



Identifikasi Tingkat Kematangan Buah Tomat Menggunakan Metode Color Processing dan KNN



C2:

152022109 Shafira Kurnia
152022117 Angeline Apriliana
152022120 Husnul Arifah
152022126 Crystal Fandy
152022147 M. Mirsab A.

KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa. Atas rahmat dan hidayah-Nya, penulis bisa menyelesaikan laporan yang berjudul “Aplikasi Identifikasi Tingkat Kematangan Buah Tomat dengan Metode Analisis Warna dan KNN”. Adapun pembuatan laporan ini ditujukan sebagai Tugas Akhir Mata Kuliah IFB-208 Pengolahan Citra Digital.

Tidak lupa penulis mengucapkan rasa terima kasih kepada Ibu Irma Amelia Dewi, S.Kom, M.T. selaku dosen Mata Kuliah Pengolahan Citra Digital, Teh Ayala Qaulam Putri, Kang Rifqi Lukmansyah, dan Teh Yuren Prisilla selaku Asisten Laboratorium Pengolahan Citra Digital yang telah membantu penulis dalam merancang laporan ini. Penulis juga mengucapkan terima kasih kepada teman-teman yang telah memberikan masukan dalam pembuatan laporan ini.

Laporan ini berisikan mengenai aplikasi untuk menjalankan identifikasi tingkat kematangan buah tomat yang terdapat tiga kategori yaitu matang, setengah matang dan mentah. Penulis menyadari adanya kekurangan pada laporan ini. Oleh karena itu, saran dan kritik senantiasa diharapkan demi perbaikan laporan ini. Penulis juga berharap agar laporan ini dapat memberikan pengetahuan tentang Pengolahan Citra Digital kepada orang-orang.

Bandung, 18 Mei 2024

Penulis

DAFTAR ISI

KATA PENGANTAR.....	1
DAFTAR ISI.....	2
DAFTAR GAMBAR.....	4
BAB I PENDAHULUAN.....	6
1.1 Latar Belakang.....	6
1.2 Tujuan Penelitian.....	7
1.3 Rumusan Masalah.....	7
1.4 Ruang Lingkup.....	7
1.5 Batasan Masalah.....	7
1.6 Pembagian Tugas.....	8
BAB II LANDASAN TEORI.....	9
2.1 Pengolahan Citra Digital.....	9
2.2 Python.....	9
2.3 Pycharm.....	9
2.4 HSV (Hue, Saturation, Value).....	10
2.5 KNN.....	10
2.6 Library.....	11
2.6.1 OpenCV.....	11
2.6.2 Matplotlib.....	11
2.6.3 Numpy.....	11
2.6.4 Qt5.....	12
2.6.5 Scikit Learn.....	12
BAB III METODE PENELITIAN.....	13
3.1 Deskripsi Sistem.....	13
3.2 Alur Proses Sistem Kerja.....	14
3.2.1 Blok Diagram.....	14
3.2.2 Flowchart.....	15
3.3 Parameter Objek yang Diteliti.....	15
3.3.1. Data Parameter HSV Buah Tomat Matang.....	16
3.3.2. Data Parameter HSV Buah Tomat Setengah Matang.....	17
3.3.3. Data Parameter HSV Buah Tomat Mentah.....	17
3.4 Operasi pre-processing.....	17
3.4.1 Konversi Gambar ke RGB.....	17
3.4.2 Konversi Gambar ke HSV.....	18
3.4.3 Masking Warna.....	18
3.4.4 Menghitung Rata-rata Warna.....	19
3.4.5 Operasi Morfologi.....	20
3.4.6 Identifikasi Kontur.....	21

BAB IV IMPLEMENTASI DAN PENGUJIAN.....	22
4.1 Coding.....	22
4.1.1 Import Library.....	22
4.1.2 Mengatur Atribut Aplikasi.....	22
4.1.3 Inisialisasi.....	22
4.1.4 Membuat Antarmuka Pengguna Pertama.....	23
4.1.5 Membuat Antarmuka Pengguna Kedua.....	23
4.1.6 Memuat Gambar dan Mengkonversikan Gambar.....	23
4.1.7 Menampilkan Gambar di Antarmuka Pengguna.....	24
4.1.8 Melakukan Pra-pemrosesan Gambar.....	24
4.1.9 Melakukan Prediksi Kategori.....	26
4.1.10 Melakukan Konversi.....	26
4.1.11 Melakukan Masking.....	27
4.1.12 Melakukan Morfologi.....	28
4.1.13 Melakukan Kontur.....	31
4.1.14 Menampilkan Proses Pengolahan Citra.....	32
4.1.15 Menampilkan Inisialisasi Aplikasi Qt.....	32
4.1.16 Membuat Dataset.....	33
4.2 Design GUI.....	35
4.2.1 Tampilan Pertama.....	35
4.2.2 Tampilan About.....	35
4.2.3 Tampilan Kedua.....	36
4.3 Gambar Sebelum dan Sesudah diproses.....	37
BAB V KESIMPULAN DAN SARAN.....	48
5.1 Kesimpulan.....	48
5.2 Saran.....	50
DAFTAR PUSTAKA.....	51

DAFTAR GAMBAR

Gambar 2.1 Python.....	9
Gambar 2.2 PyCharm IDE.....	9
Gambar 2.3 Ruang Warna HSV.....	10
Gambar 2.4 Algoritma KNN.....	10
Gambar 2.5 Qt5.....	12
Gambar 3.1 Blok Diagram Aplikasi.....	14
Gambar 3.2 Flowchart Aplikasi.....	15
Gambar 3.3 Flowchart Konversi Gambar ke RGB.....	17
Gambar 3.4 Flowchart Konversi Gambar ke HSV.....	18
Gambar 3.5 Flowchart Masking Warna.....	19
Gambar 3.6 Flowchart Menghitung Mean Warna (1).....	19
Gambar 3.7 Flowchart Menghitung Mean Warna (2).....	20
Gambar 3.8 Flowchart Morfologi.....	20
Gambar 3.9 Flowchart Identifikasi Kontur.....	21
Gambar 4.1 Koding Import Library.....	22
Gambar 4.2 Koding Mengatur Atribut Aplikasi.....	22
Gambar 4.3 Koding Inisialisasi.....	22
Gambar 4.4 Koding Membuat Antarmuka Pengguna Pertama.....	23
Gambar 4.5 Koding Membuat Antarmuka Pengguna Kedua.....	23
Gambar 4.6 Koding Memuat Gambar dan Mengkonversikan Gambar.....	24
Gambar 4.7 Koding Menampilkan Gambar di Antarmuka Pengguna.....	24
Gambar 4.8 Kodingan Melakukan Pra-Pemrosesan Gambar (1).....	24
Gambar 4.9 Koding Melakukan Pra-pemrosesan Gambar (2).....	25
Gambar 4.10 Koding Melakukan Pra-pemrosesan Gambar (3).....	25
Gambar 4.11 Koding Melakukan Prediksi Kategori.....	26
Gambar 4.12 Koding Melakukan Konversi.....	26
Gambar 4.13 Koding Melakukan Masking Ripe.....	27
Gambar 4.14 Koding Melakukan Masking Un-Ripe.....	27
Gambar 4.15 Koding Melakukan Masking Half Ripe.....	28
Gambar 4.16 Koding Melakukan Masking All.....	28
Gambar 4.17 Koding Morfologi Ripe.....	29
Gambar 4.18 Koding Morfologi Un-Ripe.....	29
Gambar 4.19 Koding Morfologi Half Ripe.....	29
Gambar 4.20 Koding Morfologi All.....	31
Gambar 4.21 Koding Kontur All.....	31
Gambar 4.22 Koding Proses Pengolahan Citra.....	32
Gambar 4.23 Koding Inisialisasi Aplikasi Qt.....	32
Gambar 4.24 Koding Membuat Dataset (1).....	33

Gambar 4.25 Koding Membuat Dataset (2).....	33
Gambar 4.26 Koding Membuat Dataset (3).....	34
Gambar 4.27 Koding Membuat Dataset (4).....	34
Gambar 4.28 UI Tampilan Pertama.....	35
Gambar 4.29 UI Tampilan About.....	36
Gambar 4.30 UI Tampilan Kedua.....	36

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi citra digital telah menjadi alat inovatif yang semakin penting dalam berbagai bidang, termasuk pertanian, di mana ia memainkan peran vital dalam identifikasi kematangan buah dan sayuran seperti tomat. Dengan memanfaatkan kamera dan sensor, teknologi ini mampu menangkap gambar dengan detail luar biasa. Analisis citra digital memungkinkan pendekripsi karakteristik visual yang signifikan, seperti warna, tekstur, dan bentuk, yang merupakan indikator utama kematangan.

Dalam pertanian tradisional, penilaian kematangan tomat umumnya dilakukan secara manual dengan mengandalkan indra penglihatan dan pengalaman petani, yang sering kali menghasilkan ketidakstabilan dan subjektivitas. Metode digital ini mengantikan proses manual dengan pendekripsi yang lebih objektif dan akurat, memungkinkan setiap tomat dievaluasi dengan kriteria yang seragam.

Keuntungan utama dari metode ini adalah peningkatan akurasi dan konsistensi dalam penentuan kematangan. Warna tomat, misalnya, dapat dianalisis dengan menggunakan model warna yang kompleks untuk mendekripsi transisi dari hijau ke merah, yang merupakan indikator utama kematangan. Dengan analisis citra digital, proses evaluasi menjadi lebih cepat dan efisien, sangat penting dalam skala produksi besar di mana waktu dan tenaga kerja menjadi faktor signifikan.

Selain itu, penggunaan citra digital memberikan manfaat ekonomi dan lingkungan. Dengan memastikan tomat dipanen pada tingkat kematangan yang optimal, nilai gizi dan rasa buah dapat dimaksimalkan, yang pada gilirannya mengikat kepuasan konsumen dan potensi penjualan. Tomat yang dipanen pada waktu yang tepat memiliki daya simpan yang lebih baik, mengurangi kerugian pasca panen akibat pembusukan atau penurunan kualitas. Dari perspektif lingkungan, efisiensi panen yang lebih tinggi berarti penggunaan sumber daya seperti air dan pupuk dapat dioptimalkan, mendukung praktik pertanian yang berkelanjutan.

Secara keseluruhan, identifikasi kematangan tomat menggunakan citra digital tidak hanya merepresentasikan kemajuan teknologi dalam bidang pertanian, tetapi juga menawarkan solusi praktis untuk meningkatkan kualitas hasil panen, efisiensi operasional, dan keberlanjutan lingkungan. Dengan perkembangan terus-menerus dalam teknologi

pencitraan dan analisis data, metode ini memiliki potensi besar untuk diterapkan pada berbagai jenis tanaman lainnya.

1.2 Tujuan Penelitian

Tujuan yang hendak dicapai dalam Laporan ini adalah sebagai berikut:

- a. Mengetahui cara melakukan pendekripsi kematangan tomat pada gambar.
- b. Mengetahui cara melakukan identifikasi kematangan tomat.

1.3 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam rangka percobaan identifikasi kematangan tomat yaitu sebagai berikut:

- a. Bagaimana cara melakukan pendekripsi kematangan tomat pada gambar?
- b. Bagaimana cara melakukan identifikasi kematangan tomat?

1.4 Ruang Lingkup

Laporan ini secara khusus mengedepankan perhatian pada:

1. Pengenalan terhadap berbagai modul yang digunakan adalah OpenCV, PyQt5, dan beberapa modul pendukung lainnya.
2. Pembuatan antarmuka menggunakan aplikasi Qt Designer bawaan dari python untuk memasukkan button dan fungsi memasukkan load image.
3. Pembuatan prototipe aplikasi yang menggabungkan semua komponen fungsi.
4. Menguji aplikasi dengan berbagai gambar tomat untuk memastikan akurasi dan konsistensi dalam identifikasi tingkat kematangan.

