

# **KLASIFIKASI JENIS MAKANAN MENGGUNAKAN NEIGHBOR WEIGHTED K-NEAREST NEIGHBOR DENGAN SELEKSI FITUR INFORMATION GAIN**

## **SKRIPSI**

Untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun oleh:  
Vriza Wahyu Saputra  
NIM: 155150200111246



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

## PENGESAHAN

KLASIFIKASI JENIS MAKANAN DARI CITRA SMARTPHONE MENGGUNAKAN  
NEIGHBOR WEIGHTED K-NEAREST NEIGHBOR DENGAN SELEKSI FITUR  
INFORMATION GAIN  
SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun Oleh :  
Vriza Wahyu Saputra  
NIM: 155150200111246

Skripsi ini telah diuji dan dinyatakan lulus pada  
26 April 2019  
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Yuita Arum Sari, S.Kom, M.Kom  
NIK: 201609 880715 2 001

Dosen Pembimbing II



Agus Wahyu Widodo, S.T, M.Cs  
NIP: 19740805 200112 1 001

Mengetahui

Ketua Jurusan Teknik Informatika



Iri 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, 26 April 2019



Vriza Wahyu Saputra

NIM: 155150200111246

## PRA KATA

Dengan menyebut nama Allah SWT Yang Maha Pengasih Lagi Maha Penyayang. Puji syukur atas khadirat Allah SWT berkat pertolongan, kasih sayang dan kuasa-Nya penulis dapat menyelesaikan skripsi yang berjudul “Klasifikasi Jenis Makanan dari Citra *Smartphone* menggunakan *Neighbor Weighted K-Nearest Neighbor* dengan Seleksi Fitur *Information Gain*” dengan baik. Penulis menyadari menyadari skripsi ini tidak dapat selesai tanpa bantuan dan dukungan dari beberapa pihak. Oleh karena itu pada kesempatan ini, penulis menyampaikan rasa hormat dan terimakasih yang sebesar-besarnya kepada:

1. Ibu Yuita Arum Sari, S.Kom., M.Kom dan Bapak Agus Wahyu Widodo , S.T, M.Cs selaku dosen pembimbing I dan dosen pembimbing II yang telah dengan sabar banyak memberikan bimbingan, ilmu dan masukan dalam penyusunan skripsi ini.
2. Keluarga dari penulis terutama orang tua, kakak dan adik penulis yang tidak pernah lelah untuk memberikan do’a, motivasi, kasih sayang serta dukungan materiil.
3. Sahabat dan teman-teman penulis terkhususnya buat Muhammad Abdan Mulia, Yohana Yunita Putri, Febristya Anugrah F., Putri Indhira Utami dan masih banyak lagi yang telah setia menemani mengerjakan laporan skripsi.
4. Keluarga Eksekutif Mahasiswa Informatika dan terkhusus buat Badan Pengurus Harian serta Advokesma yang telah mengisi hari-hari penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
5. Seluruh civitas akademika Universitas Brawijaya terutama mas Hermawan Dwi Putra yang telah banyak memberi bantuan, ilmu dan dukungan selama penulis menempuh pendidikan di Teknik Informatika Universitas Brawijaya hingga proses penyelesaian skripsi ini.
6. PT. Artajasa Pembayaran Elektronis yang memberikan bantuan biaya pendidikan kepada penulis, kesempatan magang dan juga memberikan pelatihan mengenai *soft skill* dan *practical skill* yang nantinya akan penulis hadapi dalam dunia kerja.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Oleh karena itu penulis menerima saran dan kritik yang bersifat membangun demi perbaikan kearah kesempurnaan.

Malang, 26 April 2019

Penulis

Vrizawahyu22@gmail.com

## ABSTRAK

**Vriza Wahyu Saputra, Klasifikasi Jenis Makanan dari Citra *Smartphone* menggunakan *Neighbor Weighted K-Nearest Neighbor* dengan Seleksi Fitur *Information Gain***

**Pembimbing: Yuita Arum Sari, S.Kom., M.Kom. dan Agus Wahyu Widodo, S.T, M.Cs**

*Smartphone* dengan kemampuan sensor kamera yang kuat dapat digunakan untuk menganalisis foto dan pengenalan objek. Makanan adalah salah satu objek fotografi yang populer dan dapat menjadikan sesuatu yang menimbulkan rasa ingin memasak dan mencicipinya. Untuk memasak dibutuhkan resep masakan sebagai alat bantu untuk membuat masakan karena tidak semua orang tahu bagaimana cara membuat masakan. Tetapi untuk mencari resep makanan seseorang harus tau nama makanan yang akan dimasak. Untuk itu dibutuhkan teknik pencarian resep makanan dengan masukan citra makanan untuk mempermudah pencarian. Terdapat beberapa langkah metode yang dilakukan untuk melakukan pengenalan jenis makanan yaitu *preprocessing*, ekstraksi fitur, seleksi fitur dan klasifikasi. Ekstraksi fitur digunakan untuk memperoleh ciri yang terdapat pada citra makanan. Metode ekstraksi fitur yang digunakan adalah Color Moments untuk fitur warna dan *Gray Level Cooccurrence Matrix (GLCM)* untuk fitur tekstur. Seleksi fitur digunakan untuk mengurangi atribut yang tidak relevan. Metode seleksi fitur yang digunakan adalah *Information Gain*. Klasifikasi digunakan untuk melakukan proses pengenalan citra makanan yang sebelumnya tidak diketahui jenisnya. Metode klasifikasi yang digunakan adalah *Neighbor Weighted K-Nearest Neighbor (NWKNN)*. Total data citra makanan yang digunakan adalah sebanyak 23 jenis makanan, 529 data latih tidak seimbang dan 23 data uji. Pengujian dilakukan untuk mengetahui akurasi dari metode *NWKNN* dan juga mengetahui pengaruh seleksi fitur *Information Gain*. Hasil pengujian dengan metode *K-Fold Cross Validation* diperoleh akurasi rata-rata tertinggi sebesar 72,53% dengan pembagian data uji sebanyak 30, jumlah fitur sebanyak 10, nilai *K* pada *NWKNN* sebanyak 3 dan perhitungan jarak menggunakan *Cosine Similarity*. Selain itu pada pengujian pengaruh *Information Gain* menghasilkan akurasi tertinggi yaitu 86,96% dengan jumlah fitur terbaik sebanyak 15 fitur. Hal ini dapat disimpulkan bahwa metode *NWKNN* dapat menjawab permasalahan data tidak seimbang dan *Information Gain* dapat mengetahui fitur terbaik untuk klasifikasi.

Kata Kunci: ekstraksi fitur, seleksi fitur, klasifikasi, *Color Moments*, *GLCM*, *Information Gain*, *NWKNN*.

## ABSTRACT

**Vriza Wahyu Saputra, Food Type Classification from Smartphone Image using Neighbor Weighted K-Nearest Neighbor with Information Gain Feature Selection**

**Supervisors: Yuita Arum Sari, S.Kom., M.Kom. and Agus Wahyu Widodo, S.T, M.Cs**

*Smartphones with powerful camera sensor capabilities can be used to analyze photos and object recognition. Food is one of the popular photography objects and seeing it makes you want to cook or taste it. Cooking requires recipes as a tool to make dishes because not everyone knows how to make dishes. But to find a food recipe one must know the name of the food to be cooked. For this reason, it takes a food recipe search technique by inputting food images to facilitate search. There are several steps in the method that are carried out to introduce food types namely preprocessing, feature extraction, feature selection and classification. Feature extraction is used to obtain the characteristics found in food images. The feature extraction method used is Color Moments for color features and Gray Level Counseling Matrix (GLCM) for texture features. Feature selection is used to reduce irrelevant attributes. The feature selection method used is Information Gain. Classification is used to process the image recognition of foods of unknown type. The classification method used is Neighbor Weighted K-Nearest Neighbor (NWKNN). The total food image data used is as many as 23 types of food, 529 training data is not balanced and 23 test data. Tests were carried out to determine the accuracy of the NWKNN method and also to know the effect of the Information Gain feature selection. The test results with the K-Fold Cross Validation method obtained the highest average accuracy of 72,53% by dividing the test data by 30, the number of features by 10, the K value on NWKNN by 3 and calculating distances using Cosine Similarity. On other hands, the testing of the Information Gain effect resulted in the highest accuracy of 86.96% with the 15 best features. It can be concluded that the NWKNN method can solve the problem of unbalanced data and Information Gain can find out the best features for classification.*

**Keyword:** feature extraction, feature selection, classification, *Color Moments*, *GLCM*, *Information Gain*, *NWKNN*.

## DAFTAR ISI

KLASIFIKASI JENIS MAKANAN MENGGUNAKAN NEIGHBOR WEIGHTED K-NEAREST NEIGHBOR DENGAN SELEKSI FITUR INFORMATION GAIN.....	i
PENGESAHAN .....	<b>Error! Bookmark not defined.</b>
PERNYATAAN ORISINALITAS .....	<b>Error! Bookmark not defined.</b>
PRA KATA.....	iv
ABSTRAK.....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR .....	xii
BAB 1 PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah .....	3
1.3 Tujuan .....	3
1.4 Manfaat.....	3
1.5 Batasan masalah .....	3
1.6 Sistematika pembahasan .....	3
BAB 2 LANDASAN KEPUSTAKAAN .....	5
2.1 Tinjauan Pustaka .....	5
2.2 Ekstraksi Fitur.....	7
2.2.1 Transformasi Warna RGB ke HSV .....	8
2.2.2 Color Moments .....	10
2.2.3 Gray Level Cooccurrence Matrix (GLCM) .....	11
2.3 Seleksi Fitur .....	16
2.3.1 Information Gain .....	17
2.4 Klasifikasi.....	18
2.4.1 K-Nearest Neighbor (K-NN) .....	18
2.4.2 Neighbor Weighted K-Nearest Neighbor (NWKNN) .....	19
2.4.3 Manhattan Distance .....	20
2.4.4 Euclidean Distance .....	20

2.4.5 Cosine Similarity .....	21
2.5 Pengujian Algoritme .....	21
2.5.1 Pengujian Akurasi.....	21
2.5.2 Pengujian <i>k-Fold Cross Validation</i> .....	21
BAB 3 METODOLOGI .....	23
3.1 Tipe Penelitian .....	23
3.2 Strategi Penelitian.....	23
3.3 Peralatan Pendukung.....	24
3.4 Lokasi Penelitian .....	25
3.5 Perancangan Algoritme .....	25
3.6 Pengujian dan Analisis Algoritme .....	25
3.7 Teknik Pengumpulan Data .....	26
3.8 Data Penelitian.....	27
BAB 4 PERANCANGAN.....	31
4.1 Perancangan Algoritme .....	31
4.1.1 Perancangan Algoritme <i>Color Moments</i> .....	32
4.1.2 Perancangan Algoritme <i>Gray Level Cooccurrence Matrix</i> .....	36
4.1.3 Perancangan Algoritme <i>Information Gain</i> .....	54
4.1.4 Perancangan Algoritme <i>K-Nearest Neighbor</i> .....	55
4.1.5 Perancangan Algoritme <i>Neighbor Weighted K-Nearest Neighbor</i> .....	56
4.1.6 Perancangan Algoritme Perhitungan Jarak <i>Cosine Similarity</i> .....	57
4.2 Perhitungan Manualisasi .....	58
4.2.1 Perhitungan <i>Color Moments</i> .....	58
4.2.2 Perhitungan <i>Gray Level Co-Occurrence Matrix</i> .....	63
4.2.3 Perhitungan <i>Information Gain</i> .....	92
4.2.4 Perhitungan KNN.....	94
4.2.5 Perhitungan NWKNN .....	98
4.3 Perancangan Pengujian .....	100
4.3.1 Pengaruh Jumlah Fitur pada Seleksi Fitur <i>Information Gain</i> .....	100
4.3.2 Pengaruh Nilai K NKWNN terhadap Akurasi .....	101
4.3.3 Perbandingan Perhitungan Jarak NWKNN .....	101



4.3.4 Perbandingan Akurasi Metode KNN dan NWKNN .....	102
4.3.5 Pengujian <i>k-Fold Cross Validation</i> .....	102
BAB 5 IMPLEMENTASI .....	104
5.1 Batasan Implementasi .....	104
5.2 Implementasi Algoritme .....	104
5.2.1 Implementasi Color Moments .....	104
5.2.2 Implementasi GLCM .....	107
5.2.3 Implementasi Information Gain .....	116
5.2.4 Implementasi Klasifikasi KNN dan NWKNN .....	121
5.2.5 Implementasi Pengujian .....	126
BAB 6 PENGUJIAN DAN ANALISIS .....	129
6.1 Pengujian dan Analisis Pengaruh Jumlah Fitur pada Information Gain .....	129
6.2 Pengujian dan Analisis Pengaruh Nilai K NKWNN .....	132
6.3 Pengujian dan Analisis Perhitungan Jarak NWKNN .....	134
.....	135
6.4 Pengujian dan Analisis Perbandingan Akurasi Metode KNN dan NWKNN .....	136
6.5 Pengujian <i>K-Fold Cross Validation</i> .....	137
BAB 7 KESIMPULAN .....	139
7.1 Kesimpulan .....	139
7.2 Saran .....	140
DAFTAR PUSTAKA .....	141

## DAFTAR TABEL

Tabel 2.1 Tinjauan Pustaka .....	5
Tabel 3.1 Tabel dataset citra makanan .....	27
Tabel 4.1 Nilai warna R citra .....	58
Tabel 4.2 Nilai warna G citra .....	59
Tabel 4.3 Nilai warna B citra .....	59
Tabel 4.4 Nilai warna H citra .....	59
Tabel 4.5 Nilai warna S citra .....	60
Tabel 4.6 Nilai warna V citra .....	60
Tabel 4.7 Nilai <i>Grayscale</i> Citra .....	63
Tabel 4.8 Pembentukan Matriks Co-Occurrence piksel (1,1) sudut $0^{\circ}$ dan $d=1$ ...	63
Tabel 4.9 Hasil Matriks Co-Occurrence piksel (1,1) sudut $45^{\circ}$ dan $d=1$ .....	64
Tabel 4.10 Pembentukan Matriks <i>Co-Occurrence</i> piksel (1,1) sudut $45^{\circ}$ dan $d=1$	64
Tabel 4.11 Hasil Matriks <i>Co-Occurrence</i> piksel (1,1) sudut $90^{\circ}$ dan $d=1$ .....	65
Tabel 4.12 Pembentukan Matriks <i>Co-Occurrence</i> piksel (1,1) sudut $90^{\circ}$ dan $d=1$	65
Tabel 4.13 Hasil Matriks <i>Co-Occurrence</i> piksel (1,1) sudut $90^{\circ}$ dan $d=1$ .....	65
Tabel 4.14 Pembentukan Matriks Co-Occurrence piksel (1,1) sudut $135^{\circ}$ dan $d=1$ .....	66
Tabel 4.15 Hasil Matriks <i>Co-Occurrence</i> piksel (1,1) sudut $135^{\circ}$ dan $d=1$ .....	66
Tabel 4.16 Nilai $Px + y$ Sudut $0^{\circ}$ .....	73
Tabel 4.17 Nilai $Px - y$ Sudut $0^{\circ}$ .....	74
Tabel 4.18 Matriks Co-occurrence Simetris Baris ke 1 Sudut $0^{\circ}$ .....	74
Tabel 4.19 Nilai $Px$ sudut $0^{\circ}$ .....	75
Tabel 4.20 Nilai $Py$ sudut $0^{\circ}$ .....	75
Tabel 4.21 Nilai $Px + y$ Sudut $45^{\circ}$ .....	77
Tabel 4.22 Nilai $Px - y$ Sudut $45^{\circ}$ .....	78
Tabel 4.23 Matriks Co-occurrence Simetris Baris ke 1 Sudut $0^{\circ}$ .....	79
Tabel 4.24 Nilai $Px$ sudut $0^{\circ}$ .....	79
Tabel 4.25 Nilai $Py$ sudut $0^{\circ}$ .....	79
Tabel 4.26 Nilai $Px + y$ Sudut $90^{\circ}$ .....	82
Tabel 4.27 Nilai $Px - y$ Sudut $90^{\circ}$ .....	82

Tabel 4.28 Matriks Co-occurrence Simetris Baris ke 1 Sudut $0^{\circ}$ .....	83
Tabel 4.29 Nilai $Px$ sudut $0^{\circ}$ .....	83
Tabel 4.30 Nilai $Py$ sudut $0^{\circ}$ .....	83
Tabel 4.31 Nilai $Px + y$ Sudut $135^{\circ}$ .....	86
Tabel 4.32 Nilai $Px - y$ Sudut $135^{\circ}$ .....	87
Tabel 4.33 Matriks Co-occurrence Simetris Baris ke 1 Sudut $135^{\circ}$ .....	87
Tabel 4.34 Nilai $Px$ sudut $0^{\circ}$ .....	88
Tabel 4.35 Nilai $Py$ sudut $0^{\circ}$ .....	88
Tabel 4.36 Data Latih .....	89
Tabel 4.37 Data Uji .....	91
Tabel 4.38 Tabel Data Kelompok .....	92
Tabel 4.39 Pengurutan Hasil <i>Information Gain</i> .....	94
Tabel 4.40 Hasil <i>Cosine Similarity</i> .....	95
Tabel 4.41 Hasil <i>Cosine Similarity</i> terurut .....	96
Tabel 4.42 Kelompok Nilai <i>Cosim</i> dengan $K = 7$ .....	97
Tabel 4.43 Nilai <i>Cosim</i> terurut dengan $K = 7$ .....	98
Tabel 4.44 Hasil Perhitungan Skor .....	99
Tabel 4.45 Rancangan Pengujian Pengaruh Jumlah Fitur <i>Information Gain</i> .....	100
Tabel 4.46 Rancangan Pengujian Pengaruh Nilai $K$ NWKNN .....	101
Tabel 4.47 Rancangan Pengujian Perbandingan Perhitungan Jarak .....	101
Tabel 4.48 Rancangan Pengujian Perbandingan Akurasi KNN dan NWKNN .....	102
Tabel 4.49 Pengujian K-Fold Cross Validation .....	103
Tabel 6.1 Hasil Pengujian Pengaruh Jumlah Fitur pada <i>Information Gain</i> .....	129
Tabel 6.2 Hasil Pengujian Pengaruh Nilai $K$ NWKNN .....	133
Tabel 6.3 Hasil Pengujian Perhitungan Jarak NWKNN .....	134
Tabel 6.4 Tabel Pengujian Data Uji Ke-2 <i>Cosim</i> dan Euclidean .....	135
Tabel 6.5 Hasil Pengujian Perbandingan Akurasi Metode KNN dan NWKNN ....	136
Tabel 6.6 Hasil Pengujian <i>K-Fold Cross Validation</i> .....	137

## DAFTAR GAMBAR

Gambar 2.1 Penentuan Awal Nilai Matrix Coocurrence.....	13
Gambar 2.2 Hasil pembentukan nilai GLCM simetris .....	13
Gambar 2.3 Hasil GLCM matriks setelah dinormalisasi .....	14
Gambar 3.1 Diagram alir proses .....	24
Gambar 3.2 Citra dalam kondisi utuh pada 3 kemiringan .....	26
Gambar 3.3 Citra dalam kondisi dimakan $\frac{1}{4}$ pada 3 kemiringan .....	26
Gambar 3.4 Citra dalam kondisi dimakan $\frac{1}{2}$ pada 3 kemiringan .....	26
Gambar 3.5 Citra dalam kondisi dimakan $\frac{3}{4}$ pada 3 kemiringan .....	27
Gambar 4.1 Diagram Alir Sistem .....	32
Gambar 4.2 Diagram Alir Color Moments .....	33
Gambar 4.3 Diagram Alir Perhitungan <i>Mean</i> .....	34
Gambar 4.4 Diagram Alir Perhitungan <i>Standar Deviasi</i> .....	35
Gambar 4.5 Diagram Alir Perhitungan <i>Skewness</i> .....	36
Gambar 4.6 Diagram Alir GLCM .....	37
Gambar 4.7 Diagram Alir Matrix Co-occurrence.....	39
Gambar 4.8 Diagram Alir Ekstraksi Fitur ASM.....	40
Gambar 4.9 Diagram Alir Ekstraksi Fitur <i>Contrast</i> .....	41
Gambar 4.10 Diagram Alir Ekstraksi Fitur <i>Variance</i> .....	42
Gambar 4.11 Diagram Alir Ekstraksi Fitur <i>Inverse Difference Moment (IDM)</i> .....	43
Gambar 4.12 Diagram Alir Ekstraksi Fitur <i>Correlation</i> .....	45
Gambar 4.13 Diagram Alir Ekstraksi Fitur <i>Sum Average</i> .....	46
Gambar 4.14 Diagram Alir Ekstraksi Fitur <i>Sum Entropy</i> .....	47
Gambar 4.15 Diagram Alir Ekstraksi Fitur <i>Sum Variance</i> .....	48
Gambar 4.16 Diagram Alir Ekstraksi Fitur <i>Entropy</i> .....	49
Gambar 4.17 Diagram Alir Ekstraksi Fitur <i>Difference Entropy</i> .....	50
Gambar 4.18 Diagram Alir Ekstraksi Fitur <i>Difference Variance</i> .....	51
Gambar 4.19 Diagram Alir Ekstraksi Fitur Information Measure Of Correlation .	53
Gambar 4.20 Diagram Alir <i>Information Gain</i> .....	54
Gambar 4.21 Digram Alir K-NN .....	55
Gambar 4.22 Diagram Alir NWKNN .....	56

Gambar 4.23 Diagram Alir Cosine Similarity .....	57
Gambar 4.24 Sampel Citra Rendang 200 x 200 .....	58
Gambar 4.25 Sampel Citra Rendang 10 x 10 .....	58
Gambar 6.1 Grafik Standar Deviasi 15 Fitur.....	130
Gambar 6.2 Grafik Standar Deviasi 5 Fitur.....	130
Gambar 6.3 Grafik Kemiripan Fitur Tekstur .....	131
Gambar 6.4 Citra Makanan dengan Tekstur Mirip .....	131
Gambar 6.5 Standar Deviasi 25 Fitur .....	132
Gambar 6.6 Standar Deviasi 50 Fitur .....	132
Gambar 6.7 Visualisasi Hasil Data Uji ke 3 NWKNN.....	133
Gambar 6.8 Grafik Hasil Pengujian Perhitungan Jarak <i>NWKNN</i> .....	135
Gambar 6.9 Grafik Hasil Pengujian Perbandingan Akurasi Metode KNN dan NWKNN .....	136
Gambar 6.10 Grafik Hasil Pengujian <i>K-Fold Cross Validation</i> .....	138

## BAB 1 PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang awal terkait topik atau judul beserta metode yang digunakan. Selain latar belakang juga akan disampaikan terkait tujuan serta manfaat yang diharapkan oleh penulis.

### 1.1 Latar belakang

*Smartphone* digunakan oleh miliaran orang di seluruh dunia dan dilengkapi dengan berbagai jenis sensor dan prosesor yang kuat. Salah satunya adalah kemampuan mengambil gambar yang beresolusi tinggi. Peningkatan kamera dan peningkatan daya komputasi ini memungkinkan *smartphone* untuk menangkap dan menganalisis foto atau video untuk berbagai aplikasi, seperti pengenalan gerakan dan perilaku, pengenalan objek, diagnosis penyakit dan pemantauan kondisi kesehatan (Nejati *et al.*, 2016). Makanan adalah salah satu objek yang populer untuk media fotografi dan menjadikan sesuatu yang menimbulkan selera sehingga menggugah rasa untuk memasak dan mencicipinya (Arsita *et al.*, 2017). Tetapi tidak setiap orang tahu bagaimana cara membuat setiap makanan, sehingga dibutuhkan resep masakan sebagai alat bantu untuk membuat masakan. Keterbatasan pengetahuan mengenai nama makanan juga memberikan permasalahan untuk menemukan resep makanan yang dicari (Rahayuni, Sari dan Adinugroho, 2019). Untuk mempermudah pencarian resep makanan berdasarkan citra makanan maka dibutuhkan teknik pencarian dengan masukan berupa citra makanan. Pengguna akan memasukkan *query* berupa citra yang selanjutnya akan di *retrieve*. Teknik tersebut dikenal dengan teknik *Content Based Image Retrieval (CBIR)* (Layona, Tunardi dan Tanoto, 2014). Terdapat beberapa langkah untuk proses pembelajaran yaitu *preprocessing*, ekstraksi fitur meliputi warna, tekstur dan klasifikasi gambar dengan metode seperti *Bayesian*, *Support Vector Machine* dan sebagainya (Sasano, Han & Chen, 2017).

Manusia melihat warna karena adanya cahaya yang dipantulkan oleh objek (Kadir dan Susanto, 2013). Ruang warna *Red, Green dan Blue (RGB)* dinyatakan oleh tiga warna primer dalam fisik, sehingga makna fisiknya jelas. Ruang warna *Hue, Saturation dan Value (HSV)* merupakan sistem warna non linier yang konsisten dengan karakteristik persepsi manusia tentang warna dengan setiap elemen ruang warna terpisah sehingga cocok untuk pemrosesan gambar (Liu, Liu dan Chen, 2014). Ekstraksi fitur warna dengan ruang warna *RGB* dan *HSV* untuk mendeteksi *Rare Colored Capsul (RCC)* menghasilkan akurasi sempurna yaitu 100%. Tingkat deteksi kesalahan dari bukan *RCC* adalah 0,0067% (Liu, Liu and Chen, 2014). Ekstraksi fitur warna dengan metode *color moments* merupakan metode yang efektif untuk menganalisis citra karena memiliki dimensi fitur vektor terendah dan memiliki kompleksitas komputasi lebih rendah dibandingkan dengan ekstraksi fitur dengan metode *color correlogram*, *color structure descriptor* dan *color histogram* (Patil *et al.*, 2011). Untuk mengidentifikasi makanan tidak hanya dilihat dari warnanya saja, tetapi pada setiap jenis makanan mempunyai tekstur yang berbeda. Salah satu metode

dalam ekstraksi fitur tekstur yaitu *Gray Level Co-occurrence Matrix (GLCM)*, dimana metode ini melakukan analisis terhadap piksel citra untuk mengetahui jejak tingkat keabuan yang sering muncul (Xie *et al.*, 2010). Kompleksitas tekstur citra sulit untuk didefinisikan dan dikuantifikasi, namun GLCM bisa dipakai untuk mengkuantifikasi dan membandingkan berbagai aspek tekstur citra (Honeycutt dan Plotnick, 2008). Teknik ekstraksi fitur *GLCM* juga menghasilkan akurasi yang paling tinggi yaitu sebesar 95,2% pada citra tekstur *UMD* dibandingkan dengan ekstraksi yang lain seperti *LBP*, *ILBP*, *Gabor* dan *Granulometry* (Siqueira, Schwartz and Pedrini, 2012). Penelitian penyakit pada daun tebu dengan ekstraksi fitur *GLCM* dan *Color Moments* menghasilkan akurasi 97% (Dewi dan Ginardi, 2014). Pada penelitian tersebut menyimpulkan penghapusan fitur dapat memperbaiki akurasi karena ekstraksi fitur bentuk kurang merepresentasikan pola penyakit pada daun tebu.

Untuk mengurangi dimensi atribut yang kurang relevan maka perlu teknik seleksi fitur untuk menghapus fitur tersebut. Seleksi fitur *Information Gain* merupakan metode seleksi fitur yang sederhana dengan melakukan pemeringkatan atribut dimana metode ini banyak digunakan pada kategori teks, *microarray* dan citra (Chormunge dan Jena, 2016). Penggunaan seleksi fitur *Information Gain* dapat menghasilkan akurasi yang lebih baik daripada tanpa menggunakan *Information Gain*. Penelitian pada klasifikasi penyakit jantung menunjukkan bahwa tanpa menggunakan *Information Gain* pada sebaran kelas seimbang dan tidak seimbang yaitu 61,54% dan 73,08% sedangkan dengan menggunakan *Information Gain* menunjukkan hasil pada sebaran kelas seimbang dan tidak seimbang yaitu 84,62% dan 88,46% (Aini, Sari dan Arwan, 2018). Selain itu pada penelitian deteksi intrusi pada jaringan dengan menggunakan seleksi fitur *Information Gain*, *Chi Square* dan *Relief* menyimpulkan bahwa *Information Gain* dan *Chi Square* menghasilkan akurasi yang lebih baik daripada seleksi fitur *Relief* (Sheen dan Rajesh, 2008).

Klasifikasi digunakan untuk menemukan kelas yang tidak diketahui sebelumnya. Salah satu metode dalam klasifikasi yaitu metode *K-Nearest Neighbor* yaitu metode klasifikasi yang didasarkan atas kedekatan ketetanggaan nilai kelas (Fadila *et al.*, 2016). Dalam *KNN* untuk menghitung kedekatan ketetangannya dapat menggunakan persamaan *Euclidean distance*, *Manhattan Distance* atau *Cosine Similarity*. Implementasi metode *KNN* sangat simpel dan mudah hanya mengatur satu parameter *K*. Hal ini juga memudahkan untuk menelusuri keputusan kelas sehingga mudah untuk menganalisis dan mengubah model (Suyanto, 2018). Seringkali pada data latih terdapat kondisi dimana antara kelas yang satu dengan kelas yang lain mempunyai data dengan jumlah yang berbeda. Kondisi seperti itu disebut dengan data tidak seimbang. Metode *Neighbor Weighted K-Nearest Neighbor* menjawab permasalahan data tidak seimbang yaitu dengan pemberian bobot pada kelas yang berasal dari kategori mayoritas maka diberi bobot kecil dan pada kelas yang berasal dari kategori minoritas maka akan diberi bobot lebih besar (Ridok dan Latifah, 2015).

Dari hasil uraian sebelumnya, maka penulis memutuskan untuk melakukan penelitian dengan judul “Klasifikasi Jenis Makanan dari Citra *Smartphone* menggunakan *Neighbor Weighted K-Nearest Neighbor* dengan Seleksi Fitur *Information Gain*”. Penelitian ini diharapkan mampu mempermudah masyarakat ketika mencari resep pencarian makanan.

## 1.2 Rumusan masalah

Berdasarkan latar belakang permasalahan yang telah dipaparkan, maka dapat dirumuskan permasalahan pada penelitian ini adalah:

1. Bagaimana tingkat akurasi dari metode *Neighbor Weighted K-Nearest Neighbor*?
2. Bagaimana pengaruh seleksi fitur *Information Gain* pada klasifikasi jenis makanan dari citra *smartphone* menggunakan *Neighbor Weighted K-Nearest Neighbor*?

## 1.3 Tujuan

Berdasarkan rumusan masalah yang ada penelitian ini bertujuan untuk:

1. Mengetahui tingkat akurasi dari metode *Neighbor Weighted K-Nearest Neighbor*.
2. Mengetahui pengaruh seleksi fitur *Information Gain* pada klasifikasi jenis makanan dari citra *smartphone* menggunakan *Neighbor Weighted K-Nearest Neighbor*.

## 1.4 Manfaat

Dengan adanya penelitian ini diharapkan:

1. Bermanfaat untuk menambah pengalaman dalam menerapkan metode yang akan digunakan.
2. Bermanfaat untuk mencari resep makanan melalui citra.
3. Bermanfaat untuk menambah informasi.

## 1.5 Batasan masalah

Dalam penelitian ini, acuan yang digunakan sebagai batasan masalah adalah:

1. Dataset yang digunakan berasal dari data primer yang diambil di Ged F Fakultas Ilmu Komputer Universitas Brawijaya pada siang hari pukul 10:00-12:00.
2. Citra makanan yang digunakan sebanyak 23 jenis.
3. Citra makanan berbentuk tunggal.

## 1.6 Sistematika pembahasan

Pembuatan laporan penelitian ini dibuat dengan sistematika sebagai berikut.



## BAB I PENDAHULUAN

Pada bagian pendahuluan berisi tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah serta sistematika penulisan dari skripsi ini.

## BAB II LANDASAN KEPUSTAKAAN

Pada bagian ini berisi tentang dasar teori yang mendukung penelitian dan yang berhubungan dengan penelitian ini untuk menguatkan dasar teori yang sudah ada serta menjadi dasar penelitian tentang makanan, ekstraksi fitur menggunakan *Color Moment* dan *Gray Level Cooccurrence Matrix*, seleksi fitur *Information Gain* dan Klasifikasi *Neighbor Weighted K-Nearest Neighbor (NWKNN)*.

## BAB III METODOLOGI PENELITIAN

Pada bagian metodologi penelitian berisi tentang serangkaian langkah yang dilakukan peneliti untuk menyelesaikan permasalahan dalam implementasi seleksi fitur *Information Gain* pada citra *smartphone* dengan metode *Neighbor-Weighted K-Nearest Neighbor (NWKNN)* untuk identifikasi jenis makanan.

## BAB IV PERANCANGAN

Pada bagian perancangan berisi tentang perencanaan rancangan algoritme, rancangan pengujian, serta manualisasi dengan menggunakan metode *NWKNN* untuk klasifikasi makanan.

## BAB V IMPLEMENTASI

Pada bagian implementasi ini akan dibahas bagaimana implementasi dari algoritme berdasarkan perancangan yang telah dibuat.

## BAB VI PENGUJIAN DAN ANALISIS

Pada bagian pengujian dan analisis berisi pembahasan proses dan hasil pengujian terhadap algoritme berdasarkan hasil implementasi, serta analisis dari pengujian tersebut.

## BAB VII PENUTUP

Pada bagian penutup berisi kesimpulan dari penelitian yang telah dilakukan serta saran-saran untuk pengembangan lebih lanjut.

## BAB 2 LANDASAN KEPUSTAKAAN

Bab 2 ini berisi penjelasan dan uraian kajian pustaka, teori, konsep ataupun metode dari kepustakaan penelitian lainnya yang berkaitan dengan permasalahan pada penelitian ini. Pada landasan kepustakaan terdapat dasar teori dari berbagai sumber pustaka penelitian lainnya yang menunjang penelitian ini.

### 2.1 Tinjauan Pustaka

Terdapat beberapa penelitian sebelumnya yang memiliki keterkaitan dengan permasalahan penelitian ini. Penelitian-penelitian tersebut berisi teori dan metode yang sama tetapi obyeknya berbeda, teori dan metode yang sama tetapi obyeknya berbeda dan teori, obyek dan metodenya sama. Dalam tabel akan dijelaskan mengenai persamaan dan perbedaan dari penelitian sebelumnya serta rencana untuk penelitian ini. Tabel Tinjauan Pustaka dapat dilihat pada Tabel 2.1.

**Tabel 2.1 Tinjauan Pustaka**

No	Nama Penulis, Tahun dan Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1.	Liu, F., Liu, X. and Chen, Y. (2014) 'An efficient detection method for rare colored capsule based on RGB and HSV color space', <i>Proceedings - 2014 IEEE International Conference on Granular Computing, GrC 2014</i> , pp. 175–178. doi: 10.1109/GRC.2014.6982830.	Menggunakan color space RGB dan HSV.	Tidak menggunakan metode klasifikasi.	Menggunakan klasifikasi <i>Neighbor Weighted K-Nearest Neighbor (NWKNN)</i> .
2.	Patil, J. K. <i>et al.</i> (2011) 'Color Feature Extraction of Tomato Leaf Diseases', <i>International Journal of Engineering Trends and Technology</i> , 2(2), pp. 72–74. Available at: <a href="http://www.international">http://www.international</a>	Menggunakan ekstraksi fitur <i>Color Moments</i> .	Tidak menggunakan ekstraksi fitur tekstur.	Menggunakan ekstraksi fitur <i>Color Moments</i> dan <i>GLCM</i>

	aljournalssrg.org.			
3.	Xie, Z. <i>et al.</i> (2010) 'Texture image retrieval based on gray level co-occurrence matrix and singular value decomposition', <i>2010 International Conference on Multimedia Technology, ICMT 2010</i> , (1), pp. 3–5. doi: 10.1109/ICMULT.2010.5629822.	Menggunakan ekstraksi fitur GLCM	Tidak menggunakan ekstraksi fitur <i>Color Moments</i>	Menggunakan ekstraksi fitur <i>GCLM</i> dan <i>Color Moments</i>
4.	Dewi, R. K. and Ginardi, R. V. H. (2014) 'Identifikasi Penyakit pada Daun Tebu dengan Gray Level Co-Occurrence Matrix dan Color Moments', <i>Jurnal Teknologi Informasi dan Ilmu Komputer</i> , 1(2), p. 70. doi: 10.25126/jtiik.201412114.	Ekstraksi Fitur menggunakan <i>Color Moments</i> dan <i>Gray Level Cooccurrence Matrix</i>	Metode klasifikasi menggunakan <i>Support Vector Machine (SVM)</i> .	Metode klasifikasi menggunakan <i>Neighbor Weighted K-Nearest Neighbor (NWKNN)</i> .
5.	Aini, S. H. A., Sari, Y. A. and Arwan, A. (2018) 'Seleksi Fitur Information Gain untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode K-Nearest Neighbor dan Naïve Bayes', <i>Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer</i> , 2(9), pp. 2546–2554.	Seleksi fitur <i>Information Gain</i>	Metode klasifikasi menggunakan <i>K-Nearest Neighbor(KNN)</i>	Metode klasifikasi menggunakan <i>Neighbor Weighted K-Nearest Neighbor (NWKNN)</i>
6.	Chormunge, S. and Jena, S. (2016) 'Efficient	Seleksi fitur <i>Information</i>	Metode klasifikasi	Metode klasifikasi

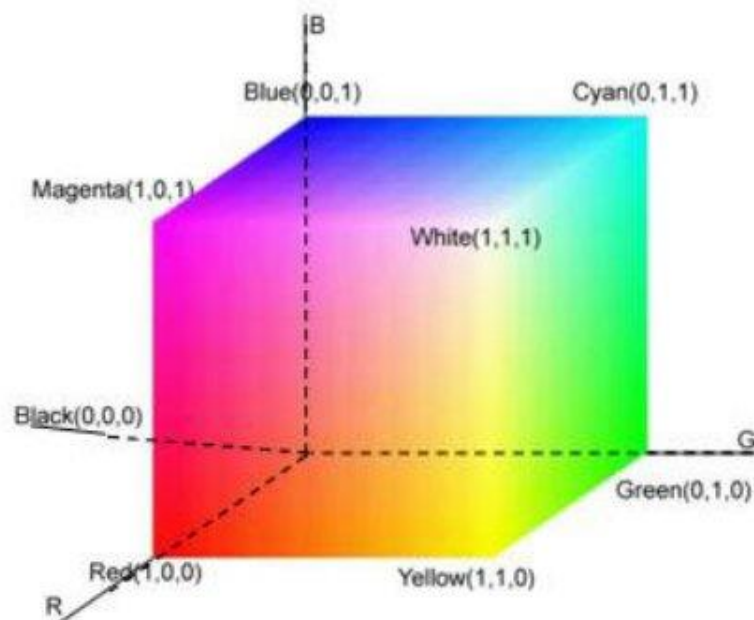
	Feature Subset Selection Algorithm for High Dimensional Data', <i>International Journal of Electrical and Computer Engineering (IJECE)</i> , 6(4), p. 1880. doi: 10.11591/ijece.v6i4.9800.	<i>Gain</i>	menggunakan <i>Naïve Bayes</i>	menggunakan <i>Neighbor Weighted K-Nearest Neighbor (NWKNN)</i>
7.	Fadila, P. N. et al. (2016) 'Identifikasi Jenis Attention Deficit Hyperactivity Disorder (Adhd) Pada Anak Usia Dini Menggunakan Metode Neighbor Weighted K-Nearest Neighbor (Nwknn)', <i>Teknologi Informasi dan Ilmu Komputer (JTIK)</i> , 3(3), pp. 194–200.	Menggunakan algoritme klasifikasi <i>Neighbor Weight K-Nearest Neighbor (NWKNN)</i>	Dataset tidak menggunakan citra makanan dan tidak melalui ekstraksi fitur dan seleksi fitur.	Menggunakan dataset citra makanan dan terdapat metode ekstraksi fitur dan seleksi fitur.
8.	Ridok, A. and Latifah, R. (2015) 'Klasifikasi Teks Bahasa Indonesia Pada Corpus Tak Seimbang Menggunakan NWKNN', <i>Konferensi Nasional Sistem dan Informatika 2015</i> , pp. 222–227.	Menggunakan algoritme klasifikasi <i>Neighbor Weight K-Nearest Neighbor (NWKNN)</i>	Dataset menggunakan teks Bahasa Indonesia dan tidak melalui ekstraksi fitur	Menggunakan dataset citra makanan dan terdapat metode ekstraksi fitur dan seleksi fitur

## 2.2 Ekstraksi Fitur

Ekstraksi fitur adalah proses pengambilan ciri sebuah objek yang dapat menggambarkan karakteristik dari objek tersebut (Satria dan Mushthofa, 2013). Fitur yang dibutuhkan dapat membedakan antara satu objek dengan objek yang lain tetapi diharapkan dengan jumlah yang sedikit. Terdapat beberapa fitur dalam citra yaitu fitur warna, bentuk dan tekstur.

### 2.2.1 Transformasi Warna RGB ke HSV

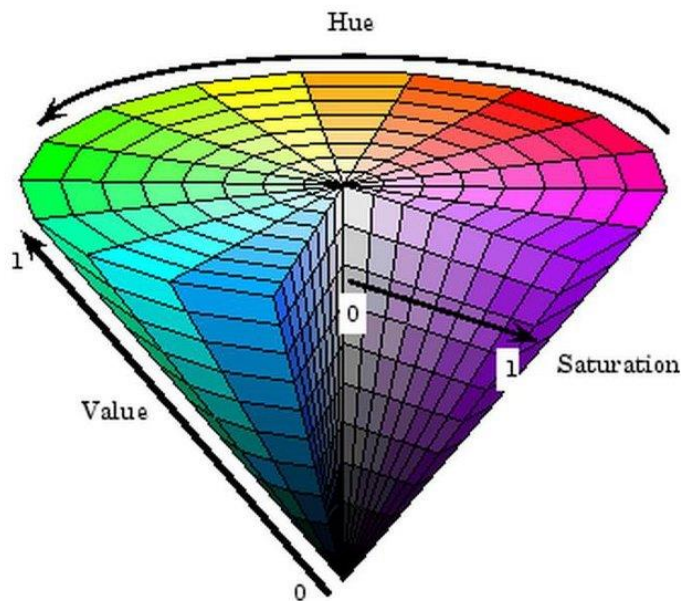
Manusia melihat warna karena adanya cahaya yang dipantulkan oleh objek. Dalam hal ini, spektrum cahaya kromatis berkisar antara 400-700 nm. Istilah kromatis berarti kualitas warna cahaya yang ditentukan oleh panjang gelombang (Kadir dan Susanto, 2013). Warna RGB atau *Red*, *Green* dan *Blue* adalah warna yang paling sering digunakan pada citra. Ruang warna dinyatakan oleh tiga warna primer secara fisik, sehingga fisik dan maknanya jelas. Rumus perbedaan warna yang umum digunakan dalam *RGB Color Space* yaitu perbedaan warna jarak dan perbedaan warna sudut (Feng, Xiaoyu dan Yi, 2014). Ruang warna *RGB* direpresentasikan dalam bentuk kubus dengan resolusi 24 bit ditunjukkan pada Gambar 2.1.



**Gambar 2.1 Representasi Ruang Warna RGB**

Sumber: (Kolivand dan Sunar, 2011)

Ruang warna *HSV* ditunjukkan oleh tiga elemen warna yaitu *Hue*, *Saturation* dan *Value* dimana ruang warna ini direpresentasikan dalam bentuk kerucut. *HSV* memiliki warna yang non linier yang konsisten dengan persepsi manusia. Setiap warna memiliki ruang yang terpisah, sehingga cocok untuk pemrosesan gambar (Feng, Xiaoyu dan Yi, 2014). Ruang warna *HSV* direpresentasikan dalam bentuk kerucut ditunjukkan pada Gambar 2.2.



**Gambar 2.2 Representasi Ruang Warna HSV**

Sumber: (Kemal dan Nihat, 2014)

Untuk mentransformasikan dari ruang *RGB* ke *HSV* dapat dilihat pada rumus berikut (Kadir dan Susanto, 2013).

$$r = \frac{R}{(R + G + B)}, g = \frac{G}{(R + G + B)}, b = \frac{B}{(R + G + B)} \quad (2.1)$$

1. *Value* merupakan intensitas gray level (cerah atau gelapnya suatu warna). Pada Gambar 2.2 *value* berada dalam perubahan warna putih menuju abu-abu dan terakhir pada ujung kerucut berwarna hitam. Rumus *value* yaitu nilai maksimal dari ruang warna RGB.

$$V (value) = \max(r, g, b) \quad (2.2)$$

2. *Saturation* adalah tingkat kemurnian warna yang menyatakan seberapa banyak cahaya putih tercampur di dalam *hue*. Pada Gambar 2.2 *saturation* berada dalam warna yang terletak pada tepi terluar kerucut hingga nuansa warna yang mendekati titik pusat kerucut.

$$S (saturation) = \begin{cases} 0, & \text{jika } V = 0 \\ 1 - \frac{\min(r, g, b)}{V}, & > 0 \end{cases} \quad (2.3)$$

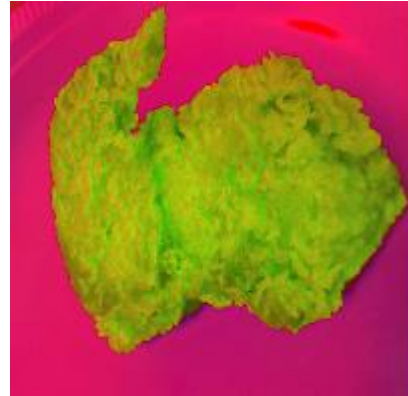
3. *Hue* adalah warna yang dikenal oleh mata manusia seperti merah dan biru. *Hue* digunakan untuk memperoleh sudut yang tepat antara 0-360. Sudut warna yang terbentuk merepresentasikan komponen kromatik yaitu mulai dari sudut 0° (merah), 30° (oranye), 60° (kuning), 120° (hijau), 180° (cyan), 240° (biru), dan 300° (ungu).

$$H(\text{hue}) = \begin{cases} 0, & \text{jika } S = 0 \\ 60^\circ * \left( \frac{g - b}{S * V} \right), & \text{jika } V = r \\ 60^\circ * \left( \frac{b - r}{S * V} + 2 \right), & \text{jika } V = b \\ 60^\circ * \left( \frac{r - g}{S * V} + 4 \right), & \text{jika } V = b \end{cases} \quad (2.4)$$

Berikut contoh hasil transformasi warna citra RGB ke HSV ditunjukkan pada Gambar 2.3.



(a)



(b)

**Gambar 2.3 Transformasi citra (a) RGB ke (b) HSV**

### 2.2.2 Color Moments

*Color moments* merupakan salah satu metode ekstraksi fitur yang dapat digunakan untuk membedakan citra berdasarkan warna. Konsep dari *Color Moments* adalah probabilitas distribusi warna yang menginterpretasikan distribusi warna yang digunakan untuk membedakan citra (Dewi dan Ginardi, 2014). Metode ini adalah yang efektif untuk analisis citra berdasarkan warna karena metode tersebut memiliki dimensi vektor fitur yang paling rendah dan juga kompleksitas komputasional yang paling rendah jika dibandingkan dengan metode lainnya seperti *Color Histogram*, *Color Correlogram* dan *Color Structure Descriptor* (Patil et al., 2011).

