

LAPORAN TUGAS I

PEMROSESAN CITRA DIGITAL

Image Enhancement

Diajukan sebagai tugas Mata Kuliah IF4073 Pemrosesan Citra Digital



Disusun Oleh:

Jason Rivalino 13521008

Juan Christopher Santoso 13521116

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

DAFTAR ISI.....	2
DAFTAR GAMBAR.....	3
BAB I	
DESKRIPSI MASALAH.....	5
BAB II	
GRAPHICAL USER INTERFACE (GUI) PROGRAM.....	7
BAB III	
PENJELASAN KODE DAN PENGUJIAN PROGRAM.....	12
I. Kode Program dan Penjelasan.....	12
A. Kode program fungsi utama.....	12
B. Kode program fungsi pendukung untuk gambar.....	29
C. Kode program fungsi pendukung untuk histogram.....	31
D. Kode program fungsi pendukung untuk vektor.....	35
II. Hasil Pengujian Program.....	37
A. Pengujian program untuk menampilkan histogram citra.....	37
B. Pengujian program untuk perbaikan kualitas pada citra.....	41
a. Perbaikan kualitas citra dengan image brightening.....	41
b. Perbaikan kualitas citra dengan citra negatif dan balikan citra negatif.....	42
c. Perbaikan kualitas citra dengan transformasi Log.....	44
d. Perbaikan kualitas citra dengan transformasi pangkat.....	45
e. Perbaikan kualitas citra dengan Contrast Stretching.....	46
C. Pengujian Program untuk perataan histogram (Histogram Equalization).....	48
D. Pengujian Program untuk Histogram Specification.....	52
III. Analisis Cara Kerja Program.....	55
A. Analisis cara kerja program untuk menghitung dan menampilkan histogram.....	55
B. Analisis cara kerja program untuk perbaikan kualitas citra.....	56
a. Analisis untuk pencerahan citra (image brightening).....	56
b. Analisis untuk citra negatif dan balikan citra negatif.....	56
c. Analisis untuk transformasi log.....	57
d. Analisis untuk transformasi pangkat.....	57
e. Analisis untuk peregangan kontras (contrast stretching).....	58
C. Analisis cara kerja program untuk histogram equalization.....	58
D. Analisis cara kerja program untuk histogram specification.....	60
BAB IV	
KESIMPULAN.....	63
DAFTAR PUSTAKA.....	64
LAMPIRAN.....	65

DAFTAR GAMBAR

Gambar 2.1 Tampilan GUI dari program Image Enhancement untuk menampilkan histogram.....	7
Gambar 2.2 Tampilan GUI dari program Image Enhancement untuk perbaikan kualitas citra dengan Image Brightening.....	8
Gambar 2.3 Tampilan GUI dari program Image Enhancement untuk perbaikan kualitas citra dengan Image Negative.....	8
Gambar 2.4 Tampilan GUI dari program Image Enhancement untuk perbaikan kualitas citra dengan Image Log Transformation.....	9
Gambar 2.5 Tampilan GUI dari program Image Enhancement untuk perbaikan kualitas citra dengan Image Power Transformation.....	9
Gambar 2.6 Tampilan GUI dari program Image Enhancement untuk perbaikan kualitas citra dengan Image Contrast Stretching.....	10
Gambar 2.7 Tampilan GUI dari program Image Enhancement untuk perataan histogram (Histogram Equalization).....	10
Gambar 2.8 Tampilan GUI dari program Image Enhancement untuk Histogram Specification.....	11
Gambar 3.1 Pengujian menampilkan informasi histogram citra grayscale untuk boat.bmp.....	37
Gambar 3.2 Pengujian menampilkan informasi histogram citra grayscale untuk aerial.tiff.....	38
Gambar 3.3 Pengujian menampilkan informasi histogram citra grayscale untuk einstein.png.....	38
Gambar 3.4 Pengujian menampilkan informasi histogram citra grayscale untuk gedung.jpg.....	38
Gambar 3.5 Pengujian menampilkan informasi histogram citra berwarna untuk monarch.bmp.....	39
Gambar 3.6 Pengujian menampilkan informasi histogram citra berwarna untuk strawberries_coffee.tif.	39
Gambar 3.7 Pengujian menampilkan informasi histogram citra berwarna untuk caster.tif.....	40
Gambar 3.8 Pengujian perbaikan kualitas citra image brightening untuk peanut.tif.....	41
Gambar 3.9 Pengujian perbaikan kualitas citra image brightening untuk satur.jpg.....	41
Gambar 3.10 Pengujian perbaikan kualitas citra negatif untuk gedung.jpg.....	42
Gambar 3.11 Pengujian perbaikan kualitas citra balikan negatif untuk gedung.jpg.....	42
Gambar 3.12 Pengujian perbaikan kualitas citra negatif untuk mountain.jpg.....	43
Gambar 3.13 Pengujian perbaikan kualitas citra balikan negatif untuk mountain.jpg.....	43
Gambar 3.14 Pengujian perbaikan kualitas citra transformasi log untuk woman.tif.....	44
Gambar 3.15 Pengujian perbaikan kualitas citra transformasi log untuk baboon24.bmp.....	44
Gambar 3.16 Pengujian perbaikan kualitas citra transformasi pangkat untuk einstein.tiff.....	45
Gambar 3.17 Pengujian perbaikan kualitas citra transformasi pangkat untuk sungai.png.....	45
Gambar 3.18 Pengujian perbaikan kualitas citra transformasi pangkat untuk sungai.png.....	46
Gambar 3.19 Pengujian perbaikan kualitas citra Contrast Stretching untuk gunung.jpg.....	46
Gambar 3.20 Pengujian perbaikan kualitas citra Contrast Stretching untuk gedung.png.....	47
Gambar 3.21 Pengujian perbaikan kualitas citra Contrast Stretching untuk lake2.jpg.....	47
Gambar 3.22 Pengujian Histogram Equalization untuk peanut.tif.....	48
Gambar 3.23 Pengujian Histogram Equalization untuk aerial.tiff.....	48
Gambar 3.24 Pengujian Histogram Equalization untuk indian.png.....	49
Gambar 3.25 Pengujian Histogram Equalization untuk mobil.gif.....	49
Gambar 3.26 Pengujian Histogram Equalization untuk goldengate.jpg.....	50
Gambar 3.27 Pengujian Histogram Equalization untuk gedung.png.....	50

Gambar 3.28 Pengujian Histogram Equalization untuk satur.jpg.....	51
Gambar 3.29 Pengujian Histogram Equalization untuk sakura.png.....	51
Gambar 3.30 Hasil pertama pengujian Histogram Specification untuk girl.bmp + goldhill.bmp.....	52
Gambar 3.31 Hasil kedua pengujian Histogram Specification untuk girl.bmp + goldhill.bmp.....	52
Gambar 3.32 Hasil pertama pengujian Histogram Specification untuk bridge.jpg + flower.jpg.....	53
Gambar 3.33 Hasil kedua pengujian Histogram Specification untuk bridge.jpg + flower.jpg.....	53
Gambar 3.34 Hasil pertama pengujian Histogram Specification untuk planet.jpg + mountain_snow.png.....	54
Gambar 3.35 Hasil kedua pengujian Histogram Specification untuk mountain_snow.png + planet.jpg...	54
Gambar 3.31 Gambaran proses dari implementasi Histogram Equalization.....	60
Gambar 3.32 Gambaran proses dari implementasi Histogram Matching.....	62

BAB I

DESKRIPSI MASALAH

Pada tugas 1 dari mata kuliah IF4073 Pemrosesan Citra Digital, terdapat sejumlah program yang harus dibuat dengan menggunakan MATLAB antara lain sebagai berikut:

1. Menghitung dan menampilkan histogram citra grayscale dan citra berwarna dengan 256 derajat keabuan. Tidak dibolehkan menggunakan fungsi *imhist* atau *hist* di dalam MATLAB, namun boleh untuk dibandingkan hasilnya. Uji program dengan menggunakan contoh tujuh buah citra berikut dan dua citra tambahan (grayscale dan berwarna) yang anda cari sendiri.

Input : Satu buah citra

Output : Histogram citra masukan

2. Melakukan perbaikan kualitas pada citra-citra berikut dengan metode:
 - a. *Image brightening* ($s = r + b$ dan $s = ar + b$, a dan b adalah parameter input dari pengguna)
 - b. Citra negatif dan balikan citra negatif
 - c. Transformasi log ($s = c \log(1 + r)$, c dan r adalah parameter input dari pengguna)
 - d. Transformasi pangkat ($s = cr^\gamma$, c dan γ adalah parameter input dari pengguna)
 - e. Peregangan kontras (*contrast stretching*) untuk citra kontras-rendah, citra kontras gelap, citra kontras terang, baik citra grayscale maupun citra berwarna. Pemilihan r_{min} dan r_{max} didasarkan pada nilai keabuan terendah dan tertinggi di dalam citra masukan (nilai r_{min} dan r_{max} pada histogram citra dicari secara otomatis, bukan diisi manual). Program tersebut menampilkan citra masukan, histogram citra masukan, citra luaran, dan histogram citra luaran. Gunakan fungsi histogram yang anda buat pada program 1. Gunakan beberapa citra uji berikut dan tambahkan beberapa citra uji lain yang anda cari di internet (citra kontras-rendah)

Input : Satu buah citra

Output :

1. Histogram citra input.
2. Citra hasil perbaikan.
3. Histogram citra hasil perbaikan

3. Melakukan perataan histogram (*histogram equalization*) untuk citra yang terlalu gelap, terlalu terang, dan citra kontras-rendah, baik untuk citra grayscale maupun untuk citra berwarna. Program tersebut dapat menampilkan citra masukan, histogram citra masukan, citra luaran, dan histogram citra luaran. Gunakan fungsi histogram yang telah anda buat pada program 1. Tidak diperbolehkan menggunakan fungsi *imhist*, *hist*, dan *histeq* di dalam MATLAB, tetapi boleh dibandingkan hasilnya. Program perataan histogram harus dibuat sendiri. Gunakan enam buah citra uji di bawah ini dan tambahkan beberapa citra uji lain yang anda cari dari internet.

Input : Satu buah citra

Output :

1. Citra input.
2. Histogram citra input.
3. Citra hasil perataan histogram (*histogram equalization*).
4. Histogram citra hasil perataan histogram.

4. Melakukan perbaikan citra dengan teknik *histogram specification* (*histogram matching*). Masukan program adalah citra yang akan diperbaiki kualitasnya dan citra referensi yang histogramnya dijadikan sebagai spesifikasi histogram. Ukuran kedua citra adalah sama. Uji program anda dengan beberapa citra grayscale dan citra berwarna sebagai berikut ini, dan tambahkan beberapa citra lain yang anda cari dari internet.

Input :

1. Citra yang akan diperbaiki.
2. Citra referensi.

Output :

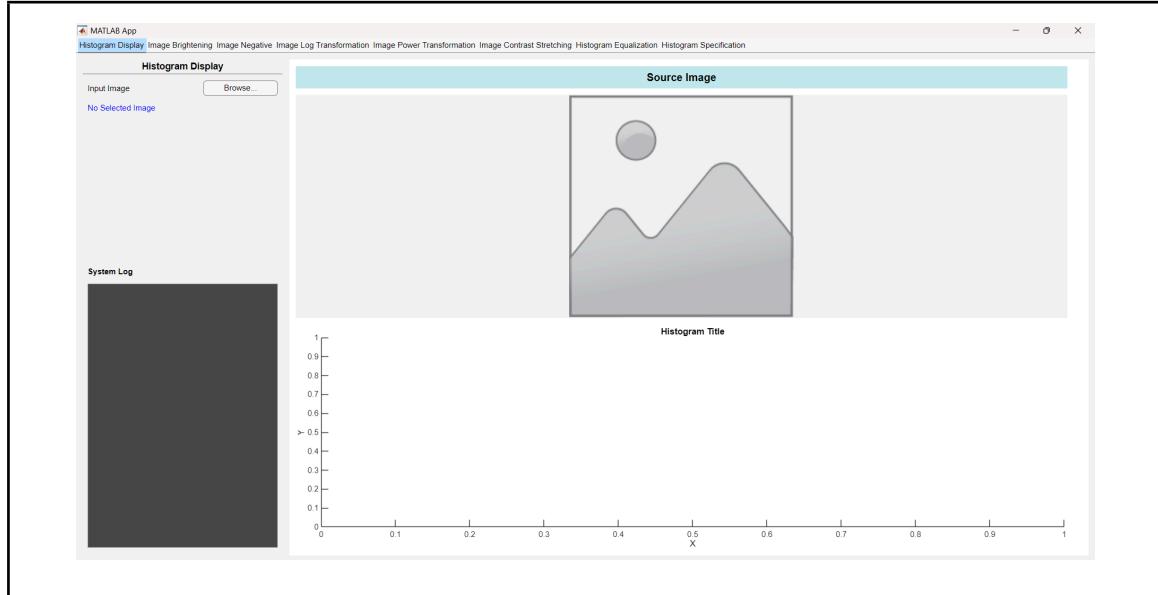
1. Citra input.
2. Histogram citra input.
3. Citra referensi.
4. Histogram citra referensi.
5. Citra hasil histogram specification.
6. Histogram citra hasil histogram specification

BAB II

GRAPHICAL USER INTERFACE (GUI) PROGRAM

Berikut merupakan tampilan layar (*screenshot*) untuk GUI dari program *Image Enhancement* yang dibuat oleh kelompok kami dengan menggunakan MATLAB:

A. Layar *Histogram Display*



Gambar 2.1 Tampilan GUI dari program *Image Enhancement* untuk menampilkan histogram

Sumber: Dokumen Penulis

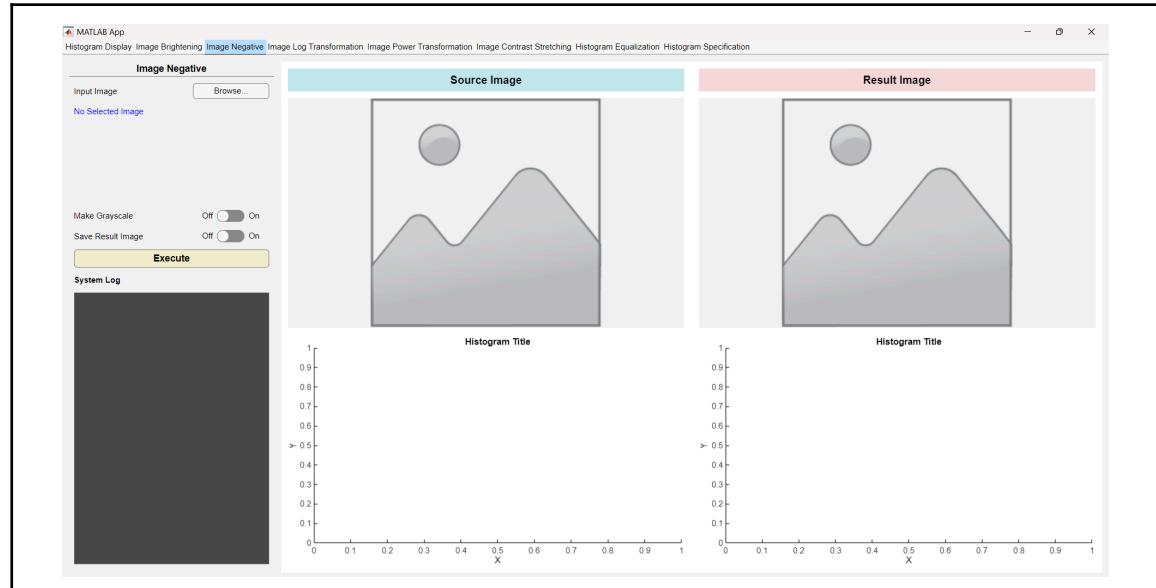
B. Layar *Image Brightening*



Gambar 2.2 Tampilan GUI dari program *Image Enhancement* untuk perbaikan kualitas citra dengan *Image Brightening*

