

Tugas 1 IF4073 Pemrosesan Citra Digital

Semester I Tahun 2024/2025

Image Enhancement



Disusun oleh

Akhmad Setiawan 13521164

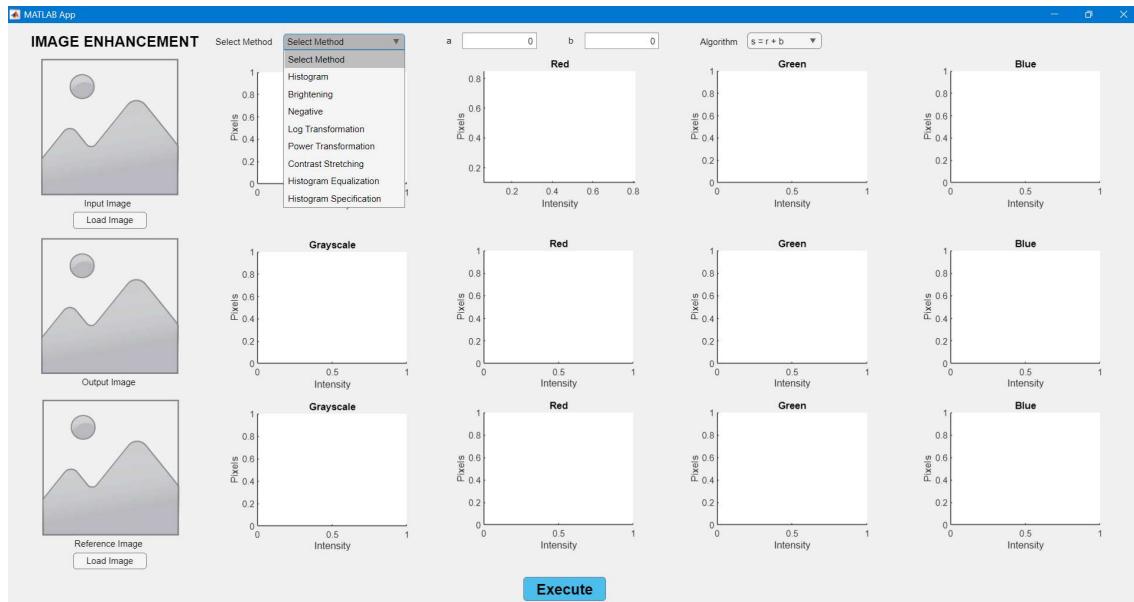
**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024**

Daftar Isi

Daftar Isi.....	2
GUI Program Matlab.....	3
1. Histogram.....	6
1.1. Kode Program Histogram.....	6
1.2. Hasil Eksekusi Program Histogram.....	8
1.3. Analisis Program Histogram.....	11
2. Image Enhancement.....	13
2.1. Kode Program.....	13
- Brightening image.....	13
- Negative image.....	14
- Log Transformation image.....	14
- Power-law Transformation image.....	15
- Contrast Stretching image.....	16
2.2. Hasil Eksekusi Image Enhancement.....	17
- Brightening image.....	17
- Negative image.....	18
- Log Transformation image.....	20
- Power-law Transformation image.....	22
- Contrast Stretching image.....	24
2.3. Analisis Image Enhancement.....	26
1. brightenImage().....	26
2. negativelImage().....	26
3. logImage().....	27
4. powerlImage().....	27
5. contrastStretchlImage.....	28
3. Histogram Equalization.....	29
3.1. Kode Program Histogram Equalization.....	29
3.2. Hasil Eksekusi Histogram Equalization.....	31
3.3. Analisis Histogram Equalization.....	37
4. Histogram Specification.....	39
4.1 Kode Program Histogram Specification.....	39
4.2 Hasil Eksekusi Histogram Specification.....	41
4.3 Analisis Histogram Specification.....	44
Lampiran.....	45

GUI Program Matlab

GUI dibuat dengan menggunakan Matlab App Designer. Berikut adalah komponen-komponen umum yang terdapat pada aplikasi berikut



Terdapat dropdown untuk memilih beberapa metode yang akan dilakukan. Pada beberapa pilihan, field di sebelah kanan akan muncul sehingga pengguna dapat memasukkan nilai variabel untuk konfigurasi pencerahan sesuai dengan nilai yang diinginkan.

Di bawahnya, terdapat 3 placeholder untuk menampilkan gambar input, gambar output, dan gambar referensi (untuk Histogram Specification). Di sebelah masing-masing gambar, terdapat chart untuk menampilkan histogram yang sesuai dengan gambar tersebut. Histogram akan ditampilkan sesuai dengan mode warnanya, apakah grayscale atau berwarna.

Di paling bawah aplikasi, terdapat tombol Execute untuk menjalankan proses image enhancement sesuai dengan pilihan dropdown dan input gambar pengguna. Di bawah ini adalah contoh GUI setelah menerima input dari pengguna.

IMAGE ENHANCEMENT

Select Method

Histogram Specification ▾



Input Image

Load Image

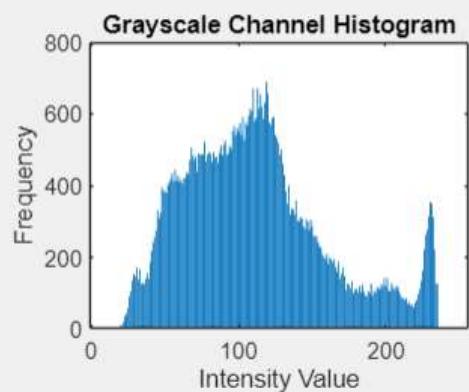
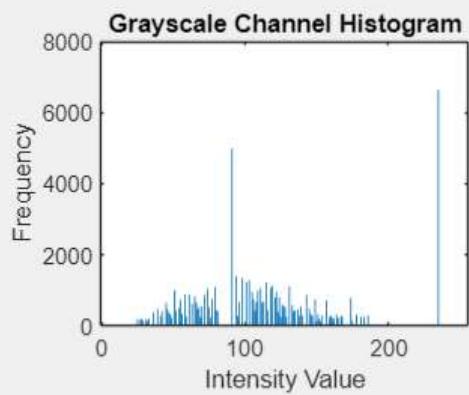
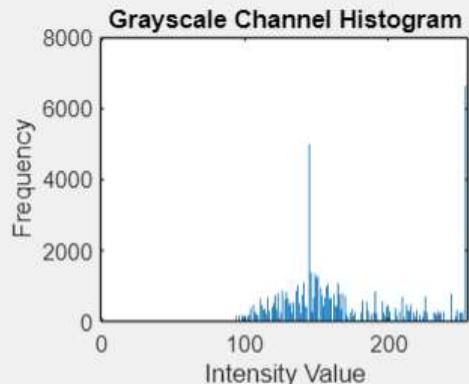


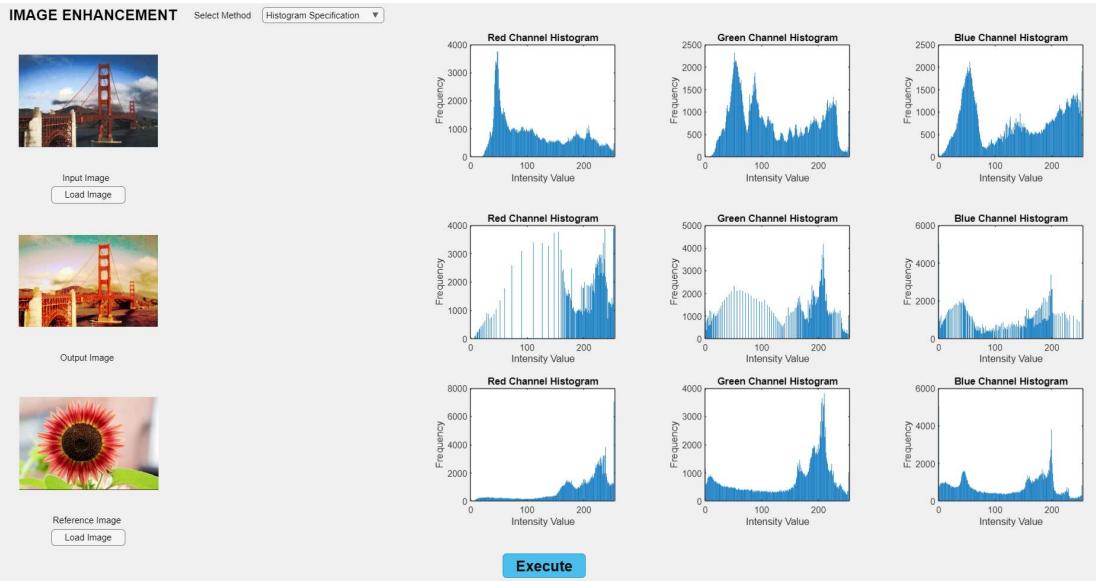
Output Image



Reference Image

Load Image



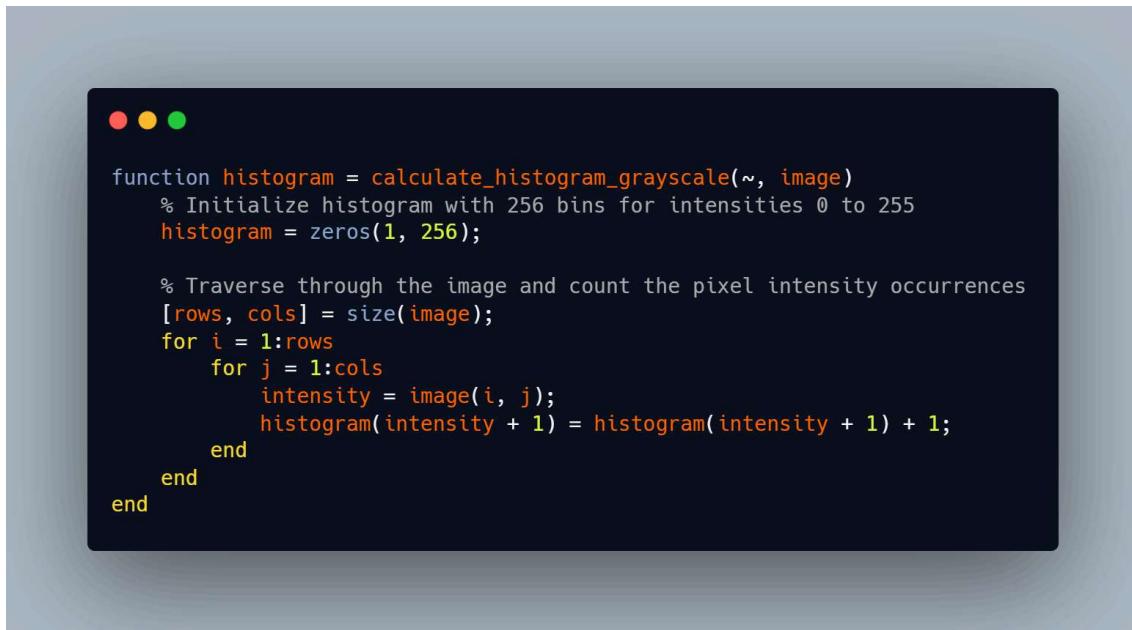


1. Histogram

1.1. Kode Program Histogram

Berikut kode program untuk menghasilkan satu histogram untuk citra *grayscale*.

- Masukan: image
- Luaran: histogram grayscale

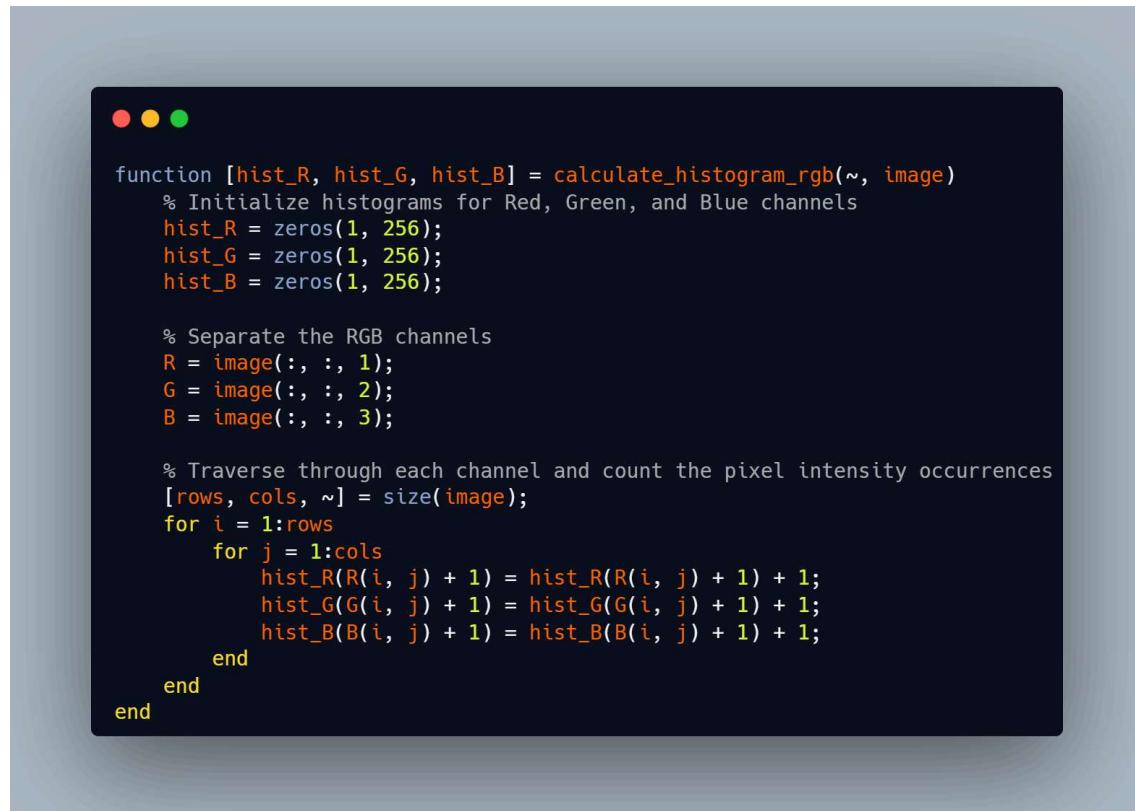


```
function histogram = calculate_histogram_grayscale(~, image)
    % Initialize histogram with 256 bins for intensities 0 to 255
    histogram = zeros(1, 256);

    % Traverse through the image and count the pixel intensity occurrences
    [rows, cols] = size(image);
    for i = 1:rows
        for j = 1:cols
            intensity = image(i, j);
            histogram(intensity + 1) = histogram(intensity + 1) + 1;
        end
    end
end
```

Berikut kode program untuk menghasilkan 3 histogram berwarna (RGB).

