UTS

PENGOLAHAN CITRA

NAMA : MUHAMMAD FADEL DESYAPUTRA

NIM : 202331098

KELAS : E

DOSEN : Dr. DARMA RUSIDI, S.T., M.Kom

NO.PC : 17

ASISTEN : 1. FAUZAN ARROYYAN

2. ABDUR RASYID RIDHO

3.

4.

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1Latar Belakang

Perkembangan teknologi komputer saat ini telah mendorong kemajuan dalam berbagai bidang, termasuk dalam pengolahan citra digital. Pengolahan citra digital (Digital Image Processing) merupakan teknik yang memungkinkan gambar digital dimanipulasi untuk berbagai tujuan, seperti peningkatan kualitas visual, ekstraksi informasi, hingga identifikasi objek. Teknologi ini telah banyak diterapkan dalam dunia medis, pertanian, robotika, keamanan, dan lain sebagainya.

Dalam praktikum Pengolahan Citra Digital, mahasiswa ditugaskan untuk menerapkan teori-teori dasar pengolahan citra dalam kasus nyata, seperti deteksi warna pada gambar, segmentasi objek berdasarkan warna, serta peningkatan kualitas citra yang mengalami backlight. Melalui praktik ini, mahasiswa diharapkan dapat memahami konsep dasar seperti model warna RGB dan HSV, histogram, thresholding, masking, serta transformasi brightness dan kontras.

1.2Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam laporan ini adalah:

1. Bagaimana cara mendeteksi dan mengekstrak warna tertentu dari citra digital menggunakan model warna RGB dan HSV?
2. Bagaimana menentukan dan mengaplikasikan ambang batas (threshold) untuk segmentasi warna dalam gambar?
3. Bagaimana cara memperbaiki citra yang memiliki pencahayaan backlight agar objek di dalam gambar terlihat lebih jelas?

1.3Tujuan Masalah

Tujuan dari praktikum dan penyusunan laporan ini adalah:

1. Menerapkan teknik deteksi warna dalam citra menggunakan model RGB dan HSV.
2. Melakukan segmentasi warna menggunakan thresholding dan masking untuk menghasilkan citra biner.
3. Melakukan peningkatan kualitas citra melalui transformasi brightness dan kontras guna memperbaiki gambar backlight.

1.4Manfaat Masalah

Adapun manfaat dari kegiatan praktikum ini antara lain:

1. Mahasiswa memperoleh pemahaman praktis dalam menggunakan berbagai teknik pengolahan citra digital.
2. Meningkatkan keterampilan dalam menggunakan alat bantu seperti Jupyter Notebook dan OpenCV untuk memanipulasi citra.
3. Memahami keterkaitan antara teori pengolahan citra dan penerapannya dalam dunia nyata, seperti pada sistem identifikasi otomatis dan perbaikan visual citra.

BAB II

LANDASAN TEORI

A. PENGOLAHAN CITRA DIGITAL

Pengolahan citra digital adalah proses manipulasi terhadap gambar digital dengan tujuan meningkatkan kualitas citra, mengekstrak informasi penting, atau mengidentifikasi objek tertentu di dalam gambar. Teknik ini bekerja dengan memproses data piksel secara matematis untuk menghasilkan interpretasi yang berguna.

Menurut Jumadi et al. (2021), pengolahan citra melibatkan pemrosesan data dua dimensi (citra) untuk keperluan identifikasi, klasifikasi, serta peningkatan kualitas gambar agar mudah diinterpretasi oleh manusia atau sistem computer.

B. MODEL WARNA RGB DAN HSV

1. RGB (Red, Green, Blue)

Model warna RGB merupakan sistem representasi warna berdasarkan pencampuran tiga warna dasar cahaya: merah, hijau, dan biru. Tiap piksel pada citra berwarna 24-bit mengandung tiga kanal intensitas warna tersebut. Pengolahan warna pada citra RGB sering dilakukan dengan cara memisahkan ketiga kanal tersebut.

