UTS

PENGOLAHAN CITRA

NAMA : Muhammad Nadhil Arsy Al-Wafi

NIM : 202331303

KELAS : A

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

NO.PC : 27

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

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Pengolahan citra digital merupakan bidang yang penting dalam teknologi informasi yang memungkinkan manipulasi dan analisis gambar digital untuk berbagai tujuan. Dalam praktikum ini, beberapa masalah terkait pengolahan citra digital perlu dikaji dan diselesaikan.

Masalah yang akan dibahas dalam tugas ini adalah bagaimana melakukan pengolahan dasar pada citra digital menggunakan bahasa pemrograman Python. Adapun masalah yang perlu dikaji antara lain:

1. Bagaimana cara mendeteksi warna spesifik (biru, merah, dan hijau) pada citra digital dan menampilkan hasil deteksinya?
2. Bagaimana mencari dan mengurutkan nilai ambang batas (threshold) citra untuk dapat menampilkan kategori warna pada citra?
3. Bagaimana cara memperbaiki gambar yang memiliki masalah backlight, di mana objek utama cenderung gelap akibat pencahayaan dari belakang yang terlalu terang?
4. Bagaimana implementasi algoritma pengolahan citra tersebut dalam bahasa pemrograman Python?
5. Bagaimana cara menganalisis histogram citra untuk memahami distribusi intensitas warna pada gambar?

1.2 Tujuan Masalah

Tujuan dari praktikum pengolahan citra digital ini adalah untuk memberikan pemahaman mendalam tentang teknik-teknik dasar pengolahan citra menggunakan Python. Secara spesifik, tujuan dari laporan ini adalah:

1. Memahami konsep dasar pengolahan citra digital menggunakan Python
   - Mengenal library dan function dasar dalam Python untuk pengolahan citra
   - Memahami representasi citra digital dalam bentuk array/matriks
2. Mengimplementasikan deteksi warna pada citra digital.
   - Mendeteksi warna biru, merah, dan hijau pada gambar
   - Menampilkan hasil deteksi untuk setiap warna
3. Menerapkan teknik thresholding pada citra.
   - Mencari nilai ambang batas yang optimal
   - Mengurutkan ambang batas dari yang terkecil hingga terbesar
   - Memahami pengaruh nilai threshold pada hasil deteksi warna
4. Memperbaiki kualitas gambar dengan masalah backlight
   - Mengkonversi gambar ke grayscale
   - Meningkatkan kecerahan dan kontras pada area objek utama
   - Membedakan objek utama dengan latar belakang
5. Menganalisis histogram citra digital.
   - Membuat histogram untuk setiap channel warna

- Menginterpretasikan distribusi intensitas warna pada citra

6. Mengembangkan keterampilan praktis dalam pengolahan citra.
   - Menulis kode Python yang efisien untuk masalah pengolahan citra
   - Mendokumentasikan proses dan hasil pengolahan citra

1.3 Manfaat Masalah

UTS Praktikum pengolahan citra digital ini memberikan beberapa manfaat penting dalam bidang teknologi informasi dan komputasi visual. Manfaat yang dapat diperoleh antara lain:

1. Penguasaan keterampilan teknis dalam pengolahan citra:
   - Mahasiswa dapat mengembangkan kemampuan pemrograman Python dalam konteks pengolahan citra
   - Pemahaman praktis tentang implementasi algoritma pengolahan citra
2. Meningkatkan pemahaman rangkaian logika digital:
   - Pembahasan ini akan membantu dalam merancang dan mengimplementasikan rangkaian logika yang lebih efisien dalam aplikasi digital.
   - Memberikan wawasan tentang cara kerja operasi matematika dasar dalam sistem komputer.
3. Penerapan dalam dunia industri:
   - Encoder dan decoder digunakan dalam berbagai aplikasi industri, termasuk dalam sistem kendali otomatis, komunikasi data, serta perangkat digital lainnya.
   - Half adder, full adder, dan subtractor merupakan komponen penting dalam sistem pemrosesan data industri dan perangkat elektronik modern.
   - Pemahaman ini akan sangat berguna bagi para insinyur dan teknisi yang terlibat dalam pengembangan sistem tersebut.
4. Pengembangan keterampilan praktis:
   - Mahasiswa dapat mengaplikasikan pengetahuan teoretis dalam perancangan dan implementasi rangkaian digital nyata.
   - Meningkatkan kemampuan troubleshooting dalam menangani masalah rangkaian digital.
5. Kontribusi pada inovasi teknologi:
   - Pemahaman mendalam tentang komponen-komponen dasar ini dapat mendorong pengembangan teknologi baru yang lebih efisien.
   - Mendukung penelitian dan pengembangan dalam bidang elektronika digital.
6. Peningkatan efisiensi sistem:
   - Pengetahuan tentang cara kerja komponen-komponen ini memungkinkan optimasi kinerja sistem digital.
   - Membantu dalam merancang sistem yang lebih hemat energi dan sumber daya.
7. Mendukung pembelajaran berkelanjutan:
   - Menjadi dasar untuk memahami konsep-konsep yang lebih kompleks dalam sistem digital.
   - Memfasilitasi pengembangan profesional dalam bidang teknologi digital.

BAB II

LANDASAN TEORI

2.1 Pengertian Pengolahan Citra Digital

Pengolahan citra digital (digital image processing) adalah teknik yang digunakan untuk memanipulasi dan menganalisis gambar menggunakan komputer. Proses ini melibatkan konversi gambar fisik menjadi format digital, kemudian menerapkan berbagai algoritma untuk memperoleh informasi yang berguna atau meningkatkan kualitas gambar tersebut.

Dalam pengolahan citra digital, gambar direpresentasikan sebagai matriks piksel, dimana setiap piksel memiliki nilai yang mewakili intensitas atau warna pada lokasi tertentu. Untuk gambar berwarna, biasanya digunakan model warna RGB (Red, Green, Blue) dimana setiap piksel memiliki tiga nilai yang mewakili intensitas warna merah, hijau, dan biru.