- Masukan: image
- Luaran: 3 histogram masing-masing untuk channel Red, Green, dan Blue



```
function [hist_R, hist_G, hist_B] = calculate_histogram_rgb(~, image)
    % Initialize histograms for Red, Green, and Blue channels
    hist_R = zeros(1, 256);
    hist_G = zeros(1, 256);
    hist_B = zeros(1, 256);

    % Separate the RGB channels
    R = image(:, :, 1);
    G = image(:, :, 2);
    B = image(:, :, 3);

    % Traverse through each channel and count the pixel intensity occurrences
    [rows, cols, ~] = size(image);
    for i = 1:rows
        for j = 1:cols
            hist_R(R(i, j) + 1) = hist_R(R(i, j) + 1) + 1;
            hist_G(G(i, j) + 1) = hist_G(G(i, j) + 1) + 1;
            hist_B(B(i, j) + 1) = hist_B(B(i, j) + 1) + 1;
        end
    end
end
```

1.2. Hasil Eksekusi Program Histogram

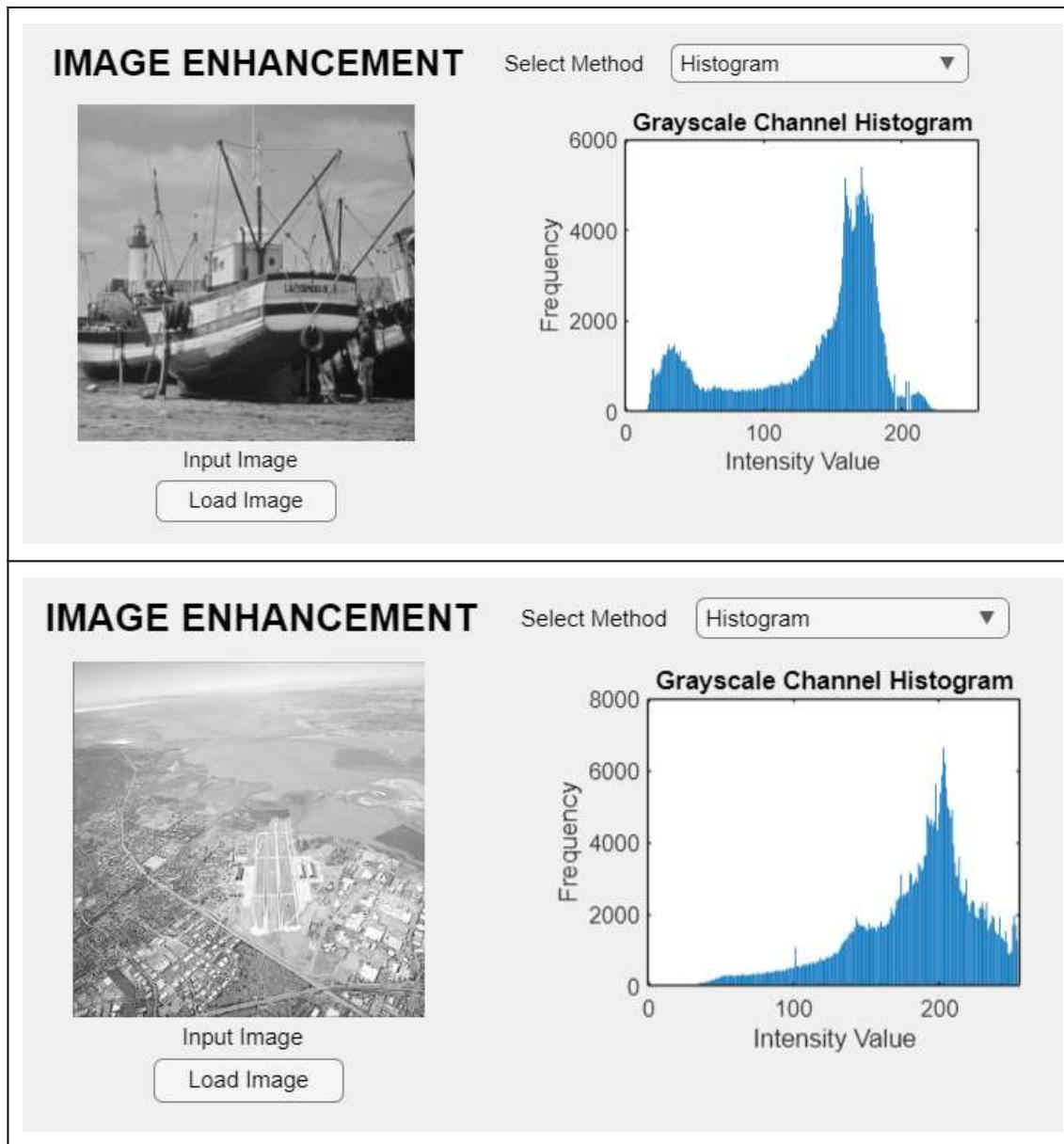
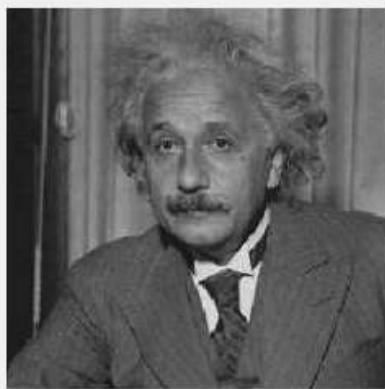


IMAGE ENHANCEMENT

Select Method

Histogram ▾



Input Image

Load Image

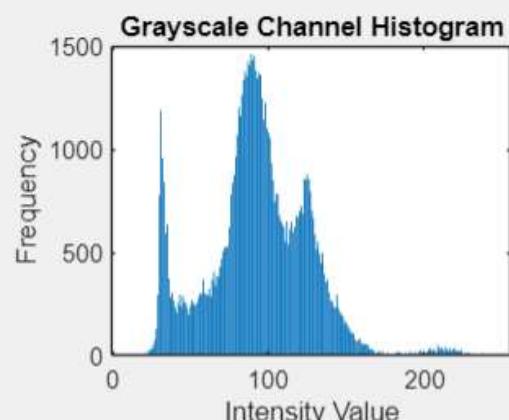


IMAGE ENHANCEMENT

Select Method

Histogram ▾



Input Image

Load Image

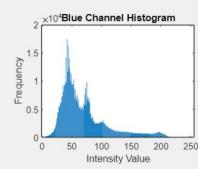
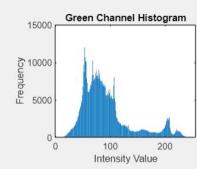
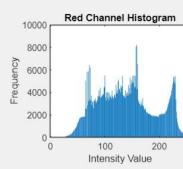


IMAGE ENHANCEMENT

Select Method

Histogram ▾



Input Image

Load Image

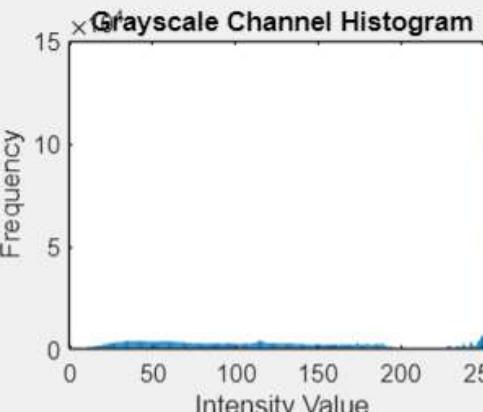


IMAGE ENHANCEMENT

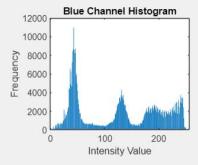
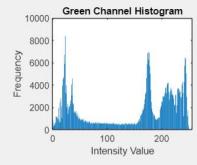
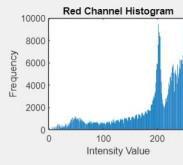
Select Method

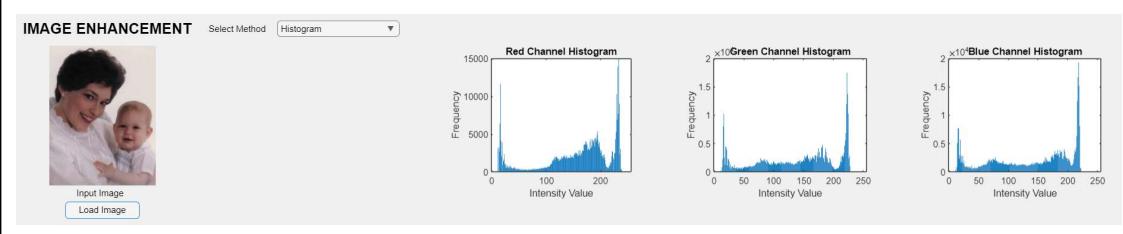
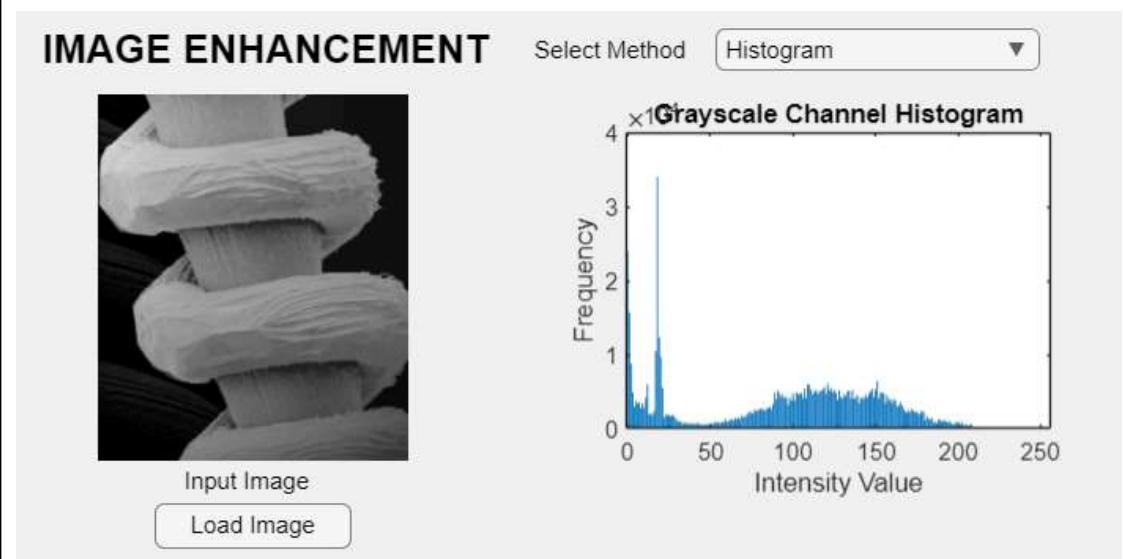
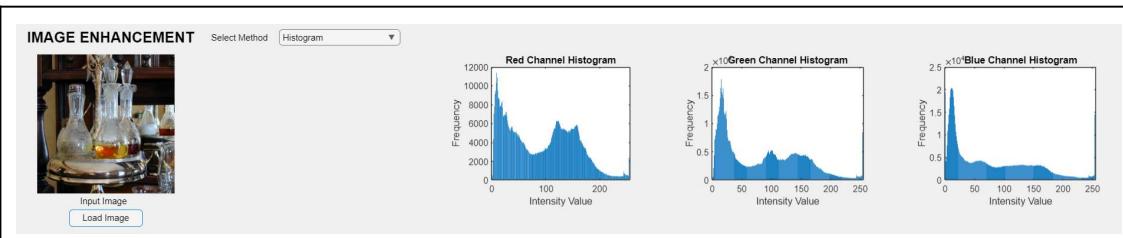
Histogram ▾



Input Image

Load Image





1.3. Analisis Program Histogram

Analisis Umum:

Fungsi untuk menghitung histogram menggunakan pendekatan *brute force* dengan melakukan perulangan untuk setiap piksel dalam gambar. Meskipun sederhana dan mudah dipahami, pendekatan ini dapat menjadi lambat untuk gambar dengan resolusi tinggi karena kompleksitasnya adalah $O(N)$, di mana N adalah jumlah piksel dalam gambar.

Histogram memberikan informasi penting mengenai distribusi intensitas piksel dalam gambar. Untuk gambar grayscale, histogram dapat digunakan untuk analisis kontras atau brightness. Pada gambar RGB, distribusi intensitas dari ketiga saluran dapat digunakan untuk analisis warna.

1. Fungsi calculate_histogram_grayscale()

- Tujuan: Fungsi ini bertugas menghitung histogram dari gambar grayscale.

- Langkah-langkah:

Histogram diinisialisasi sebagai array 1×256 dengan nilai nol. Ini mencerminkan intensitas piksel dari 0 hingga 255, sesuai dengan nilai grayscale (hitam ke putih).

Gambar grayscale diteruskan ke fungsi, di mana ukurannya diambil menggunakan $[rows, cols] = \text{size}(\text{image})$, yang berarti gambar hanya memiliki dua dimensi (baris dan kolom).

