

LAPORAN

Disusun untuk memenuhi Tugas Besar mata kuliah IFB-208 Pengolahan Citra Digital
yang diberikan oleh: **Bapak Rizka Milandga Milenio, S.T., M.T.**



Disusun oleh :

- | | |
|------------------------------|---------------|
| 1. Verenada Arsy Mardatillah | (15-2023-058) |
| 2. Aliyya Rahmawati Putri | (15-2023-093) |
| 3. Ridayanti Wardani | (15-2023-168) |

Kelas CC

INSTITUT TEKNOLOGI NASIONAL BANDUNG
FAKULTAS TEKNOLOGI INDUSTRI
INFORMATIKA
2025

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa, yang telah memberikan rahmat dan hidayah-Nya sehingga Kami dapat menyelesaikan penyusunan laporan ini guna memenuhi Ujian Akhir Semester mata kuliah IFB-208 Pengolahan Citra Digital.

Penyusunan laporan dan pengerjaan proyek ini tidak akan berjalan lancar tanpa adanya bimbingan, dukungan, dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini kami ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Rizka Milandga Milenio, S.T., M.T. selaku dosen mata kuliah Pengolahan Citra Digital, atas bimbingan, ilmu, dan motivasi yang telah diberikan kepada kami.
2. Rekan-rekan kelompok, yang telah bekerja sama dan memberikan saran serta masukan yang berharga dalam proses pengerjaan proyek ini.

Laporan ini membahas proses perancangan dan implementasi program untuk melakukan ekstraksi tiga fitur utama—yaitu warna, bentuk, dan tekstur—dari dataset citra sampah. Selain itu, laporan ini juga mencakup implementasi program klasifikasi menggunakan model *Support Vector Machine (SVM)* dan *K-Nearest Neighbors (KNN)* sebagai studi tambahan untuk menguji efektivitas fitur yang diekstraksi.

Kami menyadari sepenuhnya bahwa laporan ini masih jauh dari kata sempurna karena keterbatasan pengetahuan dan pengalaman yang kami miliki. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari para pembaca demi kesempurnaan laporan di masa mendatang.

Akhir kata, kami berharap semoga laporan proyek akhir ini dapat memberikan manfaat dan menambah wawasan bagi pembaca serta bagi pengembangan ilmu pengetahuan, khususnya di bidang pengolahan citra digital.

Bandung, 10 Juni 2025

Tim Penyusun

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR	iv
DAFTAR TABEL.....	vi
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	2
1.3 Tujuan.....	2
1.4 Batasan Masalah	3
1.5 Pembagian Tugas Kelompok	3
BAB II TINJAUAN PUSTAKA.....	4
2.1 Pengolahan Citra Digital.....	4
2.2 Struktur Citra Digital	4
2.3 Pra-pemrosesan Citra	5
2.4 Ekstraksi Fitur.....	7
2.5 Klasifikasi Citra	8
2.6 Metrik Evaluasi.....	9
BAB III PERANCANGAN SISTEM (REVISI)	10
3.1 Desain Umum Sistem	10
3.2 Struktur Folder Proyek.....	10
3.3 Alur Kerja Sistem	13
3.4 Pengumpulan Data Citra	14
3.5 Desain Implementasi.....	15
BAB IV IMPLEMENTASI DAN PENGUJIAN (REVISI)	19
4.1 Lingkungan Implementasi.....	19
4.2 Implementasi Proses Pelatihan (Train.py)	20
4.2.1 Pra-pemrosesan Citra.....	21
4.2.2 Ekstraksi Fitur Warna (Kategori Kertas)	23
4.2.3 Ekstraksi Fitur Bentuk (Kategori Organik).....	25
4.2.4 Ekstraksi Fitur Tekstur (Kategori Plastik)	28
4.2.5 Padding Fitur	30
4.2.6 Kombinasi Fitur.....	30
4.2.7 Pembentukan Dataset	31
4.2.8 Pelatihan Evaluasi dan Model.....	32
4.3 Implementasi Tahap Prediksi (Main.py)	37

4.3.1	Pemilihan Citra Input oleh Pengguna	38
4.3.2	Pembacaan dan Pra-pemrosesan Citra Input.....	38
4.3.3	Ekstraksi Fitur dari Citra Input	38
4.3.4	Pemuatan Aset yang Tersimpan.....	39
4.3.5	Standardisasi dan Prediksi Individual	40
4.3.6	Strategi Keputusan Final (Ensemble)	40
4.3.7	Tampilan Hasil dan Visualisasi Fitur:.....	41
4.4	Hasil Pengujian	42
4.4.1	Hasil Trainning Model.....	42
4.4.2	Hasil Klasifikasi	43
BAB V PENUTUP		54
5.1	Kesimpulan	54
5.2	Saran	55
LAMPIRAN.....		57
A.	Potongan Kode Program – (ekstraksi_klasifikasi.py).....	57
B.	Potongan Kode Program – (main_klasifikasi.py)	63

DAFTAR GAMBAR

Gambar 3.1 Alur Kerja Sistem	14
Gambar 3.2 Sampel Plastik	15
Gambar 3.3 Sampel Kertas	15
Gambar 3.4 Sampel Organik	15
Gambar 4.1 Pra-Pemrosesan	21
Gambar 4.2 Fungsi Resizing	22
Gambar 4.3 Fungsi Denoising.....	22
Gambar 4.4 Fungsi Contrast Enhancement	23
Gambar 4.5 Perhitungan Histogram HSV	24
Gambar 4.6 Implementasi dalam Klasifikasi.....	25
Gambar 4.7 Ekstraksi Bentuk.....	25
Gambar 4.8 Implementasi dalam Klasifikasi.....	27
Gambar 4.9 Ekstreaksi Tekstur	28
Gambar 4.10 Implementasi dalam Klasifikasi.....	30
Gambar 4.11 Pembentuka Dataset hasil Pelatihan	32
Gambar 4.12 Pembentuka Dataset hasil Testing	32
Gambar 4.13 Standarisasi Fitur	33
Gambar 4.14 Model KNN.....	33
Gambar 4.15 Model SVM.....	34
Gambar 4.16 Hasil SVM Warna	35
Gambar 4.17 Hasil SVM Tekstur.....	35
Gambar 4.18 Hasil SVM Bentuk	35
Gambar 4.19 Hasil SVM Kombinasi.....	35
Gambar 4.20 Hasi Keseluruhan.....	36
Gambar 4.21 Hasi Perbandingan Keseluruhan	37
Gambar 4.22 Tahap Prediksi	37
Gambar 4.23 Pemilihan Citra	38
Gambar 4.24 Pra-Pemrosesan Citra Input.....	38
Gambar 4.25 Ekstrasi Input dan Citra Input	39
Gambar 4.26 Pemuatan Aset.....	39
Gambar 4.27 Standarisasi dan Prediksi	40
Gambar 4.28 Gambar Asli 1	43
Gambar 4.29 Hasil Pengujian Bentuk 1	44
Gambar 4.30 Hasil Pengujian Bentuk Statik 1	44
Gambar 4.31 Gambar Asli 2	44
Gambar 4.32 Hasil Pengujian Bentuk 2	45
Gambar 4.33 Hasil Pengujian Bentuk Statik 2	45
Gambar 4.34 Gambar Asli 3	45
Gambar 4.35 Hasil Pengujian Bentuk 3	46
Gambar 4.36 Hasil Pengujian Bentuk Statik 3	46
Gambar 4.37 Gambar Asli 4	47
Gambar 4.38 Hasil Pengujian Tekstur 1.....	47
Gambar 4.39 Hasil Pengujian Tekstu Statik 1	47
Gambar 4.40 Gambar Asli 5	48
Gambar 4.41 Hasil Pengujian Tekstur 2.....	48
Gambar 4.42 Hasil Pengujian Tekstu Statik 2.....	48
Gambar 4.43 Gambar Asli 6	49
Gambar 4.44 Hasil Pengujian Tekstur 3.....	49

Gambar 4.45 Hasil Pengujian Tekstu Statik 3	49
Gambar 4.46 Gambar Asli 7	50
Gambar 4.47 Gambar Asli 8	51
Gambar 4.48 Hasil Pengujian Warna 2	51
Gambar 4.49 Hasil Pengujian Warna Statik 2	51
Gambar 4.50 Gambar Asli 9	52
Gambar 4.51 Hasil Pengujian Warna 3	52
Gambar 4.52 Hasil Pengujian Warna Statik 3	52
Gambar A.1 Kode Program ekstraksi_klasifikasi.py	57
Gambar A.2 Kode Program ekstraksi_klasifikasi.py	58
Gambar A.3 Kode Program ekstraksi_klasifikasi.py	59
Gambar A.4 Kode Program ekstraksi_klasifikasi.py	60
Gambar A.5 Kode Program ekstraksi_klasifikasi.py	61
Gambar A.6 Kode Program ekstraksi_klasifikasi.py	62
Gambar B.1 Kode Program ekstraksi_klasifikasi.py	63
Gambar B.2 Kode Program main_klasifikasi.py	64
Gambar B.3 Kode Program ekstraksi_klasifikasi.py	65
Gambar B.4 Kode Program ekstraksi_klasifikasi.py	66
Gambar B.5 Kode Program ekstraksi_klasifikasi.py	67

DAFTAR TABEL

Tabel 1.1 Pembagian Tugas Kelompok.....	3
Tabel 3.1 Struktur Folder Proyek.....	10
Tabel 4.1 Library yang Digunakan.....	19
Tabel 4.2 Akurasi Model Klasifikasi.....	42

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengolahan Citra Digital (PCD) merupakan bidang ilmu yang krusial dalam era modern, memungkinkan komputer untuk "melihat" dan menginterpretasikan informasi dari citra. Aplikasi PCD sangat luas, mencakup berbagai sektor mulai dari medis, keamanan, hingga pertanian dan industri. Dalam konteks pengelolaan lingkungan, khususnya permasalahan sampah, PCD memiliki potensi besar untuk membantu dalam proses identifikasi dan klasifikasi jenis sampah secara otomatis. Peningkatan volume sampah menjadi masalah global yang mendesak, dan identifikasi yang akurat dapat mendukung upaya daur ulang dan pengelolaan limbah yang lebih efisien.

Proyek ini berfokus pada penerapan konsep PCD untuk mengatasi permasalahan tersebut melalui ekstraksi fitur citra. Citra digital memiliki berbagai karakteristik visual yang dapat dianalisis, seperti warna, bentuk, dan tekstur. Fitur-fitur ini sangat penting untuk mengenali dan membedakan berbagai objek dalam citra. Dalam tugas akhir mata kuliah Pengolahan Citra Digital (PCD), kami ditugaskan untuk membuat sebuah proyek yang melibatkan ekstraksi fitur dari citra. Citra yang digunakan wajib memiliki model warna RGB.

Oleh karena itu, proyek ini bertujuan untuk merancang dan mengimplementasikan program yang mampu melakukan ekstraksi fitur warna, bentuk, dan tekstur dari dataset citra sampah berbasis RGB. Selain itu, sebagai bagian dari penilaian bonus, proyek ini juga akan mengeksplorasi implementasi program klasifikasi citra menggunakan model pembelajaran mesin seperti Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN) untuk menguji efektivitas fitur yang telah diekstraksi dalam mengenali kategori sampah. Dengan demikian, diharapkan proyek ini dapat memberikan kontribusi dalam pengembangan sistem identifikasi sampah otomatis yang lebih baik.

1.2 Rumusan Masalah

Merujuk pada urgensi identifikasi dan klasifikasi sampah melalui pengolahan citra digital yang telah diuraikan dalam latar belakang, berikut merupakan permasalahan yang ingin diselesaikan melalui proyek ini:

1. Bagaimana merancang dan mengimplementasikan program untuk melakukan ekstraksi fitur warna, bentuk, dan tekstur dari dataset citra sampah berformat RGB?
2. Bagaimana mengintegrasikan dan memanfaatkan library Python yang telah ditentukan (OpenCV, scikit-image, Pillow, NumPy, mahotas) dalam proses ekstraksi fitur citra ini?
3. Bagaimana mengevaluasi performa model klasifikasi machine learning, yaitu Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN), dalam mengklasifikasikan jenis sampah berdasarkan fitur-fitur yang telah diekstraksi?

1.3 Tujuan

Tujuan utama dari proyek ini adalah mengembangkan sistem identifikasi dan klasifikasi objek sampah otomatis berbasis pengolahan citra digital. Sistem ini diharapkan dapat mengekstraksi karakteristik visual dari citra sampah untuk mendukung pengelolaan limbah yang efisien. Tujuan khusus dari proyek ini meliputi:

1. Membangun minimal tiga program terpisah untuk ekstraksi fitur warna, bentuk, dan tekstur dari dataset citra sampah berformat RGB.
2. Menerapkan berbagai teknik pra-pemrosesan citra, termasuk konversi ruang warna dan penyesuaian kondisi citra, guna meningkatkan kualitas ekstraksi fitur.
3. Mengimplementasikan dan mengevaluasi performa minimal dua model klasifikasi *machine learning* (SVM dan KNN) untuk mengenali kategori sampah berdasarkan fitur yang diekstraksi, serta menganalisis akurasi.

1.4 Batasan Masalah

1. Proyek ini secara spesifik menggunakan citra dengan model warna RGB sebagai input utama untuk semua proses pengolahan citra.
2. Jumlah data yang digunakan minimal 60 citra RGB objek sampah. Kategorisasi objek sampah akan mencakup minimal tiga kategori (plastik, kertas, dan organik).
3. Pengembangan program terbatas pada penggunaan *library* Python yang telah ditentukan: OpenCV (cv2), scikit-image (skimage), Pillow (PIL), NumPy, scikit-learn, dan mahotas. Penggunaan *library* di luar daftar ini dilarang.
4. Fokus utama ekstraksi fitur adalah pada fitur warna, bentuk, dan tekstur, dengan penugasan spesifik untuk setiap kategori sampah (misalnya, plastik dengan tekstur, kertas dengan warna, dan organik dengan bentuk).

1.5 Pembagian Tugas Kelompok

Proyek akhir ini dikerjakan oleh kelompok yang terdiri dari 3 orang. Pembagian tugas dilakukan secara merata untuk memastikan setiap anggota memiliki kontribusi yang signifikan dalam semua tahapan proyek, mulai dari perancangan, implementasi, hingga penyusunan laporan dan persiapan presentasi. Berikut adalah rincian pembagian tugas setiap anggota:

Tabel 1.1 Pembagian Tugas Kelompok

NRP	Nama	Kontribusi
15-2023-058	Verenada Arsy Mardatillah	Penulisan Latar Belakang, Tujuan, serta Implementasi Ekstraksi Fitur Bentuk (Organik)
15-2023-093	Aliyya Rahmawati Putri	Ekstraksi Fitur Warna (Kertas), Penulisan Kesimpulan & Saran, serta Penyusunan Laporan
15-2023-168	Ridayanti Wardani	Implementasi Ekstraksi Fitur Tekstur (Plastik), Pengujian Klasifikasi, dan Analisis Akurasi

BAB II

TINJAUAN PUSTAKA

2.1 Pengolahan Citra Digital

Pengolahan Citra Digital (PCD) adalah bidang ilmu dan teknologi yang menggunakan komputer untuk memanipulasi citra digital. Tujuan utama PCD adalah meningkatkan kualitas citra untuk interpretasi visual manusia atau mempersiapkan citra untuk analisis mesin. Proses PCD umumnya melibatkan berbagai tahapan, mulai dari akuisisi citra, pra-pemrosesan, segmentasi, ekstraksi fitur, hingga klasifikasi atau interpretasi citra. PCD banyak diterapkan dalam berbagai disiplin ilmu seperti kedokteran, keamanan, geografi, hingga identifikasi objek otomatis.

2.2 Struktur Citra Digital

Citra digital adalah representasi visual dari objek atau *scene* dalam bentuk diskrit yang dapat diproses oleh komputer. Citra digital terdiri dari piksel (*picture element*), yaitu unit terkecil dari informasi visual. Setiap piksel memiliki nilai yang merepresentasikan intensitas warna pada posisi tertentu.

- Model Warna RGB (Red, Green, Blue):

Model warna RGB adalah model warna aditif di mana warna-warna primer Merah, Hijau, dan Biru digabungkan dalam berbagai intensitas untuk menghasilkan spektrum warna yang luas. Dalam citra RGB, setiap piksel direpresentasikan oleh tiga komponen warna (merah, hijau, biru), masing-masing dengan rentang nilai tertentu (misalnya, 0-255). Citra yang digunakan dalam proyek ini WAJIB memiliki model warna RGB.

- Model Warna Lain untuk Pra-pemrosesan dan Ekstraksi Fitur:

Selain RGB, terdapat beberapa model warna lain yang dapat digunakan untuk tujuan tertentu atau sebagai bagian dari pra-pemrosesan citra, seperti yang diterapkan dalam proyek ini:

- Grayscale:

Citra *grayscale* atau citra skala keabuan hanya memiliki satu kanal warna yang merepresentasikan intensitas cahaya (hitam hingga putih). Konversi RGB ke *grayscale* sering digunakan untuk menyederhanakan pemrosesan dan mengurangi kompleksitas data, terutama untuk ekstraksi fitur yang tidak bergantung pada warna, seperti tekstur dan bentuk. Dalam proyek ini, *grayscale* digunakan untuk ekstraksi fitur tekstur dan bentuk.

- HSV (Hue, Saturation, Value):

Model warna HSV merepresentasikan warna berdasarkan *hue* (corak warna), *saturation* (kemurnian warna), dan *value* (kecerahan). Model ini sering lebih intuitif bagi manusia dan dapat memisahkan informasi iluminasi (*value*) dari informasi warna (*hue* dan *saturation*), yang berguna dalam beberapa kasus pra-pemrosesan. Dalam proyek ini, HSV digunakan untuk ekstraksi fitur warna.

- CMYK (Cyan, Magenta, Yellow, Key/Black):

Model warna CMYK adalah model warna subtraktif yang umum digunakan dalam pencetakan. Meskipun tidak umum digunakan dalam ekstraksi fitur citra digital secara langsung, eksplorasi konversi ke CMYK dapat menjadi bagian dari "kreativitas dan *extra effort*" dalam pra-pemrosesan.

2.3 Pra-pemrosesan Citra

Pra-pemrosesan citra adalah tahapan awal dalam pengolahan citra digital yang bertujuan untuk meningkatkan kualitas citra agar sesuai untuk analisis lebih lanjut. Dalam proyek ini, beberapa teknik pra-pemrosesan yang diterapkan meliputi:

- Konversi Ruang Warna Mengubah model warna citra dari satu representasi ke representasi lain (misalnya, dari RGB ke *grayscale*, HSV, atau CMYK) dapat membantu dalam menonjolkan fitur tertentu atau menyederhanakan

data. Teknik ini digunakan untuk mengubah citra RGB asli menjadi HSV untuk ekstraksi warna, dan *grayscale* untuk ekstraksi bentuk dan tekstur.

- Normalisasi, Menyesuaikan rentang intensitas piksel dalam citra untuk memastikan konsistensi. Ini penting untuk memastikan fitur yang diekstraksi memiliki skala yang seragam.
- Segmentasi, Membagi citra menjadi beberapa segmen atau objek yang lebih bermakna. Dalam konteks ekstraksi bentuk, segmentasi sering diperlukan untuk mengisolasi objek yang akan dianalisis. Teknik seperti *adaptive thresholding* dan operasi morfologi (*opening*) digunakan untuk segmentasi dalam proyek ini.
- Penyesuaian Kondisi Citra, Menyesuaikan citra terhadap variasi kondisi seperti terang, redup, atau berembun. Ini penting untuk memastikan konsistensi hasil ekstraksi fitur terlepas dari kondisi pengambilan citra. Teknik yang diterapkan dalam proyek ini untuk penyesuaian kondisi citra meliputi:
 - Denoising (Penghilangan Noise) Menggunakan teknik seperti `cv2.fastNlMeansDenoisingColored` untuk mengurangi noise pada citra, yang dapat mengganggu akurasi ekstraksi fitur.
 - Contrast Enhancement (Peningkatan Kontras) Menggunakan teknik seperti CLAHE (*Contrast Limited Adaptive Histogram Equalization*) untuk meningkatkan kontras citra, membantu menonjolkan detail dan batas objek, terutama pada citra dengan pencahayaan kurang optimal.
- Resizing (Penyesuaian Ukuran) Mengubah ukuran citra menjadi dimensi yang seragam (misalnya, 100x100 piksel) untuk memastikan konsistensi input bagi proses ekstraksi fitur dan model klasifikasi.

