

**UTS**  
**PENGOLAHAN CITRA**



NAMA : Rafi Indra Pramudhito Zuhayr

NIM : 202331291

KELAS : F

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

NO.PC : 2

ASISTEN : 1. Sasikirana Ramadhanty Setiawan Putri

2. Rizqy Amanda

3. Ridho Chaerullah

4. Sakura Amastasya Salsabila Setiyanto

**INSTITUT TEKNOLOGI PLN**  
**TEKNIK INFORMATIKA**  
**2025/2026**

## DAFTAR ISI

DAFTAR ISI .....	2
BAB I .....	3
<b>1.1 Rumusan Masalah</b> .....	3
<b>1.2 Tujuan Masalah</b> .....	3
<b>1.3 Manfaat Masalah</b> .....	4
BAB II .....	5
2.1. Model Warna HSV .....	5
2.2. Deteksi Warna Menggunakan Masking.....	5
2.3. Histogram Citra.....	5
2.4. Thresholding .....	5
2.5. Perbaikan Citra Backlight.....	6
BAB III .....	7
3.1 Membaca dan Menampilkan Citra Asli .....	7
3.2 Deteksi Warna RGB Menggunakan Ruang Warna HSV .....	8
3.3 Pembuatan Histogram Warna .....	9
3.4 Konversi ke Grayscale dan Thresholding .....	10
3.5 Perbaikan Gambar Backlight .....	11
BAB IV .....	13
DAFTAR PUSTAKA .....	14

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Rumusan Masalah**

Dalam dunia digital saat ini, pengolahan citra memainkan peran penting dalam berbagai bidang seperti kedokteran, industri, keamanan, hingga media sosial. Kemampuan untuk menganalisis dan memodifikasi citra menjadi keahlian yang dibutuhkan dalam era informasi ini. Praktikum UTS Pengolahan Citra Digital memberikan tugas untuk menerapkan beberapa teknik dasar dalam pemrosesan citra yang bertujuan meningkatkan kualitas dan mengekstraksi informasi dari sebuah gambar.

Beberapa masalah yang dirumuskan dari tugas UTS Praktikum ini adalah:

- Bagaimana mendeteksi warna spesifik seperti merah, hijau, dan biru secara akurat pada sebuah citra menggunakan ruang warna yang tepat?
- Bagaimana distribusi intensitas piksel dalam citra dapat dianalisis melalui histogram, dan bagaimana hasilnya digunakan untuk memahami karakteristik citra?
- Bagaimana metode thresholding (baik otomatis maupun manual) dapat diterapkan untuk memisahkan objek dari latar belakang?
- Bagaimana citra dengan pencahayaan tidak merata (backlight) dapat diperbaiki sehingga objek utama (misalnya wajah) menjadi lebih jelas

#### **1.2 Tujuan Masalah**

Tujuan dari UTS ini adalah untuk menerapkan dan memahami beberapa teknik penting dalam pengolahan citra digital, antara lain:

- Mampu melakukan deteksi warna dengan menggunakan konversi ruang warna dari BGR ke HSV, serta memfilter warna tertentu berdasarkan nilai hue, saturation, dan value.
- Menampilkan dan menganalisis histogram untuk mengetahui distribusi piksel berdasarkan tingkat intensitas warna pada citra asli maupun hasil deteksi.
- Menerapkan metode thresholding otomatis (Otsu) dan manual untuk proses segmentasi citra dan mengamati efek dari masing-masing metode.
- Mengembangkan metode sederhana untuk memperbaiki citra wajah dengan pencahayaan buruk melalui peningkatan brightness dan kontras.

