UTS

PENGOLAHAN CITRA

NAMA : Muh Bintang Putra Erfin

NIM : 202331125

KELAS : G

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : -

ASISTEN : 1. FAUZAN ARROYAN

2. ABDUR RASYID RIDHO

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Apa yang dimaksud dengan pengolahan citra digital dan mengapa penting untuk dipelajari?
2. Bagaimana cara kerja dan penerapan metode dasar seperti histogram, gambar biner, dan grayscale dalam pengolahan citra?
3. Bagaimana teknik threshold dan deteksi warna RGB digunakan dalam segmentasi dan analisis gambar?
4. Apa saja hasil yang diperoleh dari proses-proses tersebut dalam konteks praktikum?

1.2 Tujuan Masalah

1. Menjelaskan konsep dasar dan pentingnya pengolahan citra digital dalam dunia teknologi saat ini.
2. Mempelajari dan menerapkan berbagai teknik dasar pengolahan citra seperti histogram, grayscale, binary image, thresholding, dan pendeteksian warna RGB.
3. Menganalisis hasil dari setiap teknik pengolahan citra yang telah diaplikasikan pada gambar.
4. Memberikan pemahaman praktis mengenai bagaimana komputer memproses dan mengenali informasi visual dari gambar digital.

1.3 Manfaat Masalah

1. Menambah wawasan dan pemahaman mahasiswa mengenai pengolahan citra digital.
2. Membantu dalam pengembangan aplikasi yang melibatkan visualisasi dan analisis gambar.
3. Memberikan dasar pengetahuan yang berguna untuk penelitian lebih lanjut di bidang computer vision dan artificial intelligence.
4. Melatih keterampilan teknis dalam menggunakan bahasa pemrograman (Python) dan pustaka-pustaka seperti OpenCV dan matplotlib dalam pemrosesan gambar.

BAB II
LANDASAN TEORI

1. Pengolahan Citra Digital

Pengolahan citra digital adalah teknik manipulasi gambar digital yang bertujuan untuk meningkatkan kualitas citra, mengekstrak informasi, atau mempermudah analisis. Teknik ini sangat penting dalam berbagai bidang, seperti pertanian, medis, dan teknologi informasi. Misalnya, dalam penelitian yang dilakukan oleh Hidayah et al. (2025), pengolahan citra digunakan untuk mendeteksi kualitas daun sawi. Dengan menerapkan algoritma tertentu, sistem dapat secara otomatis menilai kondisi daun, yang membantu para petani dalam menentukan kualitas hasil panen mereka.

Contoh implementasi lain dari pengolahan citra digital adalah dalam sistem pengenalan wajah. Dalam aplikasi keamanan, citra wajah diambil dan diproses untuk membandingkan dengan database wajah yang sudah ada. Metode ini dapat meningkatkan akurasi dalam identifikasi dan memberikan solusi keamanan yang lebih baik.

2. Gambar Biner

Gambar biner adalah citra yang hanya terdiri dari dua nilai pixel, yaitu hitam dan putih. Proses ini biasanya dilakukan melalui teknik thresholding, di mana nilai intensitas pixel dibandingkan dengan ambang batas tertentu untuk menentukan apakah pixel tersebut akan menjadi hitam atau putih. Penggunaan gambar biner sangat efektif dalam deteksi objek, karena memudahkan identifikasi dan penghitungan objek dalam citra. Dalam penelitian oleh Adianto et al. (2024), penggunaan gambar biner membantu dalam memisahkan objek dari latar belakang yang kompleks.

Contoh implementasi gambar biner dapat ditemukan dalam aplikasi pemrograman untuk mendeteksi objek dalam citra. Dalam sistem pengawasan, gambar yang diambil dari kamera dapat diproses menjadi gambar biner untuk mendeteksi gerakan atau keberadaan objek di area tertentu. Dengan cara ini, sistem dapat memberi peringatan kepada pengguna jika ada aktivitas yang mencurigakan.

3. Thresholding

Thresholding adalah teknik yang digunakan untuk mengonversi citra grayscale menjadi gambar biner dengan menentukan ambang batas tertentu. Teknik ini sangat penting dalam memisahkan objek dari latar belakang, sehingga memudahkan analisis lebih lanjut. Dalam penelitian Hidayah et al. (2025), thresholding diterapkan sebagai langkah awal dalam preprocessing citra untuk meningkatkan akurasi deteksi objek. Dengan menetapkan ambang yang tepat, sistem dapat lebih akurat dalam mengenali objek yang diinginkan.

Contoh implementasi thresholding dapat dilihat dalam aplikasi pengolahan citra untuk mengidentifikasi bagian tertentu dari gambar, seperti dalam pengolahan citra medis. Dalam analisis citra medis, thresholding digunakan untuk mendeteksi tumor atau area yang mencurigakan dalam gambar MRI atau CT scan, sehingga dokter dapat lebih mudah melakukan diagnosis.

4. Grayscale

Grayscale adalah representasi citra yang hanya menggunakan satu channel warna, yaitu intensitas cahaya. Ini mengubah citra berwarna menjadi hitam-putih, yang menyederhanakan analisis dan memungkinkan fokus pada kecerahan dan kontras. Dalam berbagai metode preprocessing, grayscale sering digunakan untuk meningkatkan kualitas citra sebelum analisis lebih lanjut dilakukan. Penelitian oleh Adianto et al. menunjukkan bahwa konversi ke grayscale dapat meningkatkan akurasi deteksi objek.

Contoh implementasi grayscale dapat ditemukan dalam aplikasi pengenalan karakter optik (OCR). Dalam sistem OCR, citra teks yang diambil dari dokumen sering kali dikonversi menjadi grayscale untuk memudahkan pengenalan karakter. Dengan cara ini, sistem dapat lebih efektif dalam membaca teks, terlepas dari variasi warna dan latar belakang.

5. Histogram

Histogram adalah representasi grafis dari distribusi intensitas pixel dalam gambar. Alat ini memungkinkan analisis kualitas citra, serta membantu dalam menentukan threshold untuk segmentasi. Hidayah et al. (2025) menjelaskan bahwa histogram equalization dapat digunakan untuk meningkatkan kontras gambar, sehingga objek yang diinginkan lebih terlihat jelas. Dengan membagi intensitas pixel ke dalam interval tertentu, pengguna dapat dengan mudah mengidentifikasi area dengan kontras rendah.

Contoh implementasi histogram dapat ditemukan dalam aplikasi pengolahan gambar, seperti software editing foto. Dalam aplikasi ini, pengguna dapat melihat histogram gambar untuk menyesuaikan pencahayaan dan kontras gambar, sehingga meningkatkan kualitas visual sebelum mencetak atau membagikannya.

