

# Laporan Tugas 1

## IF4073 Interpretasi dan Pengolahan Citra

### Image Enhancement

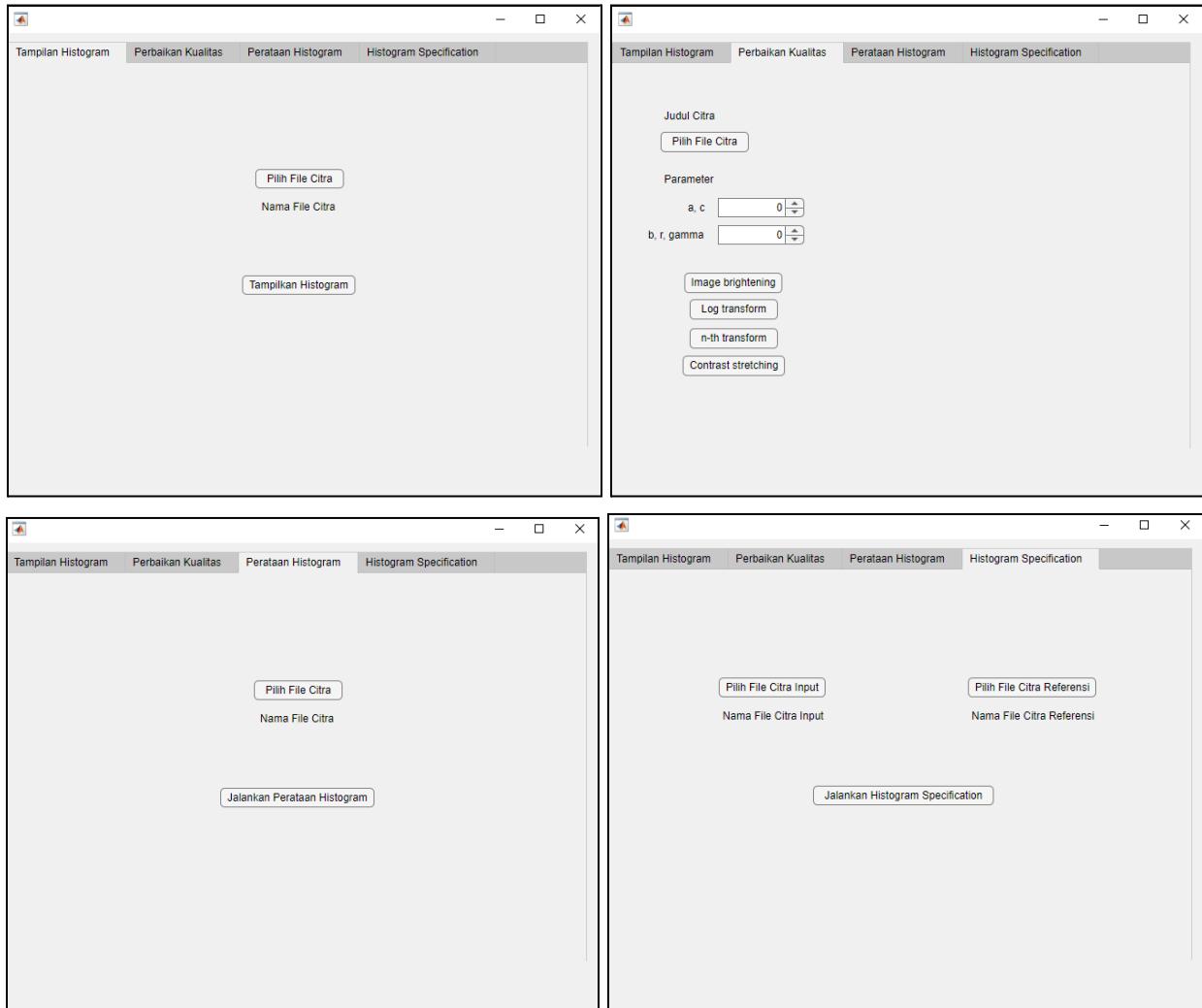


**Disusun oleh:**

Jauhar Wibisono	/ 13519160
Tanur Rizaldi Rahardjo	/ 13519214

**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2022**

# 1. Screenshot GUI Program



## 2. Rincian Setiap Program

### 2.1. Menghitung dan Menampilkan Histogram

#### Kode Program

Kode untuk menghitung dan menampilkan histogram citra, secara urut, adalah seperti berikut.

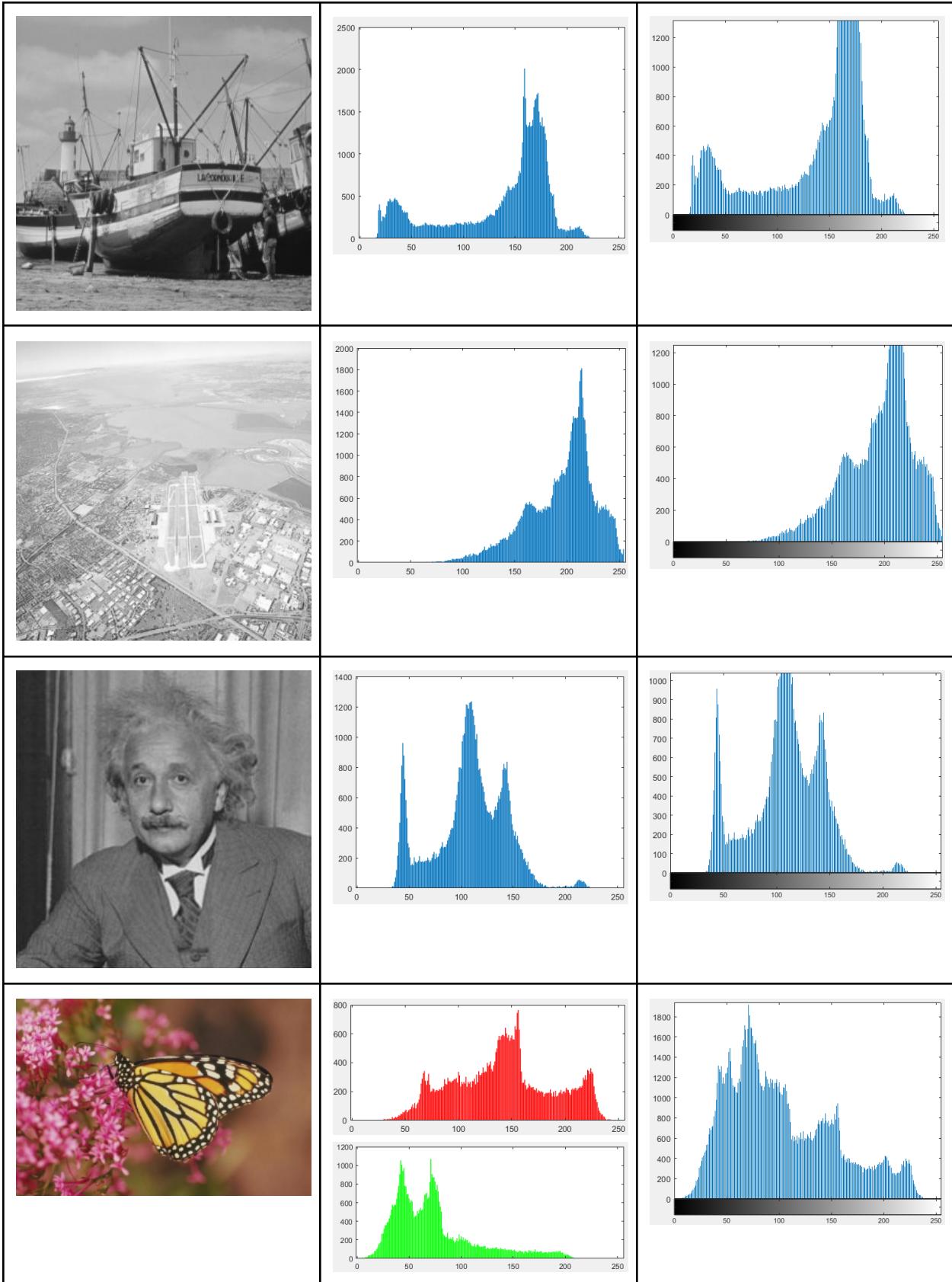
```
% Returns frequency array of intensity values in range [0, 256]
function res = GetFrequency(~, image)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numColor, 256]);
    for c = 1:numColor
        for i = 1:numRow
            for j = 1:numColumn
                pixelValue = image(i, j, c) + 1; % +1 because arrays are one-indexed
                res(c, pixelValue) = res(c, pixelValue) + 1;
            end
        end
    end
end
```

