UTS
PENGOLAHAN CITRA DIGITAL

Nama	: Daffa Laksmana Naufal
NIM	202331090
Kelas	: D
Nama Dosen	: Ir. Darma Rusjdi, M.kom
No. Bangku	21
Nama Asisten	:

1. Davina Najwa Ermawan
2. Fakhrul Fauzi Nugraha Tarigan
3. Viana Salsabila Fairuz Syahla
4. Muhammad Hanief Febriansyah

LABORATORIUM EMBEDDED SYSTEM
INSTITUT TEKNOLOGI PLN
2024

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Rumusan Masalah

Pada Ujian Tengah Semester (UTS) Praktikum Pengolahan Citra Digital ini, rumusan masalah yang dihadapi adalah bagaimana mengimplementasikan dan menganalisis beberapa teknik dasar pengolahan citra menggunakan bahasa pemrograman Python. Secara khusus, mahasiswa ditugaskan untuk melakukan pengolahan gambar dalam format grayscale, meningkatkan kualitas citra dengan metode tertentu, serta menerapkan operasi spasial seperti deteksi tepi dan filtering. Permasalahan lainnya mencakup kemampuan mahasiswa dalam memahami serta memvisualisasikan perubahan citra secara matematis dan visual.

1.2 Tujuan Masalah

Tujuan dari praktikum UTS ini adalah untuk menguji kemampuan mahasiswa dalam menerapkan konsep-konsep dasar pengolahan citra digital seperti konversi ke grayscale, histogram, transformasi logaritmik, perbaikan kontras, serta filtering menggunakan metode mean, gaussian, dan edge detection (Sobel dan Prewitt). Selain itu, mahasiswa juga diharapkan mampu mengevaluasi hasil pengolahan tersebut melalui visualisasi citra dan grafik, serta memahami perbedaan antar metode yang digunakan.

1.3 Manfaat Masalah

Manfaat yang diperoleh dari pelaksanaan UTS praktikum ini antara lain adalah meningkatkan pemahaman mahasiswa dalam bidang pengolahan citra digital secara praktis dan teoritis. Melalui tugas ini, mahasiswa dapat mengenali peran penting teknik pengolahan citra dalam berbagai aplikasi dunia nyata seperti pengenalan pola, analisis medis, dan sistem pengawasan visual. Selain itu, mahasiswa juga dapat mengasah kemampuan pemrograman serta berpikir kritis dalam menyelesaikan permasalahan citra digital secara terstruktur.

BAB II
LANDASAN TEORI

2.1 Deteksi Warna dalam Citra (Color Detection)

Deteksi warna dilakukan untuk mengekstraksi bagian tertentu dari citra berdasarkan nilai warna. Salah satu pendekatan umum adalah dengan mengubah representasi warna dari BGR ke HSV (Hue, Saturation, Value), karena HSV lebih sesuai untuk segmentasi warna dibandingkan ruang BGR atau RGB.

Hue: Menentukan jenis warna (merah, hijau, biru, dll.).

Saturation: Menentukan kejenuhan warna.

Value (Brightness): Menentukan tingkat terang warna.

Dengan menentukan rentang nilai HSV (lower_bound dan upper_bound), kita bisa mendeteksi piksel-piksel yang memiliki warna tertentu menggunakan fungsi cv2.inRange().

2.2 Ambang Batas Warna (Thresholding)

Thresholding digunakan untuk mengubah citra menjadi biner berdasarkan kriteria tertentu, misalnya intensitas atau nilai warna tertentu.

- Ambang terkecil dan terbesar dalam konteks warna biasanya merujuk pada nilai Hue minimum hingga Hue maksimum yang dapat mencakup seluruh variasi warna dalam gambar.
- Setelah didapatkan batas HSV minimum dan maksimum dari semua piksel, nilai-nilai tersebut bisa diurutkan untuk memahami sebaran warna dari yang paling rendah ke yang tertinggi.

2.3 Konversi ke Grayscale

Grayscale adalah representasi citra dalam skala abu-abu (0–255), tanpa informasi warna. Konversi dilakukan dengan mengekstrak luminance (kecerahan) dari warna.

OpenCV melakukannya dengan rumus:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Fungsi cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) digunakan untuk mengubah citra berwarna menjadi grayscale.

