

UTS
PENGOLAHAN CITRA



NAMA : Muhammad Ichwan Fauzi
NIM : 202331227
KELAS : E
DOSEN : Dr. Darma Rusjdi, S.T., M.Kom
NO.PC : 15
ASISTEN : 1. FAUZAN ARROYAN
2. ABDUR RASYID RIDHO

INSTITUT TEKNOLOGI PLN
TEKNIK INFORMATIKA
2025

DAFTAR ISI**Contents**

DAFTAR ISI	2
BAB I PENDAHULUAN	3
1.1 Rumusan Masalah.....	3
1.2 Tujuan Masalah.....	3
1.3 Manfaat Masalah	3
BAB II LANDASAN TEORI	4
2.1 Pengantar Pengolahan Citra Digital.	4
2.2 Operasi Dasar pada Citra Grayscale.	4
2.3 Analisis Histogram dan Histogram Equalization	4
2.4 Pengolahan Citra Berwarna.	5
2.5 Histogram pada Citra Warna.	5
BAB III HASIL	7
BAB IV PENUTUP.....	22
DAFTAR PUSTAKA	23

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Pada praktikum UTS Pengolahan Citra Digital ini, permasalahan utama yang diangkat adalah bagaimana cara melakukan deteksi warna dasar (merah, hijau, biru) pada sebuah citra yang dibuat secara manual, serta bagaimana cara mengatur nilai ambang batas (threshold) warna tersebut agar hasil deteksinya akurat. Selain itu, terdapat pula tantangan dalam memperbaiki kualitas citra backlight agar objek utama (wajah) bisa terlihat lebih jelas. Jadi, pertanyaannya adalah:

- Bagaimana teknik deteksi warna bisa digunakan untuk mengidentifikasi warna tulisan pada gambar secara akurat?
- Bagaimana menentukan ambang batas warna yang tepat berdasarkan komponen HSV?
- Bagaimana cara meningkatkan kualitas gambar yang gelap akibat backlight agar lebih informatif?

1.2 Tujuan Masalah

Tujuan dari pelaksanaan praktikum ini adalah untuk mengaplikasikan konsep dasar pengolahan citra dalam mendeteksi warna tertentu (merah, hijau, biru) menggunakan Python dan OpenCV. Selain itu, praktikum ini juga bertujuan agar mahasiswa dapat memahami cara kerja masking warna berbasis HSV, melakukan eksperimen terhadap nilai ambang batas warna, serta memperbaiki citra yang terkena efek backlight. Di akhir praktikum, diharapkan mahasiswa mampu:

- Mendeteksi warna spesifik dalam citra berdasarkan rentang HSV.
- Menganalisis histogram warna hasil deteksi.
- Menentukan dan mengurutkan nilai ambang batas warna.
- Mengolah gambar backlight agar objek utama lebih terlihat jelas.

1.3 Manfaat Masalah

Manfaat yang diperoleh dari praktikum ini yaitu menambah pemahaman dan pengalaman mahasiswa dalam menerapkan teknik pengolahan citra digital secara langsung. Praktikum ini memberikan gambaran nyata bagaimana proses segmentasi warna dan perbaikan citra bisa diaplikasikan pada gambar yang diambil sendiri. Selain itu, mahasiswa juga jadi lebih terbiasa menggunakan tools seperti OpenCV dan memahami pentingnya representasi warna dalam pemrosesan gambar. Dalam jangka panjang, pemahaman ini bermanfaat untuk pengembangan sistem deteksi visual, pengenalan objek, serta pengolahan gambar dalam berbagai bidang teknologi informasi.

BAB II

LANDASAN TEORI

2.1 Pengantar Pengolahan Citra Digital.

Pengolahan citra digital adalah bidang yang membahas bagaimana gambar dalam bentuk digital diproses dan dimodifikasi untuk mendapatkan informasi atau hasil visual tertentu. Citra digital sendiri bisa dianggap sebagai kumpulan angka yang merepresentasikan intensitas warna atau kecerahan di tiap titik pada gambar. Biasanya, gambar ini diubah menjadi bentuk matriks, di mana setiap elemen dari matriks disebut piksel. Nah, piksel-piksel inilah yang jadi dasar dari semua proses dalam pengolahan citra.

Dalam kehidupan sehari-hari, pengolahan citra digunakan dalam banyak hal, misalnya untuk deteksi wajah di kamera ponsel, analisis gambar medis seperti MRI atau rontgen, hingga dalam bidang pertanian untuk mendeteksi kondisi tanaman dari foto udara. Tapi sebelum masuk ke aplikasi yang kompleks, kita perlu pahami dulu dasar-dasar seperti peningkatan kualitas gambar, histogram, hingga pemrosesan warna.

2.2 Operasi Dasar pada Citra Grayscale.

Pada tahap awal pengolahan citra, biasanya yang digunakan adalah citra grayscale atau citra hitam putih, yang hanya memiliki satu kanal intensitas. Salah satu teknik dasar adalah **operasi titik**. Ini adalah proses pengubahan piksel secara langsung berdasarkan nilai intensitasnya, tanpa memperhitungkan piksel tetangga. Contohnya seperti **pembuatan citra negatif**, di mana warna terang jadi gelap dan sebaliknya.

Selain itu, ada teknik **thresholding** yang fungsinya mengubah citra grayscale menjadi citra biner. Misalnya, kita bisa menentukan bahwa semua piksel dengan nilai di atas 128 akan diubah menjadi putih, sedangkan sisanya jadi hitam. Ini sering digunakan dalam segmentasi objek agar bisa memisahkan bagian penting dari gambar.

Lalu ada juga proses **kontras stretching**, atau peregangan kontras, yang dilakukan untuk memperjelas perbedaan antara bagian gelap dan terang dalam gambar. Teknik ini sangat berguna jika gambar terlihat terlalu datar atau tidak memiliki detail yang menonjol.

