UAS

PENGOLAHAN CITRA

NAMA : Daniel David Hamonangan Hutabarat

NIM : 202331129

KELAS : F

DOSEN : Dr. Dra. Dwina Kuswardani, M.kom

NO.PC : 8

ASISTEN : 1. Sasikirana Ramadhanty Setiawan Putri

2. Rizqy Amanda

3. Ridho Chaerullah

4. Sakura Amastasya

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

Contents

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Bagaimana cara mengimplementasikan filter spasial, khususnya filter Median menggunakan pustaka OpenCV dan filter Mean secara manual, untuk memodifikasi dan menghaluskan citra digital?
2. Bagaimana metode segmentasi berbasis warna dapat diterapkan secara efektif untuk mengidentifikasi dan memisahkan objek-objek yang berbeda yaitu buah dan daun dari latar belakangnya dalam sebuah citra kompleks?
3. Bagaimana dampak dari dua teknik kompresi citra yang berbeda, yaitu kompresi lossy JPEG dan kuantisasi warna, terhadap kualitas visual dan ukuran data sebuah citra?

1.2 Tujuan Masalah

1. Menerapkan filter Median dan Mean pada sebuah citra diri, serta mampu membandingkan hasil visual dari kedua filter tersebut, dengan salah satunya merupakan implementasi algoritma manual.
2. Membangun alur kerja segmentasi yang mampu mengisolasi objek buah dan daun secara terpisah dari citra aslinya. Ini termasuk keberhasilan dalam menggunakan konversi ruang warna HSV, thresholding, operasi morfologi, dan analisis kontur.
3. Melakukan kompresi citra melalui dua metode berbeda dan menganalisis hasilnya.

1.3 Manfaat Masalah

1. Memperoleh pemahaman mendalam tentang konsep filtering sebagai teknik dasar untuk pra-pemrosesan dan perbaikan kualitas citra.
2. Menguasai alur kerja segmentasi citra, yang merupakan langkah krusial dan menjadi fondasi untuk aplikasi tingkat lanjut seperti deteksi objek, pengenalan pola, dan analisis citra medis.
3. Memahami prinsip dasar di balik kompresi citra dan perbedaannya, terutama antara kompresi untuk efisiensi penyimpanan dan kompresi untuk penyederhanaan informasi

BAB II
LANDASAN TEORI

2.1. Pengertian Filtering Citra

filtering citra adalah sebuah teknik dalam pemrosesan gambar digital yang digunakan untuk menghilangkan berbagai jenis noise atau derau. Noise merupakan variasi acak pada kecerahan atau informasi warna yang mengganggu gambar asli. Proses. filtering ini bertujuan untuk menarik atau membuang noise yang timbul pada sumber gambar asli melalui penerapan filter yang sesuai dan interpretasi matematis. Terdapat berbagai jenis filter, seperti linear, adaptif, dan non-linear. Beberapa metode filter bekerja dengan cara membuat estimasi nilai setiap piksel menjadi lebih selaras dengan nilai piksel-piksel di sekitarnya, yang seringkali bertujuan untuk menghaluskan gambar dan mengurangi tingkat variasi intensitas antar piksel. Filter yang lebih canggih, seperti filter adaptif, bahkan dirancang untuk dapat mempertahankan detail penting seperti tepian gambar sambil tetap melakukan proses penghilangan noise.

2.2. Jenis-Jenis Filtering Citra

1. Mean Filter

   diklasifikasikan sebagai salah satu jenis filter linear. Cara kerja filter ini adalah dengan membawa estimasi nilai setiap piksel menjadi lebih selaras dengan nilai-nilai piksel tetangganya. Meskipun merupakan dasar bagi filter-filter non-linear yang lebih kompleks , filter linear seperti mean filter memiliki kelemahan signifikan. Secara umum, filter ini cenderung kurang baik dalam mempertahankan detail gambar, seperti merusak tepi yang tajam, garis-garis halus, dan fitur-fitur detail lainnya. Sebuah studi pada tahun 2020 oleh S. Tripathy dan T. Swarnkar menunjukkan bahwa mean filter tidak memadai untuk membersihkan impulse noise dalam jumlah yang berlebihan. Filter ini bekerja dengan baik secara keseluruhan, namun performanya akan menurun ketika probabilitas kemunculan impulse noise dan non-impulse noise menjadi tinggi. Meskipun demikian, mean filter dapat digunakan untuk menangani beberapa jenis noise seperti Gaussian noise, salt and pepper noise, dan speckle noise

2. Median Filter

   Median filter merupakan contoh dari filter non-linear dan digambarkan sebagai metodologi yang mudah digunakan untuk menghaluskan gambar. Fungsi utama dari filter ini adalah untuk mengurangi besarnya variasi intensitas antara satu piksel dengan piksel lainnya. Mekanisme kerjanya adalah dengan mengurutkan semua nilai piksel di dalam sebuah area lingkungan (jendela) ke dalam urutan menaik, kemudian mengganti nilai piksel yang sedang diproses dengan nilai piksel yang berada di tengah (median) dari urutan tersebut. Jika jumlah piksel dalam lingkungan tersebut genap, maka nilai rata-rata dari dua piksel yang berada di tengah digunakan sebagai nilai pengganti. Beberapa penelitian menyoroti keunggulan filter median. Sebuah studi pada tahun 2020 oleh S. Tripathy dan T. Swarnkar menyimpulkan bahwa filter median adalah pendekatan yang masuk akal (reasonable) karena kualitas gambar yang dihasilkannya lebih baik jika dibandingkan dengan teknik lain. Studi lain pada tahun 2017 oleh Hambal dkk. juga berhasil mendapatkan teknik filter median yang mampu menghasilkan gambar bebas noise dengan tingkat degradasi kualitas gambar yang lebih rendah. Berdasarkan hasil analisis dan perbandingan nilai MSE (Mean Squared Error) serta PSNR (Peak Signal-to-Noise Ratio), disimpulkan bahwa filter

median merupakan filter yang paling sesuai untuk menghilangkan jenis noise salt & pepper.

