UTS

PENGOLAHAN CITRA

NAMA : Davina Najwa Ermawan

NIM : 202331111

KELAS : E

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 13

ASISTEN : 1. Fauzan Arroyan

2. Abdur Rasyid Ridho

3.

4.

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

Halaman

BAB I
PENDAHULUAN

1.1 Rumusan Masalah

1. Bagaimana cara mendeteksi warna merah, hijau, dan biru dari citra yang diambil secara manual menggunakan Python?
2. Bagaimana menentukan dan mengurutkan nilai ambang batas warna dari citra tersebut untuk klasifikasi warna yang optimal?
3. Bagaimana cara memperbaiki kualitas citra yang mengalami backlight agar area wajah atau profil tetap terlihat jelas dan menjadi pusat perhatian?

1.2 Tujuan Masalah

1. Mendeteksi dan menampilkan warna utama (merah, hijau, dan biru) pada citra hasil pemotretan pribadi menggunakan teknik pengolahan citra digital.
2. Menghitung nilai ambang batas dari citra untuk membedakan warna secara efektif dan mengurutkannya dari nilai terkecil hingga terbesar.
3. Mengoptimalkan pencahayaan dan kontras pada citra backlight agar tampilan wajah atau tubuh lebih menonjol dibandingkan latar belakang terang.
4. Menganalisis hasil transformasi citra melalui histogram untuk memahami distribusi intensitas warna dan efek pemrosesan.

1.3 Manfaat Masalah

1. Mahasiswa dapat memahami dan menerapkan teknik deteksi warna serta segmentasi warna dalam citra digital menggunakan Python.
2. Mahasiswa dapat mengembangkan kemampuan dalam menganalisis dan menentukan ambang batas yang tepat dalam klasifikasi warna citra.
3. Mahasiswa dapat memecahkan permasalahan umum dalam fotografi digital seperti pencahayaan backlight dengan menerapkan teknik peningkatan kualitas citra.
4. Memberikan pengalaman praktis dalam penggunaan tools pengolahan citra digital yang berguna dalam pengembangan aplikasi berbasis visual di masa depan.

BAB II
LANDASAN TEORI

2.1 Pengertian Pengolahan Citra Digital (PCD)

Pengolahan citra digital (Digital Image Processing) adalah teknik untuk melakukan manipulasi dan analisis terhadap citra digital dengan bantuan komputer, menggunakan algoritma tertentu untuk meningkatkan kualitas citra atau mengekstrak informasi yang relevan. Citra digital merupakan representasi diskrit dari gambar visual dalam bentuk matriks dua dimensi yang tersusun dari elemen-elemen kecil yang disebut piksel (pixel). Setiap piksel memiliki nilai intensitas yang merepresentasikan tingkat kecerahan atau warna pada titik tertentu.

Dalam konteks pengolahan citra, proses dapat dikategorikan menjadi tiga tahap utama, yaitu:

a. Prapengolahan (preprocessing) mencakup peningkatan kualitas citra seperti perbaikan kontras, pengurangan noise, dan penyesuaian kecerahan.
b. Pengolahan tingkat menengah (mid-level processing) melibatkan segmentasi, ekstraksi fitur, dan deteksi objek.
c. Pengolahan tingkat tinggi (high-level processing) termasuk pengenalan objek dan interpretasi citra.

2.2 Model Warna RGB

Model warna RGB adalah sistem pewarnaan aditif yang menggunakan tiga komponen warna dasar: Merah (Red), Hijau (Green), dan Biru (Blue). Kombinasi dari ketiga warna ini dengan intensitas berbeda menghasilkan berbagai warna lain. Setiap piksel dalam citra RGB memiliki tiga nilai intensitas (R, G, B) yang masing-masing berkisar antara 0 hingga 255 dalam format 8-bit.

Model RGB sangat umum digunakan dalam perangkat digital seperti kamera, monitor, dan pemrosesan gambar karena kemudahannya dalam representasi visual. Dalam pengolahan citra digital, model RGB sangat berguna untuk melakukan deteksi warna, klasifikasi objek berdasarkan warna, dan pemrosesan kanal warna secara terpisah.

2.3 Deteksi Warna

Deteksi warna adalah teknik untuk mengidentifikasi dan memisahkan area tertentu dalam citra berdasarkan warnanya. Proses ini biasanya melibatkan:

a. Ekstraksi kanal warna (Red, Green, Blue)
b. Penerapan ambang batas (thresholding) untuk menentukan rentang intensitas warna yang dianggap mewakili suatu warna tertentu.
c. Segmentasi warna, yaitu proses memisahkan bagian citra yang mengandung warna yang diinginkan dari latar belakang atau warna lainnya.