1.5 Batasan Masalah

Untuk memastikan fokus dan keterbatasan yang jelas dalam penelitian ini, beberapa batasan masalah telah ditetapkan sebagai berikut:

1. Aplikasi kurang optimal dalam memprediksi kematangan apabila pada *background* terdapat rentang warna antara hijau, merah, dan kuning.
2. Apabila dalam satu gambar terdapat banyak buah tomat maka yang muncul adalah hasil yang paling dominan ada.

1.6 Pembagian Tugas

Dalam pelaksanaan project tugas akhir mata kuliah Pengolahan Citra Digital kelompok kami telah membagi tugas secara jelas untuk memastikan efisiensi dan efektivitas dalam penggerjaan setiap bagian. Pembagian tugas adalah sebagai berikut:

- a. Shafira Kurnia - 152022109
 - Perancangan Aplikasi Project
 - Perancangan GUI
 - Cover Laporan
 - Edit Video
 - Membuat Kata Pengantar
 - BAB IV (4.3)
 - BAB V
- a. Angeline Apriliana - 152022117
 - Membuat Daftar Isi dan Daftar Gambar
 - BAB II
 - BAB IV (4.1)
- b. Husnul Arifah - 152022120
 - BAB III (3.3, 3.4)
- c. Crystal Fandy - 152022126
 - BAB I
 - BAB III (3.1, 3.4)
 - BAB IV (4.1, 4.2)
- d. M. Mirsab Anwar - 152022147
 - BAB III (3.2)

BAB II

LANDASAN TEORI

2.1 Pengolahan Citra Digital

Pengolahan citra digital (*Digital Image Processing*) adalah sebuah disiplin ilmu yang mempelajari tentang teknik-teknik mengolah citra yang berkaitan dengan perbaikan kualitas terhadap suatu gambar (meningkatkan kontras, perubahan warna, restorasi citra), transformasi gambar (translasi, rotasi transformasi, skala, geometrik), melakukan pemilihan citra ciri (*feature images*) yang optimal untuk tujuan analisis, melakukan penyimpanan data yang sebelumnya dilakukan reduksi dan kompresi, transmisi data, dan waktu proses data. Secara umum pengolahan citra digital dapat diartikan sebagai pemrosesan gambar dua dimensi dengan menggunakan komputer. Citra digital tersebut merupakan sebuah *array* (larik) yang berisikan nilai-nilai *real* maupun komplek yang dapat direpresentasikan dengan deretan bit tertentu.

2.2 Python



Gambar 2.1 Python

Python merupakan salah satu bahasa pemrograman tingkat tinggi yang bersifat *interpreter*, *interactive*, *object oriented*, dan dapat beroperasi hampir di semua *platform* seperti Mac, Linux, dan Windows. Pada Python sendiri tersedia sangat banyak library yang dapat dimanfaatkan sesuai dengan kebutuhan seperti OpenCV, Numpy, PyQt, dan lain sebagainya.

2.3 Pycharm

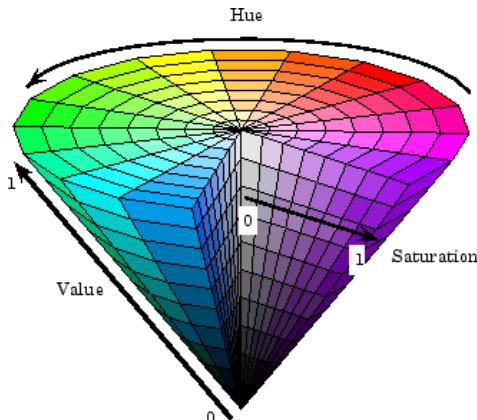


Gambar 2.2 PyCharm IDE

PyCharm adalah Python *Integrated Development Environment* (IDE) yang dikembangkan oleh JetBrains yang secara spesifik digunakan untuk penulisan, pengujian dan

memelihara kode dari bahasa pemrograman Python dengan lebih efisien. Fitur-fitur di dalamnya sudah cukup banyak, mulai dari *auto-complete code*, *coding assistance and analysis*, *syntax and error highlighting*, dan banyak lagi.

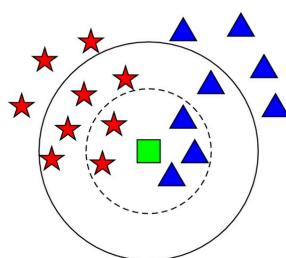
2.4 HSV (Hue, Saturation, Value)



Gambar 2.3 Ruang Warna HSV

Ruang Warna H-S-V HSV mendefinisikan warna dalam terminologi Hue, Saturation dan Value. Keuntungan HSV adalah terdapat warna-warna yang sama dengan yang ditangkap oleh indera manusia juga merupakan hasil perubahan dari ruang warna RGB yang memiliki bentuk kubus menjadi bentuk kerucut (Sanusi, 2020). Sudut warna *hue* (H) merupakan sudut yang berputar melingkar terhadap sumbu tegak yang mana warna merah berada tepat berada pada sudut nol derajat. Komponen warna *saturation* (S) memiliki nilai di antara 0 sampai 1 yang berarti 0 adalah *gray*, dan 1 merupakan warna murni. Sedangkan pada komponen *value* (V) memiliki rentang nilai 0 sampai dengan 1 yang berarti, 0 adalah hitam dan 1 adalah putih.

2.5 KNN



Gambar 2.4 Algoritma KNN

KNN atau *K-Nearest Neighbors* adalah teknik klasifikasi yang memanfaatkan nilai K data terdekat untuk mengklasifikasikan data baru ke dalam kategori tertentu. Kelas dari suatu

data adalah kelas yang terbanyak dari K data terdekat dengannya. Dalam Algoritma KNN, jarak antara data testing dan data training ditentukan untuk melakukan pengujian. Jarak dari kedua data dihitung dengan *Euclidean distance*.

2.6 Library

Library pada Python merupakan suatu kumpulan kode yang telah ditulis sebelumnya (berisi function, class, modul) dan dapat digunakan kembali untuk program lainnya. Berikut beberapa *library* yang digunakan :

2.6.1 OpenCV

OpenCV(*Open Source Computer Vision Library*) adalah perpustakaan komputer *open source* visi dan perangkat lunak pembelajaran. OpenCV dibangun untuk menyediakan infrastruktur umum untuk aplikasi visi komputer dan untuk mempercepat penggunaan persepsi mesin dalam produk komersial. Menjadi produk berlisensi BSD, OpenCV memudahkan bisnis untuk memanfaatkan dan memodifikasi. kode *library* ini memiliki lebih dari 2500 algoritma yang dapat dioptimalkan, yang mencakup satu set lengkap visi komputer klasik dan *state-of-the-art* computer vision dan mesin pembelajaran algoritma.

2.6.2 Matplotlib

Matplotlib adalah pustaka yang digunakan untuk membuat visualisasi statis, animasi, dan interaktif dengan Python. Matplotlib dapat digunakan untuk membuat berbagai visualisasi data seperti diagram batang, *scatter*, *boxplot*, *pie chart*, dan lain sebagainya. Selain itu, Matplotlib digunakan untuk memvisualisasikan data dalam 2D dan 3D dan menghasilkan gambar berkualitas tinggi yang dapat disimpan dalam berbagai format gambar seperti JPEG dan PNG (Hunter, 200).

2.6.3 Numpy

NumPy adalah paket dasar untuk melakukan komputasi ilmiah di Python yang menyediakan objek *array* multidimensi dan objek turunan seperti *masked arrays* dan *matrices*. Hal ini juga menawarkan berbagai operasi aljabar, statistik dasar, simulasi acak, matematika, logika, manipulasi bentuk, pengurutan, pemilihan, I/O, transformasi Fourier diskrit, dan lain-lain. (Harris et al., 2020).

2.6.4 Qt5



Gambar 2.5 Qt5

PyQT5 adalah sebuah *toolkit widget Graphic User Interface* (GUI), yaitu merupakan antarmuka grafik antara bahasa pemrograman Python dan QT. PyQT menyediakan sejumlah pustaka yang dapat digunakan oleh bahasa pemrograman python untuk menjalankan GUI dari QT. Rancangan perangkat lunak akuisisi data detektor gamma buatan Rusia ini memanfaatkan editor *qtdesigner* dari PyQT untuk membuat antarmuka berbasis grafik dan menggunakan bahasa pemrograman Python sebagai program yang berada dibalik antarmuka pengguna berbasis grafik tersebut.

2.6.5 Scikit Learn

Scikit-Learn adalah perpustakaan pembelajaran mesin perangkat lunak gratis untuk bahasa pemrograman Python di bawah lisensi 3-Clause BSD yang merupakan alat sederhana dan efisien untuk analisis data.

BAB III

METODE PENELITIAN

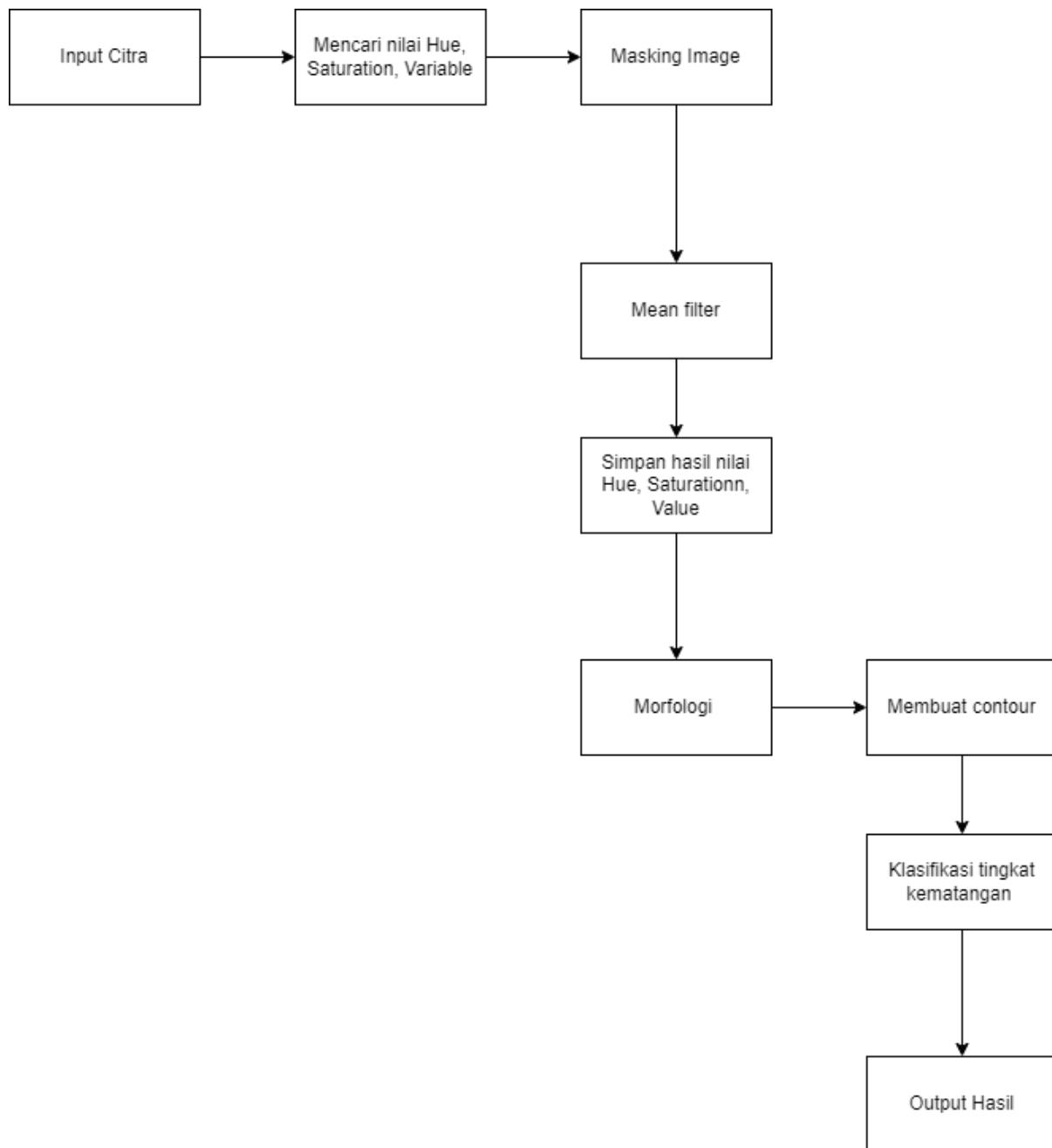
3.1 Deskripsi Sistem

Sistem ini dirancang untuk mendeteksi tingkat kematangan tomat menggunakan bahasa pemrograman Python. Implementasi sistem melibatkan beberapa tahapan penting, termasuk pengembangan algoritma dan penggunaan modul pemrosesan citra. Sistem ini menggunakan berbagai sampel tomat yaitu tomat matang, setengah matang, dan mentah. Kemudian melakukan deteksi tomat tersebut menggunakan dataset *template*. Hasil deteksi tomat akan ditampilkan pada antarmuka pengguna menggunakan PyQt5 dan OpenCV. Hasil deteksi kematangan tomat harus di pra proses agar hasilnya menjadi tepat, tahap-tahap pra proses diantaranya:

- a. Melakukan konversi gambar ke RGB agar menyesuaikan format warna untuk analisis yang lebih tepat.
- b. Melakukan konversi dari gambar RGB ke ruang warna HSV untuk memisahkan warna, kecerahan, dan kejemuhan gambar, memudahkan segmentasi warna untuk mendeteksi kematangan tomat.
- c. Membuat *masking* warna untuk area tomat yang matang, mentah, dan setengah matang.
- d. Menghitung rata-rata warna untuk setiap *mask* yang dihasilkan untuk mempersiapkan data untuk proses klasifikasi.
- e. Menerapkan operasi morfologi untuk membersihkan *noise* pada hasil segmentasi *mask*.
- f. Melakukan identifikasi kontur pada masing-masing *mask* untuk menentukan batas-batas area yang tersegmentasi.

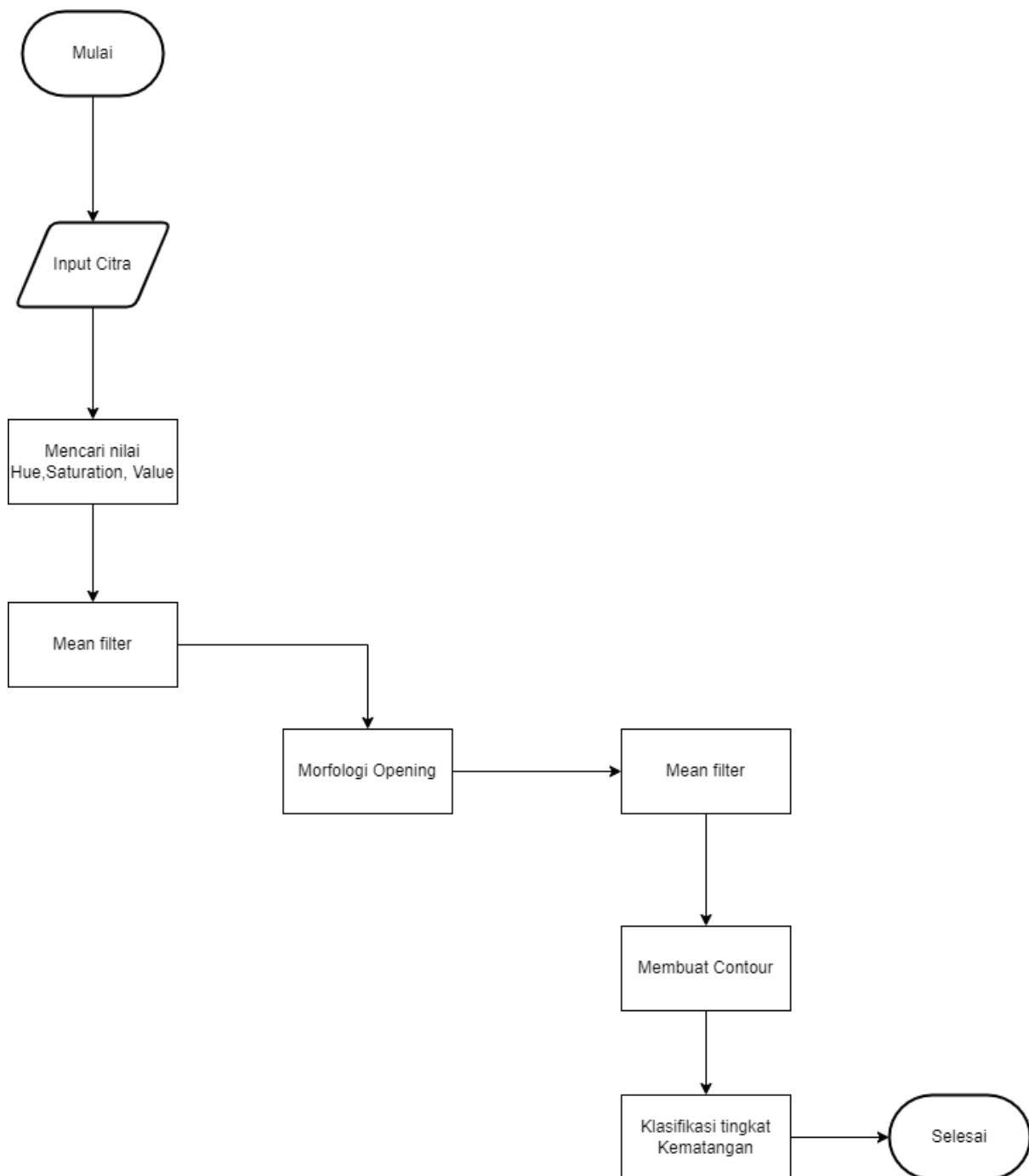
3.2 Alur Proses Sistem Kerja

3.2.1 Blok Diagram



Gambar 3.1 Blok Diagram Aplikasi

3.2.2 Flowchart



Gambar 3.2 Flowchart Aplikasi

3.3 Parameter Objek yang Diteliti

Pada penelitian ini, parameter atau objek yang diteliti adalah kematangan buah tomat yang dilihat dari warnanya. Klasifikasi menggunakan data berupa dataset yang diambil dari kaggle, yang mana dataset berbentuk *images* yang dipisah menjadi tiga kategori pada tiga *folder* yang berbeda, yaitu matang, setengah matang, dan mentah.

Untuk dapat mengklasifikasi apakah buah tomat termasuk ke dalam kategori matang, setengah matang, atau mentah; sistem aplikasi terlebih dahulu di *training* dan dilakukan *testing* terhadap setiap parameternya, dengan cara mengarahkan *path* ke setiap *folder* yang berisi dataset berupa *images* untuk masing-masing kategori, kemudian dilakukan pra-proses gambar dengan mengubah ukuran gambar menjadi 64 x 64 piksel, setelah itu dilakukan konversi *channel* warna dari BGR ke HSV agar dapat dilakukan ekstraksi warna yang lebih akurat.

Sesudah melakukan perubahan ukuran piksel dan konversi *channel* warna, dilakukan proses *masking* untuk mendeteksi warna menggunakan parameter rentang warna bawah dan atas untuk masing-masing warna merah untuk matang, kuning untuk setengah matang, dan hijau untuk mentah. Selanjutnya menghitung rata-rata dari nilai yang didapat pada proses *masking*, dan akan diambil tiga nilai rata-rata untuk setiap warna (*hue*, *saturation*, *value*) serta dilakukan penggabungan menjadi satu vektor, yang datanya akan digunakan untuk model *K-Nearest Neighbors* (KNN). Setelah pemrosesan gambar dari tiga folder itu selanjutnya akan dilakukan penyimpanan fitur dan label dalam *list* yang kemudian akan dikonversi menjadi numpy *array* untuk melakukan analisis lebih lanjut.

Kemudian akan dilakukan pembagian dataset menjadi set pelatihan dan pengujian, melatih model KNN dengan set pelatihan, membuat prediksi menggunakan set pengujian dan mengevaluasi akurasi model, serta akan menggunakan nilai tiga sebagai parameter tetangga terdekat untuk menentukan kelasnya. Langkah selanjutnya adalah menyimpan model ke dalam *file* berekstensi *joblib*, kemudian membuat prediksi pada data pengujian dan terakhir melakukan evaluasi model dengan cara menghitung akurasi model yang akan dicetak dalam persentase.

Berikut adalah nilai parameter rentang warna bawah dan atas dari buah tomat untuk masing-masing klasifikasinya:

3.3.1. Data Parameter HSV Buah Tomat Matang

No	Nama Data	Range Data	
		Lower	Upper
1	<i>Hue</i>	0	10
2	<i>Saturation</i>	100	255
3	<i>Value</i>	100	255

3.3.2. Data Parameter HSV Buah Tomat Setengah Matang

No	Nama Data	Range Data	
		Lower	Upper
1	<i>Hue</i>	10	14
2	<i>Saturation</i>	100	255
3	<i>Value</i>	100	255

3.3.3. Data Parameter HSV Buah Tomat Mentah

No	Nama Data	Range Data	
		Lower	Upper
1	<i>Hue</i>	14	47
2	<i>Saturation</i>	100	255
3	<i>Value</i>	100	255

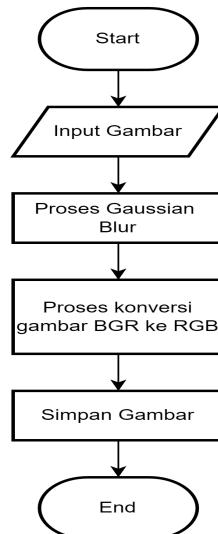
3.4 Operasi pre-processing

3.4.1 Konversi Gambar ke RGB

a. Tujuan

Mengubah format gambar ke RGB untuk menyesuaikan format warna dan memudahkan analisis selanjutnya.

b. Alur Proses



Gambar 3.3 Flowchart Konversi Gambar ke RGB

3.4.2 Konversi Gambar ke HSV

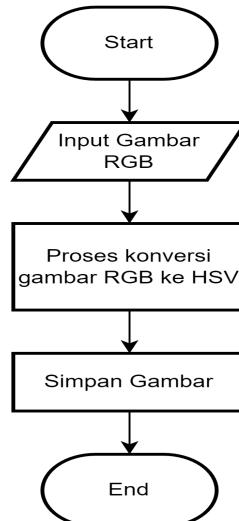
a. Tujuan

Mengubah gambar dari RGB ke ruang warna HSV untuk memisahkan warna, kecerahan, dan kejemuhan, sehingga segmentasi warna menjadi lebih mudah dan akurat.

b. Rumus

$$h(\text{hue}) = \begin{cases} 0, & \text{jika } \max - \min \\ 60^\circ \times \left(\frac{G - B}{\max - \min} \bmod 6 \right), & \text{jika } \max = R \\ 60^\circ \times \left(\frac{B - R}{\max - \min} + 2 \right), & \text{jika } \max = G \\ 60^\circ \times \left(\frac{R - G}{\max - \min} + 4 \right), & \text{jika } \max = B \end{cases}$$
$$s(\text{saturation}) = \begin{cases} 0, & \text{jika } \max - \min \\ \frac{\max - \min}{V}, & \text{otherwise} \end{cases}$$
$$V(\text{value}) = \max$$

c. Alur Proses



Gambar 3.4 Flowchart Konversi Gambar ke HSV

3.4.3 Masking Warna

a. Tujuan

Membuat mask untuk area tomat yang matang, mentah, dan setengah matang, sehingga area-area yang relevan dapat dipisahkan dan dianalisis secara spesifik.

b. Alur Proses



Gambar 3.5 Flowchart Masking Warna

3.4.4 Menghitung Rata-rata Warna

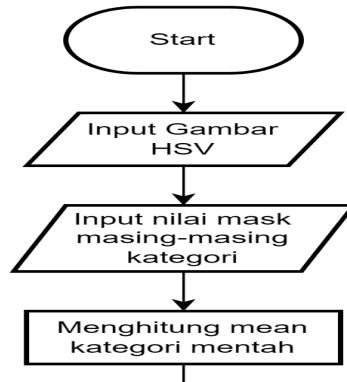
a. Tujuan

Menghitung rata-rata warna untuk setiap mask yang dihasilkan agar dapat menyiapkan data dalam bentuk fitur numerik untuk proses klasifikasi

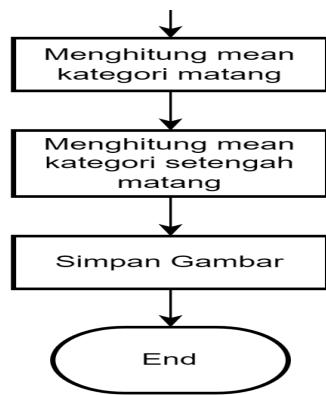
b. Rumus

$$\text{Rata-rata Warna} = \left(\frac{1}{n} \sum_{x,y} C(x,y) \right)$$

c. Alur Proses



Gambar 3.6 Flowchart Menghitung Mean Warna (1)



Gambar 3.7 Flowchart Menghitung Mean Warna (2)

3.4.5 Operasi Morfologi

a. Tujuan

Menerapkan operasi morfologi seperti opening untuk membersihkan noise pada hasil segmentasi mask, meningkatkan keakuratan deteksi.

