

UTS
PENGOLAHAN CITRA



NAMA : RIZAL WIRA PAMBUDI

NIM : 202331175

KELAS : A

DOSEN : Dwina Kuswardani, Dr. ,Dra.,M.Kom

NO.PC : 09

ASISTEN : 1. CLARENCA SWEETDIVA PEREIRA

2. VIANA SALSABILA FAIRUZ SYAHLA

3. KASHRINA MASYID AZKA

4. SASIKIRANA RAMADHANTY SETIAWAN PUTRI

INSTITUT TEKNOLOGI PLN
TEKNIK INFORMATIKA
2024/2025

DAFTAR ISI

Daftar Isi

Ketik judul bab (Tingkat 1)	1
Ketik judul bab (Tingkat 2).....	2
Ketik judul bab (Tingkat 3).....	3
Ketik judul bab (Tingkat 1)	4
Ketik judul bab (Tingkat 2).....	5
Ketik judul bab (Tingkat 3).....	6

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Apa saja teori-teori yang dipelajari dalam Pemrosesan Citra Digital?
2. Apa saja jenis-jenis citra yang digunakan dalam Pemrosesan Citra Digital?
3. Bagaimana cara kerja Teknik Peningkatan Kualitas pada Citra secara Digital?
4. Bagaimana mengimplementasikan Teknik Peningkatan Kualitas Citra Digital untuk pengenalan warna?
5. Bagaimana mengimplementasikan Teknik Peningkatan Kualitas Citra Digital untuk meningkatkan kualitas citra digital yang memiliki kekurangan kualitas dari gambar aslinya?

1.2 Tujuan Masalah

Untuk memberikan pengetahuan teoritis baru yang berasal dari kajian teori mengenai teori-teori dalam mata kuliah Pemrosesan Citra Digital dan Implementasinya dalam meningkatkan kualitas Citra Digital

1.3 Manfaat Masalah

- Mempelajari Ilmu Teori Pemrosesan Digital yang berasal dari Kajian Ilmiah mengenai Teknik Peningkatan Kualitas Citra Digital
- Memahami jenis-jenis warna dalam citra, dan memahami cara untuk melakukan splitting citra
- Paham cara menggunakan histogram untuk menganalisis kadar warna pada citra
- Menggunakan data analisis warna dan metodologi peningkatan kualitas citra sehingga praktikan dapat menentukan aksi yang tepat untuk secara efisien meningkatkan kualitas citra.

BAB II

LANDASAN TEORI

OpenCV

OpenCV (Open Source Computer Vision Library) merupakan pustaka sumber terbuka yang dioptimalkan untuk penerapan visi komputer dan pengolahan citra digital. Dalam penelitian terbaru, OpenCV diimplementasikan untuk berbagai aplikasi seperti pengenalan wajah dengan tingkat akurasi mencapai 85% menggunakan metode PCA Eigenface serta kemampuan membedakan wajah yang terdapat pada database dan yang tidak terdapat dalam database¹⁸. Library ini menyediakan berbagai fungsi dan algoritma untuk pengolahan citra yang mendukung bahasa pemrograman Python dengan kemampuan memproses berbagai jenis citra.

Jupyter Notebook

Jupyter Notebook adalah aplikasi web interaktif yang memungkinkan peneliti dan praktisi untuk mengembangkan, menjalankan, dan mendokumentasikan kode program secara langsung dengan antarmuka yang intuitif. Dalam pengolahan citra digital, Jupyter Notebook sering digunakan untuk prototyping cepat dan visualisasi hasil pengolahan citra secara real-time. Keunggulan Jupyter Notebook terletak pada kemampuannya mengkombinasikan kode, visualisasi, dan narasi dalam satu dokumen yang komprehensif, sehingga mempermudah analisis data citra dan berbagi hasil penelitian dengan lebih efektif.

Pyplot

Pyplot merupakan modul dari library Matplotlib di Python yang banyak digunakan untuk visualisasi data dalam pengolahan citra digital. Modul ini menyediakan antarmuka yang mirip dengan MATLAB untuk menciptakan berbagai plot visual dari data citra. Dalam penelitian pengolahan citra digital, Pyplot sering digunakan untuk menampilkan hasil pengolahan citra, membandingkan citra sebelum dan sesudah diproses, serta memvisualisasikan berbagai fitur citra seperti distribusi warna dan profil intensitas. Kemampuan Pyplot untuk membuat visualisasi interaktif membantu peneliti dalam menganalisis dan menginterpretasikan hasil pengolahan citra dengan lebih mendalam.

Histogram

Histogram dalam pengolahan citra digital merupakan representasi grafis dari distribusi intensitas piksel dalam suatu citra. Teknik ini sangat penting untuk analisis dan pemrosesan citra, termasuk dalam perbaikan kontras, ekualisasi, dan segmentasi citra. Histogram memberikan informasi statistik tentang persebaran nilai intensitas piksel yang membantu dalam penentuan nilai ambang batas (threshold) yang optimal untuk pemisahan objek dari latar belakangnya. Dalam implementasi pengolahan citra modern, histogram sering menjadi langkah awal untuk analisis karakteristik citra sebelum proses pengolahan lebih lanjut.

Konsep Dasar Citra Digital

Pengelolaan Citra Digital

Pengolahan citra digital adalah disiplin ilmu yang mempelajari teknik-teknik untuk mengolah dan memanipulasi citra menggunakan komputer. Menurut Kusumanto dan Tompunu (2011), citra

harus direpresentasikan secara numerik dengan nilai diskrit agar dapat diolah dengan mudah, proses ini disebut dengan digitalisasi citra¹⁹. Pengolahan citra digital telah diterapkan di berbagai bidang dan aplikasi untuk meningkatkan kualitas gambar, mengekstrak informasi berguna, dan melakukan analisis kompleks yang tidak dapat dilakukan secara manual²⁰.

Teknik Pengambilan Gambar Citra

Teknik pengambilan gambar citra merupakan langkah awal yang krusial dalam proses pengolahan citra digital. Penelitian terbaru menunjukkan bahwa kualitas akuisisi citra sangat mempengaruhi hasil akhir pengenalan objek, seperti yang ditunjukkan pada penelitian pengenalan wajah di mana faktor pencahayaan, posisi wajah, dan jarak pengambilan gambar menghasilkan nilai akurasi yang berbeda¹⁸. Pengambilan citra yang tepat dengan memperhatikan pencahayaan, sudut, dan resolusi akan menghasilkan data citra berkualitas tinggi yang memudahkan proses pengolahan selanjutnya.

Jenis Citra

BGR

BGR (Blue, Green, Red) adalah format warna standar yang digunakan oleh OpenCV untuk merepresentasikan citra berwarna. Format ini merupakan kebalikan dari format RGB konvensional, di mana urutan kanal warnanya adalah biru, hijau, dan merah. Dalam implementasi pengolahan citra menggunakan OpenCV, pemahaman tentang format BGR menjadi penting karena perbedaan urutan kanal dapat mempengaruhi hasil pengolahan citra. Saat bekerja dengan citra berwarna di OpenCV, konversi antara ruang warna BGR dan ruang warna lainnya seperti RGB atau HSV sering dilakukan untuk memfasilitasi ekstraksi fitur tertentu.