Terdapat 3 jenis perhitungan untuk metode *Color Moments* menurut (Dewi dan Ginardi, 2014) yaitu:

1. *Color Moments 1 (mean)* adalah rata-rata nilai pixel ( $P_{ij}$ ) pada masing-masing warna. Rumus untuk color moments 1 dapat dilihat pada Persamaan 2.5.

$$\mu = \frac{1}{N} \sum_{j=1}^N P_{ij} \quad (2.5)$$

dimana:

$\mu$  = mean

$P_{ij}$  = piksel j pada channel warna i

N = nilai maksimal seluruh piksel pada citra

2. *Color Moments 2 (standar deviasi)*. Variance adalah luas sebaran distribusi data yang dapat dihitung dengan Persamaan 2.6.

$$\sigma = \sqrt{\frac{1}{N} \sum_{j=1}^N ((P_{ij} - \mu)^2)} \quad (2.6)$$

dimana:

$\sigma$  = standar deviasi

$P_{ij}$  = piksel  $j$  pada channel warna  $i$

N = jumlah dari seluruh piksel pada citra

$\mu$  = rata-rata nilai pixel

3. *Color Moments 3 (skewness)* digunakan untuk menentukan derajat ketidaksimetrisan pada distribusi warna. Skewness adalah pengukuran dimana sebuah distribusi warna dikatakan simetris apabila seimbang antara kiri dengan kanan pada *center point* dan nilai *skewness* bernilai 0. Distribusi warna dikatakan condong ke kiri jika nilai *skewness* bernilai negative, sebaliknya jika distribusi warna condong ke kanan maka nilai *skewness* bernilai positif. Rumus *Color Moments 3* dapat dilihat pada Persamaan 2.7.

$$\theta = \frac{\frac{1}{N} \sum_{j=1}^N (P_{ij} - \mu)^3}{\sigma^3} \quad (2.7)$$

dimana:

$\theta$  = skewness

$P_{ij}$  = piksel  $j$  pada channel warna  $i$

N = jumlah dari seluruh piksel pada citra

$\mu$  = rata-rata nilai pixel

$\sigma$  = standar deviasi

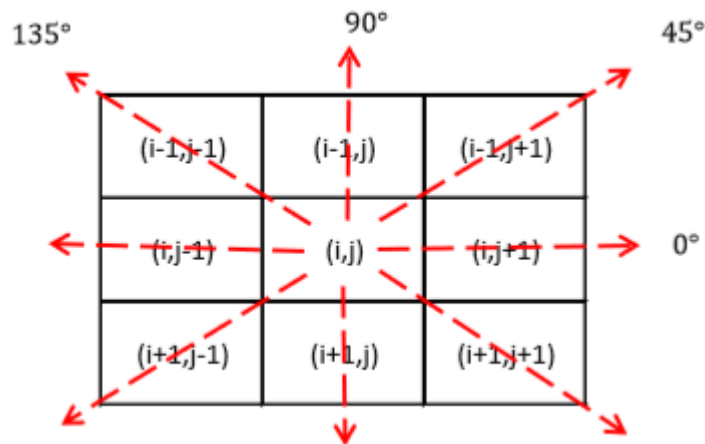
### 2.2.3 Gray Level Cooccurrence Matrix (GLCM)

GLCM merupakan salah satu metode yang digunakan untuk ekstraksi fitur tekstur pada citra dimana metode ini melakukan analisis terhadap *pixel* citra untuk mengetahui jejak tingkat keabuan yang berpasangan (bertetangga) yang sering muncul (Xie *et al.*, 2010). GLCM menggunakan perhitungan orde kedua yaitu perhitungan hubungan antar pasangan dua piksel citra (Kadir dan Susanto, 2013). Proses GLCM adalah dengan membentuk sebuah matrik *co-occurrence* (kejadian) pada citra dan dilanjutkan dengan mengekstraksi fitur dari matriks tersebut (Kasim dan Harjoko, 2014).

Untuk membentuk sebuah matriks *co-occurrence* diperoleh dengan cara menjumlahkan kejadian satu level nilai piksel bertetangga dengan satu level nilai



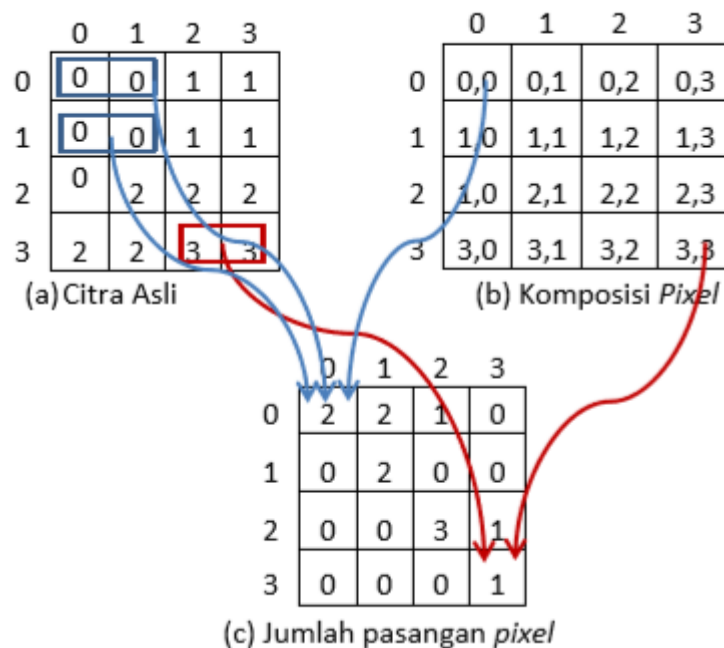
piksel lain dalam jarak ( $d$ ) dan orientasi sudut ( $\theta$ ) tertentu. Sudut dibentuk dari nilai koordinat gambar menggunakan *GLCM* adalah  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ ,  $135^\circ$  sedangkan jarak ditetapkan biasanya 1 piksel. Sudut yang terbentuk dapat dilihat pada Gambar 2.14.



**Gambar 2.14 Pixel dengan pengambilan berbagai sudut**

Sumber: (Kadir dan Susanto, 2013)

Sudut  $0^\circ$  dibentuk dari hubungan antara dua piksel tetangga secara horizontal. Sudut  $45^\circ$  adalah dari hubungan antara dua piksel tetangga yang membentuk sudut  $45^\circ$  arah timur laut. Sudut  $90^\circ$  adalah dari hubungan antara dua piksel tetangga secara vertical. Sudut  $135^\circ$  adalah dari hubungan antara dua piksel tetangga yang membentuk sudut  $135^\circ$  arah barat laut. Ilustrasi pada Gambar 2.2 menunjukkan hubungan antara dua piksel tetangga membentuk sudut  $0^\circ$  secara horizontal dengan contoh jumlah pasangan piksel yang diambil antara 0,0 dan 3,3 dan jarak ( $d$ ) = 1. Ilustrasi perhitungan dapat dilihat pada Gambar 2.4.



**Gambar 2.4 Penentuan Awal Nilai *Matrix Framework***

Sumber : (Kadir dan Susanto, 2013)

Matriks pada gambar diatas adalah salah satu matriks *framework* pada citra yang selanjutnya diproses menjadi matriks yang simetris. Untuk memperoleh matriks simetris yaitu dengan cara menjumlahkan nilai matriks dengan *transposenya*. Pembentukan matriks tersebut dapat dilihat pada Gambar 2.12.

$$\begin{pmatrix} 2 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

**Gambar 2.5 Hasil pembentukan nilai GLCM simetris**

Sumber: (Kadir dan Susanto, 2013)

Hasil dari matriks simetris akan dilakukan normalisasi supaya tidak memiliki ketergantungan terhadap ukuran citra. Matriks normalisasi jika dijumlahkan setiap piksel dari citra akan menghasilkan nilai 1. Berikut matriks yang telah dinormalisasi dapat dilihat pada Gambar 2.4.

$\frac{4}{24}$	$\frac{2}{24}$	$\frac{1}{24}$	$\frac{0}{24}$
$\frac{2}{24}$	$\frac{4}{24}$	$\frac{0}{24}$	$\frac{0}{24}$
$\frac{1}{24}$	$\frac{0}{24}$	$\frac{6}{24}$	$\frac{1}{24}$
$\frac{0}{24}$	$\frac{0}{24}$	$\frac{1}{24}$	$\frac{2}{24}$
$\frac{24}{24}$	$\frac{24}{24}$	$\frac{24}{24}$	$\frac{24}{24}$

**Gambar 2.6 Hasil GLCM matriks setelah dinormalisasi**

Sumber: (Kadir dan Susanto, 2013)

Terdapat 14 fitur GLCM yang diusulkan oleh (Haralick dan Shanmugam, 1973) yang bisa dihitung dari hasil matriks *co-occurrence*. Fitur-fitur tersebut diantaranya:

1. *Homogeneity, Angular Second Moment (ASM)*

$$ASM = \sum_i^l \sum_{j=0}^w \{P(i,j)^2\} \quad (2.8)$$

*ASM* atau *energy* digunakan untuk mengukur homogenitas gambar pada matriks *co-occurrence*. Nilai *energy* semakin besar apabila pasangan piksel yang memenuhi kondisi matriks *co-occurrence* terkoneksi pada beberapa koordinat dan semakin kecil apabila letak koordinatnya menyebar.

2. *Contrast*

$$Contrast = \sum_{n=0}^G n^2 \left\{ \sum_{i=0}^G \sum_{j=0}^G P(i,j) \right\}, |i-j| = n \quad (2.9)$$

*Contrast* digunakan untuk mengukur variasi atau perbedaan intensitas tingkat keabuan dalam citra. Nilai kontras semakin besar apabila perbedaan intensitas citra tinggi dan semakin kecil apabila perbedaan intensitas rendah.

3. *Correlation*

$$Correlation = \frac{(i * j) * P(i,j) - (\mu_x * \mu_y)}{\sigma_x * \sigma_y} \quad (2.10)$$

*Correlation* digunakan untuk mengukur korelasi dan ketergantungan antar *pixel* dengan tingkat keabuan *i* dan *pixel* dengan tingkat keabuan *j* pada citra.

4. *Sum Of Square, Variance*

$$Variance = \sum_{i=0}^G \sum_{j=0}^G (i - \mu)^2 P(i,j) \quad (2.11)$$

*Variance* digunakan untuk mengukur sebaran atau variasi nilai keabuan pada matriks *cooccurrence* awal. Citra dengan sebaran derajat keabuan yang kecil akan menghasilkan *variance* yang kecil pula.

5. *Local Homogeneity, Inverse Difference Moment (IDM)*

$$IDM = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{1}{1 + (i - j)^2} P(i, j)^2 \quad (2.12)$$

*IDM* atau *homogenitas* digunakan untuk mengukur homogenitas variasi intensitas citra dengan tingkat keabuan sejenis. Nilai homogenitas semakin besar apabila variasi intensitas citra kecil (citra yang homogen) dan semakin kecil apabila variasi intensitas citra besar (citra yang tidak homogen).

6. *Sum Average*

$$AVER = \sum_{i=2}^{2G} iP_{x+y}(i) \quad (2.13)$$

*Sum Average* digunakan untuk mengukur banyaknya nilai rata-rata *pixel* pada distribusi tingkat keabuan.

7. *Sum Entropy*

$$SENT = - \sum_{i=2}^{2G} iP_{x+y}(i) \log(P_{x+y}(i)) \quad (2.14)$$

*Sum Entropy* digunakan untuk mengukur banyaknya tingkat keabu-abuan yang acak.

8. *Sum Variance*

$$SVAR = \sum_{i=2}^{2G} (i - SENT)^2 P_{x+y}(i) \quad (2.15)$$

*Sum Variance* digunakan untuk mengukur seberapa banyak variasi tingkat keabuan dari nilai rata-rata.

9. *Entropy*

$$Entropy = - \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i, j) * \log(P(i, j)) \quad (2.16)$$

*Entropy* digunakan untuk mengukur tingkat ketidakaturan bentuk atau distribusi intensitas citra matriks co-occurrence. Nilai *entropy* semakin besar apabila citra tidak homogen dan semakin kecil apabila citra homogen.

10. *Difference Entropy*

$$DENT = - \sum_{i=0}^{G-1} iP_{x-y}(i) \log(P_{x-y}(i)) \quad (2.17)$$

Difference Entropy digunakan untuk mengukur variasi perbedaan mikro (lokal).

#### 11. Difference Variance

$$DVAR = - \sum_{i=0}^{G-1} i^2 P_{x-y}(i) \quad (2.18)$$

Difference Variance digunakan untuk mengukur variasi *pixel* lokal.

#### 12. Information Measure Of Correlation 1

$$HXY = - \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \log(P(i,j)) \quad (2.19)$$

**HX** dan **HY** merupakan entropy dari  $P_x, P_y$  (2.20)

$$HXY1 = - \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \log(P_x(i)P_y(j)) \quad (2.21)$$

$$HXY2 = - \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P_x(i)P_y(j) \log(P_x(i)P_y(j)) \quad (2.22)$$

$$IMoC1 = \frac{HXY - HXY1}{\max(HX, HY)} \quad (2.23)$$

#### 13. Information Measure Of Correlation 2

$$IMoC2 = (1 - \exp[-2.0(HXY2 - HXY)])^{1/2} \quad (2.24)$$

#### 14. Maxima Correlation Coefficient

Dari 14 fitur diatas yang diusulkan oleh Haralick, terdapat 1 fitur yang tidak dipakai yaitu *Maxima Corelation Coefficient*. Fitur tersebut tidak digunakan karena ketidakstabilan dalam komputasi (Haralick dan Shanmugam, 1973; Mahardika, Sari dan Dewi, 2018).

### 2.3 Seleksi Fitur

Seleksi fitur adalah metode seleksi untuk mengurangi atribut-atribut yang tidak relevan. Seleksi fitur akan memilih fitur terbaik berdasarkan banyaknya data yang mempunyai informasi lebih banyak. Ada 2 pendekatan pada metode seleksi fitur yaitu *wrapper* dan *filter* (Aini, Sari dan Arwan, 2018). Metode *wrapper* merupakan metode seleksi fitur dengan melakukan evaluasi dan pemilihan atribut dengan memperkirakan akurasi pada saat pembelajaran. Metode ini menghilangkan atribut dan menguji dampak penghilangan fitur

tersebut. Fitur yang mempunyai perbedaan yang signifikan dan mempunyai dampak untuk akurasi lebih baik akan dijadikan fitur untuk klasifikasi. Sedangkan metode *filter* merupakan pendekatan seleksi fitur yang menggunakan karakteristik umum data itu sendiri dan bekerja secara terpisah dengan algoritme pembelajaran sehingga biasanya lebih cepat dan terukur.

### 2.3.1 Information Gain

*Information Gain* adalah metode pemeringkatan atribut yang paling sederhana, banyak digunakan dalam aplikasi kategorisasi teks, untuk analisis data *microarray* dan analisis data gambar (Chormunge dan Jena, 2016). Metode ini akan menghapus fitur-fitur yang kurang relevan. Atribut yang digunakan pada metode seleksi dengan pendekatan *filter* ini adalah atribut kelas nominal. Tidak menutup kemungkinan untuk setiap fitur berisi atribut campuran seperti nominal, ordinal dan kontinu. Untuk mengubah atribut kontinu menjadi data kategorikal maka dilakukan teknik diskretisasi (Hermawati, 2013). Berikut langkah-langkah diskretisasi yaitu:

1. Menentukan jangkauan

$$\text{Jangkauan } (J) = \text{data terbesar} - \text{data terkecil} \quad (2.25)$$

2. Menentukan banyaknya kelas dengan aturan *Sturges*

$$k = 1 + 3,3 * \log n \quad (2.26)$$

Dimana:

k = banyaknya kelas

n = banyaknya data

3. Menentukan panjang/interval kelas

$$\text{interval } (i) = J/k \quad (2.27)$$

4. Memasukkan data tunggal ke dalam data kelompok

Menurut (Aini, Sari dan Arwan, 2018) langkah-langkah dari metode *Information Gain* yaitu:

1. Menghitung nilai *entropy* dengan Persamaan 2.28. *Entropy* adalah suatu parameter untuk mengukur heterogenitas dalam suatu himpunan data (Suyanto, 2018).

$$\text{Entropy } (S) = \sum_i^c -P_i \log_2 P_i \quad (2.28)$$

Dimana:

c = jumlah nilai yang ada pada kelas klasifikasi

$P_i$  = Peluang munculnya sampel untuk kelas  $i$

2. Menghitung nilai dari *Information Gain* dengan Persamaan 2.29.

$$Gain(S, A) = Entropy(S) - \sum Values(A) \frac{|S_v|}{|S|} Entropy(S_v) \quad (2.29)$$

Dimana:

$A$	= atribut
$v$	= nilai yang mungkin untuk atribut $A$
$ S_v $	= jumlah sampel untuk nilai $v$
$ S $	= jumlah untuk seluruh sampel data
$Entropy(S_v)$	= <i>entropy</i> untuk sampel-sampel yang memiliki nilai $v$

## 2.4 Klasifikasi

Klasifikasi adalah salah satu metode pada data mining yang bersifat *supervised* dimana proses identifikasi dilakukan dengan pembelajaran pada data yang terdapat kelasnya. Terdapat 2 langkah yang dilakukan yaitu langkah pembelajaran berfungsi untuk membangun model yang digunakan dan langkah klasifikasi berfungsi untuk memprediksi label kelas dari data yang diberikan (Han, Kamber dan Pei, 2012).

### 2.4.1 K-Nearest Neighbor (K-NN)

*K-NN* merupakan salah satu metode klasifikasi yang didasarkan pada fungsi jarak untuk membandingkan pasangan sampel data. Metode ini termasuk kategori *lazy learning* karena data latih menunggu hingga diberikannya data uji untuk dilakukan klasifikasi (Fadila *et al.*, 2016). Metode yang diusulkan oleh T.M. Cover dan P.E. Hart san telah menjadi metode yang paling banyak digunakan dalam memecahkan kasus-kasus klasifikasi (Harfiya, Widodo dan Wihandika, 2017). Metode *K-NN* memberikan kemudahan untuk melakukan implementasi dengan hanya mengatur satu parameter  $K$ . Hal ini juga memudahkan untuk menelusuri keputusan kelas yang dibuat sehingga mudah untuk menganalisis dan mengubah model (Suyanto, 2018). Berdasarkan dataset yang digunakan, metode *K-NN* mempunyai dataset input kontinu dan output diskrit. Langkah-langkah metode *K-NN* yaitu (Harfiya, Widodo dan Wihandika, 2017):

1. Definisikan nilai  $K$ .  
Nilai  $K$  ditentukan untuk memilih berapa data yang diambil dari data yang jaraknya paling dekat.
2. Penentuan jarak antara data latih dengan data uji.  
Jarak digunakan untuk memilih individu yang mirip. Jarak yang digunakan pada penelitian ini yaitu *Manhattan Distance*, *Euclidean Distance* dan *Cosine Similarity*.
3. Urutkan data berdasarkan hasil penentuan jarak.  
Data latih dan data uji yang telah ditentukan jaraknya maka diurutkan sesuai dengan metode jarak yang digunakan.

4. Bentuk kelompok berdasarkan nilai dengan ketetanggaan terdekat.  
Dari hasil pengurutan pilih sebanyak  $K$  data terdekat yang kelompokkan berdasarkan kelasnya.
5. Pilih kelompok yang paling sering muncul.  
Dari kelompok yang terbentuk pilih kelompok yang datanya sering banyak muncul untuk dijadikan prediksi kelas dari data uji.

#### 2.4.2 Neighbor Weighted K-Nearest Neighbor (NWKNN)

*Neighbor Weighted K-Nearest Neighbor (NWKNN)* merupakan pengembangan dari metode *K-Nearest Neighbor* dimana pada proses klasifikasi terdapat pembobotan untuk setiap jenis/kelas data (Fadila *et al.*, 2016). Metode ini muncul karena adanya permasalahan data tidak seimbang. Data tidak seimbang adalah dataset pada data latih yang mempunyai jumlah jenis/kelas antara satu kelas dengan kelas yang lain berbeda secara signifikan.

Langkah-langkah dari metode ini hampir sama dengan metode *KNN*. Bedanya adalah terdapat pembobotan untuk setiap jenis/kelas untuk menentukan klasifikasi terhadap data uji. Langkah-langkah dari metode *NWKNN* yaitu:

1. Menentukan parameter  $K$ .
2. Menghitung nilai kedekatan ketetanggaan antara data uji terhadap data latih dengan menggunakan *Manhattan Distance*, *Euclidean Distance* dan *Cosine Similarity*
3. Urutkan data berdasarkan hasil penentuan jarak.  
Data latih dan data uji yang telah ditentukan jaraknya maka diurutkan sesuai dengan metode jarak yang digunakan.
4. Bentuk kelompok berdasarkan nilai dengan ketetanggaan terdekat.  
Dari hasil pengurutan pilih sebanyak  $K$  data terdekat yang kelompokkan berdasarkan kelasnya.
5. Penentuan bobot.

Persamaan perhitungan bobot:

$$Weight_i = \frac{1}{\left( \frac{Num(c)_i^d}{Min\{Num(c_n^d) | n = 1, \dots, k\}} \right)^{1/exp}} \quad (2.30)$$

Dimana:

- $Num(C_i^d)$  = banyaknya data latih pada kelas  $i$
- $Num(C_n^d)$  = banyaknya data latih  $d$  pada kelas  $i$ , dimana  $i$  terdapat dalam himpunan  $k$  tetangga terdekat
- $Exp$  = Eksponen (Nilai  $exp$  lebih dari 1)



$Weight_i$  = bobot kelas ke  $i$

6. Perhitungan skor untuk memperoleh hasil klasifikasi

$$Skor(X, C_i) = Weight_i * \left( \sum_{dj \in NWKNN(x)} Sim(q, d_j) * \delta(d_j, C_i) \right) \quad (2.31)$$

Dimana:

$Skor(X, C_i)$  = skor dari tetangga terdekat dari data uji  $x$  pada kelas  $i$

$dj \in NWKNN(x)$  = data latih  $dj$  pada kumpulan tetangga terdekat dari data uji  $x$

$Sim(q, d_j)$  = nilai Cosim antara data uji  $q$  dan data latih ke  $j$

$\delta(d_j, C_i)$  = akan bernilai 1 jika nilai jarak anggota  $C_i$  dan bernilai 0 jika jarak bukan anggota  $C_i$

$C_i$  = jenis atau kelas ke  $i$

### 2.4.3 Manhattan Distance

*Manhattan distance* atau biasa disebut L1 distance merupakan pengembangan dari metode Lp norm. Metode ini bekerja dengan menghitung jarak yang paling pendek antara dua poin. Perhitungan jarak antara  $p = (x_1, y_1)$  dengan  $q = (x_2, y_2)$  menggunakan *Manhattan Distance* ditunjukkan pada Persamaan 2.21.

$$MH(p, q) = |x_1 - x_2| + |y_1 - y_2| \quad (2.32)$$

Jika memiliki variable sepanjang  $n$ , misalnya  $p = (x_1, x_2, x_3, \dots, x_n)$  dan  $q = (y_1, y_2, y_3, \dots, y_n)$ . Maka untuk mengukur jarak menggunakan *Manhattan Distance* dapat ditunjukkan pada Persamaan 2.22 berikut.

$$MH(p, q) = |x_1 - y_1| + |x_2 - y_2| + |x_3 - y_3| + \dots + |x_n - y_n| \quad (2.33)$$

### 2.4.4 Euclidean Distance

Jarak Euclidean adalah salah satu metode pengukuran jarak dari dua atau vektor yang paling sering digunakan yaitu dengan menghitung akar dari kuadrat perbedaan dua vektor (Hartono, 2017). Metode ini digunakan untuk mendeteksi tingkat ketidaksamaan citra dengan mengisi nilai vektor  $p$  dan  $q$  dengan nilai fitur citra yang akan dideteksi tingkat ketidaksamaannya. Persamaan dari pengukuran jarak ini dapat dilihat pada Persamaan 2.23.

$$d(x, y) = \sqrt{\sum_{r=1}^n (x_i - y_i)^2} \quad (2.34)$$

Dimana:

$d(x, y)$  : Jarak Euclidean

$x_i$  : data pada  $x$  ke  $i$

$y_i$  : data pada  $y$  ke  $i$

### 2.4.5 Cosine Similarity

*Cosine Similarity (CoSim)* adalah metode perhitungan jarak berdasarkan kemiripan antara dua buah vektor (Putri, Ridok dan Indriati, 2013). Semakin mirip vektor antara data latih dan data uji maka akan semakin sesuai.

Persamaan yang digunakan untuk menghitung *Cosim* dapat dilihat pada Persamaan 2.24 (Putri, Ridok dan Indriati, 2013).

$$d(q, d_j) = \frac{\vec{d_j} \circ \vec{q}}{|\vec{d_j}| \circ |\vec{q}|} = \frac{\sum_{i=1}^m w_{ij} \circ w_{iq}}{\sqrt{\sum_{i=1}^m w_{ij}^2 \circ \sum_{i=1}^m w_{iq}^2}} \quad (2.35)$$

Keterangan:

$q$  : Data Uji

$d_j$  : Data Latih ke  $j$

$w_{ij}$  : Nilai data latih  $i$  kolom  $j$

$w_{iq}$  : Nilai data uji  $i$  kolom  $q$

## 2.5 Pengujian Algoritme

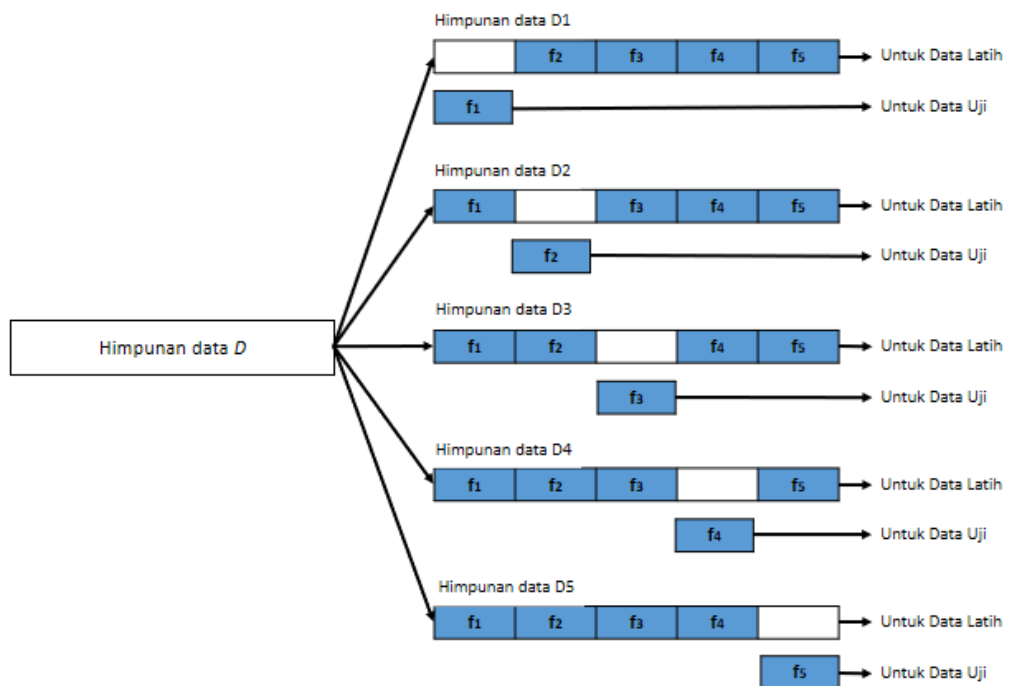
### 2.5.1 Pengujian Akurasi

Akurasi merupakan suatu cara untuk mengetahui berapa banyak nilai yang memiliki kesamaan dengan data sebenarnya. Akurasi dihasilkan dalam bentuk presentase antara jumlah identifikasi nilai benar dibagi dengan keseluruhan data. Berikut persamaan untuk menghitung akurasi:

$$akurasi = \frac{jumlah\ identifikasi\ benar}{jumlah\ data} \times 100\% \quad (2.36)$$

### 2.5.2 Pengujian *k-Fold Cross Validation*

Metode *k-Fold Cross Validation* adalah metode yang membagi sejumlah himpunan data  $D$  menjadi  $k$  (biasanya disebut *fold*) yang saling bebas:  $f_1, f_2, \dots, f_k$ , sehingga masing-masing *fold* berisi  $1/k$  bagian data (Suyanto, 2018). Himpunan data dibangun sebanyak  $k$  yaitu  $D_1, D_2, \dots, D_k$  dengan masing-masing berisi  $(k - 1)$  *fold* untuk data latih dan 1 *fold* untuk data uji. Ilustrasi *k-Fold Cross Validation* dapat dilihat pada Gambar 2.7



**Gambar 2.7 k-Fold Cross Validation**

Gambar 2.7 membagi himpunan data  $D$  sebanyak  $k = 5$  dengan masing-masing sub himpunan data  $D_1$  berisi  $f_2, f_3, f_4$  dan  $f_5$  sebagai data latih dan  $f_1$  sebagai data uji dan seterusnya. Dengan metode *k-Fold Cross Validation* ini dapat mengukur kualitas klasifikasi yang dibangun.

## BAB 3 METODOLOGI

Pada Bab 3 dijelaskan mengenai metodologi dan langkah-langkah yang akan dilakukan pada penelitian untuk klasifikasi makanan dari citra *smarthpone*.

### 3.1 Tipe Penelitian

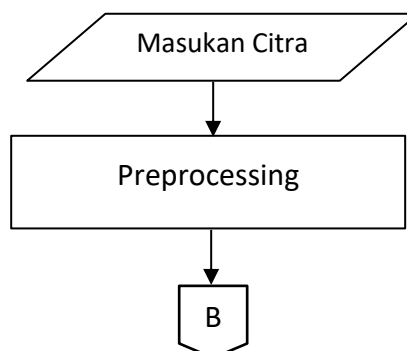
Tipe penelitian yang diterapkan pada penelitian klasifikasi jenis makanan dari citra smartphone menggunakan *NWKNN* dengan seleksi fitur *Information Gain* adalah tipe non-implementatif analitik.

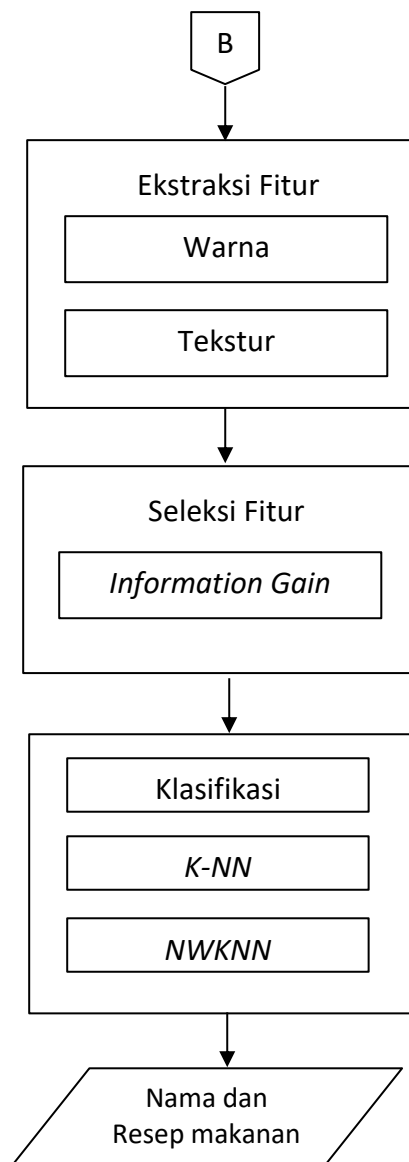
### 3.2 Strategi Penelitian

Strategi pengambilan data pada penelitian ini menggunakan data dari citra makanan dari piring plastik putih sebagai alas. Data diambil pada tanggal 28 September 2018 yang dilakukan oleh Tim Food Project Fakultas Ilmu Komputer di ged F 9. Data yang telah diambil kemudian dilakukan *rename* sesuai dengan kode dan nama makanan. Rincian pengumpulan data dapat dilihat sebagai berikut.

1. Citra dengan makanan tunggal berbentuk padat
2. Dalam satu citra diambil tiga jenis kemiringan
3. Dalam satu citra diambil dengan kondisi yaitu kondisi utuh, dimakan  $\frac{1}{4}$ , dimakan  $\frac{1}{2}$  dan dimakan  $\frac{3}{4}$ .
4. Tingkat intensitas cahaya yakni dengan cahaya matahari pada pukul 09:00 sampai 12:00
5. Tempat pengambilan dataset di Gedung F 9 Fakultas Ilmu Komputer Universitas Brawijaya.
6. Smartphone yang digunakan adalah *Xiaomi Redmi 3 Pro*.
7. Jumlah dataset yang diambil sebanyak 23 jenis makanan dengan 529 data latih tidak seimbang dan 23 data uji.

Selanjutnya data yang telah disimpan akan dilakukan proses yang ada pada Gambar 3.1.





**Gambar 3.1 Diagram alir proses**

### **3.3 Peralatan Pendukung**

Peralatan pendukung dalam melakukan proses pengerjaan penelitian ini berupa spesifikasi perangkat keras dan perangkat lunak.

Spesifikasi peralatan pendukung perangkat keras yang digunakan dalam penelitian ini dapat dilihat rinciannya sebagai berikut.

1. *Processor: Intel® Core™ i5-8520UR CPU @ 2.00GHz*
2. *Memory: 8 GB*

3. *VGA: NVIDIA Geforce 930 Mx*
4. *SSD 120 GB*
5. *Monitor 14 inch*
6. *Keyboard*
7. *Mouse*

Spesifikasi peralatan pendukung perangkat lunak yang digunakan dalam penelitian ini dapat dilihat rinciannya sebagai berikut.

1. *Operating System Microsoft Windows 10 Pro*
2. Bahasa pemrograman menggunakan Python 3.6.4
3. Pustaka yang digunakan adalah Numpy, OpenCV 2
4. *Integrated Development Environment* menggunakan *Spyder*

### 3.4 Lokasi Penelitian

Penelitian klasifikasi jenis makanan dari citra *smartphone* menggunakan *NWKNN* dengan seleksi fitur *Information Gain* dilakukan di Laboratorium Riset Komputasi Cerdas FILKOM-UB.

### 3.5 Perancangan Algoritme

Perancangan algoritme adalah tahapan yang digunakan sebelum implementasi dari algoritme. Perancangan algoritme bertujuan untuk merancang alur dari program yang akan dibuat mulai dari *preprocessing*, ekstraksi fitur dengan metode *color moments* dan *GLCM*, seleksi fitur *Information Gain* dan klasifikasi menggunakan *KNN* dan *NWKNN*. Secara umum masukan dan keluaran dari sistem ini adalah:

1. Masukan yang digunakan berupa citra makanan.
2. Citra diproses dengan urutan *preprocessing*, ekstraksi fitur dengan metode *color moments* dan *GLCM*, seleksi fitur *Information Gain* dan klasifikasi menggunakan *KNN* dan *NWKNN*. Bahasa pemrograman yang digunakan dalam sistem ini adalah *python*.
3. Keluaran yang dihasilkan dari sistem ini adalah hasil kelas klasifikasi dari citra makanan. Hasil yang dikeluarkan oleh sistem akan dilakukan evaluasi dari algoritme.

### 3.6 Pengujian dan Analisis Algoritme

Pengujian dan analisis algoritme dilakukan untuk melihat seberapa bagus dari sistem yang telah dibuat. Analisis yang dilakukan harus sesuai dengan hasil pengujian yang ada. Pengujian yang dilakukan sebagai berikut.

1. Pengujian dan Analisis Pengaruh Jumlah Fitur pada *Information Gain*

2. Pengujian dan Analisis Pengaruh Nilai  $K$   $NKWN$
3. Pengujian dan Analisis Perhitungan Jarak  $NWKNN$
4. Pengujian dan Analisis Perbandingan Akurasi Metode  $KNN$  dan  $NWKNN$
5. Pengujian  $K$ -Fold Cross Validation

### 3.7 Teknik Pengumpulan Data

Data diambil langsung berupa citra makanan dengan bantuan smartphone yang dilakukan oleh Team Food Project FILKOM UB tahun 2018. Pengambilan data dilakukan pada pencahayaan pada pukul 09:00 sampai 12:00 dan diambil pada 3 kemiringan.



**Gambar 3.2 Citra dalam kondisi utuh pada 3 kemiringan**



**Gambar 3.3 Citra dalam kondisi dimakan ¼ pada 3 kemiringan**



**Gambar 3.4 Citra dalam kondisi dimakan ½ pada 3 kemiringan**








**Gambar 3.5 Citra dalam kondisi dimakan  $\frac{3}{4}$  pada 3 kemiringan**







### 3.8 Data Penelitian








Dataset penelitian menggunakan citra makanan dengan 23 jenis citra dapat dilihat pada Tabel 3.1.

**Tabel 3.1 Tabel dataset citra makanan**

No	Nama Makanan	Citra	Jumlah
1	Donat		34
2	Indomie Goreng		8
3	Mie Gepeng		8
4	Telur Dadar		8
5	Ayam		26



6	Rendang		35
7	Mentimun		35
8	Selada		8
9	Kemangi		8
10	Tomat		38
11	Strawbery		24

12	Pisang Kuning		41
13	Jeruk Orange		37
14	Jeruk Ijo Orange		26
15	Nasi Kuning		20
16	Nasi Merah		8
17	Oreo		39
18	Soba Mie		8

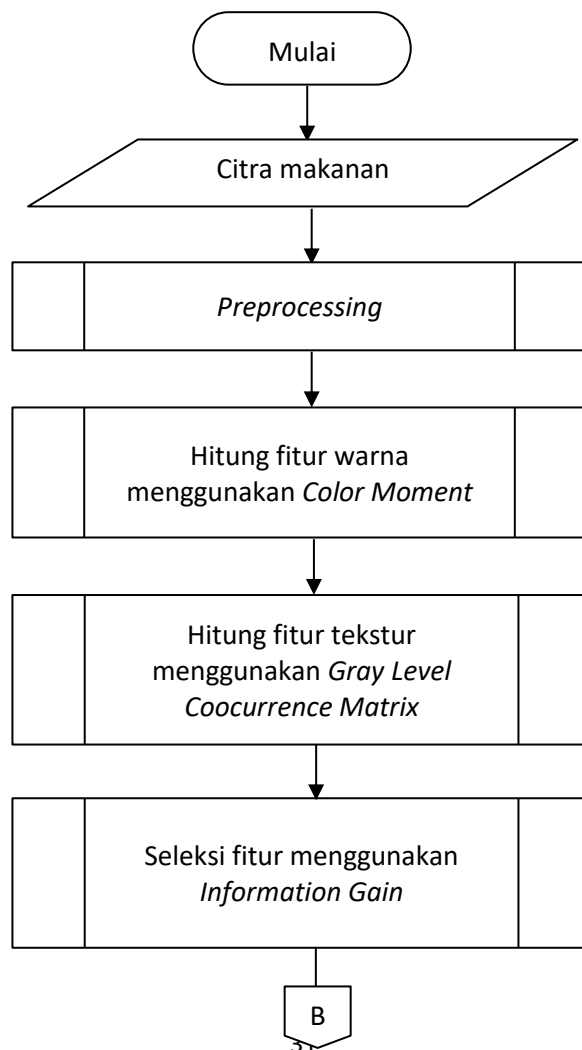
19	Biskuat		41
20	Milo Nugets		18
21	Genji Pie		27
22	Happy Tos		9
23	Gery Saluut		23
<b>Total</b>			529

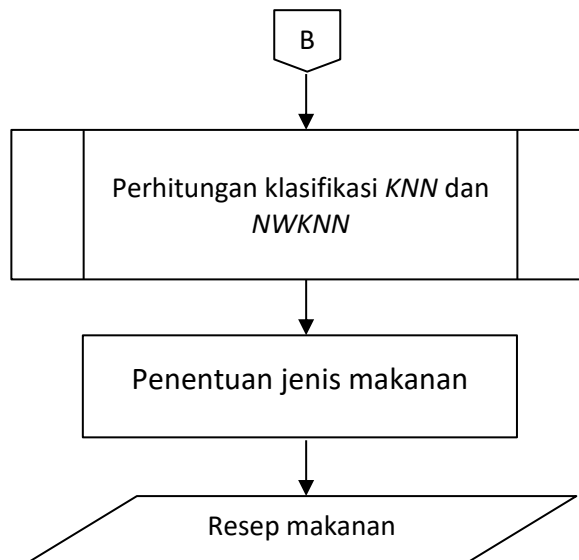
## BAB 4 PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan yang akan digunakan sebagai acuan untuk menerapkan seleksi fitur *Information Gain* pada klasifikasi citra smartphone menggunakan *Neighbor Weighted K-Nearest Neighbor (NWKNN)* dengan menggunakan pengukuran *Cosine Similarity*. Pada perancangan terdiri dari sub bab Perancangan Algoritme dan Perhitungan Manualisasi.

### 4.1 Perancangan Algoritme

Rancangan algoritme dimulai dengan memasukkan citra makanan. Dari hasil masukan citra makanan dilakukan preprocessing, dilakukan ekstraksi fitur warna *Color Moments* menghasilkan 3 fitur dan ekstraksi fitur tekstur *Gray Level Cooccurrence Matrix* menghasilkan 13 fitur. Kemudian hasil dari fitur-fitur tersebut akan dilakukan seleksi fitur menggunakan *Information Gain* untuk mengurangi dimensi atribut. Setelah itu fitur yang didapatkan dilakukan klasifikasi menggunakan NWKNN dan menghasilkan keluaran kelas dari citra. Rancangan algoritme secara keseluruhan dapat dilihat pada Gambar 4.1.

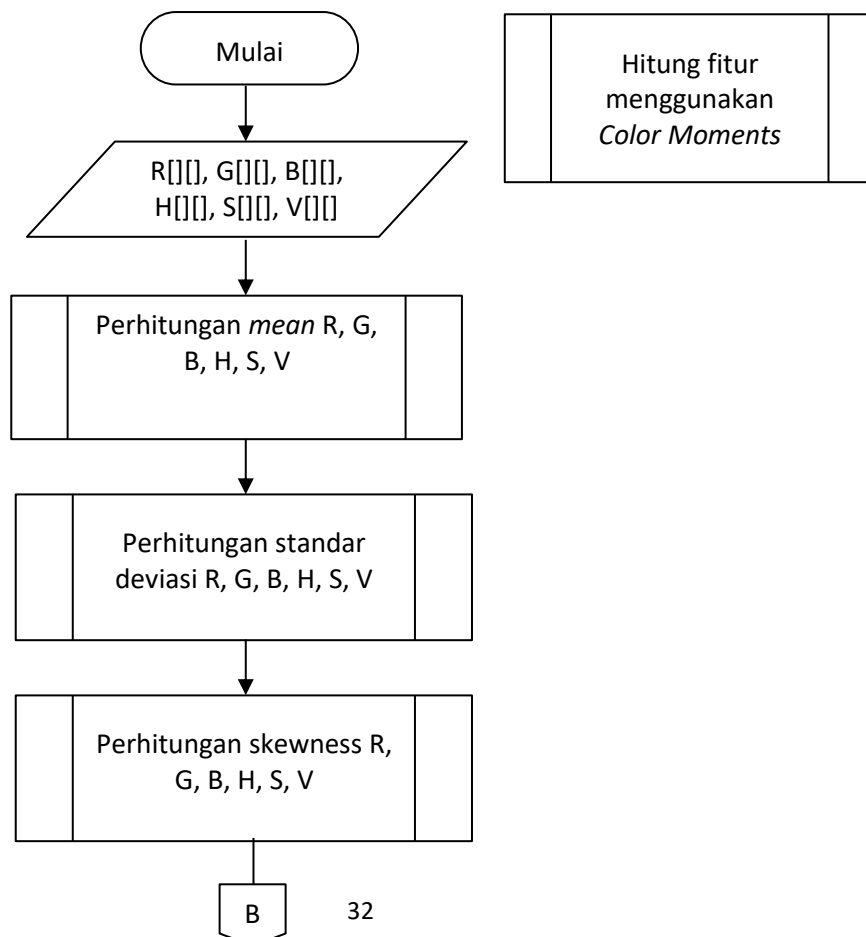


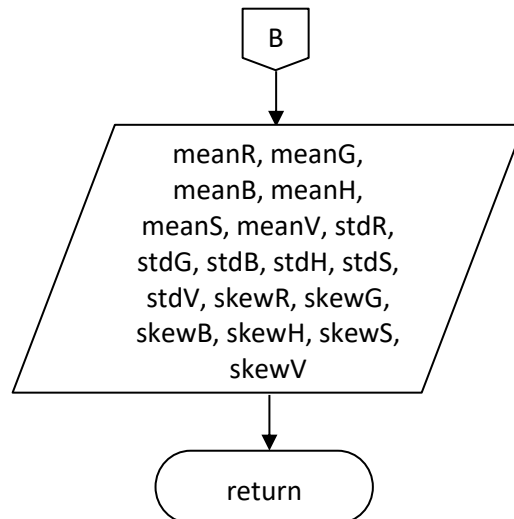


**Gambar 4.1 Diagram alir sistem**

#### 4.1.1 Perancangan Algoritme *Color Moments*

Perancangan algoritme *Color Moments* dengan masukan berupa citra dari warna R,G,B,H,S dan V akan dihitung fitur *mean*, standar deviasi dan *skewness*. Berikut diagram alir algoritme ekstraksi fitur *Color Moments* ditunjukkan pada Gambar 4.2.