2.4 Peningkatan Kecerahan (Brightness Enhancement)

Brightness ditingkatkan dengan **menambahkan nilai tertentu ke semua piksel** citra grayscale. Fungsi cv2.add() biasanya digunakan agar tidak terjadi overflow.

$$I'(x, y) = I(x, y) + \beta$$

- Nilai β adalah offset kecerahan (misalnya 50).
- Penambahan ini membuat citra tampak lebih terang.

2.5 Peningkatan Kontras (Contrast Enhancement)

Peningkatan kontras bertujuan untuk menyebarkan nilai intensitas agar detail lebih terlihat. Salah satu teknik paling umum adalah **Histogram Equalization**.

cv2.equalizeHist(gray_image)

Prinsip histogram equalization:

- Histogram disebarkan secara merata ke seluruh rentang intensitas 0–255.
- Membantu menonjolkan perbedaan antara objek terang dan gelap.

2.6 Peningkatan Brightness dan Kontras secara Bertahap

Langkah-langkah pemrosesan dilakukan berurutan untuk menunjukkan efek masing-masing metode:

- Citra grayscale biasa.
- Grayscale + brightness.
- Grayscale + contrast.
- Grayscale + brightness + contrast.

Hal ini bertujuan untuk:

- Menilai pengaruh masing-masing metode secara visual.
- Memahami bagaimana distribusi intensitas berubah (melalui histogram).

Histogram

Histogram citra menunjukkan **jumlah piksel untuk setiap level intensitas (0–255)**. Dalam analisis citra, histogram berguna untuk:

- Melihat kontras citra.
- Menilai efek pemrosesan seperti histogram equalization.
- Menentukan sebaran warna atau kecerahan.

Ringkasan Proses:

1. Deteksi warna menggunakan HSV.
2. Tentukan batas ambang warna terkecil dan terbesar → urutkan.
3. Ubah gambar ke grayscale.
4. Tambah brightness untuk citra grayscale.
5. Equalize histogram untuk menaikkan kontras.
6. Kombinasikan brightness + kontras → hasil paling optimal secara visual.

BAB III
PEMBAHASAN

3.1 Penjelasan:

Mengimpor pustaka yang dibutuhkan:

- cv2: OpenCV untuk pengolahan citra.
- numpy: Untuk manipulasi array.
- matplotlib.pyplot: Untuk menampilkan gambar.

Penjelasan:

- Membaca gambar (Nama.jpg) dalam format BGR (standar OpenCV).
- Mengubah gambar ke format RGB agar sesuai dengan tampilan matplotlib.
- Konversi ke HSV (Hue, Saturation, Value) untuk deteksi warna lebih stabil.

Penjelasan:

- Menentukan **batas bawah dan atas HSV** untuk deteksi warna merah, hijau, dan biru.
- Untuk merah, digunakan dua rentang karena warna merah berada di ujung awal dan akhir hue pada HSV (0–10 dan 160–180).
- cv2.inRange() menghasilkan mask biner (putih untuk piksel yang cocok).
- cv2.bitwise_or() menggabungkan dua mask merah.

```
[1]: import cv2
     import numpy as np
     import matplotlib.pyplot as plt

[2]: img = cv2.imread("Nama.jpg")
     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

[3]: # Merah
     lower_red1 = np.array([0, 100, 100])
     upper_red1 = np.array([10, 255, 255])
     lower_red2 = np.array([160, 100, 100])
     upper_red2 = np.array([180, 255, 255])
     mask_red = cv2.bitwise_or(cv2.inRange(hsv, lower_red1, upper_red1),
                               cv2.inRange(hsv, lower_red2, upper_red2))

     # Hijau
     lower_green = np.array([40, 100, 100])
     upper_green = np.array([80, 255, 255])
     mask_green = cv2.inRange(hsv, lower_green, upper_green)

     # Biru
     lower_blue = np.array([100, 100, 100])
     upper_blue = np.array([140, 255, 255])
     mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
```

3.2 Penjelasan Fungsi:

Fungsi ini digunakan untuk menyoroti teks yang tidak berwarna target, namun tidak mengganggu background putih.

Langkah-langkahnya:

Membuat salinan gambar berwarna putih (result) sebagai canvas.

white_bg_mask mendeteksi piksel yang "cukup putih" (nilai R, G, B ≥ 180) – dianggap background.

text_mask menentukan piksel yang bukan warna target (mask_color == 0) dan bukan background putih.

Piksel yang lolos seleksi text_mask diwarnai hitam ([0, 0, 0]).

Mengembalikan hasil akhir berupa gambar dengan latar putih dan teks non-target jadi hitam.