2.4 Ekstraksi Fitur

Ekstraksi fitur adalah proses untuk mendapatkan informasi deskriptif dari citra yang dapat digunakan untuk tujuan klasifikasi atau analisis lebih lanjut. Dalam proyek ini, tiga jenis fitur utama akan diekstraksi:

- Fitur Warna menggambarkan distribusi warna dalam suatu citra atau objek. Salah satu metode yang umum digunakan adalah *Color Histogram* yang merupakan representasi statistik distribusi warna dalam suatu citra. Histogram warna merekam seberapa sering setiap warna muncul dalam citra, memberikan gambaran global tentang komposisi warna citra. Dalam implementasi ini, histogram warna diekstraksi dari ruang warna HSV dengan $16 \times 16 \times 16$ bin untuk menangkap informasi *hue*, *saturation*, dan *value*. Library OpenCV digunakan untuk menghitung histogram warna.
- Fitur Bentuk merepresentasikan karakteristik geometris objek dalam citra. Dalam proyek ini, metode yang digunakan adalah **Hu Moments** dan **Contour Features**. Hu Moments merupakan tujuh nilai invariansi momen yang dihitung dari suatu bentuk, yang tidak berubah terhadap translasi, skala, dan rotasi. Hu Moments sering digunakan untuk pengenalan bentuk karena sifat invarian-nya. OpenCV menyediakan fungsi untuk menghitung Hu Moments. Sebelum perhitungan Hu Moments, dilakukan pra-pemrosesan citra seperti konversi ke *grayscale*, *Gaussian blurring*, *adaptive thresholding*, dan deteksi kontur untuk mengisolasi objek. Hasil Hu Moments kemudian di-*transformasi log* untuk meningkatkan diskriminasi fitur. *Contour Features* adalah kurva yang menghubungkan semua titik kontinu di sepanjang batas objek, memiliki warna atau intensitas yang sama. Fitur kontur dapat memberikan informasi tentang bentuk objek. Library OpenCV dan scikit-image mendukung ekstraksi fitur kontur.
- Fitur Tekstur menggambarkan pola berulang atau karakteristik spasial permukaan suatu objek dalam citra. Metode yang digunakan dalam proyek ini antara lain:
 - Gray-Level Co-occurrence Matrix (GLCM) adalah matriks yang menghitung frekuensi kemunculan pasangan piksel dengan nilai

intensitas tertentu pada jarak dan arah tertentu. Dari GLCM, berbagai properti tekstur seperti *contrast*, *dissimilarity*, *homogeneity*, *energy*, dan *correlation* dapat diekstraksi. Scikit-image digunakan untuk ekstraksi fitur tekstur GLCM ini.

- Local Binary Patterns (LBP) adalah operator deskriptor tekstur yang kuat, yang melabeli piksel citra dengan membandingkan ambang batas tetangganya. LBP sering digunakan untuk klasifikasi tekstur dan pengenalan wajah. Scikit-image mendukung fitur tekstur seperti LBP.

2.5 Klasifikasi Citra

Klasifikasi citra adalah proses penugasan label kategori ke citra atau objek berdasarkan fitur-fitur yang diekstraksi. Dalam proyek ini, dua model *machine learning* akan digunakan:

- K-Nearest Neighbors (KNN) adalah algoritma klasifikasi non-parametrik yang mengklasifikasikan titik data baru berdasarkan mayoritas kelas dari k tetangga terdekatnya dalam ruang fitur. Algoritma ini mudah diimplementasikan dan sering digunakan sebagai *baseline*. Dalam proyek ini, **optimasi *hyperparameter* (*n_neighbors*) dilakukan menggunakan *GridSearchCV*** untuk menemukan nilai k terbaik guna meningkatkan performa model. Scikit-learn menyediakan implementasi untuk model KNN.
- Support Vector Machine (SVM) adalah model *machine learning* yang mencari *hyperplane* terbaik untuk memisahkan kelas-kelas dalam ruang fitur. SVM efektif dalam ruang berdimensi tinggi dan fleksibel karena dapat menggunakan fungsi *kernel* yang berbeda. Dalam proyek ini, **optimasi *hyperparameter* (*C*, *gamma*, dan *kernel*) dilakukan menggunakan *GridSearchCV***. Khususnya, *kernel* **Radial Basis Function (RBF)** digunakan untuk mengakomodasi pemisahan data yang kompleks. Scikit-learn menyediakan implementasi untuk model SVM.

2.6 Metrik Evaluasi

Untuk mengukur performa model klasifikasi, metrik evaluasi digunakan:

- Akurasi (Accuracy), mengukur proporsi prediksi benar dari total keseluruhan prediksi. Akurasi dihitung sebagai jumlah prediksi benar dibagi dengan total jumlah data. Scikit-learn menyediakan alat untuk menghitung akurasi.
- Confusion Matrix, *Confusion Matrix* adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi pada sekumpulan data uji yang nilai benarnya diketahui. Matriks ini memungkinkan visualisasi kinerja algoritma, terutama dalam mengidentifikasi di mana model melakukan kesalahan. Scikit-learn menyediakan alat untuk membuat *confusion matrix*.
- Classification Report (Precision, Recall, F1-Score), Selain akurasi, *classification report* menyediakan metrik yang lebih detail per kelas, yaitu:
 - Precision adalah prediksi positif yang benar (True Positives) dari semua prediksi positif yang dibuat oleh model (True Positives + False Positives).
 - Recall (Sensitivity) Proporsi kasus positif aktual yang diidentifikasi dengan benar (True Positives) dari semua kasus positif yang sebenarnya (True Positives + False Negatives).
 - F1-Score atau Rata-rata harmonik dari *precision* dan *recall*. Metrik ini sangat berguna ketika kelas tidak seimbang, karena memberikan keseimbangan antara *precision* dan *recall*. Scikit-learn menyediakan fungsi untuk menghasilkan *classification report*.

BAB III

PERANCANGAN SISTEM

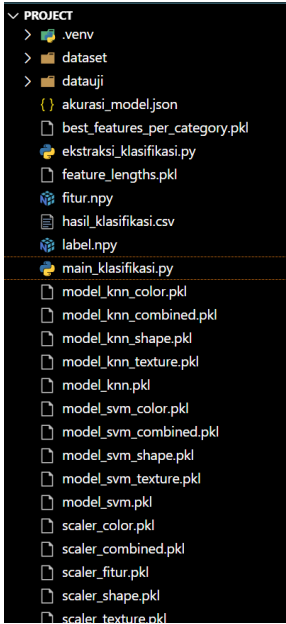
(REVISI)

3.1 Desain Umum Sistem

Sistem ini menggambarkan arsitektur menyeluruh dari proyek identifikasi dan klasifikasi sampah berbasis pengolahan citra digital. Sistem dirancang untuk mengintegrasikan berbagai tahapan kunci, mulai dari akuisisi citra, pra-pemrosesan citra yang ditingkatkan, ekstraksi beragam fitur visual yang relevan (warna, bentuk, dan tekstur) untuk setiap citra, hingga tahap klasifikasi menggunakan algoritma *machine learning*. Pendekatan modular diterapkan untuk setiap tahapan, memungkinkan fleksibilitas dalam pengembangan dan pemeliharaan. Organisasi file dan folder proyek juga dirancang secara sistematis untuk memastikan kemudahan navigasi dan pengelolaan aset, kode sumber, serta hasil keluaran yang dihasilkan.

3.2 Struktur Folder Proyek

Tabel 3.1 Struktur Folder Proyek

	<p>PROJECT - PCD FIX/</p> <ul style="list-style-type: none"> — .venv/ — dataset/ — datauji/ — akurasi_model.json — best_features_per_category.pkl — ekstraksi_klasifikasi.py — feature_lengths.pkl — fitur.npy — hasil_klasifikasi.csv
---	--

	<ul style="list-style-type: none"> — label.npy — main_klasifikasi.py — model_knn_color.pkl — model_knn_combined.pkl — model_knn_shape.pkl — model_knn_texture.pkl — model_knn.pkl — model_svm_color.pkl — model_svm_combined.pkl — model_svm_shape.pkl — model_svm_texture.pkl — model_svm.pkl — scaler_color.pkl — scaler_combined.pkl — scaler_fitur.pkl — scaler_shape.pkl — scaler_texture.pkl — requirements.txt
--	---

- .venv/: Lingkungan virtual Python yang mengisolasi dependensi proyek, memastikan kompatibilitas antar *library* dan menghindari konflik versi.
- dataset/: Folder ini berfungsi sebagai repositori utama untuk citra sampah yang digunakan sebagai **data pelatihan (training data)**. Di dalamnya berisi sub-folder untuk setiap kategori sampah (kertas/, organik/, plastik/).
- datauji/: Folder ini berisi citra-citra sampah yang digunakan sebagai **data pengujian (testing data)**. Struktur di dalamnya serupa dengan dataset/, dengan sub-folder untuk setiap kategori sampah yang terpisah dari data pelatihan.

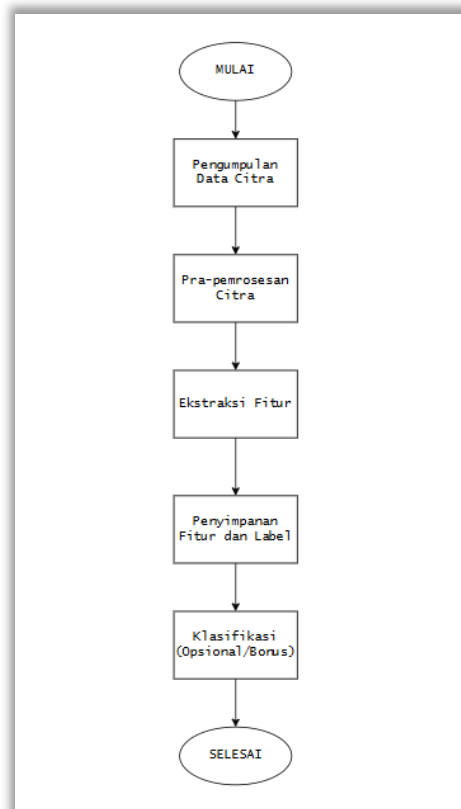
- `akurasi_model.json`: File JSON ini menyimpan hasil akurasi atau metrik evaluasi model klasifikasi yang dieksekusi selama proses pelatihan.
- `best_features_per_category.pkl`: File biner (pickle) ini menyimpan informasi mengenai kombinasi fitur dan jenis model (KNN atau SVM) yang memberikan performa F1-Score terbaik untuk setiap kategori sampah selama proses pelatihan. Ini digunakan untuk panduan dalam prediksi.
- `ekstraksi_klasifikasi.py`: Skrip Python ini kemungkinan besar berisi fungsi-fungsi utama untuk melakukan pra-pemrosesan citra, ekstraksi fitur (warna, tekstur, bentuk), serta logika pelatihan dan evaluasi model klasifikasi.
- `feature_lengths.pkl`: File biner (pickle) yang menyimpan panjang dimensi fitur untuk setiap jenis fitur (warna, tekstur, bentuk, dan kombinasi) setelah proses *padding*. Ini penting untuk memastikan konsistensi dimensi input saat melakukan prediksi pada gambar baru.
- `fitur.npy`: File biner NumPy yang menyimpan data fitur-fitur yang telah diekstraksi (seringkali sudah dalam bentuk gabungan atau *padded*) dari seluruh dataset citra pelatihan. File ini merupakan representasi numerik dari karakteristik visual.
- `hasil_klasifikasi.csv`: File berformat CSV yang berisi hasil dari proses klasifikasi, seperti prediksi kelas untuk setiap citra yang diuji, serta label aslinya.
- `label.npy`: File biner NumPy yang menyimpan label kelas (kategori: kertas, organik, plastik) yang sesuai dengan citra-citra pada dataset pelatihan. Data ini krusial untuk pelatihan dan pengujian model klasifikasi.
- `main_klasifikasi.py`: Skrip Python utama yang mengorkestrasi alur klasifikasi, termasuk pemuatan model dan *scaler* yang telah dilatih, serta proses prediksi pada gambar baru yang diberikan oleh pengguna.
- `model_knn_color.pkl`, `model_knn_combined.pkl`, `model_knn_shape.pkl`, `model_knn_texture.pkl`, `model_knn.pkl`: File-file serialisasi (pickle) ini berisi model K-Nearest Neighbors (KNN) yang telah dilatih. Model-model ini dispesialisasi berdasarkan jenis fitur yang digunakan untuk pelatihannya.

(misalnya, `_color` untuk fitur warna, `_shape` untuk bentuk, `_texture` untuk tekstur, dan `_combined` untuk fitur gabungan). `model_knn.pkl` mungkin merupakan model KNN umum atau hasil dari iterasi sebelumnya.

- `model_svm_color.pkl`, `model_svm_combined.pkl`, `model_svm_shape.pkl`, `model_svm_texture.pkl`, `model_svm.pkl`: Serupa dengan model KNN, file-file ini berisi model Support Vector Machine (SVM) yang telah dilatih, disesuaikan berdasarkan jenis fitur yang digunakan.
- `scaler_color.pkl`, `scaler_combined.pkl`, `scaler_fitur.pkl`, `scaler_shape.pkl`, `scaler_texture.pkl`: File-file serialisasi (pickle) ini berisi objek `StandardScaler` yang digunakan untuk normalisasi fitur. Setiap *scaler* dilatih secara terpisah untuk jenis fitur yang berbeda, memastikan bahwa data input untuk setiap model dinormalisasi dengan cara yang konsisten dengan data pelatihan mereka. `scaler_fitur.pkl` mungkin merupakan *scaler* untuk fitur gabungan yang lebih awal atau umum.
- `requirements.txt`: File teks yang mencantumkan semua *library* Python yang dibutuhkan oleh proyek beserta versi spesifiknya. Ini memfasilitasi replikasi lingkungan pengembangan yang sama di sistem lain.

3.3 Alur Kerja Sistem

Diagram alir pada Gambar 3.1 menggambarkan alur kerja umum sistem identifikasi dan klasifikasi sampah yang dirancang. Proses dimulai dengan Pengumpulan Data Citra, di mana citra-citra sampah dari berbagai kategori disiapkan. Selanjutnya, citra-citra tersebut melalui tahap Pra-pemrosesan Citra untuk meningkatkan kualitas dan relevansi data. Setelah pra-pemrosesan, dilakukan Ekstraksi Fitur (warna, bentuk, dan tekstur) untuk mendapatkan representasi numerik dari karakteristik citra. Fitur dan label yang diekstraksi kemudian disimpan pada tahap Penyimpanan Fitur dan Label. Tahap berikutnya adalah Klasifikasi, yang bersifat opsional/bonus, di mana model *machine learning* dilatih dan dievaluasi menggunakan fitur yang telah diekstraksi. Proses ini berakhir setelah semua tahapan selesai.



Gambar 3.1 Alur Kerja Sistem

3.4 Pengumpulan Data Citra

Dataset citra yang digunakan dalam proyek ini merupakan koleksi citra objek sampah. Sumber data citra dapat berasal dari pengambilan langsung atau dari sumber *online* yang relevan. Proyek ini menggunakan 90 citra RGB dengan 73 Data *Training* dan 17 Data *Testing* yang dibagi menjadi tiga kategori objek sampah, yaitu plastik, kertas, dan organik. Dengan pembagian kategori sampah:

- Plastik: Untuk kategori ini, ekstraksi fitur tekstur menjadi fokus utama.
- Kertas: Untuk kategori ini, ekstraksi fitur warna menjadi fokus utama.
- Organik: Untuk kategori ini, ekstraksi fitur bentuk menjadi fokus utama.

- Contoh Sampel Citra:



Gambar 3.2 Sampel Plastik



Gambar 3.3 Sampel Kertas



Gambar 3.4 Sampel Organik

3.5 Desain Implementasi

Desain implementasi dari sistem ini mengacu pada tahapan-tahapan yang telah dirancang sebelumnya, yaitu mulai dari akuisisi citra, pra-pemrosesan yang komprehensif, ekstraksi beragam fitur, hingga klasifikasi. Implementasi dilakukan menggunakan bahasa Python dengan *library* yang telah ditentukan seperti OpenCV, NumPy, Pillow, scikit-image, mahotas, dan scikit-learn.

Proses implementasi dibagi menjadi dua bagian besar, yaitu program ekstraksi fitur dan program klasifikasi citra. Setiap bagian dibangun secara modular agar mudah diuji, dikembangkan, dan dipelihara secara terpisah.

- **Ekstraksi Fitur** Program ekstraksi fitur dirancang untuk mengekstrak tiga jenis fitur visual utama dari setiap citra, tanpa membatasi pada kategori sampah tertentu. Ini memungkinkan fleksibilitas dalam analisis dan klasifikasi gabungan:

- **Fitur Warna:** Ekstraksi dilakukan menggunakan histogram HSV. Histogram dihitung dari masing-masing kanal H, S, dan V dalam $16 \times 16 \times 16$ bin, menghasilkan vektor fitur dengan 4096 dimensi. Normalisasi dilakukan untuk memastikan rentang nilai distribusi yang seragam.
- **Fitur Tekstur:** Ekstraksi dilakukan dengan menggabungkan dua metode:
 - **Gray-Level Co-occurrence Matrix (GLCM)** Berbagai properti tekstur diekstrak dari GLCM, termasuk *contrast*, *dissimilarity*, *homogeneity*, *energy*, dan *correlation*.
 - **Local Binary Patterns (LBP)**, Histogram **LBP** diekstrak untuk menangkap pola tekstur lokal. Fitur-fitur ini digabungkan menjadi satu vektor fitur tekstur yang komprehensif.
- **Fitur Bentuk:** Ekstraksi dilakukan menggunakan **Hu Moments**, yaitu tujuh nilai invarian yang tidak berubah terhadap translasi, skala, dan rotasi. Sebelum perhitungan Hu Moments, citra melalui serangkaian pra-pemrosesan untuk isolasi bentuk (lihat bagian Pra-pemrosesan Citra). Nilai Hu Moments kemudian di-*transformasi log* untuk meningkatkan diskriminasi.

Setiap hasil ekstraksi fitur (warna, tekstur, bentuk) dikonversi ke dalam bentuk *array* numerik dan disesuaikan panjangnya dengan *padding* agar seragam. Fitur-fitur ini kemudian disimpan secara internal atau digunakan untuk membentuk fitur gabungan. Label kategori untuk setiap citra disimpan terpisah.

- **Pra-pemrosesan Citra** Sebelum dilakukan ekstraksi fitur, setiap citra melewati tahap pra-pemrosesan untuk meningkatkan kualitas dan efektivitas analisis. Langkah-langkah yang diterapkan meliputi:
 - **Resize Citra**, Semua citra diubah ukurannya menjadi dimensi seragam (misalnya, 100×100 piksel) menggunakan interpolasi `cv2.INTER_AREA` untuk memastikan konsistensi input.