2.3. PengertianSegmentasi Citra

Segmentasi citra adalah sebuah proses atau teknik penting dalam pengolahan citra yang bertujuan untuk membagi atau mempartisi suatu citra menjadi beberapa daerah (region) atau segmen. Setiap segmen yang dihasilkan merupakan kelompok piksel yang memiliki kemiripan atribut atau karakteristik visual tertentu. Tujuan utama dari segmentasi adalah untuk menyederhanakan penyajian gambar menjadi sesuatu yang lebih bermakna dan mudah untuk dianalisis. Dengan melakukan segmentasi, objek dapat dipisahkan dari latar belakangnya atau dari objek lain di dalam citra. Proses ini memungkinkan setiap objek diambil secara individu untuk digunakan sebagai masukan bagi proses selanjutnya, seperti pengenalan pola. Manfaatnya antara lain untuk mempermudah identifikasi objek serta memperjelas karakter tulisan pada dokumen historis sehingga lebih mudah dibaca

2.4. Metode Segmentasi Berbasis Thresholding

Metode thresholding merupakan salah satu teknik segmentasi yang paling sederhana. Prinsip dasarnya adalah melakukan pemisahan piksel berdasarkan variasi intensitas antara piksel objek dan piksel latar belakang. Dalam prosesnya, setiap nilai intensitas piksel pada citra akan dibandingkan dengan sebuah nilai ambang batas (threshold) yang telah ditentukan. Hasil dari metode ini adalah sebuah citra biner, yaitu citra yang hanya memiliki dua kemungkinan tingkat keabuan: hitam dan putih. Tahapan umum yang dilakukan adalah mengonversi citra asli menjadi citra grayscale terlebih dahulu, kemudian citra grayscale tersebut dikonversi menjadi citra biner menggunakan metode thresholding. Terdapat berbagai jenis operasi thresholding, salah satunya adalah Threshold Binary, di mana jika intensitas piksel lebih tinggi dari ambang batas, nilainya akan diubah menjadi nilai maksimum (putih), dan sebaliknya akan diubah menjadi nol (hitam). Salah satu variasi dari metode ini adalah Otsu Thresholding, sebuah algoritma yang dapat membagi histogram citra keabuan ke dalam dua daerah berbeda secara otomatis untuk menentukan nilai ambang.

2.5. Metode Segmentasi Berbasis Clustering (K-Means)

K-Means Clustering adalah salah satu algoritma yang dapat diterapkan untuk proses segmentasi citra. K-Means merupakan metode clustering non-hirarki yang berusaha mempartisi data ke dalam satu atau lebih cluster (kelompok). Metode ini bekerja dengan mengelompokkan data yang memiliki karakteristik sama ke dalam satu cluster yang sama, dan data dengan karakteristik berbeda ke cluster yang lain. Saat diimplementasikan pada citra digital, konsep dasarnya adalah melakukan pembagian piksel-piksel pada citra ke dalam beberapa kelompok berdasarkan kedekatan warna antara satu piksel dengan piksel lainnya. Algoritma K-Means merupakan model berbasis centroid, di mana sebuah objek (piksel) akan dimasukkan ke dalam suatu cluster jika memiliki jarak terpendek terhadap centroid (nilai titik tengah) dari cluster tersebut.

2.6. Definisi Kompresi

Kompresi citra digital adalah sebuah proses atau upaya untuk mereduksi ukuran data dengan menghasilkan representasi digital yang lebih padat, namun tetap dapat mewakili kuantitas informasi yang terkandung di dalamnya. Tujuan utama dari kompresi citra adalah untuk mengurangi redundansi (pengulangan data) yang terdapat dalam gambar, sehingga gambar tersebut dapat disimpan atau ditransmisikan secara lebih efisien. Seiring berkembangnya teknologi digital yang menghasilkan data berkualitas tinggi dengan ukuran yang semakin besar, kompresi menjadi solusi untuk menghemat ruang penyimpanan dan waktu transmisi data. Proses ini mentransformasi data atau simbol tanpa menimbulkan perubahan yang signifikan bagi mata manusia yang mengamatinya.

2.7. Jenis Kompresi

1. Kompresi Lossless adalah metode di mana tidak ada informasi data yang hilang atau berkurang selama proses kompresi. Setelah melalui proses dekompresi, keseluruhan data hasil akan sama persis dengan data aslinya. Teknik ini sangat menjaga keutuhan informasi dan biasanya digunakan untuk data-data penting yang tidak boleh berubah sedikitpun, seperti dokumen atau kode program. Kelemahan dari metode ini adalah rasio kompresi yang dihasilkannya cenderung sangat rendah.
2. Kompresi Lossy adalah metode yang bekerja dengan cara menghilangkan sebagian informasi dari data asli. Teknik ini mengutamakan rasio kompresi yang sangat tinggi dengan mengorbankan beberapa informasi yang hilang, di mana kehilangan tersebut masih dapat ditolerir oleh persepsi mata manusia. Kompresi jenis ini dapat mengurangi ukuran data secara signifikan dan sering digunakan dalam bidang multimedia, seperti untuk memperkecil ukuran file gambar atau video. Proses kompresi lossy umumnya melibatkan tiga tahapan: Transformasi, Kuantisasi, dan Pengkodean, di mana tahap kuantisasi menjadi sumber utama hilangnya data dan bersifat tidak dapat diubah (irreversible).

BAB III
HASIL

3.1 Penjelasan Program

pada ujian akhir semester ini saya mendapatkan 3 soal yang berasal dari 3 materi yang berbeda yaitu filtering citra, segmentasi, dan kompresi

1. Filtering Citra

   Ketentuan khusus Gambar:

   - Gambar merupakan bentuk foto diri.
   - Gambar diambil dari jarak minimal 2 meter dari objek (diri sendiri).

   Ketentuan khusus Code :

   - Gunakan OpenCV untuk median.
   - Buatlah code perhitungan manual untuk mean filtering.
   - Pastikan output sama dengan hasil contoh di soal.