Sumber: Dokumen Penulis

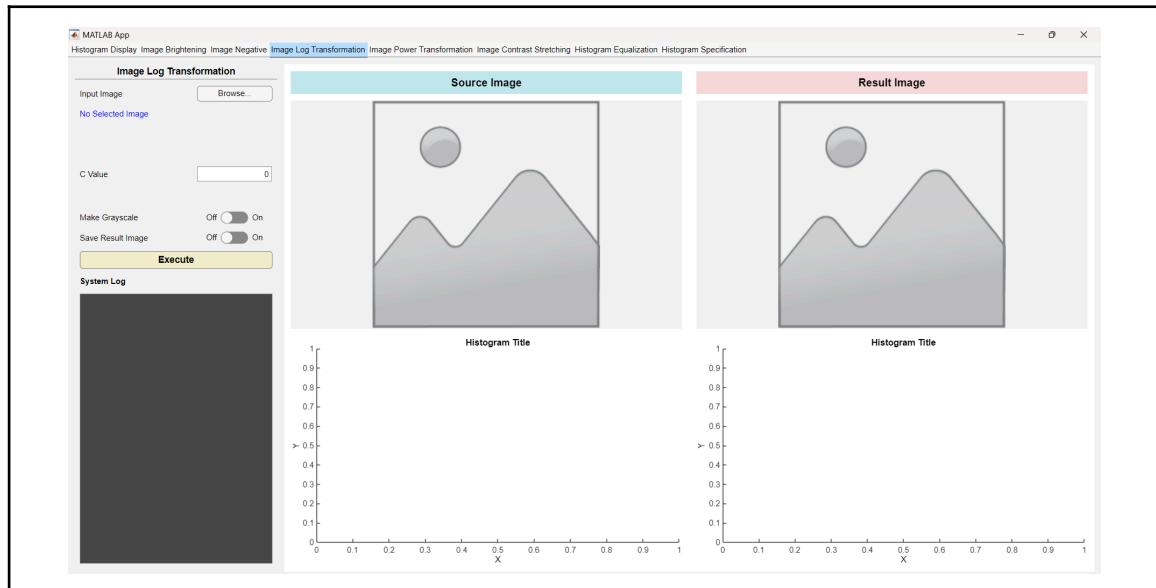
C. Layar *Image Negative*



Gambar 2.3 Tampilan GUI dari program *Image Enhancement* untuk perbaikan kualitas citra dengan *Image Negative*

Sumber: Dokumen Penulis

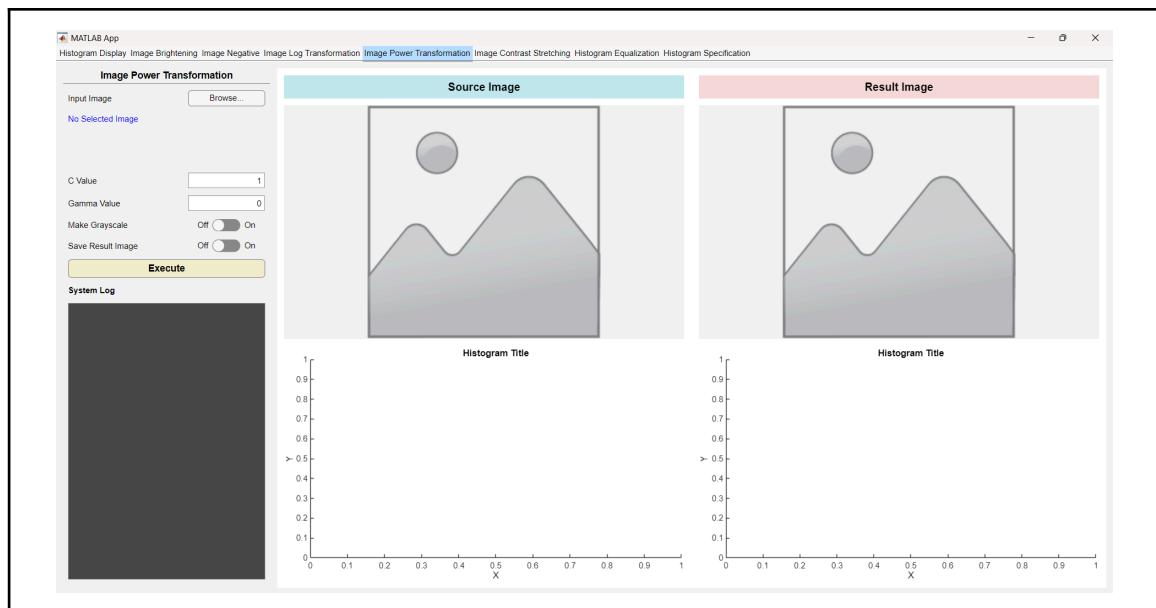
D. Layar *Image Log Transformation*



Gambar 2.4 Tampilan GUI dari program *Image Enhancement* untuk perbaikan kualitas citra dengan *Image Log Transformation*

Sumber: Dokumen Penulis

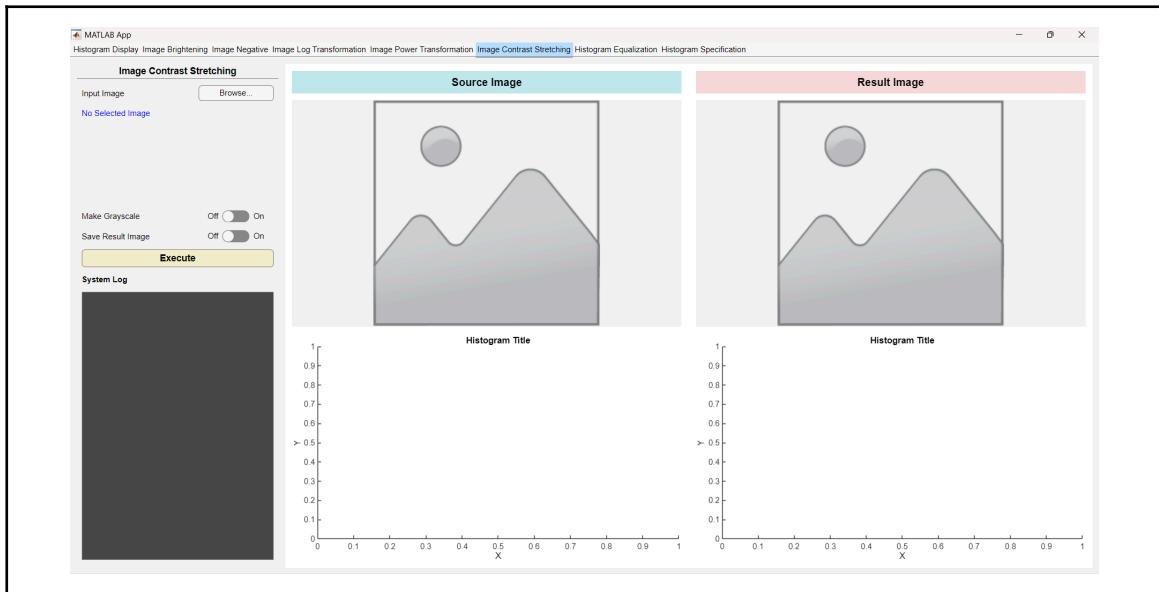
E. Layar *Image Power Transformation*



Gambar 2.5 Tampilan GUI dari program *Image Enhancement* untuk perbaikan kualitas citra dengan *Image Power Transformation*

Sumber: Dokumen Penulis

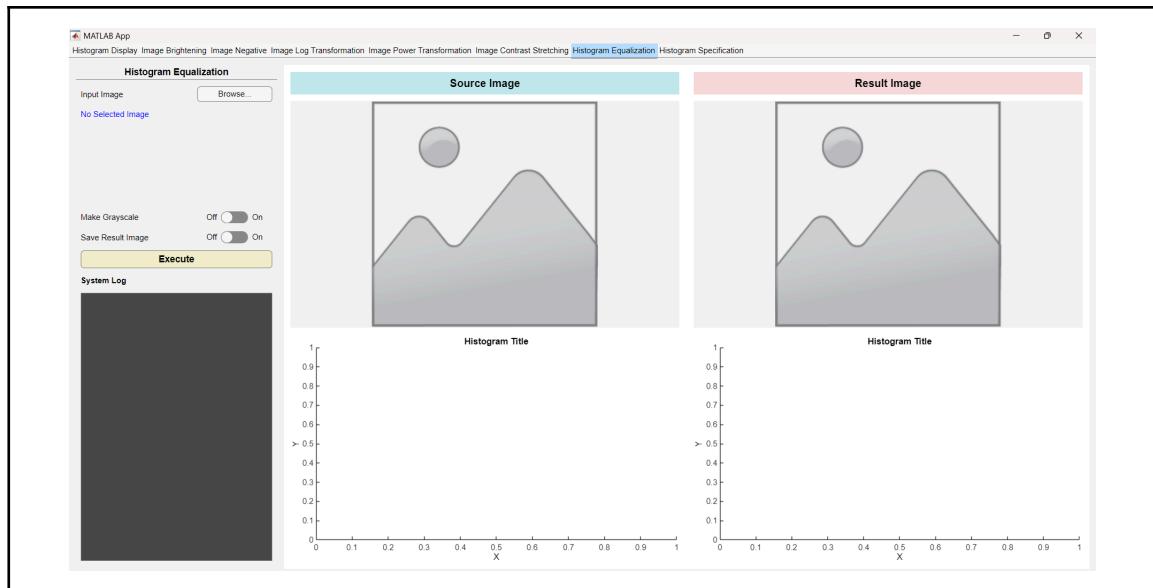
F. Layar *Image Contrast Stretching*



Gambar 2.6 Tampilan GUI dari program *Image Enhancement* untuk perbaikan kualitas citra dengan *Image Contrast Stretching*

Sumber: Dokumen Penulis

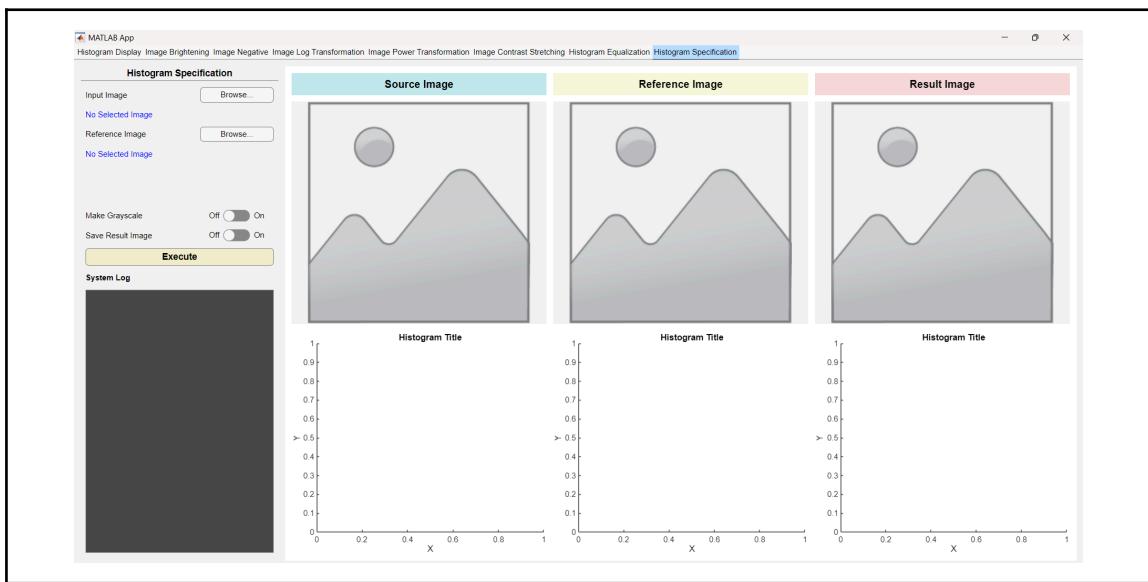
G. Layar *Histogram Equalization*



Gambar 2.7 Tampilan GUI dari program *Image Enhancement* untuk perataan histogram (*Histogram Equalization*)

Sumber: Dokumen Penulis

H. Layar *Histogram Specification*



Gambar 2.8 Tampilan GUI dari program *Image Enhancement* untuk *Histogram Specification*

Sumber: Dokumen Penulis

BAB III

PENJELASAN KODE DAN PENGUJIAN PROGRAM

I. Kode Program dan Penjelasan

Untuk pengerjaan kode program, pengerjaan dilakukan dengan membuat berbagai fungsi yang diperlukan untuk menjalankan fungsionalitas program. Berikut merupakan rincian penjabarannya:

A. Kode program fungsi utama

a. *func_img_brightening*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra untuk proses *Image Brightening* untuk membuat citra menjadi lebih terang. Berikut merupakan rincian kodennya:

```
function [result_img, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] =
func_img_brightness(img, img_name, is_grayscale,
is_saved, a, b)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');

    log = "";

    % Img details
    [rows, cols, num_channels] = size(img);
    log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

    % Create placeholder for new image
    if (num_channels == 1 | is_grayscale)
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels == 1))
            % If full colored but want to be grayscaled
            img = rgb2gray(img);
        end
        is_gray = true;
        result_img = zeros(rows, cols, 1, 'uint8');
    else
        % If full colored
        result_img = img;
        is_gray = false;
    end
end
```

```

    else
        log = log + sprintf("[PROCESS] Processing full
color image!\n");
        is_gray = false;
        result_img = zeros(rows, cols, 3, 'uint8');
    end

    for r = 1:rows
        for c = 1:cols
            if (is_gray)
                curr_pixel = double(img(r,c));
                result_img(r, c) = validate_pixel(a *
curr_pixel + b);
            else
                curr_pixel_r = double(img(r, c, 1));
                curr_pixel_g = double(img(r, c, 2));
                curr_pixel_b = double(img(r, c, 3));

                result_img(r, c, 1) =
validate_pixel(uint8(a * curr_pixel_r + b));
                result_img(r, c, 2) =
validate_pixel(uint8(a * curr_pixel_g + b));
                result_img(r, c, 3) =
validate_pixel(uint8(a * curr_pixel_b + b));
            end
        end
    end

    % Show result image
    log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

    % Make histogram
    [hist_data, hist_data_r, hist_data_b, hist_data_g] =
histogram_calculate(result_img);

    if is_saved
        % Write image
        if is_grayscale == 1
            suffix = "grayscale_";
        else
            suffix = "";

```

```

        end
        img_out_name = strcat("image_brightening_",
suffix, img_name);
        write_image(result_img, img_out_name);
        log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
    end
end

```

b. *func_img_negative*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra untuk proses citra negatif dan balikan citra negatif. Berikut merupakan rincian kodennya:

```

function [result_img, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] = func_img_negative(img,
img_name, is_grayscale, is_saved)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');

    log = "";

    % Img details
    [rows, cols, num_channels] = size(img);
    log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

    % Create placeholder for new image
    if (num_channels == 1 | is_grayscale)
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels == 1))
            % If full colored but want to be grayscaled
            img = rgb2gray(img);
        end
        is_gray = true;
        result_img = zeros(rows, cols, 1, 'uint8');
    else
        log = log + sprintf("[PROCESS] Processing full
color image!\n");
        is_gray = false;
    end

```

```

        result_img = zeros(rows, cols, 3, 'uint8');
    end

PIXEL_MAX_VAL = 255;

for r = 1:rows
    for c = 1:cols
        if (is_gray)
            curr_pixel = img(r,c);
            result_img(r, c) =
validate_pixel(PIXEL_MAX_VAL - curr_pixel);
        else
            curr_pixel_r = img(r, c, 1);
            curr_pixel_g = img(r, c, 2);
            curr_pixel_b = img(r, c, 3);

            result_img(r, c, 1) =
validate_pixel(PIXEL_MAX_VAL - curr_pixel_r);
            result_img(r, c, 2) =
validate_pixel(PIXEL_MAX_VAL - curr_pixel_g);
            result_img(r, c, 3) =
validate_pixel(PIXEL_MAX_VAL - curr_pixel_b);
        end
    end
end

% Show result image
log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

% Make histogram
[hist_data, hist_data_r, hist_data_b, hist_data_g] =
histogram_calculate(result_img);

if is_saved
    % Write image
    if is_grayscale == 1
        suffix = "grayscale_";
    else
        suffix = "";
    end
    img_out_name = strcat("image_negative_", suffix,

```

```

        img_name);
        write_image(result_img, img_out_name);
        log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
    end

end

```

c. *func_img_log_transformation*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra untuk proses transformasi log pada citra. Berikut merupakan rincian kodennya:

```

function [result_img, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] =
func_img_log_transformation(img, img_name, is_grayscale,
is_saved, c_coef)

% Initialization
addpath('./functions/image');
addpath('./functions/histogram');

log = "";

% Img details
[rows, cols, num_channels] = size(img);
log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

% Create placeholder for new image
if (num_channels == 1 | is_grayscale)
    log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
    if (not(num_channels == 1))
        % If full colored but want to be grayscaled
        img = rgb2gray(img);
    end
    is_gray = true;
    result_img = zeros(rows, cols, 1, 'uint8');
else
    log = log + sprintf("[PROCESS] Processing full
color image!\n");
    is_gray = false;
end

