UTS

PENGOLAHAN CITRA

NAMA : SULTAN MALIK MUZZAKKI

NIM : 202331020

KELAS : D

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

NO.PC :

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

Isi

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

- Bagaimana cara mendeteksi warna primer (merah, hijau, biru) dari citra hasil pemotretan pribadi menggunakan metode pemrosesan citra digital?
- Bagaimana menentukan nilai ambang batas (threshold) dari suatu citra untuk mengisolasi warna secara optimal?
- Bagaimana teknik memperbaiki citra backlight untuk menonjolkan objek (manusia) yang tertutup bayangan akibat pencahayaan latar?
- Bagaimana menampilkan hasil tersebut dalam bentuk visualisasi histogram dan menganalisisnya secara objektif?

1.2 Tujuan Masalah

- Mengimplementasikan metode deteksi warna RGB menggunakan bahasa Python.
- Menentukan dan mengurutkan ambang batas terkecil hingga terbesar dari citra yang dianalisis.
- Mengoptimalkan kualitas citra backlight melalui peningkatan brightness dan kontras.
- Menganalisis hasil olahan dalam bentuk gambar dan histogram serta menyimpulkan kualitas transformasi yang dilakukan.

1.3 Manfaat Masalah

- Memberikan pengalaman langsung dalam menerapkan teori pemrosesan citra digital.
- Meningkatkan keterampilan teknis dalam penggunaan Python untuk image processing.
- Mengembangkan kemampuan analisis visual terhadap hasil citra dan histogram.
- Menumbuhkan kreativitas dalam menyelesaikan masalah visualisasi dan pemrosesan gambar nyata.

BAB II

LANDASAN TEORI

Pengolahan citra digital merupakan salah satu bidang dalam ilmu komputer yang berfokus pada pengolahan gambar dalam format digital menggunakan algoritma tertentu. Pengolahan citra memiliki peran yang sangat penting dalam berbagai aplikasi, mulai dari bidang medis, pertahanan, keamanan, hingga hiburan dan media sosial. Dalam praktiknya, pengolahan citra memungkinkan komputer untuk mengenali, memodifikasi, serta mengambil informasi dari gambar digital secara otomatis. Salah satu pendekatan yang paling mendasar dalam pengolahan citra digital adalah melalui analisis warna, segmentasi gambar, dan perbaikan kualitas gambar berdasarkan intensitas pencahayaan.

Citra digital berwarna pada umumnya direpresentasikan dalam ruang warna RGB (Red, Green, Blue), di mana setiap piksel terdiri atas tiga komponen warna dasar tersebut. Deteksi warna pada citra RGB dilakukan dengan memisahkan masing-masing kanal warna untuk dianalisis secara individual. Proses ini bertujuan untuk mengetahui sebaran intensitas warna dan mendeteksi area yang didominasi oleh warna tertentu. Sebagai contoh, apabila ingin mendeteksi warna merah pada suatu gambar, maka dilakukan pemisahan kanal warna merah dan pengamatan nilai intensitasnya dibandingkan dengan kanal hijau dan biru. Hasil dari deteksi warna ini dapat digunakan dalam berbagai aplikasi seperti pelacakan objek, klasifikasi warna, dan segmentasi berbasis warna.

Dalam mendukung proses deteksi dan analisis warna, digunakan pula histogram citra yang menggambarkan distribusi frekuensi intensitas piksel dalam gambar. Histogram dapat menunjukkan sebaran nilai terang-gelap pada suatu gambar dan memberikan informasi apakah gambar tersebut memiliki kontras yang baik atau tidak. Histogram warna RGB, yang terdiri dari tiga grafik terpisah untuk warna merah, hijau, dan biru, memberikan gambaran yang jelas mengenai proporsi dan dominasi warna dalam citra. Melalui analisis histogram, kita dapat menentukan strategi lanjutan dalam pengolahan citra, seperti peningkatan kontras, penyesuaian kecerahan, hingga pemisahan objek dari latar belakang.