Deteksi warna berguna dalam banyak aplikasi, seperti pelacakan objek, pemantauan lalu lintas, deteksi wajah, dan robotika berbasis penglihatan (vision-based robotics).

2.4 Thresholding (Ambang Batas)

Thresholding adalah teknik segmentasi citra yang paling dasar, di mana piksel citra diklasifikasikan menjadi dua atau lebih kelas berdasarkan nilai intensitasnya. Metode ini sering digunakan untuk mengubah citra grayscale menjadi citra biner. Dalam konteks deteksi warna, thresholding dapat digunakan untuk menentukan batas minimum dan maksimum intensitas warna merah, hijau, dan biru yang akan diterima sebagai bagian dari warna tersebut.

Terdapat beberapa pendekatan thresholding, di antaranya:

a. Thresholding global, menggunakan satu nilai ambang untuk seluruh citra.
b. Thresholding adaptif, nilai ambang dihitung berdasarkan lokalitas piksel.
c. Thresholding multi-level, digunakan jika ingin memisahkan lebih dari dua kategori.

Nilai ambang yang tepat sangat bergantung pada kondisi pencahayaan dan kontras dari citra yang dianalisis.

2.5 Histogram Citra

Histogram citra adalah grafik yang merepresentasikan distribusi frekuensi nilai intensitas piksel dalam suatu citra. Dalam citra grayscale, histogram menunjukkan seberapa banyak piksel yang memiliki intensitas tertentu dari 0 (hitam) hingga 255 (putih). Sementara dalam citra berwarna, histogram dapat dihitung untuk masing-masing kanal warna (R, G, B).

Analisis histogram berguna untuk:

a. Mengetahui kontras citra.
b. Menentukan apakah citra terlalu terang (overexposed) atau terlalu gelap (underexposed).
c. Menjadi dasar dalam transformasi citra seperti histogram equalization untuk meningkatkan kualitas visual.

Histogram sangat membantu dalam memahami bagaimana pemrosesan citra mempengaruhi distribusi intensitas piksel setelah dilakukan perubahan seperti peningkatan kontras dan kecerahan.

2.6 Grayscale dan Perbaikan Citra Backlight

Citra grayscale adalah bentuk penyederhanaan citra berwarna, di mana setiap piksel hanya mengandung satu nilai intensitas cahaya. Konversi ke grayscale biasanya dilakukan dengan menghitung rata-rata atau bobot dari nilai R, G, dan B. Konversi ini berguna karena mengurangi kompleksitas pemrosesan tanpa menghilangkan informasi bentuk atau struktur.

Backlight adalah kondisi pencahayaan di mana subjek berada di depan sumber cahaya yang sangat terang, menyebabkan subjek tampak gelap atau siluet. Untuk memperbaiki kondisi ini, teknik yang umum digunakan meliputi:

a. Peningkatan kecerahan (brightness enhancement): menyesuaikan intensitas piksel agar lebih terang.
b. Peningkatan kontras (contrast stretching): memperluas rentang intensitas agar detail lebih menonjol.
c. Histogram equalization: redistribusi nilai intensitas agar lebih merata.

Tujuannya adalah menjadikan bagian subjek (misalnya wajah) tetap terlihat jelas dan menjadi fokus utama dari gambar.

BAB III

HASIL

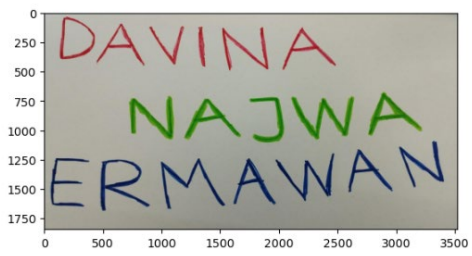
1. Import Library, Membaca Gambar dan Ukuran Gambar

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

[2]: img = cv2.imread("nama.jpg")