   Contoh output:

   - Median filtering
   - Mean filtering

2. Deteksi Daun (gambar buah bebas asalkan ada daunnya, tidak boleh sama dengan teman lain, dan harus foto sendiri, bukan diambil dari internet/lainnya)

   Ketentuan khusus Gambar:

   - Gambar diambil dengan memiliki buah serta daun.

   Ketentuan khusus Code:

   Output yang dibutuhkan:

   - Citra asli
   - Mask mangga (sesuaikan dengan gambar sendiri)
   - Segmentasi daun (seusaikan dengan gambar sendiri)
   - Segmentasi daun

3. Kompresi

   Ketentuan khusus Gambar:

   - Gambar merupakan bentuk foto diri.
   - Gambar diambil dari jarak minimal 1 meter dari objek (diri
   - sendiri).

   Ketentuan khusus Code:

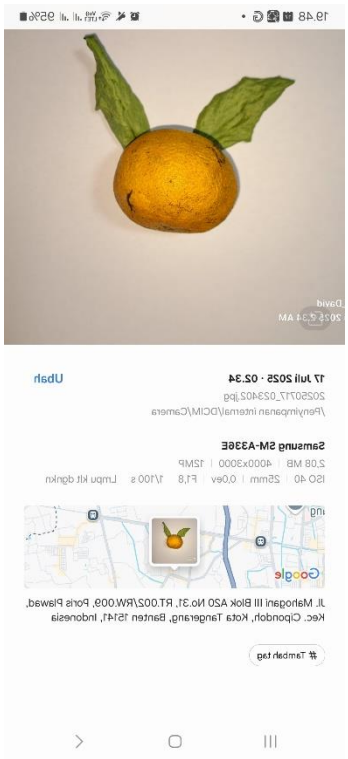
   - Lakukan kompresi lossy dengan kualitas 10% (quality=10) dari gambar asli. -
   - Lakukan kuantisasi warna RGB ke dalam 4 level per channel dari gambar
   - asli.

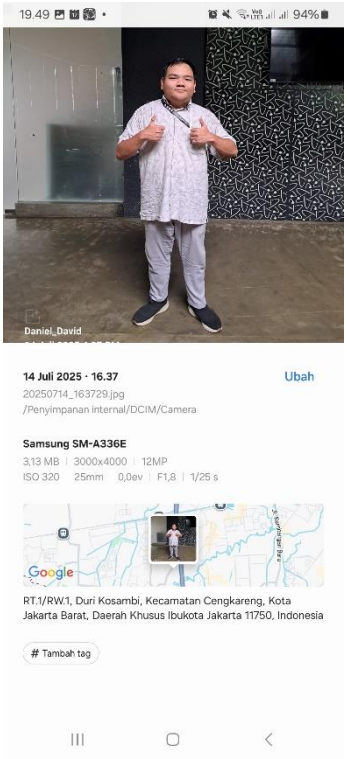
   Tampilkan hasil visual sebagai berikut:

   - Citra asli (RGB)
   - Citra hasil kompresi lossy (RGB)
   - Citra hasil kuantisasi warna (RGB)

   Tampilkan ketiganya dalam satu figure menggunakan matplotlib, masing-masing dengan judul dan ukuran file/in-memory-nya.

Bukti Gambar :





3.2 Filtering Citra

Filtering Citra

```
In [1]: import cv2
        import numpy as np
        import matplotlib.pyplot as plt
        #202331129 Daniel David Hamonangan Hutabarat
```

1. cv2 merupakan library yang menyediakan fungsi-fungsi tingkat tinggi untuk membaca, memanipulasi, dan menganalisis citra digital, seperti imread, cvtColor, dan medianBlur.
2. Numpy adalah library untuk komputasi numerik yang efisien.
3. matplotlib.pyplot adalah library untuk visualisasi data. Pustaka ini digunakan untuk menampilkan citra asli dan citra hasil filtering dalam sebuah jendela plot yang terstruktur, seperti yang terlihat pada output akhir.

```
In [2]: img_bgr = cv2.imread('gambar1.jpg')
        img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
        img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
        #202331129 Daniel David Hamonangan Hutabarat
```

4. cv2.imread('gambar1.jpg') berfungsi untuk membaca file citra gambar1.jpg dan memuatnya ke dalam memori sebagai sebuah array NumPy.
5. cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB) melakukan konversi warna dari BGR ke RGB

6. cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY) mengubah citra berwarna menjadi citra skala keabuan (grayscale).

Median filter

In [3]:
```
median_bgr = cv2.medianBlur(img_bgr, ksize=5)
median_rgb = cv2.cvtColor(median_bgr, cv2.COLOR_BGR2RGB)
#202331129 Daniel David Hamonangan Hutabarat
```

7. cv2.medianBlur(img_bgr, ksize=5) adalah fungsi yang menerapkan filter median pada img_bgr, dimana setiap piksel pada citra, nilai intensitasnya diganti dengan nilai median dari piksel-piksel tetangganya dalam sebuah kernel berukuran 5 x 5 Filter ini sangat efektif dalam mengurangi noise pada gambar
8. Hasil dari medianBlur kemudian dikonversi ke format RGB agar dapat ditampilkan dengan benar. Hasil akhirnya ditampilkan pada subplot After Filter Median.