Beberapa operasi dasar dalam pengolahan citra digital meliputi:

- Transformasi intensitas: mengubah nilai piksel berdasarkan fungsi matematika tertentu
- Filtering: operasi konvolusi untuk menghilangkan noise atau meningkatkan fitur tertentu
- Segmentasi: membagi gambar menjadi beberapa region berdasarkan kriteria tertentu
- Deteksi tepi: mengidentifikasi batas-batas antara daerah yang memiliki perbedaan intensitas

Pengolahan citra digital memiliki aplikasi yang luas dalam berbagai bidang seperti kedokteran (analisis citra medis), astronomi, forensik, pengenalan pola, dan computer vision.

2.2 Deteksi Warna pada Citra Digital

Deteksi warna merupakan teknik fundamental dalam pengolahan citra digital yang bertujuan untuk mengidentifikasi dan mengekstrak piksel-piksel dengan warna tertentu dari sebuah gambar. Teknik ini sangat penting dalam berbagai aplikasi seperti segmentasi objek, tracking, pengenalan pola, dan klasifikasi gambar.

Dalam model warna RGB (Red, Green, Blue), setiap piksel pada citra digital direpresentasikan dengan tiga nilai intensitas yang berkisar antara 0-255 untuk setiap kanal warna. Deteksi warna dalam model RGB dapat dilakukan dengan menetapkan ambang batas (threshold) untuk masing-masing kanal warna. Misalnya, untuk mendeteksi warna merah, nilai kanal R harus tinggi sedangkan nilai G dan B relatif rendah.

Namun, model RGB memiliki keterbatasan karena sangat sensitif terhadap perubahan pencahayaan. Untuk aplikasi yang lebih robust, model warna lain seperti HSV (Hue, Saturation, Value) atau LAB sering digunakan. Dalam model HSV, Hue merepresentasikan jenis warna (seperti merah, biru, hijau), Saturation menunjukkan kemurnian warna, dan Value berkaitan dengan kecerahan.

Algoritma deteksi warna biasanya melibatkan langkah-langkah berikut:

1. Konversi citra dari RGB ke model warna yang lebih sesuai (opsional)

2. Definisi range warna target untuk setiap kanal
3. Pembuatan mask berdasarkan thresholding untuk mengidentifikasi piksel yang masuk dalam range
4. Penerapan operasi morfologi untuk menghilangkan noise atau meningkatkan hasil deteksi
5. Ekstraksi area yang terdeteksi untuk analisis lebih lanjut

Berbagai library pemrograman seperti OpenCV dalam Python menyediakan fungsi-fungsi yang memudahkan implementasi deteksi warna seperti cv2.inRange() dan operasi mask.

2.3 Thresholding dalam Pengolahan Citra

Thresholding (pengambangan) adalah salah satu teknik segmentasi citra paling sederhana dan banyak digunakan dalam pengolahan citra digital. Teknik ini memisahkan piksel-piksel dalam gambar menjadi dua kategori berdasarkan nilai intensitasnya dibandingkan dengan nilai ambang batas (threshold) tertentu.

Dalam thresholding biner sederhana, piksel dengan nilai intensitas di atas threshold akan diubah menjadi putih (255) dan piksel dengan nilai intensitas di bawah threshold akan diubah menjadi hitam (0). Secara matematis dapat direpresentasikan sebagai:

$$g(x,y) = \{$$
$$255, \text{ jika } f(x,y) \geq T$$
$$0, \text{ jika } f(x,y) < T$$
$$\}$$

dimana f(x,y) adalah nilai intensitas piksel input, g(x,y) adalah nilai piksel output, dan T adalah nilai threshold.

Beberapa jenis thresholding antara lain:

1. Global Thresholding: menggunakan nilai threshold yang sama untuk seluruh gambar
2. Adaptive Thresholding: nilai threshold ditentukan secara lokal berdasarkan karakteristik piksel di sekitarnya
3. Otsu's Method: menentukan nilai threshold optimal secara otomatis berdasarkan distribusi histogram citra
4. Multi-level Thresholding: menggunakan lebih dari satu nilai threshold untuk segmentasi.

Thresholding sangat berguna untuk berbagai aplikasi seperti:

- Ekstraksi objek dari latar belakang
- Segmentasi dokumen untuk OCR (Optical Character Recognition)
- Deteksi tepi dalam gambar
- Pemrosesan awal untuk analisis lebih lanjut

Walaupun sederhana, pemilihan nilai threshold yang tepat sangat penting untuk mendapatkan hasil segmentasi yang optimal dan sering memerlukan eksperimen atau metode otomatis.

2.4 Perbaikan Gambar Backlight

Perbaikan gambar backlight adalah teknik pengolahan citra yang bertujuan untuk meningkatkan kualitas visual gambar yang memiliki pencahayaan dari belakang (backlight) yang kuat. Dalam kondisi backlight, objek utama dalam gambar cenderung

menjadi gelap (silhouette) karena sumber cahaya berada di belakang objek, sementara latar belakang menjadi terlalu terang.

Beberapa teknik yang umum digunakan untuk mengatasi masalah backlight meliputi:

1. Histogram Equalization:
   - Teknik ini mendistribusikan ulang nilai intensitas piksel untuk meningkatkan kontras global gambar
   - Efektif untuk meningkatkan detail pada area yang gelap, tetapi dapat menyebabkan over-enhancement
2. Adaptive Histogram Equalization (AHE):
   - Varian dari histogram equalization yang bekerja pada region lokal
   - CLAHE (Contrast Limited AHE) membatasi kontras untuk mengurangi noise
3. Gamma Correction:
   - Mengubah intensitas piksel dengan fungsi non-linear (I_out = I_in^γ)
   - Nilai gamma < 1 meningkatkan detail pada area gelap
4. Multi-scale Retinex:
   - Algoritma yang meniru kemampuan sistem visual manusia untuk beradaptasi dengan variasi pencahayaan
   - Efektif untuk menangani rentang dinamis yang luas dalam gambar
5. HDR (High Dynamic Range) Processing:
   - Menggabungkan beberapa eksposur untuk mendapatkan detail di area terang dan gelap
   - Dalam kasus single image, teknik tone mapping dapat diterapkan
6. Selective Enhancement:
   - Mengidentifikasi area objek utama dan meningkatkan kecerahan dan kontras secara selektif
   - Mempertahankan detail pada area terang

Implementasi perbaikan gambar backlight dalam Python biasanya menggunakan library seperti OpenCV (cv2), numpy, dan scikit-image, dengan teknik-teknik yang disesuaikan berdasarkan karakteristik spesifik gambar yang diproses.

