Shivani Bhakta
A13832428

Homework 5

1. pdf: $f_X(x) = \frac{\lambda}{2} e^{-\lambda|x|}$ for random var X.

three-level quantizer q for X:

$$q(x) = \begin{cases} +b & \text{for } x > a \\ 0 & \text{for } -a \leq x \leq +a \\ -b & \text{for } x < -a \end{cases}$$

For the centroid condition to be met we know that

$$b = r_k = \frac{\int_{t_k}^{t_{k+1}} u\, p_u(u)\, du}{\int_{t_k}^{t_{k+1}} p_u(u)\, du} = \mathbb{E}[u \mid u \in k^{th} \text{ interval}]$$

$k = 1, 2, 3, \cdots, L$

We only look at positive x. $t_k \to a$ $t_{k+1} \to \infty$

$$b = \frac{\int_a^{\infty} x \frac{\lambda}{2} e^{-\lambda x}\, dx}{\int_a^{\infty} \frac{\lambda}{2} e^{-\lambda x}\, dx}$$

using $\displaystyle\int_a^{\infty} x e^{-\lambda x}\, dx = \frac{a e^{-\lambda a}}{\lambda} + \frac{e^{-\lambda a}}{\lambda^2}$

we get

$$b = \frac{\dfrac{\cancel{e^{-\lambda a}}}{\lambda^{\cancel{2}}}\left(a\cancel{\lambda}+1\right)}{-\dfrac{e^{-\lambda x}}{\lambda}\Bigg|_a^{\infty} \approx + \dfrac{\cancel{e^{-\lambda a}}}{\cancel{\lambda}}}$$

$$b = \frac{a\lambda + 1}{\lambda} = a + \frac{1}{\lambda}$$

b The nearest neighbor condition for optimality says that the decision boundary to be midpoint between the reconstruction levels, i.e.

$$t_k = \frac{r_k + r_{k+1}}{2}$$

$$a = \frac{b+0}{2} = \frac{b}{2}$$

since $b = a + \frac{1}{\lambda}$ we can use it to also satisfy centroid condition w/ nn-condition.

$\therefore 2a = a + \frac{1}{\lambda}$ $\quad \therefore a = \frac{1}{\lambda}$

$\therefore b = \frac{2}{\lambda}$

2 Lloyd Algorithm for quantizer Design.

$$T = \{1, 2, 3, 4, 8, 9, 12\} = \{t_1, t_2 \cdots t_K\}$$

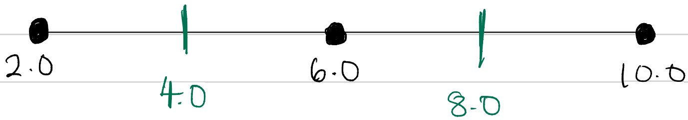
$K=7$

$$C_0 = \{2.0, 6.0, 10.0\} = \{r_1, r_2, r_3\}$$

= codebook

Round 1:

Given $\{r_i\}$, set the $\{t_i\}$ halfway in between.



$$D_1 = \{4.0, 8.0\}$$

Set the $\{r_i\}$ to be the center of mass

$$C_1 = \left\{ \frac{1+2+3+4}{4}, 8, \frac{9+12}{2} \right\}$$

$$C_1 = \{2.5, 8.0, 10.5\}$$

Choose left

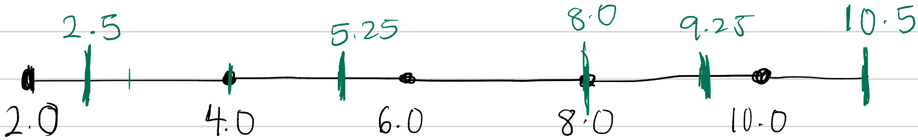
$$C_1 = \left\{ \frac{1+2+3}{3}, 4.0, \frac{8+9+12}{3} \right\}$$