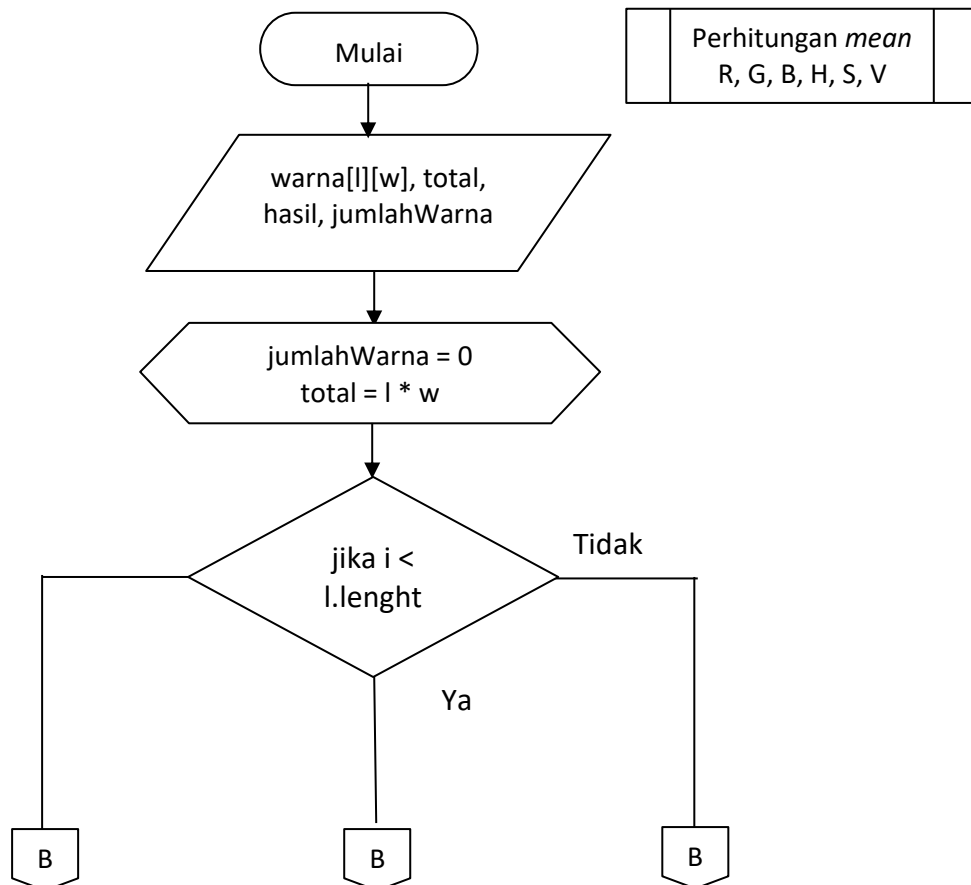


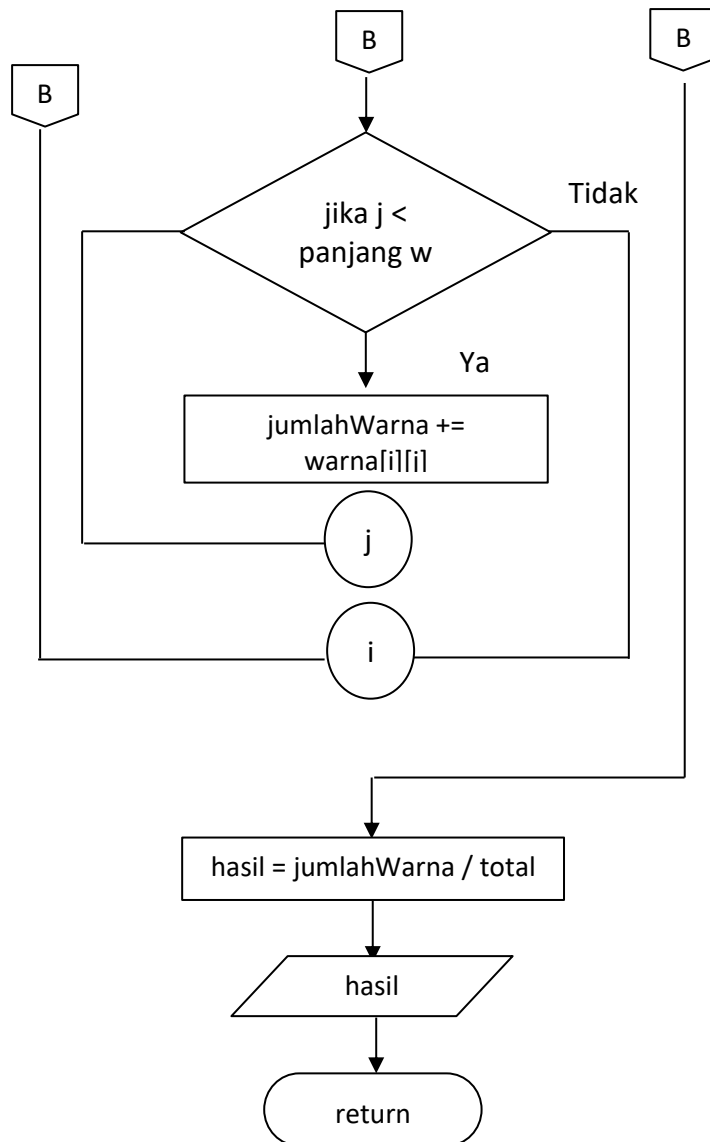


**Gambar 4.2 Diagram alir *Color Moments***

#### 4.1.1.1 Perancangan Algoritme *Color Moments* Perhitungan *Mean*

Perhitungan fitur pertama yang digunakan pada algoritme *Color Moments* adalah *mean* yaitu menghitung rata-rata dari setiap citra makanan. Berikut diagram alir perhitungan *mean* ditunjukkan pada Gambar 4.3.

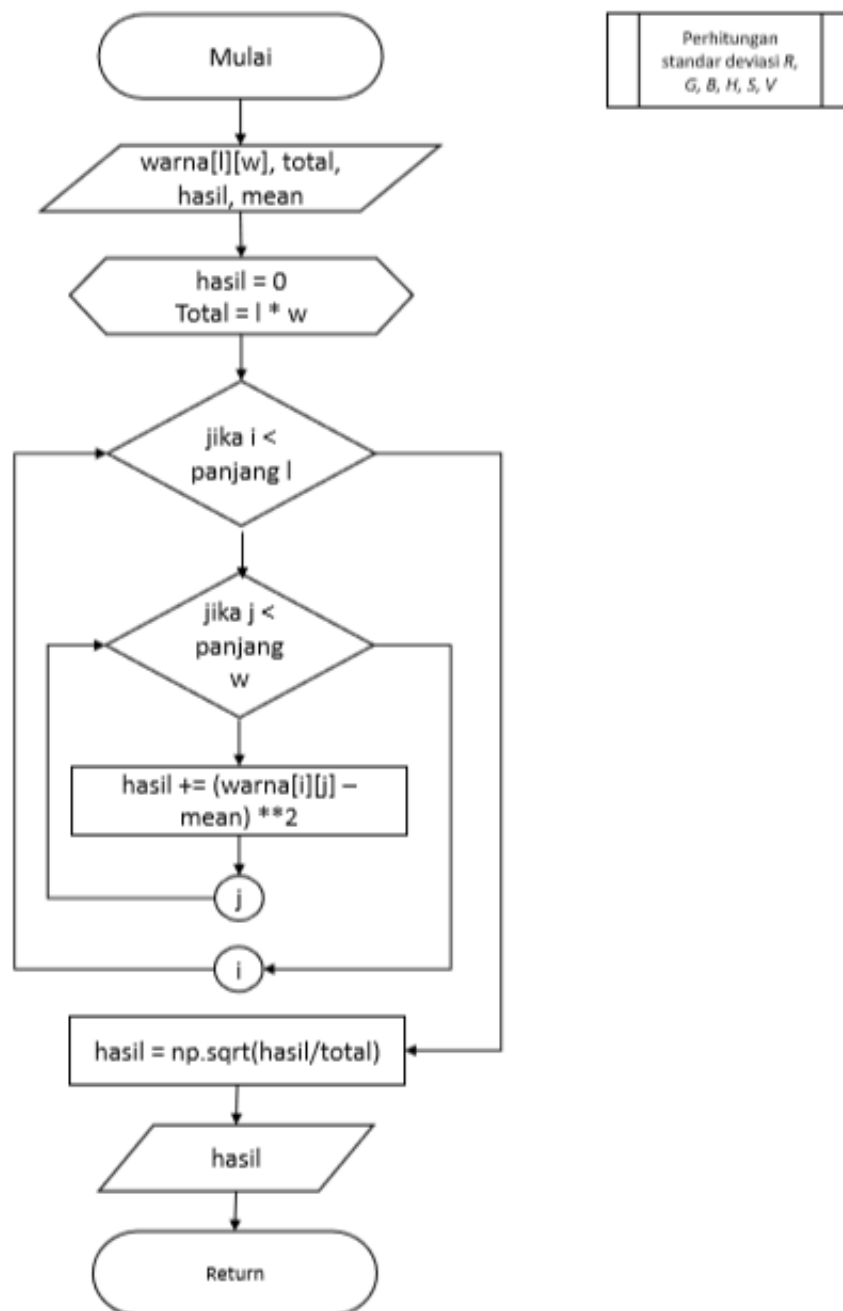




**Gambar 4.3 Diagram alir perhitungan *Mean***

#### **4.1.1.2 Perancangan Algoritme *Color Moments* Perhitungan *Standar Deviasi***

Perhitungan fitur kedua yang digunakan pada algoritme *Color Moments* adalah *Standar Deviasi*. Berikut diagram alir perhitungan *mean* ditunjukkan pada Gambar 4.4.

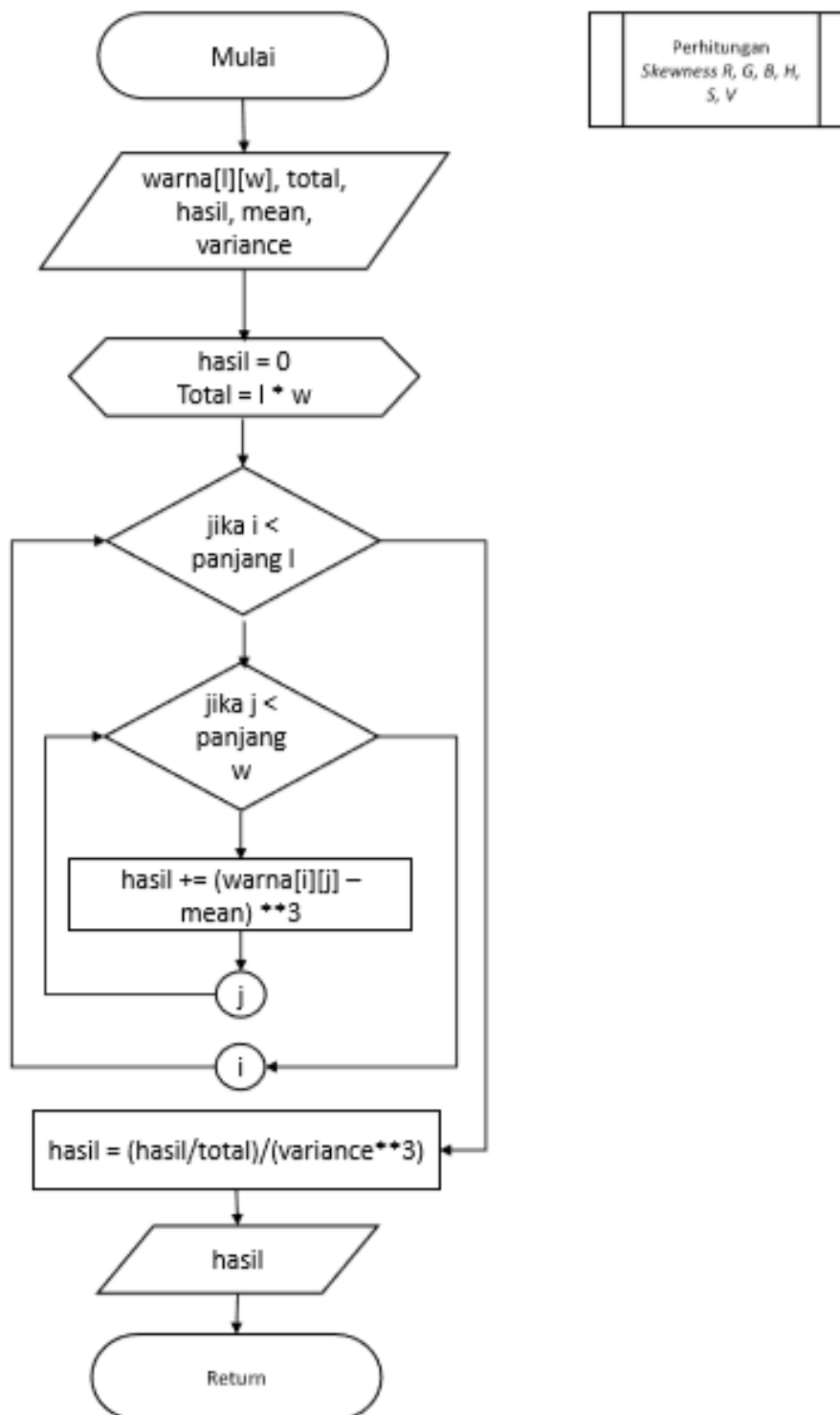


**Gambar 4.4 Diagram alir perhitungan *Standar Deviasi***

#### 4.1.1.3 Perancangan Algoritme *Color Moments* Perhitungan *Skewness*

Perhitungan fitur ketiga yang digunakan pada algoritme *Color Moments* adalah *Skewness*. Berikut diagram alir perhitungan *mean* ditunjukkan pada Gambar 4.5.





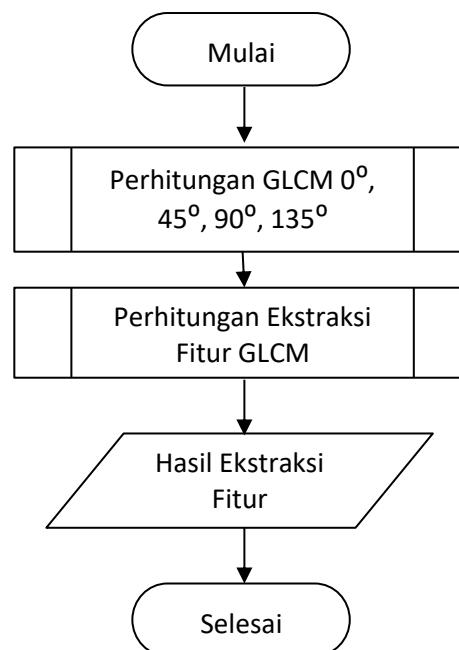
**Gambar 4.5 Diagram alir perhitungan *Skewness***

#### **4.1.2 Perancangan Algoritme *Gray Level Cooccurrence Matrix***

Perancangan algoritme *GLCM* dilakukan setelah *preprocessing* citra makanan yang tersegmentasi. Ubah citra berwarna menjadi grayscale. Perhitungan pertama setelah *preprocessing* yaitu menghitung matriks awal yang

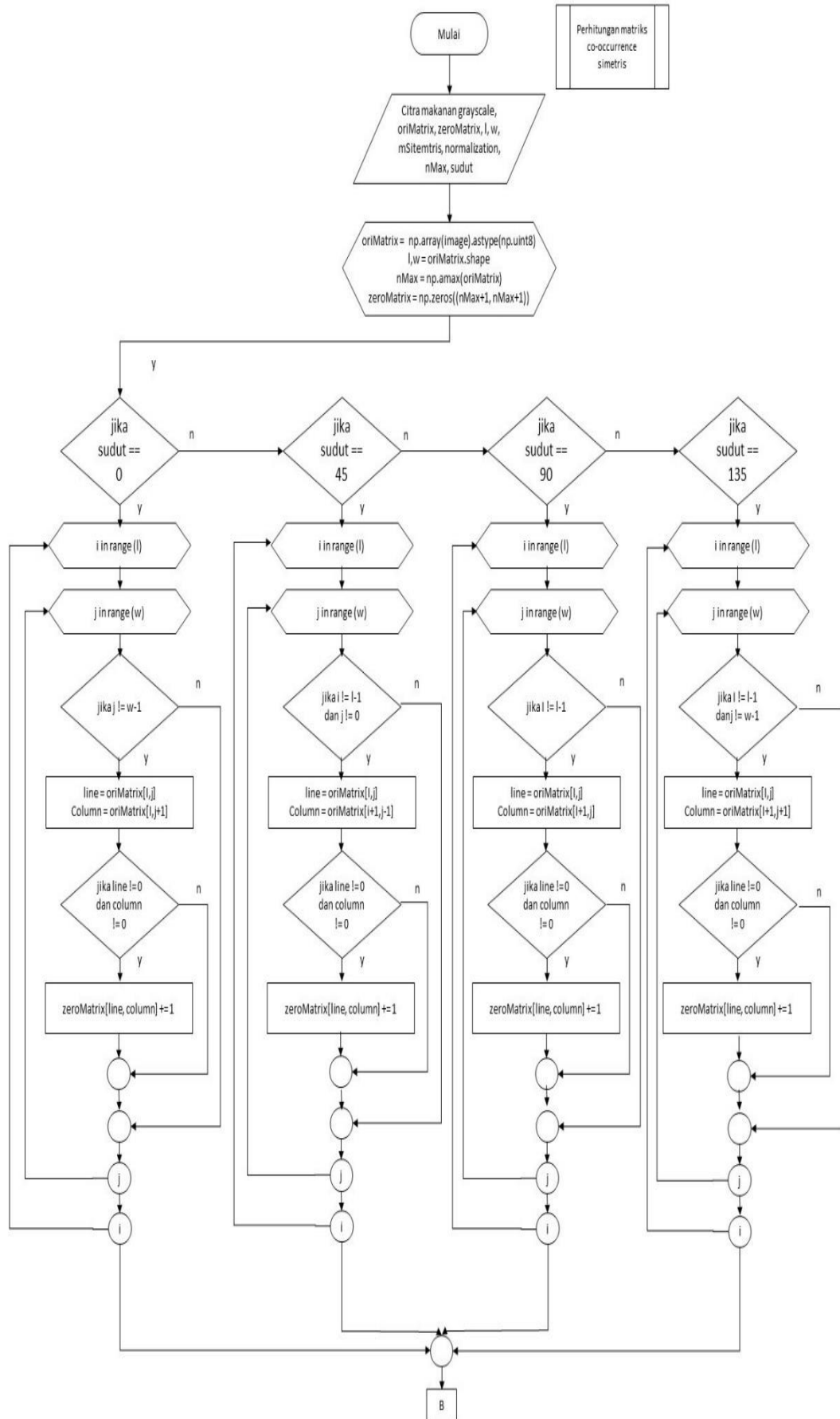
dilakukan dengan sudut 0, 45, 90, 135. Setelah didapatkan 4 sudut dari matriks awal lakukan perhitungan penjumlahan antara matriks awal dengan matriks transpose agar mendapatkan matriks baru yaitu matriks *co-occurrence* simetris. Hasil dari matriks simetris dilakukan normalisasi dengan membagi nilai matriks dengan total keseluruhan nilai matriks. Dari hasil matriks yang telah di normalisasi, dilakukan perhitungan ekstraksi fitur sebanyak 13 fitur., yaitu Angular Second Moment (ASM), Contrast, Correlation, Sum of Squares: Variance, Inverse Difference Moment (IDM), Sum Average, Sum Variance, Sum Entropy, Entropy, Difference Variance, Difference Entropy, Information Measure of Correlation 1 dan Information Measure of Correlation 2. Berikut diagram alir dapat ditunjukkan pada Gambar 4.6.

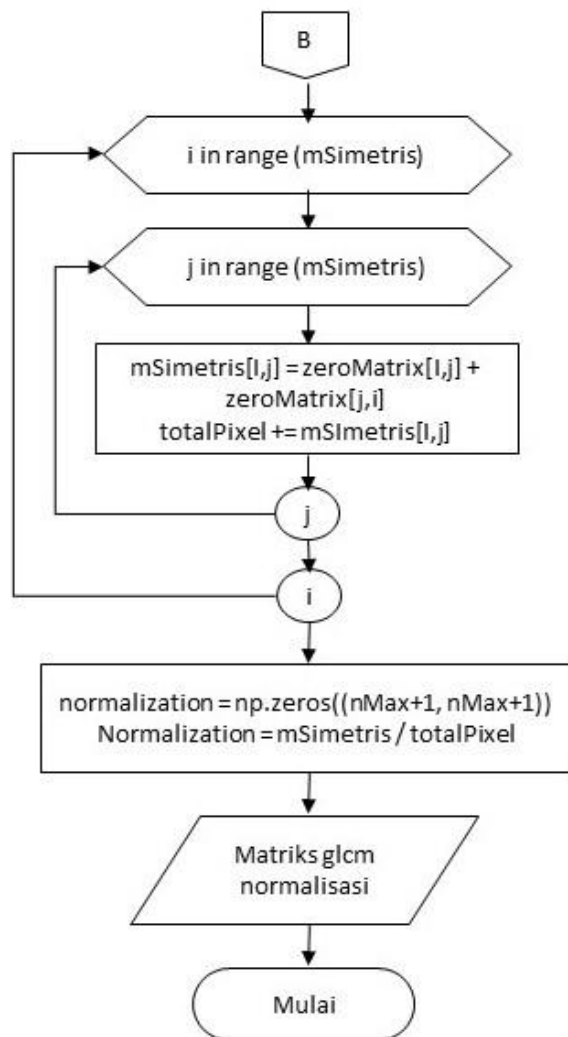
**Gambar 4.6 Diagram alir GLCM**



#### **4.1.2.1 Perancangan Algoritme GLCM Perhitungan Matriks *Co-occurrence***

Proses algoritme perhitungan matriks *cooccurrence* diawali dengan menerima masukan citra makanan berupa warna grayscale yang telah tersegmentasi. Buat matriks *cooccurrence* dengan baris dan kolom sepanjang nilai maksimal dari citra masukan. Hitung masing-masing sudut dari citra dan hitung jumlahnya ke dalam matriks *co-occurrence*. Hitung perkalian matriks antara matriks *co-occurrence* dengan matriks transposenya untuk mendapatkan matriks simetris. Lakukan normalisasi matriks simetris dengan cara setiap nilai dalam piksel dibagi dengan jumlah keseluruhan nilai piksel. Berikut diagram alir proses perhitungan matriks *co-occurrence* ditunjukkan pada Gambar 4.7.

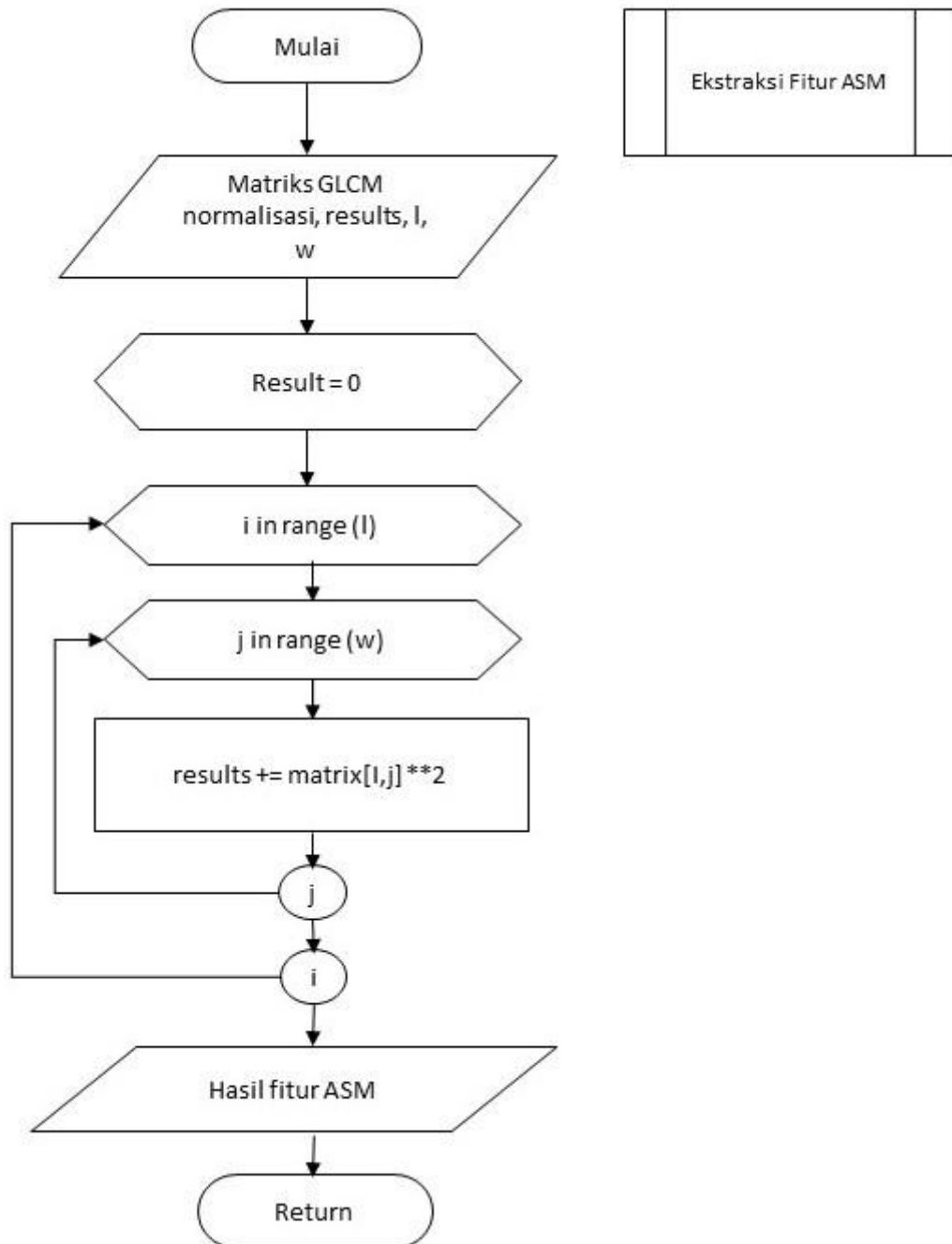




**Gambar 4.7 Diagram alir *Matrix Co-occurrence***

#### 4.1.2.2 Ekstraksi Fitur Angular Second Moments (ASM)

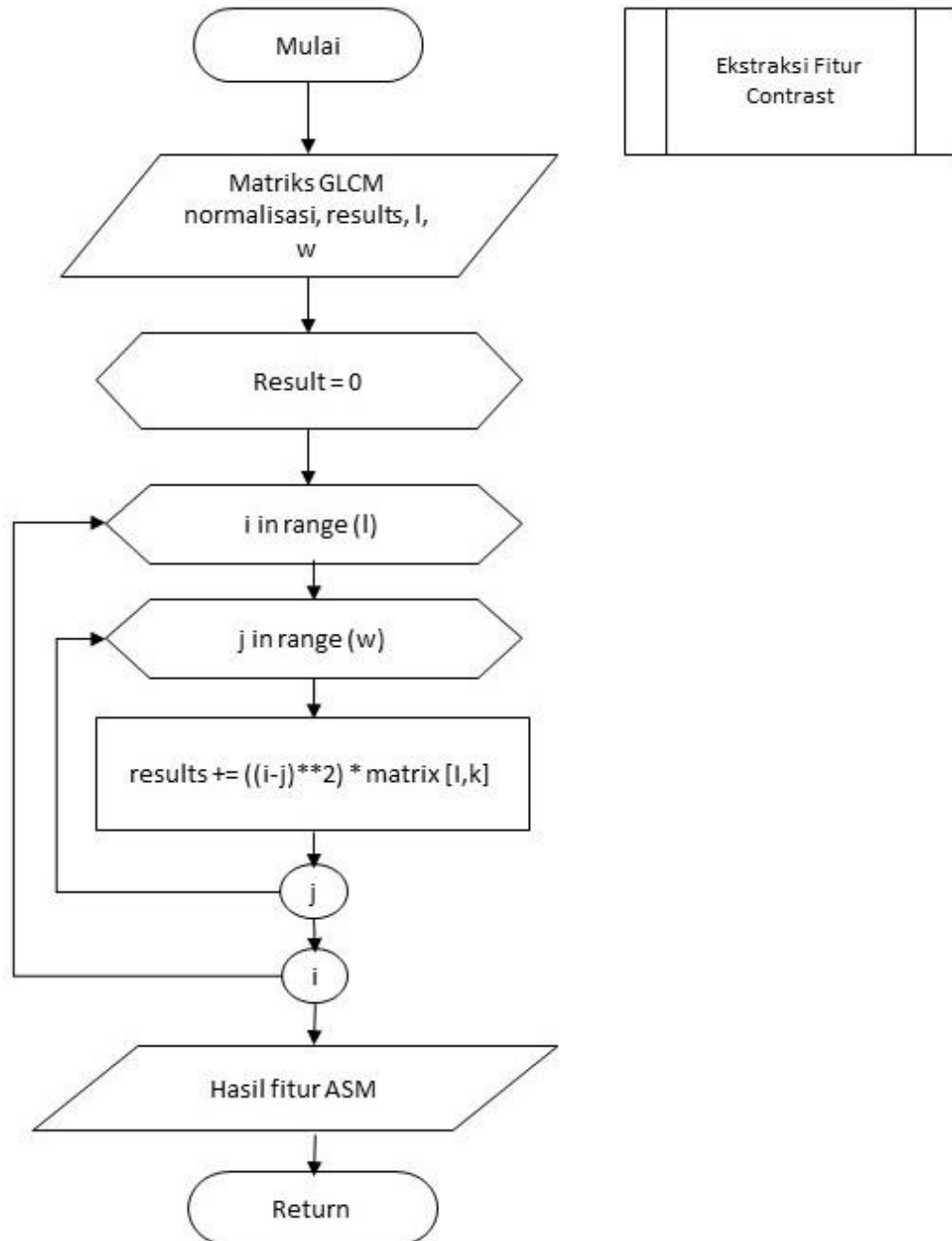
Berikut diagram alir ekstraksi fitur GLCM dengan fitur ASM ditunjukkan pada Gambar 4.8.



Gambar 4.8 Diagram alir ekstraksi fitur ASM

#### 4.1.2.3 Ekstraksi Fitur Contrast

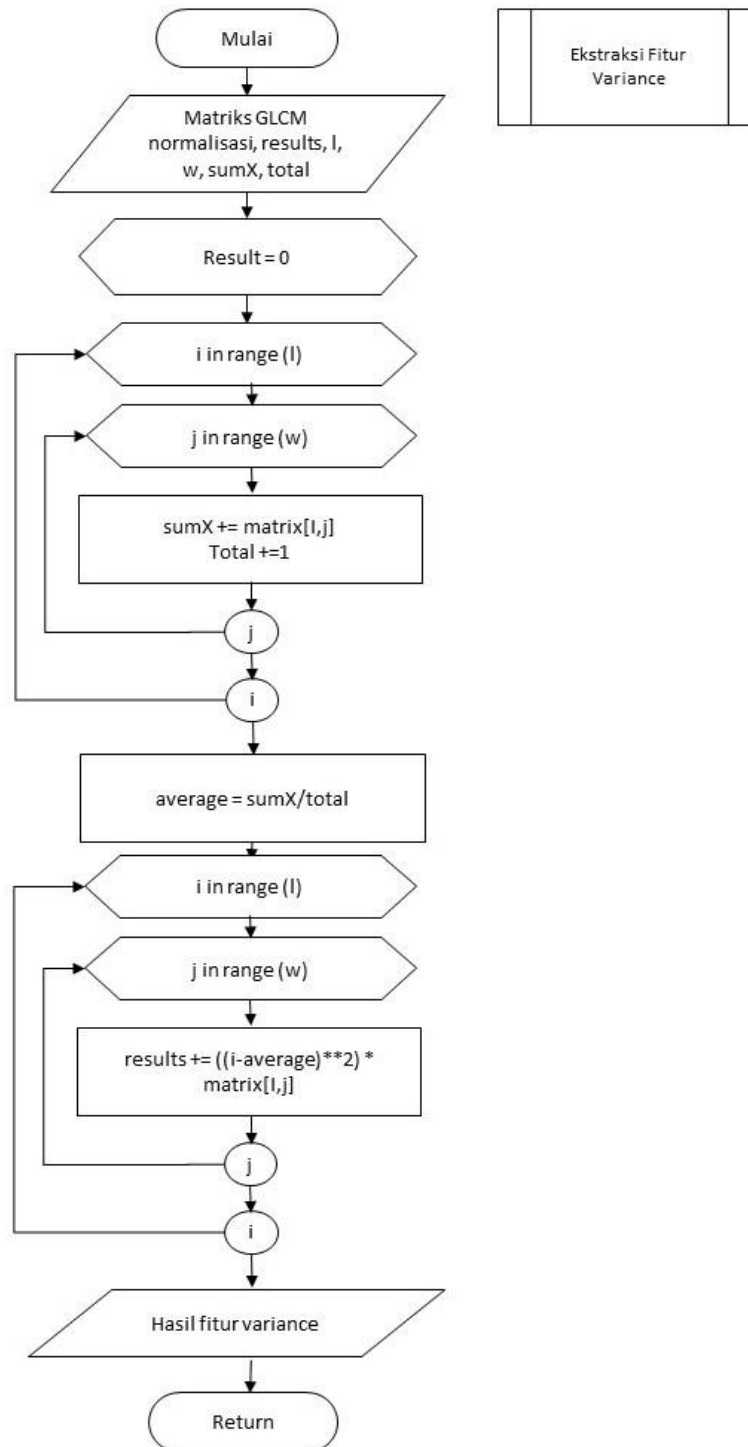
Berikut diagram alir ekstraksi fitur GLCM dengan fitur ASM ditunjukkan pada Gambar 4.9.



Gambar 4.9 Diagram alir ekstraksi fitur Contrast

#### 4.1.2.4 Ekstraksi Fitur *Variance*

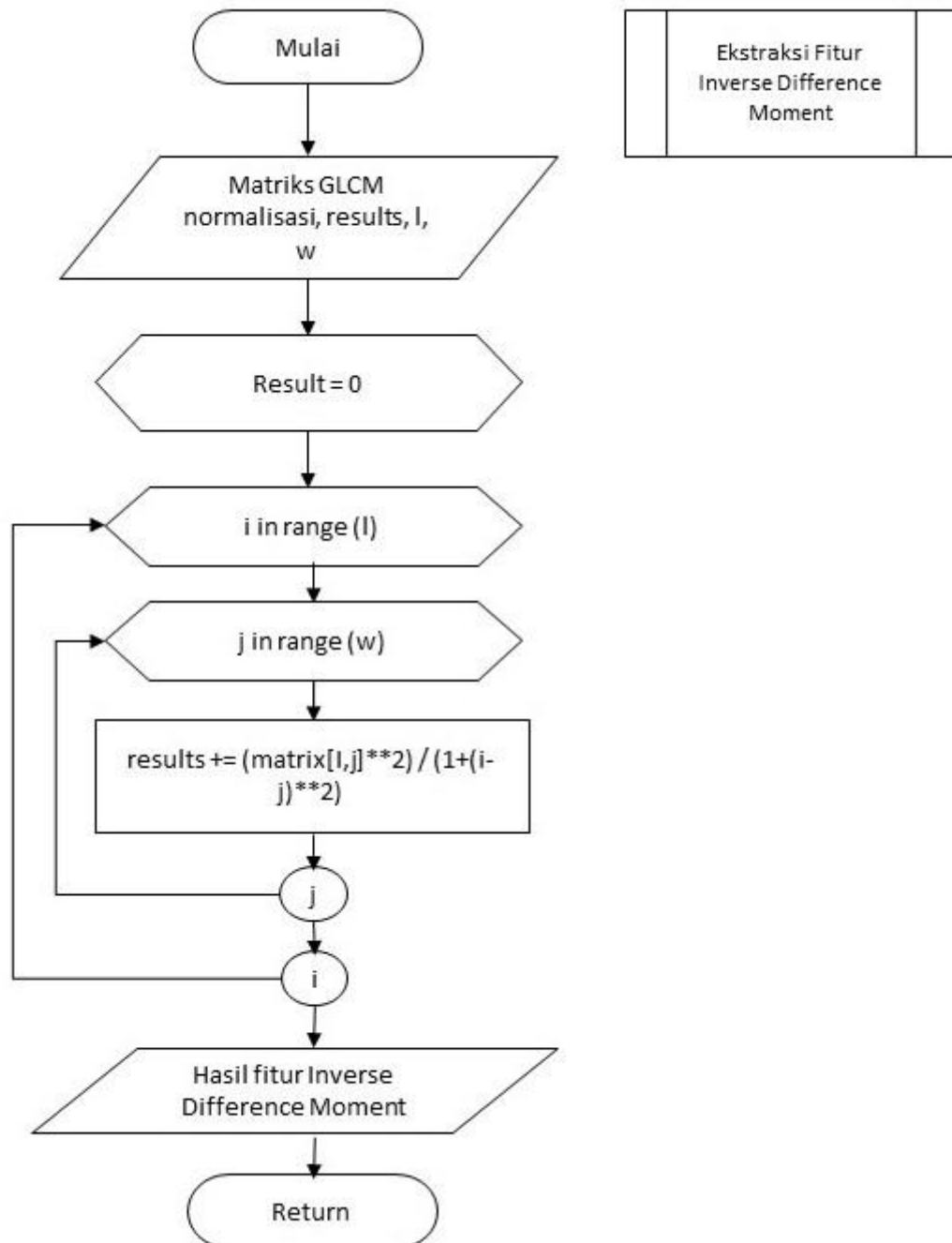
Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Variance* ditunjukkan pada Gambar 4.10.



Gambar 4.10 Diagram alir ekstraksi fitur *Variance*

#### 4.1.2.5 Ekstraksi Fitur *Inverse Difference Moment (IDM)*

Berikut diagram alir ekstraksi fitur GLCM dengan fitur *IDM* ditunjukkan pada Gambar 4.11.

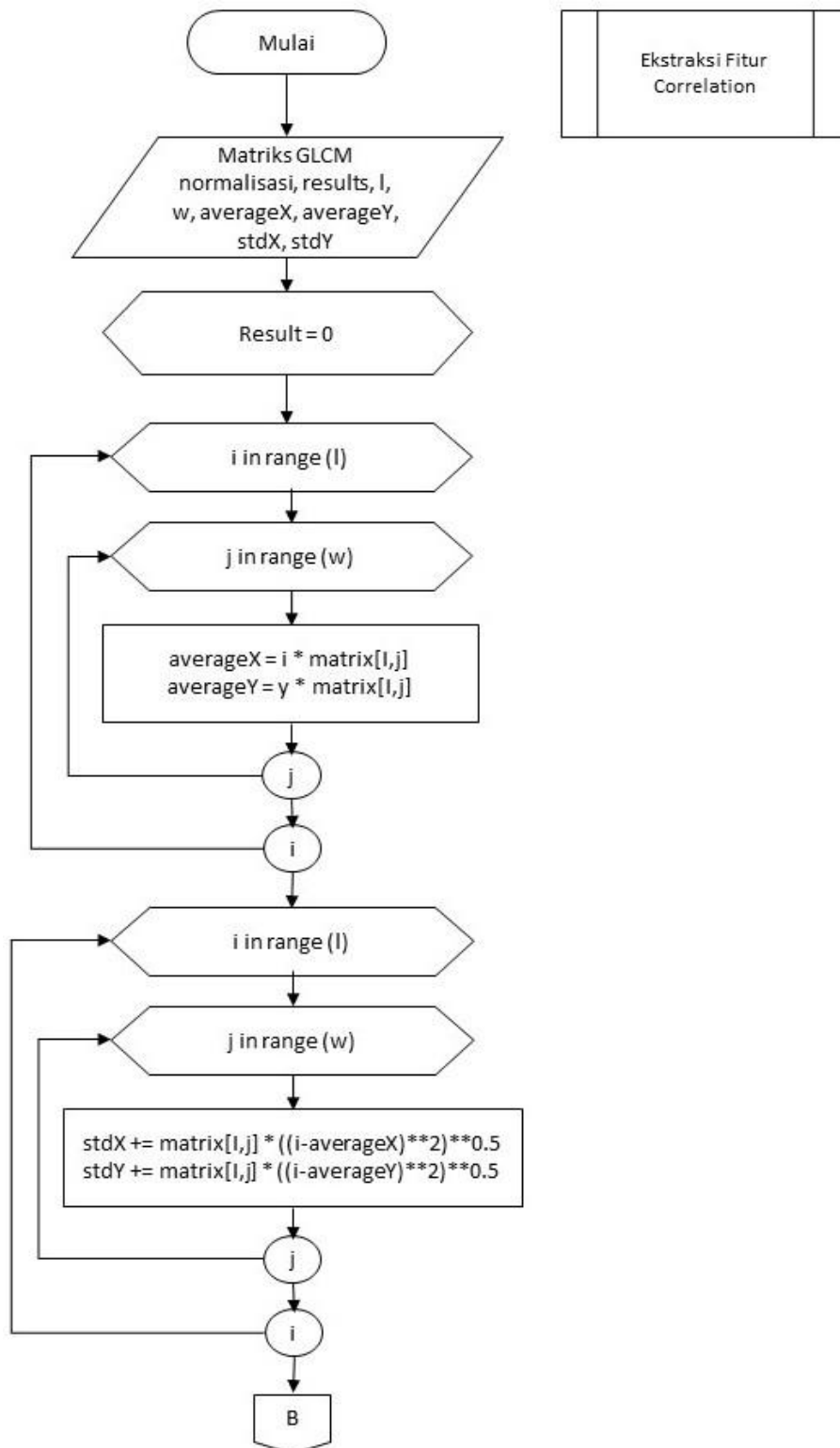


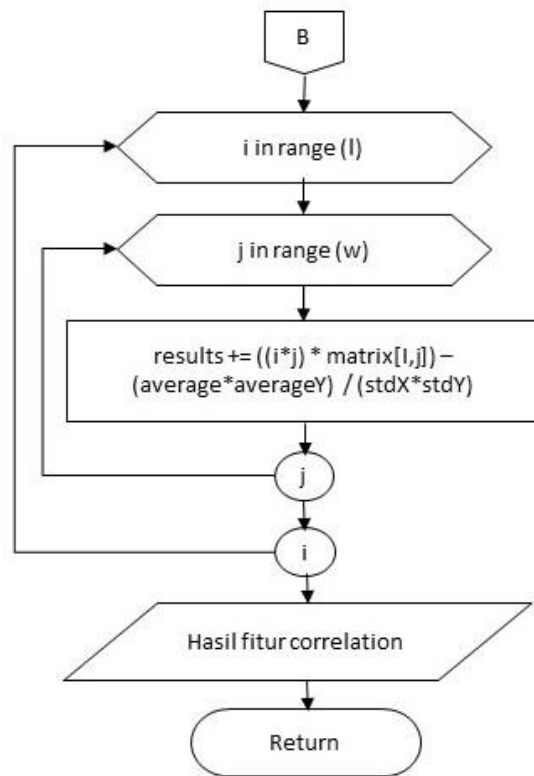
Gambar 4.11 Diagram alir ekstraksi fitur *Inverse Difference Moment (IDM)*



#### 4.1.2.6 Ekstraksi Fitur *Correlation*

Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Correlation* ditunjukkan pada Gambar 4.12.

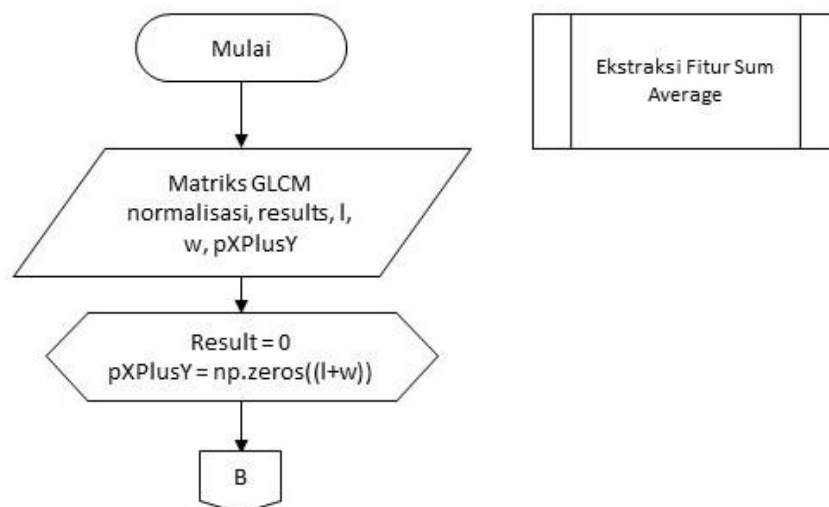


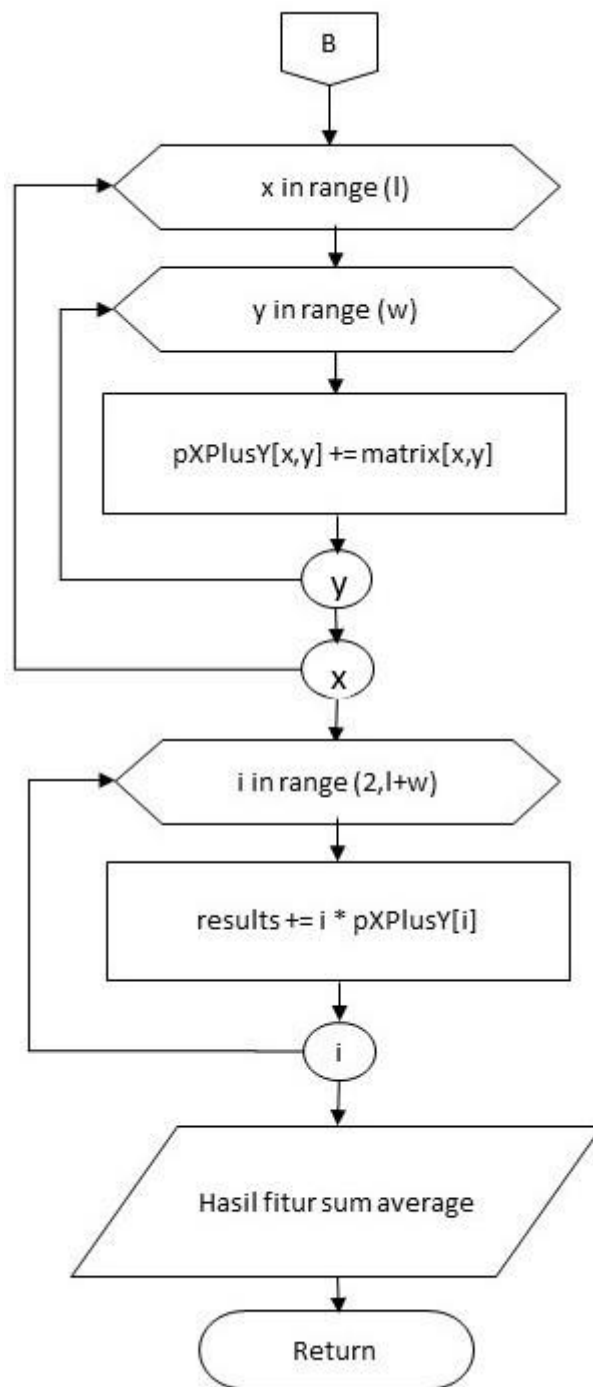


**Gambar 4.12** Diagram alir ekstraksi fitur *Correlation*

#### 4.1.2.7 Ekstraksi Fitur *Sum Average*

Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Sum Average* ditunjukkan pada Gambar 4.13.