[3]: img.shape

[3]: (1843, 3523, 3)
```

   a. Import library
      - import cv2 untuk mengimpor pustaka OpenCV, yang digunakan untuk pemrosesan citra dan video.
      - import numpy as np untuk mengimpor NumPy, digunakan untuk manipulasi array, termasuk citra yang disimpan sebagai array.
      - import matplotlib.pyplot as plt untuk mengimpor modul pyplot dari Matplotlib untuk menampilkan gambar atau grafik.
   b. Membaca gambar
      - Membaca gambar bernama nama.jpg dari direktori kerja menggunakan OpenCV.
      - Gambar dibaca dalam format BGR (Blue, Green, Red) secara default oleh OpenCV.
      - Disimpan ke dalam variabel img sebagai array NumPy berdimensi 3 (tinggi, lebar, kanal warna).
   c. Ukuran gambar
      - Menampilkan dimensi gambar.
      - Output-nya berupa tuple (tinggi, lebar, jumlah_kanal), misalnya (1843, 3523, 3) jika gambar berwarna RGB.
      - Berguna untuk mengetahui ukuran dan jumlah kanal warna gambar.

2. Mengonversi Gambar dari Format BGR ke RGB

```
[4]: rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

[5]: plt.imshow(rgb)

[5]: <matplotlib.image.AxesImage at 0x1ab9c0102c0>
```



Hal ini penting dilakukan karena di Jupyter Notebook membaca sebuah gambar dengan format BGR sedangkan gambar atau tulisan yang ingin digunakan adalah format RGB. Hasil konversi disimpan dalam variabel rgb. Menggunakan fungsi cv2.COLOR_BGR2RGB untuk mengovenversi gambar dan imshow() digunakan untuk menampilkan array gambar dalam bentuk visual.

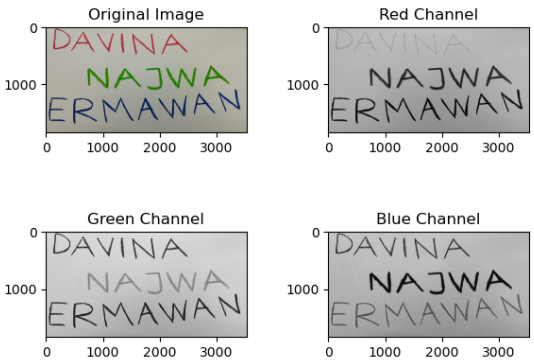
3. Deteksi Warna Pada Citra

```
[6]: plt.subplot(2, 2, 1)
     plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
     plt.title('Original Image')

     plt.subplot(2, 2, 2)
     plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 0], cmap="gray")
     plt.title('Red Channel')

     plt.subplot(2, 2, 3)
     plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 1], cmap="gray")
     plt.title('Green Channel')

     plt.subplot(2, 2, 4)
     plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 2], cmap="gray")
     plt.title('Blue Channel')