2.3 Analisis Histogram dan Histogram Equalization

Untuk memahami bagaimana nilai intensitas tersebar dalam gambar, digunakanlah **histogram**. Histogram citra adalah grafik yang menunjukkan jumlah piksel untuk setiap tingkat kecerahan (biasanya dari 0 sampai 255). Dengan melihat histogram, kita bisa tahu apakah gambar terlalu gelap, terlalu terang, atau punya kontras yang rendah.

Misalnya, kalau semua piksel terkumpul di nilai rendah (sekitar 0–50), artinya gambar gelap. Kalau di nilai tinggi (sekitar 200–255), gambar cenderung terlalu terang. Nah, untuk memperbaiki distribusi ini, digunakan teknik **histogram equalization**, yaitu proses untuk meratakan histogram agar citra memiliki kontras yang lebih seimbang. Teknik ini sangat efektif untuk membuat detail yang tadinya tersembunyi jadi lebih terlihat, terutama di gambar medis atau foto dengan pencahayaan buruk.

2.4 Pengolahan Citra Berwarna.

Setelah paham cara kerja citra grayscale, kita beralih ke **citra berwarna**. Berbeda dari grayscale yang hanya memiliki satu nilai per piksel, citra berwarna punya tiga nilai intensitas: **Red (merah)**, **Green (hijau)**, dan **Blue (biru)**—ini yang dikenal sebagai model **RGB**. Kombinasi dari ketiga warna dasar ini bisa menghasilkan berbagai macam warna yang kita lihat di layar monitor.

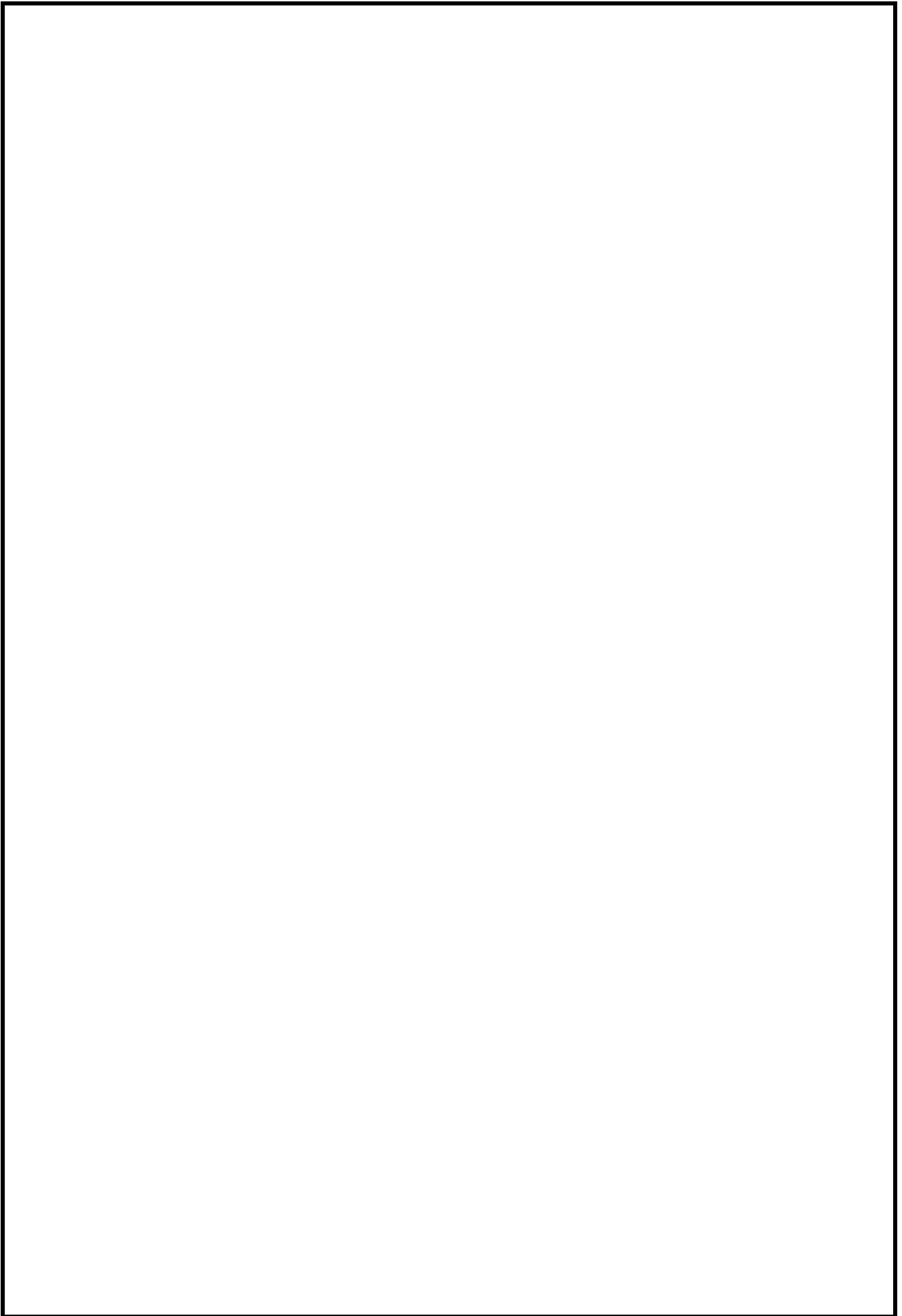
Dalam pemrosesan citra warna, biasanya tiap kanal (R, G, dan B) bisa diproses secara terpisah. Misalnya kita bisa menaikkan intensitas kanal merah saja kalau ingin gambar terlihat lebih kemerahan. Namun, tidak semua operasi cocok diterapkan langsung di RGB, karena perubahan di satu kanal bisa memengaruhi tampilan warna secara keseluruhan. Untuk itu, kadang dilakukan konversi ke model warna lain seperti **HSV (Hue, Saturation, Value)**. Model ini lebih mendekati cara manusia memahami warna.

