Part 1.

1. The equations for log transformation:

s = c \* log(1 + r), where: c is a constant and r is the intensity of a pixel.

The effect of log transformation is to stretch low intensity values and compress high intensity values.

The equations for power-law transformation:

s = c \* rg , where: c is a constant, r is the intensity of a pixel, and g is a parameter controlling the power calculation.

The effect of power-law transformation is to enrich the functionality of log transformations. By defining different g values, different parts in the grey level can be stretched or compressed.

The image before transformation:

A picture containing tree, outdoor, sky

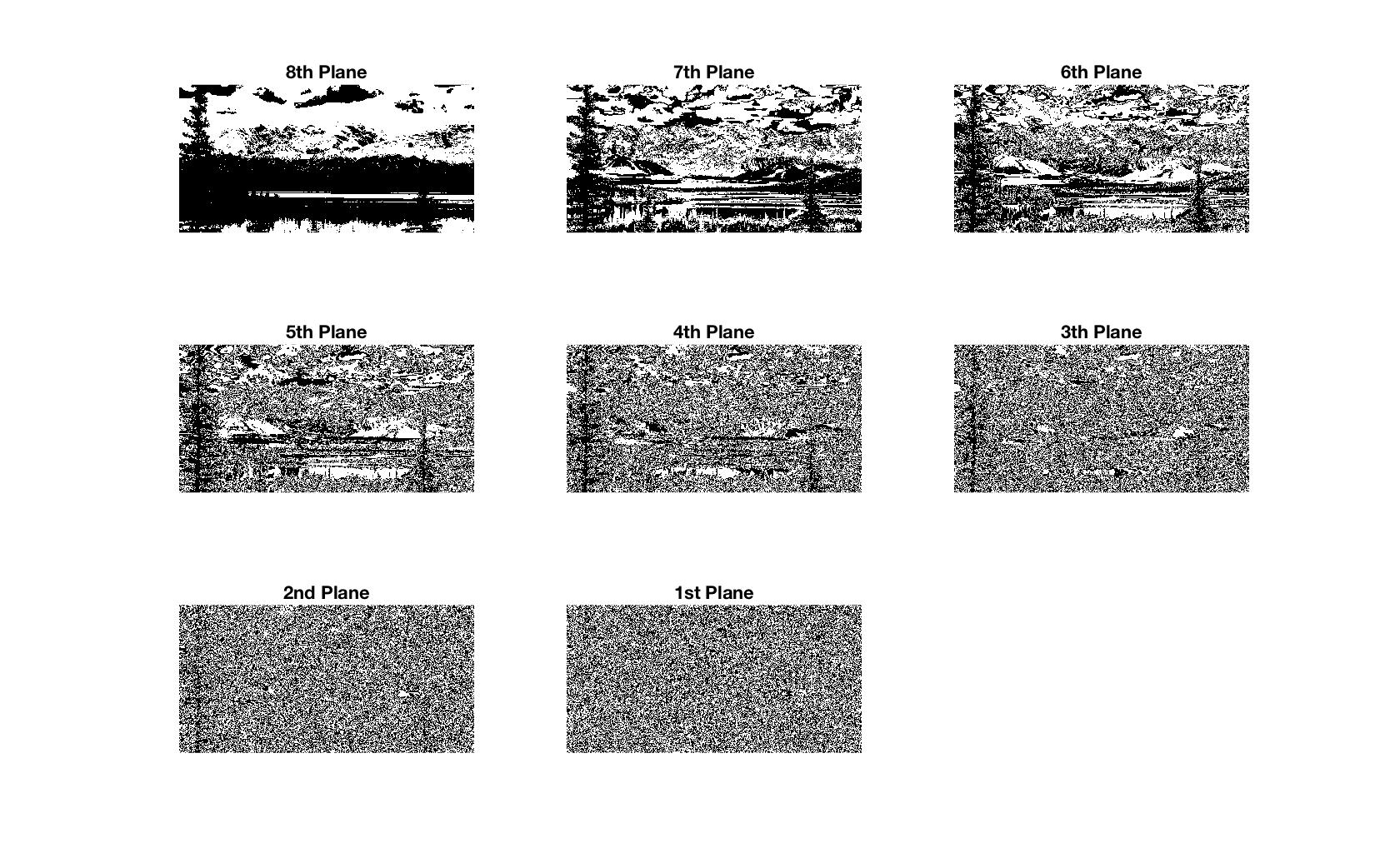
Description automatically generated

The images with different r transformation:

|  |  |
| --- | --- |
| A close up of a tree  Description automatically generated | A close up of a tree  Description automatically generated |
| r = 0.3 | r = 3 |

When a power law transformation with r = 0.3 is applied, the intensity levels tend to grow larger toward 1 under the effect of 0.3 power, which results in a brighter image and some effects like “wash-out”. When r = 3 is applied, intensity levels shrink toward 0, so they just get darker, reducing the wash-outs.

2. Images of bits slicing:



The reconstructed image from the highest 4 big planes:

A body of water

Description automatically generated

3.

For the original image:

|  |  |  |
| --- | --- | --- |
| A close up of text on a white background  Description automatically generated | A screenshot of a cell phone  Description automatically generated | A large body of water  Description automatically generated |
| hist. of before equalization | hist. of after equalization | image equalized |