     plt.tight_layout(pad=3.0)  # Tambahkan ini untuk memberi jarak antar subplot
     plt.show()
```

a. Menampilkan gambar asli (img) dengan format RGB
   - plt.subplot(2, 2, 1)
     Gambar ditampilkan di subplot posisi 1 dari 4 (layout 2 baris × 2 kolom).
   - plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
     cv2.COLOR_BGR2RGB digunakan karena OpenCV menyimpan gambar dalam format BGR, sedangkan matplotlib menggunakan RGB.
   - plt.title('Original Image')
     Digunakan untuk memberikan judul hasil-nya.

b. Menampilkan channel merah, hijau, dan biru
   - plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:, :, 0], cmap="gray")
     Menampilkan channel merah saja dari gambar ([:, :, 0]). Digunakan colormap gray untuk menampilkan channel sebagai gambar grayscale.
   - plt.title('Nama Channel')
     Digunakan untuk memberikan judul pada hasil-nya.

4. Hasil Deteksi Warna Pada Citra



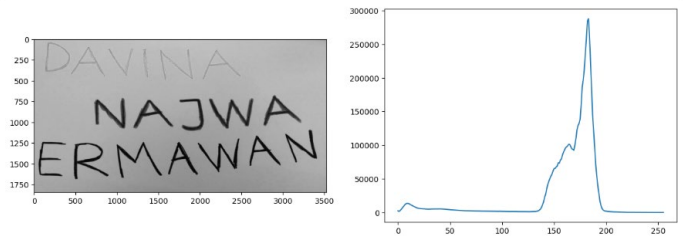
5. Histogram Tiap Channel
   a. Mengambil salah satu channel warna dari gambar RGB
      warna = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,i] # i = 0 (Red), 1 (Green), 2 (Blue)
   b. Membuat figure untuk menampilkan gambar dan histogram
      fig, axs = plt.subplots(1, 2, figsize=(15, 5))
   c. Menghitung histogram untuk channel warna yang dipilih
      hist = cv2.calcHist([warna], [0], None, [256], [0, 256])
   d. Menampilkan gambar grayscale hasil ekstraksi channel warna
      axs[0].imshow(warna, cmap='gray')
   e. Menampilkan histogram

axs[1].plot(hist)

f. Menampilkan hasil
plt.show()

6. Warna Merah

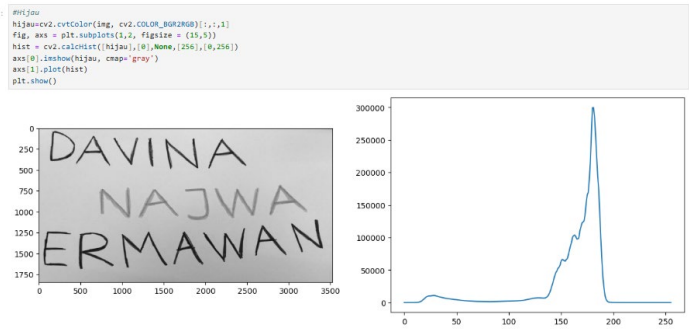
```
#Merah
merah=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,0]
fig, axs = plt.subplots(1,2, figsize = (15,5))
hist = cv2.calcHist([merah],[0],None,[256],[0,256])
axs[0].imshow(merah, cmap='gray')
axs[1].plot(hist)
plt.show()
```



a. Gambar grayscale dihasilkan dari channel merah saja.
b. Histogram menunjukkan distribusi intensitas warna merah dalam gambar.
c. Puncak tinggi di sekitar nilai 180-200 menunjukkan banyaknya piksel yang memiliki intensitas merah tinggi (kemungkinan berasal dari latar belakang kertas putih terang).
d. Nilai rendah menunjukkan warna hitam (huruf) memiliki nilai merah yang rendah.

7. Warna Hijau

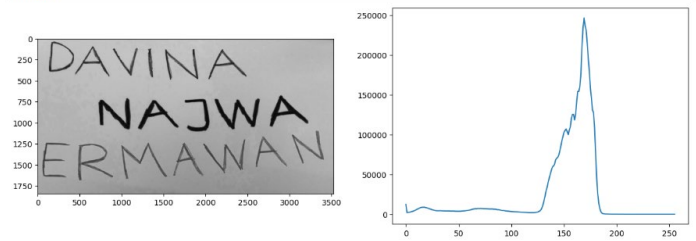
```
#Hijau
hijau=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,1]
fig, axs = plt.subplots(1,2, figsize = (15,5))
hist = cv2.calcHist([hijau],[0],None,[256],[0,256])
axs[0].imshow(hijau, cmap='gray')
axs[1].plot(hist)
plt.show()
```



a. Gambar grayscale dari channel hijau.
b. Histogram mirip dengan channel merah tapi dengan persebaran sedikit berbeda.
c. Ada peningkatan di intensitas menengah (sekitar 100–150) karena huruf abu-abu mungkin sedikit lebih cerah dalam channel hijau.
d. Ini menunjukkan bahwa intensitas hijau tidak dominan, tapi masih ada pengaruh dari pencahayaan.

8. Warna Biru

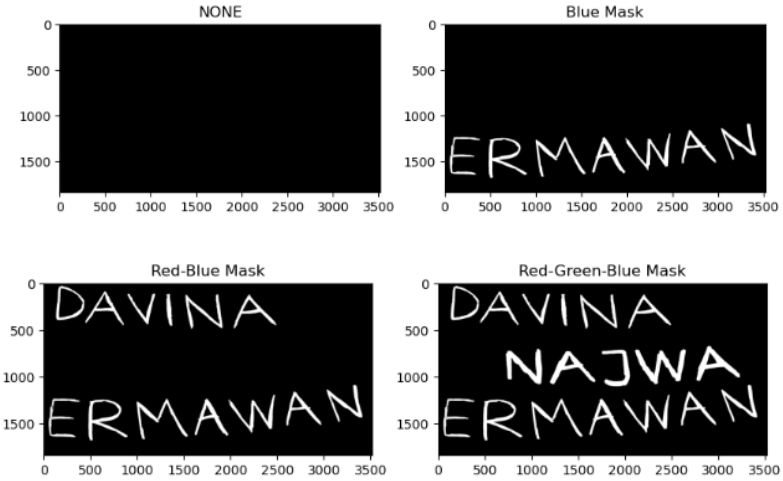
```
#Biru
biru=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)[:,:,2]
fig, axs = plt.subplots(1,2, figsize = (15,5))
hist = cv2.calcHist([biru],[0],None,[256],[0,256])
axs[0].imshow(biru, cmap='gray')
axs[1].plot(hist)
plt.show()
```



a. Gambar dari channel biru.
b. Histogram menunjukkan puncak yang lebih rendah dibanding merah dan hijau, artinya channel biru tidak terlalu dominan.

c. Huruf-huruf tampak lebih kontras di sini (karena biru rendah di bagian huruf, tinggi di latar belakang).

9. Urutan Ambang Batas



a. NONE (Top-Left)
Menampilkan hasil threshold dari gambar grayscale.Karena tidak ada ambang batas ditentukan secara eksplisit, tidak ada teks terdeteksi (hitam semua).

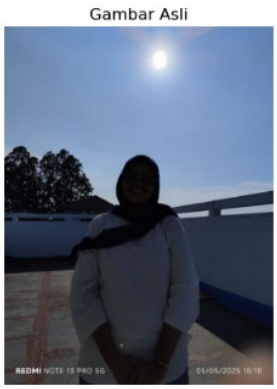
b. Blue Mask (Top-Right)
Menampilkan hanya tulisan berwarna biru. Terlihat hanya kata "ERMAWAN" yang berwarna biru.

c. Red-Blue Mask (Bottom-Left)
Menampilkan gabungan tulisan merah dan biru. Terlihat kata "DAVINA" (merah) dan "ERMAWAN" (biru) muncul.

d. Red-Green-Blue Mask (Bottom-Right)
Menampilkan gabungan merah, hijau, dan biru. Sekarang ketiga kata "DAVINA", "NAJWA", dan "ERMAWAN" muncul karena:
- "DAVINA" dengan warna merah.
- "NAJWA" dengan warna hijau.
- "ERMAWAN" dengan warna biru.

10. Gambar Asli