```

```

        result_img = zeros(rows, cols, 3, 'uint8');
    end

    for r = 1:rows
        for c = 1:cols
            if (is_gray)
                curr_pixel = double(img(r,c));
                result_img(r, c) = validate_pixel(c_coef
* log10(1 + curr_pixel));
            else
                curr_pixel_r = double(img(r, c, 1));
                curr_pixel_g = double(img(r, c, 2));
                curr_pixel_b = double(img(r, c, 3));

                result_img(r, c, 1) =
validate_pixel(uint8(c_coef * log10(1 + curr_pixel_r)));
                result_img(r, c, 2) =
validate_pixel(uint8(c_coef * log10(1 + curr_pixel_g)));
                result_img(r, c, 3) =
validate_pixel(uint8(c_coef * log10(1 + curr_pixel_b)));
            end
        end
    end

    % Show result image
    log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

    % Make histogram
    [hist_data, hist_data_r, hist_data_b, hist_data_g] =
histogram_calculate(result_img);

    if is_saved
        % Write image
        if is_grayscaled == 1
            suffix = "grayscale_";
        else
            suffix = "";
        end
        img_out_name =
strcat("image_log_transformation_", suffix, img_name);
        write_image(result_img, img_out_name);

```

```

        log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
    end
end

```

d. *func_img_power_transformation*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra untuk proses transformasi pangkat pada citra. Berikut merupakan rincian kodennya:

```

function [result_img, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] =
func_img_power_transformation(img, img_name,
is_grayscale, is_saved, c_coef, n)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');

    log = "";

    % Img details
    [rows, cols, num_channels] = size(img);
    log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

    % Create placeholder for new image
    if (num_channels == 1 | is_grayscale)
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels == 1))
            % If full colored but want to be grayscale
            img = rgb2gray(img);
        end
        is_gray = true;
        result_img = zeros(rows, cols, 1, 'uint8');
    else
        log = log + sprintf("[PROCESS] Processing full
color image!\n");
        is_gray = false;
        result_img = zeros(rows, cols, 3, 'uint8');
    end

```

```

PIXEL_MAX_VAL = 255;

for r = 1:rows
    for c = 1:cols
        if (is_gray)
            curr_pixel = double(img(r,c));
            result_img(r, c) = validate_pixel(c_coef
* (curr_pixel ^ n));
        else
            curr_pixel_r = double(img(r, c, 1))/
PIXEL_MAX_VAL;
            curr_pixel_g = double(img(r, c, 2))/
PIXEL_MAX_VAL;
            curr_pixel_b = double(img(r, c, 3))/PIXEL_MAX_VAL;

            result_img(r, c, 1) =
validate_pixel(uint8(c_coef * (curr_pixel_r ^ n) *
PIXEL_MAX_VAL));
            result_img(r, c, 2) =
validate_pixel(uint8(c_coef * (curr_pixel_g ^ n) *
PIXEL_MAX_VAL));
            result_img(r, c, 3) =
validate_pixel(uint8(c_coef * (curr_pixel_b ^ n) *
PIXEL_MAX_VAL));
        end
    end
end

% Show result image
log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

% Make histogram
[hist_data, hist_data_r, hist_data_b, hist_data_g] =
histogram_calculate(result_img);

if is_saved
    % Write image
    if is_grayscale == 1
        suffix = "grayscale_";
    else

```

```

        suffix = "";
    end
    img_out_name =
strcat("image_power_transformation_", suffix, img_name);
    write_image(result_img, img_out_name);
    log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
end
end

```

e. *func_img_contrast_stretching*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra untuk proses peregangan kontras (*contrast stretching*) pada citra. Berikut merupakan rincian kodennya:

```

function [result_img, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] =
func_img_contrast_stretching(img, img_name,
is_grayscale, is_saved)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');

    log = "";

    % Img details
    [rows, cols, num_channels] = size(img);
    log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

    % Create placeholder for new image
    if (num_channels == 1 | is_grayscale)
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels == 1))
            % If full colored but want to be grayscaled
            img = rgb2gray(img);
        end
        is_gray = true;
        result_img = zeros(rows, cols, 1, 'uint8');
    else

```

```

        log = log + sprintf("[PROCESS] Processing full
color image!\n");
        is_gray = false;
        result_img = zeros(rows, cols, 3, 'uint8');
    end

    if (is_gray)
        max_val = search_max_pixel(img, rows, cols);
        min_val = search_min_pixel(img, rows, cols);
        fprintf('[INFO] Max value: %d, Min value: %d\n',
max_val, min_val);

        for r = 1:rows
            for c = 1: cols
                curr_pixel = img(r,c);
                result_img(r, c) =
normalize_pixel(curr_pixel, max_val, min_val);
            end
        end
    else
        r_aspect = img(:,:,1); % Red channel
        max_val_r = search_max_pixel(r_aspect, rows,
cols);
        min_val_r = search_min_pixel(r_aspect, rows,
cols);
        fprintf('[INFO] Red : Max value: %d, Min value:
%d\n', max_val_r, min_val_r);

        g_aspect = img(:,:,2); % Green channel
        max_val_g = search_max_pixel(g_aspect, rows,
cols);
        min_val_g = search_min_pixel(g_aspect, rows,
cols);
        fprintf('[INFO] Green : Max value: %d, Min value:
%d\n', max_val_g, min_val_g);

        b_aspect = img(:,:,3); % Blue channel
        max_val_b = search_max_pixel(b_aspect, rows,
cols);
        min_val_b = search_min_pixel(b_aspect, rows,
cols);
        fprintf('[INFO] Blue : Max value: %d, Min value:

```

```

%d\n', max_val_b, min_val_b);

for r = 1:rows
    for c = 1: cols
        curr_pixel_r = double(img(r, c, 1));
        curr_pixel_g = double(img(r, c, 2));
        curr_pixel_b = double(img(r, c, 3));

        result_img(r, c, 1) =
normalize_pixel(curr_pixel_r, max_val_r, min_val_r);
        result_img(r, c, 2) =
normalize_pixel(curr_pixel_g, max_val_g, min_val_g);
        result_img(r, c, 3) =
normalize_pixel(curr_pixel_b, max_val_b, min_val_b);
    end
end

% Show result image
log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

% Make histogram
[hist_data, hist_data_r, hist_data_b, hist_data_g] =
histogram_calculate(result_img);

if is_saved
    % Write image
    if is_grayscale == 1
        suffix = "grayscale_";
    else
        suffix = "";
    end
    img_out_name =
strcat("image_construct_stretching_", suffix, img_name);
    write_image(result_img, img_out_name);
    log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
end
end

```

f. *func_histogram_equalizer*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra dengan proses perataan histogram (*histogram equalizer*). Berikut merupakan rincian kodennya:

```
function [imgEqualized, hist_data, hist_data_r,
hist_data_b, hist_data_g, log] =
func_histogram_equalizer(img, img_name, is_grayscale,
is_saved)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');

    log = "";

    % Img details
    [rows, cols, num_channels] = size(img);
    log = log + sprintf("[INFO] An image size [%d, %d] is
inputted!\n", rows, cols);

    if (num_channels == 1 | is_grayscale)
        % Create placeholder for new image
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels == 1))
            % If full colored but want to be grayscale
            img = rgb2gray(img);
        end
        is_gray = true;
    else
        log = log + sprintf("[PROCESS] Processing full
color image!\n");
        is_gray = false;
    end

    if not(is_gray) % Jika gambar RGB, proses setiap
channel (R, G, B) secara terpisah
        % Pisahkan gambar menjadi 3 channel: R, G, B
        imgR = img(:, :, 1);
        imgG = img(:, :, 2);
        imgB = img(:, :, 3);
```

```

        % Equalize setiap channel
        imgEqualizedR = equalizer_operation(imgR);
        imgEqualizedG = equalizer_operation(imgG);
        imgEqualizedB = equalizer_operation(imgB);

        % Gabungkan kembali ketiga channel menjadi gambar
        % RGB yang sudah di-equalize
        imgEqualized = cat(3, imgEqualizedR,
        imgEqualizedG, imgEqualizedB);
    else
        % Jika gambar adalah grayscale
        imgEqualized = equalizer_operation(img);
    end

    % Show result image
    log = log + sprintf("[DISPLAYING] Here is displayed
the initial and the result image\n");

    % Menampilkan histogram dari gambar yang telah
    % di-equalize
    [hist_data, hist_data_r, hist_data_b, hist_data_g] =
    histogram_calculate(imgEqualized);

    if is_saved
        % Write image
        if is_grayscaled == 1
            suffix = "grayscale_";
        else
            suffix = "";
        end
        img_out_name = strcat("histogram_equalizer_",
suffix, img_name);
        write_image(imgEqualized, img_out_name);
        log = log + sprintf("[SAVED] The image is saved
in %s\n", img_out_name);
    end
end

```

g. *func_histogram_specification*

Fungsi ini digunakan untuk memproses perbaikan kualitas citra dengan proses *histogram specification*. Berikut merupakan rincian kodennya:

```

function [result_img, result_histogram,
result_histogram_r, result_histogram_b,
result_histogram_g, log] =
func_histogram_specification(img_source,
img_name_source,img_reference,
img_name_reference,is_grayscaled, is_saved)
    % Initialization
    addpath('./functions/image');
    addpath('./functions/histogram');
    addpath('./functions/vector');

    % Function
    function [final_hist_data, mapped_hist] =
match_histogram(hist_data_source, hist_data_ref,
img_size, MAX_PIXEL_VAL)
        % Source
        norm_hist_data_source = hist_data_source /
img_size;
        cmltv_norm_hist_data_source =
make_cumulative(norm_hist_data_source);
        result_hist_data_source =
round(cmltv_norm_hist_data_source * MAX_PIXEL_VAL);
        new_hist_data_source =
map_new_hist(result_hist_data_source, hist_data_source);

        % Ref
        norm_hist_data_ref = hist_data_ref / img_size;
        cmltv_norm_hist_data_ref =
make_cumulative(norm_hist_data_ref);
        result_hist_data_ref =
round(cmltv_norm_hist_data_ref * MAX_PIXEL_VAL);

        len = length(result_hist_data_source);
        mapped_hist = zeros(1, len);
        final_hist_data = zeros(1, len);
        for i = 1: len
            idx_to_map_from_source =
result_hist_data_source(i);
            idx_nearest =
get_index_nearest_value(result_hist_data_ref,
idx_to_map_from_source);
            n = new_hist_data_source(i);
            mapped_hist(i) = n;
        end
    end
end

```

```

        mapped_hist(i) = idx_nearest;
        final_hist_data(idx_nearest) =
final_hist_data(idx_nearest) + n;
        % fprintf("%d %d %d %d\n", i,
idx_to_map_from_source, idx_nearest, n);
    end
end

log = "";

% Image details
[rows_s, cols_s, num_channels_s] = size(img_source);
[rows_r, cols_r, num_channels_r] =
size(img_reference);

MAX_PIXEL_VAL = 255;

if (not (rows_s == rows_r) | not (cols_s == cols_r))
    log = log + sprintf('[FAILED] The two inputted
images do not have the same size!\n');
    [result_img, result_histogram,
result_histogram_r, result_histogram_b,
result_histogram_g] = deal([]);
    return;
elseif (not (num_channels_s == num_channels_r))
    log = log + sprintf('[FAILED] The two inputted
images do not share the same color type!\n');
    [result_img, result_histogram,
result_histogram_r, result_histogram_b,
result_histogram_g] = deal([]);
    return;
else
    % Both image rows and cols should be the same
    log = log + sprintf("[INFO] An image size [%d,
%d] is inputted!\n", rows_s, cols_s);

    % Create placeholder for new image
    if (num_channels_s == 1 | is_grayscale)
        log = log + sprintf("[PROCESS] Processing
grayscale image!\n");
        if (not(num_channels_s == 1))
            % If full colored but want to be

```

```

grayscaled
    img_source = rgb2gray(img_source);
    img_reference = rgb2gray(img_reference);
end
is_gray = true;
result_img = zeros(rows_s, cols_s, 1,
'uint8');
else
    log = log + sprintf("[PROCESS] Processing
full color image!\n");
    is_gray = false;
    result_img = zeros(rows_s, cols_s, 3,
'uint8');
end

% Process Image
[hist_data_source, hist_data_r_source,
hist_data_b_source, hist_data_g_source] =
histogram_calculate(img_source);
[hist_data_ref, hist_data_r_ref, hist_data_b_ref,
hist_data_g_ref] = histogram_calculate(img_reference);
img_size = rows_s * cols_s;

[result_histogram, mapped_hist] =
match_histogram(hist_data_source, hist_data_ref,
img_size, MAX_PIXEL_VAL);
result_histogram_r = [];
result_histogram_g = [];
result_histogram_b = [];
if (is_gray)
    for r= 1:rows_s
        for c=1:cols_s
            curr_pixel = img_source(r,c);
            result_img(r, c) =
mapped_hist(curr_pixel+1);
        end
    end
else
    [result_histogram_r, mapped_hist_r] =
match_histogram(hist_data_r_source, hist_data_r_ref,
img_size, MAX_PIXEL_VAL);
    [result_histogram_g, mapped_hist_g] =

```

```

match_histogram(hist_data_g_source, hist_data_g_ref,
img_size, MAX_PIXEL_VAL);
    [result_histogram_b, mapped_hist_b] =
match_histogram(hist_data_b_source, hist_data_b_ref,
img_size, MAX_PIXEL_VAL);

for r= 1:rows_s
    for c=1:cols_s
        curr_pixel_r = img_source(r, c, 1);
        curr_pixel_g = img_source(r, c, 2);
        curr_pixel_b = img_source(r, c, 3);

        result_img(r, c, 1) =
mapped_hist_r(curr_pixel_r+1);
        result_img(r, c, 2) =
mapped_hist_g(curr_pixel_g+1);
        result_img(r, c, 3) =
mapped_hist_b(curr_pixel_b+1);
    end
end
end

% Show result image
log = log + sprintf("[DISPLAYING] Here is
displayed the source, reference, and result image");

if is_saved
    % Write image
    if is_grayscale == 1
        suffix = "grayscale_";
    else
        suffix = "";
    end
    img_out_name =
strcat("histogram_specification_", suffix,
img_name_source, img_name_reference);
    write_image(result_img, img_out_name);
    log = log + sprintf("[SAVED] The image is
saved in %s\n", img_out_name);
end
end

```

```
end
```

B. Kode program fungsi pendukung untuk gambar

a. *read_image*

Fungsi ini digunakan untuk memproses pembacaan gambar yang diinput dalam program. Berikut merupakan rincian kodenya:

```
function img = read_image(img_name)
    img_path = strcat("./img_in/" + img_name);
    img = imread(img_path);
end
```

b. *write_image*

Fungsi ini digunakan untuk memproses penyimpanan hasil citra baru yang telah dilakukan proses perbaikan. Berikut merupakan rincian kodenya:

```
function write_image(img, img_name)
    disp("The image will be written inside the `img_out` folder!")
    img_path = strcat("./img_out/", img_name);
    imwrite(img, img_path);
    fprintf("The image is successfully written in %s",
    img_path);
end
```

c. *validate_pixel*

Fungsi ini digunakan untuk memvalidasi pixel yang ada apakah berada dalam range antara 0-255. Berikut merupakan rincian kodenya:

```
function result = validate_pixel(pixel)
    max_val = 255;
    min_val = 0;
    result = min(pixel, max_val);
    result = max(result, min_val);
end
```

d. *normalize_pixel*

Fungsi ini digunakan untuk menormalisasi pixel yang sebelumnya memiliki range 0-255 ke dalam bentuk range 0-1. Berikut merupakan rincian kodenya:

```

function result = normalize_pixel(pixel, max_val,
min_val)
    result = uint8((pixel - min_val) * (255 / (max_val -
min_val)));
end

```

e. *search_max_pixel*

Fungsi ini digunakan untuk mencari nilai pixel tertinggi yang ada dalam keseluruhan citra. Berikut merupakan rincian kodenya:

```

function max_val = search_max_pixel(img, rows, cols)
    max_val = 0;
    for r = 1:rows
        for c = 1:cols
            if (img(r,c) > max_val)
                max_val = img(r,c);
            end
        end
    end
end

```

f. *search_min_pixel*

Fungsi ini digunakan untuk mencari nilai pixel tertinggi yang ada dalam keseluruhan citra. Berikut merupakan rincian kodenya:

```

function min_val = search_min_pixel(img, rows, cols)
    min_val = 255;
    for r = 1:rows
        for c = 1:cols
            if (img(r,c) < min_val)
                min_val = img(r,c);
            end
        end
    end
end