RGB

RGB (Red, Green, Blue) adalah model warna aditif yang umum digunakan dalam representasi dan pengolahan citra digital. Penelitian menunjukkan bahwa citra RGB memiliki keterbatasan dalam mengatur kontras pencahayaan yang sulit dibandingkan dengan model warna lain¹⁹. Setiap piksel dalam citra RGB terdiri dari tiga nilai yang merepresentasikan intensitas warna merah, hijau, dan biru, yang ketika dikombinasikan dapat menghasilkan berbagai macam warna yang terlihat oleh mata manusia.

Grayscale

Grayscale atau citra abu-abu merupakan representasi citra di mana nilai setiap piksel hanya menunjukkan intensitas dari hitam ke putih. Dalam pengolahan citra digital, konversi ke skala abu-abu sering menjadi langkah awal penting sebelum melakukan teknik pengolahan lebih lanjut²⁰. Citra grayscale mengandung informasi luminance (kecerahan) tanpa informasi warna, sehingga mengurangi kompleksitas komputasi dan meningkatkan efisiensi dalam berbagai algoritma pengolahan citra seperti deteksi tepi dan segmentasi.

Binary

Citra biner adalah jenis citra yang hanya memiliki dua nilai piksel, yaitu hitam dan putih (0 dan 1). Hasil dari proses segmentasi menggunakan teknik thresholding adalah citra biner yang dapat memisahkan objek dari latar belakangnya dengan jelas²⁰. Penggunaan citra biner sangat efektif untuk identifikasi objek, seperti yang diterapkan dalam penelitian penghitungan ikan hias menggunakan metode BLOB di mana objek dapat dideteksi melalui identifikasi kumpulan piksel dengan warna yang berbeda¹⁷.

Splitting Citra

Splitting citra adalah teknik pemisahan citra menjadi komponen-komponen penyusunnya, seperti memisahkan citra RGB menjadi kanal R, G, dan B yang terpisah. Teknik ini memungkinkan pemrosesan khusus pada masing-masing kanal warna secara independen. Dalam aplikasi pengolahan citra modern, splitting citra digunakan untuk menganalisis karakteristik spektral objek dan meningkatkan fitur-fitur tertentu dalam citra. Keuntungan dari splitting citra adalah kemampuannya untuk mengidentifikasi pola atau anomali yang mungkin tidak terlihat dalam citra komposit asli.

Teknik Perbaikan Kualitas Citra

Operasi Titik

Operasi titik dalam pengolahan citra digital adalah teknik manipulasi nilai piksel berdasarkan nilai piksel itu sendiri tanpa mempertimbangkan nilai piksel tetangganya. Teknik ini meliputi penyesuaian kecerahan, kontras, dan thresholding yang diterapkan pada implementasi pengolahan citra untuk meningkatkan kualitas visual. Pada penelitian penerapan teknik ambang batas (thresholding), operasi titik digunakan untuk memisahkan objek dari latar belakang dengan mengubah gambar menjadi gambar biner²⁰, yang memudahkan proses analisis dan ekstraksi informasi dari citra.

Operasi Spasial

Operasi spasial adalah teknik pengolahan citra yang menghitung nilai piksel berdasarkan nilai piksel di sekitarnya dalam jendela atau kernel tertentu. Dalam penelitian terbaru, operasi spasial seperti pengurangan noise menggunakan keburaman rata-rata diterapkan sebelum melakukan thresholding untuk meningkatkan kualitas segmentasi²⁰. Teknik ini efektif untuk menghilangkan gangguan pada citra, menghaluskan tepi objek, atau bahkan meningkatkan ketajaman tepi objek melalui berbagai jenis filter spasial.

Operasi Transformasi

Operasi transformasi dalam pengolahan citra melibatkan konversi representasi citra dari domain spasial ke domain frekuensi atau domain lainnya. Dalam penelitian pengolahan citra digital modern, transformasi citra memungkinkan analisis komponen frekuensi tinggi dan rendah dari citra secara terpisah. Transformasi seperti Fourier Transform dan Discrete Cosine Transform (DCT) banyak digunakan dalam kompresi citra dan pemfilteran berbasis frekuensi. Dalam teknik pengolahan citra canggih, operasi transformasi menawarkan cara alternatif untuk menganalisis dan memodifikasi karakteristik citra yang sulit dilakukan di domain spasial.

Konkurensi

Konkurensi dalam pengolahan citra digital merujuk pada kemampuan untuk melakukan beberapa operasi pengolahan secara bersamaan atau paralel, yang dapat secara signifikan meningkatkan kinerja dan efisiensi. Dalam implementasi modern pengolahan citra, teknik konkurensi memanfaatkan arsitektur multi-core CPU, GPU, atau teknologi komputasi paralel lainnya untuk mempercepat operasi pengolahan citra yang intensif komputasi. Pendekatan konkurensi sangat bermanfaat untuk aplikasi real-time seperti pemrosesan video atau pengolahan citra dalam jumlah besar, di mana waktu pemrosesan menjadi faktor kritis.

Konvolusi

Konvolusi adalah operasi matematika fundamental dalam pengolahan citra digital yang menggabungkan dua fungsi untuk menghasilkan fungsi ketiga. Dalam konteks pengolahan citra, konvolusi diterapkan dengan menggeser kernel (matriks bobot) di atas citra untuk menghitung nilai piksel baru berdasarkan kombinasi linear piksel tetangga. Teknik konvolusi mendasari berbagai operasi pengolahan citra seperti penghalusan (smoothing), penajaman (sharpening), dan deteksi tepi (edge detection). Keefektifan operasi konvolusi sangat ditentukan oleh desain kernel yang sesuai dengan karakteristik citra dan tujuan pengolahan.