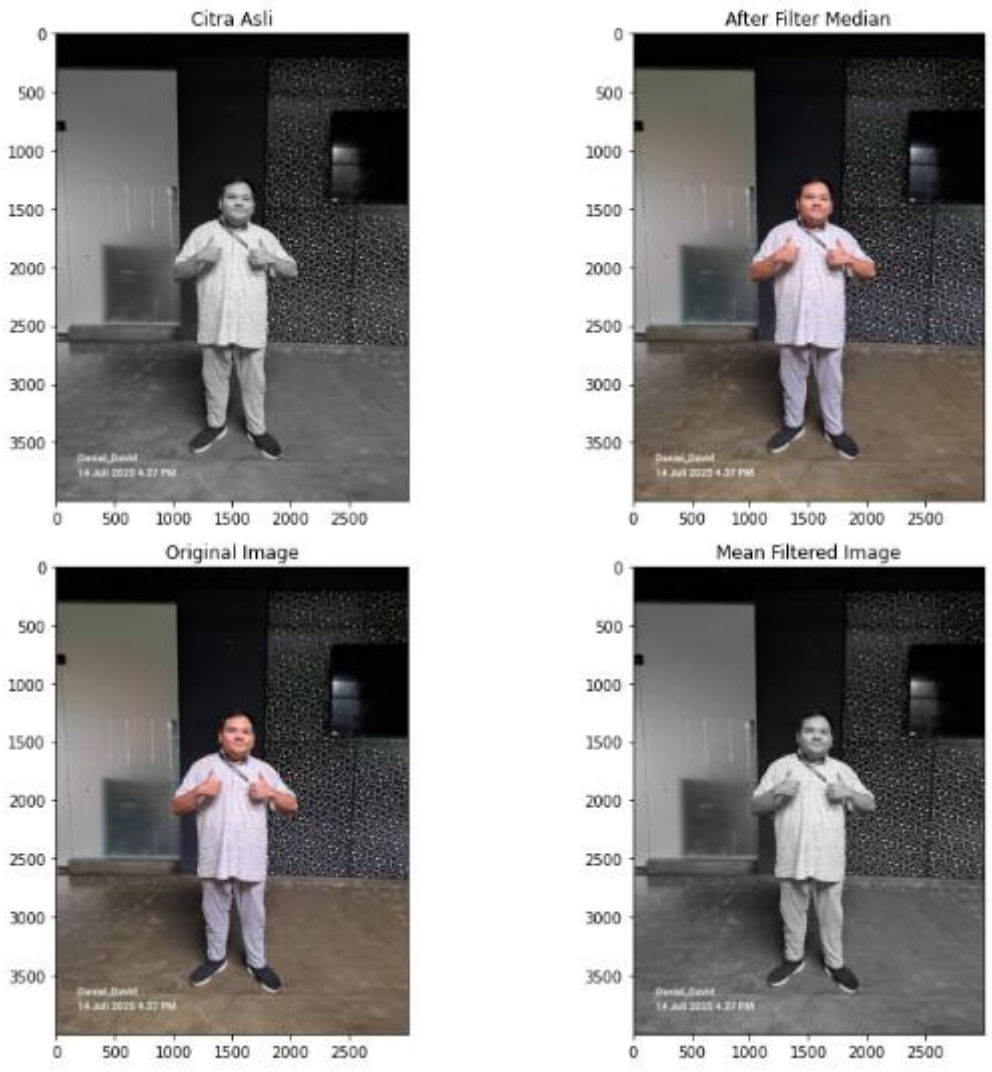
Mean filter

In [4]:
```
def wrap_reflect(image, pad):
    return np.pad(image, pad_width=pad, mode='reflect')


def meanFilter(originalImg, wrappedImg, kernelSize: int):
    filteredImage = np.zeros_like(originalImg, dtype=np.uint8)
    image_h, image_w = originalImg.shape
    w = kernelSize // 2
    area = kernelSize * kernelSize

    for i in range(w, image_h - w):
        for j in range(w, image_w - w):
            total = wrappedImg[i-w:i-w+kernelSize, j-w:j-w+kernelSize].sum()
            filteredImage[i, j] = total // area

    return filteredImage

#202331129 Daniel David Hamonangan Hutabarat
k = 5
pad = k // 2
wrapped = wrap_reflect(img_gray, pad)
mean_gray = meanFilter(img_gray, wrapped, k)
```

9. Fungsi wrap_reflect menangani masalah pemrosesan piksel di batas citra. Metode padding reflect menambahkan piksel di sekitar citra dengan mencerminkan nilai piksel dari tepi.
10. Fungsi meanFilter adalah implementasi inti dari mean filter.
11. kernel k=5, menghitung padding yang diperlukan, menerapkan padding ke citra grayscale, dan akhirnya memanggil meanFilter untuk memproses citra. Hasilnya
12. for i in range(w, image_h - w): dan for j in range(w, image_w - w): berfungsi untuk mengunjungi setiap piksel pada citra. Perulangan i berjalan untuk setiap baris, dan perulangan j berjalan untuk setiap kolom

```
In [5]: fig, axs = plt.subplots(2, 2, figsize=(12, 10))

        axs[0,0].imshow(img_gray, cmap='gray')
        axs[0,0].set_title("Citra Asli")

        axs[0,1].imshow(median_rgb)
        axs[0,1].set_title("After Filter Median")

        axs[1,0].imshow(img_rgb)
        axs[1,0].set_title("Original Image")

        axs[1,1].imshow(mean_gray, cmap='gray')
        axs[1,1].set_title("Mean Filtered Image")

        plt.tight_layout()
        plt.show()
        #202331129 Daniel David Hamonangan Hutabarat
```

13. bagian program ini menampilkan output dengan plot 2 x 2 yang akan menampilkan citra asli, citra hasil median, dan citra hasil filter mean



14. didapatkan hasil berupa :
    - Gambar ini dihasilkan oleh baris img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY). Proses ini menghilangkan informasi warna dan hanya menyisakan informasi intensitas cahaya untuk setiap piksel. Tujuannya adalah untuk menyederhanakan citra menjadi satu kanal, yang kemudian digunakan sebagai input untuk implementasi Mean Filter manual.
    - After Filter Median, Ini adalah hasil dari aplikasi Filter Median pada citra asli berwarna dengan ukuran kernel 5x5. Gambar ini adalah output dari cv2.medianBlur(img_bgr, ksize=5). Filter median bekerja dengan mengganti nilai setiap piksel dengan nilai tengah (median) dari piksel-piksel di sekitarnya. Hasilnya adalah citra yang terlihat lebih halus.
    - original image Gambar ini ditampilkan setelah dibaca oleh cv2.imread() dan dikonversi dari BGR ke RGB oleh cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB). Konversi ini penting agar matplotlib dapat menampilkan warna dengan benar
    - Mean Filtered Image, Ini adalah hasil dari aplikasi Filter Rata-rata manual pada citra grayscale dengan ukuran kernel 5x5. ini adalah output dari fungsi meanFilter yang telah dibuat. Filter ini mengganti nilai setiap piksel dengan nilai rata-rata (mean) dari piksel-piksel di sekitarnya. Efek dari perataan nilai ini adalah efek buram