- Denoising:, Teknik *non-local means denoising* (cv2.fastNlMeansDenoisingColored) diterapkan untuk mengurangi *noise* pada citra, yang dapat mempengaruhi akurasi ekstraksi fitur.
 - Contrast Enhancement, Contrast Limited Adaptive Histogram Equalization (CLAHE) diterapkan untuk meningkatkan kontras citra, membantu menonjolkan detail objek, terutama pada pencahayaan yang bervariasi.
 - Konversi Ruang Warna, Citra dikonversi ke ruang warna yang sesuai kebutuhan fitur yang akan diekstraksi (misalnya RGB ke HSV untuk warna, dan *grayscale* untuk bentuk dan tekstur).
 - Normalisasi Nilai Piksel, Nilai intensitas piksel dinormalisasi untuk meningkatkan konsistensi hasil ekstraksi.
 - Segmentasi (khusus untuk ekstraksi bentuk), Untuk ekstraksi bentuk, dilakukan proses tambahan yaitu *Gaussian blurring* dan *adaptive thresholding* untuk mengubah citra menjadi biner, diikuti dengan deteksi kontur untuk mengisolasi objek dari latar belakang.
- **Klasifikasi Citra** Sebagai tahap tambahan, sistem juga mengimplementasikan proses klasifikasi citra untuk mengevaluasi efektivitas fitur yang telah diekstraksi. Dua algoritma *machine learning* digunakan:
- K-Nearest Neighbors (KNN) Klasifikasi dilakukan berdasarkan kedekatan fitur dengan data pelatihan dalam ruang fitur. Model KNN dilatih dengan optimasi *hyperparameter* (n_neighbors) menggunakan *GridSearchCV*.
 - Support Vector Machine (SVM) Digunakan untuk menemukan *hyperplane* terbaik yang memisahkan kelas-kelas citra berdasarkan fitur. Model SVM dilatih dengan optimasi *hyperparameter* (C, gamma, dan *kernel* RBF) menggunakan *GridSearchCV*.

Model klasifikasi dilatih menggunakan data fitur yang telah diekstrak (baik fitur individual maupun gabungan) dan label kategorinya. Hasil evaluasi akurasi dan *classification report* disimpan, dan *confusion matrix* juga divisualisasikan. Model yang telah dilatih dan *scaler* normalisasi disimpan dalam format .pkl untuk penggunaan kembali tanpa perlu pelatihan ulang.

- **Struktur Modular** Setiap komponen sistem diimplementasikan secara terpisah dalam skrip dan file eksternal untuk meningkatkan keterbacaan, fleksibilitas, dan kemudahan pengelolaan:

- `ekstraksi_klasifikasi.py` (atau `train_model.py`): Mengintegrasikan proses pra-pemrosesan, ekstraksi fitur, pelatihan model, dan evaluasi awal.
- `main_klasifikasi.py` (atau `predict_image.py`): Bertanggung jawab untuk memuat model dan *scaler* yang telah dilatih, serta melakukan prediksi pada gambar baru.
- **File eksternal:**
 - `.npy`: Menyimpan data fitur (`fitur.npy`) dan label (`label.npy`).
 - `.pkl`: Menyimpan model KNN dan SVM yang telah dilatih untuk setiap jenis fitur (misalnya, `model_knn_color.pkl`, `model_svm_texture.pkl`, `model_knn_combined.pkl`), *scaler* yang sesuai (misalnya, `scaler_color.pkl`, `scaler_combined.pkl`), serta informasi fitur terbaik per kategori (`best_features_per_category.pkl`) dan panjang fitur (`feature_lengths.pkl`).
 - `.csv`: Menyimpan hasil klasifikasi (`hasil_klasifikasi.csv`).
 - `.json`: Menyimpan hasil akurasi model (`akurasi_model.json`).

Desain modular ini memudahkan pemeliharaan dan memungkinkan penambahan fitur baru di masa mendatang, seperti visualisasi data yang lebih interaktif atau prediksi berbasis antarmuka pengguna grafis.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

(REVISI)

4.1 Lingkungan Implementasi

Implementasi sistem dilakukan dalam lingkungan pengembangan lokal menggunakan perangkat komputasi yang mendukung pemrosesan citra digital. Seluruh proses pengembangan, pengujian, dan evaluasi dijalankan dengan bahasa pemrograman Python serta library pendukung yang telah ditentukan dalam ruang lingkup proyek.

Lingkungan pengembangan mencakup sistem operasi Windows 10/11 dan IDE seperti Visual Studio Code atau PyCharm. Semua dependensi proyek dikelola menggunakan virtual environment dengan bantuan pip serta file requirements.txt.

Berikut ini adalah daftar library utama beserta fungsi utamanya dalam sistem:

Tabel 4.1 Library yang Digunakan

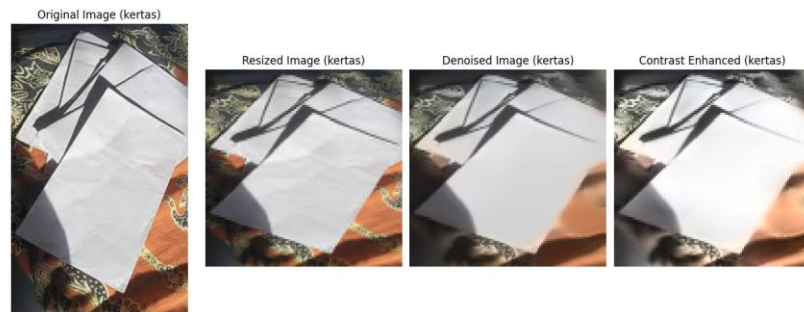
No	Library	Fungsi / Peran Utama
1	NumPy	Operasi numerik tingkat tinggi seperti manipulasi array, padding, flattening fitur
2	OpenCV	Pengolahan citra: konversi ruang warna (RGB → HSV/Grayscale), histogram, thresholding, kontur
3	scikit-image	Ekstraksi fitur tekstur menggunakan GLCM (Gray-Level Co-occurrence Matrix) dan properti statistiknya
4	scikit-learn	Model klasifikasi KNN dan SVM, pembagian data (train-test split), evaluasi akurasi, confusion matrix

5	joblib	Menyimpan dan memuat model terlatih (KNN & SVM) dalam format .pkl
6	pandas	Menyimpan hasil klasifikasi ke dalam file CSV
7	Matplotlib	Visualisasi hasil klasifikasi: confusion matrix, grafik akurasi, dan plot citra
8	os (builtin)	Navigasi direktori, memeriksa eksistensi file/folder
9	collections (builtin)	Menghitung jumlah citra per kategori dengan Counter()
10	radom	Untuk sampling acak dataset (jika sample_per_class digunakan).
11	tkinter, filedialog	Membuat antarmuka pengguna grafis sederhana untuk memilih file gambar pada bagian prediksi.

4.2 Implementasi Proses Pelatihan (Train.py)

Pada bagian ini akan dijelaskan langkah-langkah implementasi kode program Training, mulai dari tahapan pra-pemrosesan citra, ekstraksi fitur berdasarkan kategori sampah, hingga proses klasifikasi citra menggunakan model pembelajaran mesin KNN dan SVM.

4.2.1 Pra-pemrosesan Citra



Gambar 4.1 Pra-Pemrosesan

Sebelum fitur dapat diekstraksi, citra terlebih dahulu melalui tahap pra-pemrosesan untuk meningkatkan konsistensi dan efektivitas analisis. Pada proyek ini, beberapa langkah pra-pemrosesan yang diterapkan meliputi:

a. Pembacaan Citra

Citra masukan dibaca dari direktori menggunakan pustaka OpenCV (cv2). Mengubah data citra dari format file (misalnya .jpg, .png) menjadi array numerik yang dapat diproses oleh komputer.

b. Pengubahan Ukuran (Resizing)

Semua citra diubah ukurannya menjadi dimensi yang seragam, yaitu 100×100 piksel. Fungsi `cv2.resize()` digunakan dengan metode interpolasi `cv2.INTER_AREA`.

Tujuan:

- **Standardisasi Dimensi:** Memastikan semua citra memiliki ukuran yang konsisten, yang sangat penting untuk memastikan vektor fitur yang diekstrak memiliki panjang yang sama.
- **Efisiensi Komputasi:** Mengurangi jumlah piksel pada citra berukuran besar, sehingga mempercepat proses komputasi pada tahap ekstraksi fitur dan pelatihan model.
- **Mengurangi Variabilitas:** Membantu mengurangi variabilitas yang disebabkan oleh perbedaan resolusi atau ukuran citra asli.

```
img_resized = cv2.resize(img, target_size, interpolation=cv2.INTER_AREA)
axes[current_plot_index].imshow(cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB))
axes[current_plot_index].set_title(f"Resized Image {title_suffix}")
axes[current_plot_index].axis('off')
current_plot_index += 1
```

Gambar 4.2 Fungsi Resizing

c. Denoising (Pengurangan Noise)

Citra yang sudah diubah ukurannya kemudian diproses untuk mengurangi *noise* (gangguan) yang mungkin ada. Fungsi `cv2.fastNlMeansDenoisingColored()` diterapkan pada citra. Parameter yang digunakan seperti `hColor=10`, `h=10`, `templateWindowSize=7`, dan `searchWindowSize=21` dikonfigurasi untuk efek denoising yang optimal.

Tujuan:

- Meningkatkan Kualitas Citra: Menghilangkan bintik-bintik, *grain*, atau *noise* lain yang dapat mengganggu representasi visual citra.
- Memperjelas Fitur: Citra yang lebih bersih akan menghasilkan fitur yang lebih jelas dan akurat, karena *noise* tidak akan disalahartikan sebagai pola fitur.

```
if denoise:
    img_to_denoise = img_resized.copy() # Denoise the resized image
    denoised_img = cv2.fastNlMeansDenoisingColored(img_to_denoise, None, 10, 10, 7, 21)
    axes[current_plot_index].imshow(cv2.cvtColor(denoised_img, cv2.COLOR_BGR2RGB))
    axes[current_plot_index].set_title(f"Denoised Image {title_suffix}")
    axes[current_plot_index].axis('off')
    img_resized = denoised_img
    current_plot_index += 1
```

Gambar 4.3 Fungsi Denoising

d. Peningkatan Kontras (Contrast Enhancement)

Kontras citra ditingkatkan, terutama pada area lokal, untuk membuat detail objek lebih menonjol. Metode **CLAHE** (Contrast Limited Adaptive Histogram Equalization) digunakan.

- Citra dikonversi dari ruang warna BGR ke LAB (`cv2.cvtColor(img, cv2.COLOR_BGR2LAB)`).

- Hanya kanal L (Lightness/Luminositas) yang diekstrak dan diterapkan CLAHE (`cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))`).
- Kanal L yang sudah ditingkatkan kontrasnya kemudian digabungkan kembali dengan kanal A dan B, lalu dikonversi kembali ke BGR.

Tujuan:

- Menonjolkan Detail: CLAHE secara adaptif meningkatkan kontras di berbagai bagian citra, yang sangat berguna untuk citra dengan pencahayaan yang tidak merata.
- Mempermudah Ekstraksi Fitur: Peningkatan kontras membantu algoritma ekstraksi fitur (terutama tekstur dan bentuk) untuk mengidentifikasi tepi, pola, dan struktur objek dengan lebih baik.

```
if enhance_contrast:
    img_to_enhance = img_resized.copy() # Enhance the (possibly) denoised image
    lab = cv2.cvtColor(img_to_enhance, cv2.COLOR_BGR2LAB)
    l_channel = lab[:, :, 0]
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
    cl = clahe.apply(l_channel)
    lab[:, :, 0] = cl
    contrast_enhanced_img = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    axes[current_plot_index].imshow(cv2.cvtColor(contrast_enhanced_img, cv2.COLOR_BGR2RGB))
    axes[current_plot_index].set_title(f"Contrast Enhanced {title_suffix}")
    axes[current_plot_index].axis('off')
    img_resized = contrast_enhanced_img # Final processed image
    current_plot_index += 1
```

Gambar 4.4 Fungsi Contrast Enhancement

4.2.2 Ekstraksi Fitur Warna (Kategori Kertas)

Fitur warna digunakan untuk mewakili distribusi spektrum warna dari citra. Dalam proyek ini, fitur warna diekstraksi menggunakan histogram HSV tiga dimensi, yang mampu menangkap informasi hue (warna), saturation (kejenuhan), dan value (kecerahan) secara bersamaan.

a. Alasan Pemilihan HSV

Model warna HSV dipilih karena lebih stabil terhadap pencahayaan dan lebih mendekati persepsi manusia terhadap warna dibanding model RGB. Histogram HSV juga cenderung lebih efektif untuk objek berwarna mencolok seperti kertas.

b. Perhitungan Histogram HSV

Dihitung histogram 3D dari kanal Hue, Saturation, dan Value. Histogram ini merepresentasikan distribusi frekuensi piksel berdasarkan nilai HSV-nya.

- **[0, 1, 2]** : Menentukan kanal Hue, Saturation, dan Value.
- **[16, 16, 16]** : Jumlah *bin* untuk setiap kanal (masing-masing 16 bin).
- **[0, 180, 0, 256, 0, 256]** : Rentang nilai untuk setiap kanal (Hue: 0-180, Saturation: 0-256, Value: 0-256).

Tujuan:

Menghasilkan representasi numerik yang ringkas dari karakteristik warna citra. Setiap bin dalam histogram menunjukkan seberapa sering kombinasi warna tertentu muncul.

```
def extract_color(img):  
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  
    hist = cv2.calcHist([hsv], [0, 1, 2], None, [8, 8, 8],  
                        [0, 180, 0, 256, 0, 256])  
    cv2.normalize(hist, hist)  
    return hist.flatten()
```

Gambar 4.5 Perhitungan Histogram HSV

c. Normalisasi dan Perataan (Flattening)

Histogram yang telah dihitung dinormalisasi dan kemudian diratakan menjadi vektor 1D. `cv2.normalize(hist, hist)` diikuti dengan `hist.flatten()`.

Tujuan :

Normalisasi memastikan bahwa histogram fitur memiliki skala yang seragam, tidak peduli ukuran citra atau jumlah piksel. Perataan mengubah histogram 3D menjadi vektor tunggal yang siap untuk input model *machine learning*.

Dari proses ini akan menghasilkan sebuah vektor fitur warna dengan panjang $16 \times 16 \times 16 = 4096$ elemen.

d. Implementasi dalam Klasifikasi

Fungsi ini dipanggil secara otomatis jika gambar terdeteksi kertas. Hasil ekstraksi akan menjadi vektor fitur yang digunakan sebagai input model KNN dan SVM.

```
color_feat = extract_color(processed_img)
```

Gambar 4.6 Implementasi dalam Klasifikasi

4.2.3 Ekstraksi Fitur Bentuk (Kategori Organik)

```
def extract_shape(img):  
    """  
    Mengekstrak fitur bentuk menggunakan Hu Moments setelah segmentation.  
    """  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
    blur = cv2.GaussianBlur(gray, (5, 5), 0)  
    thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
    | | | | | | | | cv2.THRESH_BINARY_INV, 11, 2)  
    kernel = np.ones((3, 3), np.uint8)  
    opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=3)  
  
    contours, _ = cv2.findContours(opening, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)  
    hu = np.zeros((7,), dtype=np.float32)  
  
    if contours:  
        cnt = max(contours, key=cv2.contourArea)  
        moments = cv2.moments(cnt)  
        if moments['m00'] != 0:  
            hu = cv2.HuMoments(moments).flatten()  
            hu = -np.sign(hu) * np.log10(np.abs(hu) + 1e-7)  
    return hu
```

Gambar 4.7 Ekstraksi Bentuk

Fitur bentuk digunakan untuk mengenali karakteristik geometris objek dalam citra. Pada proyek ini, fitur bentuk diekstraksi menggunakan Hu Moments, yaitu tujuh nilai invarian yang menggambarkan bentuk dan tidak terpengaruh oleh rotasi, skala, maupun translasi.

a. Alasan Pemilihan Hu Moments

Kategori organik (misalnya, daun, sisa makanan) cenderung memiliki bentuk yang khas dan tidak teratur. Oleh karena itu, representasi bentuk sangat efektif untuk membedakan jenis sampah ini dari kategori lain.

b. Konversi ke Grayscale

Mempersiapkan citra untuk segmentasi dan deteksi kontur. Konversi gambar ke grayscale.

c. Penghalusan (Gaussian Blur)

Mengurangi *noise* halus dan detail yang tidak relevan, yang dapat membantu proses segmentasi dan membuat tepi objek lebih jelas.

d. Thresholding Aдаftif

Citra biner dihasilkan dengan menerapkan *adaptive thresholding*. Piksel diklasifikasikan sebagai objek atau latar belakang berdasarkan ambang batas yang dihitung secara lokal untuk setiap area.

```
cv2.adaptiveThreshold(blur, 255,  
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY_INV, 11, 2)
```

- **ADAPTIVE_THRESH_GAUSSIAN_C** : Menggunakan rata-rata tertimbang Gaussian dari area tetangga.
- **THRESH_BINARY_INV** : Mengubah intensitas piksel di atas ambang batas menjadi 0 dan di bawah menjadi 255 (membalikkan biner).

Tujuan :

Efektif dalam segmentasi objek pada citra dengan pencahayaan yang tidak merata, memisahkan objek sampah dari latar belakangnya.

e. Deteksi Kontur

Kontur objek (batas luar) ditemukan pada citra biner. Kontur terbesar dianggap sebagai objek sampah utama.

```
cv2.findContours(thresh, cv2.RETR_EXTERNAL,  
cv2.CHAIN_APPROX_SIMPLE)
```

- **cv2.RETR_EXTERNAL** : Mengambil hanya kontur luar.
- **cv2.CHAIN_APPROX_SIMPLE** : Mengompres segmen horizontal, vertikal, dan diagonal menjadi titik akhir saja.

Tujuan :

Mengidentifikasi batas-batas objek yang kemudian akan digunakan untuk menghitung fitur bentuk.

f. Hu Moments

Hu Moments adalah serangkaian tujuh nilai invarian yang dihitung dari momen citra. Invarian ini berarti nilai-nilai ini tidak berubah meskipun objek dirotasi, diskalakan, atau ditranslasikan.

```
cv2.HuMoments(moments).flatten()
```

- `cv2.moments(cnt)` : Menghitung momen spasial dan pusat dari kontur.
- `hu = -np.sign(hu) * np.log10(np.abs(hu) + 1e-7)` : Transformasi logaritmik diterapkan untuk membuat Hu Moments lebih *robust* terhadap perubahan skala dan lebih mudah digunakan dalam klasifikasi.

Tujuan :

Menyediakan deskripsi bentuk objek yang ringkas dan kuat, terlepas dari orientasi atau ukurannya dalam citra.

g. Implementasi dalam Klasifikasi

Fungsi ini dipanggil secara otomatis jika gambar terdeteksi Organik.

```
shape_feat_hu, shape_feat_opening_img, shape_feat_img_contour_drawn =  
extract_shape(processed_img)
```

Gambar 4.8 Implementasi dalam Klasifikasi

4.2.4 Ekstraksi Fitur Tekstur (Kategori Plastik)

```
def extract_texture(img):  
    """  
    Mengekstrak fitur tekstur menggunakan GLCM (contrast, dissimilarity, homogeneity, energy, correlation)  
    dan Local Binary Patterns (LBP).  
    """  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
  
    distances = [1, 2]  
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]  
    glcm = graycomatrix(gray, distances, angles, 256, symmetric=True, normed=True)  
  
    contrast = graycoprops(glcm, 'contrast').mean()  
    dissimilarity = graycoprops(glcm, 'dissimilarity').mean()  
    homogeneity = graycoprops(glcm, 'homogeneity').mean()  
    energy = graycoprops(glcm, 'energy').mean()  
    correlation = graycoprops(glcm, 'correlation').mean()  
    glcm_features = [contrast, dissimilarity, homogeneity, energy, correlation]  
  
    radius = 3  
    n_points = 8 * radius  
    lbp = local_binary_pattern(gray, n_points, radius, method='uniform')  
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))  
    hist = hist.astype("float")  
    hist /= (hist.sum() + 1e-7)  
  
    return np.array(glcm_features + hist.tolist()).flatten()
```

Gambar 4.9 Ekstreaksi Tekstur

Fitur tekstur digunakan untuk menggambarkan pola permukaan objek dalam citra, seperti kasar, halus, atau bertekstur. Dalam proyek ini, fitur tekstur diekstraksi menggunakan teknik GLCM (Gray-Level Co-occurrence Matrix) dan dihitung propertinya yaitu contrast.

a. Konversi ke Grayscale

Analisis tekstur (GLCM dan LBP) biasanya dilakukan pada citra skala abu-abu karena hanya berfokus pada variasi intensitas piksel, bukan warna.

b. Gray Level Co-occurrence Matrix (GLCM)

GLCM mengukur seberapa sering pasangan piksel dengan intensitas tertentu muncul pada jarak dan arah tertentu. Dari GLCM, diekstraksi lima properti statistik.

```
graycomatrix(gray, distances, angles, 256, symmetric=True,  
normed=True)
```

- `distances = [1, 2]`: Mengukur hubungan piksel pada jarak 1 dan 2 piksel.
- `angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]`: Mengukur hubungan pada 0° (horizontal), 45°, 90° (vertikal), dan 135°.