Pada praktik pendeteksian warna di Bab III, metode ini digunakan untuk mengekstrak tiap warna (merah, hijau, biru) dari gambar dan mengonversinya ke grayscale agar intensitas warna dominan terlihat lebih.

2. HSV (Hue, Saturation, Value).

Model HSV lebih mendekati persepsi manusia terhadap warna.

- Hue (H) menyatakan jenis warna (misalnya merah, biru) dalam satuan derajat.
- Saturation (S) menunjukkan kejenuhan warna.
- Value (V) menunjukkan tingkat kecerahan warna.

Konversi dari RGB ke HSV penting dalam deteksi warna karena HSV memisahkan informasi warna dari pencahayaan, menjadikannya lebih stabil dalam berbagai kondisi pencahayaan.

Dalam praktiknya, nilai-nilai H tertentu digunakan untuk mendeteksi warna spesifik:

- Merah: H = 0–10 dan 170–180
- Hijau: H = 36–86
- Biru: H = 100–140

C. THRESHOLDING DAN MASKING

Thresholding adalah teknik segmentasi citra berdasarkan nilai ambang untuk memisahkan objek dari latar belakang. Dalam HSV, threshold diterapkan pada kanal Hue untuk menyorot warna tertentu.

Masking digunakan untuk mengisolasi bagian citra yang memenuhi kriteria tertentu, seperti warna yang berada dalam rentang nilai Hue yang diinginkan.

Dalam praktik UTS, tiga mask dibuat: mask merah, hijau, dan biru, serta beberapa kombinasi, untuk menghasilkan citra biner yang menampilkan area berwarna tertentu dengan warna putih (1), dan lainnya hitam (0).

D. HISTOGRAM WARNA

Histogram citra menunjukkan distribusi nilai intensitas piksel dalam sebuah gambar. Histogram warna dapat digunakan untuk menganalisis komposisi warna suatu citra, dan sangat membantu dalam membedakan warna dominan pada gambar.

Pada praktik, histogram digunakan untuk mengevaluasi intensitas masing-masing kanal warna setelah konversi ke grayscale.

E. PENINGKATAN KUALITAS CITRA

Untuk memperbaiki gambar backlight, dua teknik digunakan:

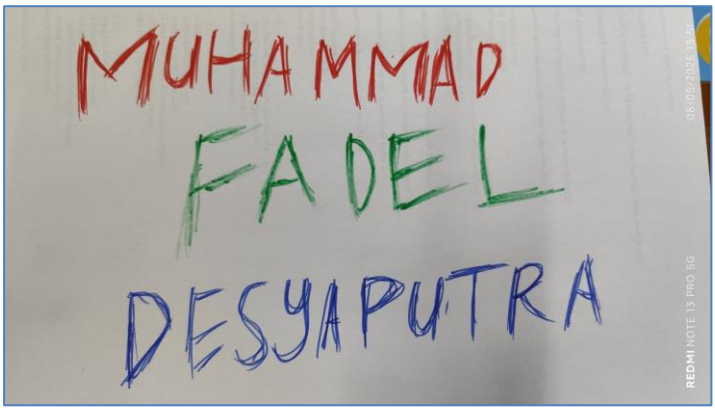
- Brightness adjustment: Menambah nilai pada setiap piksel untuk membuat gambar lebih terang.
- Contrast adjustment: Mengalikan nilai piksel untuk meningkatkan perbedaan intensitas.