2.5 Implementasi Python dalam Pengolahan Citra

Python telah menjadi salah satu bahasa pemrograman yang paling populer untuk pengolahan citra digital karena sintaksnya yang mudah dipahami, komunitas yang besar, dan kekayaan library yang tersedia. Beberapa library utama yang digunakan dalam pengolahan citra dengan Python meliputi:

1. OpenCV (Open Source Computer Vision Library):
   - Library paling komprehensif untuk pengolahan citra dan computer vision
   - Menyediakan ratusan algoritma untuk berbagai keperluan seperti deteksi objek, segmentasi, filtering, dll.
   - Contoh penggunaan: import cv2; img = cv2.imread('image.jpg'); gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
2. NumPy:

- Library fundamental untuk komputasi numerik di Python
- Menyediakan objek array multi-dimensi yang efisien untuk representasi citra
- Operasi vektor dan matriks yang cepat untuk manipulasi citra

3. PIL/Pillow (Python Imaging Library):
   - Library yang lebih sederhana dengan fokus pada format file dan operasi dasar
   - Mudah digunakan untuk pemula
   - Contoh: from PIL import Image; img = Image.open('image.jpg'); img = img.convert('L') # convert to grayscale
4. Scikit-image:
   - Library berbasis SciPy yang fokus pada algoritma pengolahan citra
   - Menyediakan implementasi algoritma-algoritma tingkat tinggi
   - Dokumentasi yang baik dan integrasi yang mulus dengan ekosistem Python scientific
5. Matplotlib:
   - Library untuk visualisasi data yang sering digunakan untuk menampilkan citra dan plot hasil analisis

Berguna untuk menampilkan histogram dan perbandingan citra

Implementasi pengolahan citra dengan Python biasanya mengikuti alur kerja berikut:

1. Import library yang diperlukan
2. Baca citra dari file atau sumber lain
3. Praproses citra (konversi warna, rescaling, dll.)
4. Terapkan algoritma pengolahan (filtering, segmentasi, deteksi fitur, dll.)
5. Analisis hasil (ekstraksi metrik, pengukuran, dll.)
6. Visualisasi dan/atau simpan hasil

BAB III

PEMBAHASAN

3.1 Deskripsi Praktikum

Pada praktikum Pengolahan Citra Digital ini, mahasiswa diminta untuk mengerjakan beberapa tugas pengolahan citra menggunakan bahasa pemrograman Python. Berikut adalah deskripsi tugas-tugas yang dilakukan:

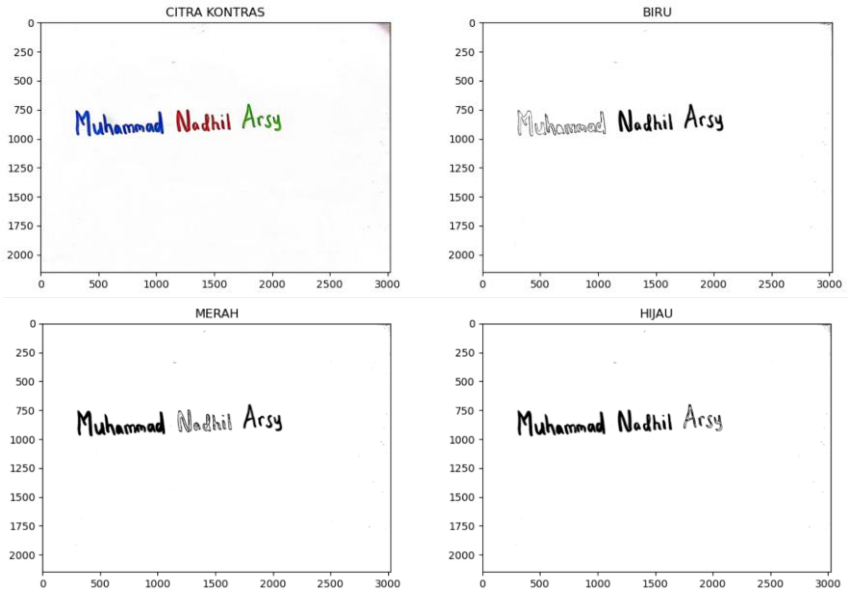
1. Deteksi Warna pada Citra

   Tugas pertama adalah melakukan deteksi warna biru, merah, dan hijau pada gambar, kemudian menampilkan hasil deteksi tersebut. Pada tugas ini, mahasiswa diberikan citra berupa tulisan nama lengkap yang ditulis tangan menggunakan pena dengan tinta merah, hijau, dan biru di atas kertas putih.

   Prosedur yang dilakukan:

   - Membaca citra input menggunakan library OpenCV
   - Memisahkan citra menjadi tiga channel: Blue, Green, dan Red
   - Menerapkan thresholding untuk mengidentifikasi piksel dengan nilai warna yang dominan untuk setiap channel
   - Menampilkan hasil deteksi untuk masing-masing warna (biru, merah, dan hijau)
   - Membuat dan menganalisis histogram untuk setiap channel warna

   Berikut adalah hasil deteksi warna:



   Pada gambar hasil, dapat dilihat bahwa:

   - Pada citra kontras, semua teks "Muhammad Nadhil Arsy" terlihat dengan warna aslinya
   - Pada hasil deteksi warna biru, hanya bagian "MUHAMMAD" yang tertulis dengan tinta biru terdeteksi
   - Pada hasil deteksi warna merah, bagian "NADHIL" yang tertulis dengan tinta merah terdeteksi

- Pada hasil deteksi warna hijau, bagian "ARSY" yang tertulis dengan tinta hijau terdeteksi