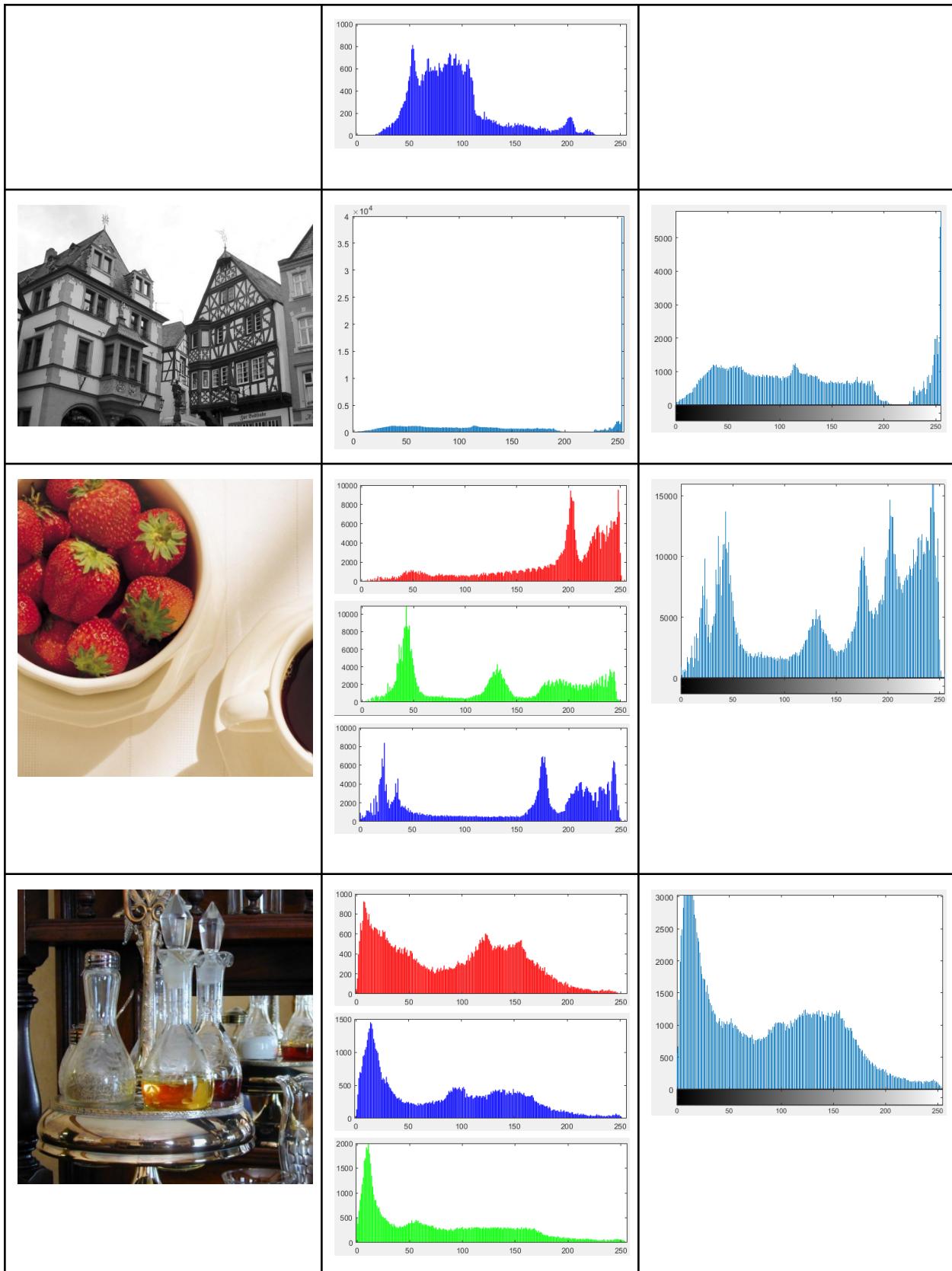
```
% Shows color histogram of an image
function ShowImageHistogram(comp, image)
    freq = comp.GetFrequency(image);
    [numColor, ~] = size(freq);

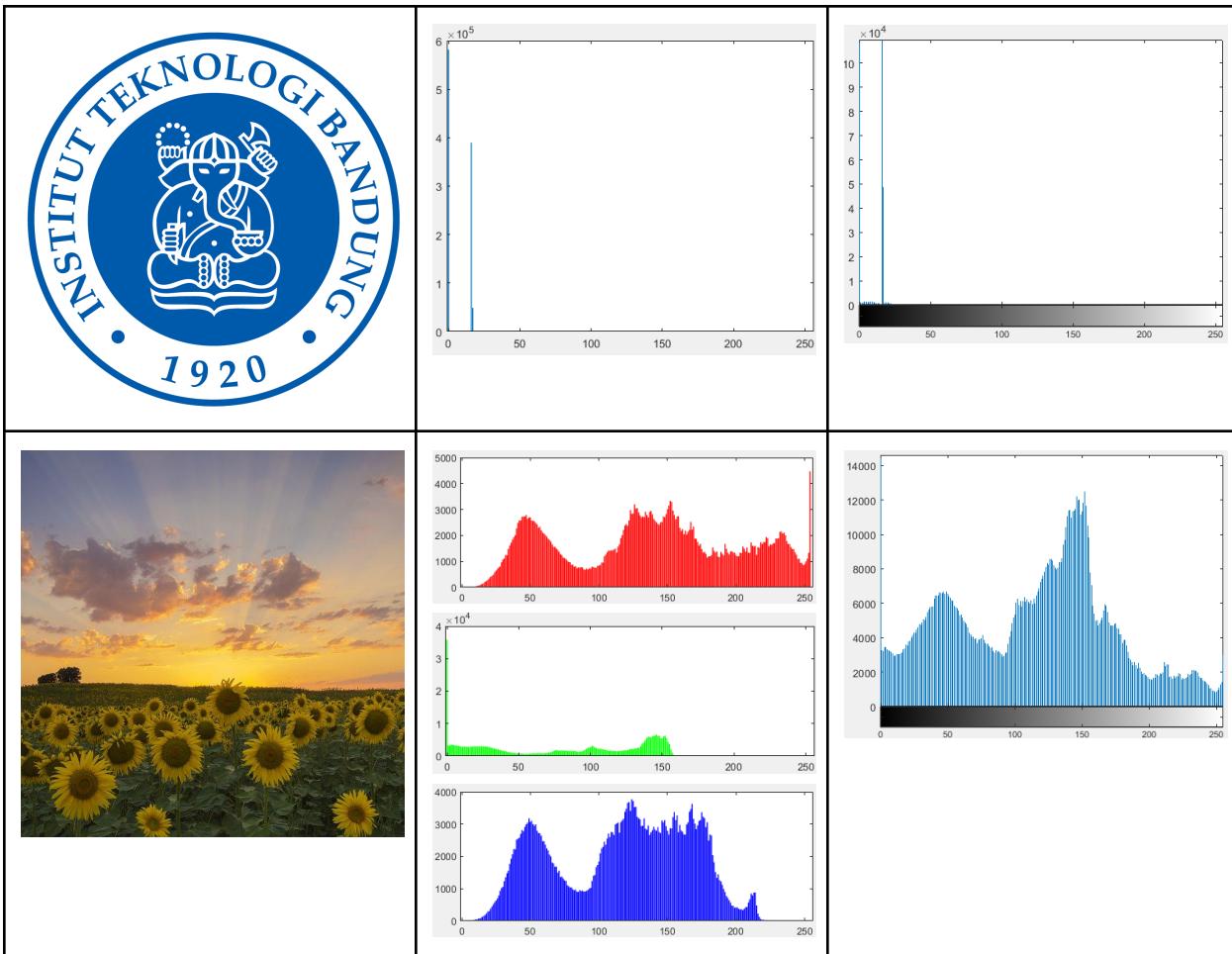
    if numColor == 1 % grayscale
        figure("Name", "Histogram")
        bar(0:1:255, freq);
    else % RGB
        figure("Name", "Histogram Red")
        bar(0:1:255, freq(1, :), "red");
        figure("Name", "Histogram Blue")
        bar(0:1:255, freq(2, :), "blue");
        figure("Name", "Histogram Green")
        bar(0:1:255, freq(3, :), "green");
    end
end
```