### **1.3 Manfaat Masalah**

Manfaat yang diperoleh dari UTS praktikum ini meliputi:

- Memberikan pemahaman yang lebih dalam terhadap konsep dasar dalam pengolahan citra digital.
- Melatih kemampuan teknis dalam menerapkan metode deteksi warna, histogram analisis, thresholding, dan peningkatan kualitas citra menggunakan Python dan OpenCV.
- Meningkatkan kemampuan analisis dan interpretasi terhadap hasil citra digital, serta memperkuat pemahaman teori dengan praktik langsung.
- Memberikan dasar yang kuat bagi mahasiswa untuk mengembangkan sistem pengolahan citra lebih kompleks di masa depan, seperti sistem pengenalan wajah, klasifikasi objek, atau pengawasan berbasis citra.

## **BAB II**

### **LANDASAN TEORI**

Pengolahan citra digital (Digital Image Processing) adalah proses memanipulasi data gambar menggunakan algoritma komputer. Dalam konteks praktikum ini, digunakan beberapa teknik dasar namun sangat penting. Berikut adalah teori-teori pendukung dari setiap bagian UTS praktikum:

#### **2.1. Model Warna HSV**

Model HSV (Hue, Saturation, Value) merepresentasikan warna berdasarkan rona (hue), intensitas warna (saturation), dan terang-gelap (value). Model ini lebih intuitif dibanding BGR (Blue, Green, Red), khususnya dalam deteksi warna.

Hue mewakili jenis warna (0–179 di OpenCV), Saturation menunjukkan kejenuhan warna (0–255), dan Value menunjukkan kecerahan warna (0–255). Konversi BGR ke HSV memungkinkan kita untuk lebih mudah membedakan warna dominan pada citra.

#### **2.2. Deteksi Warna Menggunakan Masking**

Setelah mengubah citra ke HSV, filter warna diterapkan dengan `cv2.inRange()` yang menghasilkan binary mask berdasarkan range HSV yang diinginkan. Mask ini kemudian digunakan untuk menampilkan bagian dari citra yang sesuai warna tersebut. Proses ini sangat penting dalam aplikasi pelacakan objek atau segmentasi warna.

#### **2.3. Histogram Citra**

Histogram pada citra digital menunjukkan jumlah piksel untuk setiap tingkat intensitas. Dalam konteks grayscale, sumbu x menunjukkan intensitas (0-255) dan sumbu y menunjukkan jumlah piksel. Analisis histogram dapat mengindikasikan kontras citra, kecerahan, dan distribusi warna. Histogram RGB atau HSV juga dapat digunakan untuk mengevaluasi dominasi warna dalam gambar.

#### **2.4. Thresholding**

Thresholding merupakan teknik untuk mengubah citra grayscale menjadi biner (hitam-putih), dengan menetapkan nilai ambang batas. Threshold Manual menetapkan nilai ambang secara tetap, misal 100 atau 150. Threshold Otsu merupakan metode otomatis yang mencari ambang optimal dengan meminimalkan variansi intra-kelas dan memaksimalkan variansi antar-kelas (Otsu, 1979). Thresholding sangat berguna dalam ekstraksi objek dari latar belakang, OCR, atau segmentasi citra medis.

### **2.5. Perbaikan Citra Backlight**

Backlight adalah kondisi pencahayaan di mana cahaya datang dari belakang objek, menyebabkan wajah atau objek utama tampak gelap. Teknik perbaikan yang digunakan dalam UTS praktikum termasuk:

- **Brightness Adjustment.**  
Menambahkan konstanta pada semua piksel untuk meningkatkan kecerahan.
- **Contrast Stretching.**  
Menyesuaikan rentang nilai piksel agar distribusinya lebih merata dan detail lebih terlihat.

Kombinasi keduanya mampu meningkatkan visibilitas pada citra yang terkena pencahayaan buruk.

