



دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف دوم درس مبانی بینایی کامپیوتر

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: بهار ۱۴۰۱/۱۴۰۰

مدرس: دکتر نادر کریمی

دستیاران آموزشی: بهنام ساعدی - محمدرضا مزروعی

۱

ابتدا متغیرهای زیر را تعریف می‌کنیم.

H و W : ابعاد تصویر

I : تصویر اصلی

J : تصویر مورد بررسی

MAX_I : بزرگ‌ترین سطح روشنایی محتمل در تصویر (مقدار پیش‌فرض ۲۵۵)

MIN_I : کوچک‌ترین سطح روشنایی محتمل در تصویر (مقدار پیش‌فرض ۰)

۱.۱ MAE

۱.۱.۱ حداکثر

حالتی را تصور می‌کنیم که اختلاف سطح روشنایی هر دو پیکسل متناظر در تصویر حداکثر باشد (تصویر تمام سیاه و تمام سفید). بنابراین به ازای هر i و j دامنه، $I(i, j)$ برابر MAX_I و $J(i, j)$ برابر MIN_I یا بالعکس. پس داریم:

$$\max(\frac{1}{H.W} \sum_{i=1}^H \sum_{j=1}^W |I(i, j) - J(i, j)|) = \frac{1}{H.W} \times H.W \times |MAX_I - MIN_I| = |MAX_I - MIN_I|$$

که در حالت پیش‌فرض این مقدار برابر ۲۵۵ است.

۲.۱.۱ حداقل

حالتی را تصور می‌کنیم که اختلاف سطح روشنایی هر دو پیکسل متناظر در تصویر حداقل باشد (تصاویر برابر باشند). بنابراین به ازای هر i و j دامنه، $I(i, j)$ برابر $J(i, j)$ است. پس داریم:

$$\min(\frac{1}{H.W} \sum_{i=1}^H \sum_{j=1}^W |I(i, j) - J(i, j)|) = \frac{1}{H.W} \times H.W \times |I(i, j) - I(i, j)| = 0$$

۲.۱ MSE

۱.۲.۱ حداکثر

مشابه قسمت قبل، حالتی را تصور می‌کنیم که اختلاف سطح روشنایی هر دو پیکسل متناظر در تصویر حداکثر باشد (تصویر تمام سیاه و تمام سفید). بنابراین به ازای هر i و j دامنه، $I(i, j)$ برابر MAX_I و $J(i, j)$ برابر MIN_I یا بالعکس. پس داریم:

$$\max(\frac{1}{H.W} \sum_{i=1}^H \sum_{j=1}^W (I(i, j) - J(i, j))^2) = \frac{1}{H.W} \times H.W \times (MAX_I - MIN_I)^2 = (MAX_I - MIN_I)^2$$

که در حالت پیش‌فرض این مقدار برابر ۶۵۰۲۵ است.

۲.۲.۱ حداقل

مشابه قسمت قبل، حالتی را تصور می‌کنیم که اختلاف سطح روشنایی هر دو پیکسل متناظر در تصویر حداقل باشد (تصاویر برابر باشند). بنابراین به ازای هر i و j دامنه، $I(i, j)$ برابر $J(i, j)$ است. پس داریم:

$$\min(\frac{1}{H.W} \sum_{i=1}^H \sum_{j=1}^W (I(i, j) - J(i, j))^2) = \frac{1}{H.W} \times H.W \times (I(i, j) - I(i, j))^2 = 0$$

۳.۱ PSNR

برای بیشینه (کمینه) کردن تابع $10 \log_{10}(\frac{MAX_I^2}{MSE})$ کافیت تابع $\frac{MAX_I^2}{MSE}$ بیشینه (کمینه) کنیم.

۱.۳.۱ حداکثر

با فرض ناصفر بودن مقدار MAX_I داریم:

$$\max(10 \log_{10}(\frac{MAX_I^2}{MSE})) = \lim_{MSE \rightarrow 0}(10 \log_{10}(\frac{MAX_I^2}{MSE})) = \infty$$

پس به ازای دو تصویر با MSE نزدیک به صفر (دو تصویر تقریباً برابر)، مقدار PSNR به بی‌نهایت میل می‌کند.