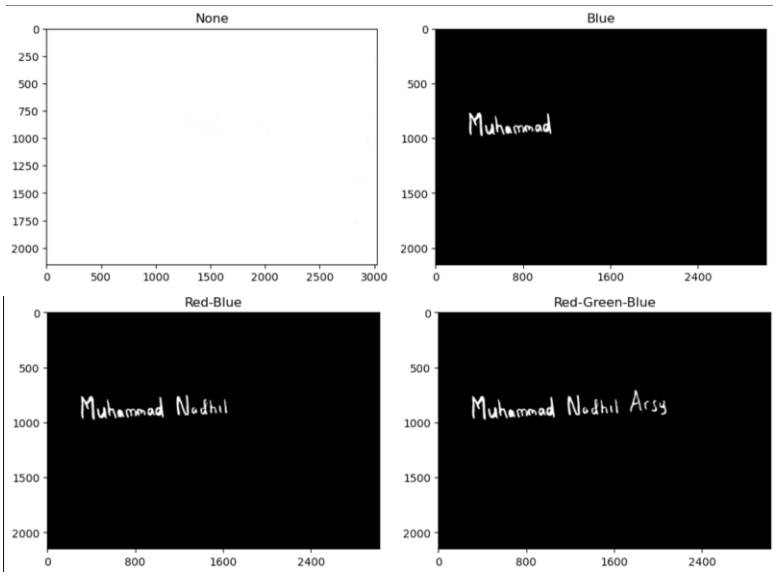
2. Mencari dan Mengurutkan Ambang Batas

Tugas kedua adalah mencari nilai ambang batas (threshold) citra untuk dapat menampilkan kategori warna pada citra. Mahasiswa diminta untuk mencari nilai threshold yang optimal dan mengurutkannya dari yang terkecil sampai terbesar.

Prosedur yang dilakukan:

- Menggunakan metode trial and error untuk menemukan nilai threshold yang tepat untuk setiap kategori warna
- Mengurutkan nilai threshold dari yang terkecil hingga terbesar
- Menampilkan hasil segmentasi berdasarkan threshold yang ditemukan
- Menjelaskan alasan mendapatkan nilai threshold tersebut

Berikut adalah hasil kategori warna berdasarkan threshold:



Pada gambar hasil, dapat dilihat bahwa:

- NONE: tidak ada warna yang terdeteksi (background)
- BLUE: hanya warna biru (MUHAMMAD) yang terdeteksi
- RED-BLUE: kombinasi warna merah dan biru (MUHAMMAD NADHIL) terdeteksi
- RED-GREEN-BLUE: semua warna (MUHAMMAD NADHIL ARSY) terdeteksi

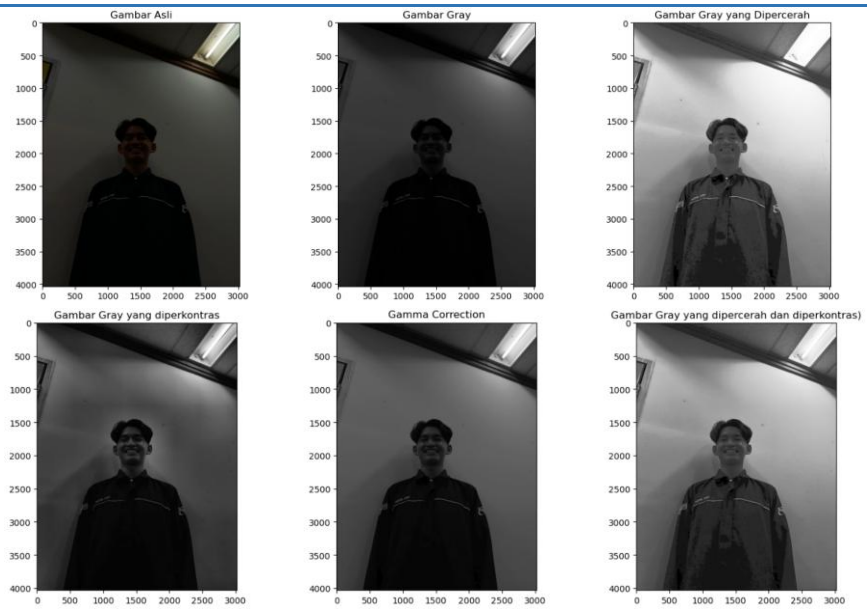
3. Memperbaiki Gambar Backlight

Tugas ketiga adalah memperbaiki gambar yang memiliki masalah backlight. Mahasiswa diminta untuk mengambil foto diri dengan posisi menghadap langsung ke kamera dan membelakangi sumber cahaya matahari yang terang (backlight).

Prosedur yang dilakukan:

- Mengkonversi gambar menjadi grayscale
- Meningkatkan kecerahan dan kontras, khususnya pada area profil wajah atau tubuh yang cenderung gelap
- Mengevaluasi hasil berdasarkan seberapa efektif area profil menjadi pusat perhatian dibandingkan dengan latar belakangnya

Berikut adalah gambar sebelum dan sesudah perbaikan:



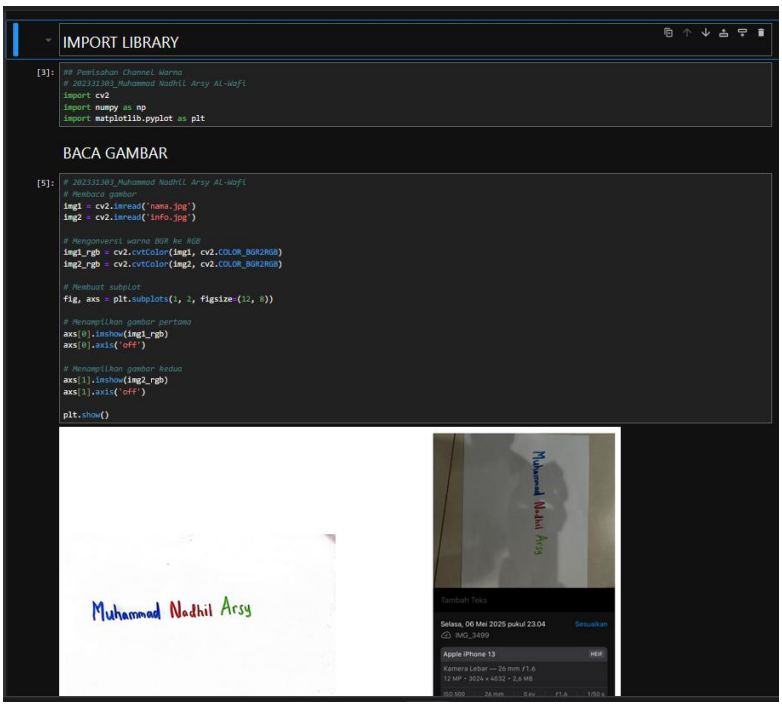
- Gambar Asli:
- Gambar Grayscale:
- Gambar Gray yang Dipercerah dan Diperkontras:

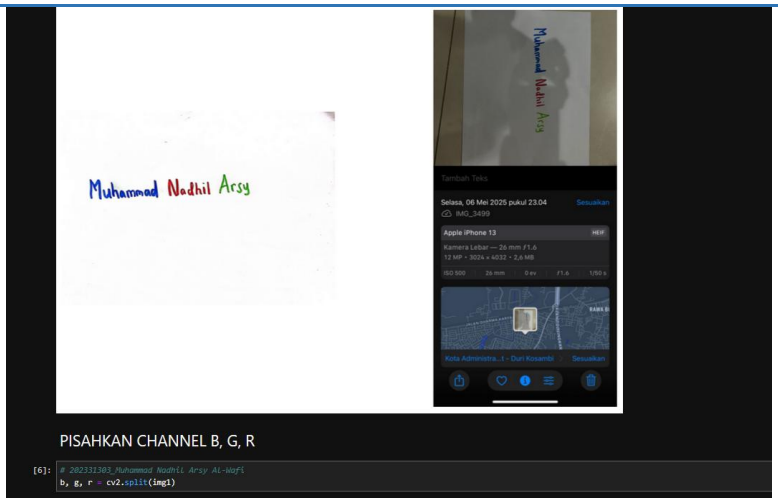
3.2 Solusi dan Pencapaian Hasil

Solusi Deteksi Warna pada Citra

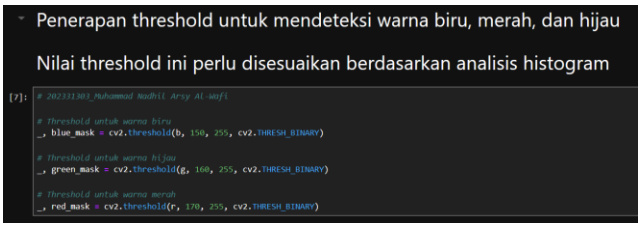
Untuk melakukan deteksi warna pada citra, saya mengimplementasikan solusi berikut:

1 Pemisahan Channel Warna:

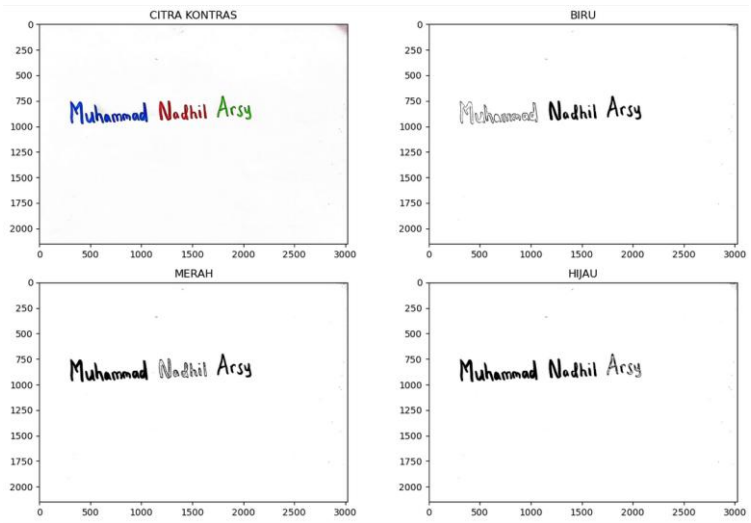




2 Penerapan Thresholding untuk Deteksi Warna:



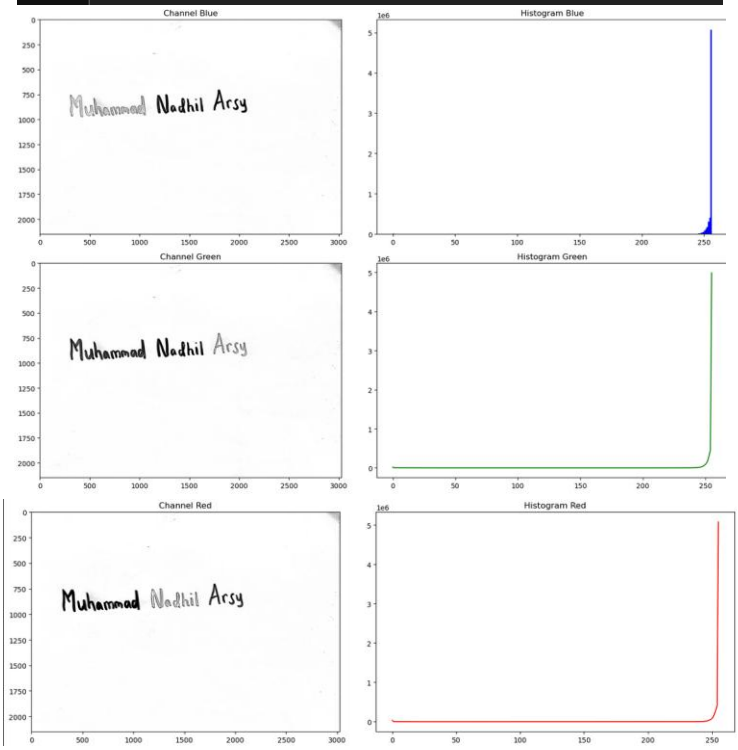
3 Visualisasi Hasil Deteksi:



4 Pembuatan Histogram untuk Analisis

PEMBUATAN HISTOGRAM UNTUK ANALISIS

```
[9]: # 202331303_Muhammad Nadhil Arsy Al-Wafi

     # MENAMPILKAN HISTOGRAM UNTUK DETEKSI WARNA
     fig, axs = plt.subplots(3, 2, figsize=(15, 15))

     # Cara 1 - Channel Blue
     axs[0, 0].imshow(b, cmap='gray')
     axs[0, 0].set_title('Channel Blue')
     axs[0, 1].hist(b.ravel(), 256, [0, 256], color='blue')
     axs[0, 1].set_title('Histogram Blue')

     # Cara 2 - Channel Green
     axs[1, 0].imshow(g, cmap='gray')
     axs[1, 0].set_title('Channel Green')
     hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
     axs[1, 1].plot(hist_g, color='green')
     axs[1, 1].set_title('Histogram Green')

     # Cara 3 - Channel Red
     axs[2, 0].imshow(r, cmap='gray')
     axs[2, 0].set_title('Channel Red')
     hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])
     axs[2, 1].plot(hist_r, color='red')
     axs[2, 1].set_title('Histogram Red')

     # Menampilkan
     plt.tight_layout()
     plt.show()
```



