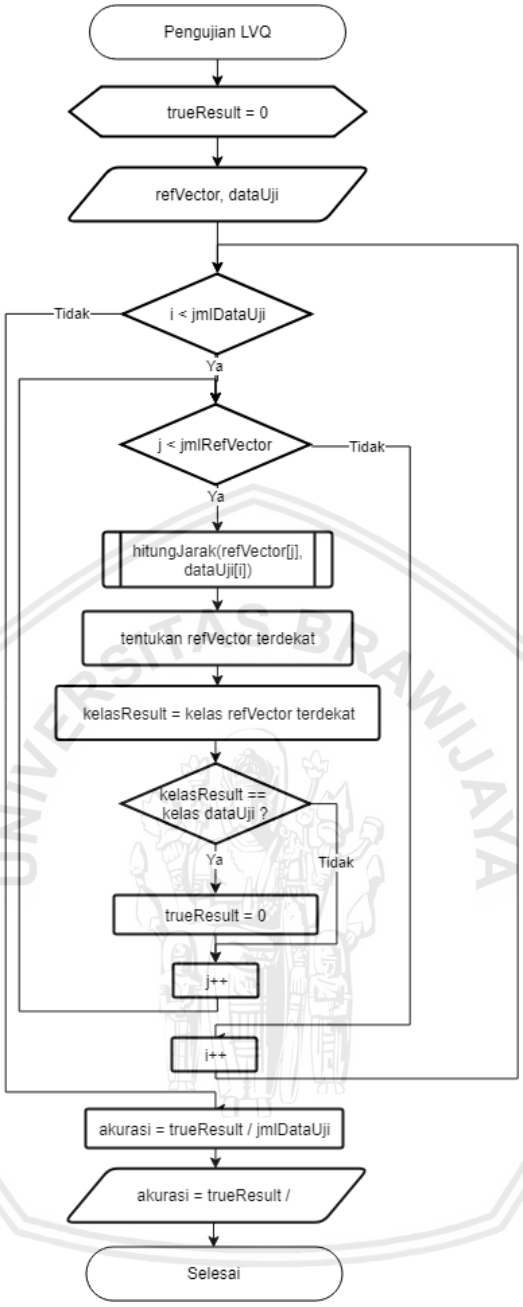
Tahap 9: Lakukan pengujian kondisi berhenti. Iterasi dihentikan setelah mencapai iterasi tertentu atau setelah nilai alpha mencapai nilai yang kecil.

4.1.6.2 Pengujian LVQ3

Proses pengujian LVQ3 bertujuan untuk mengetahui hasil akurasi dari klasifikasi LVQ3. Proses ini membutuhkan masukan berupa bobot masing-masing kelas serta kumpulan data uji yang berbeda dari data latih.

Berdasarkan Gambar 4.18 dapat dijelaskan tahapan-tahapan proses pengujian LVQ3.0 adalah sebagai berikut.

1. Inisialisasi bobot dari masing-masing *reference vector* yang telah didapatkan dari proses pelatihan.
2. Inisialisasi data uji yang telah dinormalisasi menggunakan metode *min-max normalization*.
3. Lakukan perhitungan jarak *euclidean* antara masing-masing data uji dan masing-masing *reference vector*.
4. Tentukan kelas hasil pengujian dengan mengacu pada kelas reference vector terdekat untuk masing-masing data uji.
5. Hitung nilai akurasi dari proses pengujian.



Gambar 4.18 Diagram Alir Pengujian LVQ3

Tahapan-tahapan perhitungan manual untuk proses pengujian LVQ3.0 adalah sebagai berikut.

Tahap 1: Inisialisasi bobot dari *reference vector* yang telah didapatkan dari proses pelatihan. Sebagai contoh, didapatkan nilai bobot terakhir yang ditunjukkan pada Tabel 4.21.

Tahap 2: Inisialisasi data uji yang telah dinormalisasi menggunakan metode min-max normalization. Sebagai contoh, didapatkan nilai data uji yang telah dinormalisasi dan ditunjukkan pada Tabel 4.22.

Tabel 4.21 Bobot Hasil Pelatihan LVQ3

No	Id RefVector	Nilai Fitur				Kelas
		Fitur 1	Fitur 2	Fitur 3	Fitur 4	
1	001_0001	0.090651	0.00087	0.00087	0.005599	1
2	002_0001	0.233624	0.001605	0.001605	0.137817	2
3	003_0001	0.156368	0.000327	0.000327	0.036564	3
4	004_0001	0.056382	0.001695	0.001695	0.011412	4

Tabel 4.22 Data Uji Ternormalisasi

No	Id dataUji	Nilai Fitur				Kelas
		Fitur 1	Fitur 2	Fitur 3	Fitur 4	
1	004_0003	0.052965	0.001667	0.001667	0.009161	4
2	001_0003	0.114599	0.000524	0.000524	0.013553	1
3	002_0003	0.115274	0.000424	0.000424	0.025795	2
4	003_0003	0.059574	0.001424	0.001424	0.021694	3

Tahap 3: Lakukan perhitungan jarak *Euclidean* antara masing-masing data uji dan masing-masing reference vector. Sebagai contoh, perhitungan antara data uji 004_0003 pada Tabel 4.22 dengan refVector 001_0001 adalah sebagai berikut.

$$EDist = \sqrt{\begin{matrix}(0.052965 - 0.090651)^2 + (0.001667 - 0.00087)^2 + \\ (0.001667 - 0.00087)^2 + (0.009161 - 0.005599)^2\end{matrix}}$$

$$= \sqrt{\begin{matrix}(-0.037686)^2 + (0.000797)^2 + \\ (0.000797)^2 + (0.003562)^2\end{matrix}}$$

$$= \sqrt{\begin{matrix}0.001420212 + 6.34662 \times 10^{-7} + \\ 6.34662 \times 10^{-7} + 1.3 \times 10^{-5}\end{matrix}}$$

$$= \sqrt{0.001434}$$

$$= 0.037871$$

Sehingga didapatkan jarak antara masing-masing data uji dengan masing-masing *reference vector* yang ditunjukkan pada Tabel 4.23.

Tahap 4: Tentukan kelas hasil pengujian dengan mengacu pada kelas reference vector terdekat untuk masing-masing data uji. Sebagai contoh, citra dan kelas hasil pengujian untuk data pada Tabel 4.22 ditunjukkan dengan Tabel 4.24.

Tabel 4.23 Jarak Euclidean antara Data Uji dan Refence Vector

No	Id dataUji	Reference Vector				Kelas
		001_0001	002_0001	003_0001	004_0001	
1	004_0003	0.037871	0.221788	0.106989	0.004092	4
2	001_0003	0.025239	0.172078	0.047689	0.05828	1
3	002_0003	0.031852	0.162968	0.042482	0.06065	2
4	003_0003	0.035006	0.209232	0.097942	0.010773	3

Tahap 5: Hitung nilai akurasi dari proses pengujian menggunakan Persamaan 2.34. Sebagai contoh, nilai akurasi untuk data pada Tabel 4.24 adalah sebagai berikut:

$$Akurasi = \frac{jumlah\ data\ yang\ relevan}{jumlah\ data\ keseluruhan} \times 100\%$$
$$= \frac{1}{4} \times 100\% = 25\%$$

Tabel 4.24 Citra dan kelas Hasil Pengujian

No	Id dataUji	Kelas Hasil	Kelas Target
1	004_0003	 4 (Donat)	 4 (Donat)
2	001_0003	 3 (Stroberi)	 1 (Tomat)
3	002_0003	 1 (Tomat)	 2 (Jeruk Oranye)

Tabel 4.24 Citra dan kelas Hasil Pengujian (lanjutan)

No	Id dataUji	Kelas Hasil	Kelas Target
4	003_0003	 4 (Donat)	 3 (Genji Pie)

4.2 Perancangan Antarmuka

Perancangan antarmuka ini berguna untuk menggambarkan implementasi dari klasifikasi citra makanan berdasarkan ekstraksi fitur *Haralick* dan *CIE Lab Color Moment* menggunakan *Learning Vector Quantization* pada platform web.

4.2.1 Perancangan Antarmuka Halaman Beranda

Perancangan antarmuka beranda ini berguna untuk merancang tampilan beranda dari sistem. Halaman beranda berisi beberapa judul sistem dan tombol navigasi pada masing-masing menu, yaitu Beranda, Klasifikasi, Data dan Pengembang. Menu beranda digunakan untuk menuju beranda sistem. Menu klasifikasi digunakan untuk melakukan tahapan klasifikasi pada sistem. Menu data digunakan untuk menampilkan kelas data yang tersedia pada sistem. Sedangkan menu pengembang digunakan untuk menampilkan profil pengembang sistem. Gambar menunjukkan perancangan dari halaman beranda.

4.2.2 Perancangan Antarmuka Section Klasifikasi

Perancangan antarmuka klasifikasi ini berguna untuk merancang tampilan *section* klasifikasi dari sistem. Section *klasifikasi* ini berisi empat tahapan utama dalam proses klasifikasi, yaitu unggah citra, segmentasi, ekstraksi fitur dan klasifikasi. Proses unggah citra berfungsi untuk mengunggah citra yang akan diklasifikasi untuk kemudian di simpan oleh sistem. Proses segmentasi berfungsi untuk melakukan segmentasi pada citra yang telah disimpan oleh sistem. Proses ekstraksi fitur berfungsi untuk melakukan ekstraksi fitur pada citra yang telah disegmentasi sebelumnya. Proses klasifikasi digunakan untuk melakukan proses klasifikasi melalui perhitungan jarak pada masing-masing *reference vector*.

Gambar 4.19 Perancangan Antarmuka Beranda

Gambar dan Gambar menunjukkan perancangan dari *section* klasifikasi. Masing-masing tahapan sistem dilakukan satu per satu. Sebagai contoh, sebelum melakukan proses pengunggahan citra, tombol segmentasi, ekstraksi dan klasifikasi berada pada kondisi *disabled* hingga pengguna melakukan proses simpan citra.

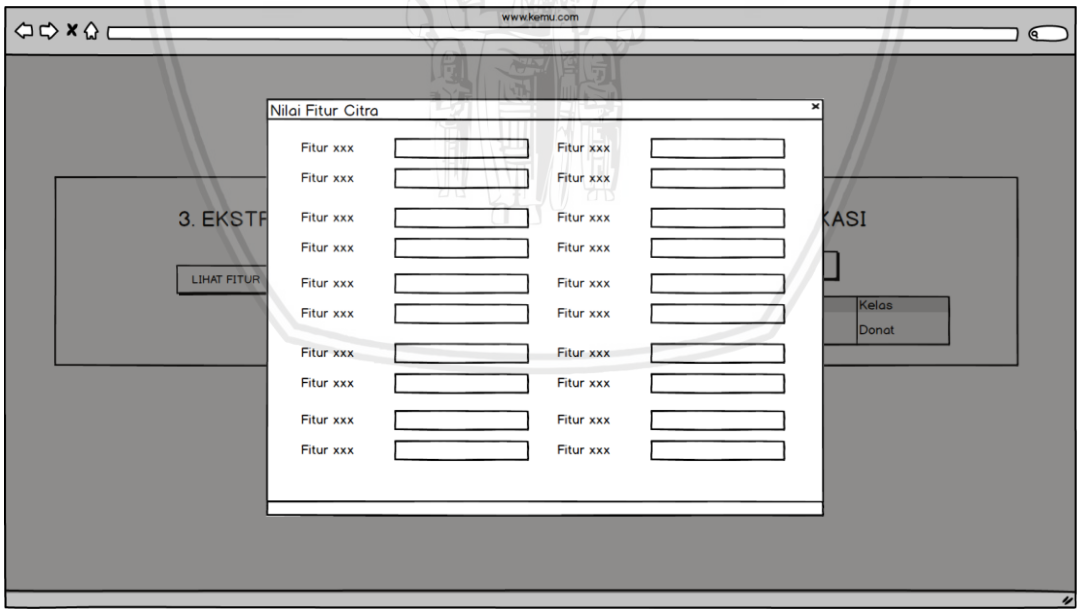
Gambar 4.20 Perancangan Antarmuka Section Klasifikasi untuk Proses Unggah Citra dan Segmentasi

4.2.3 Perancangan Antarmuka Modal Screen Nilai Fitur

Perancangan antarmuka nilai fitur berfungsi untuk merancang tampilan hasil ekstraksi ftur dari citra yang akan diproses oleh sistem. Nilai fitur ditampilkan pada *text field* sejumlah jenis fitur dengan didahului oleh *label* yang bertuliskan jenis

fitur yang dimaksud. Seluruh *text field* dan label ditampilkan pada sebuah modal screen yang akan muncul dengan menekan tombol "LIHAT FITUR". Sebelum melakukan proses ekstraksi fitur, nilai pada masing-masing *text field* berisi "Belum Ditentukan". Gambar menunjukkan perancangan dari *modal screen* nilai fitur.

Gambar 4.21 Perancangan Antarmuka Section Klasifikasi untuk Proses Ekstraksi Fitur dan Klasifikasi



Gambar 4.22 Perancangan Antarmuka Nilai Fitur

4.3 Skenario Pengujian

Bagian ini membahas mengenai perancangan skenario pengujian yang akan dilakukan pada penelitian ini. Pengujian dibutuhkan untuk mengetahui bagaimana

kualitas dari hasil penelitian ini. Daftar skenario pengujian yang akan dilakukan adalah sebagai berikut.

1. Skenario Pengujian Learning Rate
2. Skenario Pengujian Pengali Learning Rate
3. Skenario Pengujian Nilai m
4. Skenario Pengujian Epsilon
5. Skenario Pengujian Maksimum Iterasi
6. Skenario Pengujian Learning Rate Minimal
7. Skenario Pengujian Akurasi

Nilai *k* yang digunakan pada pengujian k-fold cross validation adalah sebesar 10. Selain itu, sebelum melaksanakan beberapa pengujian di atas, dibutuhkan sebuah kondisi parameter awal yang akan dijadikan acuan pada saat pengujian. Kondisi parameter awal tersebut ditunjukkan pada Tabel 4.25.

