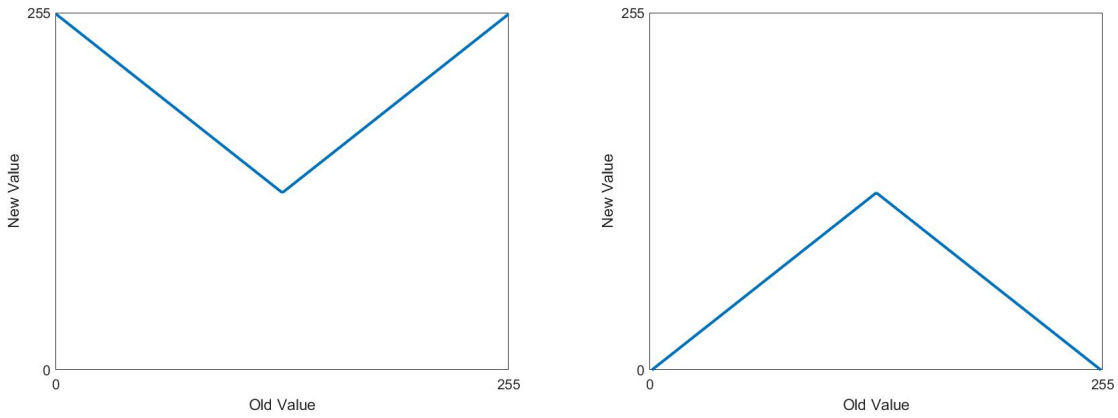
Digital Image Processing HW3

電機四乙10828241 陳大荃

October 19, 2022

1. Problem Description

Use MATLAB to perform point processing by following images.



(a) Conversion 1 (b) Conversion 2

Figure 1: Conversions

2. Program/Code

2.1 Conversion 1

The full function is available on GitHub: `https://github.com/belongtothenight/DIP_Code/blob/main/src/gimg_arith_op_1.m`

```
function img = ao1(img)
    [ih, iw] = size(img);
    for i = 1:ih
        for j = 1:iw
            if img(i, j) > 127
                continue
            else
                img(i, j) = 255 - img(i, j);
            end
        end
    end
end
```

This is the code in MATLAB to execute this function.

```
>> imgsrc = 'lena.tif';
>> imgdst = 'output.jpg';
>> speedtest = 0;
>> show = 1;
>> export = 0;
>> gimg_arith_op_1(imgsrc, imgdst, showimg)
```

2.2 Conversion 2

The full function is available on GitHub: `https://github.com/belongtothenight/DIP_Code/blob/main/src/gimg_arith_op_2.m`

```
function img = ao1(img)
```

```
    [ih, iw] = size(img);
    for i = 1:ih
        for j = 1:iw
            if img(i, j) > 127
                img(i, j) = 255 - img(i, j);
            else
                continue
            end
        end
    end
end
```

This is the code in MATLAB to execute this function.

```
>> imgsrc = 'lena.tif';
>> imgdst = 'output.jpg';
>> speedtest = 0;
>> show = 1;
>> export = 0;
>> gimg_arith_op_2(imgsrc, imgdst, showimg)
```

3. Result and Discussion

3.1 Method

My approach to this is to go through each pixel by for loop and turn its value to (255 - value) if its value is smaller than 127 for the first conversion, turn its value to (255 - value) if its value is larger than 127 for the second conversion.

Each conversion takes about 0.02 seconds, which is a lot longer than the conversions in homework 2, taking 0.00099945 seconds.

3.2 Result

(a) Conversion 1 (b) Original Image (c) Conversion 2

Figure 2: Result

3.3 Discussion

3.3.1 Colored Conversion

Colored images can also be converted by separating R, G, and B, converting them one by one, and turning them back into one.

(a) Conversion 1 (b) Original Image (c) Conversion 2

Figure 3: Result

3.3.2 Faster Convertion

Initially, the "for" loop is used to process all the elements. But using the vectorized function can significantly reduce conversion time.

For conversion 1, the following function can be used. Full code is updated to code1-gray and code1-colored.

```
img = (img > 127) .* img + (img <= 127) .* (255 - img);
```

For conversion 2, the following function can be used. Full code is updated to code2-gray and code2-colored.

```
img = (img > 127) .* (255 - img) + (img <= 127) .* img;
```

The following table is the data tested with my laptop.
1. "g1": 225x255 grayscale Lena picture. Conversion 1.
2. "g2": 225x255 grayscale Lena picture. Conversion 2.
3. "c1": 225x255 colored Lena picture. Conversion 1.
4. "c2": 225x255 colored Lena picture. Conversion 2.