```

C. Kode program fungsi pendukung untuk histogram

a. *histogram_calculate*

Fungsi ini digunakan untuk mencari nilai keseluruhan yang terdapat pada range persebaran histogram antara 0-255. Berikut merupakan rincian kodenya:

```
function [histogram_data, histogram_data_r,
histogram_data_g, histogram_data_b] =
histogram_calculate(img)

    % Fungsi manual untuk menghitung histogram tanpa
    % fungsi terpisah

    % Menghitung histogram secara manual untuk gambar
    % grayscale atau channel RGB
    [rows, cols, numChannels] = size(img); % Dapatkan
    ukuran gambar

    % Cek apakah gambar merupakan Grayscale atau RGB
    if numChannels == 1 % Kondisi Gambar Grayscale

        % Inisialisasi histogram grayscale
        grayscale_hist = zeros(1, 256); % Array untuk
        menyimpan nilai intensitas 0-255

        % Loop melalui setiap pixel untuk menghitung
        % histogram grayscale
        for i = 1:rows
            for j = 1:cols
                intensity = img(i, j); % Ambil nilai
                intensitas pixel
                grayscale_hist(intensity + 1) =
                grayscale_hist(intensity + 1) + 1; % Tambahkan ke
                histogram
            end
        end

        % Assign histogram grayscale sebagai output
        histogram_data = grayscale_hist;
        histogram_data_r = [];
        histogram_data_g = [];
        histogram_data_b = [];

    end
```

```

elseif numChannels == 3 % Kondisi Gambar RGB
    % Memisahkan kondisi warna RGB
    img = double(img); % Konversi ke double untuk
proses histogram

    % Menghitung histogram combined untuk grayscale
dari gambar RGB
    combined_hist = zeros(1, 256); % Array untuk
combined histogram
    grayscale_img = rgb2gray(uint8(img)); % Konversi
gambar RGB menjadi grayscale

    for i = 1:rows
        for j = 1:cols
            intensity = grayscale_img(i, j);
            combined_hist(intensity + 1) =
combined_hist(intensity + 1) + 1;
        end
    end

    % Assign combined histogram sebagai output
histogram_data = combined_hist;

    % Informasi histogram untuk RGB
    for i = 1:3
        % Inisialisasi histogram untuk channel RGB
        channel_hist = zeros(1, 256);

        % Hitung histogram untuk channel tertentu (R,
G, atau B)
        for row = 1:rows
            for col = 1:cols
                intensity = img(row, col, i); %
Ambil intensitas untuk channel i (R, G, atau B)
                channel_hist(intensity + 1) =
channel_hist(intensity + 1) + 1;
            end
        end

        if (i == 1)
            histogram_data_r = channel_hist;
        elseif(i == 2)

```

```

        histogram_data_g = channel_hist;
    else
        histogram_data_b = channel_hist;
    end
end
else
    disp('File format is not supported.');
    histogram_data = []; % Empty ketika file format
tidak support
end
end

```

b. *histogram_show*

Fungsi ini digunakan untuk memvisualisasikan hasil kalkulasi histogram yang didapat ke dalam bentuk grafik. Berikut merupakan rincian kodennya:

```

function histogram_show(histogram_data, histogram_data_r,
histogram_data_g, histogram_data_b, is_grayscale,
additional_title_text)
    figure('Color', 'w', 'Position', [50, 50, 1200,
600]);
    if (is_grayscale)
        % Menampilkan histogram grayscale
        subplot(1, 1, 1);
        bar(0:255, histogram_data, 'FaceColor',
'Magenta', 'EdgeColor', 'Magenta'); % Plot histogram
        title('Grayscale Histogram',
additional_title_text);
        set(gca, 'XLim', [0 255], 'YLim', [0
max(histogram_data)]); % Set properti sumbu x dan sumbu y
    else
        subplot(2, 3, 2);
        bar(0:255, histogram_data, 'FaceColor',
'Magenta', 'EdgeColor', 'Magenta'); % Plot histogram
        title('Combined Histogram ',
additional_title_text);
        set(gca, 'XLim', [0 255], 'YLim', [0
max(histogram_data)]); % Set properti sumbu x dan sumbu y
    end
    for i= 1:3

```

```

        if (i == 1)
            channel_hist = histogram_data_r;
        elseif(i == 2)
            channel_hist = histogram_data_g;
        else
            channel_hist = histogram_data_b;
        end

        clr = 'rgb';           % Identifikasi RGB
        clrTxt = {'Red', 'Green', 'Blue'};
        subplot(2, 3, i + 3);
        % Plot histogram RGB
        bar(0:255, channel_hist, 'FaceColor', clr(i),
        'EdgeColor', 'none'); % Plot histogram
        set(gca, 'XLim', [0 255], 'YLim', [0
max(channel_hist)]); % Set properti sumbu
        title([clrTxt{i}, ' Histogram ',
additional_title_text]);
        xlabel('Intensity');
    end
end
end

```

c. *equalizer_operation*

Fungsi ini digunakan untuk melakukan kalkulasi perhitungan untuk pemerataan histogram yang akan dipakai untuk proses *histogram equalization*. Berikut merupakan rincian kodennya:

```

% Function untuk melakukan histogram equalization pada
% satu channel (grayscale)

function imageEqualized = equalizer_operation(channel)
    % Ukuran total pixel dalam channel
    [rows, cols] = size(channel);
    totalPixels = rows * cols;

    % Langkah 1: Menghitung histogram secara manual (PDF)
    counts = zeros(1, 256); % Array untuk menghitung
    setiap intensitas 0-255
    for i = 1:totalPixels
        pixelValue = channel(i);

```

```

        counts(pixelValue + 1) = counts(pixelValue + 1) +
1;
end

% Langkah 2: Hitung PDF (Probability Density
Function)
pdf = counts / totalPixels;

% Langkah 3: Hitung CDF (Cumulative Distribution
Function)
cdf = cumsum(pdf);

% Langkah 4: Terapkan transformasi equalization ke
channel menggunakan fungsi transformasi  $T(r_k) = (L-1) * CDF(r_k)$ 
transformFunc = uint8(255 * cdf); % Normalisasi CDF

% Menerapkan transformasi ke channel asli
imageEqualized = zeros(size(channel), 'uint8');
for i = 1:totalPixels
    imageEqualized(i) = transformFunc(channel(i) +
1); % +1 karena indeks MATLAB dimulai dari 1
end
end

```

D. Kode program fungsi pendukung untuk vektor

a. *get_index_nearest_value*

Fungsi ini digunakan untuk mendapatkan indeks terdekat yang ada dalam vektor. Berikut merupakan rincian kodennya:

```

function idx = get_index_nearest_value(vector, val)
len = length(vector);
idx = 1;
while (idx < len & val >= vector(idx))
    idx = idx + 1;
end
end

```

b. *make_cumulative*

Fungsi ini digunakan untuk menghitung kumulatif vektor. Berikut merupakan rincian kodennya:

```
function result_vector = make_cumulative(vector)
    v_length = length(vector);
    result_vector = zeros(1, v_length);
    for i = 1:v_length
        if (i == 1)
            result_vector(i) = vector(i);
        else
            result_vector(i) = vector(i) +
result_vector(i-1);
        end
    end
end
```

c. *map_new_hist*

Fungsi ini digunakan untuk melakukan proses *mapping* pada histogram yang baru dibentuk untuk *histogram specification*. Berikut merupakan rincian kodennya:

```
function new_hist_data =
map_new_hist(rounded_equalized_hist, initial_hist)
    len = length(rounded_equalized_hist);
    new_hist_data = zeros(1, len);
    for i= 1: len
        n = initial_hist(i);
        equalized_idx = rounded_equalized_hist(i);
        new_hist_data(equalized_idx+1) =
new_hist_data(equalized_idx+1) + n;
    end
end
```

d. *sum_value*

Fungsi ini digunakan untuk melakukan penambahan skalar terhadap seluruh nilai vektor yang ada. Berikut merupakan rincian kodennya:

```
function sum = sum_value(vector)
    sum = 0;
    for i=1:length(vector)
```

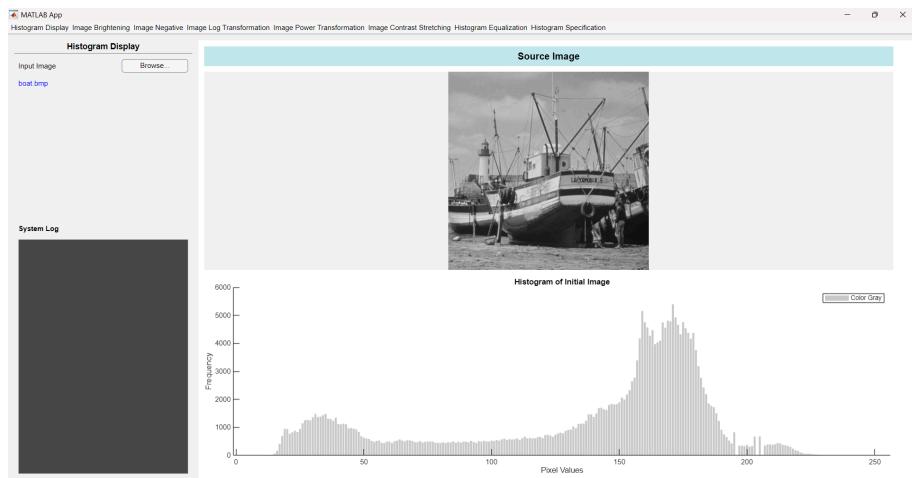
```
    sum= sum + vector(i);  
end  
end
```

II. Hasil Pengujian Program

A. Pengujian program untuk menampilkan histogram citra

a. Pengujian citra *grayscale*

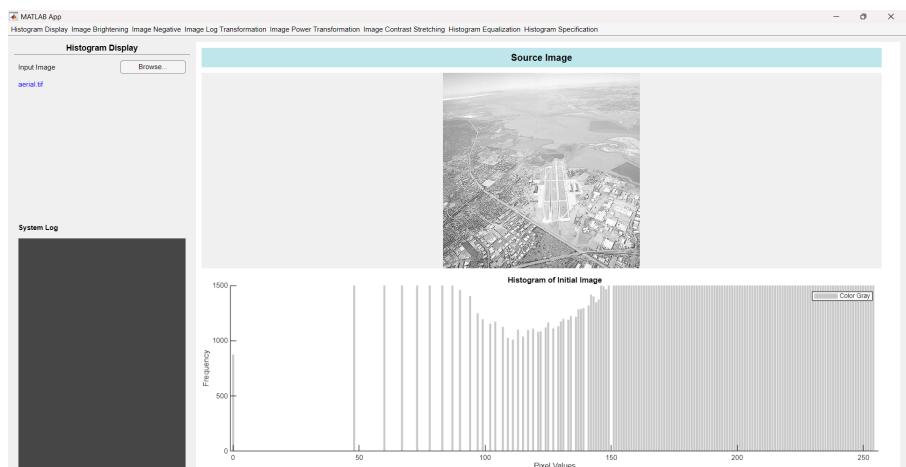
1. Gambar *boat.bmp*



Gambar 3.1 Pengujian menampilkan informasi histogram citra *grayscale* untuk *boat.bmp*

Sumber: Dokumen Penulis

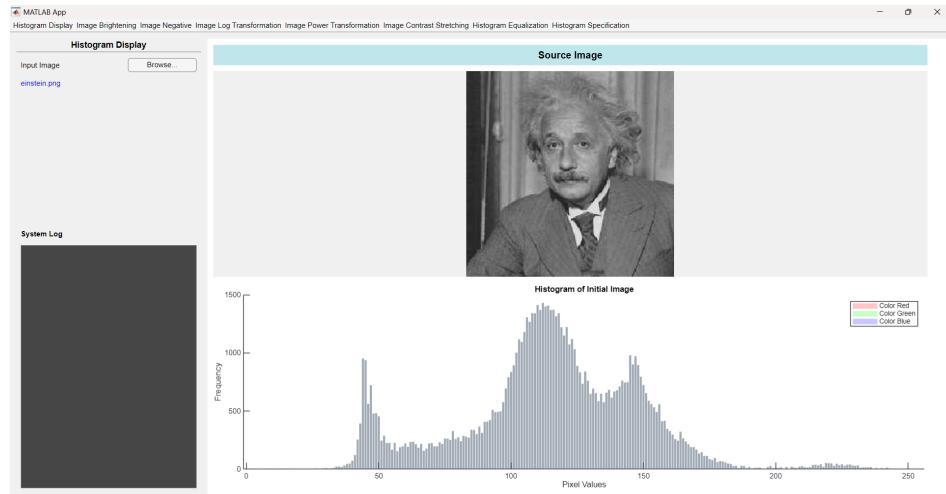
2. Gambar *aerial.tif*



Gambar 3.2 Pengujian menampilkan informasi histogram citra grayscale untuk *aerial.tiff*

Sumber: Dokumen Penulis

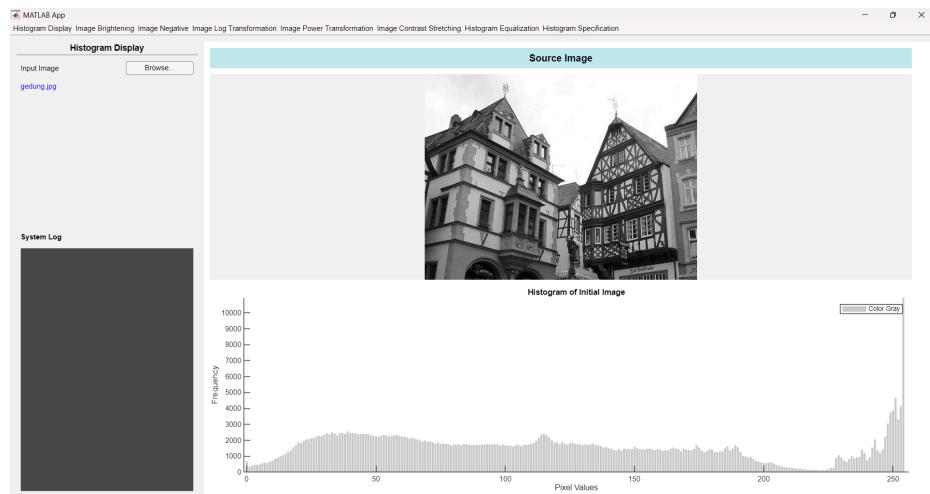
3. Gambar *einstein.png*



Gambar 3.3 Pengujian menampilkan informasi histogram citra grayscale untuk *einstein.png*

Sumber: Dokumen Penulis

4. Gambar *gedung.jpg*

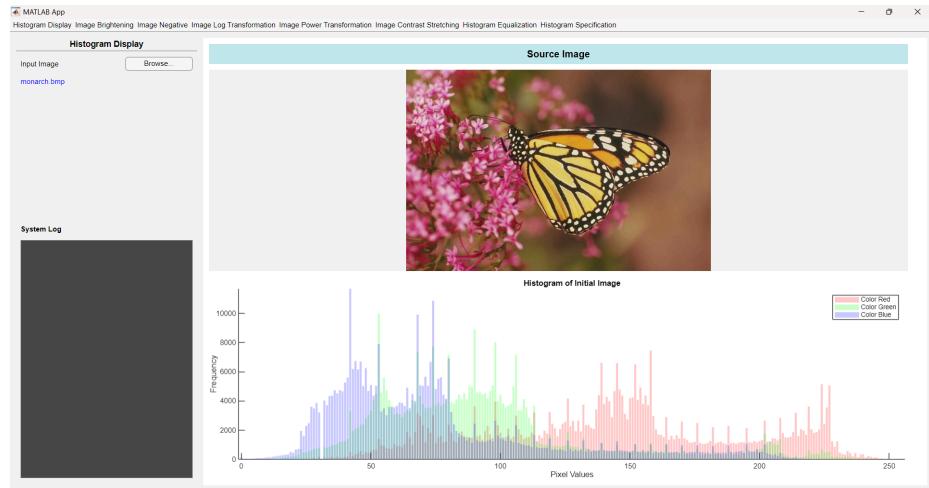


Gambar 3.4 Pengujian menampilkan informasi histogram citra grayscale untuk *gedung.jpg*