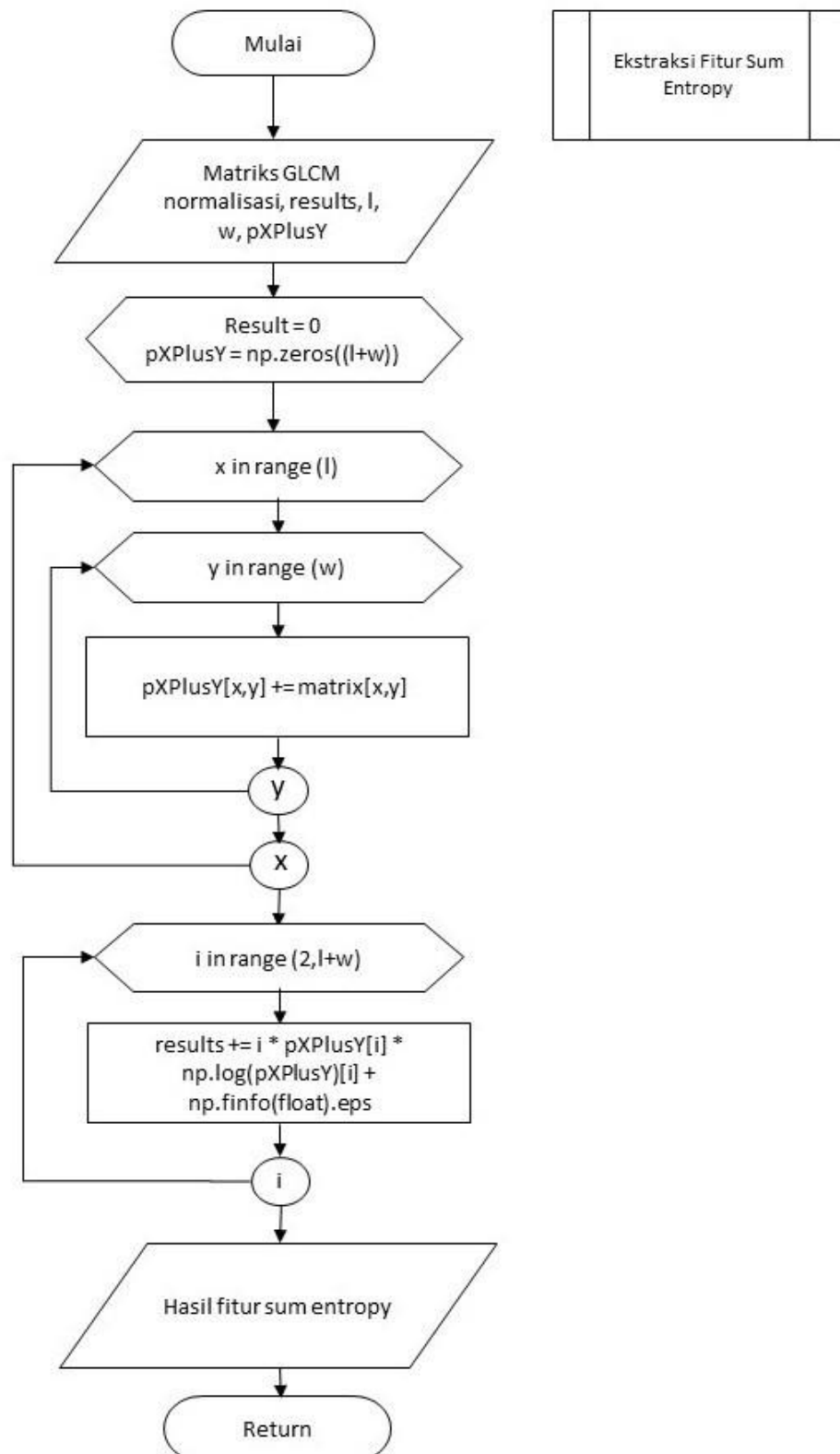




**Gambar 4.13** Diagram alir ekstraksi fitur *Sum Average*

#### **4.1.2.8 Ekstraksi Fitur *Sum Entropy***

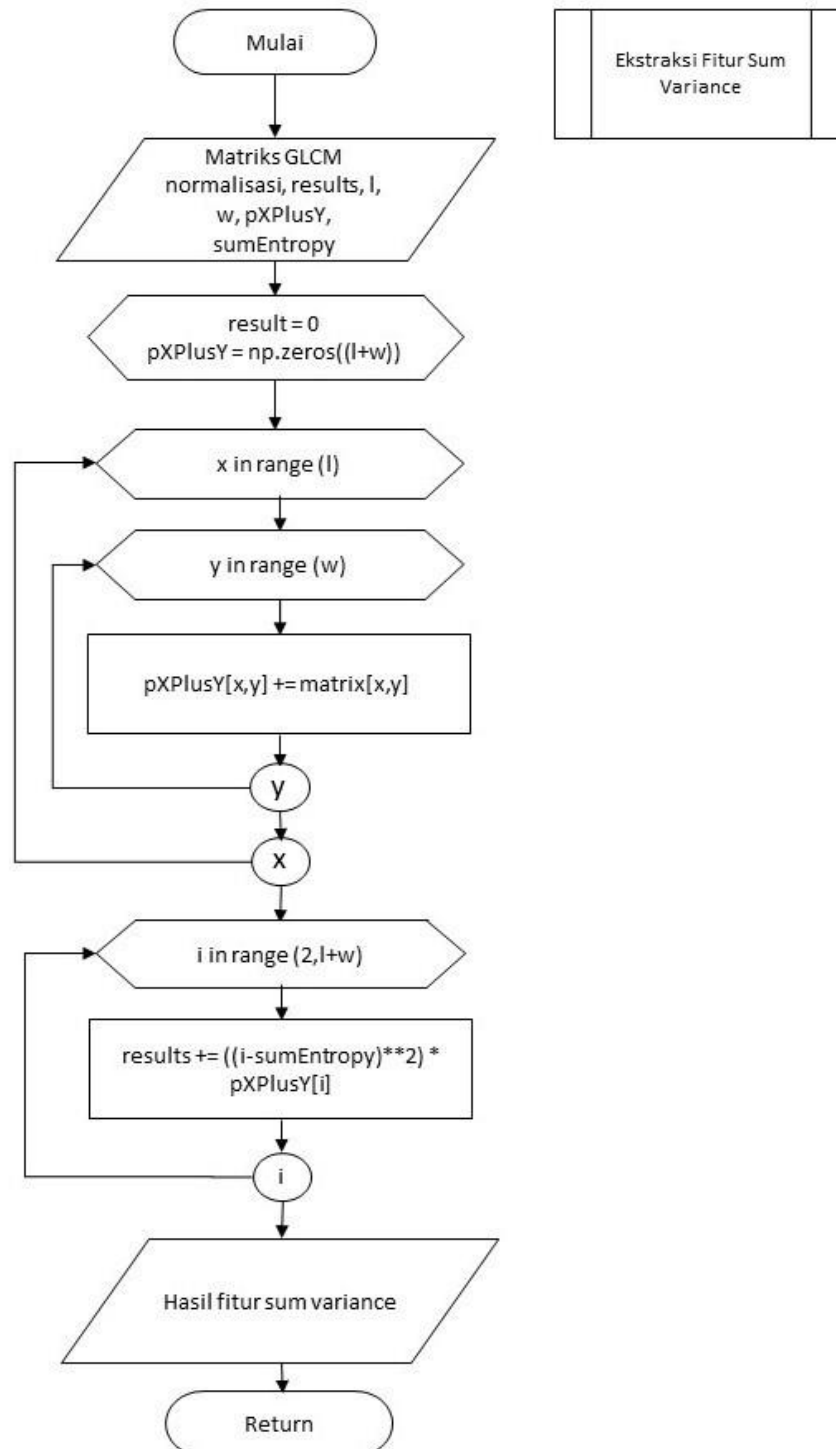
Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Sum Average* ditunjukkan pada Gambar 4.14.



**Gambar 4.14 Diagram alir ekstraksi fitur *Sum Entropy***

#### 4.1.2.9 Ekstraksi Fitur Sum Variance

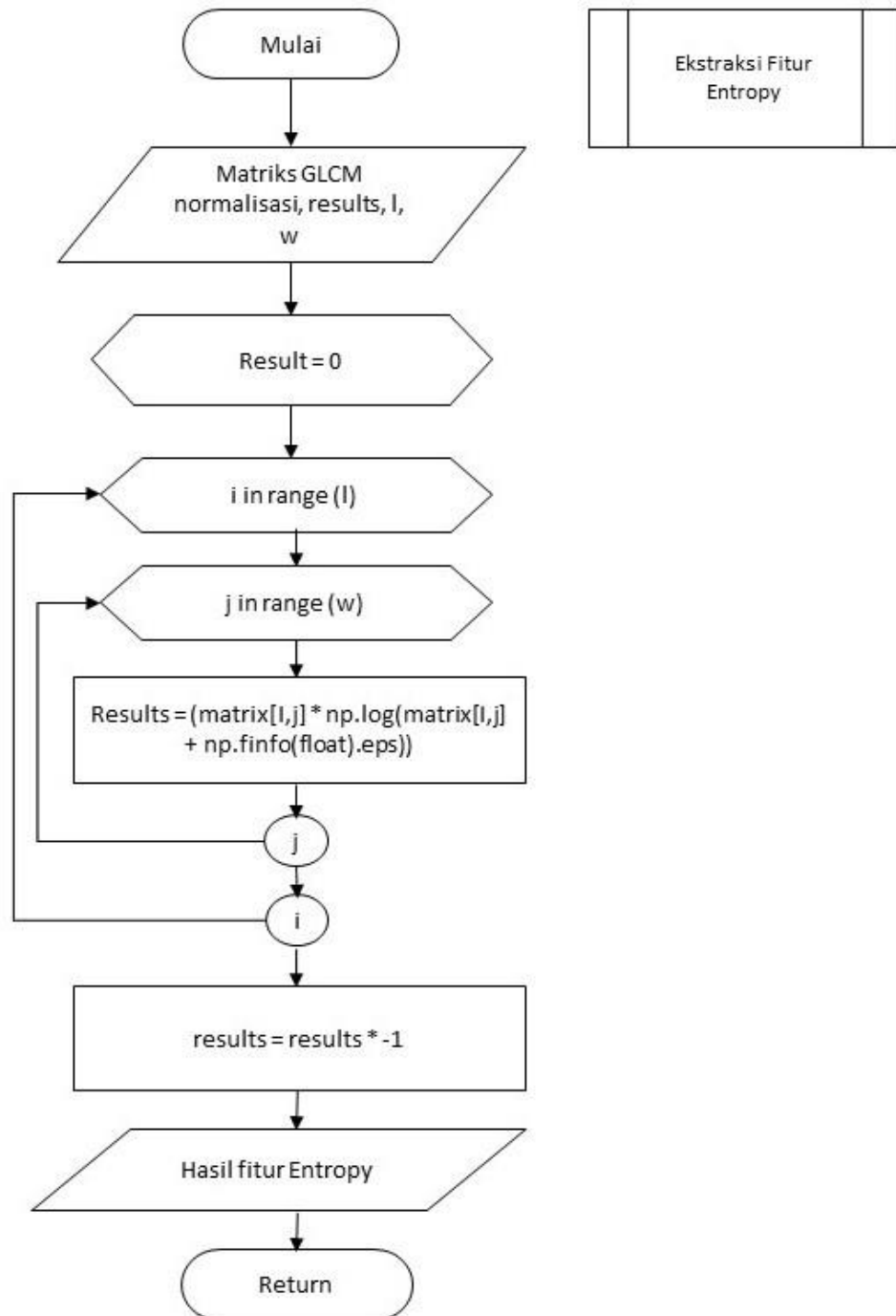
Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Sum Variance* ditunjukkan pada Gambar 4.15.



Gambar 4.15 Diagram alir ekstraksi fitur *Sum Variance*

#### 4.1.2.10 Ekstraksi Fitur Entropy

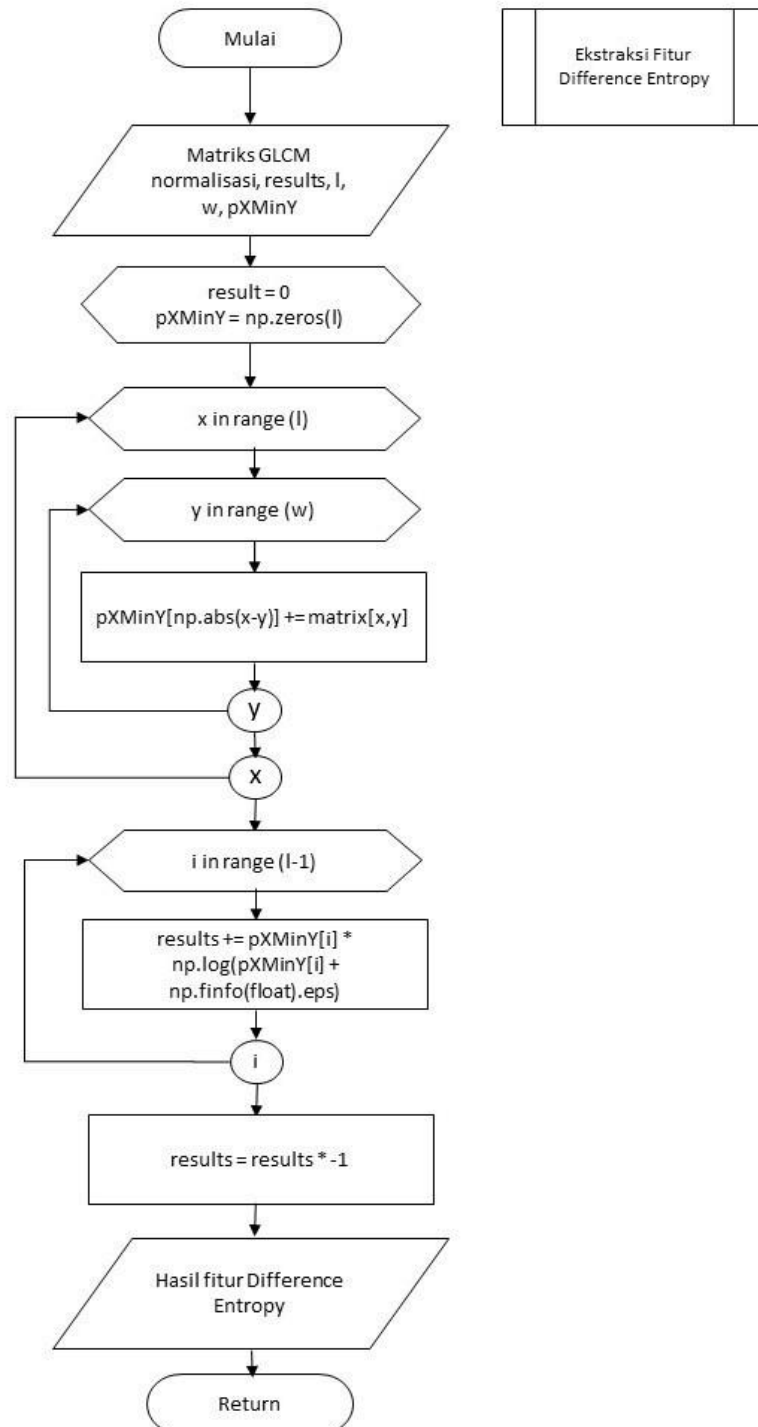
Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Entropy* ditunjukkan pada Gambar 4.16.



Gambar 4.16 Diagram alir ekstraksi Fitur *Entropy*

#### 4.1.2.11 Ekstraksi Fitur *Difference Entropy*

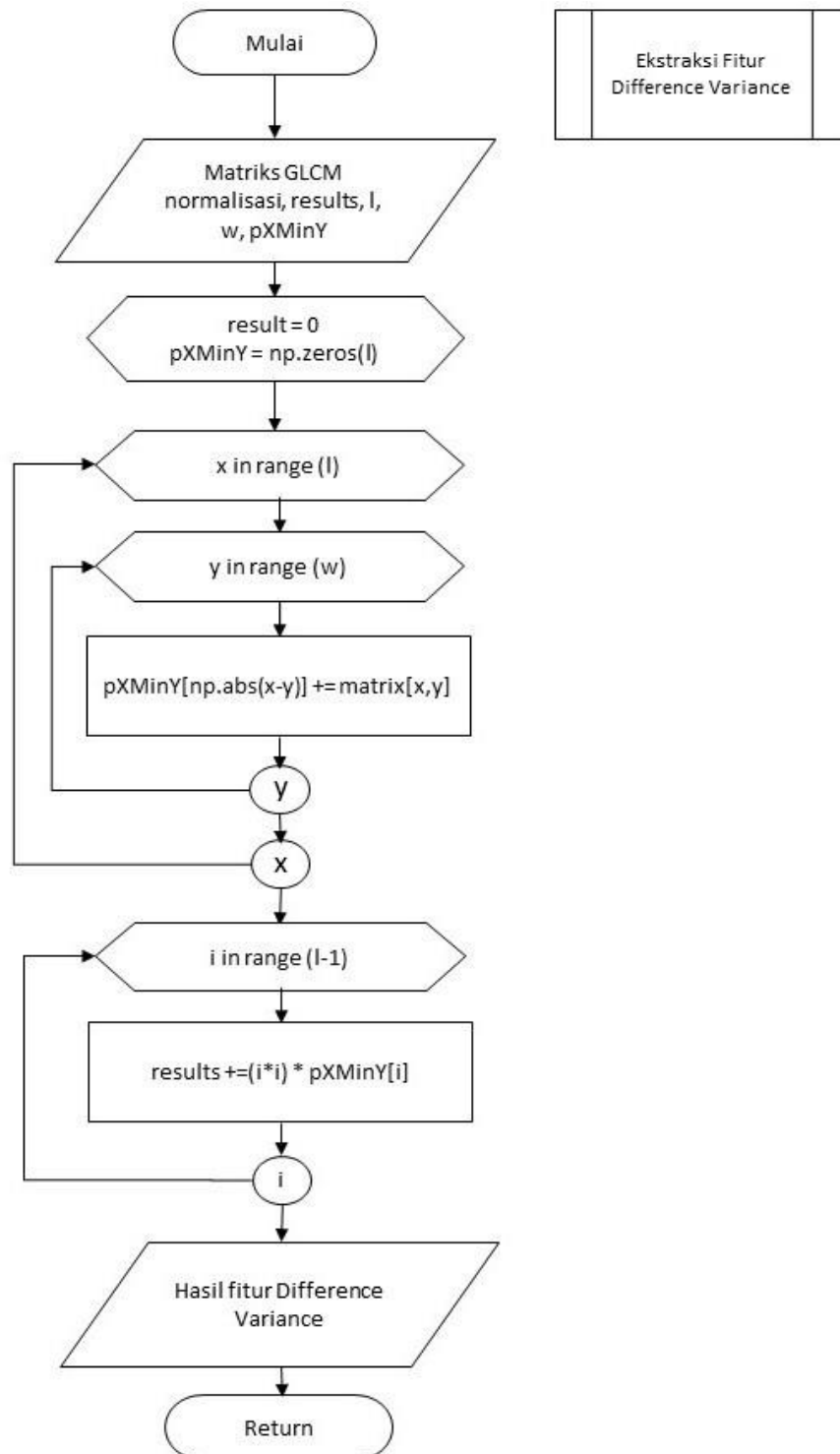
Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Difference Entropy* ditunjukkan pada Gambar 4.17.



Gambar 4.17 Diagram alir ekstraksi fitur *Difference Entropy*

#### 4.1.2.12 Ekstraksi Fitur *Difference Variance*

Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Difference Variance* ditunjukkan pada Gambar 4.18.

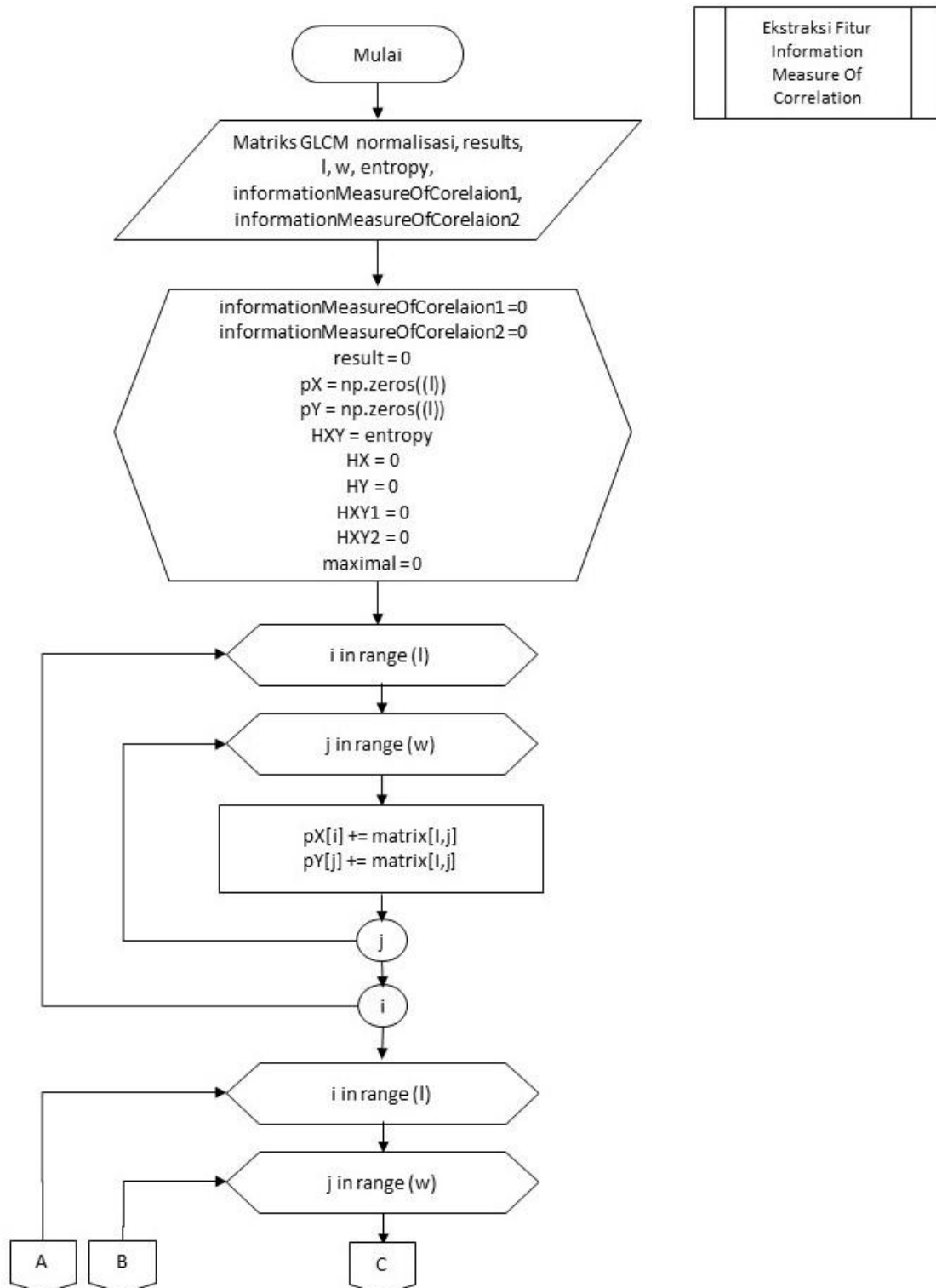


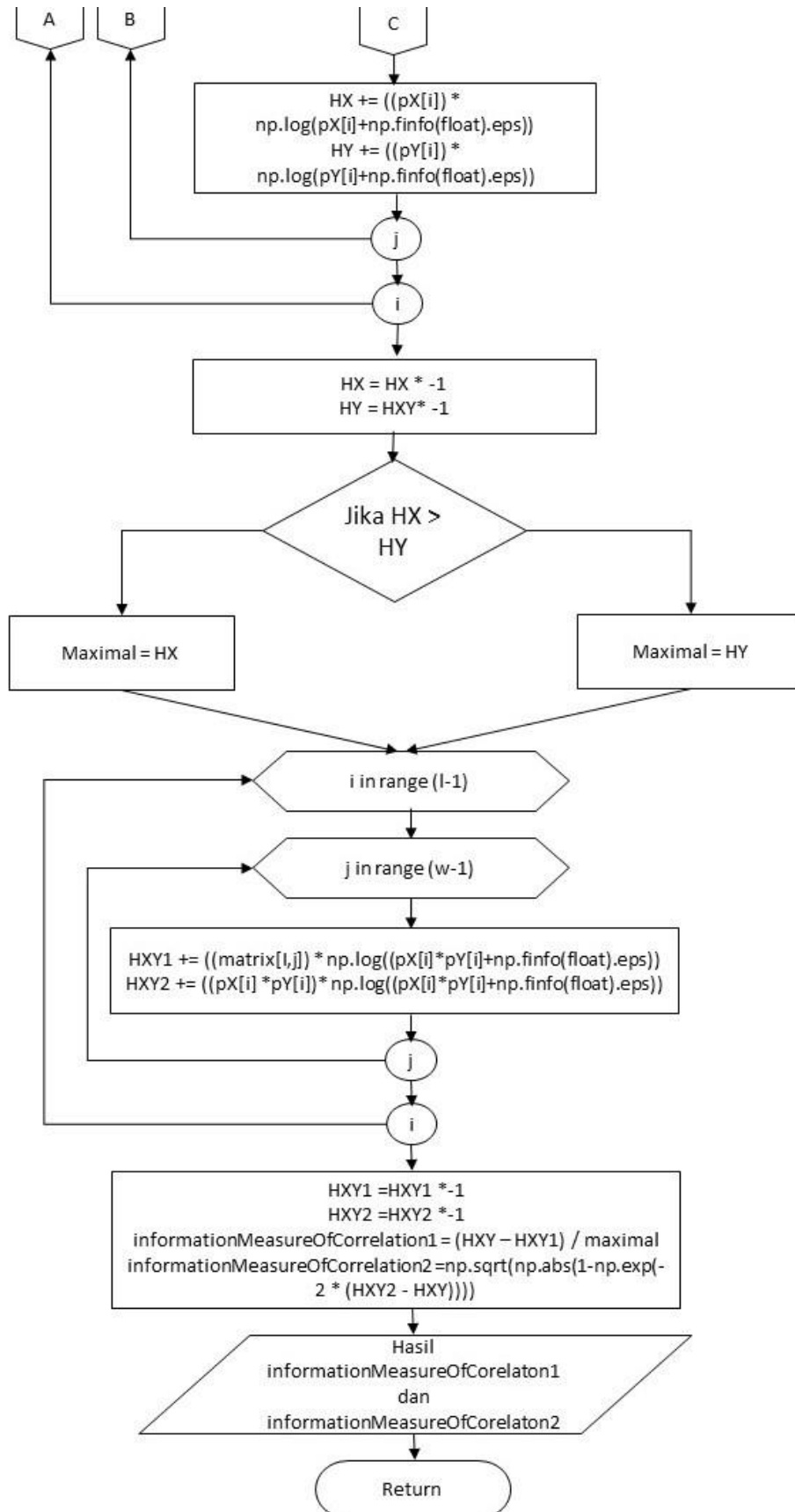
Gambar 4.18 Diagram alir ekstraksi fitur *Difference Variance*



#### 4.1.2.13 Ekstraksi Fitur Information Measure Of Correlation

Berikut diagram alir ekstraksi fitur GLCM dengan fitur *Information Measure Of Correlation 1* dan *Information Measure Of Correlation 2* ditunjukkan pada Gambar 4.19.

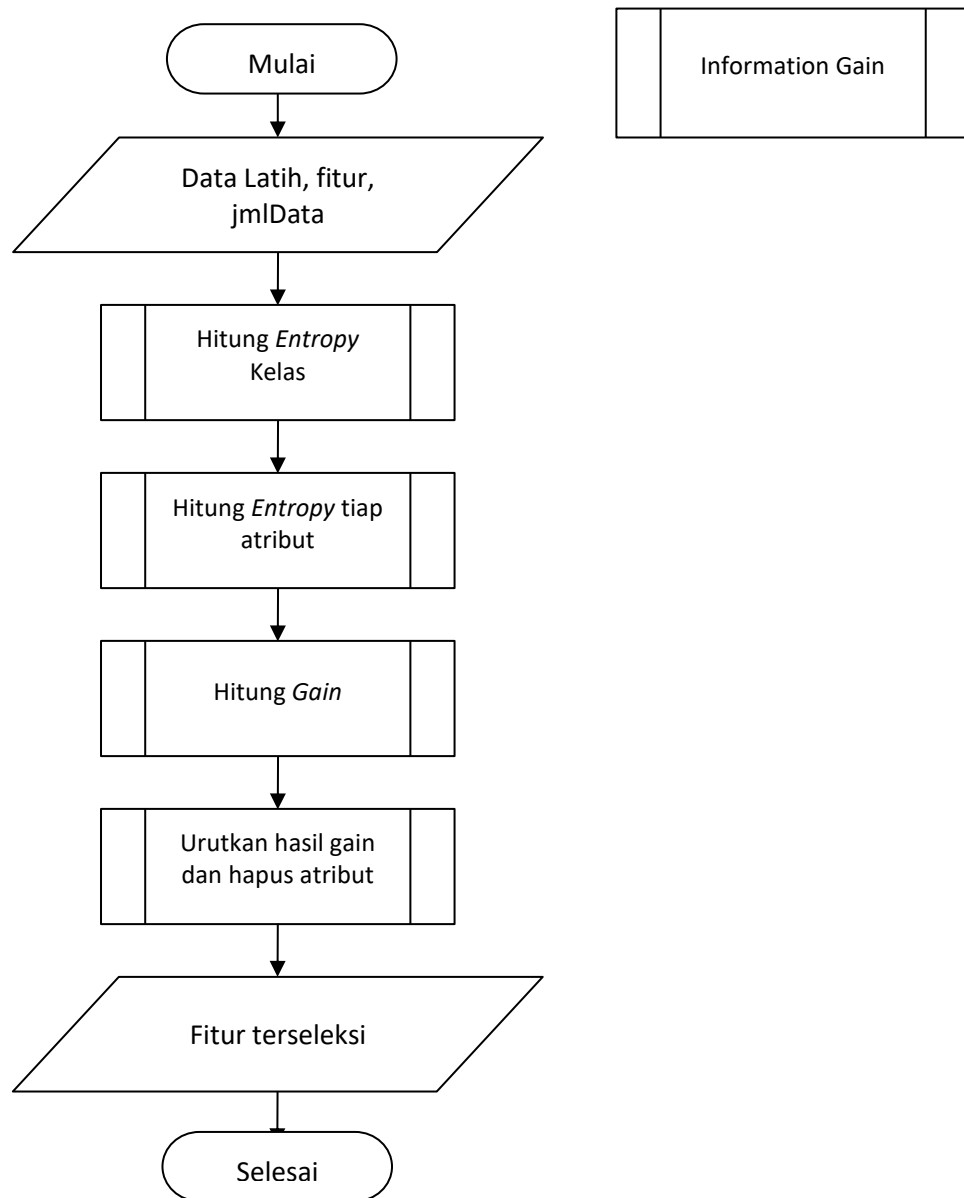




**Gambar 4.19** Diagram alir ekstraksi fitur *Information Measure Of Correlation*

#### 4.1.3 Perancangan Algoritme *Information Gain*

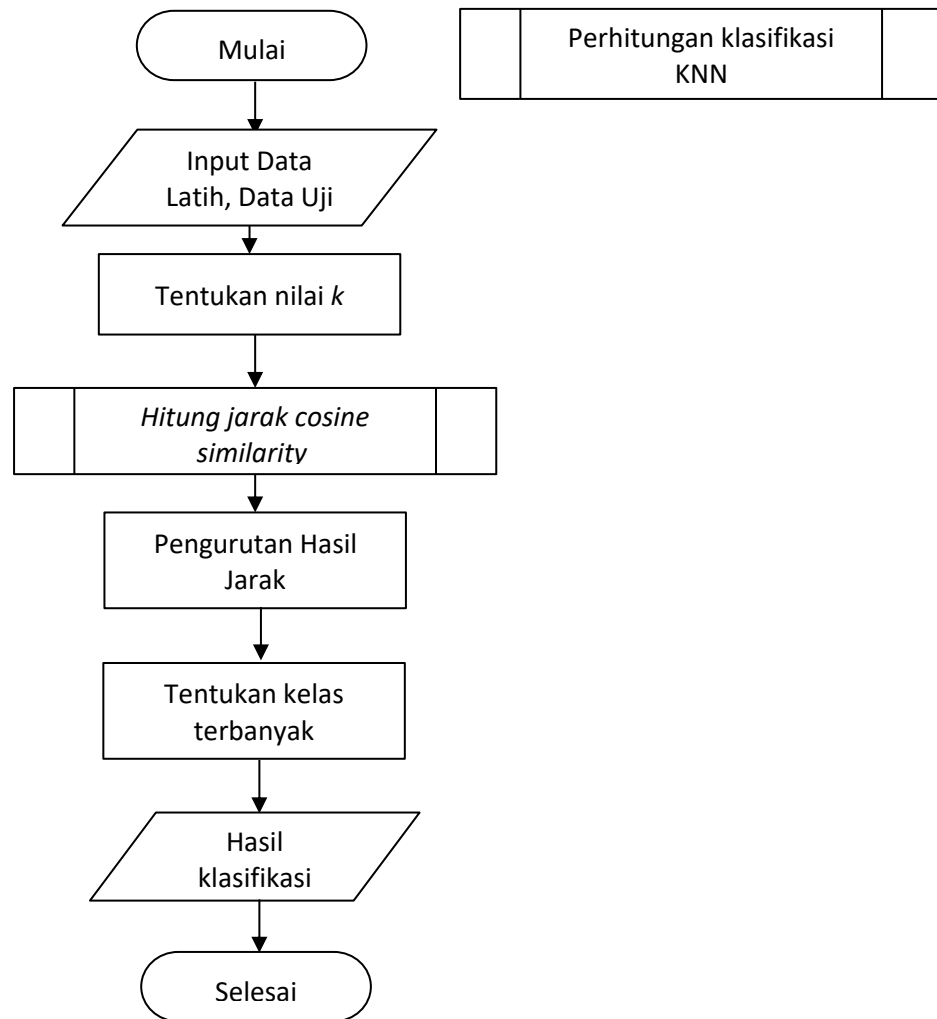
Tahapan pada *Information Gain* adalah menghitung *entropy* dari kelas makanan. Ubah data tunggal pada masing-masing atribut menjadi data kelompok. Proses selanjutnya adalah menghitung *entropy* pada masing-masing atribut/fitur. Proses terakhir adalah menghitung *gain* dari masing-masing atribut dan diurutkan berdasarkan nilai terbesar.  $K$  teratas akan diambil dan akan dijadikan fitur untuk proses klasifikasi. Tahapan proses *Information Gain* dapat dilihat pada Gambar 4.20.



Gambar 4.20 Diagram alir *Information Gain*

#### 4.1.4 Perancangan Algoritme *K-Nearest Neighbor*

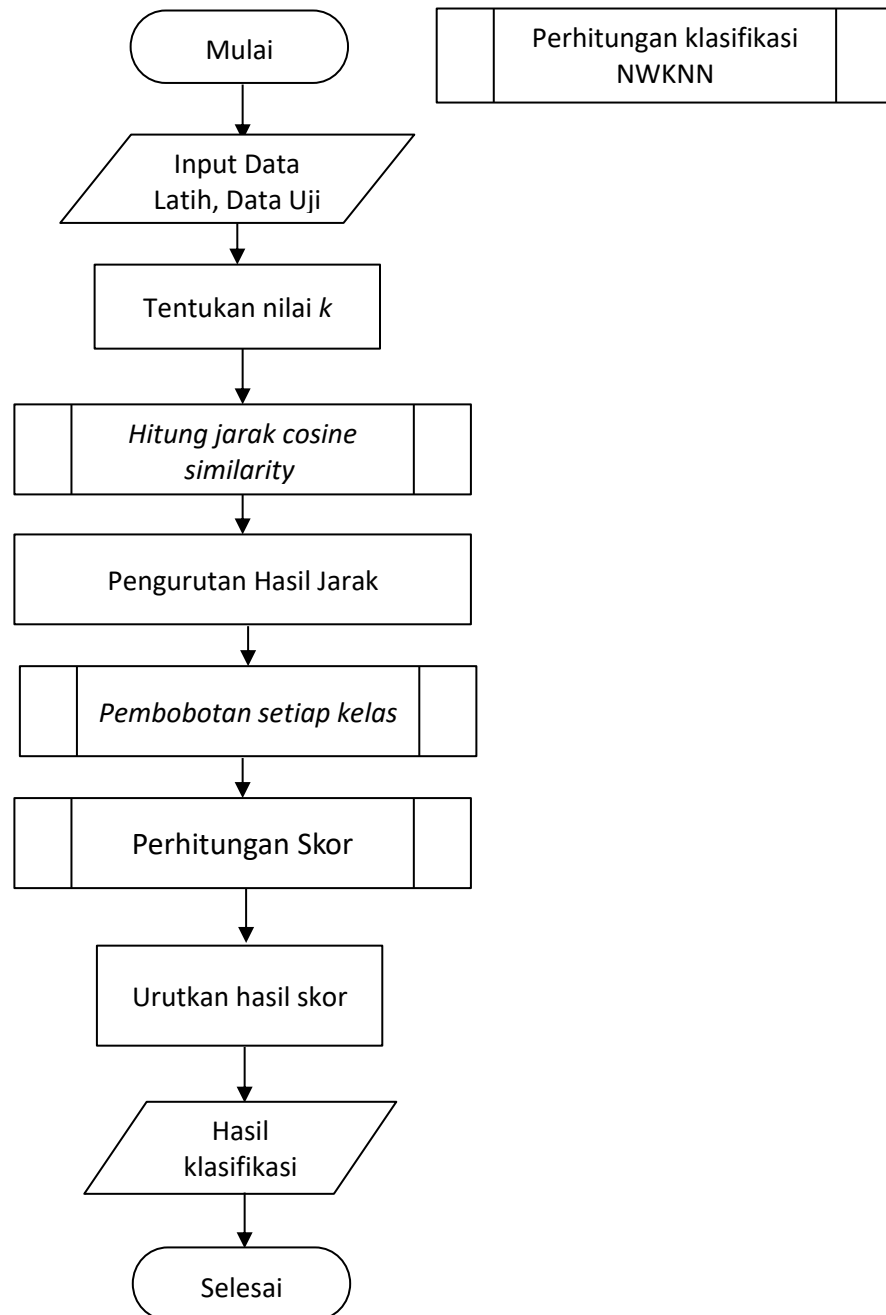
Tahapan perancangan klasifikasi menggunakan *K-NN* dimulai dari penentuan *K*, input data latih dan data uji. Hitung jarak kemiripan antara data latih dengan data uji menggunakan cosine similarity. Hasilnya akan diurutkan berdasarkan nilai terbesar. Tentukan kelas dengan melihat kelas terbanyak dari *K*. Setelah itu akan didapatkan hasil klasifikasi. Berikut diagram alir *K-NN* ditunjukkan pada Gambar 4.21.



**Gambar 4.21** Digram alir *K-NN*

#### 4.1.5 Perancangan Algoritme Neighbor Weighted *K*-Nearest Neighbor

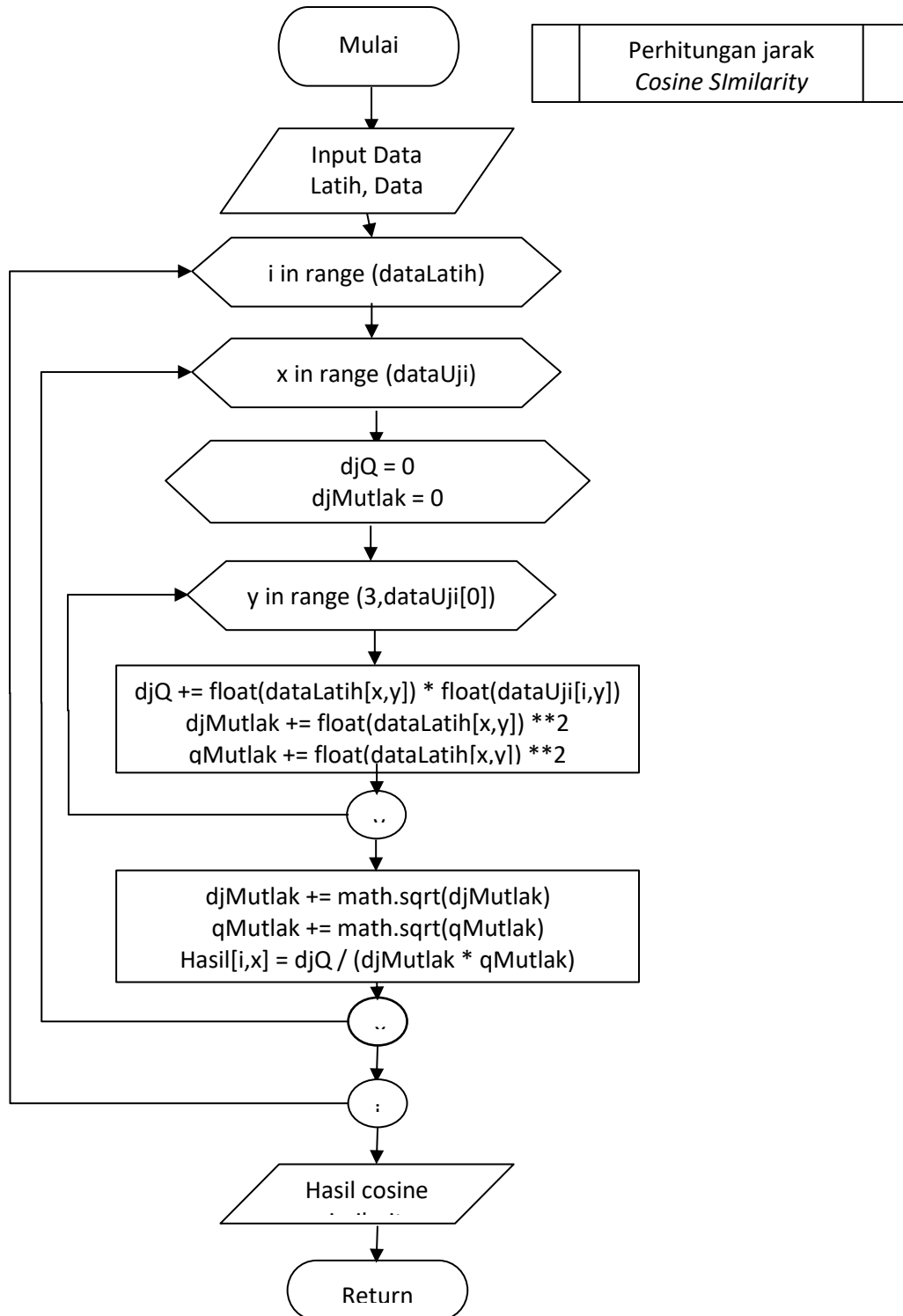
Tahapan perancangan klasifikasi *NWKNN* hampir sama dengan *K*-*NN*. Perbedaannya pada langkah perhitungan bobot dan perhitungan skor. Berikut diagram alir algoritme *NWKNN* ditunjukkan pada Gambar 4.22.



**Gambar 4.22** Diagram alir *NWKNN*

#### 4.1.6 Perancangan Algoritme Perhitungan Jarak *Cosine Similarity*

Tahapan perhitungan jarak menggunakan *Cosine Similarity* dapat dilihat pada gambar 4.23.



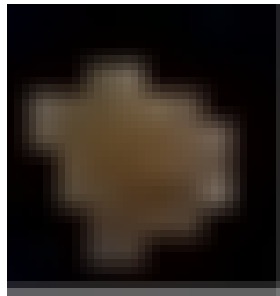
**Gambar 4.23** Diagram alir *Cosine Similarity*

## 4.2 Perhitungan Manualisasi

Perhitungan manualisasi pada penelitian ini dilakukan pada citra yang telah melalui tahapan *pre-processing* mulai dari *Gaussian Blur*, transformasi warna RGB ke HSV, *tresholding*, transformasi warna HSV ke *Grayscale*, *Tresholding*, *Opening* dan *Bitwise*. Sampel data yang akan dilakukan perhitungan manualisasi yaitu satu sampel makanan donat sebesar 200 x 200 piksel yang ditunjukkan pada Gambar 4.24 . Pada perhitungan manualisasi gambar akan di *resize* sebesar 10 x 10 piksel yang ditunjukkan pada Gambar 4.25 dikarenakan gambar terlalu besar dan terlalu banyak piksel pada gambar asli.



Gambar 4.24 Sampel Citra Rendang 200 x 200



Gambar 4.25 Sampel Citra Rendang 10 x 10

### 4.2.1 Perhitungan *Color Moments*

Pada hasil *preprocessing* menghasilkan warna *R,G,B* yang dapat dilihat pada Gambar 4.24. Karena warna yang diambil pada ekstraksi fitur *Color Moments* ini selain *R,G,B* adalah *H,S,V* maka nilai *HSV* harus dicari terlebih dahulu. Nilai *H,S,V* didapatkan melalui *library Open CV* yang dapat dilihat pada Tabel 4.1 sampai Tabel 4.5.