$$= \{2.0, 4.0, 9.67\}$$

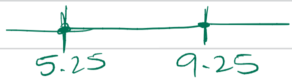
Chose right

Choose left

Round 2: Set $\{t_i\}$ halfway in between



$$D_2 = \left\{ \frac{2.5+8}{2} = 5.25, \ \frac{8+10.5}{2} = 9.25 \right\}$$



$$C_2 = \left\{ \frac{1+2+3+4}{4}, \ \frac{8+9}{2}, \ 12 \right\}$$

$$= \{2.5, 8.5, 12.0\}$$

Round 3: Set $\{t_i\}$ halfway

$$D_3 = \left\{ \frac{2.5+8.5}{2} = 5.5, \ \frac{8.5+12.0}{2} = 10.25 \right\}$$

$$C_3 = \left\{ \frac{1+2+3+4}{4}, \ \frac{8+9}{2}, \ 12 \right\}$$

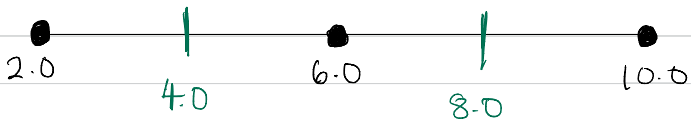
$$= \{2.5, 8.5, 12.0\}$$

Since $C_2 = C_3 \Rightarrow$ converged.

Choose right

Round 1:

Given $\{r_i\}$, set the $\{t_i\}$ halfway in between.



$$D_1 = \{4.0, 8.0\}$$

Set the $\{r_i\}$ to be the center of mass

$$C_1 = \left\{ \frac{1+2+3}{3}, 4.0, \frac{8+9+12}{3} \right\}$$

$$= \{2.0, 4.0, 9.67\}$$

Choose right

Round 2:

$$D_1 = \left\{ \frac{2.0+4.0}{2}, \frac{4.0+9.67}{2} \right\}$$

$$= \{3.0, 6.833\}$$

$$C_2 = \left\{ \frac{1+2}{2}, \frac{3+4}{2}, \frac{8+9+12}{3} \right\}$$

$$C_2 = \{1.5, 3.5, 9.67\}$$

choose right

Round 3: set $\{t_i\}$ halfway in between

$$D_3 = \left\{ \frac{1.5+3.5}{2} = 2.5, \quad \frac{3.5+9.67}{2} = 6.58 \right\}$$
$$= \{2.5, 6.58\}$$

$$C_3 = \left\{ \frac{1+2}{2}, \frac{3+4}{2}, \frac{8+9+12}{3} \right\}$$

$$C_3 = \{1.5, 3.5, 9.67\}$$

Since $C_2 = C_3 \Rightarrow$ converged.

The tie-breaking rule (Right or left) converges in same number of iterations.

