KLASIFIKASI JENIS CITRA MAKANAN TUNGGAL BERDASARKAN FITUR *LOCAL BINARY PATTERNS* DAN *HUE SATURATION VALUE* MENGGUNAKAN *IMPROVED K-NEAREST NEIGHBOR*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Sarah Najla Adha
NIM: 155150200111116

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

PENGESAHAN

KLASIFIKASI JENIS CITRA MAKANAN TUNGGAL BERDASARKAN FITUR *LOCAL BINARY PATTERNS* DAN *HUE SATURATION VALUE* MENGGUNAKAN *IMPROVED K-NEAREST NEIGHBOR*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Sarah Najla Adha
NIM: 155150200111116

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Desember 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Yuita Arum Sari, S. Kom, M. Kom	Randy Cahya Wihandika, S.ST., M.Kom
NIK: 201609 880715 2 001	NIK: 201405 880206 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 13 Desember 2018

Sarah Najla Adha

NIM: 155150200111116

PRAKATA

Penulis ucapkan puji syukur pada Allah SWT, yang telah melimpahkan rahmat dan karunia-nya, sehingga penulis dapat menyelesaikan tugas akhir skripsi dengan judul "Klasifikasi Jenis Citra Makanan Tunggal Berdasarkan Fitur *Local Binary Patterns* dan *Hue Saturation Value* Menggunakan *Improved K-Nearest Neighbor*".

Pada kesempatan ini penulis juga menyampaikan rasa terima kasih kepada pihak – pihak yang telah membantu penulis selama penyusunan tugas akhir skripsi, diantaranya:

1. Ibu Yuita Arum Sari, S.Kom., M.Kom dan Bapak Randy Cahya Wihandika, S.ST., M.Kom selaku dosen pembimbing I dan dosen pembimbing II yang telah meluangkan waktu untuk membimbing dan mengarahkan penulis dalam penyusunan skripsi ini.
2. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T, M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Ketua Prodi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Keluarga dari penulis terutama orang tua dan kakak penulis yang telah memberikan kasih sayang, doa, serta motivasi.
6. Teman-teman seperjuangan kelas Induksi Riset-F, Devinta Setyaningtyas A, Frisma Yessy N, Karunia Ayuningsih, Raiza Nifrina, dan Dwi Ardiyanti atas dukungan, ilmu, motivasi dan hiburan yang diberikan kepada penulis dalam penyelesaian skripsi ini.

Penulis menyadari atas ketidaksempurnaan dalam penyusunan skripsi ini baik dalam teknik penyajian materi maupun pembahasan. Demi kemajuan penulis, saran dan kritik yang sifatnya membangun sangat penulis harapkan. Semoga laporan ini dapat bermanfaat bagi pembaca secara umum dan penulis. Akhir kata penulis ucapkan banyak terima kasih.

Malang, 13 Desember 2018

Penulis

sarahnajladha@student.ub.ac.id

ABSTRAK

Sarah Najla Adha, Klasifikasi Jenis Citra Makanan Tunggal Berdasarkan Fitur *Local Binary Patterns* dan *Hue Saturation Value* Menggunakan *Improved K-Nearest Neighbor*

Pembimbing: Yuita Arum Sari, S.Kom., M.Kom dan Randy Cahya Wihandika, S.ST., M.Kom.

Makanan adalah bahan pokok yang diperlukan oleh tiap makhluk hidup. Makanan yang memiliki kualitas buruk atau tidak sehat dapat menyebabkan penyakit ataupun alergi. Untuk menghindari hal tersebut, teknologi yang berkembang saat ini dapat dimanfaatkan untuk membuat sistem klasifikasi jenis makanan. Klasifikasi jenis makanan dapat dilakukan dengan pengolahan citra digital. Pengolahan citra digital digunakan untuk menganalisis fitur yang terdapat pada citra. Pada penelitian ini, fitur yang digunakan untuk mengklasifikasi jenis citra makanan adalah fitur warna dan tekstur. Ekstraksi fitur warna dilakukan dengan ruang warna *Hue Saturation Value* (HSV) dan fitur tekstur dengan metode *Local Binary Patterns* (LBP). Klasifikasi dilakukan dengan metode *Improved K-Nearest Neighbor (Improved K-NN).* Hasil pengujian nilai *k* menunjukkan bahwa didapatkan akurasi tertinggi sebesar 90,476% dengan nilai *k*=1. Ketika fitur yang digunakan hanya fitur warna, didapatkan nilai akurasi tertinggi 90,476% dengan nilai *k*=1. Ketika fitur yang digunakan hanya fitur tekstur, didapatkan nilai akurasi tertinggi 85,714% dengan nilai *k*=1. Hasil pengujian metode klasifikasi menunjukkan bahwa metode *Improved K-NN* menghasilkan akurasi yang lebih tinggi dari metode K-NN dengan rata-rata akurasi 80,306%. Sehingga hasil klasifikasi terbaik didapatkan dengan penggunaan kombinasi fitur warna dan tekstur dengan metode klasifikasi *Improved K-NN.*

Kata kunci: makanan, *local binary patterns*, *hue saturation value*, *improved k-nn*.

ABSTRACT

Sarah Najla Adha, *Single Food Image Types Classification Based on Local Binary Patterns and Hue Saturation Value Features Using Improved K-Nearest Neighbor*

***Advisor*: Yuita Arum Sari, S.Kom., M.Kom dan Randy Cahya Wihandika, S.ST., M.Kom.**

Food is the basic ingredient needed by every living thing. Foods that have poor or unhealthy quality can cause illness or allergies. To avoid this, current technology can be used to create a food classification system. Classification of food types can be done by digital image processing. Digital image processing is used to analyze features contained in the image. In this study, the features used to classify the types of food images are color and texture features. Color feature extraction is done by Hue Saturation Value (HSV) color space and texture features using the Local Binary Patterns (LBP) method. Classification is done by the Improved K-Nearest Neighbor (Improved K-NN) method. The test results of the k value indicate that the highest accuracy is obtained at 90,476% with a value of k = 1. When the feature used is only a color feature, the highest accuracy value is obtained at 90,476% with a value of k = 1. When the feature used is only a texture feature, the highest accuracy value is obtained 85,714% with a value of k = 1. The results of testing the classification method showed that the Improved K-NN method produced higher accuracy than the K-NN method with an average accuracy of 80.306%. So the best classification results are obtained by using a combination of color and texture features with the Improved K-NN classification method.

Keywords: food, hue saturation value, local binary patterns, improved k-nn.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah serta sistematika pembahasan dari penelitian yang dilakukan.

1.1 Latar Belakang

Makanan adalah bahan pokok yang diperlukan oleh tiap makhluk hidup. Fungsi makanan antara lain untuk pertumbuhan dan perkembangan tubuh, pemeliharaan dan perbaikan sel-sel tubuh yang rusak, pengaturan metabolisme tubuh, penjaga keseimbangan tubuh, pertahanan tubuh terhadap penyakit dan penghasil energi. Agar makanan berfungsi sebagaimana mestinya, kualitas makanan harus diperhatikan. Kualitas tersebut termasuk nilai gizi yang terdapat pada makanan (Amaliyah, 2017). Makanan yang memiliki kualitas buruk atau tidak sehat dapat menyebabkan penyakit ataupun alergi. Untuk menghindari hal tersebut, teknologi yang berkembang saat ini dapat dimanfaatkan untuk membuat sistem klasifikasi jenis makanan. Klasifikasi jenis makanan dapat digunakan untuk meningkatkan kesadaran masyarakat dalam memperhatikan makanan yang dikonsumsi (Farinella et.al, 2016).

Klasifikasi jenis makanan dapat dilakukan dengan pengolahan citra digital. Pengolahan citra digital digunakan untuk menganalisis ciri atau fitur yang terdapat pada citra. Fitur-fitur yang dapat dianalisis dari citra antara lain adalah fitur tekstur, warna dan bentuk. Pada pengenalan citra, fitur yang umum digunakan adalah fitur warna dan tekstur. Dengan menggunakan fitur warna dan tekstur, kemiripan antar citra dapat dikenali lebih baik sehingga cocok digunakan untuk klasifikasi (Soman, Ghorpade, & Sonone, 2012). Pada penelitian yang dilakukan Arivazhagan et al (2010) dilakukan klasifikasi citra makanan dengan fitur warna pada ruang warna *Hue Saturation Value* (HSV) dan fitur tekstur dengan metode *Gray Level Co-Occurance Matrix* (GLCM). Hasilnya didapatkan rata-rata akurasi sebesar 86%. Namun pada penelitian ini citra makanan yang digunakan hanya citra buah.

Fitur warna dapat diekstraksi dari berbagai macam ruang warna. Ruang warna pada citra antara lain *red, green, blue* (RGB), HSV, YUV dan lain-lain. Pada penelitian yang dilakukan oleh Ong dan Punzalan (2014) dilakukan perbandingan antara ruang warna RGB dan HSV dalam mengekstraksi informasi warna pada larutan pewarna hijau. Larutan pewarna hijau diklasifikasikan ke empat belas kelas berdasarkan konsentrasi dari larutan. Hasilnya menunjukkan bahwa lebih mudah mengetahui informasi warna pada grafik dari ruang warna HSV karena data lebih terkelompok berdasarkan kelasnya daripada grafik pada ruang warna RGB. Selain itu, ruang warna HSV merepresentasikan warna sesuai dengan penglihatan manusia, sehingga diperlukan transformasi ruang warna dari RGB ke HSV. Transformasi ruang warna RGB ke HSV tidak memerlukan kompleksitas

perhitungan yang tinggi dibandingkan transformasi ruang warna RGB ke CIE (Singh & Hemachandran, 2012).

Fitur tekstur dapat diekstraksi dengan berbagai macam metode. Metode yang dapat digunakan untuk mendapatkan fitur tekstur antara lain *Local Binary Patterns* (LBP)*, Gray Level Co-Occurrence Matrices* (GLCM) dan lain-lain. Pada penelitian yang dilakukan oleh Reddy, Reddy, dan Reddy (2014) dilakukan pengklasifikasian tekstur kain dengan metode GLCM dan LBP. Akurasi dari hasil klasifikasi kedua metode ekstraksi fitur tekstur tersebut dibandingkan. Hasil klasifikasi dengan metode *Support Vector Machine* (SVM) menunjukkan bahwa metode LBP menghasilkan tingkat akurasi sebesar 50%, lebih besar daripada tingkat akurasi GLCM yaitu 31,25%. LBP merupakan ukuran tekstur *grayscale* yang invarian terhadap pencahayaan yang berbeda (Turiyanto, Purwanto, & Dikairono, 2014).

Setelah fitur warna dan tekstur didapatkan, dilakukan pengklasifikasian. Metode-metode yang dapat digunakan untuk klasifikasi antara lain Naïve-Bayes, *K-Nearest Neighbor (K-NN), Improved K-Nearest Neighbor* dan lain-lain. *Improved K-Nearest Neighbor* adalah pengembangan dari metode K-NN, yang memiliki performa lebih baik daripada K-NN. Pada penelitian yang dilakukan Wagle, Mangai dan Kumar (2013) dalam mengklasifikasikan citra medis dengan K-NN dan *Improved* K-NN, ketika nilai *k* bernilai 5 akurasi yang dihasilkan *Improved K-NN* sebesar 94,44% sedangkan akurasi yang dihasilkan K-NN adalah 88,89%.

Berdasarkan permasalahan di atas, penulis mengusulkan untuk melakukan pengklasifikasian citra makanan untuk mengetahui jenis makanan. Fitur yang digunakan untuk klasifikasi citra makanan adalah warna dan tekstur. Metode yang digunakan untuk fitur tekstur adalah LBP dan untuk fitur warna menggunakan ruang warna HSV. Pengklasifikasian fitur menggunakan metode *Improved K-Nearest Neighbor* dengan harapan mendapatkan hasil akurasi yang maksimal.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah penelitian ini adalah sebagai berikut.

1. Bagaimana hasil evaluasi klasifikasi jenis citra makanan tunggal berdasarkan fitur HSV dan LBP dengan *Improved K-Nearest Neighbor*?
2. Bagaimana pengaruh ekstraksi fitur HSV dan LBP terhadap hasil klasifikasi jenis citra makanan tunggal?

1.3 Tujuan

Tujuan dalam penelitian ini adalah sebagai berikut.

1. Mengetahui hasil evaluasi klasifikasi jenis citra makanan tunggal berdasarkan fitur LBP dan HSV dengan *Improved K-Nearest Neighbor*.

2. Mengetahui pengaruh ekstraksi fitur HSV dan LBP terhadap hasil klasifikasi pengenalan citra makanan tunggal.

1.4 Manfaat

Manfaat yang didapatkan dari penelitian ini adalah sebagai berikut.

1. Memenuhi salah satu persyaratan kurikulum yang telah diwajibkan oleh Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
2. Mengimplementasikan ilmu yang diperoleh selama perkuliahan khususnya dibidang komputasi cerdas.

1.5 Batasan Masalah

Pada penelitian ini, permasalahan dibatasi oleh hal-hal sebagai berikut.

1. Data yang digunakan merupakan data primer berupa dua puluh satu jenis citra makanan tunggal.
2. Data diambil menggunakan perangkat *smartphone* iPhone 7+.
3. Data latih yang digunakan sebanyak 210 data. Sepuluh data latih dari masing-masing kelas.
4. Data uji yang digunakan sebanyak 105 data. Lima data uji dari masing-masing kelas.

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penyusunan laporan ini bertujuan untuk memberikan gambaran dan penjelasan dari penelitian ini. Laporan penelitian ini terdiri dari beberapa bab antara lain sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini memuat latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini memuat dasar teori yang berhubungan dengan topik penelitian dan kajian pustaka dari penelitian-penelitian terdahulu tentang HSV, LBP dan *Improved K-NN* yang dapat menguatkan dasar teori.

BAB 3 METODOLOGI

Bagian ini menjelaskan urutan aktivitas yang dilakukan serta penjelasannya secara lengkap oleh peneliti dalam melakukan penelitian.

BAB 4 PERANCANGAN

Bagian ini menjelaskan tentang perancangan dari sistem klasifikasi citra makanan tunggal. Perancangan sistem meliputi diagram alir dan manualisasi.

BAB 5 IMPLEMENTASI

Bagian ini menjelaskan tentang implementasi yang dilakukan untuk membuat sistem klasifikasi citra makanan tunggal.

BAB 6 ANALISIS DAN PENGUJIAN

Bagian ini menjelaskan tentang analisis serta pengujian dari penelitian yang dilakukan mengenai klasifikasi citra makanan tunggal.

BAB 7 PENUTUP

Bagian ini menjelaskan tentang kesimpulan yang dapat diambil dari penelitian yang telah dilakukan, serta saran terhadap penelitian yang dilakukan dan harapan untuk penelitian selanjutnya.

BAB 2
LANDASAN KEPUSTAKAAN

Bab ini berisi pejabaran teori-teori yang bersangkutan dengan penulisan skripsi ini. Teori-teori tersebut termasuk penelitian sebelumnya, *pre-processing,* LBP, HSV*,* normalisasi, klasifikasi *Improved K-Nearest Neighbor,* dan akurasi.

2.1 Kajian Pustaka

Pada penelitian yang diteliti oleh Reddy, Reddy, dan Reddy dilakukan klasifikasi tekstur kain dengan menggunakan metode LBP dan GLCM. Tekstur kain diklasifikasikan berdasarkan kerusakan yang ada pada kain tersebut. Hasil ekstraksi fitur kain dari kedua metode kemudian diklasifikasikan dengan metode SVM, K-NN, dan *Probabilistic Neural Networks* (PNN). Dari ketiga metode klasifikasi yang digunakan hasilnya menunjukkan bahwa klasifikasi tekstur kain dengan metode LBP menghasilkan tingkat akurasi yang lebih tinggi daripada hasil klasifikasi dengan metode GLCM.

Pada penelitian yang diteliti oleh Amynarto, Sari dan Wihandika (2018) dilakukan pengenalan emosi berdasarkan ekspresi mikro dengan metode LBP dan metode klasifikasi *K-Nearest Neighbor* (K-NN). Metode LBP digunakan untuk mendapatkan fitur ekspresi mikro pada wajah. Fitur-fitur ekspresi mikro yang didapatkan diklasifikasikan berdasarkan emosinya dengan metode *K-Nearest Neighbor* (K-NN). Pengujian dilakukan dengan mencari nilai *k* yang memberikan hasil akurasi yang optimal. Hasil pengujian nilai *k* menunjukkan bahwa saat *k* bernilai 5 dan 7 didapatkan nilai akurasi yang sama yaitu 56,03%.