Penjelasan:

- Menyimpan hasil fungsi highlight_non_color_text() untuk tiap warna (biru, merah, hijau).
- mask_blue, mask_red, dan mask_green adalah mask biner hasil threshold HSV pada cell sebelumnya.
- Output-nya berupa gambar teks berwarna hitam di atas latar putih untuk masing-masing warna non-target.

Penjelasan:

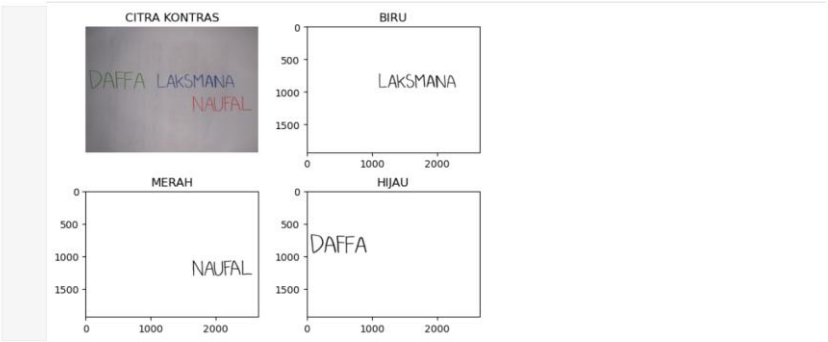
- Membuat figure kosong dengan ukuran 12x8 inci, sebagai persiapan untuk plotting gambar hasil highlight warna (biasanya akan dilanjutkan dengan plt.subplot() dan plt.imshow() di cell berikutnya).

```
[4]: # Fungsi perbaikan dengan deteksi latar putih menggunakan threshold
def highlight_non_color_text(mask_color):
    result = np.ones_like(img_rgb) * 255  # background putih

    # Deteksi background putih "cukup terang" (bukan teks)
    white_bg_mask = np.all(img_rgb >= [180, 180, 180], axis=-1)

    # Teks non-target adalah piksel yang:
    # - Tidak terdeteksi warna target (mask_color == 0)
    # - Bukan background putih
    text_mask = np.logical_and(mask_color == 0, ~white_bg_mask)

    # Buat teks jadi hitam
    result[text_mask] = [0, 0, 0]

    return result

[5]: # Gambar hasil
highlight_blue = highlight_non_color_text(mask_blue)
highlight_red = highlight_non_color_text(mask_red)
highlight_green = highlight_non_color_text(mask_green)

[6]: # Tampilkan hasil dalam layout 2x2
plt.figure(figsize=(12, 8))

[6]: <Figure size 1200x800 with 0 Axes>
     <Figure size 1200x800 with 0 Axes>
```

3.3 Penjelasan Fungsi :

- Menampilkan gambar asli (img_rgb) pada posisi subplot pertama (baris 1, kolom 1).
  - Judul gambar: "CITRA KONTRAS".
  - plt.axis('off'): menyembunyikan sumbu untuk tampilan lebih bersih.
  - Menampilkan hasil highlight_blue (yakni teks non-biru).
  - Posisi subplot kedua (baris 1, kolom 2).
  - Sumbu tetap ditampilkan (axis('on')) untuk referensi piksel.
  - Menampilkan hasil `highlight_red`, yaitu teks non-merah.
  - Ditempatkan di subplot ketiga (baris 2, kolom 1).
  - Menampilkan `highlight_green` (teks non-hijau).
  - Posisi subplot keempat (baris 2, kolom 2).
  - `plt.tight_layout()` mengatur tata letak agar **semua subplot tidak saling bertumpuk** dan lebih rapi.
  - `plt.show()` menampilkan semua gambar di figure yang telah dibuat.

```
[7]: # Citra asli
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title("CITRA KONTRAS")
plt.axis('off')

# Biru
plt.subplot(2, 2, 2)
plt.imshow(highlight_blue)
plt.title("BIRU")
plt.axis('on')

# Merah
plt.subplot(2, 2, 3)
plt.imshow(highlight_red)
plt.title("MERAH")
plt.axis('on')

# Hijau
plt.subplot(2, 2, 4)
plt.imshow(highlight_green)
plt.title("HIJAU")
plt.axis('on')

plt.tight_layout()
plt.show()
```



3.4 Penjelasan Fungsi

- cv2.imread: Membaca gambar 'NamaDiri.jpg' dalam format BGR (default OpenCV).
- cv2.cvtColor:
- Mengubah gambar menjadi format RGB untuk keperluan visualisasi.
- Mengubah gambar menjadi format HSV untuk pemrosesan warna (karena HSV lebih stabil untuk segmentasi warna dibanding RGB).