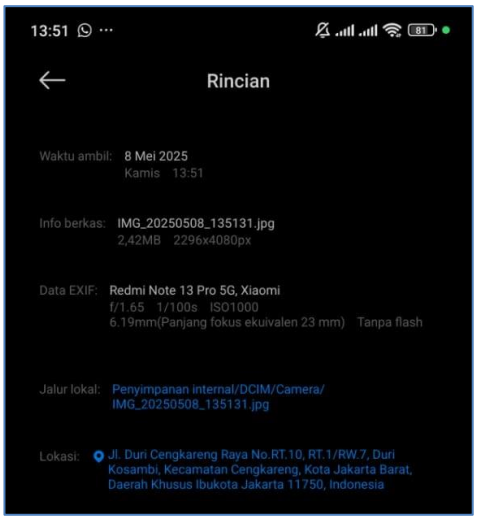
Teknik ini sesuai dengan prinsip dasar peningkatan kualitas citra agar lebih mudah diinterpretasikan oleh sistem atau manusia.

BAB III

HASIL

A. DETEKSI WARNA PADA CITRA





Soal pertama adalah mendeteksi warna pada citra. Disini kita memiliki foto yang berisikan suatu kertas yang bertuliskan nama kita dan ditulis menggunakan 3 warna, yaitu warna hijau, biru dan merah (RGB). Disini kita akan mendeteksi warna pada foto yang kita miliki..

Untuk medeteksi foto diatas, kita akan melakukan coding di jupyter agar memudahkan kita mendapatkan hasil yang kita inginkan. Untuk mendeteksi warna di jupyter, pertama-tama kita akan memasukkan library yang dibutuhkan lalu kita membaca gambar yang ingin kita deteksi warnanya menggunakan fungsi yang ada di jupyter tersebut.

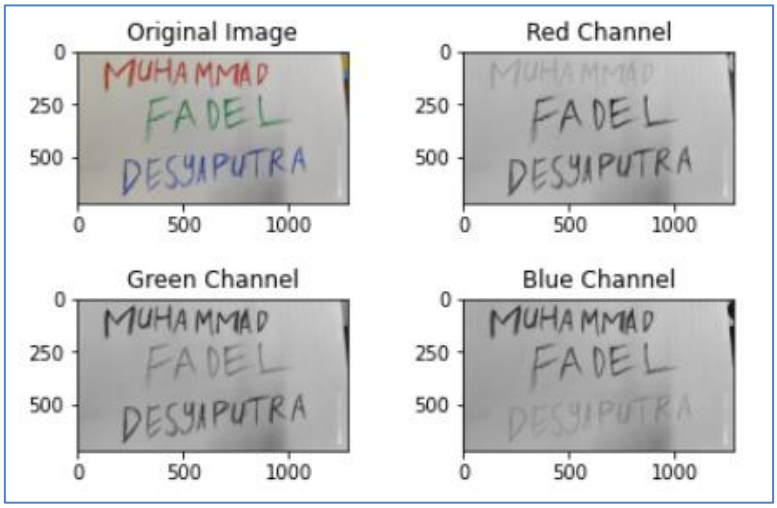
```
In [1]: import cv2
        import matplotlib.pyplot as plt
        import numpy as np

In [2]: img = cv2.imread('Nama.jpeg')
```

Setelah Membaca gambar, kita akan melakukan pendeteksian dengan cara sebagai berikut

```
In [6]: plt.subplot(2, 2, 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title('Original Image')

        plt.subplot(2, 2, 2)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 0], cmap="gray")
        plt.title('Red Channel')

        plt.subplot(2, 2, 3)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 1], cmap="gray")
        plt.title('Green Channel')

        plt.subplot(2, 2, 4)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 2], cmap="gray")
        plt.title('Blue Channel')

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

Disini, kita akan menampilkan warna dengan cara menggunakan fungsi dari cv2 dengan mengambil salah satu nilai dari RGB yang kita miliki, lalu kita baca dengan bentuk grayscale sehingga warna yang tidak dipilih tidak akan terbaca.

Hasil :

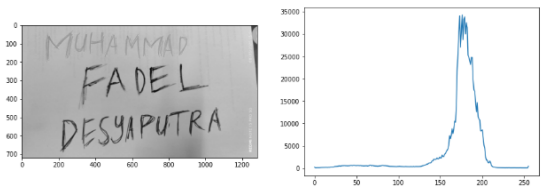


Dengan mengambil tiap warna dan dibaca dengan bentuk grayscale. Maka, warna yang dipilih yang ditampilkan akan terlihat lebih terang dikarenakn nilainya yang tinggi, contoh : kita memilih mengambil nilai merah, maka gambar akan menunjukkan setiap yang berwarna merah akan menjadi paling cerah diantara warna lainnya.

Untuk histogramnya sendiri bis akita lihat di bawah ini

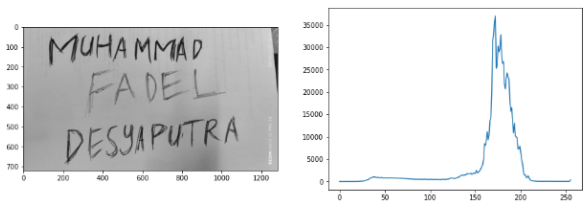
- Merah