#### Contoh Hasil Eksekusi

Citra	Hasil Program Buatan Sendiri	Hasil <i>imhist</i>
-------	------------------------------	---------------------







## Analisis

Pada kode untuk menghitung histogram, terdapat perulangan yang digunakan untuk melihat pixel-pixel citra pada setiap baris, kolom, dan kanal warna. Untuk setiap pixel tersebut, kemunculannya dicatat dalam sebuah larik. Karena larik di Matlab memiliki indeks yang dimulai dari satu, intensitas pixel dan indeks larik memiliki rentang nilai yang berbeda. Untuk mengatasi masalah tersebut, dilakukan translasi terhadap nilai intensitas pixel, yaitu sebesar 1 ke arah positif.

Kode untuk menampilkan histogram memerhatikan dua kasus, yaitu citra *grayscale* dan citra berwarna (RGB). Dua kasus tersebut dapat dibedakan berdasarkan banyak kanal warna pada histogram. Untuk citra *grayscale* program menampilkan satu buah histogram, sedangkan untuk citra berwarna program menampilkan tiga buah histogram untuk masing-masing warna merah, biru, dan hijau.

Pada contoh hasil eksekusi, terlihat bahwa histogram yang ditampilkan program yang dibuat sudah menyerupai hasil *imhist*. Terlihat juga beberapa perbedaan antara hasil program yang dibuat dengan hasil *imhist*. Pertama, program yang dibuat menampilkan histogram secara utuh, sedangkan *imhist* memotong bagian atas histogram untuk menjaga agar frekuensi rendah tetap terlihat. Kedua, program yang dibuat

membagi histogram untuk citra berwarna menjadi tiga, sedangkan *imhist* menjumlahkan intensitas ketiga warna untuk mendapatkan satu histogram.

