Laporan Tugas I

Brightness & Contrast

Name:	**Ikhwanul Abiyu Dhiyya'ul Haq**
NRP:	5024211048
Lecturer:	Dr. Eko Mulyanto Yuniarno, S.T.,M.T.
Subject:	Image & Video Processing

Table of Contents

1 Pendahuluan

Pengolahan gambar adalah bagian penting dari banyak aplikasi teknologi, termasuk pengenalan pola, visi komputer, dan grafika komputer. Dalam konteks ini, penyesuaian brightness (kecerahan) dan contrast (kontras) adalah dua operasi dasar yang sering digunakan untuk meningkatkan kualitas gambar atau mengubah tampilannya sesuai kebutuhan tertentu. Penambahan brightness akan mengangkat seluruh tingkat kecerahan gambar, sedangkan penambahan contrast akan membedakan perbedaan antara warna dan elemen dalam gambar.

Laporan ini bertujuan untuk membandingkan dua metode berbeda untuk melakukan penambahan dan pengurangan brightness serta contrast dalam pemrosesan gambar. Metode pertama melibatkan penggunaan OpenCV, sebuah perpustakaan sumber terbuka yang populer dalam pemrosesan gambar dan komputer. Metode kedua adalah implementasi manual yang tidak melibatkan OpenCV. Perbandingan ini akan memberikan wawasan tentang keuntungan dan kerugian masing-masing pendekatan.

1.1 Pentingnya Penyesuaian Brightness dan Contrast

Penyesuaian brightness dan contrast memiliki aplikasi luas dalam berbagai konteks. Misalnya, dalam fotografi, penambahan brightness dapat membuat gambar terlihat lebih terang dan jelas, sementara penambahan contrast dapat membuat detail lebih tajam. Dalam pemrosesan medis, penyesuaian ini dapat membantu dalam analisis citra medis untuk diagnosis yang lebih akurat. Selain itu, dalam produksi video dan sinematografi, penyesuaian brightness dan contrast adalah komponen penting dari editing visual.

1.2 Penggunaan OpenCV

OpenCV adalah perpustakaan perangkat lunak sumber terbuka yang sangat populer dalam pemrosesan gambar dan pengenalan pola. Ini menawarkan berbagai algoritma dan fungsi yang dapat digunakan untuk berbagai tugas pemrosesan gambar, termasuk penambahan dan pengurangan brightness serta contrast. Menggunakan OpenCV dapat mempercepat proses pengolahan gambar dan memungkinkan pengguna untuk mengakses algoritma yang telah dioptimalkan secara ekstensif.

1.3 Tujuan Laporan

Adapun tujuan laporan ini yaitu :

1. Membandingkan hasil penambahan dan pengurangan brightness serta contrast antara penggunaan OpenCV dan implementasi manual.

2. Menganalisis kelebihan dan kekurangan masing-masing metode dalam hal kecepatan, akurasi, dan kompleksitas implementasi.

3. Memberikan rekomendasi tentang penggunaan metode yang lebih sesuai dalam konteks tertentu.

Selanjutnya, laporan ini akan membahas metode yang digunakan dalam penambahan dan pengurangan brightness serta contrast, menampilkan kode sumber yang relevan, dan melakukan perbandingan hasil yang diperoleh dari kedua metode tersebut.

2 Metode

Dalam bagian ini, akan dijelaskan metode yang digunakan untuk melakukan penambahan dan pengurangan brightness serta contrast dengan menggunakan OpenCV dan metode implementasi manual (tanpa OpenCV).

$$g(i,j) = \alpha \cdot f(i,j) + \beta$$

Rumus di atas merupakan rumus yang dipakai oleh pustaka OpenCV, yang digunakan pada program *contrast* dan *brightness*. Dimana alpha merupakan *gain* dan beta merupakan *bias*, yang juga biasa disebut dengan *contrast* dan *brightness*

2.1 Menggunakan OpenCV

2.1.1 Deskripsi

Metode pertama yang digunakan adalah menggunakan OpenCV, perpustakaan pemrosesan gambar yang populer. OpenCV menyediakan berbagai algoritma dan fungsi yang mempermudah proses penambahan dan pengurangan brightness serta contrast.