Salah satu metode yang digunakan untuk memisahkan bagian penting dari citra adalah thresholding atau pengambangan ambang batas. Thresholding merupakan teknik segmentasi citra yang digunakan untuk memisahkan objek dari latar belakang berdasarkan nilai intensitas piksel. Pada metode ini, piksel yang memiliki intensitas lebih tinggi dari nilai ambang akan dianggap sebagai bagian dari objek, sementara sisanya dianggap sebagai latar belakang. Penentuan nilai ambang ini dapat dilakukan secara manual (fixed threshold) maupun otomatis (seperti metode Otsu dan adaptive thresholding). Dalam konteks deteksi warna, thresholding sangat berguna untuk mengisolasi warna tertentu dari citra dengan cara menetapkan batas nilai untuk kanal warna tertentu agar hanya warna yang diinginkan yang tampil.

Selain itu, tantangan lain yang sering dihadapi dalam pengolahan citra adalah menangani gambar dengan pencahayaan tidak ideal, seperti pada kondisi backlight. Citra backlight adalah citra yang diambil dengan latar belakang yang lebih terang dibandingkan subjek

utama, sehingga bagian wajah atau tubuh menjadi lebih gelap dan kurang terlihat jelas. Untuk mengatasi hal tersebut, dilakukan serangkaian langkah perbaikan kualitas gambar yang meliputi konversi ke grayscale, peningkatan kecerahan (brightness enhancement), serta peningkatan kontras (contrast adjustment). Konversi ke grayscale menghilangkan informasi warna sehingga hanya intensitas piksel yang dianalisis. Selanjutnya, peningkatan brightness digunakan untuk mencerahkan bagian gelap citra, dan peningkatan kontras diterapkan untuk menonjolkan perbedaan antara area terang dan gelap, sehingga detail pada wajah atau tubuh subjek menjadi lebih terlihat.

Implementasi dari semua proses tersebut sangat terbantu dengan adanya perangkat lunak dan bahasa pemrograman modern. Salah satu bahasa pemrograman yang paling sering digunakan dalam pengolahan citra digital adalah Python. Python menyediakan berbagai pustaka (library) yang mendukung manipulasi citra seperti OpenCV untuk pengolahan gambar, NumPy untuk manipulasi matriks piksel, serta Matplotlib untuk visualisasi data dan histogram. Dengan bantuan pustaka-pustaka tersebut, pengguna dapat mengembangkan algoritma pengolahan citra secara efisien, cepat, dan mudah divisualisasikan hasilnya.

Dengan memahami konsep-konsep dasar seperti deteksi warna, histogram, thresholding, perbaikan citra backlight, serta pemanfaatan Python dan pustaka-pustakanya, mahasiswa diharapkan mampu menerapkan pengetahuan ini dalam praktik dan menghasilkan solusi yang tepat terhadap permasalahan visual yang sering terjadi pada citra digital.

BAB III

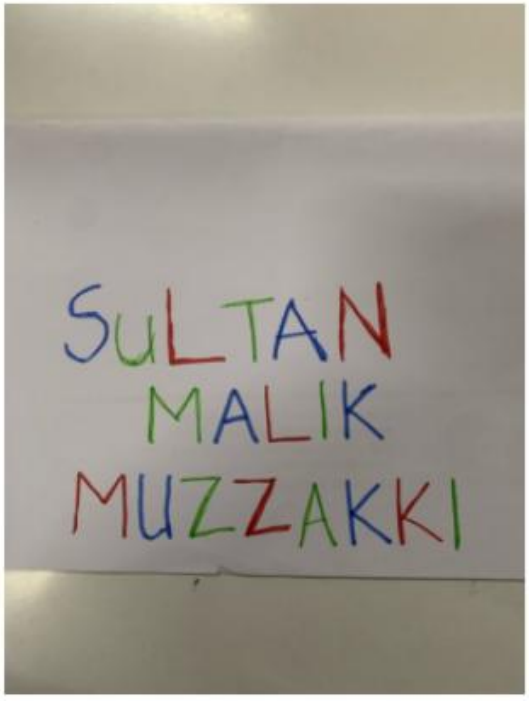
HASIL

Ini adalah hasil praktikum yang telah saya buat :