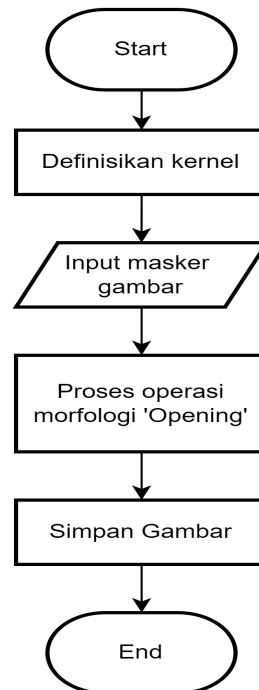
b. Rumus

$$A \circ B = (A \ominus B) \cup B$$

Padanan fungsi:

$$A \circ B = \{(B)z | (B)z \in A\}$$

c. Alur Proses



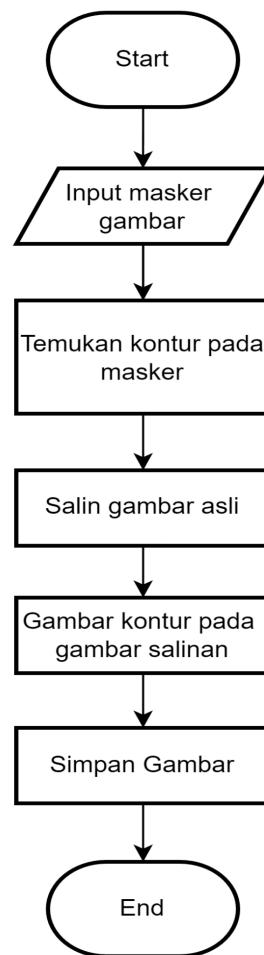
Gambar 3.8 Flowchart Morfologi

3.4.6 Identifikasi Kontur

a. Tujuan

Menentukan kontur pada masing-masing mask untuk mengetahui batas-batas area yang tersegmentasi, membantu dalam visualisasi dan verifikasi hasil analisis.

b. Alur Proses



Gambar 3.9 Flowchart Identifikasi Kontur

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Coding

4.1.1 Import Library

```
1  from joblib import load
2  import cv2
3  import sys
4  import numpy as np
5  from PyQt5.uic import loadUi
6  from PyQt5 import QtCore, QtWidgets
7  from PyQt5.QtGui import QImage, QPixmap
8  from PyQt5.QtWidgets import QMainWindow, QFileDialog
9  import matplotlib.pyplot as plt
```

Gambar 4.1 Koding Import Library

4.1.2 Mengatur Atribut Aplikasi

```
11  QtWidgets.QApplication.setAttribute(QtCore.Qt.AA_EnableHighDpiScaling, True) # enable highdpi scaling
12  QtWidgets.QApplication.setAttribute(QtCore.Qt.AA_UseHighDpiPixmaps, True) # use highdpi icons
```

Gambar 4.2 Koding Mengatur Atribut Aplikasi

Digunakan untuk mengatur atribut aplikasi dalam PyQt5 terkait penanganan DPI (*dots per inch*) tinggi. Pengaturan ‘AA_EnableHighDpiScalling’ mengaktifkan penskalaan DPI tinggi. Pengaturan ‘AA_UseHighDpiPixmaps’ mengatur penggunaan ikon dengan resolusi tinggi yang sesuai dengan *layer* dengan DPI tinggi.

4.1.3 Inisialisasi

```
14  class MainWindow(QtWidgets.QMainWindow):
15      def __init__(self):
16          super(MainWindow, self).__init__()
17          self.Image = None
18          self.model_filename = 'knn_model.joblib'
19          self.knn_model = load(self.model_filename)
20          self.prediction_value = None # Add a class attribute for prediction
21          self.initUI()
```

Gambar 4.3 Koding Inisialisasi

Koding bagian ini adalah bagian dari metode inisialisasi dalam kelas ‘MainWindow’ yang menggunakan PyQt5. Fungsi dari kode ini adalah untuk menginisialisasi objek dengan mengatur atribut-atribut penting dan mempersiapkan antarmuka pengguna.

4.1.4 Membuat Antarmuka Pengguna Pertama

```
23     def initUI(self):
24         loadUi('first.ui', self)
25         self.pushButton.clicked.connect(self.load_second_ui)
```

Gambar 4.4 Koding Membuat Antarmuka Pengguna Pertama

Koding bagian ini berfungsi untuk menginisialisasi antarmuka pengguna (UI) awal aplikasi. Kemudian menghubungkan fungsi yang akan dieksekusi ketika tombol ‘pushButton’ pada antarmuka tersebut di klik.

4.1.5 Membuat Antarmuka Pengguna Kedua

```
27     def load_second_ui(self):
28         loadUi('second2.ui', self)
29         self.btn_loadimage.clicked.connect(self.load_image)
30         self.actionKonversi.triggered.connect(self.konversi)
31
32         self.actionMask_Ripe.triggered.connect(self.maskRipe)
33         self.actionMask_Unripe.triggered.connect(self.maskUnripe)
34         self.actionMask_Half_Ripe.triggered.connect(self.maskHalfripe)
35         self.actionAll_Mask.triggered.connect(self.maskAll)
36
37         self.actionMorfologi_Mask_Ripe.triggered.connect(self.morfoRipe)
38         self.actionMorfologi_Mask_Unripe.triggered.connect(self.morfoUnripe)
39         self.actionMorfologi_Mask_Half_Ripe.triggered.connect(self.morfoHalfripe)
40         self.actionAll_Morfologi_Mask.triggered.connect(self.morfoAll)
41
42         self.actionContour_Half_Ripe.triggered.connect(self.contourAll)
```

Gambar 4.5 Koding Membuat Antarmuka Pengguna Kedua

Kodingan bagian ini untuk membuat antarmuka pengguna kedua dari file ‘second2.ui’ setelah tombol pada antarmuka pengguna pertama ditekan. Pada kodingan ini menghubungkan berbagai tombol atau opsi antarmuka pengguna kedua dengan fungsi-fungsi yang akan dieksekusi saat tombol atau opsi tersebut diaktifkan.

4.1.6 Memuat Gambar dan Mengkonversikan Gambar

Koding bagian ini mengizinkan pengguna untuk memilih gambar dari sistem file, kemudian memuat gambar tersebut ke dalam aplikasi, mengkonversinya ke format yang sesuai untuk ditampilkan di antarmuka pengguna, dan mengaktifkan fungsi-fungsi terkait saat tombol pada antarmuka pengguna ditekan.

```

51     def load_image(self):
52         file_dialog = QFileDialog()
53         file_dialog.setFileMode(QFileDialog.AnyFile)
54         file_dialog.setNameFilter("Images (*.png *.xpm *.jpg *.jpeg *.bmp)")
55         if file_dialog.exec_():
56             file_names = file_dialog.selectedFiles()
57             image_path = file_names[0]
58             self.Image = cv2.imread(image_path)
59             self.Image = cv2.cvtColor(self.Image, cv2.COLOR_BGR2RGB)
60             if self.Image is None:
61                 print("Error: Unable to load image.")
62             else:
63                 self.Image = cv2.GaussianBlur(self.Image, (5, 5), 0)
64                 # self.Image = self.apply_kmeans_segmentation(self.Image)
65                 self.displayImage()
66                 self.btn_proses.clicked.connect(lambda: self.predict_image_category(image_path))
67                 self.btn_lihatproses.clicked.connect(self.lihat_proses)
68             except Exception as e:
69                 print("Error:", e)
70

```

Gambar 4.6 Koding Memuat Gambar dan Mengkonversikan Gambar

4.1.7 Menampilkan Gambar di Antarmuka Pengguna

```

63     def displayImage(self):
64         qformat = QImage.Format_RGB888
65         img = QImage(self.Image, self.Image.shape[1], self.Image.shape[0], self.Image.strides[0], qformat)
66         self.label.setPixmap(QPixmap.fromImage(img))
67         self.label.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
68         self.label.setScaledContents(True)

```

Gambar 4.7 Koding Menampilkan Gambar di Antarmuka Pengguna

Koding di atas merupakan sebuah metode yang digunakan untuk menampilkan gambar di antarmuka pengguna. Ini mengambil citra yang sudah dimuat dan mengkonversinya ke format yang dapat ditampilkan oleh label di antarmuka.

4.1.8 Melakukan Pra-pemrosesan Gambar

```

78     def preprocess_image(self, image_path):
79         image_asli = cv2.imread(image_path)
80         cv2.imwrite("images/0_Original.png", image_asli)
81         image = cv2.cvtColor(image_asli, cv2.COLOR_BGR2RGB) # Convert to RGB
82         cv2.imwrite("images/0_RGB.png", image)
83         hsv_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV) # Convert to HSV
84         cv2.imwrite("images/0_HSV.png", hsv_image)
85
86         # Unripe Mask
87         lower_unripe = np.array([14, 100, 100], dtype=np.uint8)
88         upper_unripe = np.array([47, 255, 255], dtype=np.uint8)
89         mask_unripe = cv2.inRange(hsv_image, lower_unripe, upper_unripe)
90         cv2.imwrite("images/1_unripe.png", mask_unripe)

```

Gambar 4.8 Kodingan Melakukan Pra-Pemrosesan Gambar (1)

```

92     # Ripe Mask
93     lower_half_ripe = np.array([0, 100, 100], dtype=np.uint8)
94     upper_half_ripe = np.array([10, 255, 255], dtype=np.uint8)
95     mask_ripe = cv2.inRange(hsv_image, lower_half_ripe, upper_half_ripe)
96     cv2.imwrite("images/1_ripe.png", mask_ripe)
97
98     # Half Ripe Mask
99     lower_ripe = np.array([10, 100, 100], dtype=np.uint8)
100    upper_ripe = np.array([14, 255, 255], dtype=np.uint8)
101    mask_half_ripe = cv2.inRange(hsv_image, lower_ripe, upper_ripe)
102    cv2.imwrite("images/1_halfripe.png", mask_half_ripe)
103
104    mean_ripe = cv2.mean(hsv_image, mask=mask_ripe)[:3]
105    mean_unripe = cv2.mean(hsv_image, mask=mask_unripe)[:3]
106    mean_half_ripe = cv2.mean(hsv_image, mask=mask_half_ripe)[:3]
107
108    features = np.concatenate([mean_ripe, mean_unripe, mean_half_ripe])