## 2.2. Perbaikan Kualitas Citra

### Kode Program

Berikut adalah kode program untuk melakukan perbaikan kualitas citra, terurut sesuai dengan spesifikasi : *Image Brightening*, *Log transform*, *Power-law transform*, dan *Contrast stretching*

```
% Image Brightening
function res = ImageBrightening(~, image, a, b)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    for i = 1:numRow
        for j = 1:numColumn
            for c = 1:numColor
                % Clamp [0, 256]
                res(i, j, c) = image(i, j, c) * a + b;
                res(i, j, c) = max(0, min(255, res(i, j, c)));
            end
        end
    end

    res = uint8(res);
end
```

```
% Log transform
function res = ImageLogTransform(~, image, cr)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    for i = 1:numRow
        for j = 1:numColumn
            for c = 1:numColor
                % Clamp [0, 256]
                res(i, j, c) = round(cr * log10(1 + double(image(i, j, c))));
                res(i, j, c) = max(0, min(255, res(i, j, c)));
            end
        end
    end

    res = uint8(res);
end
```

```
% Power law transform
function res = ImageNthTransform(~, image, cr, gamma)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    for i = 1:numRow
        for j = 1:numColumn
            for c = 1:numColor
                % Clamp [0, 256], normed with 8-bit depth in mind
                res(i, j, c) = cr * power(double(image(i, j, c))/255, gamma)*255;
                res(i, j, c) = max(0, min(255, res(i, j, c)));
            end
        end
    end

    res = uint8(res);
end
```

```
% Contrast stretching
function res = ImageContrastStretching(~, image)
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    rmin = [255, 255, 255];
    rmax = [0, 0, 0];

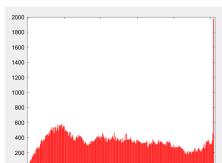
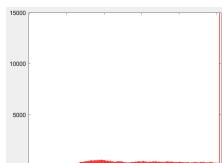
    for i = 1:numRow
        for j = 1:numColumn
            for c = 1:numColor
                if (rmax(c) < image(i, j, c))
                    rmax(c) = image(i, j, c);
                end

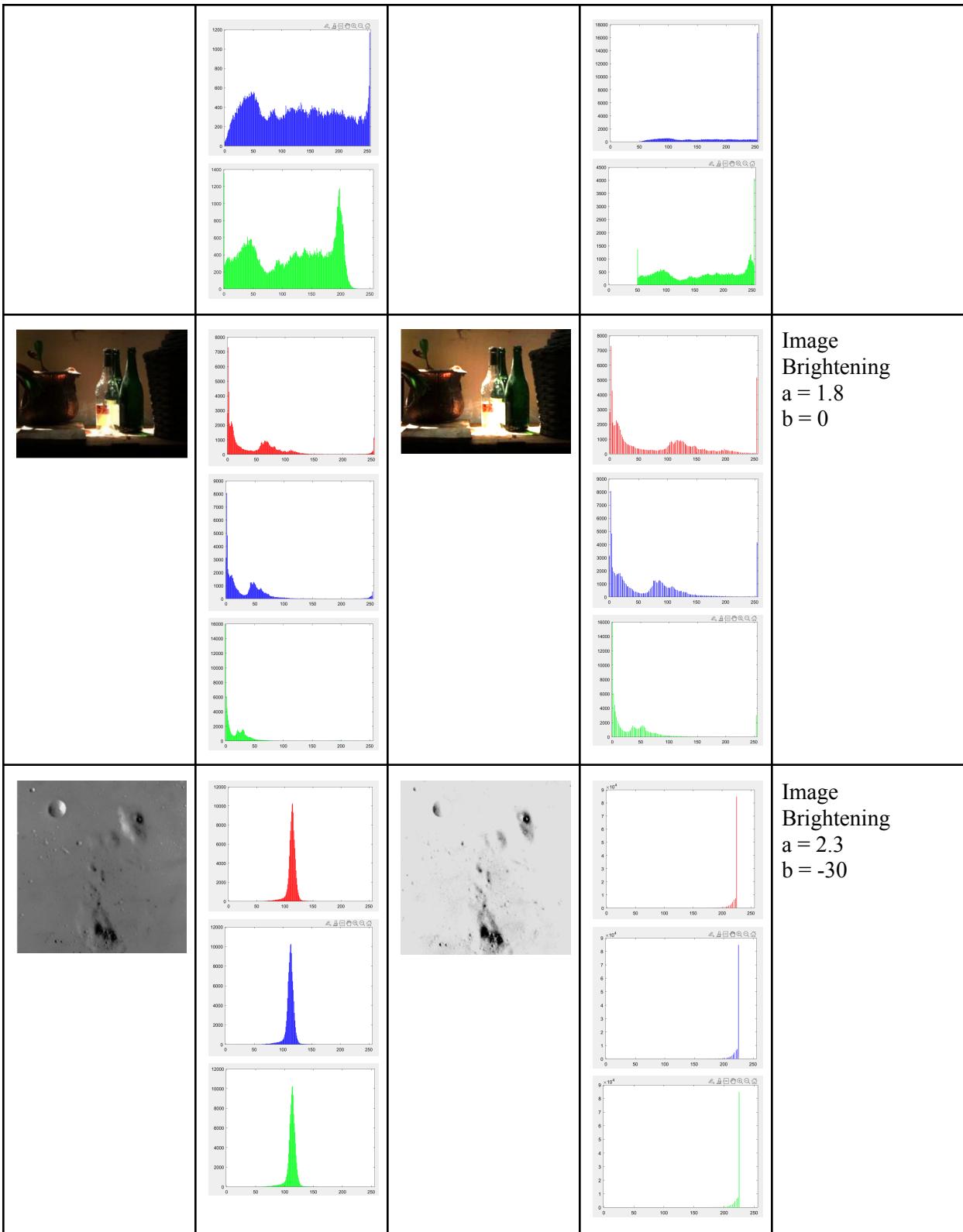
                if (rmin(c) > image(i, j, c))
                    rmin(c) = image(i, j, c);
                end
            end
        end

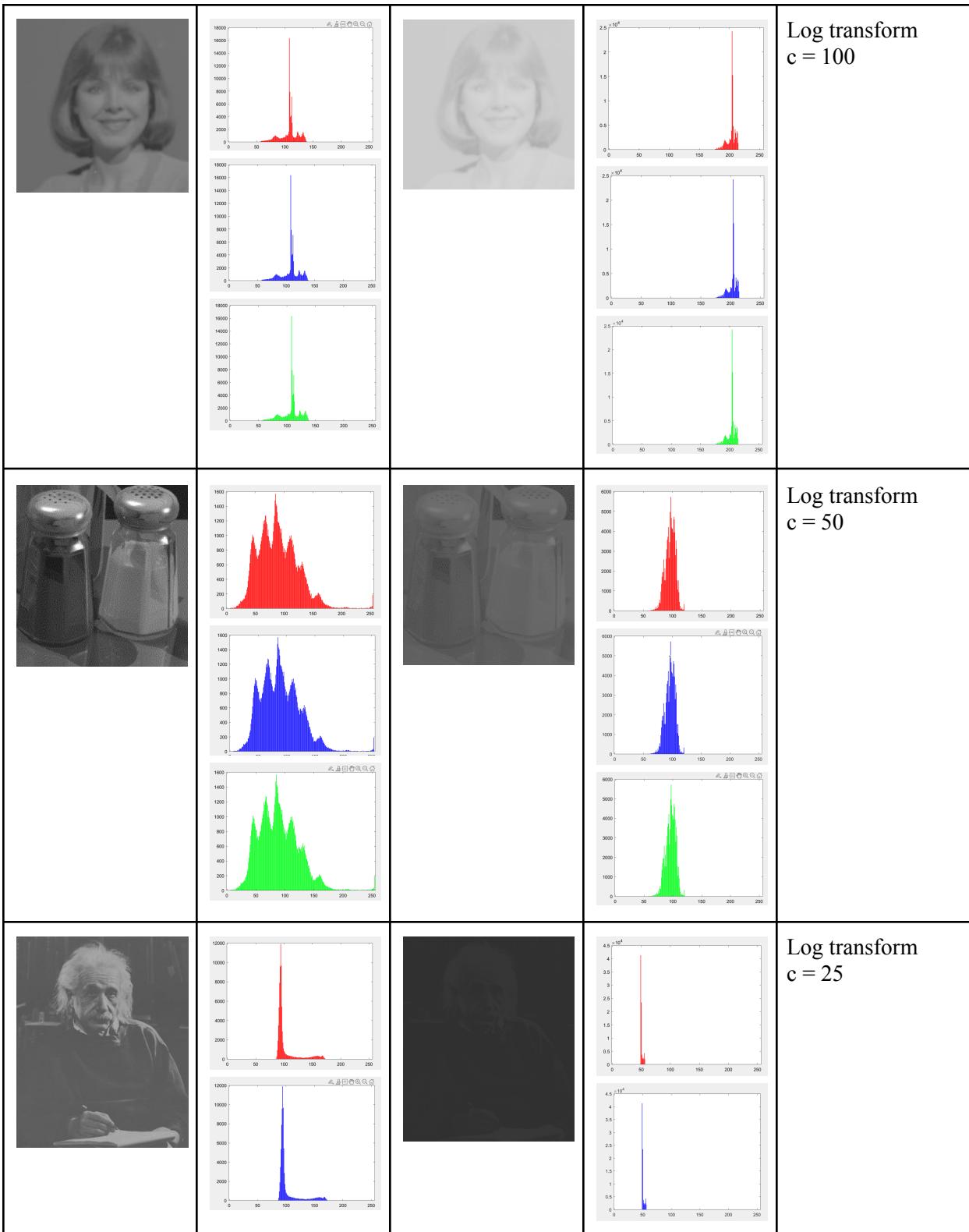
        for i = 1:numRow
            for j = 1:numColumn
                for c = 1:numColor
                    % Clamp [0, 256]
                    res(i, j, c) = (image(i, j, c) - rmin(c))*(255/(rmax(c)-rmin(c)));
                    res(i, j, c) = max(0, min(255, res(i, j, c)));
                end
            end
        end
    end

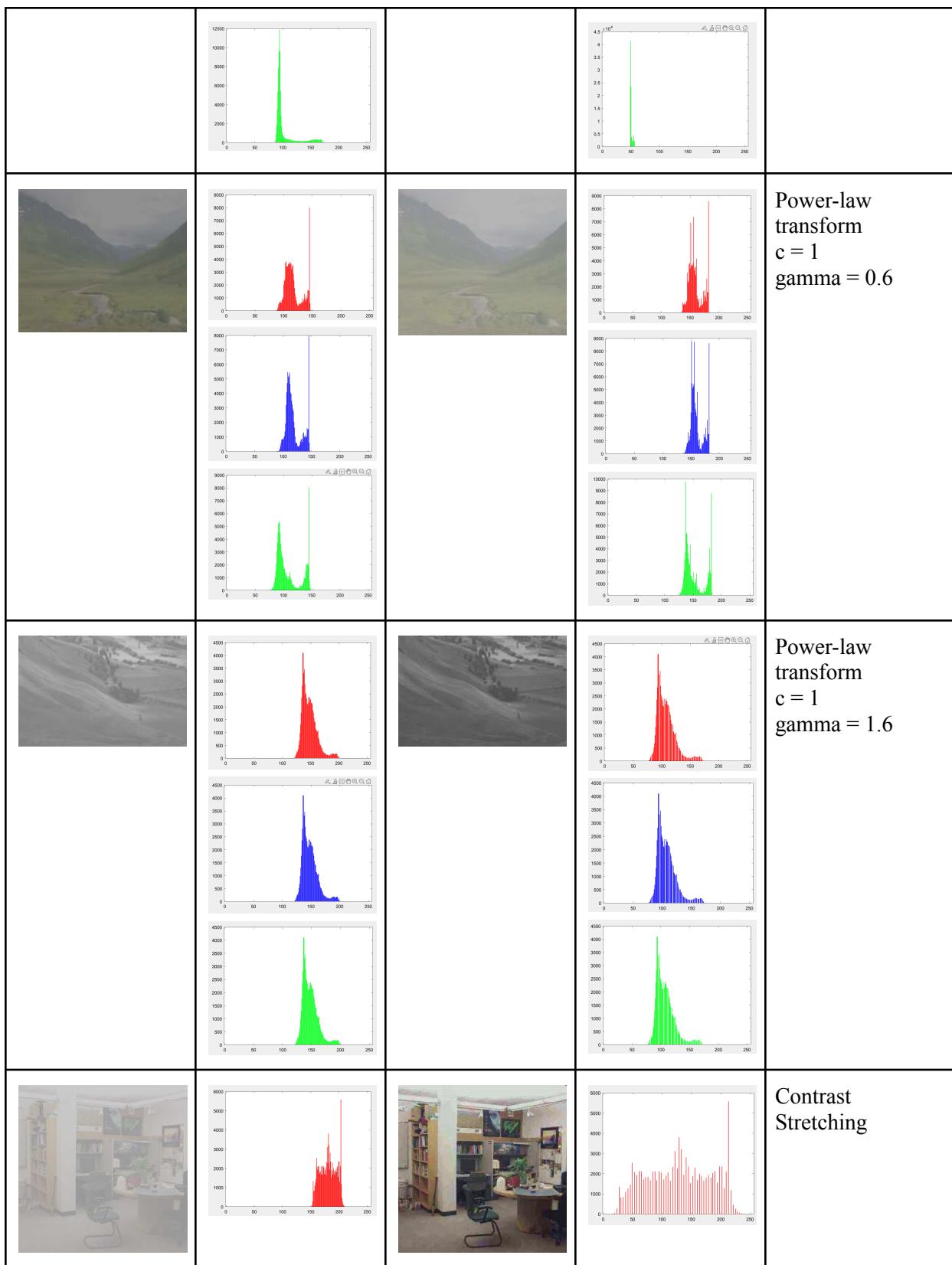
    res = uint8(res);
end
```