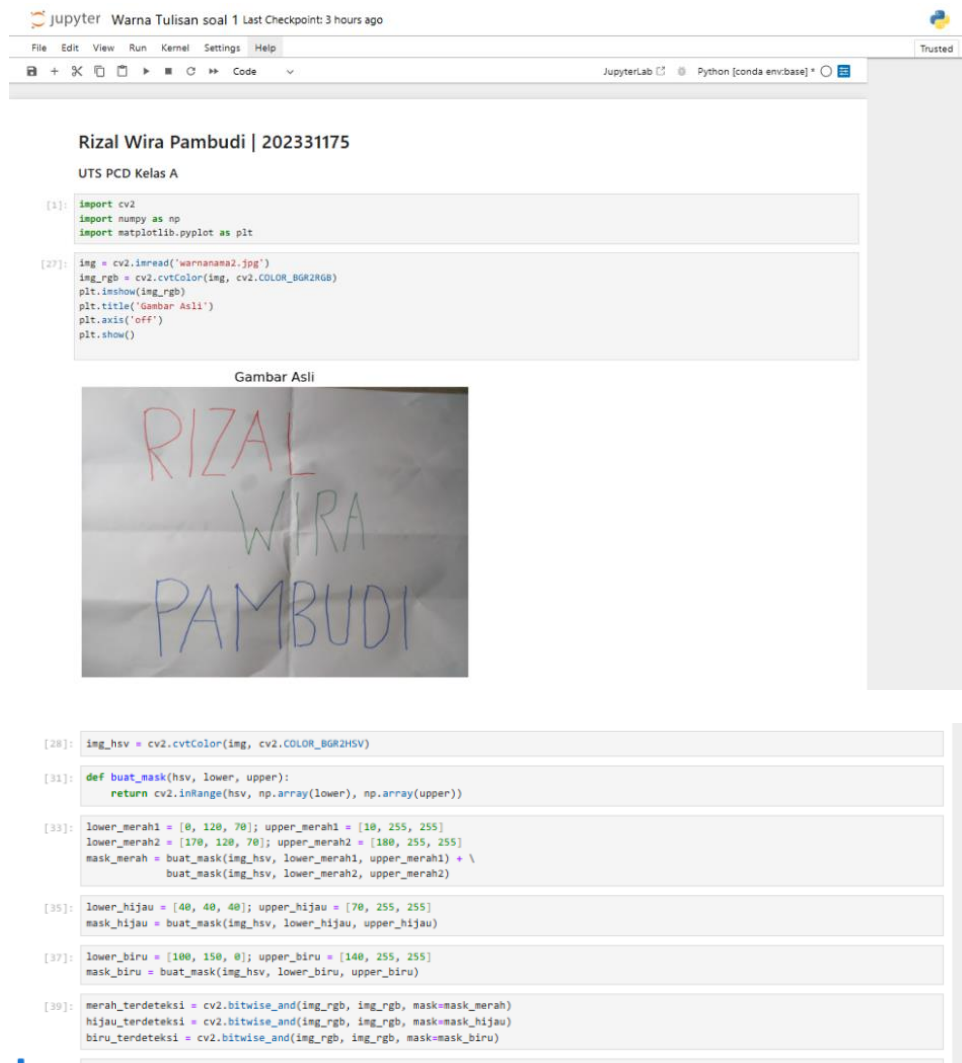
Filter Batas

Filter batas atau deteksi tepi adalah teknik pengolahan citra yang berfokus pada identifikasi titik-titik dalam citra digital di mana terjadi perubahan intensitas yang signifikan. Implementasi filter batas sering melibatkan penerapan teknik ambang batas (thresholding) untuk memisahkan objek dari latar belakangnya, seperti yang dijelaskan dalam penelitian tentang pengolahan citra digital dengan penerapan teknik ambang batas menggunakan OpenCV[20]. Berbagai metode deteksi tepi seperti Sobel, Prewitt, dan Canny memungkinkan ekstraksi informasi penting tentang bentuk dan struktur objek dalam citra, yang sangat berguna untuk aplikasi seperti segmentasi citra, pengenalan pola, dan rekonstruksi objek 3D.

BAB III

HASIL

Soal 1



Pada gambar ini saya import selalu library yang dibutuhkan, lalu saya load gambar saya kedalam variable `img`, lalu di `imshow`, lalu di jadikan warna HSV, yang penting untuk masking. Lalu lanjut dengan memperoleh warna warna dominan dan menambahkan mask nya, lalu membuat variabel merah, hijau, biru untuk memasukkan hasil warna terdeteksi.


```
[41]: titles = ['Gambar Asli', 'Merah Terdeteksi', 'Hijau Terdeteksi', 'Biru Terdeteksi']
      images = [img_rgb, merah_terdeteksi, hijau_terdeteksi, biru_terdeteksi]

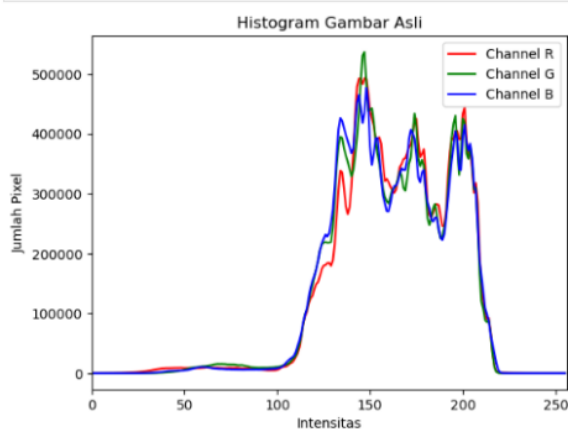
      plt.figure(figsize=(12, 8))
      for i, (im, title) in enumerate(zip(images, titles), 1):
          plt.subplot(2, 2, i)
          plt.imshow(im)
          plt.title(title)
          plt.axis('off')
      plt.tight_layout()
      plt.show()
```