3.3 Segmentasi

Segmentasi

In [14]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
#202331129 Daniel David Hamonangan Hutabarat
```

1. import cv2, Baris ini berfungsi untuk mengimpor pustaka OpenCV dengan mengimpor cv2, program mendapatkan akses ke sekumpulan besar fungsi dan algoritma tingkat tinggi untuk melakukan operasi seperti membaca file citra, melakukan transformasi ruang warna, menerapkan filter, mendeteksi objek
2. import numpy as np, Perintah ini mengimpor pustaka NumPy
3. import matplotlib.pyplot as plt, Baris ini mengimpor modul pyplot dari Matplotlib dengan alias plt. Matplotlib adalah sebuah library visualisasi di Python

In [15]:
```
img_bgr = cv2.imread('gambar3.jpg')
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
hsv_img = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)
#202331129 Daniel David Hamonangan Hutabarat
```

4. img_bgr = cv2.imread('gambar3.jpg') Baris program ini mengeksekusi fungsi imread() dari pustaka OpenCV untuk membaca data dari sebuah file citra yang bernama gambar3.jpg dan menyimpannya ke dalam variabel img_bgr

5. img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB), Perintah ini melakukan konversi BGR menjadi RGB
6. hsv_img = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV) Baris ini mengonversi citra dari ruang warna RGB ke HSV (Hue, Saturation, Value) HSV memisahkan komponen warna murni (Hue), tingkat kemurnian/kejenuhan warna (Saturation), dan tingkat kecerahan/intensitas cahaya (Value) menjadi kanal-kanal yang terpisah.

```
In [16]: lower_orange = np.array([10, 100, 100])
         upper_orange = np.array([25, 255, 255])
         mask_buah_awal = cv2.inRange(hsv_img, lower_orange, upper_orange)


         lower_green = np.array([30, 40, 40])
         upper_green = np.array([90, 255, 255])
         mask_daun_awal = cv2.inRange(hsv_img, lower_green, upper_green)
         #202331129 Daniel David Hamonangan Hutabarat
```

7. lower_orange = np.array([10, 100, 100]), upper_orange = np.array([25, 255, 255]), mask_buah_awal = cv2.inRange(hsv_img, lower_orange, upper_orange) kode ini bertujuan untuk mengisolasi semua piksel yang memiliki warna oranye. ruang warna HSV. Nilai [10, 100, 100] dan [25, 255, 255] merepresentasikan rentang untuk [Hue, Saturation, Value]. Hue 10-25 menargetkan spektrum warna oranye-kekuningan. Saturation 100-255 dan Value 100-255 memastikan bahwa hanya warna yang cukup jenuh dan terang yang akan dipilih, menghindari warna pudar atau gelap.
8. cv2.inRange(). Fungsi ini bekerja dengan cara memindai setiap piksel pada hsv_img. Jika nilai HSV dari sebuah piksel berada di dalam rentang yang didefinisikan oleh lower_orange dan upper_orange, maka piksel tersebut pada citra keluaran (mask_buah_awal) akan diberi nilai putih . Jika berada di luar rentang, akan diberi nilai hitam
9. lower_green = np.array([30, 40, 40]), upper_green = np.array([90, 255, 255]), mask_daun_awal = cv2.inRange(hsv_img, lower_green, upper_green), fungsi kode ini masih sama dengan sebelumnya bedanya ini untuk warna hijau.

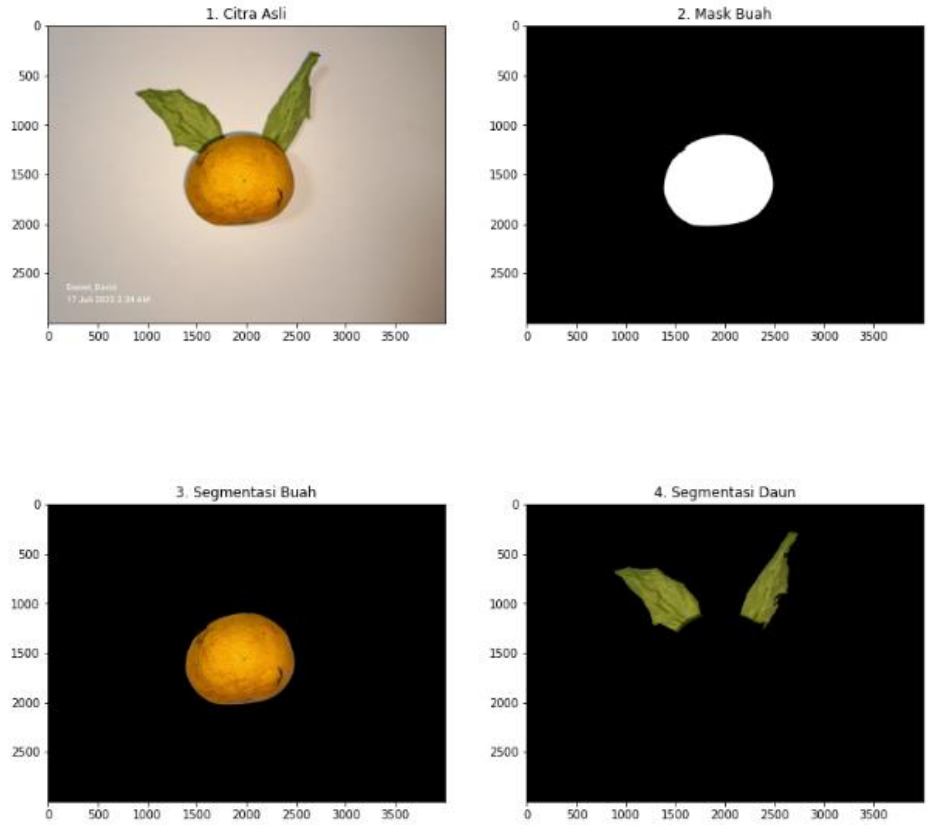
```
kernel = np.ones((5,5),np.uint8)
mask_buah_cleaned = cv2.morphologyEx(mask_buah_awal, cv2.MORPH_CLOSE, kernel)
mask_daun_cleaned = cv2.morphologyEx(mask_daun_awal, cv2.MORPH_CLOSE, kernel)



