

**TUGAS INDIVIDU
PENGOLAHAN CITRA DIGITAL**



Dosen Pengampu:

Achmad Junaidi, S.Kom, M.Kom

Penulis:

Doding Laswadana (21081010336)

Kelas Pengolahan Citra Digital – C081

PRODI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR SURABAYA

2024

Tugas Individu Pengolahan Citra Digital

1. Memisahkan gambar menjadi 3 layer RGB

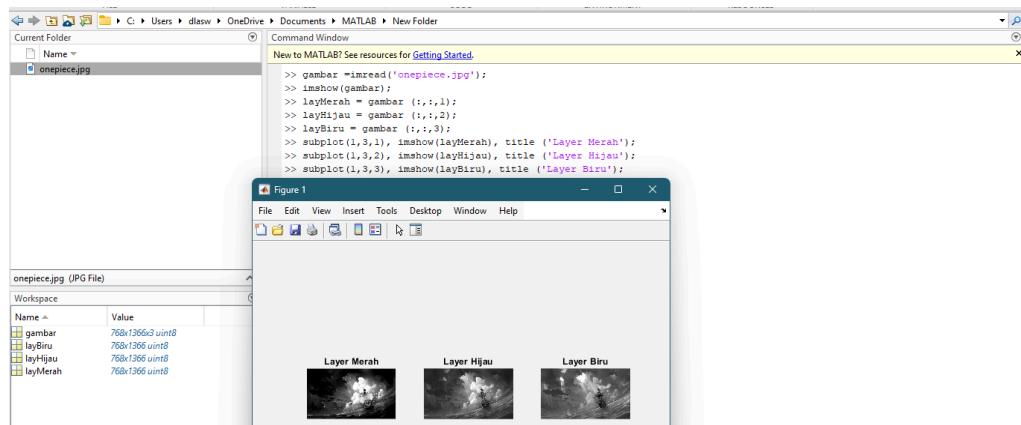
Source code :

```
1. gambar = imread('onepiece.jpg');
2. layMer = gambar (:,:,1);
3. layHijau= gambar (:,:,2);
4. layBiru= gambar (:,:,3);
5. subplot(1,3,1), imshow(layMerah), title ('Layer Merah);
6. subplot(1,3,2), imshow(layHijau), title ('Layer Hijau');
7. subplot(1,3,3), imshow(layBiru), title ('Layer Biru');
```

Penjelasan kode :

Baris ke satu menggunakan function ‘imread’ untuk membaca sebuah gambar. Setelah gambar terbaca maka langkah selanjutnya adalah baris ke 2-4, yaitu mengekstrak gambar ke dalam masing-masing layer R, G, B kemudian ditampung ke dalam variabelnya masing-masing. Setelah gambar terekstrak ke dalam masing-masing layer, kemudian ditampilkan menggunakan function ‘imshow’.

Screenshot Hasil :





Gambar Asli



Layer Red



Layer Green



Layer Blue

2. Grayscale

- Manual

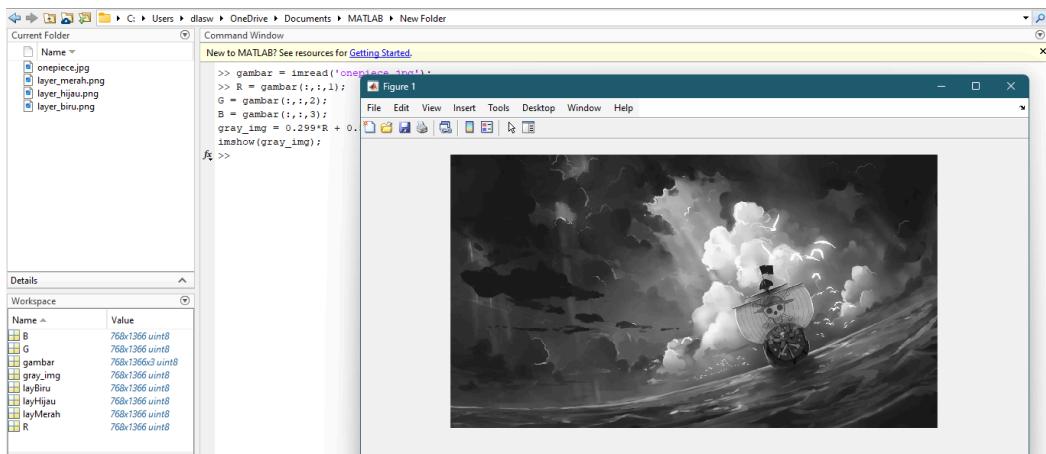
Source code :

```
1. gambar = imread ('onepiece.jpg');
2. R = gambar (:,:,1);
3. G = gambar (:,:,2);
4. B = gambar (:,:,3);
5. gray_img = 0.299*R + 0.587*G + 0.114*B;
6. imshow (gray_img);
```