2.1.2 Langkah-langkah

1. Mengimpor *library* OpenCV ke dalam *Visual Studio Code*.
2. Memuat gambar yang akan diproses ke dalam format yang sesuai.
3. Menggunakan fungsi OpenCV yang tersedia untuk melakukan penambahan atau pengurangan brightness serta contrast.

2.2 Tanpa Menggunakan OpenCV

2.2.1 Deskripsi

Metode kedua melibatkan implementasi manual untuk penambahan dan pengurangan brightness serta contrast. Ini tidak melibatkan penggunaan perpustakaan OpenCV.

1. Memuat gambar ke dalam struktur data yang sesuai, seperti matriks piksel.
2. Iterasi melalui setiap piksel dalam gambar.
3. Untuk penambahan brightness, tambahkan nilai kecerahan yang ditentukan ke setiap saluran warna (R, G, B) dari setiap piksel.
4. Untuk pengurangan brightness, kurangkan nilai kecerahan yang ditentukan dari setiap saluran warna (R, G, B) dari setiap piksel.
5. Untuk penambahan contrast, perkalian nilai kecerahan setiap piksel dengan faktor kontras yang ditentukan.
6. Untuk pengurangan contrast, bagi nilai kecerahan setiap piksel dengan faktor kontras yang ditentukan.
7. Pastikan bahwa nilai piksel tetap dalam rentang yang valid, misalnya, antara 0 hingga 255.

Dengan metode ini, kami akan dapat membandingkan hasil dari kedua pendekatan ini dalam hal hasil pengolahan gambar, waktu yang diperlukan, dan kompleksitas implementasi. Selanjutnya, kami akan menampilkan kode sumber untuk masing-masing metode ini dan melakukan perbandingan hasil yang diperoleh.

3 Sumber Kode

Dalam bagian ini, akan disajikan kode sumber untuk melakukan penambahan dan pengurangan brightness serta contrast dengan menggunakan OpenCV (Metode 2.1) dan implementasi manual (tanpa OpenCV, Metode 2.2). Kode sumber ini akan memberikan pandangan praktis tentang cara masing-masing metode diimplementasikan dalam pemrosesan gambar.

3.1 Penambahan & Pengurangan *Brightness*

3.1.1 Menggunakan OpenCV

```
import cv2
import time
import numpy as np
import os

# create a directory to save the time_taken.txt
if not os.path.exists('brightness'):
    os.makedirs('brightness')

brightness = input('Enter the brightness value: ')

start_time = time.time()

img = cv2.imread('itsSurabaya2.jpg',0)

# Menambah kecerahan
img_brightness = cv2.addWeighted(img, 1, np.zeros(img.shape, img.dtype), 0,
    brightness)

cv2.imshow('image', img)
cv2.imshow('brightness', img_brightness)

time_taken = time.time() - start_time
print("%.4f seconds" % time_taken)

cv2.imwrite('brightness/brightness_with_opencv.jpg', img_brightness)
with open('brightness/time_taken_br_with_opencv.txt', 'a') as f:
    f.write(str(time_taken) + '\n')

cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.1.2 Tanpa menggunakan OpenCV

```
import cv2
import numpy as np
import time
import os

if not os.path.exists('brightness'):
    os.makedirs('brightness')

brightness = int(input("Enter brightness value : "))

start_time = time.time()

img = cv2.imread('itsSurabaya2.jpg',0)

img_brightness = np.double(img)+brightness # 50 is the brightness value, convert
    to double to prevent overflow
```

```
img_brightness[img_brightness>255] = 255 # if the value is more than 255, set it to 255
img_brightness[img_brightness<0] = 0 # if the value is less than 0, set it to 0
# with if else syntax
# for i in range(img.shape[0]):
#     for j in range(img.shape[1]):
#         for k in range(img.shape[2]):
#             if img_brightness[i, j, k] > 255:
#                 img_brightness[i, j, k] = 255
#             elif img_brightness[i, j, k] < 0:
#                 img_brightness[i, j, k] = 0
img_brightness = np.uint8(np.floor(img_brightness)) # convert the value to uint8, balikin lagi

time_taken = time.time() - start_time

cv2.imshow('image', img)
cv2.imshow('brightness', img_brightness)

print("%.4f seconds" % time_taken)

cv2.imwrite('brightness/brightness_without_opencv.jpg', img_brightness)
with open('brightness/time_taken_br_without_opencv.txt', 'a') as f:
    f.write(str(time_taken) + '\n')


cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.2 Penambahan & Pengurangan *Contrast*