Sumber: Dokumen Penulis

b. Pengujian citra berwarna

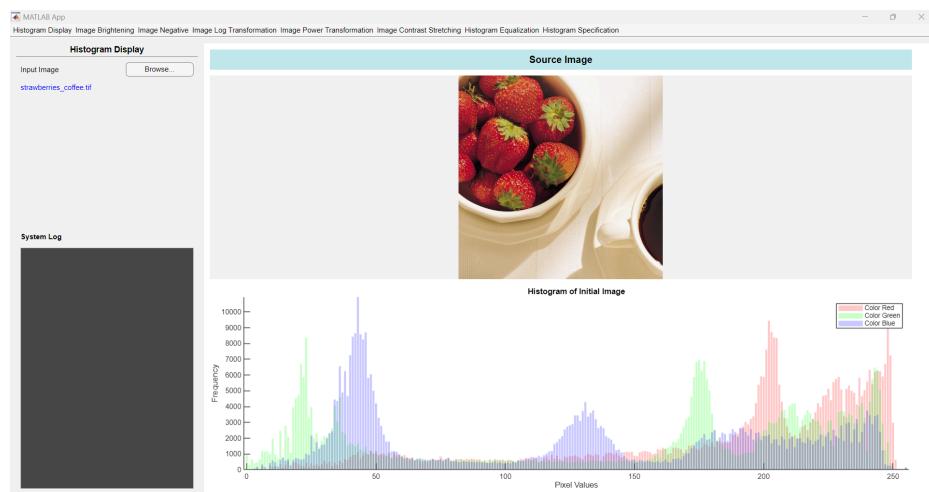
1. Gambar *monarch.bmp*



Gambar 3.5 Pengujian menampilkan informasi histogram citra berwarna untuk *monarch.bmp*

Sumber: Dokumen Penulis

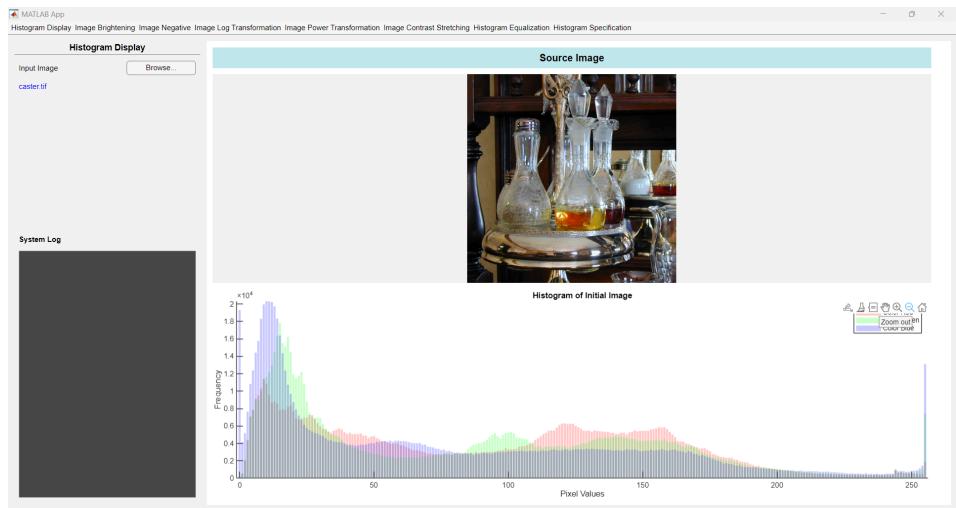
2. Gambar *strawberries_coffee.tif*



Gambar 3.6 Pengujian menampilkan informasi histogram citra berwarna untuk *strawberries_coffee.tif*

Sumber: Dokumen Penulis

3. Gambar *caster.tif*



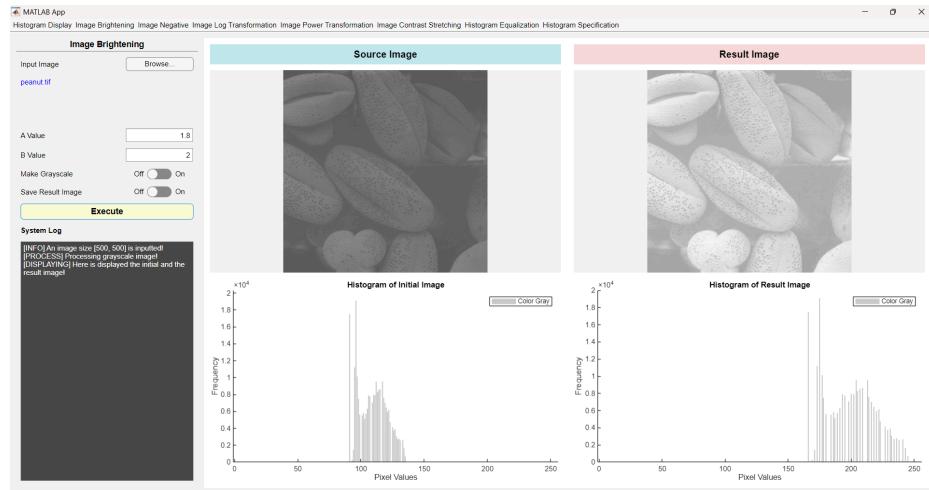
Gambar 3.7 Pengujian menampilkan informasi histogram citra berwarna untuk *caster.tif*

Sumber: Dokumen Penulis

B. Pengujian program untuk perbaikan kualitas pada citra

a. Perbaikan kualitas citra dengan *image brightening*

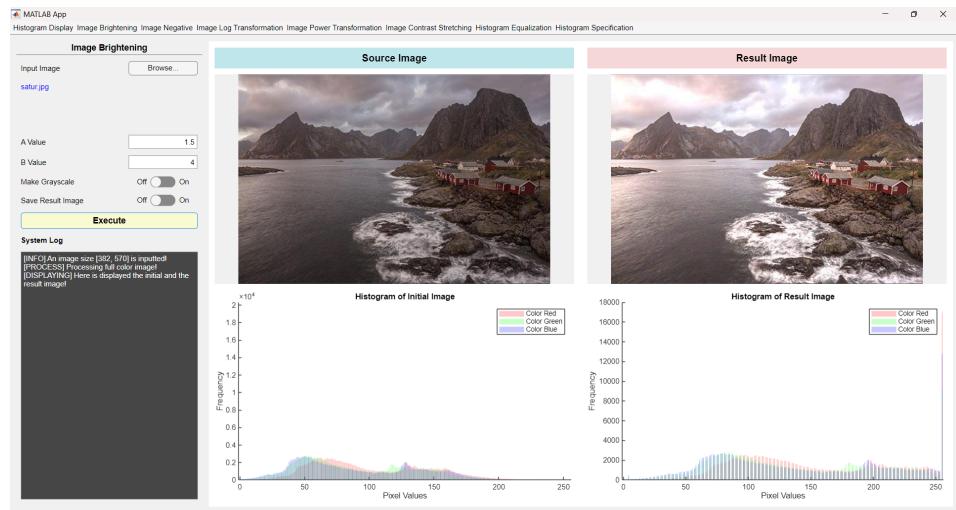
1. Gambar *peanut.tif* (*grayscale*)



Gambar 3.8 Pengujian perbaikan kualitas citra *image brightening* untuk *peanut.tif*

Sumber: Dokumen Penulis

2. Gambar *satur.jpg* (berwarna)

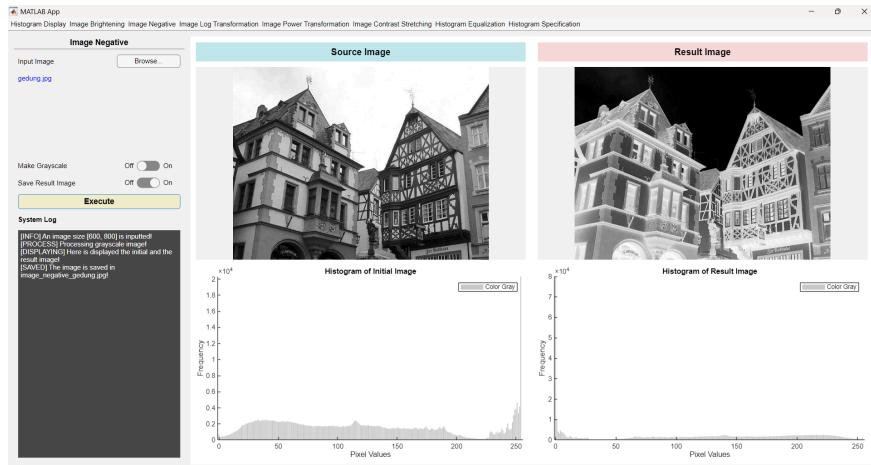


Gambar 3.9 Pengujian perbaikan kualitas citra *image brightening* untuk *satur.jpg*

Sumber: Dokumen Penulis

b. Perbaikan kualitas citra dengan citra negatif dan balikan citra negatif

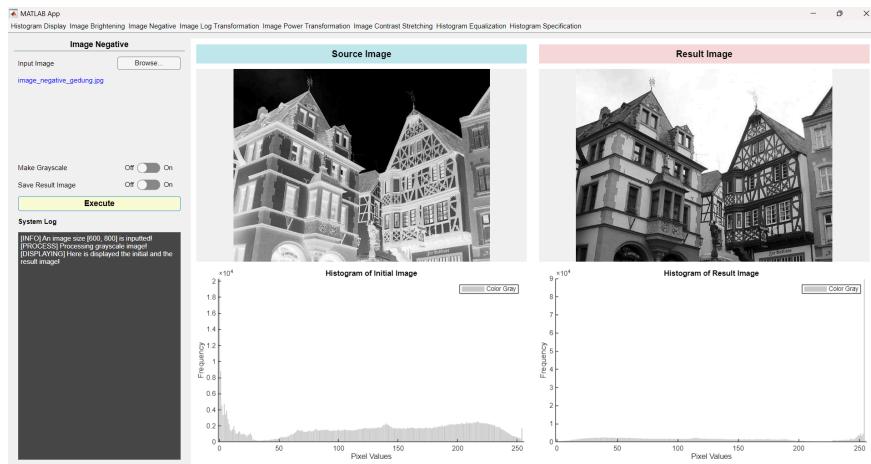
1. Citra negatif pada gambar *gedung.jpg* (*grayscale*)



Gambar 3.10 Pengujian perbaikan kualitas citra negatif untuk
gedung.jpg

Sumber: Dokumen Penulis

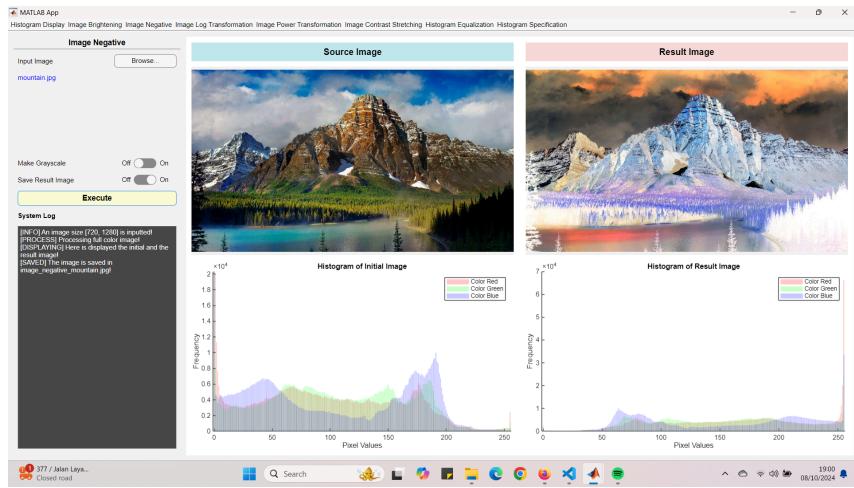
2. Citra balikan negatif pada gambar *gedung.jpg* (*grayscale*)



Gambar 3.11 Pengujian perbaikan kualitas citra balikan negatif untuk
gedung.jpg

Sumber: Dokumen Penulis

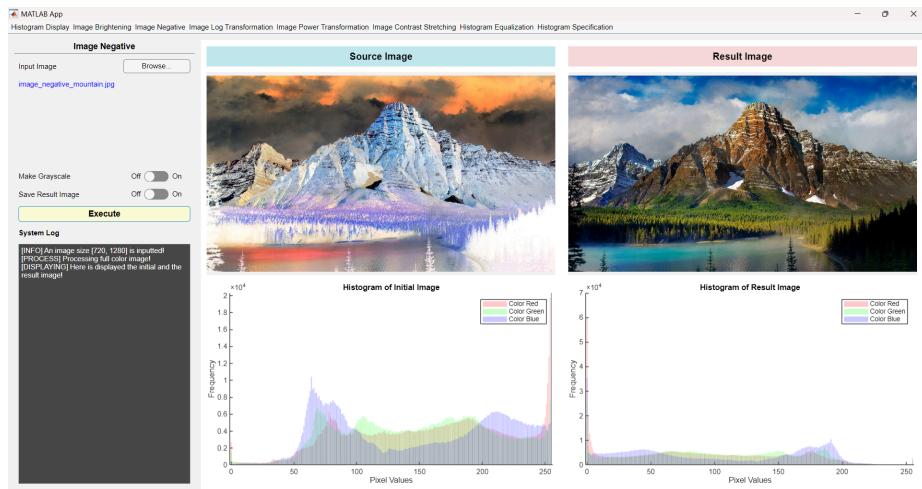
3. Citra negatif pada gambar *mountain.jpg* (berwarna)



Gambar 3.12 Pengujian perbaikan kualitas citra negatif untuk *mountain.jpg*

Sumber: Dokumen Penulis

4. Citra balikan negatif pada gambar *mountain.jpg* (berwarna)

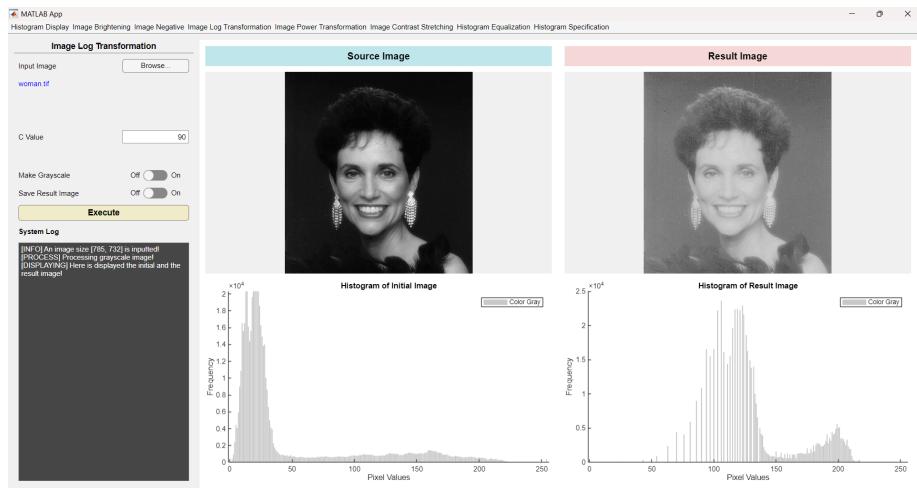


Gambar 3.13 Pengujian perbaikan kualitas citra balikan negatif untuk *mountain.jpg*

Sumber: Dokumen Penulis

c. Perbaikan kualitas citra dengan transformasi Log

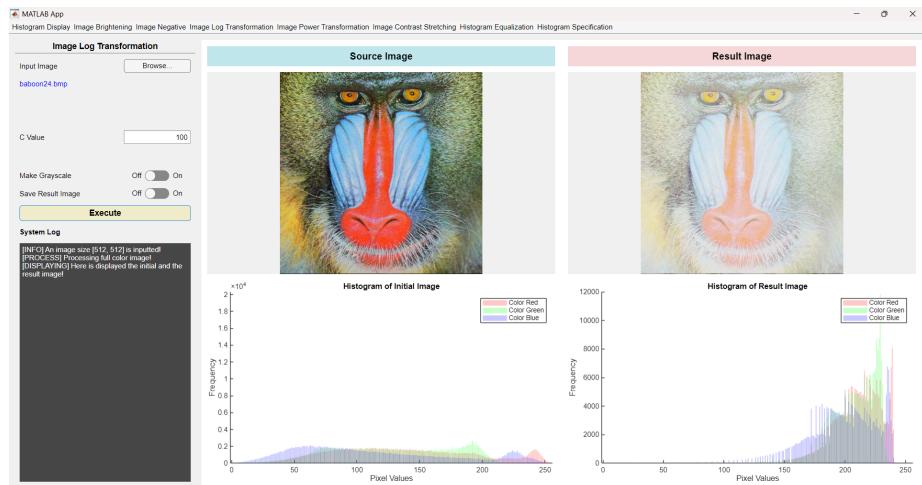
1. Transformasi log untuk gambar *woman.tif* (*grayscale*)



Gambar 3.14 Pengujian perbaikan kualitas citra transformasi log untuk *woman.tif*

