UTS

PENGOLAHAN CITRA

NAMA : Diaz Andhika Nugraha

NIM : 202331123

KELAS : D

DOSEN : Darma Rusjdi., Ir., M.Kom

NO.PC : 14

ASISTEN : 1. Fakhrul Fauzi Nugraha Tarigan

2. Muhammad Hanief Febriansyah

3. Clarenca Sweetdiva Pereira

4. Sakura Amastasya Salsabila Setiyanto

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Bagaimana cara mendeteksi warna biru, merah, dan hijau secara terpisah dalam sebuah citra digital?
2. Bagaimana menentukan nilai ambang batas (threshold) untuk mengelompokkan piksel ke dalam kategori warna tertentu?
3. Bagaimana teknik pengolahan citra dapat digunakan untuk memperbaiki citra yang memiliki pencahayaan belakang (backlight)?

1.2 Tujuan Masalah

1. Mendeteksi dan menampilkan komponen warna biru, merah, dan hijau secara terpisah dari sebuah citra RGB.
2. Menentukan nilai ambang batas intensitas warna agar citra dapat diklasifikasikan berdasarkan kategori warnanya.
3. Menerapkan teknik pengolahan citra untuk meningkatkan kualitas gambar backlight melalui penyesuaian kecerahan dan kontras.

1.3 Manfaat Masalah

1. Memberikan pemahaman tentang cara kerja ekstraksi kanal warna (R, G, B) dalam citra digital.
2. Membantu dalam proses segmentasi warna berdasarkan ambang batas untuk keperluan analisis objek atau klasifikasi.
3. Meningkatkan kemampuan dalam memperbaiki kualitas visual gambar dengan kondisi pencahayaan yang buruk, khususnya backlight.

BAB II

LANDASAN TEORI

2.1 Deteksi warna biru, merah, dan hijau pada gambar

Ruang warna adalah sistem untuk merepresentasikan dan menggambarkan warna dalam gambar atau citra digital. Setiap ruang warna memiliki cara tersendiri untuk mengekspresikan warna-warna yang berbeda, dan digunakan untuk berbagai tujuan seperti pemrosesan gambar, analisis warna, kompresi gambar, dan pengenalan objek. Dalam pengolahan citra menggunakan MATLAB, dua ruang warna yang sering digunakan adalah Ruang Warna RGB (Red, Green, Blue) dan Ruang Warna YCbCr (Luminance, Chrominance Blue, Chrominance Red).

Ruang warna RGB menggunakan tiga komponen warna utama: merah (Red), hijau (Green), dan biru (Blue), untuk mendefinisikan warna dalam sebuah gambar. Setiap piksel dalam citra RGB diwakili oleh tiga nilai intensitas, yaitu untuk warna merah (R), hijau (G), dan biru (B). Rentang nilai untuk masing-masing komponen ini umumnya berkisar dari 0 hingga 255, di mana nilai 0 menunjukkan ketiadaan warna dan nilai 255 menunjukkan intensitas warna maksimum.

Citra RGB (Red, Green, Blue) adalah jenis citra yang paling sering digunakan dalam pengolahan citra digital. Citra ini melibatkan tiga saluran warna: merah (R), hijau (G), dan biru (B). Setiap piksel dalam citra RGB diwakili oleh kombinasi intensitas dari ketiga saluran ini. Kombinasi intensitas dari ketiga saluran ini menciptakan spektrum warna yang luas, yang dapat menghasilkan warna yang berbeda.

2.2 Mencari nilai ambang batas citra untuk dapat menampilan kategori warna pada citra

Thresholding adalah salah satu metode segmentasi citra yang memisahkan antara obyek dengan background dalam suatu citra berdasarkan pada perbedaan kecerahannya atau gelap terangnya. Region citra yang cenderung gelap akan dibuat semakin gelap, sedangkan region citra yang cenderung terang akan dibuat semakin terang . Thresholding digunakan untuk men-segmentasi gambar dengan pengaturan semua piksel yang nilai intensitasnya di atas ambang batas nilai menjadi latar depan dan semua piksel yang tersisa menjadi latar belakang .

Salah satu metode perbaikan citra yang dapat digunakan adalah Metode Thresholding (pengambangan). Secara sederhana metode ini akan memisahkan tingkat keabuan citra berdasarkan batas ambang tertentu

Metode ini cocok digunakan untuk perbaikan citra dokumen dikarenakan biasanya hanya dibutuhkan dua warna yaitu warna hitam untuk isi dokumen dan warna putih untuk latar belakang

Penelitian diawali dengan memasukkan input citra digital. Citra digital yang dimasukkan bisa berupa citra berwarna maupun grayscale dengan format JPEG atau JPG. Langkah selanjutnya adalah mengubah citra inputan menjadi citra grayscale. Jika citra inputan adalah citra berwarna, maka harus dikonversi dahulu menjadi citra grayscale. Sedangkan jika citra inputan sudah dalam bentuk citra grayscale, maka tidak dilakukan proses konversi. Setelah itu dilakukan proses filtering citra menggunakan metode-metode Thresholding