Tabel 4.25 Kondisi Awal Parameter

No	Nama Parameter	Nilai
1	Proporsi Data Latih dan Data Uji	80:20
2	Learning rate (Alpha)	0,2
3	Pengali learning rate	0,8
4	Nilai m	0,1
5	Epsilon	0,4
6	Iterasi maksimal	100
7	Learning rate minimal	0,0000001

4.3.1 Skenario Pengujian Learning Rate

Skenario pengujian learning rate secara umum digambarkan dengan Tabel 4.26.

Tabel 4.26 Skenario Pengujian Learning Rate

No	Learning rate	Nilai Akurasi
1	0,05	
2	0,1	
3	0,15	
4	0,2	
5	0,25	
6	0,3	

4.3.2 Skenario Pengujian Pengali Learning Rate

Skenario pengujian pengali learning rate secara umum digambarkan dengan Tabel 4.27.

Tabel 4.27 Skenario Pengujian Pengali Learning Rate

No	Pengali learning rate	Nilai Akurasi
1	0,1	
2	0,2	
3	0,3	
4	0,4	
5	0,5	
6	0,6	
7	0,7	
8	0,8	
9	0,9	

4.3.3 Skenario Pengujian Nilai m

Skenario pengujian nilai *m* secara umum digambarkan dengan Tabel 4.27.

Tabel 4.28 Skenario Pengujian Nilai m

No	Nilai m	Nilai Akurasi
1	0,05	
2	0,1	
3	0,15	
4	0,2	
5	0,25	
6	0,3	
7	0,35	
8	0,4	

4.3.4 Skenario Pengujian Epsilon

Skenario pengujian *epsilon* secara umum digambarkan dengan Tabel 4.27.

Tabel 4.29 Skenario Pengujian Epsilon

No	Epsilon	Nilai Akurasi
1	0,1	
2	0,2	
3	0,3	

4	0,4	
5	0,5	
6	0,6	
7	0,7	
8	0,8	
9	0,9	

4.3.5 Skenario Pengujian Iterasi Maksimal

Skenario pengujian iterasi maksimal secara umum digambarkan dengan Tabel 4.30.

Tabel 4.30 Skenario Pengujian Iterasi Maksimal

No	Pengali learning rate	Nilai Akurasi
1	10	
2	20	
3	30	
4	40	
5	50	
6	60	

4.3.6 Skenario Pengujian Learning Rate Minimal

Skenario pengujian learning rate minimal secara umum digambarkan dengan Tabel 4.31.

Tabel 4.31 Skenario Pengujian Learning Rate Minimal

No	Pengali learning rate	Nilai Akurasi
1	0,001	
2	0,0001	
3	0,00001	
4	0,000001	
5	0,0000001	
6	0,00000001	

4.3.7 Skenario Pengujian Akurasi

Skenario pengujian akurasi dilakukan menggunakan nilai parameter yang memiliki nilai akurasi terbesar pada masing-masing pengujian. Pengujian dilakukan pada tiga jenis fitur, yaitu fitur tekstur, fitur warna dan kedua fitur tekstur dan warna.

BAB 5 IMPLEMENTASI

Bab ini membahas tentang implementasi sistem estimasi berdasarkan perancangan yang telah disusun pada bab sebelumnya.

5.1 Implementasi Sistem

Subbab ini membahas tentang implementasi sistem berdasarkan perancangan yang telah dilakukan sebelumnya. Sistem ini menggunakan bahasa pemrograman Python pada proses pelatihan dan pengujian.

5.1.1 Implementasi Proses Inisialisasi Data Latih dan Data Uji

Inisialisasi data latih dan data uji bertujuan untuk memasukkan keseluruhan data ke dalam sistem dan membaginya dalam keadaan terpisah antara *reference vector*, data latih dan data uji. Proses inisialisasi data latih dan data uji mengacu pada Subbab 4.1.1. Masukan dari proses ini adalah seluruh data citra. Keluaran dari proses ini adalah data citra yang telah terbagi menjadi data *reference vector,* data latih dan data uji. Proses inisialisasi data latih dan data uji secara lengkap dijelaskan dengan Kode Sumber 1.

Kode Sumber 1: Proses Inisialisasi Data Latih dan Data Uji

```
1   import csv
2   import numpy as np
3   def read_csv(file_name):
4     array_2D = []
5     with open(file_name, 'rb') as csvfile:
6       read = csv.reader(csvfile, delimiter=';')
7       for row in read:
8          array_2D.append(map(int,row))
9     return array_2D
10  data1 = read_csv('data/pengujian134/dataTrain134.csv'
11  data2 = read_csv('data/pengujian134/dataClass134.csv')
12  data3 = read_csv('data/pengujian134/dataTest134.csv')
13  dataTrain = ((np.array(data1[:]))[:,1:-
      1]).astype(np.float16).tolist()
14  dataC = ((np.array(data2[:]))[:,1:-
      1]).astype(np.float16).tolist()
15  dataT = ((np.array(data3[:]))[:,1:-
      1]).astype(np.float16).tolist()
16  classDataTrain = ((np.array(data1[:]))[:,-
      1:]).astype(int).tolist()
17  classDataClass = ((np.array(data2[:]))[:,-
      1:]).astype(int).tolist()
18  classDataTest = ((np.array(data3[:]))[:,-
      1:]).astype(int).tolist()
```

Penjelasan:

1. Baris ke-1 sampai ke-2 berfungsi untuk meng-*import library csv* dan *numpy*.
2. Baris ke-3 sampai ke-9 adalah method *read_csv* yang berfungsi untuk membaca file csv dan mengembalikan nilai berupa array berisi data citra.

3. Baris ke-10 sampai ke-13 adalah proses pemanggilan method read_csv untuk file data latih, *vector reference* dan data uji. Hasil dari kembalian method *read_csv* disimpan pada variabel *data1*, *data2* dan *data3*.
4. Baris ke-13 sampai ke-15 berfungsi untuk membuat array *dataTrain*, *dataC* dan *dataT* untuk menyimpain nilai fitur dari data latih, *vector reference* dan data uji.
5. Baris ke-16 sampai ke-18 berfungi untuk membuat array *classDataTrain*, *classDataClass* dan *classDataTest* untuk menyimpain kelas dari masing-masing data latih, *vector reference* dan data uji.

5.1.2 Implementasi Proses Preprosesing

Preprosesing bertujuan untuk menghilangkan bagian-bagian yang tidak diperlukan pada citra untuk proses selanjutnya. Proses preprosesing pada penelitian ini terdiri dari transformasi warna RGB to grayscale, transformasi warna RGB to LAB dan segmentasi.

5.1.2.1 Implementasi Transformasi Warna RGB to Grayscale

Proses transformasi warna RGB to grayscale bertujuan untuk mengubah nilai intensitas citra RGB menjadi nilai citra keabuan. Proses transformasi warna RGB to *grayscale* mengacu pada Subbab 4.1.2.1 Masukan pada proses ini adalah nilai intensitas citra pada ketiga channel RGB. Keluaran dari proses ini adalah nilai intensitas citra grayscale. Proses transformasi warna RGB to *grayscale* secara lengkap dijelaskan dengan Kode Sumber 2.

Kode Sumber 2: Proses Transformasi Warna RGB to Grayscale

```
1  import cv2
2  strFile = 001_0001.jpg'
3  rgbImg = cv2.imread(strFile)
4  grayImg = RGBtoGray(rgbImg)
5  def RGBtoGray(rgbImg):
6     grayImg = np.zeros_like(rgbImg[:,:,0])
7     for i in range(len(grayImg)):
8        for j in range(len(grayImg[i])):
9           grayImg[i][j] = 0.299*rgbImg[i][j][2] +
   0.587*rgbImg[i][j][1] + 0.114*rgbImg[i][j][0]
10    return grayImg
```

Penjelasan:

1. Baris ke-1 berfungsi untuk meng-*import library cv2.*
2. Baris ke-2 sampai ke-3 berfungsi untuk membaca file citra dengan nama file *001_0001.jpg* dan disimpan pada variabel *rgbImg.*
3. Baris ke-4 adalah proses pemanggilan method *RGBtoGray* dengan argumen berupa variabel *rgbImg*.
4. Baris ke-5 adalah inisialisasi method RGBtoGray dengan parameter citra *grayscale.*
5. Baris ke-6 berfungsi untuk membuat array kosong bernama *grayImg* dengan ukuran yang sama dengan array *rgbImg*.

6. Baris ke-7 sampai ke-9 adalah perulangan untuk masing-masing piksel array rgbImg yang berisi operasi transformasi RGB menjadi grayscale menggunakan Persamaan 2.6.
7. Baris ke-10 bertujuan untuk mengembalikan nilai dengan variabel *grayImg.*

5.1.2.2 Implementasi Transformasi Warna RGB to LAB

Proses transformasi warna RGB to LAB bertujuan untuk mengubah nilai intensitas citra dalam ruang warna RGB menjadi nilai citra dalam ruang warna LAB. Proses transformasi warna RGB to LAB mengacu pada Subbab 4.1.2.2. Masukan pada proses ini adalah nilai intensitas citra pada ketiga channel RGB. Keluaran dari proses ini adalah nilai intensitas citra pada ruang warna LAB. Proses transformasi warna RGB to LAB secara lengkap dijelaskan dengan Kode Sumber 3.

Kode Sumber 3: Proses Transformasi Warna RGB to LAB

```
1  import cv2
2  import numpy as np
3  def normalize(channel):
4     for i in range(len(channel)):
5        for j in range(len(channel[i])):
6           channel[i][j] = channel[i][j] / 255
7     return channel
8  def func(t):
9     if(t > th):
10       return np.cbrt(t)
11    else:
12       return 7.787*t + np.divide(16.0,116.0)
13 def convBGRtoLAB(rgbImg):
14    rgbImgFloat = rgbImg.astype(np.float64)
15    blueNorm = np.zeros_like(rgbImgFloat[:,:,0])
16    greenNorm = np.zeros_like(rgbImgFloat[:,:,1])
17    redNorm = np.zeros_like(rgbImgFloat[:,:,2])
18    blueNorm = normalize(rgbImgFloat[:,:,0])
19    greenNorm = normalize(rgbImgFloat[:,:,1])
20    redNorm = normalize(rgbImgFloat[:,:,2])
21    merged = cv2.merge((redNorm,greenNorm,blueNorm))
22    matriksKonv = np.array([[0.412453,0.357580,0.180423],
   [0.212671,0.715160,0.072169],[0.019334,0.119193,0.950227]])
23    Xn = 0.950456
24    Zn = 1.088754
25    xyz = merged.copy()
26    for i in range(len(xyz)):
27       for j in range(len(xyz[i])):
28          xyz[i][j] = np.matmul(matriksKonv, xyz[i][j])
29          xyz[i][j][0] = xyz[i][j][0]/Xn
30          xyz[i][j][2] = xyz[i][j][2]/Zn
31    Lab = np.zeros_like(xyz)
32    for i in range(len(Lab)):
33       for j in range(len(Lab[i])):
34          if(xyz[i][j][1] > th):
35             Lab[i][j][0] = (116*np.cbrt(xyz[i][j][1]))-16
36          else:
37             Lab[i][j][0] = 903.3 * xyz[i][j][1]
38          Lab[i][j][1]  =  500  *  (func(xyz[i][j][0])  -
   func(xyz[i][j][1]))
```

```
39          Lab[i][j][2] = 200 * (func(xyz[i][j][1]) -
    func(xyz[i][j][2]))
40      return Lab

41  strFile = 001_0001.jpg'
42  rgbImg = cv2.imread(strFile)
43  Lab = convBGRtoLAB(rgbImg)
```

Penjelasan:

1. Baris ke-1 dan ke-2 berfungsi untuk meng-*import library cv2* dan *numpy.*
2. Baris ke-3 sampai ke-7 adalah method *normalize* dengan parameter nilai piksel pada *channel* tertentu. Method *normalize* berfungsi untuk melakukan normalisasi pada piksel RGB menggunakan Persamaan 2.2. Nilai kembalian dari method *normalize* adalah nilai piksel yang telah dinormalisasi pada *channel* tertentu.
3. Baris ke-8 sampai ke 12 adalah method *func* dengan parameter berupa variabel *t*. Method *func* berfungsi untuk mendapatkan nilai dari fungsi *f(t)* pada Persamaan 2.23. Nilai kembalian dari method *func* adalah hasil perhitungan fungsi *f(t)* dengan nilai *t* tertentu.
4. Baris ke-13 adalah inisialisasi method *convRGBtoLAB* dengan parameter berupa array citra *rgbImg*.
5. Baris ke-14 berfungsi untuk melakukan konversi tipe data dari citra *rgbImg* dan disimpan pada variabel *rgbImgFloat*. Konversi tipe data dilakukan dari tipe data *integer* menjadi *float*.
6. Baris ke-15 sampai ke-17 berfungsi untuk membuat array kosong dengan ukuran yang sama dengan masing-masing *channel* RGB.
7. Baris ke-18 sampai ke-20 berfungsi untuk mengisi nilai variabel *redNorm*, *greenNorm* dan *blueNorm* dengan nilai piksel yang telah dinormalisasi pada masing-masing channel RGB.
8. Baris ke-21 berfungsi untuk menggabungkan hasil normalisasi pada variabel *merged*.
9. Baris ke-22 sampai ke-24 adalah inisialisasi nilai variabel *matriksKonv*, *Xn* dan *Zn*.
10. Baris ke-25 berfungsi untuk menyalin array *merged* dan menyimpannya pada variabel *xyz*.
11. Baris ke-26 sampai ke-30 adalah perulangan untuk melakukan perhitungan nilai masing-masing *channel* ruang warna tristimulus *XYZ*.
12. Baris ke-31 berfungsi untuk membuat array kosong bernama *Lab* dengan ukuran yang sama dengan array *xyz*.
13. Baris ke-32 sampai ke-39 adalah perulangan pada masing-masing indeks array *Lab* untuk melakukan perhitungan nilai transformasi XYZ ke LAB menggunakan Persamaan 2.20 sampai Persamaan 2.23.