Dua loop (for $i = 1:rows$ dan for $j = 1:cols$) digunakan untuk menjelajahi setiap piksel pada gambar. Nilai intensitas piksel tersebut diambil, dan nilai histogram pada indeks yang sesuai ditingkatkan.

Misalnya, jika nilai intensitas piksel adalah 150, maka elemen ke-151 dari histogram ditambah 1. Perlu diperhatikan bahwa MATLAB menggunakan indeks berbasis 1, maka intensitas ditambah 1 (intensity + 1).

- Hasil:

Fungsi ini menghasilkan array histogram yang mengandung jumlah kemunculan dari setiap nilai intensitas piksel dalam gambar grayscale. Bentuk outputnya adalah vektor 1×256 , di mana setiap elemen menyimpan jumlah piksel dengan intensitas tertentu.

2. Fungsi calculate_histogram_rgb()

- Tujuan: Fungsi ini bertugas menghitung histogram untuk gambar berwarna dengan tiga saluran (Red, Green, dan Blue).

- Langkah-langkah:

Tiga histogram diinisialisasi, masing-masing untuk saluran Red (R), Green (G), dan Blue (B) dengan array 1x256 yang semuanya nol.

Gambar dipisahkan menjadi tiga matriks berbeda: R untuk saluran merah, G untuk saluran hijau, dan B untuk saluran biru.

Fungsi kemudian menjelajahi setiap piksel dari ketiga saluran tersebut menggunakan dua loop (for i = 1:rows dan for j = 1:cols).

Untuk setiap piksel, nilai intensitas dari masing-masing saluran (R, G, B) diambil, dan elemen histogram yang sesuai ditambah 1.

Misalnya, jika nilai intensitas piksel pada saluran merah adalah 200, maka elemen ke-201 dari histogram merah (hist_R) ditambah 1, dan hal yang sama dilakukan untuk saluran hijau dan biru.

- Hasil:

Fungsi ini menghasilkan tiga histogram, hist_R, hist_G, dan hist_B, yang mencerminkan distribusi intensitas piksel untuk masing-masing saluran warna. Setiap histogram memiliki 256 elemen yang menggambarkan jumlah piksel dengan intensitas tertentu pada saluran tersebut.

2. Image Enhancement

2.1. Kode Program

- Brightening image



The screenshot shows a MATLAB code editor window with a dark theme. The code is a function named `brightenImage` that takes an application object `app` as input. It reads an input image from the application's input source and gets the selected algorithm from a dropdown menu. It initializes a brightened image by copying the original image. The function then uses a switch statement to apply different brightening algorithms based on the selected algorithm. The cases include linear brightening (`s = r + b`) and linear brightening with scaling (`s = ar + b`). If the selected algorithm is not implemented, it displays an alert message. Finally, it generates a temporary file name, saves the brightened image to it, and sets the output image's `ImageSource` to the temporary file, displaying the brightened image in the application.

```
function brightenImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Get the selected algorithm from the dropdown
    selectedAlgorithm = app.BrighteningAlgorithmDropDown.Value;

    % Initialize the brightened image
    brightenedImage = inputImage; % Start with the original image

    switch selectedAlgorithm
        case 's = r + b'
            % Apply linear brightening: new pixel value = original pixel value + a
            brightenedImage = double(inputImage) + app.BValue;
            % Ensure pixel values are within valid range
            brightenedImage = uint8(clamp(app, brightenedImage, 0, 255));

        case 's = ar + b'
            % Apply linear brightening: new pixel value = a * original pixel value + b
            brightenedImage = app.AValue * double(inputImage) + app.BValue;
            % Ensure pixel values are within valid range
            brightenedImage = uint8(clamp(app, brightenedImage, 0, 255));

        otherwise
            uialert(app.UIFigure, 'Selected algorithm is not implemented.', 'Algorithm Error');
            return;
        end

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the brightened image to the temporary file
    imwrite(brightenedImage, tempFile); % Write the image to a PNG file

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile; % Display the brightened image in the app

    app.showOutputHistogram(brightenedImage);
end
```

- Negative image

```
function negativeImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Calculate the negative image
    negativeImage = uint8(255 - double(inputImage));

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the negative image to the temporary file
    imwrite(negativeImage, tempFile); % Write the image to a PNG file

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile; % Display the negative image in the
app
    app.showOutputHistogram(negativeImage);
end
```

- Log Transformation image

```
function logImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Convert the image to double precision for logarithmic calculations
    doubleImageData = im2double(inputImage);

    % Get the value of c (this will be a scaling constant for log transformation)
    c = app.AValue; % Using AValue as 'c'

    % Apply the log transformation: s = c * log(1 + r)
    logTransformedImage = c * log(1 + doubleImageData);

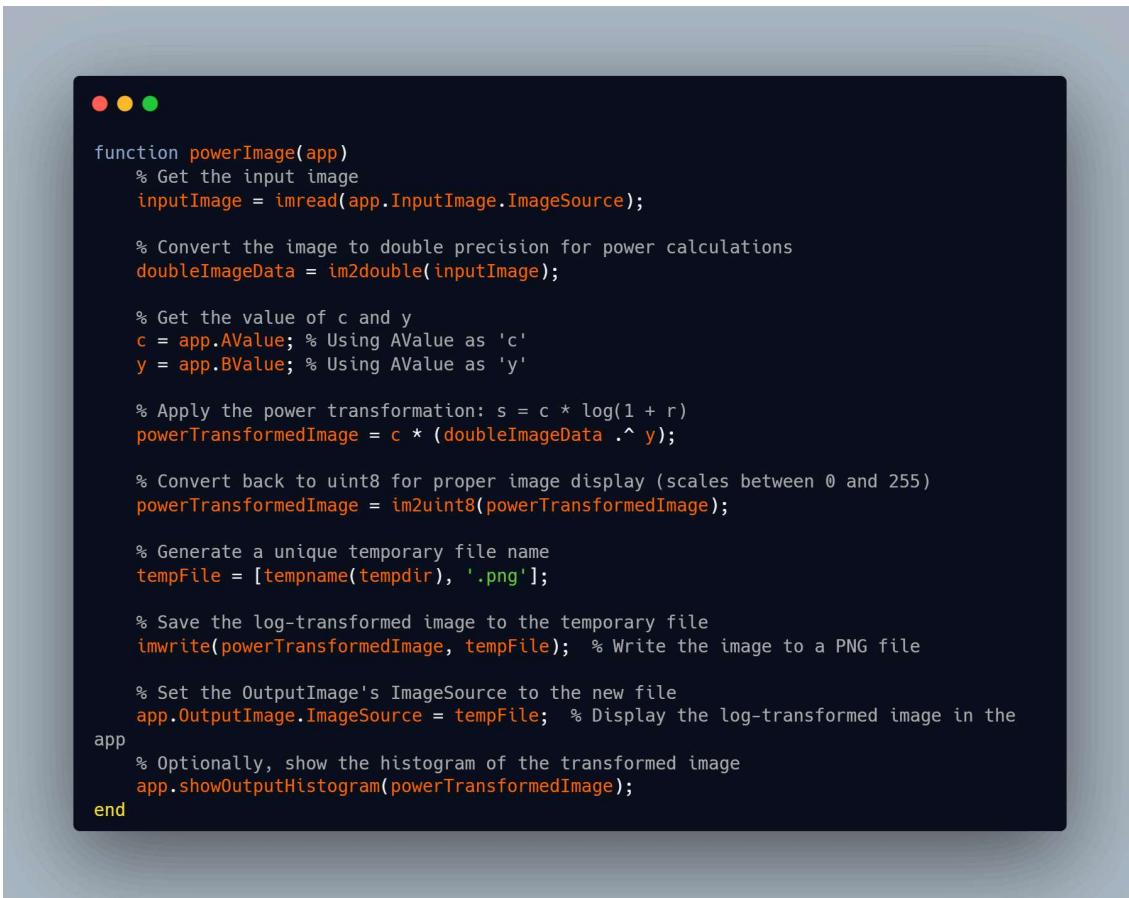
    % Convert back to uint8 for proper image display (scales between 0 and 255)
    logTransformedImage = im2uint8(logTransformedImage);

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the log-transformed image to the temporary file
    imwrite(logTransformedImage, tempFile); % Write the image to a PNG file

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile; % Display the log-transformed image in the
app
    % Optionally, show the histogram of the transformed image
    app.showOutputHistogram(logTransformedImage);
end
```

- Power-law Transformation image



```
function powerImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Convert the image to double precision for power calculations
    doubleImageData = im2double(inputImage);

    % Get the value of c and y
    c = app.AValue; % Using AValue as 'c'
    y = app.BValue; % Using AValue as 'y'

    % Apply the power transformation: s = c * log(1 + r)
    powerTransformedImage = c * (doubleImageData .^ y);

    % Convert back to uint8 for proper image display (scales between 0 and 255)
    powerTransformedImage = im2uint8(powerTransformedImage);

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the log-transformed image to the temporary file
    imwrite(powerTransformedImage, tempFile); % Write the image to a PNG file

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile; % Display the log-transformed image in the
app
    % Optionally, show the histogram of the transformed image
    app.showOutputHistogram(powerTransformedImage);
end
```

- Contrast Stretching image



The screenshot shows a MATLAB application window with a dark theme. The title bar says "Contrast Stretching image". The main area contains the following MATLAB code:

```
function contrastStretchImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Check if the input image is RGB or grayscale
    [~,~, numChannels] = size(inputImage);

    % Initialize the contrast-stretched image
    contrastStretchedImage = zeros(size(inputImage), 'like',
inputImage);
    % Apply contrast stretching for each channel independently
    for c = 1:numChannels
        % Convert to double for processing
        channel = im2double(inputImage(:,:,c));

        % Find the min and max of the current channel
        minValue = min(channel(:));
        maxValue = max(channel(:));

        % Stretch the channel using contrast stretching formula
        stretchedChannel = (channel - minValue) / (maxValue - minValue);

        % Convert back to the original data type
        contrastStretchedImage(:,:,:,c) = im2uint8(stretchedChannel);
    end

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

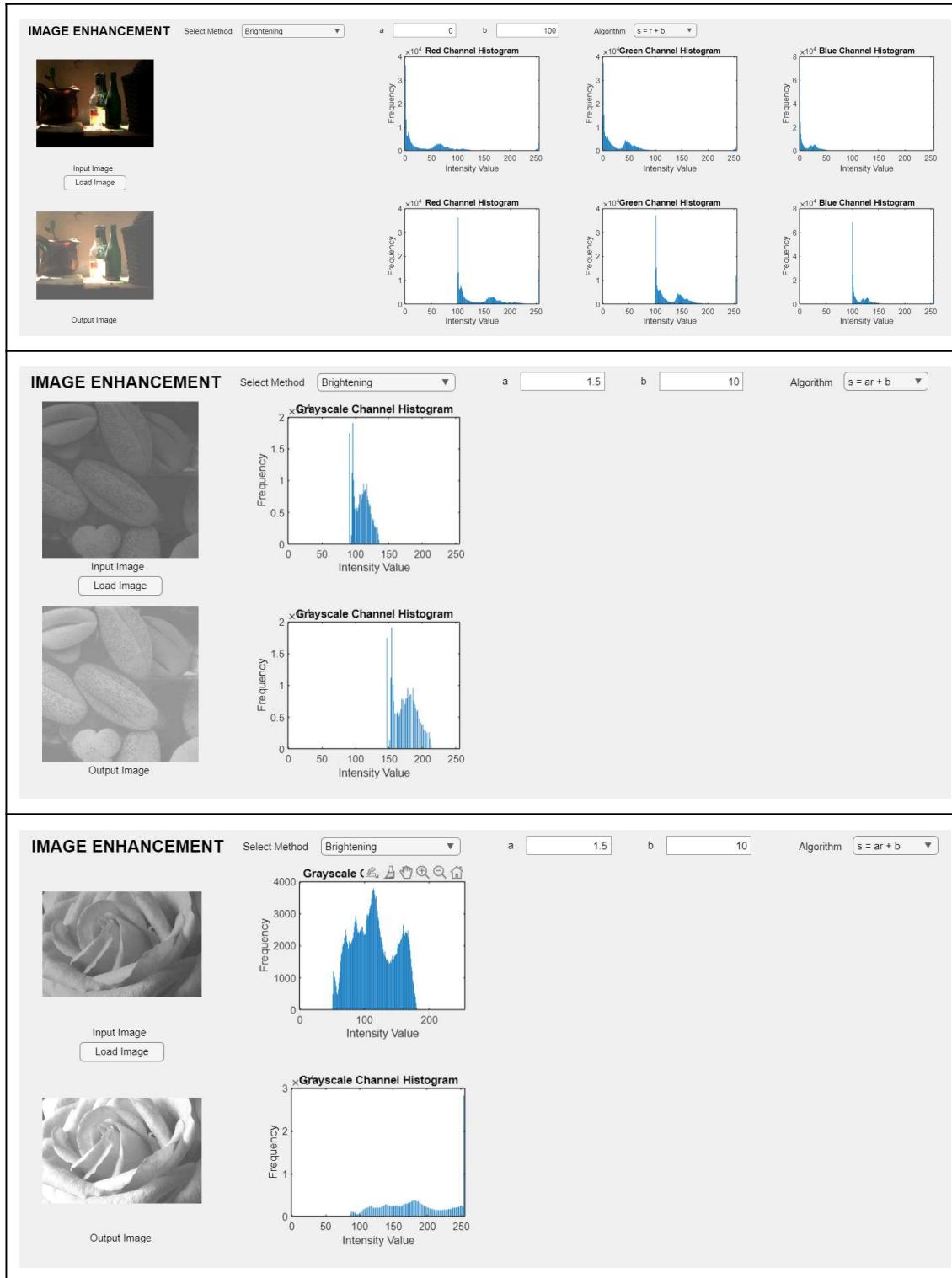
    % Save the contrast-stretched image to the temporary file
    imwrite(contrastStretchedImage, tempFile);

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile;

    % Display histogram for the stretched image
    app.showOutputHistogram(contrastStretchedImage);
end
```