6. Model Warna HSV (Hue, Saturation, Value)

Model warna HSV memisahkan warna menjadi tiga komponen: Hue (warna), Saturation (kecerahan warna), dan Value (intensitas). Model ini lebih efektif dalam membedakan objek berdasarkan warna dibandingkan model RGB. Dalam penelitian Hidayah et al. (2025), HSV digunakan untuk mendeteksi kualitas daun sawi dengan memisahkan objek berdasarkan nilai Hue tertentu. Teknik ini sangat berguna dalam aplikasi yang memerlukan diferensiasi warna yang halus.

Contoh implementasi model warna HSV dapat ditemukan dalam aplikasi pengenalan warna, seperti dalam robotika. Robot dapat menggunakan sensor untuk mendeteksi warna objek di sekitarnya berdasarkan model HSV, memungkinkan robot untuk mengenali dan berinteraksi dengan objek berdasarkan warna mereka.

7. Color Blob Detection

Color Blob Detection adalah teknik yang digunakan untuk mendeteksi area (blob) dalam citra yang memiliki karakteristik warna serupa. Teknik ini memungkinkan deteksi objek meskipun terdapat kesamaan warna dengan latar belakang. Dalam penelitian Adianto et al. (2024), metode ini diterapkan untuk mendeteksi objek seperti daun sawi dan menunjukkan bagaimana teknik ini dapat menghitung luas dan jumlah objek yang terdeteksi.

Contoh implementasi Color Blob Detection dapat ditemukan dalam aplikasi pengawasan video. Dalam sistem ini, algoritma dapat digunakan untuk mendeteksi dan mengidentifikasi objek tertentu dalam video secara real-time, seperti dalam aplikasi keamanan yang memantau area publik.

8. Preprocessing

Preprocessing adalah serangkaian teknik yang diterapkan sebelum analisis citra untuk meningkatkan kualitas citra. Langkah-langkah dalam preprocessing termasuk pengurangan noise, peningkatan kontras, dan thresholding adaptif. Pengurangan noise bertujuan untuk menghilangkan gangguan visual yang dapat mempengaruhi hasil analisis, sedangkan peningkatan kontras membantu memperjelas perbedaan antara objek dan latar belakang. Hidayah et al. (2025) menekankan bahwa preprocessing sangat penting untuk memastikan citra berkualitas baik, sehingga meningkatkan akurasi deteksi.

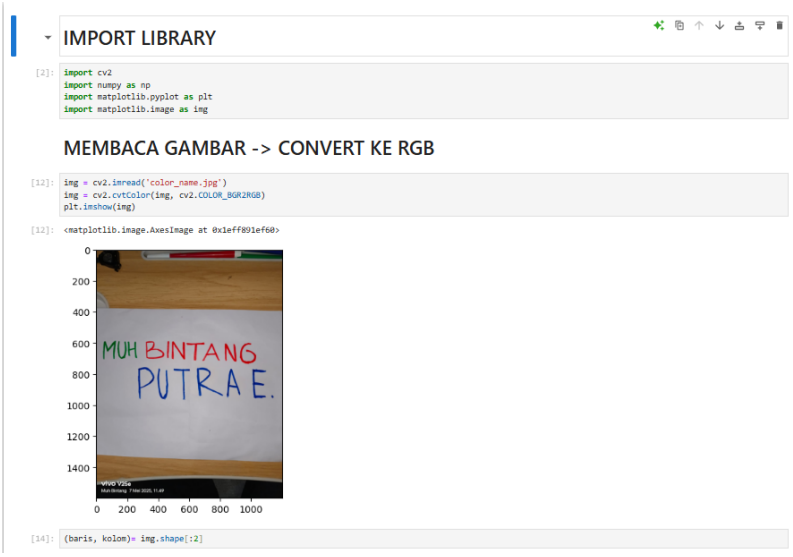
Contoh implementasi preprocessing dapat ditemukan dalam aplikasi pengolahan citra medis. Dalam analisis citra MRI, preprocessing digunakan untuk meningkatkan kualitas gambar sebelum dilakukan analisis untuk mendeteksi kelainan. Dengan menggunakan teknik seperti Gaussian Blur dan histogram equalization, dokter dapat mendapatkan citra yang lebih jelas dan informatif.

BAB III

HASIL

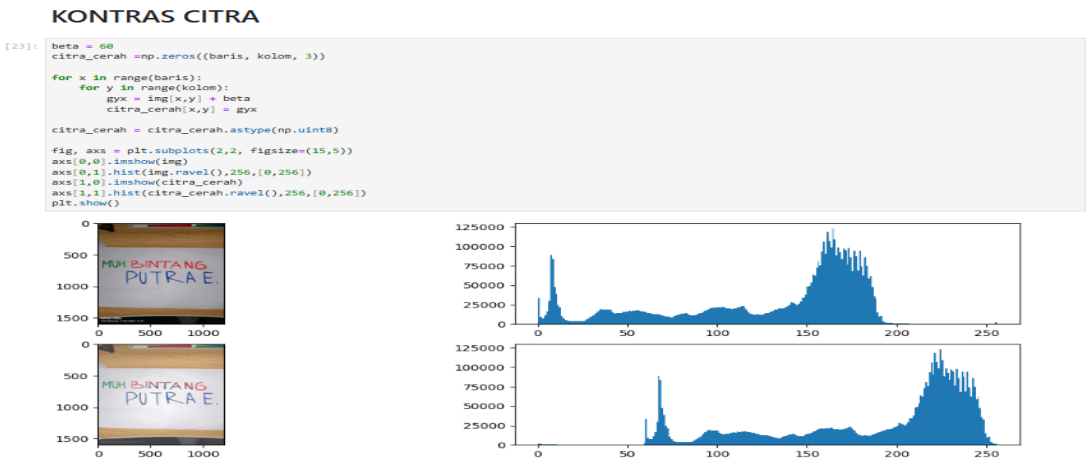
1.Deteksi Warna Pada Citra

Seperti yang sudah ditentukan, kita akan mendeteksi berbagai warna RGB (Red, Green, Blue) , lalu menganalisa satu persatu gambar yang dihasilkan dan juga melihat histogram dari gambar tersebut



Tentunya hal pertama yang akan kita dilakukan dengan cara meng import library-library yang dibutuhkan seperti cv2 untuk membaca dan menampilkan gambar, numpy untuk membaca piksel gambar secara matematis, dan matplotlib untuk mengeksekusi gambar, dalam hal ini berarti mengedit gambar sesuai kebutuhan dari soal itu sendiri

Lalu langsung saja kita gunakan cv2 untuk membaca gambar yang sudah kita sediakan yang warna awalnya berformat BGR langsung kita convert ke RGB, setelahnya kita inisialisasikan baris serta kolom dari gambar tersebut.



Nah setelah itu, kita akan coba untuk meningkatkan kontras dari citra gambar nya, dengan memakai looping for, yang dimana kita inisialisakian x nya dengan parameter baris, dan y dengan paramaeter kolom nya, untuk tambahan kontras nya sendiri kita taruh di angka 60.