Properti GLCM : `graycoprops(glc, 'property')`

- Contrast: Mengukur perbedaan intensitas piksel tetangga. Semakin tinggi, semakin besar kontras teksturnya.
- Dissimilarity: Mirip dengan kontras, mengukur ketidaksamaan intensitas.
- Homogeneity: Mengukur kedekatan distribusi elemen GLCM ke diagonal. Semakin tinggi, semakin seragam teksturnya.
- Energy: Mengukur jumlah keseragaman dalam distribusi piksel. Semakin tinggi, semakin teratur teksturnya.
- Correlation: Mengukur linearitas hubungan piksel tetangga. Semakin tinggi, semakin kuat hubungan linear.

Tujuan :

Memberikan informasi tentang struktur spasial tekstur citra, seperti kekasaran, kehalusan, dan arah pola.

c. Local Binary Pattern (LBP)

LBP adalah operator deskriptor tekstur yang memberikan label pada setiap piksel dalam citra skala abu-abu dengan ambang batas lingkungannya dan memperlakukannya sebagai angka biner.

```
local_binary_pattern(gray, n_points, radius, method='uniform')
```

- `radius = 3`: Menentukan radius lingkungan piksel.
- `n_points = 8 * radius`: Menentukan jumlah titik sampel di sekitar piksel pusat (24 titik untuk radius 3).
- `method='uniform'`: Menggunakan pola LBP uniform, yang mengurangi jumlah pola dan lebih *robust*.

Histogram LBP akan Dihitung histogram dari nilai LBP yang dihasilkan, yang kemudian dinormalisasi.

Tujuan :

Menangkap pola tekstur lokal mikroskopis yang tidak tertangkap oleh GLCM. Ini sangat efektif untuk representasi tekstur yang efisien dan deskriptif.

d. Implementasi dalam Klasifikasi

Fungsi ini dipanggil secara otomatis jika gambar terdeteksi Plastik.

```
texture_feat = extract_texture(processed_img)
```

Gambar 4.10 Implementasi dalam Klasifikasi

4.2.5 Padding Fitur

Setelah ketiga jenis fitur diekstrak, panjang masing-masing vektor fitur dapat bervariasi (terutama jika ada masalah dalam deteksi kontur untuk fitur bentuk, atau perbedaan dalam jumlah bin LBP jika tidak diatur seragam). Agar model klasifikasi dapat menerima input yang konsisten, semua vektor fitur di-*padding* (ditambah nol) hingga mencapai panjang maksimum yang ditemukan dalam dataset pelatihan.

```
np.pad(feature, (0, max_len - len(feature)))
```

Tujuan :

Memastikan semua data yang dimasukkan ke model memiliki dimensi yang seragam, mencegah error dan memungkinkan model untuk melatih dan memprediksi dengan benar.

4.2.6 Kombinasi Fitur

Untuk eksplorasi lebih lanjut, ketiga vektor fitur (warna, tekstur, bentuk) dapat digabungkan menjadi satu vektor fitur yang lebih besar.

```
np.hstack((color_feat, texture_feat, shape_feat))
```

Tujuan :

Mengevaluasi apakah kombinasi informasi dari berbagai aspek visual (warna, tekstur, bentuk) dapat menghasilkan performa klasifikasi yang lebih baik dibandingkan penggunaan fitur tunggal.

4.2.7 Pembentukan Dataset

Fungsi `load_and_extract_features` bertanggung jawab untuk memuat citra dari direktori dataset, melakukan pra-pemrosesan, mengekstrak fitur, dan mengorganisir data menjadi format yang siap untuk pelatihan dan pengujian.

1. Iterasi Direktori

Fungsi ini mengiterasi melalui setiap subfolder (kategori sampah) dalam direktori **dataset/** (untuk training) dan **datauji/** (untuk testing).

2. Pengambilan Sampel (untuk Contoh)

Untuk data pelatihan, parameter `sample_per_class=150` digunakan untuk mengambil 150 gambar secara acak dari setiap kelas, menjaga keseimbangan kelas dan mengurangi waktu pelatihan jika dataset sangat besar. Untuk data pengujian, semua gambar digunakan (`sample_per_class=None`).

3. Proses per Gambar, Untuk setiap gambar yang dipilih:

- Gambar dibaca menggunakan `cv2.imread()`.
- Pra-pemrosesan dilakukan menggunakan `preprocess_image()`.
- Semua fitur (warna, tekstur, bentuk) diekstrak.
- Fitur di-*padding* sesuai panjang maksimum yang telah ditentukan.
- Vektor fitur dan label kelas disimpan.

4. Penyimpanan Panjang Fitur

Panjang maksimum dari setiap jenis fitur (warna, tekstur, bentuk) yang diekstrak dari dataset pelatihan disimpan ke `feature_lengths.pkl`. Ini penting agar data pengujian dan data prediksi memiliki panjang fitur yang konsisten.

5. Output

Fungsi mengembalikan kumpulan array NumPy untuk fitur warna, tekstur, bentuk, dan label yang sesuai untuk data pelatihan (X_train_color, X_train_texture, X_train_shape, y_train_labels)

```
X_train_color, X_train_texture, X_train_shape, y_train_labels, feature_lengths_map = load_and_extract_features(  
    training_dataset_path,  
    sample_per_class=150,  
    denoise=True,  
    enhance_contrast=True,  
    show_example_processing=True  
)
```

Gambar 4.11 Pembentuka Dataset hasil Pelatihan

dan pengujian (X_test_color, X_test_texture, X_test_shape, y_test_labels)

```
X_test_color, X_test_texture, X_test_shape, y_test_labels, _ = load_and_extract_features(  
    testing_dataset_path,  
    sample_per_class=None,  
    target_size=(100, 100),  
    denoise=True,  
    enhance_contrast=True,  
    feature_lengths_map=feature_lengths_map,  
    show_example_processing=True  
)
```

Gambar 4.12 Pembentuka Dataset hasil Testing

4.2.8 Pelatihan Evaluasi dan Model

Fungsi `run_models_for_feature_evaluation` mengelola proses pelatihan dan evaluasi model klasifikasi.

1. Standardisasi Fitur:

Sebelum pelatihan, setiap kumpulan fitur (warna, tekstur, bentuk, dan gabungan) distandardisasi menggunakan `StandardScaler`. Ini mengubah distribusi fitur sehingga memiliki rata-rata nol dan variansi satu.

Standardisasi sangat penting untuk algoritma seperti KNN dan SVM yang peka terhadap skala data, memastikan bahwa fitur dengan rentang nilai yang lebih besar tidak mendominasi proses pembelajaran.

```

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_data['train'])
X_test_scaled = scaler.transform(X_data['test'])

# Save scaler for later use in prediction
joblib.dump(scaler, f'scaler_{feature_type}.pkl')
scalers_to_save[feature_type] = scaler # Store scaler reference

current_feature_results = {}

```

Gambar 4.13 Standarisasi Fitur

`scaler.fit_transform()` pada data pelatihan dan `scaler.transform()` pada data pengujian. *Scaler* ini kemudian disimpan (`joblib.dump(scaler, f'scaler_{feature_type}.pkl')`) untuk digunakan kembali saat prediksi.

2. Model K-Nearest Neighbors (KNN)

```

print(f"Melatih model KNN untuk {feature_type}...")
param_grid_knn = {'n_neighbors': [3, 5, 7, 9, 11]}
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5, verbose=0, n_jobs=-1)
grid_knn.fit(X_train_scaled, y_train)
knn = grid_knn.best_estimator_
print(f"KNN Best parameters ({feature_type}): {grid_knn.best_params_}")

knn_pred = knn.predict(X_test_scaled)
knn_acc = accuracy_score(y_test, knn_pred)
print(f"Akurasi KNN ({feature_type}) pada data uji: {knn_acc:.4f} ({knn_acc * 100:.2f}%)")
knn_report = classification_report(y_test, knn_pred, output_dict=True, zero_division=0, labels=unique_labels)
print("Classification Report KNN (Data Uji):\n", classification_report(y_test, knn_pred, zero_division=0, labels=unique_labels))

current_feature_results['knn_acc'] = knn_acc
current_feature_results['knn_report'] = knn_report

joblib.dump(knn, f'model_knn_{feature_type}.pkl')

```

Gambar 4.14 Model KNN

Algoritma KNN dikalibrasi dan dilatih untuk setiap jenis fitur. Optimasi Hyperparameter, `GridSearchCV` digunakan untuk menemukan nilai `n_neighbors` terbaik dalam rentang `[3, 5, 7, 9, 11]` melalui *5-fold cross-validation*.

`GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5, n_jobs=-1).`

Model terbaik (`grid_knn.best_estimator_`) kemudian digunakan untuk prediksi. Model KNN terbaik disimpan ke `model_knn_{feature_type}.pkl`.

3. Model Support Vector Machine (SVM)

```
print("Melatih model SVM untuk {feature_type}...")
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf']
}
grid_svm = GridSearchCV(SVC(probability=True), param_grid_svm, cv=5, verbose=0, n_jobs=-1)
grid_svm.fit(X_train_scaled, y_train)
svm = grid_svm.best_estimator_
print(f"SVM Best parameters ({feature_type}): {grid_svm.best_params_}")

svm_pred = svm.predict(X_test_scaled)
svm_acc = accuracy_score(y_test, svm_pred)
print(f"Akurasi SVM ({feature_type}) pada data uji: {svm_acc:.4f} ({svm_acc * 100:.2f}%)")
svm_report = classification_report(y_test, svm_pred, output_dict=True, zero_division=0, labels=unique_labels)
print("Classification Report SVM (Data Uji):\n", classification_report(y_test, svm_pred, zero_division=0, labels=unique_labels))

current_feature_results['svm_acc'] = svm_acc
current_feature_results['svm_report'] = svm_report

joblib.dump(svm, f'model_svm_{feature_type}.pkl')
```

Gambar 4.15 Model SVM

Algoritma SVM dilatih untuk setiap jenis fitur. Optimasi Hyperparameter, `GridSearchCV` digunakan untuk menemukan kombinasi `C` (parameter regularisasi) dan `gamma` (koefisien kernel RBF) terbaik dalam rentang `[0.1, 1, 10, 100]` dan `[1, 0.1, 0.01, 0.001]` dengan `kernel='rbf'` melalui *5-fold cross-validation*.

`GridSearchCV(SVC(probability=True), param_grid_svm, cv=5, n_jobs=-1)`. Parameter `probability=True` penting agar model dapat mengestimasi probabilitas prediksi. Model terbaik (`grid_svm.best_estimator_`) kemudian digunakan untuk prediksi. Model SVM terbaik disimpan ke `model_svm_{feature_type}.pkl`.

4. Kombinasi Fitur:

Fitur warna, tekstur, dan bentuk digabungkan secara horizontal (`np.hstack`) untuk membentuk satu vektor fitur gabungan. Kedua model (KNN dan SVM) juga dilatih dan dievaluasi menggunakan fitur gabungan ini. Mengevaluasi apakah integrasi berbagai jenis fitur dapat meningkatkan performa klasifikasi secara keseluruhan.

5. Evaluasi Model:

Setelah model dilatih, performanya dievaluasi pada data pengujian (`X_test, y_test`).

- Metrik :
 - Akurasi:
`accuracy_score` untuk mengukur proporsi prediksi yang benar.
 - Classification Report

`classification_report` untuk detail presisi, *recall*, dan F1-score untuk setiap kelas, serta rata-rata makro (rata-rata unweighted F1-score).

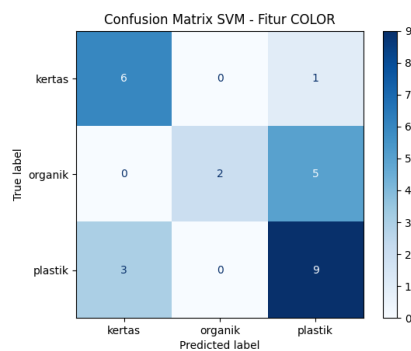
- Confusion Matrix

`confusion_matrix` dan `ConfusionMatrixDisplay.plot` untuk visualisasi performa klasifikasi per kelas, menunjukkan True Positives, False Positives, dll.

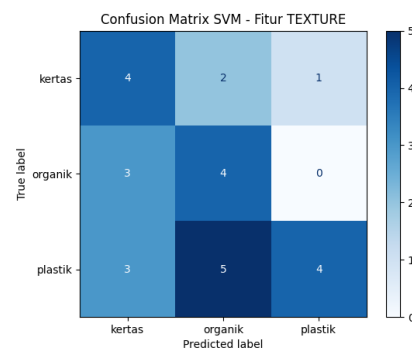
- Visualisasi

Confusion Matrix ditampilkan untuk setiap model SVM pada setiap jenis fitur.

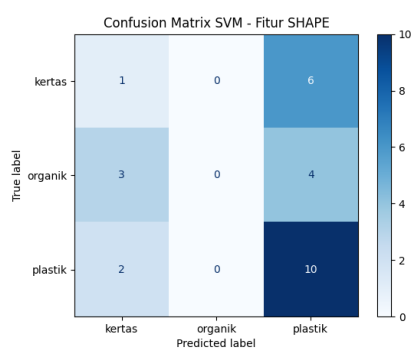
Hasil Evaluasi Model :



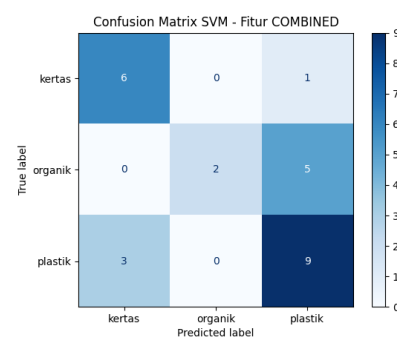
Gambar 4.16 Hasil SVM Warna



Gambar 4.17 Hasil SVM Tekstur



Gambar 4.18 Hasil SVM Bentuk



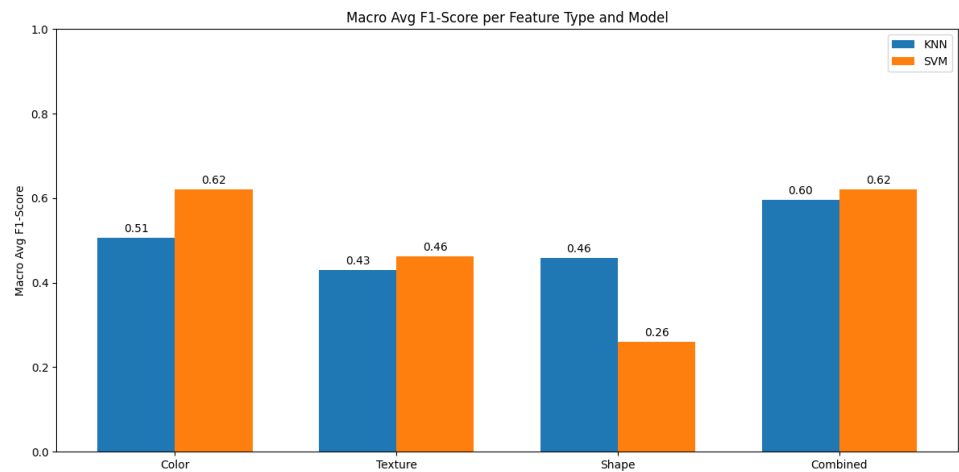
Gambar 4.19 Hasil SVM Kombinasi

Interpretasi:

- Baris menunjukkan label sebenarnya (True label)
- Kolom menunjukkan hasil prediksi model (Predicted label)
- Nilai diagonal utama menunjukkan jumlah prediksi yang benar
- Nilai di luar diagonal menunjukkan kesalahan klasifikasi.

6. Penentuan Fitur Terbaik per Kategori:

Setelah semua model dievaluasi, program menganalisis hasil *classification report* untuk mengidentifikasi kombinasi jenis fitur dan model (KNN atau SVM) yang memberikan F1-Score tertinggi untuk **setiap kategori sampah secara individual**.



Gambar 4.20 Hasil Keseluruhan

Informasi ini sangat krusial untuk fase prediksi, di mana sistem akan menggunakan model yang terbukti paling andal untuk kelas tertentu.

```

--- RINGKASAN PERFORMA FITUR PER KATEGORI ---

Untuk kategori: KERTAS
Fitur COLOR (SVM) - F1-Score: 0.7500
Fitur COLOR (KNN) - F1-Score: 0.5185
Fitur TEXTURE (SVM) - F1-Score: 0.4706
Fitur TEXTURE (KNN) - F1-Score: 0.5333
Fitur SHAPE (SVM) - F1-Score: 0.1538
Fitur SHAPE (KNN) - F1-Score: 0.5000
Fitur COMBINED (SVM) - F1-Score: 0.7500
Fitur COMBINED (KNN) - F1-Score: 0.5600
Kesimpulan: Fitur 'COLOR' dengan model 'SVM' memberikan F1-Score terbaik (0.7500) untuk 'KERTAS'

Untuk kategori: ORGANIK
Fitur COLOR (SVM) - F1-Score: 0.4444
Fitur COLOR (KNN) - F1-Score: 0.6000
Fitur TEXTURE (SVM) - F1-Score: 0.4444
Fitur TEXTURE (KNN) - F1-Score: 0.3333
Fitur SHAPE (SVM) - F1-Score: 0.0000
Fitur SHAPE (KNN) - F1-Score: 0.5000
Fitur COMBINED (SVM) - F1-Score: 0.4444
Fitur COMBINED (KNN) - F1-Score: 0.7273
Kesimpulan: Fitur 'COMBINED' dengan model 'KNN' memberikan F1-Score terbaik (0.7273) untuk 'ORGANIK'

Untuk kategori: PLASTIK
Fitur COLOR (SVM) - F1-Score: 0.6667
Fitur COLOR (KNN) - F1-Score: 0.4000
Fitur TEXTURE (SVM) - F1-Score: 0.4706
Fitur TEXTURE (KNN) - F1-Score: 0.4211
Fitur SHAPE (SVM) - F1-Score: 0.6250
Fitur SHAPE (KNN) - F1-Score: 0.3750
Fitur COMBINED (SVM) - F1-Score: 0.6667
Fitur COMBINED (KNN) - F1-Score: 0.5000
Kesimpulan: Fitur 'COLOR' dengan model 'SVM' memberikan F1-Score terbaik (0.6667) untuk 'PLASTIK'

```

Gambar 4.21 Hasi Perbandingan Keseluruhan

Ringkasan fitur terbaik per kategori ini disimpan ke `best_features_per_category.pkl`, untuk lebih detailnya ada pada lampiran.