```

Gambar 4.9 Koding Melakukan Pra-pemrosesan Gambar (2)

```

110    # Morphological operations to clean up masks
111    kernel = np.ones((5, 5), np.uint8)
112    mask_ripe = cv2.morphologyEx(mask_ripe, cv2.MORPH_OPEN, kernel)
113    cv2.imwrite("images/2_ripe.png", mask_ripe)
114    mask_unripe = cv2.morphologyEx(mask_unripe, cv2.MORPH_OPEN, kernel)
115    cv2.imwrite("images/2_unripe.png", mask_unripe)
116    mask_half_ripe = cv2.morphologyEx(mask_half_ripe, cv2.MORPH_OPEN, kernel)
117    cv2.imwrite("images/2_halfripe.png", mask_half_ripe)
118
119    # Find contours
120    contours_ripe, _ = cv2.findContours(mask_ripe, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
121    contours_unripe, _ = cv2.findContours(mask_unripe, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
122    contours_half_ripe, _ = cv2.findContours(mask_half_ripe, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
123
124    # Create copies of the original image
125    ripe_image = image.copy()
126    unripe_image = image.copy()
127    half_ripe_image = image.copy()
128
129    # Draw contours on the copies
130    cv2.drawContours(ripe_image, contours_ripe, -1, (0, 0, 255), thickness=2)
131    cv2.drawContours(unripe_image, contours_unripe, -1, (0, 0, 255), thickness=2)
132    cv2.drawContours(half_ripe_image, contours_half_ripe, -1, (0, 0, 255), thickness=2)
133
134    # Save the images
135    cv2.imwrite("images/3_ripe.png", ripe_image)
136    cv2.imwrite("images/3_unripe.png", unripe_image)
137    cv2.imwrite("images/3_halfripe.png", half_ripe_image)
138
139    # Return image and contours
140    return features

```

Gambar 4.10 Koding Melakukan Pra-pemrosesan Gambar (3)

Koding bagian ini merupakan metode untuk melakukan pra-pemrosesan gambar, ‘preprocess_image’ melakukan pra pemrosesan gambar untuk ekstraksi fitur berdasarkan warna dalam ruang warna HSV, melakukan operasi morfologi untuk membersihkan masker, dan menemukan serta menggambar kontur pada gambar asli. Fungsi ini kemudian mengembalikan fitur-fitur yang diekstraksi dari gambar. Kode ini bertujuan untuk melakukan pra pemrosesan gambar buah sehingga dapat

diekstraksi fiturnya berdasarkan warna HSV, dibersihkan dari noise, ditemukan konturnya, dan akhirnya menghasilkan vektor fitur yang dapat digunakan untuk klasifikasi lebih lanjut.

4.1.9 Melakukan Prediksi Kategori

```
118     def predict_image_category(self, image_path):
119         features = self.preprocess_image(image_path)
120         feature = features.reshape(1, -1)
121         prediction = self.knn_model.predict(feature)
122         print(prediction)
123
124         if prediction == 0:
125             category = 'Unripe'
126         elif prediction == 1:
127             category = 'Ripe'
128         elif prediction == 2:
129             category = 'Half-Ripe'
130         else:
131             category = 'Unknown'
132
133         self.label_prediction.setText(f'Prediction: {category}')
134         self.prediction_value = prediction # Store the prediction in an attribute
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
```

Gambar 4.11 Koding Melakukan Prediksi Kategori

Kodingan di atas merupakan metode untuk melakukan prediksi kategori kematangan tomat berdasarkan fitur yang diekstraksi dari citra.

4.1.10 Melakukan Konversi

```
137     def konversi(self):
138         try:
139             # Load and display saved images
140             img_RGB = cv2.imread('images/0_RGB.png')
141             img_HSV = cv2.imread('images/0_HSV.png')
142
143             if img_RGB is None or img_HSV is None:
144                 raise FileNotFoundError("One or both images not found")
145
146             plt.subplot(121), plt.imshow(img_RGB, cmap='gray', interpolation='bicubic'), plt.title('Image RGB')
147             plt.xticks([]), plt.yticks([])
148             plt.subplot(122), plt.imshow(img_HSV, cmap='gray', interpolation='bicubic'), plt.title('Image HSV')
149             plt.xticks([]), plt.yticks([])
150             plt.show()
151
152         except Exception as e:
153             print("An error occurred:", e)
```

Gambar 4.12 Koding Melakukan Konversi

Kodingan ini adalah metode dalam kelas yang bertujuan untuk memuat dan menampilkan gambar dalam format RGB dan HSV yang telah disimpan sebelumnya.

4.1.11 Melakukan Masking

```
def maskRipe(self):
    try:
        # Load and display saved images
        mask_ripe = cv2.imread('images/1_ripe.png')

        if mask_ripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(121), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('Image Asli')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(mask_ripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)
```

Gambar 4.13 Koding Melakukan Masking *Ripe*

```
def maskUnripe(self):
    try:
        # Load and display saved images
        mask_unripe = cv2.imread('images/1_unripe.png')

        if mask_unripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(121), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('Image Asli')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(mask_unripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Unripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)
```

Gambar 4.14 Koding Melakukan Masking *Un-Ripe*

```

def maskHalfripe(self):
    try:
        # Load and display saved images
        mask_halfripe = cv2.imread('images/1_halfripe.png')

        if mask_halfripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(121), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('Image Asli')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(mask_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)

```

Gambar 4.15 Koding Melakukan Masking *Half Ripe*

```

def maskAll(self):
    try:
        # Load and display saved images
        img_asli = cv2.imread('images/0_Original.png')
        mask_ripe = cv2.imread('images/1_ripe.png')
        mask_unripe = cv2.imread('images/1_unripe.png')
        mask_halfripe = cv2.imread('images/1_halfripe.png')

        if img_asli is None or mask_halfripe is None or mask_unripe is None or mask_ripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(221), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('ORIGINAL')
        plt.xticks([]), plt.yticks([])
        plt.subplot(222), plt.imshow(mask_ripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Ripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(223), plt.imshow(mask_unripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Unripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(224), plt.imshow(mask_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)

```

Gambar 4.16 Koding Melakukan Masking *All*

Kodingan ini adalah fungsi metode mask dari sebuah kelas yang bertujuan untuk menampilkan gambar asli dengan mask dari gambar tersebut yang sudah di proses. Pada proses masking ini gambar melewati tiga tahap masking, masking pertama untuk masking ripe atau matang, kemudian masking unripe atau mentah, dan masking half ripe atau setengah matang. Kemudian hasilnya ditampilkan ditampilkan dalam sebuah plot.

4.1.12 Melakukan Morfologi

Kodingan ini adalah metode dari sebuah kelas yang bertujuan untuk menampilkan hasil morfologi dari gambar asli yang telah diproses untuk area ripe (matang), unripe (mentah), dan half ripe (setengah matang). Pada proses morfologi ini

gambar melewati tiga tahap morfologi, morfologi pertama untuk morfologi ripe, unripe, dan half ripe. Kemudian hasilnya ditampilkan dalam sebuah plot.

```
def morfoRipe(self):
    try:
        # Load and display saved images
        mask_ripe = cv2.imread('images/1_ripe.png')
        morfo_ripe = cv2.imread('images/2_ripe.png')

        if morfo_ripe is None or mask_ripe is None:
            raise FileNotFoundError("One or both images not found")
```

Gambar 4.17 Koding Morfologi *Ripe*

```
def morfoUnripe(self):
    try:
        # Load and display saved images
        mask_unripe = cv2.imread('images/1_unripe.png')
        morfo_unripe = cv2.imread('images/2_unripe.png')

        if mask_unripe is None or morfo_unripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(121), plt.imshow(mask_unripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Unripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(morfo_unripe, cmap='gray', interpolation='bicubic'), plt.title('Morfologi Mask Unripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)
```

Gambar 4.18 Koding Morfologi *Un-Ripe*

```
def morfoHalfripe(self):
    try:
        # Load and display saved images
        mask_halfripe = cv2.imread('images/1_halfripe.png')
        morfo_halfripe = cv2.imread('images/2_halfripe.png')

        if mask_halfripe is None or morfo_halfripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(121), plt.imshow(mask_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(122), plt.imshow(morfo_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Morfologi Mask Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)
```

Gambar 4.19 Koding Morfologi *Half Ripe*

```

def morfoAll(self):
    try:
        # Load and display saved images
        img_asli = cv2.imread('images/0_Original.png')
        mask_ripe = cv2.imread('images/2_ripe.png')
        mask_unripe = cv2.imread('images/2_unripe.png')
        mask_halfripe = cv2.imread('images/2_halfripe.png')

        if img_asli is None or mask_halfripe is None or mask_unripe is None or mask_ripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(221), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('ORIGINAL')
        plt.xticks([]), plt.yticks([])
        plt.subplot(222), plt.imshow(mask_ripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Ripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(223), plt.imshow(mask_unripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Unripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(224), plt.imshow(mask_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Mask Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)

```

Gambar 4.20 Koding Morfologi *All*

4.1.13 Melakukan Kontur

Kodingan ini adalah metode tambahan dengan tujuan untuk menampilkan gambar asli bersama dengan gambar yang menunjukkan kontur atau garis tepi dari area ripe atau matang, unripe atau mentah, dan half ripe atau setengah matang yang dihasilkan dari proses pengolahan citra.

```

def contourAll(self):
    try:
        # Load and display saved images
        c_ripe = cv2.imread('images/3_ripe.png')
        c_unripe = cv2.imread('images/3_unripe.png')
        c_halfripe = cv2.imread('images/3_halfripe.png')

        if c_halfripe is None or c_unripe is None or c_ripe is None:
            raise FileNotFoundError("One or both images not found")

        plt.subplot(221), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('ORIGINAL')
        plt.xticks([]), plt.yticks([])
        plt.subplot(222), plt.imshow(c_ripe, cmap='gray', interpolation='bicubic'), plt.title('Contour Ripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(223), plt.imshow(c_unripe, cmap='gray', interpolation='bicubic'), plt.title('Contour Unripe')
        plt.xticks([]), plt.yticks([])
        plt.subplot(224), plt.imshow(c_halfripe, cmap='gray', interpolation='bicubic'), plt.title('Contour Half Ripe')
        plt.xticks([]), plt.yticks([])
        plt.show()

    except Exception as e:
        print("An error occurred:", e)

```

Gambar 4.21 Koding Kontur *All*

4.1.14 Menampilkan Proses Pengolahan Citra

Kodingan ini adalah metode yang digunakan untuk menampilkan proses pengolahan citra dari gambar asli ke berbagai tahap pengolahan citra dari gambar asli ke berbagai tahap pengolahan berdasarkan kategori prediksi oleh model K-Nearest Neighbors (KNN). Fungsi ini menampilkan gambar asli, hasil masking, morfologi, dan kontur dari gambar yang dianalisis.

```
def lihat_proses(self):
    prediction = self.prediction_value # Use the stored prediction value
    if prediction == 1:
        mask = cv2.imread('images/1_ripe.png')
        morfo = cv2.imread('images/2_ripe.png')
        contour = cv2.imread('images/3_ripe.png')
    if prediction == 0:
        mask= cv2.imread('images/1_unripe.png')
        morfo = cv2.imread('images/2_unripe.png')
        contour = cv2.imread('images/3_unripe.png')
    if prediction == 2:
        mask = cv2.imread('images/1_halfripe.png')
        morfo = cv2.imread('images/2_halfripe.png')
        contour = cv2.imread('images/3_halfripe.png')

    plt.subplot(221), plt.imshow(self.Image, cmap='gray', interpolation='bicubic'), plt.title('ORIGINAL')
    plt.xticks([]), plt.yticks([])
    plt.subplot(222), plt.imshow(mask, cmap='gray', interpolation='bicubic'), plt.title('Masking')
    plt.xticks([]), plt.yticks([])
    plt.subplot(223), plt.imshow(morfo, cmap='gray', interpolation='bicubic'), plt.title('Morphology')
    plt.xticks([]), plt.yticks([])
    plt.subplot(224), plt.imshow(contour, cmap='gray', interpolation='bicubic'), plt.title('Contour')
    plt.xticks([]), plt.yticks([])
    plt.show()
```

Gambar 4.22 Koding Proses Pengolahan Citra

4.1.15 Menampilkan Inisialisasi Aplikasi Qt

```
app = QtWidgets.QApplication(sys.argv)
window = MainWindow()
window.setWindowTitle('KELOMPOK C2')
window.show()
sys.exit(app.exec_())
```

Gambar 4.23 Koding Inisialisasi Aplikasi Qt

Kode ini bertujuan untuk menginisialisasi aplikasi berbasis Qt, menampilkan jendela utama yang berjudul “KELOMPOK C2”, dan menjalankan loop utama aplikasi untuk menangani peristiwa dan interaksi pengguna.