Dalam HSV, Hue menunjukkan jenis warna (misalnya merah, kuning, biru), Saturation menyatakan seberapa pekat warna tersebut, dan Value adalah tingkat kecerahan. Jika kita ingin memfilter objek berdasarkan warna, model HSV ini sangat membantu karena kita bisa cukup mengambil nilai Hue tertentu saja. Misalnya, mendeteksi objek biru tinggal ambil rentang Hue yang sesuai.

2.5 Histogram pada Citra Warna.

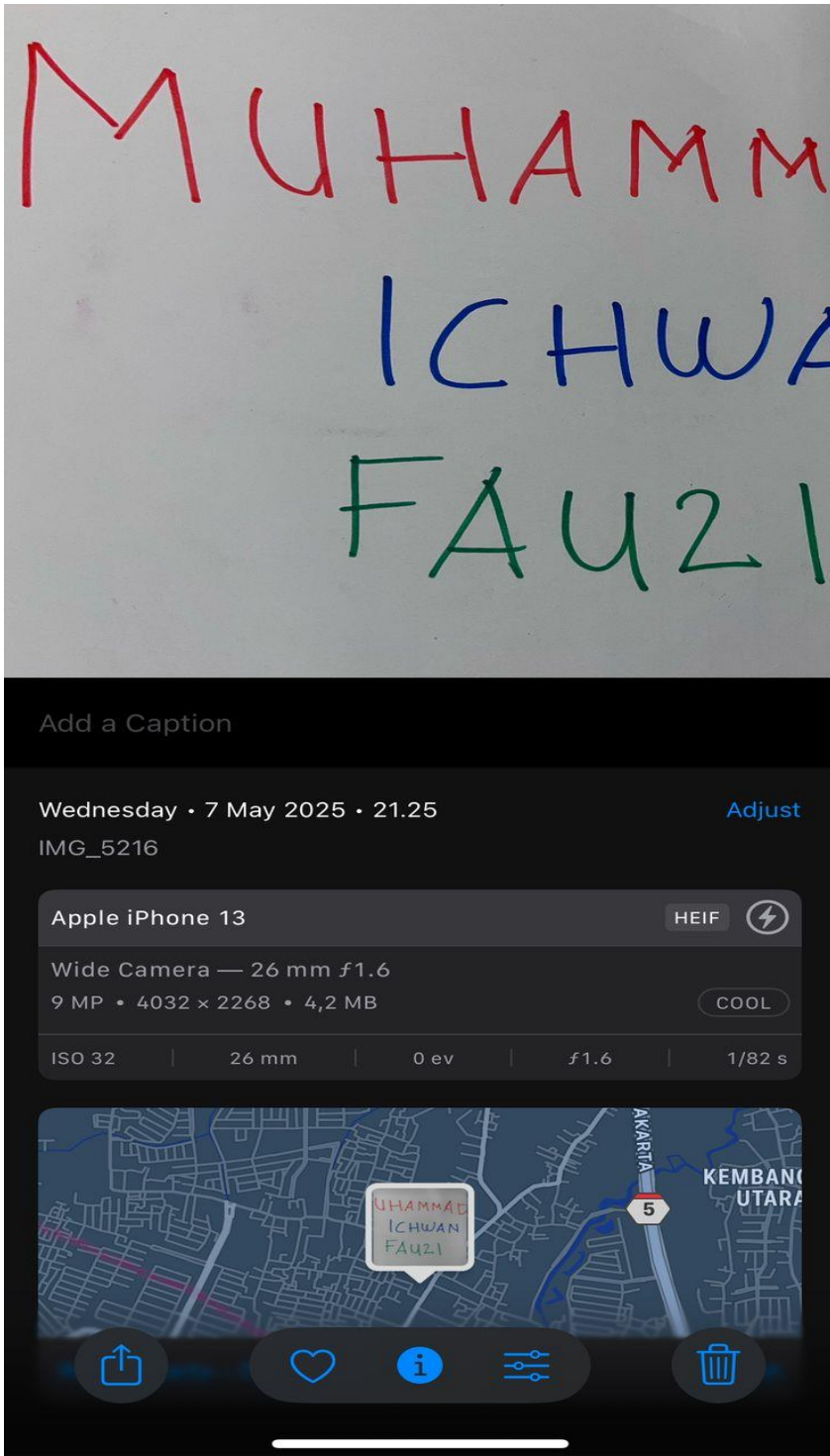
Histogram tidak hanya berlaku untuk citra grayscale, tapi juga bisa digunakan pada citra warna. Di sini, histogram dihitung untuk masing-masing kanal RGB. Misalnya, kita bisa melihat sebaran nilai warna merah dalam satu histogram, hijau di histogram lain, dan begitu juga biru. Dari sini, kita bisa tahu apakah warna dalam gambar cenderung dominan ke arah tertentu, atau kurang merata.

Dalam praktiknya, **histogram equalization** juga bisa dilakukan untuk masing-masing kanal warna, meskipun ini harus dilakukan dengan hati-hati karena bisa menyebabkan warna tampak tidak alami. Salah satu solusi yang sering digunakan adalah dengan mengonversi citra ke model HSV lalu hanya menerapkan equalization pada kanal Value (V), sehingga pencahayaan meningkat tanpa mengubah warna secara drastis.