2.2. Hasil Eksekusi Image Enhancement

- Brightening image



- Negative image

IMAGE ENHANCEMENT Select Method

 Input Image

 Output Image

Red Channel Histogram
Frequency vs. Intensity Value (0-255)

Green Channel Histogram
Frequency vs. Intensity Value (0-255)

Blue Channel Histogram
Frequency vs. Intensity Value (0-255)

IMAGE ENHANCEMENT Select Method

 Input Image

 Output Image

Grayscale Channel Histogram
Frequency vs. Intensity Value (0-255)

Grayscale Channel Histogram
Frequency vs. Intensity Value (0-255)

IMAGE ENHANCEMENT

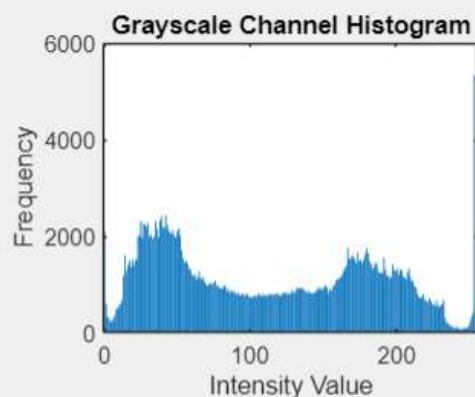
Select Method

Negative

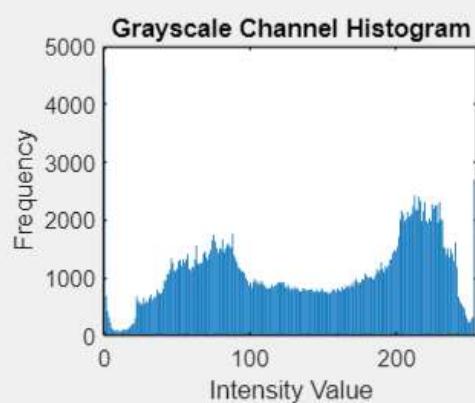


Input Image

Load Image



Output Image



- Log Transformation image

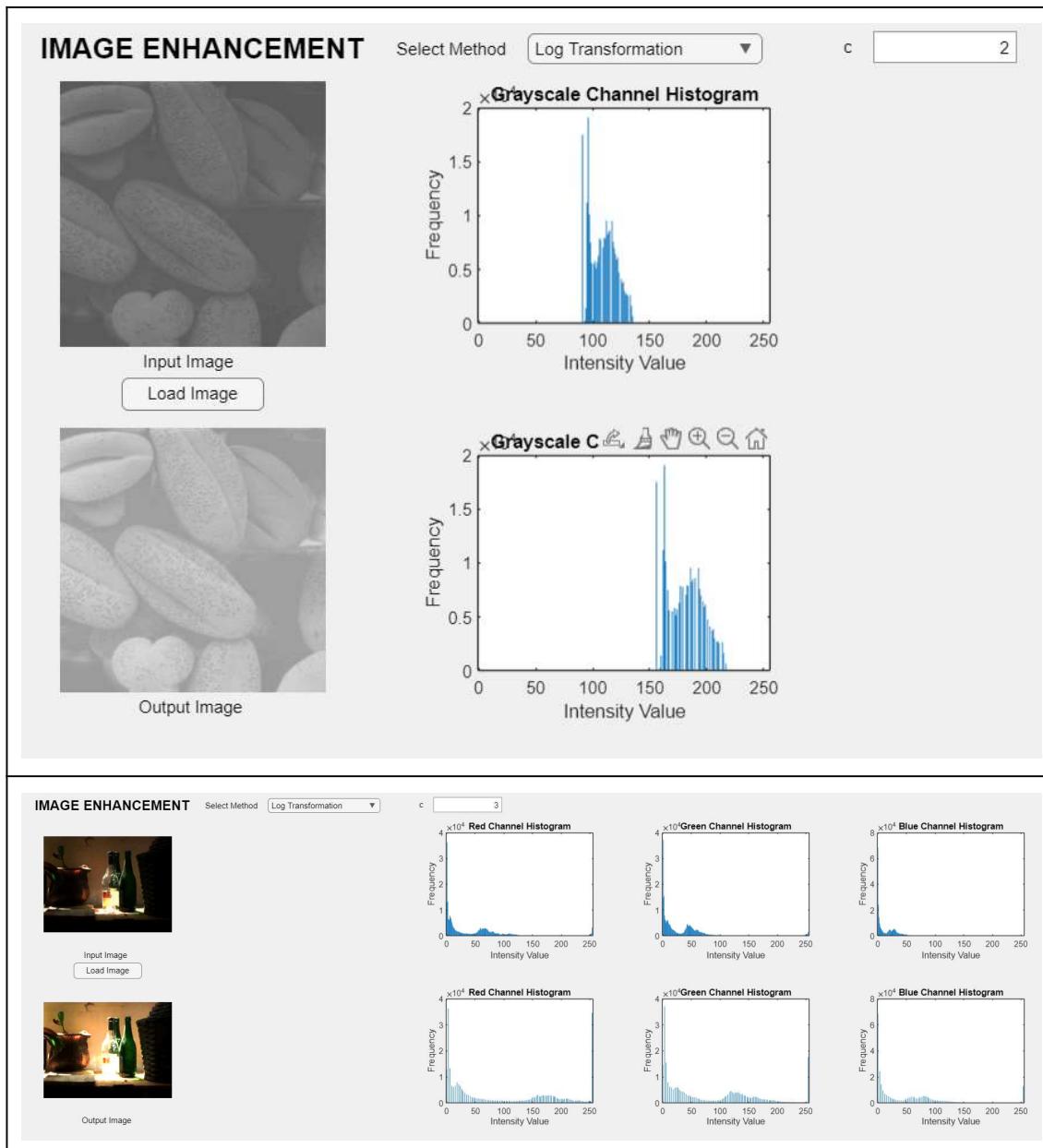


IMAGE ENHANCEMENT

Select Method

Log Transformation ▾

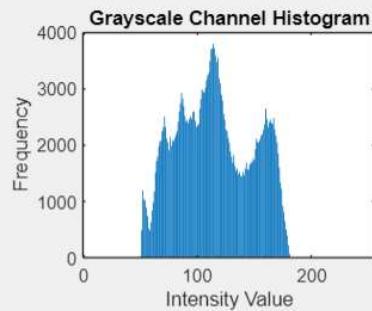
c

2



Input Image

Load Image



Output Image

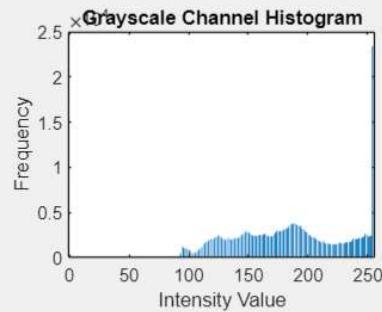
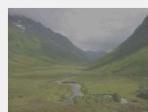


IMAGE ENHANCEMENT

Select Method

Log Transformation ▾



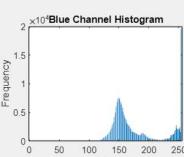
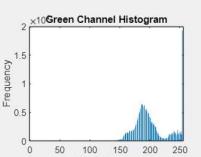
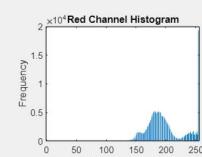
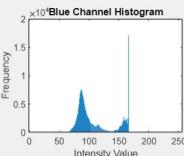
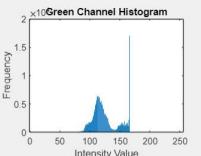
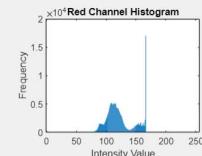
Input Image
Load Image



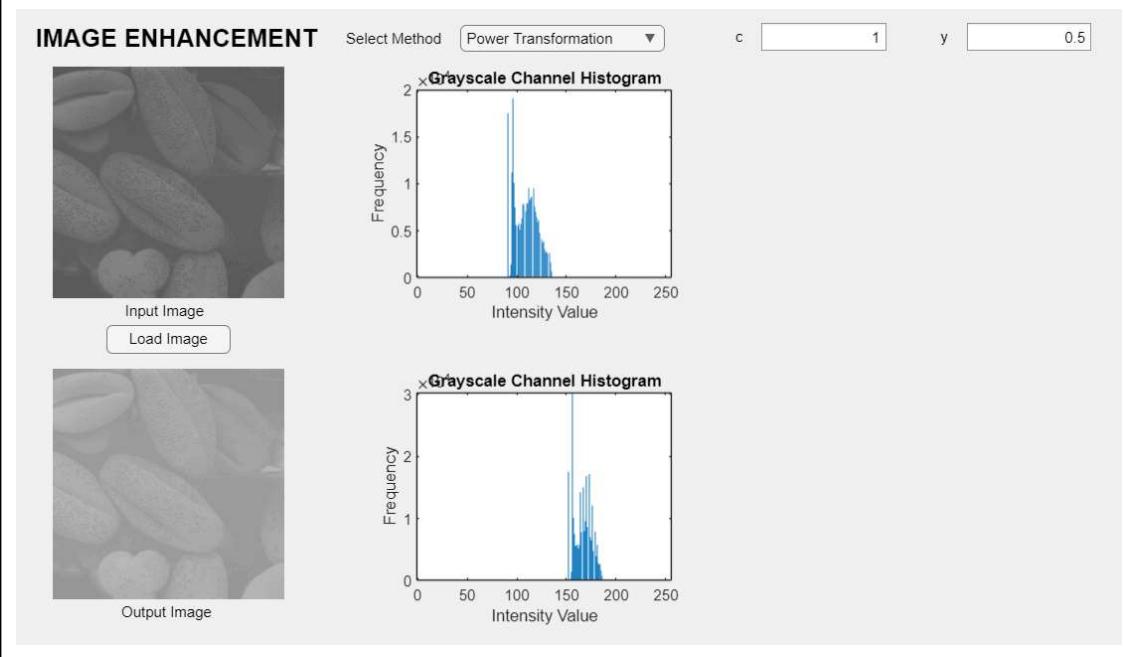
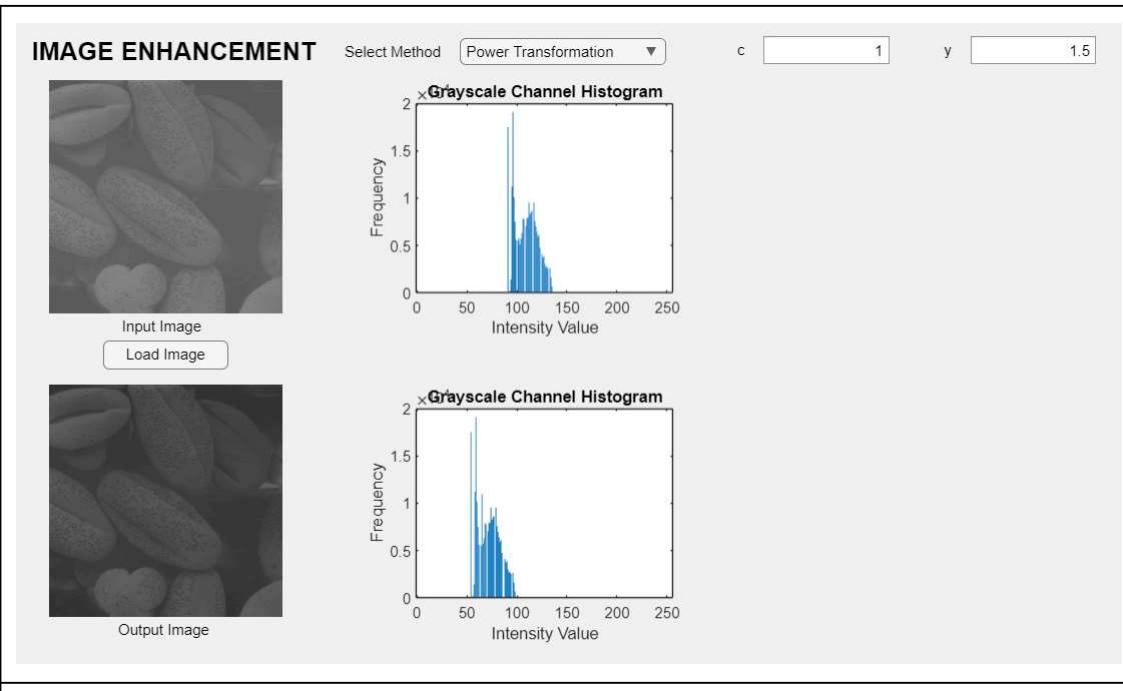
Output Image