Lalu untuk menampilkan histogram nya, kita memakai fig dan axs, sebagai baris dan kolom, ibarat seperti sebuah matriks / array untuk penyusunan tata letak dari gambar sebelum dan sesudah nya beserta histogram nya. Bisa kita lihat juga setelah penambahan dari kontrasnya terlihat jelas pada gambar histogram di samping nya, yang dimana cukup menggeser ke dari kiri ke kanan dengan baik.

DETEKSI WARNA CITRA

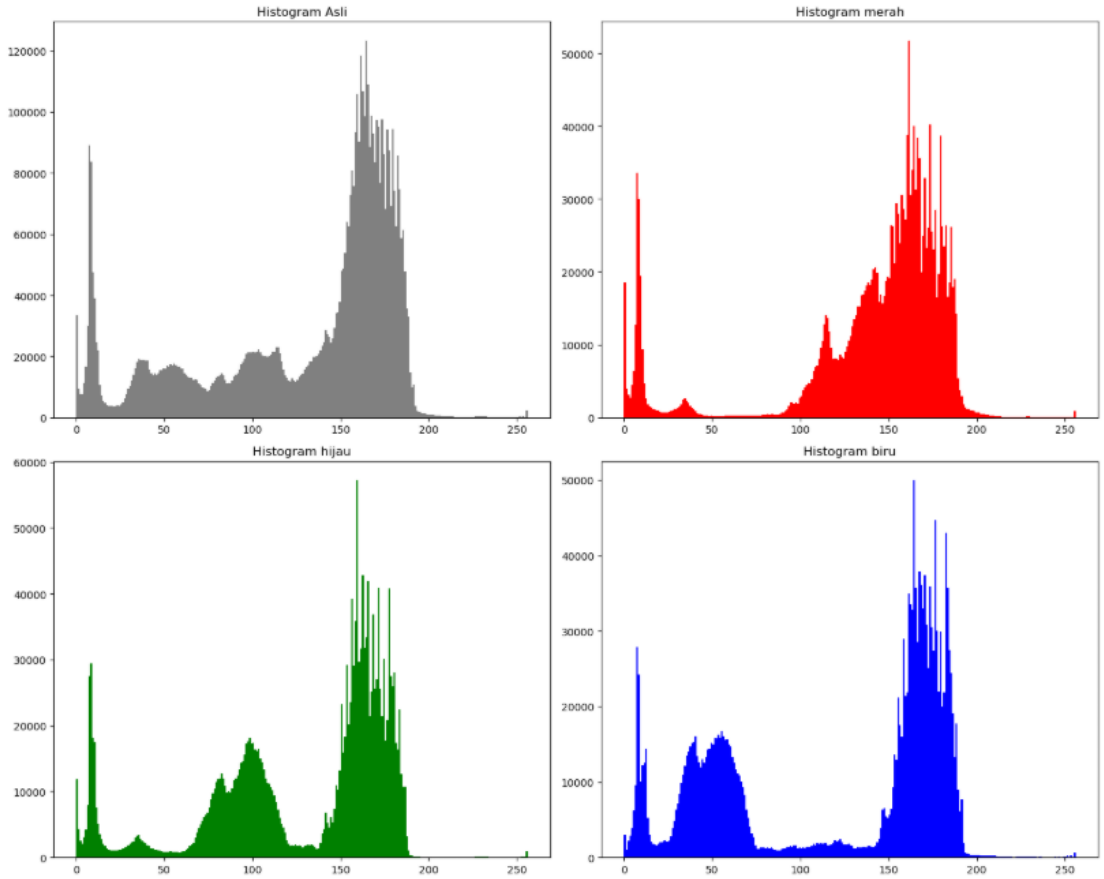
```
[37]: r = img[:, :, 0]
      g = img[:, :, 1]
      b = img[:, :, 2]

      f, axes = plt.subplots(2, 2, figsize=(15, 12))

      # Citra Kontras
      axes[0, 0].set_title('CITRA KONTRAS')
      axes[0, 0].imshow(img)
      axes[0, 0].axis('off')

      # Warna Merah
      axes[0, 1].set_title('Merah')
      axes[0, 1].imshow(r, cmap='gray')
      axes[0, 1].axis('on')

      # Warna Hijau
      axes[1, 0].set_title('Hijau')
      axes[1, 0].imshow(g, cmap='gray')
      axes[1, 0].axis('on')

      # Warna Biru
      axes[1, 1].set_title('Biru')
      axes[1, 1].imshow(b, cmap='gray')
      axes[1, 1].axis('on')

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

Selanjutnya kita akan coba untuk mendeteksi dari ketiga warna nya, dan menampilkan nya secara bersamaan dan berurutan, yang dimana cukup simpel kita hanya perlu menginisialisasikan nilai dari rgb itu sendiri dengan value nya gambar kita ubah, untuk re sendiri 0, untuk hijau 1, dan untuk biru 2.

Setelahnya kita hanya perlu menampilkan nya dengan cara yang sama seperti tadi, cuman disini figsize nya kita kecilkan, karna kita akan memuat 4 gambar (2 baris 2 kolom), maka dengan axes saja kita panggil, dan tampilan gambar nya pun seperti diatas. Bisa dilihat untuk masing-masing warna nya sendiri terdeteksi dengan warna abu yang lebih muda.

HISTOGRAM GAMBAR

```
[51]: f, axes = plt.subplots(2, 2, figsize=(15, 12))

axes[0, 0].set_title('Histogram Asli')
axes[0, 0].hist(img.flatten(), bins=256, range=[0,256], color='gray')

axes[0, 1].set_title('Histogram merah')
axes[0, 1].hist(r.flatten(), bins=256, range=[0,256], color='r')

axes[1, 0].set_title('Histogram hijau')
axes[1, 0].hist(g.flatten(), bins=256, range=[0,256], color='g')

axes[1, 1].set_title('Histogram biru')
axes[1, 1].hist(b.flatten(), bins=256, range=[0,256], color='b')

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



Selanjutnya kita akan menganalasiskan nya satu per satu. Bisa kita lihat pada histogram asli nya puncak besar nya dikisaran 160-180, yang dimana menunjukan pensebaran pikselnya terang atau mendekati warna putih, cukup kompleks karna gambar juga memiliki kontras yang cukup tinggi dari gelap ke terang. Adapun untuk Histogram merah, distribusinya juga cukup tinggi pada rentang 130-190, yang artinya bagian terang merah ini cukup dominan.