Fungsi ini bertujuan untuk:

1. Membuat mask berdasarkan rentang HSV (cv2.inRange).
2. Menampilkan area gambar yang termasuk mask menggunakan operasi bitwise_and.
3. Konversi ke grayscale lalu lakukan thresholding untuk mendapatkan hasil biner (hitam-putih).
    - Nilai piksel di atas 10 dijadikan 255 (putih), lainnya 0 (hitam).

Output: gambar biner yang menandai area dengan warna tertentu.

- Menentukan rentang HSV untuk 3 warna utama:
- Biru
- Merah
- Hijau
- Format: [Hue_min, Saturation_min, Value_min], [Hue_max, Saturation_max, Value_max].

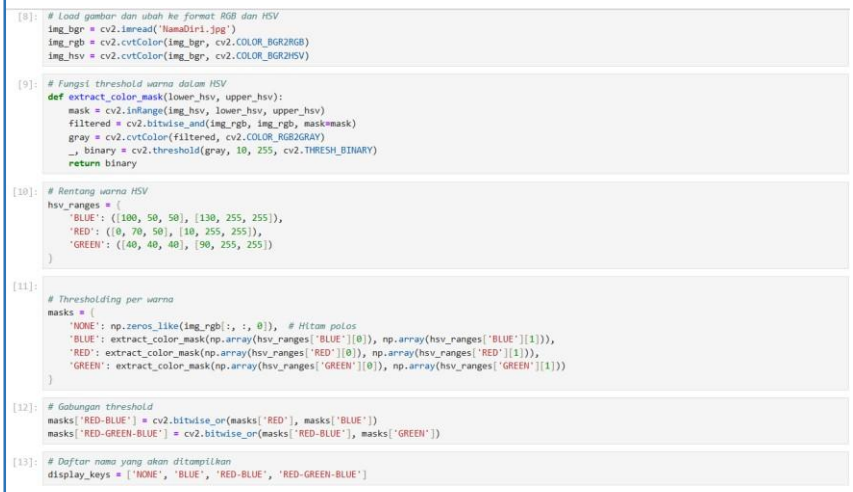
- Membuat dictionary masks yang menyimpan **hasil threshold** untuk masing-masing warna.
- 'NONE': mask kosong (seluruhnya hitam), untuk perbandingan awal.
- extract_color_mask: fungsi yang dipanggil untuk masing-masing warna.

Menggabungkan hasil threshold:

- 'RED-BLUE': menggabungkan area merah dan biru.
- 'RED-GREEN-BLUE': menambahkan hijau ke hasil sebelumnya.
- cv2.bitwise_or: operasi logika "OR" piksel-per-piksel (jika salah satu putih → hasilnya putih).

Menyimpan **urutan kunci** yang ingin ditampilkan nanti pada bagian visualisasi (misalnya subplot).

- Berguna untuk loop agar bisa menampilkan hasil setiap langkah threshold.

