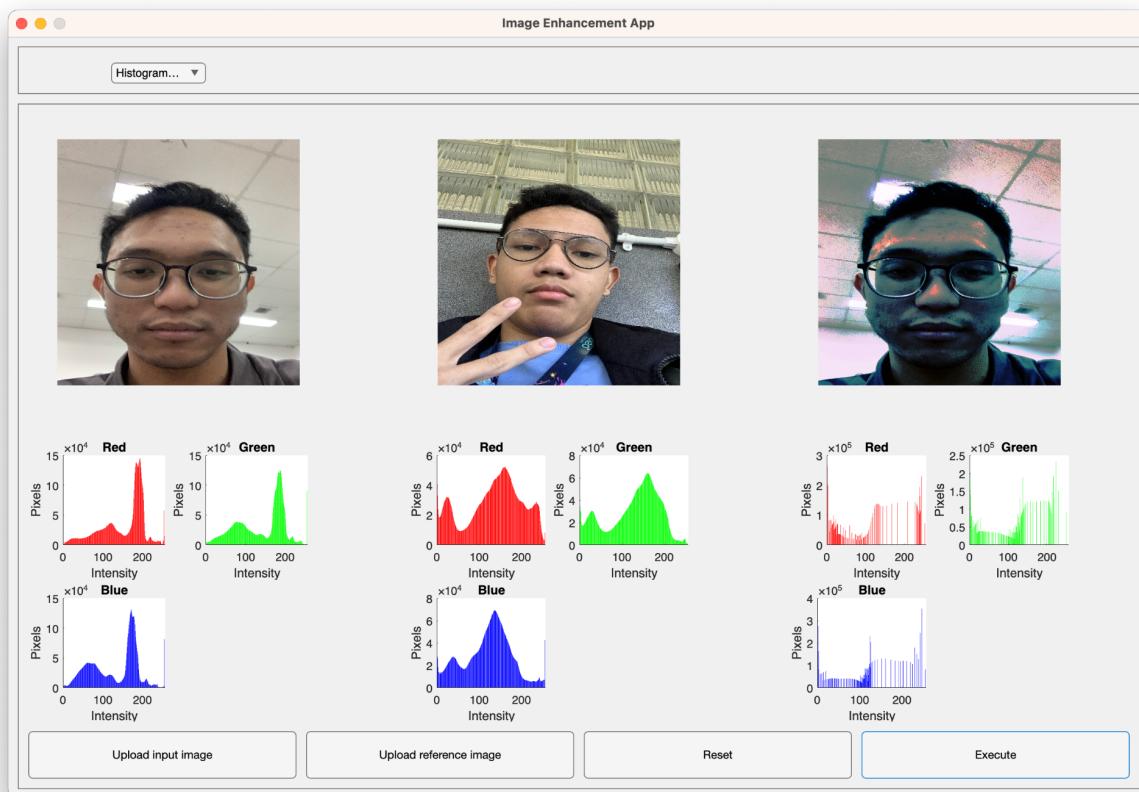


Tugas 1 IF4073 Interpretasi dan Pengolahan Citra



disusun oleh:

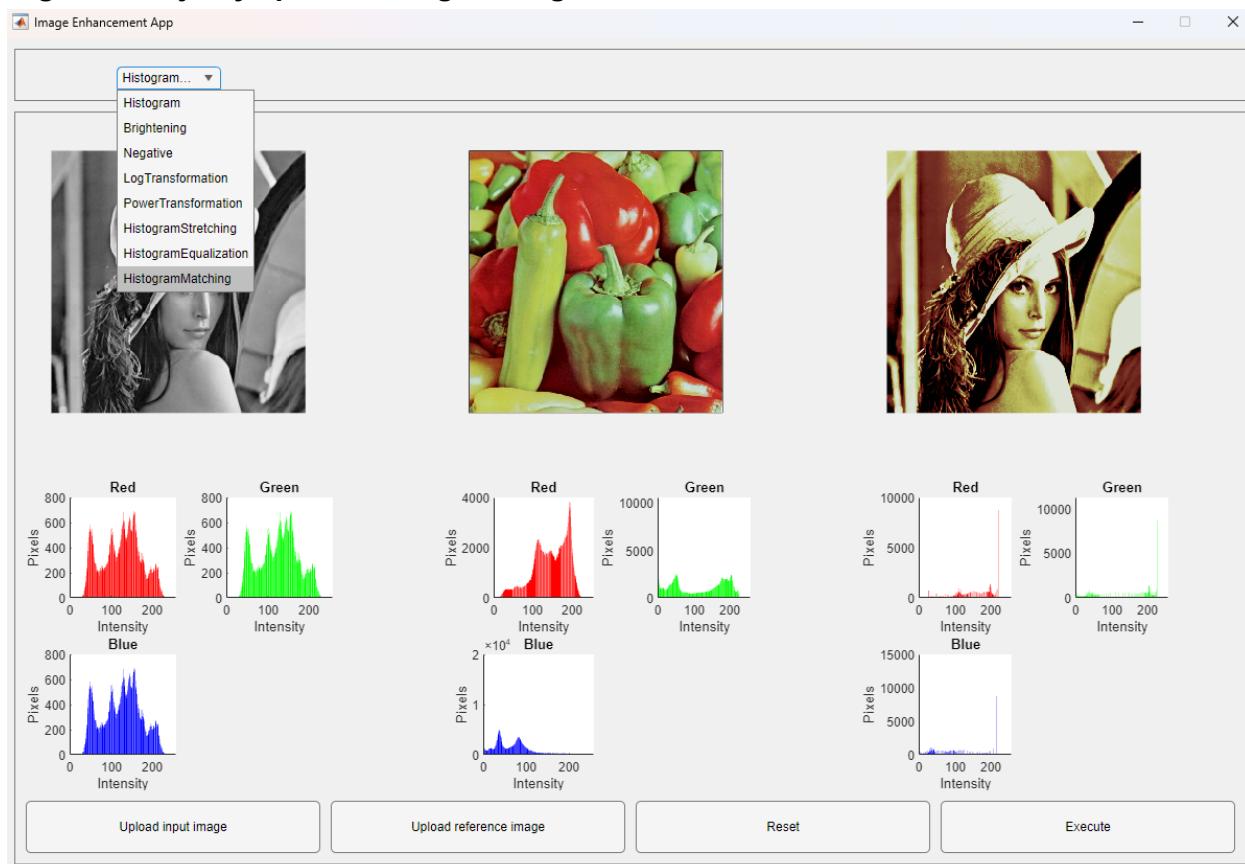
13520065 Rayhan Kinan Muhamnad
13520119 Marchotridyo

Screenshot GUI program

Berikut disajikan laman GUI program yang kami buat secara umum. Dalam GUI program ini, terdapat:

1. Sebuah *pull-down menu* untuk mengganti halaman. Halaman yang bisa dipilih antara lain halaman Histogram, Brightening, Negative, LogTransformation, PowerTransformation, HistogramStretching, HistogramEqualization, dan HistogramMatching.
2. Panel utama yang menampilkan gambar-gambar serta histogramnya.
3. Tombol-tombol dan/atau *input field* untuk pengguna melakukan interaksi terhadap halaman, seperti untuk meng-*upload* gambar atau mengatur parameter-parameter perubahan.

Untuk lebih detailnya, GUI untuk setiap fitur yang kami buat akan kami lampirkan pada bagian selanjutnya pada masing-masing fitur.



Program 1: Menampilkan histogram

Pada bagian ini, kami akan menjelaskan detail program pertama kami yaitu program untuk menampilkan histogram dari gambar.

Kode program

Berikut adalah fungsi histogram yang kami buat.



```
function histogram = hist(imageData)
    arguments
        imageData uint8
    end % arguments

    % Make a 1 dimensional array of zeros with length 256
    histogram = zeros(1, 256);

    % Loop through the image
    [m, n] = size(imageData);
    for i = 1:m
        for j = 1:n
            % val + 1 because of arrays in MATLAB are one-indexed
            % not zero-indexed
            val = imageData(i, j);
            histogram(val + 1) = histogram(val + 1) + 1;
        end
    end
end % hist
```

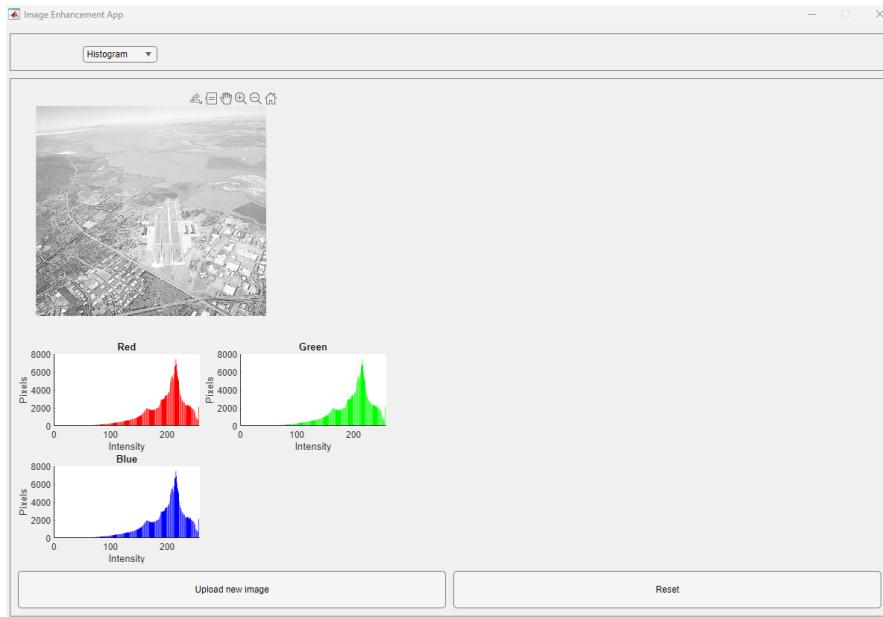
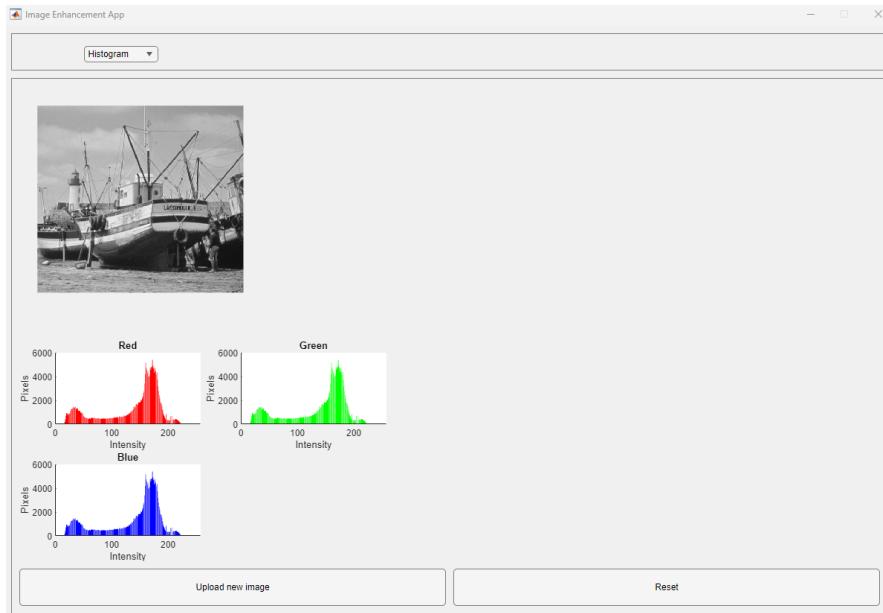
Dan berikut adalah pemanggilannya baik untuk gambar berwarna maupun gambar grayscale:

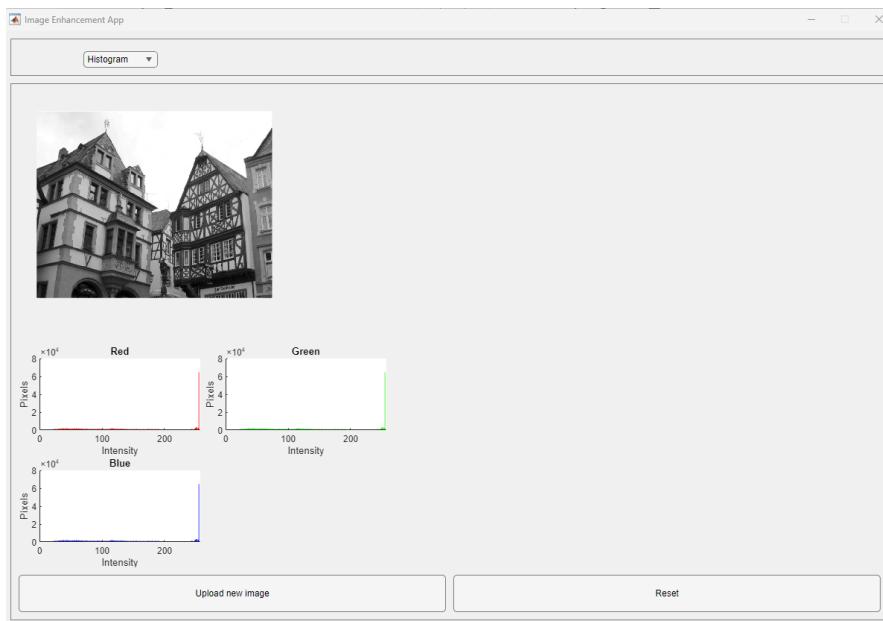
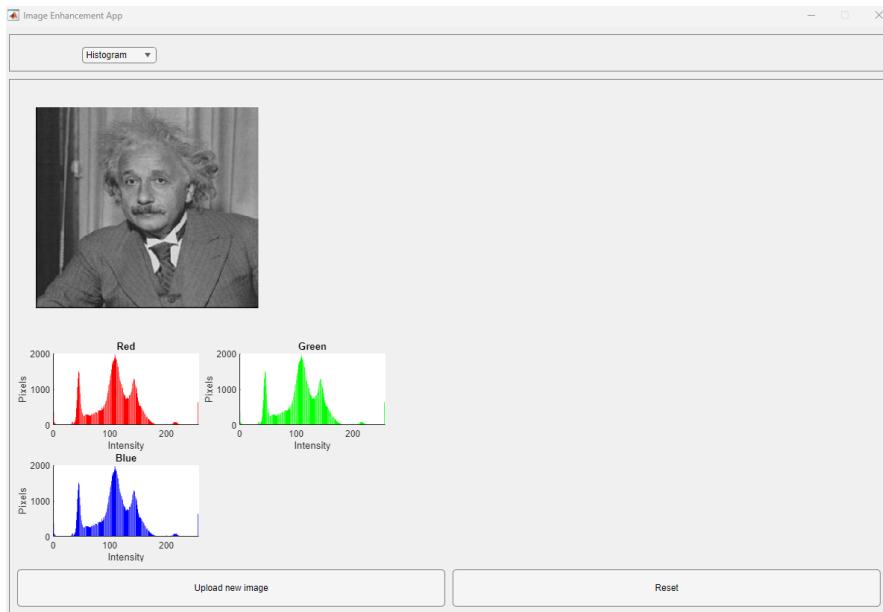
```
1      % Get Histogram
2      function [histRed, histGreen, histBlue] = GetHistogram(obj)
3          arguments
4              obj wrappers.ColoredImageWrapper
5          end
6
7          histRed = utils.Histogram.hist(obj.ImageData(:, :, 1));
8          histGreen = utils.Histogram.hist(obj.ImageData(:, :, 2));
9          histBlue = utils.Histogram.hist(obj.ImageData(:, :, 3));
10         end
11
```

```
1      % Get Histogram
2      function [histRed, histGreen, histBlue] = GetHistogram(obj)
3          arguments
4              obj wrappers.GrayscaleImageWrapper
5          end
6
7          histRed = utils.Histogram.hist(obj.ImageData);
8          histBlue = utils.Histogram.hist(obj.ImageData);
9          histGreen = utils.Histogram.hist(obj.ImageData);
10         end
```

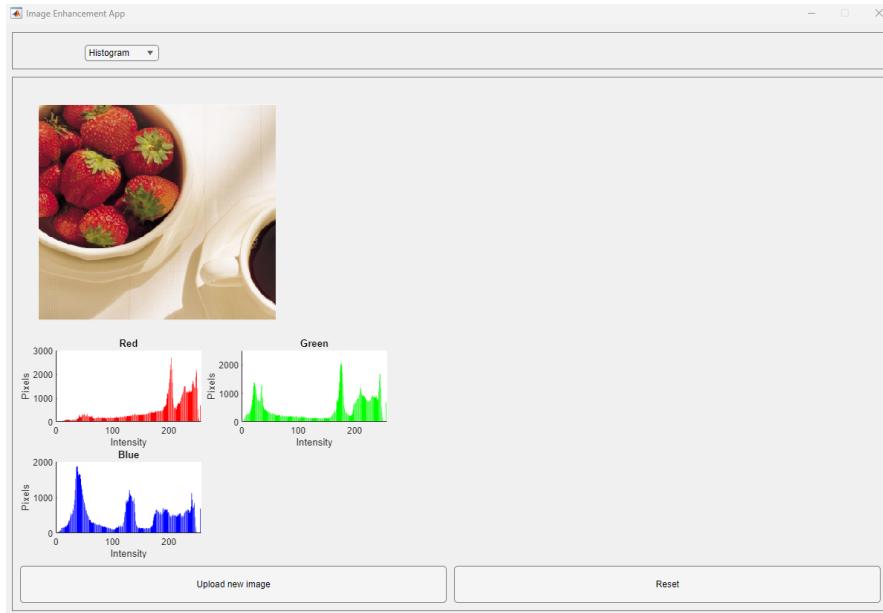
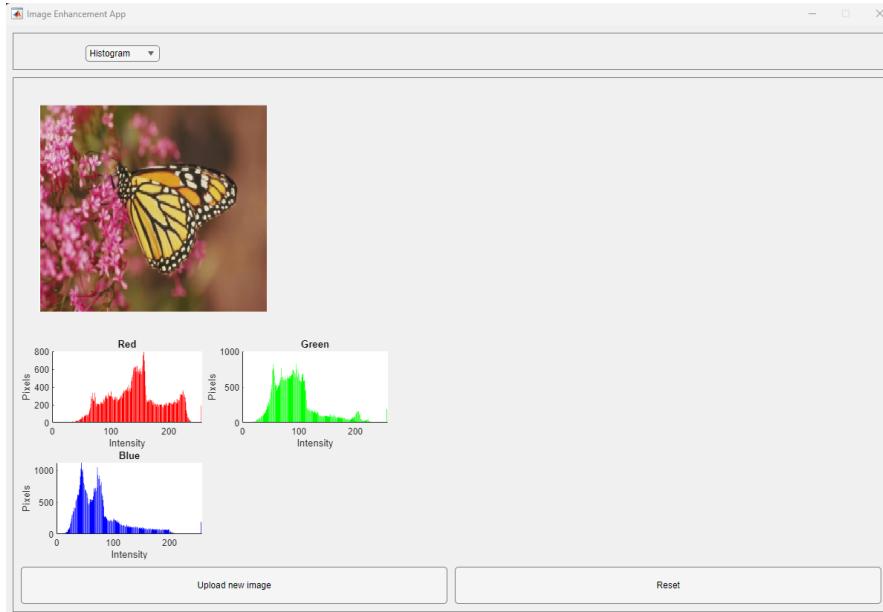
Contoh hasil eksekusi program

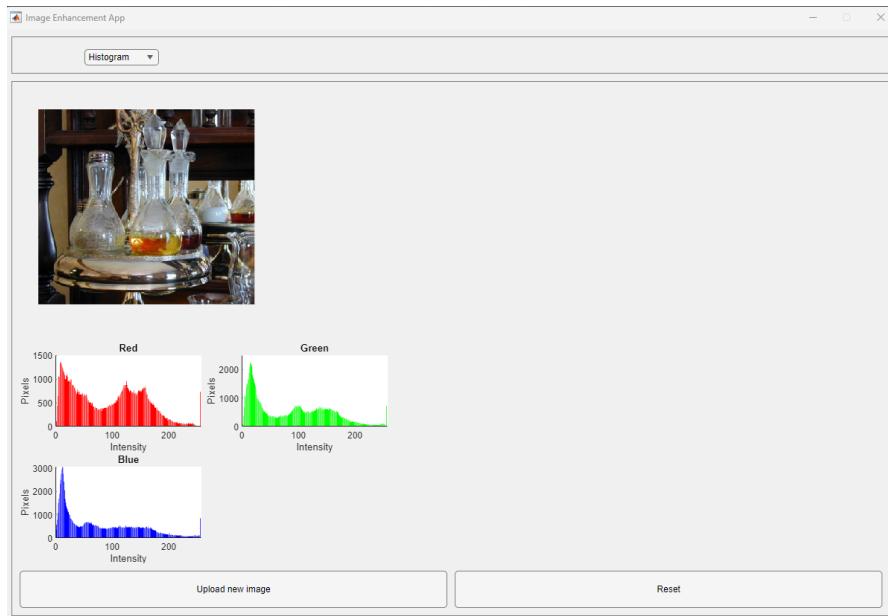
Eksekusi pada gambar grayscale





Eksekusi pada gambar berwarna





Analisis cara kerja algoritma

Fungsi `hist` secara umum bekerja dengan cara berikut:

- Inisialisasi sebuah larik bernama `histogram`. `histogram[i]` menyatakan frekuensi kemunculan pixel dengan gray level `i`.
- Iterasi terhadap semua pixel pada gambar lalu catat kemunculan gray levelnya dengan cara memodifikasi larik `histogram`.