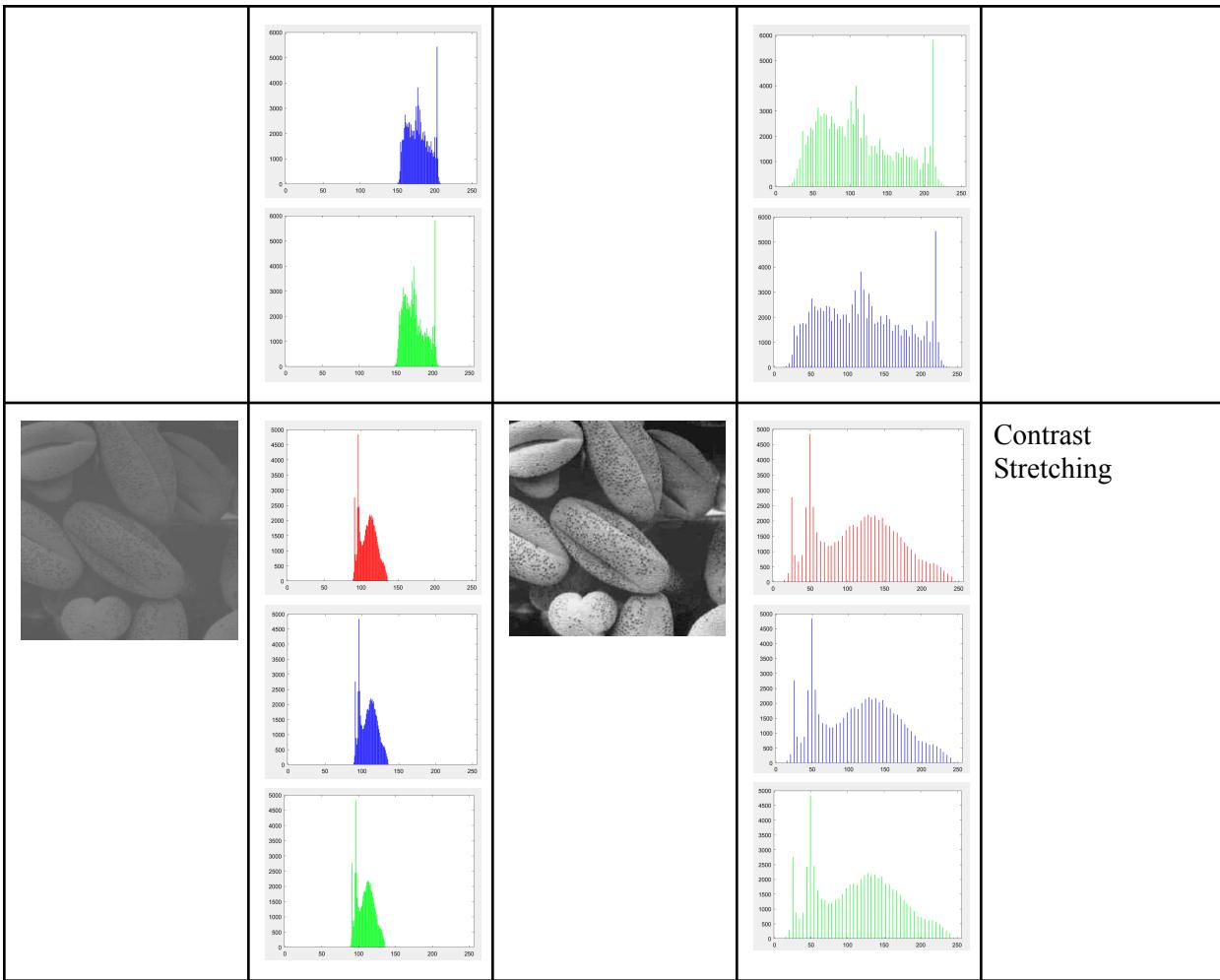
## Contoh Hasil Eksekusi

Citra Input	Histogram Input	Citra Output	Histogram Output	Parameter
				Image Brightening a = 1 b = 50









## Analisis

Untuk kode algoritma untuk *image enhancement*, sebagian besar sudah cukup *straightforward* dan *self explanatory*, sesuai dengan penjelasan dari kelas. Namun setelah mengetes beberapa kali pada input yang berbeda, pada kasus ekstrim, beberapa artifak mungkin muncul pada gambar. Kemungkinan besar hal ini disebabkan oleh beberapa artifak kompresi yang di-*enhance* secara tidak sengaja oleh algoritma. Kemungkinan lain adalah *numeric precision* yang terbatas dapat membuat kalkulasi aritmatika tidak sesuai dengan operasi pada bilangan riil. Contoh kasus ekstrim tersebut adalah *Power-law transform* dengan parameter  $c=1$  dan  $\gamma=0.3$ .

Selain yang disebutkan, hasil program sudah cukup sesuai dengan ekspektasi dari penjelasan yang diberikan dikelas.

## 2.3. Perataan Histogram

### Kode Program

Program perataan histogram memiliki dua komponen, yaitu fungsi yang menghitung pemetaan dan fungsi yang mengaplikasikan pemetaan. Dua komponen tersebut secara urut yaitu seperti berikut.

```
% Returns histogram equalization result, in float
function res = GetHistEqMapping(comp, image)
    [numRow, numColumn, numColor] = size(image);
    freq = comp.GetFrequency(image);

    res = zeros([numColor, 256]);
    for c = 1:numColor
        for i = 1:256
            res(c, i) = freq(c, i) ./ (numRow*numColumn) * 256;
            if i > 1
                res(c, i) = res(c, i) + res(c, i-1);
            end
        end
    end
end
```