Gambar yang Saya tampilkan merupakan hasil pemotretan kertas putih yang bertuliskan nama lengkap saya “SULTAN MALIK MUZZAKKI” menggunakan tiga warna tinta yaitu merah, hijau, dan biru. Gambar ini diambil secara pribadi sesuai dengan ketentuan tugas UTS Pengolahan Citra Digital, di mana mahasiswa diminta menuliskan nama menggunakan pena berwarna merah, hijau, dan biru secara bebas pada kertas putih. Gambar kemudian dibaca menggunakan pustaka OpenCV dalam Python melalui fungsi cv2.imread(), yang secara default memuat gambar dalam format warna BGR. Agar tampilan warna sesuai saat ditampilkan dengan Matplotlib, gambar tersebut dikonversi ke format RGB menggunakan cv2.cvtColor(). Selanjutnya, gambar divisualisasikan menggunakan plt.imshow() dan plt.axis('off') digunakan untuk menyembunyikan sumbu koordinat agar tampilan lebih bersih. Tujuan dari proses ini adalah untuk mempersiapkan gambar sebagai bahan pengolahan lebih lanjut seperti deteksi warna dan analisis histogram dalam tugas praktikum UTS.

[21]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread("Sultan.jpg")
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(rgb)
plt.axis('off')
```

[21]:
```
(-0.5, 1199.5, 1599.5, -0.5)
```



Pada bagian bawah, kode menggunakan fungsi plt.subplot() untuk membagi tampilan menjadi empat bagian dalam satu figur. Gambar pertama merupakan **"Original Image"**, yaitu citra asli berwarna yang menampilkan tulisan "SULTAN MALIK MUZZAKKI" dengan tinta merah, hijau, dan biru. Selanjutnya, citra dipisahkan ke dalam tiga saluran warna:

1. **Red Channel**: Menampilkan hanya komponen warna merah dalam bentuk citra grayscale. Tulisan yang berwarna merah terlihat lebih terang, sedangkan bagian lain tampak lebih gelap, menandakan bahwa intensitas warna merah dominan pada bagian tersebut.
2. **Green Channel**: Menampilkan saluran warna hijau. Tulisan "MALIK" terlihat lebih terang dibanding bagian lain karena ditulis dengan tinta hijau, menunjukkan intensitas tinggi pada channel ini.
3. **Blue Channel**: Menampilkan saluran warna biru. Huruf-huruf biru dari kata "SULTAN" dan "MUZZAKKI" tampak lebih terang, menandakan bahwa bagian tersebut memiliki kandungan warna biru yang dominan.

Setiap channel ditampilkan dalam skala abu-abu (cmap="gray") untuk memperjelas perbedaan intensitas piksel dalam masing-masing kanal warna. Dengan demikian, hasil visualisasi ini sangat membantu dalam proses analisis dan deteksi warna, karena kita dapat dengan jelas melihat bagian mana dari citra yang mendominasi pada setiap channel warna RGB. Proses ini merupakan bagian penting dari analisis awal sebelum melakukan segmentasi atau pemrosesan lebih lanjut seperti thresholding dan histogram analisis.

```
plt.subplot(2, 2, 1)
plt.imshow(rgb)
plt.title('Original Image')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(rgb[:,:,0], cmap="gray")
plt.title('Red Channel')
plt.axis('off')

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

plt.subplot(2, 2, 4)
plt.imshow(rgb[:,:,2], cmap="gray")
plt.title('Blue Channel')
plt.axis('off')

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

Gambar yang ini menunjukkan proses pembuatan histogram intensitas dari masing-masing saluran warna (Red, Green, dan Blue) pada citra digital yang menampilkan tulisan tangan nama "SULTAN MALIK MUZZAKKI". Tujuan dari proses ini adalah untuk menganalisis distribusi intensitas piksel dari setiap kanal warna dalam bentuk grafik, sehingga kita dapat memahami karakteristik pencahayaan dan sebaran warna dalam gambar secara lebih mendalam.

Setiap bagian pada tampilan ini terbagi menjadi dua kolom: di sisi kiri ditampilkan citra grayscale dari masing-masing channel warna (merah, hijau, biru), dan di sisi kanan ditampilkan histogram dari masing-masing channel tersebut.