Pada penelitian yang diteliti oleh Ong dan Punzalan (2014) dilakukan perbandingan ekstraksi fitur warna menggunakan ruang warna HSV dan RGB. Objek penelitian yang digunakan adalah larutan pewarna hijau yang mirip dengan warna alga. Terdapat empat belas kelas warna berdasarkan konsentrasi larutan pewarna. Selanjutnya dilakukan pengamatan pada grafik kumpulan data tiap kelas menggunakan ruang warna HSV dan RGB. Hasilnya menunjukkan bahwa lebih mudah mengetahui informasi warna pada grafik dari ruang warna HSV karena data lebih terkelompok berdasarkan kelasnya daripada grafik pada ruang warna RGB.

Pada penelitian yang diteliti oleh Usuman, Dharmawan dan Frisky (2012) dilakukan pendeteksian kulit manusia dengan segmentasi warna pada tipe citra HSV. Penelitian ini dilakukan dengan mencari nilai *Hue* dan *Saturation* pada citra HSV dan menentukan rentang nilai kulit manusia. Selanjutnya citra disegmentasi dan dilakukan *low pass filtering* pada hasil segmentasi untuk menghilangkan *noise.* Hasilnya kemudian dideteksi untuk membedakan kulit manusia yang berupa muka dan selain muka. Metode ini dapat bekerja dengan kecepatan rata-rata 1,4 *frame* per detik dengan akurasi sebesar 87,86%. Hasil deteksi dipengaruhi oleh gerakan objek, posisi dan arah objek wajah terhadap kamera,

jumlah objek orang dalam satu *frame*, jarak objek kulit manusia dengan kamera serta pencahayaan.

Pada penelitian yang diteliti oleh Wagle, Mangai dan Kumar (2013) dilakukan pengklasifikasian citra medis menggunakan metode K-NN dan *Improved K-NN.* Objek penelitian yang digunakan merupakan citra retina yang dibedakan menjadi kelas normal dan kelas parah. Penelitian ini dimulai dengan melakukan *pre-processing* data latih. Selanjutnya dilakukan ekstraksi fitur dengan fitur statistik. Fitur yang diekstraksi antara lain rata-rata, deviasi standar, *skewness,* dan kurtosis. Kemudian dilakukan seleksi fitur dengan teknik yang disebut CfssubsetEval. Hasil seleksi fitur selanjutnya digunakan untuk menghitung bobot dari citra tersebut untuk kemudian digunakan untuk pengklasifikasian dengan *Improved K-NN.* Hasil dari penelitian tersebut menunjukkan bahwa pengklasifikasian dengan metode *Improved K-NN* memiliki tingkat akurasi yang lebih tinggi daripada pengklasifikasian dengan K-NN tradisional. Saat *k* bernilai 5, *Improved K-NN* memiliki tingkat akurasi sebesar 94,44% sedangkan K-NN memiliki tingkat akurasi sebesar 88,89%.

2.2 *Pre-Processing*

Pre-processing adalah tahapan awal dalam pengolahan citra digital. *Pre-processing* merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada citra sebelum dilakukan tahapan selanjutnya. Pada penelitian ini terdapat beberapa tahapan yang dilakukan dalam *pre-processing.* Tahapan yang dilakukan antara lain:

1. Mengubah ukuran citra menjadi 500x500 piksel.
2. Mengonversi citra RGB ke citra HSV.
3. Mengubah citra HSV ke *grayscale.*
4. Memberikan filter *Gaussian Blur* dengan kernel 5x5.
5. Melakukan segmentasi citra dengan *Otsu* dengan *threshold* 120 dan mengubah citra menjadi biner.
6. Melakukan erosi dan dilasi pada citra.
7. Melakukan *masking*.

2.3 *Hue Saturation Value* (HSV)

Hue Saturation Value (HSV) adalah dua representasi alternatif dari model warna RGB. Ruang warna merepresentasikan warna seperti yang dilihat mata manusia (Neneng & Fernando, 2017). HSV mencirikan warna berdasarkan *hue, saturation* dan *value (brightness).* Untuk mengonversi citra RGB ke ruang warna HSV dilakukan perhitungan dengan persamaan berikut (Singh & Hemachandran, 2012).

$$H = cos^{-1} \frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}} \quad (2.1)$$

$$S = 1 - \frac{3[\min(R,G,B)}{R+G+B} \quad (2.2)$$

$$V = \left(\frac{R+G+B}{3}\right) \quad (2.3)$$

Terdapat modifikasi perhitungan pengubahan ruang warna dari RGB ke HSV yang dibuat oleh OpenCV. Persamaan yang digunakan oleh OpenCV menggunakan operasi yang lebih sederhana dan menghasilkan citra HSV yang baik. Untuk mengonversi citra RGB ke ruang warna HSV dilakukan perhitungan dengan persamaan modifikasi berikut (OpenCV, 2018).

$$V = Maximum\ (R, G, B) \quad (2.4)$$

$$S = \begin{cases} 0, \ jika\ V = 0 \\ \frac{Maximum\ (R,G,B) - Minimum\ (R,G,B)}{Maximum\ (R,G,B)}, \ jika\ V > 0 \end{cases} \quad (2.5)$$

$$H = \begin{cases} 0, \ jika\ Max = Min & (2.6) \\ \frac{60*(G-B)}{Max-Min}, \ jika\ V = R & (2.7) \\ 60 * \left[2 + \frac{B-R}{Max-Min}\right], \ jika\ V = G & (2.8) \\ 60 * \left[4 + \frac{R-G}{Max-Min}\right], \ jika\ V = B & (2.9) \end{cases}$$

Pada ruang warna HSV, *hue* merupakan warna pada citra. *Saturation* adalah intensitas dari warna pada citra. *Brightness* merupakan tingkat cahaya dari warna tersebut (PM & Chezian, 2013).

2.3.1 Ekstraksi Fitur Warna HSV

Untuk mengkalsifikasikan fitur warna HSV, digunakan beberapa nilai dari masing-masing *channel* H, S, dan V. Nilai-nilai yang digunakan antara lain nilai *mean,* deviasi standar, dan *skewness.* Tujuan dari perhitungan nilai *mean,* deviasi standar, dan *skewness* adalah untuk menemukan citra dengan komposisi warna yang mirip untuk pengklasifikasian. *Mean,* deviasi standar, dan *skewness* dijelaskan dan dapat dihitung dengan persamaan berikut.

1. *Mean*

   Mean merupakan rata-rata nilai piksel (P_{ij}) pada tiap channel H, S, dan V. Rumus perhitungan *mean* ditunjukkan pada Persamaan 2.10.

   $$\mu = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} P_{ij} \quad (2.10)$$

   dimana M dan N secara berturut-turut adalah nilai panjang dan lebar piksel pada gambar (Sari, Dewi, & Fatichah, 2014).

2. Deviasi standar

Deviasi standar merupakan nilai statistik yang menunjukkan persebaran data dalam sampel. Rumus perhitungan deviasi standar ditunjukkan pada Persamaan 2.11.

$$\sigma, = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij} - \mu)^2} \quad (2.11)$$

dimana M dan N secara berturut-turut adalah nilai panjang dan lebar piksel pada gambar.

3. *Skewness*

Skewness merupakan kemiringan suatu kurva dilihat dari perbedaan letak mean, median, dan modusnya. Rumus perhitungan *skewness* ditunjukkan pada Persamaan 2.12.

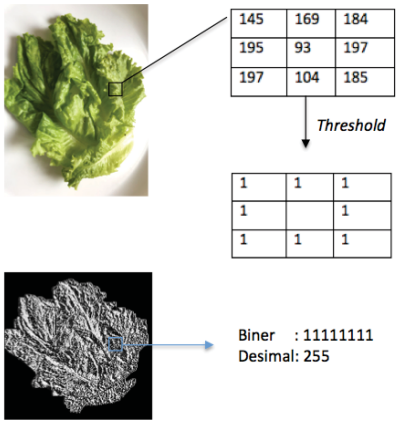
$$\theta = \sqrt[3]{(\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij} - \mu)^3} \quad (2.12)$$

dimana M dan N secara berturut-turut adalah nilai panjang dan lebar piksel pada gambar (Singh & Hemachandran, 2012).

2.4 *Local Binary Patterns* (LBP)

LBP merupakan ukuran tekstur *grayscale* yang invarian terhadap pencahayaan yang berbeda pertama kali diperkenalkan oleh Ojala et al (1994). LBP ampuh mendeskripsikan suatu tekstur, mempunyai pembeda yang akurat dan juga memiliki toleransi terhadap perubahan *grayscale* yang *monotonic* (Turiyanto, Purwanto, & Dikairono, 2014).

LBP adalah kode biner yang merepresentasikan tekstur lokal. Untuk menghitung nilai LBP, dilakukan pembandingan intensitas nilai piksel tengah dengan intensitas piksel tetangganya. Ilustrasi perhitungan LBP ditunjukkan pada Gambar 2.1.



Gambar 2.1 Perhitungan piksel LBP

Nilai desimal dari 8-bit yang dihasilkan dapat dinyatakan dalam persamaan berikut (Cusano, Napoletano, & Schettini, 2014).

$$LBP_{n,r}(\hat{g}) = \sum_{p=0}^{n-1} s(g_p - \hat{g})2^p \quad (2.13)$$

Fungsi $s(x)$ didefinisikan sebagai berikut.

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad (2.14)$$

Keterangan:

$\hat{g}$: nilai piksel

g_p : nilai piksel tetangga

n : jumlah tetangga

r : nilai radius

2.5 Normalisasi Data

Normalisasi data merupakan salah satu proses *pre-processing* data. Dengan normalisasi dapat ditentukan rentang yang baru dari sekumpulan data. Salah satu metode sederhana yang dapat digunakan untuk normalisasi adalah *Min-Max Normalization. Min-Max Normalization* menormalisasi data dan tetap menjaga hubungan antar data (Patro & Sahu, 2015). Untuk melakukan normalisasi data dengan *Min-Max Normalization* data dihitung dengan Persamaan 2.15.

$$A' = \left(\frac{A - \min value\ of\ A}{\max value\ of\ A - \min value\ of\ A}\right) * (D - C) + C \quad (2.15)$$

Keterangan:

A' : Data hasil normalisasi

A : Data sebelum dinormalisasi

Min value of A : Nilai terkecil dari *A*

Max value of A : Nilai terbesar dari *A*

C : Nilai terkecil dari *A'*

D : Nilai terbesar dari *A'*

2.6 *Improved K-Nearest Neighbor*

Improved K-Nearest Neighbor adalah pengembangan dari metode K-NN, dimana terdapat perbedaan pada penentuan nilai *k*. Pada *Improved K-NN* dilakukan perhitungan bobot pada tiap data latih berdasarkan data uji. Perhitungan bobot dilakukan dengan langkah-langkah sebagai berikut (Wagle, Mangai, & Kumar, 2013).

1. Tentukan nilai S_{size} yaitu ukuran ketetanggan dari data latih untuk mencari bobot dari data latih.

2. Bobot data latih dihitung dengan Persamaan 2.16.

$$I\,Weight(a) = \frac{1}{S_{size}} \sum_{i=1}^{S_{size}} f(class(a), class(S_i(a)) \quad (2.16)$$

Keterangan:

class(a) : kelas data

$S_i(a)$: data tetangga ke-*i*

$class(S_i(a))$: kelas data $S_i(a)$

Fungsi yang digunakan untuk perhitungan bobot ditunjukkan pada Persamaan 2.17.

$$f(x, y) = \begin{cases} 1, x = y \\ 0, x \neq y \end{cases} \quad (2.17)$$

Selanjutnya setiap data uji diklasifikasikan dengan langkah-langkah sebagai berikut (Wagle, Mangai, & Kumar, 2013).

1. Hitung jarak antara data uji dan data latih dengan jarak Euclidean yang ditunjukkan pada Persamaan 2.18.

$$D(a, b) = \sum_{i=1}^{A} \sqrt{(a_i - b_i)} \quad (2.18)$$

Keterangan:

A : jumlah fitur

a_i : fitur data latih ke-*i*

b_i : fitur data uji ke-*i*

2. Dengan jarak yang telah dihitung, maka didapatkan data ketetanggaan sejumlah k.
3. Dengan bobot yang telah dihitung sebelumnya, dilakukan perhitungan bobot untuk data uji dengan Persamaan 2.19.

$$W(i) = I\,Weight(i) * \frac{1}{D + 0.5} \quad (2.19)$$

4. Selanjutnya dilakukan penjumlahan bobot data uji dan bobot dari data ketetanggaan yang memiliki kelas yang sama.
5. Kelas data uji ditentukan dari bobot kelas yang memiliki jumlah terbesar.

2.7 Akurasi

Akurasi adalah metode evaluasi yang menghitung tingkat kedekatan antara nilai prediksi atau nilai yang diberikan oleh sistem dengan nilai sebenarnya (Andono, Sutojo, & Muljono, 2017). Rumus perhitungan nilai akurasi ditunjukkan pada Persamaan 2.20.

$$Akurasi = \frac{jumlah\ prediksi\ benar}{jumlah\ seluruh\ data} * 100\% \quad (2.20)$$

BAB 3
METODOLOGI

Bab ini berisi penjelasan tentang metode dan tahapan yang dilakukan dalam penelitian. Hal-hal yang dibahas pada bab ini antara lain tipe penelitian, strategi penelitian, lokasi penelitian, metode pengumpulan data, peralatan pendukung serta jadwal penelitian.

3.1 Tipe Penelitian

Tipe penelitian ini adalah penelitian non-implentatif analitik. Penelitian non-implementatif menitikberatkan investigasi terhadap analisis hubungan antar fenomena yang sedang dikaji untuk kemudian menghasilkan hasil analisis ilmiah.

3.2 Strategi Penelitian

Strategi penelitian menjelaskan tentang alur dari sistem yang ditunjukkan pada Gambar 3.1.

Gambar 3.1 Diagram alir strategi penelitian

Pada penelitian ini masukan yang digunakan berupa citra makanan tunggal. Kemudian dilakukan *pre-processing* pada citra untuk memfokuskan citra ke objek penelitian. Selanjutnya dilakukan ekstraksi fitur warna dengan HSV dan ekstraksi fitur tekstur dengan metode LBP. Hasil ekstraksi dari fitur warna dan tekstur kemudian dinormalisasi dan diklasifikasikan berdasarkan jenis makanan dengan metode *Improved K-Nearest Neighbor (K-NN)*. Hasil klasifikasi berupa nama jenis makanan merupakan keluaran dari sistem ini.

3.3 Lokasi Penelitian

Penelitian ini berlokasi di Lantai 9 Gedung F Ruang Grup Riset Computer Vision F9.3, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

3.4 Metode Pengumpulan Data

Data yang digunakan pada penelitian ini diambil oleh peneliti langsung dengan menggunakan kamera perangkat *smartphone.* Data yang diambil berupa citra makanan tunggal dengan ketentuan sebagai berikut.

1. Objek penelitian yang diambil citranya berupa dua puluh satu jenis makanan tunggal.
2. Waktu pengambilan citra adalah pukul 10.00 – 13.00 WIB pada tanggal 28 Agusutus 2018 di Lantai 9 Gedung F Ruang Grup Riset Computer Vision F9.3, Fakultas Ilmu Komputer, Universitas Brawijaya.
3. *Smartphone* yang digunakan bertipe iPhone 7+.
4. Pengambilan citra makanan tunggal dilakukan secara tegak lurus dan dari kemiringan yang berbeda.
5. Citra makanan diambil sebelum dimakan, setelah dimakan ¼ bagian dan setelah dimakan ½ bagian. Namun pada citra ayam goreng, selada, nasi merah, Soba Mie, dan Milo Nugget citra diambil hanya dalam keadaan sebelum dimakan.

Jenis makanan tunggal yang digunakan pada penelitian ini ditunjukkan pada Tabel 3.1. Sampel data uji dan data latih ditunjukkan pada Tabel 3.2.

Tabel 3.1 Jenis dataset

No	Jenis Makanan
1	Donat
2	Roti gandum
3	Roti tawar
4	Telur ceplok
5	Ayam goreng
6	Rendang
7	Timun
8	Selada
9	Tomat

Tabel 3.1 Jenis dataset (Lanjutan)

No	Jenis Makanan
10	Stroberi
11	Pisang hijau
12	Pisang kuning
13	Jeruk oranye
14	Nasi merah
15	Oreo
16	Beng-beng
17	Soba mie
18	Gery Salut
19	Biskuat
20	Milo Nugget
21	Genji Pie

Tabel 3.2 Sampel data uji dan data latih

No	Jenis Makanan	Data Latih	Data Uji
1	Donat		
2	Roti gandum		
3	Tomat		
4	Pisang hijau		
5	Oreo		

3.5 Teknik Analisis Data

Data hasil klasifikasi dianalisis untuk mengetahui kinerja sistem. Hasil klasifikasi dibandingkan dengan kelas sebenarnya untuk mengetahui nilai akurasi. Analisis dilakukan berdasarkan nilai *k* yang digunakan pada klasifikasi dan fitur yang digunakan untuk klasifikasi.