Tabel 4.1 Nilai warna R citra

x,y	0	1	2	3	4	5	6	7	8	9
0	13	9	5	0	0	3	7	13	9	4
1	7	3	0	0	0	0	1	5	5	4
2	0	0	0	62	89	0	0	0	2	2
3	0	78	54	47	59	75	65	0	0	0

4	0	67	46	39	39	47	64	86	0	0
5	0	0	51	33	31	43	55	64	0	0
6	0	0	58	42	27	26	41	96	0	2
7	7	0	0	40	37	45	50	0	0	2
8	11	7	0	53	50	0	0	0	0	2
9	10	7	2	0	0	0	0	0	0	0

**Tabel 4.2 Nilai warna G citra**

0	0	0	0	0	1	2	1	3	0	1
1	0	0	0	0	1	0	1	0	0	0
2	0	0	0	82	108	0	0	0	0	0
3	0	93	77	77	90	102	83	0	0	0
4	1	85	72	73	76	79	88	99	0	0
5	0	0	75	64	67	74	79	79	0	0
6	2	0	74	66	56	53	62	107	0	0
7	0	0	0	55	56	65	65	0	0	1
8	0	2	0	59	59	0	0	0	0	1
9	2	0	0	0	0	0	0	2	1	1

**Tabel 4.3 Nilai warna B citra**

x,y	0	1	2	3	4	5	6	7	8	9
0	0	0	1	0	0	0	0	0	0	0
1	0	2	3	4	4	1	1	0	0	0
2	4	7	8	93	121	10	8	5	1	1
3	6	102	92	96	113	123	100	11	6	3
4	7	96	89	97	104	108	110	115	7	3
5	4	8	93	89	97	105	103	95	6	1
6	2	4	87	88	83	79	83	121	4	0
7	0	1	7	71	77	83	81	7	1	0
8	0	1	3	70	72	10	9	3	0	0
9	0	0	1	4	6	4	3	1	0	0

**Tabel 4.4 Nilai warna H citra**

x,y	0	1	2	3	4	5	6	7	8	9
0	120	120	126	0	60	100	116	113	120	113
1	120	140	0	0	8	0	0	120	120	120
2	0	0	0	19	18	0	0	0	135	135
3	0	19	18	18	17	17	15	0	0	0
4	4	19	18	18	17	16	16	13	0	0
5	0	0	17	17	16	15	15	15	0	0
6	30	0	17	16	16	15	15	13	0	120
7	120	0	0	15	14	16	15	0	0	105
8	120	115	0	11	12	0	0	0	0	105
9	114	120	135	0	0	0	0	45	60	60



**Tabel 4.5 Nilai warna S citra**

x,y	0	1	2	3	4	5	6	7	8	9
0	255	255	255	0	255	255	255	255	255	255
1	255	255	255	255	255	255	0	255	255	255
2	255	255	255	85	67	255	255	255	255	255
3	255	60	105	130	122	100	89	255	255	255
4	255	77	123	152	159	144	107	64	255	255
5	255	255	115	160	174	151	119	83	255	255
6	255	255	85	133	172	171	129	53	255	255
7	255	255	255	111	132	117	98	255	255	255
8	255	219	255	62	78	255	255	255	0	255
9	255	255	255	255	255	255	255	255	255	255

**Tabel 4.6 Nilai warna V citra**

x,y	0	1	2	3	4	5	6	7	8	9
0	13	9	5	0	1	3	7	13	9	4
1	7	3	3	4	4	1	1	5	5	4
2	4	7	8	93	121	10	8	5	2	2
3	6	102	92	96	113	123	100	11	6	3
4	7	96	89	97	104	108	110	115	7	3
5	4	8	93	89	97	105	103	95	6	1
6	2	4	87	88	83	79	83	121	4	2
7	7	1	7	71	77	83	81	7	1	2
8	11	7	3	70	72	10	9	3	0	2
9	10	7	2	4	6	4	3	2	1	1

Seperti yang dijelaskan pada Bab 2, terdapat 3 perhitungan manual *Color Moments* yaitu *Mean*, *Variance* dan *Skewness*. Berikut perhitungan dari masing-masing ekstraksi fitur:

Sebelum dihitung *mean*, *variance* dan *skewness* terlebih dahulu dihitung total piksel dari masing-masing citra yaitu:

$$Total = l * w = 10 * 10 = 100$$

#### 1. *Mean*

a) Citra Warna R

$$MeanR = \frac{13 + 9 + 5 + \dots + 0 + 0}{100}$$

$$= 19.01$$

b) Citra Warna G

$$MeanG = \frac{0 + 0 + 0 + \dots + 1 + 1}{100}$$

$$= 25.22$$

c) Citra Warna B

$$\begin{aligned} \text{Mean}B &= \frac{0 + 0 + 1 + \dots + 0 + 0}{100} \\ &= 33.29 \end{aligned}$$

d) Citra Warna H

$$\begin{aligned} \text{Mean}H &= \frac{120 + 120 + 126 + \dots + 60 + 60}{100} \\ &= 36.67 \end{aligned}$$

e) Citra Warna S

$$\begin{aligned} \text{Mean}S &= \frac{255 + 255 + 255 + \dots + 0 + 255}{100} \\ &= 200.11 \end{aligned}$$

f) Citra Warna V

$$\begin{aligned} \text{Mean}V &= \frac{13 + 9 + 5 + \dots + 1 + 1}{100} \\ &= 34.67 \end{aligned}$$

## 2. Variance

a) Citra Warna R

$$\begin{aligned} \text{Variance}R &= \sqrt{\frac{(13 - 19.01)^2 + (9 - 19.01)^2 + \dots + (0 - 19.01)^2}{100}} \\ &= 26.1696 \end{aligned}$$

b) Citra Warna G

$$\begin{aligned} \text{Variance}G &= \sqrt{\frac{(0 - 25.22)^2 + (0 - 25.22)^2 + \dots + (1 - 25.22)^2}{100}} \\ &= 36.446 \end{aligned}$$

c) Citra Warna B

$$\begin{aligned} \text{Variance}B &= \sqrt{\frac{(0 - 33.29)^2 + (0 - 33.29)^2 + \dots + (1 - 33.29)^2}{100}} \\ &= 44.173 \end{aligned}$$

d) Citra Warna H

$$\begin{aligned} \text{Variance}H &= \sqrt{\frac{(120 - 36.7)^2 + (120 - 36.7)^2 + \dots + (60 - 36.7)^2}{100}} \\ &= 48.35 \end{aligned}$$

e) Citra Warna S

*VarianceS*

$$= \sqrt{\frac{(255 - 200.11)^2 + (255 - 200.11)^2 + \dots + (255 - 200.11)^2}{100}}$$

$$= 77.524$$

f) Citra Warna V

$$VarianceV = \sqrt{\frac{(13 - 34.67)^2 + (9 - 34.67)^2 + \dots + (1 - 34.67)^2}{100}}$$

$$= 43.227$$

### 3. Skewness

a) Citra Warna R

*SkewnessR*

$$= \left( \frac{((13 - 19.01)^3 + (13 - 19.01)^3 + \dots + (13 - 19.01)^3)}{100} \right) / (26.17)^3$$

$$= 1.16546$$

b) Citra Warna G

*SkewnessG*

$$= \left( \frac{((0 - 25.22)^3 + (0 - 25.22)^3 + \dots + (1 - 25.22)^3)}{100} \right) / (36.446)^3$$

$$= 0.899$$

c) Citra Warna B

*SkewnessB*

$$= \left( \frac{((0 - 33.29)^3 + (0 - 33.29)^3 + \dots + (1 - 33.29)^3)}{100} \right) / (44.173)^3$$

$$= 0.8222$$

d) Citra Warna H

*SkewnessH*

$$= \left( \frac{((120 - 36.7)^3 + (120 - 36.7)^3 + \dots + (60 - 36.7)^3)}{100} \right) / (48.35)^3$$

$$= 1.0763$$

e) Citra Warna S

$$\begin{aligned}
 & \text{Skewness } S \\
 &= \left( \frac{(255 - 200.11)^3 + (255 - 200.11)^3 + \dots + (255 - 200.11)^3}{100} \right) // (77.524)^3 \\
 &= -0.9831
 \end{aligned}$$

f) Citra Warna V

$$\begin{aligned}
 & \text{Skewness } V \\
 &= \left( \frac{(13 - 34.67)^3 + (9 - 34.67)^3 + \dots + (1 - 34.67)^3}{100} \right) // (43.227)^3 \\
 &= 0.826
 \end{aligned}$$

#### 4.2.2 Perhitungan Gray Level Co-Occurrence Matrix

Citra pada hasil *preprocessing* masih berwarna RGB maka harus diubah dalam warna *grayscale* terlebih dahulu. Nilai *Grayscale* ditunjukkan pada Tabel 4.7.

**Tabel 4.7 Nilai Grayscale Citra**

x,y	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	2	1	3	1	1
1	1	1	1	1	2	0	1	1	1	0
2	1	2	2	83	110	3	2	1	1	1
3	2	94	79	79	93	105	86	3	2	1
4	3	86	74	76	80	84	92	102	2	1
5	1	2	78	68	72	80	83	82	2	0
6	2	1	76	70	61	58	66	110	1	0
7	1	0	2	58	60	68	68	2	0	1
8	1	2	1	62	62	3	3	1	0	1
9	2	1	1	1	2	1	1	1	1	1

Berikut langkah-langkah perhitungan manual ekstraksi fitur GLCM :

- Sebelum dilakukan ekstraksi fitur pada GLCM dilakukan pembentukan matriks *co-occurrence* dengan 4 sudut yaitu 0°, 45°, 90° dan 135°. Jarak antar piksel yaitu 1 piksel. Berikut hasil pembentukan matriks *co-occurrence* dengan pixel (1, 1).

a) Sudut 0°

**Tabel 4.8 Pembentukan Matriks Co-Occurrence piksel (1,1) sudut 0° dan d=1**

x,y	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	2	1	3	1	1
1	1	1	1	1	2	0	1	1	1	0

2	1	2	2	83	110	3	2	1	1	1
3	2	94	79	79	93	105	86	3	2	1
4	3	86	74	76	80	84	92	102	2	1
5	1	2	78	68	72	80	83	82	2	0
6	2	1	76	70	61	58	66	110	1	0
7	1	0	2	58	60	68	68	2	0	1
8	1	2	1	62	62	3	3	1	0	1
9	2	1	1	1	2	1	1	1	1	1

Berdasarkan Tabel 4.8 diatas telah terbentuk matriks *Co-Occurrence* piksel (1,1) dan jarak 1 berjumlah 15 pasang. Perhitungan *Co-occurrence* seperti di atas juga berlaku pada semua piksel yang memiliki nilai piksel ketetanggaan. Nilai matriks *Co-occurrence* sudut  $0^\circ$  ditunjukkan pada Tabel 4.8 dimana x sebagai piksel asal dan y sebagai piksel tetangga.

**Tabel 4.9 Hasil Matriks Co-Occurrence piksel (1,1) sudut  $45^\circ$  dan d=1**

x,y	0	1	...	50	51	...	109	110
0	0	0	...	0	0	...	0	0
1	0	16	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
50	0	0	...	0	0	...	0	0
51	0	0	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
109	0	0	...	0	0	...	0	0
110	0	1	...	0	0	...	0	0

b) Sudut  $45^\circ$

**Tabel 4.10 Pembentukan Matriks Co-Occurrence piksel (1,1) sudut  $45^\circ$  dan d=1**

x,y	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	2	1	3	1	1
1	1	1	1	1	2	0	1	1	1	0
2	1	2	2	83	110	3	2	1	1	1
3	2	94	79	79	93	105	86	3	2	1
4	3	86	74	76	80	84	92	102	2	1
5	1	2	78	68	72	80	83	82	2	0
6	2	1	76	70	61	58	66	110	1	0
7	1	0	2	58	60	68	68	2	0	1
8	1	2	1	62	62	3	3	1	0	1
9	2	1	1	1	2	1	1	1	1	1

Berdasarkan Tabel 4.10 diatas telah terbentuk matriks *Co-Occurrence* piksel (1,1) dan jarak 1 berjumlah 11 pasang. Perhitungan *Co-occurrence* seperti di atas juga berlaku pada semua piksel yang memiliki nilai piksel ketetanggaan. Nilai matriks *Co-occurrence* sudut  $45^\circ$  ditunjukkan pada Tabel 4.10 dimana x sebagai piksel asal dan y sebagai piksel tetangga.

**Tabel 4.11 Hasil Matriks *Co-Occurrence* piksel (1,1) sudut 90° dan d=1**

x,y	0	1	...	50	51	...	109	110
0	0	0	...	0	0	...	0	0
1	0	11	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
50	0	0	...	0	0	...	0	0
51	0	0	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
109	0	0	...	0	0	...	0	0
110	0	0	...	0	0	...	0	0

c) Sudut 90°

**Tabel 4.12 Pembentukan Matriks *Co-Occurrence* piksel (1,1) sudut 90° dan d=1**

x,y	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	2	1	3	1	1
1	1	1	1	1	2	0	1	1	1	0
2	1	2	2	83	110	3	2	1	1	1
3	2	94	79	79	93	105	86	3	2	1
4	3	86	74	76	80	84	92	102	2	1
5	1	2	78	68	72	80	83	82	2	0
6	2	1	76	70	61	58	66	110	1	0
7	1	0	2	58	60	68	68	2	0	1
8	1	2	1	62	62	3	3	1	0	1
9	2	1	1	1	2	1	1	1	1	1

Berdasarkan tabel 4.12 diatas telah terbentuk matriks *Co-Occurrence* piksel (1,1) dan jarak 1 berjumlah 15 pasang. Perhitungan *Co-occurrence* seperti di atas juga berlaku pada semua piksel yang memiliki nilai piksel ketetanggaan. Nilai matriks *Co-occurrence* sudut 90° ditunjukkan pada Tabel 4.12 dimana x sebagai piksel asal dan y sebagai piksel tetangga.

x,y	0	1	...	50	51	...	109	110
0	0	0	...	0	0	...	0	0
1	0	15	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
50	0	0	...	0	0	...	0	0
51	0	0	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
109	0	0	...	0	0	...	0	0
110	0	0	...	0	0	...	0	0

**Tabel 4.13 Hasil Matriks *Co-Occurrence* piksel (1,1) sudut 90° dan d=1**

d) Sudut 135°

**Tabel 4.14 Pembentukan Matriks Co-Occurrence piksel (1,1) sudut 135° dan d=1**

x,y	0	1	2	3	4	5	6	7	8	9
0	1	1	1	0	1	2	1	3	1	1
1	1	1	1	1	2	0	1	1	1	0
2	1	2	2	83	110	3	2	1	1	1
3	2	94	79	79	93	105	86	3	2	1
4	3	86	74	76	80	84	92	102	2	1
5	1	2	78	68	72	80	83	82	2	0
6	2	1	76	70	61	58	66	110	1	0
7	1	0	2	58	60	68	68	2	0	1
8	1	2	1	62	62	3	3	1	0	1
9	2	1	1	1	2	1	1	1	1	1

Berdasarkan Tabel 4.14 diatas telah terbentuk matriks *Co-Occurrence* piksel (1,1) dan jarak 1 berjumlah 13 pasang. Perhitungan *Co-occurrence* seperti di atas juga berlaku pada semua piksel yang memiliki nilai piksel ketetanggaan. Nilai matriks *Co-occurrence* sudut 135° ditunjukkan pada Tabel 4.14 dimana x sebagai piksel asal dan y sebagai piksel tetangga.

**Tabel 4.15 Hasil Matriks Co-Occurrence piksel (1,1) sudut 135° dan d=1**

x,y	0	1	...	50	51	...	109	110
0	0	0	...	0	0	...	0	0
1	0	13	...	0	0	...	0	1
...	...	...	...	...	...	...	...	...
50	0	0	...	0	0	...	0	0
51	0	0	...	0	0	...	0	0
...	...	...	...	...	...	...	...	...
109	0	0	...	0	0	...	0	0
110	0	0	...	0	0	...	0	0

2. Langkah selanjutnya setelah mendapatkan matriks *co-occurrence* yaitu membuat matrix *co-occurrence* simetris dengan menjumlahkan matriks *co-occurrence* dengan matriks transpose nya.

a. Sudut 0°

$$mSimetris = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 16 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix} +$$

b. Sudut  $45^\circ$

67



c. Sudut  $90^\circ$

[illegible]

d. Sudut  $135^\circ$

$$mSimestris = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 13 & \dots & 0 & 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix} +$$

$$\begin{pmatrix}
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 13 & \dots & 0 & 0 & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 0 & \dots & 0 & 0
\end{pmatrix}
=
\begin{pmatrix}
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 26 & \dots & 0 & 0 & \dots & 0 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 0 & \dots & 0 & 0
\end{pmatrix}$$

3. Langkah terakhir sebelum melakukan ekstraksi fitur yaitu melakukan normalisasi matriks simetris pada setiap sudut. Normalisasi dilakukan dengan cara membagi nilai setiap piksel dengan jumlah nilai piksel matriks simetris. Hasil normalisasi pada matriks jika dijumlahkan akan bernilai 1.

a. Sudut  $0^\circ$

Jumlah nilai piksel dalam matriks simetris pada sudut  $0^\circ$  adalah 154 sehingga didapatkan hasil matriks simetris normalisasi sebagai berikut.

$$mNormalize = \frac{\begin{pmatrix}
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 32 & \dots & 0 & 0 & \dots & 0 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 0 & \dots & 0 & 0
\end{pmatrix}}{154}
=
\begin{pmatrix}
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0.2078 & \dots & 0 & 0 & \dots & 0 & 0.0065 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0.0065 & \dots & 0 & 0 & \dots & 0 & 0
\end{pmatrix}$$

b. Sudut  $45^\circ$

Jumlah nilai piksel dalam matriks simetris pada sudut 45° adalah 140 sehingga didapatkan hasil matriks simetris normalisasi sebagai berikut.

$$\begin{aligned}
 mNormalize &= \frac{\begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 22 & \dots & 0 & 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}}{140} \\
 &= \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0.1572 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}
 \end{aligned}$$

c. Sudut 90°

Jumlah nilai piksel dalam matriks simetris pada sudut 90° adalah 154 sehingga didapatkan hasil matriks simetris normalisasi sebagai berikut.

$$\begin{aligned}
 mNormalize &= \frac{\begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 30 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}}{154} \\
 &= \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0.1949 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}
 \end{aligned}$$

d. Sudut 135°

Jumlah nilai piksel dalam matriks simetris pada sudut 135° adalah 138 sehingga didapatkan hasil matriks simetris normalisasi sebagai berikut.

$$mNormalize = \frac{\begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 26 & \dots & 0 & 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}}{138}$$

$$= \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0.1885 & \dots & 0 & 0 & \dots & 0 & 0.0073 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0.0073 & \dots & 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

4. Menghitung ekstraksi fitur tekstur *GLCM* berdasarkan matriks normalisasi. Terdapat 13 fitur yang digunakan yaitu *Angular Second Moment (ASM)*, *Contrast*, *Correlation*, *Sum of Squares: Variance*, *Inverse Difference Moment (IDM)*, *Sum Average (AVER)*, *Sum Entropy (SENT)*, *Sum Variance (SVAR)*, *Entropy*, *Difference Entropy (DENT)*, *Difference Variance (DVAR)*, *Information Measure of Correlation 1* dan *Information Measure of Correlation 2*. Contoh perhitungan berikut dimulai dari baris kedua matriks simetris dikarenakan baris pertama nilai piksel adalah 0. Berikut perhitungan ekstraksi fitur *GLCM* yaitu:

a. Sudut  $0^\circ$

- *Angular Second Moment (ASM)*

$$\begin{aligned} ASM &= 0^2 + 0.2078^2 + 0.0909^2 + 0.0195^2 + \dots + 0^2 \\ &= 0 + 0,04318 + 0,0083 + 0,00038 + \dots + 0 \\ &= 0.06494 \end{aligned}$$

- *Contrast*

$$\begin{aligned} Contrast &= ((1 - 0)^2 * 0) + ((1 - 1)^2 * 0.2078) + ((1 - 2)^2 * 0.0909) + ((1 - 3)^2 * 0.0195) + \dots \\ &\quad + ((110 - 110)^2 * 0) \\ &= 0 + 0 + 0.0909 + 0,078 + \dots + 0 \\ &= 1286.221 \end{aligned}$$

- *Sum of Squares: Variance*

Sebelum dilakukan perhitungan *variance* akan dihitung dulu rata-rata dari matriks simetris. Perhitungan dilakukan sebagai berikut.

$$\mu = \frac{(0 + 0.2078 + 0.0909 + 0.0195 + \dots + 0)}{12321} = 0.00000812$$

Hitung *variance* dengan cara berikut :

$$\begin{aligned} \text{Variance} &= ((1 - 0.00000812) * 0)^2 + (2 - 0.00000812)^2 \\ &\quad * 0.2078) + \dots + ((110 - 0.00000812)^2 * 0) \\ &= 0 + 0,83119325 + \dots + 0 \\ &= 2759.6892 \end{aligned}$$

- *Inverse Difference Moment (IDM)*

$$\begin{aligned} IDM &= \left( \frac{0^2}{1 + ((1 - 0)^2)} \right) + \left( \frac{(0.2078)^2}{1 + ((1 - 1)^2)} \right) + \dots \\ &\quad + \left( \frac{0^2}{1 + ((110 - 110)^2)} \right) \\ &= 0 + 0.0432 + \dots + 0 \\ &= 0.05273 \end{aligned}$$

- *Correlation*

Sebelum melakukan perhitungan *correlation* terlebih dahulu untuk menghitung rata-rata dari  $P_x$ ,  $P_y$  dan standar deviasi  $P_x$ ,  $P_y$ .

$$\begin{aligned} \mu_x &= (1 * 0) + (1 * 0.2078) + (1 * 0.0909) + \dots + (110 * 0) \\ &= 0 + 0.2078 + 0.0909 + \dots + 0 \\ &= 34.70779 \end{aligned}$$

$$\begin{aligned} \mu_y &= (0 * 0) + (1 * 0.2078) + (2 * 0.0909) + \dots + (110 * 0) \\ &= 0 + 0.2078 + 0.1818 + \dots + 0 \\ &= 34.70779 \end{aligned}$$

$$\begin{aligned} \sigma_x &= (0 * \sqrt{(1 - 34.70779)^2}) + (0.2078 * \\ &\quad \sqrt{(1 - 34.70779)^2}) + \dots + (0 * \sqrt{(110 - 34.70779)^2}) \\ &= 0 + 7,0044 + \dots + 0 \\ &= 37.9128 \end{aligned}$$

$$\begin{aligned} \sigma_y &= (0 * \sqrt{(1 - 34.70779)^2}) + (0.2078 * \\ &\quad \sqrt{(2 - 34.70779)^2}) + \dots + (0 * \sqrt{(110 - 34.70779)^2}) \\ &= 0 + 5,7966 + \dots + 0 \\ &= 37.9128 \end{aligned}$$

Setelah didapatkan rata-rata dan standar deviasi hitung *correlation* dengan cara dibawah ini:

$$\begin{aligned}
Correlation &= \frac{((1 * 0) * 0) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\
&+ \frac{((1 * 1) * 0.2078) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\
&+ \dots \\
&+ \frac{((110 * 110) * 0) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\
&= -8209.3219
\end{aligned}$$

Sebelum melakukan perhitungan *Sum Average*, *Sum Entropy* dan *Sum Variance* terlebih dahulu dihitung  $P_{x+y}$ .

$$\begin{aligned}
P_{(4)} &= P_{(0+4)} + P_{(1+3)} + P_{(2+2)} + P_{(3+1)} + P_{(4+0)} \\
&= 0 + 0.0194805 + 0.012987 + 0.0194805 + 0 \\
&= 0.05194805
\end{aligned}$$

Hasil perhitungan  $P_{x+y}$  ditunjukkan pada Tabel 4.16 dibawah ini.

$x+y$	0	1	2	...	220	221
$P_{x+y}$	0	0	0.20779	...	0	0

**Tabel 4.16 Nilai  $P_{x+y}$  Sudut 0°**

- *Sum Average (AVER)*

$$\begin{aligned}
AVER &= (2 * 0.20779) + \dots + (220 * 0) + (221 * 0) \\
&= 69.41558
\end{aligned}$$

- *Sum Entropy (SENT)*

$$\begin{aligned}
SENT &= (2 * 0.20779 * \log(0.20779)) + \dots + (220 * 0 * 0) \\
&\quad + (221 * 0 * 0) \\
&= 288.92594
\end{aligned}$$

- *Sum Variance (SVAR)*

$$\begin{aligned}
SVAR &= ((2 * 288.92594)^2 * 0.20779) + \dots \\
&\quad + ((220 * 288.92594)^2 * 0) \\
&\quad + ((221 * 288.92594)^2 * 0) \\
&= 53118.8295
\end{aligned}$$

- *Entropy*

$$\begin{aligned}
Entropy &= -((0 * 0) + (0.2078 * \log(0.2078)) \\
&\quad + (0.0909 * \log(0.0909)) + \dots + (0 * 0)) \\
&= 3.71315
\end{aligned}$$

Sebelum melakukan perhitungan *Difference Entropy* dan *Difference Variance*, hitung  $P_{x-y}$  dengan cara sebagai berikut.

$$\begin{aligned} P_{(0)} &= P_{(|0-0|)} + P_{(|1-1|)} + P_{(|2-2|)} + \dots + P_{(|110-110|)} \\ &= 0 + 0.20779 + 0.012987 + \dots + 0 \\ &= 0.27273 \end{aligned}$$

Hasil perhitungan  $P_{x-y}$  ditunjukkan pada Tabel 4.17 dibawah ini.

$x-y$	0	1	2	...	109	110
$P_{x-y}$	0.27273	0.22077	0.0649	...	0.01298	0

**Tabel 4.17 Nilai  $P_{x-y}$  Sudut  $0^\circ$**

- *Difference Entropy (DENT)*

$$\begin{aligned} DENT &= -((0 * 0.27273 * \log(0.27273)) \\ &\quad + (1 * 0.22077 * \log(0.22077)) \\ &\quad + (2 * 0.0649 * \log(0.0649)) + \dots + (110 * 0 * 0)) \\ &= 74.49298 \end{aligned}$$

- *Difference Variance (DVAR)*

$$\begin{aligned} DVAR &= ((0^2 * 0.27273) + (1^2 * 0.22077) + (2^2 * 0.0649) + \dots \\ &\quad + (110^2 * 0)) \\ &= 1286.2207 \end{aligned}$$

Sebelum dilakukan perhitungan Information Measure of Correlatin 1 dan 2, terlebih dahulu dihitung nilai  $P_x, P_y, HX, HY, HXY1, HXY2$ .

Dibawah ini contoh perhitungan  $P_{(x=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.18.

**Tabel 4.18 Matriks Co-occurrence Simetris Baris ke 1 Sudut  $0^\circ$**

$x/y$	0	1	2	...	110
1	0	0.20779	0.0909	...	0.00649

$$\begin{aligned} P_{(x=1)} &= P_{(1,0)} + P_{(1,1)} + P_{(1,2)} + \dots + P_{(1,110)} \\ &= 0 + 0.20779 + 0.0909 + \dots + 0.00649 \\ &= 0.33116 \end{aligned}$$

Hasil perhitungan  $P_x$  dapat dilihat pada Tabel 4.19.

$x$	0	1	2	...	110
$P_x$	0	0.33116	0.162337	...	0

**Tabel 4.19 Nilai  $P_x$  sudut  $0^\circ$**

Dibawah ini contoh perhitungan  $P_{(y=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.20.

y/x	0	1	2	...	110
1	0	0.20779	0.0909	...	0.00649

$$\begin{aligned}
 P_{(y=1)} &= P_{(0,1)} + P_{(1,1)} + P_{(2,1)} + \dots + P_{(110,1)} \\
 &= 0 + 0.20779 + 0.0909 + \dots + 0.00649 \\
 &= 0.33116
 \end{aligned}$$

Hasil perhitungan  $P_y$  dapat dilihat pada Tabel 4.20.

**Tabel 4.20 Nilai  $P_y$  sudut  $0^\circ$**

y	0	1	2	...	110
$P_y$	0	0.33116	0.162337	...	0

$$\begin{aligned}
 HX &= -(0 * (0) + 1 * \log(0.33116) + 2 * \log(0.162337) + \dots + 110 \\
 &\quad * (0)) \\
 &= 261.45353
 \end{aligned}$$

$$\begin{aligned}
 HY &= -(0 * (0) + 1 * \log(0.33116) + 2 * \log(0.162337) + \dots + 110 \\
 &\quad * (0)) \\
 &= 261.45353
 \end{aligned}$$

$$\max\{HX, HY\} = 261.45353$$

$$\begin{aligned}
 HXY1 &= -(0 * \log(0 * 0) + 0.20779 * \log(0.33116 * 0.33116) + \dots \\
 &\quad + 0 * \log(0 * 0)) \\
 &= 4.7537005
 \end{aligned}$$

$$\begin{aligned}
 HXY2 &= -(0 * 0 * \log(0 * 0) + 0.33116 * 0.33116 \\
 &\quad * \log(0.33116 * 0.33116) + \dots + 0 * 0 * \log(0 * 0)) \\
 &= 4.506755
 \end{aligned}$$

- *Information Measure of Correlation 1*

$$IMoC1 = \frac{(3.71315 - 4.7537005)}{261.45353} = -0.003979$$

- *Information Measure of Correlation 2*

$$IMoC2 = \sqrt{|1 - \exp(-2 * (4.506755 - 3.71315))|} = 0.8919104$$

b. Sudut  $45^\circ$

- Angular Second Moment (ASM)



$$\begin{aligned}
 ASM &= 0^2 + 0^2 + 0^2 + 0^2 + \dots + 0^2 \\
 &= 0 + 0 + 0 + 0 + \dots + 0 \\
 &= 0.04143
 \end{aligned}$$

- Contrast

$$\begin{aligned}
 Contrast &= ((1 - 0)^2 * 0) + ((1 - 1)^2 * 0.1572) + ((1 - 2)^2 * 0.05) + ((1 - 3)^2 * 0.0286) + \dots + ((1 - 110)^2 * 0) \\
 &= 0 + 0 + 0.05 + 0.1144 + \dots + 0 \\
 &= 1843.215
 \end{aligned}$$

- Sum of Squares: Variance

Sebelum dilakukan perhitungan *variance* akan dihitung dulu rata-rata dari matriks simetris. Perhitungan dilakukan sebagai berikut :

$$\mu = \frac{(0 + 0.1572 + 0.05 + 0.0286 + \dots + 0)}{12321} = 0.00000812$$

Hitung *variance* dengan cara berikut :

$$\begin{aligned}
 Variance &= ((1 - 0.00000812) * 0)^2 + (2 - 0.00000812)^2 * 0.1572 + \dots + ((110 - 0.00000812)^2 * 0) \\
 &= 0 + 0.6278 + \dots + 0 \\
 &= 2907.873
 \end{aligned}$$

- Inverse Difference Moment (IDM)

$$\begin{aligned}
 IDM &= \left( \frac{0^2}{1 + ((1 - 0)^2)} \right) + \left( \frac{(0.1572)^2}{1 + ((1 - 1)^2)} \right) + \dots \\
 &\quad + \left( \frac{0^2}{1 + ((110 - 110)^2)} \right) \\
 &= 0 + 0.0247 + \dots + 0 \\
 &= 0.03298
 \end{aligned}$$

- Correlation

Sebelum melakukan perhitungan *correlation* terlebih dahulu untuk menghitung rata-rata dari  $P_x$ ,  $P_y$  dan standar deviasi  $P_x$ ,  $P_y$ .

$$\begin{aligned}
 \mu_x &= (1 * 0) + (1 * 0.1572) + (1 * 0.05) + \dots + (110 * 0) \\
 &= 0 + 0.1572 + 0.05 + \dots + 0 \\
 &= 36.76429
 \end{aligned}$$

$$\begin{aligned}
 \mu_y &= (0 * 0) + (1 * 0.1572) + (2 * 0.05) + \dots + (110 * 0) \\
 &= 0 + 0.1572 + 0.1 + \dots + 0
 \end{aligned}$$

$$\begin{aligned}
&= 36.76429 \\
\sigma_x &= (0 * \sqrt{(1 - 36.76429)^2}) + (0.1572 * \sqrt{(1 - 36.76429)^2}) + \dots + (0 * \sqrt{(110 - 36.76429)^2}) \\
&= 0 + 5,62214 + \dots + 0 \\
&= 38.15837 \\
\sigma_y &= (0 * \sqrt{(1 - 36.76429)^2}) + (0.1572 * \sqrt{(2 - 36.76429)^2}) + \dots + (0 * \sqrt{(110 - 36.76429)^2}) \\
&= 0 + 5,4649 + \dots + 0 \\
&= 38.15837
\end{aligned}$$

Setelah didapatkan rata-rata dan standar deviasi hitung *correlation* dengan cara dibawah ini :

$$\begin{aligned}
Correlation &= \frac{((1 - 0) * 0) - (36.76429 - 36.76429)}{(38.15837 * 38.15837)} \\
&+ \frac{((1 - 1) * 0.1572) - (36.76429 - 36.76429)}{(38.15837 * 38.15837)} \\
&+ \dots \\
&+ \frac{((110 - 110) * 0) - (36.76429 - 36.76429)}{(38.15837 * 38.15837)} \\
&= -9450.9006
\end{aligned}$$

Sebelum melakukan perhitungan Sum Average, Sum Entropy dan Sum Variance terlebih dahulu dihitung  $P_{x+y}$ .

$$\begin{aligned}
P_{(4)} &= P_{(0+4)} + P_{(1+3)} + P_{(2+2)} + P_{(3+1)} + P_{(4+0)} \\
&= 0 + 0.028572 + 0.012987 + 0.0285714 + 0 \\
&= 0.12857
\end{aligned}$$

Hasil perhitungan  $P_{x+y}$  ditunjukkan pada Tabel 4.21 dibawah ini.

**Tabel 4.21 Nilai  $P_{x+y}$  Sudut 45°**

$x+y$	0	1	2	...	220	221
$P_{x+y}$	0	0	0.157143	...	0	0

- *Sum Average (AVER)*

$$\begin{aligned}
AVER &= (0 * 0) + (1 * 0) + (2 * 0.157143) + \dots + (220 * 0) \\
&\quad + (221 * 0) \\
&= 73.52857
\end{aligned}$$

- *Sum Entropy (SENT)*

$$\begin{aligned}
 SENT &= -(2 * 0.157143 * \log(0.157143)) + \dots + (220 * 0 * 0) \\
 &\quad + (221 * 0 * 0) \\
 &= 295.56034
 \end{aligned}$$

- Sum Variance (SVAR)

$$\begin{aligned}
 SVAR &= ((2 * 295.56034)^2 * 0.157143) + \dots \\
 &\quad + ((220 * 295.56034)^2 * 0) \\
 &\quad + ((221 * 295.56034)^2 * 0) \\
 &= 53679.95484
 \end{aligned}$$

- Entropy

$$\begin{aligned}
 Entropy &= -((0 * 0) + (0.15714 * \log(0.15714)) \\
 &\quad + (0.05 * \log(0.05)) + \dots + (0 * 0)) \\
 &= 3.95822
 \end{aligned}$$

Sebelum melakukan perhitungan *Difference Entropy* dan *Difference Variance*, hitung  $P_{x-y}$  dengan cara sebagai berikut :

$$\begin{aligned}
 P_{(0)} &= P_{(|0-0|)} + P_{(|1-1|)} + P_{(|2-2|)} + \dots + P_{(|110-110|)} \\
 &= 0 + 0.15715 + 0.0714285 + \dots + 0 \\
 &= 0.22857
 \end{aligned}$$

Hasil perhitungan  $P_{x-y}$  ditunjukkan pada Tabel 4.22 dibawah ini.

**Tabel 4.22 Nilai  $P_{x-y}$  Sudut 45°**

$x-y$	0	1	2	...	109	110
$P_{x-y}$	0.22857	0.1285	0.14285	...	0	0

- *Difference Entropy (DENT)*

$$\begin{aligned}
 DENT &= -((0 * 0.22857 * \log(0.22857)) \\
 &\quad + (1 * 0.1285 * \log(0.1285)) \\
 &\quad + (2 * 0.14285 * \log(0.14285)) + \dots \\
 &\quad + (110 * 0 * 0)) \\
 &= 101.36904
 \end{aligned}$$

- *Difference Variance (DVAR)*

$$\begin{aligned}
 DVAR &= ((0^2 * 0.22857) + (1^2 * 0.1285) + (2^2 * 0.14285) + \dots \\
 &\quad + (110^2 * 0)) \\
 &= 1843.2143
 \end{aligned}$$

Sebelum dilakukan perhitungan Information Measure of Correlatin 1 dan 2, terlebih dahulu dihitung nilai  $P_x, P_y, HX, HY, HXY1, HXY2$ .

Dibawah ini contoh perhitungan  $P_{(x=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.23.

**Tabel 4.23 Matriks Co-occurrence Simetris Baris ke 1 Sudut 0°**

x/y	0	1	2	...	110
1	0	0.15714	0.05	...	0

$$\begin{aligned}
 P_{(x=1)} &= P_{(1,0)} + P_{(1,1)} + P_{(1,2)} + \dots + P_{(1,110)} \\
 &= 0 + 0.15714 + 0.05 + \dots + 0 \\
 &= 0.27857
 \end{aligned}$$

Hasil perhitungan  $P_x$  dapat dilihat pada Tabel 4.24.

**Tabel 4.24 Nilai  $P_x$  sudut 0°**

x	0	1	2	...	110
$P_x$	0	0.27857	0.1857	...	0

Dibawah ini contoh perhitungan  $P_{(y=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.25.

y/x	0	1	2	...	110
1	0	0.15714	0.05	...	0

$$\begin{aligned}
 P_{(y=1)} &= P_{(0,1)} + P_{(1,1)} + P_{(2,1)} + \dots + P_{(110,1)} \\
 &= 0 + 0.27857 + 0.1857 + \dots + 0 \\
 &= 0.15714
 \end{aligned}$$

Hasil perhitungan  $P_y$  dapat dilihat pada Tabel 4.25.

**Tabel 4.25 Nilai  $P_y$  sudut 0°**

y	0	1	2	...	110
$P_y$	0	0.15714	0.05	...	0

$$\begin{aligned}
 HX &= -(0 * (0) + 0.15714 * \log(0.15714) + 0.05 * \log(0.05) + \dots \\
 &\quad + 0 * (0))
 \end{aligned}$$

$$= 274.83653$$

$$\begin{aligned}
 HY &= -(0 * (0) + 0.15714 * \log(0.15714) + 0.05 * \log(0.05) + \dots \\
 &\quad + 0 * (0))
 \end{aligned}$$

$$= 274.83653$$

$$\max\{HX, HY\} = 274.83653$$

$$\begin{aligned}
 HXY1 &= -(0 * \log(0 * 0) + 0.15714 * \log(0.15714 * 0.15714) + \dots \\
 &\quad + 0 * \log(0 * 0))
 \end{aligned}$$

$$= 4.997027$$

$$\begin{aligned}
 HXY2 &= -(0 * 0 * \log(0 * 0) + 0.15714 * 0.15714 \\
 &\quad * \log(0.15714 * 0.15714) + \dots + 0 * 0 * \log(0 * 0))
 \end{aligned}$$

$$= 4.7828694$$

- *Information Measure of Correlation 1*

$$IMoC1 = \frac{(3.95822 - 4.997027)}{274.83653} = -0.00377974$$

- *Information Measure of Correlation 2*

$$IMoC2 = \sqrt{|1 - \exp(-2 * (4.7828694 - 3.95822))|} = 0.898786$$

c. Sudut 90°

- Angular Second Moment (ASM)

$$\begin{aligned} ASM &= 0^2 + 0^2 + 0^2 + 0^2 + \dots + 0^2 \\ &= 0 + 0 + 0 + 0 + \dots + 0 \\ &= 0.06123 \end{aligned}$$

- Contrast

$$\begin{aligned} Contrast &= ((1 - 0)^2 * 0) + ((1 - 1)^2 * 0.1949) + ((1 - 2)^2 * 0.0909) + ((1 - 3)^2 * 0.03246) + \dots \\ &\quad + ((1 - 110)^2 * 0) \\ &= 0 + 0 + 0.0909 + 0,001 + \dots + 0 \\ &= 1346.61 \end{aligned}$$

- Sum of Squares: Variance

Sebelum dilakukan perhitungan *variance* akan dihitung dulu rata-rata dari matriks simetris. Perhitungan dilakukan sebagai berikut :

$$\mu = \frac{(0 + 0.1949 + 0.0909 + 0.03247 + \dots + 0)}{12321} = 0.00000812$$

Hitung *variance* dengan cara berikut :

$$\begin{aligned} Variance &= ((1 - 0.00000812) * 0)^2 + (2 - 0.00000812)^2 * 0.1948) + \dots + ((110 - 0.00000812)^2 * 0) \\ &= 0 + 0,7791 + \dots + 0 \\ &= 2759.67618 \end{aligned}$$

- Inverse Difference Moment (IDM)

$$\begin{aligned} DM &= \left( \frac{0^2}{1 + ((1 - 0)^2)} \right) + \left( \frac{(0.1949)^2}{1 + ((1 - 1)^2)} \right) + \dots \\ &\quad + \left( \frac{0^2}{1 + ((110 - 110)^2)} \right) \\ &= 0 + 0.0247 + \dots + 0 \end{aligned}$$

$$= 0.03798$$

- *Correlation*

Sebelum melakukan perhitungan *correlation* terlebih dahulu untuk menghitung rata-rata dari  $P_x$ ,  $P_y$  dan standar deviasi  $P_x$ ,  $P_y$ .

$$\begin{aligned}\mu_x &= (1 * 0) + (1 * 0.1949) + (1 * 0.0909) + \dots + (110 * 0) \\ &= 0 + 0.2078 + 0.0909 + \dots + 0 \\ &= 34.70779\end{aligned}$$

$$\begin{aligned}\mu_y &= (0 * 0) + (1 * 0.1949) + (2 * 0.0909) + \dots + (110 * 0) \\ &= 0 + 0.2078 + 0.1818 + \dots + 0 \\ &= 34.70779\end{aligned}$$

$$\begin{aligned}\sigma_x &= (0 * \sqrt{(1 - 34.70779)^2}) + (0.1949 * \sqrt{(1 - 34.70779)^2}) + \dots + (0 * \sqrt{(110 - 34.70779)^2}) \\ &= 0 + 6.5696 + \dots + 0 \\ &= 37.9128\end{aligned}$$

$$\begin{aligned}\sigma_y &= (0 * \sqrt{(1 - 34.70779)^2}) + (0.1949 * \sqrt{(2 - 34.70779)^2}) + \dots + (0 * \sqrt{(110 - 34.70779)^2}) \\ &= 0 + 5.7966 + \dots + 0 \\ &= 37.9128\end{aligned}$$

Setelah didapatkan rata-rata dan standar deviasi hitung *correlation* dengan cara dibawah ini :

$$\begin{aligned}Correlation &= \frac{((1 - 0) * 0) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\ &+ \frac{((1 - 1) * 0.1949) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\ &+ \dots \\ &+ \frac{((110 - 110) * 0) - (34.70779 - 34.70779)}{(37.9128 * 37.9128)} \\ &= -8239.52971\end{aligned}$$

Sebelum melakukan perhitungan Sum Average, Sum Entropy dan Sum Variance terlebih dahulu dihitung  $P_{x+y}$ .

$$\begin{aligned}P_{(4)} &= P_{(0+4)} + P_{(1+3)} + P_{(2+2)} + P_{(3+1)} + P_{(4+0)} \\ &= 0 + 0.0324675 + 0.02597402 + 0.0324675 + 0\end{aligned}$$

$$= 0.0909$$

Hasil perhitungan  $P_{x+y}$  ditunjukkan pada Tabel 4.26 dibawah ini.

**Tabel 4.26 Nilai  $P_{x+y}$  Sudut  $90^\circ$**

$x+y$	0	1	2	...	220	221
$P_{x+y}$	0	0	0.194805	...	0	0

- *Sum Average (AVER)*

$$\begin{aligned} AVER &= (0 * 0) + (1 * 0) + (2 * 0.194805) + \dots + (220 * 0) \\ &\quad + (221 * 0) \\ &= 69.41558 \end{aligned}$$

- *Sum Entropy (SENT)*

$$\begin{aligned} SENT &= -(2 * 0.194805 * \log(0.194805)) + \dots + (220 * 0 * 0) \\ &\quad + (221 * 0 * 0) \\ &= 288.22547 \end{aligned}$$

- *Sum Variance (SVAR)*

$$\begin{aligned} SVAR &= ((2 * 288.22547)^2 * 0.194805) + \dots \\ &\quad + ((220 * 288.22547)^2 * 0) \\ &\quad + ((221 * 288.22547)^2 * 0) \\ &= 52751.361 \end{aligned}$$

- *Entropy*

$$\begin{aligned} Entropy &= -((0 * 0) + (0.1948 * \log(0.1948)) \\ &\quad + (0.0909 * \log(0.0909)) + \dots + (0 * 0)) \\ &= 3.700025 \end{aligned}$$

Sebelum melakukan perhitungan *Difference Entropy* dan *Difference Variance*, hitung  $P_{x-y}$  dengan cara sebagai berikut :

$$\begin{aligned} P_{(0)} &= P_{(|0-0|)} + P_{(|1-1|)} + P_{(|2-2|)} + \dots + P_{(|110-110|)} \\ &= 0 + 0.15714 + 0.071428 + \dots + 0 \\ &= 0.2286 \end{aligned}$$

Hasil perhitungan  $P_{x-y}$  ditunjukkan pada Tabel 4.27 dibawah ini.

**Tabel 4.27 Nilai  $P_{x-y}$  Sudut  $90^\circ$**

$x-y$	0	1	2	...	109	110
$P_{x-y}$	0.2285	0.12857	0.0143	...	0	0

- *Difference Entropy (DENT)*

$$\begin{aligned}
DENT &= -((0 * 0.2285 * \log(0.2285)) \\
&\quad + (1 * 0.12857 * \log(0.12857)) \\
&\quad + (2 * 0.0143 * \log(0.0143)) + \dots + (110 * 0 * 0)) \\
&= 101.36904
\end{aligned}$$

- *Difference Variance (DVAR)*

$$\begin{aligned}
DVAR &= ((0^2 * 0.2285) + (1^2 * 0.12857) + (2^2 * 0.0143) + \dots \\
&\quad + (110^2 * 0)) \\
&= 1843.21428
\end{aligned}$$

Sebelum dilakukan perhitungan Information Measure of Correlatin 1 dan 2, terlebih dahulu dihitung nilai  $P_x, P_y, HX, HY, HXY1, HXY2$ .

Dibawah ini contoh perhitungan  $P_{(x=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.28.

**Tabel 4.28 Matriks Co-occurrence Simetris Baris ke 1 Sudut 0°**

x/y	0	1	2	...	110
1	0	0.1948	0.0909	...	0.

$$\begin{aligned}
P_{(x=1)} &= P_{(1,0)} + P_{(1,1)} + P_{(1,2)} + \dots + P_{(1,110)} \\
&= 0 + 0.1948 + 0.0909 + \dots + 0 \\
&= 0.33116
\end{aligned}$$

Hasil perhitungan  $P_x$  dapat dilihat pada Tabel 4.29.

**Tabel 4.29 Nilai  $P_x$  sudut 0°**

x	0	1	2	...	110
$P_x$	0	0.33116	0.162337	...	0

Dibawah ini contoh perhitungan  $P_{(y=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.30.

y/x	0	1	2	...	110
1	0	0.1948	0.0909	...	0.

$$\begin{aligned}
P_{(y=1)} &= P_{(0,1)} + P_{(1,1)} + P_{(2,1)} + \dots + P_{(110,1)} \\
&= 0 + 0.1948 + 0.0909 + \dots + 0 \\
&= 0.33116
\end{aligned}$$

Hasil perhitungan  $P_y$  dapat dilihat pada Tabel 4.30.