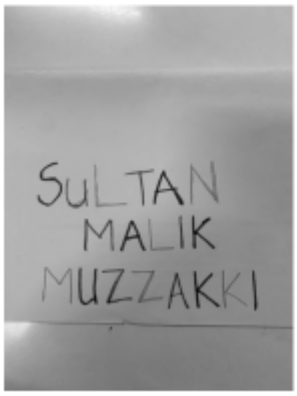
1. **Histogram Warna Merah (Red Channel)**
   Citra grayscale di bagian atas kiri menunjukkan komponen warna merah dalam citra. Di sebelah kanannya, grafik histogram memperlihatkan distribusi intensitas piksel merah. Terlihat adanya dua hingga tiga puncak tajam, yang menunjukkan banyaknya piksel merah dengan intensitas tertentu — ini terutama berasal dari bagian tulisan yang menggunakan tinta merah (misalnya pada kata “SULTAN” dan “MUZZAKKI”). Nilai intensitas yang tinggi mengindikasikan bahwa terdapat area dengan warna merah yang sangat dominan.

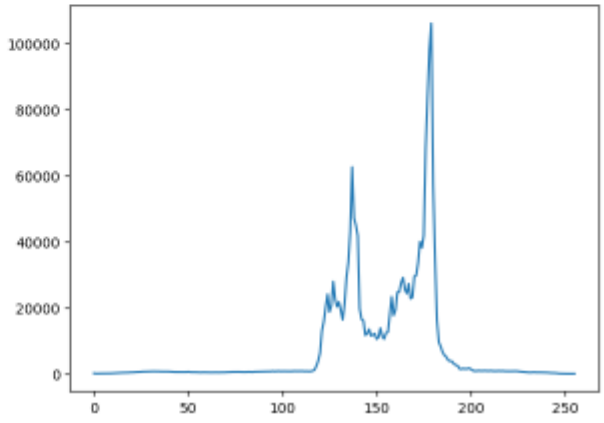
2. **Histogram Warna Hijau (Green Channel)**
   Gambar kedua dari atas menunjukkan citra grayscale dari kanal hijau, dengan penekanan pada bagian tulisan “MALIK” yang menggunakan tinta hijau. Histogramnya menunjukkan puncak intensitas yang menandakan konsentrasi piksel hijau dalam area tersebut. Bentuk histogram yang memiliki beberapa puncak dapat menandakan adanya variasi bayangan atau pencahayaan pada permukaan kertas putih.

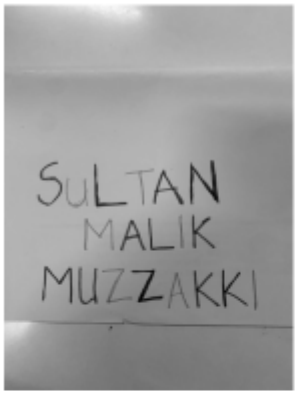
3. **Histogram Warna Biru (Blue Channel)**
   Gambar paling bawah menunjukkan channel biru, dengan fokus pada tulisan berwarna biru yang terdapat di beberapa huruf dalam nama. Histogram pada kanal biru juga menunjukkan puncak-puncak yang sesuai dengan area dominan biru. Intensitasnya cenderung sedikit lebih rendah daripada merah dan hijau, yang bisa jadi karena proporsi penggunaan warna biru lebih sedikit pada gambar ini.

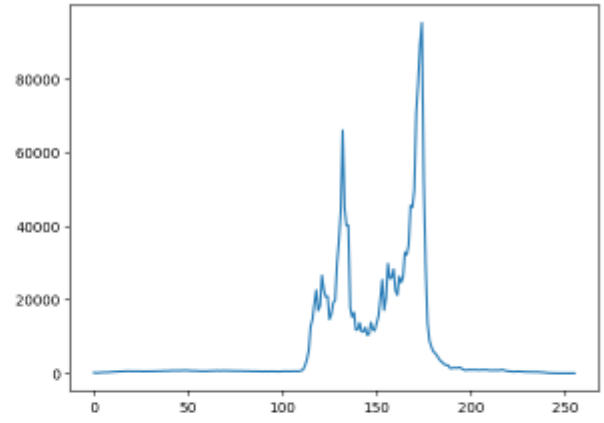
Secara keseluruhan, visualisasi histogram ini sangat membantu dalam mengenali distribusi dan kekuatan warna pada setiap channel. Informasi dari histogram juga berguna dalam proses berikutnya seperti penentuan ambang batas (thresholding), koreksi pencahayaan, atau segmentasi citra. Analisis ini menunjukkan bahwa setiap warna dalam citra memiliki distribusi yang berbeda dan dapat diproses lebih lanjut secara selektif berdasarkan intensitasnya.