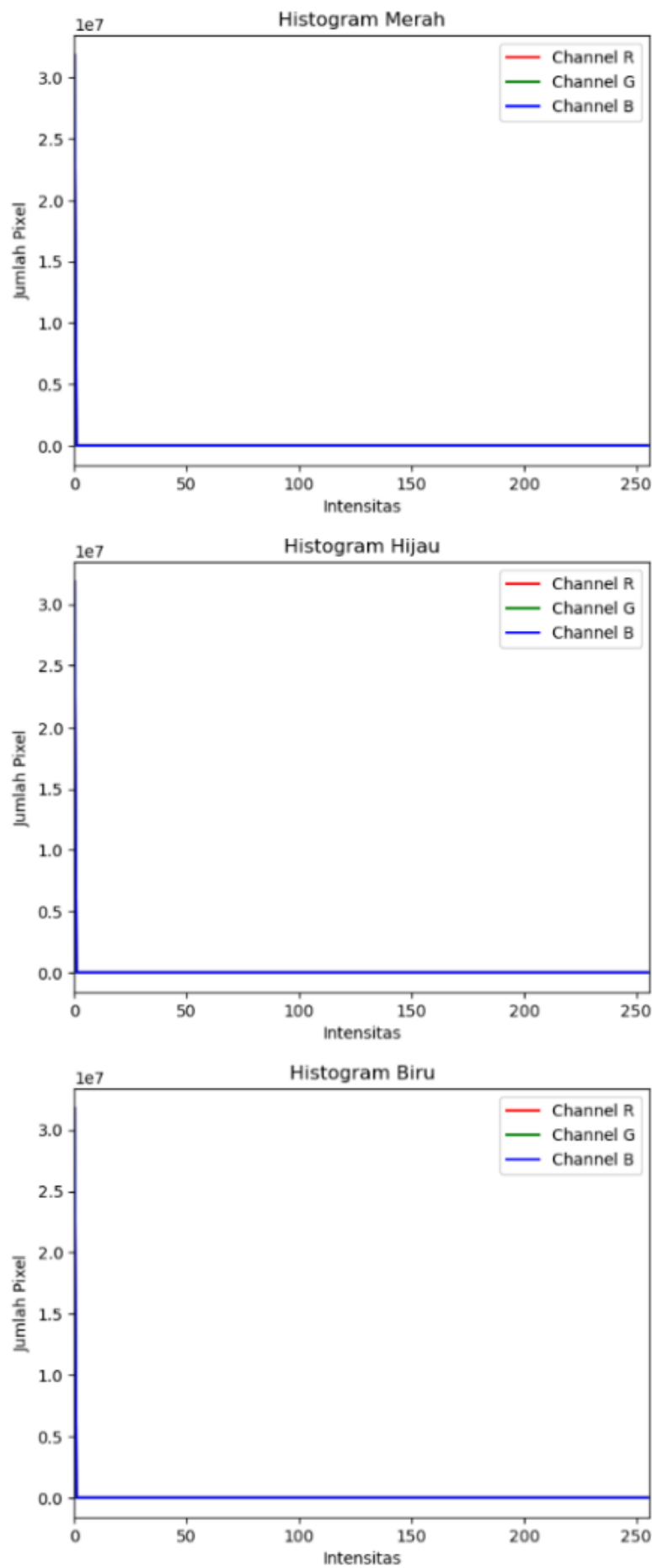
Lalu menggunakan pyplot membuat masing masing gambar hasil masking dari warna warna sebelumnya.

```
[42]: def plot_histogram(image, judul):
      plt.figure()
      channel_colors = ('r', 'g', 'b')
      for idx, col in enumerate(channel_colors):
          hist = cv2.calcHist([image], [idx], None, [256], [0, 256])
          plt.plot(hist, color=col, label=f'Channel {col.upper()}')
      plt.title(judul)
      plt.xlabel('Intensitas')
      plt.ylabel('Jumlah Pixel')
      plt.xlim([0, 256])
      plt.legend()
      plt.show()
```

```
[43]: plot_histogram(img_rgb, 'Histogram Gambar Asli')
      plot_histogram(merah_terdeteksi, 'Histogram Merah')
      plot_histogram(hijau_terdeteksi, 'Histogram Hijau')
      plot_histogram(biru_terdeteksi, 'Histogram Biru')
```

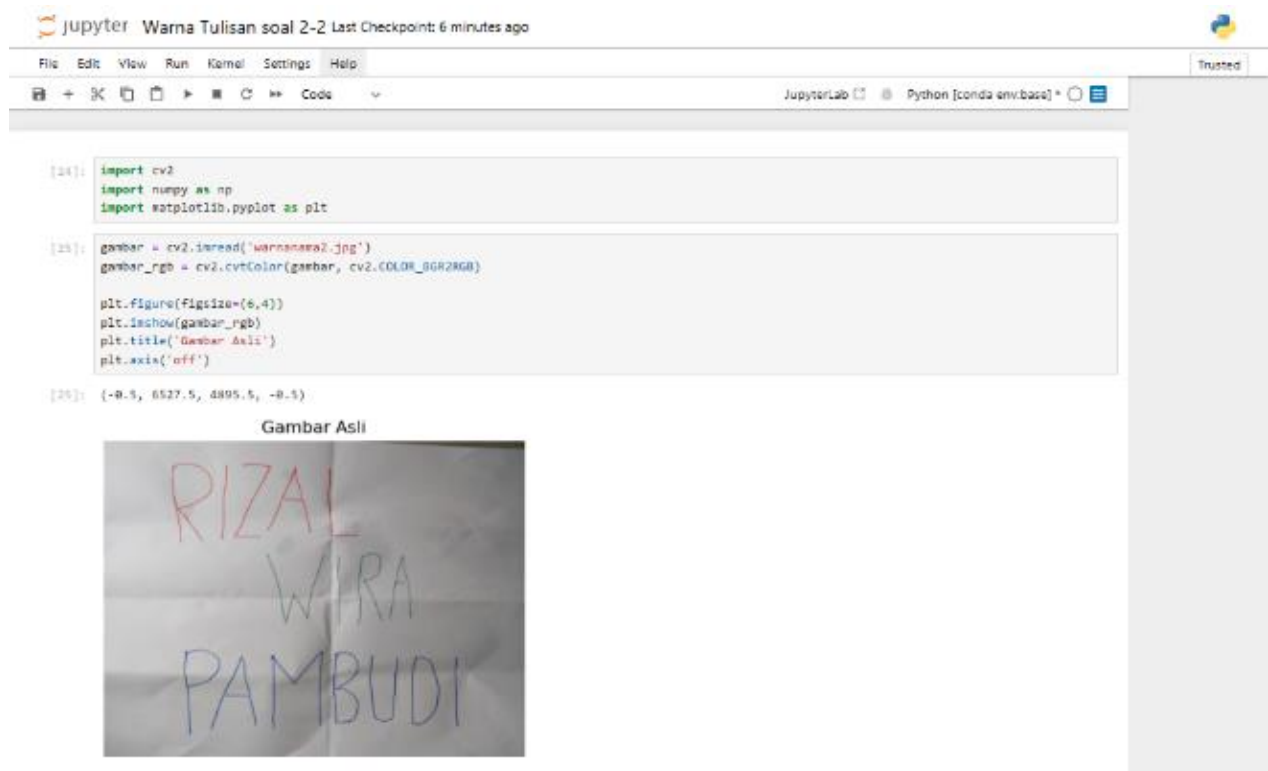


Lalu kita membuat histogram untuk mendata frekuensi dominan dari channel RGB nya



Ini adalah histogram tapi masing masing warna channel terpisah.

Soal 2

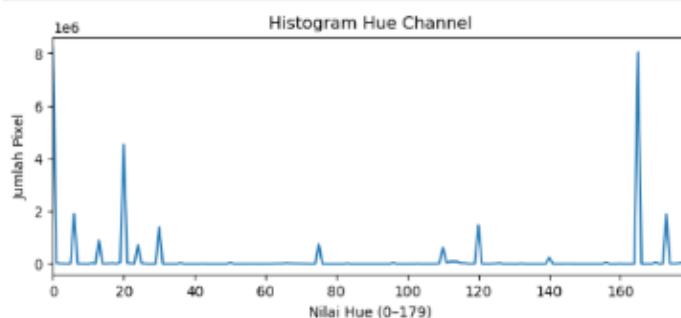


Mulai dari Gambar asli kita import dan librry yang diperlukan. Lalu menkonversi warnanya jadi channel HSV. Lalu diekstrak hue nya, Lalu kita membuat histogram hue untuk memperoleh frekuensi threshold untuk mengetahui kandidat ambang batas.

```
[26]: gambar_hsv = cv2.cvtColor(gambar, cv2.COLOR_BGR2HSV)
hue = gambar_hsv[:, :, 0]

[27]: hist_hue = cv2.calcHist([hue], [0], None, [180], [0, 180]).flatten()

plt.figure(figsize=(8,3))
plt.plot(hist_hue)
plt.title('Histogram Hue Channel')
plt.xlabel('Nilai Hue (0-179)')
plt.ylabel('Jumlah Pixel')
plt.xlim([0, 179])
plt.show()
```



```

+ [28]: lembah = []
        for i in range(1,179):
            if hist_hue[i] < hist_hue[i-1] and hist_hue[i] < hist_hue[i+1]:
                lembah.append(i)

        print("Semua kandidat valley:", lembah)
        thresholds = [20, 90, 150]
        print("Threshold terpilih:", thresholds)

Semua kandidat valley: [4, 7, 10, 12, 14, 18, 23, 27, 29, 31, 33, 37, 39, 41, 44, 47, 49, 53, 57, 59, 61, 64, 72, 74, 78, 82, 84, 86, 91,
94, 97, 100, 102, 104, 107, 111, 113, 115, 119, 121, 123, 127, 129, 131, 134, 137, 139, 143, 149, 154, 157, 159, 162, 164, 168, 171, 174,
176, 178]
Threshold terpilih: [20, 90, 150]