contours_buah, _ = cv2.findContours(mask_buah_cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours_daun, _ = cv2.findContours(mask_daun_cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

height, width, _ = img_rgb.shape
mask_buah_final = np.zeros((height, width), dtype=np.uint8)
mask_daun_final = np.zeros((height, width), dtype=np.uint8)


if contours_buah:
    fruit_contour = max(contours_buah, key=cv2.contourArea)
    cv2.drawContours(mask_buah_final, [fruit_contour], -1, (255), thickness=cv2.FILLED)


if contours_daun:
    cv2.drawContours(mask_daun_final, contours_daun, -1, (255), thickness=cv2.FILLED)


mask_daun_final = cv2.bitwise_and(mask_daun_final, cv2.bitwise_not(mask_buah_final))


segmentasi_buah = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_buah_final)
segmentasi_daun = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_daun_final)
#202331129 Daniel David Hamonangan Hutabarat
```

10. kernel = np.ones((5,5),np.uint8) Baris ini berfungsi untuk membuat sebuah kernel, yaitu sebuah matriks kecil yang digunakan dalam operasi morfologi. Fungsi. np.ones((5,5),np.uint8) dari NumPy membuat sebuah matriks berukuran 5x5 piksel yang semua elemennya bernilai 1.
11. mask_buah_cleaned = cv2.morphologyEx(mask_buah_awal, cv2.MORPH_CLOSE, kernel) Perintah ini menerapkan operasi morfologi yang disebut Closing pada masker buah awal (mask_buah_awal). Fungsi cv2.morphologyEx digunakan untuk operasi morfologi yang lebih kompleks.
12. mask_daun_cleaned = cv2.morphologyEx(mask_daun_awal, cv2.MORPH_CLOSE, kernel)
13. Baris ini melakukan operasi yang identik dengan baris sebelumnya, namun diterapkan pada mask daun awal (mask_daun_awal).
14. contours_buah, _ = cv2.findContours(mask_buah_cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) Fungsi cv2.findContours digunakan untuk mendeteksi dan mengekstrak semua garis batas dari objek putih pada mask_buah_cleaned.
15. contours_daun, _ = cv2.findContours(mask_daun_cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
16. Baris ini melakukan proses pencarian kontur yang sama persis, namun pada masker daun yang sudah dibersihkan (mask_daun_cleaned)
17. height, width, _ = img_rgb.shape Perintah ini mengambil dimensi dari citra asli berwarna (img_rgb). img_rgb.shape mengembalikan sebuah tuple berisi. Nilai tinggi dan lebar ini kemudian disimpan dalam variabel height dan width
18. mask_buah_final = np.zeros((height, width), dtype=np.uint8) dan mask_daun_final = np.zeros((height, width), dtype=np.uint8), Baris ini menciptakan sebuah gambar hitam kosong dengan dimensi yang sama persis dengan citra asli. Fungsi

19. if contours_buah: kode ini memeriksa apakah list contours_buah berisi sesuatu. Jika tidak ada kontur yang terdeteksi, list tersebut akan kosong dan blok kode di bawahnya akan dilewati, mencegah terjadinya error.
20. fruit_contour = max(contours_buah, key=cv2.contourArea), Jika kontur ditemukan, baris ini akan mencari satu kontur dengan area terbesar. Fungsi max() digunakan pada list contours_buah dengan key=cv2.contourArea, yang berarti kriteria untuk mencari nilai "maksimum" adalah luas area dari setiap kontur.
21. mask_daun_final = cv2.bitwise_and(mask_daun_final, cv2.bitwise_not(mask_buah_final)) cv2.bitwise_not(mask_buah_final) membalik masker buah (area buah menjadi hitam, sisanya putih). Kemudian, cv2.bitwise_and dilakukan antara masker daun dengan masker buah yang terbalik. Hasilnya adalah area masker daun akan tetap ada hanya jika area tersebut tidak tumpang tindih dengan area buah, secara efektif "memotong" bagian daun yang mungkin overlap dengan buah.

```
In [18]: fig, axs = plt.subplots(2, 2, figsize=(12, 12))
         fig.tight_layout(pad=4.0)


         axs[0, 0].imshow(img_rgb)
         axs[0, 0].set_title('1. Citra Asli')

         axs[0, 1].imshow(mask_buah_final, cmap='gray')
         axs[0, 1].set_title('2. Mask Buah')

         axs[1, 0].imshow(segmentasi_buah)
         axs[1, 0].set_title('3. Segmentasi Buah')

         axs[1, 1].imshow(segmentasi_daun)
         axs[1, 1].set_title('4. Segmentasi Daun')

         plt.show()
         #202331129 Daniel David Hamonangan Hutabarat
```

22. kode ini untuk menampilkan output dari program berupa citra asli, citra mask buah, citra segmentasi buah dan citra segmentasi daun.



23. Citra Asli, Gambar ini adalah hasil dari dua baris kode awal: cv2.imread() yang memuat gambar, dan cv2.cvtColor() yang mengubah format warnanya menjadi RGB.
24. Mask Buah, Gambar ini tidak menampilkan warna asli, melainkan hanya informasi spasial—area putih menunjukkan lokasi buah, dan area hitam menunjukkan segala sesuatu yang bukan buah.
25. Segmentasi Buah, Gambar ini menampilkan hasil akhir dari segmentasi buah, di mana objek buah berhasil diisolasi dari lingkungannya. Output ini dihasilkan oleh operasi cv2.bitwise_and(img_rgb, img_rgb, mask=mask_buah_final).
26. Segmentasi Daun, Gambar ini menunjukkan objek daun yang telah berhasil dipisahkan dari buah dan latar belakang menggunakan operasi cv2.bitwise_and, namun dengan mask_daun_final sebagai acuannya.

3.4 Kompresi

Kompresi