BAB III

HASIL



```
: import cv2
import numpy as np
from matplotlib import pyplot as plt
```

```
: img = cv2.imread("iwannn.jpg")
```

Baca Citra Asli

```
: img.shape
```

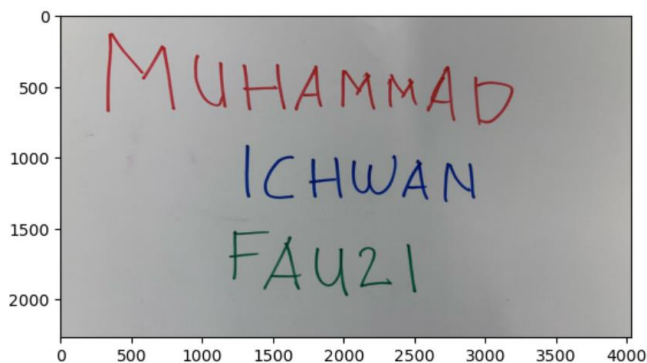
```
: (2268, 4032, 3)
```

```
: rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

- cv2 dari OpenCV dipakai buat baca dan proses gambar.
- numpy kita pakai karena semua gambar digital itu bentuknya array (matriks angka).
- matplotlib.pyplot biasanya dipakai buat nampilin gambar di Jupyter.
- cv2.imread() buat ngebaca gambar yang filenya bernama iwannn.jpg, pastiin file gambar iwannn.jpg ini udah ada di folder kerja yang sama.
- Img.shape buat ngeliat ukuran gambar. Output-nya (2268, 4032, 3) artinya: **2268** piksel tinggi (jumlah baris), **4032** piksel lebar (jumlah kolom), **3** artinya gambar ini berwarna, dengan 3 kanal warna (RGB — merah, hijau, biru)
- cv2.cvtColor() itu fungsinya buat ngonversi format warna.
- cv2.COLOR_BGR2RGB artinya kamu ngubah urutan kanal warna dari BGR ke RGB, biar warnanya tampil sesuai aslinya waktu ditampilin pakai matplotlib.

```
plt.imshow(rgb)
```

<matplotlib.image.AxesImage at 0x137190c4710>



Ini buat nampilin gambar rgb di notebook.


```
plt.subplot(2, 2, 1)
plt.imshow(rgb)
plt.title('Gambar Asli')

plt.subplot(2, 2, 2)
plt.imshow(rgb[:, :, 0], cmap="gray")
plt.title('Merah')

plt.subplot(2, 2, 3)
plt.imshow(rgb[:, :, 1], cmap="gray")
plt.title('Hijau')

plt.subplot(2, 2, 4)
plt.imshow(rgb[:, :, 2], cmap="gray")
plt.title('Biru')

plt.tight_layout(pad=3.0)
plt.show()
```

`plt.subplot(2, 2, 1)` Ini adalah subplot pertama (posisi 1 dari grid 2x2).

`plt.imshow(rgb)` nampilin gambar asli dalam format RGB.

`plt.title('Gambar Asli')` Dikasih judul "Gambar Asli" supaya orang tahu ini gambar awalnya.

`plt.subplot(2, 2, 2)` Subplot kedua: kamu nampilin **kanal merah** dari gambar.

`plt.imshow(rgb[:, :, 0])` artinya kamu ambil layer pertama (merah) dari array gambar.

, `cmap="gray"`) supaya meskipun ini data warna, tetap ditampilkan dalam bentuk gradasi abu-abu — putih berarti intensitas merahnya tinggi, hitam berarti rendah.

`plt.title('Merah')`

`plt.subplot(2, 2, 3)` Subplot ketiga, bagian kanal hijau.

`plt.imshow(rgb[:, :, 1], cmap="gray")` Sama kayak tadi, tapi ambil layer indeks 1.

`plt.title('Hijau')`

`plt.subplot(2, 2, 4)` Subplot keempat buat kanal biru.

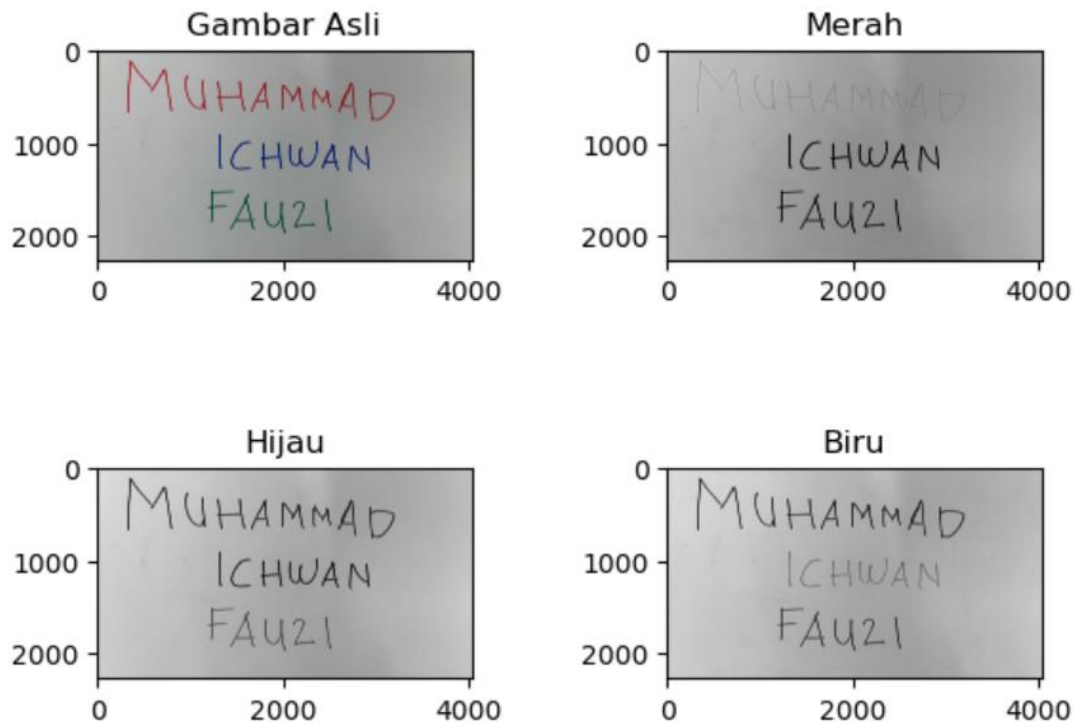
`plt.imshow(rgb[:, :, 2], cmap="gray")` ambil layer biru dari gambar.

`plt.title('Biru')`

`tight_layout()` ini biar semua subplot rapi, nggak tumpang tindih.

`pad=3.0` ngasih jarak antar gambar biar kelihatan lebih lega.

`plt.show()` buat nampilin semua plot di atas.