14. Baris ke-40 bertujuan untuk mengembalikan nilai berupa matriks *Lab.*
15. Baris ke-41 dan ke-42 membaca file citra dengan nama file *001_0001.jpg* dan disimpan pada variabel *rgbImg*.
16. Baris ke-43 adalah proses pemanggilan method *convBGRtoLAB* dengan argumen berupa variabel *rgbImg*.

5.1.2.3 Implementasi Segmentasi

Proses segmentasi bertujuan untuk memisahkan citra target dari latar belakang (*background*). Proses segmentasi mengacu pada Subbab 4.1.2.3. Masukan pada proses ini adalah citra RGB asli. Keluaran dari proses ini adalah citra yang telah terpisah dari latar belakang dengan intensitas background bernilai 255 (putih). Proses segmentasi secara lengkap dijelaskan dengan Kode Sumber 4.

Kode Sumber 4: Proses Segmentasi

```
1  import cv2
2  import numpy as np
3  import fiturWarna as fw
4  def segmentation(rgbImg):
5     labNorm = fw.convBGRtoLAB(rgbImg)
6     lab = np.zeros_like(rgbImg)
7     for i in range(len(labNorm)):
8        for j in range(len(labNorm[i])):
9           lab[i][j][0] = labNorm[i][j][0] * 255 / 100
10          lab[i][j][1] = labNorm[i][j][1] + 128
11          lab[i][j][2] = labNorm[i][j][2] + 128
12    first = lab[:,:,0]
13    second = lab[:,:,1]
14    third = lab[:,:,2]
15    stdb = np.std((np.array(third)))
16    segmen2 = second.copy()
17    if (stdb<2):
18       segmen = first.copy()
19       for i in range(len(first)):
20          for j in range(len(first[i])):
21             if(segmen[i][j] > 200):
22                segmen[i][j] = 0
23             else:
24                segmen[i][j] = 255
25    else:
26       segmen = third.copy()
27          for i in range(len(first)):
28             for j in range(len(first[i])):
29                if(segmen[i][j] < 140):
30                   segmen[i][j] = 0
31                else:
32                   segmen[i][j] = 255
33    for i in range(len(second)):
34       for j in range(len(second[i])):
35          if(segmen2[i][j] < 130):
36             segmen2[i][j] = 0
37          else:
38             segmen2[i][j] = 255
39    mask = np.bitwise_or(segmen,segmen2)
40    result = rgbImg[:,:,:]
41    result[mask == 0] = (255, 255, 255)
```

```
42      return result

43  strFile = 001_0001.jpg'
44  rgbImg = cv2.imread(strFile)
45  lab = segmentation(rgbImg)
```

Penjelasan:

1. Baris ke-1 sampai ke-3 berfungsi untuk meng-*import library cv2, numpy dan fiturWarna.*
2. Baris ke-4 adalah inisialisasi method *segmentation* dengan parameter berupa citra RGB.
3. Baris ke-5 adalah proses pemanggilan method *convBGRtoLAB* dengan argumen array *rgbImg* dan menyimpan nilai kembaliannya pada variabel *labNorm*.
4. Baris ke-6 adalah inisialisasi array kosong bernama *lab* dengan ukuran yang sama dengan array *rgbImg*.
5. Baris ke-7 sampai ke-11 adalah perulangan pada masing-masing piksel array *labNorm* untuk melakukan denormalisasi pada nilai citra LAB menggunakan Persamaan 2.3 sampai Persamaan 2.5.
6. Baris ke-12 sampai ke-14 adalah proses pemisahan nilai dari masing-masing channel LAB.
7. Baris ke-15 adalah proses perhitungan nilai standar deviasi dari channel B pada citra LAB dan disimpan pada variabel *stdb*.
8. Baris ke-16 adalah proses pengisian nilai variabel *segmen2* dengan channel *A* pada citra LAB.
9. Baris ke-17 adalah inisialisasi percabangan untuk nilai variabel *stdb* kurang dari 2.
10. Baris ke-18 adalah proses pengisian nilai variabel *segmen* dengan channel *L* pada citra LAB.
11. Baris ke-19 sampai ke-24 adalah perulangan untuk masing-masing indeks *segmen* dengan melakukan *thresholding* menggunakan Persamaan 2.7. Jika nilai piksel lebih dari 200, maka nilai akan diubah menjadi 0 (hitam). Selain itu, nilai piksel akan diubah menjadi 255 (putih).
12. Baris ke-25 adalah inisialisasi percabangan untuk nilai variabel *stdb* lebih dari sama dengan 2.
13. Baris ke-26 adalah proses pengisian nilai variabel *segmen* dengan channel *B* pada citra LAB.
14. Baris ke-27 sampai ke-32 adalah perulangan untuk masing-masing indeks *segmen* dengan melakukan *thresholding* menggunakan Persamaan 2.7. Jika nilai piksel kurang dari 140, maka nilai akan diubah menjadi 0 (hitam). Selain itu, nilai piksel akan diubah menjadi 255 (putih).

15. Baris ke-33 sampai ke-38 adalah perulangan untuk masing-masing indeks *segmen2* dengan melakukan *thresholding* menggunakan Persamaan 2.7. Jika nilai piksel kurang dari 130, maka nilai akan diubah menjadi 0 (hitam). Selain itu, nilai piksel akan diubah menjadi 255 (putih).
16. Baris ke-39 berfungsi untuk melakukan operasi *bitwise OR* pada array *segmen* dan *segmen2* serta menyimpannya pada variabel *mask*.
17. Baris ke-40 adalah proses inisialisasi variabel *result* dengan nilai yang sama dengan array *rgbImg*.
18. Baris ke-41 adalah proses masking pada array result dengan syarat nilai variabel *mask* pada indeks yang sama adalah 0. Piksel yang memenuhi syarat tersebut diubah nilainya menjadi 255 pada masing-masing *channel* RGB.
19. Baris ke-42 adalah proses pengembalian nilai dengan nilai array *result*.
20. Baris ke-43 dan ke-44 membaca file citra dengan nama file *001_0001.jpg* dan disimpan pada variabel *rgbImg*.
21. Baris ke-45 adalah proses pemanggilan method *segmentation* dengan argumen berupa variabel *rgbImg*.

5.1.3 Implementasi Proses Ekstraksi Fitur Tekstur

Proses ekstraksi fitur tekstur bertujuan untuk mendapatkan nilai fitur tekstur menggunakan sepuluh fitur tekstur yang telah dijelaskan pada bagian Perancangan. Proses ekstraksi fitur tekstur terdiri dari perhitungan matriks co-occurence, perhitungan matriks probabilitas dan perhitungan nilai fitur tekstur.

5.1.3.1 Menghitung Matriks Co-Occurence

Proses perhitungan matriks *co-occurence* bertujuan untuk mendapatkan nilai matriks untuk jumlah setiap nilai piksel yang bersebelahan. Proses perhitungan matriks *co-occurence* mengacu pada Subbab 4.1.3.1. Masukan pada proses ini adalah citra LAB yang telah disegmentasi. Keluaran dari proses ini berupa matriks *co-occurence*. Proses perhitungan matriks *co-occurence* secara lengkap dijelaskan dengan Kode Sumber 5.

Kode Sumber 5: Proses Perhitungan Matriks Co-Occurence

```
1  def getValue(i, j, grayImg):
2     if((i < 0) or (j < 0) or (i >= len(grayImg)) or (j >=
   len(grayImg[0]))):
3        return -1
4     else:
5        return grayImg[i][j]
6  def getCoMatrix(rgbImg):
7     grayImg = RGBtoGray(rgbImg)
8     coMatrix0 = np.zeros((256,256), dtype=int)
9     coMatrix45 = np.zeros((256,256), dtype=int)
10    coMatrix90 = np.zeros((256,256), dtype=int)
11    coMatrix135 = np.zeros((256,256), dtype=int)
12    for i in range(len(grayImg)):
13       for j in range(len(grayImg[i])):
```

```
14          if (getValue(i, j+1, grayImg) != -1 and
    (getValue(i, j+1, grayImg) < 247 or getValue(i, j, grayImg)
15  < 247)):
                coMatrix0[getValue(i,j,grayImg)]
16  [getValue(i,j+1,grayImg)] += 1
                coMatrix45[getValue(i,j,grayImg)][getValue(i-
17  1,j+1,grayImg)] += 1
                coMatrix90[getValue(i,j,grayImg)][getValue(i-
18  1,j,grayImg)] += 1
                coMatrix135[getValue(i,j,grayImg)][getValue(i-
19  1,j-1,grayImg)] += 1
       return coMatrix0, coMatrix45, coMatrix90, coMatrix135
```

Penjelasan:

1. Baris ke-1 sampai ke-5 adalah method *getValue* dengan parameter indeks *i* dan *j*, serta citra *grayscale* yang dimaksud. Method *getValue* berfungsi untuk mendapatkan nilai dari suatu indeks pada citra *grayscale.* Jika indeks yang dimasukkan tidak melebihi ukuran citra maka method mengembalikan nilai piksel pada indeks tersebut. Jika indeks yang dimasukkan melebihi ukuran citra, maka method mengembalikan nilai -1.
2. Baris ke-6 adalah inisialisasi method *getCoMatrix* dengan parameter citra RGB. Method *getCoMatrix* berfungsi untuk mendapatkan matriks *co-occurence* dari suatu citra *grayscale*.
3. Baris ke-7 adalah proses pemanggilan method *RGBtoGray* dengan argumen berupa variabel *rgbImg*.
4. Baris ke-8 sampai ke-11 adalah inisialisasi variabel *coMatrix* untuk semua sudut. Ukuran variabel *coMatrix* adalah 256 x 256.
5. Baris ke-12 sampai ke-18 adalah perulangan pada setiap indeks *grayImg* untuk menghitung nilai dari matriks *co-occurence*. Nilai dari variabel *coMatrix* akan bertambah jika indeks nilai indeks *i* dan *j* memenuhi syarat pada method *getValue*.
6. Baris ke-19 adalah proses pengembalian nilai berupa nilai *coMatrix* untuk semua sudut.

5.1.3.2 Menghitung Matriks Probabilitas

Proses perhitungan matriks probabilitas bertujuan untuk mendapatkan nilai peluang dari matriks *co-occurence* yang telah didapatkan sebelumnya. Proses perhitungan matriks probabilitas mengacu pada Subbab 4.1.3.2. Masukan pada proses ini adalah matriks *co-occurence* untuk empat sudut. Keluaran dari proses ini berupa matriks probabilitas untuk empat fitur. Proses perhitungan matriks probabilitas secara lengkap dijelaskan dengan Kode Sumber 6.

Kode Sumber 6: Proses Perhitungan Matriks Probabilitas

```
1   def sumCM(coMatrix):
2      sumValue = 0
3      for i in range(len(coMatrix)):
4         for j in range(len(coMatrix[i])):
5            sumValue += coMatrix[i][j]
6      return sumValue
```

```
7   def getProbMatrix(coMatrix, sumValue):
8      probMatrix = coMatrix.copy()
9      probMatrix = probMatrix.astype(np.float64)
10     for i in range(len(probMatrix)):
11        for j in range(len(probMatrix[i])):
12           probMatrix[i][j]    =    np.divide(probMatrix[i][j],
    sumValue)
13     return probMatrix
```

Penjelasan:

1. Baris ke-1 adalah inisialisasi method *sumCM* dengan parameter berupa array *coMatrix*.
2. Baris ke-2 adalah inisialisasi variabel *sumValue* dengan nilai 0.
3. Baris ke-3 sampai ke-5 adalah perulangan pada setiap piksel *coMatrix* untuk melakukan penambahan nilai pada variabel sumValue dengan nilai *coMatrix* pada indeks tertentu.
4. Baris ke-6 adalah proses pengembalian nilai berupa nilai variabel *sumValue*.
5. Baris ke-7 adalah inisialisasi method *getProbMatrix* dengan parameter berupa *coMatrix* dan *sumValue* dari *coMatrix* tersebut.
6. Baris ke-8 dan ke-9 adalah proses menyalin variabel *coMatrix* pada variabel *probMatrix* dan mengubah tipe datanya menjadi *float*.
7. Baris ke-10 sampai ke-12 adalah perulangan pada setiap indeks *probMatrix* untuk melakukan perhitungan *probMatrix*.
8. Baris ke-13 adalah proses pengembalian nilai berupa nilai dari variabel *probMatrix*.

5.1.3.3 Menghitung Fitur Tekstur

Proses perhitungan fitur tekstur bertujuan untuk mendapatkan nilai fitur tekstur berdasarkan matriks probabilitas yang telah didapatkan sebelumnya. Proses perhitungan fitur tekstur mengacu pada Subbab 4.1.3.3. Masukan pada proses ini adalah matriks probabilitas untuk empat sudut. Keluaran dari proses ini berupa sepuluh nilai fitur untuk masing-masing citra. Proses perhitungan fitur tekstur secara lengkap dijelaskan dengan Kode Sumber 7.