۲.۳.۱ حداقل

اگر I تصویر تمام سیاه (MAX_I ناچیز باشد)، و J تصویر غیر تمام سیاه باشد، آنگاه چون MAX_I تقریباً صفر است داریم:

$$\min(10 \log_{10}(\frac{MAX_I^2}{MSE})) = \lim_{MAX_I \rightarrow 0}(10 \log_{10}(\frac{MAX_I^2}{MSE})) = -\infty$$

۲

سطح روشنایی هر پیکسل در تصویر خاکستری‌گونه از MIN_I (پیش‌فرض ۰) تا MAX_I (پیش‌فرض ۲۵۵) متغیر است. بدترین MSE که در واقع بزرگ‌ترین MSE است زمانی رخ می‌دهد که اختلاف سطح روشنایی هر دو پیکسل متناظر در تصاویر ماکسیمم باشد. در نتیجه برای ساخت چنین تصویری فاصله‌ی سطح روشنایی هر پیکسل در تصویر اولیه را از مقادیر MIN_I و MAX_I به دست می‌آوریم (قدر مطلق تفاضل). در صورتی که سطح روشنایی به MIN_I (سیاه) نزدیک‌تر بود، مقدار MAX_I (سفید) و در صورتی که به MAX_I (سفید) نزدیک‌تر بود، مقدار MIN_I (سیاه) را در پیکسل متناظر قرار می‌دهیم. در این حالت چون قدر مطلق تفاضل‌ها به ازای همه پیکسل‌های تصویر ماکسیمم هستند پس تضمین می‌شود که ماکسیمم MSE (بدترین MSE) را نسبت به تصویر ورودی داریم.

۳

۱.۳ الف

Function ۱.۱.۳

```

1 function J = toBlackWhite(I)
2 %TOBLACKWHITE Summary of this function goes here
3 % RGB to black and white with maximum psnr
4 J = uint8(zeros(size(I)));
5 for i = 1: size(I, 1)
6     for j = 1: size(I, 2)

```

```

7         avg_of_rgb = round(double(I(i, j, 1)) / 3 + double(I(i, j, 2) / 3) + double(I
            (i, j, 3) / 3));
8         if(255 - avg_of_rgb <= avg_of_rgb)
9             J(i, j, 1) = 255;
10            J(i, j, 2) = 255;
11            J(i, j, 3) = 255;
12        else
13            J(i, j, 1) = 0;
14            J(i, j, 2) = 0;
15            J(i, j, 3) = 0;
16        end
17    end
18 end
19 end

```

Driver code ۲.۱.۳

```

1 clc
2 clear
3 close all
4 imtool close all
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 I = imread("images\i3.jpg");
7 J = uint8(zeros(size(I)));
8 J = toBlackWhite(I);
9 psnr(J, I)
10 figure, imshow(I, []);
11 figure, imshow(J, []);

```

ب ۲.۳

Function ۱.۲.۳

```

1 function J = floydSteinberg(I)
2 %FLOYDSTEINBERG Summary of this function goes here
3     I = int16(I);
4     J = zeros(size(I));
5     for i = 1: size(I, 1)
6         for j = 1: size(I, 2)

```

```

7         old_value = I(i, j, 1);
8         remaining = int16(-1);
9         if(255 - I(i, j, 1) < I(i, j, 1))
10             I(i, j, :) = 255;
11             remaining = -int16(int16(255 - old_value));
12         else
13             I(i, j, :) = 0;
14             remaining = int16(old_value);
15         end
16         if(j < size(J, 2))
17             I(i, j + 1, :) = I(i, j + 1, 1) + floor((7 / 16) * remaining);
18             if(i < size(J, 1))
19                 I(i + 1, j + 1, :) = I(i + 1, j + 1, 1) + floor((1 / 16) * remaining)
20             ;
21         end
22         if(i < size(I, 1))
23             I(i + 1, j, :) = I(i + 1, j, 1) + floor((5 / 16) * remaining);
24             if(j > 1)
25                 I(i + 1, j - 1, :) = I(i + 1, j - 1, 1) + floor((3 / 16) * remaining)
26             ;
27         end
28     end
29 end
30 J = uint8(I);
31 end