**Tabel 4.30 Nilai  $P_y$  sudut 0°**

y	0	1	2	...	110
---	---	---	---	-----	-----



$P_y$	0	0.33116	0.162337	...	0
-------	---	---------	----------	-----	---

$$HX = -(0 * (0) + 0.33116 * \log(0.33116) + 0.162337 * \log(0.162337) + \dots + 0 * (0))$$

$$= 260.83706$$

$$HY = -(0 * (0) + 0.33116 * \log(0.33116) + 0.162337 * \log(0.162337) + \dots + 0 * (0))$$

$$= 260.83706$$

$$\max\{HX, HY\} = 260.83706$$

$$HXY1 = -(0 * \log(0 * 0) + 0.1948 * \log(0.33116 * 0.33116) + \dots + 0 * \log(0 * 0))$$

$$= 4.7537005$$

$$HXY2 = -(0 * 0 * \log(0 * 0) + 0.33116 * 0.33116 * \log(0.33116 * 0.33116) + \dots + 0 * 0 * \log(0 * 0))$$

$$= 4.496128$$

- *Information Measure of Correlation 1*

$$IMoC1 = \frac{(3.700025 - 4.7537005)}{260.83706} = -0.0039966$$

- *Information Measure of Correlation 2*

$$IMoC2 = \sqrt{|1 - \exp(-2 * (4.496128 - 3.700025))|} = 0.892482$$

d. Sudut  $135^\circ$

- *Angular Second Moment (ASM)*

$$ASM = 0^2 + 0^2 + 0^2 + 0^2 + \dots + 0^2$$

$$= 0 + 0 + 0 + 0 + \dots + 0$$

$$= 0.06123$$

- *Contrast*

$$Contrast = ((1 - 0)^2 * 0) + ((1 - 1)^2 * 0.1885) + ((1 - 2)^2 * 0.058) + ((1 - 3)^2 * 0.0217) + \dots + ((1 - 110)^2 * 0)$$

$$= 0 + 0 + 0.058 + 0.00047 + \dots + 0$$

$$= 1524.333$$

- *Sum of Squares: Variance*

Sebelum dilakukan perhitungan *variance* akan dihitung dulu rata-rata dari matriks simetris. Perhitungan dilakukan sebagai berikut.

$$\mu = \frac{(0 + 0.2078 + 0.0909 + 0.0195 + \dots + 0)}{12321} = 0.00000812$$

Hitung *variance* dengan cara berikut.

$$\begin{aligned} \text{Variance} &= ((1 - 0.00000812) * 0)^2 + (2 - 0.00000812)^2 \\ &\quad * 0.1885) + \dots + ((110 - 0.00000812)^2 * 0) \\ &= 0 + 0.7539 + \dots + 0 \\ &= 2991.6968 \end{aligned}$$

- *Inverse Difference Moment (IDM)*

$$\begin{aligned} IDM &= \left( \frac{0^2}{1 + ((1 - 0)^2)} \right) + \left( \frac{(0.1885)^2}{1 + ((1 - 1)^2)} \right) + \dots \\ &\quad + \left( \frac{0^2}{1 + ((110 - 110)^2)} \right) \\ &= 0 + 0.0247 + \dots + 0 \\ &= 0.03553 \end{aligned}$$

- *Correlation*

Sebelum melakukan perhitungan *correlation* terlebih dahulu untuk menghitung rata-rata dari  $P_x$ ,  $P_y$  dan standar deviasi  $P_x$ ,  $P_y$ .

$$\begin{aligned} \mu_x &= (1 * 0) + (1 * 0.1885) + (1 * 0.0579) + \dots + (110 * 0) \\ &= 0 + 0.1885 + 0.0579 + \dots + 0 \\ &= 37.77536 \end{aligned}$$

$$\begin{aligned} \mu_y &= (0 * 0) + (1 * 0.1885) + (2 * 0.1885) + \dots + (110 * 0) \\ &= 0 + 0.1885 + 0.377 + \dots + 0 \\ &= 37.77536 \end{aligned}$$

$$\begin{aligned} \sigma_x &= (0 * \sqrt{(1 - 37.77536)^2}) + (0.1885 * \\ &\quad \sqrt{(1 - 37.77536)^2}) + \dots + (0 * \sqrt{(110 - 37.77536)^2}) \\ &= 0 + 6,9322 + \dots + 0 \\ &= 38.32756 \end{aligned}$$

$$\begin{aligned} \sigma_y &= (0 * \sqrt{(1 - 37.77536)^2}) + (0.1885 * \\ &\quad \sqrt{(2 - 37.77536)^2}) + \dots + (0 * \sqrt{(110 - 37.77536)^2}) \\ &= 0 + 6,7437 + \dots + 0 \\ &= 37.77536 \end{aligned}$$

Setelah didapatkan rata-rata dan standar deviasi hitung *correlation* dengan cara dibawah ini:

$$\begin{aligned}
 Correlation &= \frac{((1 - 0) * 0) - (37.77536 - 37.77536)}{(37.77536 * 37.77536)} \\
 &+ \frac{((1 - 1) * 0.1885) - (37.77536 - 37.77536)}{(37.77536 * 37.77536)} \\
 &+ \dots \\
 &+ \frac{((110 - 110) * 0) - (37.77536 - 37.77536)}{(37.77536 * 37.77536)} \\
 &= -9738.9976
 \end{aligned}$$

Sebelum melakukan perhitungan Sum Average, Sum Entropy dan Sum Variance terlebih dahulu dihitung  $P_{x+y}$ .

$$\begin{aligned}
 P_{(4)} &= P_{(0+4)} + P_{(1+3)} + P_{(2+2)} + P_{(3+1)} + P_{(4+0)} \\
 &= 0 + 0.021739 + 0 + 0.0217391304 + 0 \\
 &= 0.043478
 \end{aligned}$$

Hasil perhitungan  $P_{x+y}$  ditunjukkan pada Tabel 4.31 dibawah ini.

$x+y$	0	1	2	...	220	221
$P_{x+y}$	0	0	0.188406	...	0	0

**Tabel 4.31 Nilai  $P_{x+y}$  Sudut  $135^\circ$**

- *Sum Average (AVER)*

$$\begin{aligned}
 AVER &= (0 * 0) + (1 * 0) + (2 * 0.188406) + \dots + (220 * 0) \\
 &\quad + (221 * 0) \\
 &= 75.550724
 \end{aligned}$$

- *Sum Entropy (SENT)*

$$\begin{aligned}
 SENT &= -(2 * 0.188406 * \log(0.188406)) + \dots + (220 * 0 * 0) \\
 &\quad + (221 * 0 * 0) \\
 &= 305.25103
 \end{aligned}$$

- *Sum Variance (SVAR)*

$$\begin{aligned}
 SVAR &= ((2 * 305.25103)^2 * 0.188406) + \dots \\
 &\quad + ((220 * 305.25103)^2 * 0) \\
 &\quad + ((221 * 305.25103)^2 * 0) \\
 &= 57496.80048
 \end{aligned}$$

- *Entropy*

$$\begin{aligned}
Entropy &= -((0 * 0) + (0.1884 * \log(0.1884)) \\
&\quad + (0.05797 * \log(0.05797)) + \dots + (0 * 0)) \\
&= 3.87386
\end{aligned}$$

Sebelum melakukan perhitungan *Difference Entropy* dan *Difference Variance*, hitung  $P_{x-y}$  dengan cara sebagai berikut.

$$\begin{aligned}
P_{(0)} &= P_{(0-0)} + P_{(1-1)} + P_{(2-2)} + \dots + P_{(110-110)} \\
&= 0 + 0.1884 + 0 + \dots + 0 \\
&= 0.20289
\end{aligned}$$

Hasil perhitungan  $P_{x-y}$  ditunjukkan pada Tabel 4.32 dibawah ini.

**Tabel 4.32 Nilai  $P_{x-y}$  Sudut 135°**

$x-y$	0	1	2	...	109	110
$P_{x-y}$	0.20289	0.20289	0.04347	...	0.0145	0

- *Difference Entropy (DENT)*

$$\begin{aligned}
DENT &= -((0 * 0.20289 * \log(0.20289)) \\
&\quad + (1 * 0.20289 * \log(0.20289)) \\
&\quad + (2 * 0.04347 * \log(0.04347)) + \dots \\
&\quad + (110 * 0 * 0)) \\
&= 89.4661
\end{aligned}$$

- *Difference Variance (DVAR)*

$$\begin{aligned}
DVAR &= ((0^2 * 0.20289) + (1^2 * 0.20289) + (2^2 * 0.04347) + \dots \\
&\quad + (110^2 * 0)) \\
&= 1524.3333
\end{aligned}$$

Sebelum dilakukan perhitungan Information Measure of Correlatin 1 dan 2, terlebih dahulu dihitung nilai  $P_x, P_y, HX, HY, HXY1, HXY2$ .

Dibawah ini contoh perhitungan  $P_{(x=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.33.

**Tabel 4.33 Matriks Co-occurrence Simetris Baris ke 1 Sudut 135°**

$x/y$	0	1	2	...	110
1	0	0.1884	0.05797	...	0.0072

$$\begin{aligned}
P_{(x=1)} &= P_{(1,0)} + P_{(1,1)} + P_{(1,2)} + \dots + P_{(1,110)} \\
&= 0 + 0.1884 + 0.0579 + \dots + 0.0072 \\
&= 0.304348
\end{aligned}$$

Hasil perhitungan  $P_x$  dapat dilihat pada Tabel 4.34.

**Tabel 4.34 Nilai  $P_x$  sudut  $0^\circ$**

x/y	0	1	2	...	110
1	0	0.30435	0.1449	...	0

Dibawah ini contoh perhitungan  $P_{(x=1)}$  dengan matriks simetris baris ke 1 dapat dilihat pada Tabel 4.35.

x/y	0	1	2	...	110
1	0	0.1884	0.05797	...	0.0072

$$\begin{aligned}
 P_{(y=1)} &= P_{(0,1)} + P_{(1,1)} + P_{(2,1)} + \dots + P_{(110,1)} \\
 &= 0 + 0.1884 + 0.05797 + \dots + 0.0072 \\
 &= 0.304348
 \end{aligned}$$

Hasil perhitungan  $P_y$  dapat dilihat pada Tabel 4.35.

**Tabel 4.35 Nilai  $P_y$  sudut  $0^\circ$**

x/y	0	1	2	...	110
1	0	0.30435	0.1449	...	0

$$\begin{aligned}
 HX &= -(0 * (0) + 0.30435 * \log(0.30435) + 0.1449 * \log(0.1449) \\
 &\quad + \dots + 0 * (0)) \\
 &= 275.10807
 \end{aligned}$$

$$\begin{aligned}
 HY &= -(0 * (0) + 0.30435 * \log(0.304356) + 0.1449 * \\
 &\quad * \log(0.1449) + \dots + 0 * 0 * (0)) \\
 &= 275.10807
 \end{aligned}$$

$$\max\{HX, HY\} = 275.10807$$

$$\begin{aligned}
 HXY1 &= -(0 * \log(0 * 0) + 0.1884 * \log(0.30435 * 0.30435) + \dots \\
 &\quad + 0 * \log(0 * 0)) \\
 &= 5.001964
 \end{aligned}$$

$$\begin{aligned}
 HXY2 &= -(0 * 0 * \log(0 * 0) + 0.30435 * 0.30435 \\
 &\quad * \log(0.30435 * 0.30435) + \dots + 0 * 0 * \log(0 * 0)) \\
 &= 4.78448
 \end{aligned}$$

- *Information Measure of Correlation 1*

$$IMoC1 = \frac{(3.87386 - 5.001964)}{275.10807} = -0.0041005$$

- *Information Measure of Correlation 2*

$$IMoC2 = \sqrt{|1 - \exp(-2 * (4.78448 - 3.87386))|} = 0.915519$$

Tabel 4.36 Data Latih

No	Kode Makanan	Nama Makanan	0 Derajat				
			ASM	Contrast	Variance	IDM	Correlation
1	001_0001_XiaomiRedmi3Pro.jpg	Donat	0,001160236	154,0178556	12108,83952	0,000847332	-449641,9875
2	001_0002_XiaomiRedmi3Pro.jpg	Donat	0,001176186	145,4950398	11621,89689	0,000854336	-433506,3053
3	001_0004_XiaomiRedmi3Pro.jpg	Donat	0,001272376	138,0402589	14338,4657	0,000845171	-593410,1793
4	001_0005_XiaomiRedmi3Pro.jpg	Donat	0,001260446	132,8750899	14399,45517	0,000829043	-622305,7608
5	001_0006_XiaomiRedmi3Pro.jpg	Donat	0,001617893	153,1487307	19499,53432	0,001089566	-1009811,155
6	001_0007_XiaomiRedmi3Pro.jpg	Donat	0,001621127	150,5773775	19443,83621	0,001092818	-1022539,292
7	009_0204_XiaomiRedmi3Pro.jpg	Rendang	0,001849989	143,4823191	5864,160598	0,001053206	-1095777,419
8	009_0205_XiaomiRedmi3Pro.jpg	Rendang	0,00180035	158,6146908	5850,709512	0,001050192	-1174671,273
9	009_0207_XiaomiRedmi3Pro.jpg	Rendang	0,00200361	108,6515827	6224,058117	0,001080088	-1498445,377
10	009_0208_XiaomiRedmi3Pro.jpg	Rendang	0,002392528	93,64505651	6227,225363	0,001392113	-1385294,896
11	009_0210_XiaomiRedmi3Pro.jpg	Rendang	0,002024123	116,4574916	6247,721714	0,001104209	-1350096,44
12	009_0211_XiaomiRedmi3Pro.jpg	Rendang	0,002422684	140,5919372	8542,792947	0,001522093	-1689733,716
13	009_0212_XiaomiRedmi3Pro.jpg	Rendang	0,002475351	144,0676519	8531,68697	0,001579873	-1786233,277
14	007_0160_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001000937	255,4883237	11519,68874	0,00067126	-1001207,393
15	007_0161_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001019917	248,900653	11517,85779	0,000687591	-996797,8631
16	007_0162_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001020119	172,4882463	11914,3773	0,000651531	-700015,0113
17	007_0163_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001009999	174,4247652	11957,4485	0,000641364	-723198,2223
18	007_0164_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001220262	158,9091324	11537,68921	0,000831188	-660981,7813
19	007_0165_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001329343	147,4105411	12882,30643	0,000859784	-1139762,174
20	007_0166_XiaomiRedmi3Pro.jpg	Telur Dadar	0,001321209	151,7390631	12882,52058	0,000858992	-1159266,533

0 Derajat							
Average	Sum Entropy	Sum Variance	Entropy	DEntropy	DVariance	IMoC1	IMoC2
205,5310649	1160,288266	917599,6545	7,844243967	24,36472833	-154,0178556	-0,001931074	0,989026799
201,0372412	1133,608654	875615,5805	7,801633694	23,42043928	-145,4950398	-0,001963664	0,989411921
226,5834174	1247,487758	1048121,472	7,431466237	17,68645929	-138,0402589	-0,002234018	0,992828521
227,1232816	1249,853767	1051857,627	7,440245659	17,59461669	-132,8750899	-0,002177299	0,992634192
268,2640098	1418,407119	1328708,606	7,164051109	19,10231809	-153,1487307	-0,002019111	0,989843977
267,7915047	1415,759955	1323744,064	7,15996403	18,69805784	-150,5773775	-0,002001155	0,989844404
147,024096	692,3672808	299096,2794	7,128302788	26,44037478	-143,4823191	-0,001153921	0,916133589
146,7496784	692,0643002	299076,8058	7,223831182	29,08969576	-158,6146908	-0,001008182	0,901403862
152,7215462	708,8888935	310785,8433	6,939382961	21,73080585	-108,6515827	-0,001254083	0,9299805
152,7184043	706,8329675	308535,3107	6,765631323	18,32955885	-93,64505651	-0,001480833	0,946166963
152,7261852	712,5139973	314911,5522	6,949462301	22,19301421	-116,4574916	-0,001308758	0,93223081
177,9131056	839,5052873	440081,7354	6,891164314	21,51992502	-140,5919372	-0,001306003	0,952897322
177,6112946	838,4022335	439081,5858	6,893556512	21,86142196	-144,0676519	-0,001248624	0,952716591
203,1561033	1104,15914	816357,3493	8,070636078	36,86337942	-255,4883237	-0,001087057	0,960164336
203,114782	1103,526965	815309,028	8,055220598	35,9923804	-248,900653	-0,001097992	0,961160779
205,3059673	1141,947528	882631,9086	7,82481663	24,82542531	-172,4882463	-0,001511035	0,983688681
205,7230698	1143,694729	885124,2343	7,835688473	25,28741038	-174,4247652	-0,001480306	0,983210799
201,6351123	1118,717581	846375,3983	7,719477595	22,65189042	-158,9091324	-0,00159793	0,985535429
217,2587931	1144,96095	864811,7413	7,506538965	21,91278271	-147,4105411	-0,001534485	0,977951669
217,1193059	1145,296183	865749,8812	7,538113325	22,62704426	-151,7390631	-0,00148448	0,976921405

**Tabel 4.37 Data Uji**

No	Kode Makanan	Nama Makanan	0 Derajat				
			ASM	Contrast	Variance	IDM	Correlation
1	001_0003_XiaomiRedmi3Pro.jpg	Donat	0,001335664	136,1292339	14305,13717	0,000906297	-582739,0586
2	009_0206_XiaomiRedmi3Pro.jpg	Rendang	0,001631753	158,9212693	5912,582329	0,000898251	-1346835,995
3	007_0159_XiaomiRedmi3Pro.jpg	Telor Dadar	0,001053767	249,9393662	11524,77265	0,000722993	-937407,2208

0 Derajat							
Average	Sum Entropy	Sum Variance	Entropy	DEntropy	DVariance	IMoC1	IMoC2
225,9437931	1245,151786	1044818,778	7,436467358	17,75053125	-136,1292339	-0,002216931	0,992743965
147,8022008	697,5624503	303882,262	7,264864869	29,47613712	-158,9212693	-0,00093928	0,897821418
203,2593231	1103,488905	814948,1136	8,056450194	36,34292415	-249,9393662	-0,001131086	0,960364564



### 4.2.3 Perhitungan *Information Gain*

Untuk melakukan manualisasi *Information Gain* pada citra makanan, dataset yang digunakan sebanyak 23 data yang diambil dari dataset asli. Data latih yang digunakan sebanyak 20 data dan terdapat 3 kelas yaitu kelas Donat, Rendang dan Telor Dadar. Data uji yang digunakan sebanyak 3 data dengan kelas Donat, Rendang dan Telor Dadar. Fitur yang digunakan terdapat 13 fitur yang didapatkan dari fitur *GLCM 0 Derajat* Tabel 4.36 menunjukkan data latih yang digunakan untuk perhitungan manual dan Tabel 4.37 merupakan data uji.

#### Langkah 1. Menghitung Entropy

Perhitungan entropy dapat dihitung menggunakan Persamaan (2.6). Terdapat dua kelas dalam perhitungan manual entropy, yaitu kelas Donat, Rendang dan Telor Dadar.

$$\begin{aligned}
 Entropy(S) &= \sum_i^c -P_i \log_2 P_i \\
 &= (-P_{Donat} \log_2 P_{Donat}) + (-P_{Rendang} \log_2 P_{Rendang}) \\
 &\quad + (-P_{TelorDadar} \log_2 P_{TelorDadar}) \\
 &= \left(-\frac{6}{20} \log_2 \frac{6}{20}\right) + \left(-\frac{7}{20} \log_2 \frac{7}{20}\right) + \left(-\frac{7}{20} \log_2 \frac{7}{20}\right) \\
 &= 1,58129
 \end{aligned}$$

#### Langkah 2. Mengubah Data Tunggal ke Data Kelompok

Tentukan jangkauan dengan mencari selisih antara data terbesar dan data terkecil. Dibawah ini contoh data dari fitur *ASM*.

$$J = 0,002475351 - 0,001000937 = 0,0014744$$

Tentukan banyaknya banyaknya kelas yang akan dibentuk dengan aturan *Sturges*.

$$k = 1 + 3,3 \log 20 = 1 + 3,3 \cdot 1,301 = 5,2933 \approx 5$$

Tentukan panjang kelas atau interval kelas

$$i = \frac{J}{k} = \frac{0,0014744}{5} = 0,0002949$$

Masukkan data tunggal ke dalam bentuk tabel dan tentukan masing-masing data ke dalam kelas makanan.

**Tabel 4.38 Tabel Data Kelompok**

No	Kelas	Frekuensi		
		Donat	Rendang	Telor Dadar
1	0,00100094 – 0,0012958	4	0	5
2	0,0012959 – 0,0015908	0	0	2

3	0,0015909 – 0,0018857	2	2	0
4	0,0018858 – 0,0021806	0	2	0
5	0,0021807 – 0,0024755	0	3	0

### Langkah 3. Menghitung Nilai *Information Gain* dari Masing-Masing Atribut

Pada tahap ini dilakukan perhitungan *entropy* dari masing-masing atribut berdasarkan kelas yang sudah dikelompokkan sebelumnya lalu menghitung nilai gain dengan Persamaan 2.19.

Dibawah ini contoh perhitungan *entropy* menggunakan fitur ASM.

$$\begin{aligned}
 Entropy(S_{Kelas1}) &= \left( -\frac{4}{4+0+5} \log_2 \frac{4}{4+0+5} \right) + \left( -\frac{0}{4+0+5} \log_2 \frac{0}{4+0+5} \right) \\
 &+ \left( -\frac{5}{4+0+5} \log_2 \frac{5}{4+0+5} \right) = 0,99107606
 \end{aligned}$$

$$\begin{aligned}
 Entropy(S_{Kelas2}) &= \left( -\frac{0}{0+0+2} \log_2 \frac{0}{0+0+2} \right) + \left( -\frac{0}{0+0+2} \log_2 \frac{0}{0+0+2} \right) \\
 &+ \left( -\frac{2}{0+0+2} \log_2 \frac{2}{0+0+2} \right) = 0
 \end{aligned}$$

$$\begin{aligned}
 Entropy(S_{Kelas3}) &= \left( -\frac{2}{2+2+0} \log_2 \frac{2}{2+2+0} \right) + \left( -\frac{2}{2+2+0} \log_2 \frac{2}{2+2+0} \right) \\
 &+ \left( -\frac{0}{2+2+0} \log_2 \frac{0}{2+2+0} \right) = 1
 \end{aligned}$$

$$\begin{aligned}
 Entropy(S_{Kelas4}) &= \left( -\frac{0}{0+2+0} \log_2 \frac{0}{0+2+0} \right) + \left( -\frac{2}{0+2+0} \log_2 \frac{2}{0+2+0} \right) \\
 &+ \left( -\frac{0}{0+2+0} \log_2 \frac{0}{0+2+0} \right) = 0
 \end{aligned}$$

$$\begin{aligned}
 Entropy(S_{Kelas5}) &= \left( -\frac{0}{0+3+0} \log_2 \frac{0}{0+3+0} \right) + \left( -\frac{3}{0+3+0} \log_2 \frac{3}{0+3+0} \right) \\
 &+ \left( -\frac{0}{0+3+0} \log_2 \frac{0}{0+3+0} \right) = 0
 \end{aligned}$$

Setelah didapatkan *entropy* dari masing-masing atribut di dalam kelas dapat dilakukan perhitungan *Information Gain* dari atribut ASM.

$$\begin{aligned}
Gain(S, ASM) &= 1,58129 \\
&- \left\{ \left( \left( \frac{4+0+5}{20} \right) x 0,99107606 \right) + \left( \left( \frac{0+0+2}{20} \right) x 0 \right) \right. \\
&+ \left( \left( \frac{2+2+0}{20} \right) x 1 \right) + \left( \left( \frac{0+2+0}{20} \right) x 0 \right) + \left. \left( \left( \frac{0+3+0}{20} \right) x 0 \right) \right\} \\
&= 0,9353067
\end{aligned}$$

#### Langkah 4. Mengurutkan Hasil Information Gain

Urutkan hasil *Information Gain* dari yang terbesar ke terkecil untuk selanjutnya akan dipilih hasil tersebut untuk proses klasifikasi.

**Tabel 4.39 Pengurutan Hasil *Information Gain***

No	Fitur/Atribut	Hasil
1	OSum Variance	1,36386512
2	OVariance	1,23739887
3	OAverage	1,23739887
4	OSum Entropy	1,18129090
5	OIMoC2	1,08129090
6	OIMoC1	1,06306450
7	OASM	0,93530667
8	OEntropy	0,90080888
9	OIDM	0,82532013
10	OCorrelation	0,73855325
11	OContrast	0,62958101
12	ODVariance	0,62958101
13	ODEntropy	0,57860351

#### Langkah 5. Menentukan jumlah fitur untuk proses klasifikasi

Tentukan jumlah atribut dari hasil *Information Gain* terbesar ke terkecil untuk selanjutnya akan dilakukan proses klasifikasi. Dalam perhitungan manual ini dipilih jumlah atribut sebanyak 10 maka akan menghasilkan atribut OSum Variance, OVariance, OAverage, OSum Entropy, OIMoC2, OIMoC1, OASM, OEntropy, OIDM, OCorrelation.

#### 4.2.4 Perhitungan KNN

Perhitungan manual klasifikasi *K-Nearest Neighbor* (KNN) dilakukan dengan dataset sebanyak 20 dengan 13 fitur seperti pada Tabel 4.36 dan data uji sebanyak 3. Terdapat 3 kelas yang terdapat pada data latih yaitu kelas donat, rendang dan telur dadar.

### Langkah 1. Menentukan nilai K

Tentukan manualisasi metode NWKNN menggunakan K sebesar 3

### Langkah 2. Menghitung jarak ketetangaan

Pada langkah ini akan dilakukan perhitungan jarak ketetangaan menggunakan 3 metode yaitu *Manhattan Distance*, *Euclidean Distance* dan *Cosine Similarity*.

Pada perhitungan menggunakan metode *Cosine Similarity* rumus ditunjukkan pada persamaan 2.. Berikut contoh perhitungan manual antara data uji ke – 1 dengan data latih ke – 1.

$$d(q, d_j) = \frac{\vec{d_j} \circ \vec{q}}{|\vec{d_j}| \circ |\vec{q}|} = \frac{\sum_{i=1}^m w_{ij} \circ w_{iq}}{\sqrt{\sum_{i=1}^m w_{ij}^2 \circ \sum_{i=1}^m w_{iq}^2}}$$

Dalam kasus ini  $w_{ij}$  merupakan nilai dari setiap fitur pada data latih mulai dari  $i = 1$  sampai  $i = 13$  (dari fitur pertama sampai fitur ke tiga belas) dengan data latih ke  $j = 1$  yaitu data latih pertama,  $w_{iq}$  merupakan nilai dari setiap fitur pada data uji mulai dari  $i = 1$  sampai  $i = 13$  (dari fitur pertama sampai fitur ke tiga belas) dengan data uji ke  $q = 1$ .

$$\begin{aligned}\vec{d_j} \circ \vec{q} &= ((0,001160236 * 0,001335664) + (154,0178556 * 136,1292339) \\ &\quad + (12108,839 * 14305,13717) + \dots + (0,9890268 * 0,99274396)) \\ &= 1220924050230,871\end{aligned}$$

$$\begin{aligned}|\vec{d_j}| \circ |\vec{q}| &= \sqrt{0,001160236^2 + 154,0178556^2 + 12108,839^2 + \dots + 0,9890268^2} \\ &\quad \times \\ &\quad \sqrt{0,001335664^2 + 136,1292339^2 + 14305,13717^2 + \dots + 0,9927439^2} \\ &= 1222649442647,6794\end{aligned}$$

$$d(q, d_j) = \frac{1220924050230,871}{1222649442647,6794} = 0,99858881$$

Hasil keseluruhan perhitungan *cosim* antara data uji dengan data latih dapat dilihat pada tabel 4.40.

**Tabel 4.40 Hasil *Cosine Similarity***

No	Data Uji 1 ke Data Latih		Data Uji 2 ke Data Latih		Data Uji 3 ke Data Latih	
	Hasil Cosim	Kelas	Hasil Cosim	Kelas	Hasil Cosim	Kelas
1	0,998589	Donat	0,626882	Donat	0,921253	Donat
2	0,998797	Donat	0,630054	Donat	0,922831	Donat
3	0,999980	Donat	0,672124	Donat	0,942762	Donat
4	0,999676	Donat	0,686119	Donat	0,948948	Donat

5	0,990065	Donat	0,765456	Donat	0,978999	Donat
6	0,988930	Donat	0,770486	Donat	0,980570	Donat
7	0,699885	Rendang	0,999007	Rendang	0,900804	Rendang
8	0,687526	Rendang	0,999624	Rendang	0,893220	Rendang
9	0,654306	Rendang	0,999848	Rendang	0,872182	Rendang
10	0,665310	Rendang	0,999996	Rendang	0,879251	Rendang
11	0,672750	Rendang	0,999974	Rendang	0,883976	Rendang
12	0,691493	Rendang	0,999459	Rendang	0,895669	Rendang
13	0,681493	Rendang	0,999817	Rendang	0,889469	Rendang
14	0,929417	Telor Dadar	0,895092	Telor Dadar	0,999501	Telor Dadar
15	0,929982	Telor Dadar	0,894407	Telor Dadar	0,999548	Telor Dadar
16	0,986948	Telor Dadar	0,778588	Telor Dadar	0,983003	Telor Dadar
17	0,984503	Telor Dadar	0,787628	Telor Dadar	0,985567	Telor Dadar
18	0,988128	Telor Dadar	0,773860	Telor Dadar	0,981598	Telor Dadar
19	0,915955	Telor Dadar	0,910126	Telor Dadar	0,997787	Telor Dadar
20	0,912865	Telor Dadar	0,913262	Telor Dadar	0,997251	Telor Dadar

### Langkah 3. Mengurutkan hasil perhitungan jarak ketetanggaan

Setelah mendapatkan hasil dari perhitungan *Cosim* antara data uji dan data latih maka urutkan hasil perhitungan mulai dari yang terbesar sampai yang terkecil. Semakin besar nilainya maka semakin dekat atau semakin mirip jarak ketetanggannya. Pada Tabel 4.41. adalah hasil dari *Cosim* yang telah diurutkan.

**Tabel 4.41 Hasil *Cosine Similarity* terurut**

No	Data Uji 1 ke Data Latih		Data Uji 2 ke Data Latih		Data Uji 3 ke Data Latih	
	Hasil Cosim	Kelas	Hasil Cosim	Kelas	Hasil Cosim	Kelas
1	0,999980	Donat	0,999996	Rendang	0,999548	Telor Dadar
2	0,999676	Donat	0,999974	Rendang	0,999501	Telor Dadar
3	0,998797	Donat	0,999848	Rendang	0,997787	Telor Dadar
4	0,998589	Donat	0,999817	Rendang	0,997251	Telor Dadar

5	0,990065	Donat	0,999624	Rendang	0,985567	Telor Dadar
6	0,988930	Donat	0,999459	Rendang	0,983003	Telor Dadar
7	0,988128	Telor Dadar	0,999007	Rendang	0,981598	Telor Dadar
8	0,986948	Telor Dadar	0,913262	Telor Dadar	0,980570	Donat
9	0,984503	Telor Dadar	0,910126	Telor Dadar	0,978999	Donat
10	0,929982	Telor Dadar	0,895092	Telor Dadar	0,948948	Donat
11	0,929417	Telor Dadar	0,894407	Telor Dadar	0,942762	Donat
12	0,915955	Telor Dadar	0,787628	Telor Dadar	0,922831	Donat
13	0,912865	Telor Dadar	0,778588	Telor Dadar	0,921253	Donat
14	0,699885	Rendang	0,773860	Telor Dadar	0,900804	Rendang
15	0,691493	Rendang	0,770486	Donat	0,895669	Rendang
16	0,687526	Rendang	0,765456	Donat	0,893220	Rendang
17	0,681493	Rendang	0,686119	Donat	0,889469	Rendang
18	0,672750	Rendang	0,672124	Donat	0,883976	Rendang
19	0,665310	Rendang	0,630054	Donat	0,879251	Rendang
20	0,654306	Rendang	0,626882	Donat	0,872182	Rendang

#### Langkah 4. Bentuk kelompok berdasarkan nilai ketetanggaan terdekat

Dari hasil pengurutan pada Tabel 4.42 maka pilih sebanyak K yaitu 7 yang akan dikelompokkan berdasarkan kelasnya.

**Tabel 4.42 Kelompok Nilai *Cosim* dengan K =7**

Data Uji ke -	Jumlah Kelas		
	Donat	Rendang	Telor Dadar
1	6	0	1
2	0	7	0
3	0	0	7

#### Langkah 5. Pilih kelompok yang paling sering muncul

Dari tabel 4.41 data yang paling sering banyak muncul pada data uji 1 adalah Donat maka akan dipilih kelas Donat untuk data uji 1 sebagai prediksi.

Data uji 2 yang paling sering muncul adalah Rendang maka akan dipilih kelas Rendang untuk data uji 2 sebagai prediksi. Data uji 3 yang paling sering muncul adalah Telor Dadar maka akan dipilih kelas Telor Dadar untuk data uji 3 sebagai prediksi.

#### 4.2.5 Perhitungan NWKNN

Pada perhitungan *Neighbor Weighted K-Nearest Neighbor (NWKNN)* hampir sama dengan metode KNN. Perbedaan yaitu dengan melakukan pembobotan dari setiap kelas pada data latih. Untuk itu perhitungan manual NWKNN dilakukan dengan melanjutkan perhitungan manual KNN pada langkah ke 3 dengan Tabel 4.40.

##### Langkah 1. Pembobotan setiap kelas

Pembobotan dilakukan pada setiap kelas yang ada pada data latih. Kelas mayoritas akan diberi bobot lebih kecil dan kelas minoritas akan diberi bobot lebih besar. Terdapat kelas yang ada pada data latih yaitu kelas Donat dengan jumlah 6, kelas Rendang dengan jumlah 7 dan kelas Telor Dadar dengan jumlah 7. Berikut contoh perhitungan manual pembobotan setiap kelas dengan eksponen adalah 4.

$$Weight_{Donat} = \frac{1}{\left(\frac{6}{6}\right)^{1/4}} = 1$$

$$Weight_{Rendang} = \frac{1}{\left(\frac{7}{6}\right)^{1/4}} = 0,96219$$

$$Weight_{TelorDadar} = \frac{1}{\left(\frac{7}{6}\right)^{1/4}} = 0,96219$$

##### Langkah 2. Perhitungan Skor

Setelah dilakukan pembobotan dan mendapatkan nilai bobot dari masing-masing kelas, kemudian lakukan perhitungan skor pada setiap jenis kelas makanan yang termasuk K tetangga terdekat untuk mengetahui hasil kelas dari data uji. Berikut pada Tabel 4.43 hasil Cosim yang telah diurutkan dengan K = 7.

**Tabel 4.43 Nilai Cosim terurut dengan K = 7**

No	Data Uji 1 ke Data Latih		Data Uji 2 ke Data Latih		Data Uji 3 ke Data Latih	
	Hasil Cosim	Kelas	Hasil Cosim	Kelas	Hasil Cosim	Kelas
1	0,999980	Donat	0,999996	Rendang	0,999548	Telor Dadar
2	0,999676	Donat	0,999974	Rendang	0,999501	Telor Dadar

3	0,998797	Donat	0,999848	Rendang	0,997787	Telor Dadar
4	0,998589	Donat	0,999817	Rendang	0,997251	Telor Dadar
5	0,990065	Donat	0,999624	Rendang	0,985567	Telor Dadar
6	0,988930	Donat	0,999459	Rendang	0,983003	Telor Dadar
7	0,988128	Telor Dadar	0,999007	Rendang	0,981598	Telor Dadar

Pada data uji ke 1 dengan K = 7 terdapat hanya kelas Donat dan Telor Dadar, maka skor yang dihitung untuk data uji ke 1 adalah skor untuk kelas Donat dan Telor Dadar.

$$\begin{aligned}
 Skor_{Donat} &= 1 \\
 &\quad * ((0,999980 * 1) + (0,999676 * 1) + (0,998797 * 1) \\
 &\quad + (0,998589 * 1) + (0,990065 * 1) + (0,988930 * 1) \\
 &\quad + (0,988128 * 0)) = 5,976036
 \end{aligned}$$

$$\begin{aligned}
 Skor_{TelorDadar} &= 0,96219 \\
 &\quad * ((0,999980 * 0) + (0,999676 * 0) + (0,998797 * 0) \\
 &\quad + (0,998589 * 0) + (0,990065 * 0) + (0,988930 * 0) \\
 &\quad + (0,988128 * 1)) = 0,950772
 \end{aligned}$$

Berdasarkan perhitungan skor pada masing-masing data uji maka dapat dilihat pada data uji ke 1 mengidentifikasi kelas Donat dan pada data uji ke 2 mengidentifikasi kelas Rendang. Berikut Tabel 4.44 hasil perhitungan skor data uji terhadap data latih.

**Tabel 4.44 Hasil Perhitungan Skor**

No	Data uji 1 ke data latih				
	Kelas	Skor	Data Asli	Hasil Identifikasi	Kevalidan
1	Donat	5,976036	Donat	Donat	Valid
2	Telor Dadar	0,950772			
No	Data uji 2 ke data latih				
	Kelas	Skor	Data Asli	Hasil Identifikasi	Kevalidan
1	Rendang	6,73318	Rendang	Rendang	Valid



No	Data uji 3 ke data latih				
	Kelas	Skor	Data Asli	Hasil Identifikasi	Kevalidan
1	Telor Dadar	6,681731	Telor Dadar	Telor Dadar	Valid

### 4.3 Perancangan Pengujian

Pengujian pada penelitian ini digunakan untuk menguji terkait algoritme yang digunakan agar dapat memberikan keluaran berupa nama makanan. Pengujian yang dilakukan antara lain pengaruh nilai K pada seleksi fitur *Information Gain* terhadap akurasi NWKNN, perbandingan perhitungan jarak *Manhattan Distance*, *Euclidean Distance* dan *Cosine Similarity* terhadap NWKNN, pengaruh nilai K NWKNN terhadap akurasi, perbandingan antara KNN dan NWKNN, pengujian *K-Fold Cross Validation*.

#### 4.3.1 Pengaruh Jumlah Fitur pada Seleksi Fitur *Information Gain*

Pengujian ini bertujuan untuk melihat pengaruh jumlah fitur pada *Information Gain* terhadap akurasi NWKNN. Hasil perancangan pengujian akan ditampilkan pada tabel 4.45.

**Tabel 4.45 Rancangan Pengujian Pengaruh Jumlah Fitur *Information Gain***

Jumlah Data Uji	Jumlah Fitur	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	5	5	Cosine Similarity	
	10			
	15			
	20			
	25			
	30			
	35			
	40			
	45			
	50			

#### 4.3.2 Pengaruh Nilai K NKWNN terhadap Akurasi

Nilai K atau nilai ketetanggaan merupakan salah satu nilai yang mempengaruhi hasil akurasi metode NWKNN. Hasil perancangan pengujian akan ditampilkan pada tabel 4.46.

**Tabel 4.46 Rancangan Pengujian Pengaruh Nilai K NWKNN**

Jumlah Data Uji	Nilai K Information Gain	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	10	3	Cosine Similarity	
		5		
		7		
		9		
		11		
		13		
		15		
		17		
		19		
		21		

#### 4.3.3 Perbandingan Perhitungan Jarak NWKNN

Pengujian ini dilakukan untuk mengetahui keoptimalan hasil akurasi metode NWKNN dengan perhitungan jarak *Cosine Similarity*, *Euclidean Distance* dan *Manhattan Distance*. Hasil perancangan pengujian akan ditampilkan pada Tabel 4.47.

**Tabel 4.47 Rancangan Pengujian Perbandingan Perhitungan Jarak**

Jumlah Data Uji	Jumlah Fitur	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	15	3	<i>Cosine Similarity</i>	
			<i>Euclidean Distance</i>	
			<i>Manhattan Distance</i>	
23	15	5	<i>Cosine</i>	

			<i>Similarity</i>	
			<i>Euclidean Distance</i>	
			<i>Mahnattan Distance</i>	
23	15	5	<i>Cosine Similarity</i>	
			<i>Euclidean Distance</i>	
			<i>Mahnattan Distance</i>	

#### 4.3.4 Perbandingan Akurasi Metode KNN dan NWKNN

Pengujian ini dilakukan untuk mengetahui perbandingan hasil akurasi antara KNN dan NWKNN. Hasil perancangan pengujian akan ditampilkan pada Tabel 4.48.

**Tabel 4.48 Rancangan Pengujian Perbandingan Akurasi KNN dan NWKNN**

Jumlah Data Uji	Jumlah Fitur	Nilai K	Perhitungan Jarak	Hasil Akurasi KNN (%)	Hasil Akurasi NWKNN (%)
23	15	3	Cosine Similarity		
23	15	5	Cosine Similarity		
23	15	7	Cosine Similarity		

#### 4.3.5 Pengujian *k-Fold Cross Validation*

Pengujian ini dilakukan untuk mengetahui keoptimalan hasil akurasi pada saat perubahan data uji terhadap data latih. Pengujian ini dilakukan dengan membagi sebanyak K data latih. Masing masing K digunakan untuk dijadikan data

uji dan yang lainnya digunakan sebagai data latih. Hasil perancangan pengujian akan ditampilkan pada Tabel 4.48.

**Tabel 4.49 Pengujian K-Fold Cross Validation**

<i>Fold</i>	Jumlah Fitur Information Gain	Nilai <i>K</i>	Perhitungan Jarak	Hasil Rata - rata Akurasi KNN (%)	Hasil Rata- rata Akurasi NWKNN (%)
5	10	3	Cosine Similarity		
10					
15					
20					
25					
30					

## BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi algoritme yang telah dibuat berdasarkan perancangan pada bab 4. Implementasi yang ada meliputi batasan implementasi dan implementasi algoritme.

### 5.1 Batasan Implementasi

Batasan yang digunakan dalam mengimplementasikan sistem identifikasi jenis makanan dengan menggunakan metode NWKNN adalah sebagai berikut.

1. Sistem dibangun dengan menggunakan bahasa python 3.6 dengan IDE spyder.
2. Data yang digunakan untuk implementasi dimasukkan ke dalam file csv.
3. Metode klasifikasi yang digunakan dalam mengenali citra makanan adalah K-NN dan NWKNN.
4. Data berasal dari data primer yang diambil di gedung F lantai 9 Fakultas Ilmu Komputer Universitas Brawijaya.
5. Input yang digunakan sistem berupa citra makanan tunggal.
6. Output yang diterima oleh *user* adalah hasil klasifikasi kelas makanan.
7. Fitur yang digunakan sebanyak 70 dengan rincian fitur color moments sebanyak 3 dengan warna R,G,B,H,S,V dan fitur GLCM sebanyak 13 dengan 4 sudut.

### 5.2 Implementasi Algoritme

Pada implementasi algoritma akan dijelaskan terkait *code* dari klasifikasi citra makanan yang mengacu pada bab perancangan sub bab perancangan proses yang meliputi proses perhitungan pada setiap langkah pada metode ekstraksi fitur, seleksi fitur dan klasifikasi.

#### 5.2.1 Implementasi Color Moments

Proses ekstraksi fitur *Color Moments* dengan warna RGB dan HSV mempunyai 3 fitur yaitu *mean*, standar deviasi dan *skewness*. Gambar awal citra berupa warna RGB maka tambah warna dengan warna HSV mengacu pada warna RGB. Setelah mendapatkan semua warna maka ekstraksi fitur dari warna bisa didapatkan. Fungsi pengambilan warna RGB ke HSV dapat ditunjukkan pada *Sourcecode* 5.1.

Sourcecode 5.1 : Implementasi fungsi readImages()	
1	def readImages(self, imageName):
2	imgRGB = cv2.imread(imageName)
3	imgHSV = cv2.cvtColor(imgRGB, cv2.COLOR_BGR2HSV)
4	return imgRGB, imgHSV

Berikut ini merupakan penjelasan dari Sourcecode 5.1:

1. Baris 1 adalah deklarasi fungsi dengan nama `readImages` dengan parameter nama file image.
2. Baris 2 adalah pemanggilan image dengan bantuan library `cv2`
3. Baris 3 digunakan mengubah image RGB menjadi HSV
4. Baris 4 mengembalikan nilai `imgRGB` dan `imgHSV`

Implementasi *sourcecode* untuk menyimpan fitur-fitur dari color moments ditunjukkan pada *Sourcecode 5.2*.

Sourcecode 5.2 : Implementasi fungsi <code>featureExtraction()</code>	
1	<code>def featureExtraction(self, image):</code>
2	<code>    colorMoments = np.zeros(3,np.float)</code>
3	<code>    l, w = image.shape</code>
4	
5	<code>    colorMoments[0] = self.mean(image, l, w)</code>
6	<code>    colorMoments[1] = self.std(image, colorMoments[0], l, w)</code>
7	<code>    colorMoments[2] = self.skewness(image, colorMoments[0],</code>
8	<code>    colorMoments[1], l, w)</code>
9	
10	<code>    return colorMoments</code>

Berikut ini merupakan penjelasan dari *Sourcecode 5.2*:

5. Baris 1 adalah deklarasi fungsi dengan nama `featureExtraction` dengan parameter nama file image.
6. Baris 2 membuat array dengan panjang 3 dan disimpan pada variable *colorMoments*.
7. Baris 3 mengambil panjang baris dan kolom dari gambar
8. Baris 5 Menghitung mean dengan memanggil fungsi `mean` dengan parameter `image, l` dan `w`.
9. Baris 6 menghitung standar deviasi dengan memanggil fungsi `std` dengan parameter `image, l` dan `w`.
10. Baris 8 mengitung *skewness* dengan memanggil fungsi `skewness` dengan paremetr `mean, standar deviasi, l` dan `w`
11. Baris 10 pengembalian nilai fitur `colorMoments`

Implementasi *sourcecode* untuk menghitung *mean* ditunjukkan pada *Sourcecode 5.3*.

Sourcecode 5.3 : Implementasi fungsi <code>mean()</code>	
1	<code>def mean(self, image, l, w):</code>
2	<code>    total = l * w</code>
3	<code>    pij = 0</code>
4	<code>    for i in range(l):</code>
5	<code>        for j in range(w):</code>
6	<code>            pij += image[i][j]</code>
7	
8	<code>    results = pij/total</code>
9	<code>    return results</code>

Berikut ini merupakan penjelasan dari *Sourcecode* 5.3:

1. Baris 1 adalah deklarasi fungsi dengan nama *mean* dengan parameter nama file *image*.
2. Baris 2 -8 adalah proses perhitungan *mean*
3. Baris 10 pengembalian nilai hasil fitur *mean*

Implementasi *sourcecode* untuk menghitung standar deviasi ditunjukkan pada *Sourcecode* 5.4

Sourcecode 5.4 : Implementasi fungsi std()	
1	def std(self, image, mean, l, w):
2	total = l * w
3	results = 0
4	for i in range(l):
5	for j in range(w):
6	results += ((image[i][j] - mean)**2)
7	
8	results = math.sqrt(results/total)
9	
10	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.4:

4. Baris 1 adalah deklarasi fungsi dengan nama *std* dengan parameter nama file *image*.
5. Baris 2 -8 adalah proses perhitungan standar deviasi
6. Baris 10 pengembalian nilai hasil fitur standar deviasi

Implementasi *sourcecode* untuk menghitung *skewness* ditunjukkan pada *Sourcecode* 5.5

Sourcecode 5.5 : Implementasi fungsi std()	
1	def skewness(self, image, mean, std, l, w):
2	total = l * w
3	results = 0
4	
5	for i in range(l):
6	for j in range(w):
7	results += ((image[i][j] - mean)**3)
8	
9	skewness = (results / total) / (std**3)
10	
11	return skewness

Berikut ini merupakan penjelasan dari *Sourcecode* 5.4:

7. Baris 1 adalah deklarasi fungsi dengan nama *skewness* dengan parameter nama file *image*, *mean*, *std*, *l* dan *w*.
8. Baris 2 -9 adalah proses perhitungan *skewness*
9. Baris 10 pengembalian nilai hasil fitur *skewness*

### 5.2.2 Implementasi GLCM

Implementasi ekstraksi fitur GLCM dengan input citra makanan yang telah disegmentasi akan diekstraksi menggunakan 13 fitur yaitu *Angular Second Moment (ASM)*, *Contrast*, *Correlation*, *Sum of Squares: Variance*, *Inverse Difference Moment (IDM)*, *Sum Average (AVER)*, *Sum Entropy (SENT)*, *Sum Variance (SVAR)*, *Entropy*, *Difference Entropy (DENT)*, *Difference Variance (DVAR)*, *Information Measure of Correlation 1* dan *Information Measure of Correlation 2*. Terdapat 2 fungsi utama untuk ekstraksi fitur GLCM yaitu `getMatrix` untuk menentukan matrix *cooccurrence* simetris yang sudah dinormalisasi dan `featureExtraction` untuk menyimpan 13 fitur pada GLCM. Sebelum melakukan ekstraksi fitur, masukan berupa citra tersegmentasi akan diubah menjadi citra *grayscale*. Berikut sourcecode fungsi mengubah citra RGB menjadi *grayscale* ditunjukkan pada Sourcecode 5.6.