```
In [7]: #Merah
merah=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,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()
```



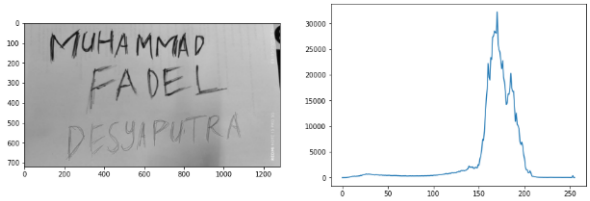
- Hijau

```
In [8]: #Hijau
hijau=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,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()
```



- Biru

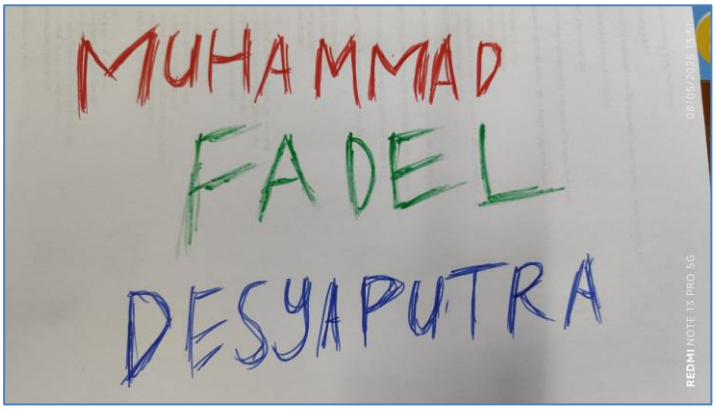
```
In [9]: #Biru
biru=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,2]
fig, axs = plt.subplots(1,2, figsize = (15,5))
hist = cv2.calcHist([biru],[0],None,[256],[0,256])
axs[0].imshow(biru, cmap='gray')
axs[1].plot(hist)
plt.show()
```

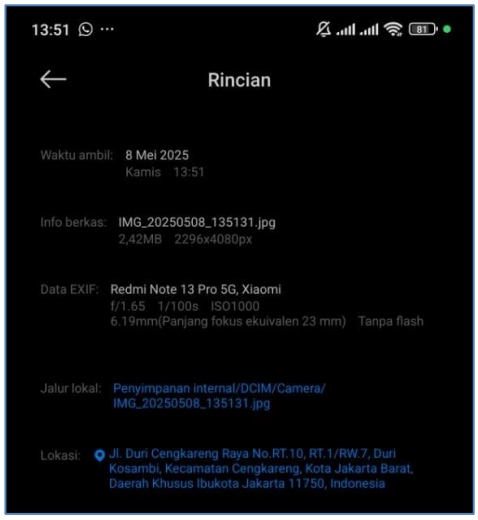


Bisa dilihat dari histogram perbedaan nilai di tiap warnanya yang mana tiap gambar memiliki nilai warna yang berbeda-beda.

B. MENCARI DAN MENGURUTKAN AMBANG BATAS

Foto :





Disini kita akan mencari ambang batas untuk tiap warnanya. Untuk mencarinya, kita dapat lakukan hal berikut.