```

Menggunakan looping fungsi append dari hue , kita bisa set threshold ambang batas dan menemukan kandidat ambang batas yang tepat

```

+ [29]: thresh1, thresh2, thresh3 = thresholds
        s_min, v_min = 50, 20

        kategori = np.zeros_like(hue, dtype=np.uint8)

        kategori[np.logical_or(hue <= thresh1, hue >= thresh3)] = 1
        kategori[np.logical_and(hue > thresh1, hue <= thresh2)] = 2
        kategori[np.logical_and(hue > thresh2, hue < thresh3)] = 3

[30]: def mask_hsv(h_min, h_max):
        lower = np.array([h_min, s_min, v_min])
        upper = np.array([h_max, 255, 255])
        m = cv2.inRange(gambar_hsv, lower, upper)
        # buka (noise kecil hilang) lalu tutup (lubang kecil tertutup)
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))
        m = cv2.morphologyEx(m, cv2.MORPH_OPEN, kernel, iterations=1)
        m = cv2.morphologyEx(m, cv2.MORPH_CLOSE, kernel, iterations=1)
        return m

[31]: mask_orang = cv2.bitwise_or(
        mask_hsv(0, thresh1),
        mask_hsv(thresh3, 170)
    )
    mask_hijau = mask_hsv(thresh1+1, thresh2)
    mask_biru = mask_hsv(thresh2+1, thresh3-1)

```

Dari beberapa ambang batas yang ditemukan, disortirlah jadi tiga jenis threshold yang merepresentasikan masing masing channel warna RGB yaitu thresh1,thresh2,thresh3.

```
[32]: merah_gambar = cv2.bitwise_and(gambar_rgb, gambar_rgb, mask=mask_merah)
hijau_gambar = cv2.bitwise_and(gambar_rgb, gambar_rgb, mask=mask_hijau)
biru_gambar = cv2.bitwise_and(gambar_rgb, gambar_rgb, mask=mask_biru)

[47]: fig, axes = plt.subplots(2, 2, figsize=(12,6))

axes[0,0].imshow(np.zeros_like(gambar_rgb)) # NONE (hitam)
axes[0,0].set_title('NONE')

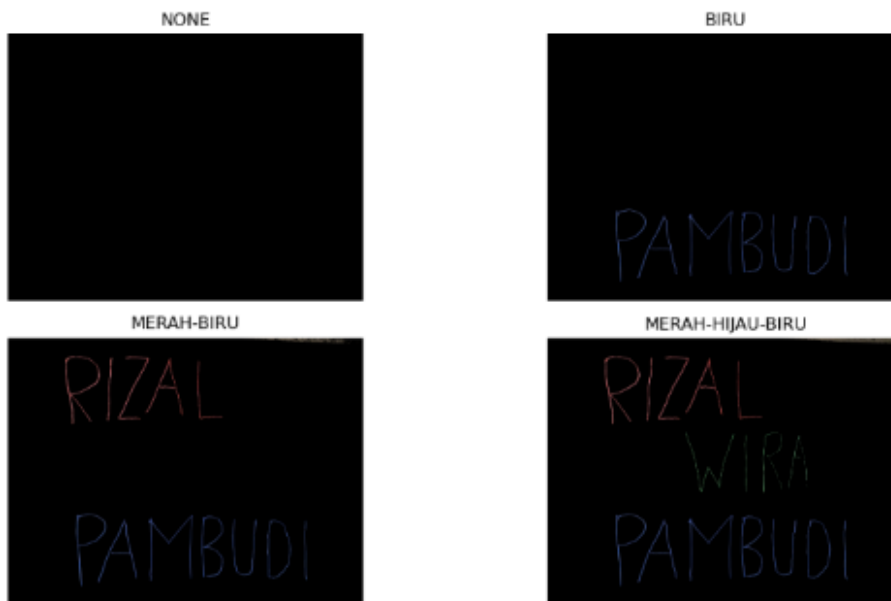
axes[0,1].imshow(biru_gambar) # BIRU
axes[0,1].set_title('BIRU')

axes[1,0].imshow(cv2.bitwise_or(merah_gambar, biru_gambar)) # MERAH-BIRU
axes[1,0].set_title('MERAH-BIRU')

axes[1,1].imshow(cv2.bitwise_or(
    merah_gambar,
    cv2.bitwise_or(hijau_gambar, biru_gambar)
))
axes[1,1].set_title('MERAH-HIJAU-BIRU')

for ax in axes.ravel():
    ax.axis('off')

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

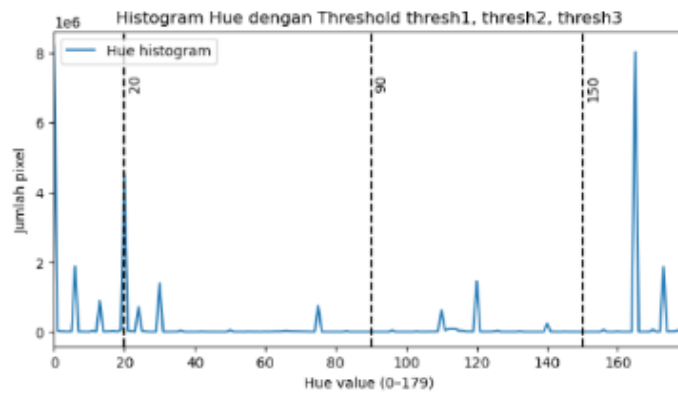


Setelah itu kita membuat masking HSV untuk masing masing threshold agar bisa dioutputkan dalam plotting seperti di gambar,