Sourcecode 5.6 : Implementasi fungsi <code>readImages()</code>	
1	<code>def readImages(self, imageName):</code>
2	<code>    image = cv2.imread(imageName)</code>
3	<code>    imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)</code>
4	
5	<code>    return imageGray</code>

Berikut penjelasan dari Sourcecode 5.6:

1. Baris 1 adalah deklarasi fungsi dengan nama `readImages` dengan parameter nama file image.
2. Baris 2 adalah pemanggilan image dengan bantuan library `cv2`
3. Baris 3 digunakan mengubah image RGB menjadi *Grayscale*
4. Baris 4 mengembalikan nilai `imgRGB` dan *Grayscale*

#### 5.2.2.2 Implementasi Matrix Cooccurrence GLCM

Implementasi fungsi dari perhitungan *matriks cooccurrence* ditunjukkan pada Sourcecode 5.7.

Sourcecode 5.7 : Implementasi Matrix Cooccurrence GLCM	
1	<code>def getMatrix(self, image, sudut):</code>
2	
3	<code>    np.set_printoptions(threshold=np.inf)</code>
4	<code>    oriMatrix = np.array(image).astype(np.uint8)</code>
5	
6	<code>    l, w = oriMatrix.shape</code>
7	<code>    nMax = np.amax(oriMatrix)</code>
8	<code>    zeroMatrix = np.zeros((nMax+1, nMax+1), dtype=int)</code>
9	<code>    if(sudut == 0):</code>
10	<code>        for i in range(l):</code>
11	<code>            for j in range(w):</code>
12	<code>                if(j != w-1):</code>
13	<code>                    #print(i,j, i,j+1)</code>
14	<code>                    #print(oriMatrix[i,j], oriMatrix[i,j+1])</code>
15	<code>                    #print()</code>
16	<code>                    line = oriMatrix[i,j]</code>
17	<code>                    column = oriMatrix[i,j+1]</code>
18	<code>                    if (line != 0 and column != 0):</code>
19	<code>                        zeroMatrix[line, column] +=1</code>
20	
21	<code>    elif(sudut == 45):</code>



```

22         for i in range(l):
23             for j in range(w):
24                 if(i != l-1 and j != 0):
25                     #print(i,j, i+1,j-1)
26                     #print(oriMatrix[i,j], oriMatrix[i+1,j-1])
27                     line = oriMatrix[i,j]
28                     column = oriMatrix[i+1,j-1]
29                     if (line != 0 and column != 0):
30                         zeroMatrix[line,column] +=1
31
32     elif(sudut == 90):
33         for i in range(l):
34             for j in range(w):
35                 if(i != l-1):
36                     #print(i,j, i+1,j)
37                     #print(oriMatrix[i,j], oriMatrix[i+1,j])
38                     line = oriMatrix[i,j]
39                     column = oriMatrix[i+1,j]
40                     if (line != 0 and column != 0):
41                         zeroMatrix[line,column] +=1
42
43     elif(sudut == 135):
44         for i in range(l):
45             for j in range(w):
46                 if(i != l-1 and j != w-1):
47                     #print(i,j, i+1,j+1)
48                     #print(oriMatrix[i,j], oriMatrix[i+1,j+1])
49                     line = oriMatrix[i,j]
50                     column = oriMatrix[i+1,j+1]
51                     if (line != 0 and column != 0):
52                         zeroMatrix[line,column] +=1
53
54     #Membuat matrix co-occurence simetris
55     mSimetris = np.zeros((nMax+1,nMax+1))
56     totalPixel = 0
57     for i in range(len(mSimetris)):
58         for j in range(len(mSimetris)):
59             mSimetris[i,j] = zeroMatrix[i,j]+zeroMatrix[j,i]
60             totalPixel += mSimetris[i,j]
61
62     #Matrix di normalisasi
63     normalization = np.zeros((nMax+1,nMax+1))
64     normalization = mSimetris/totalPixel
65
66     return normalization

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.7:

1. Baris 1 adalah deklarasi fungsi dengan nama *getMatrix* dengan parameter nama file image dan sudut.
2. Baris 3 - 8 instansiasi variable
3. Baris 9 – 19 penentuan matrix *cooccurence* sudut  $0^{\circ}$
4. Baris 21 – 30 penentuan matrix *cooccurence* sudut  $45^{\circ}$
5. Baris 32 – 41 penentuan matrix *cooccurence* sudut  $90^{\circ}$
6. Baris 43 – 52 penentuan matrix *cooccurence* sudut  $135^{\circ}$
7. Baris 54 – 60 menghitung matriks simetris dengan mengalikan matriks *cooccurence* dengan transposenya
8. Baris 63 – 64 menghitung normalisasi dari matriks simetris

9. Baris 66 mengembalikan nilai matriks cooccurrence simetris ternormalisasi

### 5.2.2.3 Implementasi Ekstraksi Fitur

Fungsi untuk menyimpan 13 fitur dari GLCM adalah fungsi `featureExtraction`. Di dalam fungsi tersebut memanggil fungsi lagi berupa perhitungan dari setiap fitur GLCM. Berikut ini ekstraksi fitur GLCM ditunjukkan pada *Sourcecode 5.8*.

Sourcecode 5.8 : Implementasi fitur <code>featureExtraction()</code>	
1	<code>def featureExtraction(self, matrix):</code>
2	<code>    feature = np.zeros(13,np.double)</code>
3	<code>    l, w = matrix.shape</code>
4	<code>    feature[0] = self.angularSecondMoment(matrix, l, w)</code>
5	<code>    feature[1] = self.contrast(matrix, l, w)</code>
6	<code>    feature[2] = self.variance(matrix, l, w)</code>
7	<code>    feature[3] = self.inverseDifferentMoment(matrix, l, w)</code>
8	<code>    feature[4] = self.correlation(matrix, l, w)</code>
9	<code>    feature[5] = self.sumAverage(matrix, l, w)</code>
10	<code>    feature[6] = self.sumEntropy(matrix, l, w)</code>
11	<code>    feature[7] = self.sumVariance(matrix, l, w, feature[6])</code>
12	<code>    feature[8] = self.entropy(matrix, l, w)</code>
13	<code>    feature[9] = self.differenceEntropy(matrix, l, w)</code>
14	<code>    feature[10] = self.differenceVariance(matrix, l, w)</code>
15	<code>    feature[11], feature[12] =</code>
16	<code>    self.informationMeasureOfCorrelation(matrix, l, w, feature[8])</code>
17	
18	<code>    return feature</code>

Berikut ini merupakan penjelasan dari *Sourcecode 5.8*:

1. Baris 1 adalah deklarasi fungsi dengan nama `featureExtraction` dengan parameter nama file citra.
2. Baris 2 membuat array dengan nama `feature` dengan panjang 13
3. Baris 3 mencari panjang baris dan kolom dari citra
4. Baris 4 – 16 memanggil fungsi dari masing-masing fitur dan disimpan dalam array `feature`
5. Pengembalian nilai `feature` dari GLCM

Implementasi fungsi dari perhitungan ekstraksi fitur *Angular Second Moment (ASM)* ditunjukkan pada *Sourcecode 5.9*.

Sourcecode 5.9 : Implementasi Ekstraksi Fitur Angular Second Moment (ASM)	
1	<code>def angularSecondMoment(self, matrix, l, w):</code>
2	<code>    results = 0</code>
3	<code>    for i in range(l):</code>
4	<code>        for j in range(w):</code>
5	<code>            results += (matrix[i,j]**2)</code>
6	<code>    return results</code>

Berikut ini merupakan penjelasan dari *Sourcecode 5.9*:

6. Baris 1 adalah deklarasi fungsi dengan nama `angularSecondMoment` dengan parameter `matrix`, `l` dan `w`.
7. Baris 3 dan 4 perulangan sampai panjang baris dan kolom

8. Baris 5 Setiap piksel dihitung dengan dipangkatkan 2 dan hasilnya disimpan pada variable results
9. Baris 6 Mengembalikan hasil dari ASM

Implementasi fungsi dari perhitungan ekstraksi fitur *Contrast* ditunjukkan pada *Sourcecode* 5.10.

Sourcecode 5.10 : Implementasi Ekstraksi Fitur Contrast	
1	def contrast(self, matrix, l, w):
2	results = 0
3	for i in range(l):
4	for j in range(w):
5	results += ((i-j)**2) * matrix[i,j]
6	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.10:

1. Baris 1 adalah deklarasi fungsi dengan nama contrast dengan parameter matrix, l dan w.
2. Baris 3 dan 4 perulangan sampai panjang baris dan kolom
3. Baris 5 proses perhitungan fitur *Contrast*
4. Baris 6 pengembalian nilai hasil *Contrast*

Implementasi fungsi dari perhitungan ekstraksi fitur *Variance* ditunjukkan pada *Sourcecode* 5.11.

Sourcecode 5.11 : Implementasi Ekstraksi Fitur Variance	
1	def variance(self, matrix, l, w):
2	sumX = 0
3	total = 0
4	results = 0
5	for i in range(l):
6	for j in range(w):
7	sumX += matrix[i,j]
8	total += 1
9	average = sumX/total
10	
11	for i in range(l):
12	for j in range(w):
13	results += ((i-average)**2) * matrix[i,j]
14	
15	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.11:

1. Baris 1 adalah deklarasi fungsi dengan nama variance dengan parameter matrix, l dan w
2. Baris 2 – 4 inisialisasi dengan nilai 0
3. Baris 5 – 9 menghitung jumlah nilai tiap piksel citra, jumlah piksel citra dan dihitung rata-ratanya
4. Baris 11 – 13 menghitung *variance*
5. Baris 15 mengembalikan nilai fitur *variance*

Implementasi fungsi dari perhitungan ekstraksi fitur *Inverse Difference Matrix (IDM)* ditunjukkan pada *Sourcecode 5.12*.

Sourcecode 5.12 : Implementasi Ekstraksi Fitur Inverse Difference Matrix (IDM)	
1	def inverseDifferenceMoment(self, matrix, l, w):
2	results = 0
3	for i in range(l):
4	for j in range(w):
5	results += (matrix[i][j]**2) / (1+(i-j)**2)
6	return results

Berikut ini merupakan penjelasan dari *Sourcecode 5.12*:

1. Baris 1 adalah deklarasi fungsi dengan nama *inverseDifferenceMoment* dengan parameter *matrix*, *l* dan *w*
2. Baris 2 inisialisasi variable *results* dengan nilai 0
3. Baris 3-5 proses menghitung fitur *Inverse Difference Matrix (IDM)*
4. Baris 6 mengembalikan nilai hasil fitur *IDM*

Implementasi fungsi dari perhitungan ekstraksi fitur *Entropy* ditunjukkan pada *Sourcecode 5.13*.

Sourcecode 5.13 : Implementasi Ekstraksi Fitur Entropy	
1	def entropy(self, matrix, l, w):
2	results = 0
3	
4	for i in range(l):
5	for j in range(w):
6	results += (matrix[i][j] * np.log(matrix[i][j] +
7	np.finfo(float).eps))
8	
9	results = results * -1
10	return results

Berikut ini merupakan penjelasan dari *Sourcecode 5.13*:

5. Baris 1 adalah deklarasi fungsi dengan nama *entropy* dengan parameter *matrix*, *l* dan *w*
6. Baris 2 digunakan untuk inisialisasi variable *results* dengan nilai 0
7. Baris 4-9 adalah proses perhitungan *entropy*
8. Baris 10 mengembalikan nilai hasil fitur *entropy*

Implementasi fungsi dari perhitungan ekstraksi fitur *Correlation* ditunjukkan pada *Sourcecode 5.14*.

Sourcecode 5.14 : Implementasi Ekstraksi Fitur Correlation	
1	def correlation(self, matrix, l, w):
2	results = 0
3	averageX = 0
4	averageY = 0
5	stdX = 0

6	stdY = 0
7	
8	for i in range(l):
9	for j in range(w):
10	averageX += i * matrix[i,j]
11	averageY += j * matrix[i,j]
12	
13	for i in range(l):
14	for j in range(w):
15	stdX += matrix[i,j] * ((i - averageX)**2)**0.5
16	stdY += matrix[i,j] * ((j - averageY)**2)**0.5
17	
18	for i in range(l):
19	for j in range(w):
20	results += ((i*j) * matrix[i,j]) -
21	(averageX*averageY) / (stdX*stdY)
22	
23	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.14:

1. Baris 1 adalah deklarasi fungsi dengan nama *correlation* dengan parameter *matrix*, *l* dan *w*
2. Baris 2-6 inialisasi variable dengan nilai 0
3. Baris 8-21 proses perhitungan *correlation*
4. Mengembalikan nilai hasil *correlation*

Implementasi fungsi dari perhitungan ekstraksi fitur *Sum Average* ditunjukkan pada *Sourcecode* 5.15.

Sourcecode 5.15 : Implementasi Ekstraksi Fitur Sum Average	
1	def sumAverage(self, matrix, l, w):
2	pXPlusY = np.zeros((l+w))
3	results = 0
4	
5	for x in range(l):
6	for y in range(w):
7	pXPlusY[x+y] += matrix[x,y]
8	
9	for i in range (2, l+w):
10	results += i * pXPlusY[i]
11	
12	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.15:

1. Baris 1 adalah deklarasi fungsi dengan nama *sumAverage* dengan parameter *matrix*, *l* dan *w*
2. Baris 2-3 inialisasi variable
3. Baris 5-10 proses perhitungan *Sum Average*
4. Mengembalikan nilai hasil *Sum Average*

Implementasi fungsi dari perhitungan ekstraksi fitur *Sum Entropy* ditunjukkan pada *Sourcecode* 5.16.

Sourcecode 5.16 : Implementasi Ekstraksi Fitur Sum Entropy	
1	def sumEntropy(self, matrix, l, w):
2	pXPlusY = np.zeros((l+w))
3	results = 0
4	
5	for x in range(l):
6	for y in range(w):
7	pXPlusY[x+y] += matrix[x,y]
8	
9	for i in range (2, l+w):
10	results += i * pXPlusY[i] * np.log(pXPlusY[i] +
11	np.finfo(float).eps)
12	
13	results = results * -1
14	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.16:

1. Baris 1 adalah deklarasi fungsi dengan nama *sumEntropy* dengan parameter *matrix*, *l* dan *w*
2. Baris 2-3 inialisasi variable
3. Baris 5-13 proses perhitungan *Sum Entropy*
4. Mengembalikan nilai hasil *Sum Entropy*

Implementasi fungsi dari perhitungan ekstraksi fitur *Sum Variance* ditunjukkan pada *Sourcecode* 5.17.

Sourcecode 5.17 : Implementasi Ekstraksi Fitur Sum Variance	
1	def sumVariance(self, matrix, l, w, sumEntropy):
2	pXPlusY = np.zeros((l+w))
3	results = 0
4	
5	for x in range(l):
6	for y in range(w):
7	pXPlusY[x+y] += matrix[x,y]
8	
9	for i in range(2, l+w):
10	results += ((i - sumEntropy)**2) * pXPlusY[i]
11	
12	return results

Berikut ini merupakan penjelasan dari *Sourcecode* 5.17:

1. Baris 1 adalah deklarasi fungsi dengan nama *sumVariance* dengan parameter *matrix*, *l* dan *w*
2. Baris 2-3 inialisasi variable
3. Baris 5-10 proses perhitungan *Sum Variance*
4. Mengembalikan nilai hasil *Sum Variance*

Sourcecode 5.18 : Implementasi Ekstraksi Fitur Difference Entropy	
1	def differenceEntropy(self, matrix, l, w):
2	pXMinY = np.zeros((l))
3	results = 0
4	
5	for x in range(l):

```

6         for y in range(w):
7             pXMinY[np.abs(x-y)] += matrix[x,y]
8             #print(str(x)+'                                '+str(y)+'
9             '+'{0:.3f}'.format(matrix[x][y]))
10            for i in range(l-1):
11                results += i * pXMinY[i] * np.log(pXMinY[i] +
12                np.finfo(float).eps)
13
14            results = results * -1
15            return results

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.18:

1. Baris 1 adalah deklarasi fungsi dengan nama `differenceEntropy` dengan parameter `matrix`, `l` dan `w`
2. Baris 2-3 inialisasi variable
3. Baris 5-14 proses perhitungan *Difference Entropy*
4. Mengembalikan nilai hasil *Difference Entropy*

Implementasi fungsi dari perhitungan ekstraksi fitur *Difference Variance* ditunjukkan pada *Sourcecode* 5.19.

```

Sourcecode 5.19 : Implementasi Ekstraksi Fitur Difference Variance
1  def differenceVariance(self, matrix, l, w):
2      pXMinY = np.zeros((l))
3      results = 0
4
5      for x in range(l):
6          for y in range(w):
7              pXMinY[np.abs(x-y)] += matrix[x,y]
8
9      for i in range(l-1):
10         results += (i*i) * pXMinY[i]
11
12     return results

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.19:

1. Baris 1 adalah deklarasi fungsi dengan nama `differenceVariance` dengan parameter `matrix`, `l` dan `w`
2. Baris 2-3 inialisasi variable
3. Baris 5-10 proses perhitungan *Difference Variance*
4. Mengembalikan nilai hasil *Difference Variance*

Implementasi fungsi dari perhitungan ekstraksi fitur *Information Measure Of Correlation 1* dan *Information Measure Of Correlation 2* ditunjukkan pada *Sourcecode* 5.19.

Sourcecode 5.20 : Implementasi Ekstrasi Fitur Information Measure Of Correlation	
1	def informationMeasureOfCorrelation(self, matrix, l, w, entropy):
2	informationMeasureOfCorrelation1 = 0
3	informationMeasureOfCorrelation2 = 0
4	

```

5      pX = np.zeros((l))
6      pY = np.zeros((l))
7
8      HXY = entropy
9      HX = 0
10     HY = 0
11
12     HXY1 = 0
13     HXY2 = 0
14
15     maximal = 0
16
17     for i in range(l-1):
18         for j in range(w-1):
19             pX[i] += matrix[i,j]
20             pY[j] += matrix[i,j]
21     #print(pX)
22     for i in range(l-1):
23         for j in range(w-1):
24             HX += ((pX[i]) * np.log(pX[i] +
25 np.finfo(float).eps))
26             HY += ((pY[i]) * np.log(pY[i] +
27 np.finfo(float).eps))
28
29     HX = HX * -1
30     HY = HY * -1
31
32     if HX > HY:
33         maximal = HX
34     else:
35         maximal = HY
36
37     for i in range(l-1):
38         for j in range(w-1):
39             HXY1 += ((matrix[i,j]) * np.log((pX[i]*pY[j]) +
40 np.finfo(float).eps))
41             HXY2 += (pX[i] * pY[j] * np.log((pX[i]*pY[j]) +
42 np.finfo(float).eps))
43
44     HXY1 = HXY1 * -1
45     HXY2 = HXY2 * -1
46     print(HXY2)
47     informationMeasureOfCorrelation1 = (HXY - HXY1)/maximal
48     informationMeasureOfCorrelation2 = np.sqrt(np.abs(1 -
49 np.exp(-2 * (HXY2 - HXY))))
50
51     return informationMeasureOfCorrelation1,
52     informationMeasureOfCorrelation2

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.20:

1. Baris 1 adalah deklarasi fungsi dengan nama `informationMeasureOfCorrelation` dengan parameter `matrix`, `l`, `w` dan `entropy`
2. Baris 2-15 inialisasi variable
3. Baris 17-30 proses perhitungan *HX* dan *HY*
4. Baris 32-35 proses mencari maksimal antara *HX* dan *HY*
5. Baris 37-35 proses perhitungan *HXY1* dan *HXY2*



6. Baris 47-49 proses perhitungan *informationMeasureOfCorrelation1* dan *informationMeasureOfCorrelation2*
7. Mengembalikan nilai hasil *informationMeasureOfCorrelation1* dan *informationMeasureOfCorrelation2*

### 5.2.3 Implementasi Information Gain

Proses selanjutnya setelah mendapatkan fitur dari ekstraksi *Color Moments* dan *GLCM* adalah seleksi fitur *Information Gain*. Terdapat beberapa langkah dalam perhitungan *Information Gain* yaitu menghitung entropy, mengubah data tunggal menjadi data kelompok, menghitung nilai *Gain* pada masing-masing atribut, mengurutkan hasil *Information Gain* dan menentukan jumlah fitur untuk proses klasifikasi. Berikut implementasi dari perhitungan *Information Gain* dapat dilihat pada Sourcecode 5.21.

```

1  def informationGain(self,data):
2      fitur =
3      ['0ASM','0Contrast','0Variance','0IDM','0Correlation',
4      '0Average','0Sum Entropy','0Sum Variance','0Entropy',
5      '0DEntropy','0DVariance','0IMoC1','0IMoC2','45ASM',
6      '45Contrast','45Variance','45IDM','45Correlation',
7      '45Average','45Sum Entropy','45Sum Variance','45Entropy',
8      '45DEntropy','45DVariance','45IMoC1','45IMoC2',
9      '90ASM','90Contrast','90Variance','90IDM','90Correlation',
10     '90Average','90Sum Entropy','90Sum Variance','90Entropy',
11     '90DEntropy','90DVariance','90IMoC1','90IMoC2','135ASM',
12     '135Contrast','135Variance','135IDM','135Correlation',
13     '135Average','135Sum Entropy','135Sum Variance','135Entropy',
14     '135DEntropy','135DVariance','135IMoC1','135IMoC2','RMean',
15     'RVariance','RSkewness','GMean','GVariance','GSkewness',
16     'BMean','BVariance','BSkewness','HMean','HVariance',
17     'HSkewness','SMean','SVariance','SSkewness','VMean',
18     'VVariance','VSkewness']
19     jmlData = len(data)
20
21     #MENGHITUNG ENTROPY UNTUK KELAS
22     entropyS = 0
23     for i in range(len(jmlKelasI)):
24         entropyS += (-(jmlKelasI[i][1]/jmlData)
25 * math.log2(jmlKelasI[i][1]/jmlData))
26
27     #MENGUBAH DATA TUNGGAL MENJADI DATA KELOMPAK
28     jmlKelasAttr = []
29     for i in range(len(data[0])): #len(data[0])
30         if i > 2:
31             jmlKelasAttr.append(self. ubahTunggal2Kelompok(data,
32 jmlKelasI, i))
33
34     #MENGHITUNG ENTROPY PADA MASING-MASING ATRIBUT
35     hslInformationGain = []
36     for z in range(len(jmlKelasAttr)):
37         entropySAttr = np.zeros(len(jmlKelasAttr[z]))
38         jmlDataAttr = np.zeros(len(jmlKelasAttr[z]))
39         for i in range(len(jmlKelasAttr[z])):
40             for j in range(len(jmlKelasAttr[z][i])):
41                 jmlDataAttr[i] += jmlKelasAttr[z][i][j][1]
42
43             for j in range(len(jmlKelasAttr[z][i])):

```

```

44         if jmlKelasAttr[z][i][j][1] != 0 :
45             entropySAttr[i] +=
46 (- (jmlKelasAttr[z][i][j][1]/jmlDataAttr[i])
47 * math.log2 (jmlKelasAttr[z][i][j][1]/jmlDataAttr[i]))
48         else:
49             entropySAttr[i] += 0
50
51         #MENGHITUNG SIGMA DARI ENTROPY MASING MASING ATRIBUT
52         sigmaEntropy = 0
53         for i in range(len(entropySAttr)):
54             sigmaEntropy += (jmlDataAttr[i]/len(data)) *
55 entropySAttr[i]
56
57         hasil = entropyS - sigmaEntropy
58         hslInformationGain.append([])
59         hslInformationGain[z].append(fitur[z])
60         hslInformationGain[z].append(hasil)
61
62         hslInformationGain = sorted(hslInformationGain, key=lambda x:
63 x[1], reverse=True)
64         return hslInformationGain

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.21:

1. Baris 1 adalah deklarasi fungsi dengan nama *InformationGain* dengan parameter *data*
2. Baris 2-18 inialisasi array *fitur*
3. Baris 19 menyimpan jumlah data dari data latih
4. Baris 21-25 proses menghitung entropy untuk kelas
5. Baris 27-32 proses mengubah data tunggal menjadi data kelompok dengan memanggil fungsi *ubahTunggal2Kelompok()* menghasilkan array 3 dimensi dengan nama *jmlKelasAttr*. Terdapat 70 fitur dengan masing-masing fitur memiliki kelas dan masing-masing kelas terdapat jumlah frekuensi dari masing-masing kategori makanan. Kategori/kelas makanan terdapat 16 kelas.
6. Baris 35 inialisasi array dengan nama *hslInformationGain*
7. Baris 36 perulangan sejumlah banyaknya fitur yaitu 70 fitur
8. Baris 37-49 proses menghitung entropy dari 70 atribut/fitur. Setiap fitur akan dihitung entropy dari masing-masing kelas.
9. Baris 51-55 menghitung jumlah entropy dari masing-masing kelas pada fitur.
10. Baris 57 menghitung hasil dari *Information Gain* masing-masing fitur
11. Baris 58-60 adalah proses menambahkan hasil *Information Gain* ke dalam array
12. Baris 62-63 adalah proses mengurutkan hasil dari yang terbesar ke yang terkecil

Proses untuk mengecek jumlah kelas makanan dari data latih dapat dilihat pada *Sourcecode* 5.22.

Sourcecode 5.22 : Implementasi Cek Jumlah Kelas Makanan

```

1  def cekJumlahKelas(self, data):
2      result = []
3      result.append([])
4      result[0].append(data[0][2])
5      result[0].append(int(1))
6      temp = 0
7
8      for i in range(len(data)):
9          if i > 0 :
10             cek = False
11             for j in range(len(result)):
12                 if data[i][2] == result[j][0]:
13                     result[temp][1] += 1
14                     cek = True
15
16             if cek == False:
17                 temp += 1
18                 result.append([])
19                 result[temp].append(data[i][2])
20                 result[temp].append(int(1))
21
22     return result

```

Berikut ini merupakan penjelasan dari *Sourcecode 5.22*:

1. Baris 1 adalah deklarasi fungsi dengan nama `cekJumlahKelas` dengan parameter `data`
2. Baris 2-6 inisialisasi array `result` dengan nilai pertama adalah kelas makanan pertama dan nilai 1.
3. Baris 8-20 adalah proses mencari jumlah masing-masing kelas makanan
4. Mengembalikan nilai `results`

Proses untuk mengubah data tunggal pada masing-masing fitur ke dalam data kelompok dapat dilihat pada fungsi `ubahTunggal2Kelompok()` dengan Sourcecode 5.23.

Sourcecode 5.23 : Implementasi Ubah Data Tunggal Ke Data Kelompok

```

1  def ubahTunggal2Kelompok(self, data, jmlKelasI, attr):
2      #MENYIMPAN ATRIBUT KE attr
3      atribut = []
4      for i in range(len(data)):
5          atribut.append(float(data[i][attr]))
6
7      #MENGUBAH DATA TUNGGAL KE DALAM DATA KELOMPOK
8      J = float(max(atribut)) - float(min(atribut))
9
10     #Menentukan banyaknya kelas dengan aturan sturgess
11     k = round(1 + 3.3 * np.log10(len(atribut)))
12
13     #Menentukan panjang kelas ke i
14     pjgKelas = J/k
15     pjgKelas = round(pjgKelas, 7)
16
17
18     #INISIALIASI ARRAY [[['Donat', 0], ['Rendang', 0]],
19     [['Donat', 0], ['Rendang', 0]]]

```

```

20     result = []
21     for i in range(int(k)):
22         result.append([])
23         for j in range(len(jmlKelasI)):
24             result[i].append([])
25             result[i][j].append(jmlKelasI[j][0])
26             result[i][j].append(0)
27
28     batas0 = min(atribut)
29     batas1 = min(atribut)
30     for x in range(int(k)):
31         batas1 += pjgKelas
32         for y in range(len(atribut)):
33             if x == 0:
34                 if atribut[y] >= batas0 and atribut[y] <=
35 batas1:
36                     for z in range(len(jmlKelasI)):
37                         cek = False
38                         if data[y][2] == jmlKelasI[z][0]:
39                             result[x][z][1] += 1
40                             cek = True
41                             if cek == True:
42                                 break
43                     else:
44                         if atribut[y] > batas0 and atribut[y] <= batas1:
45                             for z in range(len(jmlKelasI)):
46                                 cek = False
47                                 if data[y][2] == jmlKelasI[z][0]:
48                                     result[x][z][1] += 1
49                                     cek = True
50                                     if cek == True:
51                                         break
52         batas0 += pjgKelas
53     return result

```

Berikut ini merupakan penjelasan dari *Sourcecode* 5.23:

1. Baris 1 adalah deklarasi fungsi dengan nama `ubahTunggal2Kelompok()` dengan parameter `data`, `jmlKelasI`, `attr`
2. Baris 3-5 menyimpan nilai per fitur ke dalam array dengan nama `atribut`
3. Baris 8 berfungsi untuk menentukan jangkauan dengan menghitung selisih antara nilai maksimal dan minimal
4. Baris 11 berfungsi untuk menentukan kelas yang dibuat pada fitur
5. Baris 14-15 berfungsi untuk menentukan panjang dari masing-masing kelas
6. Baris 18-26 adalah inisialisasi array 3 dimensi. Dimensi pertama adalah dengan panjang jumlah kelas. Dimensi kedua dengan panjang kategori makanan yaitu sebanyak 16 jenis. Dimensi ketiga kolom pertama adalah nama jenis makanan dan kolom kedua adalah nilai 0.
7. Baris 28-52 adalah proses menghitung jumlah jenis makanan pada masing-masing kelas data kelompok
8. Mengembalikan nilai hasil data kelompok

Proses terakhir dari Information Gain adalah mengambil sejumlah k fitur yang akan digunakan untuk proses klasifikasi. Berikut implementasi dapat dilihat pada *Sourcecode 5.24*.

Sourcecode 5.24 : Implementasi Menentukan Jumlah Fitur

```

1 def hapusAtribut(self, k, dataLatih, dataUji, hasilIG):
2     fitur = ['No', 'Kode', 'Nama', '0ASM', '0Contrast',
3     '0Variance', '0IDM', '0Correlation', '0Average', '0Sum Entropy',
4     '0Sum Variance', '0Entropy', '0DEntropy', '0DVariance', '0IMoC1',
5     '0IMoC2', '45ASM', '45Contrast', '45Variance', '45IDM', '45Correlation',
6     '45Average', '45Sum Entropy', '45Sum Variance', '45Entropy',
7     '45DEntropy', '45DVariance', '45IMoC1', '45IMoC2', '90ASM', '90Contrast',
8     '90Variance', '90IDM', '90Correlation', '90Average',
9     '90Sum Entropy', '90Sum Variance', '90Entropy', '90DEntropy',
10    '90DVariance', '90IMoC1', '90IMoC2', '135ASM', '135Contrast',
11    '135Variance', '135IDM', '135Correlation', '135Average',
12    '135Sum Entropy', '135Sum Variance', '135Entropy',
13    '135DEntropy', '135DVariance', '135IMoC1', '135IMoC2',
14    'RMean', 'RVariance', 'RSkewness', 'GMean', 'GVariance',
15    'GSkewness', 'BMean', 'BVariance', 'BSkewness', 'HMean',
16    'HVariance', 'HSkewness', 'SMean', 'SVariance', 'SSkewness',
17    'VMean', 'VVariance', 'VSkewness']
18    dataLatih = np.insert(dataLatih, [0], fitur, axis = 0)
19    dataUji = np.insert(dataUji, [0], fitur, axis = 0)
20
21    dtHapus = len(dataLatih[0]) - k - 3
22    sortHasilIG = sorted(hasilIG, key=lambda x: x[1])
23
24    for i in range(dtHapus):
25        cek = False
26        for j in range(len(dataUji[0])):
27            if j > 2:
28                #print(sortHasilIG[i][0], dataUji[0][j])
29                if dataUji[0][j] == sortHasilIG[i][0]:
30                    #print(dataUji[0][j], sortHasilIG[i][0])
31                    dataUji = np.delete(dataUji, np.s_[j],
32                    axis=1)
33                    dataLatih = np.delete(dataLatih, np.s_[j],
34                    axis=1)
35                    cek = True
36                    if cek == True:
37                        break
38                dataUji = np.delete(dataUji, np.s_[0], axis=0)
39                dataLatih = np.delete(dataLatih, np.s_[0], axis=0)
40    return dataLatih, dataUji

```

Berikut ini merupakan penjelasan dari *Sourcecode 5.24*:

1. Baris 1 adalah deklarasi fungsi dengan nama `hapusAtribut()` dengan parameter `k`, `dataLatih`, `dataUji`, `hasilIG`
2. Baris 2-17 inisialisasi fitur
3. Baris 18-19 proses insert fitur ke array `dataLatih` dan `dataUji`
4. Baris 21 menghitung jumlah fitur yang akan dihapus
5. Baris 22 proses mengurutkan data hasil IG dari yang terkecil
6. Baris 24-37 proses menghapus fitur yang tidak diperlukan
7. Baris 38-39 proses menghapus baris 0 dari `dataLatih` dan `dataUji`

## 8. Mengembalikan nilai dataLatih dan dataUji

### 5.2.4 Implementasi Klasifikasi KNN dan NWKNN

Proses klasifikasi akan dilakukan menggunakan *KNN* dan *NWKNN*. Proses *KNN* terbagi menjadi 2 yaitu menghitung jarak dan mengurutkan jarak. Hasil urutan diambil sejumlah *K* dari yang terbesar lalu dicari kelas terbanyak. Proses *NWKNN* adalah lanjutan dari *KNN* dengan proses setelah mendapatkan hasil pengurutan jarak dilakukan pembobotan setiap kelas dan dilakukan perhitungan skor.

#### 5.2.4.1 Implementasi KNN

Berikut implementasi dari algoritma *KNN* dapat dilihat pada Sourcecode 5.35.

Sourcecode 5.25 : Implementasi Algoritma KNN

```
1 def KNN(self, k, metodeJarak, dataLatih, dataUji):
2     hasil = np.zeros((len(dataUji),len(dataLatih)), )
3     namaMakanan = np.empty((len(dataUji),len(dataLatih)), dtype
4 = object)
5
6     if metodeJarak == "Manhattan":
7         #PERHITUNGAN JARAK MENGGUNAKAN EUCLIDEAN DISTANCE
8         for i in range(len(dataUji)):
9             for x in range(len(dataLatih)):
10                namaMakanan[i][x] = dataLatih[x][2]
11                for y in range(len(dataLatih[0])):
12                    if(y > 2):
13                        #hasil[i][x]
14 self.euclidean(float(dataLatih[x][y]), float(dataUji[i][y])) +=
15                        hasil[i][x] +=
16 np.power((float(dataLatih[x][y])-float(dataUji[i][y])),2)
17                        hasil[i][x] = np.sqrt(hasil[i][x])
18                elif metodeJarak == "Euclidean":
19                    #PERHITUNGAN JARAK MENGGUNAKAN MANHATTAN DISTANCE
20                    for i in range(len(dataUji)):
21                        for x in range(len(dataLatih)):
22                            namaMakanan[i][x] = dataLatih[x][2]
23                            for y in range(len(dataLatih[0])):
24                                if(y > 2):
25                                    #hasil[i][x]
26 self.manhattan(float(dataLatih[x][y]), float(dataUji[i][y])) +=
27                                    hasil[i][x] +=
28 np.abs(float(dataLatih[x][y])-float(dataUji[i][y]))
29                elif metodeJarak == "Cosim":
30                    #PERHITUNGAN JARAK MENGGUNAKAN COSINE SIMILARITY
31                    for i in range(len(dataUji)): #len(dataUji)
32                        for x in range(len(dataLatih)): #len(dataLatih)
33                            djQ = 0
34                            djMutlak = 0
35                            qMutlak = 0
36                            namaMakanan[i][x] = dataLatih[x][2]
37                            for y in range(3,len(dataLatih[0])):
38                                djQ += float(dataLatih[x][y]) *
39 float(dataUji[i][y])
40                                djMutlak += (float(dataLatih[x][y])**2)
41                                qMutlak += (float(dataUji[i][y])**2)
42                            djMutlak = math.sqrt(djMutlak)
43                            qMutlak = math.sqrt(qMutlak)
```

```

44         hasil[i][x] = djQ / (djMutlak * qMutlak)
45
46     resultKNN = []
47     for i in range(len(hasil)):
48         resultKNN.append([])
49         for j in range(len(hasil[0])):
50             resultKNN[i].append([])
51             resultKNN[i][j].append(namaMakanan[i][j])
52             resultKNN[i][j].append(hasil[i][j])
53
54     for i in range(len(resultKNN)):
55         if metodeJarak == "Manhattan" or metodeJarak ==
56 "Eculidean":
57             resultKNN[i] = sorted(resultKNN[i],key=lambda x:
58 x[1])
59             elif metodeJarak == "Cosim":
60                 resultKNN[i] = sorted(resultKNN[i],key=lambda x:
61 x[1], reverse = True)
62
63     #print(resultKNN[0])
64     kelasMakanan = np.zeros((len(resultKNN),32))
65
66     for x in range(len(resultKNN)):
67         for y in range(k):
68             if (resultKNN[x][y][0] == 'Donat'):
69                 kelasMakanan[x][0] += 1
70             elif (resultKNN[x][y][0] == 'Roti Gandum'):
71                 kelasMakanan[x][1] += 1
72             elif (resultKNN[x][y][0] == 'Roti Tawar'):
73                 kelasMakanan[x][2] += 1
74             elif (resultKNN[x][y][0] == 'Indomie Goreng'):
75                 kelasMakanan[x][3] += 1
76             elif (resultKNN[x][y][0] == 'Mie Gepeng'):
77                 kelasMakanan[x][4] += 1
78             elif (resultKNN[x][y][0] == 'Telor Ceplok'):
79                 kelasMakanan[x][5] += 1
80             elif (resultKNN[x][y][0] == 'Telor Dadar'):
81                 kelasMakanan[x][6] += 1
82             elif (resultKNN[x][y][0] == 'Fried Chicken'):
83                 kelasMakanan[x][7] += 1
84             elif (resultKNN[x][y][0] == 'Rendang'):
85                 kelasMakanan[x][8] += 1
86             elif (resultKNN[x][y][0] == 'Mentimun'):
87                 kelasMakanan[x][9] += 1
88             elif (resultKNN[x][y][0] == 'Gatau'):
89                 kelasMakanan[x][10] += 1
90             elif (resultKNN[x][y][0] == 'Kubis'):
91                 kelasMakanan[x][11] += 1
92             elif (resultKNN[x][y][0] == 'Selada'):
93                 kelasMakanan[x][12] += 1
94             elif (resultKNN[x][y][0] == 'Kemangi'):
95                 kelasMakanan[x][13] += 1
96             elif (resultKNN[x][y][0] == 'Tomat'):
97                 kelasMakanan[x][14] += 1
98             elif (resultKNN[x][y][0] == 'Strawberry'):
99                 kelasMakanan[x][15] += 1
100             elif (resultKNN[x][y][0] == 'Pisang Ijo'):
101                 kelasMakanan[x][16] += 1
102             elif (resultKNN[x][y][0] == 'Pisang Kuning'):
103                 kelasMakanan[x][17] += 1
104             elif (resultKNN[x][y][0] == 'Jeruk Orange'):
105                 kelasMakanan[x][18] += 1
106             elif (resultKNN[x][y][0] == 'Jeruk Ijo Orange'):
107                 kelasMakanan[x][19] += 1
108             elif (resultKNN[x][y][0] == 'Nasi Kuning'):

```