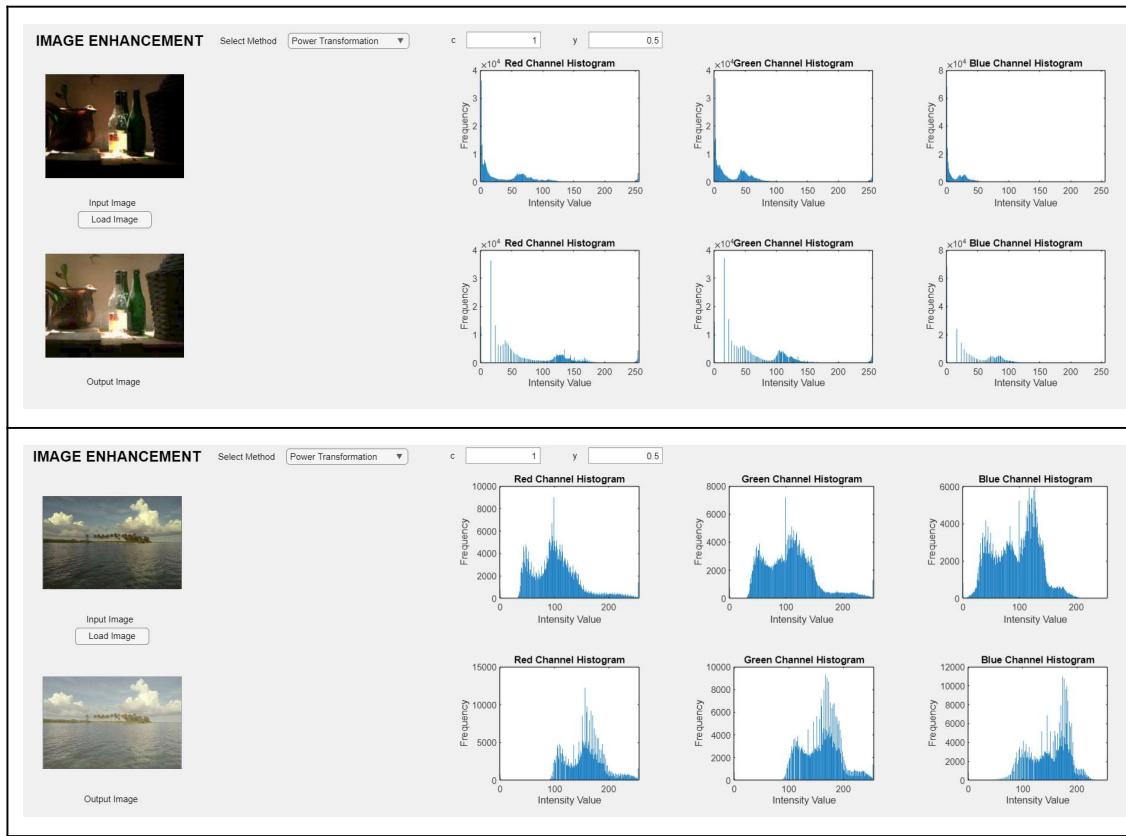
c

2



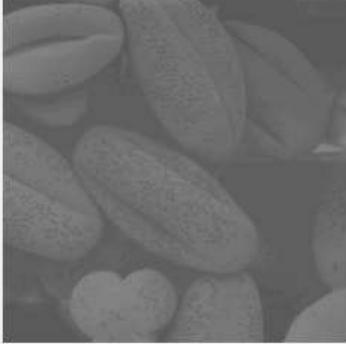
- Power-law Transformation image





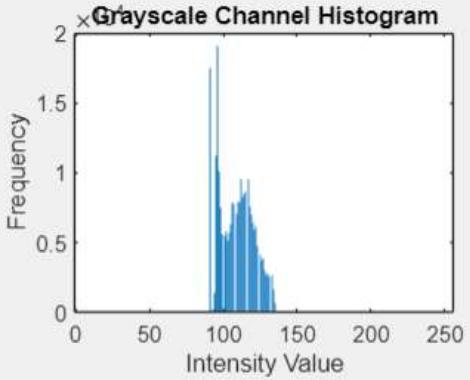
- Contrast Stretching image

IMAGE ENHANCEMENT
Select Method
Contrast Stretching ▾

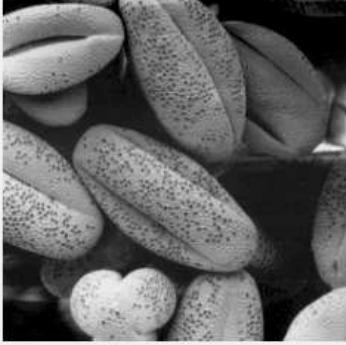


Input Image

Grayscale Channel Histogram

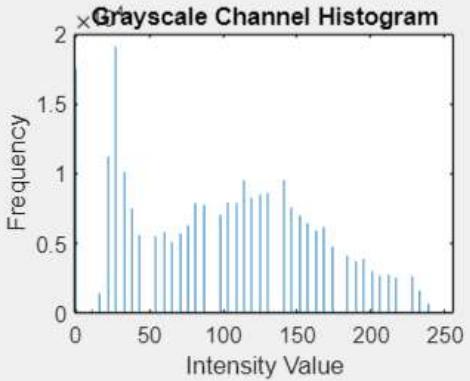


Intensity Value	Frequency
0-25	~0.1
25-50	~0.5
50-75	~1.8
75-100	~0.5
100-125	~0.8
125-150	~0.5
150-175	~0.2
175-200	~0.1
200-225	~0.1
225-250	~0.1



Output Image

Grayscale Channel Histogram



Intensity Value	Frequency
0-25	~1.2
25-50	~1.0
50-75	~0.8
75-100	~0.7
100-125	~0.8
125-150	~0.7
150-175	~0.6
175-200	~0.4
200-225	~0.3
225-250	~0.2

IMAGE ENHANCEMENT
Select Method
Contrast Stretching ▾

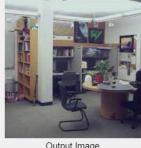


Input Image

$\times 10^4$ Red Channel Histogram

$\times 10^4$ Green Channel Histogram

$\times 10^4$ Blue Channel Histogram

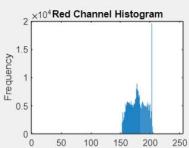


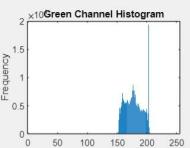
Output Image

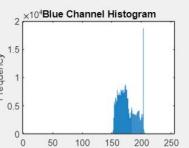
$\times 10^4$ Red Channel Histogram

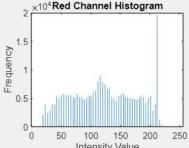
$\times 10^4$ Green Channel Histogram

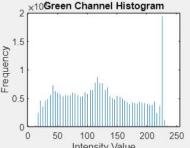
$\times 10^4$ Blue Channel Histogram











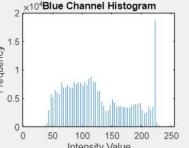
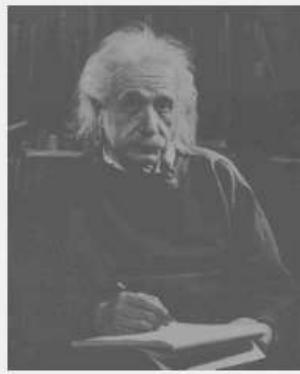


IMAGE ENHANCEMENT

Select Method

Contrast Stretching ▾



Input Image

Load Image



Output Image

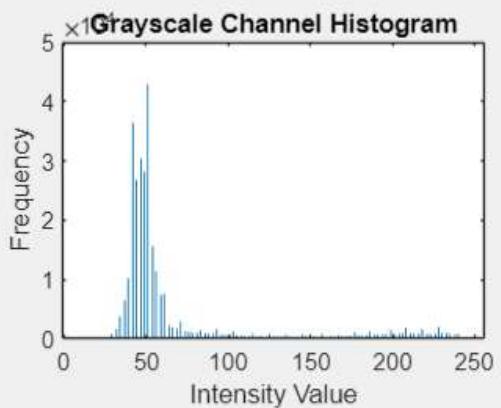
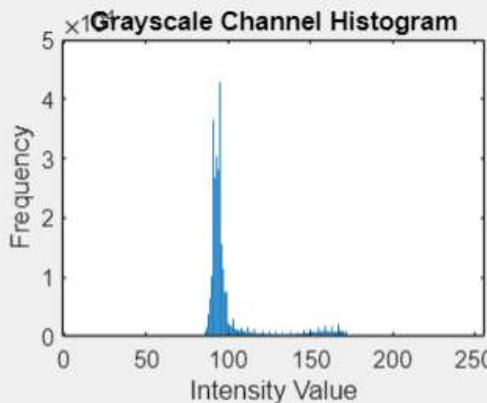


IMAGE ENHANCEMENT

Select Method Contrast Stretching ▾

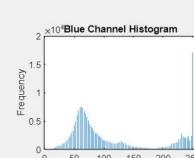
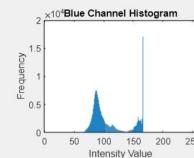
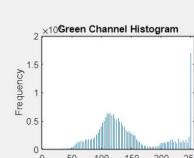
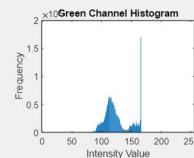
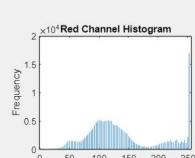
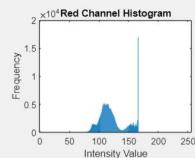


Input Image

Load Image



Output Image



2.3. Analisis Image Enhancement

Analisis Umum:

- Brightening image: Fungsi brightenImage menambah atau mengalikan nilai piksel dengan faktor tertentu untuk meningkatkan kecerahan. Ini berguna untuk mengoreksi gambar yang terlalu gelap.
- Negative image: Fungsi negativelImage mengubah gambar menjadi negatif dengan membalik nilai intensitas, yang bisa digunakan untuk analisis visual atau efek estetis.
- Transformasi Log: Digunakan untuk meningkatkan detail pada area gelap, transformasi ini berguna pada gambar yang memiliki kontras rendah di area gelap.
- Transformasi Pangkat: Fleksibel untuk menyesuaikan kontras dan kecerahan gambar. Ini memungkinkan kontrol lebih halus dibandingkan transformasi linier.
- Peregangan Kontras: Meningkatkan keseluruhan kontras gambar dengan memperluas rentang intensitas piksel. Berguna untuk gambar dengan kontras yang kurang.

Setiap fungsi ini memiliki aplikasi tersendiri dalam pemrosesan citra, tergantung pada karakteristik gambar yang ingin ditingkatkan.

1. **brightenImage()**

- Tujuan: Fungsi ini meningkatkan kecerahan gambar berdasarkan algoritma yang dipilih.
- Algoritma:

Terdapat dua opsi algoritma dalam dropdown:

- $s = r + b$: Nilai piksel baru adalah nilai piksel asli ditambah dengan konstanta BValue. Hasilnya adalah gambar yang lebih terang karena intensitas piksel meningkat.
- $s = ar + b$: Nilai piksel baru adalah AValue dikali dengan nilai piksel asli, lalu ditambah dengan BValue. Ini memberikan kontrol lebih besar atas peningkatan kecerahan.

Clamping: Nilai piksel dibatasi antara 0 dan 255 agar tetap dalam rentang valid.

- Hasil:

Gambar yang dihasilkan akan lebih terang, tergantung pada nilai AValue dan BValue. Histogram dari gambar hasil biasanya akan bergeser ke kanan, mencerminkan peningkatan intensitas piksel secara keseluruhan.

2. **negativelImage()**

- Tujuan: Fungsi ini membuat gambar negatif dari gambar asli.

- Algoritma:
Gambar negatif dihitung dengan mengurangkan nilai piksel asli dari 255. Ini membalikkan intensitas warna sehingga piksel terang menjadi gelap dan sebaliknya.
- Hasil:
Gambar negatif memiliki warna yang terbalik. Histogram dari gambar negatif akan menjadi cerminan dari histogram gambar asli, di mana distribusi intensitas yang semula berada di nilai rendah (gelap) akan berpindah ke nilai tinggi (terang), dan sebaliknya.

3. logImage()

- Tujuan: Fungsi ini menerapkan transformasi logaritmik pada gambar untuk meningkatkan detail pada area gelap.
- Algoritma:
Gambar dikonversi menjadi tipe double agar sesuai untuk perhitungan logaritmik.
Transformasi logaritmik dilakukan dengan rumus: $s = c * \log(1 + r)$, di mana r adalah nilai intensitas asli, dan c adalah konstanta skala (AValue).
Transformasi logaritmik sangat efektif untuk memperjelas detail dalam area yang lebih gelap.
- Hasil:
Gambar hasil akan memiliki peningkatan detail di bagian-bagian yang lebih gelap, sementara area yang lebih terang akan terlihat lebih rata. Histogram gambar cenderung terkonsentrasi pada intensitas yang lebih rendah karena sifat dari fungsi logaritmik yang memperlambat perubahan di bagian nilai tinggi.

4. powerImage()

- Tujuan: Fungsi ini menerapkan transformasi pangkat (power-law transformation) pada gambar.
- Algoritma:
Transformasi ini menggunakan rumus: $s = c * (r^y)$, di mana r adalah nilai intensitas asli, c adalah faktor skala (AValue), dan y adalah eksponen (BValue).
Jika nilai $y < 1$, gambar menjadi lebih terang; jika $y > 1$, gambar menjadi lebih gelap.
- Hasil:
Hasil transformasi pangkat bervariasi tergantung nilai y . Jika $y < 1$, gambar akan tampak lebih terang, dengan histogram bergeser ke kanan. Jika $y > 1$, gambar akan lebih gelap, dan histogram akan bergeser ke kiri.

Transformasi pangkat memungkinkan fleksibilitas yang lebih besar dalam mengubah kontras dan kecerahan gambar dibandingkan dengan transformasi linier.

5. **contrastStretchImage**

- Tujuan: Fungsi ini melakukan peregangan kontras pada gambar untuk memanfaatkan seluruh rentang intensitas (0-255).

- Algoritma:

Gambar diproses per kanal (untuk RGB), di mana nilai piksel setiap kanal diskalakan antara nilai minimum dan maksimum dari kanal tersebut.

Transformasi ini menggunakan rumus: $(\text{channel} - \text{minVal}) / (\text{maxVal} - \text{minVal})$, yang menyebarkan intensitas piksel ke seluruh rentang 0 hingga 255.

- Hasil:

Gambar hasil akan memiliki kontras yang lebih tinggi, dengan distribusi intensitas yang lebih tersebar merata. Histogram gambar setelah peregangan kontras akan menyebar lebih luas, menunjukkan peningkatan dalam rentang dinamis gambar.

3. Histogram Equalization

3.1. Kode Program Histogram Equalization

Fungsi manualHistogramEqualization

- Tujuan:

Fungsi ini bertujuan untuk melakukan histogram equalization pada satu channel gambar (biasanya digunakan untuk gambar grayscale atau per channel RGB). Histogram equalization adalah teknik untuk meningkatkan kontras gambar dengan menyamakan distribusi intensitas pixel dalam gambar.