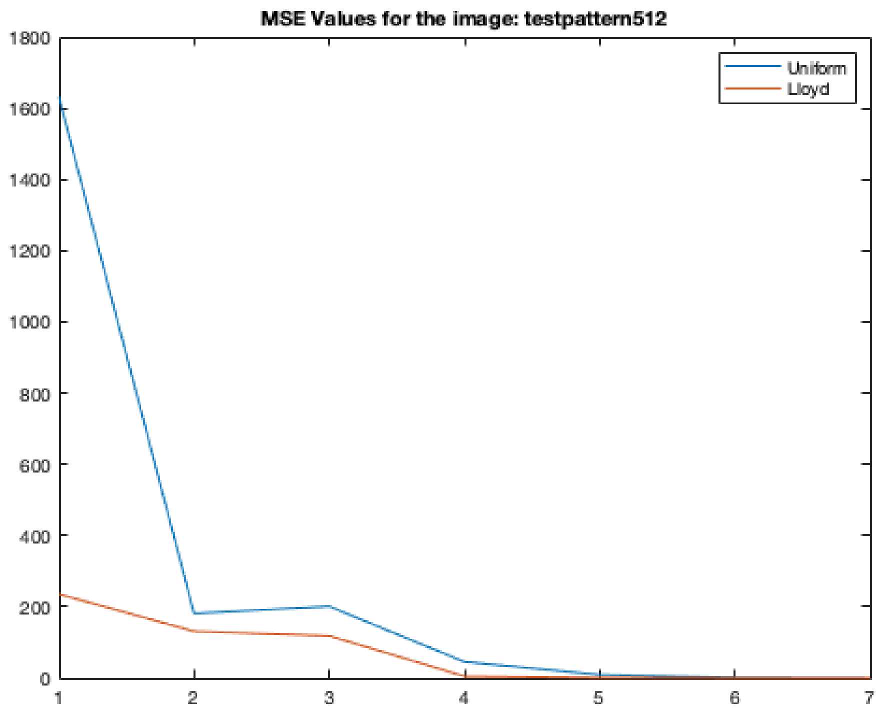
Contents

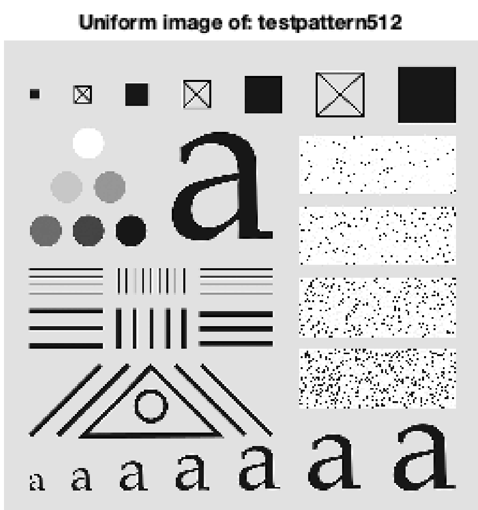
- ECE 253 Homework 5
- *Part(ab)*
- *_Part(c)_*