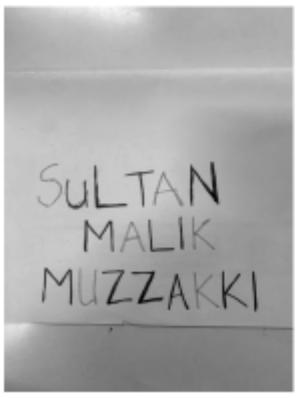
[3]:
```
merah=rgb[:,:,0]
fig, axs = plt.subplots(1,2, figsize = (15,5))
hist = cv2.calcHist([merah],[0],None,[256],[0,256])
axs[0].imshow(merah, cmap='gray')
axs[0].axis('off')
axs[1].plot(hist)
plt.show()

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[0].axis('off')
axs[1].plot(hist)
plt.show()

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[0].axis('off')
axs[1].plot(hist)
plt.show()
```

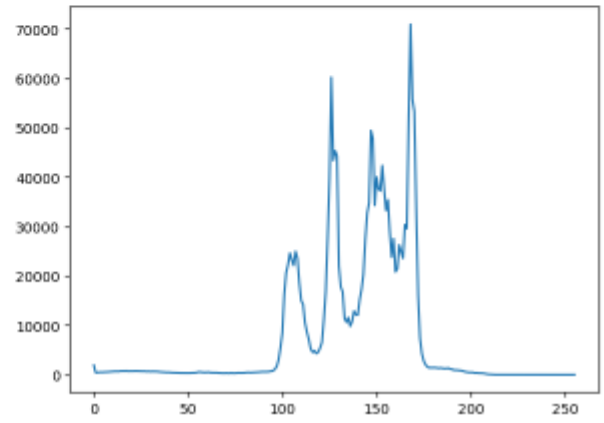












Gambar di atas menunjukkan hasil proses segmentasi warna berdasarkan ruang warna HSV yang dilakukan untuk mendeteksi dan memisahkan tulisan tangan berwarna merah, hijau, dan biru pada gambar. Proses diawali dengan mengubah format gambar dari BGR ke HSV karena ruang warna HSV lebih efektif dalam memisahkan warna berdasarkan karakteristik hue (warna), saturation (kejenuhan), dan value (kecerahan). Kemudian, dilakukan pembuatan *mask* untuk masing-masing warna dengan menggunakan fungsi cv2.inRange() berdasarkan rentang nilai hue yang sesuai, yaitu merah (0–10), hijau (36–70), dan biru (94–126).

Masing-masing mask menghasilkan citra biner (hitam-putih) di mana bagian putih menunjukkan keberadaan warna yang sesuai. Setelah mask individual berhasil dibuat, dilakukan penggabungan antara dua atau tiga warna menggunakan operasi logika bitwise. Gambar pertama bertuliskan “NONE” menunjukkan hasil kosong karena tidak ada mask yang aktif secara visual pada tahap itu. Gambar kedua dengan judul “Blue Mask” menampilkan huruf-huruf yang ditulis dengan tinta biru seperti “S”, “A”, dan “K” yang tampak terang, menunjukkan bahwa bagian-bagian tersebut dikenali sebagai biru.