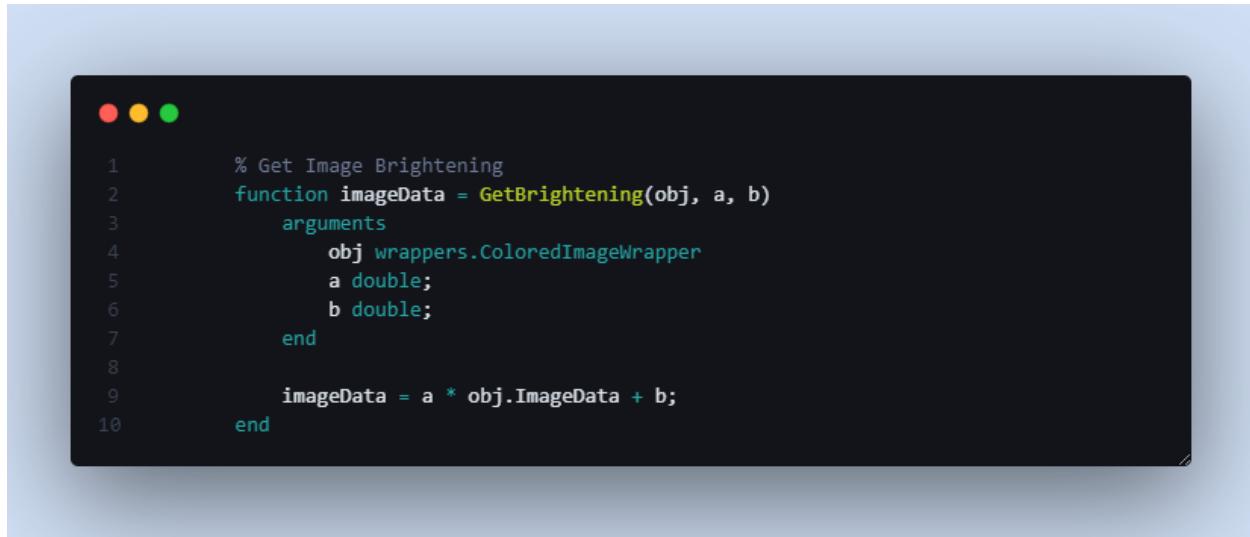
Apabila gambar berukuran $M \times N$, maka kompleksitasnya adalah $O(M \times N)$.

Perhatikan bahwa suatu image berwarna tersusun atas 3 layer warna, ada 1 layer gray level yang menunjukkan warna merah, 1 layer gray level yang menunjukkan warna hijau, dan 1 layer gray level yang menunjukkan warna biru. Histogramnya berbeda untuk setiap warnanya. Sedangkan, pada gambar grayscale, gambar tersusun hanya dari 1 layer gray level saja sehingga apabila dicoba diuraikan menjadi R, G, dan B, semuanya bernilai sama.

Program 2: Memperbaiki citra dengan beberapa operasi

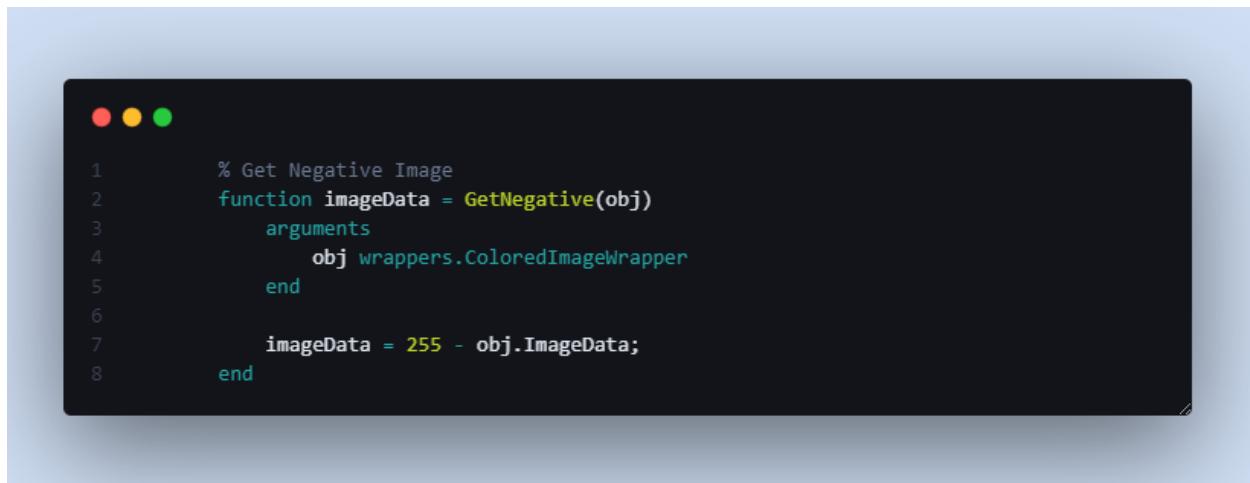
Kode program

Image brightening



```
1      % Get Image Brightening
2      function imageData = GetBrightening(obj, a, b)
3          arguments
4              obj wrappers.ColoredImageWrapper
5              a double;
6              b double;
7          end
8
9          imageData = a * obj.ImageData + b;
10         end
```

Citra negatif



```
1      % Get Negative Image
2      function imageData = GetNegative(obj)
3          arguments
4              obj wrappers.ColoredImageWrapper
5          end
6
7          imageData = 255 - obj.ImageData;
8      end
```

Transformasi log

```
1      % Get Log Transformation
2      function imageData = GetLogTransformation(obj, c)
3          arguments
4              obj wrappers.ColoredImageWrapper
5              c double;
6          end
7
8          doubleImageData = im2double(obj.ImageData);
9          doubleImageDataProcessed = c * log(1 + doubleImageData);
10         imageData = im2uint8(doubleImageDataProcessed);
11     end
```

Transformasi pangkat

```
1      % Get Power Law Transformation
2      function imageData = GetPowerTransformation(obj, c, gamma)
3          arguments
4              obj wrappers.ColoredImageWrapper
5              c double;
6              gamma double;
7          end
8
9          doubleImageData = im2double(obj.ImageData);
10         doubleImageDataProcessed = c * doubleImageData .^ gamma;
11         imageData = im2uint8(doubleImageDataProcessed);
12     end
```

Peregangan kontras

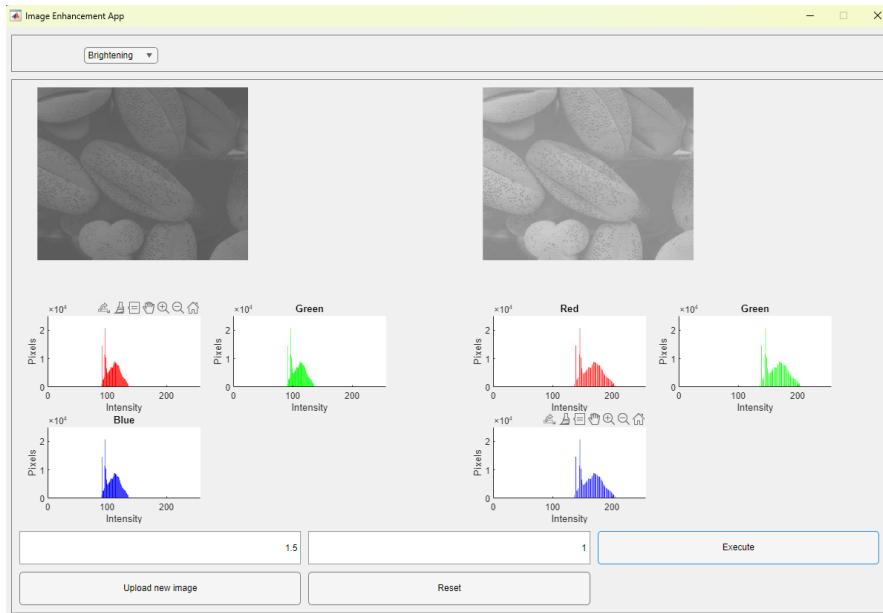
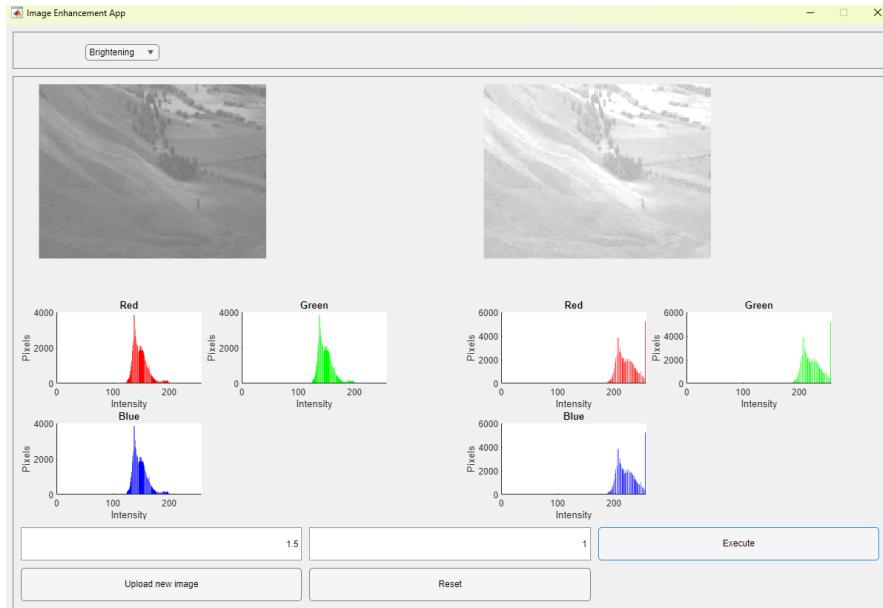
```
● ○ ■  
1 function newData = histstretch(inputData)  
2     arguments  
3         inputData uint8  
4     end %arguments  
5  
6     doubleInputData = im2double(inputData);  
7  
8     % Find the minimum and maximum value  
9     minValue = min(min(doubleInputData));  
10    maxValue = max(max(doubleInputData));  
11  
12    % Compute new image data  
13    doubleNewData = (doubleInputData - minValue) / (maxValue - minValue);  
14  
15    % Convert to uint8  
16    newData = im2uint8(doubleNewData);  
17 end
```