```
[54]: hue_channel = gambar_hsv[:, :, 0]
hist_hue = cv2.calcHist([hue_channel], [0], None, [180], [0, 180]).flatten()

plt.figure(figsize=(8,4))
plt.plot(hist_hue, label='Hue histogram')
for t in (thresh1, thresh2, thresh3):
    plt.axvline(t, color='k', linestyle='--')
    plt.text(t+1, max(hist_hue)*0.9, str(t), rotation=90, va='top')

plt.title('Histogram Hue dengan Threshold thresh1, thresh2, thresh3')
plt.xlabel('Hue value (0-179)')
plt.ylabel('Jumlah pixel')
plt.xlim(0,179)
plt.legend()
plt.show()
```



Disini kita memisahkan thresh 1,2,dan 3 untuk memperjelas ambang batas yang diambil yang dimana merah 20 batasnya, di hijau ada 90, dan di biru ada 150 sesuai yang ditentukan

Soal 3

jupyter

Peningkatan Citra Last Checkpoint: 4 hours ago

File

Edit

View

Run

Kernel

Settings

Help

+

JupyterLab

Python [conda env:base] *

Trusted

Rizal Wira Pambudi | 202331175

UTS PCD Kelas A

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

[3]: asli = cv2.imread('selfie1.jpg')

[5]: asli_rgb = cv2.cvtColor(asli, cv2.COLOR_BGR2RGB)

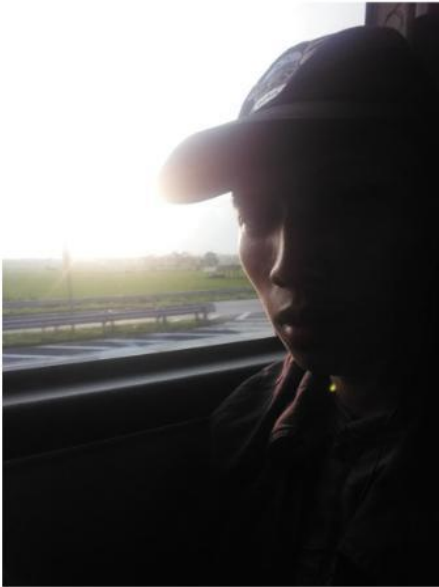
[7]: ## buat dimensi gambar
baris, kolom, _ = asli.shape

[9]: gambar_gray = cv2.cvtColor(asli, cv2.COLOR_BGR2GRAY)


[11]: plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title('Gambar Asli')
plt.imshow(asli_rgb)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('Gambar Grayscale')
plt.imshow(gambar_gray, cmap='gray')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Gambar Asli



Gambar Grayscale



Mulai dari mengimport library dan gambar, lalu meimshow kan nya untuk asli dan grayscale


```
[13]: beta = 70

[17]: gambar_gray_dicerahkan = np.zeros((baris, kolom), dtype=np.uint8)

for x in range(baris):
    for y in range(kolom):
        nilai = min(gambar_gray[x, y] + beta, 255)
        gambar_gray_dicerahkan[x, y] = nilai

[23]: fig, axs = plt.subplots(2, 2, figsize=(15, 10))

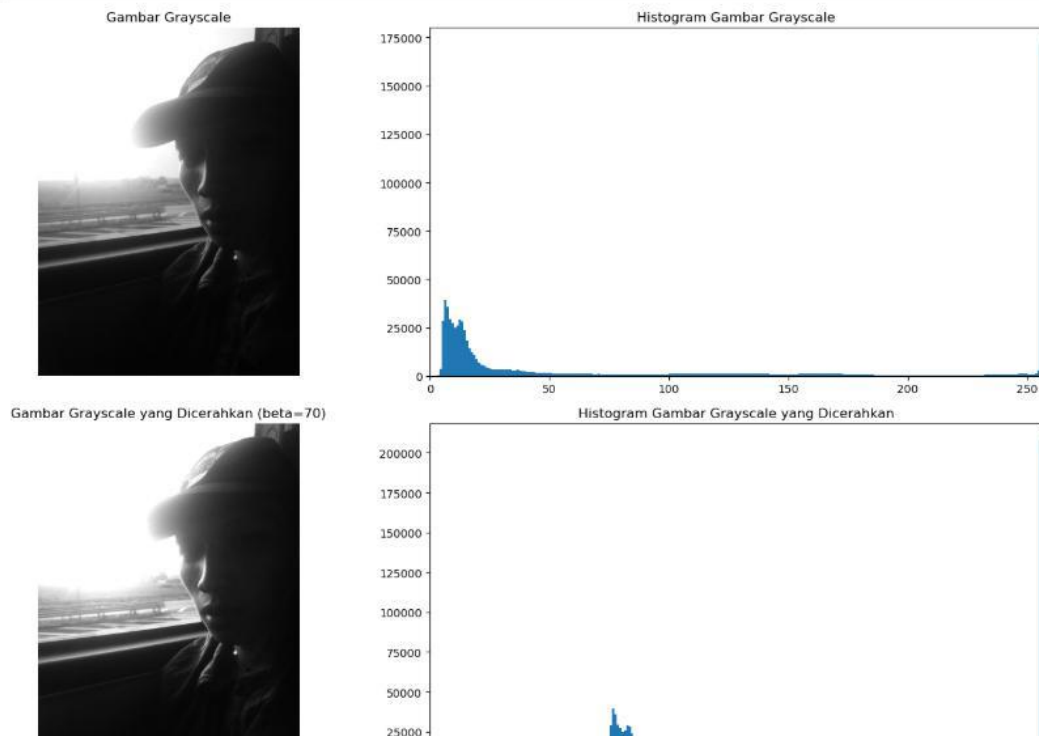
# Gambar grayscale asli
axs[0, 0].imshow(gambar_gray, cmap='gray')
axs[0, 0].set_title('Gambar Grayscale')
axs[0, 0].axis('off')

# Histogram gambar grayscale asli
axs[0, 1].hist(gambar_gray.ravel(), 256, [0, 256])
axs[0, 1].set_title('Histogram Gambar Grayscale')
axs[0, 1].set_xlim([0, 256])

# Gambar grayscale yang dicerahkan
axs[1, 0].imshow(gambar_gray_dicerahkan, cmap='gray')
axs[1, 0].set_title(f'Gambar Grayscale yang Dicerahkan (beta={beta})')
axs[1, 0].axis('off')

# Histogram gambar grayscale yang dicerahkan
axs[1, 1].hist(gambar_gray_dicerahkan.ravel(), 256, [0, 256])
axs[1, 1].set_title('Histogram Gambar Grayscale yang Dicerahkan')
axs[1, 1].set_xlim([0, 256])