Penjelasan kode :

Baris ke satu menggunakan function ‘imread’ untuk membaca sebuah gambar. Setelah gambar terbaca maka langkah selanjutnya adalah baris ke 2-4, yaitu mengekstrak gambar ke dalam masing-masing layer RGB kemudian ditampung ke dalam variabelnya masing-masing. Setelah gambar terekstrak ke dalam masing-masing layer, nilai dari masing-masing layer dikalikan dengan nilai bobot di atas untuk agar menghasilkan efek grayscale. kemudian ditampilkan menggunakan function ‘imshow’.

Screenshot hasil :



Gambar grayscale menggunakan cara manual

- Function Matlab

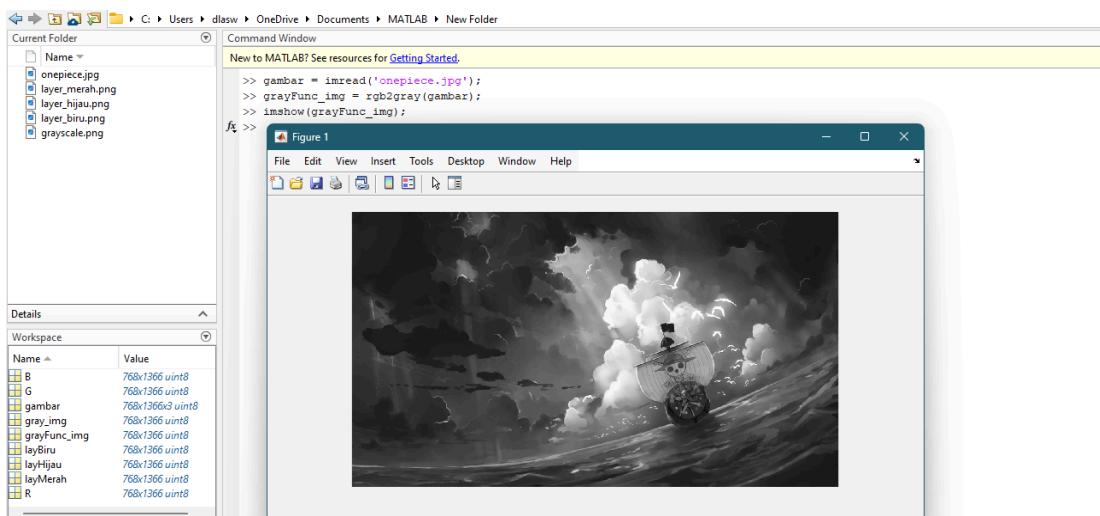
Source code :

```
1. gambar = imread ('mushokuBot.jpg');
2. grayFunc_img = rgb2gray(gambar);
3. imshow(grayFunc_img);
```

Penjelasan kode :

Baris ke satu menggunakan function ‘imread’ untuk membaca sebuah gambar. Baris ke dua memanggil function ‘rgb2array’ untuk mengconvert gambar ke dalam Grayscale secara otomatis, kemudian ditampilkan menggunakan function ‘imshow’.

Screenshot hasil :



Gambar grayscale menggunakan function 'rgb2gray'