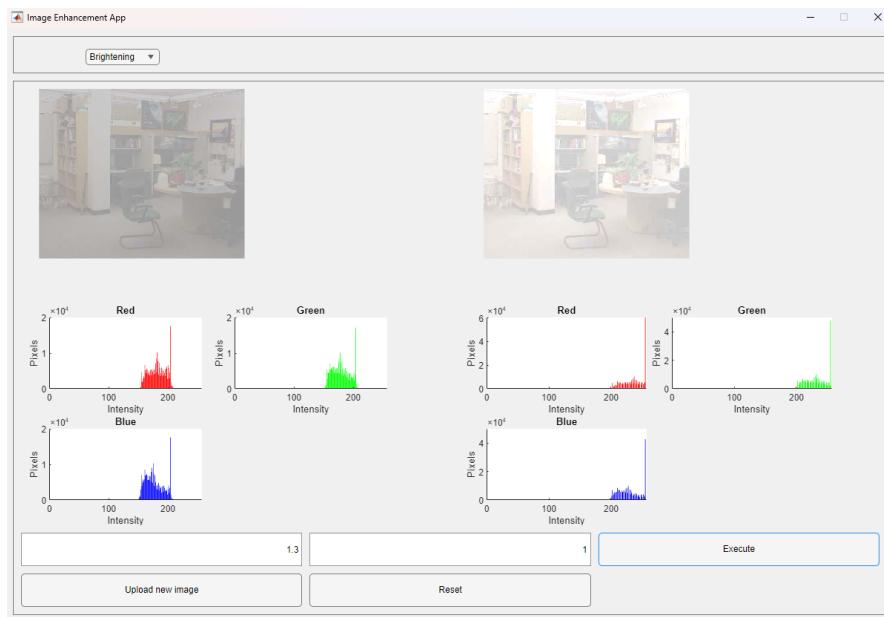
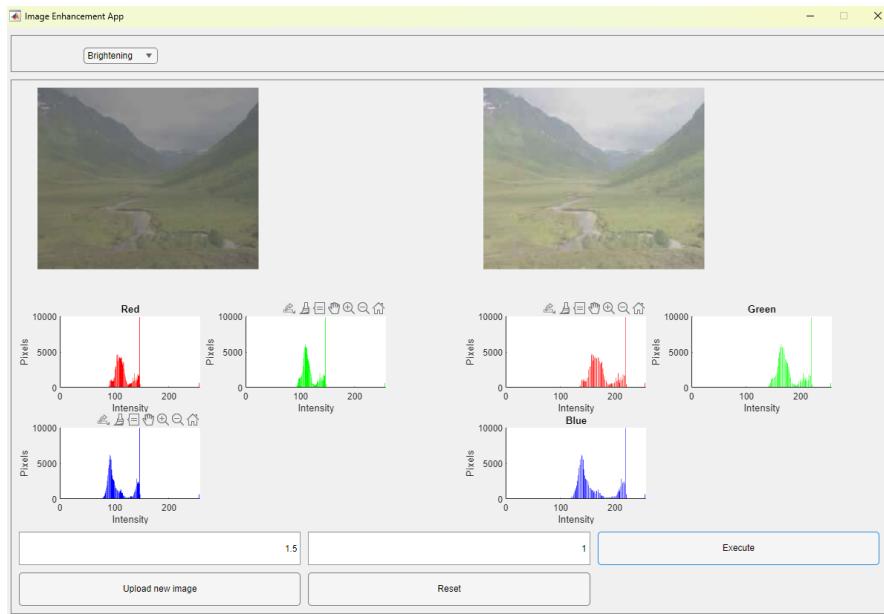
```
● ○ ■  
1 % Get Histogram Stretched  
2 function imageData = GetHistogramStretchedImage(obj)  
3     arguments  
4         obj wrappers.ColoredImageWrapper  
5     end  
6  
7     % Stretch the histogram for each channel  
8     redChan = utils.Histogram.histstretch(obj.ImageData(:, :, 1));  
9     greenChan = utils.Histogram.histstretch(obj.ImageData(:, :, 2));  
10    blueChan = utils.Histogram.histstretch(obj.ImageData(:, :, 3));  
11  
12    imageData = cat(3, redChan, greenChan, blueChan);  
13 end
```

```
● ○ ■  
1 % Get Histogram Stretched  
2 function imageData = GetHistogramStretchedImage(obj)  
3     imageData = utils.Histogram.histstretch(obj.ImageData);  
4 end
```

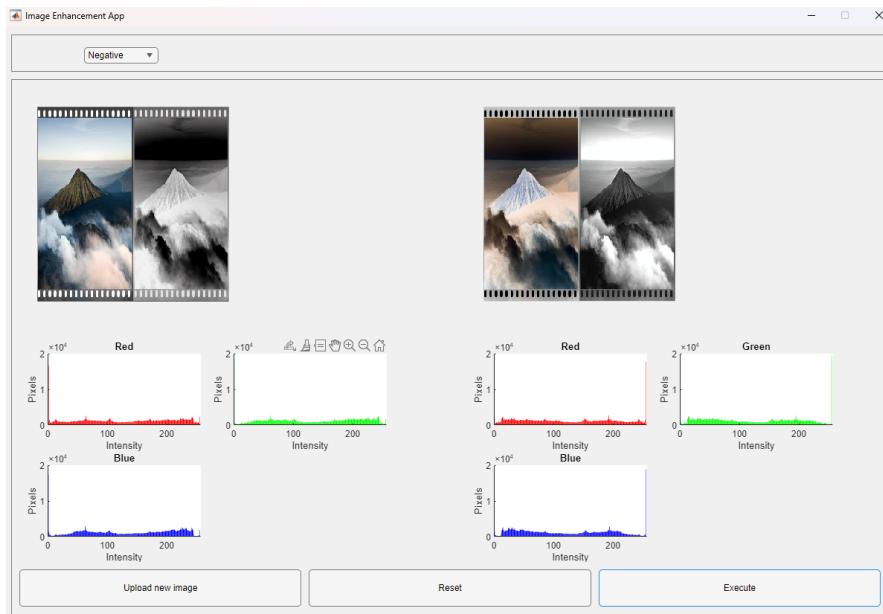
Contoh hasil eksekusi program

Image brightening

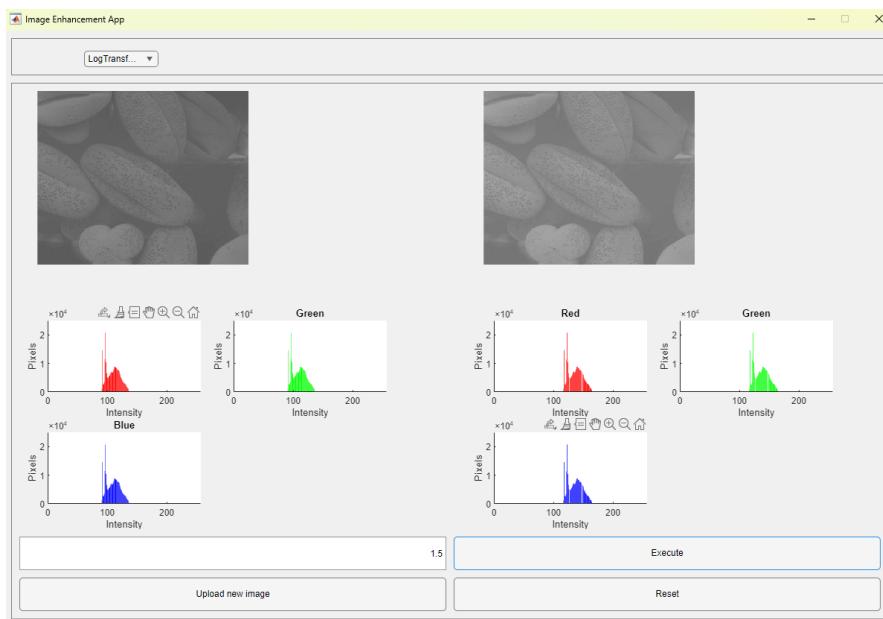
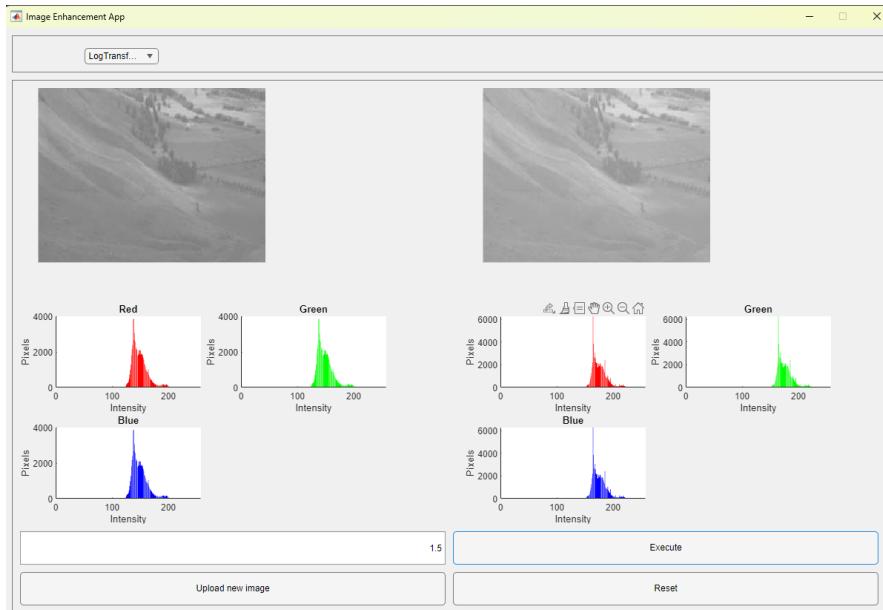


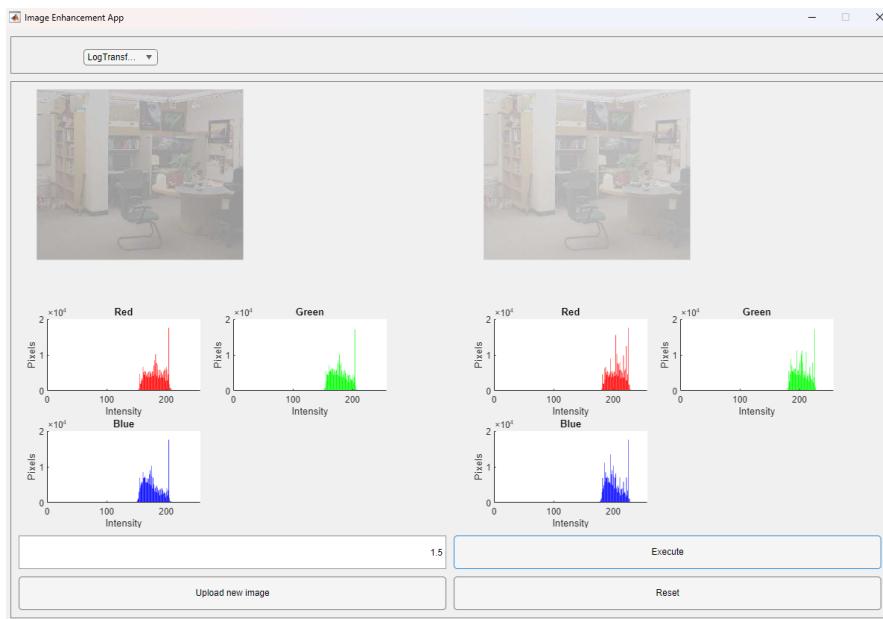
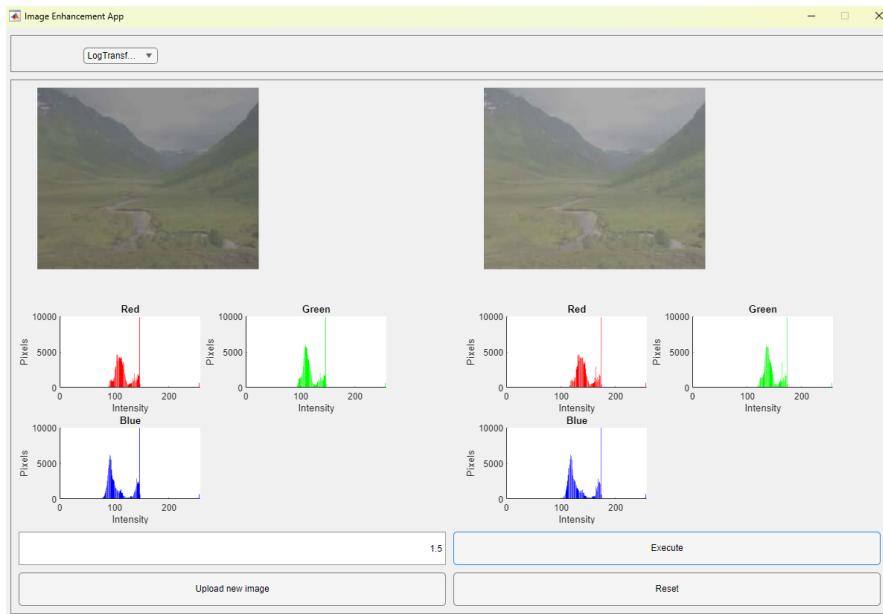