Gambar ketiga “Red-Blue Mask” merupakan kombinasi dari mask merah dan biru, sehingga menampilkan lebih banyak huruf dari nama “SULTAN” dan “MUZZAKKI” yang ditulis dengan tinta merah dan biru. Gambar keempat yaitu “Red-Green-Blue Mask” adalah gabungan dari ketiga mask warna utama, dan hasilnya menampilkan seluruh tulisan “SULTAN MALIK MUZZAKKI” secara lengkap. Hasil ini membuktikan bahwa teknik segmentasi warna berbasis HSV mampu mendeteksi warna primer dalam citra dengan cukup baik dan akurat, serta dapat digunakan sebagai dasar untuk proses pengolahan citra lanjutan seperti klasifikasi warna atau pelacakan objek.

```
[5]
image_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
gray = cv2.cvtColor(img, 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 = cv2.bitwise_or(mask_red, mask_blue)
combined_mask2 = cv2.bitwise_or(combined_mask1, mask_green)

(_, binary1) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY)

axs[0, 0].imshow(binary1, cmap='gray')
axs[0, 0].set_title('NONE')

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

axs[1, 0].imshow(combined_mask1, cmap='gray')
axs[1, 0].set_title('Red-Blue Mask')

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

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

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

ode tersebut digunakan untuk menampilkan gambar asli bernama *"Sultan2.jpg"* menggunakan pustaka OpenCV dan Matplotlib. Pertama, gambar dibaca dalam format BGR, kemudian diubah menjadi format RGB agar bisa ditampilkan dengan benar di Matplotlib. Setelah itu, gambar ditampilkan tanpa sumbu koordinat dan diberi judul **"Gambar asli"**. Hasil output menunjukkan gambar seseorang yang berdiri di luar ruangan, dengan latar belakang berupa bangunan, tiang listrik, kabel, dan pepohonan. Gambar ini merupakan tampilan awal sebelum dilakukan proses pengolahan citra lebih lanjut.

```
img2 = cv2.imread("Sultan2.jpg")
rgb2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
baris = rgb2.shape[0]
kolom = rgb2.shape[1]
plt.imshow(rgb2)
plt.title("Gambar asli")
plt.axis('off')
```

```
(-0.5, 1199.5, 1599.5, -0.5)
```

Kode tersebut digunakan untuk mengubah gambar berwarna menjadi **grayscale** (abu-abu) secara manual dan kemudian menampilkannya. Berikut penjelasan dalam bentuk kalimat:

Pertama, dibuat sebuah fungsi bernama show(x, y) yang digunakan untuk menampilkan gambar dengan pewarnaan skala abu-abu. Fungsi ini memanfaatkan imshow() dari Matplotlib dengan parameter cmap='gray', yang artinya gambar akan ditampilkan dalam format grayscale, lalu diberi judul dan sumbu koordinatnya disembunyikan.

Selanjutnya, gambar grayscale dibuat dengan cara menghitung **rata-rata dari tiga kanal warna (R, G, B)** untuk setiap piksel. Nilai rata-rata tersebut kemudian diubah ke dalam tipe data uint8 agar sesuai dengan format citra digital 8-bit.

Proses perhitungan dilakukan melalui dua *loop* bersarang: for i in range(baris) dan for j in range(kolom), yang mengiterasi setiap piksel dalam gambar. Nilai rata-rata dari piksel RGB di posisi [i, j] dihitung menggunakan np.mean() dan disimpan ke dalam matriks grayM.

Terakhir, gambar hasil konversi grayscale ditampilkan dengan memanggil fungsi show(grayM, "Grayscale").

Hasil output-nya adalah gambar yang sama seperti sebelumnya, namun kini dalam format abu-abu. Warna-warna asli telah diubah menjadi intensitas terang-gelap berdasarkan nilai rata-rata ketiga kanal warna. Ini merupakan tahap awal penting dalam banyak aplikasi pengolahan citra seperti deteksi tepi atau pengenalan objek.

[11]:
```
def show(x,y):
    plt.imshow(x, cmap='gray')
    plt.title(y)
    plt.axis('off')
    plt.show()
```

[13]:
```
grayM = np.zeros((baris, kolom), dtype=np.uint8)
for i in range(baris):
    for j in range(kolom):
        grayM[i,j]=np.uint8(np.mean(rgb2[i,j]))