```

Driver code ۲.۲.۳

```

1 clc
2 clear
3 close all
4 imtool close all
5 %%%%%%%%%%%%%%%
6 I = imread("images\i3b.png");
7 J = floydSteinberg(I);

```

```

8 figure, imshow(I, []);
9 figure, imshow(J, []);
10 x = I(:, :, 1);
11 y = J(:, :, 1);

```

۳.۳ ج

```

1 clc
2 clear
3 close all
4 imtool close all
5 %%%%%%%%%%%
6 I = imread("images\i3b.png");
7 J = floydSteinberg(I);
8 K = toBlackWhite(I);
9 fs_psnr = psnr(J, I);
10 bw_psnr = psnr(K, I);
11 figure, imshow(I, []);
12 figure, imshow(J, []);
13 figure, imshow(K, []);

```

با مقایسه‌ی PSNR در دو تصویر تولید شده در قسمت‌های قبل درمی‌یابیم که بهتر بودن PSNR یک تصویر نسبت به تصویر دیگر الزاما به معنی بهتر بودن کیفیت بصری آن تصویر نیست. همانطور که می‌بینیم، تصویر تولید شده توسط الگوریتم Floyd-Steinberg با وجود داشتن PSNR پایین‌تر نسبت به تصویر تولید شده توسط الگوریتم حریصانه (تابع toBlackWhite) جزئیات تصویر را بهتر مشخص می‌کند و کیفیت بصری بهتری دارد. پس نتیجه می‌گیریم که PSNR در همه جا معیار دقیقی برای مقایسه‌ی کیفیت تصاویر نیست.



شکل ۱: خروجی الگوریتم Floyd-Steinberg



شکل ۲: خروجی الگوریتم حریصانه

۴

Function ۱.۴

```

1 function J = My_Imresize_BL(Input_Image, Resizing_Factor)
2 %MY_IMRESIZE_BL Summary of this function goes here
3     old_row = size(Input_Image, 1);
4     old_column = size(Input_Image, 2);
5     new_row = ceil(old_row * Resizing_Factor);
6     new_column = ceil(old_column * Resizing_Factor);
7     J = uint8(zeros(new_row, new_column, 3));
8     r_ratio = double(old_row - 1) / double(new_row - 1);
9     c_ratio = double(old_column - 1) / double(new_column - 1);
10    for i = 1: new_row
11        i_floor = uint32(floor(r_ratio * i)) + 1;
12        i_ceil = uint32(ceil(r_ratio * i));
13        if (i_floor > old_row)
14            i_floor = old_row;
15        end
16        if (i_ceil > old_row)
17            i_ceil = old_row;
18        end
19        y = uint32(r_ratio * i - i_floor);
20
21        for j = 1: new_column
22            j_floor = uint32(floor(c_ratio * j)) + 1;

```

```

23     j_ceil = uint32(ceil(c_ratio * j));
24     if (j_floor > old_column)
25         j_floor = old_column;
26     end
27     if (j_ceil > old_column)
28         j_ceil = old_column;
29     end
30     x = uint32(c_ratio * j - j_floor);
31
32     a = uint32(zeros(1, 3));
33     b = uint32(zeros(1, 3));
34     c = uint32(zeros(1, 3));
35     d = uint32(zeros(1, 3));
36     e = uint32(zeros(1, 3));
37     %if (i_floor <= old_row && i_ceil <= old_row && j_floor <= old_column &&
    j_ceil <= old_column)
38         a = uint32(Input_Image(i_floor, j_floor, :));
39         b = uint32(Input_Image(i_floor, j_ceil, :));
40         c = uint32(Input_Image(i_ceil, j_floor, :));
41         d = uint32(Input_Image(i_ceil, j_ceil, :));
42         e = uint32(round(a + (b - a) * x + (c - a) * y + (a - b - c + d) * x * y)
    );
43         J(i, j, :) = e;
44     %end
45 end
46 end
47 end