2.3 Grayscale, grayscale dipercerah,grayscale diperkontras, greyscale dipercerah dan diperkontras

Citra biner adalah jenis citra yang hanya memiliki dua nilai intensitas yang mungkin untuk setiap piksel, yaitu hitam atau putih, biasanya direpresentasikan dengan 0 (hitam) dan 1 (putih). Citra biner dihasilkan dari proses thresholding, di mana nilai intensitas piksel dalam citra grayscale dikonversi menjadi nilai biner berdasarkan ambang tertentu. Jika nilai piksel melebihi ambang tersebut, piksel tersebut dianggap putih (objek), dan jika tidak, dianggap hitam (latar belakang).

Citra Grayscale adalah jenis citra di mana setiap pikselnya direpresentasikan oleh satu saluran warna yang hanya menyatakan tingkat kecerahan atau intensitas. Dalam citra Grayscale, informasi warna diabaikan dan hanya informasi kecerahan yang dipertahankan. Setiap piksel dalam citra Grayscale memiliki nilai intensitas tunggal yang berkisar dari 0 (hitam) hingga 255 (putih) di mana 0 adalah nilai kecerahan tergelap (hitam) dan 255 adalah nilai kecerahan paling terang (putih).

Ada tiga hal yang perlu diperhatikan dalam proses perbaikan citra, yaitu brightness, colormap, dan contrast. Ketiga hal tersebut mempunyai fungsi dan struktur fungsi masing-masing yang sangat berguna untuk mengolah gambar sesuai yang perlukan.Proses brightnessbertujuan untuk mendapatkan warna gambar terang dan gelap. Proses brightness mengubah komposisi warna pada gambar menjadi warna putih terang atau warna gelap sesuai dengan intensitas warna yang diinginkan. Fungsi brightness pertama-tama akan menentukan kecerahan dari gambar.

Proses contrast pada gambar bertujuan untuk mengatur color gray scale pada gambar. Contrast akan bekerja dengan baik ketika warna gambar dipadukan dengan intensitas kecerahan Pengujian ini dilakukan untuk mengetahui tingkat keberhasilan. Selain itu dengan adanya pengujian, dapat diketahui adanya kelemahan atau kekuranganyang ada pada citra, sehingga dapat dilakukan beberapa perbaikan yang diperlukan. Titik berat pengujian terletak pada pengujian perangkat lunak saja. Sedangkan untuk perangkat kerasnya yang merupakan piranti pendukung, tidak akan dibahas, karena perangkat keras sudah ada dalam bentuk jadi. Brigthness, Proses ini akan memberikan tingkat kecerahan pada suatu citra. Dengan menaikan nilai brightness menjadi tinggi grayscale mengarah ke 255 sehingga citra menjadi lebih terang.

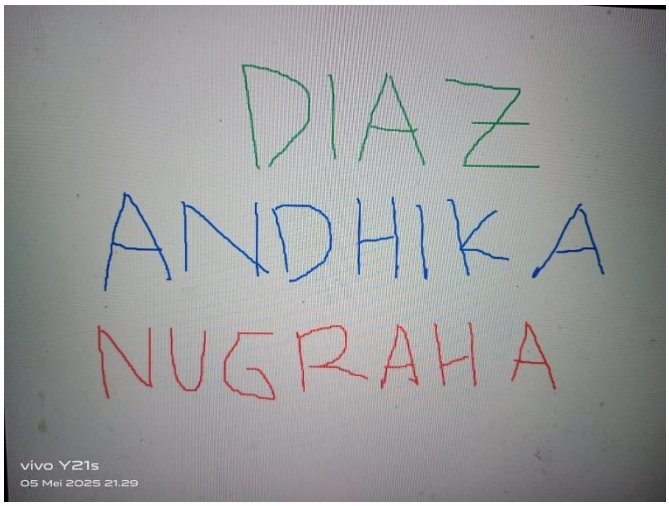
Contrast, Proses ini mengatur color gray scale pada gambar. Contrast akan bekerja dengan baik ketika warna gambar dipadukan dengan intensitas kecerahan. Perubahan citra ini dapat dilihat seperti gambar di bawah ini.