```
merah=rgb[:, :, 0]
fig, axs = plt.subplots(1, 2, figsize = (15, 5))
hist = cv2.calcHist([merah], [0], None, [256], [0, 256])
axs[0].imshow(merah, cmap='gray')
axs[1].plot(hist)
plt.show()
```

`rgb[:, :, 0]` artinya ngambil layer ke-0 (karena urutannya RGB, jadi 0 = merah).

`fig, axs = plt.subplots(1, 2, figsize = (15, 5))` Ini buat bikin dua plot berdampingan (1 baris, 2 kolom).

`figsize=(15, 5)` bikin plotnya panjang ke samping, biar lega dan kelihatan jelas.

`hist = cv2.calcHist([merah], [0], None, [256], [0, 256])` hitung histogram dari kanal merah pakai OpenCV.

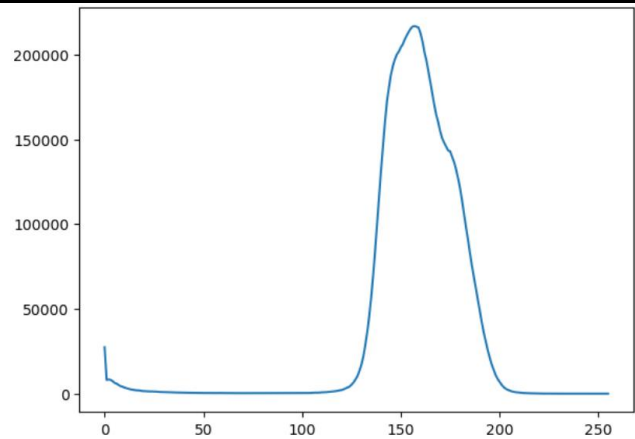
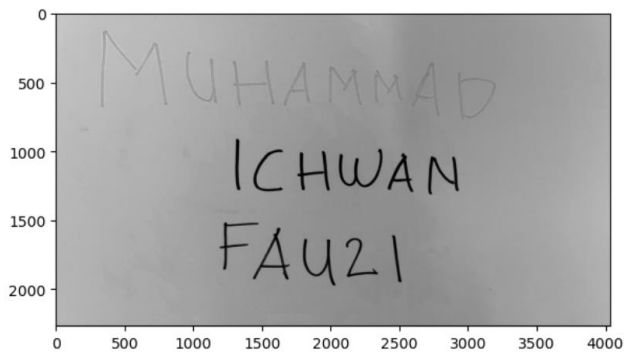
`cv2.calcHist` itu fungsinya buat ngitung distribusi intensitas warna.

`[256]` berarti kamu bagi intensitasnya jadi 256 level (dari 0–255, sesuai grayscale).

Output-nya adalah array yang nunjukin berapa banyak piksel yang punya nilai intensitas tertentu.

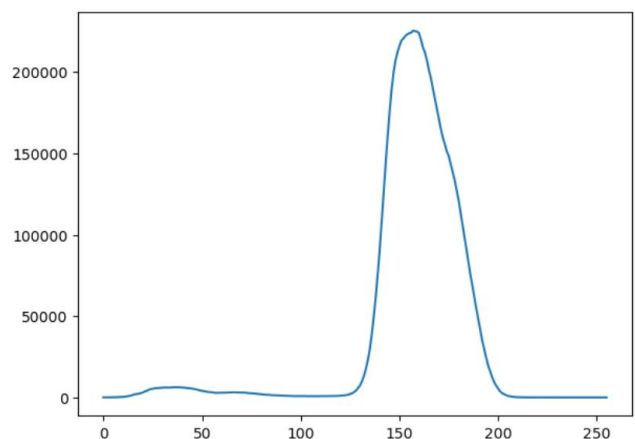
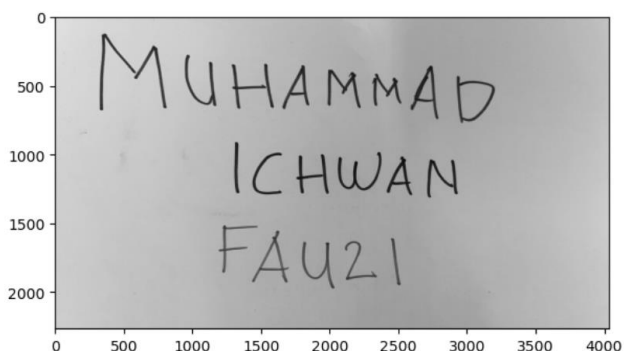
`cmap='gray'` biar jelas perbedaan terang-gelapnya.

`axs[1].plot(hist)` menampilkan histogram di sebelah kanan



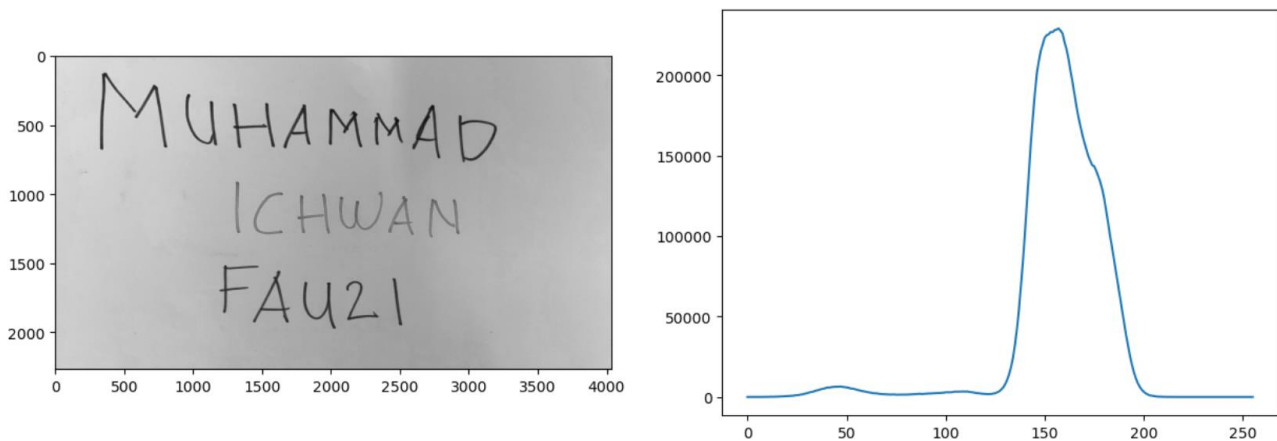
```
hijau=rgb[:, :, 1]
fig, axs = plt.subplots(1, 2, figsize = (15, 5))
hist = cv2.calcHist([hijau], [0], None, [256], [0, 256])
axs[0].imshow(hijau, cmap='gray')
axs[1].plot(hist)
plt.show()
```

Mengambil kanal hijau, Karena urutan RGB itu (0 = merah, 1 = hijau, 2 = biru), maka `rgb[:, :, 1]` adalah channel hijau. Lalu sama kayak kanal merah, lalu menampilkan gambar kanal biru dan histogram-nya.