3.2.1 Menggunakan OpenCV

```
import cv2
import numpy as np
import time
import os

# create a directory to save the time_taken.txt
if not os.path.exists('contrast'):
    os.makedirs('contrast')

img = cv2.imread('itsSurabaya2.jpg',0)
alpha = float(input("Enter alpha value : ")) # koma akan menurunkan kontras, bilangan bulat akan menaikkan kontras

start_time = time.time()
# check if the image grayscale or not
if len(img.shape) > 2:
    # h = img.shape[0]
    # l = img.shape[1]
    # c = img.shape[2]

    img_out = np.zeros(img.shape, img.dtype)

    img_out = cv2.convertScaleAbs(img, alpha=alpha, beta=0)
else:
    # h = img.shape[0]
    # l = img.shape[1]

    img_out = np.zeros(img.shape, img.dtype)

```

```
    img_out = cv2.convertScaleAbs(img, alpha=alpha, beta=0)
time_taken = time.time() - start_time

cv2.imshow('image', img)
cv2.imshow('increased / decreased contrast', img_out)

cv2.imwrite('contrast/contrast_with_opencv.jpg', img_out)
print("%.4f seconds" % time_taken)

with open('contrast/time_taken_contrast_with_opencv.txt', 'a') as f:
    f.write(str(time_taken) + '\n')

cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.2.2 Tanpa menggunakan OpenCV

```
import cv2
import numpy as np
import time
import os

if not os.path.exists('contrast'):
    os.makedirs('contrast')

img = cv2.imread('itsSurabaya2.jpg',0)

alpha = float(input("Enter alpha value : ")) # koma akan menurunkan kontras,
    bilangan bulat akan menaikkan kontras

start_time = time.time()

# check if the image grayscale or not
if len(img.shape) > 2:
    h = img.shape[0]
    l = img.shape[1]
    c = img.shape[2]

    img_out = np.zeros((h,l,c), img.dtype)

    for i in range(h):
        for j in range(l):
                for k in range(c):
                    img_out[i, j, k] = np.clip(alpha * img[i, j, k], 0, 255)
else:
    h = img.shape[0]
    l = img.shape[1]

    img_out = np.zeros((h,l), img.dtype)

    for i in range(h):
        for j in range(l):
            img_out[i, j] = np.clip(alpha * img[i, j], 0, 255)

cv2.imshow('image', img)
cv2.imshow('increased / decreased contrast', img_out)