Sumber: Dokumen Penulis

2. Transformasi log untuk gambar *baboon24.bmp*

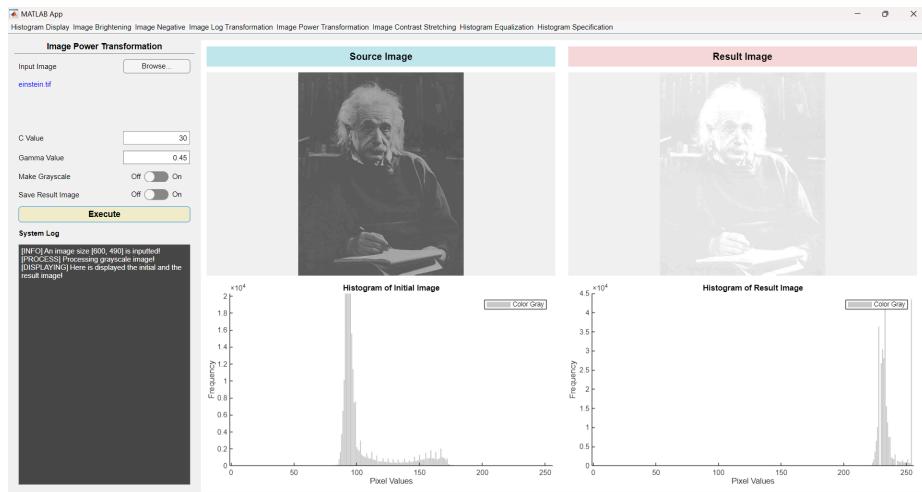


Gambar 3.15 Pengujian perbaikan kualitas citra transformasi log untuk *baboon24.bmp*

Sumber: Dokumen Penulis

d. Perbaikan kualitas citra dengan transformasi pangkat

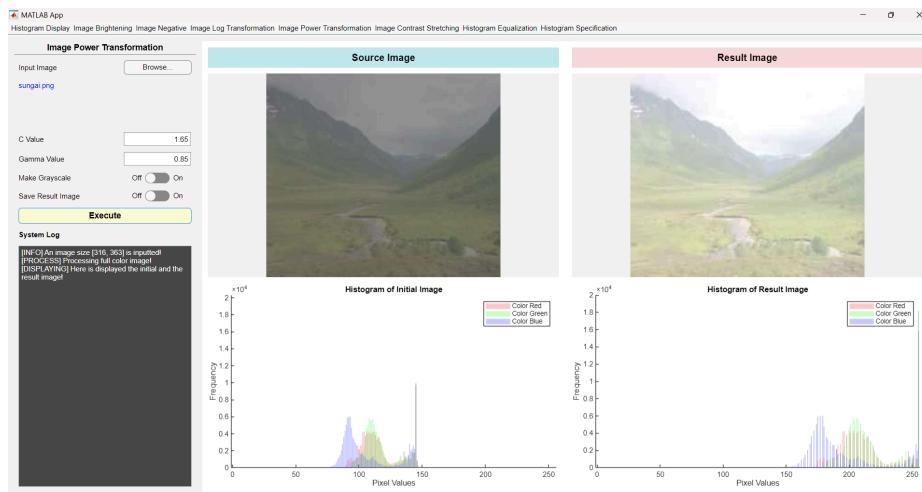
1. Transformasi pangkat untuk gambar *einstein.tiff* (*grayscale*)



Gambar 3.16 Pengujian perbaikan kualitas citra transformasi pangkat untuk *einstein.tiff*

Sumber: Dokumen Penulis

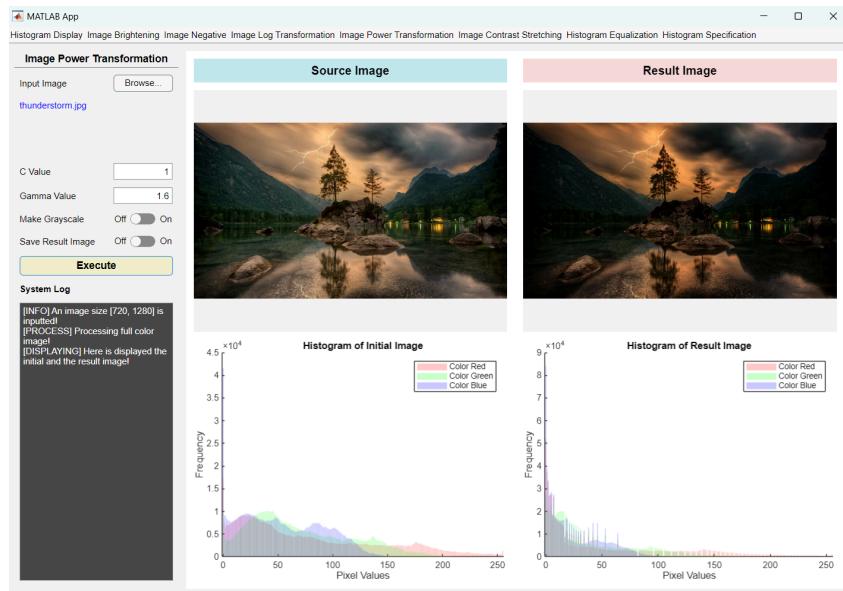
2. Transformasi pangkat untuk gambar *sungai.png* (berwarna)



Gambar 3.17 Pengujian perbaikan kualitas citra transformasi pangkat untuk *sungai.png*

Sumber: Dokumen Penulis

3. Transformasi pangkat untuk gambar *thunderstorm.png* (berwarna)

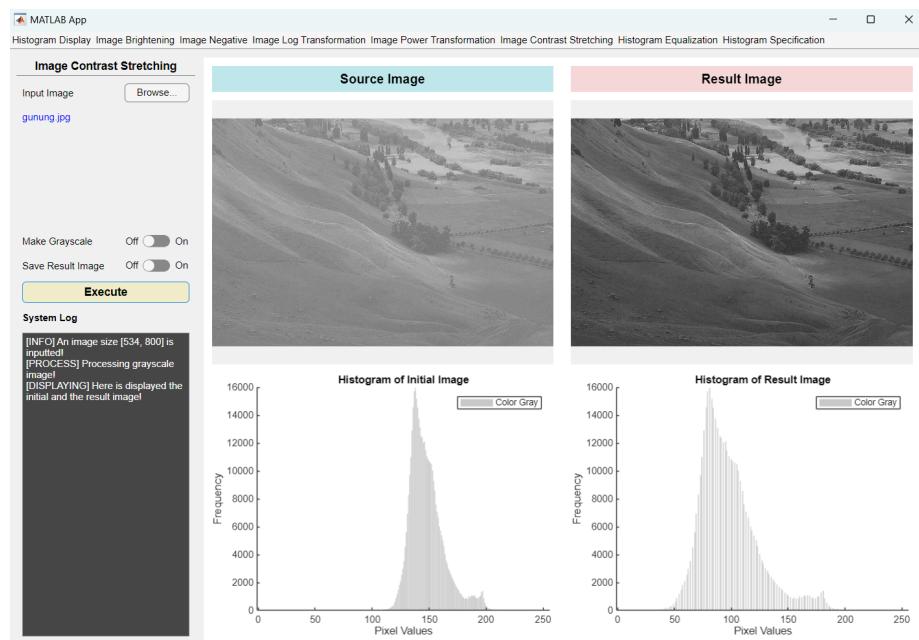


Gambar 3.18 Pengujian perbaikan kualitas citra transformasi pangkat untuk *sungai.png*

Sumber: Dokumen Penulis

e. Perbaikan kualitas citra dengan *Contrast Stretching*

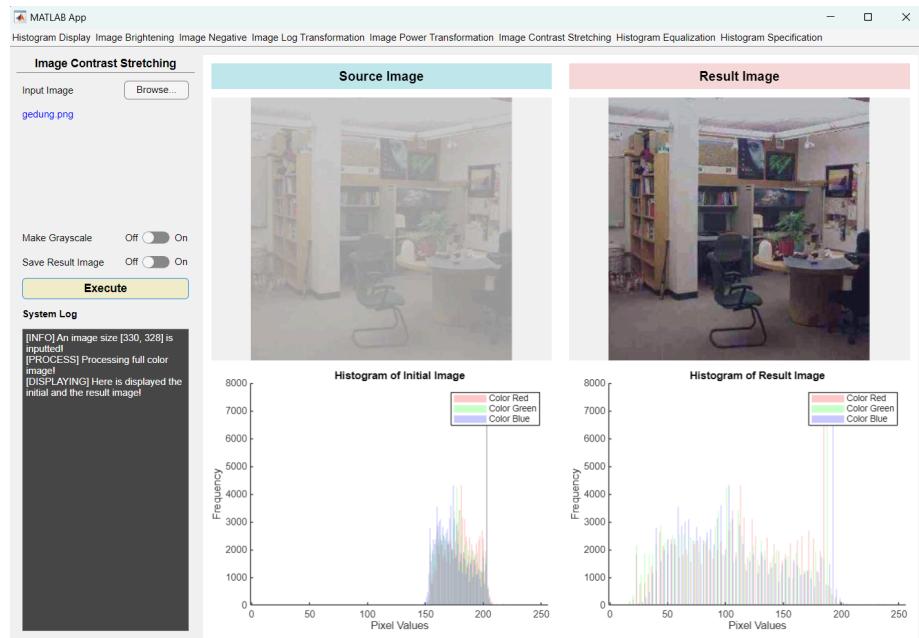
1. *Contrast Stretching* untuk gambar *gunung.jpg*



Gambar 3.19 Pengujian perbaikan kualitas citra *Contrast Stretching* untuk *gunung.jpg*

Sumber: Dokumen Penulis

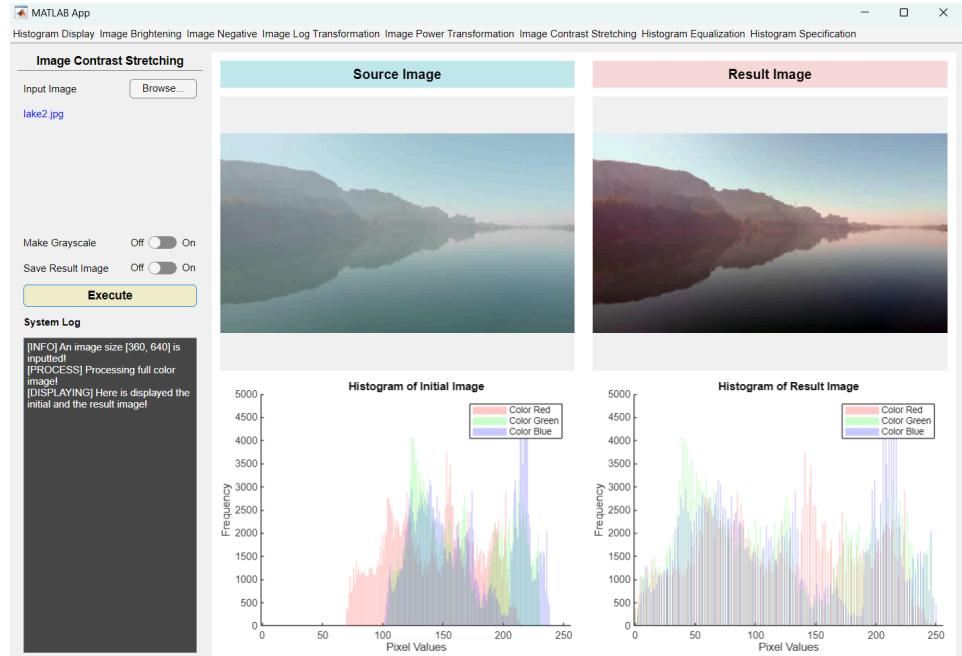
2. Contrast Stretching untuk gambar gedung.png



Gambar 3.20 Pengujian perbaikan kualitas citra *Contrast Stretching* untuk *gedung.png*

Sumber: Dokumen Penulis

3. Contrast Stretching untuk gambar lake2.jpg



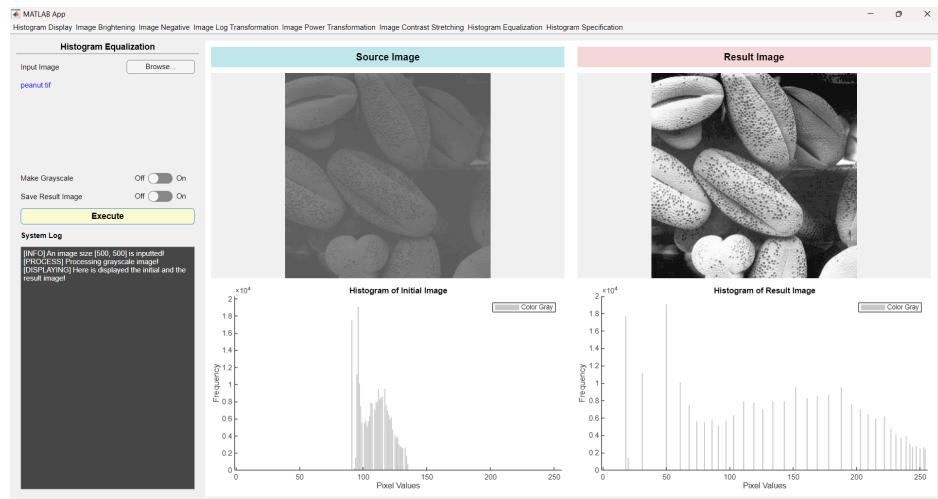
Gambar 3.21 Pengujian perbaikan kualitas citra *Contrast Stretching* untuk *lake2.jpg*