```
In [9]: import cv2
        import numpy as np
        import matplotlib.pyplot as plt
        import os
        #202331129 Daniel David Hamonangan Hutabarat
```

1. import cv2, import librart OpenCV fungsinya untuk semua tugas yang berhubungan dengan pemrosesan gambar.
2. import numpy as np, Perintah ini mengimpor pustaka NumPy
3. import matplotlib.pyplot as plt Baris ini mengimpor pustaka Matplotlib. Kegunaannya adalah untuk menampilkan gambar-gambar hasil olahan di layar dalam sebuah jendela plot, sehingga hasilnya bisa dibandingkan secara visual.

4. import os berfungsi untuk berinteraksi dengan sistem operasi komputer.

```
In [10]: nama_file_asli = 'gambar2.jpg'
         img_bgr = cv2.imread(nama_file_asli)
         img_asli_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
         ukuran_asli_in_memory = img_asli_rgb.nbytes
         #202331129 Daniel David Hamonangan Hutabarat
```

5. nama_file_asli = 'gambar2.jpg' , Baris ini berfungsi untuk menyimpan nama file gamba
6. img_bgr = cv2.imread(nama_file_asli) bagian ini membaca file gambar yang namanya telah disimpan di nama_file_asli
7. img_asli_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB) Baris ini berfungsi untuk mengubah susunan warna gambar. Fungsi cvtColor mengubah format dari BGR menjadi RGB.
8. ukuran_asli_in_memory = img_asli_rgb.nbytes, Perintah ini menghitung ukuran total data gambar mentah dalam satuan byte saat berada di memori.

```
In [11]: nama_file_lossy = 'gambar_lossy_q10.jpg'
         kualitas_jpeg = 10

         cv2.imwrite(nama_file_lossy, img_bgr, [cv2.IMWRITE_JPEG_QUALITY, kualitas_jpeg])

         ukuran_file_lossy = os.path.getsize(nama_file_lossy)

         img_lossy_bgr = cv2.imread(nama_file_lossy)
         img_lossy_rgb = cv2.cvtColor(img_lossy_bgr, cv2.COLOR_BGR2RGB)
         #202331129 Daniel David Hamonangan Hutabarat
```

9. nama_file_lossy = 'gambar_lossy_q10.jpg', Baris ini berfungsi untuk menentukan dan menyimpan nama file untuk gambar yang akan dihasilkan dari proses kompresi. Dengan menyimpannya dalam variabel nama_file_lossy.
10. kualitas_jpeg = 10, Perintah ini menetapkan nilai untuk tingkat kualitas kompresi JPEG
11. cv2.imwrite(nama_file_lossy, img_bgr, [cv2.IMWRITE_JPEG_QUALITY, kualitas_jpeg]) Fungsi cv2.imwrite menyimpan data gambar ke sebuah file baru dengan nama yang sudah ditentukan. Parameter [cv2.IMWRITE_JPEG_QUALITY, kualitas_jpeg]memerintahkan fungsi untuk menggunakan algoritma kompresi JPEG dengan tingkat kualitas 10.
12. ukuran_file_lossy = os.path.getsize(nama_file_lossy), menggunakan fungsi os.path.getsize untuk memeriksa ukuran file tersebut di dalam hard disk.
13. img_lossy_bgr = cv2.imread(nama_file_lossy), Fungsi ini membaca kembali file gambar yang baru saja dikompresi
14. img_lossy_rgb = cv2.cvtColor(img_lossy_bgr, cv2.COLOR_BGR2RGB), baris ini mengubah format warna gambar hasil kompresi dari BGR ke RGB.

```
In [12]: jumlah_level = 4

         bin_size = 256 / jumlah_level
         img_kuantisasi_rgb = (np.floor(img_asli_rgb.astype(float) / bin_size) * bin_size

         ukuran_kuantisasi_in_memory = img_kuantisasi_rgb.nbytes
         #202331129 Daniel David Hamonangan Hutabarat
```

```
In [12]:
         umlah_level
         = (np.floor(img_asli_rgb.astype(float) / bin_size) * bin_size).astype(np.uint8)

         in_memory = img_kuantisasi_rgb.nbytes
         David Hamonangan Hutabarat
```

15. jumlah_level = 4, Baris ini berfungsi untuk menetapkan jumlah tingkatan warna yang akan digunakan.
16. bin_size = 256 / jumlah_level, Perintah ini berfungsi untuk menghitung ukuran dari setiap "kelompok" atau bin warna. Logikanya, karena ada 256 kemungkinan nilai warna setiap kelompok akan memiliki rentang sebesar 256 / 4 = 64. Jadi, warna akan dikelompokkan menjadi 0-63, 64-127, 128-191, dan 192-255.
17. img_kuantisasi_rgb = (np.floor(img_asli_rgb.astype(float) / bin_size) * bin_size).astype(np.uint8), baris inti yang melakukan proses kuantisasi warna, mengubah setiap nilai warna piksel asli menjadi salah satu dari 4 level yang telah ditentukan. Prosesnya adalah: membagi nilai piksel dengan bin_size (64), membulatkannya ke bawah, lalu mengalikannya kembali dengan bin_size. Contohnya, nilai piksel 130 akan menjadi floor(130/64) * 64 -> floor(2.03) * 64 -> 2 * 64 = 128. Ini secara drastis mengurangi variasi warna pada gambar.
18. ukuran_kuantisasi_in_memory = img_kuantisasi_rgb.nbytes , Baris ini berfungsi untuk menghitung ukuran data gambar hasil kuantisasi di dalam memori.