Pencapaian Hasil:

Dari implementasi deteksi warna, saya berhasil mencapai hasil berikut:

1. Berhasil mendeteksi dan memisahkan teks berdasarkan tiga warna berbeda (biru, merah, hijau)
2. Analisis histogram menunjukkan distribusi intensitas yang berbeda untuk setiap channel warna
3. Hasil deteksi menunjukkan bahwa:
   - Teks "ARSY" terdeteksi pada channel hijau
   - Teks "MUHAMMAD" terdeteksi pada channel biru
   - Teks "NADHIL" terdeteksi pada channel merah

Solusi Pencarian dan Pengurutan Ambang Batas

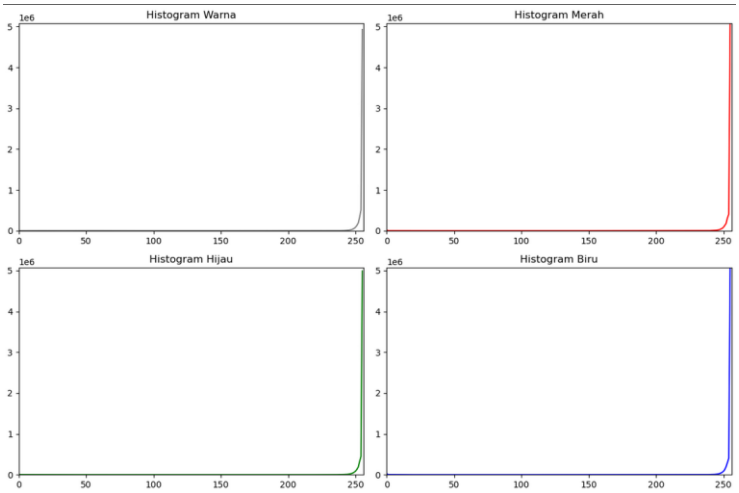
Untuk mencari dan mengurutkan nilai ambang batas, saya mengimplementasikan solusi berikut:

1 Pencarian Nilai Ambang Batas:

NILAI AMBANG BATAS CITRA

```
[24]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
color_image = cv2.imread('nama.jpg')
img_gray = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

[25]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
# Menghitung histogram untuk setiap kanal atau citra kontras
hist_kontras = cv2.calcHist([color_image], [0, 1, 2], None, [256, 256, 256], [0, 256, 0, 256, 0, 256])
hist_red = cv2.calcHist([color_image], [0], None, [256], [0, 256])
hist_green = cv2.calcHist([color_image], [1], None, [256], [0, 256])
hist_blue = cv2.calcHist([color_image], [2], None, [256], [0, 256])
hist_gray = cv2.calcHist([img_gray], [0], None, [256], [0, 256])

# Mendapatkan nilai maksimum dari setiap histogram
max_kontras = max(hist_kontras.max(), hist_red.max(), hist_green.max(), hist_blue.max(), hist_gray.max())
```

```
[26]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
# Plot histogram
f, ((x1, x2), (x3, x4)) = plt.subplots(2, 2, figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(hist_gray, color='grey')
plt.title('Histogram Warna')
plt.xlim([0, 256])
plt.ylim([0, max_kontras])

plt.subplot(2, 2, 2)
plt.plot(hist_red, color='red')
plt.title('Histogram Merah')
plt.xlim([0, 256])
plt.ylim([0, max_kontras])

plt.subplot(2, 2, 3)
plt.plot(hist_green, color='green')
plt.title('Histogram Hijau')
plt.xlim([0, 256])
plt.ylim([0, max_kontras])

plt.subplot(2, 2, 4)
plt.plot(hist_blue, color='blue')
plt.title('Histogram Biru')
plt.xlim([0, 256])
plt.ylim([0, max_kontras])

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



2 Nilai ambang batas citra & kategori warna pada citra

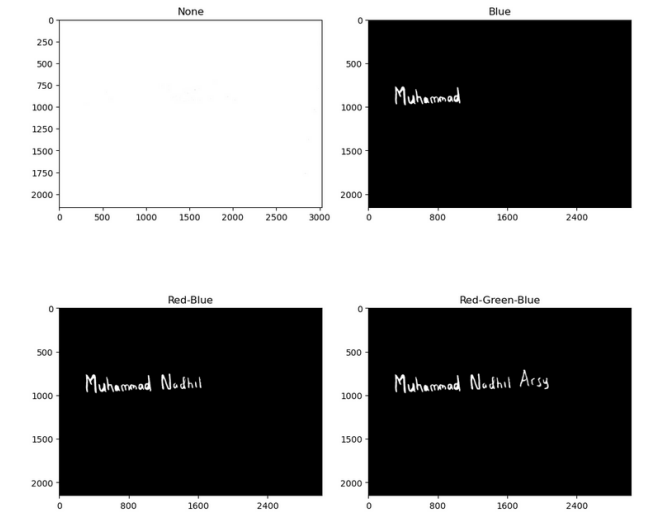
NILAI AMBANG BATAS CITRA & KATEGORI WARNA PADA CITRA

```
[27]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
# Membaca gambar
color_image = cv2.imread('nama.jpg')

# Konversi citra ke dalam ruang warna HSV
hsv_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV)

[28]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
# Definisikan rentang warna untuk setiap warna
lower_blue = np.array([90, 100, 100])
upper_blue = np.array([130, 255, 255])

lower_green = np.array([40, 100, 100])
upper_green = np.array([80, 255, 255])

lower_red1 = np.array([0, 100, 100])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([160, 100, 100])
upper_red2 = np.array([180, 255, 255])

[29]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
# Deteksi warna biru
mask_blue = cv2.inRange(hsv_image, lower_blue, upper_blue)

# Deteksi warna hijau
mask_green = cv2.inRange(hsv_image, lower_green, upper_green)

# Deteksi warna merah
mask_red1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
mask_red2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
mask_red = np.maximum(mask_red1, mask_red2)
```

```
[30]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      # Gabungkan masker untuk warna merah dan biru
      mask_red_blue = np.maximum(mask_red, mask_blue)