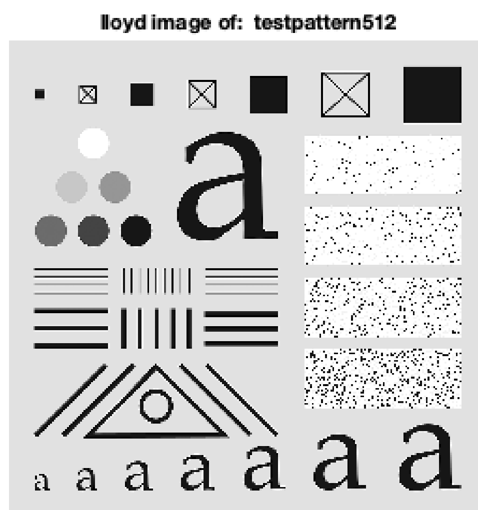
ECE 253 Homework 5

Shivani Bhakta A13832428 Problem 3 Comparing quantization for images

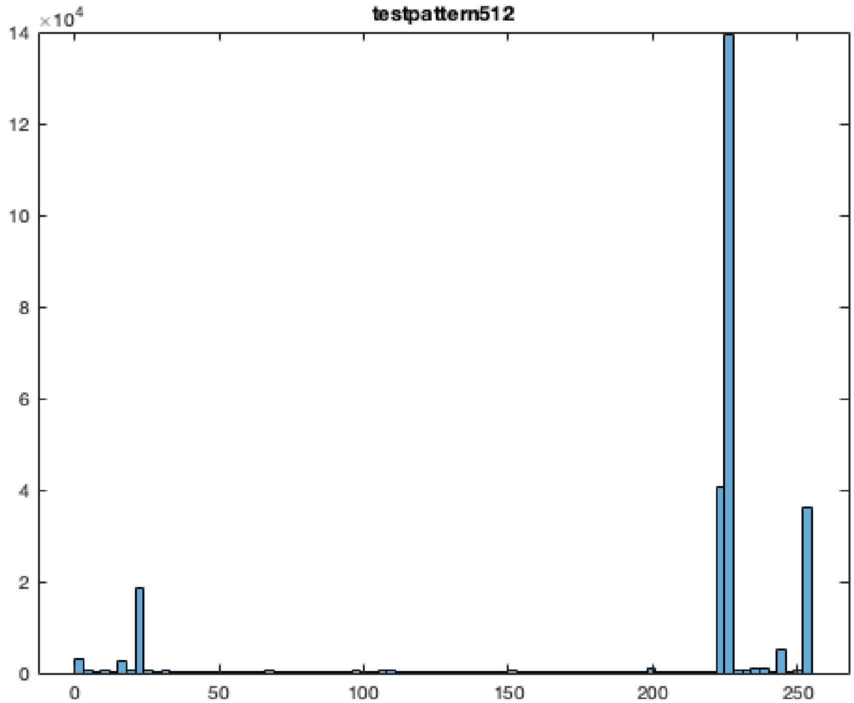
```
clear, clc, close all;
testpattern512 = imread('testpattern512.tif');
vase = imread('vase.tif');
astronaut = imread('astronaut.tif');
% input_img = testpattern512;
% input_img = vase;
% input_img = astronaut;
% vase = imread(Tiff('vase.tif', 'r'));

hw3_b(double(testpattern512), 'testpattern512')
%
```



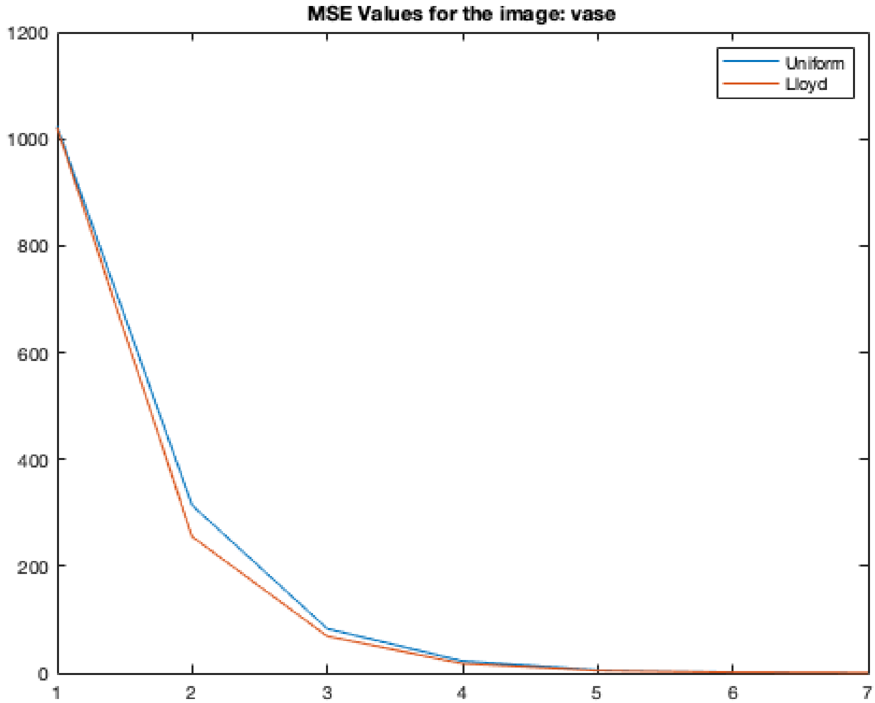




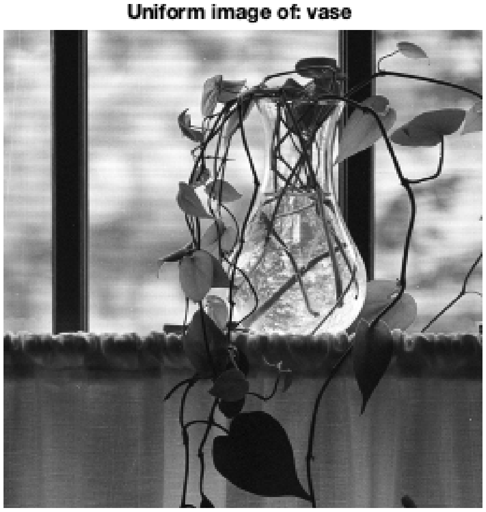

```
figure, histogram(testpattern512), title('testpattern512')
%
```