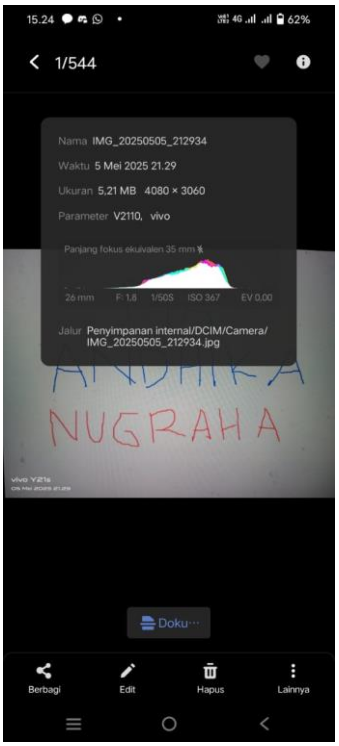
BAB III

HASIL

3.1 Deteksi Warna pada Citra

Gambar yang dilampirkan :





1. Import library

```
import cv2
import matplotlib.pyplot as plt
```

- cv2 dari OpenCV: digunakan untuk membaca dan mengolah gambar.
- matplotlib.pyplot: digunakan untuk menampilkan gambar dan histogram.

2. Baca dan Konversi Gambar

```
[27]: image_path = 'RGB_UTS.png'
      image = cv2.imread(image_path)
```

```
[29]: image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

- Membaca file gambar (RGB_UTS.png).
- OpenCV membaca gambar dalam format BGR, jadi perlu dikonversi ke format RGB agar warna ditampilkan dengan benar di matplotlib.

3. Ekstraksi kanal warna

```
[31]: r = image_rgb[:, :, 0]
      g = image_rgb[:, :, 1]
      b = image_rgb[:, :, 2]
```

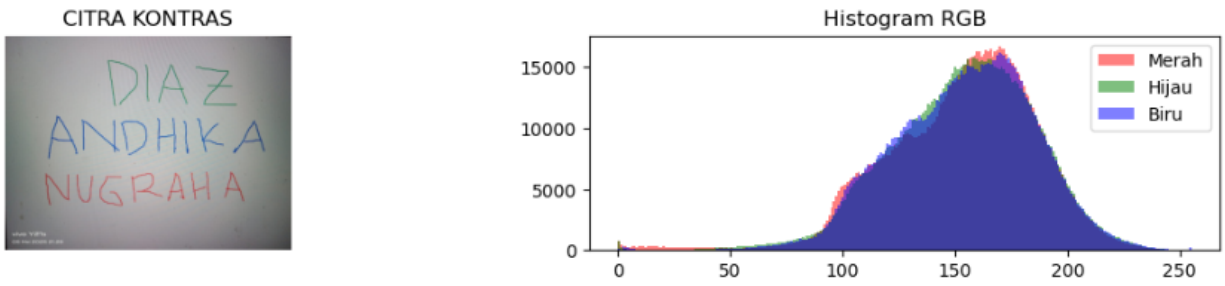
- Memisahkan **kanal Merah (Red)**, **Hijau (Green)**, dan **Biru (Blue)** dari citra RGB.
- [:, :, 0] artinya ambil semua baris dan kolom dari kanal ke-0 (merah), dst.

4. Citra Kontras dan Histogram RGB

```
[33]: plt.figure(figsize=(14, 10))

      plt.subplot(4, 2, 1)
      plt.imshow(image_rgb)
      plt.title("CITRA KONTRAS")
      plt.axis('off')

      plt.subplot(4, 2, 2)
      plt.title("Histogram RGB")
      plt.hist(r_channel.ravel(), bins=256, color='red', alpha=0.5, label='Merah')
      plt.hist(g_channel.ravel(), bins=256, color='green', alpha=0.5, label='Hijau')
      plt.hist(b_channel.ravel(), bins=256, color='blue', alpha=0.5, label='Biru')
      plt.legend()
```

- Membuat canvas besar untuk menampung semua subplot.
- Menampilkan citra RGB asli.
- Menampilkan histogram gabungan RGB.
- `.ravel()` mengubah matriks 2D menjadi array 1D agar bisa dihitung histogramnya.

```
[33]: <matplotlib.legend.Legend at 0x1ea20c481a0>
```

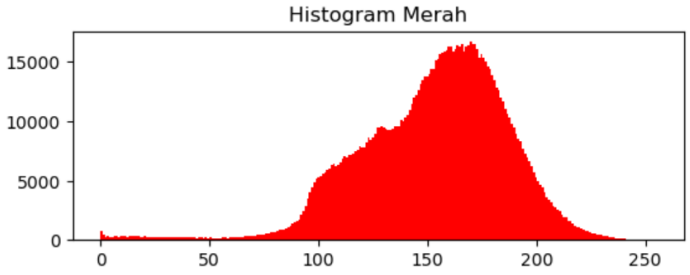


5. Kanal dan histogram Merah

```
[35]: plt.figure(figsize=(14, 10))

      plt.subplot(4, 2, 3)
      plt.imshow(r_channel, cmap='gray')
      plt.title("MERAH")
      plt.axis('off')

      plt.subplot(4, 2, 4)
      plt.hist(r_channel.ravel(), bins=256, color='red')
      plt.title("Histogram Merah")
```

- Menampilkan gambar kanal merah dalam gradasi abu-abu.
- Menampilkan histogram intensitas merah.