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



Menetapkan beta awal pencerahan yaitu 70, lalu membuat pencerahan algoritma dari beta yang kita tentukan

```
[25]: alpha = 1.5 # set nilai alpha
gambar_gray_kontras = np.zeros((baris, kolom), dtype=np.uint8)

for x in range(baris):
    for y in range(kolom):
        nilai = min(int(gambar_gray[x, y] * alpha), 255)
        gambar_gray_kontras[x, y] = nilai

fig, axs = plt.subplots(2, 2, figsize=(15, 10))

axs[0, 0].imshow(gambar_gray, cmap='gray')
axs[0, 0].set_title('Gambar Grayscale')
axs[0, 0].axis('off')

# Histogram gambar grayscale asli
axs[0, 1].hist(gambar_gray.ravel(), 256, [0, 256])
axs[0, 1].set_title('Histogram Gambar Grayscale')
axs[0, 1].set_xlim([0, 256])

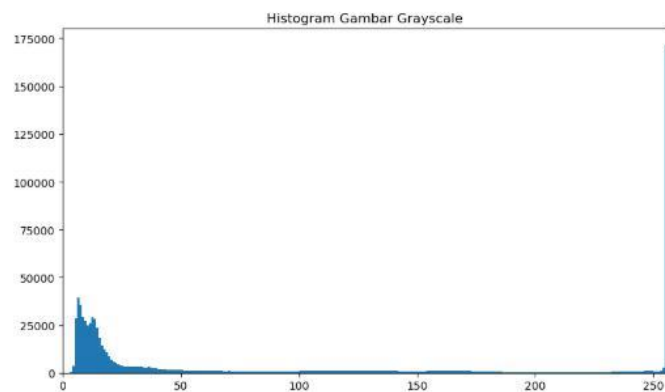
# Gambar grayscale yang dikontraskan
axs[1, 0].imshow(gambar_gray_kontras, cmap='gray')
axs[1, 0].set_title(f'Gambar Grayscale yang Dikontraskan (alpha={alpha})')
axs[1, 0].axis('off')

# Histogram gambar grayscale yang dikontraskan
axs[1, 1].hist(gambar_gray_kontras.ravel(), 256, [0, 256])
axs[1, 1].set_title('Histogram Gambar Grayscale yang Dikontraskan')
axs[1, 1].set_xlim([0, 256])

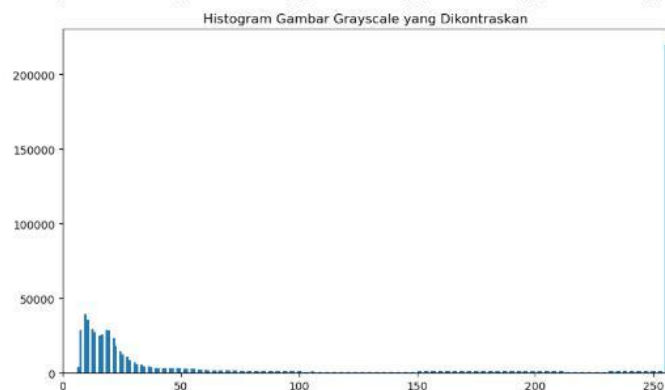
plt.tight_layout()
plt.show()
```



Gambar Grayscale



Gambar Grayscale yang Dikontraskan (alpha=1.5)



Di tahap ini kita membuat kontrasnya jadi lebih jelas dengan menentukan alpha 1.4

```
[27]: alpha = 1.2
      beta = 40 # tambahan diatas beta +70 sebelumnya

      gambar_gray_cerahkontras = np.zeros((baris, kolom), dtype=np.uint8)

      for x in range(baris):
          for y in range(kolom):
              # Kombinasi alpha dan beta, pastikan tidak melebihi 255
              nilai = min(int(gambar_gray[x, y] * alpha + beta), 255)
              gambar_gray_cerahkontras[x, y] = nilai

[29]: fig, axs = plt.subplots(2, 2, figsize=(15, 10))

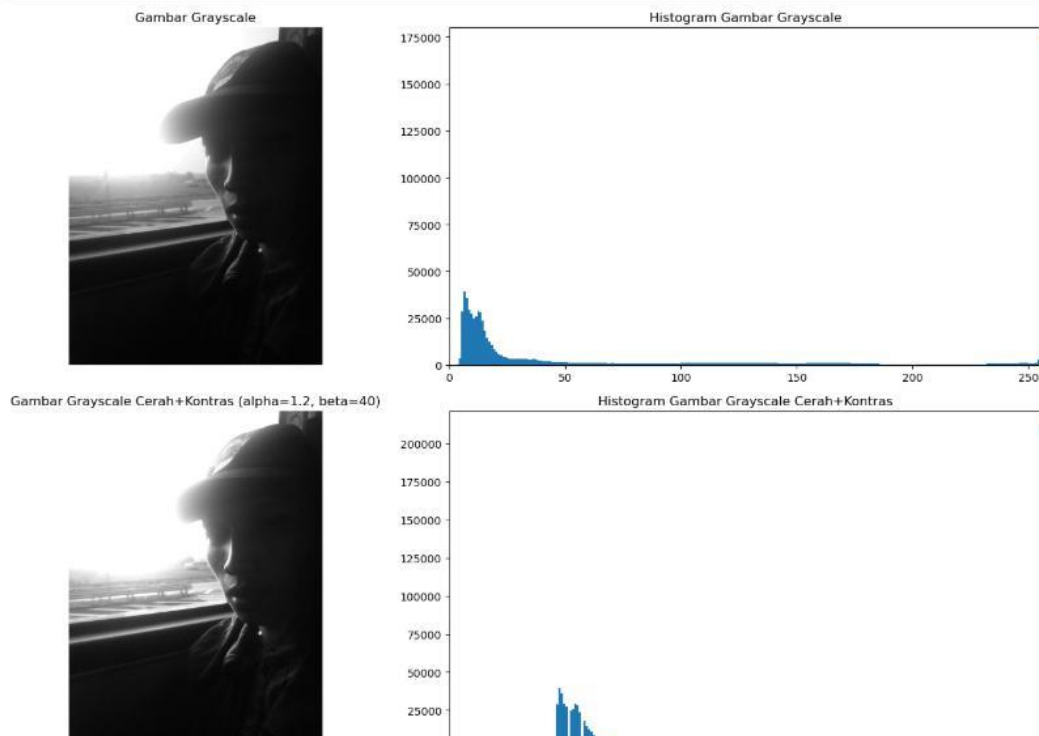
      # Gambar grayscale asli
      axs[0, 0].imshow(gambar_gray, cmap='gray')
      axs[0, 0].set_title('Gambar Grayscale')
      axs[0, 0].axis('off')

      # Histogram gambar grayscale asli
      axs[0, 1].hist(gambar_gray.ravel(), 256, [0, 256])
      axs[0, 1].set_title('Histogram Gambar Grayscale')
      axs[0, 1].set_xlim([0, 256])

      # Gambar grayscale yang dicerahkan dan dikontraskan
      axs[1, 0].imshow(gambar_gray_cerahkontras, cmap='gray')
      axs[1, 0].set_title(f'Gambar Grayscale Cerah+Kontras (alpha={alpha}, beta={beta})')
      axs[1, 0].axis('off')

      # Histogram gambar grayscale yang dicerahkan dan dikontraskan
      axs[1, 1].hist(gambar_gray_cerahkontras.ravel(), 256, [0, 256])
      axs[1, 1].set_title('Histogram Gambar Grayscale Cerah+Kontras')
      axs[1, 1].set_xlim([0, 256])

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



Menentukan abeta untuk mencerahkan dan alpha 1.2 untuk kontras, kita masukkan dalam algoritma yang menggabungkan kontras dan pencerahan agar menghasilkan hasil seperti tersebut.