4.3 Implementasi Tahap Prediksi (Main.py)

Tahap prediksi adalah aplikasi praktis dari model yang telah dilatih, di mana sistem mengklasifikasikan citra sampah baru yang belum pernah dilihat sebelumnya.

```

def predict_image_with_visuals(image_path,
                                target_size=(100, 100), # Konsisten dengan training
                                denoise=True,             # Konsisten dengan training
                                enhance_contrast=True      # Konsisten dengan training
                                ):

```

Gambar 4.22 Tahap Prediksi

Fungsi `predict_image_with_visuals` mengotomatiskan seluruh alur klasifikasi untuk sebuah citra tunggal yang dipilih oleh pengguna.

4.3.1 Pemilihan Citra Input oleh Pengguna

Melalui antarmuka grafis sederhana (`tkinter.filedialog.askopenfilename`), pengguna diminta untuk memilih file gambar dari sistem mereka. Memberikan fleksibilitas kepada pengguna untuk menguji gambar apa pun.

```
image_to_predict_path = filedialog.askopenfilename(  
    title="Pilih Gambar untuk Klasifikasi",  
    filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp *.tiff")]  
)
```

Gambar 4.23 Pemilihan Citra

`filedialog.askopenfilename()` memunculkan dialog standar untuk memilih file.

4.3.2 Pembacaan dan Pra-pemrosesan Citra Input

Citra yang dipilih pengguna dibaca (`cv2.imread`) dan kemudian di-*pipelining* melalui fungsi `preprocess_image` dengan parameter yang **sama persis** (`target_size=(100, 100)`, `denoise=True`, `enhance_contrast=True`) seperti saat pelatihan.

```
img = cv2.imread(image_path)  
if img is None:  
    print(f"❌ ERROR: Gagal membuka gambar: {image_path}")  
    return  
  
print(f"\n[INFO] Memproses gambar: {os.path.basename(image_path)}")  
  
# Lakukan pra-pemrosesan pada gambar input  
processed_img = preprocess_image(img, target_size, denoise, enhance_contrast)
```

Gambar 4.24 Pra-Pemrosesan Citra Input

Memastikan citra input memiliki format dan kualitas yang konsisten dengan data yang digunakan untuk melatih model, sehingga fitur yang diekstrak relevan dengan yang dikenali model.

4.3.3 Ekstraksi Fitur dari Citra Input

Setelah pra-pemrosesan, fitur warna, tekstur, dan bentuk diekstrak dari citra input menggunakan fungsi `extract_color`, `extract_texture`, dan `extract_shape`. Vektor fitur yang dihasilkan (`color_feat`, `texture_feat`, `shape_feat_hu`) merupakan representasi numerik dari citra.

```

color_feat = extract_color(processed_img)
texture_feat = extract_texture(processed_img)
shape_feat_hu, shape_feat_opening_img, shape_feat_img_contour_drawn = extract_shape(processed_img)

```

Gambar 4.25 Ekstraksi Input dan Citra Input

4.3.4 Pemuatan Aset yang Tersimpan

Program memuat semua aset yang telah disimpan dari fase pelatihan:

```

# Muat semua model dan scaler terpisah
loaded_models = {}
loaded_scalers = {}
feature_types = ['color', 'texture', 'shape']
class_labels = None
feature_lengths = None

try:
    for f_type in feature_types:
        loaded_models[f_type] = {
            'svm': joblib.load(f'model_svm_{f_type}.pkl'),
            'knn': joblib.load(f'model_knn_{f_type}.pkl')
        }
        loaded_scalers[f_type] = joblib.load(f'scaler_{f_type}.pkl')
        if class_labels is None:
            # Assuming all models have the same classes attribute
            class_labels = loaded_models[f_type]['svm'].classes_

    best_features_info = joblib.load('best_features_per_category.pkl')
    feature_lengths = joblib.load('feature_lengths.pkl') # Muat feature_lengths
except FileNotFoundError as e:
    print(f"✗ ERROR: File model/scaler/best_features_info/feature_lengths tidak ditemukan: {e}")
    print("Pastikan Anda sudah menjalankan script train_model.py terlebih dahulu untuk menghasilkan file-file ini.")
    return
except Exception as e:
    print(f"✗ ERROR: Gagal memuat model/scaler/info fitur: {e}")
    return

```

Gambar 4.26 Pemuatan Aset

- Model-model yang dilatih (model_svm_color.pkl, model_knn_texture.pkl, dst.).
- Scaler yang sesuai untuk setiap jenis fitur (scaler_color.pkl, dst.).
- Informasi panjang fitur (feature_lengths.pkl).
- Informasi model terbaik per kategori (best_features_per_category.pkl).

Memastikan bahwa transformasi data dan model yang digunakan untuk prediksi adalah versi yang sama dan terlatih dengan baik dari fase pelatihan.

4.3.5 Standardisasi dan Prediksi Individual

Setiap vektor fitur yang diekstrak (warna, tekstur, bentuk) di-*padding* sesuai dengan panjang fitur yang tersimpan (*feature_lengths.pkl*), kemudian distandardisasi menggunakan *scaler* yang relevan.

```
print("\n--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---")
for pred_label, proba_val, model_type, f_type, _ in all_predictions_with_proba:
    model = loaded_models[f_type][model_type]
    # Re-scale feature data to get full probability array for printing
    # feat_data needs to be correct for each feature type
    if f_type == 'color':
        current_feat_data_for_proba = color_feat_padded
    elif f_type == 'texture':
        current_feat_data_for_proba = texture_feat_padded
    else: # 'shape'
        current_feat_data_for_proba = shape_feat_padded
    full_proba = model.predict_proba([loaded_scalers[f_type].transform([current_feat_data_for_proba])[0]])[0]
    print(f" Model {model_type.upper()} {f_type.upper()}: {pred_label.upper()} (Prob: {get_proba_string(full_proba, class_labels)})")
```

Gambar 4.27 Standarisasi dan Prediksi

Setiap model yang dimuat (KNN dan SVM untuk setiap jenis fitur) kemudian digunakan untuk melakukan prediksi pada fitur yang telah diskalakan tersebut. Probabilitas prediksi (*predict_proba()*) juga diambil untuk memberikan tingkat keyakinan. Mendapatkan serangkaian prediksi dari berbagai model, masing-masing dengan tingkat keyakinannya.

4.3.6 Strategi Keputusan Final (Ensemble)

Program menentukan prediksi kelas akhir untuk citra input berdasarkan strategi *ensemble* yang menggunakan informasi dari *best_features_per_category.pkl*.

Mekanisme:

1. Semua label unik yang diprediksi oleh model individual dikumpulkan sebagai kandidat.
2. Untuk setiap *candidate_label*, program mencari entri yang sesuai di *best_features_per_category.pkl* untuk menemukan model (jenis fitur dan algoritma) yang memiliki F1-Score pelatihan tertinggi untuk kelas tersebut.
3. Label yang prediksi terbaiknya berasal dari model dengan F1-Score pelatihan tertinggi akan dipilih sebagai prediksi final.

4. Jika `best_features_per_category.pkl` tidak valid atau tidak ada kandidat yang cocok, sistem akan *fallback* ke *majority voting* dari semua prediksi individual.

Mengambil keputusan klasifikasi yang paling *robust* dan akurat dengan memanfaatkan pengetahuan tentang kekuatan model untuk setiap kelas yang diperoleh selama pelatihan. Gambar kode dilampirkan.

4.3.7 Tampilan Hasil dan Visualisasi Fitur:

Hasil prediksi final ditampilkan kepada pengguna. Sebagai tambahan, visualisasi fitur yang paling relevan dengan kelas yang diprediksi akan ditampilkan untuk memberikan *insight* visual tentang mengapa model membuat keputusan tersebut.

Implementasi:

- Jika prediksi final adalah 'kertas', `visualize_color_features` akan dipanggil untuk menampilkan citra dan histogram HSV.
- Jika prediksi final adalah 'plastik', `visualize_texture_features` akan dipanggil untuk menampilkan citra, *grayscale*, dan bar chart fitur GLCM.
- Jika prediksi final adalah 'organik', `visualize_shape_features` akan dipanggil untuk menampilkan alur segmentasi, kontur terdeteksi, dan nilai Hu Moments.

Menyediakan umpan balik yang intuitif dan membantu pengguna memahami dasar keputusan model

4.4 Hasil Pengujian

Sebagai studi lanjutan, sistem diuji menggunakan dua algoritma klasifikasi sederhana yaitu K-Nearest Neighbors (KNN) dan Support Vector Machine (SVM). Pengujian bertujuan untuk mengetahui seberapa efektif fitur yang telah diekstraksi dalam membedakan tiga kategori sampah.

4.4.1 Hasil Training Model

Dari Hasil training dan testing, kami mendapatkan kesimpulan hasil model sebagai berikut :

Tabel 4.2 Akurasi Model Klasifikasi

Kategori	Fitur	Model	F1-Score
KERTAS	COLOR	SVM	0.7500
	COLOR	KNN	0.5185
	TEXTURE	SVM	0.4706
	TEXTURE	KNN	0.5333
	SHAPE	SVM	0.1538
	SHAPE	KNN	0.5000
	COMBINED	SVM	0.7500
	COMBINED	KNN	0.5600
Kesimpulan	COLOR	SVM	0.7500
ORGANIK	COLOR	SVM	0.4444
	COLOR	KNN	0.6000
	TEXTURE	SVM	0.4444
	TEXTURE	KNN	0.3333
	SHAPE	SVM	0.0000
	SHAPE	KNN	0.5000
	COMBINED	SVM	0.4444
Kesimpulan	COMBINED	KNN	0.7273
PLASTIK	COLOR	SVM	0.6667
	COLOR	KNN	0.4000
	TEXTURE	SVM	0.4706
	TEXTURE	KNN	0.4211
	SHAPE	SVM	0.6250
	SHAPE	KNN	0.5000
Kesimpulan	COMBINED	SVM	0.6667

Berdasarkan hasil evaluasi performa fitur per kategori, model SVM menunjukkan performa terbaik pada kategori *KERTAS* dan *PLASTIK* dengan F1-Score tertinggi masing-masing sebesar 0.7500 dan 0.6667 ketika menggunakan fitur COLOR. Sementara itu, pada kategori *ORGANIK*, model KNN dengan fitur COMBINED memberikan F1-Score tertinggi

sebesar 0.7273, menunjukkan bahwa kombinasi fitur (COLOR, TEXTURE, dan SHAPE) memberikan kontribusi signifikan untuk klasifikasi sampah organik. Secara umum, fitur COLOR merupakan fitur yang paling konsisten memberikan hasil terbaik, terutama ketika dipadukan dengan model SVM, sehingga dapat diprioritaskan dalam pengembangan sistem klasifikasi awal.

4.4.2 Hasil Klasifikasi

Pengujian dilakukan dengan menjalankan file `main_klasifikasi.py` dan memasukkan citra uji dari folder `test/` sebagai input. Berdasarkan nama file, sistem secara otomatis memilih metode ekstraksi fitur yang sesuai dan melakukan klasifikasi menggunakan model KNN dan SVM.

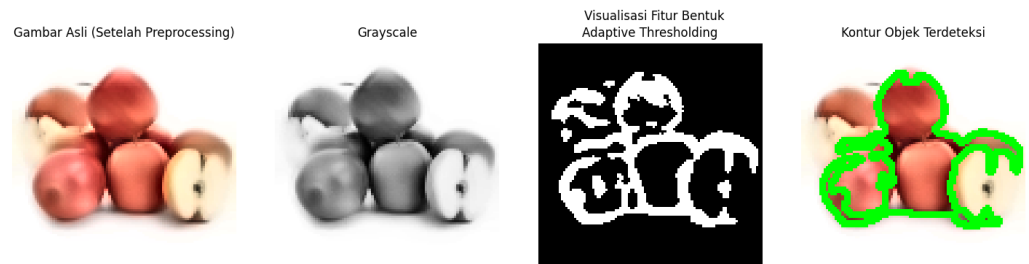
Sebagai contoh, citra `plastik.jpg` digunakan dalam pengujian. Berdasarkan model yang sudah ditrainning, ketika model mengenal gambar tersebut plastik sistem menerapkan ekstraksi tekstur menggunakan GLCM dan memprediksi hasil klasifikasinya.

1. Organik

- Percobaan **Pertama** dari gambar **baru**



Gambar 4.28 Gambar Asli 1



Gambar 4.29 Hasil Pengujian Bentuk 1

Dengan hasil, Gambar berhasil di klasifikasi sebagai organik.

```
[INFO] Memproses gambar: apel.jpeg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: ORGANIK (Prob: KERTAS: 20.35%, ORGANIK: 43.26%, PLASTIK: 36.39%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 40.00%, ORGANIK: 20.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 23.50%, ORGANIK: 66.38%, PLASTIK: 10.13%)
Model KNN TEXTURE: ORGANIK (Prob: KERTAS: 0.00%, ORGANIK: 80.00%, PLASTIK: 20.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 32.85%, ORGANIK: 26.88%, PLASTIK: 40.28%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 42.88%, ORGANIK: 42.88%, PLASTIK: 14.28%)
Tidak ada kandidat prediksi yang cocok dengan best_features_info atau best_features_info kosong.
Menggunakan Voting Mayoritas sebagai fallback akhir.

--- HASIL PREDIKSI FINAL: ORGANIK ---
Dipilih berdasarkan: Model Majority (Fallback) Fitur Voting
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Hu Moments:
Hu[1] = 5.795310e-01
Hu[2] = 3.516714e+00
Hu[3] = 2.426228e+00
Hu[4] = 3.711791e+00
Hu[5] = -6.832250e+00
Hu[6] = 6.582850e+00
Hu[7] = 6.587260e+00

Gambar: apel.jpeg
Prediksi Model Final: ORGANIK
```

Gambar 4.30 Hasil Pengujian Bentuk Statik 1

- Percobaan **Kedua** dengan gambar **baru** lain-nya



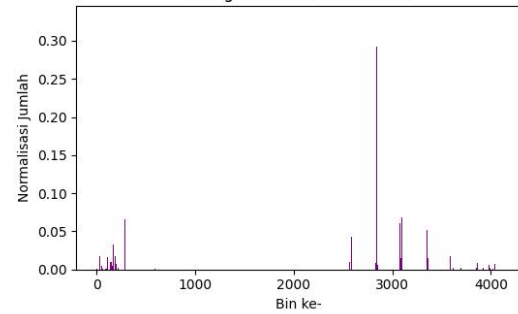
Gambar 4.31 Gambar Asli 2

Visualisasi Fitur Warna

Gambar Asli (Setelah Preprocessing)



Histogram HSV (Fitur Warna)



Gambar 4.32 Hasil Pengujian Bentuk 2

Dengan hasil, gambar gagal diklasifikasikan sebagai Organik.

```
[INFO] Memproses gambar: 18.jpg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: ORGANIK (Prob: KERTAS: 5.00%, ORGANIK: 89.10%, PLASTIK: 5.91%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 60.00%, ORGANIK: 20.00%, PLASTIK: 20.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 38.14%, ORGANIK: 44.79%, PLASTIK: 17.07%)
Model KNN TEXTURE: KERTAS (Prob: KERTAS: 60.00%, ORGANIK: 40.00%, PLASTIK: 0.00%)
Model SVM SHAPE: PLASTIK (Prob: KERTAS: 37.30%, ORGANIK: 31.53%, PLASTIK: 31.17%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 42.86%, ORGANIK: 28.57%, PLASTIK: 28.57%)
Tidak ada kandidat prediksi yang cocok dengan best_features_info atau best_features_info kosong.
Menggunakan Voting Mayoritas sebagai fallback akhir.

--- HASIL PREDIKSI FINAL: KERTAS ---
Dipilih berdasarkan: Model Majority (Fallback) Fitur Voting
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur warna (HSV): 4096

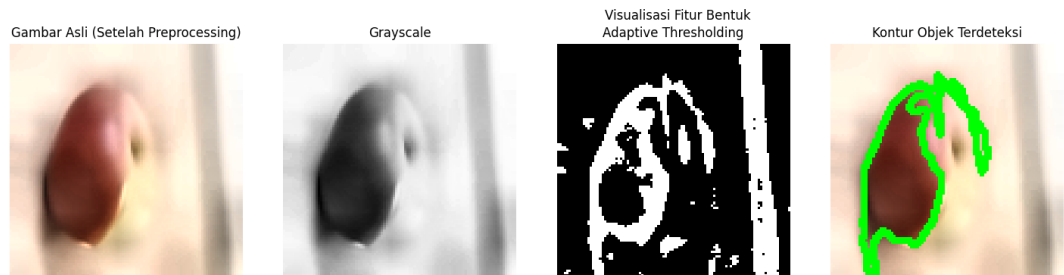
Gambar: 18.jpg
Prediksi Model Final: KERTAS
```

Gambar 4.33 Hasil Pengujian Bentuk Statik 2

- Percobaan **Ketiga** dengan gambar yang sudah di **training**



Gambar 4.34 Gambar Asli 3



Gambar 4.35 Hasil Pengujian Bentuk 3

Dengan hasil, gambar berhasil diklasifikasikan sebagai Organik.

```
[INFO] Memproses gambar: organik1.png

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: ORGANIK (Prob: KERTAS: 20.78%, ORGANIK: 43.09%, PLASTIK: 36.14%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 40.00%, ORGANIK: 20.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 21.35%, ORGANIK: 71.10%, PLASTIK: 7.55%)
Model KNN TEXTURE: ORGANIK (Prob: KERTAS: 20.00%, ORGANIK: 80.00%, PLASTIK: 0.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 31.88%, ORGANIK: 26.34%, PLASTIK: 41.78%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 57.14%, ORGANIK: 28.57%, PLASTIK: 14.29%)
Tidak ada kandidat prediksi yang cocok dengan best_features_info atau best_features_info kosong.
Menggunakan Voting Mayoritas sebagai fallback akhir.

--- HASIL PREDIKSI FINAL: ORGANIK ---
Dipilih berdasarkan: Model Majority (Fallback) Fitur Voting
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Hu Moments:
Hu[1] = 5.661027e-01
Hu[2] = 1.623675e+00
Hu[3] = 4.156924e+00
Hu[4] = 3.076435e+00
Hu[5] = 6.990766e+00
Hu[6] = 4.125762e+00
Hu[7] = 6.519243e+00

Gambar: organik1.png
Prediksi Model Final: ORGANIK
```

Gambar 4.36 Hasil Pengujian Bentuk Statik 3

2. Plastik

- Percobaan **Pertama** dengan gambar **baru**



Gambar 4.37 Gambar Asli 4



Gambar 4.38 Hasil Pengujian Tekstur 1

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Plastik.

```
[INFO] Memproses gambar: plastiks.jpeg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: PLASTIK (Prob: KERTAS: 19.33%, ORGANIK: 39.70%, PLASTIK: 40.97%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 40.00%, ORGANIK: 20.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 37.03%, ORGANIK: 40.81%, PLASTIK: 22.16%)
Model KNN TEXTURE: KERTAS (Prob: KERTAS: 80.00%, ORGANIK: 0.00%, PLASTIK: 20.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 32.37%, ORGANIK: 27.31%, PLASTIK: 40.31%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 57.14%, ORGANIK: 42.86%, PLASTIK: 0.00%)

--- HASIL PREDIKSI FINAL: PLASTIK ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.6667
Probabilitas Prediksi dari model ini: 40.97%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur tekstur (GLCM + LBP): 31

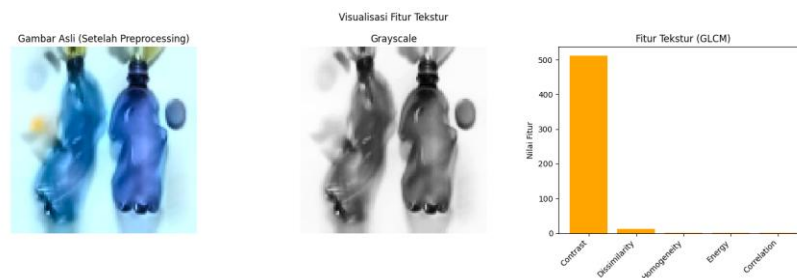
Gambar: plastiks.jpeg
Prediksi Model Final: PLASTIK
```