```

Driver code ۲.۴

```

1 clc
2 clear
3 close all
4 imtool close all
5 %%%%%%%%%%%%%%
6 I = imread("images\lena.gif");
7 rf = 0.3;

```



```
8 J = My_Imresize_BL(I, rf);  
9 figure, imshow(I, []);  
10 figure, imshow(J, []);
```

۳.۴ نمونه خروجی



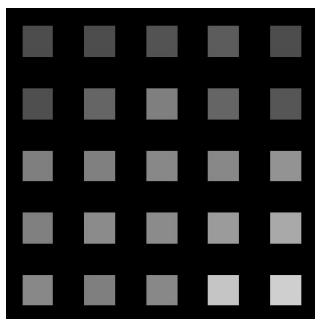
شکل ۳: تصویر اصلی



شکل ۴: تصویر با ابعاد ۳۰ درصد ابعاد تصویر اصلی

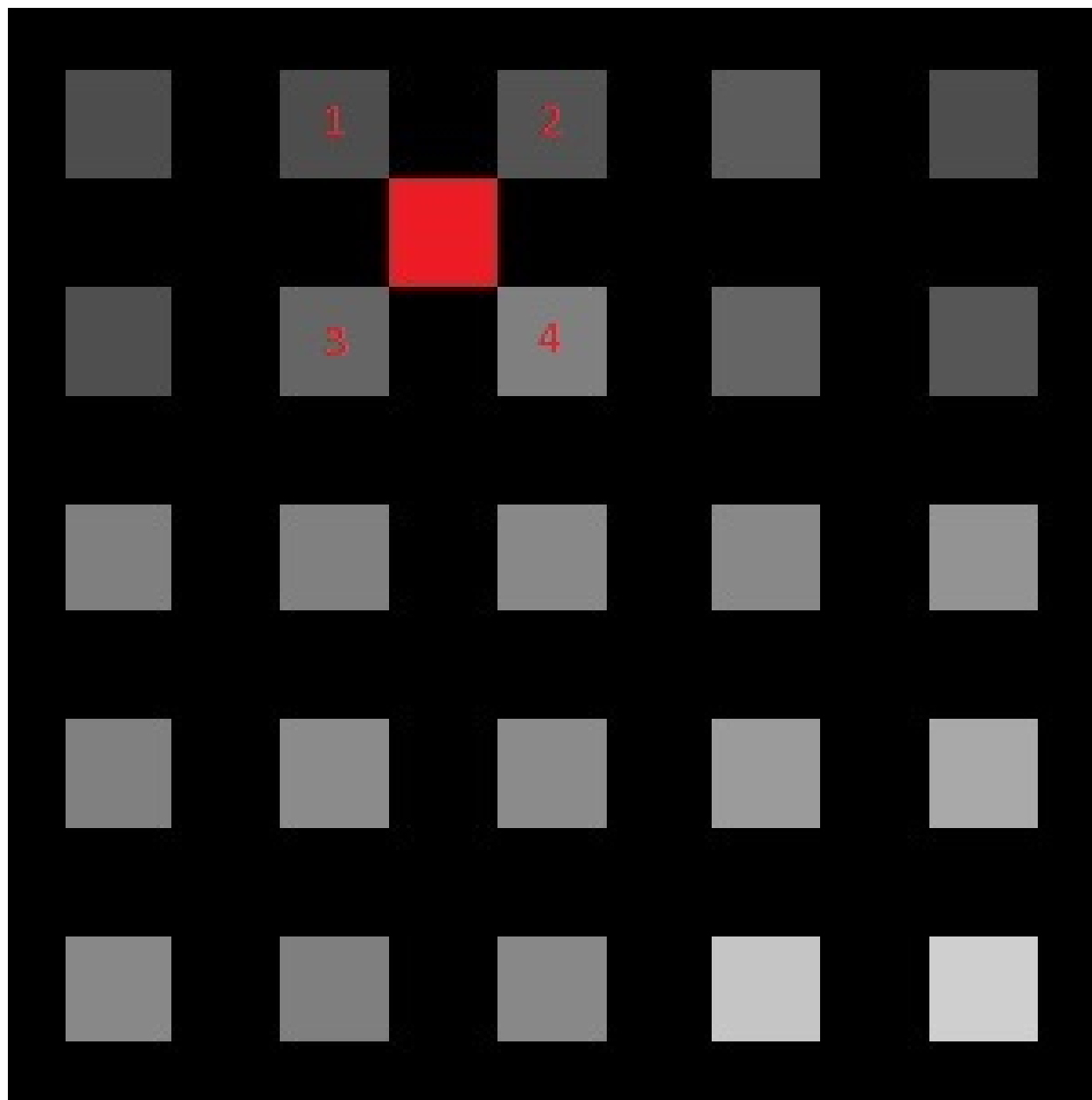
۵

پیکسل‌های تصویر را به طور یک در میان به شکل زیر می‌چینیم:



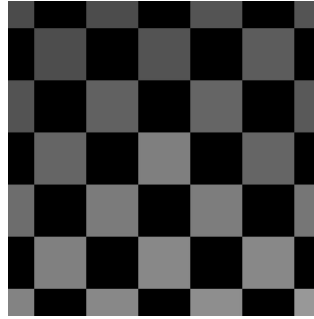
شکل ۵

سطح روشنایی پیکسل‌های قرار گرفته در سطر زوج و ستون زوج را برابر با میانگین پیکسل‌های مجاور (۴ پیکسل مشخص شده در شکل زیر در صورت وجود) قرار می‌دهیم. سطح روشنایی پیکسل‌های مرزی که دارای ۲ همسایه‌ی مورب و ۱ همسایه‌ی مورب هستند را هم به همین صورت مقداردهی می‌کنیم.



شکل ۶

پیکسل‌ها تصویر به شکل زیر در می‌آیند:



شکل ۷

در گام بعد، سطح روشنایی پیکسل‌های باقی‌مانده را برابر میانگین پیکسل‌های مجاور (بالا، پایین، چپ و راست) قرار می‌دهیم. سطح روشنایی پیکسل‌های مرزی هم به طریق مشابه تعیین می‌شوند. این الگوریتم در تابع `doubleDimensions` به شکل زیر پیاده‌سازی شده است.

۱.۵ Function

```

1 function K = doubleDimensions(J)
2     %initializing variables
3     height = 2 * size(J, 1);
4     width = 2 * size(J, 2);
5     K = uint8(zeros(height, width , 3));
6     %arranging pixels alternately
7     for i = 1: height / 2
8         for j = 1: width / 2
9             K(2 * i - 1, 2 * j - 1, :) = J(i, j, :);
10        end
11    end
12    %assigning pixels with even row and even column with average of diagonal neighbors
13    for i = 2: 2: height - 1
14        for j = 2: 2: width - 1
15            K(i, j, :) = (1 / 4) * K(i - 1, j - 1, :) + (1 / 4) * K(i - 1, j + 1, :) + (1
16                / 4) * K(i + 1, j - 1, :) + (1 / 4) * K(i + 1, j + 1, :);
17        end
18    end
19    %setting border pixels
20    for j = 2: 2: width - 1
21        K(height, j, :) = (1 / 2) * K(height - 1, j - 1, :) + (1 / 2) * K(height - 1, j +
22            1, :);