3.6 Peralatan Pendukung

Peralatan pendukung penelitian ini digunakan untuk pengembangan sistem klasifikasi citra makanan tunggal. Peralatan pendukung yang dibutuhkan antara lain adalah sebagai berikut.

1. Spesifikasi Perangkat Keras

Untuk mengimplementasikan sistem klasifikasi citra makanan tunggal dibutuhkan perangkat keras dengan spesifikasi pada Tabel 3.3.

Tabel 3.3 Spesifikasi perangkat keras

Komponen	Spesifikasi
Processor	Intel Core 2 Duo
Memory	2 GB
Harddisk	160 GB

2. Spesifikasi Perangkat Lunak

Untuk mengimplementasikan sistem klasifikasi citra makanan tunggal dibutuhkan perangkat lunak dengan spesifikasi pada Tabel 3.4.

Tabel 3.4 Spesifikasi perangkat lunak

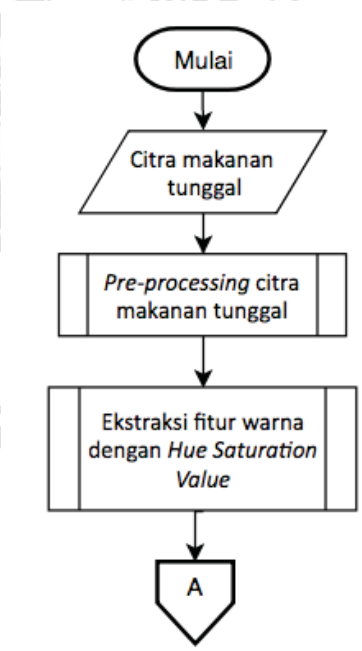
Komponen	Spesifikasi
Sistem Operasi	OS X El Capitan
Bahasa Pemrograman	Python
Tools Pemrograman	Spyder

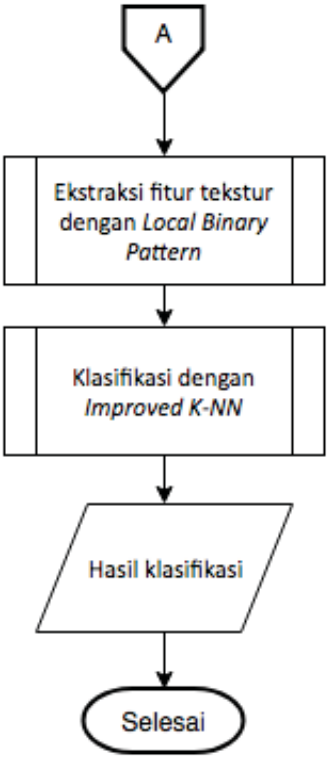
BAB 4
PERANCANGAN

Bab ini berisi penjelasan tentang penerapan tahapan perancangan dari sistem klasifikasi jenis citra makanan tunggal dengan fitur warna HSV, fitur tekstur LBP, normalisasi, dan metode klasifikasi *Improved K-NN.*

4.1 Perancangan Algoritme Proses Implementasi

Pada penelitian ini proses utama yang diimplementasikan ke program antara lain adalah *pre-processing,* ekstraksi fitur warna dengan HSV, ekstraksi fitur tekstur dengan LBP dan klasifikasi dengan metode *Improved K-NN.* Tahap *pre-processing* merupakan tahapan pengolahan citra masukan untuk menghasilkan citra yang dapat diproses pada tahap selanjutnya. Tahapan selanjutnya adalah ekstraksi fitur warna HSV. Pada tahap ini dilakukan pengubahan ruang warna citra dari RGB ke HSV kemudian dicari nilai *mean*, deviasi standar dan *skewness* dari masing-masing *channel* HSV sebagai fitur untuk klasifikasi. Tahapan selanjutnya adalah ekstraksi fitur tekstur LBP. Pada tahap ini dilakukan pengubahan citra ke *grayscale* untuk kemudian didapatkan nilai piksel baru dari perhitungan LBP. Kemudian diambil nilai histogram dari nilai piksel baru sebagai fitur untuk klasifikasi. Selanjutnya fitur-fitur yang telah didapatkan diklasifikasi dengan metode *Improved K-NN.* Alur dari proses implementasi ditunjukkan pada Gambar 4.1.

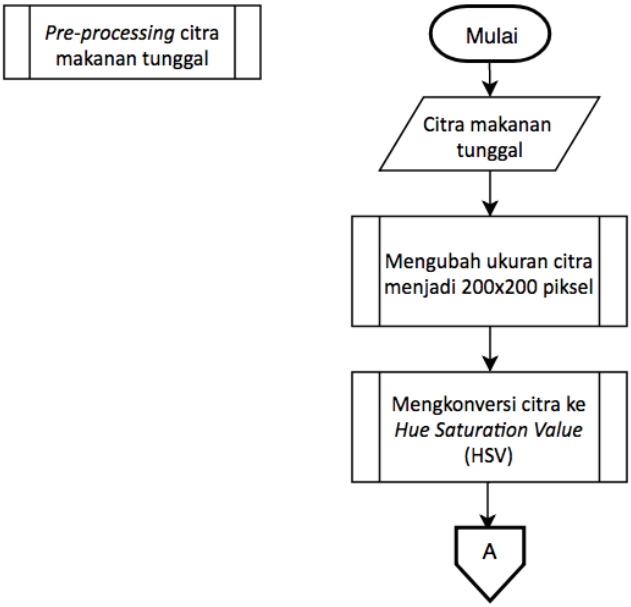




Gambar 4.1 Diagram alir perancangan sistem

4.1.1 *Pre-processing*

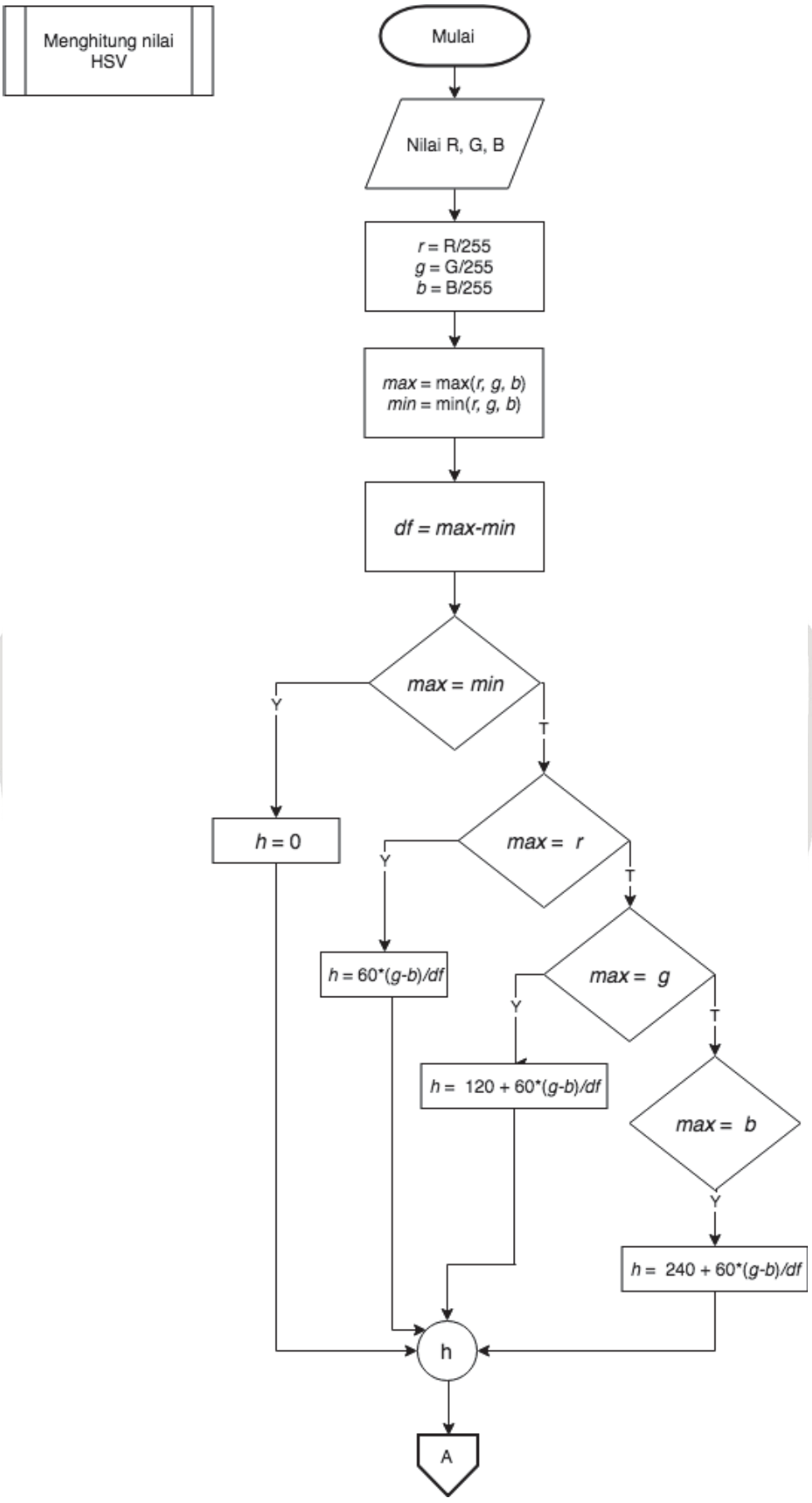
Pre-processing adalah tahapan awal dalam pengolahan citra digital. Pre-processing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada citra sebelum dilakukan tahapan selanjutnya. Alur dari tahapan pre-processing ditunjukkan pada Gambar 4.2.

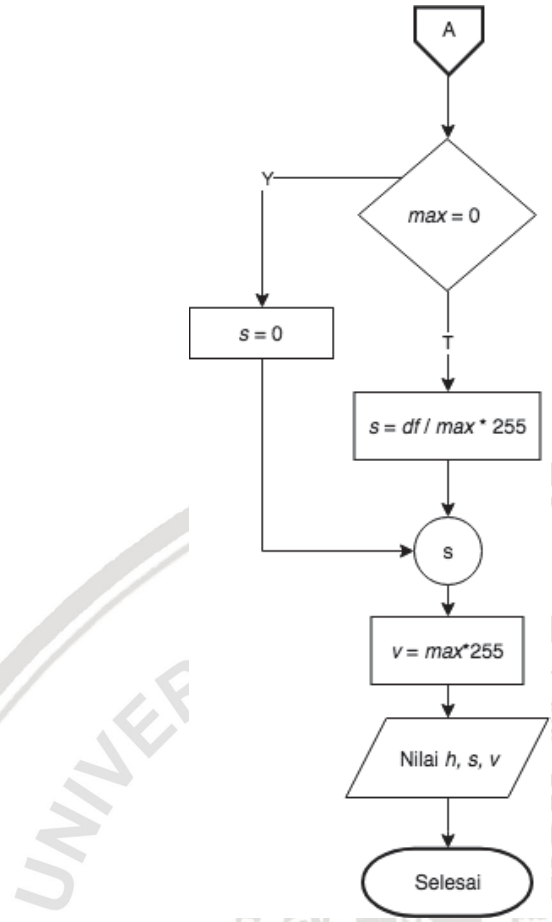


Gambar 4.2 Diagram alir *pre-processing* citra makanan tunggal

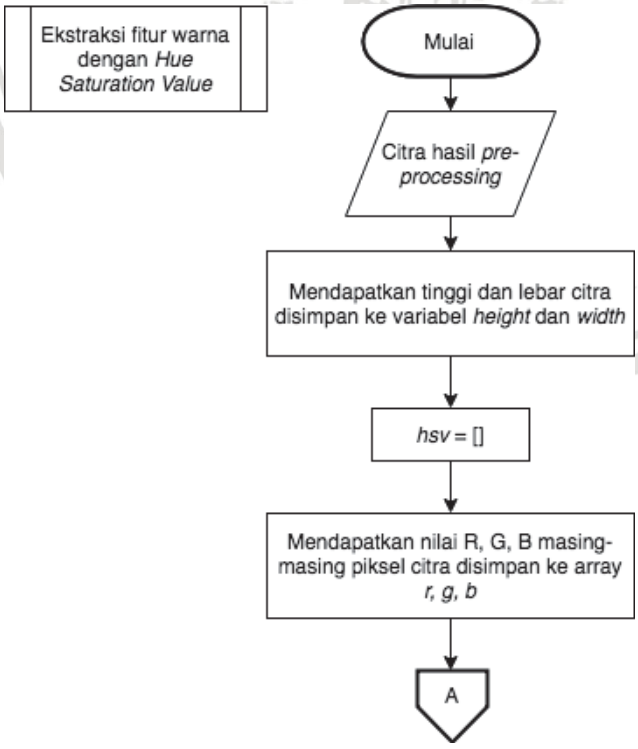
4.1.2 Ekstraksi fitur warna HSV

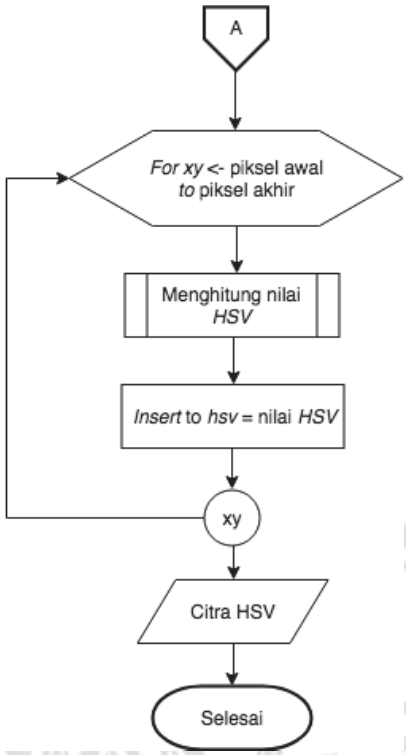
Pada ekstraksi fitur warna HSV dilakukan konversi citra dari ruang warna RGB ke HSV. Terdapat 2 proses yang dilakukan dalam tahap ini. Proses yang dilakukan antara lain melakukan perhitungan nilai HSV untuk masing-masing piksel citra dan proses penggabungan masing-masing nilai HSV pada piksel untuk menghasilkan citra HSV. Proses perhitungan nilai HSV untuk masing-masing citra ditunjukkan pada Gambar 4.3 dan proses penggabungannya ditunjukkan pada Gambar 4.4.





Gambar 4.3 Diagram alir perhitungan nilai HSV

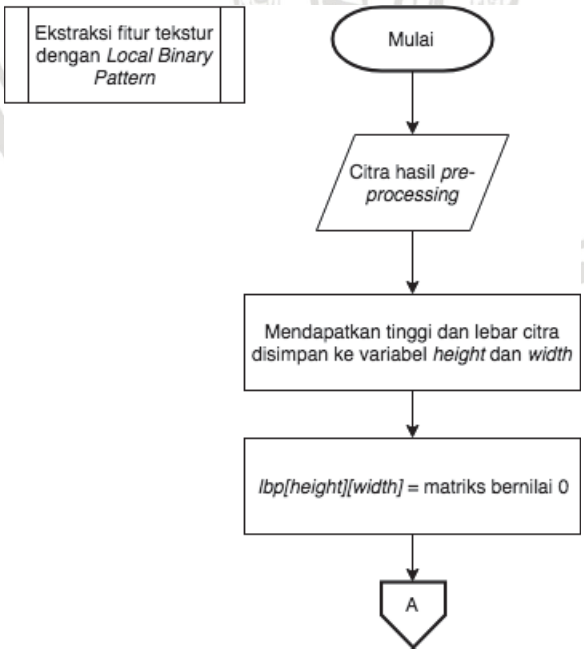


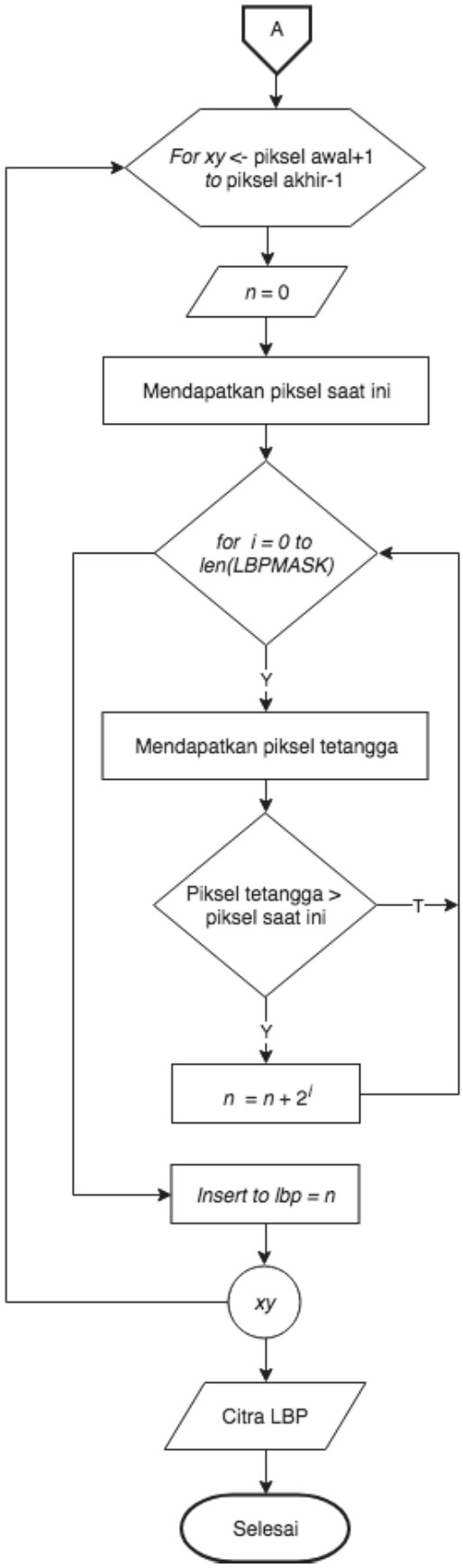


Gambar 4.4 Diagram alir ekstraksi fitur warna HSV

4.1.3 Ekstraksi fitur tekstur LBP

Pada ekstraksi fitur tekstur dengan LBP dilakukan perbandingan antara sebuah piksel dan piksel tetangganya. Selanjutnya dilakukan *thresholding* untuk mendapatkan nilai piksel LBP. Piksel tetangga yang telah melewati proses *thresholding* diurutkan kemudian diubah ke bilangan desimal. Bilangan desimal tersebut adalah nilai piksel LBP. Proses mendapatkan citra LBP ditunjukkan pada Gambar 4.5.