Kode Sumber 7: Proses Perhitungan Fitur Tekstur

```
1   # FEATURE 1 : MEAN
2   def meanX(probMatrix):
3       sumTotal = 0
4       for i in range(len(probMatrix)):
5           sumJ = 0
6           for j in range(len(probMatrix[i])):
7               sumJ += probMatrix[i][j]
8           sumTotal += i*sumJ
9       return sumTotal

10  def meanY(probMatrix):
11      sumTotal = 0
```

```
12      for j in range(len(probMatrix[0])):
13          sumI = 0
14          for i in range(len(probMatrix)):
15              sumI += probMatrix[i][j]
16          sumTotal += j*sumI
17      return sumTotal

18  # FEATURE 2 : VARIANS
19  def variansX(probMatrix, meanX):
20      sumTotal = 0
21      for i in range(len(probMatrix)):
22          sumJ = 0
23          for j in range(len(probMatrix[i])):
24              sumJ += probMatrix[i][j]
25          sumTotal += np.power(i-meanX,2)*sumJ
26      return sumTotal

27  def variansY(probMatrix, meanY):
28      sumTotal = 0
29      for j in range(len(probMatrix[0])):
30          sumI = 0
31          for i in range(len(probMatrix)):
32              sumI += probMatrix[i][j]
33          sumTotal += np.power(j-meanY,2)*sumI
34      return sumTotal

35  # FEATURE 3 : ENERGY
36  def energy(probMatrix):
37      sumTotal = 0
38      for i in range(len(probMatrix)):
39          for j in range(len(probMatrix[i])):
40              sumTotal += np.power(probMatrix[i][j],2)
41      return sumTotal

42  # FEATURE 4 : ENTROPY
43  def entropy(probMatrix):
44      sumTotal = 0
45      for i in range(len(probMatrix)):
46          for j in range(len(probMatrix[i])):
47              if (probMatrix[i][j] != 0.0):
48                  sumTotal +=
    probMatrix[i][j]*np.log(probMatrix[i][j])
49      return -sumTotal

50  # FEATURE 5 : CONTRAST
51  def contrast(probMatrix):
52      sumTotal = 0
53      for i in range(len(probMatrix)):
54          for j in range(len(probMatrix[i])):
55              sumTotal += np.power(i-j,2)*probMatrix[i][j]
56      return sumTotal

57  # FEATURE 6 : DISSIMILARITY
58  def dissimilarity(probMatrix):
59      sumTotal = 0
60      for i in range(len(probMatrix)):
61          for j in range(len(probMatrix[i])):
62              sumTotal += np.abs(i-j)*probMatrix[i][j]
63      return sumTotal
```

```
64  # FEATURE 7 : HOMOGENEITY
65  def homogeneity(probMatrix):
66      sumTotal = 0
67      for i in range(len(probMatrix)):
68          for j in range(len(probMatrix[i])):
69              sumTotal +=
    np.divide(probMatrix[i][j],1+np.power(i-j,2))
70      return sumTotal

71  # FEATURE 8 : CORRELATION
72  def correlation(probMatrix, meanX, meanY, varX, varY):
73      sumTotal = 0
74      for i in range(len(probMatrix)):
75          for j in range(len(probMatrix[i])):
76              sumTotal += np.divide((i-meanY)*(j-
    meanX)*probMatrix[i][j],varX*varY)
77      return sumTotal

78  def getFeature(coMatrix, sumCoMatrix):
79      probMatrix = getProbMatrix(coMatrix, sumCoMatrix)
80      meanXF = meanX(probMatrix)
81      meanYF = meanY(probMatrix)
82      varXF = variansX(probMatrix, meanXF)
83      varYF = variansY(probMatrix, meanYF)
84      energyF = energy(probMatrix)
85      entropyF = energy(probMatrix)
86      contrastF = contrast(probMatrix)
87      dissimilarityF = dissimilarity(probMatrix)
88      homogeneityF = homogeneity(probMatrix)
89      correlationF = correlation(probMatrix, meanXF, meanYF,
    varXF, varYF)
90      return meanXF, meanYF, varXF, varYF, energyF, entropyF,
    contrastF, dissimilarityF, homogeneityF, correlationF
```

Penjelasan:

1. Baris ke-1 sampai ke-9 adalah method *meanX* dengan parameter berupa variabel *probMatrix*. Method *meanX* berfungsi untuk melakukan perhitungan fitur *meanX* menggunakan Persamaan 2.9.
2. Baris ke-10 sampai ke-17 adalah method *meanY* dengan parameter berupa variabel *probMatrix*. Method *meanY* berfungsi untuk melakukan perhitungan fitur *meanY* menggunakan Persamaan 2.10.
3. Baris ke-18 sampai ke-26 adalah method *variansX* dengan parameter berupa variabel *probMatrix* dan *meanX*. Method *variansX* berfungsi untuk melakukan perhitungan fitur *variansX* menggunakan Persamaan 2.11.
4. Baris ke-27 sampai ke-34 adalah method *variansY* dengan parameter berupa variabel *probMatrix* dan *meanY*. Method *variansY* berfungsi untuk melakukan perhitungan fitur *variansY* menggunakan Persamaan 2.12.
5. Baris ke-35 sampai ke-41 adalah method *energy* dengan parameter berupa variabel *probMatrix*. Method *energy* berfungsi untuk melakukan perhitungan fitur *energy* menggunakan Persamaan 2.17.

6. Baris ke-42 sampai ke-49 adalah method *entropy* dengan parameter berupa variabel *probMatrix*. Method *entropy* berfungsi untuk melakukan perhitungan fitur *entropy* menggunakan Persamaan 2.18.
7. Baris ke-50 sampai ke-56 adalah method *contrast* dengan parameter berupa variabel *probMatrix*. Method *contrast* berfungsi untuk melakukan perhitungan fitur *contrast* menggunakan Persamaan 2.13.
8. Baris ke-57 sampai ke-63 adalah method *dissimilarity* dengan parameter berupa variabel *probMatrix*. Method *dissimilarity* berfungsi untuk melakukan perhitungan fitur *dissimilarity* menggunakan Persamaan 2.14.
9. Baris ke-64 sampai ke-70 adalah method homogeneity dengan parameter berupa variabel probMatrix. Method homogeneity berfungsi untuk melakukan perhitungan fitur homogeneity menggunakan Persamaan 2.16.
10. Baris ke-71 sampai ke-77 adalah method *correlation* dengan parameter berupa variabel *probMatrix*, *meanXF*, *meanYF*, *varXF*, *varYF*. Method *correlation* berfungsi untuk melakukan perhitungan fitur *correlation* menggunakan Persamaan 2.15.
11. Baris ke-78 sampai 90 adalah method *getFeature* dengan parameter variabel *coMatrix* dan *sumCoMatrix* dari *coMatrix* tersebut. method getFeature berfungsi untuk memanggil method *getProbMatrix* dan melakukan perhitungan untuk setiap fitur tekstur. Nilai kembalian dari method ini adalah keseluruhan nilai fitur tekstur yang telah dihitung.

5.1.4 Implementasi Proses Ekstraksi Fitur Warna

Proses ekstraksi fitur warna bertujuan untuk mendapatkan nilai fitur warna menggunakan nilai momen warna. Proses ekstraksi fitur warna mengacu pada Subbab 4.1.4. Masukan pada proses ini adalah citra dalam ruang warna LAB. Keluaran dari proses ini berupa empat fitur warna untuk tiga *channel* warna LAB. Proses ekstraksi fitur warna secara lengkap dijelaskan dengan Kode Sumber 8.

Kode Sumber 8: Proses Inisialisasi Data Latih dan Data Uji

```
1   def meanMoment(channel):
2       sumValue = 0
3       countValue = 0
4       for i in range(len(channel)):
5           for j in range(len(channel[i])):
6               if(channel[i][j] < 200):
7                   sumValue += channel[i][j]
8                   countValue += 1
9       if(countValue == 0):
10          return 0
11      else:
12          return sumValue/countValue

13  def varianceMoment(channel, meanChannel):
14      sumValue = 0
15      countValue = 0
16      for i in range(len(channel)):
17          for j in range(len(channel[i])):
```

```
18              if(channel[i][j] < 200):
19                  sumValue += np.power(channel[i][j] -
    meanChannel,2)
20                  countValue += 1
21      if(countValue == 0):
22          return 0
23      else:
24          return np.sqrt(sumValue/countValue)

25  def skewnessMoment(channel, meanChannel):
26      sumValue = np.int64(0)
27      countValue = 0
28      for i in range(len(channel)):
29          for j in range(len(channel[i])):
30              if(channel[i][j] < 200):
31                  sumValue += np.power(channel[i][j] -
    meanChannel,3)
32                  countValue += 1
33      if(countValue == 0):
34          return 0
35      else:
36          return np.cbrt(sumValue/countValue)

37  def kurtosisMoment(channel, meanChannel):
38      sumValue = np.int64(0)
39      countValue = 0
40      for i in range(len(channel)):
41          for j in range(len(channel[i])):
42              if(channel[i][j] < 200):
43                  sumValue += np.power(channel[i][j] -
    meanChannel,4)
44                  countValue += 1
45      if(countValue == 0):
46          return 0
47      else:
48          return np.power(sumValue/countValue,0.25)

49  def getColorMoment(channel):
50      meanChannel = meanMoment(channel)
51      varChannel = varianceMoment(channel, meanChannel)
52      skewChannel = skewnessMoment(channel, meanChannel)
53      kurtChannel = kurtosisMoment(channel, meanChannel)
54      return meanChannel, varChannel, skewChannel,
    kurtChannel
55
56  meanL, varL, skewL, kurtL = getColorMoment(Lab[:,:,0])
57  meanA, varA, skewA, kurtA = getColorMoment(Lab[:,:,1])
    meanB, varB, skewB, kurtB = getColorMoment(Lab[:,:,2])
```

Penjelasan:

1. Baris ke-1 sampai ke-12 adalah method *meanMoment* dengan parameter berupa variabel *channel*. Method *meanMoment* berfungsi untuk melakukan perhitungan momen warna *mean* menggunakan Persamaan 2.24.
2. Baris ke-13 sampai ke-24 adalah method *varianceMoment* dengan parameter berupa variabel *channel*. Method *varianceMoment* berfungsi

untuk melakukan perhitungan momen warna *varians* menggunakan Persamaan 2.25.

3. Baris ke-25 sampai ke-36 adalah method *skewnessMoment* dengan parameter berupa variabel *channel*. Method *skewnessMoment* berfungsi untuk melakukan perhitungan momen warna *skewness* menggunakan Persamaan 2.26.
4. Baris ke-37 sampai ke-48 adalah method *kurtosisMoment* dengan parameter berupa variabel *channel*. Method *kurtosisMoment* berfungsi untuk melakukan perhitungan momen warna *kurtosis* menggunakan Persamaan 2.27.
5. Baris ke-49 sampai ke-54 adalah method *getColorMoment* dengan parameter berupa variabel *channel*. Method *getColorMoment* berfungsi untuk melakukan pemanggilan method *meanMoment*, *varianceMoment*, *skewnessMoment* dan *kurtosisMoment* dan menyimpannya pada variabel *meanChannel*, *varChannel*, *skewChannel* dan *kurtChannel*. Nilai kembalian dari method ini adalah keempat nilai fitur dari *channel* tertentu
6. Baris ke-55 sampai ke-57 adalah proses pemanggilan method *getColorMoment* untuk masing-masing *channel* LAB.

5.1.5 Implementasi Proses Normalisasi

Proses normalisasi bertujuan untuk mendapatkan data dengan sebaran data yang seimbang dan proporsional. Proses normalisasi mengacu pada Subbab 4.1.5. Masukan pada proses ini adalah seluruh data fitur tekstur dan warna. Keluaran dari proses ini adalah data fitur yang telah dinormalisasi. Proses normalisasi secara lengkap dijelaskan dengan Kode Sumber 1.

Kode Sumber 9: Proses Normalisasi

```
1   import csv
2   import numpy as np

3   def read_csv(file_name):
4       array_2D = []
5       with open(file_name, 'rb') as csvfile:
6           read = csv.reader(csvfile, delimiter=';')
7           for row in read:
8               array_2D.append(row)
9       return array_2D

10  dataAsli = read_csv('data/dataNormal1.csv')
11  dataTrain          =          ((np.array(dataAsli[:]))[:,1:-
      1]).astype(np.float64).tolist()
12  maxValue = [-999999]*len(dataTrain[0])
13  minValue = [999999]*len(dataTrain[0])
14  for i in range(len(dataTrain)):
15      for j in range(len(dataTrain[i])):
16          if (dataTrain[i][j] > maxValue[j]):
17              maxValue[j] = dataTrain[i][j]
18          if (dataTrain[i][j] < minValue[j]):
19              minValue[j] = dataTrain[i][j]
```

```
20  dataNorm = np.copy(dataTrain).tolist()
21  for i in range(len(dataNorm)):
22      for j in range(len(dataNorm[i])):
23          dataNorm[i][j] = (dataNorm[i][j] - minValue[j]) /
      (maxValue[j] - minValue[j])
```

Penjelasan:

1. Baris ke-1 dan ke-2 adalah berfungsi untuk meng-*import library csv dan numpy.*
2. Baris ke-3 sampai ke-9 adalah method read_csv dengan parameter nama file csv yang berfungsi untuk membaca file csv.
3. Baris ke-10 adalah proses pemanggilan method read_csv dengan argumen nama file.
4. Baris ke-11 adalah proses pemisahan data dari id dan label kelas pada masing-masing data.
5. Baris ke-12 dan ke-13 adalah inisialisasi array maxValue dan minValue untuk menampung nilai maksimal dan minimal pada masing-masing fitur.
6. Baris ke-14 sampai ke-19 berfungsi untuk mencari nilai maksimal dan minimal untuk masing-masing fitur.
7. Baris ke-20 adalah proses menyalin array dataTrain dan menyimpannya pada array dataNorm.
8. Baris ke-21 sampai ke-23 berfungsi untuk melakukan normalisasi pada seluruh data menggunakan Persamaan 2.1.

5.1.6 Implementasi Proses Klasifikasi LVQ3

Proses klasifikasi LVQ3 merupakan proses utama pada penelitian ini. Proses ini bertujuan untuk mendapatkan prediksi kelas untuk setiap citra pada data latih maupun data uji. Proses klasifikasi LVQ3 secara umum terbagi menjadi dua, yaitu proses pelatihan dan pengujian LVQ3.

5.1.6.1 Implementasi Pelatihan LVQ3

Proses pelatihan LVQ3 bertujuan untuk mendapatkan nilai bobot pada *reference vector* yang merujuk pada masing-masing kelas. Proses pelatihan LVQ3 mengacu pada Subbab 4.1.6.1. Masukan pada proses ini adalah seluruh data latih dan *reference vector*. Keluaran dari proses ini adalah bobot *reference vector* hasil pelatihan. Proses pelatihan LVQ3 secara lengkap dijelaskan dengan Kode Sumber 11.