Sumber: Dokumen Penulis

C. Pengujian Program untuk perataan histogram (*Histogram Equalization*)

a. Pengujian citra *grayscale*

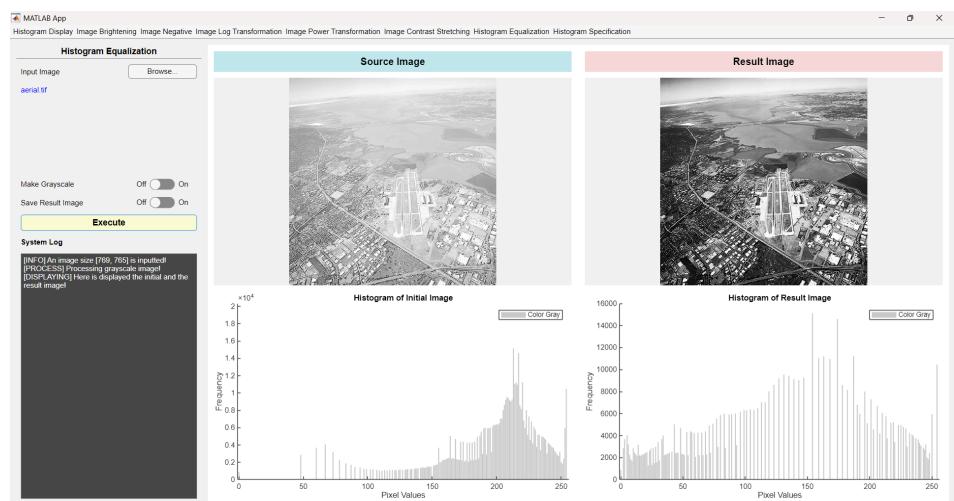
1. Gambar *peanut.tif*



Gambar 3.22 Pengujian *Histogram Equalization* untuk *peanut.tif*

Sumber: Dokumen Penulis

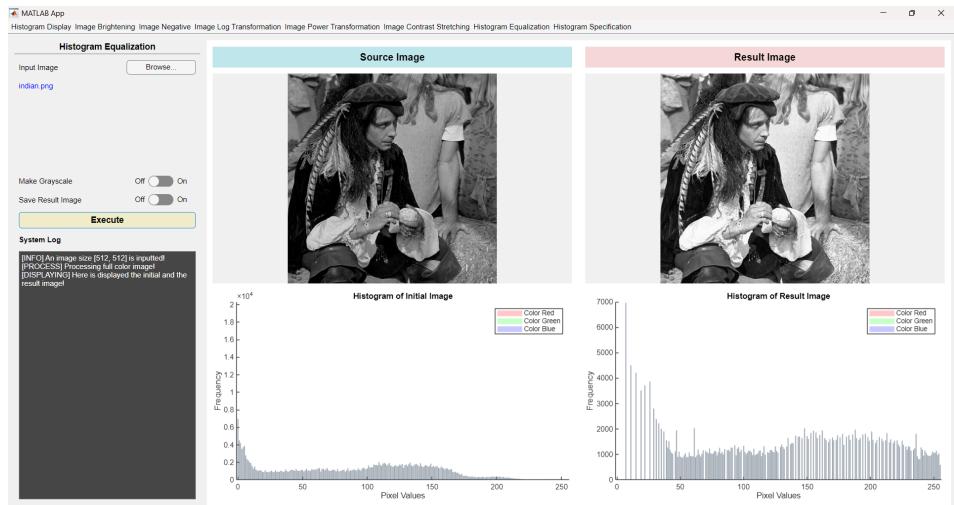
2. Gambar *aerial.tif*



Gambar 3.23 Pengujian *Histogram Equalization* untuk *aerial.tif*

Sumber: Dokumen Penulis

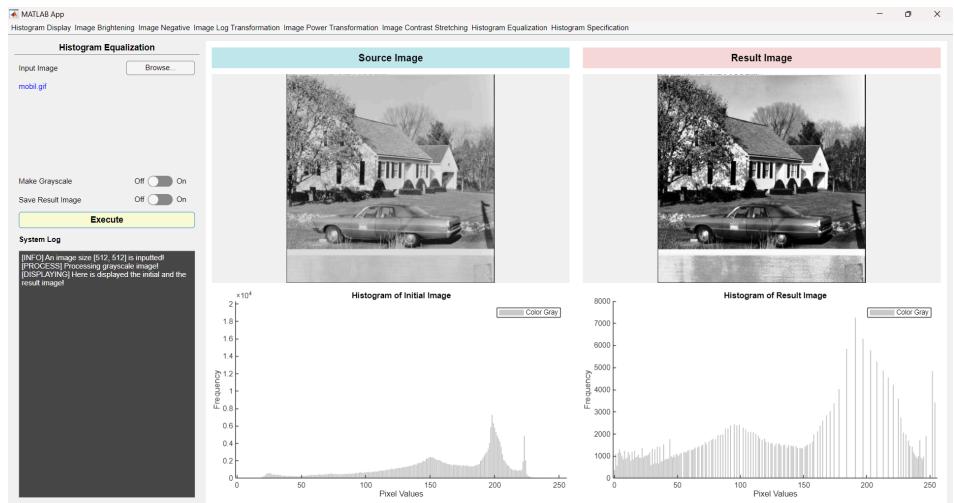
3. Gambar *indian.png*



Gambar 3.24 Pengujian *Histogram Equalization* untuk *indian.png*

Sumber: Dokumen Penulis

4. Gambar *mobil.gif*

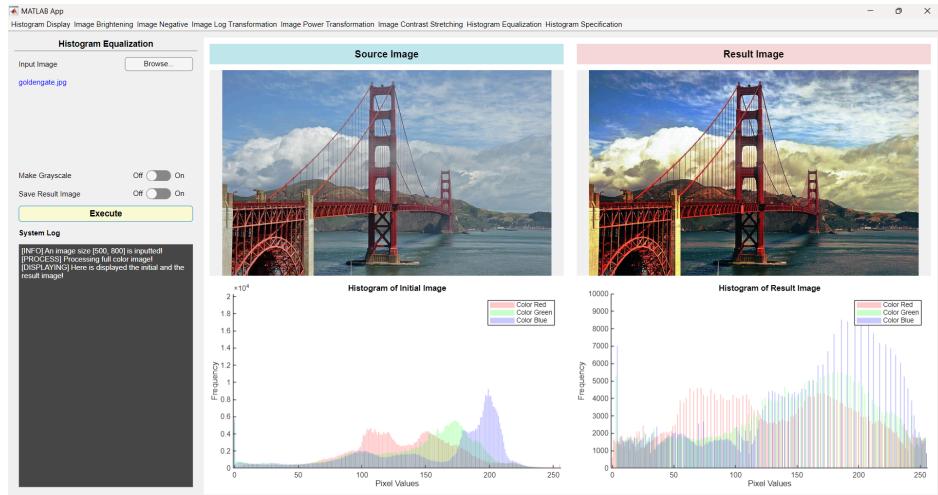


Gambar 3.25 Pengujian *Histogram Equalization* untuk *mobil.gif*

Sumber: Dokumen Penulis

b. Pengujian citra berwarna

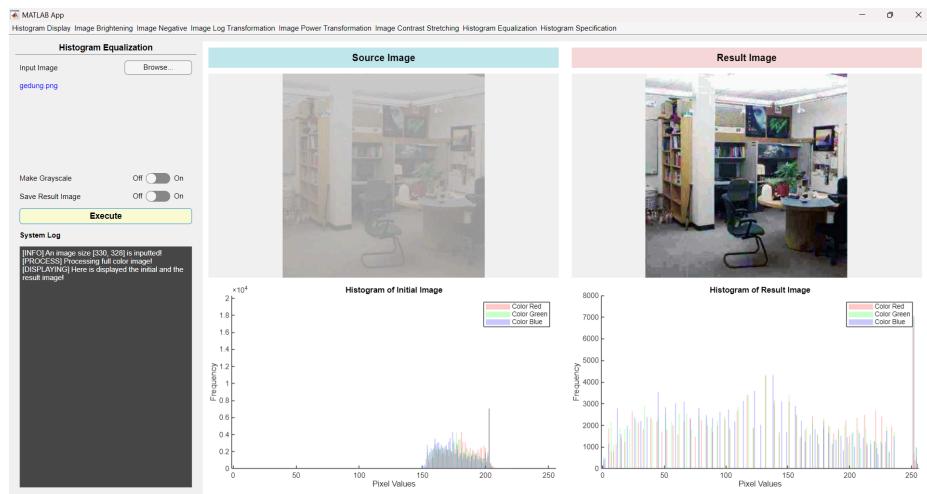
1. Gambar *goldengate.jpg*



Gambar 3.26 Pengujian *Histogram Equalization* untuk *goldengate.jpg*

Sumber: Dokumen Penulis

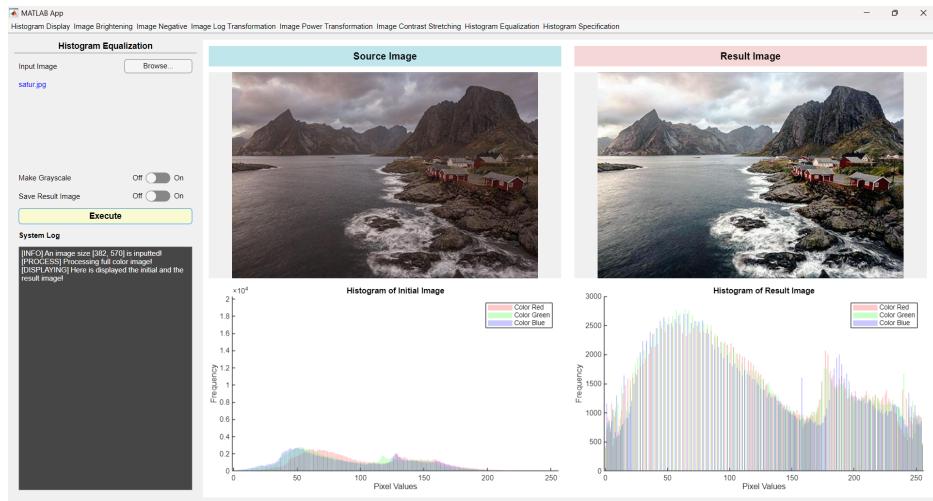
2. Gambar *gedung.png*



Gambar 3.27 Pengujian *Histogram Equalization* untuk *gedung.png*

Sumber: Dokumen Penulis

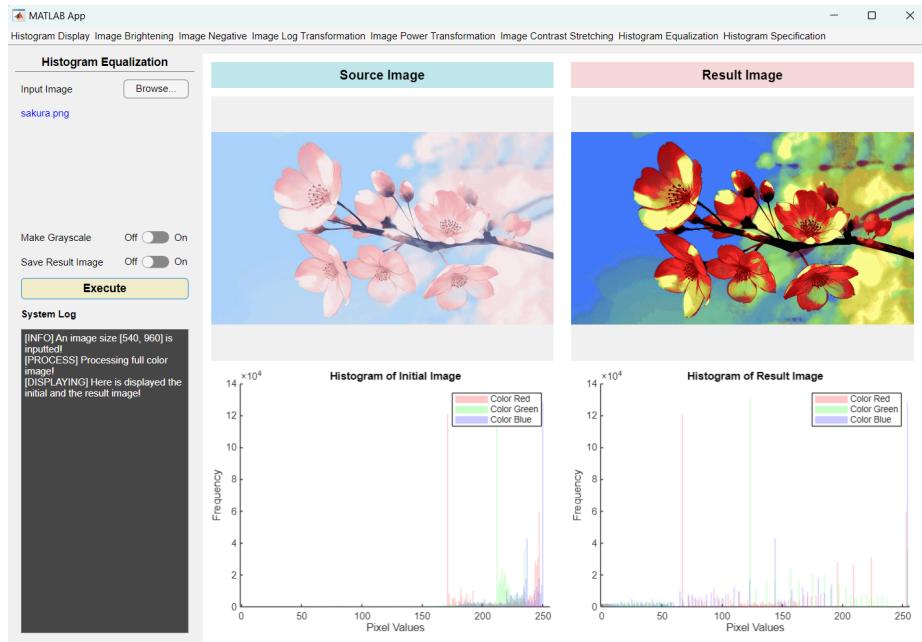
3. Gambar *satur.jpg*



Gambar 3.28 Pengujian *Histogram Equalization* untuk *satur.jpg*

Sumber: Dokumen Penulis

4. Gambar *sakura.png*



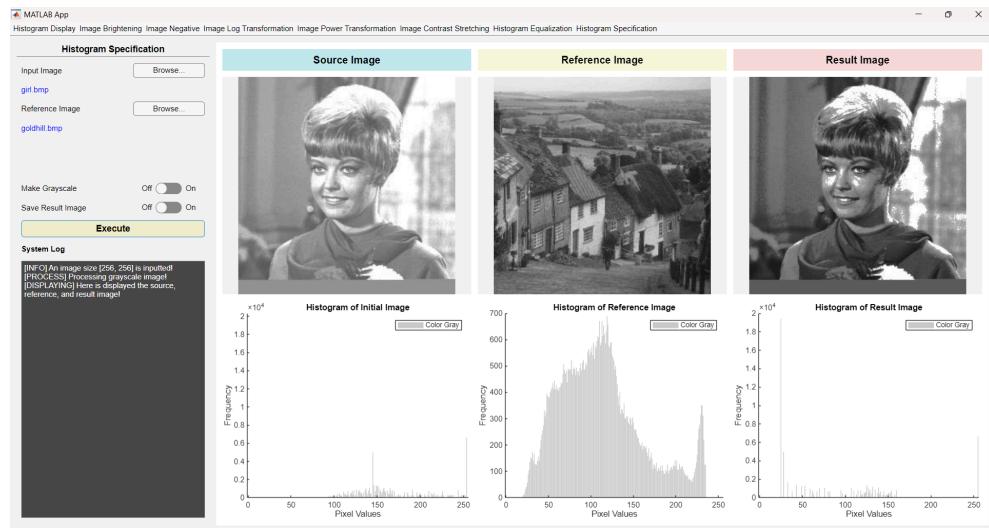
Gambar 3.29 Pengujian *Histogram Equalization* untuk *sakura.png*

Sumber: Dokumen Penulis

D. Pengujian Program untuk *Histogram Specification*

a. Pengujian citra grayscale (*girl.bmp* + *goldhill.bmp*)

Hasil 1:



Gambar 3.30 Hasil pertama pengujian *Histogram Specification* untuk
girl.bmp + *goldhill.bmp*

Sumber: Dokumen Penulis

Hasil 2:

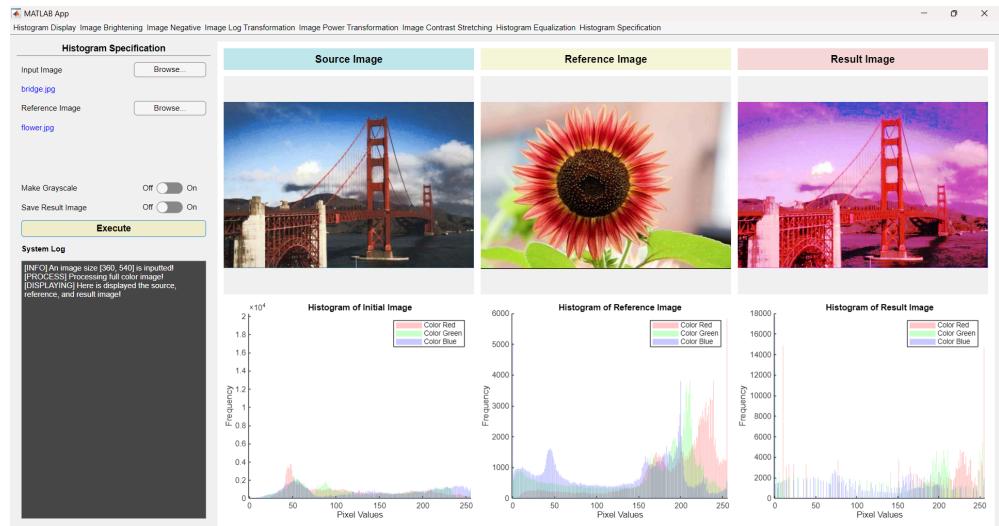


Gambar 3.31 Hasil kedua pengujian *Histogram Specification* untuk
girl.bmp + *goldhill.bmp*

Sumber: Dokumen Penulis

b. Pengujian citra berwarna (*bridge.jpg* + *flower.jpg*)

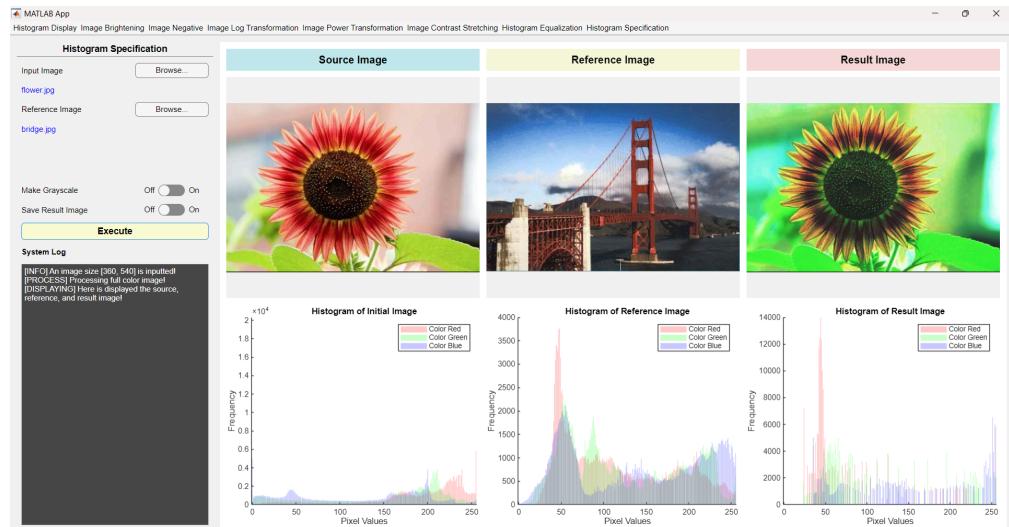
Hasil 1:



Gambar 3.32 Hasil pertama pengujian *Histogram Specification* untuk
bridge.jpg + *flower.jpg*

Sumber: Dokumen Penulis

Hasil 2:

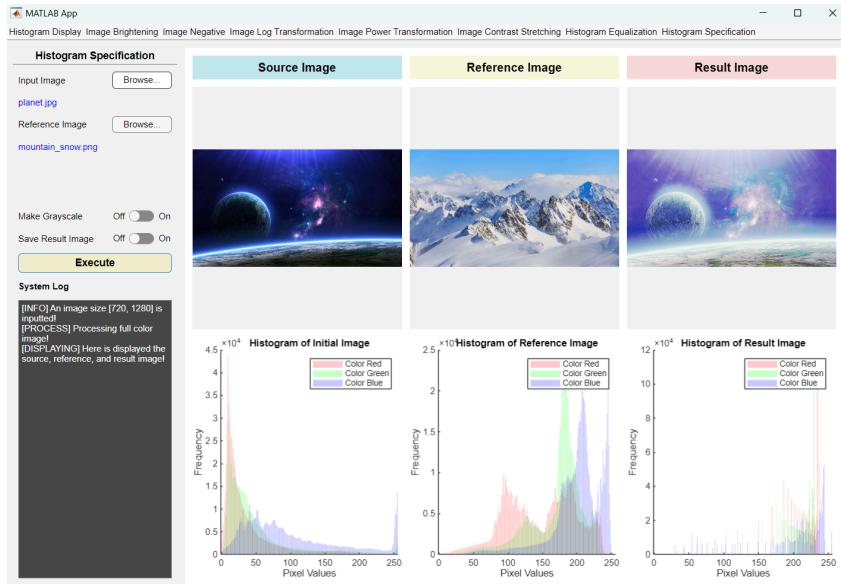


Gambar 3.33 Hasil kedua pengujian *Histogram Specification* untuk
bridge.jpg + *flower.jpg*

Sumber: Dokumen Penulis

c. Pengujian citra berwarna (*planet.jpg* + *mountain_snow.png*)