- Masukan:

Kanal gambar yang diberikan ke fungsi ini dalam bentuk matriks 2D yang berisi nilai intensitas (0-255) setiap pixel.

- Luaran:

Gambar hasil histogram equalization (equalizedChannel), yang merupakan kanal yang sama setelah distribusi intensitasnya disetarakan.



```
function equalizedChannel = manualHistogramEqualization(app, channel)
    % Calculate the histogram for the channel
    histogram = app.calculate_histogram_grayscale(channel);

    % Number of pixels in the channel
    numPixels = numel(channel);

    % Normalize the histogram (convert counts to probabilities)
    probability = histogram / numPixels;

    % Compute the cumulative distribution function (CDF)
    cdf = cumsum(probability);

    % Scale the CDF to the range [0, 255]
    cdf = uint8(255 * cdf);

    % Use the CDF to map the old pixel values to new equalized values
    [rows, cols] = size(channel);
    equalizedChannel = zeros(size(channel), 'like', channel);
    for i = 1:rows
        for j = 1:cols
            equalizedChannel(i, j) = cdf(channel(i, j) + 1);
        end
    end
end
```

Fungsi histogramEqualizationImage.

- Tujuan:

Fungsi ini menerapkan histogram equalization ke seluruh gambar, baik itu gambar grayscale atau RGB. Untuk gambar RGB, fungsi ini melakukan equalization secara independen pada setiap channel.

- Input:

Gambar asli (inputImage) diambil dari app.InputImage.ImageSource yang diberikan dalam format matriks 3D untuk gambar RGB atau matriks 2D untuk gambar grayscale.

- Output:

Gambar hasil histogram equalization yang disimpan dalam file sementara dan ditampilkan di antarmuka aplikasi.



```
function histogramEqualizationImage(app)
    % Get the input image
    inputImage = imread(app.InputImage.ImageSource);

    % Check if the input image is RGB or grayscale
    [~,~,numChannels] = size(inputImage);

    % Initialize the equalized image
    equalizedImage = zeros(size(inputImage), 'like', inputImage);

    % Apply histogram equalization for each channel
    if numChannels == 1
        % Grayscale image
        equalizedImage = manualHistogramEqualization(app, inputImage);
    else
        % RGB image
        for c = 1:numChannels
            channel = inputImage(:,:,c);
            equalizedImage(:,:,:,c) = manualHistogramEqualization(app, channel);
        end
    end

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the equalized image to the temporary file
    imwrite(equalizedImage, tempFile);

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile;

    % Display the histogram for the equalized image
    app.showOutputHistogram(equalizedImage);
end
```

3.2. Hasil Eksekusi Histogram Equalization

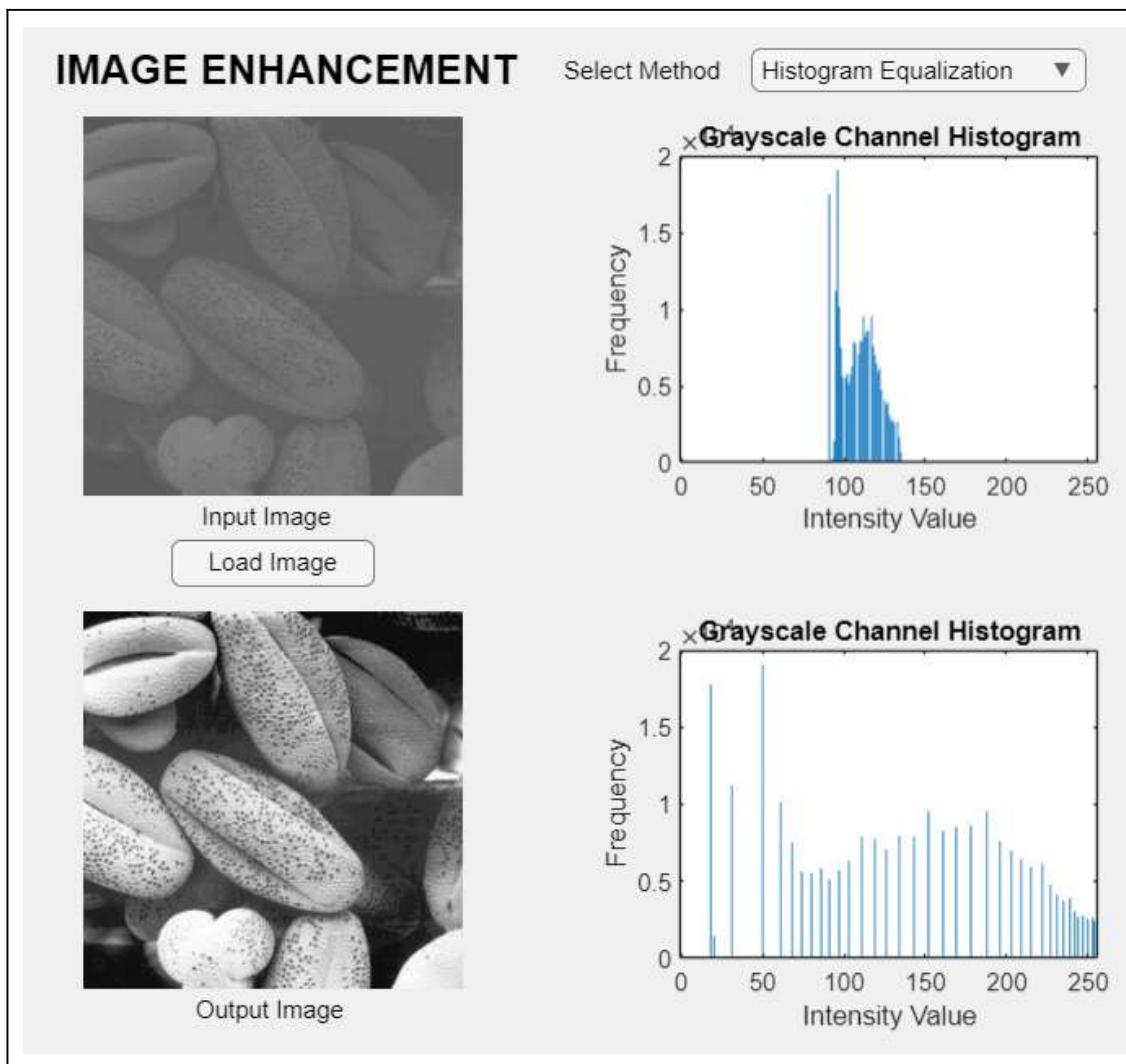


IMAGE ENHANCEMENT

Select Method

Histogram Equalization ▾



Input Image

Load Image



Output Image

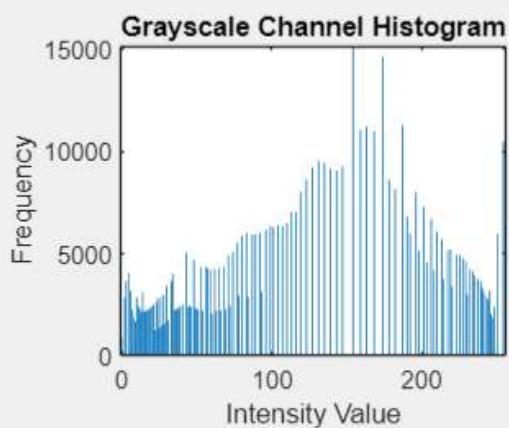
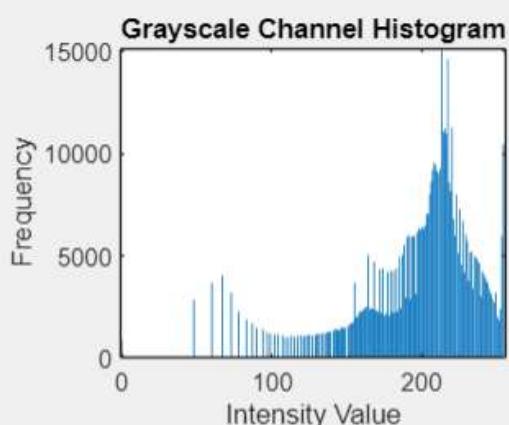


IMAGE ENHANCEMENT

Select Method Histogram Equalization ▾

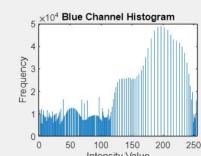
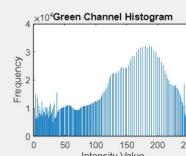
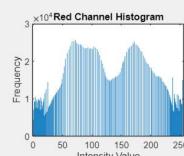
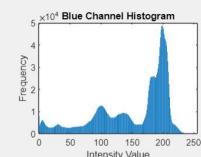
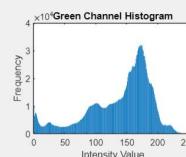
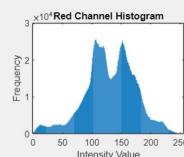


Input Image

Load Image



Output Image



Execute

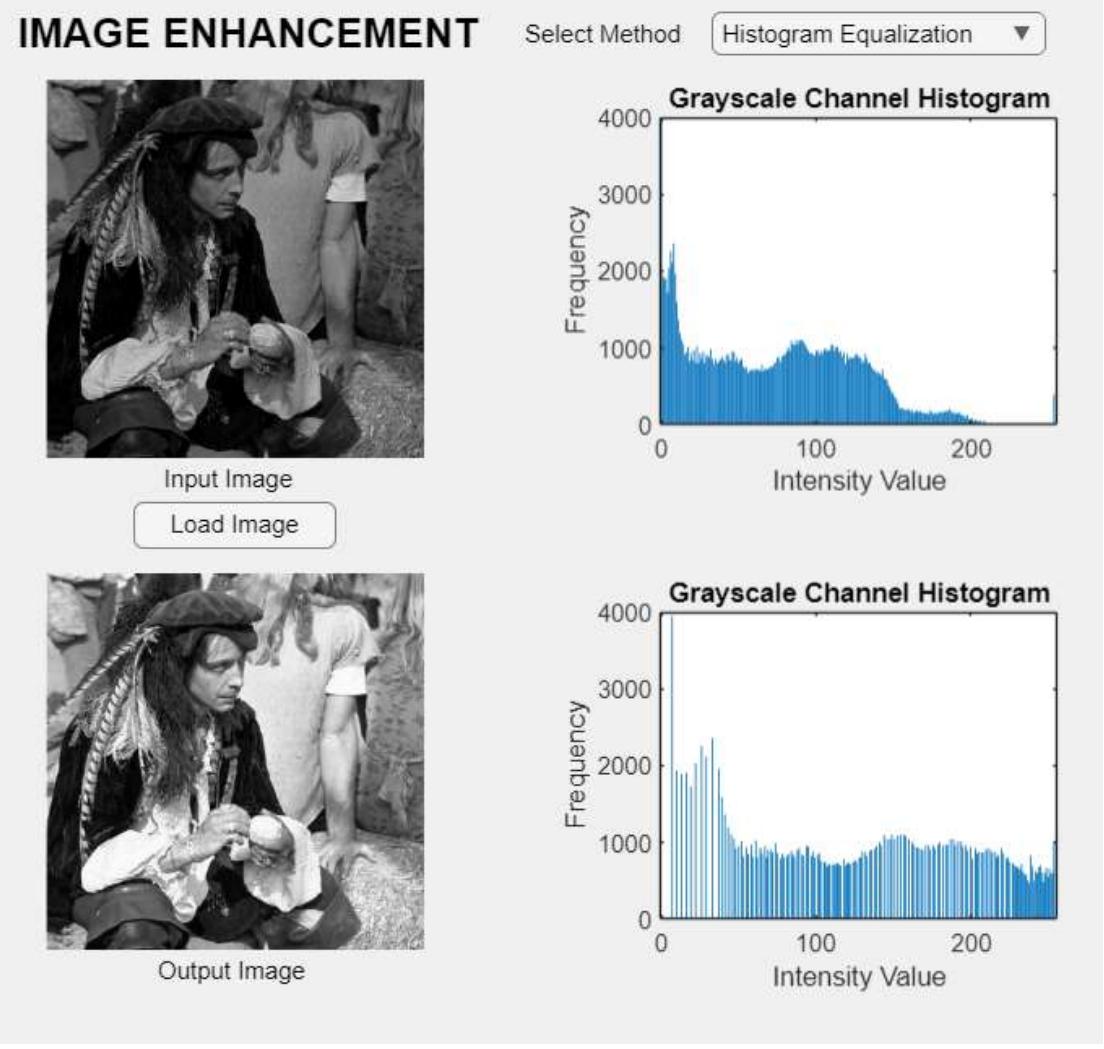
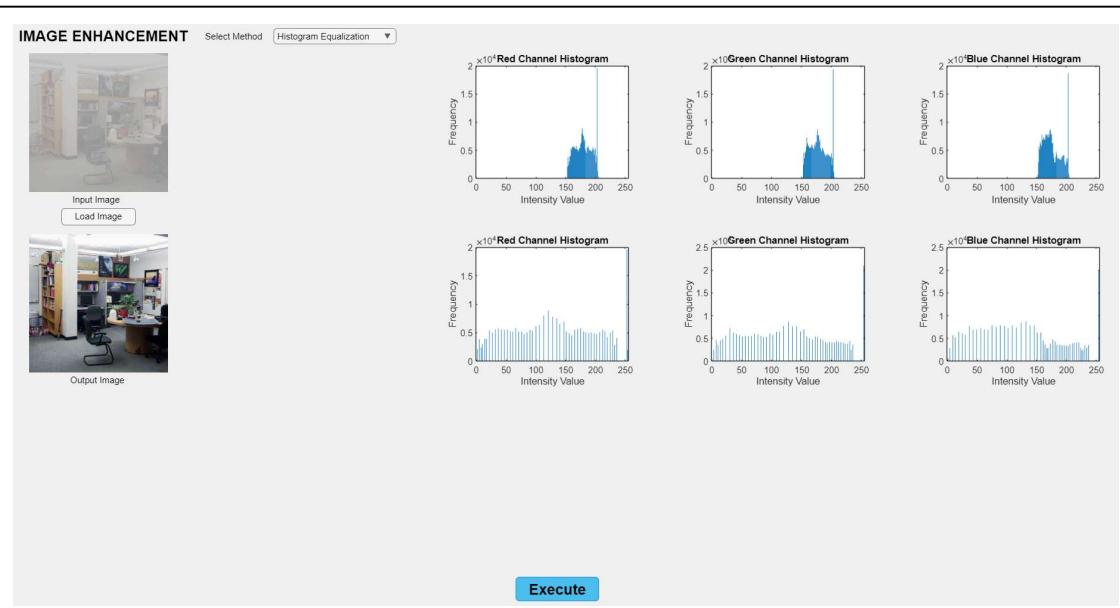