```
[44]: plt.figure(figsize=(20, 15))

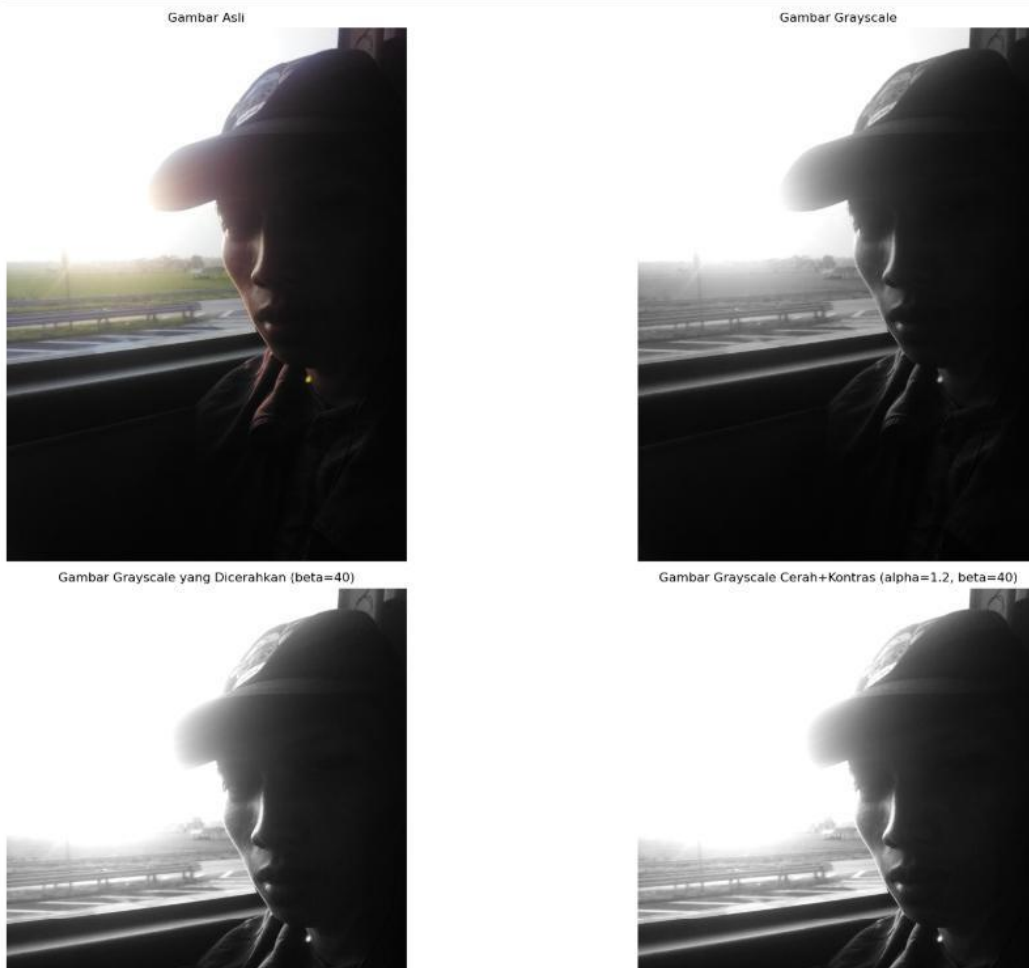
plt.subplot(2, 2, 1)
plt.title('Gambar Asli')
plt.imshow(asli_rgb)
plt.axis('off')

plt.subplot(2, 2, 2)
plt.title('Gambar Grayscale')
plt.imshow(gambar_gray, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.title(f'Gambar Grayscale yang Dicerahkan (beta={beta})')
plt.imshow(gambar_gray_dicerahkan, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.title(f'Gambar Grayscale Cerah+Kontras (alpha={alpha}, beta={beta})')
plt.imshow(gambar_gray_cerahkontras, cmap='gray')
plt.axis('off')

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



dioutputkan semua hasil dari cerah, kontras, hingga gabungan, karena tidak cukup space, maka hasil kontras dibuatkan figure baru lagi.



```
[42]: plt.figure(figsize=(20, 15))  
plt.subplot(2, 2, 1)  
plt.title(f'Gambar Grayscale Kontras')  
plt.imshow(gambar_gray_kontras, cmap='gray')  
plt.axis('off')  
  
plt.tight_layout()  
plt.show()
```



Gambar Grayscale Kontras



```
[ ]:
```


BAB IV

PENUTUP

Kesimpulan untuk landasan teori yang digunakan pada Ujian kali ini adalah bahwa pengolahan citra digital merupakan bidang yang kompleks dan multidisipliner, yang melibatkan berbagai teknik dan alat seperti OpenCV, Jupyter Notebook, dan Pyplot untuk mempermudah proses pengolahan dan visualisasi citra. Pemahaman tentang jenis-jenis citra seperti BGR, RGB, grayscale, dan binary sangat penting untuk menentukan metode pengolahan yang tepat. Teknik splitting citra memungkinkan analisis lebih mendalam pada komponen warna atau intensitas citra. Perbaikan kualitas citra melalui operasi titik, spasial, dan transformasi menjadi fondasi utama dalam meningkatkan hasil pengolahan citra. Selain itu, konsep konkurensi dan operasi konvolusi berperan penting dalam efisiensi dan efektivitas pengolahan citra, terutama dalam aplikasi real-time. Terakhir, filter batas sebagai teknik deteksi tepi membantu dalam ekstraksi fitur penting yang sangat berguna untuk berbagai aplikasi pengenalan pola dan segmentasi citra. Secara keseluruhan, landasan teori ini memberikan pemahaman yang menyeluruh dan terintegrasi untuk mendukung pelaksanaan pengolahan citra digital secara efektif dan efisien.

DAFTAR PUSTAKA

Azizah, A. Z. S., Yalis, F. N. T., Narayana, I. S., Syalom, K. A., & Rosyani, P. (2024). Pengolahan Citra Digital Dengan Penerapan Teknik Ambang Batas: Studi Kasus Menggunakan Opencv. Jurnal AI dan SPK: Jurnal Artificial Inteligent dan Sistem Penunjang Keputusan, 1(4), 283-287.

Laksono, A. T. (2022). Pengolahan Citra Digital Buah Murbei Dengan Algoritma LDA. Indonesian Journal of Engineering and Technology (INAJET), 4(2), 72-77.

Syntax Admiration. (2023). Pengolahan Citra untuk Pengenalan Wajah (Face Recognition) pada library OpenCv yang di tulis menggunakan Bahasa pemrograman Python. Jurnal Syntax Admiration.

Wardana, M. W., Maskuri, I., & Zaim, M. S. (2023). Pengolahan Citra Digital Pada Perhitungan Ikan Hias Menggunakan Metode BLOB. Jurnal Rekayasa Komputasi dan Teknologi Lunak, 6(2), 118-126.