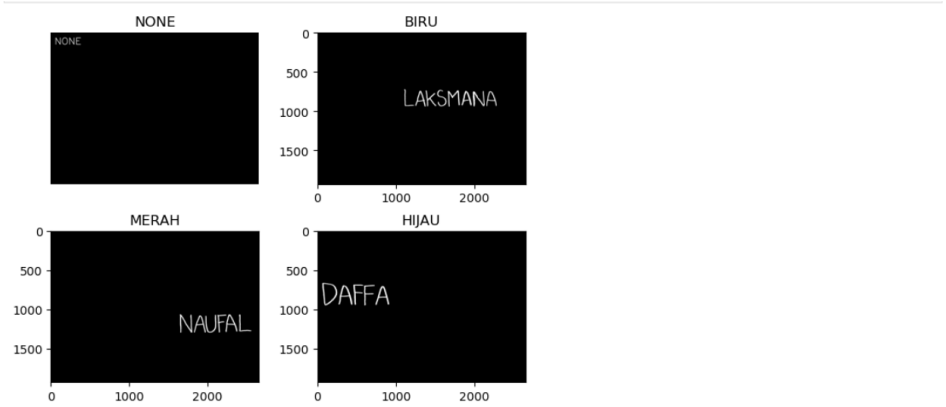


```
[8]: # Load gambar dan ubah ke format RGB dan HSV
img_bgr = cv2.imread('NamaDir1.jpg')
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)

[9]: # Fungsi threshold warna dalam HSV
def extract_color_mask(lower_hsv, upper_hsv):
    mask = cv2.inRange(img_hsv, lower_hsv, upper_hsv)
    filtered = cv2.bitwise_and(img_rgb, img_rgb, mask=mask)
    gray = cv2.cvtColor(filtered, cv2.COLOR_RGB2GRAY)
    _, binary = cv2.threshold(gray, 10, 255, cv2.THRESH_BINARY)
    return binary

[10]: # Rentang warna HSV
hsv_ranges = {
    'BLUE': ([100, 50, 50], [130, 255, 255]),
    'RED': ([0, 70, 50], [10, 255, 255]),
    'GREEN': ([40, 40, 40], [90, 255, 255])
}

[11]:
# Thresholding per warna
masks = {
    'NONE': np.zeros_like(img_rgb[:, :, 0]),  # Hitam polos
    'BLUE': extract_color_mask(np.array(hsv_ranges['BLUE'][0]), np.array(hsv_ranges['BLUE'][1])),
    'RED': extract_color_mask(np.array(hsv_ranges['RED'][0]), np.array(hsv_ranges['RED'][1])),
    'GREEN': extract_color_mask(np.array(hsv_ranges['GREEN'][0]), np.array(hsv_ranges['GREEN'][1]))
}

[12]: # Gabungan threshold
masks['RED-BLUE'] = cv2.bitwise_or(masks['RED'], masks['BLUE'])
masks['RED-GREEN-BLUE'] = cv2.bitwise_or(masks['RED-BLUE'], masks['GREEN'])

[13]: # Daftar nama yang akan ditampilkan
display_keys = ['NONE', 'BLUE', 'RED-BLUE', 'RED-GREEN-BLUE']
```

```
[14]: # Plot hasil 2x2
plt.figure(figsize=(12, 8))
for idx, key in enumerate(display_keys):
    plt.subplot(2, 2, idx + 1)
    plt.imshow(masks[key], cmap='gray')
    plt.title(key)
    plt.axis('off')
plt.tight_layout()
plt.show()
```

- Membuat figure canvas baru dengan ukuran 12x8 inch.
- Ukuran ini mengatur besar tampilan agar hasil subplot tidak terlalu kecil.
- Melakukan loop untuk setiap elemen dalam list `display_keys`.
- `enumerate` memberi index (`idx`) dan nama mask (`key`) secara bersamaan.
- Membuat subplot dalam grid 2 baris x 2 kolom.
- idx + 1: posisi subplot ke-n (karena posisi subplot dimulai dari 1, bukan 0).
  - Misalnya: idx=0 → subplot(2,2,1), idx=1 → subplot(2,2,2), dst.
- Menampilkan gambar **mask** yang dipilih.
- `cmap='gray'`: menggunakan colormap abu-abu, karena citra mask bersifat biner (hitam-putih).
- Memberi **judul** pada subplot sesuai nama mask, yaitu 'NONE', 'BLUE', 'RED-BLUE', dan 'RED-GREEN-BLUE'.

- Menyembunyikan sumbu x dan y agar tampilan bersih.
- Mengatur layout antar subplot agar tidak saling tumpang tindih (rapih otomatis).
- Menampilkan semua subplot yang telah dibuat.



Ini adalah output nya.

- Membuat **figure baru** untuk menampilkan plot histogram.
- Ukurannya 10x5 inch, cukup lebar untuk menampilkan beberapa kurva.
- Melakukan loop untuk setiap nama mask dalam `display_keys`, **kecuali** yang pertama (`'NONE'`).
- `'NONE'` adalah mask hitam polos, tidak ada nilai 255, jadi tidak berguna untuk histogram.
- Menghitung histogram dari **mask biner**.
- `cv2.calcHist` menjumlahkan berapa banyak piksel bernilai 0 hingga 255:
  - `[masks[key]]`: citra input.
  - `[0]`: kanal yang dihitung (kanal tunggal).
  - `None`: tidak ada mask tambahan.
  - `[256]`: jumlah bin (0–255).
  - `[0, 256]`: rentang nilai piksel.

Hasilnya: array berisi jumlah piksel untuk setiap nilai intensitas.

- Menampilkan histogram sebagai kurva garis.
- Diberi label sesuai nama warna ('BLUE', 'RED-BLUE', dll).

Judul grafik.

Label sumbu-x: menggambarkan nilai intensitas piksel (dalam citra biner, biasanya hanya 0 atau 255).