Kode Sumber 11: Proses Pelatihan LVQ3

```
1  # Initialize Weight Matrix
2  weightMatrix = np.zeros((len(dataClass),
   len(dataTraining[0][0])) ,dtype=np.float64)
3  for i in range(len(weightMatrix)):
4      for j in range(len(dataTraining[i][0])):
5          weightMatrix[i][j] = dataClass[i][0][j]
6  def takeFirst(elem):
7      return elem[0]
```

```
countLVQ1 = 0
countLVQ2 = 0
countLVQ21 = 0
countLVQ3 = 0
# ITERATION LVQ
iterasi = 2
for x in range(iterasi):
    # Find Euclidean Distance
    jarak = np.zeros(len(dataClass),dtype=np.float64)
    for i in range(len(dataTraining)):
        for j in range(len(dataClass)):
            jarak[j] = 0
            for k in range(len(dataTraining[i][0])):
                jarak[j] += np.power(dataTraining[i][0][k] - weightMatrix[j][k],2)
            jarak[j] = np.sqrt(jarak[j])
        jVal = []
        for j in range(len(dataClass)):
            jVal.append([jarak[j],j])
        jVal.sort(key=takeFirst)
        t = dataTraining[i][1]
        yc1 = int(jVal[0][1]) # The nearest class from the data training
        yc2 = int(jVal[1][1]) # The second nearest class from the data training
        dc1 = jVal[0][0] # The distance of the nearest class from the data training
        dc2 = jVal[1][0] # The distance of the second nearest class from the data training
        for j in range(len(weightMatrix)):
            for k in range(len(weightMatrix[j])):
                if (min(np.divide(dc1,dc2),np.divide(dc2,dc1)) > ((1-epsilon2)*(1+epsilon2))):
                    if ((classDataClass[yc1][0] == t and classDataClass[yc2][0] == t) ):
                        countLVQ3 += 1
                        if(j == yc1 or j == yc2):
                            weightMatrix[j][k] = (1-beta)*weightMatrix[j][k] + beta*dataTraining[i][0][k]
                    elif((classDataClass[yc1][0] == t and classDataClass[yc2][0] != t) or (classDataClass[yc1][0] != t and classDataClass[yc2][0] == t)):
                        countLVQ21 += 1
                        if(j == yc1):
                            if(t == classDataClass[yc1][0]):
                                weightMatrix[j][k] = (1-alpha)*weightMatrix[j][k] + alpha*dataTraining[i][0][k]
                            else:
                                weightMatrix[j][k] = (1+alpha)*weightMatrix[j][k] - alpha*dataTraining[i][0][k]
                        elif(j == yc2):
                            if(t == classDataClass[yc2][0]):
                                weightMatrix[j][k] = (1-alpha)*weightMatrix[j][k] + alpha*dataTraining[i][0][k]
                            else:
                                weightMatrix[j][k] = (1+alpha)*weightMatrix[j][k] - alpha*dataTraining[i][0][k]
                elif (j == yc1):
```

```
51                      countLVQ1 += 1
52                      if(classDataClass[j][0] == t):
53                          weightMatrix[j][k]       =       (1-
54  alpha)*weightMatrix[j][k] + alpha*dataTraining[i][0][k]
                        else:
55                          weightMatrix[j][k]                  =
56  (1+alpha)*weightMatrix[j][k]  -  alpha*dataTraining[i][0][k]
      alpha = 0.8*alpha
57
```

Penjelasan:

1. Baris ke-1 dan ke-2 adalah proses inisialisasi matriks bobot awal dengan nilai 0.
2. Baris ke-3 sampai ke-5 adalah perulangan pada setiap indeks matriks *weightMatrix* dan mengisi nilai *weightMatrix* pada indeks tersebut dengan nilai variabel *dataClass* pada indeks terkait.
3. Baris ke-6 dan ke-7 adalah method *takeFirst* dengan parameter variabel *elem* untuk melakukan pengurutan berdasarkan elemen indeks ke 0.
4. Baris ke-8 sampai ke-11 adalah inisialisasi variabel penghitung proses pada masing-masing tahapan.
5. Baris ke-12 dan ke-13 adalah inisialisasi variabel iterasi maksimal dengan nilai 2.
6. Baris ke-14 adalah inisialisasi perulangan *for* sejumlah variabel iterasi.
7. Baris ke-15 dan ke-16 adalah inisialisasi variabel jarak dengan panjang yang sama dengan panjang variabel *dataClass*.
8. Baris ke-17 sampai ke-22 adalah perulangan pada masing-masing data latih dan *reference vector* untuk melakukan perhitungan jarak *Euclidean* pada masing-masing data latih dan *reference vector*.
9. Baris ke-23 sampai ke-26 adalah proses pengurutan array jarak dimulai dari nilai yang terkecil dan disimpan pada variabel *jVal*.
10. Baris ke-27 sampai ke-31 adalah proses inisialisasi variabel *j*, *yc1*, *yc2*, *dc1* dan *dc2* berdasarkan variabel *jVal*.
11. Baris ke-32 sampai ke-56 adalah proses pengubahan bobot menggunakan algoritme LVQ3. Perubahan bobot pada masing-masing kondisi menggunakan Persamaan 2.31 sampai Persamaan 2.32.
12. Baris ke-57 adalah proses perubahan nilai *learning rate* dengan pengali bernilai 0,8.

5.1.6.2 Implementasi Pengujian LVQ3

Proses pengujian LVQ3 bertujuan untuk mendapatkan hasil prediksi kelas untuk masing-masing citra pada data uji. Proses pengujian LVQ3 mengacu pada Subbab 4.1.6.2. Masukan pada proses ini adalah bobot *reference vector* hasil pelatihan. Keluaran dari proses ini adalah hasil prediksi kelas untuk masing-masing

citra pada data uji. Proses pengujian LVQ3 secara lengkap dijelaskan dengan Kode Sumber 12.

Kode Sumber 12: Proses Pengujian LVQ3

```
1     wrongClass = 0
2     with open(testingResult.csv', 'a') as myfile:
3         wr = csv.writer(myfile, delimiter=',')
4         for z in range(len(dataTesting)):
5             testing = classDataTest[z][0]-1
6             dataTest = dataTesting[testing]
7             classResult = -1
8             minValue = 999999999
9             for i in range(len(weightMatrix)):
10                sumValue = 0
11                for j in range(len(weightMatrix[i])):
12                    sumValue += np.power(dataTesting[z][j] -
    weightMatrix[i][j],2)
13                if (sumValue < minValue):
14                    minValue = sumValue
15                    classResult = classDataClass[i][0]-1
16            result = [testing,classResult]
17            wr.writerow(result)
18            if (testing != classResult):
19                wrongClass+=1
20            akurasi    = np.divide(float(len(dataTesting) -
    wrongClass),float(len(dataTesting)))
```

Penjelasan:

9. Baris ke-1 adalah inisialisasi variabel *wrongClass* untuk menyimpan prediksi kelas yang salah pada proses pengujian.
10. Baris ke-2 dan ke-3 adalah proses pembukaan file bernama *testingResult.csv* untuk menyimpan hasil pengujian.
11. Baris ke-4 adalah inisialisasi variabel *testing* dan *dataTest*.
12. Baris ke-5 sampai ke-15 adalah proses pencarian jarak minimal antara data uji dengan masing-masing *reference vector*. Jarak dan kelas *reference vector* minimal dicatat pada variabel *classResult* dan *minValue*.
13. Baris ke-16 dan ke-17 adalah proses menulis kelas aktual dan kelas prediksi pada file *testingResult.csv*.
14. Baris ke-18 dan ke-19 adalah proses penambahan nilai *wrongClass* untuk prediksi kelas yang salah.
15. Baris ke-20 adalah proses penghitungan nilai akurasi menggunakan Persamaan 2.34.

5.2 Implementasi Antarmuka

Implementasi antarmuka pada penelitian ini dilakukan berdasarkan perancangan yang telah dilakukan sebelumnya. Pada bagian perancangan dijelaskan bahwa antarmuka sistem ini terbagi menjadi tiga bagian. Keempat bagian tersebut adalah halaman beranda, *section* klasifikasi, *modal screen* nilai

fitur. Masing-masing bagian memiliki karakteristik dan fungsi masing-masing yang telah dijelaskan sebelumnya pada bagian perancangan.

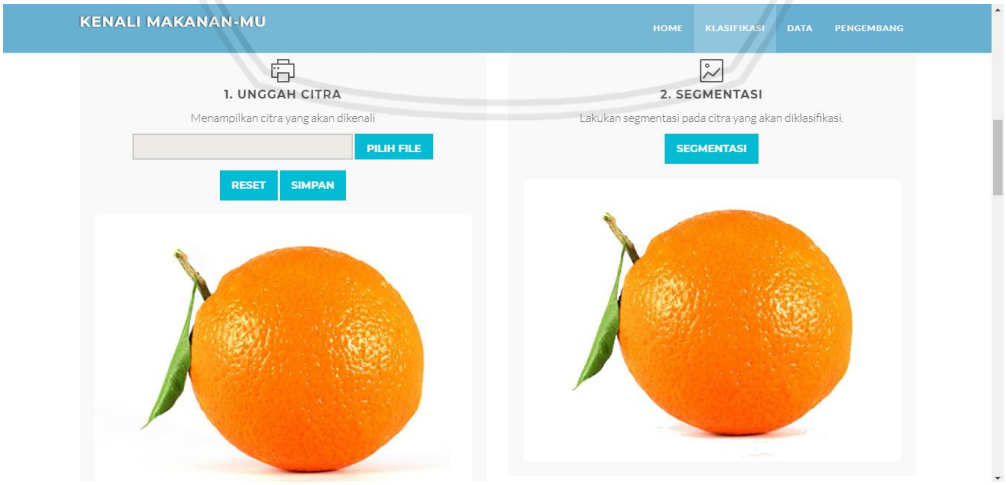
5.2.1 Antarmuka Halaman Beranda

Implementasi Halaman Beranda mengacu pada perancangan antarmuka halaman beranda pada Subbab 4.2.1. Halaman beranda berisi beberapa judul sistem dan tombol navigasi pada masing-masing menu, yaitu Beranda, Klasifikasi, Data dan Pengembang. Gambar menunjukkan tampilan dari halaman beranda.

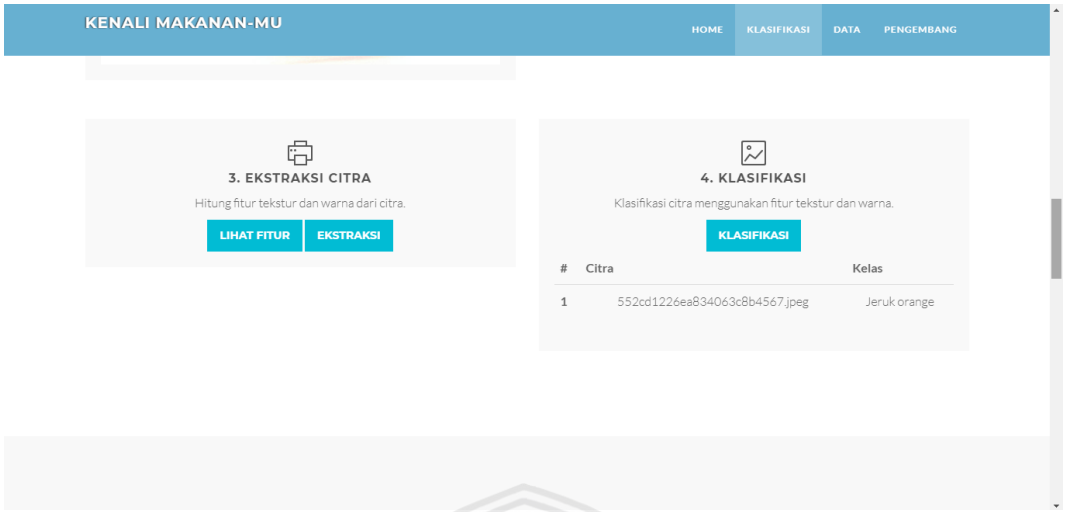
Gambar 5.1 Tampilan Halaman Beranda

5.2.2 Antarmuka Section Klasifikasi

Implementasi *Section* Klasifikasi mengacu pada perancangan antarmuka halaman beranda pada Subbab 4.2.2. Section klasifikasi ini berisi empat tahapan utama dalam proses klasifikasi, yaitu unggah citra, segmentasi, ekstraksi fitur dan klasifikasi. Gambar menunjukkan tampilan dari *section* klasifikasi.



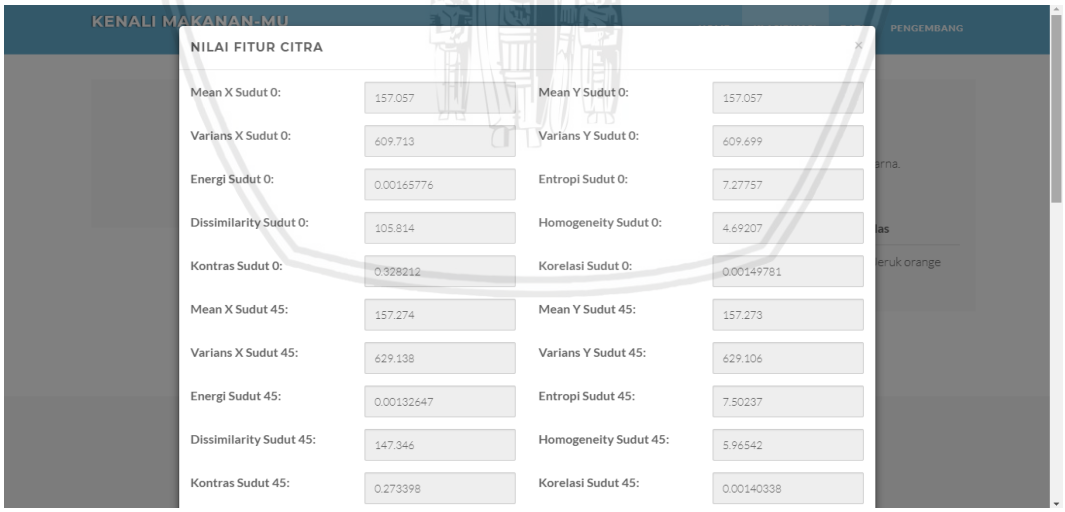
Gambar 5.2 Tampilan Section Klasifikasi untuk Proses Unggah Citra dan Segmentasi



Gambar 5.3 Tampilan Section Klasifikasi untuk Proses Ekstraksi Fitur dan Klasifikasi

5.2.3 Antarmuka Modal Screen Nilai Fitur

Implementasi *Modal Screen* Nilai Fitur mengacu pada perancangan antarmuka halaman beranda pada Subbab 4.2.3. Nilai fitur ditampilkan pada text field sejumlah jenis fitur dengan didahului oleh label yang bertuliskan jenis fitur yang dimaksud. Seluruh text field dan label ditampilkan pada sebuah modal screen yang akan muncul dengan menekan tombol “LIHAT FITUR”. Sebelum melakukan proses ekstraksi fitur, nilai pada masing-masing text field berisi “Belum Ditentukan”. Gambar menunjukkan tampilan dari *modal screen* nilai fitur.