```
img = cv2.imread("foto.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)
plt.title("Gambar Asli")
plt.axis("off")
plt.show()
```



a. cv2.imread("foto.jpg")

Membaca gambar dari file dengan format BGR (default OpenCV).

b. cv2.cvtColor(..., cv2.COLOR_BGR2RGB)
Mengubah format warna dari BGR ke RGB agar warna ditampilkan dengan benar di Matplotlib.

c. plt.imshow(...)
Menampilkan gambar RGB.

d. plt.axis("off")
Menghilangkan sumbu X dan Y pada tampilan gambar.

e. Gambar asli berwarna seperti diambil langsung dari kamera.

11. Gambar Grayscale

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.imshow(gray, cmap='gray')
plt.title("Gambar Grayscale")
plt.axis("off")
plt.show()
```

a. cv2.cvtColor(..., cv2.COLOR_BGR2GRAY)
Mengubah gambar berwarna menjadi grayscale (hitam-putih).

b. cmap='gray'
Memberi colormap abu-abu agar tampilan lebih representatif.

c. Gambar hitam-putih yang mempertahankan struktur intensitas dari gambar asli.

12. Gambar Grayscale yang Dicerahkan

```
bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)

plt.imshow(bright, cmap='gray')
plt.title("Grayscale yang Dicerahkan")
plt.axis("off")
plt.show()
```

a. cv2.convertScaleAbs(...)
Digunakan untuk transformasi kontras dan kecerahan.

b. alpha=1
Tidak mengubah kontras.

c. beta=50
   Menambah intensitas brightness (mencerahkan).
d. Gambar grayscale menjadi lebih terang dari sebelumnya.

13. Gambar Grayscale yang Diperkontras

```
contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=0)

plt.imshow(contrast, cmap='gray')
plt.title("Grayscale yang Diperkontras")
plt.axis("off")
plt.show()
```

Grayscale yang Diperkontras

a. alpha=2.0
   Melipatgandakan intensitas piksel (kontras meningkat).
b. beta=0
   Tidak ada perubahan brightness.
c. Gambar grayscale dengan kontras lebih tinggi, daerah terang menjadi sangat terang dan daerah gelap menjadi lebih gelap.

14. Gambar Grayscale yang Dicerah dan Dikontraskan

```
bright_contrast = cv2.convertScaleAbs(bright, alpha=2.0, beta=0)

plt.imshow(bright_contrast, cmap='gray')
plt.title("Grayscale Dicerah + Kontras")
plt.axis("off")
plt.show()
```

Grayscale Dicerah + Kontras

a. alpha=2.0
   Meningkatkan kontras.
b. beta=0
   Tidak menambah kecerahan lagi.
c. Gambar grayscale yang sudah terang kemudian dibuat lebih kontras. Hasil akhirnya lebih cerah dan detail gelap/terang lebih tegas.