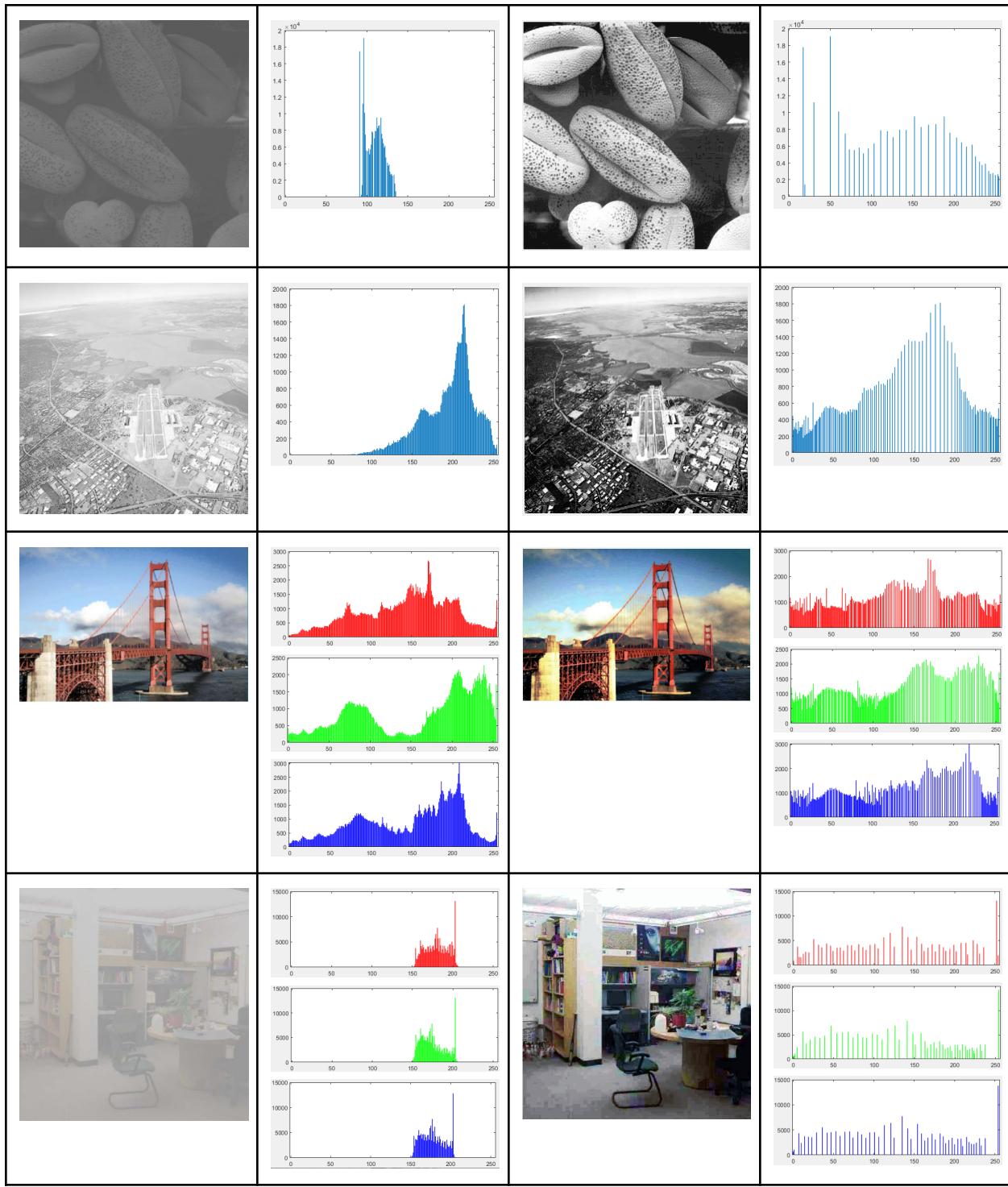
```
% Applies color mapping map to image
function res = ApplyMapping(~, image, map)
    map = round(map); % Round map to integer before applying

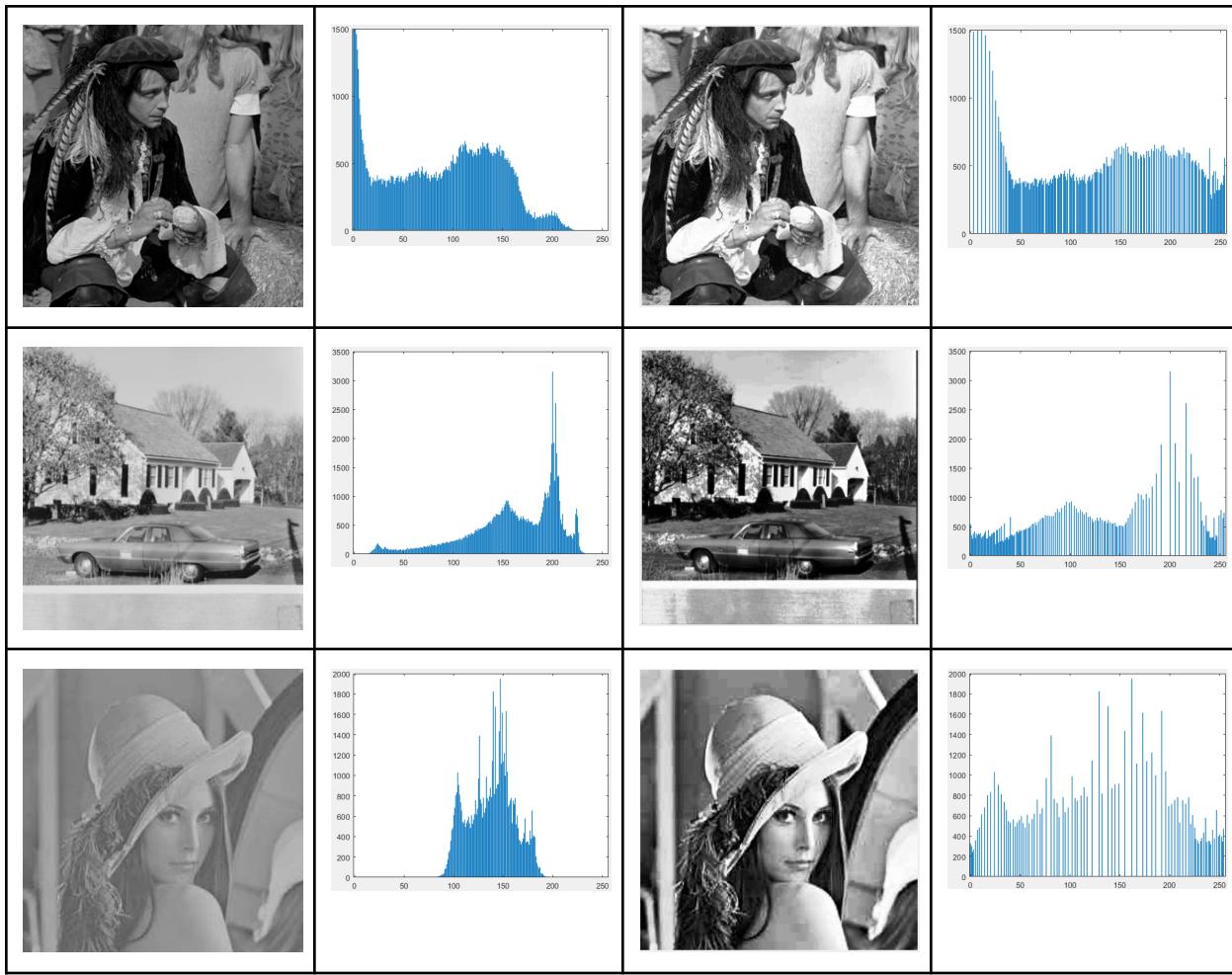
    [numRow, numColumn, numColor] = size(image);
    res = zeros([numRow, numColumn, numColor]);
    for i = 1:numRow
        for j = 1:numColumn
            for c = 1:numColor
                res(i, j, c) = map(c, image(i, j, c)+1) - 1; % -1 because map values are in (0, 256]
            end
        end
    end

    res = uint8(res);
end
```

### Contoh Hasil Eksekusi

Citra Input	Histogram Citra Input	Citra Hasil	Histogram Citra Hasil
-------------	-----------------------	-------------	-----------------------





## Analisis

Pada fungsi yang menghitung pemetaan, terdapat perulangan yang menghitung ekspresi  $\sum_{i=0}^j n_i/n \times 256$  ( $n_i$  menyatakan frekuensi intensitas  $i$ ,  $n$  menyatakan banyak pixel di citra) untuk setiap  $j$ . Hasil perhitungan tersebut merupakan pemetaan intensitas untuk perataan histogram. Pemetaan tersebut memetakan ke bilangan riil untuk menjaga akurasi perhitungan. Bilangan riil tersebut akan dibulatkan tepat sebelum mengaplikasikan pemetaan.

Cara kerja fungsi yang mengaplikasikan pemetaan cukup sederhana, yaitu dengan mengiterasi setiap pixel citra dan mentransformasikannya dengan pemetaan yang diberikan melalui argumen. Seperti pada program untuk menampilkan histogram, terdapat beberapa translasi nilai yang dilakukan untuk mengakomodasi perbedaan antara rentang nilai intensitas keabuan dan rentang nilai indeks larik Matlab.

Pada contoh hasil eksekusi terlihat bahwa perataan histogram berjalan sebagaimana mestinya. Rentang nilai histogram citra yang awalnya relatif kecil menjadi besar setelah perataan histogram diaplikasikan.

Dampaknya pun terlihat pada citra; citra yang awalnya terlihat memiliki kontras rendah dan terkesan seperti tercuci menjadi berkontras tinggi dan jelas setelah perataan histogram diaplikasikan. Pada contoh hasil eksekusi, khususnya yang pertama, terlihat salah satu kekurangan perataan histogram yang diaplikasikan, yaitu menimbulkan artefak dengan memperbesar selisih intensitas yang awalnya tidak terlihat.