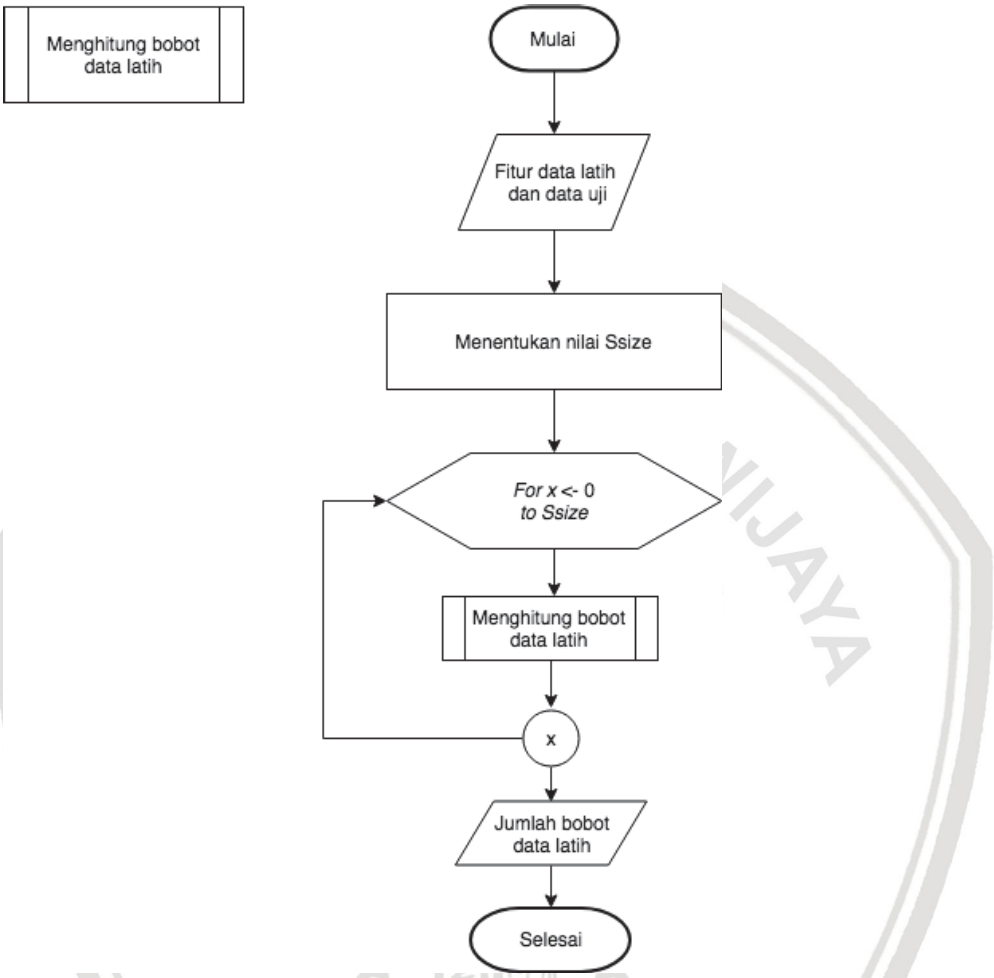




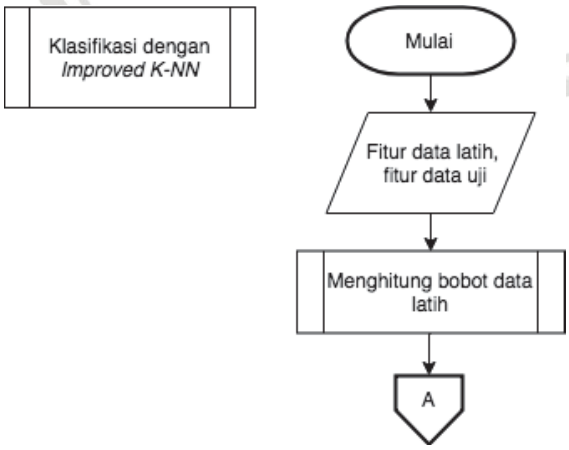
Gambar 4.5 Diagram alir ekstraksi fitur tekstur dengan LBP

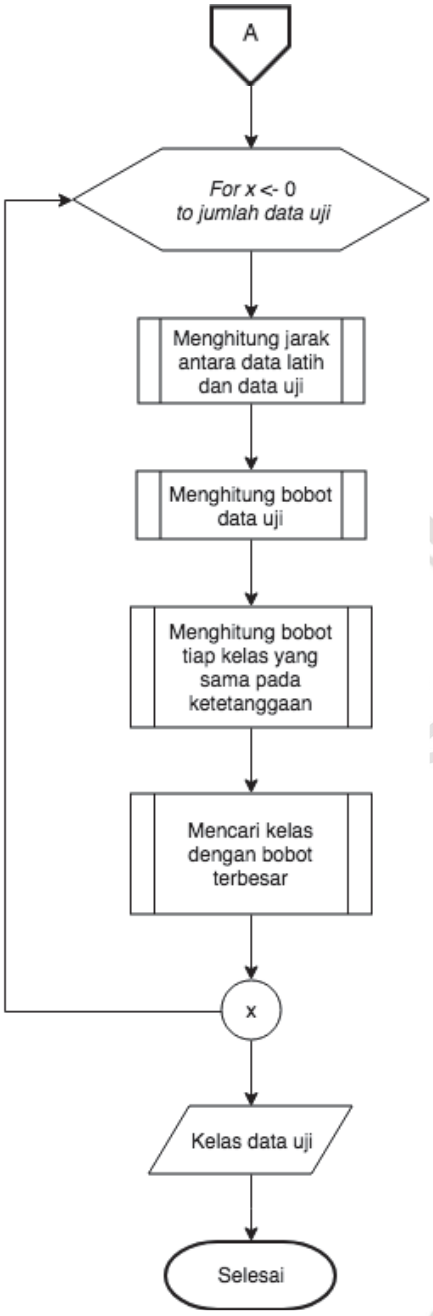
4.1.4 Klasifikasi *Improved K-NN*

Pada klasifikasi dengan *Improved K-NN* terdapat dua alur proses yang dilakukan yaitu proses pembobotan data latih dan proses klasifikasi data uji. Alur proses pembobotan data latih ditunjukkan pada Gambar 4.6 dan alur proses klasifikasi data uji ditunjukkan pada Gambar 4.7.



Gambar 4.6 Diagram alir menghitung bobot data latih





Gambar 4.7 Diagram alir klasifikasi dengan *Improved K-NN*

4.2 Perhitungan Manualisasi

Perhitungan manualisasi menjelaskan tentang perhitungan pada setiap tahapan yang dilakukan. Perhitungan yang dilakukan antara lain pada tahap *pre-processing,* konversi warna ke HSV, ekstraksi fitur HSV, konversi citra ke LBP dan klasifikasi. Sampel data yang digunakan ditunjukkan pada Gambar 4.8. Ukuran piksel yang dihitung pada perhitungan manualisasi adalah 3x3 piksel.

Gambar 4.8 Sampel data

4.2.1 Perhitungan *Pre-processing* Citra

Berikut adalah langkah-langkah yang dilakukan dalam porses *pre-processing* citra.

1. Mengubah ukuran citra menjadi 500x500 piksel.

   Citra 500x500 piksel ditunjukkan pada Gambar 4.9.

Gambar 4.9 Hasil perubahan ukuran citra

2. Mengonversi citra ke HSV.

   Citra masukan berupa citra RGB dengan nilai piksel yang ditunjukkan pada Tabel 4.1. Perubahan nilai piksel citra dari RGB ke HSV ditunjukkan pada Tabel 4.2.

Tabel 4.1 Nilai piksel citra RGB

R,G,B			
y / x	0	1	2
0	144, 146, 143	217, 166, 56	177, 186, 193
1	188, 197, 204	149, 80, 65	189, 200, 206
2	190, 199, 206	130, 99, 65	177, 188, 194

Tabel 4.2 Hasil konversi citra RGB ke HSV

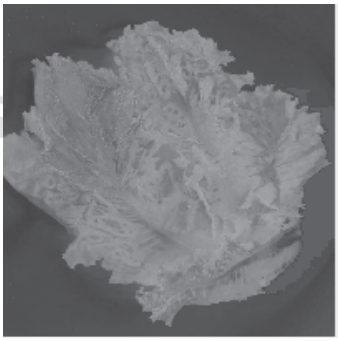
H, S, V			
y / x	0	1	2
0	50, 5, 146	20, 189, 217	103, 21, 193
1	103, 20, 204	14, 228, 149	101, 21, 206
2	103, 20, 206	16, 127, 130	101, 22, 194

3. Mengonversi citra ke *grayscale*.

Citra HSV yang telah didapatkan dari tahap sebelumnya kemudian dikonversi ke citra *grayscale*. Perubahan nilai piksel dari HSV ke *grayscale* ditunjukkan pada Tabel 4.3 dan hasil citra *grayscale* ditunjukkan pada Gambar 4.10.

Tabel 4.3 Hasil konversi citra HSV ke *grayscale*

y / x	0	1	2
0	52	178	82
1	84	180	85
2	85	115	82



Gambar 4.10 Hasil citra *grayscale*

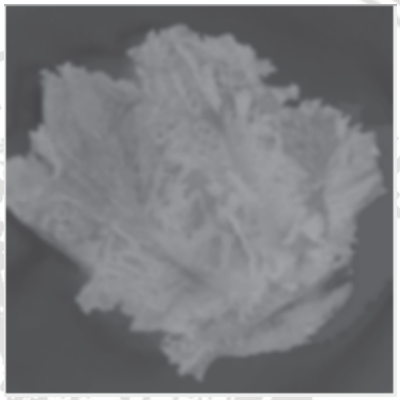
4. Memberikan filter Gaussian *Blur*

Citra *grayscale* yang telah didapatkan dari tahapan sebelumnya diberi *filter* Gaussian *Blur.* Perubahan nilai piksel setelah dilakukan *filtering* ditunjukkan pada Tabel 4.4.

Tabel 4.4 Hasil *filtering* Gaussian Blur

y / x	0	1	2
0	123	124	126
1	121	122	122
2	118	119	119

Citra hasil *filter* ditunjukkan pada Gambar 4.11.



Gambar 4.11 Hasil *filter* Gaussian *Blur*

5. Segmentasi dengan *Otsu*

Selanjutnya dilakukan segmentasi citra dengan *Otsu.* Segmentasi dilakukan untuk membedakan objek dengan latar. Perubahan nilai piksel setelah dilakukan segmentasi *Otsu* dan mengubah citra menjadi biner ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil segmentasi *Otsu*

y / x	0	1	2
0	255	255	255
1	0	255	255
2	0	0	0

Hasil segmentasi citra ditunjukkan pada Gambar 4.12.

Gambar 4.12 Hasil segmentasi citra

6. Melakukan erosi dan dilasi citra

Citra hasil segmentasi yang telah didapatkan dari tahapan sebelumnya dierosi dan dilasi. Perubahan nilai piksel setelah dilakukan erosi ditunjukkan pada Tabel 4.6 dan nilai piksel yang didapatkan setelah dilakukan dilasi ditunjukkan pada Tabel 4.7.

Tabel 4.6 Hasil erosi

y / x	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

Tabel 4.7 Hasil dilasi

y / x	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

Citra hasil erosi ditunjukkan pada Gambar 4.13 dan citra hasil dilasi ditunjukkan pada Gambar 4.14.

Gambar 4.13 Hasil erosi

Gambar 4.14 Hasil dilasi

7. Melakukan *masking.*

Citra hasil erosi dan dilasi yang telah didapatkan dari tahapan sebelumnya digabungkan dengan citra asli dengan *masking* yang menggunakan operasi *bitwise and.* Perubahan nilai piksel setelah dilakukan ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil *masking*

y / x	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

Citra hasil *masking* ditunjukkan pada Gambar 4.15.

Gambar 4.15 Hasil *masking*

4.2.2 Perhitungan HSV

Berikut adalah langkah-langkah yang dilakukan dalam proses ekstraksi fitur warna dengan HSV.

1. Membagi nilai *R,G,B* dengan 255.

Tabel 4.9 Nilai RGB

R,G,B			
y / x	0	1	2
0	144, 146, 143	217, 166, 56	177, 186, 193
1	188, 197, 204	149, 80, 65	189, 200, 206
2	190, 199, 206	130, 99, 65	177, 188, 194

Nilai *R, G* dan *B* yang digunakan adalah 144, 146, 143.

$$R = \frac{144}{255} = 0{,}564$$

$$G = \frac{146}{255} = 0{,}572$$

$$B = \frac{143}{255} = 0{,}560$$

2. Menentukan nilai maksimum dan minimum dari nilai *R, G, B*

$$Max = 0{,}572$$

$$Min = 0{,}560$$

3. Menghitung nilai *H.*

Nilai *Max = G.* Sehingga persamaan yang digunakan untuk menghitung *H* adalah Persamaan 2.8.

$$H = 60 * \left[2 + \frac{0{,}572 - 0{,}564}{0{,}572 - 0{,}560}\right]$$

$$H = 100/2$$

$$H = 50$$

4. Menghitung nilai *S.*

Untuk menghitung nilai *S* digunakan Persamaan 2.5.

$$S = \frac{0{,}572 - 0{,}560}{0{,}572}$$

$$= 0{,}0205$$

$$S = 0{,}0205 * 255$$

$$= 5{,}239$$

5. Menghitung nilai V.

Untuk menghitung nilai *S* digunakan Persamaan 2.4.

$$V = 0{,}572$$

$$V = 0{,}572 * 255$$

$= 146$

Sehingga didapatkan nilai *H, S, V* yaitu 50, 5,239, 146.

4.2.3 Perhitungan Ekstraksi Fitur HSV

Nilai HSV yang didapatkan dari perhitungan HSV diekstraksi untuk mendapatkan fitur. Fitur yang digunakan adalah *mean,* deviasi standar dan *skewness* dari masing-masing *channel* warna. Nilai HSV yang didapatkan dan perhitungan ekstraksi fitur ditunjukkan pada Tabel 4.10.