Citra negatif

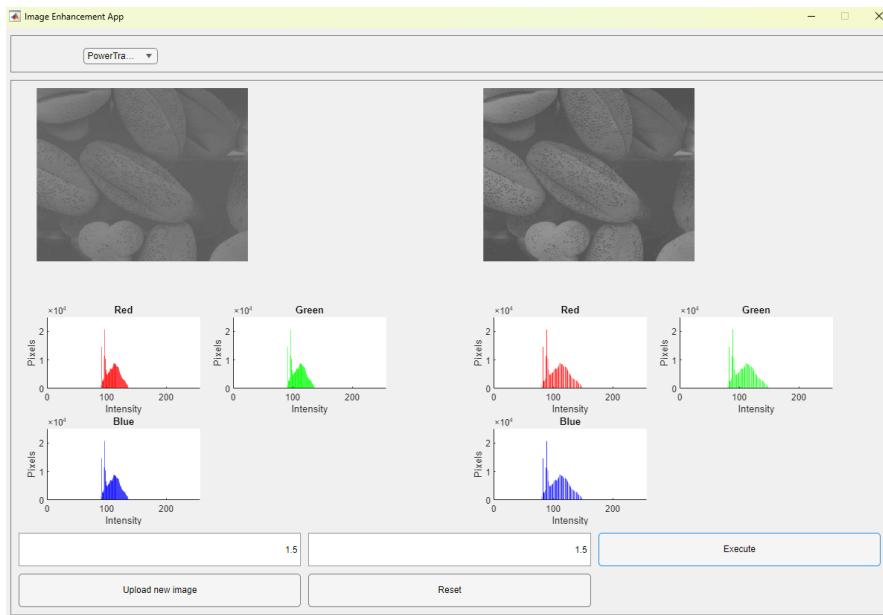
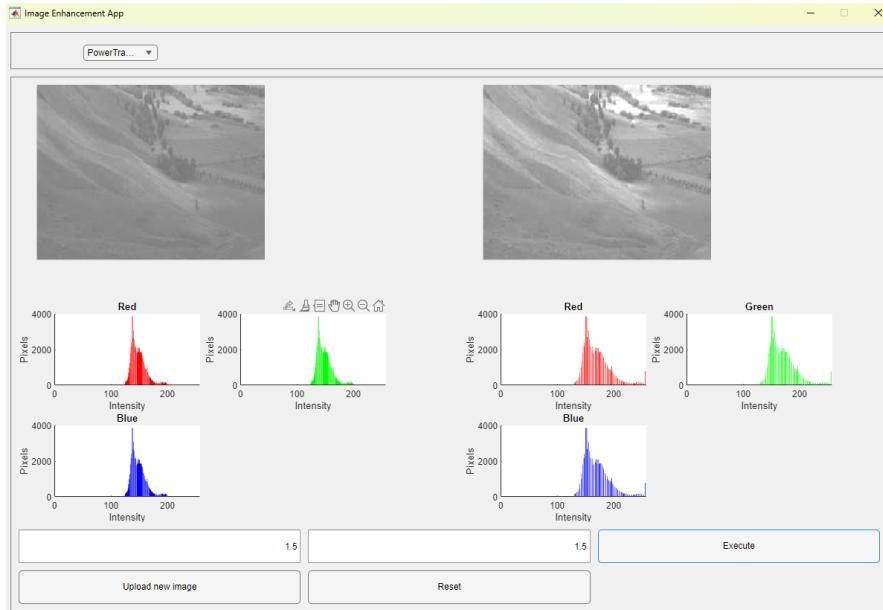


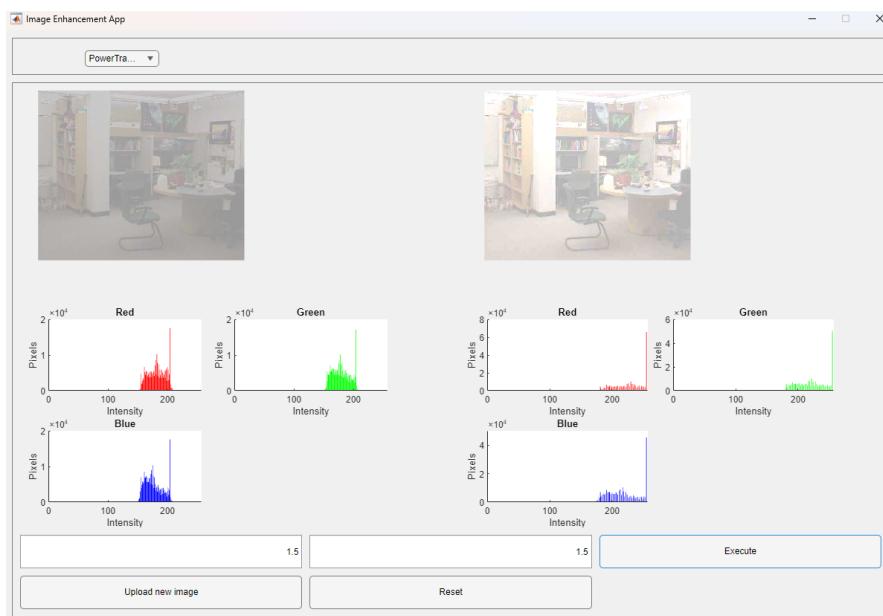
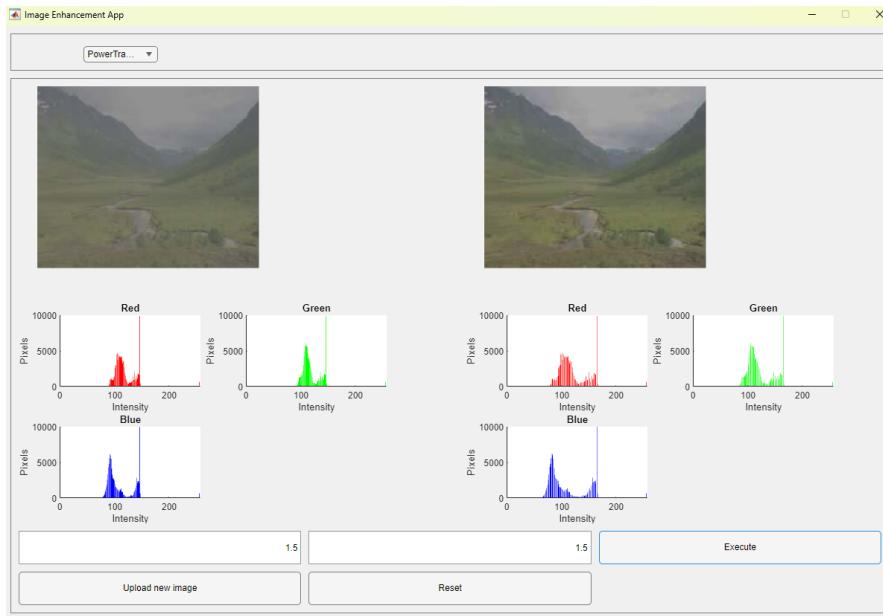
Transformasi log



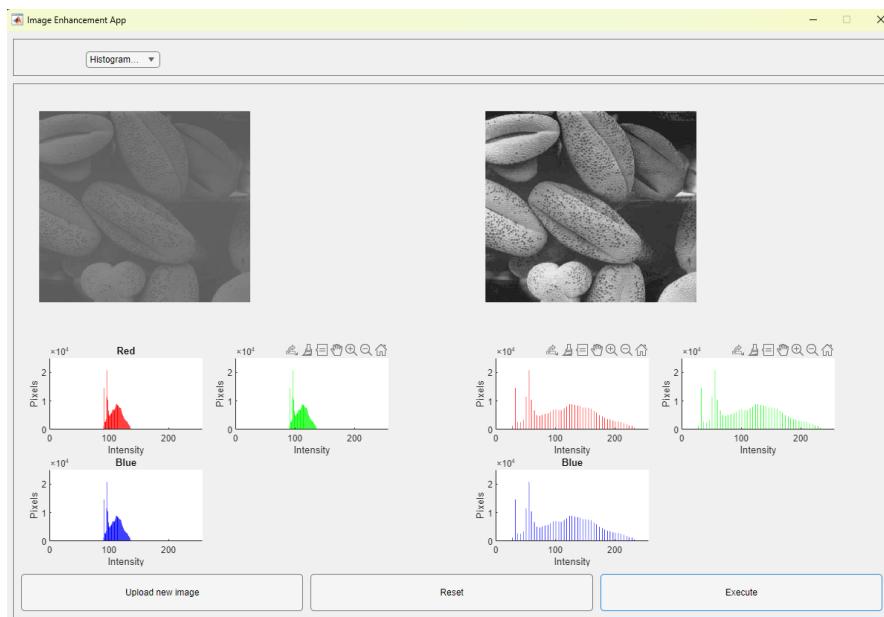
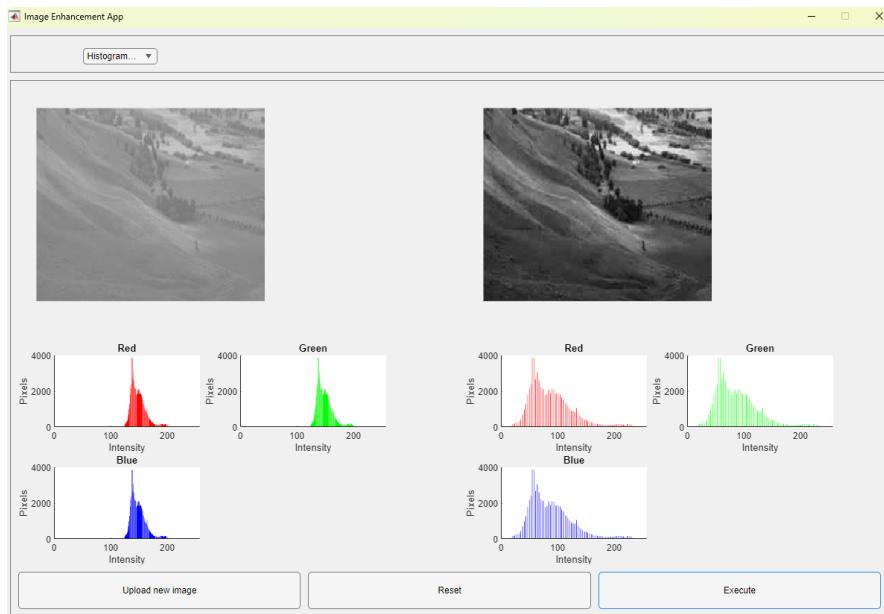