Gambar 5.4 Tampilan Modal Screen Nilai Fitur

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai hasil pengujian dan analisis terhadap pengujian yang telah dilakukan. Pengujian dilakukan sesuai dengan perancangan pengujian yang telah dilakukan sebelumnya. Pengujian yang dilakukan pada penelitian ini antara lain pengujian learning rate, pengujian pengali learning rate, pengujian iterasi maksimal, pengujian learning rate minimal dan pengujian akurasi.

Seluruh pengujian menggunakan metode *k-fold cross validation* dan akurasi sebagai metode evaluasi. Akurasi didapatkan dengan membandingkan jumlah data yang sesuai dengan target dengan jumlah data keseluruhan. Sebagai contoh, Tabel 6.1 menunjukkan pengujian hasil klasifikasi.

Tabel 6.1 Ilustrasi Pengujian Hasil Klasifikasi

No	Id dataUji	Kelas Hasil	Kelas Target	Status
1	004_0003	 4 (Donat)	 4 (Donat)	**Benar**
2	001_0003	 3 (Stroberi)	 1 (Tomat)	**Salah**
3	002_0003	 1 (Tomat)	 2 (Jeruk Oranye)	**Salah**
4	003_0003	 4 (Donat)	 3 (Genji Pie)	**Salah**

6.1 Hasil dan Analisis Pengujian Learning Rate

Pengujian *learning rate* digunakan untuk mengetahui nilai *learning rate* yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. *Learning rate* yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.1. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter pengali *learning rate* = 0,8
2. Nilai *m* = 0,1
3. Nilai *epsilon* = 0,4
4. Nilai parameter iterasi maksimal = 40
5. Nilai parameter *learning rate* minimal = 0,000001

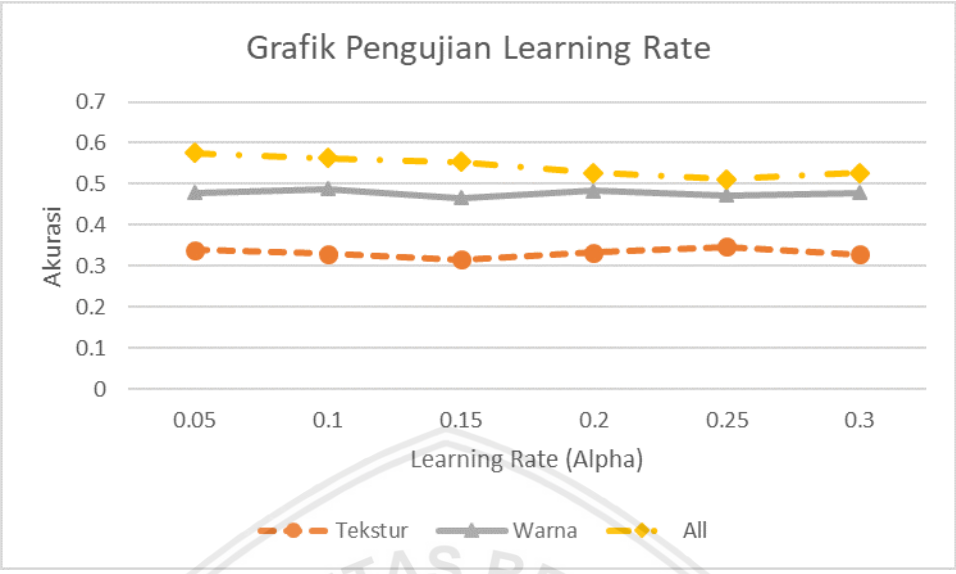
Tabel 6.2 Hasil Pengujian Learning Rate

Fitur	Learning Rate					
	0.05	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**
Tekstur	0.338882	0.330645	0.316763	0.333122	0.346947	0.329147
Warna	0.478571	0.488422	0.466187	0.484101	0.473041	0.478168
Tekstur dan Warna	0.57621	0.562615	0.553053	0.526959	0.512097	0.526728

Hasil dari pengujian learning rate secara lengkap tersaji pada Tabel 6.2. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.1. Klasifikasi menggunakan fitur tekstur mencapai akurasi maksimal sebesar 0,347 pada *learning rate* bernilai 0,25. Klasifikasi menggunakan fitur warna mencapai akurasi maksimal sebesar 0,4884 pada *learning rate* bernilai 0,1. Sedangkan klasifikasi menggunakan kedua fitur (tekstur dan warna) mencapai akurasi maksimal sebesar 0,5762 pada *learning rate* bernilai 0,05.

Nilai *learning rate* berpengaruh terhadap tingkat kecepatan pelatihan. Nilai *learning rate* yang terlalu kecil akan mengakibatkan laju pelatihan menjadi lambat dan beresiko terjadi *overfitting*. Sebaliknya nilai *learning rate* yang terlalu besar akan mengakibatkan model mudah mencapai konvergensi dini. Nilai *learning rate* juga dapat memiliki nilai yang berbeda untuk beberapa kasus yang berbeda pula. Sebagai contoh, akurasi maksimal pada model menggunakan fitur tekstur dapat mencapai akurasi maksimal pada learning rate bernilai 0,25. Hal ini dikarenakan model tersebut relatif dapat terbebas dari *local optima* yang diakibatkan oleh nilai tekstur dari beberapa kelas yang relatif sama. Nilai tekstur yang hampir sama antara satu kelas dengan kelas yang lain dapat mengakibatkan citra pengujian diklasifikasikan pada kelas yang salah.

Akurasi maksimal pada model menggunakan fitur warna saja serta fitur tekstur dan warna dicapai pada *learning rate* bernilai masing-masing 0,1 dan 0,05. Hal ini dikarenakan sistem dapat mencapai model yang paling cocok dengan nilai *learning rate* yang relatif kecil.



Gambar 6.1 Grafik Pengujian Learning Rate

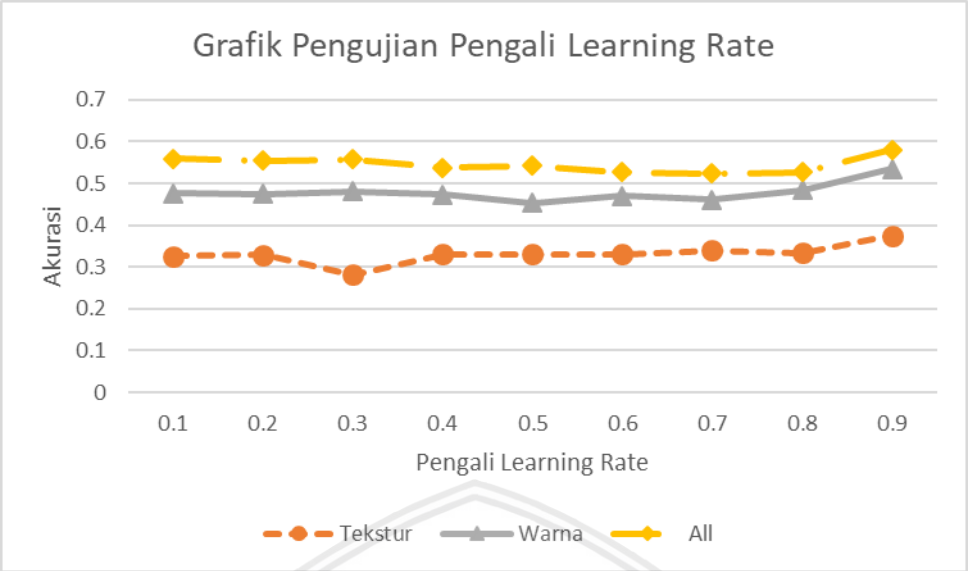
6.2 Hasil dan Analisis Pengujian Pengali Learning Rate

Pengujian pengali *learning rate* digunakan untuk mengetahui nilai pengali *learning rate* yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. Pengali *learning rate* yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.2. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter *learning rate* = 0,2
2. Nilai *m* = 0,1
3. Nilai *epsilon* = 0,4
4. Nilai parameter iterasi maksimal = 40
5. Nilai parameter *learning rate* minimal = 0,000001

Tabel 6.3 Hasil Pengujian Pengali Learning Rate

Fitur	Pengali Learning Rate								
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Tekstur	0.325	0.328	0.281	0.33	0.33	0.33	0.34	0.333	0.374
Warna	0.475	0.474	0.48	0.472	0.453	0.469	0.46	0.484	0.535
Tekstur dan Warna	0.558	0.554	0.557	0.537	0.541	0.526	0.523	0.526	0.581



Gambar 6.2 Grafik Pengujian Pengali Learning Rate

Hasil dari pengujian pengali learning rate secara lengkap tersaji pada Tabel 6.3. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.2. Klasifikasi menggunakan fitur tekstur saja, fitur warna saja maupun fitur tekstur dan warna mencapai akurasi maksimal pada nilai pengali *learning rate* sebesar 0,9 dengan akurasi sebesar masing-masing 0,374 , 0,535 , dan 0,581.

Nilai pengali learning rate berguna untuk menurunkan nilai *learning rate* dengan harapan seiring bertambahnya epoch maka pencarian model akan semakin fokus pada suatu sisi. Semakin kecil nilai pengali *learning rate* maka seiring bertambahnya epoch, nilai *learning rate* akan semakin kecil dengan selisih yang semakin besar. Sebaliknya nilai pengali *learning rate* yang besar akan mengakibatkan pencarian model tidak fokus pada satu sisi dan tidak terjebak pada *local optima*.

Sebagai contoh, akurasi maksimal pada ketiga jenis fitur (fitur tekstur saja, fitur warna saja serta fitur tekstur dan warna) didapatkan pada nilai pengali *learning rate* sebesar 0,9. Hal ini dikarenakan *learning rate* seiring bertambahnya epoch tidak mengalami pengurangan nilai yang terlalu jauh sehingga tidak mudah terjebak pada *local optima*.

6.3 Hasil dan Analisis Pengujian Nilai m

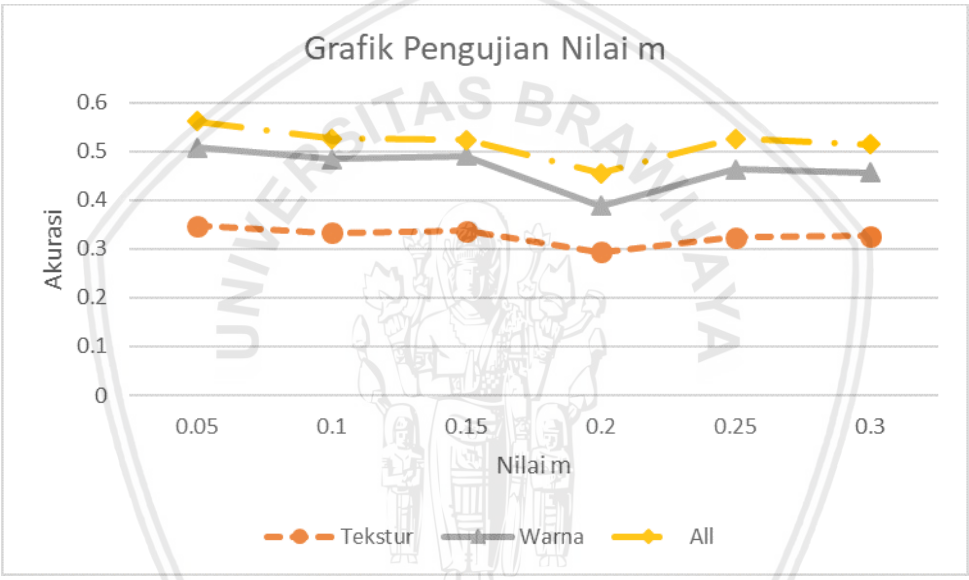
Pengujian nilai *m* digunakan untuk mengetahui nilai *m* yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. Nilai *m* yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.3. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter *learning rate* = 0,2
2. Nilai parameter pengali *learning rate* = 0,8
3. Nilai *epsilon* = 0,4

4. Nilai parameter iterasi maksimal = 40
5. Nilai parameter *learning rate* minimal = 0,000001

Tabel 6.4 Hasil Pengujian Nilai m

Fitur	Nilai *m*					
	0.05	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**
Tekstur	0.348329	0.333122	0.33773	0.293203	0.32356	0.327074
Warna	0.508353	0.484101	0.491359	0.389055	0.462903	0.457431
Tekstur dan Warna	0.561463	0.526959	0.524482	0.455588	0.527016	0.515323



Gambar 6.3 Grafik Pengujian Nilai m

Hasil dari pengujian nilai *m* secara lengkap tersaji pada Tabel 6.4. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.3. Klasifikasi menggunakan fitur tekstur saja, fitur warna saja maupun fitur tekstur dan warna mencapai akurasi maksimal pada nilai *m* sebesar 0,05 dengan akurasi sebesar masing-masing 0,3483 , 0,5083 , dan 0,5615.