Tabel 4.10 Nilai HSV

H, S, V			
y / x	**0**	**1**	**2**
0	50, 5, 146	20, 189, 217	103, 21, 193
1	103, 20, 204	14, 228, 149	101, 21, 206
2	103, 20, 206	16, 127, 130	101, 22, 194

1. *Mean*

Untuk menghitung nilai *mean* digunakan Persamaan 2.10. Berikut adalah perhitungan nilai *mean* pada *channel* warna H.

$$\mu = \frac{50 + 20 + 103 + 103 + 14 + 101 + 103 + 16 + 101}{9}$$

$$\mu = 67{,}889$$

2. Deviasi standar

Untuk menghitung nilai *mean* digunakan Persamaan 2.11. Berikut adalah perhitungan nilai deviasi standar pada *channel* warna H.

$$\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij} - \mu)^2 = (50 - 67{,}89)^2 + (20 - 67{,}89)^2 + (103 - 67{,}89)^2 + (103 - 67{,}89)^2 + (14 - 67{,}89)^2 + (101 - 67{,}89)^2 + (103 - 67{,}89)^2 + (16 - 67{,}89)^2 + (101 - 67{,}89)^2$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij} - \mu)^2 = 14100{,}889$$

$$\sigma = \sqrt[2]{\frac{14100{,}89}{9}}$$

$$\sigma = 39{,}582$$

3. *Skewness*

Untuk menghitung nilai *mean* digunakan Persamaan 2.12. Berikut adalah perhitungan nilai *skewness* pada *channel* warna H.

$$\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij}-\mu)^3 = (50-67{,}89)^3+(20-67{,}89)^3+(103-67{,}89)^3+(103-67{,}89)^3$$
$$+(14-67{,}89)^3+(101-67{,}89)^3+(103-67{,}89)^3+(16-67{,}89)^3$$
$$+(101-67{,}89)^3$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N}(P_{ij}-\mu)^3 = -209296{,}69$$

$$\sigma = \sqrt[3]{\frac{-209296{,}69}{9}}$$

$$\sigma = -28{,}54346$$

4.2.4 Perhitungan Ekstraksi Fitur LBP

Berikut adalah langkah-langkah yang dilakukan dalam proses ekstraksi fitur tekstur dengan LBP.

1. Menentukan radius dan jumlah piksel ketetanggaan.

   Pada penelitian ini digunakan radius bernilai 1 dan jumlah piksel ketetanggaan 8.

2. Melakukan *windowing* 3x3 piksel pada citra masukan.

   Contoh *windowing* 3x3 piksel pada citra masukan ditunjukkan pada Tabel 4.11.

Tabel 4.11 *Windowing* 3x3 piksel

x \ y	0	1	2
0	145	169	184
1	195	93	197
2	197	104	185

3. Melakukan *thresholding* piksel yang terpilih dengan piksel tetangganya.

   Thresholding dilakukan dengan membandingkan piksel pusat dengan piksel tetangganya. Jika piksel pusat lebih kecil atau sama dengan piksel tetangga, maka nilai binernya adalah 1. Jika piksel pusat lebih besar daripada piksel tetangga, maka nilai binernya adalah 0. Perbandingan dilakukan bergantian pada setiap piksel tetangga dengan arah sesuai dengan arah jarum jam. Hasil *thresholding* ditunjukkan pada Tabel 4.12.

Tabel 4.12 Hasil *thresholding*

x \ y	0	1	2
0	1	1	1
1	1	0	1
2	1	1	1

4. Membaca hasil biner piksel berurutan sesuai hasil *thresholding.*

   Untuk mengubah angka biner menjadi angka desimal, pembacaan hasil biner dilakukan dengan urutan yang ditunjukkan pada Tabel 4.13.

Tabel 4.13 Urutan membaca piksel

x \ y	0	1	2
0	1	2	3
1	8		4
2	7	6	5

Sehingga nilai desimal dari hasil *thresholding* dihitung sebagai berikut.

Angka biner = 11111111

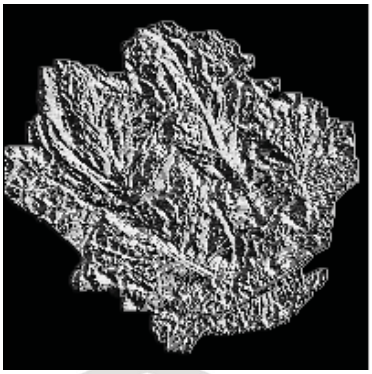
Angka desimal = 255

5. Menghitung nilai desimal dan meletakkannya pada matriks LBP

   Angka desimal yang sudah didapatkan dari langkah sebelumnya dimasukkan ke matriks LBP. Kemudian ulangi langkah 1 untuk *windowing* selanjutnya hingga semua piksel terhitung nilai desimalnya. Pergeseran *window* dilakukan hingga semua piksel dan piksel tepi diberikan nilai 0. Hasil yang didapatkan ditunjukkan pada Tabel 4.14. Hasil piksel LBP ditunjukkan pada Gambar 4.16.

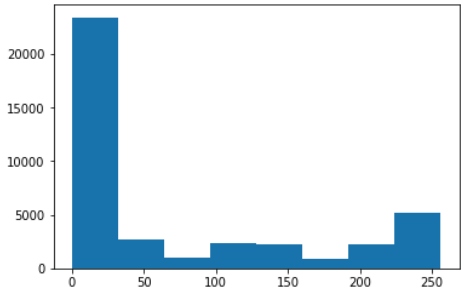
Tabel 4.14 Matriks LBP

x \ y	0	1	2	3	4
0	0	0	0	0	0
1	0	40	88	32	0
2	0	32	255	0	0
3	0	0	141	2	0
4	0	0	0	0	0



Gambar 4.16 Citra Hasil LBP

Selanjutnya tiap nilai piksel diproyeksikan ke histogram 8-*bins.* Gambar histogram hasil piksel LBP ditunjukkan pada Gambar 4.17.



Gambar 4.17 Histogram Piksel LBP

Pada Gambar 4.17 sumbu *x* menunjukkan nilai piksel dengan rentang dari nol sampai 256 yang dibagi menjadi delapan bagian atau *bins*. Sumbu *y* menunjukkan jumlah piksel dari citra yang memiliki nilai piksel sesuai dengan sumbu *x.* Nilai dari sumbu *y* pada masing-masing *bins* digunakan sebagai fitur dari LBP. Sehingga terdapat delapan fitur dari LBP.

4.2.5 Perhitungan *Improved K-NN*

Setelah fitur dari citra didapatkan, dilakukan klasifikasi untuk menentukan jenis citra dengan metode *Improved K-NN.* Perhitungan di bawah ini dilakukan dengan menggunakan 10 sampel data latih dan 1 data uji. Data latih terdiri dari 3 kelas yaitu timun, donat dan stroberi. Data uji ditunjukkan pada Gambar 4.18 dan fitur dari data latih dan uji ditunjukkan pada Tabel 4.15.

Gambar 4.18 Data uji

Tabel 4.15 Fitur data latih dan data uji

No	μ H	μ S	μ V	...	θ H	θ S	θ V	H1	...	H8	Kelas
1	4,8	48,3	52,9	...	8,5	68,8	81,4	27385	...	299	Donat
2	4,3	45,6	47,1	...	7,0	73,7	82,7	28668	...	204	Donat
3	4,1	51,6	42,7	...	7,8	95,5	82,5	30206	...	263	Donat
4	4,5	12,5	10,6	...	63,6	59,6	56,2	3661	...	234	Stroberi
5	17,8	18,2	26,5	...	96,9	61,5	89,4	34494	...	289	Stroberi
6	5,2	6,5	4,5	...	68,4	55,5	38,6	38404	...	98	Stroberi
7	8,0	13,8	5,9	...	75,0	77,7	35,7	37082	...	80	Stroberi
8	8,5	31,7	28,4	...	15,8	61,2	68,4	30213	...	288	Timun
9	8.8	35.5	36.9	...	15.2	62.8	77.7	29862	...	280	Timun
10	3.7	12.8	10.6	...	15.7	54.5	47.5	36081	...	98	Timun
11	8.5	58.1	97.0	...	22.5	47.2	72.6	21888	...	562	?

Berikut adalah langkah-langkah yang dilakukan untuk melakukan klasifikasi.

1. Menentukan jumlah ketetanggaan (S_{size}).

   Nilai S_{size} pada perhitungan ini adalah 3. Nilai S_{size} digunakan untuk menentukan jumlah tetangga terdekat saat pemilihan kelas.

2. Menghitung bobot data latih.

Bobot data latih dihitung dengan Persamaan 2.16.

$$I\,Weight(1) = \frac{1}{3}\sum_{i=1}^{3} f(class(1), class(S_i(1))$$

$$= \frac{1}{3}(2) = 0.67$$

Hasil perhitungan bobot data latih ditunjukkan pada Tabel 4.16.

Tabel 4.16 Bobot data latih

No	Kelas	*I Weight*
1	Donat	0,67
2	Donat	0,67
3	Donat	0,33
4	Stroberi	0,00
5	Stroberi	0,67
6	Stroberi	0,67
7	Stroberi	0,67
8	Timun	0,33
9	Timun	0,33
10	Timun	0,00

3. Menghitung jarak antara data uji dan data latih.

Jarak antara data uji dan data latih dihitung dengan persamaan Euclidean yang ditunjukkan pada Persamaan 2.18. Hasil perhitungan jarak antara data uji dan data latih ditunjukkan pada Tabel 4.17.

Tabel 4.17 Jarak antara data uji dan data latih

No	Kelas	*I Weight*	Jarak
1	Donat	0,67	1,433221475
2	Donat	0,67	1,768160618
3	Donat	0,33	2,167641791
4	Stroberi	0,00	4,74747163
5	Stroberi	0,67	3,283733464
6	Stroberi	0,67	4,302888101
7	Stroberi	0,67	3,958931095

Tabel 4.17 Jarak antara data uji dan data latih (Lanjutan)

No	Kelas	*I Weight*	Jarak
8	Timun	0,33	2,169256848
9	Timun	0,33	2,077943413
10	Timun	0,00	3,698189966

Sehingga didapatkan data latih sebanyak S_{size} yang memiliki jarak terkecil dari data uji yang ditunjukkan pada Tabel 4.18.

Tabel 4.18 Data latih dengan jarak terdekat

No	Kelas	*I Weight*	D
1	Donat	0,67	1,433221475
2	Donat	0,67	1,768160618
8	Timun	0,33	2,077943413

4. Menghitung bobot data ketetanggaan.

   Selanjutnya dilakukan perhitungan bobot data ketetanggaan dari bobot data latih dan jarak antara data uji dan data latih sesuai dengan Persamaan 2.19.

$$W(1) = 0{,}67 * \frac{1}{1{,}433+0{,}5}$$

$$= 0{,}344$$

   Hasil perhitungan bobot data uji ditunjukkan pada Tabel 4.19.

Tabel 4.19 Bobot data uji

No	Kelas	*I Weight*	D	W
1	Donat	0.67	1.433221475	0,344
2	Donat	0.67	1.768160618	0,293
8	Timun	0.33	2.077943413	0,128

5. Menghitung bobot tiap kelas yang sama pada ketetanggaan.

   Bobot tiap kelas dihitung dengan menjumlahkan bobot dari data yang memiliki kelas yang sama.

$$Donat = 0{,}344 + 0{,}293 = 0{,}638$$

$$Timun = 0{,}128$$

6. Mencari kelas dengan bobot terbesar.

Kelas dengan bobot terbesar yang didapatkan dari tahapan sebelumnya adalah kelas Donat. Sehingga data uji diklasifikasikan sebagai citra dengan jenis Donat.

4.2.6 Perhitungan Akurasi

Selanjutnya dilakukan perhitungan evaluasi kinerja. Evaluasi kinerja diukur dengan menghitung akurasi dari hasil klasifikasi. Pada perhitungan ini, data latih yang digunakan adalah data latih pada Tabel 4.15. Kelas hasil klasifikasi dan kelas sebenarnya pada lima data uji ditunjukkan pada Tabel 4.20.

Tabel 4.20 Hasil klasifikasi data uji

Data uji ke-	Hasil klasifikasi	Kelas sebenarnya
1	Donat	Donat
2	Donat	Donat
3	Stroberi	Stroberi
4	Timun	Timun
5	Timun	Stroberi

Dari hasil klasifikasi pada Tabel 4.20 dilakukan perhitungan nilai akurasi. Nilai akurasi dihitung dengan Persamaan 2.20 mendapatkan hasil sebagai berikut.

$$Akurasi = \frac{4}{5} x\ 100\% = 80\%$$

Sehingga nilai akurasi yang didapatkan dari klasifikasi adalah sebesar 80%.

4.3 Perancangan Skenario Pengujian

Pengujian pada penelitian ini bertujuan untuk mengetahui hasil akurasi yang didapatkan dari hasil klasifikasi. Pengujian yang dilakukan antara lain pengujian nilai *k,* pengujian fitur warna HSV, pengujian fitur tekstur LBP dan pengujian hasil akurasi metode klasifikasi *Improved K-NN* dan K-NN. Rancangan pengujian pada penelitian ini ditunjukkan pada Tabel 4.21.

Tabel 4.21 Perancangan pengujian

Nilai *k*	HSV	LBP	HSV dan LBP	HSV dan LBP dengan K-NN
1				
3				
5				
7				
9				

BAB 5
IMPLEMENTASI

Bab ini berisi penjelasan tentang implementasi yang dilakukan berdasarkan perancangan pada Bab Analisis dan Perancangan. Bab ini terdiri dari lingkungan implementasi dan implementasi algoritme. Lingkungan implementasi menjelaskan tentang lingkungan perangkat keras dan lunak yang digunakan untuk mengimplementasi sistem. Implementasi algoritme menjelaskan tentang kode sumber dari yang digunakan untuk mengimplementasi algoritme.

5.1 Lingkungan Implementasi

Terdapat perangkat-perangkat yang digunakan untuk implementasi. Perangkat tersebut antara lain perangkat keras dan perangkat lunak. Berikut adalah lingkungan dari perangkat keras dan lunak yang digunakan pada implementasi.

5.1.1 Lingkungan Perangkat Keras

Berikut adalah lingkungan perangkat keras yang digunakan dalam implementasi sistem klasifikasi jenis citra makanan tunggal.

1. *Processor:* Intel Core 2 Duo
2. *Memory:* 2 GB
3. *Graphics:* NVIDIA GeForce 9400M 256 MB
4. *Harddisk:* 160 GB
5. *Monitor* 14 inch
6. *Keyboard*

5.1.2 Lingkungan Perangkat Lunak

Berikut adalah lingkungan perangkat lunak yang digunakan dalam implementasi sistem klasifikasi jenis citra makanan tunggal.

1. *Operating System* OS X El Capitan.
2. Spyder sebagai *tools* pemrograman.
3. Microsoft Office Word 2011 untuk pembuatan laporan penelitian.
4. Microsoft Office Excel 2011 untuk pembuaatan perhitungan manual dari metode dan untuk menyimpan hasil dari perhitungan program.

5.2 Implementasi Algoritme

Pada implementasi algoritme terbagi menjadi empat bagian antara lain penjelasan tentang algoritme *pre-processing,* ekstraksi fitur warna, ekstraksi fitur

tekstur dan klasifikasi. Berikut adalah penjelasan dari algoritme yang diimplementasi.

5.2.1 Algoritme *Pre-processing*

Tahapan pertama yang dilakukan dalam pengolahan citra adalah *pre-processing.* Algoritme *pre-processing* pada penelitian ini digunakan untuk membedakan objek dengan latar dari citra masukan. Implementasi kode sumber algoritme *pre-processing* ditunjukkan pada Algoritme 1.

Algoritme 1: Fungsi Pre-processing

```
1   def preprocessing (image) :
2       img = cv2.resize(image, (500,500))
3       imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
4       imgGray = cv2.cvtColor(imgHSV, cv2.COLOR_BGR2GRAY)
5       height, width = imgGray.shape
6       blur = cv2.GaussianBlur(imgGray,(5,5),0)
7       ret3,th4                                                    =
8   cv2.threshold(blur,120,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
9       kernel = np.ones((5,5), np.uint8)
10      img_erosion = cv2.erode(th4, kernel, iterations=1)
11      img_dilation = cv2.dilate(img_erosion, kernel, iterations=1)
12      img2 = cv2.bitwise_and(img, img, mask = img_dilation)
13      return img2
```

Berikut adalah penjelasan dari kode sumber algoritme *pre-processing.*

1. Baris 1 adalah deklarasi dari fungsi *preprocessing* dengan parameter *image*.
2. Baris 2 mengubah ukuran citra dari parameter *image* menjadi 500x500 piksel dan disimpan pada variabel *img.*
3. Baris 3 mengubah ruang warna citra *img* dari RGB ke HSV dan disimpan pada variabel *imgHSV.*
4. Baris 4 mengubah ruang warna citra *imgHSV* dari HSV ke *grayscale* dan disimpan pada variabel *imgGray*.
5. Baris 5 mendapatkan tinggi dan lebar dari citra *imgGray* dan disimpan pada variabel *height* dan *width.*
6. Baris 6 melakukan proses *blurring* pada citra *imgGray* dan disimpan pada variabel *blur.*
7. Baris 7-8 melakukan proses *thresholding* dengan metode *Otsu.*
8. Baris 9 merupakan inisalisasi kernel 5x5 pada variabel *kernel.*
9. Baris 10 melakukan proses erosi pada citra hasil *thresholding* dan disimpan pada variabel *img_erosion.*
10. Baris 11 melakukan proses dilasi pada citra *img_erosion* dan disimpan pada variabel *img_dilation.*

11. Baris 12 melakukan proses *masking* pada citra *img_dilation* dan disimpan pada variabel *img2.*
12. Baris 13 merupakan nilai kembalian dari fungsi *preprocessing* yaitu variabel *img2.*

5.2.2 Algoritme Ekstraksi Fitur Warna

Untuk mendapatkan fitur warna dari citra masukan, dilakukan ekstraksi fitur warna. Ekstraksi fitur warna pada penelitian ini dilakukan dengan mengonversi ruang warna citra dari RGB ke HSV, selanjutnya dihitung *mean,* deviasi standar, dan *skewness* dari citra HSV. Berikut adalah penjelasan dari algoritme yang diimplementasi pada ekstraksi fitur warna.