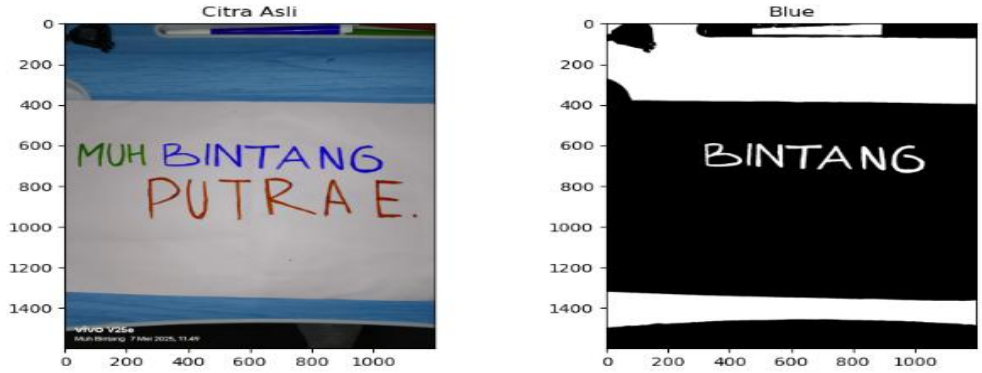
Untuk histogram hijau, kurang lebih mirip dengan histogram merah hanya saja ada puncak tambahan sekitar 50-100 yang cukup jelas, meunjukan distribusi warna hijau lebih banyak dari merah. Terakhir untuk histogram biru ini cukup menarik, karna memiliki 2 puncak dominan, yang pertama di kisaran 20-70 yang menunjukan bagian gelap dari biru itu, dan yang kedua di 150-200 yang menunjukan bagian terang dari biru tersebut, maka histogram biru kah yang paling seimbang antara warna gelap maupun terang nya.

2.Mengurutkan Ambang Batas

Sesuai dengan ketentuan soal yang sudah dibuat, kita akan mencari lalu mengurutkan amabang batas / threshold pada citra gambar untuk menampilkan sesuai katergorinya

AMBANG BATAS BIRU

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

lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

mask = cv2.inRange(hsv, lower_blue, upper_blue)

hasil = cv2.bitwise_and(img, img, mask=mask)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Citra Asli')

axs[1].imshow(mask, cmap='gray')
axs[1].set_title('Blue')

plt.show()
```



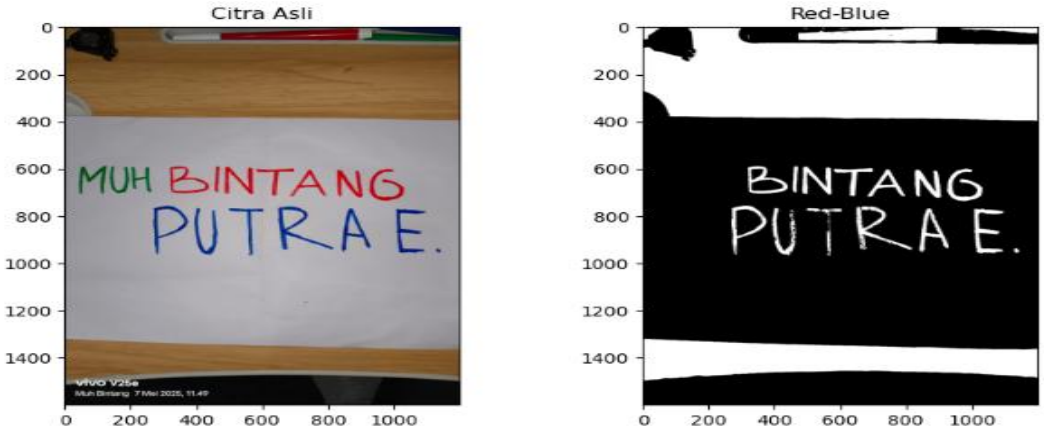
Yang pertama kita akan mencari threshold biru nya, dengan menggunakan hsv, yakni Teknik memisahkan warna menjadi 3 komponen, yang diamna Hue (warna), Saturation (kecerahan warna), dan Value (intensitas Warna).

Terlihat jelas, tulisan bintang memiliki warna biru yang sangat jelas, dengan kita menginisialisasikan lower dan uppernya dengan rentang 50-130 dengan saturasi dan value nya minimal 50.

Adapun cara kerja nya sendiri cukup simple, bisa kita lihat pada mask, terdapat inRange yang dimana piksel dalam rentang biru diset ke warna putih(255), lalu untuk mengisolasikan bagian biru dari gambar aslinya kita hanya perlu menggunakan bitwise_and()

AMBANG BATAS MERAH-BIRU

```
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

lower_red1 = np.array([0, 50, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
upper_red2 = np.array([180, 255, 255])

mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)

mask_combined1 = cv2.bitwise_or(mask_blue, mask_red1)
mask_combined1 = cv2.bitwise_or(mask_combined1, mask_red2)

hasil = cv2.bitwise_and(img, img, mask=mask_combined1)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(img)
axs[0].set_title('Citra Asli')

axs[1].imshow(mask_combined1, cmap='gray')
axs[1].set_title('Red-Blue')

plt.show()
```

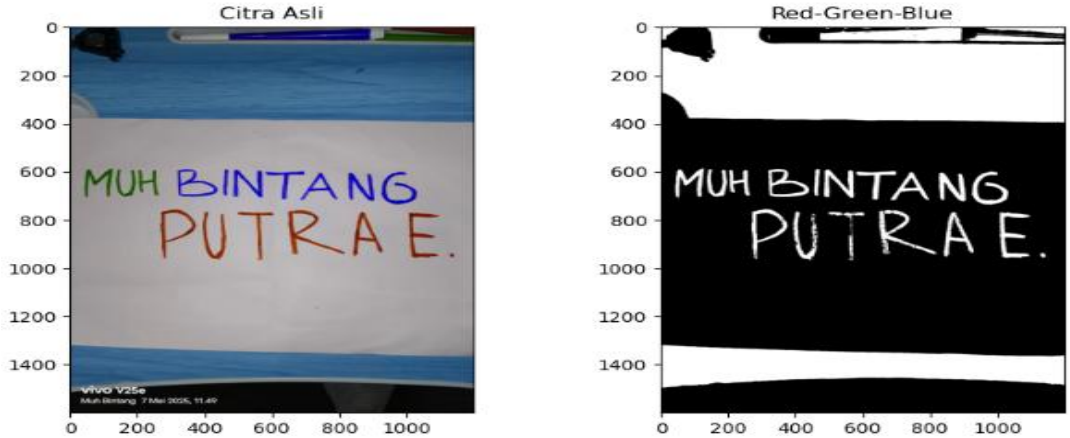


Selanjutnya kita akan membuat ambang batas nya dengan 2 warna, yakni merah dan biru, sebernya untuk codingan nya sendiri sama, hanya saja kita menambahkan warna merah pada notebook nya, gunanya untuk men setel kembali agar citra sebelumnya balik ke warna asalnya.

Nilai amabang pada warna merahnya sendiri cukup kontras ya, karna bisa dilihta pada gambar juga warna merah cukup memiliki porsi yang bagus

AMBANG BATAS RGB