3. Negative

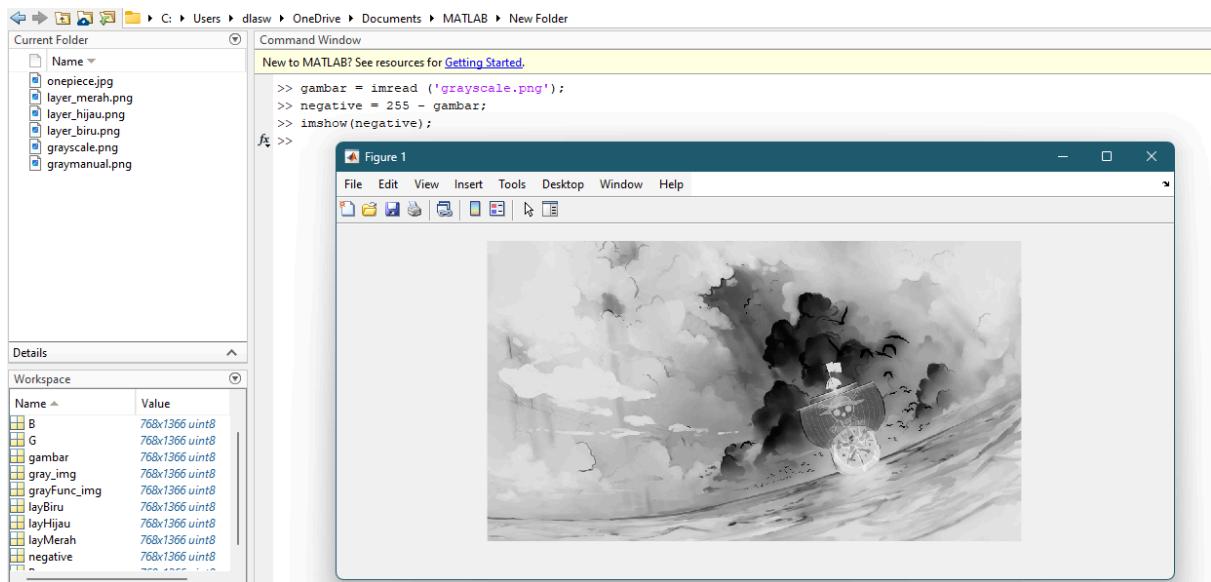
Source code :

1. `gambar = imread ('grayscale.png');`
2. `negative = 255 - gambar;`
3. `imshow(negative);`

Penjelasan kode :

Baris ke satu menggunakan function 'imread' untuk membaca sebuah gambar. Baris ke dua mengurangi 255 dengan setiap nilai pixel sehingga akan menimbulkan efek negatif ($255 - n = \text{efek negatif}$), kemudian ditampilkan menggunakan function 'imshow'.

Screenshot hasil :



Gambar negative

4. Log Transformation

Source code :

1. gambar = imread ('grayscale.png');
2. double_value = im2double(gambar);
3. output1 = 2*log(1+double_value);
4. output2 = 2.5*log(1+double_value);
5. output3 = 3*log(1+double_value);
6. subplot(2,2,1), imshow(gambar); title('gambar grayscale');
7. subplot(2,2,2), imshow(output1); title('c = 2');
8. subplot(2,2,3), imshow(output2); title('c = 2.5');

```

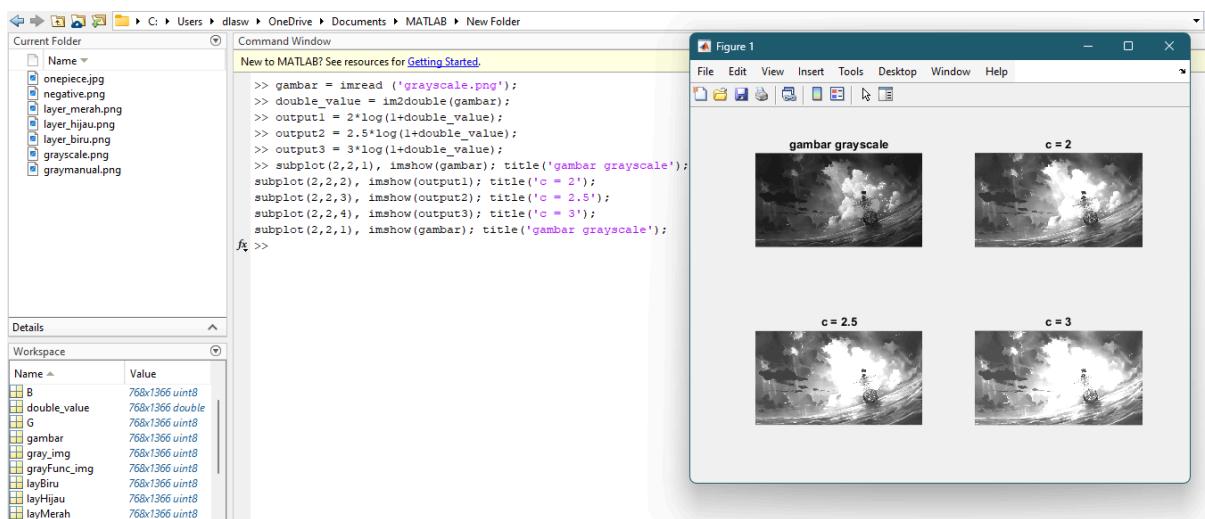
9. subplot(2,2,4), imshow(output3); title('c = 3');
10.subplot(2,2,1), imshow(gambar); title('gambar grayscale');