5.2.2.1 Algoritme Konversi RGB ke HSV

Implementasi kode sumber algoritme konversi RGB ke HSV ditunjukkan pada Algoritme 2.

Algoritme 2: Fungsi HSV

```
1   def rgb2hsv(r, g, b):
2       r, g, b = r/255.0, g/255.0, b/255.0
3       mx = max(r, g, b)
4       mn = min(r, g, b)
5       df = mx-mn
6       if mx == mn:
7           h = 0
8       elif mx == r:
9           h = (60*(g-b)/df)
10      elif mx == g:
11          h = (120 + 60*(b-r)/df)
12      elif mx == b:
13          h = (240 + 60*(r-g)/df)
14      if mx == 0:
15          s = 0
16      else:
17          s = (df/mx)*255
18      v = mx*255
19      h=h/2
20      return h,s,v
```

Berikut adalah penjelasan dari algoritme fungsi HSV.

1. Baris 1 merupakan deklarasi dari fungsi *rgb2hsv* dengan parameter *r, g,* dan *b.*
2. Baris 2 merupakan inisialisasi dari variabel *r, g,* dan *b* yaitu sama dengan nilai dari masing-masing variabel dibagi dengan 255.
3. Baris 3 mencari nilai maksimum dari variabel *r, g,* dan *b*. Selanjutnya disimpan pada variabel *mx.*
4. Baris 4 mencari nilai minimum dari variabel *r, g,* dan *b*. Selanjutnya disimpan pada variabel *mn.*

5. Baris 5 merupakan inisialisasi variabel *df* yaitu sama dengan nilai variabel *mx* dikurang dengan nilai variabel *mn.*
6. Baris 6-13 merupakan seleksi nilai variabel *mx* untuk menentukan nilai variabel *h.*
7. Baris 14-17 merupakan seleksi nilai variabel *mx* untuk menentukan nilai variabel *s.*
8. Baris 18 merupakan inisialisasi variabel *v* yaitu sama dengan nilai variabel *mx* dikali dengan 255.
9. Baris 19 merupakan inisialisasi variabel *h* yaitu sama dengan nilai variabel tersebut dibagi dengan 2.
10. Baris 20 merupakan nilai kembalian dari fungsi *rgb2hsv* yaitu variabel *h, s* dan *v.*

5.2.2.2 Algoritme Perhitungan *Mean*

Implementasi kode sumber algoritme perhitungan *mean* ditunjukkan pada Algoritme 3.

Algoritme 3: Fungsi Perhitungan *Mean*

```
1   def mean(hsv):
2       mean_h, mean_s, mean_v = 0,0,0
3       h,s, v= hsv[:,:,0],hsv[:,:,1],hsv[:,:,2]
4       h2, s2, v2 = h.ravel(), s.ravel(), v.ravel()
5       mean_h = sum(h2)/len(h2)
6       mean_s = sum(s2)/len(s2)
7       mean_v = sum(v2)/len(v2)
8       return mean_h, mean_s, mean_v
```

Berikut adalah penjelasan dari algoritme fungsi perhitungan *mean.*

1. Baris 1 adalah deklarasi fungsi *mean* dengan parameter *hsv.*
2. Baris 2 merupakan inisialisasi variabel *mean_h, mean_s,* dan *mean_v* dengan 0.
3. Baris 3 merupakan inisialisasi variabel *h, s,* dan *v* dengan nilai dari parameter *hsv.*
4. Baris 4 mengubah array 2 dimensi *h, s,* dan *v* menjadi array 1 dimensi dan disimpan pada array *h2, s2*, dan *v2.*
5. Baris 5-7 menghitung nilai *mean* dari *h2, s2,* dan *v2* dan disimpan pada variabel *mean_h, mean_s,* dan *mean_v.*
6. Baris 8 merupakan nilai kembalian dari fungsi *mean* yaitu variabel *mean_h, mean_s,* dan *mean_v.*

5.2.2.3 Algoritme Perhitungan Deviasi Standar

Implementasi kode sumber algoritme perhitungan deviasi standar ditunjukkan pada Algoritme 4.

Algoritme 4: Fungsi Perhitungan Deviasi Standar

```
1   def std(hsv):
2       std_h, std_s, std_v = 0, 0, 0
3       h,s, v= hsv[:,:,0],hsv[:,:,1],hsv[:,:,2]
4       h2, s2, v2 = h.ravel(), s.ravel(), v.ravel()
5       mean_h = mean(hsv)[0]
6       mean_s = mean(hsv)[1]
7       mean_v = mean(hsv)[2]
8       h2 = (h2 - mean_h)**2
9       s2 = (s2 - mean_s)**2
10      v2 = (v2 - mean_v)**2
11      d = sum(h2)/len(h2)
12      e = sum(s2)/len(s2)
13      f = sum(v2)/len(v2)
14      std_h = np.sqrt(d)
15      std_s = np.sqrt(e)
16      std_v = np.sqrt(f)
17      return std_h, std_s, std_v
```

Berikut adalah penjelasan dari algoritme fungsi perhitungan deviasi standar.

1. Baris 1 adalah deklarasi fungsi *std* dengan parameter *hsv.*
2. Baris 2 merupakan inisialisasi variabel *std_h, std_s,* dan *std_v* dengan 0.
3. Baris 3 merupakan inisialisasi variabel *h, s,* dan *v* dengan nilai dari parameter *hsv.*
4. Baris 4 mengubah array 2 dimensi *h, s,* dan *v* menjadi array 1 dimensi dan disimpan pada array *h2, s2*, dan *v2.*
5. Baris 5-7 merupakan inisialisasi variabel *mean_h, mean_s,* dan *mean_v* dengan memanggil fungsi *mean.*
6. Baris 8-16 menghitung nilai deviasi standar dari *h2, s2,* dan *v2* dan disimpan pada variabel *std_h, std_s,* dan *std_v.*
7. Baris 17 merupakan nilai kembalian dari fungsi *std* yaitu variabel *std_h, std_s,* dan *std_v.*

5.2.2.4 Algoritme Perhitungan *Skewness*

Implementasi kode sumber algoritme perhitungan *skewness* ditunjukkan pada Algoritme 5.

Algoritme 5: Fungsi Perhitungan *Skewness*

```
1   def skewness(hsv):
2        skw_h, skw_s, skw_v = 0, 0, 0
3        h,s, v= hsv[:,:,0],hsv[:,:,1],hsv[:,:,2]
4        h2, s2, v2 = h.ravel(), s.ravel(), v.ravel()
5        mean_h = mean(hsv)[0]
6        mean_s = mean(hsv)[1]
```

```
7       mean_v = mean(hsv)[2]
8       h2 = (h2 - mean_h)**3
9       s2 = (s2 - mean_s)**3
10      v2 = (v2 - mean_v)**3
11      d = sum(h2)/len(h2)
12      e = sum(s2)/len(s2)
13      f = sum(v2)/len(v2)
14      skw_h = np.cbrt(d)
15      skw_s = np.cbrt(e)
16      skw_v = np.cbrt(f)
17      return skw_h, skw_s, skw_v
```

Berikut adalah penjelasan dari algoritme fungsi perhitungan *skewness.*

1. Baris 1 adalah deklarasi fungsi *skewness* dengan parameter *hsv.*
2. Baris 2 merupakan inisialisasi variabel *skw_h, skw_s,* dan *skw_v* dengan 0.
3. Baris 3 merupakan inisialisasi variabel *h, s,* dan *v* dengan nilai dari parameter *hsv.*
4. Baris 4 mengubah array 2 dimensi *h, s,* dan *v* menjadi array 1 dimensi dan disimpan pada array *h2, s2*, dan *v2.*
5. Baris 5-7 merupakan inisialisasi variabel *mean_h, mean_s,* dan *mean_v* dengan memanggil fungsi *mean.*
6. Baris 8-16 menghitung nilai *skewness* dari *h2, s2,* dan *v2* dan disimpan pada variabel *skw_h, skw_s,* dan *skw_v.*
7. Baris 17 merupakan nilai kembalian dari fungsi *std* yaitu variabel *skw_h, skw_s,* dan *skw_v.*

5.2.3 Algoritme Ekstraksi Fitur Tekstur

Untuk mendapatkan fitur tekstur dari citra masukan, dilakukan ekstraksi fitur tekstur. Ekstraksi fitur tekstur pada penelitian ini dilakukan dengan algoritme LBP. Implementasi kode sumber algoritme LBP ditunjukkan pada Algoritme 6.

```
Algoritme 6: Fungsi LBP
1   def lbp(img):
2       height = img.shape[0]
3       width = img.shape[1]
4       lbp = np.zeros((height,width), np.uint8)
5       gr = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6       LBPMASK = [(-1,-1),(-1,0),(-1,1),(0,1),(1,1),(1,0),(1,-1),
7       (0,-1)]
8       for x in range(1, height-1):
9           for y in range(1, width-1):
10              n = 0
11              gv = gr[x][y]
12              for i in range(len(LBPMASK)):
13                  m = LBPMASK[i]
14                  if gr[x+m[0]][y+m[1]]>=gv:
15                      n += 1 << i
```

```
16                  lbp[x][y] = n
17          hist = plt.hist(lbp.ravel(),8,[0,256]);plt.show()
18          return hist
```

Berikut adalah penjelasan dari algoritme fungsi LBP.

1. Baris 1 adalah deklarasi fungsi *lbp* dengan parameter *img.*
2. Baris 2-3 mendapatkan tinggi dan lebar dari citra *img* dan disimpan pada variabel *height* dan *width.*
3. Baris 4 deklarasi array *lbp* yang merupakan array *numpy.*
4. Baris 5 mengubah ruang warna citra *img* dari RGB ke *grayscale* dan disimpan pada variabel *gr*.
5. Baris 6-7 merupakan inisialisasi array *LBPMASK*.
6. Baris 8-9 melakukan perulangan tiap piksel pada citra.
7. Baris 10-11 merupakan inisialisasi variabel *n* dan variabel *gv* menyimpan nilai saat ini.
8. Baris 12-13 melakukan perulangan sebanyak array *LBPMASK* dan menyimpan nilai *LBPMASK* di variabel *m.*
9. Baris 14 untuk membandingkan nilai piksel saat ini dengan piksel tetangga. Jika nilai piksel tetangga lebih besar atau sama dengan nilai piksel saat ini maka dilakukan perhitungan nilai piksel LBP.
10. Baris 15 merupakan perhitungan nilai piksel LBP.
11. Baris 17 membuat histogram dari piksel LBP pada citra dan disimpan pada array *hist.*
12. Baris 18 merupakan nilai kembalian dari fungsi *lbp* yaitu array *hist*.

5.2.4 Algoritme Klasifikasi

Untuk mengklasifikasi fitur warna dan tekstur yang didapatkan dari citra, dilakukan klasifikasi dengan algoritme *Improved K-NN*. Tahapan-tahapan yang dilakukan untuk klasifikasi dengan *Improved K-NN* antara lain normalisasi, perhitungan bobot data latih dan klasifikasi. Berikut adalah penjelasan dari algoritme yang diimplementasi untuk klasifikasi.

5.2.4.1 Algoritme Normalisasi

Implementasi kode sumber algoritme normalisasi ditunjukkan pada Algoritme 7.

```
Algoritme 7: Fungsi Normalisasi
1   def normalisasi(datalatih, datauji):
2       min_lama_latih = min(datalatih.ravel())
3       max_lama_latih = max(datalatih.ravel())
4       min_lama_uji = min(datauji.ravel())
5       max_lama_uji = max(datauji.ravel())
```

```
6       min_lama = min_lama_latih
7       if(min_lama>min_lama_uji):
8           min_lama = min_lama_uji
9       max_lama = max_lama_latih
10      if(max_lama<max_lama_uji):
11          max_lama = max_lama_uji
12      min_baru=0
13      max_baru=1000
14      datauji = (((datauji-min_lama)/(max_lama-min_lama))*(max_baru-
15  min_baru))+0
16      datalatih = (((datalatih-min_lama)/(max_lama-min_lama))*
17      (max_baru-min_baru))+0
18      return datalatih, datauji
```

Berikut adalah penjelasan dari algoritme fungsi normalisasi.

1. Baris 1 merupakan deklarasi dari fungsi *normalisasi* dengan parameter *datalatih* dan *datauji.*
2. Baris 2-3 merupakan inisialisasi dari variabel *min_lama_latih* dan *max_lama_latih* dengan nilai minimum dan maksimum dari parameter *datalatih.*
3. Baris 4-5 merupakan inisialisasi dari variabel *min_lama_uji* dan *max_lama_uji* dengan nilai minimum dan maksimum dari parameter *datalatih.*
4. Baris 6-8 merupakan inisialisasi variabel *min_lama* dengan nilai terkecil di antara variabel *min_lama_latih* dan *min_lama_uji.*
5. Baris 9-11 merupakan inisialisasi variabel *max_lama* dengan nilai terbesar di antara variabel *max_lama_latih* dan *max_lama_uji.*
6. Baris 12-13 merupakan inisialisasi variabel *min_baru* dan *max_baru* yaitu 0 dan 100.
7. Baris 14-17 merupakan inisialisasi variabel *datalatih* dan *datauji* dengan perhitungan normalisasi dari variabel *datalatih* dan *datauji.*
8. Baris 18 merupakan nilai kembalian dari fungsi *normalisasi* yaitu variabel *datalatih* dan *datauji.*

5.2.4.2 Algoritme Perhitungan Bobot Data Latih

Implementasi kode sumber algoritme perhitungan bobot data latih ditunjukkan pada Algoritme 8.

Algoritme 8: Fungsi Perhitungan Bobot Data Latih

```
1   def bobot_datalatih(data, tetangga):
2       bobot=[]
3       for i in range (0,len(data)):
4           jarak=[]
5           for j in range (0,len(data)):
6               aminb = (data[i]-data[j])**2
7               jarak.append(np.sqrt(sum(aminb)))
```

```
8           urutan_tetangga=sorted(range(len(jarak)),
9           key=jarak.__getitem__)
10          sama=0
11          for z in range (1,tetangga+1):
12              if(labels[i]==labels[urutan_tetangga[z]]):
13                  sama+=1
14          nilai = sama/tetangga
15          bobot.append(nilai)
16      return bobot
```

Berikut adalah penjelasan dari algoritme fungsi perhitungan bobot data latih.

1. Baris 1 merupakan deklarasi dari fungsi *bobot_datalatih* dengan parameter *data* dan *tetangga.*
2. Baris 2-3 merupakan deklarasi dari array *bobot*.
3. Baris 4-9 menghitung jarak antar data latih. Selanjutnya jarak diurutkan dari yang terkecil untuk menentukan tetangga dari tiap data.
4. Baris 10-15 membandingkan label data dengan label data tetangga untuk menghitung bobot dari tiap data. Bobot dari tiap data yang didapatkan disimpan pada array *bobot.*
5. Baris 16 merupakan nilai kembalian dari fungsi *bobot_datalatih* yaitu array *bobot.*

5.2.4.3 Algoritme *Improved K-NN*

Implementasi kode sumber algoritme klasifikasi dengan *Improved K-NN* ditunjukkan pada Algoritme 9.

Algoritme 9: Fungsi Klasifikasi Improved K-NN

```
1   def improved_knn(data,test,tetangga):
2       bobot = bobot_datalatih(data)
3       jarak=[]
4       nilai=[]
5       label_tetangga=[]
6       bobot_label=[]
7       for i in range (0,len(data)):
8           aminb = (data[i]-test)**2
9           jarak.append(np.sqrt(sum(aminb)))
10      urutan_tetangga=sorted(range(len(jarak)),
11      key=jarak.__getitem__)
12      for z in range (0,tetangga):
13          label_tetangga.append(labels[urutan_tetangga[z]])
14          nilai.append(bobot[urutan_tetangga[z]]/
15          (jarak[urutan_tetangga[z]]+0.5))
16      label_tetangga=np.array(label_tetangga)
17      label_unique=np.unique(label_tetangga)
18      for a in range(0,len(label_unique)):
19          total=0
20          for b in range(0, len(label_tetangga)):
21              if(label_unique[a]==label_tetangga[b]):
22                  total += nilai[b]
23           bobot_label.append(total);
```

```
24          urutan_bobot=sorted(range(len(bobot_label)),
25          key=bobot_label.__getitem__, reverse=True)
26          hasil = label_unique[urutan_bobot[0]]
27          return hasil
```

Berikut adalah penjelasan dari algoritme fungsi klasifikasi *Improved K-NN*.