```
86]: hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)


lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

lower_red1 = np.array([0, 50, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
upper_red2 = np.array([180, 255, 255])

lower_green = np.array([35, 50, 50])
upper_green = np.array([85, 255, 255])

mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask_green = cv2.inRange(hsv, lower_green, upper_green)

mask_combined2 = cv2.bitwise_or(mask_blue, mask_red1)
mask_combined2 = cv2.bitwise_or(mask_combined2, mask_red2)
mask_combined2 = cv2.bitwise_or(mask_combined2, mask_green)

hasil = cv2.bitwise_and(img, img, mask=mask_combined2)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Citra Asli')

axs[1].imshow(mask_combined2, cmap='gray')
axs[1].set_title('Red-Green-Blue')

plt.show()
```

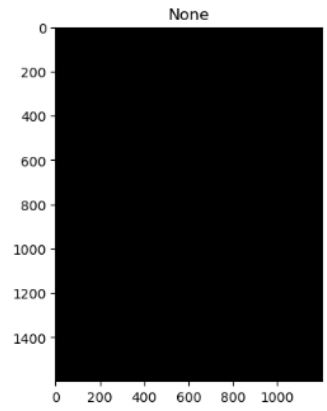


Selanjutnya kita tambahkan lagi warna hijau nya, sama seperti mencari ambang biru dan merah, kita inisialisasikan lagi hsv nya dan juga upper dan lower nya masing-masing. Dimana untuk warna biru sendiri masih di rentang 100-130, untuk warna merah rentang 0-10 untuk gelap nya, 170-180 untuk terang nya, dan untuk warna hijau sendiri rentang 50-80, setelahnya kita kasih mask, dan bitwisenya

Latar belakang dan sebagian area luar tidak terdeteksi (hitam), menunjukkan keberhasilan dalam isolasi objek teks berwarna dari latar belakang putih-kebiruan.

AMBANG BATAS NONE

```
[80]: def display_image(image, title="Image"):
          plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
          plt.title(title)
          plt.axis('on')
          plt.show()

      hsv_image = cv2.imread('color_name.jpg', cv2.IMREAD_COLOR)
      hsv_image = cv2.cvtColor(hsv_image, cv2.COLOR_BGR2HSV)

      lower_blue = np.array([100, 50, 50])
      upper_blue = np.array([130, 255, 255])

      lower_red1 = np.array([0, 50, 50])
      upper_red1 = np.array([10, 255, 255])

      lower_red2 = np.array([170, 50, 50])
      upper_red2 = np.array([180, 255, 255])

      lower_green = np.array([50, 50, 50])
      upper_green = np.array([80, 255, 255])

      blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)
      red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
      red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
      green_mask = cv2.inRange(hsv_image, lower_green, upper_green)

      red_blue_mask = cv2.bitwise_or(red_mask1, red_mask2)
      combined_mask = cv2.bitwise_or(blue_mask, cv2.bitwise_or(red_blue_mask, green_mask))

      black_image = np.zeros_like(hsv_image, dtype=np.uint8)

      black_detected_image = cv2.bitwise_and(black_image, black_image, mask=combined_mask)

      display_image(black_detected_image, "None")
```

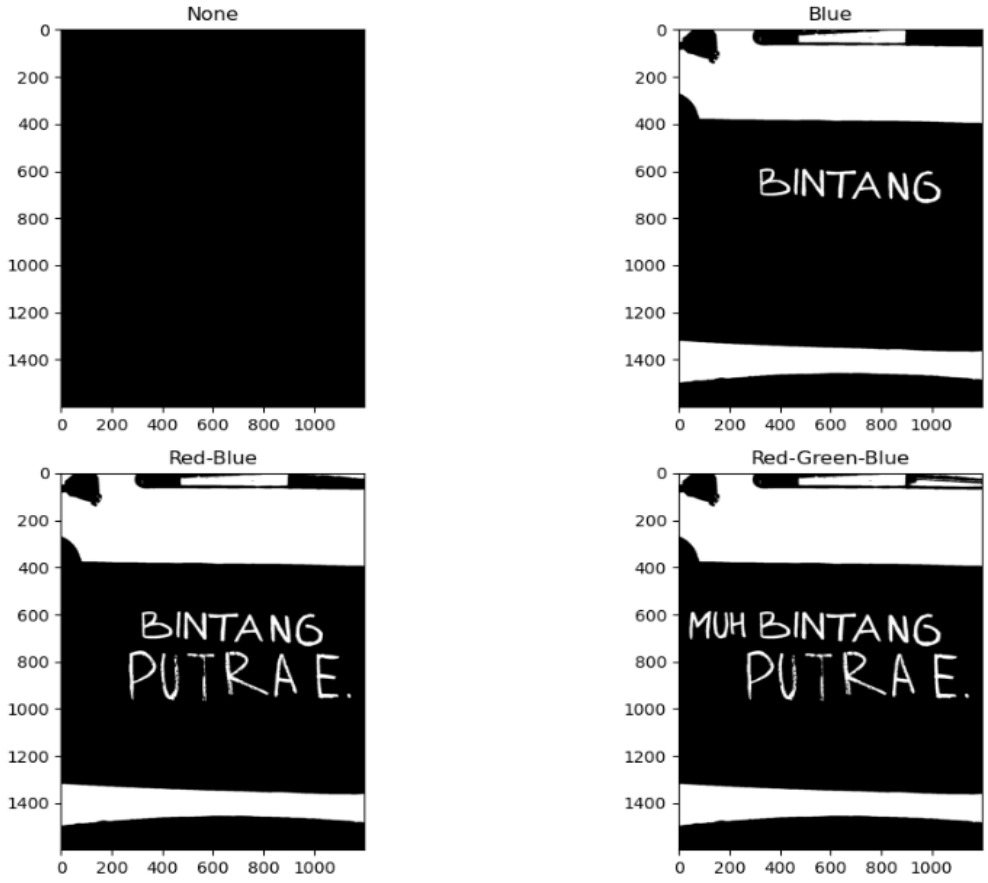


Pada batas ambang yang terakhir ini, menggabungkan ketiga warna tadi secara bersamaan, namun pada hasil akhirnya menunjukan tidak ada gambar yang terdeteksi.

Adapun ketiga warna ditentukan masing-masing:

- Biru : dari 100 ke 130
- Merah : ada 2 rentang, karena merah melintasi batas 0 dalam HSV itu sendiri
- Hijau : dari 50 ke 80

```
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

axs[0, 0].imshow(cv2.bitwise_and(black_image, black_image, mask=combined_mask))
axs[0, 0].set_title('None')

axs[0, 1].imshow(mask, cmap='gray')
axs[0, 1].set_title('Blue')

axs[1, 0].imshow(mask_combined1, cmap='gray')
axs[1, 0].set_title('Red-Blue')

axs[1, 1].imshow(mask_combined2, cmap='gray')
axs[1, 1].set_title('Red-Green-Blue')

for ax in axs.flat:
    ax.axis('on')

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



Dan ini adalah hasil pengurutan ambang batas dari terkecil hingga terbesar (kontras sesuai warnanya), dengan menggunakan cara yang sama dengan nomor 1, kita membuatnya dengan figsize 4 gambar (2 kolom 2 baris), lalu menampilkan nya dengan axes.