For the r=0.3 image:

|  |  |  |
| --- | --- | --- |
|  | A screenshot of a cell phone  Description automatically generated | A picture containing outdoor, tree, sky, flying  Description automatically generated |
| hist. of before equalization | hist. of after equalization | image equalized |

For the r=3 image:

|  |  |  |
| --- | --- | --- |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated | A picture containing outdoor, tree, sky, flying  Description automatically generated |
| hist. of before equalization | hist. of after equalization | image equalized |

It could be observed that, after equalization, all three images get a far more balanced histogram at all grey level distributions, and this is exactly the purpose of equalization: to convert the distribution of grey levels toward uniform distribution. And it could be observed from the r=0.3 and r=3 images that, both images get a more balanced brightness after equalization comparing with their origin appearance. Distributions on the two extremes largely move toward more central bins.

4. The process of histogram matching in my understanding:

1) compute the probability distribution for the input image Pr(r)

2) apply histogram equalization on the input image: s = T(r)

3) given the desired distribution, apply histogram equalization on it: s’ = G(z)

4) do the inverse mapping from s to s’, Pr(z) = G-1(s’ --> s)

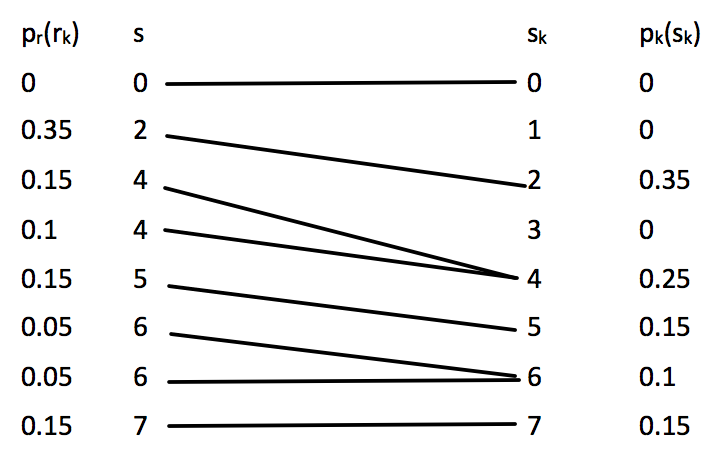
5) Get the output image z according to the inverse mapping G-1.

So here, equalization procedure acts as a bridge to equalize the two uniformed distributions. The mapping between s and s’ makes it possible to reversely map the desired output z to source image r.