Speed Test (100000 runs)(seconds)							
	for loop		vectorized		Speed Up (averaged)		
	Total Time	Averaged Time	Total Time	Averaged Time	Subtract	Divide	Percentage
g1	20.923535	0.000209	2.138721	0.000021	0.0001888	9.783201	89.778397
g2	21.005860	0.000210	2.360536	0.000024	0.0001860	8.898766	88.762487
c1	229.26762	0.002293	46.91116	0.000469	0.0018240	4.887272	79.538688
c2	204.14072	0.002041	45.52077	0.000455	0.0015860	4.484563	77.701282

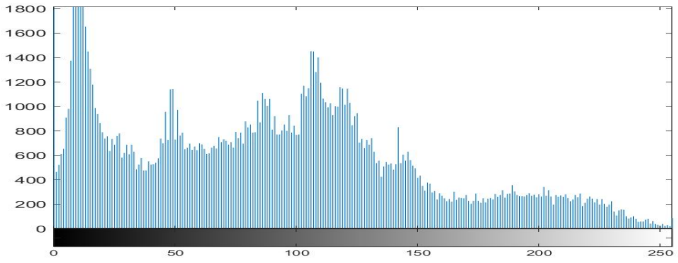
Based on the above table, using the vectorized function can largely increase the processing speed.

3.3.3 Histogram Equalization

Histogram equalization makes the histogram of images uniform.

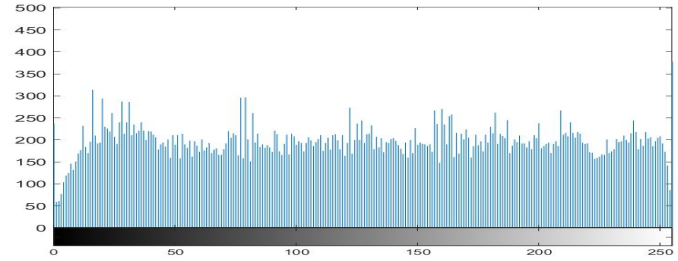
1. Code for gray-scale images.
2. Code for both gray-scale and RGB images.

(a) Original Image



(b) Original Histogram

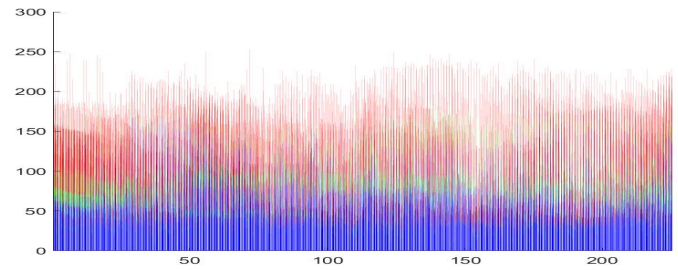
(c) Converted Image



(d) Converted Histogram

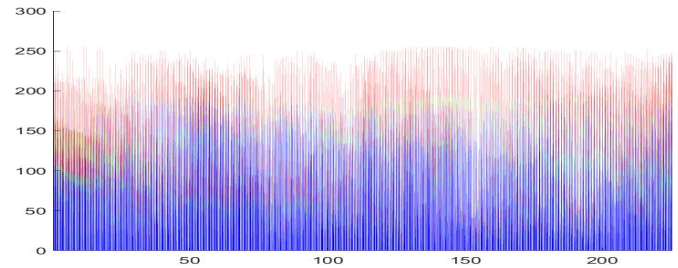
Figure 4: Gray-scale Images

(a) Original Image



(b) Original Histogram

(c) Converted Image



(d) Converted Histogram

Figure 5: Colored Images

The details of both gray-scale and RGB images have been amplified. While they have significant differences after conversion, few changes in the histogram of RGB are visible. Also, RGB equalization is bringing colors to the wrong places.

Speed test is also available for these two functions.
1. "g": 225x255 grayscale Lena picture. (10000 runs)
2. "c": 225x255 colored Lena picture. (1000 runs)

Speed Test (seconds)							
	for loop		vectorized		Speed Up (averaged)		
	Total Time	Averaged Time	Total Time	Averaged Time	Subtract	Divide	Percentage
g	203.26085	0.020326	216.71787	0.021672	-0.00135	0.937905	-6.620569
c	171.42364	0.171424	51.248595	0.051249	0.120175	3.344943	70.104127

Based on the above table, vectorization doesn't guarantee a performance boost. The reason behind this could be that much time is spent on equalizing histograms consisting of for-loops, and time saved from vectorized image value alternation can be too little compared with it.

As for the time saved on colored images, the unique value presented in the original image is a lot less than 256, which is a huge time saver since less amount of for-loop calculation is required.