Gambar 4.39 Hasil Pengujian Tekstu Statik 1

- Percobaan **Kedua** dengan gambar **baru**



Gambar 4.40 Gambar Asli 5



Gambar 4.41 Hasil Pengujian Tekstur 2

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Plastik.

```
[INFO] Memproses gambar: Screenshot 2025-06-14 201822.png

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: PLASTIK (Prob: KERTAS: 19.33%, ORGANIK: 39.70%, PLASTIK: 40.97%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 80.00%, ORGANIK: 20.00%, PLASTIK: 0.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 36.60%, ORGANIK: 47.93%, PLASTIK: 15.47%)
Model KNN TEXTURE: ORGANIK (Prob: KERTAS: 20.00%, ORGANIK: 40.00%, PLASTIK: 40.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 34.38%, ORGANIK: 28.21%, PLASTIK: 37.41%)
Model KNN SHAPE: ORGANIK (Prob: KERTAS: 28.57%, ORGANIK: 42.86%, PLASTIK: 28.57%)

--- HASIL PREDIKSI FINAL: PLASTIK ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.6667
Probabilitas Prediksi dari model ini: 40.97%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur tekstur (GLCM + LBP): 31

Gambar: Screenshot 2025-06-14 201822.png
Prediksi Model Final: PLASTIK
```

Gambar 4.42 Hasil Pengujian Tekstu Statik 2

- Percobaan **Ktiga** dengan gambar **testing**



Gambar 4.43 Gambar Asli 6



Gambar 4.44 Hasil Pengujian Tekstur 3

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Plastik.

```
[INFO] Memproses gambar: 17.jpg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: PLASTIK (Prob: KERTAS: 9.92%, ORGANIK: 7.97%, PLASTIK: 82.10%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 60.00%, ORGANIK: 0.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: PLASTIK (Prob: KERTAS: 24.45%, ORGANIK: 13.07%, PLASTIK: 62.49%)
Model KNN TEXTURE: PLASTIK (Prob: KERTAS: 0.00%, ORGANIK: 20.00%, PLASTIK: 80.00%)
Model SVM SHAPE: PLASTIK (Prob: KERTAS: 35.82%, ORGANIK: 29.15%, PLASTIK: 35.03%)
Model KNN SHAPE: PLASTIK (Prob: KERTAS: 28.57%, ORGANIK: 0.00%, PLASTIK: 71.43%)

--- HASIL PREDIKSI FINAL: PLASTIK ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.6667
Probabilitas Prediksi dari model ini: 82.10%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur tekstur (GLCM + LBP): 31

Gambar: 17.jpg
Prediksi Model Final: PLASTIK
```

Gambar 4.45 Hasil Pengujian Tekstu Statik 3

3. Kertas

- Percobaan **Pertama** dengan gambar **baru**



Gambar 4.46 Gambar Asli 7



Gambar 4.45 Hasil Pengujian Warna 1

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Kertas.

```
[INFO] Memproses gambar: plastiks.jpeg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: PLASTIK (Prob: KERTAS: 19.33%, ORGANIK: 39.70%, PLASTIK: 40.97%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 40.00%, ORGANIK: 20.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: ORGANIK (Prob: KERTAS: 37.03%, ORGANIK: 40.81%, PLASTIK: 22.16%)
Model KNN TEXTURE: KERTAS (Prob: KERTAS: 80.00%, ORGANIK: 0.00%, PLASTIK: 20.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 32.37%, ORGANIK: 27.31%, PLASTIK: 40.31%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 57.14%, ORGANIK: 42.86%, PLASTIK: 0.00%)

--- HASIL PREDIKSI FINAL: PLASTIK ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.6667
Probabilitas Prediksi dari model ini: 40.97%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur tekstur (GLCM + LBP): 31

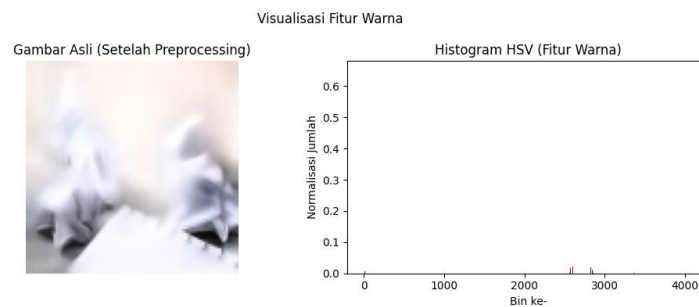
Gambar: plastiks.jpeg
Prediksi Model Final: PLASTIK
```

Gambar 4.46 Hasil Pengujian Warna Statik 1

- Percobaan **Kedua** dengan gambar **baru**



Gambar 4.47 Gambar Asli 8



Gambar 4.48 Hasil Pengujian Warna 2

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Kertas.

```
[INFO] Memproses gambar: OIP (3).jpeg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: KERTAS (Prob: KERTAS: 78.65%, ORGANIK: 5.84%, PLASTIK: 15.51%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 60.00%, ORGANIK: 0.00%, PLASTIK: 40.00%)
Model SVM TEXTURE: KERTAS (Prob: KERTAS: 52.74%, ORGANIK: 30.01%, PLASTIK: 17.26%)
Model KNN TEXTURE: KERTAS (Prob: KERTAS: 80.00%, ORGANIK: 0.00%, PLASTIK: 20.00%)
Model SVM SHAPE: PLASTIK (Prob: KERTAS: 37.80%, ORGANIK: 31.83%, PLASTIK: 30.37%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 57.14%, ORGANIK: 0.00%, PLASTIK: 42.86%)

--- HASIL PREDIKSI FINAL: KERTAS ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.7500
Probabilitas Prediksi dari model ini: 78.65%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur warna (HSV): 4096

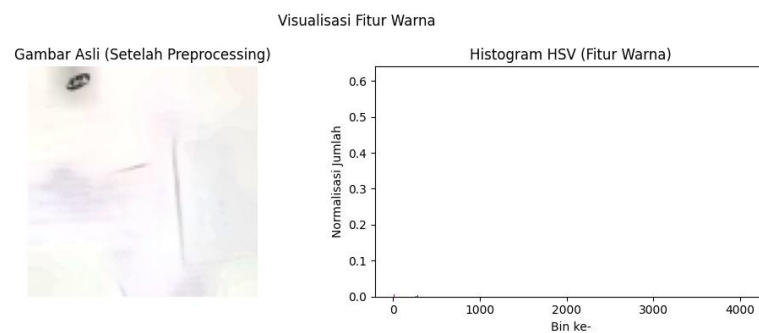
Gambar: OIP (3).jpeg
Prediksi Model Final: KERTAS
```

Gambar 4.49 Hasil Pengujian Warna Statik 2

- Percobaan **Ketiga** dengan gambar **Trainning**



Gambar 4.50 Gambar Asli 9



Gambar 4.51 Hasil Pengujian Warna 3

Dengan hasil, Gambar berhasil diklasifikasikan sebagai Kertas.

```
[INFO] Memproses gambar: kertas.jpg

--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---
Model SVM COLOR: KERTAS (Prob: KERTAS: 80.78%, ORGANIK: 4.55%, PLASTIK: 14.67%)
Model KNN COLOR: KERTAS (Prob: KERTAS: 80.00%, ORGANIK: 0.00%, PLASTIK: 20.00%)
Model SVM TEXTURE: KERTAS (Prob: KERTAS: 62.12%, ORGANIK: 14.58%, PLASTIK: 23.30%)
Model KNN TEXTURE: KERTAS (Prob: KERTAS: 60.00%, ORGANIK: 20.00%, PLASTIK: 20.00%)
Model SVM SHAPE: KERTAS (Prob: KERTAS: 35.99%, ORGANIK: 30.28%, PLASTIK: 33.73%)
Model KNN SHAPE: KERTAS (Prob: KERTAS: 57.14%, ORGANIK: 0.00%, PLASTIK: 42.86%)

--- HASIL PREDIKSI FINAL: KERTAS ---
Dipilih berdasarkan: Model SVM Fitur color
F1-Score saat Training untuk model ini: 0.7500
Probabilitas Prediksi dari model ini: 80.78%
[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:
[INFO] Panjang fitur warna (HSV): 4096

Gambar: kertas.jpg
Prediksi Model Final: KERTAS
```

Gambar 4.52 Hasil Pengujian Warna Statik 3

Kesimpulan :

Sistem klasifikasi gambar ini dirancang untuk mengidentifikasi objek berdasarkan kategori KERTAS, ORGANIK, dan PLASTIK. Dengan memanfaatkan algoritma Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN) yang dilatih menggunakan fitur-fitur seperti warna, tekstur, dan bentuk, program ini menunjukkan kinerja yang solid dalam sebagian besar kasus. Hal ini terbukti dari keberhasilan akuratnya dalam mengklasifikasikan citra "apel.jpeg" dan "organik1.png" sebagai ORGANIK, serta berbagai citra plastik dan kertas yang teridentifikasi dengan tepat. Namun, sebuah anomali terdeteksi pada citra "18.jpg"; meskipun seharusnya dikategorikan sebagai ORGANIK, program justru mengklasifikasikannya sebagai KERTAS. Deviasi ini kemungkinan besar berasal dari inkonsistensi dalam hasil prediksi antar model individual – di mana, misalnya, SVM berdasarkan warna sangat mengarah ke ORGANIK, sementara model KNN cenderung ke KERTAS – yang pada akhirnya memengaruhi keputusan akhir metode voting mayoritas, menyebabkan kesalahan klasifikasi pada kasus tersebut.

BAB V

PENUTUP

5.1 Kesimpulan

Rancangan dan implementasi program untuk ekstraksi fitur warna, bentuk, dan tekstur dari citra sampah RGB telah berhasil ditunjukkan melalui serangkaian proses komputasi yang terstruktur. Proses ini secara efektif memanfaatkan berbagai *library* Python, termasuk OpenCV untuk pengolahan citra dasar seperti pembacaan dan konversi ruang warna (misalnya, RGB ke HSV untuk fitur warna), scikit-image untuk ekstraksi fitur tekstur (misalnya, GLCM dan LBP), Pillow untuk manipulasi citra, NumPy untuk operasi array numerik yang efisien, serta mahotas untuk ekstraksi fitur bentuk seperti Hu Moments. Integrasi *library* ini memungkinkan program untuk mengekstrak representasi numerik yang komprehensif dari karakteristik visual setiap citra sampah.

Selanjutnya, fitur-fitur yang diekstraksi menjadi masukan penting bagi model klasifikasi *machine learning*. Evaluasi performa model Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN) dalam mengklasifikasikan jenis sampah (Kertas, Organik, Plastik) menunjukkan kemampuan sistem untuk melakukan prediksi. Meskipun sebagian besar klasifikasi berjalan dengan baik, seperti yang terlihat pada prediksi akurat citra apel dan organik1.png sebagai ORGANIK, serta berbagai citra plastik dan kertas, adanya kasus anomali seperti pada citra "18.jpg" yang diprediksi salah dari seharusnya ORGANIK menjadi KERTAS, mengindikasikan kompleksitas dalam pemisahan fitur dan kebutuhan akan optimasi lebih lanjut. Perbedaan prediksi antar model individu dan mekanisme *voting* mayoritas sebagai *fallback* menunjukkan bahwa setiap model memiliki kekuatan pada fitur tertentu (misalnya, SVM pada warna) namun mungkin rentan terhadap ambiguitas fitur pada citra tertentu, yang menyoroti pentingnya kalibrasi model dan strategi penggabungan keputusan.

5.2 Saran

Berdasarkan hasil dan observasi yang telah dicapai, beberapa saran dapat diajukan untuk peningkatan lebih lanjut:

1. Optimasi Ekstraksi Fitur

- **Eksplorasi Fitur Lanjutan**

Pertimbangkan untuk menambahkan fitur lain seperti Gabor filters untuk tekstur yang lebih detail, atau *moments* geometris lainnya untuk bentuk yang lebih kompleks. Uji kombinasi fitur yang berbeda untuk menemukan set fitur yang paling diskriminatif.

- **Normalisasi dan Skala Fitur**

Pastikan semua fitur dinormalisasi atau diskalakan dengan benar sebelum dimasukkan ke model klasifikasi. Ini penting untuk performa SVM dan KNN yang sensitif terhadap skala fitur.

- **Pemilihan Fitur (Feature Selection)**

Gunakan teknik pemilihan fitur seperti RFE (Recursive Feature Elimination), SelectKBest, atau Principal Component Analysis (PCA) untuk mengurangi dimensi fitur dan menghilangkan fitur yang redundan atau tidak relevan, yang dapat meningkatkan akurasi dan efisiensi komputasi.

2. Peningkatan Model Klasifikasi

- **Tuning Hiperparameter**

Lakukan *hyperparameter tuning* yang lebih ekstensif untuk model SVM dan KNN menggunakan teknik seperti *Grid Search* atau *Random Search* dengan *Cross-Validation*. Hiperparameter seperti C dan gamma untuk SVM, serta n_neighbors dan weights untuk KNN, sangat memengaruhi performa model.

- **Evaluasi Model Lebih Mendalam**

Selain probabilitas dan F1-Score, evaluasi model menggunakan metrik lain seperti *Precision*, *Recall*, *Confusion Matrix*, dan *ROC AUC* untuk

mendapatkan pemahaman yang lebih komprehensif tentang performa masing-masing kelas.

- **Ensemble Learning**

Eksplorasi penggunaan metode *ensemble learning* seperti *Random Forest*, *Gradient Boosting Machines* (misalnya XGBoost, LightGBM), atau *Stacking* yang dapat mengkombinasikan kekuatan beberapa model untuk hasil prediksi yang lebih robust dan akurat, terutama dalam menangani ambiguitas fitur.

- **Dataset Training yang Diperkaya**

Untuk mengatasi kasus gagal deteksi seperti "18.jpg", perluasan dataset training dengan variasi citra sampah yang lebih luas dan representatif, termasuk citra dengan kondisi pencahayaan atau latar belakang yang berbeda, dapat sangat membantu model untuk belajar pola yang lebih robust.

3. Mekanisme Pengambilan Keputusan

- **Strategi Voting yang Ditingkatkan**

Alih-alih voting mayoritas sederhana, implementasikan *weighted voting* di mana model yang memiliki performa lebih tinggi (berdasarkan F1-score atau metrik lain dari fase training) diberi bobot lebih besar dalam keputusan akhir.

- **Analisis Kasus Batas (Edge Cases)**

Lakukan analisis mendalam terhadap citra-citra yang salah diklasifikasikan untuk mengidentifikasi penyebab spesifik kegagalan. Ini dapat mengarahkan pada penyesuaian fitur atau penambahan model khusus untuk kasus-kasus tersebut.

LAMPIRAN

A. Potongan Kode Program – (ekstraksi_klasifikasi.py)

```
1 import os
2 import cv2
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 from collections import Counter
7 from sklearn.model_selection import GridSearchCV
8 from sklearn.neighbors import KNeighborsClassifier
9 from sklearn.svm import SVC
10 from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
11 from skimage.feature import graycomatrix, graycoprops, local_binary_pattern
12 from sklearn.preprocessing import StandardScaler
13 import joblib # Make sure joblib is imported
14 import random
15
16 # ----- EKSTRAKSI FITUR -----
17
18 def preprocess_image(img, target_size=(100, 100), denoise=False, enhance_contrast=False, show_steps=False, title_suffix=""):
19     """
20     Melakukan pra-pemrosesan gambar: resizing, denoising (opsional), dan contrast enhancement (opsional).
21     """
22     original_img = img.copy()
23
24     # Determine number of subplots needed for visualization
25     num_subplots = 1 # Original Image
26     if show_steps:
27         num_subplots += 1 # Resized Image
28         if denoise:
29             num_subplots += 1 # Denoised Image
30         if enhance_contrast:
31             num_subplots += 1 # Contrast Enhanced Image
32
33     fig, axes = plt.subplots(1, num_subplots, figsize=(4 * num_subplots, 5))
34     current_plot_index = 0
35
36     # Plot Original Image
37     axes[current_plot_index].imshow(cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB))
38     axes[current_plot_index].set_title(f"Original Image {title_suffix}")
39     axes[current_plot_index].axis('off')
40     current_plot_index += 1
41
42     # Plot Resized Image
43     img_resized = cv2.resize(img, target_size, interpolation=cv2.INTER_AREA)
44     axes[current_plot_index].imshow(cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB))
45     axes[current_plot_index].set_title(f"Resized Image {title_suffix}")
46     axes[current_plot_index].axis('off')
47     current_plot_index += 1
48
49     # Denoising
50     if denoise:
51         img_to_denoise = img_resized.copy() # Denoise the resized image
52         denoised_img = cv2.fastNlMeansDenoisingColored(img_to_denoise, None, 10, 10, 7, 21)
53         axes[current_plot_index].imshow(cv2.cvtColor(denoised_img, cv2.COLOR_BGR2RGB))
54         axes[current_plot_index].set_title(f"Denoised Image {title_suffix}")
55         axes[current_plot_index].axis('off')
56         img_resized = denoised_img # Update img_resized for the next step
57         current_plot_index += 1
58
59     # Contrast Enhancement (CLAHE)
60     if enhance_contrast:
61         img_to_enhance = img_resized.copy() # Enhance the (possibly) denoised image
62         lab = cv2.cvtColor(img_to_enhance, cv2.COLOR_BGR2LAB)
63         l_channel = lab[:, :, 0]
64         clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
65         cl = clahe.apply(l_channel)
66         lab[:, :, 0] = cl
67         contrast_enhanced_img = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
68         axes[current_plot_index].imshow(cv2.cvtColor(contrast_enhanced_img, cv2.COLOR_BGR2RGB))
69         axes[current_plot_index].set_title(f"Contrast Enhanced {title_suffix}")
70         axes[current_plot_index].axis('off')
71         img_resized = contrast_enhanced_img # Final processed image
72         current_plot_index += 1
73
74     plt.tight_layout()
75     plt.show()
76     plt.close(fig) # Close the figure to manage memory
77
78 else:
79     # Perform processing without showing steps
80     img_resized = cv2.resize(img, target_size, interpolation=cv2.INTER_AREA)
81     if denoise:
82         img_resized = cv2.fastNlMeansDenoisingColored(img_resized, None, 10, 10, 7, 21)
83     if enhance_contrast:
84         lab = cv2.cvtColor(img_resized, cv2.COLOR_BGR2LAB)
85         l_channel = lab[:, :, 0]
86         clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
```