3.Memperbaiki gambar backlight

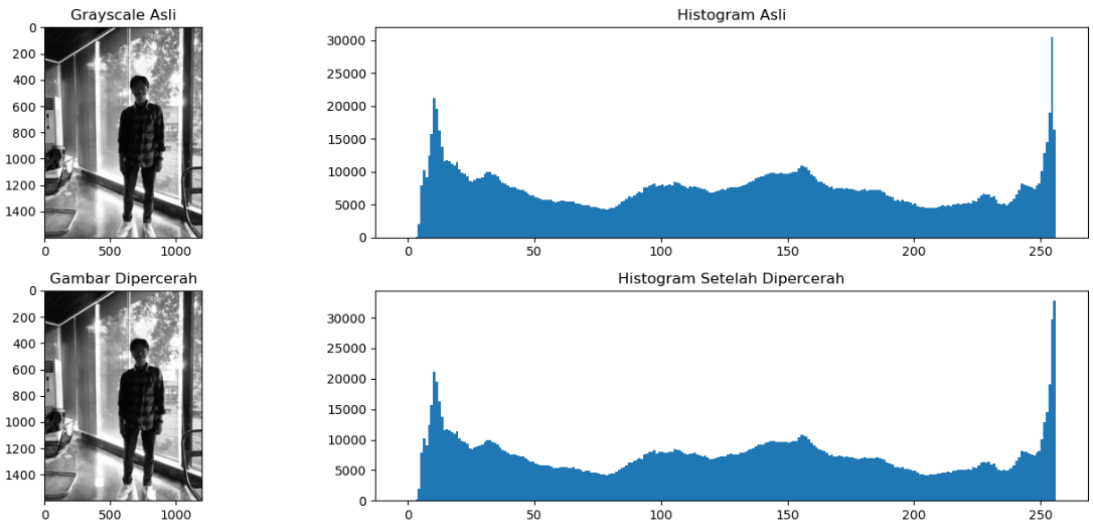
Sesuai dengan ketentuan soal yang sudah dibuat, kita akan memperbaiki gambar backlight dengan satu foto, yanga dimana kita akan mengubahnya terlebih dahulu ke grayscale

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

[2]: img = cv2.imread('backlight.jpg')
     rgb = cv2.cvtColor (img, cv2.COLOR_BGR2RGB)

[3]: baris, kolom = img.shape[:2]

[4]: gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

[5]: face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
     faces = face_cascade.detectMultiScale(gray, scaleFactor=1.09, minNeighbors=3)
```

Seperti biasa kita import kan library nya dulu, baru inisialisasi variabel image nya dengan gambar yang sudah ada. Disini kita coba conver format gambar nya dalam format rgb , setelah itu kita inisialisasikan juga kolom dan baris nya, lalu kita convert lagi ke bentuk grayscale.

Nah disini untuk menangkap objek muka, kita menggunakan yang Namanya face_cascade, gunanya untuk mendeteksi objek muka yang tertera pada gambar yang nantinya akan kita naikan kecerahan serta kontras nya.

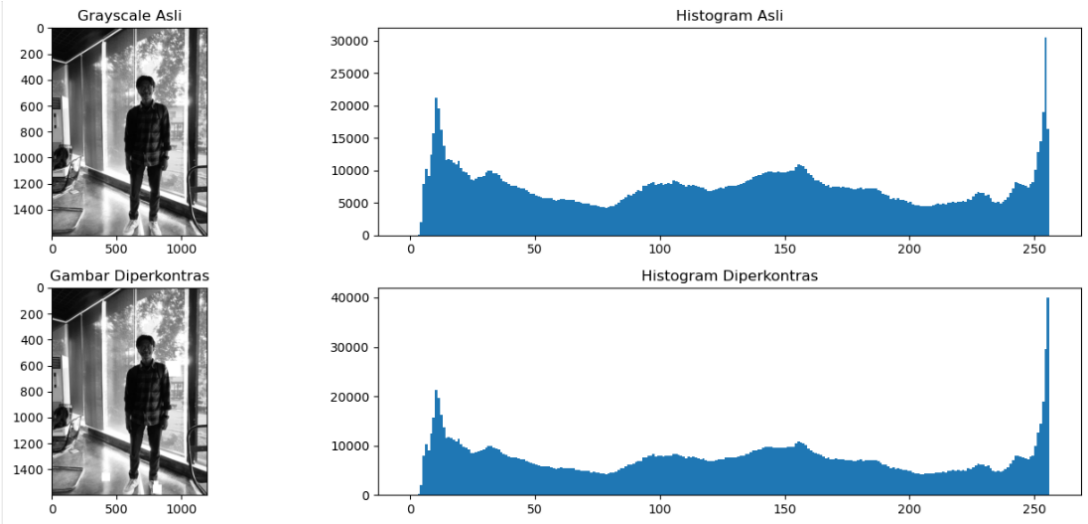
```
citra_cerah = gray.copy()
beta = 70

# Loop hanya pada area wajah
for (x, y, w, h) in faces:
    for i in range(y, y+h):
        for j in range(x, x+w):
            gyx = gray[i, j] + beta
            citra_cerah[i, j] = min(255, gyx)

citra_cerah = citra_cerah.astype(np.uint8)

# Visualisasi hasil
fig, axs = plt.subplots(2, 2, figsize=(15, 6))

axs[0, 0].imshow(gray, cmap='gray')
axs[0, 0].set_title("Grayscale Asli")
axs[0, 1].hist(gray.ravel(), 256, [0, 256])
axs[0, 1].set_title("Histogram Asli")

axs[1, 0].imshow(citra_cerah, cmap='gray')
axs[1, 0].set_title("Gambar Dipercerah")
axs[1, 1].hist(citra_cerah.ravel(), 256, [0, 256])
axs[1, 1].set_title("Histogram Setelah Dipercerah")

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



Pertama, kita akan coba untuk meningkatkan kecerahan dari gambar yang sudah kita ubah dalam bentuk grayscale, yang dimana tingkat kecerahan yang kita tambahkan sebesar 70 piksel, lalu untuk mendeteksi objek wajah nya, maka kita gunakan looping seperti biasa tapi dengan iterasi 3 kali, agar dapat mendeteksi wajah objek pada gambar.

Bisa dilihat pada histogram setelah dicerahkan, histogram nya bergeser sedikit ke kanan, menandakan peningkatan intensitas, dan distribusi lebih merata ke arah nilai terang, namun puncak pada tinggi 255 menjadi lebih jelas.