## BAB III

### HASIL

#### 3.1 Membaca dan Menampilkan Citra Asli

Mengimport library yang dibutuhkan yaitu OpenCV, numpy, dan matplotlib. Setelah librarynya di import gambar di muat ke dalam kode menggunakan function imread(), karena OpenCV menggunakan format BGR, maka gambar diubah menjadi RGB dengan cv2.cvtColor() agar sesuai dengan format matplotlib. Gambar ditampilkan menggunakan plt.imshow().

```
DETEKSI WARNA PADA CITRA - OpenCV & Matplotlib

import cv2 ## 202331291_Rafi Indra Pramudhito Zuhayr
import numpy as np
import matplotlib.pyplot as plt

[1] Python

Load gambar

img = cv2.imread('img.jpg') ## 202331291_Rafi Indra Pramudhito Zuhayr
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

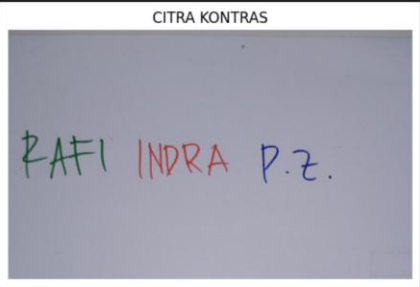
[29] Python
```

```
Tampilkan gambar asli dengan kontras disesuaikan

plt.figure(figsize=(6, 6)) ## 202331291_Rafi Indra Pramudhito Zuhayr
plt.title("CITRA KONTRAS")
plt.imshow(img_rgb)
plt.axis('off')
plt.show()

[30] Python

CITRA KONTRAS


```

### 3.2 Deteksi Warna RGB Menggunakan Ruang Warna HSV

Fungsi `detect_color()` dibuat untuk mendeteksi warna merah, hijau, dan biru berdasarkan ambang batas HSV. Warna yang dideteksi dibuatkan mask menggunakan `cv2.inRange()` lalu dikalikan dengan gambar asli memakai `cv2.bitwise_and()`. Proses dilakukan untuk masing-masing warna: merah, hijau, dan biru. Hasil ditampilkan dengan `plt.imshow()`.

```

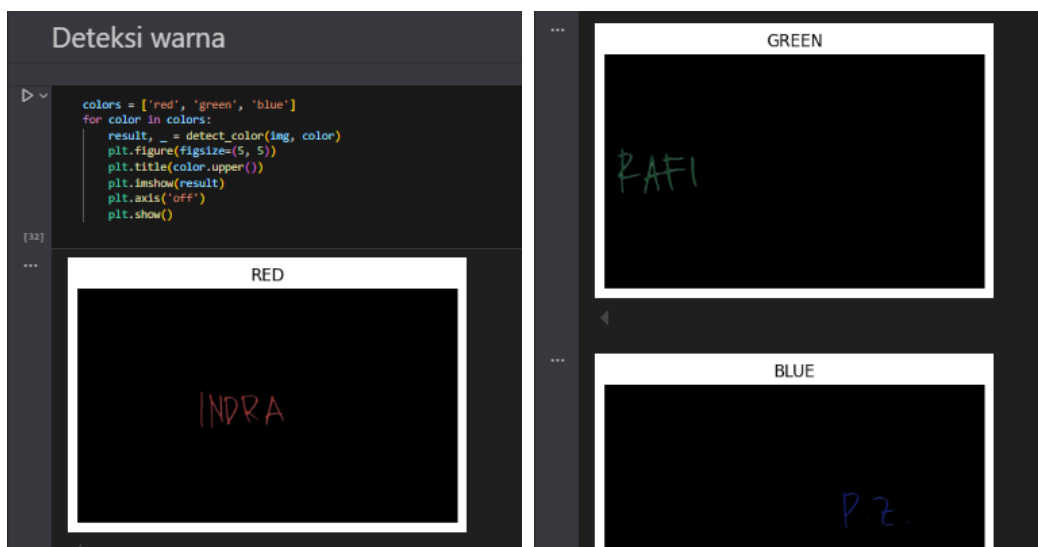
Fungsi deteksi warna berdasarkan HSV

def detect_color(img_bgr, color):
    hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)

    if color == 'red':
        lower1 = np.array([0, 120, 70])
        upper1 = np.array([10, 255, 255])
        lower2 = np.array([170, 120, 70])
        upper2 = np.array([180, 255, 255])
        mask1 = cv2.inRange(hsv, lower1, upper1)
        mask2 = cv2.inRange(hsv, lower2, upper2)
        mask = mask1 + mask2
    elif color == 'green':
        lower = np.array([35, 25, 25])
        upper = np.array([85, 255, 255])
        mask = cv2.inRange(hsv, lower, upper)
    elif color == 'blue':
        lower = np.array([100, 150, 0])
        upper = np.array([140, 255, 255])
        mask = cv2.inRange(hsv, lower, upper)
    else:
        raise ValueError("color harus 'red', 'green', atau 'blue'")

    result = cv2.bitwise_and(img_bgr, img_bgr, mask=mask)
    return cv2.cvtColor(result, cv2.COLOR_BGR2RGB), mask