```
[35]: Text(0.5, 1.0, 'Histogram Merah')
```

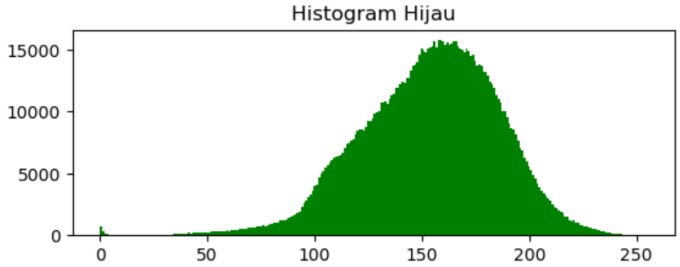


6. Kanal dan histogram Hijau

```
[37]: plt.figure(figsize=(14, 10))

      plt.subplot(4, 2, 5)
      plt.imshow(g_channel, cmap='gray')
      plt.title("HIJAU")
      plt.axis('off')

      plt.subplot(4, 2, 6)
      plt.hist(g_channel.ravel(), bins=256, color='green')
      plt.title("Histogram Hijau")
```

- Menampilkan gambar kanal hijau dalam gradasi abu-abu.
- Menampilkan histogram intensitas hijau.

```
[37]: Text(0.5, 1.0, 'Histogram Hijau')
```

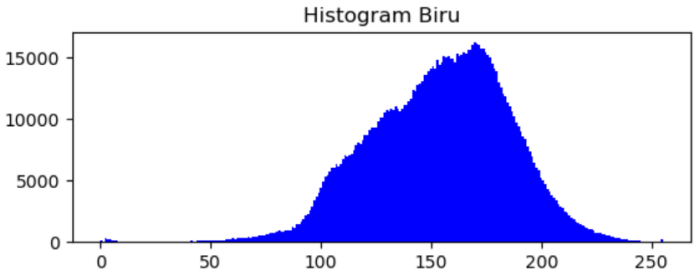


7. Kanal dan histogram Biru

```
[47]: plt.figure(figsize=(14, 10))

      plt.subplot(4, 2, 7)
      plt.imshow(b_channel, cmap='gray')
      plt.title("BIRU")
      plt.axis('off')

      plt.subplot(4, 2, 8)
      plt.hist(b_channel.ravel(), bins=256, color='blue')
      plt.title("Histogram Biru")
```

- Menampilkan gambar kanal biru dalam gradasi abu-abu.
- Menampilkan histogram intensitas biru.

```
[47]: Text(0.5, 1.0, 'Histogram Biru')
```



3.2 Mencari dan mengurutkan ambang batas terkecil sampai dengan terbesar

1. Import Library

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- **cv2**: untuk membaca dan memproses gambar/video.
- **numpy**: untuk operasi matematis dan array (struktur data gambar).

- **matplotlib.pyplot**: untuk menampilkan gambar dalam notebook atau GUI berbasis grafik.

2. Membaca gambar

```
image = cv2.imread('RGB_UTS.png')
img_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

- Fungsi cv2.imread() secara default akan membaca gambar dalam format BGR (Blue, Green, Red), bukan RGB seperti umumnya pada Matplotlib atau PIL.
- Fungsi ini digunakan untuk **mengonversi format warna** gambar dari BGR ke RGB.

3. Mendefinisikan ambang batas untuk setiap warna
   - Merah

```
[32]: lower_red = np.array([100, 0, 0])
      upper_red = np.array([255, 100, 100])
      mask_red = cv2.inRange(img_rgb, lower_red, upper_red)
```

- lower_red dan upper_red menentukan batas warna merah dalam RGB.
- cv2.inRange() menghasilkan masker biner berdasarkan rentang warna tersebut.
- Masker ini bisa digunakan untuk:
  - Menyorot area warna tertentu.
  - Membuat segmentasi warna.
  - Deteksi objek dengan warna spesifik dalam gambar.

- Hijau

```
[34]: lower_green = np.array([0, 100, 0])
      upper_green = np.array([100, 255, 100])
      mask_green = cv2.inRange(img_rgb, lower_green, upper_green)
```

- lower_red dan upper_red menentukan batas warna hijau dalam RGB.
- cv2.inRange() menghasilkan masker biner berdasarkan rentang warna tersebut.
- Masker ini bisa digunakan untuk:
  - Menyorot area warna tertentu.
  - Membuat segmentasi warna.
  - Deteksi objek dengan warna spesifik dalam gambar.
- Biru

```
[36]: lower_blue = np.array([0, 0, 100])
      upper_blue = np.array([100, 100, 255])
      mask_blue = cv2.inRange(img_rgb, lower_blue, upper_blue)
```

- lower_red dan upper_red menentukan batas warna biru dalam RGB.
- cv2.inRange() menghasilkan masker biner berdasarkan rentang warna tersebut.