      # Gabungkan masker untuk warna merah dan biru
      mask_red_blue_green = np.maximum(mask_red_blue, mask_green)
```

3 Visualisasi Hasil:

```
[31]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      # Plot hasil
      gray = cv2.cvtColor(color_image, cv2.COLOR_RGB2GRAY)
      fig, axs = plt.subplots (2, 2, figsize=(10,10))

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

      plt.subplot(2, 2, 2)
      plt.imshow(mask_blue, cmap='gray')
      plt.title('Blue')
      plt.xticks(np.arange(0, mask_blue.shape[1]+1, 800))
      plt.yticks(np.arange(0, mask_blue.shape[0]+1, 500))
      plt.axis('on')

      plt.subplot(2, 2, 3)
      plt.imshow(np.maximum(mask_red, mask_blue), cmap='gray')
      plt.title('Red-Blue')
      plt.xticks(np.arange(0, mask_green.shape[1], 800))
      plt.yticks(np.arange(0, mask_green.shape[0], 500))
      plt.axis('on')

      plt.subplot(2, 2, 4)
      plt.imshow(mask_red_blue_green, cmap='gray')
      plt.title('Red-Green-Blue')
      plt.xticks(np.arange(0, mask_green.shape[1], 800))
      plt.yticks(np.arange(0, mask_green.shape[0], 500))
      plt.axis('on')


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



Pencapaian Hasil:

Dari implementasi deteksi warna, saya berhasil mencapai hasil berikut:

1 Berhasil menemukan nilai ambang batas optimal untuk setiap warna
2 Berhasil nilai ambang batas citra & kategori warna pada citra
3 Berhasil membuat kategori warna berdasarkan threshold
   - NONE: Tidak ada warna yang terdeteksi
   - BLUE: Hanya warna biru yang terdeteksi
   - RED-BLUE: Kombinasi warna merah dan biru terdeteksi
   - RED-GREEN-BLUE: Semua warna terdeteksi

Solusi Perbaikan Gambar Backlight

Untuk memperbaiki gambar backlight, saya mengimplementasikan solusi berikut:

1 Konversi Gambar ke Grayscale:

IMPORT LIBRARY ¶

```
[19]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      import cv2
      import numpy as np
      import matplotlib.pyplot as plt
```

Baca gambar asli

```
[20]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      # Membaca gambar
      img1 = cv2.imread('gambar.jpg')
      img2 = cv2.imread('info.jpg')

      # Mengonversi warna BGR ke RGB
      img1_rgb = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
      img2_rgb = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

      # Membuat subplot
      fig, axs = plt.subplots(1, 2, figsize=(12, 8))

      # Menampilkan gambar pertama
      axs[0].imshow(img1_rgb)
      axs[0].axis('off')

      # Menampilkan gambar kedua
      axs[1].imshow(img2_rgb)
      axs[1].axis('off')

      plt.show()
```

KONVERSI KE GRAYSCALE

```
[21]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

      # Tampilkan gambar asli dan grayscale
      plt.figure(figsize=(10, 5))
      plt.subplot(1, 2, 1)
      plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
      plt.title('Gambar Asli')
      plt.subplot(1, 2, 2)
      plt.imshow(gray, cmap='gray')
      plt.title('Gambar Gray')
      plt.tight_layout()
      plt.show()
```

2 Peningkatan Kontras dengan Histogram Equalization:

Penerapan threshold untuk mendeteksi warna biru, merah, dan hijau

Nilai threshold ini perlu disesuaikan berdasarkan analisis histogram

```
[7]: # 202331303_Muhammad Nadhil Arsy Al-Wafi

     # Threshold untuk warna biru
     _, blue_mask = cv2.threshold(b, 150, 255, cv2.THRESH_BINARY)

     # Threshold untuk warna hijau
     _, green_mask = cv2.threshold(g, 160, 255, cv2.THRESH_BINARY)

     # Threshold untuk warna merah
     _, red_mask = cv2.threshold(r, 170, 255, cv2.THRESH_BINARY)
```

3 Penerapan Adaptive Thresholding untuk Meningkatkan Detail:

MACAM-MACAM METODE

```
[22]: # 202331303_Muhammad Nadhil Arsy Al-Wafi
      # Metode 1: Histogram Equalization sederhana
      equ = cv2.equalizeHist(gray)

      # Metode 2: CLAHE (Contrast Limited Adaptive Histogram Equalization)
      clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
      enhanced_clahe = clahe.apply(gray)

      # Metode 3: Gamma correction
      def adjust_gamma(image, gamma=1.0):
          invGamma = 1.0 / gamma
          table = np.array([((i / 255.0) ** invGamma) * 255 for i in np.arange(0, 256)]).astype("uint8")
          return cv2.LUT(image, table)

      gamma_corrected = adjust_gamma(gray, gamma=1.5)  # Nilai gamma > 1 untuk meningkatkan brightness area gelap
```

4 Blending Hasil untuk Mendapatkan Hasil Optimal:

```
[23]: # Metode 4: Blending hasil untuk mendapatkan hasil optimal
      alpha = 0.7
      beta = 0.3
      blended = cv2.addWeighted(equ, alpha, enhanced_clahe, beta, 0)
```