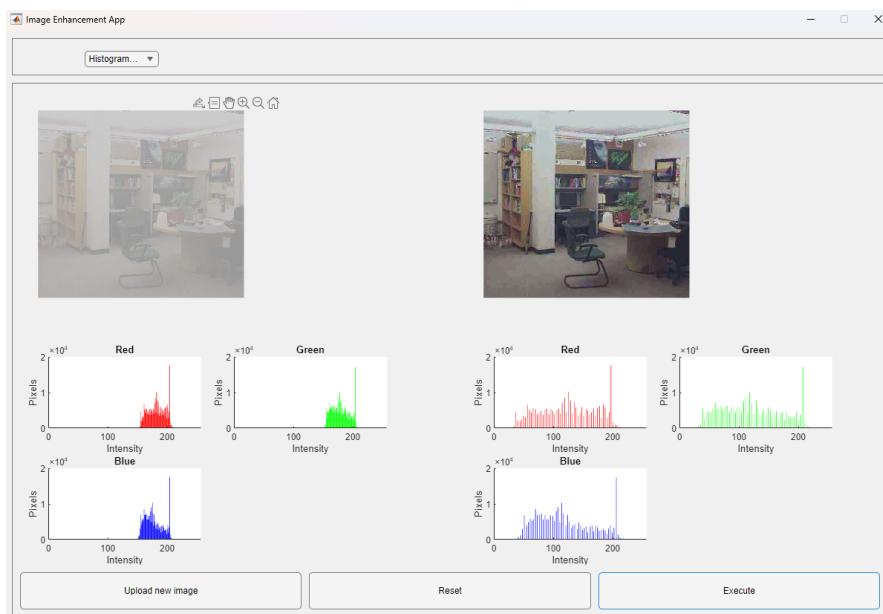
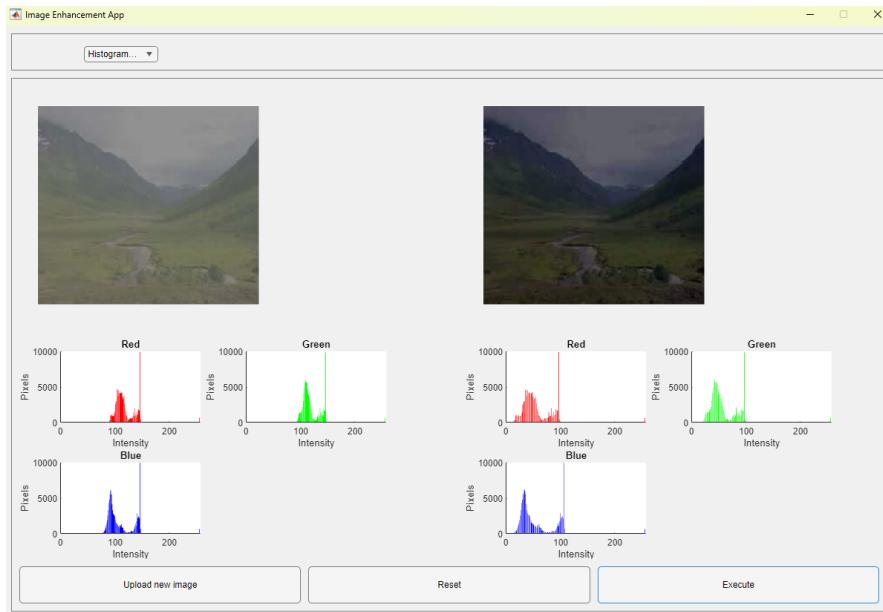
Transformasi pangkat





Peregangan kontras





Analisis cara kerja algoritma

Image brightening

Algoritma bekerja dengan cara mengubah nilai setiap pixel, misal awalnya r , menjadi $ar+b$. Pemrosesan tiap pixel dilakukan dalam waktu yang konstan, sehingga apabila ukuran gambar adalah $M \times N$, kompleksitasnya $O(MN)$.

Citra negatif

Algoritma bekerja dengan cara mengubah nilai setiap pixel, misalnya r , menjadi $255 - r$. Hal unik yang bisa dilihat adalah hasil histogramnya menjadi cerminan terhadap sumbu-y. Pemrosesan tiap pixel dilakukan dalam waktu yang konstan, sehingga apabila ukuran gambar adalah $M \times N$, kompleksitasnya $O(MN)$.

Transformasi log

Algoritma bekerja dengan cara mengubah nilai setiap pixel, misal awalnya r , menjadi $c * \log(1 + r)$. Pemrosesan tiap pixel dilakukan dalam waktu yang konstan, sehingga apabila ukuran gambar adalah $M \times N$, kompleksitasnya $O(MN)$.

Transformasi pangkat

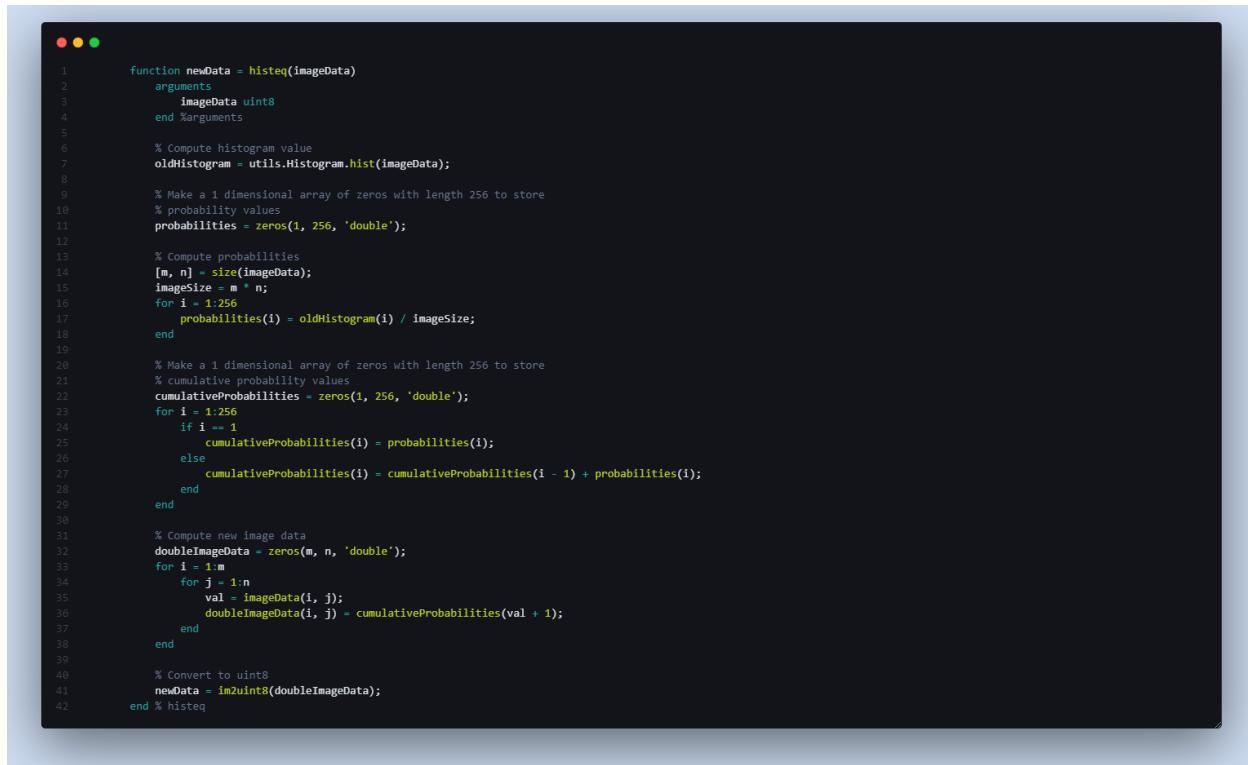
Algoritma bekerja dengan cara mengubah nilai setiap pixel, misal awalnya r , menjadi cr^y . Pemrosesan tiap pixel dilakukan dalam waktu yang konstan, sehingga apabila ukuran gambar adalah $M \times N$, kompleksitasnya $O(MN)$.

Peregangan kontras

Algoritma bekerja dengan cara mencari nilai gray level terkecil dan nilai gray level terbesar yang disimpan secara berurutan pada `minVal` dan `maxVal`. Setelah `minVal` dan `maxVal` diketahui, peregangan dilakukan dengan cara menjadikan `minVal` sebagai nilai 0 dan `maxVal` sebagai nilai gray level tertinggi menggunakan proporsi matematika sederhana (membandingkan nilai sekarang terhadap interval `[minVal, maxVal]`). Dapat dilihat bahwa hasil dari operasi ini menghasilkan histogram yang hasilnya lebih meregang. Untuk mencari kedua nilai `minVal` dan `maxVal` diperlukan perbandingan nilai dari setiap pixel, sehingga apabila ukuran gambar adalah $M \times N$, kompleksitasnya $O(MN)$.

Program 3: Perataan histogram

Kode program

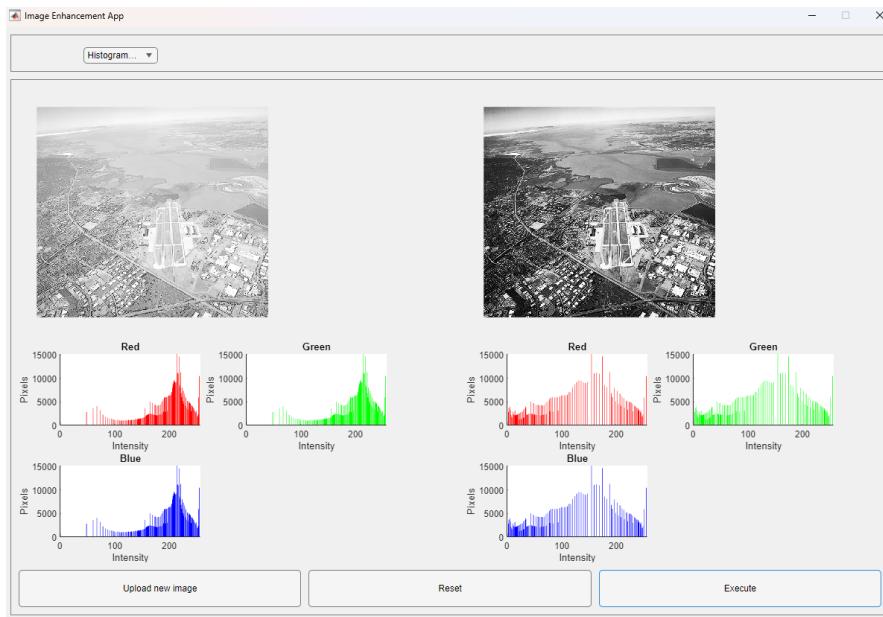
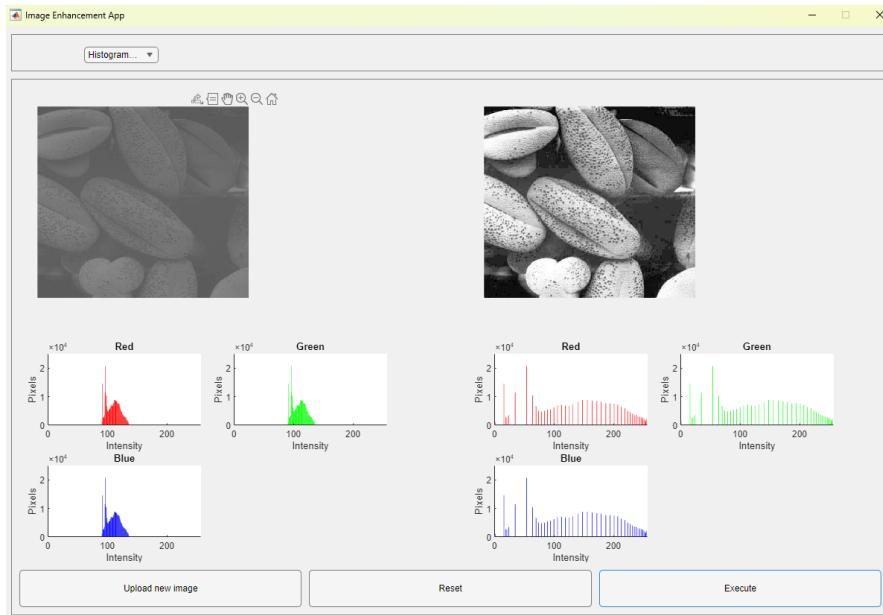