Label sumbu-y: jumlah piksel untuk tiap intensitas.

Menampilkan **label warna** di pojok grafik agar mudah dibedakan.

Menambahkan grid untuk membantu membaca nilai grafik.

Menyusun tata letak agar tidak tumpang tindih.

Menampilkan grafik ke layar.

```
[15]: # Plot histogram masing-masing hasil
      plt.figure(figsize=(10, 5))
      for key in display_keys[1:]:  # Skip 'NONE'
          hist = cv2.calcHist([masks[key]], [0], None, [256], [0, 256])
          plt.plot(hist, label=key)
      plt.title('Histogram Biner Warna')
      plt.xlabel('Intensitas (0-255)')
      plt.ylabel('Jumlah Piksel')
      plt.legend()
      plt.grid(True)
      plt.tight_layout()
      plt.show()
```

3.5 Penjelasan Fungsi

- cv2.imread: Membaca gambar dengan format default OpenCV (BGR).
- cv2.cvtColor(..., COLOR_BGR2RGB): Mengubah ke RGB agar warna tampil benar saat menggunakan matplotlib.
- Mengonversi gambar asli ke **grayscale**, biasanya digunakan sebelum proses peningkatan kontras.

`increase_brightness`: Menambahkan nilai konstan (`val`) ke semua piksel citra → citra menjadi lebih terang.

- `equalize_contrast`: Melakukan histogram equalization untuk meningkatkan kontras citra grayscale.

Membuat dictionary results berisi lima versi gambar:

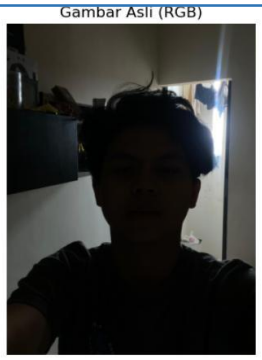
1. Gambar asli berwarna
2. Grayscale
3. Grayscale yang lebih terang
4. Grayscale dengan kontras ditingkatkan
5. Gabungan brightness & kontras

Membuat figure dengan ukuran tinggi (20 inci) agar semua gambar tertata vertikal.

- `enumerate(results.items())`: Loop semua hasil sekaligus ambil judulnya (`title`) dan gambarnya (`img`).
- `subplot(len(results), 1, idx+1)`: Menyusun citra secara **vertikal** (jumlah baris sesuai jumlah hasil).
- `cmap='gray'`: Digunakan jika gambar grayscale (jumlah channel = 1).
- `axis('off')`: Sembunyikan sumbu agar tampilan lebih bersih.
- `tight_layout`: Menghindari tumpang tindih antar judul dan gambar.
- `show()`: Menampilkan hasil plot ke layar.

UTS

Gambar Asli (RGB)



Grayscale

Grayscale + Brightness + Contrast

Grayscale + Brightness

Grayscale + Contrast

BAB IV
PENUTUP

Melalui pelaksanaan UTS Praktikum Pengolahan Citra Digital ini, mahasiswa telah memperoleh pengalaman langsung dalam menerapkan berbagai teknik pengolahan citra, mulai dari transformasi dasar hingga filtering dan deteksi tepi. Praktikum ini memberikan kontribusi besar dalam meningkatkan pemahaman mahasiswa terhadap teori yang telah dipelajari di kelas serta kemampuannya dalam mengimplementasikannya melalui bahasa pemrograman Python. Dengan demikian, diharapkan mahasiswa mampu mengembangkan kemampuan analitis dan teknis yang dapat digunakan dalam studi lanjutan maupun dunia kerja di bidang teknologi informasi, khususnya dalam pengolahan citra digital.

DAFTAR PUSTAKA

[1] " Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson Education.

Buku ini menjadi rujukan utama dalam pengolahan citra digital, membahas konsep grayscale, histogram equalization, dan color space seperti RGB dan HSV.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

Menjelaskan metode segmentasi warna, konversi warna, dan teknik thresholding dalam konteks visi komputer.

OpenCV Documentation. (2023). *OpenCV: Open Source Computer Vision Library*.

Pratt, W. K. (2007). *Digital Image Processing: PIKS Inside* (4th ed.). Wiley-Interscience.

Menjelaskan secara teknis tentang histogram, peningkatan kontras, serta algoritma pengolahan citra lainnya.

Russ, J. C. (2011). *The Image Processing Handbook* (6th ed.). CRC Press.

Membahas teknik pengolahan brightness dan contrast adjustment secara praktis serta aplikatif.