```
biru=rgb[:, :, 2]
fig, axs = plt.subplots(1, 2, figsize = (15, 5))
hist = cv2.calcHist([biru], [0], None, [256], [0, 256])
axs[0].imshow(biru, cmap='gray')
axs[1].plot(hist)
plt.show()
```

mengambil kanal biru dari gambar RGB, pakai indeks ke-2 (karena urutannya R = 0, G = 1, B = 2). Lalu sama kayak kanal merah, lalu menampilkan gambar kanal biru dan histogram-nya.



```

image_hsv = cv2.cvtColor(rgb, cv2.COLOR_RGB2HSV)

gray = cv2.cvtColor(rgb, cv2.COLOR_RGB2GRAY)

fig, axs = plt.subplots(2, 2, figsize=(10,10))

red_lower1 = np.array([0, 50, 50])
red_upper1 = np.array([10, 255, 255])
red_lower2 = np.array([170, 50, 50])
red_upper2 = np.array([180, 255, 255])

green_lower = np.array([36, 50, 50])
green_upper = np.array([86, 255, 255])

blue_lower = np.array([100, 50, 50])
blue_upper = np.array([140, 255, 255])

mask_red1 = cv2.inRange(image_hsv, red_lower1, red_upper1)
mask_red2 = cv2.inRange(image_hsv, red_lower2, red_upper2)
mask_red = cv2.bitwise_or(mask_red1, mask_red2)
mask_green = cv2.inRange(image_hsv, green_lower, green_upper)
mask_blue = cv2.inRange(image_hsv, blue_lower, blue_upper)

combined_mask1 = np.bitwise_or(mask_red, mask_blue)
combined_mask2 = np.bitwise_or(combined_mask1, mask_green)

```

`image_hsv = cv2.cvtColor(rgb, cv2.COLOR_RGB2HSV)` konversi gambar dari RGB ke HSV

`gray = cv2.cvtColor(rgb, cv2.COLOR_RGB2GRAY)` menyimpan versi grayscale-nya, yang nanti dipakai buat threshold biner.

`red_lower1 = np.array....` : Warna merah agak spesial karena terletak di dua sisi skala Hue, jadi pakai dua rentang: 0–10 dan 170–180.

`green_lower = np.array....` : juga set rentang HSV untuk warna hijau dan biru. Nilai ini hasil eksperimen umum untuk mendeteksi warna dasar.

`mask_red1 = cv2.inRange....` : `cv2.inRange()` bikin mask biner (hitam-putih), di mana hanya warna yang sesuai rentang yang jadi putih.

Untuk merah, gabung dua range pakai `bitwise_or`.

`combined_mask1`: gabungan merah + biru.

`combined_mask2`: gabungan merah + biru + hijau.

`(thresh, binary1) =` bikin versi biner dari gambar grayscale sebagai perbandingan.

```
(thresh, binary1) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY)
axs[0,0].imshow(binary1, cmap = 'gray')
axs[0,0].set_title('NONE')

plt.figure(figsize=(12, 10))
plt.subplot(1, 3, 1)
axs[0,1].imshow(mask_blue, cmap='gray')
axs[0,1].set_title('Blue')
plt.axis('off')

plt.subplot(1, 3, 2)
axs[1,0].imshow(combined_mask1, cmap='gray')
axs[1,0].set_title('Red-Blue')
plt.axis('off')

plt.subplot(1, 3, 3)
axs[1,1].imshow(combined_mask2, cmap='gray')
axs[1,1].set_title('Red-Green-Blue')
plt.axis('off')