1. Baris 1 merupakan deklarasi dari fungsi *improved_knn* dengan parameter *data, test,* dan *tetangga.*
2. Baris 2-6 merupakan deklarasi dari variabel *bobot, jarak, nilai, label_tetangga,* dan *bobot_label*.
3. Baris 7-11 menghitung jarak antara data uji dan data latih. Selanjutnya jarak diurutkan dari yang terkecil untuk menentukan tetangga dari tiap data.
4. Baris 12-15 membandingkan label data dengan label data tetangga untuk menghitung bobot dari tiap data tetangga. Bobot dari tiap data tetangga yang didapatkan disimpan pada array *bobot.*
5. Baris 16-17 merupakan inisialisasi array *label_tetangga* dan *label_unique* dengan label tetangga dan tiap jenis dari label data.
6. Baris 18-26 menghitung bobot label dari data tetangga. Bobot label kemudian diurutkan. Label yang memiliki bobot tertinggi dijadikan sebagai kelas hasil klasifikasi dan disimpan pada variabel *hasil*.
7. Baris 27 merupakan nilai kembalian dari fungsi *improved_knn* yaitu variabel *hasil.*

BAB 6
PENGUJIAN DAN ANALISIS

Bab ini berisi penjelasan tentang hasil pengujian dan analisis dari algoritme yang telah diimplementasi. Pengujian dilakukan untuk mencari nilai akurasi dari hasil klasifikasi yang didapatkan dari metode yang telah diimplementasikan. Analisis dilakukan dengan membandingkan nilai akurasi yang telah didapatkan dari hasil pengujian yang telah dilakukan.

6.1 Skenario Pengujian dan Analisis

Skenario pengujian dilakukan sesuai dengan perancangan skenario pengujian pada Bab 4. Pengujian yang dilakukan antara lain pengujian nilai *k*, pengujian fitur HSV, pengujian fitur LBP, pengujian fitur HSV dan LBP dan pengujian metode klasifikasi *Improved K-NN* dan K-NN.

Skenario pertama adalah pengujian nilai *k*. Nilai *k* merupakan variabel yang menentukan jumlah ketetanggan yang digunakan untuk mengklasifikasikan data. Pada pengujian ini, hasil akurasi dari lima nilai *k* berbeda dibandingkan untuk dicari *k* yang terbaik. Skenario kedua adalah pengujian fitur HSV. Pengujian ini bertujuan untuk mengetahui nilai akurasi yang didapatkan ketika fitur yang digunakan untuk klasifikasi adalah fitur HSV. Skenario ketiga adalah pengujian fitur LBP. Pengujian ini bertujuan untuk mengetahui nilai akurasi yang didapatkan ketika fitur yang digunakan untuk klasifikasi adalah fitur LBP. Skenario keempat adalah pengujian fitur HSV dan LBP. Pengujian ini bertujuan untuk mengetahui nilai akurasi yang didapatkan ketika fitur yang digunakan untuk klasifikasi adalah fitur HSV dan LBP. Skenario kelima adalah pengujian metode klasifikasi. Pengujian ini bertujuan untuk mengetahui nilai akurasi yang didapatkan oleh metode klasifikasi *Improved K-NN* dan K-NN.

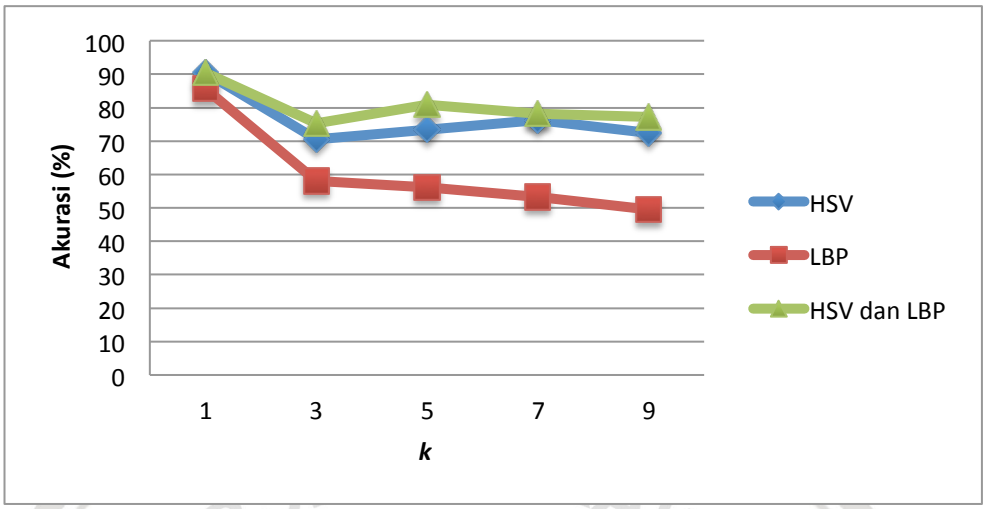
6.1.1 Pengujian Nilai *k*

Pengujian nilai *k* dilakukan dengan menggunakan fitur HSV, LBP, dan HSV dan LBP. Nilai *k* yang diujikan memiliki rentang dari satu hingga sepuluh sesuai dengan jumlah data latih pada tiap kelas. Nilai *k* yang digunakan antara lain *k=1, k=3, k=5, k=7,* dan *k=9.* Pemilihan nilai *k* ganjil dilakukan dengan tujuan agar didapatkan hanya satu kelas terbanyak dari ketetanggaan data uji untuk dijadikan kelas data uji. Hasil pengujian nilai *k* ditunjukkan pada Tabel 6.1.

Tabel 6.1 Hasil pengujian nilai *k*

k / Fitur	1	3	5	7	9
HSV	90,476%	70,476%	73,333%	76,190%	72,380%
LBP	85,714%	58,095%	56,190%	53,333%	49,532%
HSV dan LBP	90,476%	75,238%	80,952%	78,095%	77,142%

Hasil pengujian nilai *k* ditunjukkan pada grafik pada Gambar 6.1.



Gambar 6.1 Grafik hasil pengujian nilai *k*

Dari kelima nilai *k* yang digunakan hasil pengujian menunjukkan bahwa ketika *k* bernilai satu maka nilai akurasi yang dihasilkan optimal yaitu 90,476%. Selain itu ketika nilai *k* semakin besar, maka nilai akurasi akan semakin menurun.

Hasil pengujian terhadap nilai *k* menunjukkan bahwa nilai *k* yang memiliki hasil rata-rata akurasi tertinggi adalah *k*=1. Hal ini terjadi karena ketika *k* bernilai 1, data uji hanya akan mencari satu tetangga terdekat dari data latih. Sehingga bobot data latih tidak diperhitungkan dalam penentuan kelas data uji, karena hanya ada satu nilai bobot ketetanggaan. Kelas data latih terdekat akan dijadikan sebagai kelas data uji karena jumlah ketetanggaan bernilai satu.

Hasil pengujian juga menunjukkan bahwa nilai *k* yang semakin besar membuat hasil akurasi semakin menurun. Hal ini terjadi karena semakin besar nilai *k* maka jumlah ketetanggaan juga akan semakin besar. Jumlah ketetanggaan yang besar dapat mengakibatkan dua atau lebih kelas yang berbeda dianggap sebagai kelas terdekat dari satu data uji. Selanjutnya ketika bobot ketetanggaan dihitung hasilnya dapat menunjukkan bahwa kedua kelas tersebut memiliki bobot yang sama atau bobot kelas yang benar memiliki nilai yang lebih kecil. Hal ini menyebabkan terjadinya kesalahan klasifikasi sehingga nilai akurasi menurun. Contoh kesalahan pengklasifikasian karena nilai *k* ditunjukkan pada Tabel 6.2. Citra data uji yang digunakan ditunjukkan pada Gambar 6.2.

Gambar 6.2 Citra data uji

Tabel 6.2 Contoh kesalahan pengklasifikasian karena nilai *k*

k	Tetangga	Bobot/ Label data latih	Bobot Ketetanggaan (W)	Hasil Klasifikasi	Kelas Sebenarnya
1	 Dataset/Oreo/ 14(10).jpeg	0,2/Oreo	0,0532	Oreo	Oreo
3	 Dataset/Oreo/ 14(10).jpeg  Dataset/Gery Salut/17(2).jpeg  Dataset/Oreo/ 14(9).jpeg	0,2/Oreo 0/Gery Salut 0,2/Oreo	0,0532 0,0 0,0229	Oreo	Oreo
5	 Dataset/Oreo/ 14(10).jpeg  Dataset/Gery Salut/17(2).jpeg	0,2/Oreo 0/Gery Salut	0,0532 0,0	Milo Nugget	Oreo

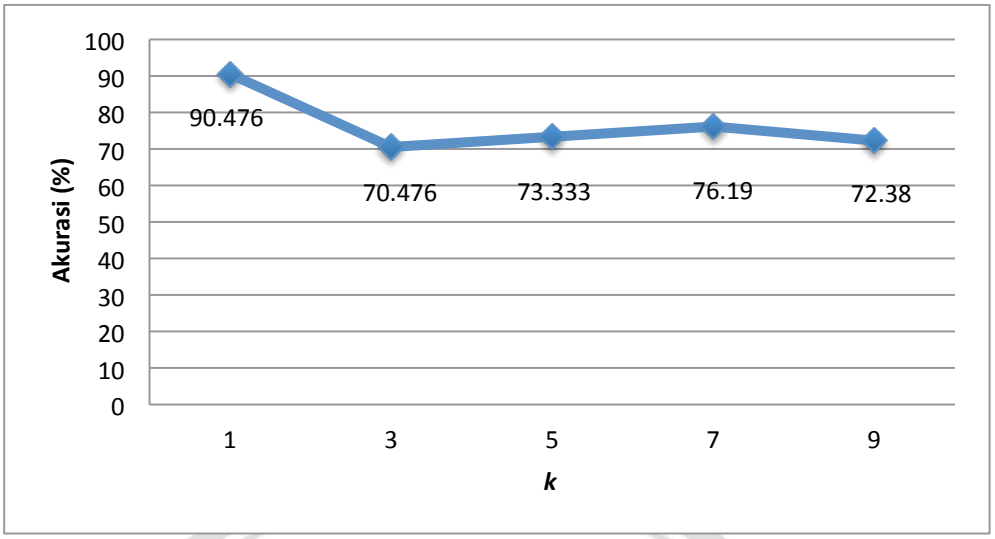
Tabel 6.2 Contoh kesalahan pengkalsifikasian karena nilai *k* (Lanjutan)

k	Tetangga	Bobot/ Label data latih	Bobot Ketetanggaan (W)	Hasil Klasifikasi	Kelas Sebenarnya
	 Dataset/Oreo/ 14(9).jpeg	0,2/Oreo	0,0229		
	 Dataset/Milo Nugget/19(6)	0,6/Milo Nugget	0,04		
	 Dataset/Milo Nugget/19(1)	0,6/Milo Nugget	0,399		

Dari Tabel 6.2 didapatkan bobot ketetanggaan kelas Oreo adalah sebesar 0,0761, kelas Gery Salut adalah 0, dan kelas Milo Nugget adalah 0,0799. Data uji diklasifikasikan sebagai Milo Nugget karena bobot ketetanggan terbesar adalah bobot ketetanggaan kelas Milo Nugget. Kelas Milo Nugget mendapatkan bobot ketetanggaan terbesar karena bobot data latih pada kelas Milo Nugget lebih besar daripada bobot data latih pada kelas Oreo. Bobot data latih pada masing-masing data latih kelas Milo Nugget yang berdekatan dengan data uji adalah 0,6. Sedangkan bobot data latih pada masing-masing data latih kelas Oreo yang berdekatan dengan data uji adalah 0,2. Bobot data latih semakin besar nilainya jika kemiripan antar data pada suatu kelas tinggi dan semakin kecil nilainya jika kemiripan antar data pada suatu kelas rendah.

6.1.2 Pengujian Fitur HSV

Pada pengujian ini dilakukan klasifikasi dengan *Improved K*-NN dan fitur yang digunakan hanya hasil ekstraksi fitur warna HSV dari citra. Pengujian fitur HSV ditunjukkan pada grafik pada Gambar 6.3.

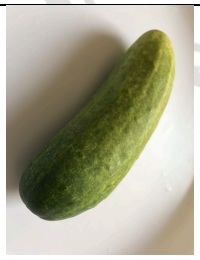
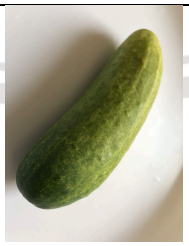


Gambar 6.3 Grafik hasil pengujian fitur HSV

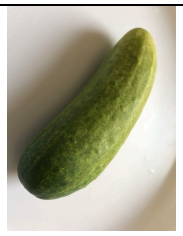
Hasil pengujian fitur HSV menunjukkan bahwa fitur HSV dapat mencirikan objek untuk diklasifikasi. Hasil akurasi dari klasifikasi dengan fitur HSV tertinggi adalah sebesar 90,476% ketika *k*=1. Nilai rata-rata akurasi yang dihasilkan dari fitur HSV adalah 76,571%.

Hasil pengujian menunjukkan bahwa ketika nilai *k* semakin besar, maka nilai akurasi akan semakin menurun. Hal ini terjadi karena jika nilai *k* semakin besar maka jumlah tetangga dari satu data uji juga akan bertambah. Pada pengujian ini, fitur yang digunakan hanya fitur warna dengan ruang warna HSV. Sehingga klasifikasi hanya berdasarkan dari fitur warna saja. Hal ini menyebabkan jika suatu jenis makanan memiliki warna yang mirip dengan jenis makanan lain, maka jenis makanan tersebut akan saling bertetangga dan memiliki jarak antar data yang kecil. Sehingga dapat menyebabkan kesalahan klasifikasi. Contoh dari kesalahan pengklasifikasian fitur HSV ditunjukkan pada Tabel 6.3.

Tabel 6.3 Contoh kesalahan pengklasifikasian fitur HSV

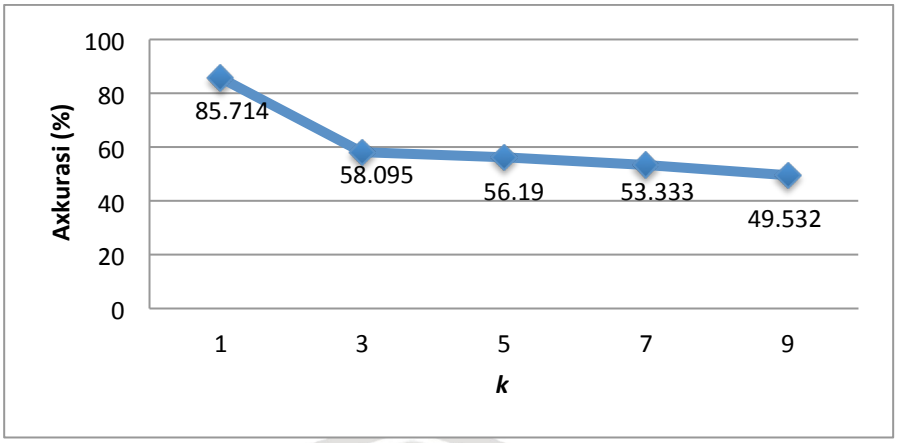
Data ke-	Data Uji	*k*	Tetangga	Hasil Klasifikasi	Kelas Sebenarnya
32	 DataUji/7(2).jpeg	1	 Dataset/Timun/ 7(2).jpeg/	Timun	Timun

Tabel 6.3 Contoh kesalahan pengklasifikasian fitur HSV (Lanjutan)

Data ke-	Data Uji	*k*	Tetangga	Hasil Klasifikasi	Kelas Sebenarnya
		3	 Dataset/Timun/ 7(2).jpeg  Dataset/Pisang Hijau/11(8).jpeg  Dataset/Pisang Hijau/11(9).jpeg	Pisang Hijau	Timun

6.1.3 Pengujian Fitur LBP

Pada pengujian ini dilakukan klasifikasi dengan *Improved K*-NN dan fitur yang digunakan hanya hasil ekstraksi fitur tekstur dengan metode LBP dari citra. Fitur LBP didapatkan dari pemetaan piksel LBP pada histogram yang dibagi menjadi delapan rentang. Sehingga fitur yang didapatkan dari ekstraksi fitur tekstur dengan LBP sebanyak delapan fitur. Pengujian fitur LBP ditunjukkan pada grafik pada Gambar 6.4.