Sometimes it is impossible to get exactly the same distribution as desired, because the mapping G-1 conforms to the rule that it finds the closest equalization, and sometimes there will be cases where several s equalizations are mapped to the same s’ equalization. This would cause little bit difference between the final output distribution and the desired one.

5.

|  |  |  |  |
| --- | --- | --- | --- |
| rk | nk | pr (rk) | s |
| r0 = 0 | 0 | 0 | 0 |
| r1 = 1 | 7 | 0.35 | 7\*(0+0.35) = 2.45 ≈ 2 |
| r2 = 2 | 3 | 0.15 | 7\*(0+0.35+0.15) = 3.5 ≈ 4 |
| r3 = 3 | 2 | 0.1 | 7\*(0+0.35+0.15+0.1) = 4.2 ≈ 4 |
| r4 = 4 | 3 | 0.15 | 7\*(0+0.35+0.15+0.1+0.15) = 5.25 ≈ 5 |
| r5 = 5 | 1 | 0.05 | 7\*(0+0.35+0.15+0.1+0.15+0.05) = 5.6 ≈ 6 |
| r6 = 6 | 1 | 0.05 | 7\*(0+0.35+0.15+0.1+0.15+0.05+0.05) = 5.95 ≈ 6 |
| r7 = 7 | 3 | 0.15 | 7\*(0+0.35+0.15+0.1+0.15+0.05+0.05+0.15) = 7 |



|  |  |
| --- | --- |
| Histogram before equalization: | Histogram after equalization: |
|  |  |

Part 2. A brief technical description of padding and shearing using Matlab

Step 1. Applying simple shear and color fill. Different values of a result in different extent of shear.

image = imread('alaska.jpg');

figure; subplot(2,2,1),imshow(image); title('Original Image');

i = 1;

for a=[0.2, 0.5, 0.7]

T = maketform('affine', [1 0 0; a 1 0; 0 0 1] );

color = [5 127 56]';

R = makeresampler({'cubic', 'nearest'}, 'fill');

B = imtransform(image, T, R, 'FillValues', color);

i = i + 1;

subplot(2,2,i),imshow(B); title(strcat('a=', num2str(a)));

end

A close up of a sign

Description automatically generated

Apply different filling colors:

i = 0;

for a=[50, 100, 150, 200]

T = maketform('affine', [1 0 0; 0.3 1 0; 0 0 1] );

color = [mod(a-30, 255) mod(a\*2, 255) mod(a\*3, 255)]';

R = makeresampler({'cubic', 'nearest'}, 'fill');

B = imtransform(image, T, R, 'FillValues', color);

i = i + 1;

subplot(2,2,i),imshow(B); title(strcat('color=', mat2str(color)));

end

A close up of a sign

Description automatically generated

Step 2. Transformation exploration

Apply meshgrid on the original image.

[U,V] = meshgrid(0:20:800,0:20:400);

gray = 0.65 \* [1 1 1];

figure; imshow(image); title("original image");

hold on;

line(U, V, 'Color', gray);

line(U', V', 'Color', gray);

A picture containing sky, building

Description automatically generated

Apply meshgrid on the sheared image.

T = maketform('affine', [1 0 0; 0.3 1 0; 0 0 1] );

color = [5 127 56]';

R = makeresampler({'cubic', 'nearest'}, 'fill');

B = imtransform(image, T, R, 'FillValues', color);

[U,V] = meshgrid(0:20:800,0:20:400);

[X,Y] = tformfwd(T,U,V);

gray = 0.65 \* [1 1 1];

figure; imshow(B); title("sheared image");

hold on;

line(X, Y, 'Color', gray);

line(X', Y', 'Color', gray);

A picture containing solar cell

Description automatically generated

Apply circle meshgrid on both original image and sheared image. Adjust gray to 1 so that they are displayed more clearly.