```





### 3.3 Pembuatan Histogram Warna

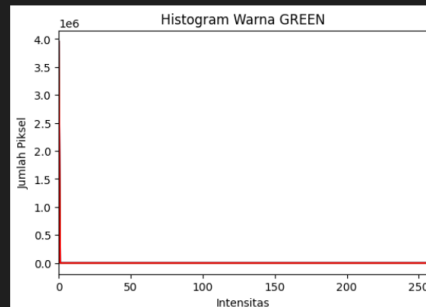
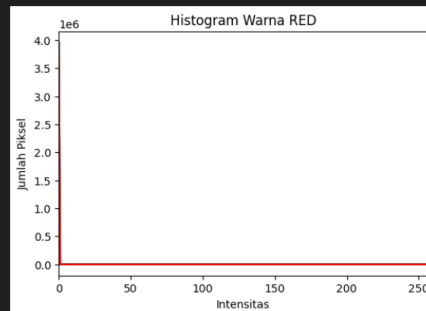
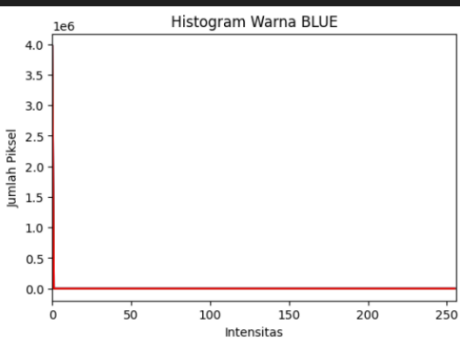
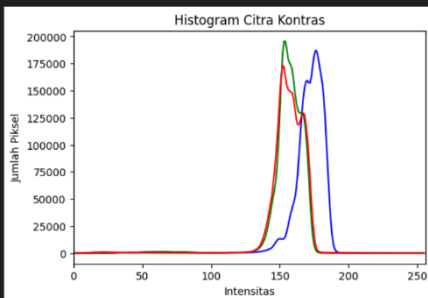
Gambar RGB diubah ke BGR untuk keperluan per channel (karena cv2.calcHist() bekerja di BGR). Histogram setiap channel (biru, hijau, merah) dihitung menggunakan cv2.calcHist(). Grafik histogram ditampilkan dengan plt.plot().

Fungsi membuat histogram dari gambar

```
def plot_histogram(image_rgb, title="Histogram"):
    chans = cv2.split(cv2.cvtColor(image_rgb, cv2.COLOR_RGB2BGR))
    colors = ('b', 'g', 'r')
    plt.figure(figsize=(6, 4))
    plt.title(title)
    plt.xlabel('Intensitas')
    plt.ylabel('Jumlah Pixel')
    for chan, col in zip(chans, colors):
        hist = cv2.calcHist([chan], [0], None, [0, 256])
        plt.plot(hist, color=col)
    plt.xlim(0, 256)
    plt.show()
```

Histogram untuk citra asli

```
plot_histogram(img_rgb, "Histogram Citra Kontras")
for color in colors:
    result, _ = detect_color(img, color)
    plot_histogram(result, f"Histogram Warna {color.upper()}")
```



### 3.4 Konversi ke Grayscale dan Thresholding

Gambar asli dikonversi ke grayscale dengan `cv2.cvtColor()`. Thresholding dilakukan secara otomatis menggunakan metode Otsu (`cv2.THRESH_OTSU`). Thresholding manual dilakukan untuk beberapa nilai ambang (50, 100, 128, 150, 200). Hasil dari setiap threshold ditampilkan dan nilai-nilai ambang batas diurutkan.