```

Penjelasan kode :

Baris ke satu menggunakan function ‘imread’ untuk membaca sebuah gambar. Baris kedua mengkonversi gambar dari tipe data uint8 menjadi tipe data double menggunakan fungsi im2double. Hal ini dilakukan untuk memungkinkan operasi matematis yang lebih akurat pada gambar. Baris ketiga hingga kelima menerapkan transformasi logaritmik pada gambar dengan tiga nilai faktor skala berbeda. output1, output2, dan output3 masing-masing dihitung dengan mengalikan hasil transformasi logaritmik dari gambar ($\log(1 + \text{double_value})$) dengan faktor skala 2, 2.5, dan 3. Fungsi log menambahkan 1 pada nilai pixel untuk menghindari $\log(0)$ yang tidak terdefinisi.

Screenshot hasil :



c = 2



c = 2.5



c = 3

5. Powerlaw Gamma

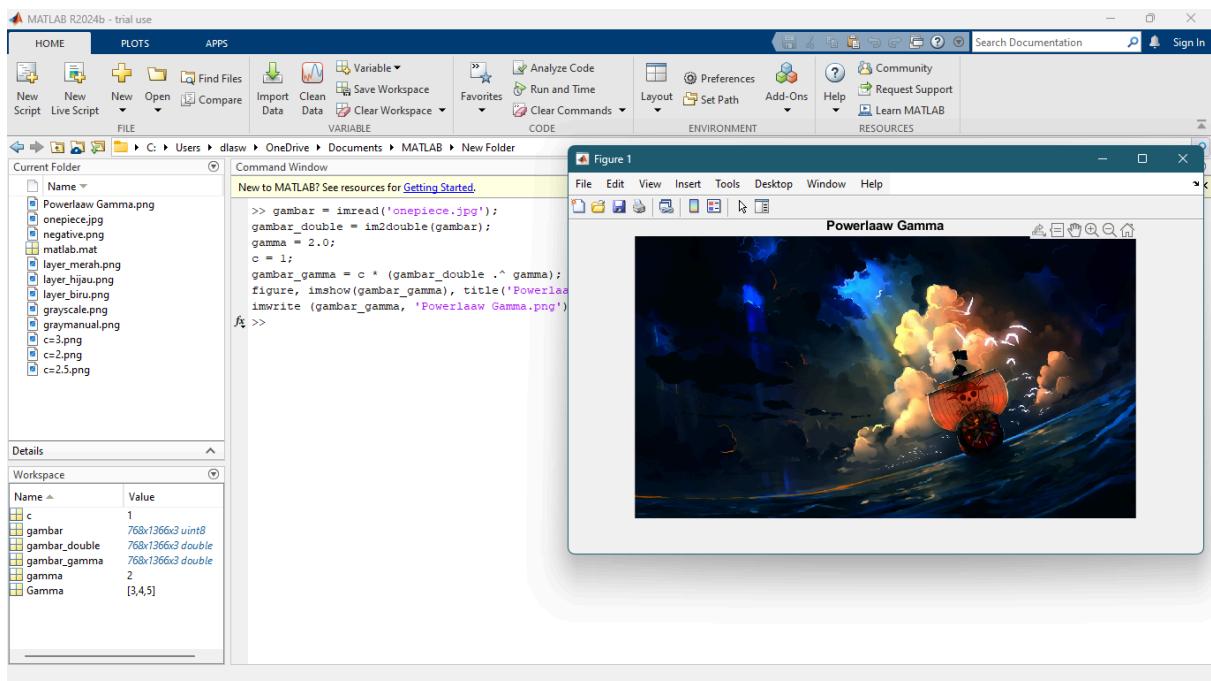
Source code :

```
1. gambar = imread('onepiece.jpg');
2. gambar_double = im2double(gambar);
3. gamma = 2.0;
4. c = 1;
5. gambar_gamma = c * (gambar_double .^ gamma);
6. figure, imshow(gambar_gamma), title('Powerlaaw Gamma');
7. imwrite (gambar_gamma, 'Powerlaaw Gamma.png');
```

Penjelasan kode :