show(grayM, "Grayscale")
```

Kode pada gambar ini digunakan untuk **mencerahkan gambar grayscale** yang sebelumnya telah dihasilkan. Berikut penjelasan lengkapnya dalam bentuk kalimat:

Pertama, ditentukan nilai beta = 20, yang merupakan nilai penambah kecerahan. Nilai ini akan ditambahkan ke setiap piksel dalam gambar grayscale.

Kemudian dibuat matriks kosong citra_cerah dengan ukuran yang sama seperti gambar asli, namun memiliki 3 kanal warna (meskipun isinya nanti tetap grayscale), dan bertipe float (default saat np.zeros() digunakan tanpa dtype).

Proses mencerahkan dilakukan dengan dua *loop* bersarang yang mengakses setiap piksel satu per satu:

- Nilai piksel pada posisi [x, y] di gambar grayscale (grayM) ditambah dengan beta.
- Hasilnya (gyx) disimpan ke dalam matriks citra_cerah pada posisi yang sama.

Setelah semua piksel diperbarui, citra_cerah dikonversi ke dalam tipe data uint8, yang sesuai untuk format citra 8-bit, agar bisa ditampilkan dengan benar.

Terakhir, gambar yang telah diberi kecerahan ditampilkan dengan fungsi show() dan diberi judul **"Grayscale yang dipercepat"**.

Hasil output menunjukkan gambar yang tampak lebih terang dibandingkan versi grayscale sebelumnya. Proses ini berguna untuk meningkatkan visibilitas detail dalam gambar yang terlalu gelap atau untuk persiapan tahap pengolahan citra lebih lanjut.

```
]: beta = 20
   citra_cerah = np.zeros((baris, kolom, 3))

   for x in range(baris):
       for y in range(kolom):
           gyx = grayM[x,y] + beta
           citra_cerah[x,y] = gyx

   citra_cerah = citra_cerah.astype(np.uint8)
   show(citra_cerah, "Grayscale yang dipercerah")
```

Kode ini digunakan untuk **meningkatkan kontras** pada gambar grayscale. Berikut penjelasan dalam bentuk kalimat:

Pertama, nilai **alpha = 1.2** ditentukan sebagai faktor penyesuaian kontras. Nilai ini digunakan untuk mengalikan setiap piksel dalam gambar grayscale. Semakin besar nilai alpha, semakin tinggi kontras yang dihasilkan.

Kemudian, dibuat sebuah matriks kosong bernama citra_kontras dengan ukuran seperti gambar asli dan 3 kanal warna (meskipun hanya grayscale), yang akan menyimpan hasil setelah kontras ditingkatkan.

Proses peningkatan kontras dilakukan menggunakan dua *loop* bersarang:

- Untuk setiap piksel [x, y], nilai intensitas dari gambar grayscale (grayM) dikalikan dengan alpha.
- Hasil perkalian disimpan ke dalam matriks citra_kontras.

Setelah itu, matriks hasil konversi dikembalikan ke tipe data uint8 agar bisa ditampilkan sebagai gambar digital 8-bit.

Terakhir, gambar ditampilkan menggunakan fungsi show() dan diberi judul **"Grayscale yang diperkontras"**.

Hasil output menunjukkan gambar yang tampak lebih tajam dan memiliki perbedaan terang-gelap yang lebih jelas dibandingkan gambar grayscale sebelumnya. Warna terang menjadi lebih terang dan warna gelap menjadi lebih gelap — inilah efek peningkatan kontras.

Peningkatan kontras seperti ini sering digunakan untuk menonjolkan detail pada objek dalam gambar, terutama saat citra tampak datar atau buram.

```
alpha = 1.2
citra_kontras = np.zeros((baris, kolom, 3))

for x in range(baris):
    for y in range(kolom):
        gyx = grayM[x,y] * alpha
        citra_kontras[x,y] = gyx