Gambar A.1 Kode Program ekstraksi_klasifikasi.py


```

85         c1 = cv2.cvtColor(c1, cv2.COLOR_BGR2LAB)
86         c1 = clahe.apply(c1)
87         lab[:, :, 0] = c1
88         img_resized = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
89
90     return img_resized
91
92
93 def extract_color(img):
94     """
95     Mengekstrak fitur warna menggunakan histogram HSV.
96     """
97     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
98     hist = cv2.calcHist([hsv], [0, 1, 2], None, [16, 16, 16], [0, 180, 0, 256, 0, 256])
99     cv2.normalize(hist, hist)
100     return hist.flatten()
101
102 def extract_texture(img):
103     """
104     Mengekstrak fitur tekstur menggunakan GLCM (contrast, dissimilarity, homogeneity, energy, correlation)
105     dan Local Binary Patterns (LBP).
106     """
107     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
108
109     distances = [1, 2]
110     angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
111     glcm = graycomatrix(gray, distances, angles, 256, symmetric=True, normed=True)
112
113     contrast = graycoprops(glcm, 'contrast').mean()
114     dissimilarity = graycoprops(glcm, 'dissimilarity').mean()
115     homogeneity = graycoprops(glcm, 'homogeneity').mean()
116     energy = graycoprops(glcm, 'energy').mean()
117     correlation = graycoprops(glcm, 'correlation').mean()
118     glcm_features = [contrast, dissimilarity, homogeneity, energy, correlation]
119
120     radius = 3
121     n_points = 8 * radius
122     lbp = local_binary_pattern(gray, n_points, radius, method='uniform')
123     (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
124     hist = hist.astype("float")
125     hist /= (hist.sum() + 1e-7)
126
127     return np.array(glcm_features + hist.tolist()).flatten()
128
129 def extract_shape(img):
130     """
131     Mengekstrak fitur bentuk menggunakan Hu Moments setelah segmentation.
132     """
133     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
134     blur = cv2.GaussianBlur(gray, (5, 5), 0)
135     thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
136     | | | | | | | | cv2.THRESH_BINARY_INV, 11, 2)
137     kernel = np.ones((3, 3), np.uint8)
138     opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=3)
139
140     contours, _ = cv2.findContours(opening, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
141     hu = np.zeros((7,), dtype=np.float32)
142
143     if contours:
144         cnt = max(contours, key=cv2.contourArea)
145         moments = cv2.moments(cnt)
146         if moments['m00'] != 0:
147             hu = cv2.HuMoments(moments).flatten()
148             hu = -np.sign(hu) * np.log10(np.abs(hu) + 1e-7)
149     return hu
150
151 # ----- PEMBUATAN DATASET (SAMPLING ACAK) -----
152
153 def load_and_extract_features(base_path, sample_per_class=None, target_size=(100,100), denoise=False, enhance_contrast=False,
154     | | | | | | | | feature_lengths_map=None, show_example_processing=False):
155     """
156     Memuat gambar dari dataset, melakukan pra-pemrosesan, dan mengekstrak SEMUA jenis fitur.
157     Jika feature_lengths_map disediakan, padding akan dilakukan sesuai panjang tersebut.
158     show_example_processing akan menampilkan langkah-langkah pra-pemrosesan untuk satu gambar.
159     """
160     all_color_features = []
161     all_texture_features = []
162     all_shape_features = []
163     all_labels = []
164
165     print(f"Memuat dataset dari: {base_path}")
166

```

Gambar A.2 Kode Program ekstraksi_klasifikasi.py

```

166
167 # Determine maximum lengths if not provided (for training data)
168 if feature_lengths_map is None:
169     # Create a dummy image for feature length determination
170     dummy_img = np.zeros((target_size[0], target_size[1], 3), dtype=np.uint8)
171     max_color_len = len(extract_color(dummy_img))
172     max_texture_len = len(extract_texture(dummy_img))
173     max_shape_len = len(extract_shape(dummy_img))
174     feature_lengths_map = {'color': max_color_len, 'texture': max_texture_len, 'shape': max_shape_len}
175 else:
176     max_color_len = feature_lengths_map['color']
177     max_texture_len = feature_lengths_map['texture']
178     max_shape_len = feature_lengths_map['shape']
179
180 example_processed_flag = False
181
182 for label in os.listdir(base_path):
183     label_path = os.path.join(base_path, label)
184     if not os.path.isdir(label_path):
185         continue
186
187     files = [f for f in os.listdir(label_path) if f.lower().endswith(('.jpg', '.png', '.jpeg'))]
188     if len(files) == 0:
189         print(f'Tidak ada gambar ditemukan di {label_path}')
190         continue
191
192     if sample_per_class:
193         sampled_files = random.sample(files, min(sample_per_class, len(files)))
194     else:
195         sampled_files = files # Use all files if no sample_per_class is specified
196
197     print(f'Mengambil {len(sampled_files)} sampel untuk kelas '{label.capitalize()}'')
198
199     for filename in sampled_files:
200         filepath = os.path.join(label_path, filename)
201         img = cv2.imread(filepath)
202         if img is None:
203             print(f'Warning: Tidak dapat membaca gambar {filepath}, dilewati.')
204             continue
205
206         try:
207             # Show example processing for the first image
208             if show_example_processing and not example_processed_flag:
209                 print(f'Menampilkan contoh pra-pemrosesan untuk gambar: {filepath}')
210                 processed_img = preprocess_image(img, target_size, denoise, enhance_contrast, show_steps=True, title_suffix=f'({label})')
211                 example_processed_flag = True
212             else:
213                 processed_img = preprocess_image(img, target_size, denoise, enhance_contrast)
214
215             color_feat = extract_color(processed_img)
216             texture_feat = extract_texture(processed_img)
217             shape_feat = extract_shape(processed_img)
218
219             color_feat = np.pad(color_feat, (0, max_color_len - len(color_feat)))
220             texture_feat = np.pad(texture_feat, (0, max_texture_len - len(texture_feat)))
221             shape_feat = np.pad(shape_feat, (0, max_shape_len - len(shape_feat)))
222
223             all_color_features.append(color_feat)
224             all_texture_features.append(texture_feat)
225             all_shape_features.append(shape_feat)
226             all_labels.append(label)
227
228         except Exception as e:
229             print(f'Error memproses {filepath}: {e}')
230             continue
231
232     print(f'Total data yang berhasil diproses: {len(all_labels)}')
233     if len(all_labels) == 0:
234         return np.array([], np.array([], np.array([], np.array([], feature_lengths_map
235
236     count_per_class = Counter(all_labels)
237     for kategori, jumlah in count_per_class.items():
238         print(f'{kategori.capitalize()}: {jumlah} gambar')
239
240     return np.array(all_color_features), np.array(all_texture_features), np.array(all_shape_features), \
241           np.array(all_labels), feature_lengths_map
242
243 # ----- TRAINING & EVALUASI UNTUK SETIAP JENIS FITUR -----
244
245
246 def run_models_for_feature_evaluation(X_train_color, X_test_color, y_train, y_test,
247                                     X_train_texture, X_test_texture,
248                                     X_train_shape, X_test_shape,
249                                     unique_labels, feature_lengths_map):

```

Gambar A.3 Kode Program ekstraksi_klasifikasi.py

```

249 unique_labels, feature_lengths_map):
250 all_results = {}
251
252 # Combined Features
253 X_train_combined = np.hstack((X_train_color, X_train_texture, X_train_shape))
254 X_test_combined = np.hstack((X_test_color, X_test_texture, X_test_shape))
255
256 # Add combined features to the dictionary
257 feature_datasets = {
258     'color': {'train': X_train_color, 'test': X_test_color},
259     'texture': {'train': X_train_texture, 'test': X_test_texture},
260     'shape': {'train': X_train_shape, 'test': X_test_shape},
261     'combined': {'train': X_train_combined, 'test': X_test_combined} # New entry
262 }
263
264 # Store scalers for all feature types, including combined
265 scalers_to_save = {}
266
267 for feature_type, X_data in feature_datasets.items():
268     print(f"\n--- Melatih dan Mengevaluasi model menggunakan fitur: {feature_type.upper()} ---")
269
270     if len(X_data['train']) == 0 or len(X_data['test']) == 0:
271         print(f"Tidak ada data training atau testing untuk fitur {feature_type}, dilwati.")
272         continue
273
274     scaler = StandardScaler()
275     X_train_scaled = scaler.fit_transform(X_data['train'])
276     X_test_scaled = scaler.transform(X_data['test'])
277
278     # Save scaler for later use in prediction
279     joblib.dump(scaler, f'scaler_{feature_type}.pkl')
280     scalers_to_save[feature_type] = scaler # Store scaler reference
281
282     current_feature_results = {}
283
284     # --- K-Nearest Neighbors (KNN) ---
285     print(f"Melatih model KNN untuk {feature_type}...")
286     param_grid_knn = {'n_neighbors': [3, 5, 7, 9, 11]}
287     grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5, verbose=0, n_jobs=-1)
288     grid_knn.fit(X_train_scaled, y_train)
289     knn = grid_knn.best_estimator_
290     print(f"KNN Best parameters ({feature_type}): {grid_knn.best_params}")
291
292     knn_pred = knn.predict(X_test_scaled)
293     knn_acc = accuracy_score(y_test, knn_pred)
294     print(f"Akurasi KNN ({feature_type}) pada data uji: {knn_acc:.4f} ({knn_acc * 100:.2f}%)")
295     knn_report = classification_report(y_test, knn_pred, output_dict=True, zero_division=0, labels=unique_labels)
296     print("Classification Report KNN (Data Uji):\n", classification_report(y_test, knn_pred, zero_division=0, labels=unique_labels))
297
298     current_feature_results['knn_acc'] = knn_acc
299     current_feature_results['knn_report'] = knn_report
300
301     joblib.dump(knn, f'model_knn_{feature_type}.pkl')
302
303     # --- Support Vector Machine (SVM) ---
304     print(f"Melatih model SVM untuk {feature_type}...")
305     param_grid_svm = {
306         'C': [0.1, 1, 10, 100],
307         'gamma': [1, 0.1, 0.01, 0.001],
308         'kernel': ['rbf']
309     }
310
311     grid_svm = GridSearchCV(SVC(probability=True), param_grid_svm, cv=5, verbose=0, n_jobs=-1)
312     grid_svm.fit(X_train_scaled, y_train)
313     svm = grid_svm.best_estimator_
314     print(f"SVM Best parameters ({feature_type}): {grid_svm.best_params}")
315
316     svm_pred = svm.predict(X_test_scaled)
317     svm_acc = accuracy_score(y_test, svm_pred)
318     print(f"Akurasi SVM ({feature_type}) pada data uji: {svm_acc:.4f} ({svm_acc * 100:.2f}%)")
319     svm_report = classification_report(y_test, svm_pred, output_dict=True, zero_division=0, labels=unique_labels)
320     print("Classification Report SVM (Data Uji):\n", classification_report(y_test, svm_pred, zero_division=0, labels=unique_labels))
321
322     current_feature_results['svm_acc'] = svm_acc
323     current_feature_results['svm_report'] = svm_report
324
325     joblib.dump(svm, f'model_svm_{feature_type}.pkl')
326
327     all_results[feature_type] = current_feature_results
328
329     # Plot Confusion Matrix for SVM
330     cm_svm = confusion_matrix(y_test, svm_pred, labels=unique_labels)
331     disp_svm = ConfusionMatrixDisplay(confusion_matrix=cm_svm, display_labels=unique_labels)
332     disp_svm.plot(cmap=plt.cm.Blues)

```

Gambar A.4 Kode Program ekstraksi_klasifikasi.py

```

333 | plt.title(f"Confusion Matrix SVM - Fitur {feature_type.upper()}")
334 | plt.show()
335 |
336 | # --- SINGKAPAN FITUR TERBAIK UNTUK SETIAP KATEGORI ---
337 | print("\n--- RINGKASAN PERFORMA FITUR PER KATEGORI ---")
338 | best_features_per_category = {}
339 |
340 | # We need to know the original lengths of combined features for padding later
341 | # This assumes that the 'feature_lengths_map' passed in already contains these.
342 | # We should add a 'combined' entry to feature_lengths_map itself from main if not present.
343 | # For now, let's just calculate it here
344 | if 'combined' not in feature_lengths_map:
345 |     feature_lengths_map['combined'] = X_train_combined.shape[1]
346 |     joblib.dump(feature_lengths_map, 'feature_lengths.pkl') # Save updated map
347 |
348 | for label in unique_labels:
349 |     print(f"\nUntuk kategori: {label.upper()}")
350 |     best_f1_score = -1
351 |     best_feature_combo = None # 'color', 'texture', 'shape', or 'combined'
352 |     best_model_type = None
353 |
354 |     # Iterate through all feature types, including 'combined'
355 |     for feature_type, results in all_results.items():
356 |         if label in results['svm_report']:
357 |             svm_f1 = results['svm_report'][label]['f1-score']
358 |             print(f"    Fitur {feature_type.upper()} (SVM) - F1-Score: {svm_f1:.4f}")
359 |             if svm_f1 > best_f1_score:
360 |                 best_f1_score = svm_f1
361 |                 best_feature_combo = feature_type
362 |                 best_model_type = 'SVM'
363 |
364 |         if label in results['knn_report']:
365 |             knn_f1 = results['knn_report'][label]['f1-score']
366 |             print(f"    Fitur {feature_type.upper()} (KNN) - F1-Score: {knn_f1:.4f}")
367 |             if knn_f1 > best_f1_score:
368 |                 best_f1_score = knn_f1
369 |                 best_feature_combo = feature_type
370 |                 best_model_type = 'KNN'
371 |
372 |     if best_feature_combo:
373 |         print(f"    Kesimpulan: Fitur '{best_feature_combo.upper()}' dengan model '{best_model_type}' memberikan F1-Score terbaik ({best_f1_score:.4f}) untuk '{label.upper()}'")
374 |         best_features_per_category[label] = {
375 |             'best_feature': best_feature_combo, # This can now be 'combined'
376 |             'best_f1_score': best_f1_score,
377 |             'best_model_type': best_model_type
378 |         }
379 |     else:
380 |         print(f"    Tidak ada data performa yang valid untuk kategori {label}.")
381 |
382 | joblib.dump(best_features_per_category, 'best_features_per_category.pkl')
383 | print("\nRingkasan fitur terbaik per kategori disiepan di: best_features_per_category.pkl")
384 |
385 | return all_results
386 |
387 | # ----- MAIN -----
388 |
389 | if __name__ == "__main__":
390 |     training_dataset_path = 'dataset'
391 |     testing_dataset_path = 'datauji'
392 |
393 |     # Check if dataset folders exist
394 |     if not os.path.exists(training_dataset_path):
395 |         print(f"Error: Folder dataset training '{training_dataset_path}' tidak ditemukan.")
396 |         print("Pastikan Anda memiliki struktur folder seperti ini:")
397 |         print("    your_script.py")
398 |         print("    dataset/")
399 |         print("        |-- plastik/")
400 |         print("        |-- kertas/")
401 |         print("        |-- organik/")
402 |         exit()
403 |
404 |     if not os.path.exists(testing_dataset_path):
405 |         print(f"Error: Folder dataset testing '{testing_dataset_path}' tidak ditemukan.")
406 |         print("Pastikan Anda memiliki folder 'datauji' dengan subfolder kategori yang sama.")
407 |         exit()
408 |
409 |     # STEP 1: LOAD AND EXTRACT FEATURES FOR TRAINING DATA
410 |     print("\n--- STEP 1: Memuat dan Mengekstrak Fitur dari Data Training ---")
411 |     X_train_color, X_train_texture, X_train_shape, y_train_labels, feature_lengths_map = load_and_extract_features(
412 |         training_dataset_path,
413 |         sample_per_class=150,
414 |         denoise=True,
415 |         enhance_contrast=True,

```

Gambar A.5 Kode Program ekstraksi_klasifikasi.py

```

415     enhance_contrast=True,
416     show_example_processing=True
417 )
418
419 # --- SAVE FEATURE LENGTHS MAP ---
420 # This will now include the 'combined' feature length after run_models_for_feature_evaluation updates it
421 # We save it here initially, and then run_models_for_feature_evaluation might update it further
422 if feature_lengths_map:
423     joblib.dump(feature_lengths_map, 'feature_lengths.pkl')
424     print(f"Panjang fitur awal disimpan ke: feature_lengths.pkl")
425 else:
426     print("Peringatan: feature_lengths_map kosong, tidak dapat menyimpan feature_lengths.pkl.")
427
428
429 if len(y_train_labels) == 0:
430     print("Dataset training kosong atau gagal dimuat. Proses training tidak dapat dilanjutkan.")
431     exit()
432
433 unique_labels = np.unique(y_train_labels)
434 print(f"\nLabel unik yang ditemukan di dataset training: {unique_labels}")
435
436 # STEP 2: LOAD AND EXTRACT FEATURES FOR TESTING DATA
437 print("\n--- STEP 2: Memuat dan Mengekstrak Fitur dari Data Testing ---")
438 X_test_color, X_test_texture, X_test_shape, y_test_labels, _ = load_and_extract_features(
439     testing_dataset_path,
440     sample_per_class=None,
441     target_size=(100, 100),
442     denoise=True,
443     enhance_contrast=True,
444     feature_lengths_map=feature_lengths_map,
445     show_example_processing=True
446 )
447
448 if len(y_test_labels) == 0:
449     print("Dataset testing kosong atau gagal dimuat. Proses evaluasi tidak dapat dilanjutkan.")
450     exit()
451
452 # STEP 3: TRAIN AND EVALUATE MODELS
453 print("\n--- STEP 3: Memulai Training dan Evaluasi Model ---")
454 all_training_results = run_models_for_feature_evaluation(
455     X_train_color, X_test_color, y_train_labels, y_test_labels,
456     X_train_texture, X_test_texture,
457     X_train_shape, X_test_shape,
458     unique_labels, feature_lengths_map
459 )
460
461 print("\n--- Proses training dan evaluasi fitur selesai ---")
462 print("Model, scaler, dan ringkasan performa telah disimpan.")
463
464 # STEP 4: Visualize Overall F1-Scores per Feature Type
465 print("\n--- STEP 4: Visualisasi Performa Keseluruhan (F1-Score) ---")
466 # Include 'combined' in feature_types for visualization
467 feature_types_for_viz = ['color', 'texture', 'shape', 'combined']
468 model_types = ['knn', 'svm']
469
470 f1_scores_overall = {f_type: {'knn': 0, 'svm': 0} for f_type in feature_types_for_viz}
471
472 for f_type, results in all_training_results.items():
473     if 'knn_report' in results and 'macro avg' in results['knn_report']:
474         f1_scores_overall[f_type]['knn'] = results['knn_report']['macro avg']['f1-score']
475     if 'svm_report' in results and 'macro avg' in results['svm_report']:
476         f1_scores_overall[f_type]['svm'] = results['svm_report']['macro avg']['f1-score']
477
478 x = np.arange(len(feature_types_for_viz))
479 width = 0.35
480
481 fig, ax = plt.subplots(figsize=(12, 6)) # Adjust figure size for more bars
482 rects1 = ax.bar(x - width/2, [f1_scores_overall[ft]['knn'] for ft in feature_types_for_viz], width, label='KNN')
483 rects2 = ax.bar(x + width/2, [f1_scores_overall[ft]['svm'] for ft in feature_types_for_viz], width, label='SVM')
484
485 ax.set_ylabel('Macro Avg F1-Score')
486 ax.set_title('Macro Avg F1-Score per Feature Type and Model')
487 ax.set_xticks(x)
488 ax.set_xticklabels([f.capitalize() for f in feature_types_for_viz])
489 ax.legend()
490 ax.set_ylim(0, 1)
491
492 def autolabel(rects):
493     """Attach a text label above each bar in *rects*, displaying its height."""
494     for rect in rects:
495         height = rect.get_height()
496         ax.annotate(f'{height:.2f}',
497                 xy=(rect.get_x() + rect.get_width() / 2, height),
498                 xytext=(0, 3), # 3 points vertical offset
499                 textcoords="offset points",
500                 ha='center', va='bottom')
501
502 autolabel(rects1)
503 autolabel(rects2)
504
505 plt.tight_layout()
506 plt.show()