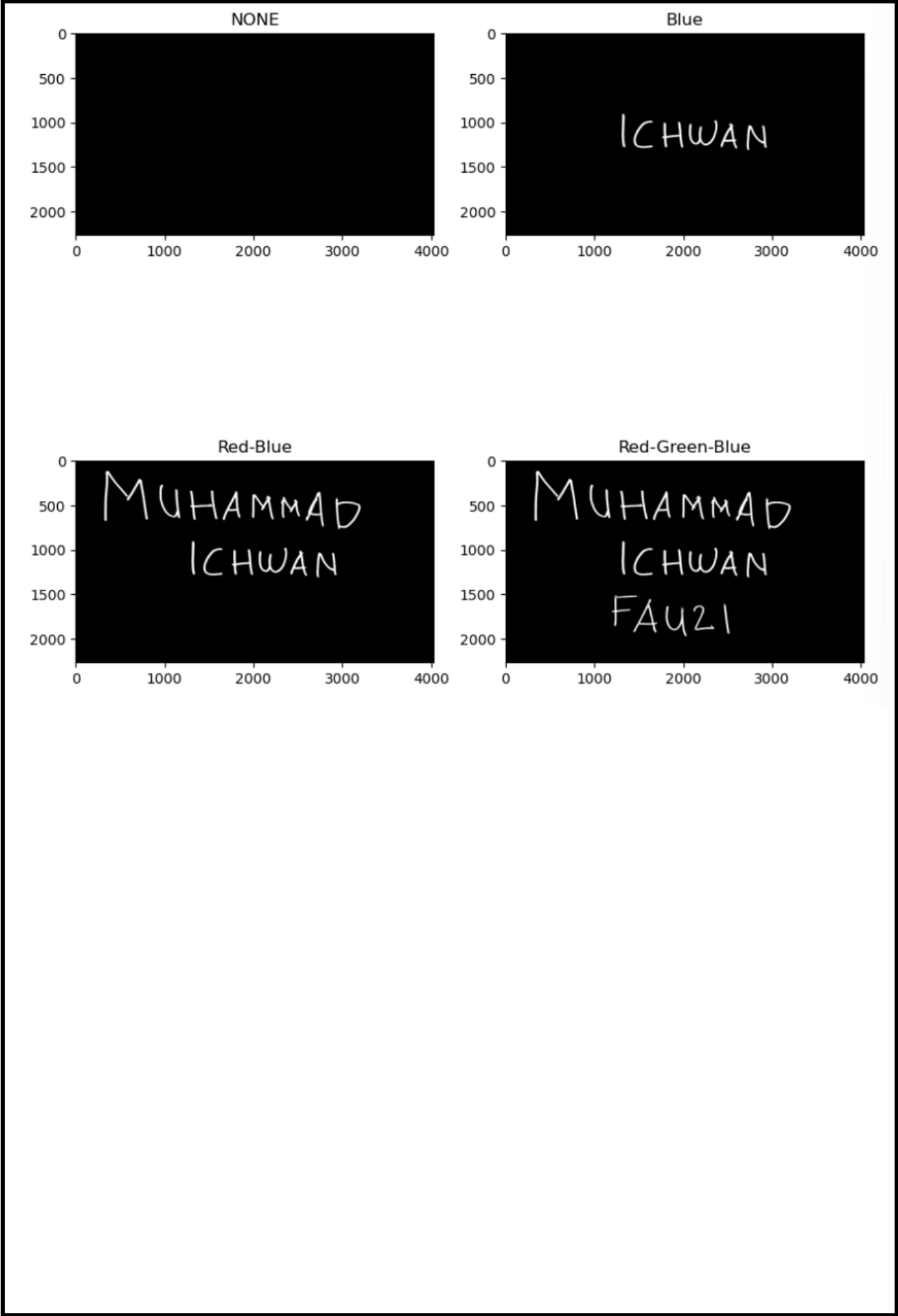
plt.show()
```

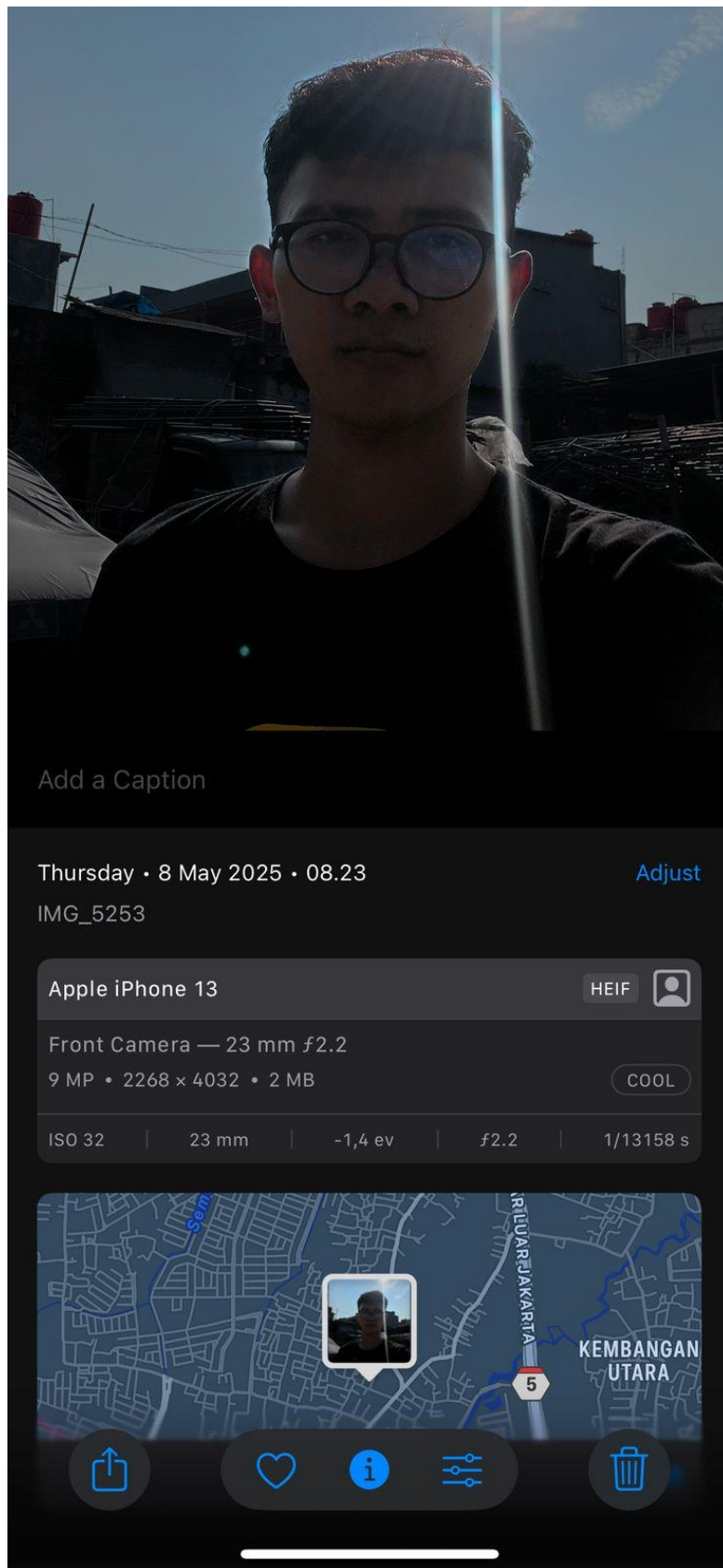
tampilkan hasil masking warna dalam bentuk subplot:

Gambar 1: hasil deteksi warna biru.

Gambar 2: kombinasi merah dan biru.

Gambar 3: kombinasi semua (merah, hijau, biru).