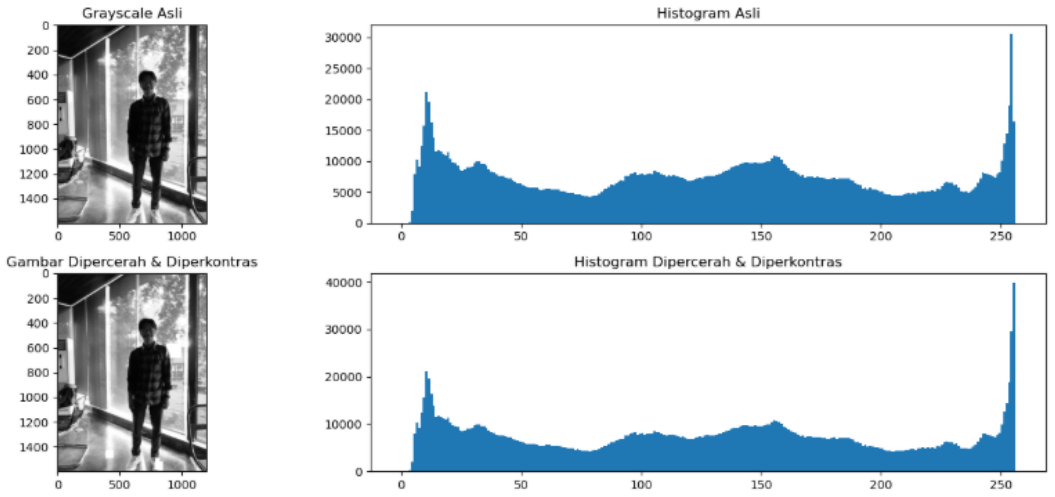
```
[7]: citra_kontras = gray.copy()
alpha = 2.5

for (x, y, w, h) in faces:
    for i in range(y, y+h):
        for j in range(x, x+w):
            gxy = gray[i, j] * alpha
            citra_kontras[i, j] = min(255, int(gxy))

citra_kontras = citra_kontras.astype(np.uint8)

# Tampilkan hasil
fig, axs = plt.subplots(2, 2, figsize=(15, 6))

axs[0, 0].imshow(gray, cmap='gray')
axs[0, 0].set_title("Grayscale Asli")
axs[0, 1].hist(gray.ravel(), 256, [0, 256])
axs[0, 1].set_title("Histogram Asli")

axs[1, 0].imshow(citra_kontras, cmap='gray')
axs[1, 0].set_title("Gambar Diperkontras")
axs[1, 1].hist(citra_kontras.ravel(), 256, [0, 256])
axs[1, 1].set_title("Histogram Diperkontras")

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



Selanjutanya kita akan coba memperkontras kan gambar nya, dengan codingan yang sama , tetapi disini untuk operasi aritmatika nya kita ganti jadi perkalian,dengan setiap pixel nya kita kalikan sebesar 2.5 piksel, maka gambar nya akan kontras.

Bisa dilihat pada histogram gambar setelah diperkontras, kontrasnya makin menajam, yang area gelap nya pun makin gelap, dan yag terang makin jadi terang, Ada puncak tajam di kedua sisi, khas dari proses kontras linier agresif. Subjek (orang) menjadi lebih menonjol terhadap latar belakang.

[8]:
```
citra_hasil = gray.copy().astype(np.float32)

alpha = 2.0
beta = 30

# Terapkan hanya pada wajah
for (x, y, w, h) in faces:
    for i in range(y, y+h):
        for j in range(x, x+w):
            gxy = alpha * gray[i, j] + beta
            citra_hasil[i, j] = min(255, gxy)

citra_hasil = citra_hasil.astype(np.uint8)

fig, axs = plt.subplots(2, 2, figsize=(15, 6))
axs[0, 0].imshow(gray, cmap='gray')
axs[0, 0].set_title("Grayscale Asli")
axs[0, 1].hist(gray.ravel(), 256, [0, 256])
axs[0, 1].set_title("Histogram Asli")

axs[1, 0].imshow(citra_hasil, cmap='gray')
axs[1, 0].set_title("Gambar Dipercerah & Diperkontras")
axs[1, 1].hist(citra_hasil.ravel(), 256, [0, 256])
axs[1, 1].set_title("Histogram Dipercerah & Diperkontras")