```
In [14]: fig, axes = plt.subplots(1, 3, figsize=(18, 6))

         axes[0].imshow(img_asli_rgb)
         axes[0].set_title(f"Asli\nUkuran: {ukuran_asli_in_memory / 1024:.2f} KB (In-Memo
         axes[0].axis('off')

         axes[1].imshow(img_lossy_rgb)
         axes[1].set_title(f"Lossy JPEG Q={kualitas_jpeg}\nUkuran: {ukuran_file_lossy / 1
         axes[1].axis('off')

         axes[2].imshow(img_kuantisasi_rgb)
         axes[2].set_title(f"Kuantisasi RGB ({jumlah_level} Level)\nUkuran: {ukuran_kuant
         axes[2].axis('off')

         plt.tight_layout()
         plt.show()

         #202331129 Daniel David Hamonangan Hutabarat
```

```
In [14]: 6))

         li_in_memory / 1024:.2f} KB (In-Memory)")


         peg}\nUkuran: {ukuran_file_lossy / 1024:.2f} KB")


         level} Level)\nUkuran: {ukuran_kuantisasi_in_memory / 1024:.2f} KB (In-Memory)")
```

19. bagian ini merupakan kode pada program yang digunakan untuk menampilkan output dari program

20. Asli, Ini adalah gambar asli tanpa perubahan ukuran: 25520.06 KB menunjukkan ukuran data gambar mentah saat berada di memori bukan ukuran file di hard disk. Ukuran ini dihitung oleh img_asli_rgb.nbytes dan merepresentasikan ukuran gambar jika setiap piksel disimpan tanpa kompresi sama sekali.
21. Lossy JPEG Q=10, Ukuran: 187.74 KB adalah ukuran file sebenarnya di hard disk setelah disimpan dengan cv2.imwrite pada kualitas JPEG 10. Program membuang sebagian data visual yang dianggap tidak terlalu penting oleh mata manusia untuk mencapai pengurangan ukuran file yang drastis

22. Kuantisasi RGB (4 Level), Ukuran: 25520.06 KB (In-Memory) menunjukkan ukurannya di memori sama dengan gambar asli. untuk disederhanakan menjadi hanya 4 level warna saja.

BAB IV
PENUTUP

Pengolahan citra digital mencakup berbagai teknik dasar yang berperan penting dalam proses ekstraksi informasi visual secara efisien dan akurat. Tiga konsep fundamental yang menjadi fokus dalam praktikum ini meliputi modifikasi citra melalui filtering spasial, segmentasi objek, serta kompresi citra untuk efisiensi data. Filtering spasial diterapkan dengan menggunakan filter Median dan Mean untuk mengurangi gangguan berupa noise dan menghasilkan efek penghalusan, sehingga kualitas visual citra meningkat sebelum memasuki tahap analisis lebih lanjut. Teknik ini mencerminkan bagaimana manipulasi piksel secara lokal dapat berdampak signifikan terhadap struktur global citra. Proses segmentasi, sebagai tahap kedua, menunjukkan peran vital dalam mengidentifikasi dan mengisolasi objek spesifik dari latar belakang berdasarkan karakteristik warna atau intensitas. Pendekatan thresholding dan pemanfaatan ruang warna HSV memungkinkan pemisahan objek seperti buah dan daun secara lebih akurat, yang sangat berguna dalam aplikasi seperti pengenalan objek, pelacakan, dan klasifikasi citra. Segmentasi tidak hanya menyederhanakan representasi citra, tetapi juga memfasilitasi proses analisis yang lebih terstruktur dan sistematis. Tahap akhir dari praktikum ini menekankan pentingnya kompresi citra, baik melalui metode lossy seperti JPEG yang mengurangi ukuran file dengan menghilangkan sebagian informasi visual, maupun melalui kuantisasi warna yang menyederhanakan palet warna guna mengurangi kompleksitas data. Kompresi menjadi aspek krusial dalam penghematan ruang penyimpanan dan percepatan transmisi data, terutama dalam sistem yang melibatkan citra dalam jumlah besar atau resolusi tinggi. Ketiga proses ini—modifikasi, segmentasi, dan kompresi—mewakili fondasi penting dalam pengolahan citra digital modern. Sinergi antara teori dan implementasi program dalam praktikum ini memperlihatkan keterkaitan erat antara teknik dasar dan aplikasi lanjut dalam berbagai bidang, seperti analisis citra medis, pertanian cerdas, pengawasan visual, hingga sistem kendaraan otonom berbasis visi komputer.

DAFTAR PUSTAKA

- Medinah, D. R. E., & Sinurat, S. (2020). Analisa dan Perbandingan Algoritma Otsu Thresholding dengan Algoritma Region Growing Pada Segmentasi Citra Digital. Journal of Computer System and Informatics (JoSYC), 2(1), 9-16.
- Mohanty, S. S., & Tripathy, S. (2021). Application of Different Filtering Techniques in Digital Image Processing. Journal of Physics: Conference Series, 2062(1), 012007. https://doi.org/10.1088/1742-6596/2062/1/012007
- Pangesti, W. E., Widagdo, G., Riana, D., & Hadianti, S. (2020). Implementasi Kompresi Citra Digital Dengan Membandingkan Metode Lossy Dan Lossless Compression Menggunakan Matlab. Jurnal Khatulistiwa Informatika, 8(1), 53-58.
- Pratama, E. F. A., Khairil, & Jumadi, J. (2022). Implementasi Metode K-Means Clustering Pada Segmentasi Citra Digital. Jurnal Media Infotama, 18(2), 291-301.
- Putra, A. B. W., Aryuna, M. T., & Malani, R. (2021). Kompresi Citra Digital Dengan Basis Komponen Warna RGB Menggunakan Metode K-Means Clustering. Jurnal Komputer Terapan, 7(1), 14-23.
- Sumardiyono, B. (2022). Segmentasi Citra Digital Paleografi Arsip VOC Menggunakan Metode Thresholding. Jurnal Rekayasa Informasi, 11(1), 17-23.
- Yan, K., Chen, Y., & Zhang, D. (2014). Gabor Surface Feature for Face Recognition. [Preprint]. arXiv. https://arxiv.org/abs/1409.2413