#### Cari ambang batas otomatis (Otsu)

```
th_otsu_val, th_otsu_img = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
print("Ambang batas Otsu:", th_otsu_val)

# Uji beberapa nilai threshold manual
thresholds = [50, 100, 128, 150, 200]
results = []

for t in thresholds:
    _, bin_img = cv2.threshold(gray, t, 255, cv2.THRESH_BINARY)
    results.append((t, bin_img))

# Tampilkan hasil threshold
for t, bin_img in results:
    plt.figure(figsize=(4, 4))
    plt.title(f"Ambang Batas = {t}")
    plt.imshow(bin_img, cmap='gray')
    plt.axis('off')
    plt.show()

# Urutkan ambang batas
sorted_thresholds = sorted(thresholds + [int(th_otsu_val)])
print("Ambang batas terurut:", sorted_thresholds)
```

[36]



### 3.5 Perbaiki Gambar Backlight

Gambar kedua dibaca menggunakan `cv2.imread()`. Dikonversi ke grayscale. Peningkatan kecerahan dilakukan dengan menambah nilai beta pada `cv2.convertScaleAbs()`. Peningkatan kontras dilakukan dengan mengubah nilai alpha. Gabungan kontras dan kecerahan dibuat menggunakan kedua parameter. Hasil ditampilkan dalam satu grid `plt.subplots()`.

```

Memperbaiki gambar Backlight

Baca Gambar Asli

img2 = cv2.imread('img2.jpg')
[37]

Konversi ke Grayscale

gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
[38]

Tingkatkan Kecerahan (Brightness)

bright = cv2.convertScaleAbs(gray, alpha=1.0, beta=40) # beta menambah brightness
[39]

Tingkatkan Kontras (Contrast)

contrast = cv2.convertScaleAbs(gray, alpha=1.5, beta=0)
[40]

Tingkatkan Kecerahan & Kontras Bersamaan

bright_contrast = cv2.convertScaleAbs(gray, alpha=1.5, beta=40)
[41]