```
img = cv2.imread("beklait.jpg")  
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
  
plt.imshow(img_rgb)  
plt.title("Gambar Asli")  
plt.axis("off")  
plt.show()
```

baca gambar beklait.jpg dan ubah dari BGR ke RGB.

Gambar Asli



```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.imshow(gray, cmap='gray')
plt.title("Gambar Grayscale")
plt.axis("off")
plt.show()
```

Gambar diubah ke grayscale supaya fokus ke intensitas pencahayaan aja, bukan warna.

Gambar Grayscale



```
bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)

plt.imshow(bright, cmap='gray')
plt.title("Grayscale yang Dicerahkan")
plt.axis("off")
plt.show()
```

Di sini mencerahkan gambar dengan menambahkan nilai $\beta=50$.

$\alpha=1$ artinya tingkat kontras tetap.

Ini bikin semua piksel jadi lebih terang secara merata.

Grayscale yang Dicerahkan



```
contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=0)

plt.imshow(contrast, cmap='gray')
plt.title("Grayscale yang Diperkontras")
plt.axis("off")
plt.show()
```

Naikkan kontras dengan $\alpha=2.0$, yang bikin perbedaan antara piksel terang dan gelap jadi lebih ekstrem.

Cocok untuk memperjelas detail di area yang tadinya datar/kurang tegas.

Grayscale yang Diperkontras



```
bright_contrast = cv2.convertScaleAbs(bright, alpha=2.0, beta=0)

plt.imshow(bright_contrast, cmap='gray')
plt.title("Grayscale Dicerah + Kontras")
plt.axis("off")
plt.show()
```

Nah ini gabungan dari dua langkah sebelumnya.

Gambar yang sudah dicerahkan, sekarang ditingkatkan lagi kontrasnya.

Hasilnya: objek yang tadinya gelap karena backlight sekarang jadi lebih jelas dan detail muncul.

Grayscale Dicerah + Kontras



BAB IV

PENUTUP

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa proses deteksi warna dan perbaikan citra dalam pengolahan citra digital sangat bergantung pada pemahaman terhadap model warna dan representasi piksel. Dengan memanfaatkan model HSV, proses segmentasi warna menjadi lebih mudah dan akurat karena pemisahan antara hue (warna), saturation (kejenuhan), dan value (kecerahan). Teknik masking warna menggunakan `cv2.inRange()` terbukti efektif dalam mendeteksi objek berdasarkan warna dominan seperti merah, hijau, dan biru.

Selain itu, eksperimen terhadap masing-masing kanal warna RGB juga memberikan wawasan lebih dalam mengenai distribusi intensitas warna melalui histogram. Ini sangat membantu dalam proses pengambilan keputusan saat menentukan ambang batas dan melihat dominasi warna dalam gambar.

Pada bagian akhir, dilakukan proses perbaikan citra backlight dengan menaikkan nilai kecerahan dan kontras. Hasilnya menunjukkan peningkatan kualitas visual, di mana objek yang sebelumnya tidak terlihat jelas menjadi lebih terang dan detail. Teknik ini sangat bermanfaat dalam kasus pencahayaan tidak ideal pada gambar.

Secara keseluruhan, praktikum ini memberikan pemahaman praktis mengenai cara kerja dasar pengolahan citra digital menggunakan Python dan OpenCV, serta menumbuhkan kemampuan mahasiswa dalam menganalisis dan memodifikasi citra secara langsung berdasarkan kebutuhan visual.

DAFTAR PUSTAKA

- Gokce Nur Yilmaz, Selin H,macioğlu, Tahsin Üstünel.** (2022). *A Novel Color Detection Model*. Journal of Artificial Intelligence and Data Science, 2(2), 76–81.
- Pratik Garg.** (2021). *Object Tracking Using HSV Values and OpenCV*. International Journal for Research in Applied Science and Engineering Technology (IJRASET).
- Henry Dang.** (2020). *Color Detection in Python with OpenCV*.
- Gowtham.** (2021). *How to Detect Colors Using OpenCV Python*.
- StayTechRich.** (2021). *Computer Vision 001: Color Detection with OpenCV*.