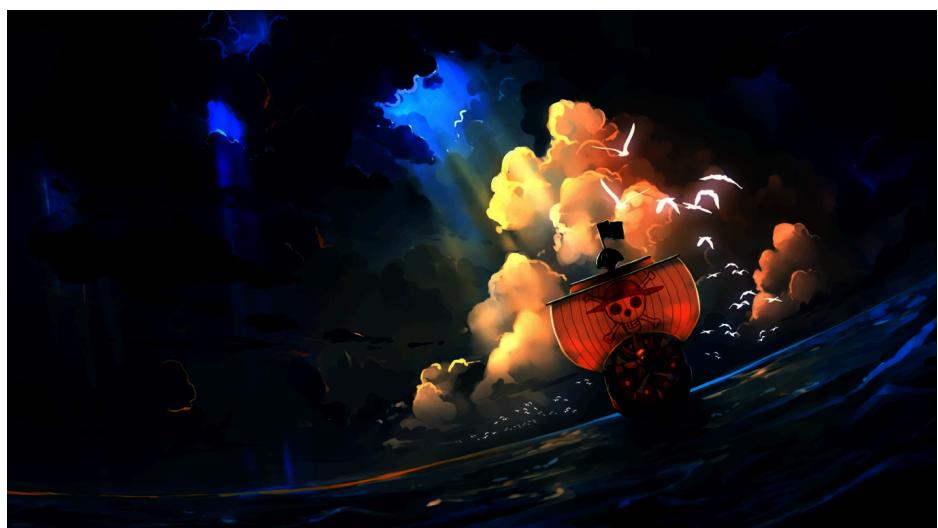
Untuk menerapkan Power-Law Transformation (Gamma Correction) pada gambar RGB di MATLAB, prinsip dasarnya adalah mengubah intensitas piksel menggunakan rumus berikut: Output = $c * (\text{Input})^{\gamma} \cdot c$: konstanta penskalaan (biasanya 1 jika tidak diperlukan penskalaan tambahan). • Input : nilai intensitas piksel gambar. • γ (gamma) : nilai gamma yang mengontrol kecerahan gambar. Nilai $\gamma > 1$ membuat gambar lebih gelap, sedangkan $\gamma < 1$ membuat gambar lebih cerah. Gamma Correction adalah proses yang melibatkan penggunaan eksponen pada nilai intensitas piksel untuk mengatur kecerahan gambar. Nilai gamma menentukan tingkat koreksi, di mana jika gamma lebih besar dari 1 ($\gamma > 1$), gambar akan menjadi lebih gelap, sedangkan jika gamma lebih kecil dari 1 ($\gamma < 1$), gambar akan terlihat lebih cerah. Sebelum melakukan operasi ini, gambar dikonversi terlebih dahulu ke tipe data double agar operasi eksponensial dapat dilakukan dengan presisi yang lebih tinggi. Dalam proses koreksi gamma, operasi eksponensial diterapkan secara element-wise pada setiap piksel di gambar menggunakan operator $.^{\gamma}$ gamma, sehingga setiap piksel mendapatkan efek koreksi sesuai nilai gamma yang diterapkan.

Screenshot hasil :





Gamma = 2



Gamma = 3



Gamma = 4

6. Histogram

Source code :

```
1. gambar = imread('grayscale.jpg');
2.
3. frekuensi = zeros(1, 256);
4.
5. for i = 1: numel(gambar)
6.     intensitas = gambar(i) + 1;
7.     frekuensi(intensitas) = frekuensi(intensitas) + 1;
8. end
9.
10.figure;
11.
12.x = 0:255;
13.bar(x, frekuensi, 'BarWidth', 1);
14.
15.title('Histogram Gambar');
16.xlabel('Nilai Intensitas Pixel');
17.ylabel('Frekuensi');
18.
19.xlim([0 255]);
```

Penjelasan kode :

Langkah pertama adalah menginisialisasi array `frekuensi` untuk menyimpan jumlah pixel untuk setiap nilai intensitas dari 0 hingga 255. Selanjutnya, loop menghitung frekuensi setiap nilai intensitas pixel dan menyimpannya dalam array `frekuensi`. kemudian membuat figure baru dan menggunakan fungsi `bar` untuk menampilkan histogram dengan lebar bar yang diatur agar tidak ada ruang di antara bar. Terakhir, `title`, `xlabel`, dan `ylabel` menambahkan judul dan label pada sumbu, dan `xlim([0 255]);` memastikan sumbu x mencakup rentang 0-255.

Screenshot hasil :