- Masker ini bisa digunakan untuk:
  - Menyorot area warna tertentu.
  - Membuat segmentasi warna.
  - Deteksi objek dengan warna spesifik dalam gambar.

4. Gabungkan kombinasi channel

```
[39]: none_mask = np.zeros_like(mask_red)                    # NONE
      blue_mask = mask_blue                                  # BLUE
      red_blue_mask = cv2.bitwise_or(mask_red, mask_blue)    # RED-BLUE
      rgb_mask = cv2.bitwise_or(red_blue_mask, mask_green)   # RED-GREEN-BLUE
```

- Membuat masker kosong (semua nilai 0 / hitam) dengan ukuran yang sama seperti mask_red.
- Menyimpan masker biru ke dalam variabel blue_mask.
- Menggabungkan dua masker: mask_red dan mask_blue dengan operasi bitwise OR.
- Menggabungkan hasil red_blue_mask dengan mask_green (masker hijau).

5. Menampilkan gambar dan histogram

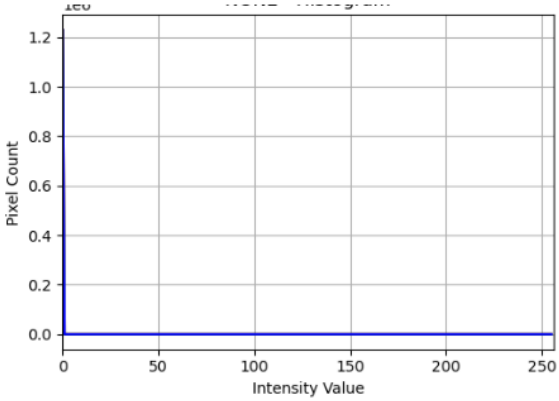
```
[42]: def show_image_and_hist(mask, title):
          rgb_mask = cv2.merge([mask, mask, mask])  # Convert grayscale to RGB for histogram
          color = ('r', 'g', 'b')

          fig, axs = plt.subplots(1, 2, figsize=(10, 4))

          # Tampilkan gambar
          axs[0].imshow(mask, cmap='gray')
          axs[0].set_title(f'{title} - Mask')
          axs[0].axis('off')

          # Tampilkan histogram
          for i, col in enumerate(color):
              hist = cv2.calcHist([rgb_mask], [i], None, [256], [0, 256])
              axs[1].plot(hist, color=col)
              axs[1].set_xlim([0, 256])
          axs[1].set_title(f'{title} - Histogram')
          axs[1].set_xlabel('Intensity Value')
          axs[1].set_ylabel('Pixel Count')
          axs[1].grid(True)

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

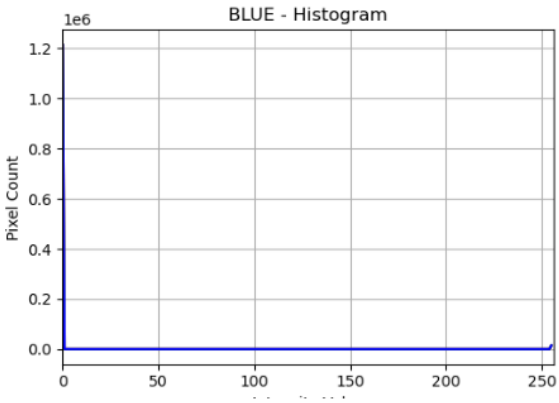
- Menampilkan gambar masker biner dalam bentuk grayscale.
- Menampilkan histogram untuk saluran warna merah, hijau, dan biru dari gambar yang diubah menjadi format RGB.
- Memberikan visualisasi yang informatif mengenai distribusi intensitas piksel dalam gambar biner.

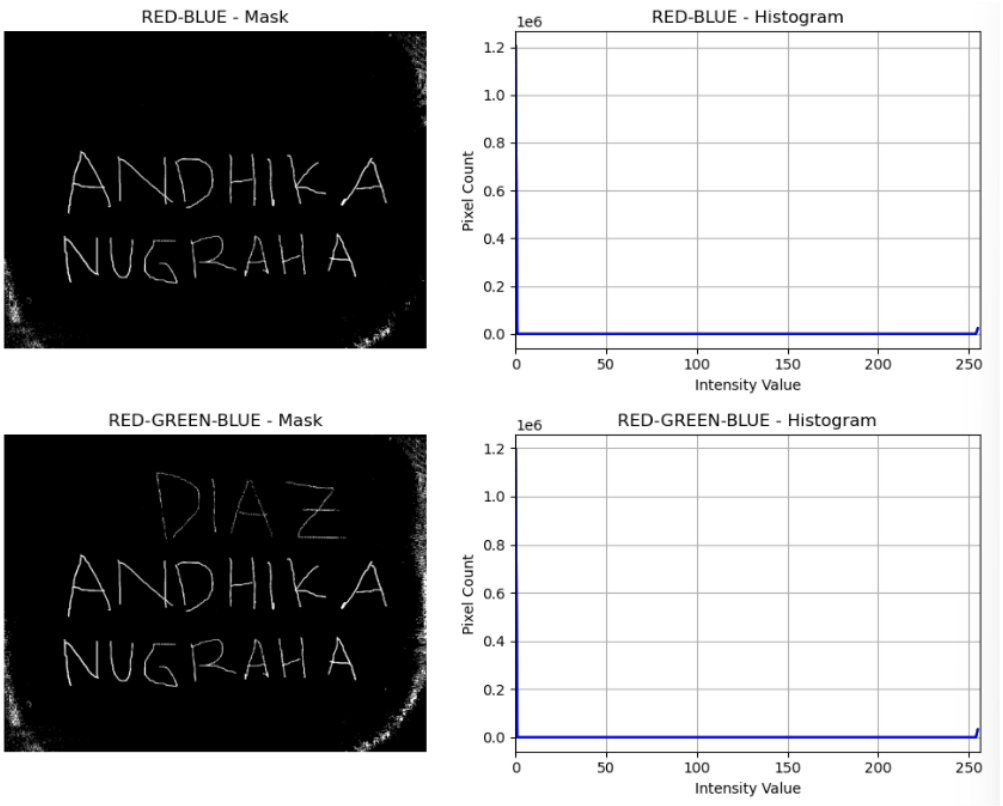
6. Tampilkan hasil