4.1.16 Membuat Dataset

```
1 import os
2 import cv2
3 import numpy as np
4 from sklearn.model_selection import train_test_split
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.metrics import accuracy_score
7 from joblib import dump
8
9 # Paths to your folders
10 ripe_folder = 'Matang'
11 unripe_folder = 'Mentah'
12 half_ripe_folder = 'Setengah Matang'
13
14 # Function to preprocess images and extract features using OpenCV
15 def preprocess_image(image_path):
16     image = cv2.imread(image_path)
17     image = cv2.resize(image, (64, 64)) # Resize to a fixed size
18
19     # Convert to HSV color space
20     hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

Gambar 4.24 Koding Membuat Dataset (1)

```
37 # Unripe Mask
38 lower_unripe = np.array([14, 100, 100], dtype=np.uint8)
39 upper_unripe = np.array([47, 255, 255], dtype=np.uint8)
40 mask_unripe = cv2.inRange(hsv_image, lower_unripe, upper_unripe)
41 cv2.imwrite("images/1_unripe.png", mask_unripe)
42
43 # Ripe Mask
44 lower_half_ripe = np.array([0, 100, 100], dtype=np.uint8)
45 upper_half_ripe = np.array([10, 255, 255], dtype=np.uint8)
46 mask_ripe = cv2.inRange(hsv_image, lower_half_ripe, upper_half_ripe)
47 cv2.imwrite("images/1_ripe.png", mask_ripe)
48
49 # Half Ripe Mask
50 lower_ripe = np.array([10, 100, 100], dtype=np.uint8)
51 upper_ripe = np.array([14, 255, 255], dtype=np.uint8)
52 mask_half_ripe = cv2.inRange(hsv_image, lower_ripe, upper_ripe)
53 cv2.imwrite("images/1_halfripe.png", mask_half_ripe)
```

Gambar 4.25 Koding Membuat Dataset (2)

```

55     # Calculate the mean color within the masked areas
56     mean_ripe = cv2.mean(hsv_image, mask=mask_ripe)[:3]
57     mean_unripe = cv2.mean(hsv_image, mask=mask_unripe)[:3]
58     mean_half_ripe = cv2.mean(hsv_image, mask=mask_half_ripe)[:3]
59
60     # Combine the mean colors into a feature vector
61     features = np.concatenate([mean_ripe, mean_unripe, mean_half_ripe])
62
63     return features
64
65
66     # Lists to hold features and labels
67     features = []
68     labels = []

```

Gambar 4.26 Koding Membuat Dataset (3)