```
image_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

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

fig, axs = plt.subplots(2, 2, figsize=(15,15))

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])
```

Pertama-tama, kita akan mengubah warna menjadi bentuk HSV terlebih dahulu agar mudah dideteksi warnanya untuk ditentukan ambangnya. HSV memiliki perbedaan dengan RGB yang mana R adalah merah, G adalah hijau dan B adalah biru. Pada HSV, H adalh hue(warna), S adalah saturation (kejenuhan) dan V adalah value (kecerahan).

Disini kita juga membuat plot kosong dengan ukuran 2x2 atau 4 kota kosong. Disini juga kita menentukan rentang warna HSV-nya tadi. Di dalam kode, kita memisahkan nilai dengan cara jika nilai H = 0-10 dan 170-180 maka warna adalah merah, dan jika nilai H = 36-86 maka warna adalah merah, dan jika nilai H = 100-140 maka warna adalah biru.

```
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)
```

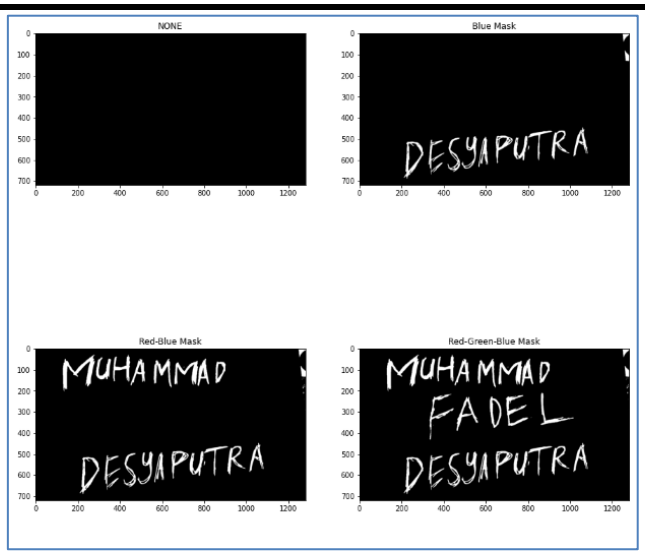
Disini kita mengubah warna yang telah kita ubah menjadi hsv tadi menjadi bentuk biner, disini kita membuat 3 mask, yang pertama mask_red sebagai penanda warna merah, yang kedua mask_green sebagai penanda warna hijau, yang ketiga mask_blue sebagai penanda warna Biru. Mask disini berguna untuk mengkhususkan warna yang akan ditampilkan dan diubah menjadi biner nantinya

Setelah mengubah menjadi mask, kita membuat mask campuran dengan mencampurkan yang merah dan biru untuk mask campuran pertama dan kita campurkan ketiga-tiganya untuk mask campuran kedua.

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

Disini kita mengubah mask yang telah dibuat menjadi bentuk biner. Setelah diubah menjadi biner, barulah disini kita menampilkan gambar tersebut.

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

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

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

plt.show()
```



Disini, maksud blue mask adalah ketika nilai hue sebelumnya diantara 100-140, maka gambar akan menjadi putih dan ketika diluar nilai tersebut maka akan menjadi hitam. Begitu pula yang campuran seperti red-blue yang mana ketika nilai hue antara 0-10, 100-140 dan 170-180 maka pikselnya akan menjadi warna putih, dan ikalau selain nilai tersebut, maka piksel menjadi warna hitam.