```

Gambar A.6 Kode Program ekstraksi_klasifikasi.py

B. Potongan Kode Program – (main_klasifikasi.py)

```

1 import os
2 import cv2
3 import numpy as np
4 import joblib
5 import matplotlib.pyplot as plt
6 from skimage.feature import graycomatrix, graycoprops, local_binary_pattern
7 from sklearn.preprocessing import StandardScaler
8 from collections import Counter
9 import tkinter as tk
10 from tkinter import filedialog
11
12 # --- EKSTRAKSI FITUR (Harus SAMA PERSIS dengan fungsi di train_model.py) ---
13
14 # Fungsi preprocess_image harus sama persis dengan yang di train_model.py
15 def preprocess_image(img, target_size=(100, 100), denoise=False, enhance_contrast=False):
16     """
17     Melakukan pra-pemrosesan gambar: resizing, denoising (opsional), dan contrast enhancement (opsional).
18     """
19     img_resized = cv2.resize(img, target_size, interpolation=cv2.INTER_AREA)
20
21     if denoise:
22         img_resized = cv2.fastNlMeansDenoisingColored(img_resized, None, 10, 10, 7, 21) # H_COLOR 10,10,7,21
23
24     if enhance_contrast:
25         lab = cv2.cvtColor(img_resized, cv2.COLOR_BGR2LAB)
26         l_channel = lab[:, :, 0]
27         clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
28         cl = clahe.apply(l_channel)
29         lab[:, :, 0] = cl
30         img_resized = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
31
32     return img_resized
33
34
35 def extract_color(img):
36     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
37     hist = cv2.calcHist([hsv], [0, 1, 2], None, [16, 16, 16], [0, 180, 0, 256, 0, 256])
38     cv2.normalize(hist, hist)
39     return hist.flatten()
40
41
42 def extract_texture(img):
43     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
44     distances = [1, 2]
45     angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
46     glcm = graycomatrix(gray, distances, angles, 256, symmetric=True, normed=True)
47
48     contrast = graycoprops(glcm, 'contrast').mean()
49     dissimilarity = graycoprops(glcm, 'dissimilarity').mean()
50     homogeneity = graycoprops(glcm, 'homogeneity').mean()
51     energy = graycoprops(glcm, 'energy').mean()
52     correlation = graycoprops(glcm, 'correlation').mean()
53     glcm_features = [contrast, dissimilarity, homogeneity, energy, correlation]
54
55     radius = 3
56     n_points = 8 * radius
57     lbp = local_binary_pattern(gray, n_points, radius, method='uniform')
58     (hist_lbp, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
59     hist_lbp = hist_lbp.astype("float")
60     hist_lbp /= (hist_lbp.sum() + 1e-7)
61
62     return np.array(glcm_features + hist_lbp.tolist()).flatten()
63
64
65 def extract_shape(img):
66     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
67     # Parameter harus sama persis dengan train_model.py
68     blur = cv2.GaussianBlur(gray, (5, 5), 0)
69     thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
70     | | | | | cv2.THRESH_BINARY_INV, 11, 2)
71     # kernel = np.ones((3, 3), np.uint8)
72     # opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=3)
73
74     contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
75
76     img_contour_drawn = img.copy()
77     hu = np.zeros((7,), dtype=np.float32)
78
79     if contours:
80         cnt = max(contours, key=cv2.contourArea)
81         cv2.drawContours(img_contour_drawn, [cnt], -1, (0, 255, 0), 2)
82
83         moments = cv2.moments(cnt)
84         if moments['m00'] != 0:
85             hu = cv2.HuMoments(moments).flatten()
86             hu = -np.sign(hu) * np.log10(np.abs(hu) + 1e-7)

```

Gambar B.1 Kode Program ekstraksi_klasifikasi.py

```

86     return hu, thresh, img_contour_drawn
87
88 # --- VISUALISASI SPESIFIK UNTUK PREDIKSI ---
89
90 def visualize_color_features(img, color_feat):
91     fig, axs = plt.subplots(1, 2, figsize=(10, 4))
92     axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
93     axs[0].set_title("Gambar Asli (Setelah Preprocessing)")
94     axs[0].axis('off')
95
96     axs[1].bar(range(len(color_feat)), color_feat, color='purple')
97     axs[1].set_title("Histogram HSV (Fitur Warna)")
98     axs[1].set_xlabel("Bin ke-")
99     axs[1].set_ylabel("Normalisasi Jumlah")
100    plt.suptitle("Visualisasi Fitur Warna")
101    plt.tight_layout()
102    plt.show()
103    print(f"[INFO] Panjang fitur warna (HSV): {len(color_feat)}")
104
105 def visualize_texture_features(img, texture_feat):
106     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Ini dari gambar yang sudah diproses
107     glcm_feat_viz = texture_feat[:5]
108
109     fig, axs = plt.subplots(1, 3, figsize=(15, 5))
110     axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
111     axs[0].set_title("Gambar Asli (Setelah Preprocessing)")
112     axs[0].axis('off')
113
114     axs[1].imshow(gray, cmap='gray')
115     axs[1].set_title("Grayscale")
116     axs[1].axis('off')
117
118     feature_names_glcm = ['Contrast', 'Dissimilarity', 'Homogeneity', 'Energy', 'Correlation']
119     axs[2].bar(feature_names_glcm, glcm_feat_viz, color='orange')
120     axs[2].set_title("Fitur Tekstur (GLCM)")
121     axs[2].set_ylabel("Nilai Fitur")
122     plt.setp(axs[2].get_xticklabels(), rotation=45, ha="right")
123
124     plt.suptitle("Visualisasi Fitur Tekstur")
125     plt.tight_layout()
126     plt.show()
127     print(f"[INFO] Panjang fitur tekstur (GLCM + LBP): {len(texture_feat)}")
128
129 def visualize_shape_features(img, shape_feat, opening_img, img_contour_drawn):
130     fig, axs = plt.subplots(1, 5, figsize=(18, 4))
131
132     axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
133     axs[0].set_title("Gambar Asli (Setelah Preprocessing)")
134     axs[0].axis('off')
135
136     axs[1].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), cmap='gray')
137     axs[1].set_title("Grayscale")
138     axs[1].axis('off')
139
140     # Rekalkulasi thresh untuk konsistensi visualisasi
141     gray_for_thresh = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
142     blur_for_thresh = cv2.GaussianBlur(gray_for_thresh, (5, 5), 0)
143     thresh_for_viz = cv2.adaptiveThreshold(blur_for_thresh, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
144     | | | | | | | | | | cv2.THRESH_BINARY_INV, 11, 2)
145
146     axs[2].imshow(thresh_for_viz, cmap='gray')
147     axs[2].set_title("Adaptive Thresholding")
148     axs[2].axis('off')
149
150     # axs[3].imshow(opening_img, cmap='gray')
151     # axs[3].set_title("Opening Morphology")
152     # axs[3].axis('off')
153
154     axs[3].imshow(cv2.cvtColor(img_contour_drawn, cv2.COLOR_BGR2RGB))
155     axs[3].set_title("Kontur Objek Terdeteksi")
156     axs[3].axis('off')
157
158     plt.suptitle("Visualisasi Fitur Bentuk")
159     plt.tight_layout()
160     plt.show()
161
162     print("[INFO] Hu Moments:")
163     for i, val in enumerate(shape_feat):
164         print(f"    Hu[{i+1}] = {val:.6e}")
165

```

Gambar B.2 Kode Program main_klasifikasi.py


```

166 # --- FUNGSI PREDIKSI UTAMA ---
167
168 def predict_image_with_visuals(image_path,
169                               target_size=(100, 100), # Konsisten dengan training
170                               denoise=True,           # Konsisten dengan training
171                               enhance_contrast=True    # Konsisten dengan training
172                               ):
173     """
174     Memuat gambar, melakukan pra-pemrosesan, mengekstrak SEMUA fitur,
175     memprediksi kelas menggunakan model terbaik berdasarkan best_features_per_category.pkl,
176     dan menampilkan visualisasi yang relevan.
177     """
178     img = cv2.imread(image_path)
179     if img is None:
180         print(f"✗ ERROR: Gagal membuka gambar: {image_path}")
181         return
182
183     print(f"\n[INFO] Memproses gambar: {os.path.basename(image_path)}")
184
185     # Lakukan pra-pemrosesan pada gambar input
186     processed_img = preprocess_image(img, target_size, denoise, enhance_contrast)
187
188     # Muat semua model dan scaler terpisah
189     loaded_models = {}
190     loaded_scalers = {}
191     feature_types = ['color', 'texture', 'shape']
192     class_labels = None
193     feature_lengths = None
194
195     try:
196         for f_type in feature_types:
197             loaded_models[f_type] = {
198                 'svm': joblib.load(f'model_svm_{f_type}.pkl'),
199                 'knn': joblib.load(f'model_knn_{f_type}.pkl')
200             }
201             loaded_scalers[f_type] = joblib.load(f'scaler_{f_type}.pkl')
202             if class_labels is None:
203                 # Assuming all models have the same classes attribute
204                 class_labels = loaded_models[f_type]['svm'].classes_
205
206             best_features_info = joblib.load('best_features_per_category.pkl')
207             feature_lengths = joblib.load('feature_lengths.pkl') # Muat feature_lengths
208     except FileNotFoundError as e:
209         print(f"✗ ERROR: File model/scaler/best_features_info/feature_lengths tidak ditemukan: {e}")
210         print("Pastikan Anda sudah menjalankan script train_model.py terlebih dahulu untuk menghasilkan file-file ini.")
211         return
212     except Exception as e:
213         print(f"✗ ERROR: Gagal memuat model/scaler/info fitur: {e}")
214         return
215
216     # --- EKSTRAKSI SEMUA JENIS FITUR DARI GAMBAR YANG SUDAH DIPROSES ---
217     color_feat = extract_color(processed_img)
218     texture_feat = extract_texture(processed_img)
219     shape_feat_hu, shape_feat_opening_img, shape_feat_img_contour_drawn = extract_shape(processed_img)
220
221     # --- NORMALISASI DAN PREDIKSI DENGAN SETIAP MODEL TERPISAH ---
222
223     all_predictions_with_proba = []
224
225     def get_proba_string(proba_array, labels):
226         proba_str = ", ".join([f'{labels[i].upper():} {p*100:.2f}%' for i, p in enumerate(proba_array)])
227         return proba_str
228
229     # Prediksi menggunakan model warna
230     # Padding fitur sebelum scaling
231     color_feat_padded = np.pad(color_feat, (0, feature_lengths['color'] - len(color_feat)))
232     color_feat_scaled = loaded_scalers['color'].transform([color_feat_padded])[0]
233
234     svm_pred_color = loaded_models['color']['svm'].predict([color_feat_scaled])[0]
235     svm_proba_color = loaded_models['color']['svm'].predict_proba([color_feat_scaled])[0]
236     all_predictions_with_proba.append((svm_pred_color, np.max(svm_proba_color), 'svm', 'color', color_feat))
237
238     knn_pred_color = loaded_models['color']['knn'].predict([color_feat_scaled])[0]
239     knn_proba_color = loaded_models['color']['knn'].predict_proba([color_feat_scaled])[0]
240     all_predictions_with_proba.append((knn_pred_color, np.max(knn_proba_color), 'knn', 'color', color_feat))
241
242     # Prediksi menggunakan model tekstur
243     texture_feat_padded = np.pad(texture_feat, (0, feature_lengths['texture'] - len(texture_feat)))
244     texture_feat_scaled = loaded_scalers['texture'].transform([texture_feat_padded])[0]
245
246     svm_pred_texture = loaded_models['texture']['svm'].predict([texture_feat_scaled])[0]
247     svm_proba_texture = loaded_models['texture']['svm'].predict_proba([texture_feat_scaled])[0]
248     all_predictions_with_proba.append((svm_pred_texture, np.max(svm_proba_texture), 'svm', 'texture', texture_feat))
249
250     knn_pred_texture = loaded_models['texture']['knn'].predict([texture_feat_scaled])[0]

```

Gambar B.3 Kode Program ekstraksi_klasifikasi.py


```

250 knn_pred_texture = loaded_models['texture']['knn'].predict([texture_feat_scaled])[0]
251 knn_proba_texture = loaded_models['texture']['knn'].predict_proba([texture_feat_scaled])[0]
252 all_predictions_with_proba.append((knn_pred_texture, np.max(knn_proba_texture), 'knn', 'texture', texture_feat))
253
254 # Prediksi menggunakan model bentuk
255 shape_feat_padded = np.pad(shape_feat_hu, (0, feature_lengths['shape'] - len(shape_feat_hu)))
256 shape_feat_scaled = loaded_scalers['shape'].transform([shape_feat_padded])[0]
257
258 svm_pred_shape = loaded_models['shape']['svm'].predict([shape_feat_scaled])[0]
259 svm_proba_shape = loaded_models['shape']['svm'].predict_proba([shape_feat_scaled])[0]
260 all_predictions_with_proba.append((svm_pred_shape, np.max(svm_proba_shape), 'svm', 'shape', shape_feat_hu))
261
262 knn_pred_shape = loaded_models['shape']['knn'].predict([shape_feat_scaled])[0]
263 knn_proba_shape = loaded_models['shape']['knn'].predict_proba([shape_feat_scaled])[0]
264 all_predictions_with_proba.append((knn_pred_shape, np.max(knn_proba_shape), 'knn', 'shape', shape_feat_hu))
265
266 # --- Tampilkan Prediksi Individual ---
267 print("\n--- PREDIKSI INDIVIDUAL DARI SETIAP MODEL (SVM & KNN) ---")
268 for pred_label, proba_val, model_type, f_type, _ in all_predictions_with_proba:
269     model = loaded_models[f_type][model_type]
270     # Re-scale feature data to get full probability array for printing
271     # feat_data needs to be correct for each feature type
272     if f_type == 'color':
273         current_feat_data_for_proba = color_feat_padded
274     elif f_type == 'texture':
275         current_feat_data_for_proba = texture_feat_padded
276     else: # 'shape'
277         current_feat_data_for_proba = shape_feat_padded
278
279     full_proba = model.predict_proba([loaded_scalers[f_type].transform([current_feat_data_for_proba])[0]])[0]
280     print(f" Model {model_type.upper()} {f_type.upper()}: {pred_label.upper()} (Prob: {get_proba_string(full_proba, class_labels)})")
281
282 # --- STRATEGI KEPUTUSAN FINAL (Menggunakan best_features_per_category.pkl) ---
283
284 final_pred_label = None
285 max_score = -1
286 chosen_model_info = None
287
288 if not best_features_info:
289     print("✗ ERROR: best_features_per_category.pkl kosong atau tidak valid.")
290     print("Menggunakan Voting Mayoritas sebagai fallback.")
291     vote_counts = Counter([p[0] for p in all_predictions_with_proba])
292     final_pred_label = vote_counts.most_common(1)[0][0]
293     chosen_model_info = {"feature_type": "Voting", "model_type": "Majority (Fallback)"}
294 else:
295     # Step 1: Kumpulkan semua prediksi unik yang didapat dari model.
296     # Ini akan menjadi kandidat untuk prediksi akhir.
297     unique_predicted_labels = set([p[0] for p in all_predictions_with_proba])
298
299     # Step 2: Untuk setiap kandidat label, cari model terbaiknya dari best_features_info
300     # dan gunakan f1-score dari training sebagai basis pemilihan.
301     for candidate_label in unique_predicted_labels:
302         if candidate_label in best_features_info:
303             best_info_for_candidate = best_features_info[candidate_label]
304
305             best_feat_type = best_info_for_candidate['best_feature']
306             best_algo_type = best_info_for_candidate['best_model_type'] # KNN atau SVM
307             best_f1_from_training = best_info_for_candidate['best_f1_score']
308
309             # Temukan prediksi dari model 'terbaik' ini untuk gambar yang sedang diproses
310             for pred_label_actual, proba_actual, model_type_actual, f_type_actual, _ in all_predictions_with_proba:
311                 if pred_label_actual == candidate_label and \
312                     f_type_actual == best_feat_type and \
313                     model_type_actual.lower() == best_algo_type.lower():
314
315                     # Gunakan f1-score dari training sebagai 'score' untuk memilih prediksi terbaik.
316                     # Semakin tinggi f1-score training, semakin kita percaya pada model ini untuk kelas tersebut.
317                     current_score = best_f1_from_training
318
319                     if current_score > max_score:
320                         max_score = current_score
321                         final_pred_label = candidate_label
322                         chosen_model_info = {
323                             "feature_type": best_feat_type,
324                             "model_type": best_algo_type,
325                             "f1_score_from_training": best_f1_from_training,
326                             "predicted_proba_for_this_model": proba_actual # Probabilitas tertinggi dari model terbaik
327                         }
328                     break # Pindah ke kandidat label berikutnya setelah menemukan model terbaiknya
329
330     # Jika kandidat_label tidak ada di best_features_info (misalnya, jika ada kelas yang tidak pernah diprediksi terbaik)
331     elif candidate_label not in best_features_info:
332         print(f"Warning: Info fitur terbaik untuk '{candidate_label}' tidak ditemukan di best_features_per_category.pkl. Ini tidak biasa.")
333

```

Gambar B.4 Kode Program ekstraksi_klasifikasi.py

```

333 # Fallback jika tidak ada prediksi yang memiliki info best_features_info yang relevan
334 if final_pred_label is None and all_predictions_with_proba:
335     print("Tidak ada kandidat prediksi yang cocok dengan best_features_info atau best_features_info kosong.")
336     print("Menggunakan Voting Mayoritas sebagai fallback akhir.")
337     vote_counts = Counter([p[0] for p in all_predictions_with_proba])
338     final_pred_label = vote_counts.most_common(1)[0][0]
339     chosen_model_info = {"feature_type": "Voting", "model_type": "Majority (Fallback)"}
340 elif final_pred_label is None: # Kasus jika tidak ada prediksi sama sekali (sangat jarang)
341     final_pred_label = "UNKNOWN"
342     chosen_model_info = {"feature_type": "N/A", "model_type": "N/A"}
343
344 print(f"\n--- HASIL PREDIKSI FINAL: {final_pred_label.upper()} ---")
345 if chosen_model_info:
346     print(f"    Dipilih berdasarkan: Model {chosen_model_info.get('model_type', '')} Fitur {chosen_model_info.get('feature_type', '')}")
347     if 'f1_score_from_training' in chosen_model_info:
348         print(f"    F1-Score saat Training untuk model ini: {chosen_model_info['f1_score_from_training']:.4f}")
349     if 'predicted_proba_for_this_model' in chosen_model_info:
350         print(f"    Probabilitas Prediksi dari model ini: {chosen_model_info['predicted_proba_for_this_model']*100:.2f}%")
351
352 # --- Tampilkan Visualisasi Sesuai HASIL PREDIKSI FINAL ---
353 print("[INFO] Menampilkan visualisasi fitur yang paling relevan dengan hasil prediksi akhir:")
354
355 if final_pred_label.lower() == 'kertas':
356     visualize_color_features(processed_img, color_feat)
357 elif final_pred_label.lower() == 'plastik':
358     visualize_texture_features(processed_img, texture_feat)
359 elif final_pred_label.lower() == 'organik':
360     visualize_shape_features(processed_img, shape_feat_hu, shape_feat_opening_img, shape_feat_img_contour_drawn)
361 else:
362     print("[INFO] Tidak ada visualisasi spesifik yang ditentukan untuk kelas ini.")
363
364 print(f"\nGambar: {os.path.basename(image_path)}")
365 print(f"Prediksi Model Final: {final_pred_label.upper()}")
366 # We can't determine original category from a random file selection, so we'll remove it.
367 # kategori_asli = os.path.basename(os.path.dirname(image_path)).lower()
368 # print(f"Kategori Asli (dari folder): {kategori_asli.upper()}")
369 print("-" * 30)
370
371 # --- INTERAKSI PENGGUNA ---
372 if __name__ == "__main__":
373     print("\n--- PROGRAM KLASIFIKASI SAMPAH ---")
374     print("Silakan pilih gambar yang ingin Anda klasifikasikan.")
375
376     # Hide the main tkinter window
377     root = tk.Tk()
378     root.withdraw()
379
380     image_to_predict_path = filedialog.askopenfilename(
381         title="Pilih Gambar untuk Klasifikasi",
382         filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp *.tiff")]
383     )
384
385     if image_to_predict_path:
386         predict_image_with_visuals(image_to_predict_path)
387     else:
388         print("❌ Tidak ada gambar yang dipilih. Program dihentikan.")
389
390
391
392

```

Gambar B.5 Kode Program ekstraksi_klasifikasi.py

Link Google Drive (Video dan Program)

<https://drive.google.com/drive/folders/1wrFZpfSfalDEOu6j8qgZnMZFcoyTyWEk?usp=sharing>