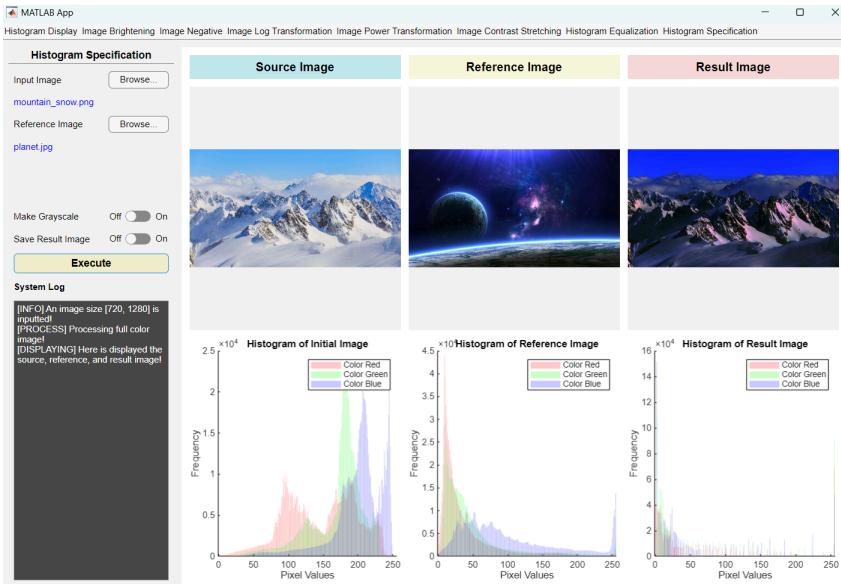
Hasil 1:



Gambar 3.34 Hasil pertama pengujian *Histogram Specification* untuk
planet.jpg + *mountain_snow.png*

Sumber: Dokumen Penulis

Hasil 2:



Gambar 3.35 Hasil kedua pengujian *Histogram Specification* untuk
mountain_snow.png + *planet.jpg*

Sumber: Dokumen Penulis

III. Analisis Cara Kerja Program

A. Analisis cara kerja program untuk menghitung dan menampilkan histogram

Kondisi perhitungan dan visualisasi histogram yang ada pada program yang dibuat yaitu melalui proses sebagai berikut:

Proses perhitungan sebaran titik warna pada program:

1. Program menerima input masukan berupa citra yang akan dilakukan pemrosesan
2. Setelah mendapatkan input citra, kemudian menghitung ukuran baris dan kolom untuk setiap pixel dalam citra (terdapat dua kondisi disini yang mungkin terjadi antara kondisi matriks hanya terdapat satu channel untuk citra Grayscale ataupun tiga channel untuk kondisi citra RGB)
3. Jika kondisi citra Grayscale, program akan melakukan looping untuk setiap pixel dan menghitung nilai kondisi warna yang terdapat pada pixel. Misalkan jika sebuah pixel memiliki nilai intensitas 150, maka nilai tersebut akan ditambahkan untuk perhitungan total dalam skala histogram
4. Cara yang sama juga kurang lebih dilakukan untuk kondisi citra RGB, namun untuk RGB, terdapat loop tambahan untuk menghitung intensitas warna pada setiap jenis pewarnaan mulai dari Red, Green, hingga Blue
5. Nilai yang telah didapatkan ini pun kemudian akan disimpan ke dalam bentuk array untuk divisualisasikan dalam bentuk grafik histogram

Proses untuk menampilkan persebaran titik warna citra pada grafik histogram:

1. Membagi visualisasi berdasarkan dua kondisi, ketika kondisi citra *grayscale* dan citra RGB.
2. Jika kondisi citra *grayscale*, program akan menampilkan histogram untuk gambar dengan sumbu x merupakan persebaran intensitas antara 0-255 dan sumbu y merupakan jumlah pixel yang terdistribusi dalam intensitas tersebut
3. Jika kondisi citra RGB, visualisasi untuk histogram kurang lebih sama untuk sumbu x dan sumbu y dengan informasi yang terdapat pada *grayscale*, namun kondisi ini juga akan menampilkan visualisasi

histogram citra untuk setiap jenis pewarnaan mulai dari Red, Green, hingga Blue

B. Analisis cara kerja program untuk perbaikan kualitas citra

a. Analisis untuk pencerahan citra (*image brightening*)

Berikut adalah langkah-langkah yang dilakukan dalam melakukan pemrosesan pencerahan citra:

1. Program input dari pengguna yakni: gambar yang akan diproses, variabel a, dan variabel b.
2. Program lalu melakukan pemrosesan sesuai dengan tipe citra (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada gambar.
3. Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
4. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
5. Setiap *pixel* dilakukan kalkulasi untuk menghasilkan *pixel* yang baru dengan rumus $new_pixel = a * pixel + b$
6. Setiap *pixel* yang dihasilkan akan dimasukkan ke dalam *image* baru yang telah dibentuk pada poin 3.
7. Program mengembalikan *image* baru tersebut sebagai citra *output*.

b. Analisis untuk citra negatif dan balikan citra negatif

Berikut adalah langkah-langkah yang dilakukan dalam melakukan pemrosesan citra negatif:

1. Program input dari pengguna yakni gambar yang akan diproses.
2. Program lalu melakukan pemrosesan sesuai dengan tipe citra (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada gambar.
3. Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
4. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.

5. Setiap *pixel* dilakukan kalkulasi untuk menghasilkan *pixel* yang baru dengan rumus $new_pixel = 255 - pixel$.
6. Setiap *pixel* yang dihasilkan akan dimasukkan ke dalam *image* baru yang telah dibentuk pada poin 3.
7. Program mengembalikan *image* baru tersebut sebagai citra *output*.

c. Analisis untuk transformasi log

Berikut adalah langkah-langkah yang dilakukan dalam melakukan pemrosesan transformasi log:

1. Program input dari pengguna yakni: gambar yang akan diproses dan variabel *c*.
2. Program lalu melakukan pemrosesan sesuai dengan tipe citra (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada gambar.
3. Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
4. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
5. Setiap *pixel* dilakukan kalkulasi untuk menghasilkan *pixel* yang baru dengan rumus $new_pixel = c * \log(1 + pixel)$.
6. Setiap *pixel* yang dihasilkan akan dimasukkan ke dalam *image* baru yang telah dibentuk pada poin 3.
7. Program mengembalikan *image* baru tersebut sebagai citra *output*.

d. Analisis untuk transformasi pangkat

Berikut adalah langkah-langkah yang dilakukan dalam melakukan pemrosesan transformasi pangkat:

1. Program input dari pengguna yakni: gambar yang akan diproses, variabel *c*, dan variabel gamma (γ).
2. Program lalu melakukan pemrosesan sesuai dengan tipe citra (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada gambar.

3. Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
 4. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
 5. Setiap *pixel* dilakukan kalkulasi untuk menghasilkan *pixel* yang baru dengan rumus $new_pixel = c * r^y$.
 6. Setiap *pixel* yang dihasilkan akan dimasukkan ke dalam *image* baru yang telah dibentuk pada poin 3.
 7. Program mengembalikan *image* baru tersebut sebagai citra *output*.
- e. Analisis untuk peregangan kontras (*contrast stretching*)

Berikut adalah langkah-langkah yang dilakukan dalam melakukan pemrosesan peregangan kontras:

1. Program input dari pengguna yakni gambar yang akan diproses.
2. Program lalu melakukan pemrosesan sesuai dengan tipe citra (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada gambar.
3. Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
4. Program melakukan pencarian nilai maksimum (*max_val*) dan nilai minimum (*min_val*) dari *pixel* yang tertera pada gambar.
5. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
6. Setiap *pixel* dilakukan kalkulasi untuk menghasilkan *pixel* yang baru dengan rumus

$$new_pixel = (pixel - min_val) * (255 / (max_val - min_val))$$
7. Setiap *pixel* yang dihasilkan akan dimasukkan ke dalam *image* baru yang telah dibentuk pada poin 3.
8. Program mengembalikan *image* baru tersebut sebagai citra *output*.

C. Analisis cara kerja program untuk *histogram equalization*

Kondisi perhitungan dan visualisasi histogram yang ada pada program yang dibuat yaitu melalui proses sebagai berikut:

Proses keseluruhan untuk perataan histogram:

1. Program menerima input masukan berupa citra yang akan dilakukan pemrosesan
2. Proses perataan histogram yang ada pada program dibagi menjadi dua kondisi, yaitu untuk kondisi citra *Grayscale* dan citra RGB. Untuk keduanya proses yang dilakukan kurang lebih sama namun pada citra RGB, perataan dilakukan dengan memisahkan kondisi jenis pewarnaan antara Red, Green, dan Blue dan diratakan secara masing-masing untuk setiap pewarnaan sebelum digabungkan kembali di akhir
3. Proses perataan histogram dilakukan dengan memanggil fungsi *equalizer_operation* untuk meratakan histogram (alur fungsi ini akan dijelaskan dibawah)
4. Setelah histogram diratakan untuk kondisi *grayscale* maupun kondisi RGB, gambar baru akan disimpan dan kemudian akan dilakukan visualisasi terhadap histogramnya

Proses perataan histogram dengan fungsi *equalizer_operation*:

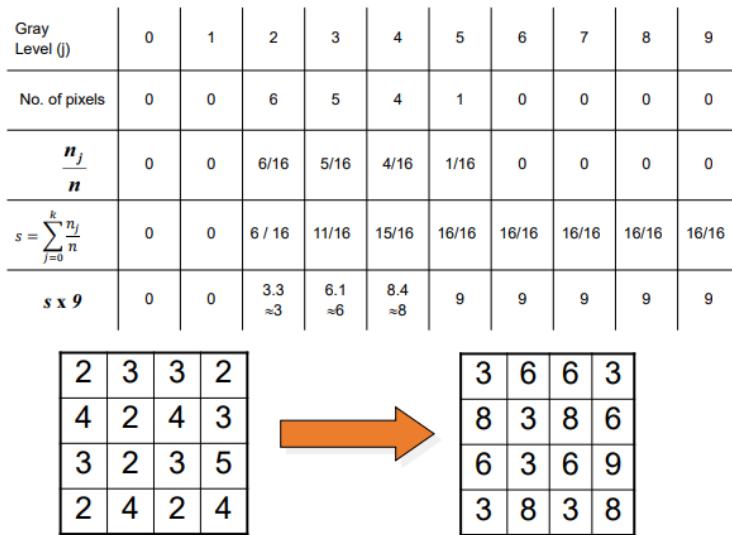
1. Fungsi akan menerima parameter berupa matriks intensitas warna dengan nilai yang dimiliki antara 0-255. Setelah menerima parameter matriks, akan dicari nilai total pixel berdasarkan matriks keseluruhan
2. Setelahnya akan dilakukan perhitungan secara manual terhadap jumlah pixel antara 0-255 dengan proses looping.
3. Dari hasil perhitungan pixel, selanjutnya dilakukan perhitungan dengan metode *Probability Density Function* (PDF) dengan proses yaitu membagi nilai pixel yang terdapat pada intensitas tertentu (n_j) dibagi dengan jumlah total pixel (n). Sehingga rumus yang ada adalah sebagai berikut:

$$\frac{n_j}{n}$$

4. Berikutnya, melakukan perhitungan dengan metode *Cumulative Distribution Function* (CDF) dengan menjumlahkan nilai dari PDF sebelumnya mulai dari 0-255
5. Kemudian, dilakukan transformasi terhadap gambar yang telah diratakan nilainya dengan mengalikan nilai CDF dengan keseluruhan range intensitas warna yaitu 255

- Dari hasil yang didapat ini, lakukan transformasi kembali untuk menyusun nilai ke dalam bentuk matriks citra kembali sama seperti awal

Referensi dari keseluruhan proses perataan histogram dapat dilihat pada gambar berikut:



Gambar 3.31 Gambaran proses dari implementasi *Histogram Equalization*

Sumber: Ali Javed, Digital Image Processing, Chapter # 3, Image Enhancement in Spatial Domain

D. Analisis cara kerja program untuk *histogram specification*

Berikut adalah langkah-langkah yang dilakukan program dalam melakukan proses *histogram specification*:

- Program input dari pengguna yakni: gambar yang akan diproses dan gambar yang akan dijadikan acuan/ referensi.
- Program lalu melakukan pemrosesan sesuai dengan tipe kedua citra tersebut (*grayscale* atau berwarna) dimulai dari pembacaan ukuran dan *pixel* pada masing-masing citra.
- Program melakukan pengecekan untuk memastikan bahwa ukuran kedua citra dan jenis ukuran kedua citra sama.
- Program membuat sebuah *image* baru dengan ukuran yang sama yang akan digunakan untuk menampung setiap *pixel* hasil pemrosesan.
- Setiap citra akan dilakukan proses serupa dengan proses perataan histogram (*histogram equalization*), yang dapat dijabarkan sebagai berikut:

- a. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
- b. Untuk frekuensi kemunculan *pixel* untuk setiap nilai *pixel* (0 hingga 255) pada gambar akan dicatat dalam sebuah *list*.
- c. Dari hasil perhitungan frekuensi kemunculan *pixel*, selanjutnya dilakukan perhitungan dengan metode *Probability Density Function* (PDF) dengan proses yaitu membagi setiap nilai kemunculan *pixel* (n_j) dengan jumlah total *pixel* (n). Rumus tersebut dapat dituliskan sebagai berikut:

$$\frac{n_j}{n}$$

- d. Berikutnya, program melakukan perhitungan dengan metode *Cumulative Distribution Function* (CDF) dengan menjumlahkan nilai kemunculan dari *pixel* pertama (n_0) hingga *pixel* tertentu (n_j).
- e. Kemudian, program melakukan transformasi terhadap gambar yang telah diratakan nilainya dengan mengalikan nilai CDF dengan keseluruhan range intensitas warna yaitu 255.
- f. Program melakukan pembulatan terhadap hasil perkalian tersebut sehingga dihasilkan rentang nilai *pixel* seperti serupa, yakni 0-255.
- g. Hasil akhir yang didapat ini adalah nilai histogram yang telah diratakan (*equalized histogram*) dan dikembalikan oleh program ke program utama.
- 6. Untuk setiap nilai pada *equalized histogram* citra pertama dicocokkan dengan *equalized histogram* citra kedua. Proses pencocokan dilakukan pada *equalized histogram* tetapi hasil akhir adalah *mapping* yang dilakukan dari nilai *pixel* citra pertama ke nilai *pixel* citra kedua.



Gambar 3.32 Gambaran proses dari implementasi *Histogram Matching*

Sumber: Ali Pourramezan Fard, TowardDataScience, Histogram Matching

7. Program melakukan iterasi untuk setiap baris *pixel* yang ada pada gambar dan setiap *pixel* yang ada pada kolom baris tersebut.
8. Untuk setiap *pixel* yang berada pada citra pertama dilakukan proses *mapping* berdasarkan *mapping* yang telah dibentuk pada poin 6. Hasil mapping ini dimasukkan sebagai *pixel* pada baris dan kolom tertentu pada *image* baru yang dihasilkan.
9. Program mengembalikan *image* baru tersebut sebagai citra *output*.

BAB IV

KESIMPULAN

Kesimpulan yang didapat dari penggerjaan tugas ini adalah sebagai berikut:

1. Histogram dapat dipergunakan untuk menampilkan informasi detail terkait dengan intensitas yang dimiliki oleh citra. Hal ini dapat membantu untuk memberikan gambaran terkait dengan kondisi citra untuk proses perbaikan.
2. Perbaikan kualitas citra (*image enhancement*) dapat dilakukan dengan beberapa cara seperti pencerahan citra (*image brightening*), pembalikan citra atau citra negatif, transformasi log, transformasi pangkat, serta dengan peregangan kontras (*contrast stretching*).
3. Perbaikan kualitas citra (*image enhancement*) juga dapat dilakukan menggunakan histogram dari citra itu sendiri, yaitu melalui proses *histogram equalization* untuk perataan histogram dan juga *histogram specification* untuk penyamaan histogram.

DAFTAR PUSTAKA

- [1] “Histogram matching in digital image processing.” Youtube. <https://www.youtube.com/watch?v=r565euxWZBs> (Diakses pada tanggal 5 Oktober 2024)
- [2] “Histogram Citra” Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/06-Image-Histogram-2024.pdf> (Diakses pada tanggal 5 Oktober 2024)
- [3] “Image Enhacement Bagian 1”. Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/08-Image-Enhancement-Bagian1-2024.pdf> (Diakses pada tanggal 5 Oktober 2024)
- [4] “Image Enhacement Bagian 2”. Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/08-Image-Enhancement-Bagian1-2024.pdf> (Diakses pada tanggal 5 Oktober 2024)

LAMPIRAN

Pranala Github: https://github.com/Gulilil/IF4073_Tugas1