```
show_image_and_hist(none_mask, "NONE")
show_image_and_hist(blue_mask, "BLUE")
show_image_and_hist(red_blue_mask, "RED-BLUE")
show_image_and_hist(rgb_mask, "RED-GREEN-BLUE")
```

- none_mask: Menampilkan masker kosong.
- blue_mask: Menampilkan masker area biru.
- red_blue_mask: Menampilkan masker gabungan merah dan biru.
- rgb_mask: Menampilkan masker untuk gabungan merah, hijau, dan biru.







3.3 Memperbaiki Gambar Backlight

1. Import library

```
[7]: from PIL import Image, ImageEnhance, ImageOps
     import matplotlib.pyplot as plt
```

- PIL (Python Imaging Library) adalah pustaka untuk memproses dan memanipulasi gambar di Python. PIL kini sering digantikan dengan Pillow, yang merupakan versi fork dan lebih aktif dikembangkan dari PIL.
- Image: Merupakan modul utama dalam PIL untuk membuka, memodifikasi, dan menyimpan gambar.
- ImageEnhance: Digunakan untuk melakukan berbagai penyesuaian gambar seperti kontras, kecerahan, ketajaman, dll.
- ImageOps: Menyediakan berbagai operasi gambar seperti pembalikan warna, pemotongan tepi, atau pengolahan gambar dengan efek khusus.

2. Membaca gambar

```
[9]: image_path = 'BL_UTS.jpg'
     original_image = Image.open(image_path)
```

- Menyimpan path atau alamat file gambar yang ingin dibuka dalam variabel image_path.
- Image.open() digunakan untuk membuka gambar yang terletak di path 'BL_UTS.jpg'.

3. Konversu ke grayscale

```
[11]: grayscale_image = ImageOps.grayscale(original_image)
```

- ImageOps.grayscale(): Fungsi ini digunakan untuk mengonversi gambar warna (RGB) menjadi grayscale (hitam-putih).
- Fungsi ini menghilangkan informasi warna dan hanya menyimpan informasi intensitas cahaya dari gambar.
- Hasilnya adalah gambar dalam format grayscale, yang disimpan dalam variabel grayscale_image.

4. Grayscale yang dipercerah

```
[16]: only_bright = ImageEnhance.Brightness(grayscale_image).enhance(1.5)
```

- ImageEnhance.Brightness(grayscale_image): Fungsi ini digunakan untuk menyesuaikan kecerahan gambar. Objek ImageEnhance.Brightness() memungkinkan kamu untuk mengubah kecerahan gambar.
- enhance(1.0): Angka 1.0 menunjukkan tidak ada perubahan pada kecerahan gambar. Jika nilai ini lebih besar dari 1, gambar akan menjadi lebih terang, dan jika lebih kecil dari 1, gambar akan menjadi lebih gelap.
    - 1.0 berarti gambar akan tetap dengan kecerahan aslinya.
    - Nilai lebih besar dari 1.0 akan meningkatkan kecerahan.
    - Nilai lebih kecil dari 1.0 akan mengurangi kecerahan.

5. Graysacle yang diperkontrass