```

```

21     end
22     for i = 2: 2: height - 1
23         K(i, width, :) = (1 / 2) * K(i - 1, width - 1, :) + (1 / 2) * K(i + 1, width - 1,
24             :);
25     end
26     K(height, width, :) = K(height - 1, width - 1, :);
27     %assigning left pixels with average of up, down, right and left neighbors
28     for i = 2: height - 1
29         for j = 2: width - 1
30             if (mod(i, 2) == 0 && mod(j, 2) == 1) || (mod(i, 2) == 1 && mod(j, 2) == 0)
31                 K(i, j, :) = (1 / 4) * K(i - 1, j, :) + (1 / 4) * K(i + 1, j, :) + (1 /
32                 4) * K(i, j - 1, :) + (1 / 4) * K(i, j + 1, :);
33             end
34         end
35     end
36     %setting border pixels
37     for j = 2: width - 1
38         if (mod(j, 2) == 0)
39             K(1, j, :) = (1 / 3) * K(1, j - 1, :) + (1 / 3) * K(1, j + 1, :) + (1 / 3) *
40             K(2, j, :);
41         else
42             K(height, j, :) = (1 / 3) * K(height, j - 1, :) + (1 / 3) * K(height, j + 1,
43             :) + (1 / 3) * K(height - 1, j, :);
44         end
45     end
46     for i = 2: height - 1
47         if (mod(i, 2) == 0)
48             K(i, 1, :) = (1 / 3) * K(i - 1, 1, :) + (1 / 3) * K(i + 1, 1, :) + (1 / 3) *
49             K(i, 2, :);
50         else
51             K(i, width, :) = (1 / 3) * K(i - 1, width, :) + (1 / 3) * K(i + 1, width, :)
52             + (1 / 3) * K(i, width - 1, :);
53         end
54     end
55     K(height, 1, :) = (1 / 3) * K(height - 1, 1, :) + (1 / 3) * K(height - 1, 2, :) + (1
56     / 3) * K(height, 2, :);

```

```

50     K(1, width, :) = (1 / 3) * K(1, width - 1, :) + (1 / 3) * K(2, width - 1, :) + (1 /
      3) * K(2, width, :);
51 end

```

Driver Code ۲.۵

```

1  clc
2  clear
3  close all
4  imtool close all
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  I = imread("images\Cameraman.png");
7  J = imread("images\LR_Cameraman.png");
8  nn = imresize(J, 2, "nearest");
9  bil = imresize(J, 2, "bilinear");
10 bic = imresize(J, 2, "bicubic");
11
12 tic
13 K = doubleDimensions(J);
14 toc
15
16 [psnr(nn, I) psnr(bil, I) psnr(bic, I) psnr(K, I)]
17 imtool(I)
18 imtool(K)

```

جدول مقادیر PSNR به شکل زیر می باشد.

نام تصویر	مقدار PSNR به ازای Resizing_Factor برابر با ۲				
	روش Nearest Neighbor	روش Bilinear	روش Bicubic	روش پیشنهادی شما	زمان اجرا بر حسب ثانیه
Boat	25.5147	27.1040	26.9296	28.8864	0.899658
Peppers	28.1116	29.9490	29.7193	32.1106	0.941563
Cameraman	28.0316	30.3392	30.4896	34.6318	0.913748
House	27.5452	29.3679	29.2722	31.5815	0.228155
متوسط PSNR	27.30075	29.190475	29.102675	31.802575	