```

74     # Process ripe images
75     for filename in os.listdir(ripe_folder):
76         if filename.endswith('.jpg'):
77             features.append(preprocess_image(os.path.join(ripe_folder, filename)))
78             labels.append(1) # Label for ripe
79
80     # Process unripe images
81     for filename in os.listdir(unripe_folder):
82         if filename.endswith('.jpg'):
83             features.append(preprocess_image(os.path.join(unripe_folder, filename)))
84             labels.append(0) # Label for unripe
85
86     # Process half-ripe images
87     for filename in os.listdir(half_ripe_folder):
88         if filename.endswith('.jpg'):
89             features.append(preprocess_image(os.path.join(half_ripe_folder, filename)))
90             labels.append(2) # Label for half-ripe
91
92     # Convert to numpy arrays
93     features = np.array(features)
94     labels = np.array(labels)
95
96     # Split the dataset into training and testing sets
97     X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
98
99     # Create and train the KNN model
100    knn = KNeighborsClassifier(n_neighbors=3) # You can choose the number of neighbors
101    knn.fit(X_train, y_train)
102
103    # Save the model to a file
104    model_filename = 'knn_model.joblib'
105    dump(knn, model_filename)
106
107    # Make predictions on the test set
108    y_pred = knn.predict(X_test)
109
110    # Evaluate the model
111    accuracy = accuracy_score(y_test, y_pred)
112    print(f'Accuracy: {accuracy * 100:.2f}%')

```

Gambar 4.27 Koding Membuat Dataset (4)

Kode ini bertujuan untuk mengembangkan model klasifikasi yang dapat mengidentifikasi tingkat kematangan buah berdasarkan gambar yang diinputkan, menggunakan algoritma K-Nearest Neighbors dan fitur warna dari gambar dalam

ruang warna HSV. Model yang dihasilkan kemudian disimpan dan siap digunakan untuk prediksi.

4.2 Design GUI

Gambar desain antarmuka pengguna (UI) dibuat menggunakan aplikasi PyCharm dengan bantuan library PyQt untuk pengembangan aplikasi GUI. Desain UI ini merupakan bagian integral dari pengembangan perangkat lunak yang memungkinkan pengguna untuk berinteraksi dengan aplikasi dengan cara yang intuitif dan efisien. Terdapat UI untuk tampilan awal, kemudian about untuk menjelaskan aplikasi yang dibuat, dan tampilan kedua untuk menampilkan citra.

4.2.1 Tampilan Pertama



Gambar 4.28 UI Tampilan Pertama

Pada tampilan pertama ini adalah tampilan awal pada saat menjalankan program, di sini pengguna dapat melakukan start dengan menekan tombol play dan pengguna juga dapat melihat pengertian aplikasi yang dibuat dengan menekan about.

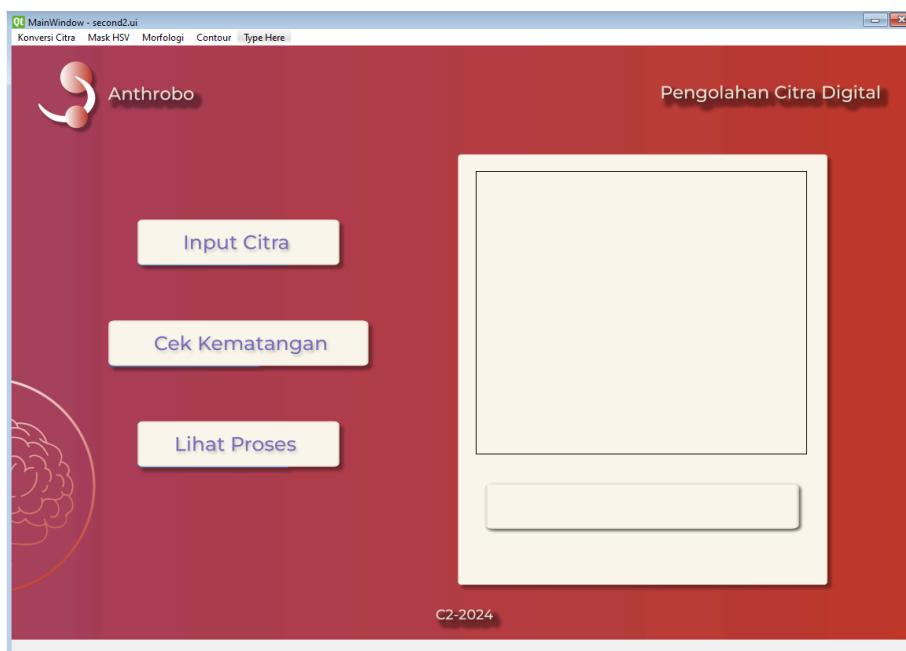
4.2.2 Tampilan About

Pada tampilan about, pengguna dapat melihat informasi mengenai aplikasi yang dibuat, meliputi nama aplikasi dan cara penggunaan aplikasi.



Gambar 4.29 UI Tampilan About

4.2.3 Tampilan Kedua

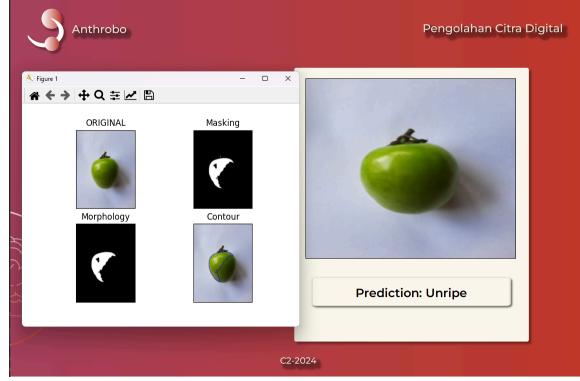
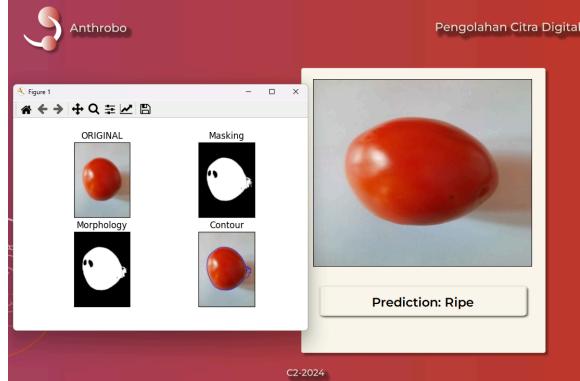


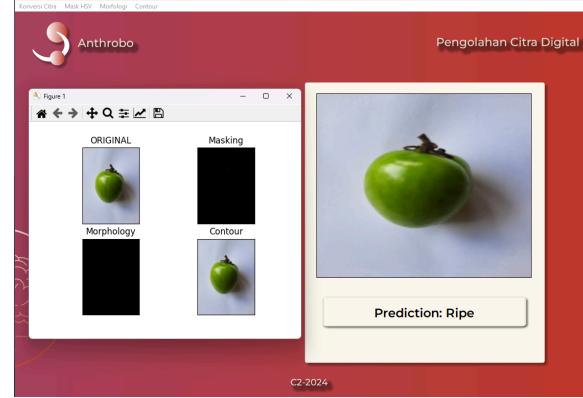
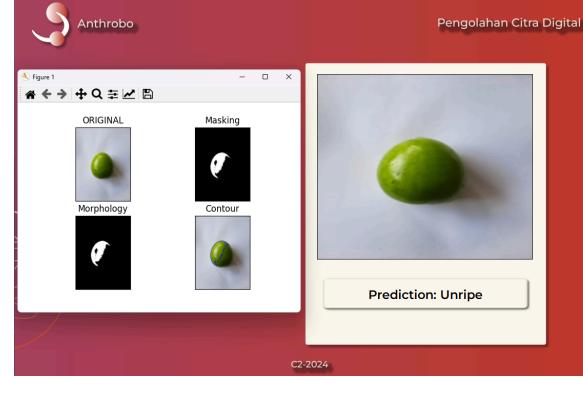
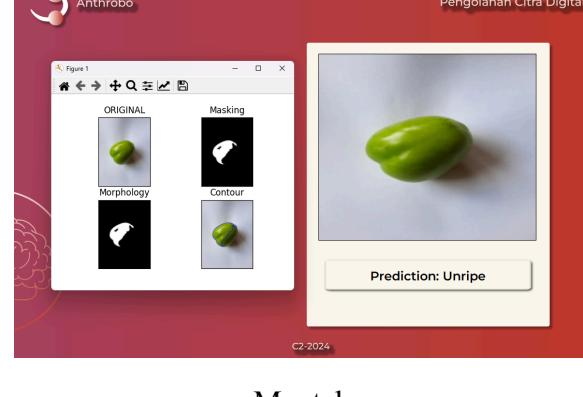
Gambar 4.30 UI Tampilan Kedua

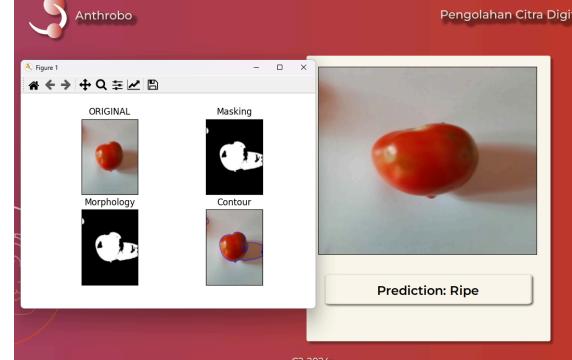
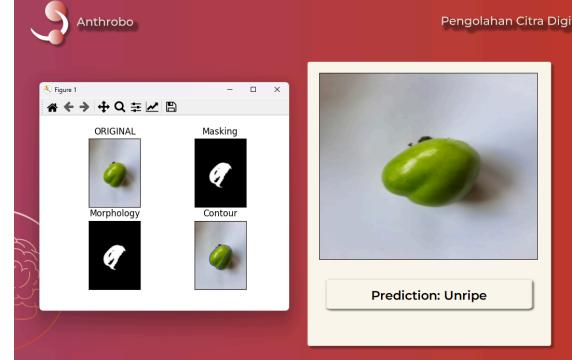
Pada tampilan kedua ini adalah tampilan ketika pengguna sudah menekan tombol play pada tampilan ui pertama. Dalam tampilan kedua ini pengguna dapat

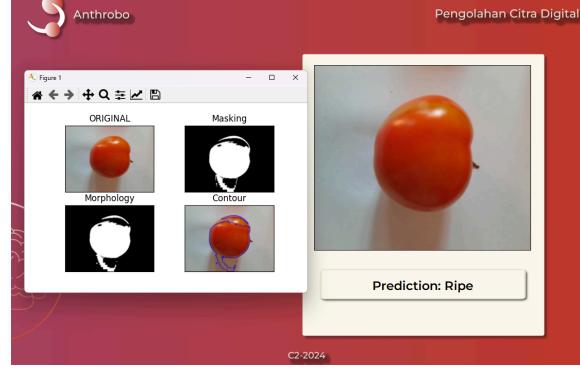
memasukkan citra dengan menekan tombol ‘Input Citra’ lalu citra dapat di proses di tombol ‘Cek Kematangan’ maka kematangan buah tomat dapat di lihat di bawah gambar yang sudah dimasukkan. Pada tombol ‘Lihat Proses’ pengguna dapat melihat hasil masking, morfologi, dan juga kontur yang disesuaikan dengan hasil kematangan.

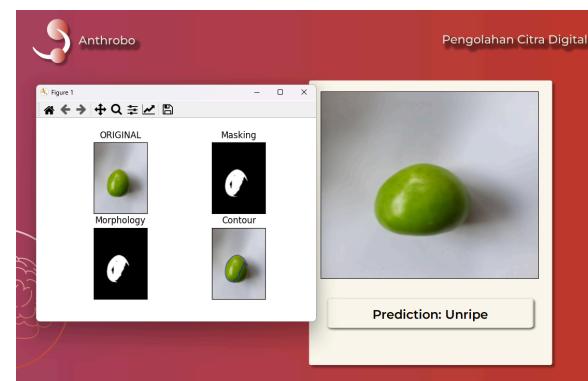
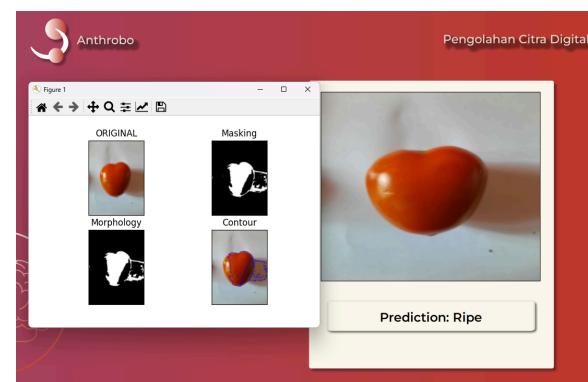
4.3 Gambar Sebelum dan Sesudah diproses

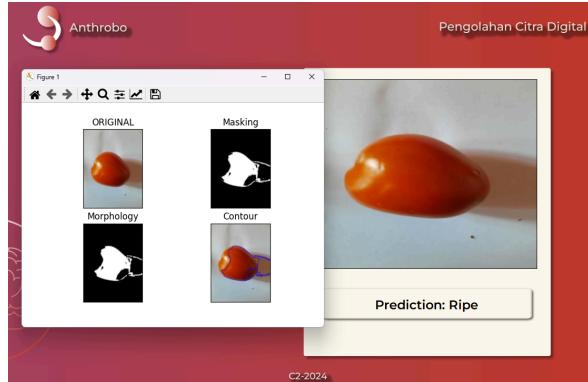
No	Gambar Asli	Tingkat Kematangan dari Hasil Aplikasi	Tingkat Kematangan Asli
1			Mentah
2			Matang

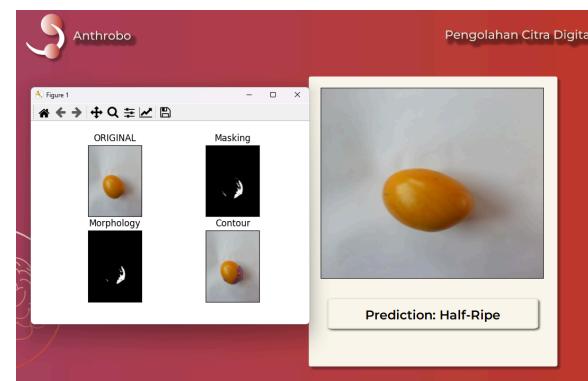
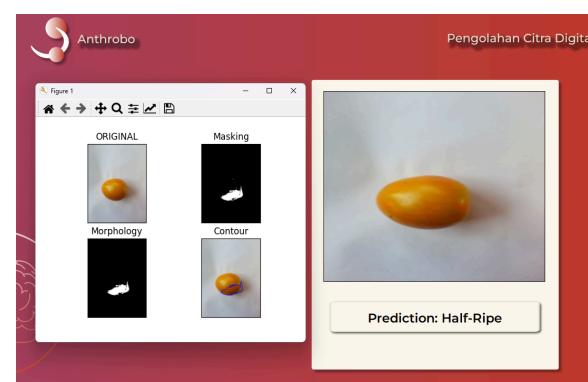
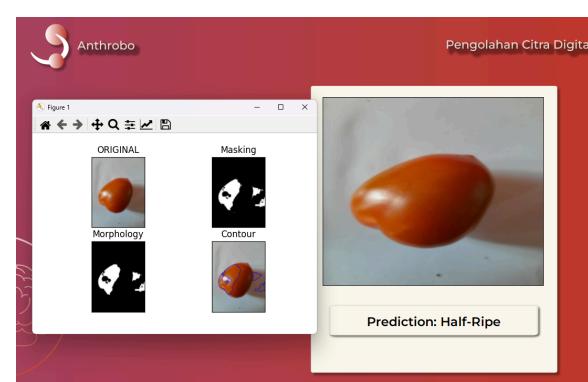
3		 Matang	Mentah
4		 Mentah	Mentah
5		 Mentah	Mentah

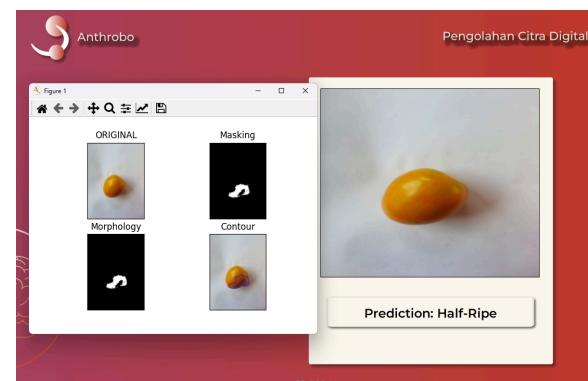
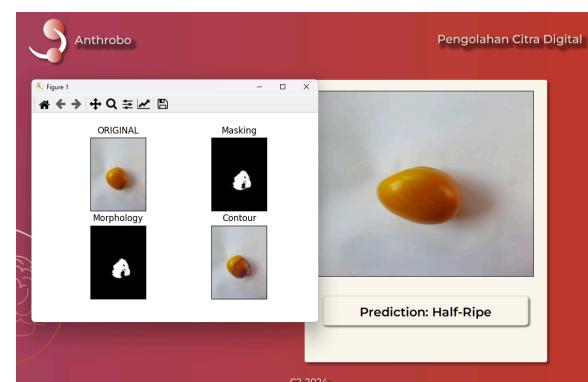
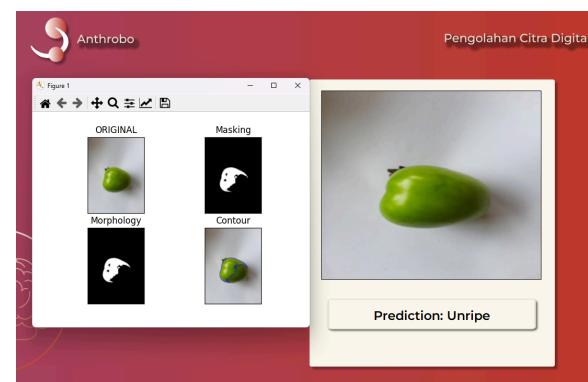
6		 Matang	
7		 Setengah Matang	
8		 Mentah	

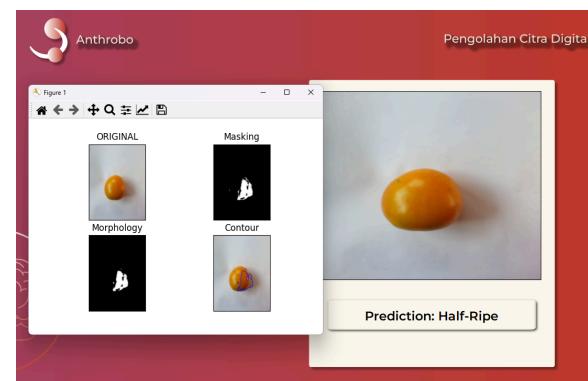
9		 <p>Pengolahan Citra Digital C2-2024</p>	<p>Matang</p>
10		 <p>Pengolahan Citra Digital C2-2024</p>	<p>Setengah Matang</p>
11		 <p>Pengolahan Citra Digital C2-2024</p>	<p>Matang</p>

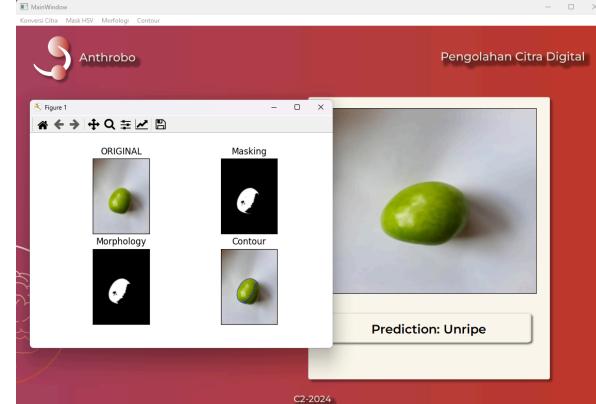
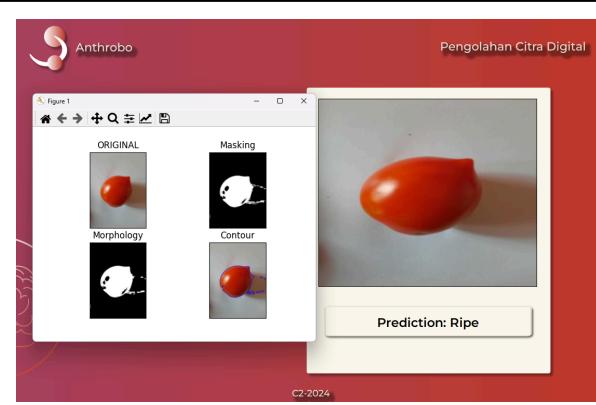
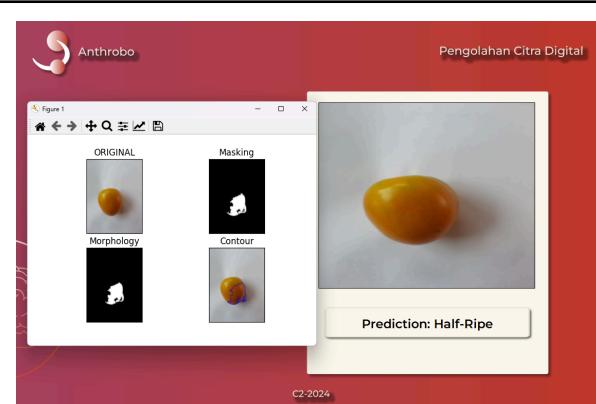
12		 Prediction: Unripe	Mentah
13		 Prediction: Ripe	Matang
14		 Prediction: Ripe	Matang

15		 Prediction: Half-Ripe	Setengah Matang
16		 Prediction: Ripe	Matang
17		 Prediction: Half-Ripe	Setengah Matang

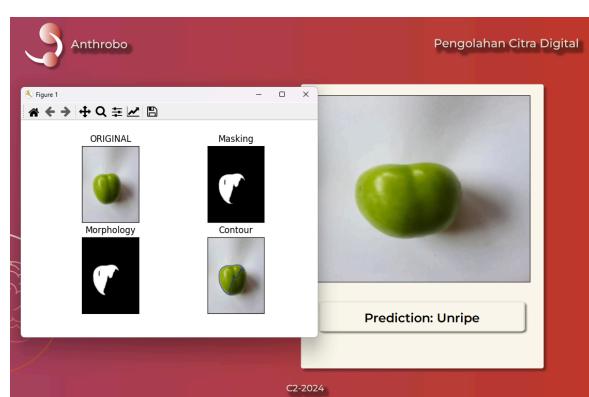
18		 Setengah Matang
19		 Setengah Matang
20		 Setengah Matang

21		 Setengah Matang	
22		 Setengah Matang	
23		 Mentah	

24			Setengah Matang
25			Matang
26			Mentah

27			Mentah
28			Matang
29			Setengah Matang

30



Mentah

Mentah

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian tingkat kematangan buah tomat dengan aplikasi yang telah dibangun, berikut merupakan keterangan dan akurasi yang diperoleh.

Gambar	Target	Hasil	Keterangan
1	Mentah	Mentah	Benar
2	Matang	Matang	Benar
3	Mentah	Matang	Salah
4	Mentah	Mentah	Benar
5	Mentah	Mentah	Benar
6	Matang	Matang	Benar
7	Setengah Matang	Setengah Matang	Benar
8	Mentah	Mentah	Benar
9	Matang	Matang	Benar
10	Setengah Matang	Setengah Matang	Benar
11	Matang	Matang	Benar
12	Mentah	Mentah	Benar
13	Matang	Matang	Benar
14	Matang	Matang	Benar
15	Setengah Matang	Setengah Matang	Benar
16	Matang	Matang	Benar

17	Setengah Matang	Setengah Matang	Benar
18	Setengah Matang	Setengah Matang	Benar
19	Setengah Matang	Setengah Matang	Benar
20	Matang	Setengah Matang	Salah
21	Setengah Matang	Setengah Matang	Benar
22	Setengah Matang	Setengah Matang	Benar
23	Mentah	Mentah	Benar