## 2.4. Perbaikan Kualitas Citra dengan *Histogram Specification*

### Kode Program

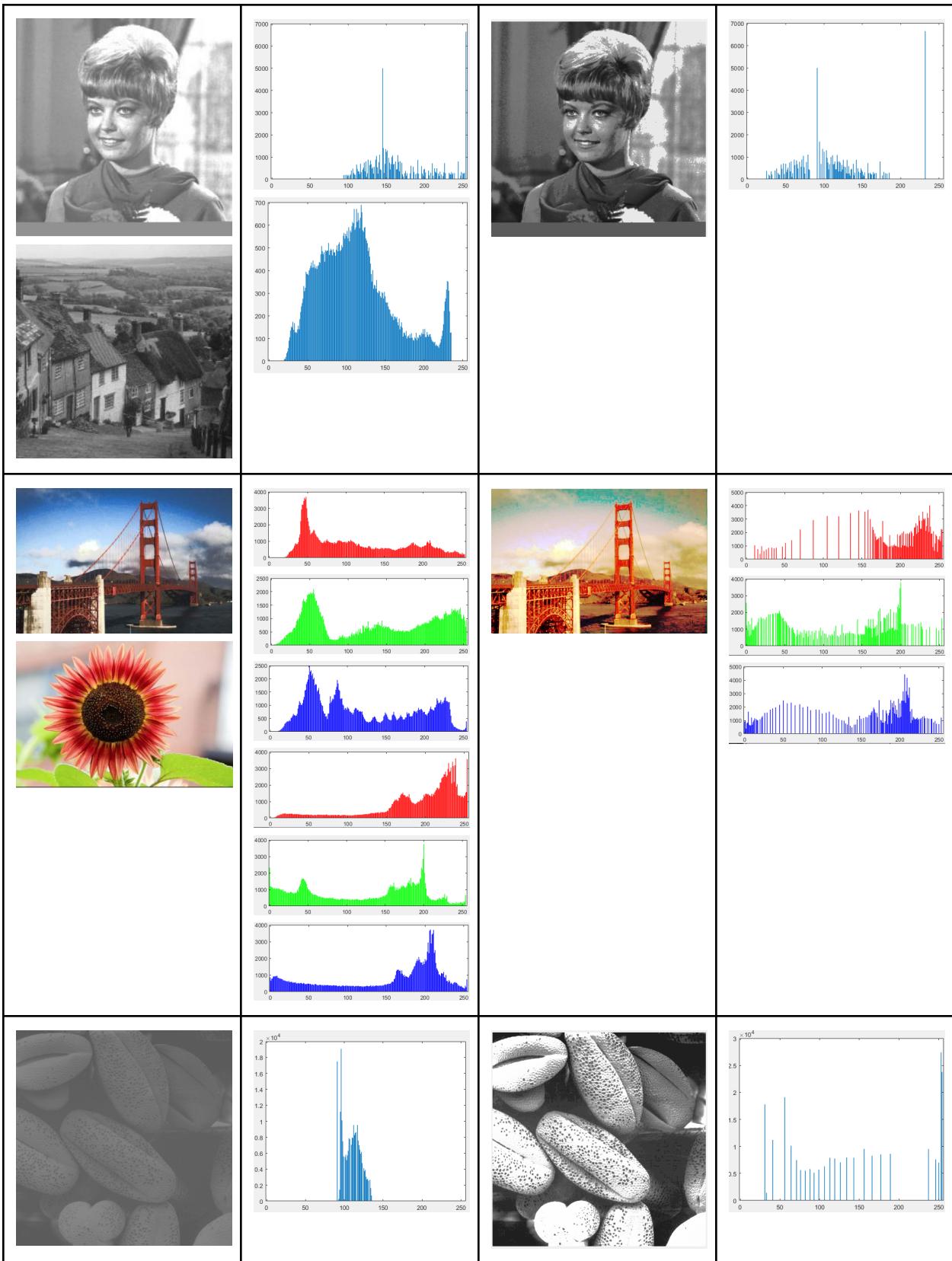
Program *histogram specification* memiliki dua komponen, yaitu fungsi yang menghitung pemetaan dan fungsi yang mengaplikasikan pemetaan. Fungsi yang mengaplikasikan pemetaan sama persis dengan yang digunakan pada program ke-3, sehingga fungsi baru yang dibuat hanya satu, yaitu seperti berikut.

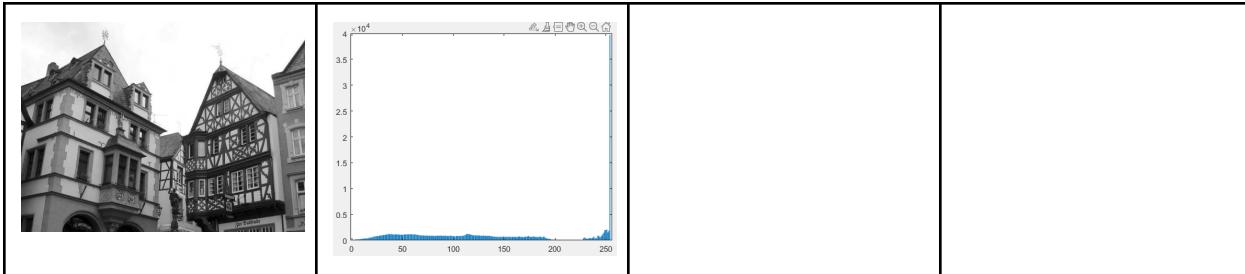
```
% Returns inverse histogram equalization result, in uint8
% res(i) = an integer in (0, 256] s.t. |map(res(i)) - i| is minimum
% if multiple possibility exists, minimum value is taken
function res = GetInverseHisteqMapping(comp, image)
    map = comp.GetHisteqMapping(image);
    [numColor, ~] = size(map);

    res = zeros([numColor, 256]);
    for c = 1:numColor
        for i = 1:256
            dist = 300;
            for j = 1:256
                if abs(map(c, j) - i) < dist
                    res(c, i) = j;
                    dist = abs(map(c, j) - i);
                end
            end
        end
    end
end
```

### Contoh Hasil Eksekusi

Citra Input dan Referensi	Histogram Citra Input dan Referensi	Citra Hasil	Histogram Citra Hasil
---------------------------	-------------------------------------	-------------	-----------------------





## Analisis

Untuk setiap nilai intensitas  $i$ , fungsi penghitung pemetaan di atas mencari nilai kodomain yang paling dekat dengan  $i$ , lalu membuat pemetaan baru dari  $i$  ke domain yang bersesuaian dengan kodomain tersebut. Perulangan pada fungsi di atas menggunakan *linear search* untuk pencarian dan memiliki kompleksitas total  $O(n^2)$  ( $n$  menyatakan kedalaman warna atau ukuran rentang nilai intensitas). Perulangan tersebut dapat dibuat lebih mangkus dengan mengganti *linear search* dengan algoritme pencarian lain, seperti *binary search* (dengan kompleksitas total  $O(n \log n)$ ) atau *two pointers* (dengan kompleksitas total  $O(n)$ ). Untuk kedalaman warna 8 bit, *linear search* dirasa cukup bagus.

Pada contoh hasil eksekusi terlihat bahwa *histogram specification* berjalan sebagaimana mestinya. Pada contoh pertama dan ketiga, terlihat bahwa kontras citra input yang awalnya rendah menjadi tinggi karena mengikuti kontras citra referensi. Pada contoh kedua, terlihat bahwa tona citra input menjadi lebih hangat atau merah mengikuti tona warna bunga citra referensi.

### 3. Alamat Github Program

<https://github.com/Jauhar-Wibisono/if4073-tugas-1>