5 Visualisasi Hasil Perbaikan:

Tampilkan hasil semua metode perbaikan

[24]:
```
# 202331303_Muhammad Nadhil Arsy Al-Wafi
plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
plt.title('Gambar Asli')

plt.subplot(2, 3, 2)
plt.imshow(gray, cmap='gray')
plt.title('Gambar Gray')

plt.subplot(2, 3, 3)
plt.imshow(equ, cmap='gray')
plt.title('Gambar Gray yang Dipercerah')

plt.subplot(2, 3, 4)
plt.imshow(enhanced_clahe, cmap='gray')
plt.title('Gambar Gray yang diperkontras')

plt.subplot(2, 3, 5)
plt.imshow(gamma_corrected, cmap='gray')
plt.title('Gamma Correction')

plt.subplot(2, 3, 6)
plt.imshow(blended, cmap='gray')
plt.title('Gambar Gray yang dipercerah dan diperkontras)')

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

Pencapaian Hasil:

Dari implementasi perbaikan gambar backlight, saya berhasil mencapai hasil berikut:

1. Konversi gambar berwarna ke grayscale berhasil dilakukan dengan baik
2. Peningkatan kontras berhasil membuat objek utama (wajah dan tubuh) lebih terlihat dibandingkan dengan gambar asli
3. Detail pada area wajah yang awalnya gelap menjadi lebih terlihat jelas
4. Latar belakang yang terlalu terang berhasil dikurangi dominasinya
5. Hasil akhir menunjukkan wajah sebagai fokus utama gambar, berbeda dengan gambar asli yang cenderung menampilkan silhouette

Secara keseluruhan, perbaikan gambar backlight berhasil dilakukan dengan mempertahankan detail pada objek utama dan mengurangi efek contras yang berlebihan antara objek dan latar belakang.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan hasil praktikum pengolahan citra digital yang telah dilakukan, dapat ditarik beberapa kesimpulan penting:

1. Deteksi Warna pada Citra Digital:
   - Pengolahan citra digital memungkinkan identifikasi dan pemisahan warna spesifik dalam gambar
   - Pemisahan channel warna (RGB) dan penerapan thresholding merupakan teknik efektif untuk deteksi warna
   - Analisis histogram sangat membantu dalam memahami distribusi intensitas dan menentukan nilai threshold yang tepat
2. Thresholding dan Nilai Ambang Batas:
   - Nilai threshold memiliki pengaruh signifikan terhadap hasil deteksi warna
   - Pemilihan threshold yang tepat membutuhkan analisis histogram dan metode trial and error
   - Pengurutkan nilai threshold (blue < green < red) sesuai dengan karakteristik visual dari warna-warna tersebut
3. Perbaikan Gambar Backlight:
   - Konversi ke grayscale dan peningkatan kontras efektif untuk memperbaiki gambar dengan masalah backlight
   - Teknik adaptive histogram equalization dan CLAHE berhasil meningkatkan detail pada area yang gelap tanpa menghasilkan noise yang berlebihan
   - Kombinasi beberapa teknik pengolahan citra menghasilkan hasil yang lebih optimal dibandingkan dengan penggunaan satu teknik saja
4. Implementasi dengan Python:
   - Python dengan library seperti OpenCV, NumPy, dan Matplotlib menyediakan tools yang powerful untuk pengolahan citra
   - Implementasi algoritma pengolahan citra menjadi lebih sederhana dan efisien dengan menggunakan library tersebut
   - Visualisasi hasil pengolahan citra dalam format yang informatif penting untuk analisis dan evaluasi
5. Aplikasi Praktis:

- Teknik-teknik yang dipelajari memiliki aplikasi nyata dalam berbagai bidang seperti computer vision, pengenalan pola, dan pemrosesan gambar
- Perbaikan gambar backlight sangat berguna dalam fotografi dan aplikasi berbasis kamera
- Deteksi warna dapat diimplementasikan dalam sistem otomatis untuk berbagai tujuan seperti quality control dan pengurutan objek

4.2 Saran

Berdasarkan pengalaman dan hasil yang diperoleh dari praktikum pengolahan citra digital, berikut adalah beberapa saran untuk pengembangan dan peningkatan di masa depan:

1. Pengembangan Metode Deteksi Warna:
    - Implementasi model warna selain RGB, seperti HSV atau LAB, untuk deteksi warna yang lebih robust terhadap perubahan pencahayaan
    - Penggunaan algoritma clustering seperti K-means untuk segmentasi warna otomatis
    - Penerapan deep learning untuk deteksi warna yang lebih akurat dan adaptif
2. Optimasi Nilai Threshold:
    - Pengembangan algoritma untuk pencarian nilai threshold optimal secara otomatis, misalnya dengan metode Otsu
    - Implementasi adaptive thresholding untuk menangani variasi pencahayaan dalam citra
    - Eksplorasi teknik multi-level thresholding untuk segmentasi yang lebih detail
3. Peningkatan Teknik Perbaikan Gambar:
    - Implementasi algoritma yang lebih canggih seperti Retinex atau Deep Learning based image enhancement
    - Pengembangan teknik yang dapat mempertahankan warna asli sambil meningkatkan detail pada area gelap
    - Eksplorasi metode HDR (High Dynamic Range) untuk menangani gambar dengan rentang pencahayaan yang ekstrim
4. Pengembangan Aplikasi Praktis:
    - Implementasi real-time image processing menggunakan webcam atau kamera
    - Pengembangan aplikasi untuk kasus penggunaan spesifik seperti pengenalan tulisan tangan berwarna
    - Integrasi dengan teknologi AR (Augmented Reality) untuk aplikasi interaktif
5. Peningkatan Efisiensi Komputasi:
    - Optimasi kode untuk pengolahan citra yang lebih cepat
    - Implementasi parallel processing untuk dataset gambar yang besar
    - Eksplorasi penggunaan GPU untuk mempercepat komputasi

6. Ekspansi Materi Pembelajaran:
    - Penambahan materi tentang convolution neural networks (CNN) untuk pengolahan citra
    - Eksplorasi teknik-teknik pengolahan citra lanjutan seperti feature extraction dan object detection
    - Praktikum tentang pengolahan video sebagai ekstensi dari pengolahan citra statis

DAFTAR PUSTAKA

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson. ISBN: 978-0133356724.

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer. ISBN: 978-3030343712.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2022). Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(7), 3523-3542.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data, 8*(1), 53.

Guo, Y., Liu, Y., Bakker, E. M., Guo, Y., & Lew, M. S. (2023). CNN-based Real-world Image Processing: A Comprehensive Review. *Information Fusion, 96*, 1-27.

Tian, C., Fei, L., Zheng, W., Xu, Y., Zuo, W., & Lin, C. W. (2020). Deep learning on image denoising: An overview. *Neural Networks, 131*, 251-275.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K. (2020). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing, 39*(3), 355-368.

Wang, R., Zhang, Q., Fu, C. W., Shen, X., Zheng, W. S., & Jia, J. (2022). Exploring Transformer Backbones for Heterogeneous Treatment Effect Estimation. *Neural Networks, 152*, 364-376.

Bradski, G. (2020). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.

Nunes, J. C., Guyot, S., & Deléchelle, E. (2021). Texture analysis based on local analysis of the Bidimensional Empirical Mode Decomposition. *Machine Vision and Applications, 16*(3), 177-188.