IMAGE ENHANCEMENT

Select Method

Histogram Equalization ▾



Input Image

Load Image



Output Image

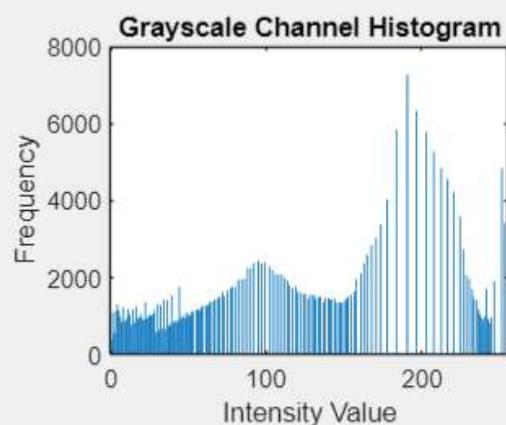
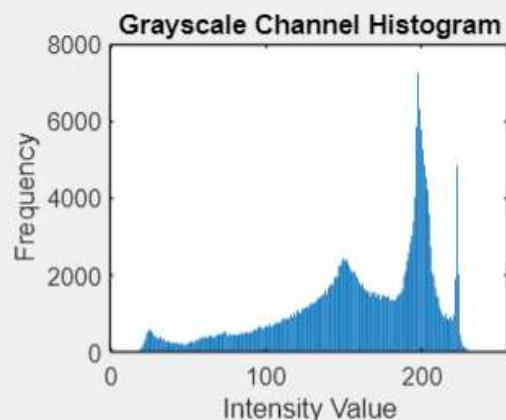


IMAGE ENHANCEMENT

Select Method Histogram Equalization ▾

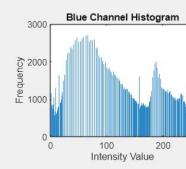
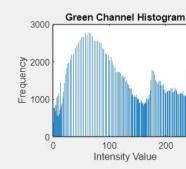
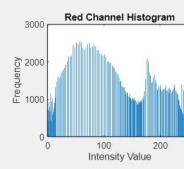
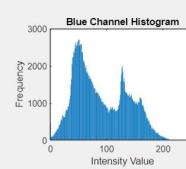
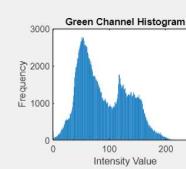
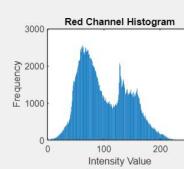


Input Image

Load Image



Output Image



Execute

IMAGE ENHANCEMENT

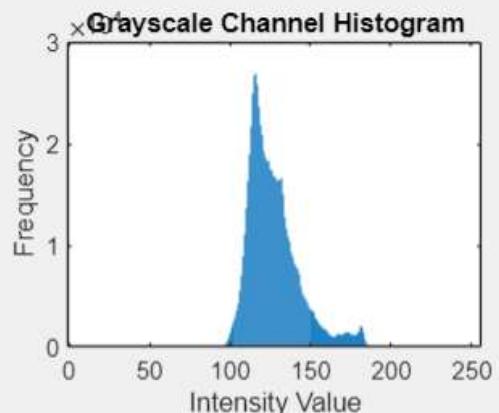
Select Method

Histogram Equalization ▾



Input Image

Load Image



Output Image

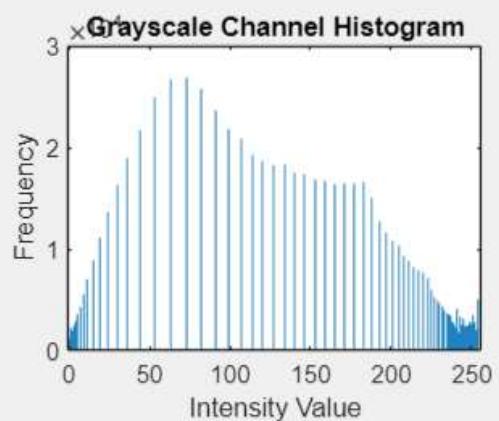


IMAGE ENHANCEMENT

Select Method Histogram Equalization ▾

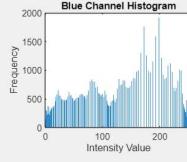
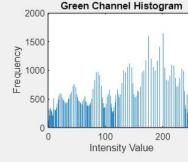
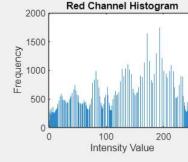
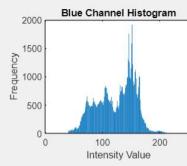
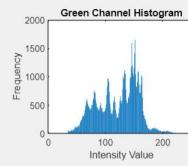
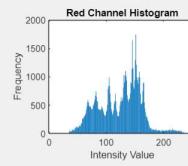


Input Image

Load Image



Output Image



Execute

IMAGE ENHANCEMENT

Select Method

Histogram Equalization ▾

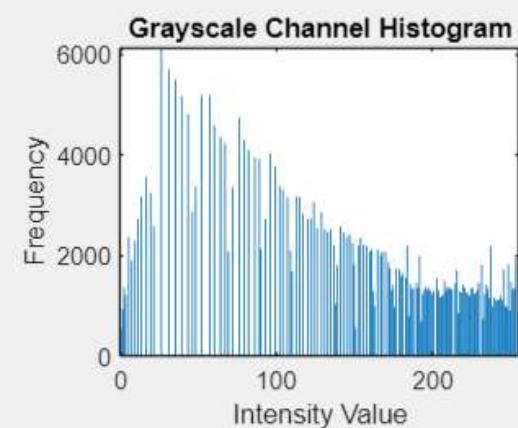
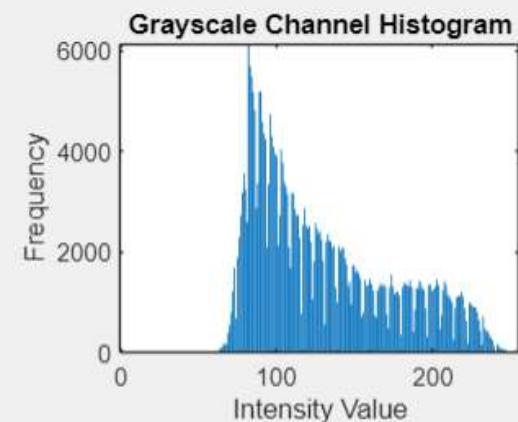


Input Image

Load Image



Output Image



3.3. Analisis Histogram Equalization

Langkah-langkah:

- Membaca Gambar:

Gambar input dibaca menggunakan imread dan disimpan dalam variabel inputImage.

- Identifikasi Jenis Gambar:

Fungsi ini memeriksa apakah gambar adalah gambar grayscale atau RGB dengan melihat jumlah channel pada gambar (numChannels).

- Equalization Berdasarkan Jumlah Channel:

Jika gambar grayscale (hanya 1 channel), maka fungsi manualHistogramEqualization dipanggil untuk langsung menerapkan equalization pada gambar tersebut. Jika gambar RGB (3 channel), histogram equalization diterapkan secara independen pada setiap channel (R, G, dan B) dengan menggunakan manualHistogramEqualization untuk masing-masing channel.

Fungsi app.calculate_histogram_grayscale(channel) menghitung histogram dari channel tersebut. Histogram ini adalah distribusi frekuensi dari nilai intensitas pixel dalam gambar (yaitu, berapa kali setiap nilai intensitas 0-255 muncul dalam channel tersebut).

Setelah histogram diperoleh, histogram tersebut dinormalisasi dengan membaginya dengan jumlah total pixel (numel(channel)), menghasilkan nilai probabilitas. Probabilitas ini menunjukkan seberapa sering setiap intensitas pixel muncul dalam gambar.

Setelah probabilitas dihitung, fungsi cumsum(probability) menghitung CDF, yang merupakan distribusi kumulatif dari probabilitas intensitas pixel. CDF membantu dalam menyertakan distribusi pixel di seluruh gambar.

CDF kemudian diskalakan dari 0 hingga 255 (uint8(255 * cdf)) sehingga dapat digunakan untuk memetakan nilai intensitas lama ke nilai intensitas baru (hasil equalization).

Dalam loop nested (for i, j), setiap nilai intensitas pixel di gambar asli diubah ke nilai baru menggunakan CDF. Nilai lama pixel digunakan sebagai indeks ke dalam CDF, dan hasilnya adalah nilai baru yang sudah disetarakan.

Fungsi ini akhirnya menghasilkan channel baru (equalizedChannel) dengan nilai intensitas pixel yang sudah disetarakan sesuai dengan distribusi kumulatif histogramnya.

- Menyimpan dan Menampilkan Gambar:

Gambar yang sudah di-equalize disimpan sebagai file gambar sementara menggunakan imwrite dan kemudian ditampilkan dalam antarmuka aplikasi dengan mengubah app.OutputImage.ImageSource.

- Menampilkan Histogram:

Setelah proses equalization, histogram gambar hasil ditampilkan dengan app.showOutputHistogram.

Kedua fungsi ini efektif dalam meningkatkan kontras gambar melalui histogram equalization, baik untuk gambar grayscale maupun RGB.

Fungsi manualHistogramEqualization memberikan hasil yang baik dengan kontrol penuh atas perhitungan histogram dan transformasi intensitas pixel, tetapi ada potensi peningkatan performa terutama untuk gambar besar.

Fungsi histogramEqualizationImage menyediakan pendekatan yang lebih umum, yang menangani gambar dengan jumlah channel berbeda dengan memanggil fungsi manual histogram equalization pada setiap channel secara independen.

4. Histogram Specification

4.1 Kode Program Histogram Specification

Fungsi histogramSpecificationImage

- Tujuan:

Fungsi ini bertujuan untuk melakukan histogram specification. Histogram specification (atau histogram matching) adalah teknik untuk memodifikasi histogram gambar input agar menyerupai histogram gambar referensi.

- Masukan:

Dua gambar, yaitu gambar input (inputImage) dan gambar referensi (referenceImage), yang diberikan sebagai input dalam aplikasi. Kedua gambar bisa berupa gambar grayscale atau RGB.

- Luaran:

Gambar hasil histogram specification (specifiedImage), yang memiliki distribusi intensitas yang disesuaikan dengan distribusi gambar referensi.



```
function histogramSpecificationImage(app)
    % Get the input and reference images
    inputImage = imread(app.InputImage.ImageSource);
    referenceImage = imread(app.ReferenceImage.ImageSource);

    % Check if the input image is RGB or grayscale
    [~, ~, numChannels] = size(inputImage);

    % Initialize the output image
    specifiedImage = zeros(size(inputImage), 'like', inputImage);

    if numChannels == 1
        % Grayscale histogram specification
        specifiedImage = manualHistogramSpecification(app, inputImage, referenceImage);
    else
        % RGB histogram specification
        for c = 1:numChannels
            inputChannel = inputImage(:, :, c);
            refChannel = referenceImage(:, :, c);
            specifiedImage(:, :, c) = manualHistogramSpecification(app, inputChannel, refChannel);
        end
    end

    % Generate a unique temporary file name
    tempFile = [tempname(tempdir), '.png'];

    % Save the specified image to the temporary file
    imwrite(specifiedImage, tempFile);

    % Set the OutputImage's ImageSource to the new file
    app.OutputImage.ImageSource = tempFile;

    % Display the histogram for the specified image
    app.showOutputHistogram(specifiedImage);
end
```

Fungsi manualHistogramSpecification

- Tujuan:

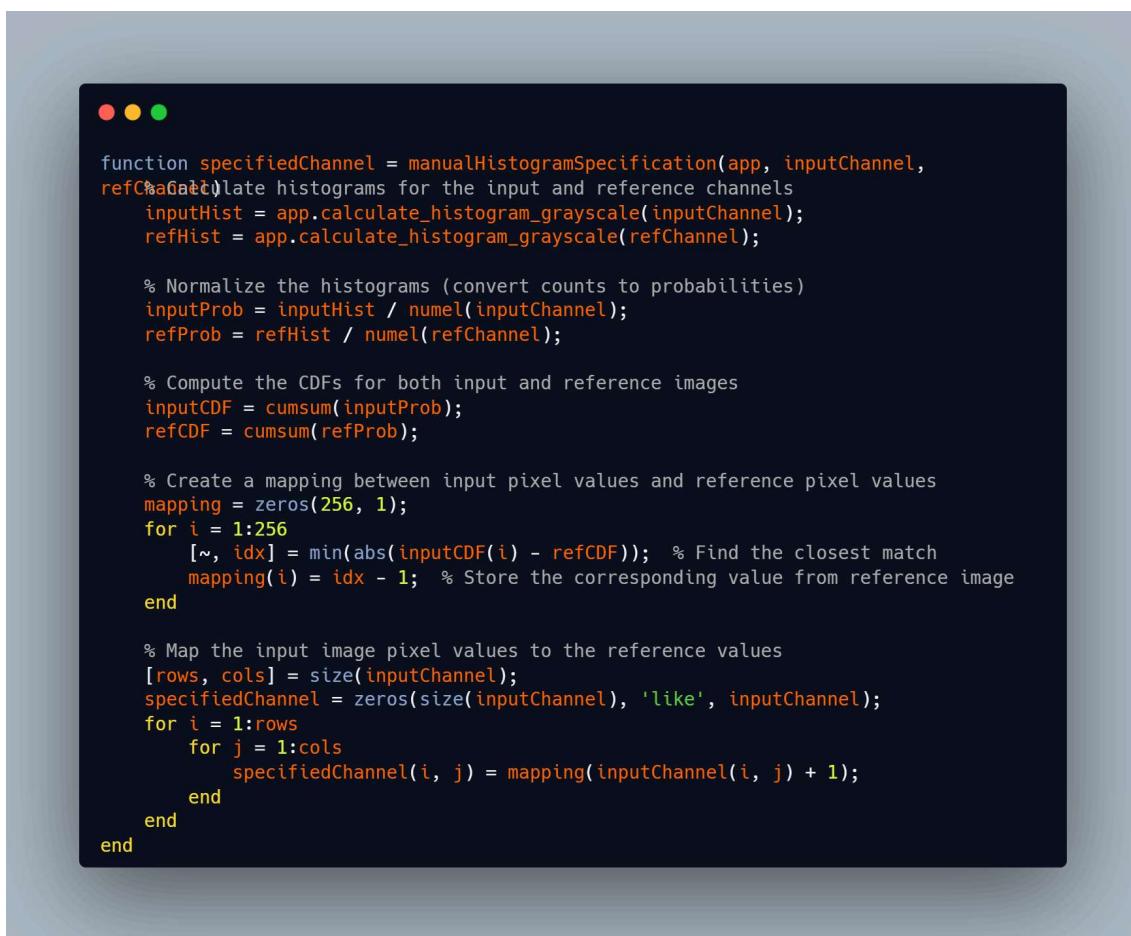
Fungsi ini menerapkan histogram specification pada sebuah channel gambar (baik grayscale atau per channel RGB). Fungsinya adalah untuk menyesuaikan histogram gambar input agar menyerupai histogram gambar referensi.