time_taken = time.time() - start_time

# save image
cv2.imwrite('contrast/contrast_without_opencv.jpg', img_out)

```

```
45 print("%.4f seconds" % time_taken)
46
47 with open('contrast/time_taken_contrast_without_opencv.txt', 'a') as f:
48     f.write(str(time_taken) + '\n')
49
50 cv2.waitKey(0)
51 cv2.destroyAllWindows()
```

4 Perbandingan Hasil

Dalam bagian ini, kami akan membandingkan hasil dari penggunaan OpenCV dan implementasi manual (tanpa OpenCV) untuk penambahan dan pengurangan brightness serta contrast. Perbandingan ini akan memberikan pemahaman yang lebih baik tentang keunggulan dan kelemahan masing-masing metode dalam pemrosesan gambar.

4.1 Penambahan & Pengurangan *brightness*

Berikut merupakan perbandingan penambahan *brightness* menggunakan fungsi *built-in* OpenCV dan operasi manual

Figure 1: Penambahan brightness dengan fungsi *built-in* OpenCV sebesar 120

Figure 2: Penambahan brightness dengan operasi manual sebesar 120

Hasil yang didapatkan dari kedua gambar memiliki kemiripan, namun dalam prosesnya memakan waktu yang berbeda, jika menggunakan fungsi *built-in* OpenCV, waktu yang dihabiskan oleh program akan relatif lebih lama, berikut data dari 10 kali run program penambahan *brightness* sebesar 120 dengan fungsi *built-in* OpenCV dan operasi manual.

Table 1: Waktu fungsi *built-in* OpenCV

Run program ke-	Waktu (s)
1	0.0594
2	0.0725
3	0.0653
4	0.0597
5	0.0799
6	0.0625
7	0.0600
8	0.0661
9	0.0658
10	0.0600

Table 2: Waktu operasi manual

Run program ke-	Waktu (s)
1	0.0064
2	0.0077
3	0.0072
4	0.0071
5	0.0059
6	0.0076
7	0.0074
8	0.0077
9	0.0068
10	0.0080

Selanjutnya, Berikut merupakan perbandingan pengurangan *brightness* menggunakan fungsi *built-in* OpenCV dan operasi manual.

Figure 3: Pengurangan brightness dengan fungsi *built-in* OpenCV sebesar 120

Figure 4: Pengurangan brightness dengan operasi manual sebesar 120

Hasil yang didapatkan dari kedua gambar memiliki kemiripan, namun dalam prosesnya memakan waktu yang berbeda, jika menggunakan fungsi *built-in* OpenCV, waktu yang dihabiskan oleh program akan relatif lebih lama juga, berikut data dari 10 kali run program pengurangan *brightness* sebesar 120 dengan fungsi *built-in* OpenCV dan operasi manual.

Table 3: Waktu fungsi *built-in* OpenCV

Run program ke-	Waktu (s)
1	0.0622
2	0.0536
3	0.0550
4	0.0565
5	0.0612
6	0.0587
7	0.0560
8	0.0611
9	0.0541
10	0.0538

Table 4: Waktu operasi manual

Run program ke-	Waktu (s)
1	0.0031
2	0.0030
3	0.0030
4	0.0031
5	0.0031
6	0.0030
7	0.0031
8	0.0030
9	0.0031
10	0.0030

4.2 Penambahan & Pengurangan *contrast*

Berikut merupakan perbandingan penambahan *contrast* menggunakan fungsi *built-in* OpenCV dan operasi manual:

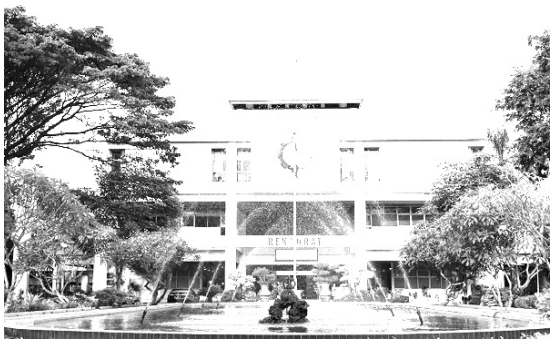

Figure 5: Penambahan *contrast* dengan fungsi *built-in* OpenCV dengan nilai alpha sebesar 3

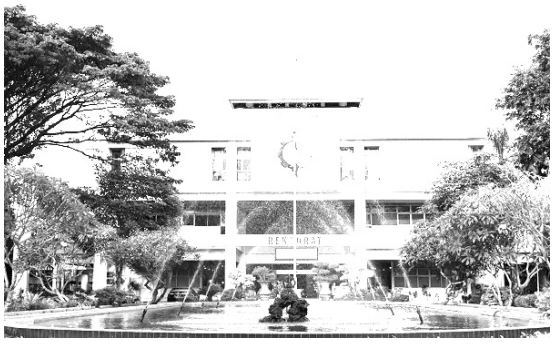

Figure 6: Penambahan *contrast* dengan operasi manual dengan nilai alpha sebesar 3

Hasil yang didapatkan dari kedua gambar memiliki kemiripan, namun dalam prosesnya memakan waktu yang berbeda, jika menggunakan operasi manual, waktu yang dihabiskan oleh program akan relatif lebih lama. Hal ini bisa terjadi karena penggunaan perulangan for sebanyak 2 kali untuk mengiterasi panjang dan lebar gambar, berikut data dari 10 kali run program penambahan *contrast* dengan nilai $\alpha = 3$ dengan fungsi *built-in* OpenCV dan operasi manual.