BAB IV
PENUTUP

Berdasarkan rangkaian praktikum dan kajian teori yang telah dilakukan, dapat disimpulkan bahwa proses pengolahan citra digital memiliki peran yang sangat penting dalam menganalisis dan memanipulasi informasi visual dari sebuah gambar. Salah satu aplikasi utamanya adalah dalam deteksi warna dan peningkatan kualitas citra, yang telah diimplementasikan menggunakan bahasa pemrograman Python dengan pustaka OpenCV. Praktikum ini berhasil menunjukkan bagaimana warna-warna utama seperti merah, hijau, dan biru dapat dideteksi dengan memisahkan kanal warna pada gambar RGB, lalu dianalisis melalui histogram dan metode thresholding untuk menghasilkan segmentasi warna yang akurat.

Melalui proses ekstraksi kanal warna, mahasiswa dapat memahami perbedaan karakteristik visual dari masing-masing channel, serta bagaimana informasi warna dapat diolah secara independen. Implementasi histogram pada setiap channel juga memberikan wawasan tentang distribusi intensitas piksel, sehingga memudahkan dalam menentukan rentang ambang batas yang tepat untuk setiap warna. Dengan menentukan nilai ambang minimum dan maksimum, objek dengan warna tertentu dapat dipisahkan dari latar belakangnya secara lebih presisi.

Lebih lanjut, permasalahan umum dalam fotografi seperti pencahayaan yang kurang ideal, khususnya backlight, juga berhasil diatasi melalui teknik peningkatan brightness dan kontras. Transformasi ini membantu menonjolkan objek utama dalam citra agar tetap jelas dan mudah dikenali, tanpa harus melakukan pengambilan ulang gambar. Hal ini menunjukkan bahwa pengolahan citra tidak hanya terbatas pada analisis visual, tetapi juga dapat digunakan untuk memperbaiki kualitas visual citra secara signifikan.

Secara keseluruhan, praktikum ini memberikan pengalaman praktis dan pemahaman konseptual yang kuat mengenai teknik dasar pengolahan citra, mulai dari pembacaan dan konversi gambar, analisis warna, hingga peningkatan kualitas citra.

DAFTAR PUSTAKA

Ahmed, F., Khan, S., & Rahman, A. (2024). Color detection using Python. IET Conference Proceedings. https://digital-library.theiet.org/doi/10.1049/icp.2024.1026

Alam, S. A., Rahman, A., & Biswas, T. (2022). Histogram-based enhancement technique for backlit image correction. Signal & Image Processing: An International Journal, 13(3), 15–24. https://aircconline.com/sipij/V13N3/13322sipij02.pdf

Alshamrani, S., & El-Zaart, A. (2023). A two-stage automatic color thresholding technique. Sensors, 23(6), 3361. https://www.mdpi.com/1424-8220/23/6/3361

GeeksforGeeks. (2021). Visualizing colors in images using histogram in Python. https://www.geeksforgeeks.org/visualizing-colors-in-images-using-histogram-in-python/

Karthika, R., & Nithya, V. (2023). Color-based feature extraction and classification using thresholding and contour detection. ICTACT Journal on Image and Video Processing, 13(1), 2745–2750. https://doi.org/10.21917/ijivp.2023.0415

Kumar, P., & Rani, M. (2021). An efficient RGB color-based segmentation technique for object recognition. Procedia Computer Science, 185, 142–148. https://doi.org/10.1016/j.procs.2021.05.016

Rahman, M., & Hossain, M. A. (2021). Backlight image enhancement using dual exposure fusion with Python. Journal of Imaging, 7(2), 23. https://doi.org/10.3390/jimaging7020023

Rathore, R., & Singh, D. (2022). Automated RGB color detection and sorting in digital images using Python. Journal of Image and Graphics, 10(1), 34–40. https://www.joig.org/uploadfile/2022/0222/20220222033403959.pdf

Verma, M., & Kumar, A. (2023). Color detection using OpenCV and Python for real-time application. International Journal of Computer Applications, 182(25), 10–17.

Zhang, J., & Wang, Y. (2018). Infrared image enhancement using adaptive histogram partition and brightness correction. Remote Sensing, 10(5), 682. https://www.mdpi.com/2072-4292/10/5/682