```

```

Simpan hasil

fig, axes = plt.subplots(2, 3, figsize=(15, 8))
fig.suptitle('Perbaikan Gambar Backlight', fontsize=16)

# Tampilkan gambar
axes[0, 0].imshow(img2)
axes[0, 0].set_title('Gambar Asli')
axes[0, 0].axis('off')

axes[0, 1].imshow(gray, cmap='gray')
axes[0, 1].set_title('Gray')
axes[0, 1].axis('off')

axes[0, 2].imshow(bright, cmap='gray')
axes[0, 2].set_title('Gray + Cerah')
axes[0, 2].axis('off')

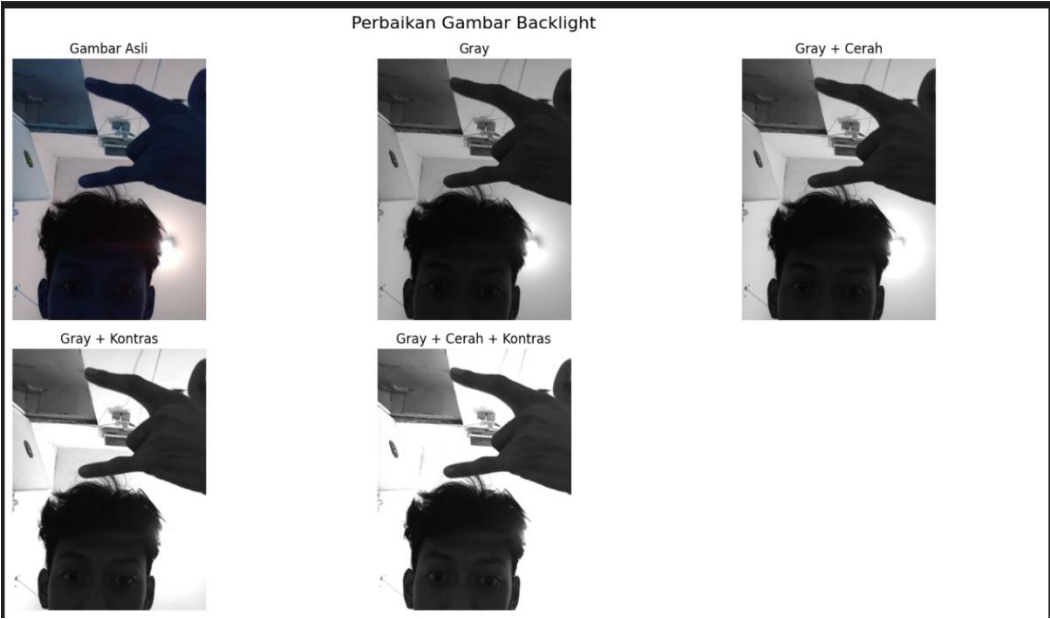
axes[1, 0].imshow(contrast, cmap='gray')
axes[1, 0].set_title('Gray + Kontras')
axes[1, 0].axis('off')

axes[1, 1].imshow(bright_contrast, cmap='gray')
axes[1, 1].set_title('Gray + Cerah + Kontras')
axes[1, 1].axis('off')

axes[1, 2].axis('off')

plt.tight_layout()
plt.show()
[43]

```



## **BAB IV**

### **PENUTUP**

Melalui UTS Praktikum ini, diperoleh pemahaman menyeluruh tentang teknik dasar pengolahan citra digital, meliputi deteksi warna, histogram, thresholding, dan perbaikan citra backlight. Penggunaan model HSV sangat efektif dalam mendeteksi warna tertentu dengan tingkat akurasi tinggi. Histogram menjadi alat visualisasi penting untuk menganalisis distribusi intensitas dalam gambar. Metode threshold Otsu dan manual masing-masing memiliki kelebihan, dan pemilihannya bergantung pada karakteristik citra. Kombinasi peningkatan brightness dan kontras sangat membantu dalam memperbaiki kualitas gambar dengan pencahayaan buruk. Hasil UTS Praktikum ini dapat menjadi fondasi untuk studi lanjutan, seperti deteksi objek otomatis, segmentasi citra kompleks, dan aplikasi real-time berbasis kamera.

## DAFTAR PUSTAKA

1. <https://journal.universitaspahlawan.ac.id/index.php/jutin/article/download/21406/15328/69302>
2. <https://journal.unpacti.ac.id/index.php/JSCE/article/download/878/504/>
3. <https://ojs.unh.ac.id/index.php/jurteki/article/download/1509/782/3173>
4. <http://download.garuda.kemdikbud.go.id/article.php?article=2119046&val=15966&title=PENGOLAHAN+CITRA+DIGITAL+DAN+HISTOGRAM+DENGAN+PHYTON+DAN+TEXT+EDITOR+PHYCHARM>
5. <https://jurnalmahasiswa.com/index.php/aidanspk/article/download/1478/977/3036>
6. <https://jurnal.tau.ac.id/index.php/snartek/article/download/711/476/2899>
7. <https://ejournal.unesa.ac.id/index.php/jinacs/article/download/47438/39943/88958>
8. <https://ejournal.itn.ac.id/index.php/jati/article/download/12444/6995/>
9. <https://journal2.unusa.ac.id/index.php/ATCSJ/article/download/4/2/4>
10. <https://jurnal.mdp.ac.id/index.php/jatisi/article/download/8110/2010/>