gray = 1 \* [1 1 1];

for u = 0:100:800

for v = 0:100:400

theta = (0 : 32)' \* (2 \* pi / 32);

uc = u + 20 \* cos(theta);

vc = v + 20 \* sin(theta);

[xc, yc] = tformfwd(T, uc, vc);

figure(h1); line(uc, vc, 'Color', gray);

figure(h2); line(xc, yc, 'Color', gray);

end

end

|  |  |
| --- | --- |
|  | A picture containing solar cell  Description automatically generated |

Step 3. Compare the 'fill', 'replicate', and 'bound' Pad Methods

1) Fill additional space around the sheared image.

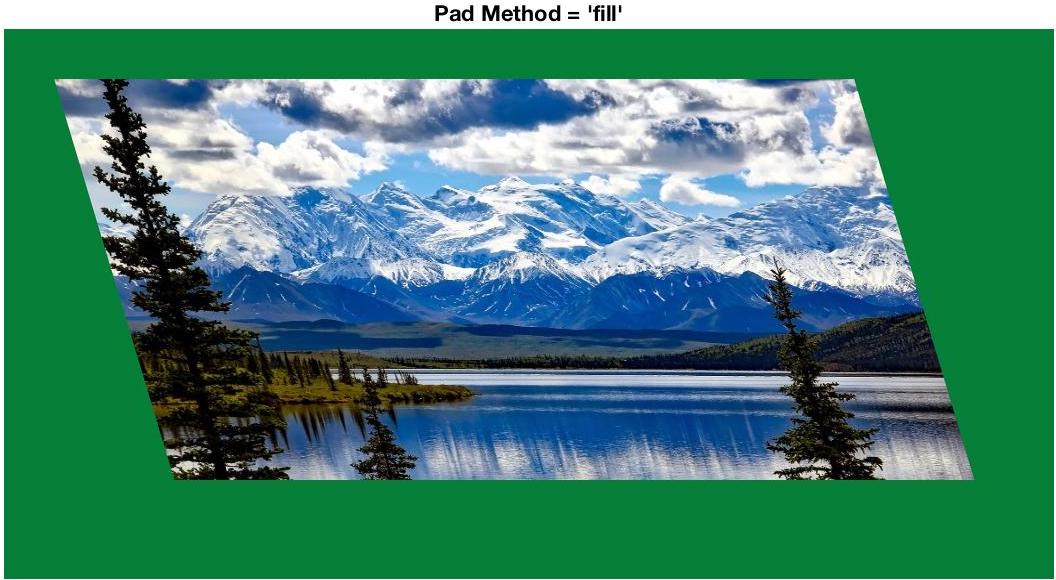
R = makeresampler({'cubic', 'nearest'},'fill');

color = [5 127 56]';

Bf = imtransform(image, T, R, 'XData',[-49 1000], 'YData',[-49 500], 'FillValues', color);

figure, imshow(Bf);

title('Pad Method = ''fill''');