۶

الف ۱.۶

Function ۱.۱.۶

```

1 function hist_vector = myHist(I)
2 %MYHIST Summary of this function goes here
3     x_axis = 0: 255;
4     y_axis = zeros(1, 256);
5     for i = 1: size(I, 1)
6         for j = 1: size(I, 2)
7             y_axis(I(i, j) + 1) = y_axis(I(i, j) + 1) + 1;
8         end
9     end
10    hist_vector = y_axis;
11    figure, bar(x_axis, y_axis);
12 end

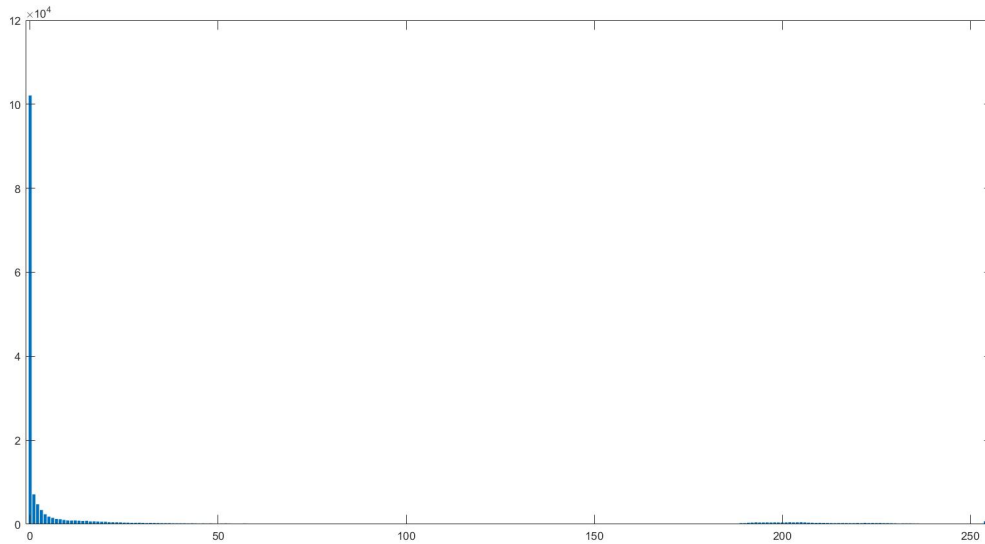
```

Driver code ۲.۱.۶

```

1 clc
2 clear
3 close all
4 imtool close all
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 I = imread("images\Image.tif");
7 myHist(I);

```



شکل ۸: هیستوگرام تصویر *Image.tif*

۲.۶ ب

همانطور که در هیستوگرام دیدیم، سطح روشنایی‌های صفر و نزدیک به صفر بیشتر پیکسل‌های این تصویر را تشکیل داده‌اند. اما سطح روشنایی‌های بزرگ‌تر و نزدیک به ۲۵۵ هم در این تصویر وجود دارد. برای بهبود کیفیت تصویر برای سطح روشنایی‌های پایین (سطح روشنایی کمتر مساوی k)، تابع تبدیلی می‌تواند مناسب باشد که بتواند سطوح روشنایی پایین را به محدوده وسیع‌تری نگاشت کند و از سوی دیگر قابلیت حفظ تقریبی سطوح روشنایی بالای قبلی را هم داشته باشد. بنابراین از تابع تبدیل لگاریتمی می‌توانیم استفاده کنیم. همچنین برای بهبود کیفیت تصویر برای سطح روشنایی‌های بالا (سطح روشنایی بزرگ‌تر از k)، بخش عمده‌ای از پیکسل‌ها دارای سطح روشنایی بالا هستند، ولی مناطقی شامل پیکسل‌هایی با سطح روشنایی پایین نیز در تصویر وجود دارد. این تبدیل، سطوح روشنایی بالا (نواحی روشن تصویر) را به محدوده گسترده‌تری نگاشت می‌کند. بنابراین تابع تبدیل معکوس لگاریتمی می‌تواند مناسب باشد (برای وضوح بیشتر قسمت سفید مقدار گاما را برابر با ۱.۲ قرار داده‌ایم و k می‌تواند ۱۲۷ باشد). توابع استفاده شده به ترتیب $J = 31.875 \times \log(I + 1)$ و $J = \frac{1}{255} I^2$ هستند. خروجی تصویر به شکل زیر می‌باشد.



شکل ۹: خروجی تابع تبدیل

Code ۳.۶

```

1  clc
2  clear
3  close all
4  imtool close all
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  I = imread("images\Image.tif");
7  gamma = 1.2;
8  alpha = double(255) ^ (1 - gamma);
9  for i = 1: size(I, 1)
10     for j = 1: size(I, 2)
11         if(I(i, j) <= 127)
12             J(i, j) = uint8(round(31.875 * log(double(I(i, j)) + 1)));
13         else

```

```
14         J(i, j) = uint8(round(alpha * double(I(i, j)) ^ gamma));
15     end
16 end
17 end
18 imtool(imadjust(J));
```

منابع