Table 5: Waktu fungsi *built-in* OpenCV

Run program ke-	Waktu (s)
1	0.000112
2	0.000121
3	0.000121
4	0.000116
5	0.000168
6	0.000171
7	0.000116
8	0.000117
9	0.000120
10	0.000124

Table 6: Waktu operasi manual

Run program ke-	Waktu (s)
1	1.3932
2	1.3640
3	1.3146
4	1.2950
5	1.3089
6	1.3052
7	1.3024
8	1.3531
9	1.3324
10	1.2934

Selanjutnya, berikut merupakan perbandingan pengurangan *contrast* dengan $\alpha = 0.3$ menggunakan fungsi *built-in* OpenCV dan operasi manual.

Figure 7: Pengurangan *contrast* dengan fungsi *built-in* OpenCV dengan nilai alpha sebesar 0.3

Figure 8: Pengurangan *contrast* dengan operasi manual dengan nilai alpha sebesar 0.3

Hasil yang didapatkan dari kedua gambar memiliki kemiripan, namun dalam prosesnya memakan waktu yang berbeda, jika menggunakan operasi manual, waktu yang dihabiskan oleh program akan relatif lebih lama. Hal ini bisa terjadi karena penggunaan perulangan for sebanyak 2 kali untuk mengiterasi panjang dan lebar gambar, berikut data dari 10 kali run program penambahan *contrast* dengan nilai $\alpha = 0.3$ dengan fungsi *built-in* OpenCV dan operasi manual.

Table 7: Waktu fungsi *built-in* OpenCV

Run program ke-	Waktu (s)
1	0.000115
2	0.000115
3	0.000117
4	0.000116
5	0.000114
6	0.000118
7	0.000109
8	0.000119
9	0.000117
10	0.000116

Table 8: Waktu operasi manual

Run program ke-	Waktu (s)
1	1.375323
2	1.382964
3	1.373070
4	1.489619
5	1.378195
6	1.365049
7	1.404826
8	1.380544
9	1.376546
10	1.370240

Selanjutnya, akan dipaparkan kesimpulan dari hasil eksperimen ini pada bab kesimpulan.

5 Kesimpulan

Dari eksperimen yang telah kami lakukan, kami dapat menyimpulkan beberapa hal penting tentang penggunaan OpenCV dan operasi manual dalam mengatur kontras dan kecerahan gambar.

- **Pengaturan Kontras:** Kami menemukan bahwa penggunaan built-in OpenCV untuk mengatur kontras pada gambar lebih cepat daripada operasi manual. Fungsi cv2.convertScaleAbs() memungkinkan kami dengan mudah mengendalikan kontras gambar dengan faktor tertentu, dan hasilnya diperoleh dengan waktu yang lebih singkat.
- **Pengaturan Kecerahan:** Sebaliknya, kami menemukan bahwa operasi manual untuk mengatur kecerahan gambar lebih cepat daripada menggunakan built-in OpenCV. Ini mungkin karena operasi penambahan dan pengurangan kecerahan dapat diimplementasikan secara sederhana tanpa memerlukan pemrosesan yang kompleks.
- **Ukuran File:** Meskipun ada perbedaan dalam waktu eksekusi antara metode pengaturan kontras dan kecerahan, ukuran file gambar tidak berubah. Penggunaan OpenCV atau operasi manual tidak memengaruhi ukuran file gambar.

Dengan demikian, pemilihan metode tergantung pada kebutuhan dan preferensi. Jika kita mengutamakan kecepatan, penggunaan built-in OpenCV mungkin menjadi pilihan yang baik untuk pengaturan kontras. Namun, jika kita memerlukan fleksibilitas dalam pengaturan kecerahan, operasi manual dapat memberikan kendali yang lebih besar. Keduanya memiliki manfaat dan kekurangan masing-masing, dan pemahaman yang baik tentang keduanya dapat membantu kita memilih metode yang sesuai.

6 Lampiran

Program dapat diakses di https://github.com/wannn-one/01-pcv-brightness-contrast