24	Setengah Matang	Setengah Matang	Benar
25	Matang	Matang	Benar
26	Mentah	Mentah	Benar
27	Mentah	Mentah	Benar
28	Matang	Matang	Benar
29	Setengah Matang	Setengah Matang	Benar
30	Mentah	Mentah	Benar

$$\text{Akurasi} = 28 : 30 \times 100\% = 93.33\%$$

Aplikasi identifikasi kematangan tomat yang menggunakan metode analisis citra dan algoritma K-Nearest Neighbors (KNN) telah berhasil mengkategorikan tomat menjadi tiga tingkat kematangan: Matang, Setengah Matang, dan Mentah. Dengan memanfaatkan teknik analisis citra, aplikasi ini mampu mengenali karakteristik visual tomat, seperti warna tomat. Algoritma KNN kemudian digunakan untuk mengklasifikasikan tomat berdasarkan fitur-fitur yang diidentifikasi dari citra tersebut. Hasil penelitian menunjukkan bahwa aplikasi ini memiliki tingkat akurasi yang tinggi dalam membedakan ketiga kategori kematangan tomat.

Selain peningkatan akurasi dalam identifikasi kematangan, aplikasi ini juga memanfaatkan berbagai teknik pemrosesan citra untuk meningkatkan ketepatan dan efisiensi.

Metode Gaussian digunakan untuk menghaluskan citra dan mengurangi noise, yang dapat mengganggu identifikasi fitur visual. Teknik lower upper bound diaplikasikan untuk segmentasi warna, membantu dalam menentukan rentang warna spesifik yang terkait dengan setiap tingkat kematangan tomat. Masking diterapkan untuk menyorot area tertentu dari citra tomat yang relevan untuk analisis lebih lanjut, sementara mean masking digunakan untuk menghitung nilai rata-rata dari area yang dimask, mengurangi efek anomali lokal.

Selain itu, teknik contour digunakan untuk mendeteksi dan menggambarkan batas objek tomat dalam citra, memungkinkan identifikasi bentuk dan ukuran yang lebih akurat. Kombinasi teknik-teknik ini dalam pemrosesan citra memastikan bahwa fitur-fitur dari tomat diisolasi dan dianalisis dengan tepat, memberikan dasar yang kuat bagi algoritma KNN untuk melakukan klasifikasi dengan akurasi tinggi. Implementasi metode-metode ini menjadikan aplikasi tidak hanya cepat dan efisien, tetapi juga andal dalam mengidentifikasi tingkat kematangan tomat secara konsisten.

5.2 Saran

Berikut adalah beberapa saran untuk pengembangan lebih lanjut dari aplikasi identifikasi kematangan tomat yang telah kami kembangkan:

1. Mengembangkan aplikasi dengan menambahkan beberapa kategori baru untuk kematangan tomat, seperti "Hampir Matang" atau "Lewat Matang", untuk memberikan klasifikasi yang lebih detail dan spesifik.
2. Meningkatkan kemampuan aplikasi untuk mendeteksi buah tomat secara otomatis dan memisahkan objek tomat dari latar belakang, menggunakan teknik segmentasi citra yang lebih canggih seperti deep learning atau metode masking yang lebih presisi.
3. Menguji aplikasi dalam berbagai kondisi pencahayaan dan lingkungan yang berbeda untuk memastikan keandalan dan ketangguhan aplikasi dalam situasi dunia nyata.

Dengan mengimplementasikan saran-saran diatas, kami berharap aplikasi ini dapat menjadi alat yang lebih komprehensif dan handal untuk identifikasi dan klasifikasi kematangan tomat.

DAFTAR PUSTAKA

- Abdillah, S. (2022). HSV COLOR CLASSIFICATION ON HAND IMAGE USING K-MEANS. *Journal Of Information System And Computer*, 1(1), 8-12. Retrieved from <https://ejournal.uniska-kediri.ac.id/index.php/JISCOMP/article/view/2772>
- Anggriawan, M. A., Ichwan, M., & Utami, D. B. (2017). Pengenalan tingkat kematangan tomat berdasarkan citra warna pada studi kasus pembangunan sistem pemilihan otomatis. *Jurnal teknik informatika dan sistem informasi*, 3(3). Retrieved from <https://journal.maranatha.edu/index.php/jutisi/article/view/695>
- Amin, M., Triyanto, J., & Istofa, I. (2020). Rancangan Perangkat Lunak Akuisisi Data Modul Detektor Gamma RosRao Berbasis Modbus Over TCP/IP Menggunakan PyQt5. *PRIMA-Aplikasi dan Rekayasa dalam Bidang Iptek Nuklir*, 17(1), 40-49. Retrieved from <https://jurnal.batan.go.id/index.php/prima/article/view/5920>
- Maliki, I. (2020). Morfologi Citra. Retrieved from <https://repository.unikom.ac.id/62706/1/CV-3-%20Morfologi%20Citra.pdf>
- Marcellino Jonathan, Muhammad Thoriqul Hafidz, Nur Ayu Apriyanti, Zaid Husaini, & Rosyani , P. (2023). MENDETEKSI PLAT NOMOR KENDARAAN DENGAN METODE YOLO (You Only Look Once) DAN SINGLE SHOT DETECTOR (SSD). *AI Dan SPK : Jurnal Artificial Intelligent Dan Sistem Penunjang Keputusan*, 1(1), 105–111. Retrieved from <https://jurnalmahasiswa.com/index.php/aidanspk/article/view/320>
- Muchtar, H., & Apriadi, R. (2019). Implementasi pengenalan wajah pada sistem penguncian rumah dengan metode template matching menggunakan open source computer vision library (opencv). RESISTOR (elektRonika kEndali telekomunikaSI tenaga liSTrik kOmputeR), 2(1), 39-42. Retrieved from <https://jurnal.umj.ac.id/index.php/resistor/article/view/4116/3031>
- Munantri, N. Z., Sofyan, H., & Florestiyanto, M. Y. (2020). Aplikasi Pengolahan Citra Digital Untuk Identifikasi Umur Pohon. *Telematika: Jurnal Informatika dan Teknologi Informasi*, 16(2), 97-104. Retrieved from <https://publikasi.dinus.ac.id/index.php/semantik/article/view/153>
- Riska, A., Darwis, H., & Astuti, W. (2023). Studi Perbandingan Kombinasi GMI, HSV, KNN, dan CNN pada Klasifikasi Daun Herbal. *Indonesian Journal of Computer Science*, 12(3). Retrieved from <http://3.8.6.95/ijcs/index.php/ijcs/article/view/3210/169>
- Swedia, E. R., & Cahyanti, M. (2010). Algoritma Transformasi Ruang Warna. *Depok Univ. Gunadarma*. Retrieved from <https://www.scribd.com/doc/84999919/Pengolahan-Citra-Digital3>

Syarovy, M., Nugroho, A. P., & Sutiarno, L. (2023). PEMANFAATAN MODEL NEURAL NETWORK DALAM GENERASI BARU PERTANIAN PRESISI DI PERKEBUNAN KELAPA SAWIT. WARTA Pusat Penelitian Kelapa Sawit, 28(1), 39-54. Retrieved from <https://warta.iopri.org/index.php/Warta/article/view/97>

Wicaksono, Arifano Teguh (2019) *Prediksi Kepribadian Berdasarkan Tulisan Tangan Dengan Metode Convolutional Neural Network*. Other thesis, Universitas Komputer Indonesia.

Retrieved from
https://elibrary.unikom.ac.id/id/eprint/1194/8/UNIKOM_Arifano%20Teguh%20Wicaksono_bab%202.pdf