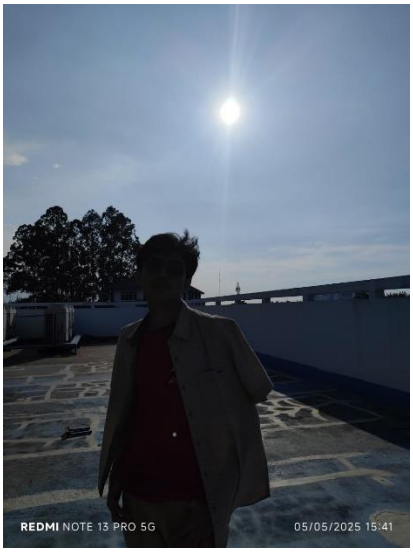
Adapun tujuan utama kita yang ingin mencari dan mengurutkannya adalah dengan menyesuaikan dengan nilai hue yang ada dan telah disebutkan sebelumnya, yaitu :

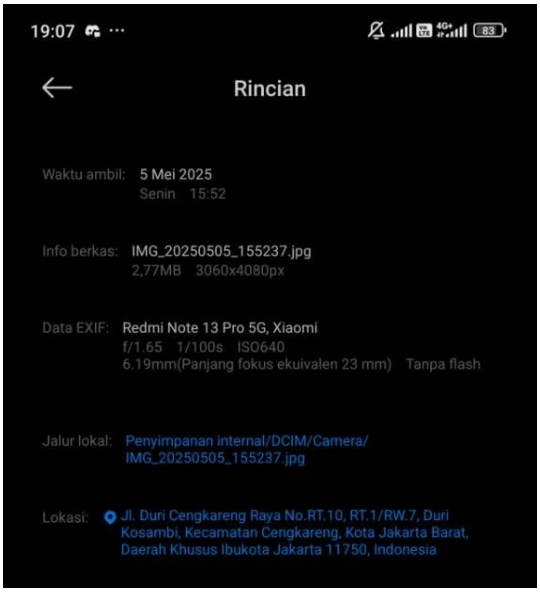
- Merah = hue 0-10 dan 170-180
- Hijau = hue 36-86
- Biru = 170-180

Hal ini didapat karena sudah dari sananya nilai hue pada HSV seperti begitu.

C. MEMPERBAIKI GAMBAR BACKLIGHT

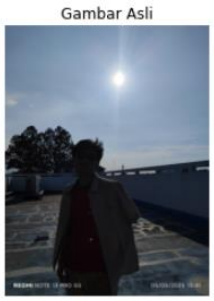
Foto :





Disini kita memiliki foto yang membelakangi cahaya atau biasa disebut backlight. Tugas saya disini adalah untuk mengubah foto tersebut agar bisa memperlihatkan wajah dari orang yang ada di foto. Untuk caranya, adalah sebagai berikut :

```
In [11]: img = cv2.imread("Foto.jpeg")
         img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

         plt.imshow(img_rgb)
         plt.title("Gambar Asli")
         plt.axis("off")
         plt.show()
```



Pertama-tama kita membaca file dari foto yang kita miliki. Diatas dapat dilihat adalah kode untuk menampilkan foto dan foto yang digunakan untuk diproses. Setelah itu, kita mengubah foto menjadi bentuk grayscale agar foto kita mudah untuk diproses nantinya.

```
In [12]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

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

Setelah dijadikan abu-abu, kita akan membuat fotonya menjadi cerah dengan menambah tiap nilai di fotonya dengan nilai 70. Hal ini dilakukan agar wajah orang di foto dapat terlihat lebih jelas.

```
In [20]: bright = cv2.convertScaleAbs(gray, alpha=1, beta=70)

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

Selain mencerahkan, untuk membuat wajah orang di foto menjadi terlihat lebih jelas kita bisa meningkatkan kontras dengan mengkalikan setiap nilai di foto dengan 3.

```
In [26]: contrast = cv2.convertScaleAbs(gray, alpha=3.0, beta=0)

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

Grayscale yang Diperkontras

Agar foto terlihat benar2 lebih jelas, kita juga dapat menggabungkan pencerahan dengan meningkatkan kontras. Untuk melakukan operasi campuran seperti itu, kita dapat menjumlahkan tiap nilai di foto terlebih dahulu lalu mengkalikan tiap nilai di foto. Disini kita menggunakan foto yang telah dicerahkan sebelumnya dan kita kalikan sendiri disini dengan 2,5. Berikut hasilnya:

```
In [22]: bright_contrast = cv2.convertScaleAbs(bright, alpha=2.5, beta=0)

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

Grayscale Dicerah + Kontras

Untuk perbandingan tiap hasil sebagai berikut :

```
In [16]: fig, axs = plt.subplots(1,5, figsize=(50,50))
         axs[0].imshow(img_rgb)
         axs[1].imshow(gray, cmap='gray')
         axs[2].imshow(bright, cmap='gray')
         axs[3].imshow(contrast, cmap='gray')
         axs[4].imshow(bright_contrast, cmap='gray')

Out[16]: <matplotlib.image.AxesImage at 0x261f8996790>
```

BAB IV
PENUTUP

A. KESIMPULAN

Berdasarkan teori yang dipelajari dan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa pengolahan citra digital merupakan pendekatan efektif dalam mengenali dan memanipulasi objek pada gambar berdasarkan fitur warna dan pencahayaan. Praktikum ini berhasil menerapkan beberapa teknik dasar pengolahan citra, di antaranya:

1. Deteksi warna menggunakan model RGB dan HSV, di mana RGB digunakan untuk mengekstrak kanal warna dasar dan HSV digunakan untuk segmentasi warna berdasarkan nilai Hue. Penerapan threshold pada kanal Hue memungkinkan pemisahan warna merah, hijau, dan biru secara akurat.

2. Pembuatan masking dan konversi ke citra biner terbukti mampu menyorot bagian citra yang sesuai dengan warna tertentu. Teknik ini efektif untuk proses segmentasi warna dan mempermudah analisis objek.

3. Histogram warna memberikan gambaran visual mengenai distribusi intensitas tiap kanal warna, yang sangat berguna untuk mengidentifikasi dominasi warna dalam gambar.

4. Perbaikan gambar backlight melalui penyesuaian brightness dan kontras menunjukkan bahwa transformasi nilai piksel dapat meningkatkan kualitas visual citra. Kombinasi pencerahan dan kontras mampu menampilkan detail objek yang sebelumnya tidak terlihat dengan jelas.

Secara keseluruhan, praktikum ini memberikan pemahaman yang kuat mengenai dasar-dasar pengolahan citra digital dan menunjukkan relevansi nyata dari teori-teori yang dipelajari terhadap permasalahan dalam dunia nyata, seperti identifikasi warna dan peningkatan kualitas gambar.

B. SARAN

Untuk pengembangan lebih lanjut, disarankan agar:

- Praktikum dilengkapi dengan analisis kuantitatif (seperti akurasi segmentasi).
- Hasil output citra disimpan dan disajikan dalam laporan agar evaluasi visual dapat dilakukan lebih objektif.
- Dieksplorasi teknik pengolahan lanjutan seperti deteksi tepi atau klasifikasi objek menggunakan metode clustering.

DAFTAR PUSTAKA

Aditya, M. R., Husni, N. L., Pratama, D. A., & Handayani, A. S. (2020). Penerapan Sistem Pengolahan Citra Digital Pendeteksi Warna pada Starbot. *TEKNIKA*.

Fadjeri, A., Saputra, B. A., Arianto, D. K., & Kurniatin, L. (2022). Karakteristik Morfologi Tanaman Selada Menggunakan Pengolahan Citra Digital. *Jurnal Ilmiah SINUS (JIS)*.

Jumadi, J., Yupianti, & Sartika, D. (2021). PENGOLAHAN CITRA DIGITAL UNTUK IDENTIFIKASI OBJEK MENGGUNAKAN METODE HIERARCHICAL AGGLOMERATIVECLUSTERING. *Jurnal Sains & Teknologi*.