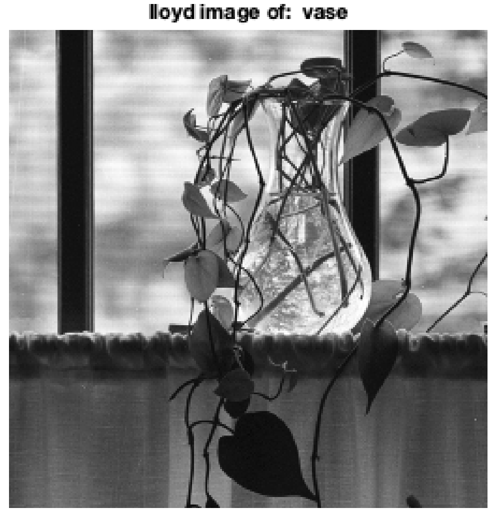

```
hw3_b(double(vase), 'vase')
%
```



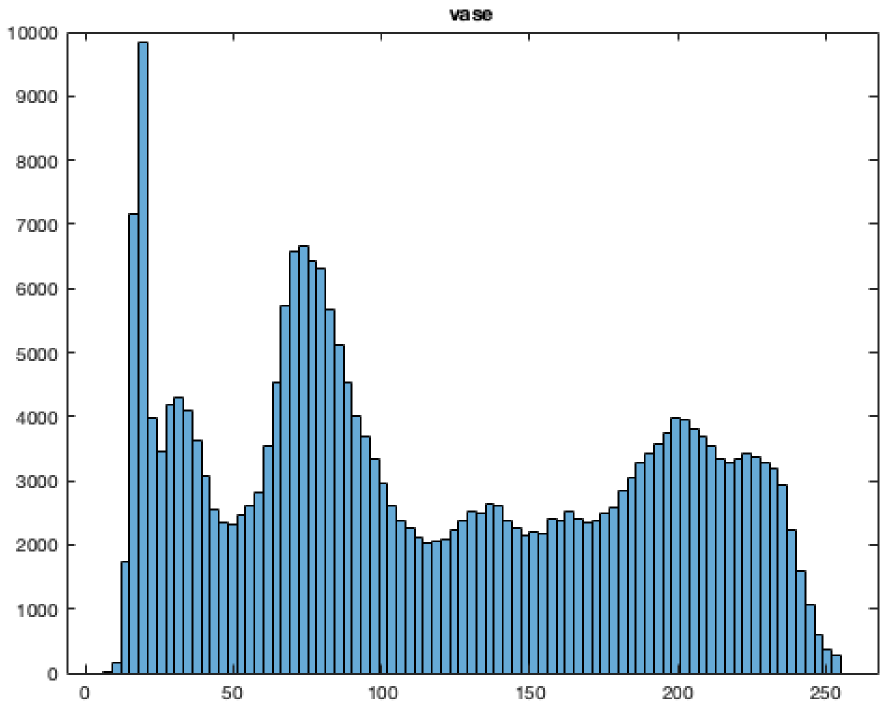
Uniform image of: vase



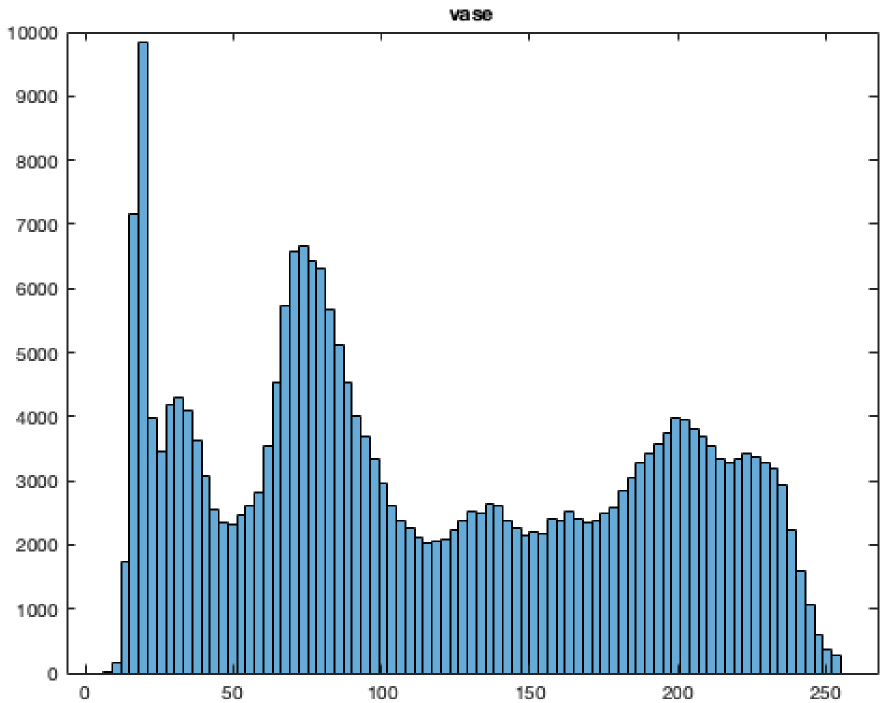