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

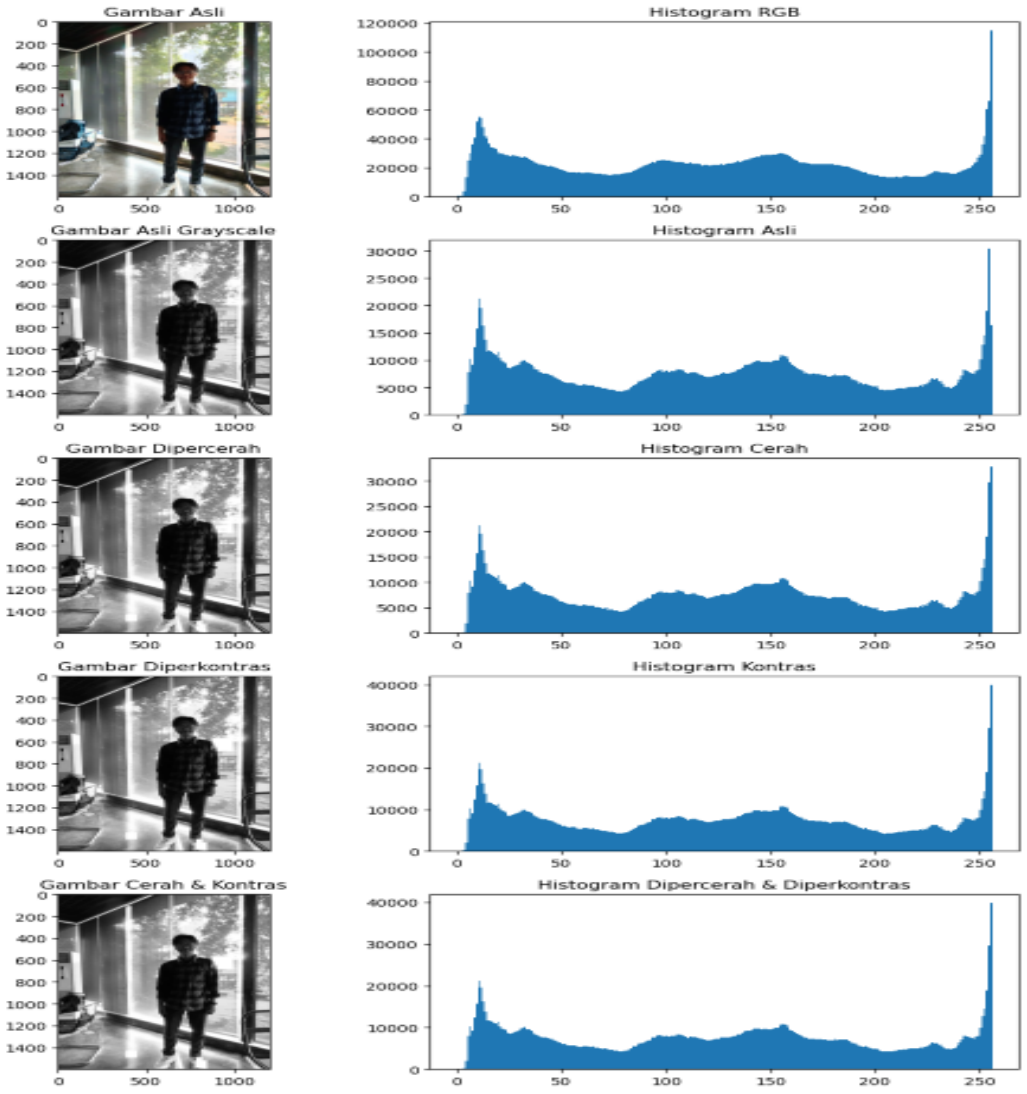


Berikutnya, kita akan coba mengagabungkan kedua percobaan diatas (mempercerah, dan memperkontraskan gambar) ke dalam satu gamabr dan coidnga, yang dimana kita tetap memakai gambar yang sama seperti sebelumnya. Lalu untuk nilai alpa nya sebesar 2.0 dan nilia beta nya sebesar 30, yang berarti gambar akan ditingkatkan kontrasnya setiap 2.0 kali, dan masing masing pixel nya akan kita tambahkan kecerahan nya sebesar 30 piksel.

Setalah di run, maka bisa dilihat pada histogram nya perubahan hanya terlihat pada area wajah, dan wajahnya pun tampak lebih jelas dan terang , bisa kita perhatikan juga area lain tidak mengalami perubahan yang cukup siginifikan.

Pada histogramnya juga, bisa diperhatikan tidakberubah secara drastic dibandingkan histogram aslinya, Ini menunjukkan **bahwa** perubahan hanya terjadi pada sebagian kecil area (wajah), sehingga tidak menggeser distribusi keseluruhan secara signifikan. Namun, ada sedikit peningkatan nilai intensitas di sisi kanan (area terang), menandakan bagian wajah menjadi lebih cerah.

```
# Visualisasi seluruh hasil
fig, axs = plt.subplots(5, 2, figsize=(10, 15))

axs[0, 0].imshow(rgb)
axs[0, 0].set_title("Gambar Asli")
axs[0, 1].hist(rgb.ravel(), 256, [0, 256])
axs[0, 1].set_title("Histogram RGB")

axs[1, 0].imshow(gray, cmap='gray')
axs[1, 0].set_title("Gambar Asli Grayscale")
axs[1, 1].hist(gray.ravel(), 256, [0, 256])
axs[1, 1].set_title("Histogram Asli")

axs[2, 0].imshow(citra_cerah, cmap='gray')
axs[2, 0].set_title("Gambar Dipercerah")
axs[2, 1].hist(citra_cerah.ravel(), 256, [0, 256])
axs[2, 1].set_title("Histogram Cerah")

axs[3, 0].imshow(citra_kontras, cmap='gray')
axs[3, 0].set_title("Gambar Diperkontras")
axs[3, 1].hist(citra_kontras.ravel(), 256, [0, 256])
axs[3, 1].set_title("Histogram Kontras")

axs[4, 0].imshow(citra_hasil, cmap='gray')
axs[4, 0].set_title("Gambar Cerah & Kontras")
axs[4, 1].hist(citra_hasil.ravel(), 256, [0, 256])
axs[4, 1].set_title("Histogram Dipercerah & Diperkontras")

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



Terakhir, kita akan menampilkan hasil dari semua gambar yang sudah kita analisis, mulai dari conver ke grayscale, lalu mempercerah, memperkontras, dan menggabungkan kedua nya., maka hasilnya bisa dilihat pada gambar diatas.

BAB IV

PENUTUP

Kesimpulan Hasil Teori

Pengolahan citra digital adalah disiplin ilmu yang luas, mencakup beragam teknik dasar yang bertujuan untuk memanipulasi dan menganalisis gambar. Teknik-teknik seperti konversi ke grayscale, thresholding, penggunaan histogram, model warna HSV, dan Color Blob Detection, serta preprocessing, memiliki peran krusial dalam meningkatkan kualitas citra dan mempermudah proses ekstraksi informasi yang relevan. Pemahaman mendalam terhadap konsep-konsep ini memungkinkan praktisi untuk mengoptimalkan proses pengolahan citra, menghasilkan analisis yang lebih akurat dan efisien.

Setiap teknik memiliki keunggulan dan aplikasi spesifik. Misalnya, konversi ke grayscale menyederhanakan analisis dengan mengurangi kompleksitas warna, sementara thresholding membantu dalam memisahkan objek dari latar belakang. Histogram memberikan wawasan tentang distribusi intensitas pixel, dan model warna HSV memungkinkan segmentasi warna yang lebih efektif. Dengan menguasai landasan teori ini, mahasiswa dan peneliti dapat mengembangkan solusi inovatif dalam berbagai bidang, mulai dari pengolahan citra medis hingga aplikasi pengawasan video.

Kesimpulan Hasil Praktikum

Hasil praktikum menunjukkan bahwa manipulasi citra melalui peningkatan kontras dan deteksi warna RGB menghasilkan perubahan signifikan pada distribusi pixel, yang tercermin dalam histogram. Peningkatan kontras menggeser histogram dan memperjelas perbedaan antara area terang dan gelap, sementara deteksi warna RGB mengungkapkan distribusi warna yang berbeda dalam citra. Penggunaan HSV efektif dalam mengisolasi warna tertentu, meskipun penggabungan ambang batas untuk beberapa warna sekaligus dapat menjadi kompleks.

Selain itu, praktikum ini juga menunjukkan bahwa perbaikan gambar backlight dapat dicapai dengan menggabungkan peningkatan kecerahan dan kontras. Peningkatan kecerahan menggeser histogram ke kanan, sementara peningkatan kontras memperjelas perbedaan antara area terang dan gelap. Kombinasi keduanya memberikan hasil yang optimal dalam memperbaiki gambar backlight, terutama pada area wajah. Secara keseluruhan, praktikum ini memberikan pengalaman praktis yang berharga dalam menerapkan teknik pengolahan citra digital untuk memanipulasi dan menganalisis gambar.

DAFTAR PUSTAKA

Sari, D. &. (2022). Deteksi dan perhitungan objek berdasarkan warna dengan HSV dan Color Blob Detection. *Jurnal JATI*, 23-34.

Sari, I. P. (2023). IImplementasi pengolahan citra digital dalam pengenalan wajah menggunakan deep learning YOLOv5. *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, 99-103.

Wicida. (2021). Implementasi color detection menggunakan algoritma Midpoint. *Jurnal* , 15-27.