- Input:

Channel gambar input (inputChannel) dan channel gambar referensi (refChannel), keduanya dalam bentuk matriks 2D yang berisi nilai intensitas pixel.

- Output:

Channel gambar hasil histogram specification (specifiedChannel), yang nilai intensitas pixelnya disesuaikan agar mengikuti distribusi histogram gambar referensi.



```
function specifiedChannel = manualHistogramSpecification(app, inputChannel, refChannel)
% Calculate histograms for the input and reference channels
    inputHist = app.calculate_histogram_grayscale(inputChannel);
    refHist = app.calculate_histogram_grayscale(refChannel);

    % Normalize the histograms (convert counts to probabilities)
    inputProb = inputHist / numel(inputChannel);
    refProb = refHist / numel(refChannel);

    % Compute the CDFs for both input and reference images
    inputCDF = cumsum(inputProb);
    refCDF = cumsum(refProb);

    % Create a mapping between input pixel values and reference pixel values
    mapping = zeros(256, 1);
    for i = 1:256
        [~, idx] = min(abs(inputCDF(i) - refCDF)); % Find the closest match
        mapping(i) = idx - 1; % Store the corresponding value from reference image
    end

    % Map the input image pixel values to the reference values
    [rows, cols] = size(inputChannel);
    specifiedChannel = zeros(size(inputChannel), 'like', inputChannel);
    for i = 1:rows
        for j = 1:cols
            specifiedChannel(i, j) = mapping(inputChannel(i, j) + 1);
        end
    end
end
```

4.2 Hasil Eksekusi Histogram Specification

IMAGE ENHANCEMENT Select Method **Histogram Specification ▾**


Input Image

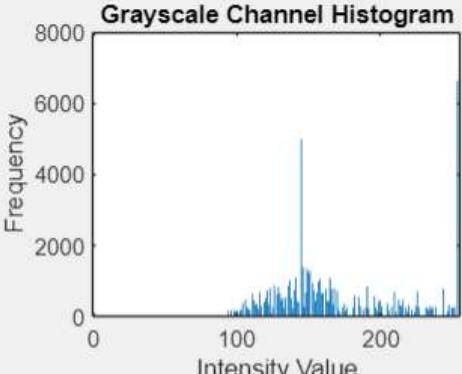
Load Image


Output Image


Reference Image

Load Image

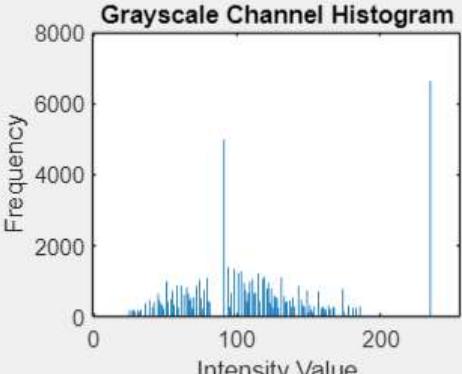
Grayscale Channel Histogram



Frequency

Intensity Value

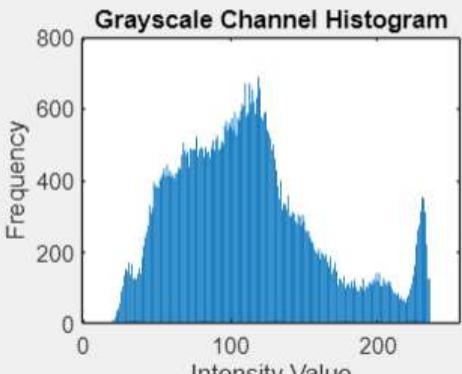
Grayscale Channel Histogram



Frequency

Intensity Value

Grayscale Channel Histogram



Frequency

Intensity Value

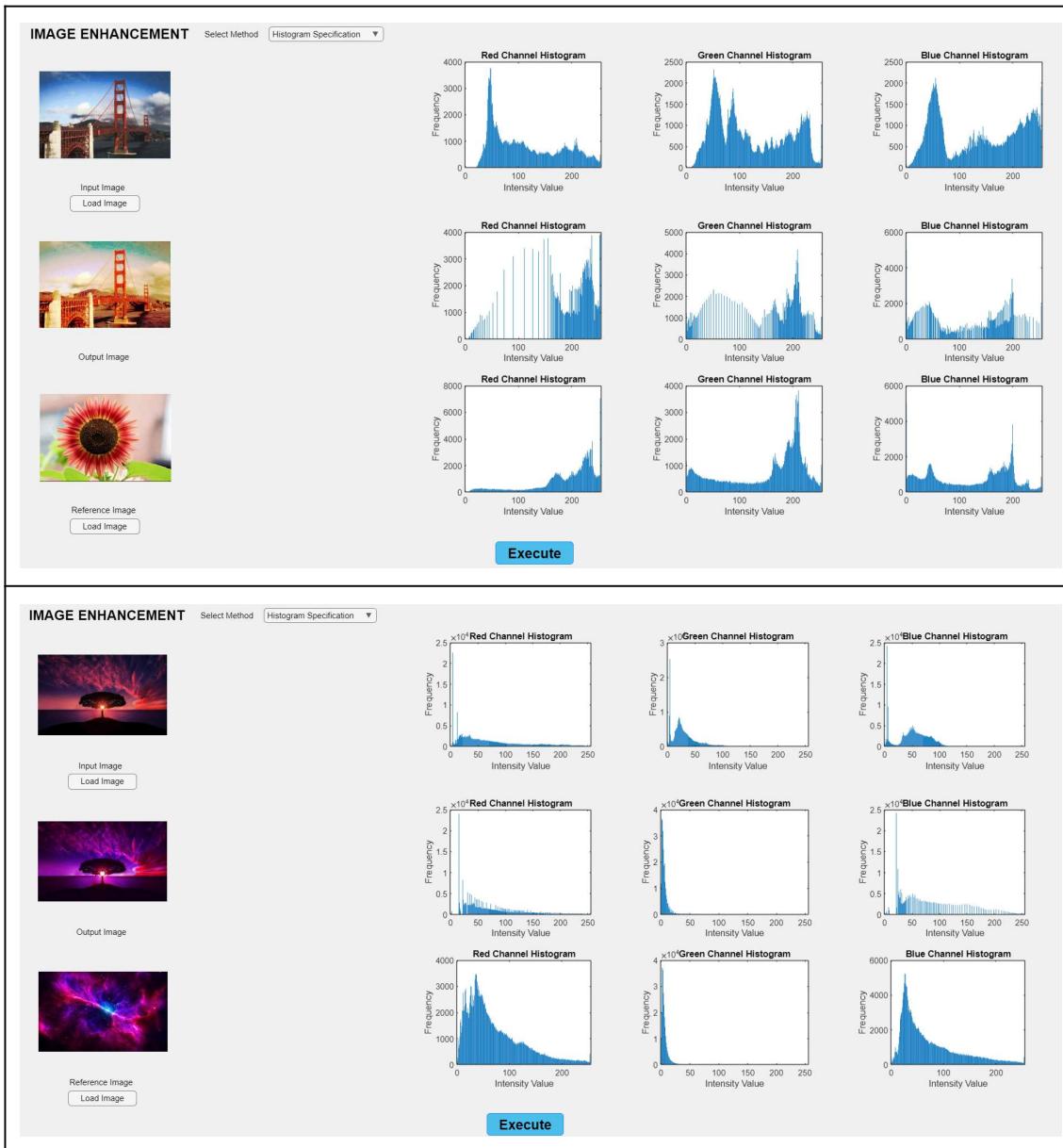


IMAGE ENHANCEMENT

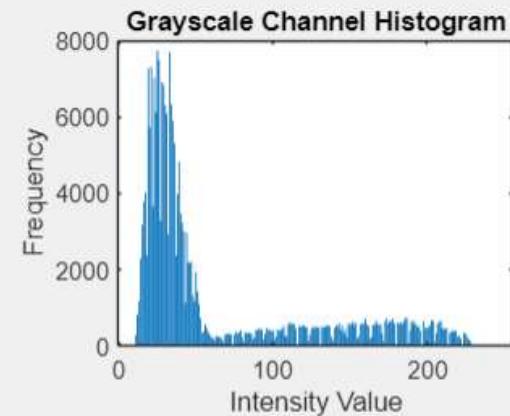
Select Method

Histogram Specification ▾

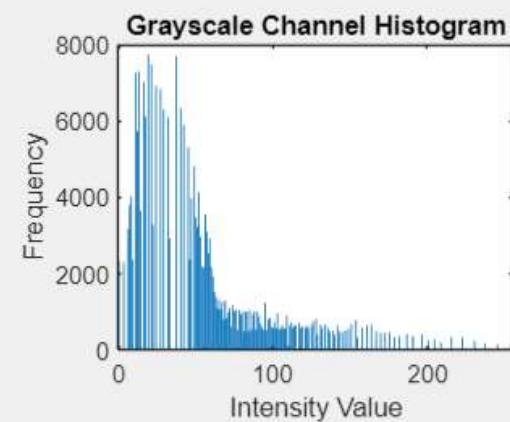


Input Image

Load Image

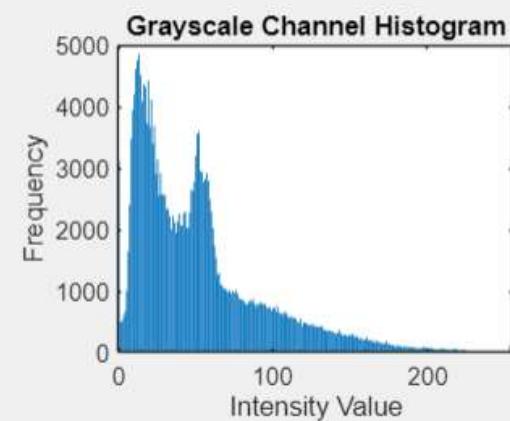


Output Image



Reference Image

Load Image



4.3 Analisis Histogram Specification

Fungsi `histogramSpecification` adalah implementasi yang efisien dalam mencocokkan distribusi histogram gambar input dengan gambar referensi. Meskipun terdapat dua loop utama (satu untuk mencocokkan CDF dan satu untuk memetakan pixel), metode ini memberikan hasil yang akurat dengan kontrol penuh atas proses histogram specification.

- Fungsi ini menangani gambar grayscale dan RGB secara terpisah, yang membuatnya fleksibel. Proses ini efisien karena hanya menggunakan perulangan untuk setiap channel dalam gambar RGB, sehingga fungsi ini dapat diterapkan pada berbagai jenis gambar.
- Dalam implementasi untuk gambar RGB, setiap channel diproses secara independen, yang memastikan bahwa distribusi intensitas warna di setiap channel disesuaikan dengan referensi yang sesuai.

Fungsi `histogramSpecificationImage` menangani gambar dengan jumlah channel yang berbeda, membuatnya fleksibel dan dapat digunakan pada gambar grayscale dan RGB. Implementasi per channel untuk gambar RGB menjamin penyesuaian yang presisi pada setiap channel warna.

- Mapping CDF: Proses pencocokan nilai intensitas antara CDF input dan referensi menggunakan loop dan mencari perbedaan minimum bisa relatif lambat, terutama untuk gambar dengan ukuran besar. Pencocokan dilakukan 256 kali (untuk semua intensitas), dan ini diulang untuk setiap pixel dalam gambar.
- Perulangan Pixel: Seperti dalam fungsi sebelumnya, terdapat loop nested (for i, j) yang memetakan setiap pixel ke nilai baru berdasarkan mapping. Pendekatan ini memungkinkan kontrol penuh, tetapi bisa memakan waktu untuk gambar dengan resolusi tinggi.
- Akurasi dan Hasil: Fungsi ini secara akurat mencocokkan distribusi intensitas pixel input dengan distribusi referensi, menghasilkan gambar yang memiliki karakteristik intensitas yang lebih mirip dengan gambar referensi.

Lampiran

Kode program github: <https://github.com/akhmadst1/IF4073-Image-Enhancement>