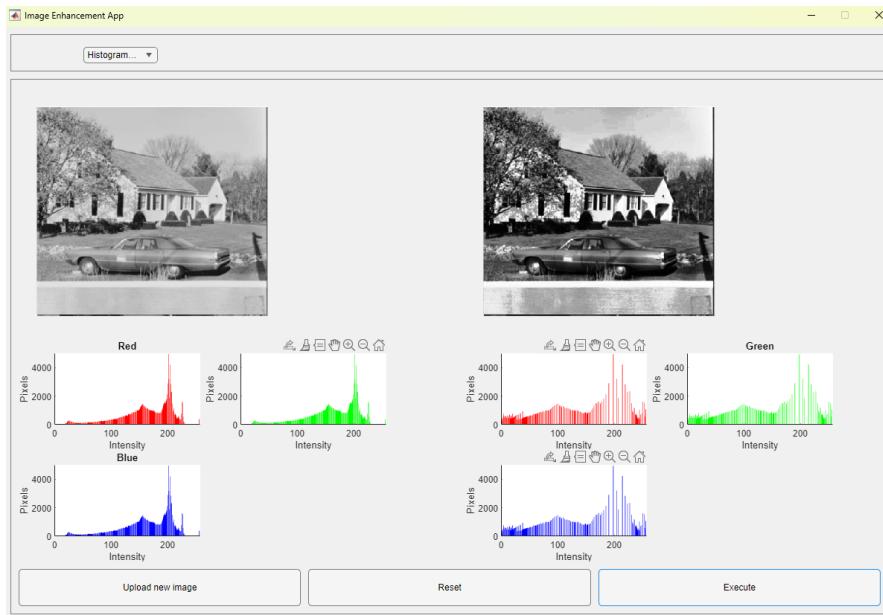
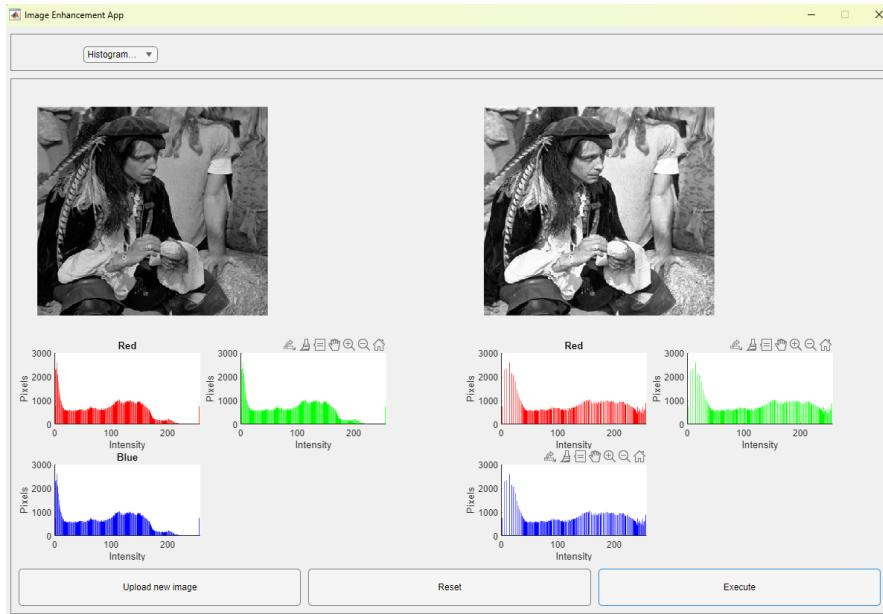
The screenshot shows a MATLAB code editor window with a dark theme. The title bar has three colored dots (red, yellow, green). The main area contains the following MATLAB code:

```
1 function newData = histeq(imageData)
2     % Arguments
3     % imageData uint8
4     % end %arguments
5
6     % Compute histogram value
7     oldHistogram = utils.Histogram.hist(imageData);
8
9     % Make a 1 dimensional array of zeros with length 256 to store
10    % probability values
11    probabilities = zeros(1, 256, 'double');
12
13    % Compute probabilities
14    [m, n] = size(imageData);
15    imageSize = m * n;
16    for i = 1:256
17        probabilities(i) = oldHistogram(i) / imageSize;
18    end
19
20    % Make a 1 dimensional array of zeros with length 256 to store
21    % cumulative probability values
22    cumulativeProbabilities = zeros(1, 256, 'double');
23    for i = 1:256
24        if i == 1
25            cumulativeProbabilities(i) = probabilities(i);
26        else
27            cumulativeProbabilities(i) = cumulativeProbabilities(i - 1) + probabilities(i);
28        end
29    end
30
31    % Compute new image data
32    doubleImageData = zeros(m, n, 'double');
33    for i = 1:m
34        for j = 1:n
35            val = imageData(i, j);
36            doubleImageData(i, j) = cumulativeProbabilities(val + 1);
37        end
38    end
39
40    % Convert to uint8
41    newData = im2uint8(doubleImageData);
42 end % histeq
```

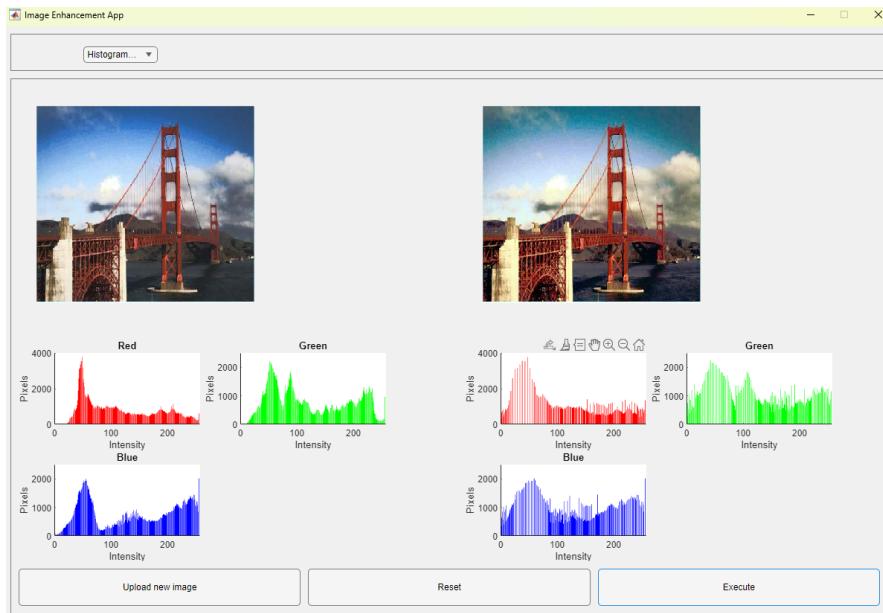
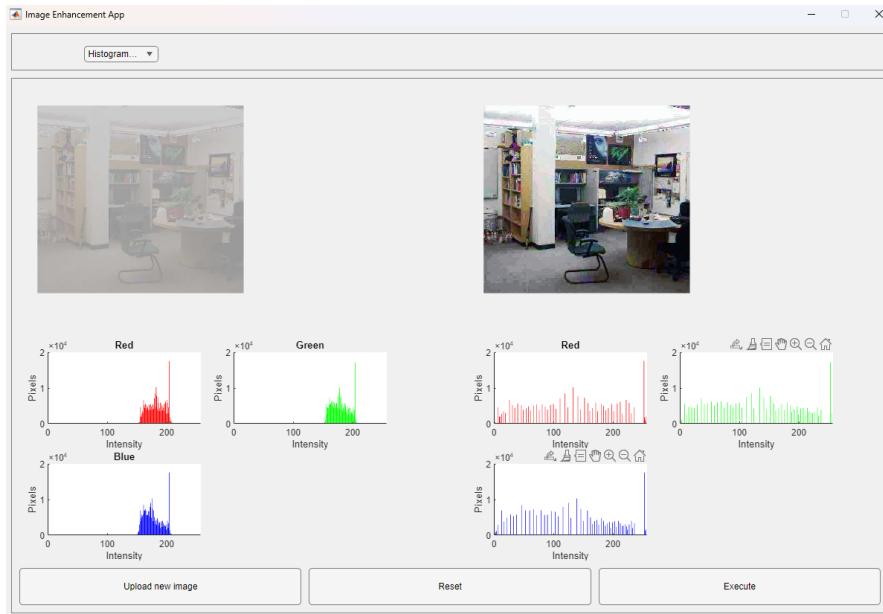
Contoh hasil eksekusi program

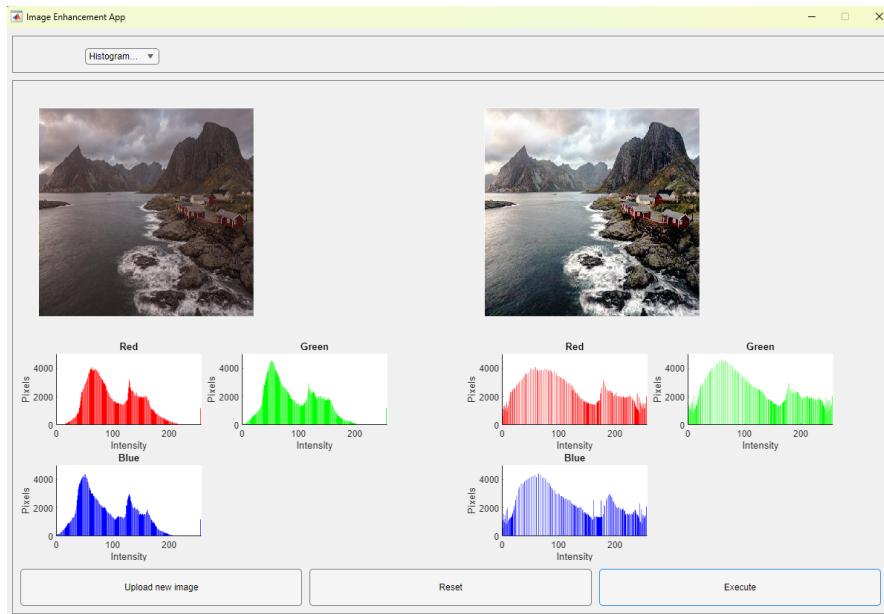
Eksekusi pada gambar grayscale





Eksekusi pada gambar berwarna





Analisis kerja algoritma

Perataan histogram bekerja dengan basis peluang. Yang dilakukan secara umum adalah sebagai berikut:

1. Menghitung histogram dari gambar. Perlu diproses setiap pixelnya, $O(MN)$.
2. Dari data gray level yang ada, misal L level, diperlukan iterasi $O(L)$ untuk mendapatkan peluang masing-masing gray level.
3. Setelah peluang masing-masing gray level, dihitung peluang kumulatif dari setiap gray level. Seperti sebelumnya, ini memiliki kompleksitas $O(L)$.
4. Setiap pixel akan diberikan nilai berupa $L * cumulativeProbability(valuePixelSekarang)$. Perhitungan ini linear dan perlu dilakukan untuk setiap pixel, jadi kompleksitasnya $O(MN)$.

Perhitungan dengan cara mengalikan nilai gray level dengan cumulative probability akan menyebabkan histogram menjadi lebih rata dari sebelumnya seperti yang bisa dilihat pada contoh hasil eksekusi program.