```

109         kelasMakanan[x][20] += 1
110     elif (resultKNN[x][y][0] == 'Nasi Merah'):
111         kelasMakanan[x][21] += 1
112     elif (resultKNN[x][y][0] == 'Oreo'):
113         kelasMakanan[x][22] += 1
114     elif (resultKNN[x][y][0] == 'Beng Beng'):
115         kelasMakanan[x][23] += 1
116     elif (resultKNN[x][y][0] == 'Soba Mie'):
117         kelasMakanan[x][24] += 1
118     elif (resultKNN[x][y][0] == 'Tim Tam'):
119         kelasMakanan[x][25] += 1
120     elif (resultKNN[x][y][0] == 'Happy Toss'):
121         kelasMakanan[x][26] += 1
122     elif (resultKNN[x][y][0] == 'Gerry Saluut'):
123         kelasMakanan[x][27] += 1
124     elif (resultKNN[x][y][0] == 'Biskuat Coklat'):
125         kelasMakanan[x][28] += 1
126     elif (resultKNN[x][y][0] == 'Milo Nuggets'):
127         kelasMakanan[x][29] += 1
128     elif (resultKNN[x][y][0] == 'Sari Roti'):
129         kelasMakanan[x][30] += 1
130     elif (resultKNN[x][y][0] == 'Genji Pie'):
131         kelasMakanan[x][31] += 1
132
133     maxValue = np.zeros((len(kelasMakanan)))
134     indexMaxValue = []
135     for x in range(len(kelasMakanan)):
136         maxValue[x] = np.amax(kelasMakanan[x])
137         indexMaxValue.append(np.where(kelasMakanan[x] ==
138 maxValue[x]))
139
140     hasilKelas = np.empty((len(kelasMakanan)), dtype = object)
141     for x in range(len(kelasMakanan)):
142         if(indexMaxValue[x][0][0] == 0):
143             hasilKelas[x] = 'Donat'
144         elif (indexMaxValue[x][0][0] == 1):
145             hasilKelas[x] = 'Roti Gandum'
146         elif (indexMaxValue[x][0][0] == 2):
147             hasilKelas[x] = 'Roti Tawar'
148         elif (indexMaxValue[x][0][0] == 3):
149             hasilKelas[x] = 'Indomie Goreng'
150         elif (indexMaxValue[x][0][0] == 4):
151             hasilKelas[x] = 'Mie Gepeng'
152         elif (indexMaxValue[x][0][0] == 5):
153             hasilKelas[x] = 'Telor Ceplok'
154         elif (indexMaxValue[x][0][0] == 6):
155             hasilKelas[x] = 'Telor Dadar'
156         elif (indexMaxValue[x][0][0] == 7):
157             hasilKelas[x] = 'Fried Chicken'
158         elif (indexMaxValue[x][0][0] == 8):
159             hasilKelas[x] = 'Rendang'
160         elif (indexMaxValue[x][0][0] == 9):
161             hasilKelas[x] = 'Mentimun'
162         elif (indexMaxValue[x][0][0] == 10):
163             hasilKelas[x] = 'Gatau'
164         elif (indexMaxValue[x][0][0] == 11):
165             hasilKelas[x] = 'Kubis'
166         elif (indexMaxValue[x][0][0] == 12):
167             hasilKelas[x] = 'Selada'
168         elif (indexMaxValue[x][0][0] == 13):
169             hasilKelas[x] = 'Kemangi'
170         elif (indexMaxValue[x][0][0] == 14):
171             hasilKelas[x] = 'Tomat'
172         elif (indexMaxValue[x][0][0] == 15):
173             hasilKelas[x] = 'Strawberry'

```



174	elif (indexMaxValue[x][0][0] == 16):
175	hasilKelas[x] = 'Pisang Ijo'
176	elif (indexMaxValue[x][0][0] == 17):
177	hasilKelas[x] = 'Pisang Kuning'
178	elif (indexMaxValue[x][0][0] == 18):
179	hasilKelas[x] = 'Jeruk Orange'
180	elif (indexMaxValue[x][0][0] == 19):
181	hasilKelas[x] = 'Jeruk Ijo Orange'
182	elif (indexMaxValue[x][0][0] == 20):
183	hasilKelas[x] = 'Nasi Kuning'
184	elif (indexMaxValue[x][0][0] == 21):
185	hasilKelas[x] = 'Nasi Merah'
186	elif (indexMaxValue[x][0][0] == 22):
187	hasilKelas[x] = 'Oreo'
188	elif (indexMaxValue[x][0][0] == 23):
189	hasilKelas[x] = 'Beng Beng'
190	elif (indexMaxValue[x][0][0] == 24):
191	hasilKelas[x] = 'Soba Mie'
192	elif (indexMaxValue[x][0][0] == 25):
193	hasilKelas[x] = 'Tim Tam'
194	elif (indexMaxValue[x][0][0] == 26):
195	hasilKelas[x] = 'Happy Toss'
196	elif (indexMaxValue[x][0][0] == 27):
197	hasilKelas[x] = 'Gerry Saluut'
198	elif (indexMaxValue[x][0][0] == 28):
199	hasilKelas[x] = 'Biskuat Coklat'
200	elif (indexMaxValue[x][0][0] == 29):
201	hasilKelas[x] = 'Milo Nuggets'
202	elif (indexMaxValue[x][0][0] == 30):
203	hasilKelas[x] = 'Sari Roti'
204	elif (indexMaxValue[x][0][0] == 31):
205	hasilKelas[x] = 'Genji Pie'
206	
207	return hasilKelas, resultKNN

Berikut ini merupakan penjelasan dari *Sourcecode* 5.25:

1. Baris 1 adalah deklarasi fungsi dengan nama `hapusAtribut()` dengan parameter `k`, `metodeJarak`, `dataLatih`, `dataUji`
2. Baris 6-17 jika `metodeJarak` adalah "Manhattan" maka akan memproses perhitungan jarak menggunakan *Manhattan Distance*
3. Baris 18-28 jika `metodeJarak` adalah "Euclidean" maka akan memproses perhitungan jarak menggunakan *Euclidean Distance*
4. Baris 29-44 jika `metodeJarak` adalah "Cosim" maka akan memproses perhitungan jarak menggunakan *Cosine Similarity*
5. Baris 46-52 inisialisasi array 3 dimensi dengan nama `resultKNN` dengan dimensi pertama adalah jumlah data hasil klasifikasi sejumlah data uji. Dimensi kedua hasil perhitungan jarak dengan panjang sesuai dengan data latih dan dimensi ketiga adalah kolom pertama adalah nama kelas makanan dari data latih dan kolom kedua adalah hasil dari perhitungan jarak.
6. Baris 54-61 proses mengurutkan hasil perhitungan jarak. Jika `metode` perhitungan jarak adalah *Manhattan* dan *Euclidean* maka akan diurutkan mulai terkecil sampai terbesar. Jika `metode` perhitungan jarak adalah *Cosine Similarity* maka akan diurutkan mulai terbesar sampai terkecil.

7. Baris 64-131 proses menghitung jumlah kelas yang dihasilkan sampai data ke k
8. Baris proses mencari index dari nilai kelas terbanyak
9. Baris proses mengganti index menjadi nama kelas makanan
10. Mengembalikan nilai hasil kelas makanan dan hasil KNN

#### 5.2.4.2 Implementasi NWKNN

Berikut implementasi dari algoritma KNN dapat dilihat pada Sourcecode 5.26.

Sourcecode 5.26 : Implementasi NWKNN	
1	def NWKNN(self, kN, dataLatih, resultKNN):
2	kelas = self.cekJumlahKelas(dataLatih)
3	data = []
4	for i in range(len(kelas)):
5	data.append(kelas[i][1])
6	
7	mins = np.amin(data)
8	for i in range(len(kelas)):
9	kelas[i].append(1 / ((kelas[i][1] / mins)**(1/4)))
10	kelas[i].append(0)
11	
12	#INISIALISASI SKOR
13	skor = []
14	for i in range(len(resultKNN)):
15	skor.append([])
16	for j in range(len(kelas)):
17	skor[i].append([])
18	skor[i][j].append(kelas[j][0])
19	skor[i][j].append(0)
20	skor[i][j].append(kelas[j][2])
21	
22	#PERHITUNGAN SKOR
23	for i in range(len(resultKNN)):
24	for j in range(len(skor[0])):
25	temp = 1
26	for k in range(len(resultKNN[0])):
27	
28	#print(str(skor[i][j][0])+"==" +str(resultKNN[i][k][0]))
29	if skor[i][j][0] == resultKNN[i][k][0]:
30	skor[i][j][1] += resultKNN[i][k][1] *1
31	else:
32	skor[i][j][1] += resultKNN[i][k][1] *0
33	#print("SKOR SAAT INI " + str(skor[i][j][1]))
34	if temp == kN:
35	break
36	temp += 1
37	skor[i][j][1] *= skor[i][j][2]
38	
39	#SORTING SKOR
40	for i in range(len(skor)):
41	skor[i] = sorted(skor[i],key=lambda x: x[1], reverse =
42	True)
43	
44	#HASIL NWKNN
45	hasil = []
46	for i in range(len(resultKNN)):
47	hasil.append(skor[i][0][0])

48	
49	return hasil

Berikut ini merupakan penjelasan dari *Sourcecode* 5.26:

1. Baris 1 adalah deklarasi fungsi dengan nama NWKNN() dengan parameter kN, dataLatih, resultKNN
2. Baris 2 mengecek jumlah kelas pada data latih
3. Baris 4-10 proses menghitung bobot dari masing-masing kelas
4. Baris 12 – 20 inisialisasi array 3 dimensi dengan nama skor. Dimensi pertama dengan panjang sejumlah data uji. Dimensi kedua dengan panjang sejumlah kelas makanan. Dimensi ketiga kolom 1 adalah nama makanan, kolom 2 adalah nilai 0 dan kolom 3 adalah hasil skor
5. Baris 22-37 proses perhitungan skor dari masing-masing kelas sesuai dengan k
6. Baris 39-42 proses mengurutkan skor
7. Baris 44-47 adalah hasil dari NWKNN dimana kelas makanan yang diambil adalah dengan skor tertinggi
8. Mengembalikan nilai hasil NWKNN

### 5.2.5 Impelementasi Pengujian

Proses implementasi algoritme terdapat 2 metode yaitu pengujian akurasi dan pengujian K-Fold Cross Validation.

#### 5.2.5.1 Implementasi Pengujian Akurasi

Berikut implementasi dari pengujian dapat dilihat pada *Sourcecode* 5.27.

Sourcecode 5.27 : Implementasi Pengujian Akurasi	
1	def ujiAkurasi(self, dtUji, hasilKelas):
2	hasil = 0
3	for i in range(len(dtUji)):
4	if dtUji[i][2] == hasilKelas[i]:
5	hasil += 1
6	
7	presentase = hasil/len(dtUji) *100
8	
9	return presentase

Berikut ini merupakan penjelasan dari *Sourcecode* 5.27:

1. Baris 1 adalah deklarasi fungsi dengan nama ujiAkurasi() dengan parameter dtUji, hasilKelas
2. Baris 2-7 proses menghitung prosentase akurasi.
3. Baris 9 mengembalikan nilai hasil presentase akurasi

### 5.2.5.2 Implementasi Pengujian K-Fold Cross Validation

Berikut implementasi dari pengujian dapat dilihat pada Sourcecode 5.28.

Sourcecode 5.28 : Implementasi Pengujian K-Fold Cross Validation	
1	def kFoldCrossValidation(self, kF,kN,metodeJarak, dtLatih):
2	kFold = []
3	bagi = len(dtLatih) / kF
4	sisa = len(dtLatih) % kF
5	
6	adders = 0
7	for i in range(kF):
8	kFold.append([])
9	for j in range(int(bagi)):
10	kFold[i].append(dtLatih[adders])
11	
12	adders +=1
13	if adders == (len(dtLatih) - sisa):
14	for k in range(sisa):
15	kFold[i].append(dtLatih[adders+k])
16	
17	klasifikasi = Klasifikasi()
18	hasilKNN = []
19	hasilNWKNN = []
20	hasilKelasKNN = []
21	hasilKelasNWKNN = []
22	resultKNN = []
23	for i in range(kF):
24	
25	#MENCARI DATA LATIH BARU = DATA LATIH LAMA - DATA UJI
26	dtLatihBaru = dtLatih
27	temp = 0
28	for j in range(len(kFold[i])):
29	for k in range(len(dtLatih)):
30	if kFold[i][j][1] == dtLatih[k][1]:
31	dtLatihBaru = np.delete(dtLatihBaru,
32	np.s_[k-temp], axis=0)
33	temp +=1
34	break
35	
36	#KLASIFIKASI KNN DAN NWKNN
37	hasilKelasKNN, resultKNN = klasifikasi.KNN(kN,
38	metodeJarak, dtLatih, kFold[i])
39	hasilKelasNWKNN = klasifikasi.NWKNN(kN, dtLatih,
40	resultKNN)
41	
42	hasilKNN.append(self.ujiAkurasi(kFold[i],
43	hasilKelasKNN))
44	hasilNWKNN.append(self.ujiAkurasi(kFold[i],
45	hasilKelasNWKNN))
46	return hasilKNN, hasilNWKNN
47	

Berikut ini merupakan penjelasan dari *Sourcecode* 5.28:

1. Baris 1 adalah deklarasi fungsi dengan nama `kFoldCrossValidation()` dengan parameter `kF`, `kN`, `metodeJarak`, `dtLatih`
2. Baris 2 inialisasi array dengan nama *kFold* yang akan digunakan sebagai dataUji
3. Baris 3 menghitung jumlah data tiap k pada *kFold*

4. Baris 4 menghitung sisa data
5. Baris 6-15 digunakan untuk memasukkan data uji dimana jumlah data uji tiap kF sebanyak nilai variabel bagi dan sisa data latih akan dimasukkan ke dalam data uji pada kF terakhir.
6. Baris 17 instansiasi objek dengan nama klasifikasi
7. Baris 18-22 inisialisasi array
8. Baris 23 perulangan sejumlah kF
9. Baris 25-34 membuat data latih baru dimana data dari data latih lama akan dikurangi dengan data uji (kFold) ke kF
10. Baris 36-46 menghitung klasifikasi menggunakan KNN dan NWKNN dan hasilnya akan dimasukkan ke dalam array hasilKNN dan hasilNWKNN
11. Baris 47 Mengembalikan nilai hasilKNN dan hasilNWKNN

## BAB 6 PENGUJIAN DAN ANALISIS

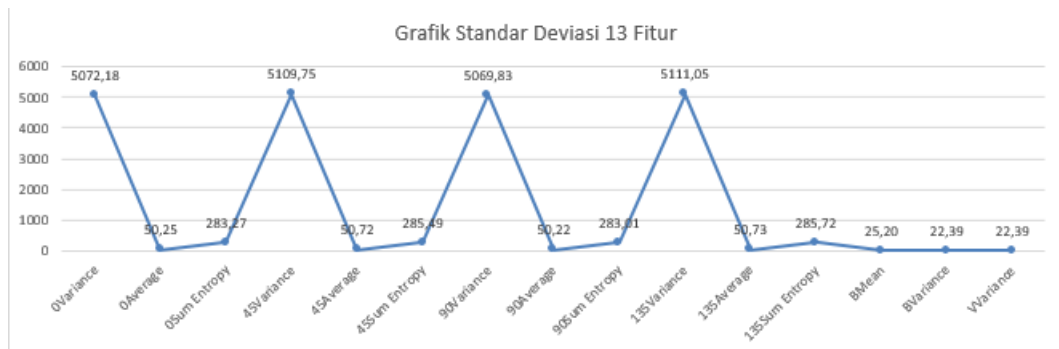
Pada bagian ini dijelaskan mengenai pengujian dan analisis dari hasil implementasi klasifikasi jenis makanan dari citra smartphone menggunakan *Neighbor Weighted K-Nearest Neighbor* dengan seleksi fitur *Information Gain*. Jumlah data latih yang digunakan yaitu sebanyak 528 dengan banyak kelas sebanyak 23 dan data uji yang digunakan sebanyak 23. Proses pengujian dilakukan berdasarkan perancangan yang dibuat yaitu terdapat 5 skenario pengujian antara lain pengaruh nilai K pada seleksi fitur *Information Gain* terhadap akurasi NWKNN, perbandingan perhitungan jarak *Manhattan Distance*, *Euclidean Distance* dan *Cosine Similarity* terhadap NWKNN, pengaruh nilai K NWKNN terhadap akurasi, perbandingan antara KNN dan NWKNN, pengujian *K-Fold Cross Validation*.

### 6.1 Pengujian dan Analisis Pengaruh Jumlah Fitur pada Information Gain

Pengujian ini dilakukan untuk mengetahui pengaruh atau tidaknya jumlah fitur yang diseleksi pada *Information Gain* terhadap akurasi. Jumlah fitur yang dipilih diseleksi pada metode *Information gain* dengan jumlah 5, 10, 15, 20, 25, 30, 35, 40, 45 dan 50. Nilai yang tetap yaitu jumlah data uji sebanyak 23, nilai K pada NWKNN yaitu 5 dan perhitungan jarak menggunakan *Cosine Similarity*. Hasil pengujian pengaruh jumlah fitur terhadap akurasi dapat dilihat pada Tabel 6.1.

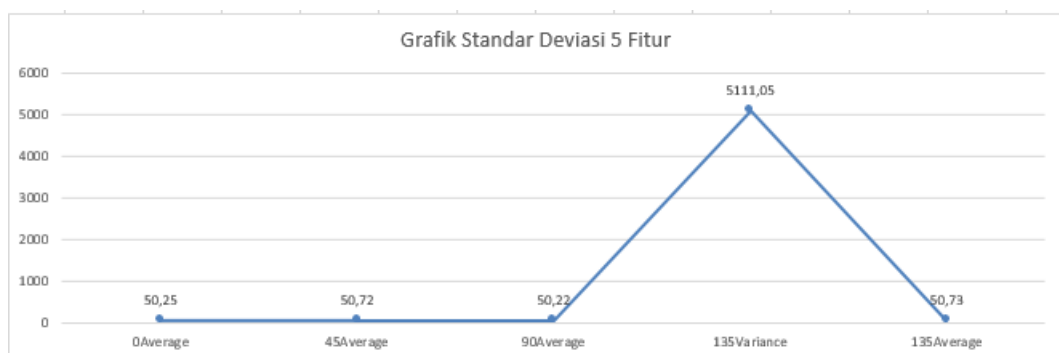
**Tabel 6.1 Hasil Pengujian Pengaruh Jumlah Fitur pada Information Gain**

Jumlah Data Uji	Jumlah Fitur	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	5	5	Cosine Similarity	47,82 %
	10			69,56 %
	15			86,96 %
	20			60,86 %
	25			65,22 %
	30			65,22 %
	35			65,22 %
	40			65,22 %
	45			65,22 %
	50			65,22 %
	60			60,86%
	70			60,86%

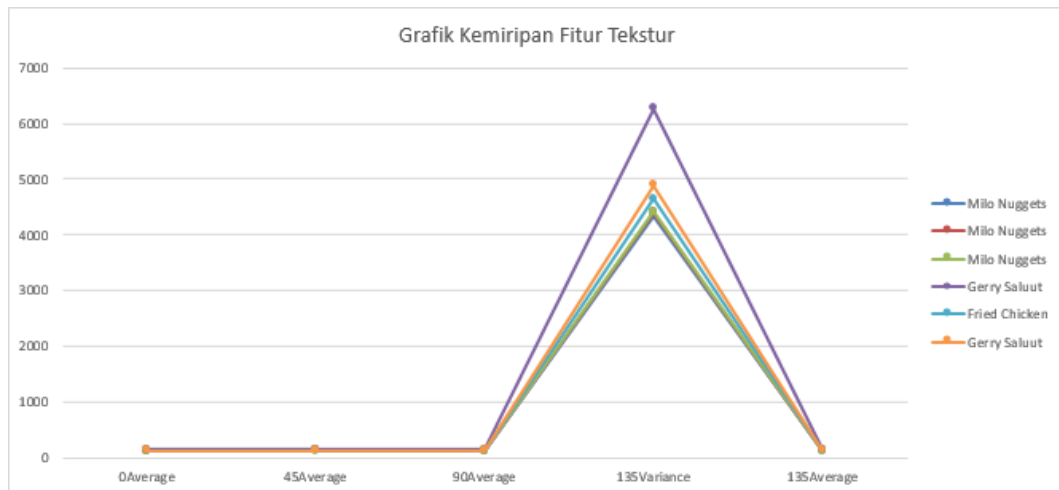


**Gambar 6.1 Grafik Standar Deviasi 15 Fitur**

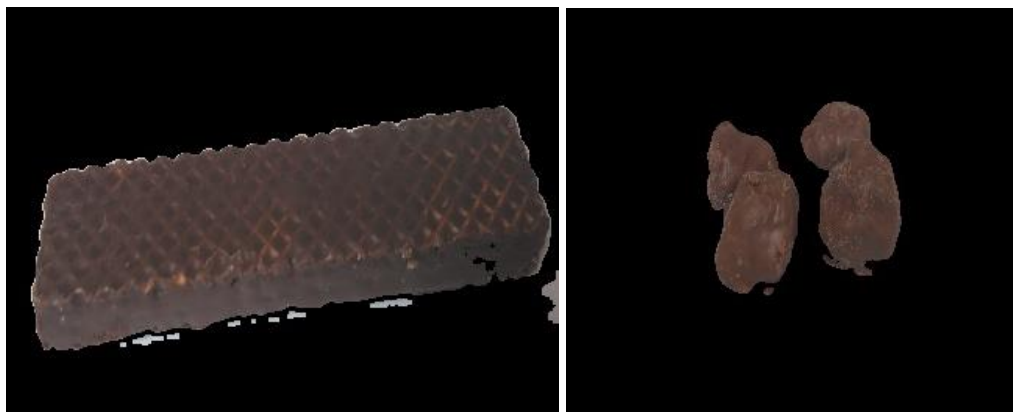
Berdasarkan hasil pengujian pengaruh jumlah fitur *Information Gain* terhadap hasil akurasi menghasilkan hasil akurasi tertinggi yaitu 86,96 % dengan jumlah fitur 15. Hal ini menunjukkan dengan jumlah fitur 15 merupakan fitur yang menghasilkan nilai yang berpengaruh untuk klasifikasi. Fitur sejumlah 15 yaitu *0Variance*, *0Average*, *0Sum Entropy*, *45Variance*, *45Average*, *45Sum Entropy*, *90Variance*, *90Average*, *90Sum Entropy*, *135Variance*, *135Average*, *135Sum Entropy*, *BMean*, *BVariance*, *VVariance* dengan masing-masing nilai standar deviasi yaitu 5072,178172 , 50,25205492 , 283,2650551 , 5109,747057 , 50,71810851 , 285,4903761 , 5069,830404 , 50,21675308 , 283,0135873 , 5111,048278 , 50,72973563 , 285,7164445 , 25,19505706 , 22,38873188 , 22,38873188. Dari Gambar 6.1 dapat dilihat bahwa hasilnya mempunyai pola yang mirip dikarenakan fitur tekstur yang dipakai sama dengan perbedaan arah dari matriks co-occurrence dan fitur warna juga sama dengan perbedaan warna B dan V. Hasil pengujian bernilai tinggi dikarenakan nilai standar deviasi yang cenderung tinggi artinya data pada data latih beragam atau bervariasi. Selain itu terdapat kombinasi fitur antara fitur tekstur dan warna yang bisa membedakan jika ada tekstur yang sama tetapi warnanya berbeda atau jika warnanya sama teksturnya bisa berbeda.



**Gambar 6.2 Grafik Standar Deviasi 5 Fitur**



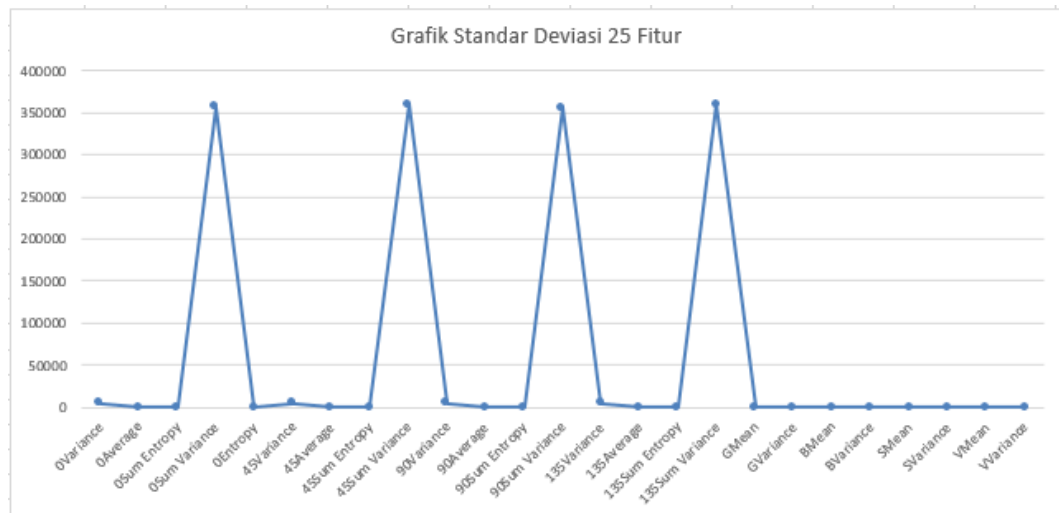
**Gambar 6.3 Grafik Kemiripan Fitur Tekstur**



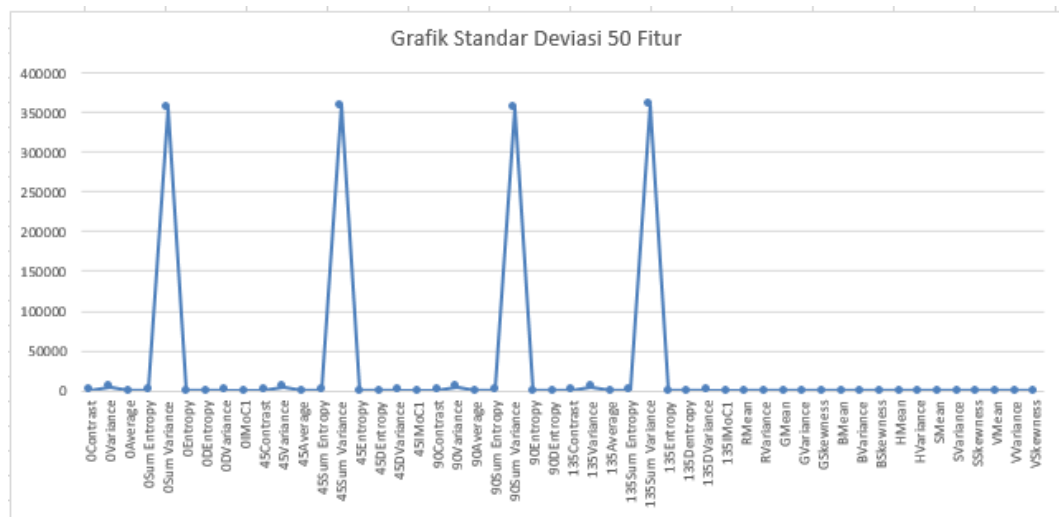
**Gambar 6.4 Citra Makanan dengan Tekstur Mirip**

Hasil pengujian terendah terdapat pada fitur dengan jumlah 5 dengan akurasi 47,82 %. Fitur tersebut adalah 0Average, 45Average, 90Average, 135Variance, 135Average dengan nilai standar deviasi yaitu 50,25205492 , 50,71810851 , 50,21675308 , 5111,048278 , 50,72973563. Hasil akurasi rendah dikarenakan fitur yang telah diseleksi hanya terdapat fitur tekstur saja dengan nilai standar deviasi rendah yang artinya keberagaman pada data latih cenderung mirip. Jika fitur tekstur saja maka hasil tekstur mirip dengan tekstur lainnya sedangkan kelas yang di klasifikasikan bukan kelas yang asli. Seharusnya fitur yang bisa menjadi pembeda adalah fitur warna. Pada data uji ke 12 dengan kelas Gerry Saluut mempunyai hasil klasifikasi dengan K = 5 yaitu kelas Milo Nugget. Tetangga terdekat yang dihasilkan yaitu Fried Chicken, Gerry Saluut, Milo Nuggets, Milo Nuggets dan Milo Nuggets. Fitur-fitur dari kelas tersebut mempunyai tingkat kemiripan yang tinggi. Hasil nilai dari masing masing fitur dapat dilihat pada grafik Gambar 6.3 dan visualisasi Gambar 6.4 diatas menunjukkan persamaan fitur tekstur pada citra.





**Gambar 6.5 Standar Deviasi 25 Fitur**



**Gambar 6.6 Standar Deviasi 50 Fitur**

Hasil pengujian pada fitur 25 sampai 50 menghasilkan akurasi yang sama yaitu 65,22 %. Hasil tersebut menunjukkan bahwa pada fitur 25 sampai 50 tidak berpengaruh pada hasil klasifikasi. Dari hasil grafik 6.3 dan 6.4 menunjukkan hasil standar deviasi membentuk pola yang sama. Dengan semakin menghapus fitur yang tidak berguna untuk klasifikasi akan mempercepat proses komputasi.

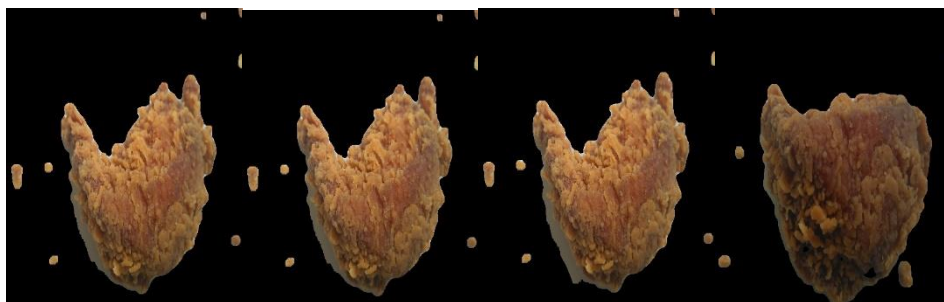
## 6.2 Pengujian dan Analisis Pengaruh Nilai K NKWNN

Pengujian pengaruh nilai K pada NKWNN dilakukan untuk mengetahui pengaruh atau tidaknya nilai K terhadap hasil akurasi ketika nilai K tersebut diubah. Pengujian ini dilakukan dengan mengatur nilai K secara acak dengan nilai 3, 5, 7, 9, 11, 13, 15, 17, 19, 21. Nilai tetap yang digunakan antara lain jumlah data uji yaitu sejumlah 23, jumlah fitur yang digunakan adalah 10 fitur dan perhitungan jarak menggunakan *Cosine Similarity*.

**Tabel 6.2 Hasil Pengujian Pengaruh Nilai K NWKNN**

Jumlah Data Uji	Jumlah Fitur	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	15	3	Cosine Similarity	91,3 %
		5		86,95 %
		7		73,91 %
		9		65,21 %
		11		69,56 %
		13		69,56 %
		15		56,52 %
		17		47,82 %
		19		56,52 %
		21		56,52 %

Hasil pengujian pengaruh nilai K NWKNN menghasilkan akurasi tertinggi dengan nilai K=3 yaitu 91,3 %. Hasil tersebut menghasilkan akurasi tertinggi dikarenakan pada data latih terdapat 3 data dengan foto yang sangat mirip dari data uji. Pada hasil uji kedua dengan kelas Fried Chicken menghasilkan skor dari NWKNN yaitu Fried Chicken dengan nilai 2,234 dan kelas yang lainnya bernilai 0. Hal ini menunjukkan bahwasanya hanya terdapat kelas Fried Chicken pada K = 3. Visualisasi gambar dari data uji dan data latih dapat dilihat pada Gambar 6.7. Gambar paling kiri menunjukkan data uji sedangkan pada 3 gambar selanjutnya adalah hasil rekomendasi kelas dari dari NWKNN.



**Gambar 6.7 Visualisasi Hasil Data Uji ke 3 NWKNN**

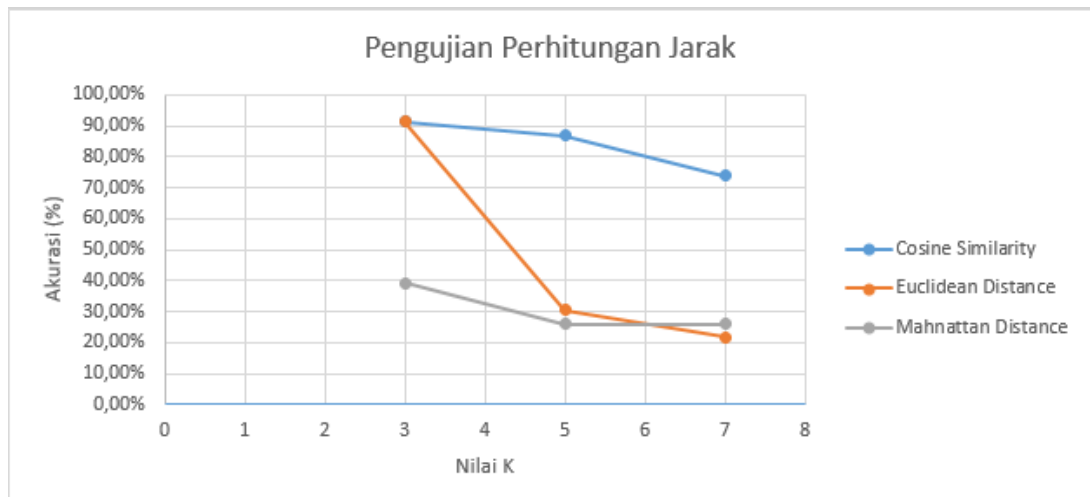
Hasil terendah pada pengaruh nilai K NWKNN adalah 47,82 % dengan K = 17. Hal ini dikarenakan jumlah K terlalu tinggi dan jumlah minimal kelas pada data latih adalah 8 sehingga kelas yang seharusnya diidentifikasi benar akan diidentifikasi salah. Pada data uji kedua dengan kelas Fried Chicken menghasilkan skor NWKNN sebanyak 3 dengan kelas Donat, Telor Dadar dan Fried Chicken dengan masing-masing nilai nya adalah 5,5717 , 3,99997 dan 2,99997.

### 6.3 Pengujian dan Analisis Perhitungan Jarak NWKNN

Pengujian ini dilakukan untuk mengetahui keoptimalan hasil akurasi metode NWKNN dengan perhitungan jarak *Cosine Similarity*, *Euclidean Distance* dan *Manhattan Distance*. Nilai K pada NWKNN diatur menjadi 3 yaitu dengan nilai 3, 5 dan 7 sedangkan jumlah data uji dan jumlah fitur tetap yaitu 23 dan 15. Hasil pengujian dapat dilihat pada Tabel 6.3.

**Tabel 6.3 Hasil Pengujian Perhitungan Jarak NWKNN**

Jumlah Data Uji	Jumlah Fitur	Nilai K NWKNN	Perhitungan Jarak	Hasil Akurasi (%)
23	15	3	<i>Cosine Similarity</i>	91,3 %
			<i>Euclidean Distance</i>	43,47 %
			<i>Manhattan Distance</i>	39,13 %
23	15	5	<i>Cosine Similarity</i>	86,95 %
			<i>Euclidean Distance</i>	30,43 %
			<i>Manhattan Distance</i>	26,08 %
23	15	7	<i>Cosine Similarity</i>	73,91 %
			<i>Euclidean Distance</i>	21,74 %
			<i>Manhattan Distance</i>	26,08 %



**Gambar 6.8 Grafik Hasil Pengujian Perhitungan Jarak NWKNN**

Hasil pengujian perhitungan jarak NWKNN menghasilkan akurasi tertinggi pada perhitungan jarak *Cosine Similarity* dibandingkan dengan *Manhattan* dan *Euclidean*. Pada masing-masing K terlihat pada K = 3 dengan akurasi *Cosim* yaitu 91,3% dan. Hasil akurasi terendah terletak pada K = 7 dengan perhitungan jarak Euclidean dengan hasil 21,74 %. Perhitungan jarak *Cosim* mengasilkan akurasi tertinggi karena *Cosim* cocok dengan metode NWKNN. Pada perhitungan jarak *Cosim* hasil akan diurutkan mulai dari tertinggi ke terendah sedangkan perhitungan jarak *Manhattan* dan *Euclidean* hasil diurutkan mulai yang terendah ke tertinggi. Pada langkah perhitungan skor akan dihitung perkalian antara bobot dengan hasil perhitungan jarak masing-masing kelas. Tabel 6.4 menunjukkan contoh kasus data uji ke 2 kelas Rendang dengan perhitungan jarak Cosim dan Euclidean yang menunjukkan hasil kelas yang berbeda yaitu pada Cosim menghasilkan kelas Rendang dan *Euclidean* menghasilka kelas Strawberry.

**Tabel 6.4 Tabel Pengujian Data Uji Ke-2 Cosim dan Euclidean**

No	Data Uji 2 ke Data Latih			
	Kelas	Hasil Cosim	Bobot	Hasil
1	Rendang	1,0	0,691	3,457
2	Rendang	0,99		
3	Rendang	0,99		
4	Rendang	0,99		
5	Rendang	0,99		
6	Strawberry	0,99	0.75	0,75
7	Mentimun	0,99	0.69	0,69
No	Data Uji 2 ke Data Latih			
	Kelas	Hasil Euclidean	Bobot	Hasil
1	Rendang	0	0,691	63,51
2	Rendang	16,9		
3	Rendang	74,95		

4	Rendang	192,34		
5	Strawberry	89,3	0,759	227,49
6	Strawberry	210,1		
7	Gerry Salut	134,97	0,767	103,65

#### 6.4 Pengujian dan Analisis Perbandingan Akurasi Metode KNN dan NWKNN

Pengujian ini dilakukan untuk mengetahui perbandingan hasil akurasi antara KNN dan NWKNN. Nilai K pada KNN dan NWKNN akan diatur sebanyak 3 kali yaitu 3, 5 dan 7. Nilai lainnya seperti jumlah data uji dan jumlah fitur diatur konstan sebanyak 23 dan 15. Hasil pengujian dapat dilihat pada Tabel 6.5.

**Tabel 6.5 Hasil Pengujian Perbandingan Akurasi Metode KNN dan NWKNN**

Jumlah Data Uji	Jumlah Fitur	Nilai K	Perhitungan Jarak	Hasil Akurasi KNN (%)	Hasil Akurasi NWKNN (%)
23	15	3	Cosine Similarity	86,95 %	91,34 %
23	15	5	Cosine Similarity	82,6 %	86,95 %
23	15	7	Cosine Similarity	56,52 %	73,91 %



**Gambar 6.9 Grafik Hasil Pengujian Perbandingan Akurasi Metode KNN dan NWKNN**

Hasil pengujian perbandingan metode K-NN dengan NWKNN menghasilkan akurasi tertinggi pada NWKNN yaitu pada K = 3 dengan hasil

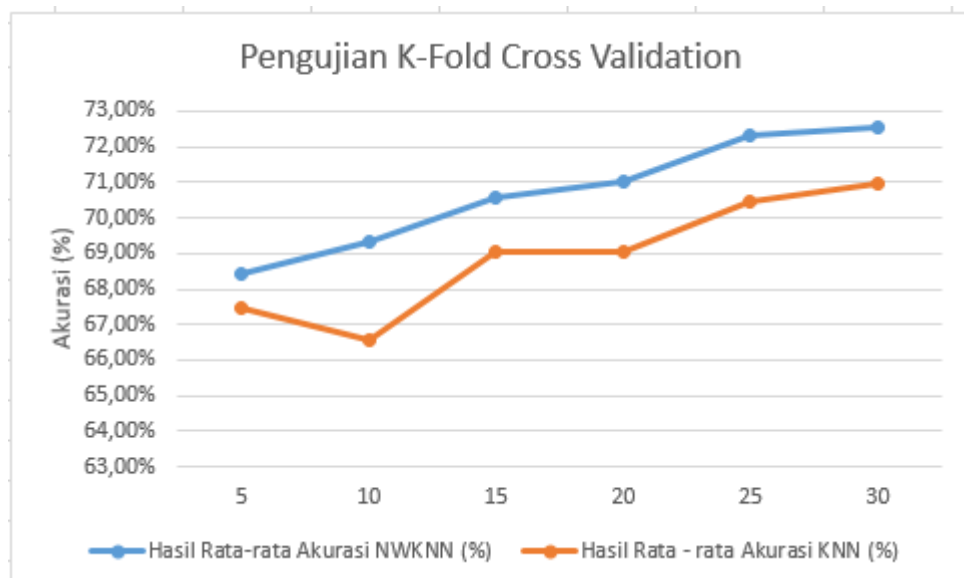
91,34% sedangkan akurasi terendah terelatak pada KNN dengan K = 7 dengan hasil 56,52%. Metode NWKNN lebih baik daripada KNN dikarenakan metode NWKNN dapat melihat data tidak seimbang pada data latih dan dibobotkan. Data latih yang lebih banyak akan dibobotkan lebih kecil sedangkan data latih lebih sedikit akan dibobotkan lebih besar.

## 6.5 Pengujian *K-Fold Cross Validation*

Pengujian ini dilakukan untuk mengetahui keoptimalan hasil akurasi pada saat perubahan data uji terhadap data latih. Pengujian ini dilakukan dengan membagi sebanyak K data latih. Masing masing K digunakan untuk dijadikan data uji dan yang lainnya digunakan sebagai data latih. Hasil pengujian *K-Fold Cross Validation* dapat dilihat pada Tabel 6.6.

**Tabel 6.6 Hasil Pengujian *K-Fold Cross Validation***

Pembagian data uji	Jumlah Fitur	Nilai K	Perhitungan Jarak	Hasil Rata - rata Akurasi KNN (%)	Hasil Rata-rata Akurasi NWKNN (%)
5	10	3	Cosine Similarity	67,46 %	68,41 %
10				66,58 %	69,33 %
15				69,05 %	70,57 %
20				69,04 %	71,02 %
25				70,43%	72.31 %
30				70,96 %	72,53 %



### **Gambar 6.10 Grafik Hasil Pengujian *K-Fold Cross Validation***

Hasil pengujian K-Fold Cross Validation dengan membandingkan 2 metode klasifikasi yaitu KNN dan NWKNN dan membagi data sejumlah 5 sampai 30 menghasilkan akurasi tertinggi pada 30 pembagian data latih dengan akurasi rata-rata adalah 72,53% dan akurasi terendah terletak pada metode KNN dengan pembagian data latih sejumlah 5. Gambar 6.10 menunjukkan bahwa semakin besar jumlah pembagiannya maka hasil akurasi akan semakin meningkat. Hal ini dikarenakan semakin besar pembagian data maka setiap data yang akan diuji jumlahnya akan semakin dan jumlah data latih lebih banyak sehingga pembelajaran yang dilakukan lebih banyak dilakukan. Pada grafik Metode NWKNN lebih baik dari metode KNN dikarenakan hasil akurasinya selalu lebih besar. Hal ini dikarenakan metode NWKNN dapat membobotkan data tidak seimbang dengan bobot terbesar diberikan kepada data latih yang sedikit sedangkan bobot terkecil diberikan pada data latih yang banyak.

## BAB 7 KESIMPULAN

### 7.1 Kesimpulan

Berdasarkan hasil penelitian tentang klasifikasi jenis citra makanan menggunakan *Neighbor Weighted K-Nearest Neighbor (NWKNN)* dengan seleksi fitur *Information Gain* dapat disimpulkan sebagai berikut.

1. Dari hasil pengujian yang sudah dilakukan, penggunaan metode *NWKNN* cocok untuk klasifikasi citra makanan dengan data tidak seimbang dengan data latih sebanyak 529 citra, data uji sebanyak 23 citra dengan kelas data latih terbanyak pada kelas Pisang Kuning dan Biskuit Coklat sebanyak 40 data sedangkan data latih terkecil pada kelas Soba Mie, Nasi Merah, Kemangi, Selada, Telor Dadar, Mie Gepeng dan Indomie Goreng sejumlah 8 data. Hasil akurasi tertinggi diperoleh pada  $K = 3$  dengan perhitungan jarak *NWKNN* dengan jumlah fitur 15 yaitu 91,3%. Akurasi terendah yang dihasilkan adalah 47,82%. Dari hasil pengujian *K-Fold Cross Validation* diperoleh akurasi rata-rata tertinggi 72,53% dengan pembagian data uji sebanyak 30. Hasil *K-Fold Cross Validation* cenderung naik ketika pembagian nilai lebih besar. Hal ini dikarenakan ketika pembagian lebih besar maka data uji akan semakin sedikit dan data latih akan semakin besar sehingga metode klasifikasi akan melakukan pembelajaran yang lebih banyak. Metode perhitungan jarak yang cocok untuk klasifikasi *NWKNN* adalah *Cosine Similarity* dengan menghasilkan akurasi jauh lebih baik daripada *Euclidean* dan *Manhattan* yaitu menghasilkan akurasi tertinggi 91,3%. Hal ini dikarenakan hasil dari *Cosine Similarity* diurutkan berdasarkan yang terbesar ke yang terkecil sehingga ketika dilakukan perkalian terhadap bobot hasilnya akan sesuai dengan kelas aslinya. Perbandingan metode KNN dan *NWKNN* menyimpulkan metode *NWKNN* lebih baik untuk kasus data tidak seimbang dengan menghasilkan akurasi lebih baik dari *NWKNN* yaitu 91,34% sedangkan KNN yaitu 86,95%.
2. Metode *Information Gain* lebih baik terhadap hasil klasifikasi karena metode ini berhasil menyeleksi fitur-fitur yang kurang relevan. *Information Gain* memilih fitur-fitur terbaik sebanyak 15 fitur yaitu 0Variance , 0Average , 0Sum Entropy , 45Variance , 45Average , 45Sum Entropy , 90Variance , 90Average , 90Sum Entropy , 135Variance , 135Average , 135Sum Entropy , BMean , BVariance , VVariance dengan akurasi sebesar 86,96%. Hal ini dikarenakan terdapat kombinasi fitur tekstur dan warna yang akan berpengaruh pada klasifikasi. *Information Gain* memilih fitur sebanyak 5 menghasilkan akurasi paling sedikit yaitu 47,82% dikarenakan fitur yang dipilih yaitu 0Average , 45Average , 90Average , 135Variance , 135Average terlalu sedikit dan kurang ada keberagaman data pada fitur tersebut. Pada fitur yang dipilih *Information Gain* sebanyak 25-50 mempunyai hasil yang konstan yaitu 65,22 %. Hal ini menunjukkan fitur-fitur tersebut tidak berpengaruh pada hasil klasifikasi.



## 7.2 Saran

Penelitian yang sudah dilakukan masih banyak yang perlu dikembangkan lagi agar menjadi lebih baik. Adapun saran yang dapat diberikan pada kelanjutan penelitian klasifikasi jenis citra makanan menggunakan *NWKNV* dengan seleksi fitur *Information Gain* yaitu:

1. Perlunya ditambahkan metode ekstraksi fitur lainnya seperti bentuk dikarenakan tekstur dan warna saja tidak cukup untuk mengidentifikasi jenis makanan.
2. Perlunya perbandingan dengan metode seleksi fitur lainnya untuk mengetahui apakah metode seleksi fitur lainnya lebih baik daripada metode seleksi fitur *Information Gain*.
3. Perlunya ditambahkan data latih yang lebih banyak, lebih mirip dalam kelas yang sama dan lebih beragam antar kelas agar citra bisa diidentifikasi lebih baik.

## DAFTAR PUSTAKA

- Aini, S. H. A., Sari, Y. A. and Arwan, A. (2018) 'Seleksi Fitur Information Gain untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode K-Nearest Neighbor dan Naïve Bayes', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(9), pp. 2546–2554.
- Arsita, A. *et al.* (2017) 'SIMULAKRA BAUDRILLARD DALAM MULTIDIMENSI POSMODERNISME : KAJIAN FOTOGRAFI MAKANAN', 13(2), pp. 85–98.
- Chormunge, S. and Jena, S. (2016) 'Efficient Feature Subset Selection Algorithm for High Dimensional Data', *International Journal of Electrical and Computer Engineering (IJECE)*, 6(4), p. 1880. doi: 10.11591/ijece.v6i4.9800.
- DEDY, Y. (2015) 'Identifikasi Kualitas Daging Sapi Berbasis Android Dengan Ekstraksi Fitur Warna Dan Klasifikasi Knn', *Skripsi, Fakultas Ilmu Komputer*, pp. 1–8.
- Dewi, R. K. and Ginardi, R. V. H. (2014) 'Identifikasi Penyakit pada Daun Tebu dengan Gray Level Co-Occurrence Matrix dan Color Moments', *Jurnal Teknologi Informasi dan Ilmu Komputer*, 1(2), p. 70. doi: 10.25126/jtiik.201412114.
- Fadila, P. N. *et al.* (2016) 'Identifikasi Jenis Attention Deficit Hyperactivity Disorder (Adhd) Pada Anak Usia Dini Menggunakan Metode Neighbor Weighted K-Nearest Neighbor (Nwknk)', *Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 3(3), pp. 194–200.
- Han, J., Kamber, M. and Pei, J. (2012) *Data Mining: Concepts and Techniques, San Francisco, CA, itd: Morgan Kaufmann*. doi: 10.1016/B978-0-12-381479-1.00001-0.
- Haralick, R. M. and Shanmugam, K. (1973) '04309314'. doi: 10.1109/TSMC.1973.4309314.
- Harfiya, L. N., Widodo, A. W. and Wihandika, R. C. (2017) 'Verifikasi Citra Tanda Tangan Berdasarkan Ciri Pyramid Histogram of Oriented Gradient ( PHOG ) Menggunakan Metode Klasifikasi K-Nearest Neighbor', 1(10), pp. 1162–1171.
- Hartono, B., SENDI\_U, V. L.-P. and 2017, U. (2017) 'Pencarian Isi Citra Menggunakan Metode Minkowski Distance', *Unisbank.Ac.Id*, (1), pp. 34–39. Available at: <https://www.unisbank.ac.id/ojs/index.php/sendu/article/view/4993>.
- Honeycutt, C. E. and Plotnick, R. (2008) 'Image analysis techniques and gray-level co-occurrence matrices (GLCM) for calculating bioturbation indices and characterizing biogenic sedimentary structures', *Computers and Geosciences*, 34(11), pp. 1461–1472. doi: 10.1016/j.cageo.2008.01.006.
- Kasim, A. A. (2014) 'Klasifikasi Citra Batik Menggunakan Jaringan Syaraf Tiruan Berdasarkan Gray Level Co- Occurrence Matrices ( GLCM )', pp. 7–13.
- Kemal, E. and Nihat, Y. (2014) 'Shifting Colors to Overcome not Realizing Objects

Problem due to Color Vision Deficiency', *Conf. on Advances in Computing, Electronics and Electrical Technology*, pp. 11–14. Available at: [https://www.researchgate.net/publication/284698928\\_Shifting\\_Colors\\_to\\_Overcome\\_not\\_Realizing\\_Objects\\_Problem\\_due\\_to\\_Color\\_Vision\\_Deficiency?\\_sg=X36VnIXthG1QcSi\\_UVTRbOVTOfMSN86bQDsQyVSTQlq5Ur0jghOMu7LLYABGskYOlc1uVpzSmQ](https://www.researchgate.net/publication/284698928_Shifting_Colors_to_Overcome_not_Realizing_Objects_Problem_due_to_Color_Vision_Deficiency?_sg=X36VnIXthG1QcSi_UVTRbOVTOfMSN86bQDsQyVSTQlq5Ur0jghOMu7LLYABGskYOlc1uVpzSmQ).

Kolivand, H. and Sunar, M. S. (2011) 'Real-Time Sky Color with Effect of Sun's Position', *International Journal of Scientific and Engineering Research*, 2(11), pp. 2–7.

Layona, R., Tunardi, Y. and Tanoto, D. F. (no date) 'IMAGE RETRIEVAL BERDASARKAN FITUR WARNA , Penelitian Text Based Image Retrieval ( TBIR )', (9), pp. 1073–1085.

Liu, F., Liu, X. and Chen, Y. (2014) 'An efficient detection method for rare colored capsule based on RGB and HSV color space', *Proceedings - 2014 IEEE International Conference on Granular Computing, GrC 2014*, pp. 175–178. doi: 10.1109/GRC.2014.6982830.

Mahardika, A., Sari, Y. A. and Dewi, C. (2018) 'Sistem Temu Kembali Citra Lubang Jalan Aspal Berdasarkan Tingkat Kerusakan Menggunakan Ekstraksi Fitur Gray Level Co-occurrence Matrix', 2(10), pp. 3811–3821.

Nejati, H. *et al.* (2016) 'Smartphone and mobile image processing for assisted living: Health-monitoring apps powered by advanced mobile imaging algorithms', *IEEE Signal Processing Magazine*, 33(4), pp. 30–48. doi: 10.1109/MSP.2016.2549996.

Patil, J. K. *et al.* (2011) 'Color Feature Extraction of Tomato Leaf Diseases', *International Journal of Engineering Trends and Technology*, 2(2), pp. 72–74. Available at: <http://www.internationaljournalsrg.org>.

Putri, P. A., Ridok and Indriati (2013) 'Implementasi Metode Improved K-Nearest Neighbor pada Analisis Sentimen Twitter Berbahasa Indonesia', *Repositori Jurnal Mahasiswa PTIIK UB*, 2, pp. 1–8.

Rahayuni, T., Sari, Y. A. and Adinugroho, S. (2019) 'Pencarian Resep Makanan Berdasarkan Citra Makanan Menggunakan Ekstraksi Fitur Simple Morphological Shape Descriptors dan Color Moment', 3(2), pp. 1901–1907.

Ridok, A. and Latifah, R. (2015) 'Klasifikasi Teks Bahasa Indonesia Pada Corpus Tak Seimbang Menggunakan NWKNN', *Konferensi Nasional Sistem dan Informatika 2015*, pp. 222–227.

Sasano, S., Han, X. H. and Chen, Y. W. (2017) 'Food recognition by combined bags of color features and texture features', *Proceedings - 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2016*, pp. 815–819. doi: 10.1109/CISP-BMEI.2016.7852822.

Satria, D. and Mushthofa (2013) 'Perbandingan Metode Ekstraksi Ciri Histogram

dan PCA untuk Mendeteksi Stoma pada Citra Penampang Daun Freycinetia', *Jurnal Ilmu Komputer Agri-Informatika*, 2, pp. 20–28.

Sheen, S. and Rajesh, R. (2008) 'Network intrusion detection using feature selection and decision tree classifier', *IEEE Region 10 Annual International Conference, Proceedings/TENCON*. doi: 10.1109/TENCON.2008.4766847.

Siqueira, F. R. De, Schwartz, W. R. and Pedrini, H. (no date) 'Multi-Scale Gray Level Co-Occurrence Matrices for Texture Description'.

Xie, Z. *et al.* (2010) 'Texture image retrieval based on gray level co-occurrence matrix and singular value decomposition', *2010 International Conference on Multimedia Technology, ICMT 2010*, (1), pp. 3–5. doi: 10.1109/ICMULT.2010.5629822.