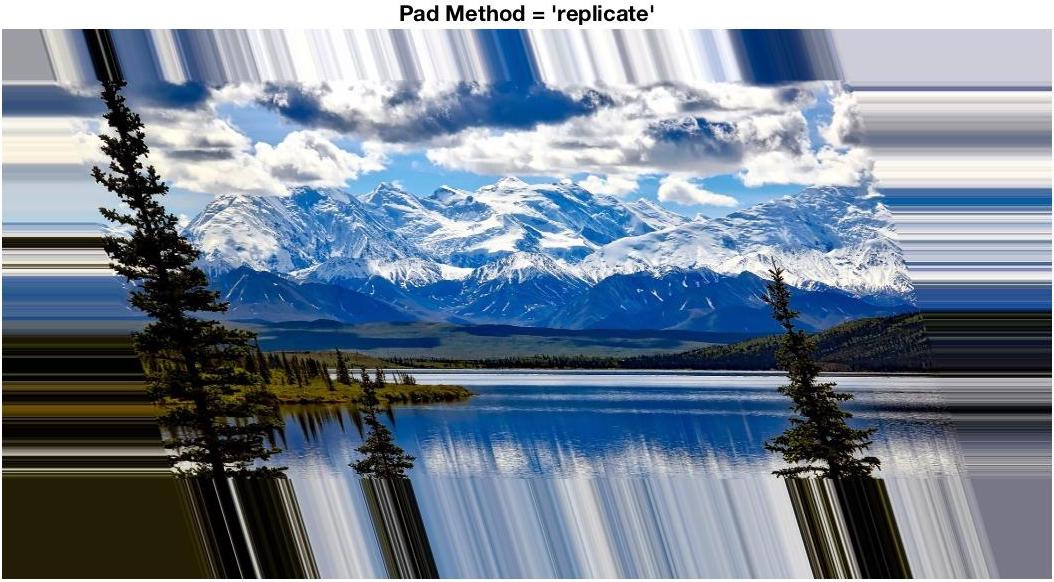
2) Replicate additional space.

R = makeresampler({'cubic', 'nearest'},'replicate');

Br = imtransform(image, T, R, 'XData',[-49 1000],'YData', [-49 500]);

figure, imshow(Br);

title('Pad Method = ''replicate''');



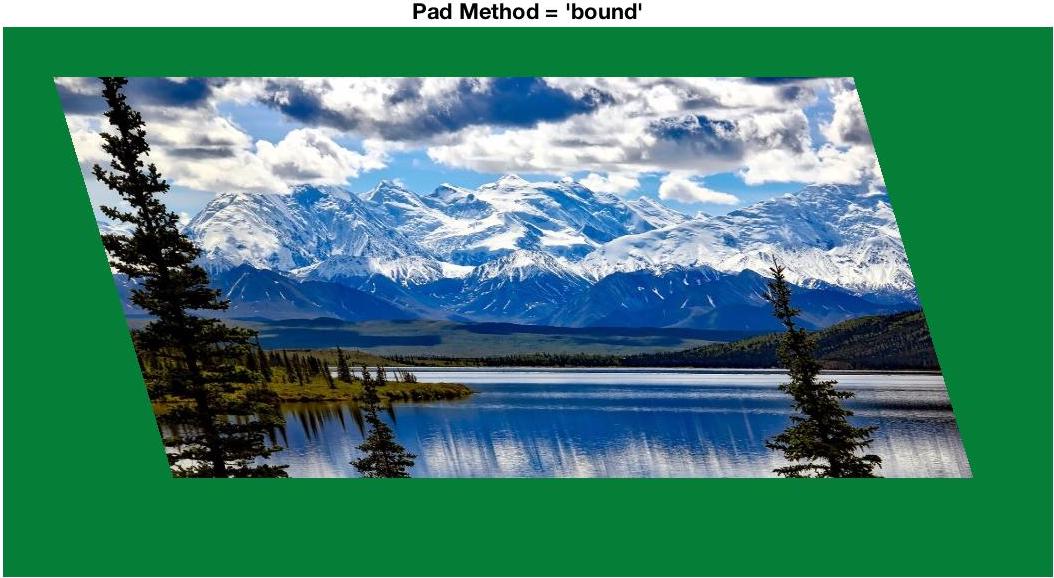
3) Try the “bound” method.

R = makeresampler({'cubic', 'nearest'}, 'bound');

Bb = imtransform(image, T, R, 'XData',[-49 1000],'YData',[-49 500], 'FillValues',[5 127 56]');

figure, imshow(Bb);

title('Pad Method = ''bound''');



Comparing with ‘fill’, the ‘Bound’ method presents a clearer strict boundary around the sheared image due to its implementation of applying replication around the edge. While ‘fill’ method does a cubic interpolation that would mix up filled value and original image values a little bit.