lloyd image of: vase

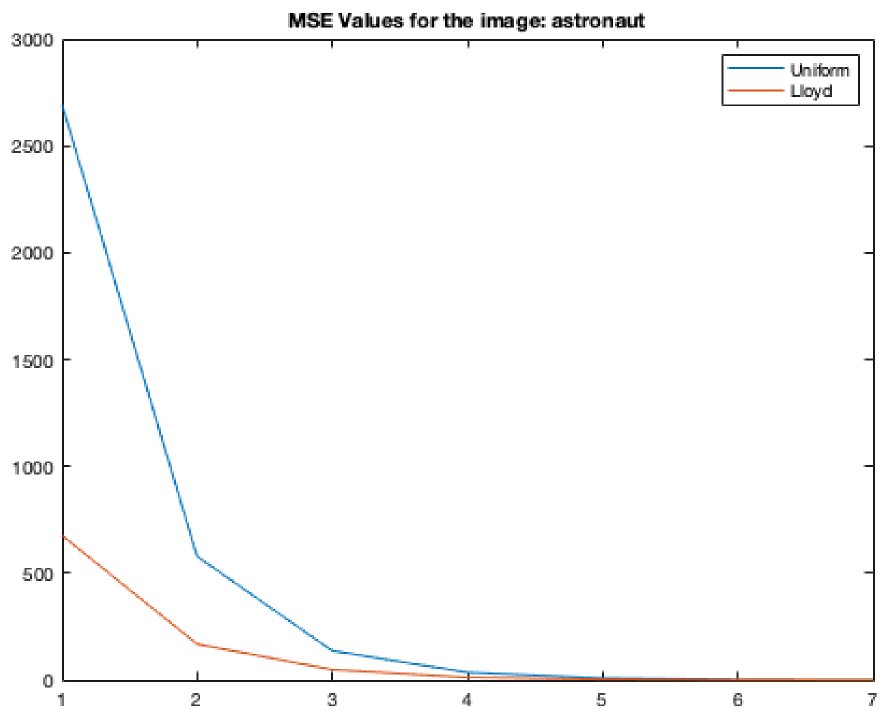


```
figure, histogram(vase), title('vase')
%
```




```
hw3_b(double(astronaut), 'astronaut')
%
```