```
[15]: only_contrast = ImageEnhance.Contrast(grayscale_image).enhance(1.0)
```

- ImageEnhance.Contrast(grayscale_image): Fungsi ini digunakan untuk menyesuaikan kontras gambar. Objek ImageEnhance.Contrast() memungkinkan kamu untuk mengubah kontras gambar.
- enhance(1.5): Nilai 1.5 meningkatkan kontras gambar.
    - Nilai 1.0 berarti kontras tidak berubah (gambar tetap seperti aslinya).
    - Nilai yang lebih besar dari 1.0, seperti 1.5, meningkatkan kontras, membuat perbedaan antara area terang dan gelap menjadi lebih tajam.
    - Nilai lebih kecil dari 1.0 (misalnya 0.5) akan mengurangi kontras, menghasilkan gambar yang lebih datar atau lebih kabur antara area terang dan gelap.

6. Graysacle yang dipercerah dan diperkontras

```
[17]: bright_then_contrast = ImageEnhance.Contrast(only_bright).enhance(1.0)
```

- ImageEnhance.Contrast(only_bright):

- Ini pertama-tama mengambil gambar yang telah ditingkatkan kecerahannya (only_bright) dan membuat objek untuk menyesuaikan kontras gambar tersebut.

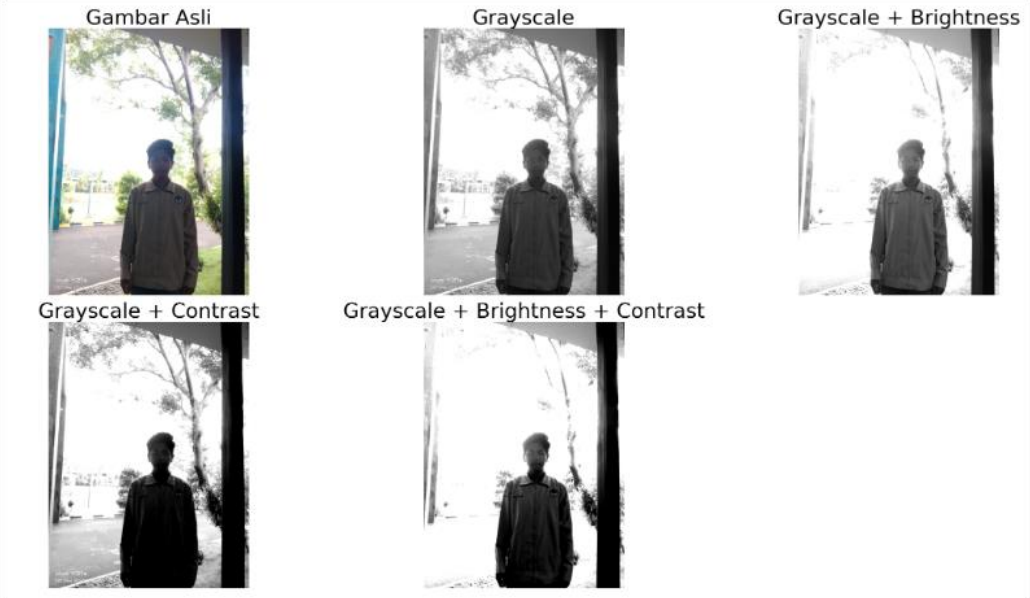
- enhance(1.5):
  - Nilai 1.5 digunakan untuk meningkatkan kontras gambar yang sudah lebih terang (only_bright).
  - Kontras akan lebih ditingkatkan, membuat perbedaan antara area terang dan gelap semakin tajam setelah gambar dibuat lebih terang.

7. Tampilkan Gambar

[21]:
```
fig, axs = plt.subplots(2, 3, figsize=(24, 12))  # Ukuran diperbesar

# Gambar asli
axs[0, 0].imshow(original_image)
axs[0, 0].set_title("Gambar Asli" , fontsize=30 )
axs[0, 0].axis("off")

# Grayscale
axs[0, 1].imshow(grayscale_image, cmap='gray')
axs[0, 1].set_title("Grayscale",fontsize=30 )
axs[0, 1].axis("off")

# Grayscale + Brightness
axs[0, 2].imshow(only_bright, cmap='gray')
axs[0, 2].set_title("Grayscale + Brightness", fontsize=30)
axs[0, 2].axis("off")

# Grayscale + Contrast
axs[1, 0].imshow(only_contrast, cmap='gray')
axs[1, 0].set_title("Grayscale + Contrast", fontsize=30)
axs[1, 0].axis("off")

# Grayscale + Brightness + Contrast
axs[1, 1].imshow(bright_then_contrast, cmap='gray')
axs[1, 1].set_title("Grayscale + Brightness + Contrast",fontsize=30)
axs[1, 1].axis("off")

# Kosongkan slot terakhir
axs[1, 2].axis("off")

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

- plt.subplots(2, 3, figsize=(24, 12)) membuat sebuah grid subplots yang terdiri dari 2 baris dan 3 kolom, dengan ukuran gambar yang diperbesar (figsize=(24, 12)).
- fig adalah objek gambar utama, dan axs adalah array yang berisi axis untuk setiap subplot.
- Gambar asli (original_image) ditampilkan pada posisi baris pertama, kolom pertama.