Nilai *m* berpengaruh terhadap nilai β yang digunakan pada proses pelatihan LVQ3. Nilai *m* yang besar akan menghasilkan nilai β yang yang tidak terlalu jauh dengan nilai *learning rate*. Hal tersebut berdampak pada proses pencarian model saat *reference vector* yang terdekat pertama dan kedua memiliki kelas yang sama. Proses pencarian model pada kondisi tersebut akan dipengaruhi oleh β yang menggantikan peran *learning rate* pada kondisi yang lain.

Sebagai contoh, akurasi maksimal pada ketiga jenis fitur (fitur tekstur saja, fitur warna saja serta fitur tekstur dan warna) didapatkan pada nilai *m* sebesar 0,05.

Hal ini dikarenakan sistem dapat mencapai model yang paling cocok dengan nilai β yang relatif kecil.

6.4 Hasil dan Analisis Pengujian Epsilon

Pengujian *epsilon* digunakan untuk mengetahui nilai *epsilon* yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. Nilai *epsilon* yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.4. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter *learning rate* = 0,2
2. Nilai parameter pengali *learning rate* = 0,8
3. Nilai m = 0,1
4. Nilai parameter iterasi maksimal = 40
5. Nilai parameter *learning rate* minimal = 0,000001

Tabel 6.5 Hasil Pengujian Epsilon

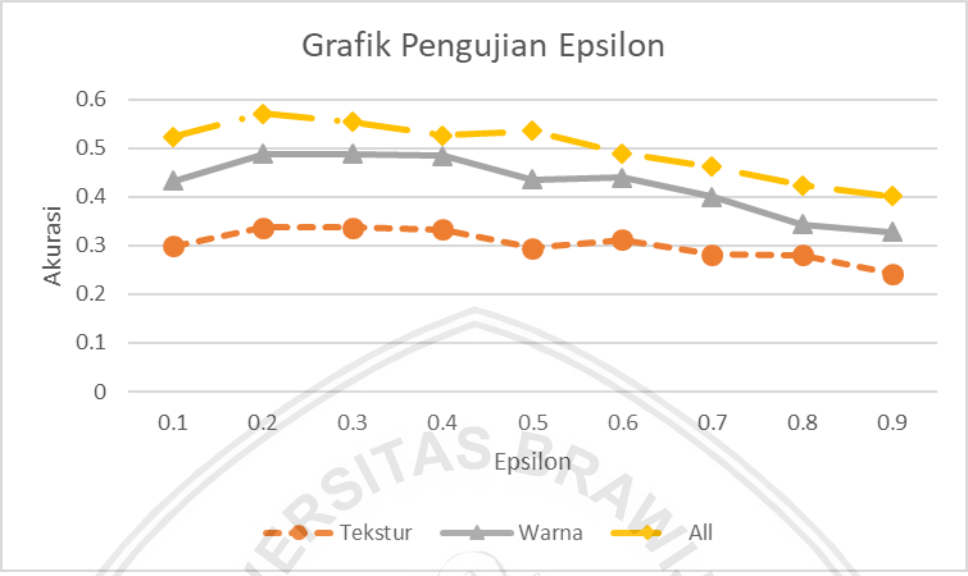
Fitur	Epsilon								
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Tekstur	0.2995	0.3377	0.3375	0.3331	0.2951	0.3119	0.2816	0.2809	0.2425
Warna	0.4331	0.488	0.488	0.4841	0.436	0.4395	0.4004	0.3436	0.3281
Tekstur dan Warna	0.5235	0.5713	0.5538	0.5269	0.5357	0.4893	0.4623	0.4245	0.4014

Hasil dari pengujian epsilon secara lengkap tersaji pada Tabel 6.5. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.4. Klasifikasi menggunakan fitur tekstur saja serta fitur tekstur dan warna mencapai akurasi maksimal pada nilai epsilon sebesar 0,2 dengan akurasi sebesar masing-masing 0,3378 dan 0,5714. Sedangkan klasifikasi menggunakan fitur warna saja mencapai akurasi maksimal sebesar 0,4881 pada nilai epsilon sebesar 0,3.

Nilai epsilon sangat berpengaruh pada pemilihan kondisi pelatihan LVQ3. Nilai epsilon yang kecil menunjukkan bahwa algoritme LVQ3 akan memperhatikan reference vector dengan jarak antara reference vector terdekat pertama dan kedua yang semakin kecil. Apabila syarat tersebut tidak terpenuhi, maka sistem akan melakukan perubahan bobot seperti pada algoritme LVQ2.1.

Sebagai contoh, akurasi maksimal pada ketiga jenis fitur (fitur tekstur saja, fitur warna saja serta fitur tekstur dan warna) didapatkan pada nilai *epsilon* sebesar 0,2 dan 0,3. Hal tersebut menunjukkan bahwa dua reference vector terdekat akan diakomodasi oleh algoritme LVQ3 dengan syarat kedua reference vector tersebut memiliki jarak yang memenuhi Persamaan 2.29 dengan nilai epsilon sebesar 0,2 atau 0,3.

Nilai epsilon yang terlalu kecil dapat mengakibatkan kelas/*reference vector* yang memiliki tingkat kemiripan tinggi tidak terakomodir oleh algoritme LVQ3. Sebaliknya nilai epsilon yang terlalu besar dapat algoritme LVQ3 mengakomodir kelas/*reference vector* yang tidak mirip.



Gambar 6.4 Grafik Pengujian Epsilon

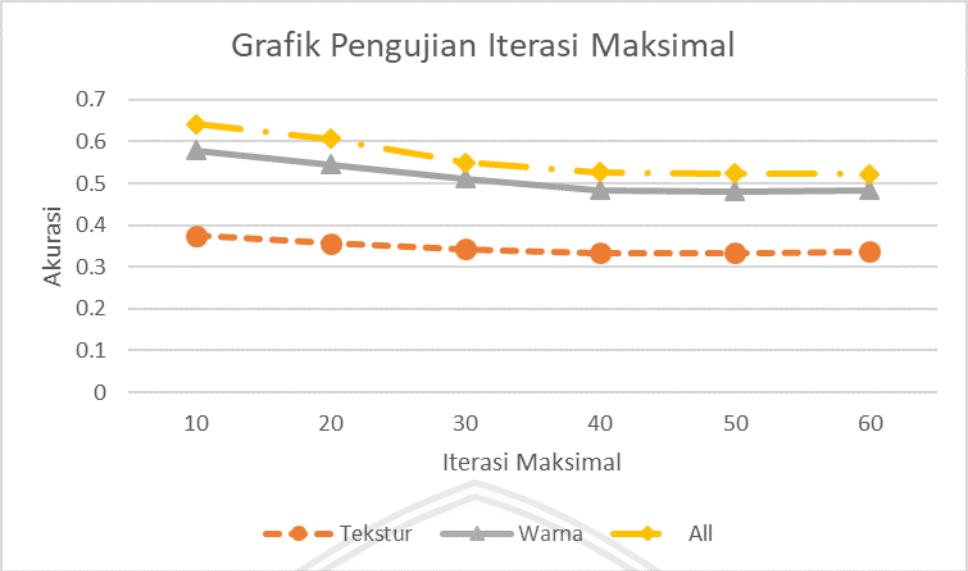
6.5 Hasil dan Analisis Pengujian Iterasi Maksimal

Pengujian iterasi maksimal digunakan untuk mengetahui iterasi maksimal yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. Iterasi maksimal yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.5. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter *learning rate* = 0,2
2. Nilai parameter pengali *learning rate* = 0,8
3. Nilai m = 0,1
4. Nilai *epsilon* = 0,4
5. Nilai parameter *learning rate* minimal = 0,000001

Tabel 6.6 Hasil Pengujian Iterasi Maksimal

Fitur	Iterasi Maksimal					
	10	20	30	40	50	60
Tekstur	0.375403	0.356164	0.34303	0.333122	0.333353	0.336982
Warna	0.578975	0.545334	0.512097	0.484101	0.480127	0.483583
Tekstur dan Warna	0.642051	0.606336	0.550346	0.526959	0.523214	0.522753



Gambar 6.5 Grafik Pengujian Iterasi Maksimal

Hasil dari pengujian iterasi maksimal secara lengkap tersaji pada Tabel 6.6. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.5. Klasifikasi menggunakan fitur tekstur saja, fitur warna saja maupun fitur tekstur dan warna mencapai akurasi maksimal pada iterasi maksimal sebesar 10 dengan akurasi sebesar masing-masing 0,3754 , 0,579 , dan 0,6421.

Hasil dari pengujian iterasi juga menunjukkan bahwa semakin lama sistem dapat membuat model semakin jauh dari nilai ideal sebelum mencapai konvergen pada titik tertentu. Hal tersebut dikarenakan semakin lama model akan mencoba mencari nilai yang dapat mengakomodir semua kelas. Sehingga hal tersebut berdampak pada data uji yang sebelumnya sudah dapat diklasifikasikan dengan benar menjadi tidak tepat.

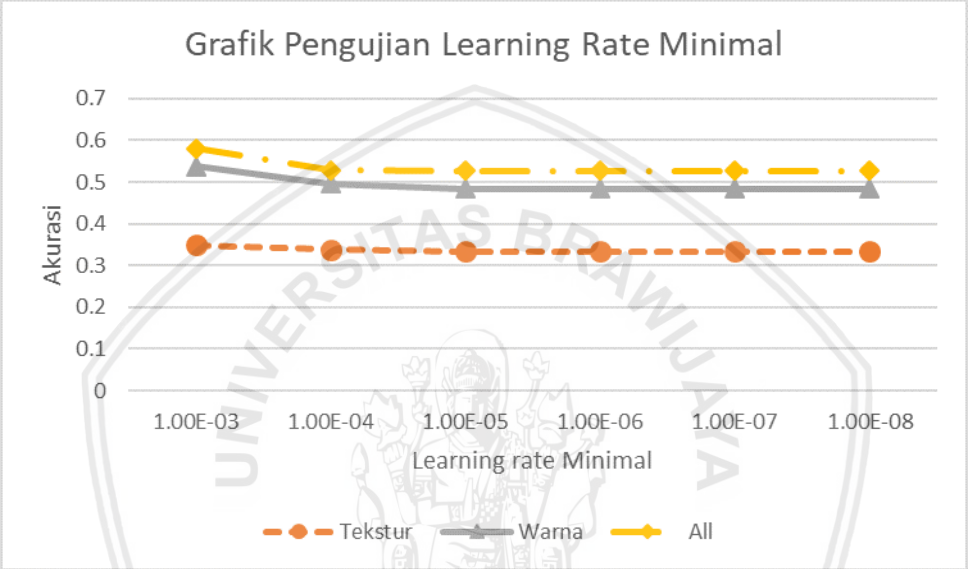
6.6 Hasil dan Analisis Pengujian Learning Rate Minimal

Pengujian *learning rate* minimal digunakan untuk mengetahui nilai *learning rate* minimal yang terbaik sehingga sistem dapat melakukan klasifikasi dengan tepat. Nilai *learning rate* minimal yang diuji menyesuaikan dengan perancangan pengujian pada Subbab 4.3.6. Adapun nilai parameter lain dalam pengujian ini antara lain:

1. Nilai parameter *learning rate* = 0,2
2. Nilai parameter pengali *learning rate* = 0,8
3. Nilai m = 0,1
4. Nilai *epsilon* = 0,4
5. Nilai parameter iterasi maksimal = 40

Tabel 6.7 Hasil Pengujian Learning Rate Minimal

Fitur	Learning rate Minimal					
	0,001	0,0001	0,00001	0,000001	0,000001	0,00000001
Tekstur	0.349194	0.337846	0.333122	0.333122	0.333122	0.333122
Warna	0.538191	0.496256	0.484101	0.484101	0.484101	0.484101
Tekstur dan Warna	0.580357	0.529147	0.526959	0.526959	0.526959	0.526959



Gambar 6.6 Grafik Pengujian Learning Rate Minimal

Hasil dari pengujian *learning rate* minimal secara lengkap tersaji pada Tabel 6.7. Hasil dari pengujian tersebut secara umum dapat digambarkan dengan Gambar 6.6. Klasifikasi menggunakan fitur tekstur saja, fitur warna saja maupun fitur tekstur dan warna mencapai akurasi maksimal pada learning rate minimal sebesar 0,001 dengan akurasi sebesar masing-masing 0,3492 , 0,5382 , dan 0,5804.

Hasil dari pengujian *learning rate* minimal juga menunjukkan bahwa semakin lama sistem dapat membuat model semakin jauh dari nilai ideal sebelum mencapai konvergen pada titik tertentu. Hal tersebut dikarenakan semakin lama model akan mencoba mencari nilai yang dapat mengakomodir semua kelas. Sehingga hal tersebut berdampak pada data uji yang sebelumnya sudah dapat diklasifikasikan dengan benar menjadi tidak tepat.

6.7 Hasil dan Analisis Pengujian Akurasi

Pengujian ini dilakukan untuk mengetahui tingkat akurasi sistem yang didapatkan. Selain itu, pengujian juga dilakukan untuk mengetahui pengaruh penggunaan fitur terhadap akurasi. Nilai parameter yang digunakan pada pengujian ini disesuaikan dengan hasil akurasi maksimal yang didapatkan dari

pengujian parameter yang telah dilakukan sebelumnya. Adapun akurasi maksimal pada masing-masing pengujian dihasilkan pada nilai parameter berikut.

1. Nilai parameter *learning rate* = 0,05
2. Nilai parameter pengali *learning rate* = 0,9
3. Nilai m = 0,05
4. Nilai *epsilon* = 0,2
5. Nilai parameter iterasi maksimal = 10
6. Nilai parameter *learning rate* minimal = 0,001

Tabel 6.8 Hasil Pengujian Akurasi

Fitur	Akurasi
Tekstur	0.317569
Warna	0.45576
Tekstur dan Warna	0.559965

Hasil dari pengujian akurasi secara lengkap tersaji pada Tabel 6.8. Klasifikasi menggunakan fitur tekstur saja, fitur warna saja serta fitur tekstur dan warna dengan nilai parameter berdasarkan hasil pengujian menghasilkan akurasi sebesar masing-masing 0,3176 , 0,4558 , dan 0,55997. Hasil pengujian akurasi menggunakan kombinasi nilai parameter terbaik masih belum dapat menghasilkan akurasi yang lebih tinggi daripada akurasi pada pengujian sebelumnya. Hal ini dikarenakan perubahan pada masing-masing parameter berpengaruh terhadap nilai optimal dari parameter lain. Algoritme optimasi dapat menjadi solusi untuk permasalahan tersebut.