citra_kontras = citra_kontras.astype(np.uint8)
show(citra_kontras, "Grayscale yang diperkontras")
```

Kode tersebut digunakan untuk menghasilkan gambar grayscale yang telah ditingkatkan **kecerahan dan kontrasnya secara bersamaan**.

Pertama, ditentukan dua parameter utama: beta = 20 untuk menambah kecerahan dan alpha = 1.2 untuk meningkatkan kontras. Kemudian, dibuat sebuah matriks kosong bernama citra_hasil untuk menyimpan hasil transformasi piksel.

Selanjutnya, setiap piksel dari gambar grayscale (grayM) diproses dengan cara mengalikan nilainya dengan alpha (untuk membuat perbedaan terang-gelap lebih tajam), lalu ditambahkan dengan beta (untuk membuat keseluruhan gambar tampak lebih terang). Hasil transformasi ini disimpan di posisi yang sesuai dalam matriks citra_hasil.

Setelah semua piksel diolah, matriks hasil dikonversi ke tipe data uint8 agar sesuai dengan format citra digital standar, lalu ditampilkan dengan judul **"Grayscale yang dipercepat dan diperkontras"**.

Hasil akhir yang ditampilkan adalah gambar grayscale dengan tampilan yang lebih terang dan lebih tajam dibandingkan sebelumnya, sehingga detail dalam gambar menjadi lebih mudah dikenali dan tampak lebih jelas. Teknik seperti ini sering digunakan untuk meningkatkan kualitas visual gambar sebelum dilakukan analisis lebih lanjut.

```
beta = 20
alpha = 1.2
citra_hasil = np.zeros((baris, kolom, 3))

for x in range(baris):
    for y in range(kolom):
        gyx = alpha *grayM[x,y] + beta
        citra_hasil[x,y] = gyx

citra_hasil = citra_hasil.astype(np.uint8)
show(citra_hasil, "Grayscale yang dipercerah dan diperkontras")
```

BAB IV

PENUTUP

Berdasarkan hasil praktikum Ujian Tengah Semester mata kuliah Pengolahan Citra Digital, dapat disimpulkan bahwa berbagai teknik dasar pengolahan citra seperti konversi ke grayscale, peningkatan kecerahan (brightness), peningkatan kontras, serta kombinasi keduanya, mampu memperbaiki kualitas visual citra, khususnya pada kasus citra dengan efek **backlight**. Dalam kasus ini, bagian objek utama dalam citra (wajah atau tubuh) yang semula gelap akibat cahaya latar yang kuat berhasil ditingkatkan visibilitasnya dengan metode pengolahan citra digital yang tepat.

Proyek ini juga mencakup proses deteksi warna (merah, hijau, biru), pencarian nilai ambang batas minimum hingga maksimum untuk segmentasi warna, serta analisis histogram dari tiap saluran warna. Seluruh tahapan ini memperkuat pemahaman tentang bagaimana informasi visual disimpan dan diinterpretasikan dalam bentuk digital melalui representasi piksel dan intensitas warna.

Secara umum, proyek ini telah membekali mahasiswa dengan pengalaman praktis dalam mengaplikasikan konsep dasar pemrosesan citra digital menggunakan bahasa pemrograman Python dan pustaka seperti OpenCV dan Matplotlib. Mahasiswa juga dilatih untuk berpikir kritis dalam memilih metode pemrosesan yang sesuai dengan kondisi citra dan tujuan yang ingin dicapai.

Melalui pelaksanaan tugas ini, mahasiswa diharapkan dapat mengembangkan keterampilan teknis serta logika analitis yang diperlukan untuk menghadapi tantangan nyata dalam dunia pengolahan citra. Pengetahuan dan pengalaman ini diharapkan dapat menjadi fondasi penting untuk pengembangan teknologi berbasis visual di masa depan, seperti sistem deteksi otomatis, pengenalan pola, serta aplikasi dalam kecerdasan buatan dan computer vision.

DAFTAR PUSTAKA

Gogor C. Setyawan, & Yosephine M.S. Mendrofa. (2022). SEGMENTASI CITRA GESTURE TANGAN BERBASIS RUANG WARNA HSV. *Infact: International Journal of Computers*, *6*(2). Retrieved from https://journal.ukrim.ac.id/index.php/JIF/article/view/304

Goenawan, A. D., Rachman, M. B. A., & Pulungan, M. P. (2022, January 29). Identifikasi Warna Pada Objek Citra Digital Secara Real Time Menggunakan Pengolahan Model Warna HSV. *Jurnal Teknik Informatika Dan Elektro*, *4*(1), 68-74. https://doi.org/https://doi.org/10.55542/jurtie.v4i1.430