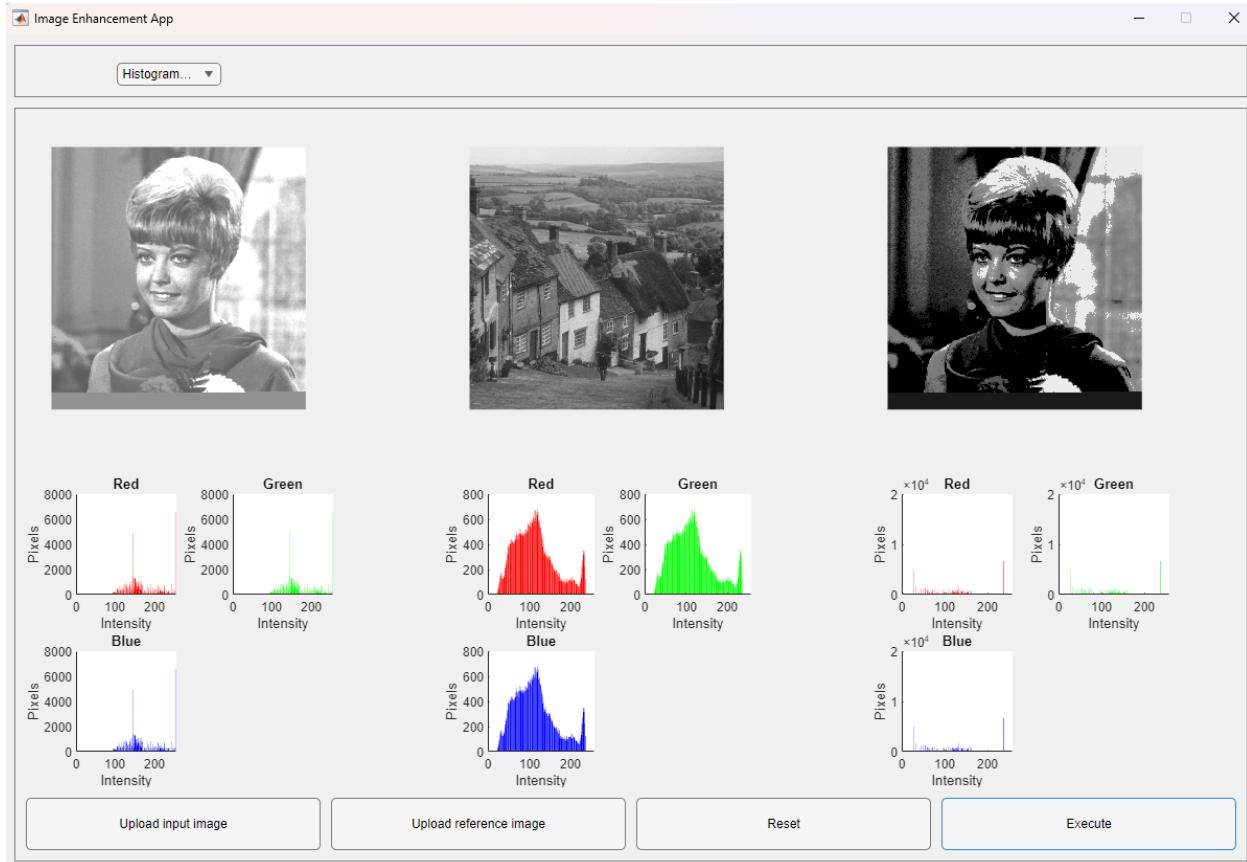
Program 4: Spesifikasi histogram

Kode program

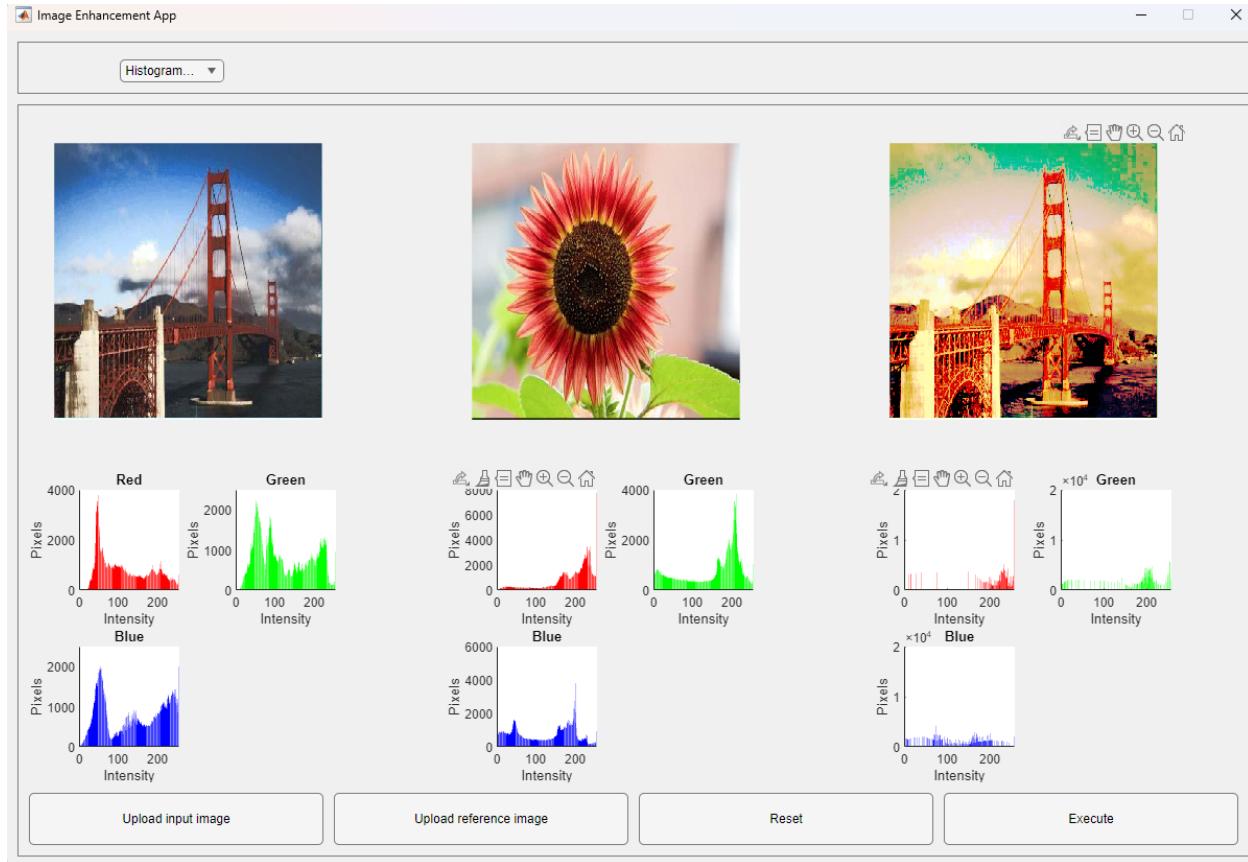
```
1  function newData = histmatch(inputData, referenceData)
2      arguments
3          inputData uint8
4          referenceData uint8
5      end %arguments
6
7      % Process inputData
8
9      % Compute histogram value
10     inputHistogram = utils.Histogram.hist(inputData);
11
12     % Make a 1 dimensional array of zeros with length 256 to store
13     % probability values
14     inputProbabilities = zeros(1, 256, 'double');
15
16     % Compute probabilities
17     [m, n] = size(inputData);
18     imageSize = m * n;
19     for i = 1:256
20         inputProbabilities(i) = inputHistogram(i) / imageSize;
21     end
22
23     % Make a 1 dimensional array of zeros with length 256 to store
24     % cumulative probability values
25     cumulativeInputProbabilities = zeros(1, 256, 'double');
26     for i = 1:256
27         if i == 1
28             cumulativeInputProbabilities(i) = inputProbabilities(i);
29         else
30             cumulativeInputProbabilities(i) = cumulativeInputProbabilities(i - 1) + inputProbabilities(i);
31         end
32     end
33
34     % Find the corresponding map for each possible grayscale
35     % value
36     inputMap = zeros(1, 256, 'uint8');
37     for i = 1:256
38         inputMap(i) = uint8(round(cumulativeInputProbabilities(i) * 255));
39     end
40
41     % Process referenceData
42
43     % Compute histogram value
44     referenceHistogram = utils.Histogram.hist(referenceData);
45
46     % Make a 1 dimensional array of zeros with length 256 to store
47     % probability values
48     referenceProbabilities = zeros(1, 256, 'double');
49
50     % Compute probabilities
51     [m, n] = size(referenceData);
52     imageSize = m * n;
53     for i = 1:256
54         referenceProbabilities(i) = referenceHistogram(i) / imageSize;
55     end
56
57     % Make a 1 dimensional array of zeros with length 256 to store
58     % cumulative probability values
59     cumulativeReferenceProbabilities = zeros(1, 256, 'double');
60     for i = 1:256
61         if i == 1
62             cumulativeReferenceProbabilities(i) = referenceProbabilities(i);
63         else
64             cumulativeReferenceProbabilities(i) = cumulativeReferenceProbabilities(i - 1) + referenceProbabilities(i);
65         end
66     end
67
68     % Create inverse mapping (refered from ppt)
69     inverseMap = zeros(1, 256, 'uint8');
70     for i = 1:256
71         s = inputMap(i);
72         currMin = abs(s - cumulativeReferenceProbabilities(1) * 255);
73         currMinIdx = 1;
74
75         for j = 1:256
76             if abs(s - cumulativeReferenceProbabilities(j) * 255) < currMin
77                 currMin = abs(s - cumulativeReferenceProbabilities(j) * 255);
78                 currMinIdx = j;
79             end
80         end
81
82         inverseMap(i) = currMinIdx;
83     end
84
85     % Generate image data
86     [m, n] = size(inputData);
87     newData = zeros(m, n, 'uint8');
88     for i = 1:m
89         for j = 1:n
90             newData(i, j) = inverseMap(inputMap(inputData(i, j) + 1) + 1);
91         end
92     end
93 end % histmatch
```

Contoh hasil eksekusi program

Eksekusi pada gambar grayscale



Eksekusi pada gambar berwarna



Analisis kerja algoritma

Untuk melakukan spesifikasi histogram, diperlukan perataan histogram terlebih dulu. Citra input dan citra referensi dilakukan perataan histogram, lalu spesifikasi histogram dilakukan dengan cara melakukan *inverse mapping*. Caranya cukup mudah:

1. Untuk suatu pixel bernilai s , cari nilai terdekatnya pada hasil perataan citra reference.
2. Misal perataan citra reference dilakukan dengan pemetaan $f(x) = y$ dengan x adalah citra lama dan y adalah citra setelah perataan histogram.
3. Mencari nilai terdekat artinya mencari nilai y terdekat dengan s .
4. Yang dicari adalah nilai x yang menyebabkan nilai y tersebut.

Setelah didapat pemetaan dari s ke x , cukup ubah setiap pixel yang ada di hasil perataan histogram citra input menjadi nilai yang ada di x . Proses *inverse mapping* yang dilakukan terhadap citra referensi inilah yang menyebabkan histogram citra input bisa dicocokkan dengan histogram citra referensi.

Alamat GitHub program

GitHub program dapat diakses pada pranala berikut:

[rayhankinan/Tugas-1-IF4073-Interpretasi-dan-Pengolahan-Citra: Matlab Program for Visualising Image Histogram and Image Enhancement \(github.com\)](https://github.com/rayhankinan/Tugas-1-IF4073-Interpretasi-dan-Pengolahan-Citra: Matlab Program for Visualising Image Histogram and Image Enhancement (github.com))