Hasil pengujian akurasi juga menunjukkan bahwa klasifikasi menggunakan kedua fitur tekstur dan warna menghasilkan akurasi yang paling tinggi. Hal tersebut dikarenakan masing-masing kelas memiliki karakteristik warna dan tekstur yang beragam.

Penggunaan salah satu fitur, baik tekstur saja maupun warna saja menghasilkan akurasi yang lebih rendah. Hal tersebut dikarenakan terdapat beberapa kelas/*reference vector* yang memiliki tekstur atau warna yang relatif sama. Sebagai contoh, pada Gambar 6.7 dapat dilihat bahwa kelas mie gepeng dan kelas *fried chicken* sekilas memiliki tekstur yang relatif mirip. Hal ini dibuktikan dengan nilai jarak antara *reference vector* yang mewakili kedua kelas tersebut menunjukkan nilai yang cukup kecil sebesar 0,014025. Jarak antar masing-masing kelas berdasarkan tekstur selengkapnya dapat dilihat pada Lampiran B.2.

Contoh lain adalah pada kelas pisang kuning dan snack *Happy Tos*. Pada Gambar 6.8 dapat dilihat bahwa kedua kelas tersebut dari sisi fitur warna saja memiliki tingkat kemiripan yang cukup tinggi. Hal ini dibuktikan dengan jarak antara reference vector yang mewakili kedua kelas tersebut memiliki nilai yang

cukup kecil sebesar 0.013496938. Jarak antar masing-masing kelas berdasarkan warna selengkapnya dapat dilihat pada Lampiran B.2.

Gambar 6.7 Contoh Citra (a) Kelas Mie Gepeng dan (b) Kelas *Fried Chicken*

Gambar 6.8 Contoh Citra (a) Kelas Pisang Kuning dan (b) Kelas *Happy Tos*

Pada hasil pengujian akurasi maupun pengujian-pengujian sebelumnya dapat dilihat dilihat bahwa akurasi sistem menggunakan fitur warna saja lebih tinggi daripada menggunakan fitur tekstur saja. Hal ini dikarenakan nilai dari fitur warna pada masing-masing citra memiliki nilai standar deviasi yang lebih bervariasi. Hal tersebut dapat dilihat dari rata-rata standar deviasi dari fitur warna sebesar 0.160574336 lebih besar dari rata-rata standar deviasi dari fitur tekstur sebesar 0.108144191. Standar deviasi dari masing-masing fitur dapat dilihat pada Tabel 6.9.

Tabel 6.9 Nilai Standar Deviasi untuk fitur tekstur

No	Nama Fitur	Standar Deviasi
1	Mean X Sudut 0	0.177872
2	Mean Y Sudut 0	0.177908
3	Varians X Sudut 0	0.081345
4	Varians Y Sudut 0	0.081389
5	Energi Sudut 0	0.076628
6	Entropi Sudut 0	0.076628

Tabel 6.9 Nilai Standar Deviasi untuk fitur tekstur (lanjutan)

7	Dissimilarity Sudut 0	0.054996
8	Homogeneity Sudut 0	0.072495
9	Kontras Sudut 0	0.16605
10	Korelasi Sudut 0	0.116132
11	Mean X Sudut 45	0.177872
12	Mean Y Sudut 45	0.177908
13	Varians X Sudut 45	0.081345
14	Varians Y Sudut 45	0.081389
15	Energi Sudut 45	0.076628
16	Entropi Sudut 45	0.076628
17	Dissimilarity Sudut 45	0.054996
18	Homogeneity Sudut 45	0.072495
19	Kontras Sudut 45	0.16605
20	Korelasi Sudut 45	0.116132
21	Mean X Sudut 90	0.177872
22	Mean Y Sudut 90	0.177908
23	Varians X Sudut 90	0.081345
24	Varians Y Sudut 90	0.081389
25	Energi Sudut 90	0.076628
26	Entropi Sudut 90	0.076628
27	Dissimilarity Sudut 90	0.054996
28	Homogeneity Sudut 90	0.072495
29	Kontras Sudut 90	0.16605
30	Korelasi Sudut 90	0.116132
31	Mean X Sudut 135	0.177872
32	Mean Y Sudut 135	0.177908
33	Varians X Sudut 135	0.081345
34	Varians Y Sudut 135	0.081389
35	Energi Sudut 135	0.076628
36	Entropi Sudut 135	0.076628
37	Dissimilarity Sudut 135	0.054996
38	Homogeneity Sudut 135	0.072495
39	Kontras Sudut 135	0.16605
40	Korelasi Sudut 135	0.116132

Tabel 6.10 Nilai Standar Deviasi untuk fitur warna

No	Nama Fitur	Standar Deviasi
1	Mean Channel L	0.165055
2	Varians Channel L	0.138385
3	Skewness Channel L	0.147951
4	Kurtosis Channel L	0.127747
5	Mean Channel A	0.111623
6	Varians Channel A	0.169721
7	Skewness Channel A	0.166887
8	Kurtosis Channel A	0.152233
9	Mean Channel B	0.144935
10	Varians Channel B	0.217482
11	Skewness Channel B	0.156763
12	Kurtosis Channel B	0.228111

BAB 7 PENUTUP

Bab ini memuat kesimpulan dari penelitian yang telah dilakukan. Selain itu, bab ini juga memuat saran untuk pengembangan penelitian ini di kemudian hari.

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan sebelumnya, terdapat beberapa kesimpulan yang didapatkan, antara lain:

1. Klasifikasi jenis makanan dari citra *smartphone* berdasarkan ekstraksi fitur Haralick dan CIE Lab Color Moment menggunakan Learning Vector Quantization menghasilkan akurasi yang cukup baik. Hal tersebut ditunjukkan dengan hasil pengujian pada beberapa parameter LVQ3 dan mencapai akurasi tertinggi sebesar 0.642051 dengan nilai *learning rate* sebesar 0.2, pengali *learning rate* sebesar 0.8, nilai *m* sebesar 0.1, nilai *epsilon* sebesar 0.4, iterasi maksimal sebesar 10 dan *learning rate* minimal sebesar 0.000001.
2. Penggunaan kedua fitur tekstur (*Haralick*) dan warna (*CIE Lab Color Moments*) berpengaruh terhadap hasil akurasi. Hal tersebut ditunjukkan dengan seluruh hasil pengujian yang menunjukkan bahwa hasil akurasi tertinggi dicapai menggunakan fitur tekstur dan warna.

7.2 Saran

Berdasarkan hasil pengujian dan analisis yang telah dilakukan sebelumnya, terdapat beberapa saran yang diberikan untuk penelitian selanjutnya, antara lain:

1. Sistem dapat dikembangkan menggunakan beberapa modifikasi dari algoritme LVQ seperti *Distinctive Sensitive LVQ. Generalized LVQ* atau *Robust Soft LVQ.*
2. Sistem dapat ditambahkan dengan metode seleksi fitur, seperti *Principal Component Analysis, Relief,* Korelasi atau *Information Gain* dengan harapan memperoleh hasil akurasi yang lebih tinggi.
3. Sistem dapat ditambahkan dengan algoritme optimasi seperti *Particle Swarm Optimization*, *Ant Colony Optimization* atau algoritma genetika untuk penentuan *reference vector* awal dan nilai parameter yang paling ideal untuk meningkatkan akurasi yang didapatkan.

DAFTAR PUSTAKA

Adinugroho, S. & Sari, Y. A., 2017. Perbandingan Jaringan Learning Vector Quantization dan Backpropagation pada Klasifikasi Daun Berbasiskan Fitur Gabungan. *Jurnal Informatika & Multimedia,* 9(2), pp. 58-64.

Allibhai, E., 2018. *Hold-out vs. Cross-validation in Machine Learning.* [Online] Available at: https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f
[Diakses 06 Desember 2018].

Bagalkote, I. S. & Vibhute, A. S., 2016. Review on: Texture Discrimination Feature Analysis for Visually Similar Texture of Different Fields. *International Journal for Scientific Research & Development,* 3(9), pp. 851-856.

Brownlee, J., 2016. *Supervised and Unsupervised Machine Learning Algorithms.* [Online]
Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/
[Diakses 29 08 2018].

Brownlee, J., 2018. *k-Fold Cross-Validation.* [Online] Available at: https://machinelearningmastery.com/k-fold-cross-validation/ [Diakses 06 Desember 2018].

Chen, M.-Y.et al., 2012. *Automatic Chinese Food Identification and Quantity Estimation.* Singapore, SIGGRAPH Asia 2012 Technical Briefs.

Dattatherya, Chalam, S. V. & Singh, M. K., 2013. A Generalized Image Authentication Based On Statistical Moments of Color Histogram. *International Journal on Recent Trends in Engineering and Technology,* 8(1), pp. 40-46.

Fausett, L., 2004. *Fundamentals of Neural Networks.* New Jersey: Prentice-Hall.

Gonzales, R. C. & Woods, R. E., 2008. *Digital Image Processing.* New Jersey: Prentice-Hall.

Guyon, I. & Elisseeff, A., 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research 3,* Volume 3, pp. 1157-1182.

Halim, I., Hardy, Dewi, C. & Angkasa, S., 2013. Aplikasi Image Retrieval Menggunakan Kombinasi Metode Color Moment dan Gabor Texture. *JSM STMIK Mikroskil,* 14(2), pp. 109-117.

Hardani, R., 2002. *Pola Makan Sehat.* Jakarta: Kharisma, Woman and Education.

Jahan, S., Rekha, H. S. & Quadri, S. A., 2018. Bird's Eye Review on Food Image Classification using Supervised Machine Learning. *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS),* VII(III), pp. 153-159.

Joutou, T. & Yanai, K., 2009. *A Food Image Recognition System With Multiple Kernel Learning.* Cairo, 16th IEEE International Conference on Image Processing (ICIP).

Kadir, A. & Susanto, A., 2013. *Teori dan Aplikasi Pengolahan Citra.* Yogyakarta: ANDI.

Kawano, Y. & Yanai, K., 2014. *Automatic Expansion of a Food Image Dataset Leveraging Existing Categories with Domain Adaptation.* Zurich, European Conference on Computer Vision.

Kho, D., 2018. *Pengertian Noise (Derau) dan Jenis-jenis Noise.* [Online] Available at: https://teknikelektronika.com/pengertian-noise-derau-dan-jenis-jenis-noise/
[Diakses 02 09 2018].

Kumar, A. A. & Chandrasekhar, S., 2012. Text Data Pre-processing and Dimensionality Reduction Techniques for Document Clustering. *International Journal of Engineering Research & Technology (IJERT),* 1(5), pp. 1-5.

Kumar, T. & Verma, K., 2010. A Theory Based on Conversion of RGB image to Grayscale. *International Journal of Computer Applications,* 7(2), pp. 7-10.

Kusumaningrum, R. & Arymurthy, A. M., 2015. CIELab Color Moments: Alternative Descriptors for LANDSAT Images Classification System. *INKOM,* 8(2), pp. 111-116.

Liu, C. et al., 2016. *DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment.* Wuhan, 14th International Conference on Smart homes and health telematics.

Madenda, S., 2015. *Pengolahan Citra dan Video Digital.* Jakarta: Erlangga.

Mutrofin, S., Izzah, A., Kurniawardhani, A. & Masrur, M., 2014. Optimasi Teknik Klasifikasi Modified k Nearest Neighbor Menggunakan Algoritma Genetika. *Jurnal GAMMA,* 10(1), pp. 1-5.

Nayef, B. H., Hussain, R. I., Sahran, S. & Abdullah, S. N. H. S., 2013. *Brain Imaging Classification Based On Learning Vector Quantization.* Sharjah, First International Conference on Communications, Signal Processing and their Applications.

Patil, N. K., Malemath, V. S. & Yadahalli, R. M., 2011. Color and Texture Based Identification and Classification of food Grains using different Color Models and Haralick features. *International Journal on Computer Science and Engineering (IJCSE),* 3(12), pp. 3669-3680.

Porebski, A., Vandenbroucke, N. & Macaire, L., 2008. *Haralick Feature Extraction from LBP Images for Color Texture Classification.* Sousse, First International Conference on Image Processing Theory, Tools and Applications.

Riduwan, 2004. *Metode Riset.* s.l.:s.n.

Saparinto, C. & Hidayati, D., 2006. *Bahan Tambahan Pangan.* s.l., s.n.

Stricker, M. & Orengo, M., 1995. *Similarity of Color Images.* Washington, Proceedings of SPIE - The International Society for Optical Engineering.

Susanti & Bistara, D. N., 2018. Hubungan Pola Makan Dengan Kadar Gula Darah Pada Penderita Diabetes Mellitus. *Jurnal Kesehatan Vokasional,* 3(1), pp. 29-34.

Tumenggung, I., 2015. Hubungan Pola Makan dengan Kejadian Gout Artritis di RSUD Toto Kabila Kabupaten Bone Bolango. *Health and Nutrition Journal,* Volume 1, pp. 1-12.

World Health Organization, 2018. *Obesity and overweight.* [Online] [Diakses 30 September 2018].

Wulanningrum, R. & Robby, B. F., 2016. Learning Vector Quantization Image for Identification Adenium. *Indonesian Journal of Electrical Engineering and Computer Science,* 4(2), pp. 383-389.

Zuroida, R., 2015. Faktor Risiko Pola Makan dan Hubungannya dengan Penyakit Jantung pada Pria dan Wanita Dewasa di Provinsi Lampung. *Jurnal Kesehatan UNILA,* 5(9), pp. 18-22.