- axs[0, 0].set_title() menambahkan judul "Gambar Asli" dengan ukuran font 30.
- axs[0, 0].axis("off") menonaktifkan sumbu agar gambar ditampilkan tanpa garis sumbu.
- Gambar grayscale (grayscale_image) ditampilkan pada posisi baris pertama, kolom kedua, dengan colormap gray.
- Gambar dengan peningkatan kecerahan (only_bright) ditampilkan pada posisi baris pertama, kolom ketiga.
- Gambar dengan peningkatan kontras (only_contrast) ditampilkan pada posisi baris kedua, kolom pertama.
- Gambar dengan peningkatan kecerahan dan kontras (bright_then_contrast) ditampilkan pada posisi baris kedua, kolom kedua.
- Menonaktifkan slot terakhir pada baris kedua, kolom ketiga.
- plt.tight_layout() memastikan bahwa subplots disusun dengan rapi tanpa tumpang tindih.
- plt.show() menampilkan gambar dan subplots yang telah disusun.



BAB IV

PENUTUP

Berdasarkan kegiatan praktikum pengolahan citra digital yang telah dilakukan, dapat disimpulkan bahwa teknik deteksi warna dan perbaikan kualitas gambar merupakan bagian penting dalam proses analisis dan manipulasi citra. Proses deteksi warna merah, hijau, dan biru dalam gambar digital dapat dilakukan dengan memisahkan masing-masing kanal warna dari format citra RGB. Dengan pendekatan ini, setiap warna utama dapat divisualisasikan dan dianalisis secara individual, yang sangat berguna dalam berbagai aplikasi seperti deteksi objek, segmentasi gambar, dan pelacakan warna. Selain itu, penggunaan metode thresholding atau ambang batas sangat membantu dalam mengelompokkan piksel berdasarkan intensitas warnanya. Dengan menentukan nilai ambang yang sesuai, citra dapat dibagi menjadi area yang merepresentasikan warna-warna tertentu, sehingga mempermudah proses identifikasi dan pengolahan bagian citra yang relevan.

Selanjutnya, permasalahan pencahayaan pada gambar, khususnya kondisi backlight, dapat diatasi dengan teknik penyesuaian brightness (kecerahan) dan contrast (kontras). Gambar yang memiliki pencahayaan dari belakang biasanya menyebabkan objek di bagian depan menjadi gelap atau tidak terlihat jelas. Dengan memanfaatkan fungsi pemrosesan gambar seperti peningkatan kecerahan dan kontras, kualitas visual gambar dapat ditingkatkan secara signifikan sehingga objek dalam gambar terlihat lebih terang dan tajam. Hal ini membuktikan bahwa manipulasi citra menggunakan pustaka seperti OpenCV dan Pillow mampu memberikan solusi praktis dalam meningkatkan kualitas gambar yang kurang optimal.

Secara keseluruhan, praktik ini memberikan pemahaman yang lebih dalam tentang bagaimana teknik dasar dalam pengolahan citra dapat diterapkan secara nyata untuk menyelesaikan permasalahan visual pada gambar. Mulai dari deteksi warna, pengelompokan warna berdasarkan nilai intensitas, hingga perbaikan pencahayaan citra, seluruh tahapan menunjukkan bahwa pemrosesan citra digital memainkan peran penting dalam berbagai bidang, termasuk pengenalan pola, sistem pengawasan visual, dokumentasi digital, dan masih banyak lagi. Dengan menguasai dasar-dasar ini, diharapkan peserta praktikum dapat mengembangkan kemampuan analisis dan pengolahan gambar secara lebih lanjut untuk keperluan akademik maupun profesional di masa mendatang.

DAFTAR PUSTAKA

Anggraeni, D. T. (2021). *Perbaikan citra dokumen hasil pindai menggunakan metode Simple, Adaptive-Gaussian, dan Otsu Binarization Thresholding*. Jurnal Expert: Jurnal Ilmiah Bidang Ilmu Eksakta, 11(2), 125–134. https://media.neliti.com/media/publications/391993-none-dbc2f496.pdf

(n.d.). *Penerapan pengolahan citra untuk perbaikan gambar 2 dimensi dengan menggunakan MATLAB*. Jurnal Teknik Informatika, Universitas Islam Lamongan. **https://jurnalteknik.unisla.ac.id/index.php/informatika/article/view/1063/654**

(n.d.). *BAB.pdf*. Repository Universitas Muhammadiyah Parepare. https://repository.umpar.ac.id/id/eprint/808/3/BAB.pdf