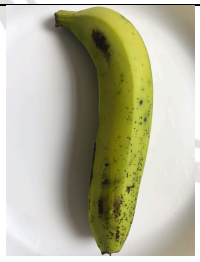
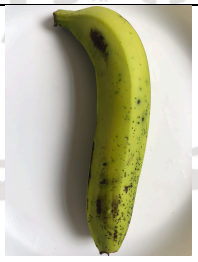


Gambar 6.4 Grafik hasil pengujian fitur LBP

Hasil pengujian fitur LBP menunjukkan bahwa fitur LBP dapat mencirikan objek untuk diklasifikasi. Hasil akurasi dari klasifikasi dengan fitur LBP tertinggi adalah sebesar 85,714% ketika *k*=1. Nilai rata-rata akurasi yang dihasilkan dari fitur LBP adalah 60,572%.

Hasil pengujian menunjukkan bahwa ketika nilai *k* semakin besar, maka nilai akurasi akan semakin menurun. Hal ini terjadi karena jika nilai *k* semakin besar maka jumlah tetangga dari satu data uji juga akan bertambah. Pada pengujian ini, fitur yang digunakan hanya fitur tekstur dengan metode LBP. Sehingga klasifikasi hanya berdasarkan dari fitur tekstur saja. Hal ini menyebabkan jika suatu jenis makanan memiliki tekstur yang mirip dengan jenis makanan lain, maka jenis makanan tersebut akan saling bertetangga dan memiliki jarak antar data yang kecil. Sehingga dapat menyebabkan kesalahan klasifikasi. Contoh dari kesalahan pengklasifikasian fitur LBP ditunjukkan pada Tabel 6.4.

Tabel 6.4 Contoh kesalahan pengklasifikasian fitur LBP

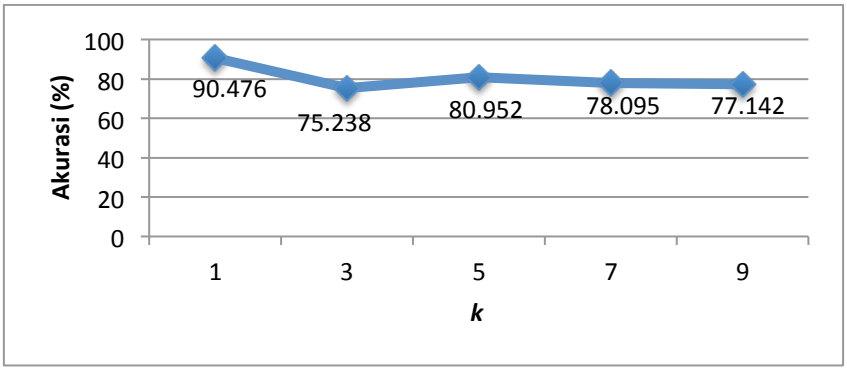
Data ke-	Data Uji	*k*	Tetangga	Hasil Klasifikasi	Kelas Sebenarnya
52	 DataUji/11(2).jpeg	1	 Dataset/Pisang Hijau/11(2).jpeg	Pisang Hijau	Pisang Hijau

Tabel 6.4 Contoh kesalahan pengklasifikasian fitur LBP (Lanjutan)

Data ke-	Data Uji	*k*	Tetangga	Hasil Klasifikasi	Kelas Sebenarnya
		3	 Dataset/Pisang Hijau/11(2).jpeg  Dataset/Pisang Kuning/12(6).jpeg  Dataset/Pisang Kuning/12(5).jpeg	Pisang Kuning	Pisang Hijau

6.1.4 Pengujian dengan Fitur HSV dan LBP

Pada pengujian ini dilakukan klasifikasi dengan *Improved K*-NN dan fitur yang digunakan adalah hasil ekstraksi fitur warna dengan ruang warna HSV dan ekstraksi fitur tekstur dengan metode LBP dari citra. Pengujian fitur HSV dan LBP ditunjukkan pada grafik pada Gambar 6.5.



Gambar 6.5 Grafik hasil pengujian fitur HSV dan LBP

Hasil pengujian fitur HSV dan LBP menunjukkan bahwa kombinasi fitur HSV dan LBP dapat mencirikan objek untuk diklasifikasi. Hasil akurasi dari klasifikasi dengan kombinasi fitur HSV dan LBP tertinggi adalah sebesar 90,476% ketika *k*=1. Nilai rata-rata akurasi yang dihasilkan dari kombinasi fitur HSV dan LBP adalah 80,380%.

Pada pengujian ini, fitur yang digunakan adalah fitur warna dan tekstur. Kesalahan klasifikasi dapat terjadi jika suatu jenis makanan memiliki fitur warna dan tekstur yang mirip sehingga sulit untuk dibedakan. Contoh jenis makanan yang memiliki fitur warna dan tekstur yang mirip antara lain roti tawar dan roti gandum dan jeruk oranye dan donat. Contoh kesalahan pengklasifikasian fitur HSV dan LBP ditunjukkan pada Tabel 6.5.

Tabel 6.5 Contoh kesalahan pengklasifikasian fitur HSV dan LBP

Data ke-	Data Uji	*k*	Tetangga/Bobot Ketetanggaan (W)	Hasil Klasifikasi	Kelas Sebenarnya
7	 DataUji/2(2).jpeg	1	 Dataset/Roti Gandum/2(5).jpeg	Roti Gandum	Roti Gandum
		3	 Dataset/Roti Gandum/2(5).jpeg	Roti Tawar	Roti Gandum

Tabel 6.5 Contoh kesalahan pengklasifikasian fitur HSV dan LBP (Lanjutan)

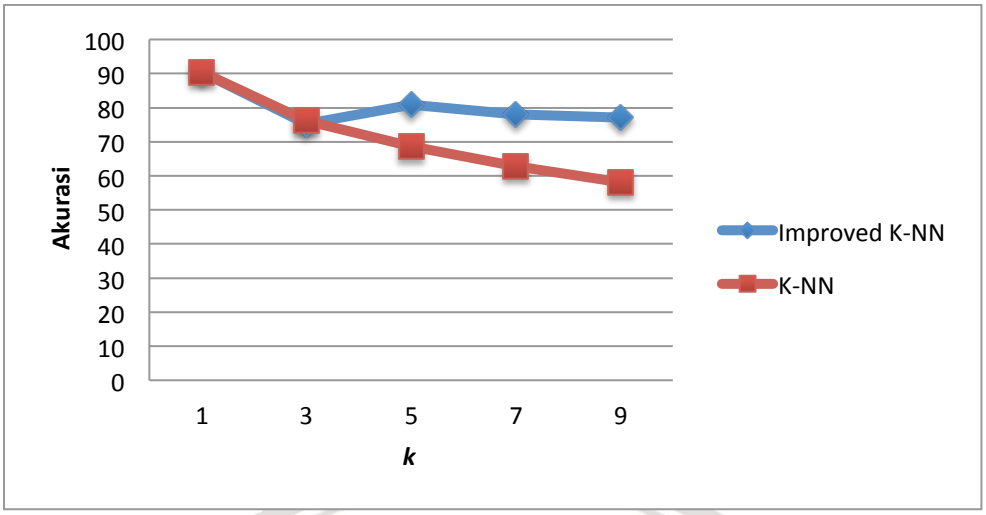
Data ke-	Data Uji	*k*	Tetangga	Hasil Klasifikasi	Kelas Sebenarnya
			 Dataset/Roti Tawar/3(4).jpeg  Dataset/Roti Tawar/3(10).jpeg		

6.1.5 Pengujian Terhadap Metode Klasifikasi

Pengujian terhadap metode klasifikasi dilakukan dengan membandingkan hasil akurasi yang didapatkan dari klasifikasi dengan metode *Improved K*-NN dan K-NN. Fitur yang digunakan pada pengujian ini adalah kombinasi ekstraksi fitur warna dengan ruang warna HSV dan ekstraksi fitur tekstur dengan metode LBP. Hasil pengujian metode klasifikasi ditunjukkan pada Tabel 6.6 dan grafik hasil pengujian ditunjukkan pada Gambar 6.6.

Tabel 6.6 Hasil pengujian metode klasifikasi

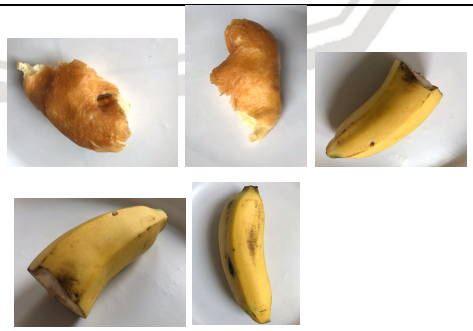
k / Metode	1	3	5	7	9	Rata-rata
Improved K-NN	90,476%	75,238%	80,952%	78,095%	77,142%	80,306%
K-NN	90,476%	76,190%	68,571%	62,857%	58,095%	71,237%



Gambar 6.6 Hasil pengujian metode klasifikasi

Hasil pengujian menunjukkan bahwa klasifikasi dengan metode *Improved K-NN* menghasilkan tingkat rata-rata akurasi yang lebih besar daripada klasifikasi dengan metode K-NN yaitu sebesar 80,306%. Hal ini disebabkan oleh adanya pembobotan data latih pada metode klasifikasi *Improved K-NN.* Pembobotan data latih memberikan bobot pada tiap data latih. Bobot data latih bernilai tinggi jika data memiliki kemiripan yang tinggi dengan data-data lain di kelasnya. Bobot data latih bernilai rendah jika data memiliki kemiripan yang rendah dengan data-data lain di kelasnya. Penentuan kelas pada klasifikasi dengan *Improved K-NN* juga dipengaruhi oleh bobot data latih saat dilakukan perhitungan bobot ketetanggaan. Ketika dilakukan perhitungan bobot ketetanggaan, tetangga yang memiliki bobot kelas yang kecil tidak akan dipilih sebagai kelas untuk data uji. Contoh perbandingan hasil klasifikasi dari kedua metode ditunjukkan pada Tabel 6.7 dan Tabel 6.8. Contoh data uji dan data tetanggaannya ditunjukkan pada Tabel 6.7. Contoh perbandingan hasil klasifikasi *Improved K-NN* dengan K-NN ditunjukkan pada Tabel 6.8.

Tabel 6.7 Contoh data uji dan tetangganya

Data ke-	Data Uji	*k*	Tetangga
4		5	

Tabel 6.8 Perbandingan hasil klasifikasi *Improved K-NN* dan K-NN

Metode	Cara mengklasifikasi	Hasil klasifikasi	Kelas Sebenarnya
Improved K-NN	Mencari bobot ketetanggaan (W) terbesar. W_{Donat} = 0,386 $W_{PisangKuning}$ = 0,303	Donat	Donat
K-NN	Mencari kelas terbanyak pada data tetangga. Donat = 2 Pisang Kuning = 3	Pisang Kuning	Donat

Dari hasil pengujian di atas, maka dapat dilihat bahwa pengklasifikasian dengan metode *Improved K-NN* memiliki tingkat akurasi yang lebih tinggi daripada pengklasifikasian dengan metode K-NN. Hal ini disebabkan oleh adanya perhitungan bobot data latih pada metode *Improved K-NN.*

BAB 7
PENUTUP

Bab ini berisi penjelasan tentang kesimpulan dan saran yang dapat diambil dari penelitian. Kesimpulan menjelaskan tentang hasil penelitian yang telah disimpulkan. Saran menjelaskan tentang saran yang dapat digunakan untuk penelitian selanjutnya.

7.1 Kesimpulan

Dari penelitian yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut.

1. Ekstraksi fitur HSV dan LBP dapat digunakan untuk mengklasifikasikan citra makanan tunggal dengan hasil akurasi terbaik sebesar 90,476%. Akurasi terbesar didapatkan dengan menggunakan kombinasi fitur HSV dan LBP dan fitur HSV saja, metode klasifikasi yang digunakan *Improved K-NN* dan nilai *k*=1.
2. Fitur HSV, LBP dan kombinasi HSV dan LBP dapat digunakan untuk mengklasifikasikan citra makanan tunggal. Klasifikasi dengan fitur HSV saja menghasilkan nilai akurasi tertinggi sebesar 90,476% dan rata-rata akurasi sebesar 76,571%. Klasifikasi dengan fitur LBP saja menghasilkan nilai akurasi tertinggi sebesar 85,714% dan rata-rata akurasi sebesar 60,572%. Klasifikasi dengan kombinasi fitur HSV dan LBP menghasilkan nilai akurasi tertinggi sebesar 90,476% dan rata-rata akurasi sebesar 80,380%. Sehingga kombinasi fitur HSV dan LBP menghasilkan nilai akurasi terbaik untuk mengklasifikasi citra makanan tunggal.

7.2 Saran

Dari hasil penelitian klasifikasi jenis citra makanan tunggal menggunakan metode HSV dan LBP masih terdapat kekurangan sehingga perlu dilakukan banyak pengembangan. Saran untuk penelitian selanjutnya adalah sebagai berikut.

1. Perlu dilakukan modifikasi nilai radius dan *point* atau jumlah ketetanggan pada metode ekstraksi fitur LBP agar hasil akurasi yang dihasilkan dari fitur tekstur lebih tinggi.
2. Perlu dilakukan penambahan data latih pada tiap jenis variasi citra agar bobot pada data latih yang didapatkan lebih tinggi.

DAFTAR PUSTAKA

Amaliyah, N. (2017). *Penyehatan Makanan dan Minuman - A.* Yogyakarta: Deepublish.

Amynarto, N., Sari, Y. A., & Wihandika, R. C. (2018). Pengenalan Emosi Berdasarkan Ekspresi Mikro Menggunakan Metode Local Binary Patterns. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* , 3230-3238.

Andono, P. N., Sutojo, T., & Muljono. (2017). *Pengolahan Citra Digital.* Yogyakarta: Penerbit ANDI.

Arivazhagan, S., Shebiah, R. N., Nidhyanandhan, S. S., & Ganesan, L. (2010). Fruit Recognition using Color and Texture Features. *Journal of Emerging Trends in Computing and Information Sciences* , 90-94.

Attokaren et al., 2017. Food Classification Using Convolutional Neural Networks.

Cusano, C., Napoletano, P., & Schettini, R. (2014). Combining Local Binary Patterns and Local Color Contrast for Texture Classification Under Varying Illumination. *Journal Optical Society of America* , 1453-1461.

Farinella, G. M., Allegra, D., Moltisanti, M., Stanco, F., & Battiato, S. (2016). Retrieval and Classification of Food Images. *Computers in Biology and Medicine* , 23-39.

Neneng, & Fernando, Y. (2017). Klasifikasi Jenis Daging Berdasarkan Analisis Citra Tekstur Gray Level Co-Occurence Matrices (GLCM) dan Warna. *Seminar Nasional Sains dan Teknologi* , 1-7.

Ong, P. M., & Punzalan, E. (2014). Comparative Analysis of RGB and HSV Color Models in Extracting Color Features of Green Dye Solutions. *DLSU Research Congress 2014* , 1-7.

OpenCV, 2018. *Color conversions.* [online] Tersedia di:< https://docs.opencv.org/3.3.0/de/d25/imgproc_color_conversions.html> [Diakses 1 Oktober 2018]

Patro, S. K., & Sahu, K. K. (2015). Normalization: A Preprocessing Stage. 1-4.

PM, N., & Chezian, M. D. (2013). Various Colour Spaces and Colour Space Conversion Algorithm. *Journal of Global Research in Computer Science* , 44-48.

Reddy, R. K., Reddy, B., & Reddy, E. (2014). An Effective GLCM and Binary Pattern Schemes Based Classification for Rotation Invariant Fabric Textures. *International Journal of Computer Engineering Science (IJCES)* , 1-16.

Sari, Y. A., Dewi, R. K., & Fatichah, C. (2014). Seleksi Fitur Menggunakan Ekstraksi Fitur Bentuk, Warna, dan Tekstur dalam Sistem Temu Kembali Citra Daun. *JUTI* , 1-8.

Singh, S., & Hemachandran, K. (2012). Content Based Image Retrieval using Color Moment and Gabor Texture Feature. *International Journal of Computer Science Issues Vol.9* , 299-309.

Soman, S., Ghorpade, M., & Sonone, V. (2012). Content Based Image Retrieval Using Advanced Color and Texture Features. *International Conference in Computational Intelligence (ICCIA)* , 1-5.

Turiyanto, M. D., Purwanto, D., & Dikairono, R. (2014). Penerapan Teknik Pengenalan Wajah Berbasis Fitur Local Binary Patterns pada Robot Pengantar Makanan. 1-6.

Usuman, I., Dharmawan, A., & Frisky, A. Z. (2012). Sistem Pendeteksi Kulit Manusia Menggunakan Segmentasi Warna Kulit Pada Tipe Citra HSV (Hue Saturation Value). *IJEIS Vol.2 No.2* , 143-154.

Wagle, S., Mangai, A., & Kumar, S. V. (2013). An Improved Medical Image Classification Model using Data Mining Techniques. *IEEE GCC Conference and Exhibition* , 114-118.