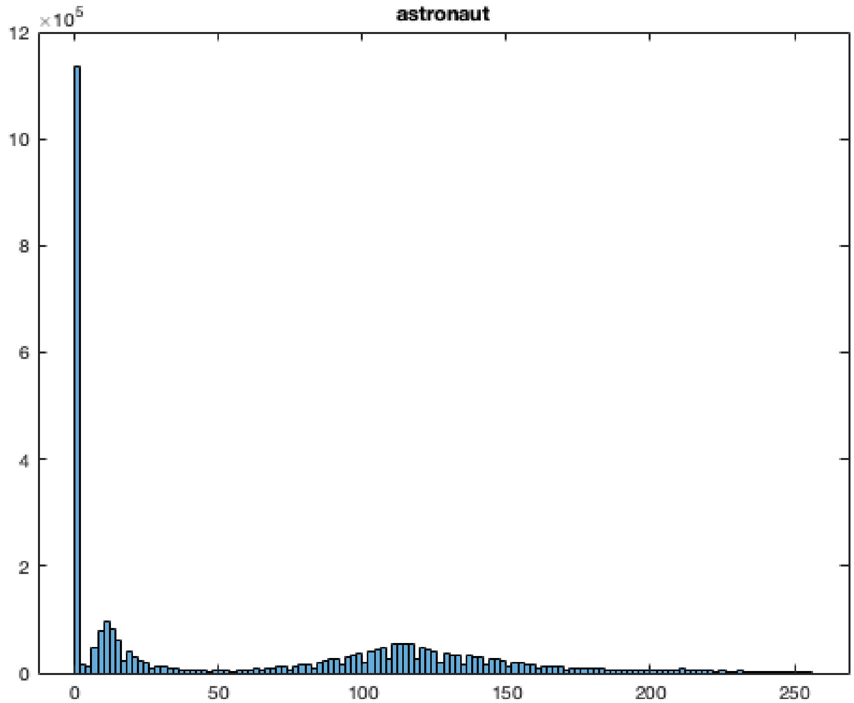

vase
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250


MSE Values for the image: astronaut
Uniform
Lloyd
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7

Uniform image of: astronaut

lloyd image of: astronaut

```
figure, histogram(astronaut), title('astronaut')
```



Part(ab)

- As it can be seen in all the images above, Lloyd Max quantization does better than uniform quantization in general. This is expected, because llyod considers the probability distribution of each pixel points and quantize the points based on that distrubution. Basically, if we see the histograms of these images, there are some parts which are densely populated, so for these parts llyod assigns more quantization levels, and for the less dense areas we don't see steps that closer to each other. This makes a use difference and help with the MSE as it can be seen from the MSE plots. This is actually very avidant in testpattern and astronut pictures and mse plots. - Looking at the historgram of vase and it's MSE, we can see that the hist is more uniformly distributed compared to testpattern, and therefore, uniform or llyod are not much different. While the mse plots of other two has some kind of spike where we have the uneven distribution in the histogram. And since uniform doesn't take these densly populated area into consideration, it had the highest mse difference compare to llyod in those regions.

Looking at techinical terms, we satisdfy noth nearest neighbor and centroid conditions in llyod, while only the centroid in uniform quantization.

*_Part(c)_*

- We see that for the testpattern image, from bit-2 to bit-3 the mse increases for the uniform quantizer. This is probably because of the for bit-2 the quantization levels were probably closer to the dense areas/spike areas, while futher in the other one, leading to increased mse for the latter.

- Although we don't see Lloyd quantizer getting worse for increasing bit rate, it might happen where there is an increase in mse. Let us say, the tie breaking rule for the boundary points, was decided such that it didn't work in the favor of those 'dense/spike' areas. We might still converge but it might not be the best. Also, some initial assumptions could turn up these results.

```
function [] = hw3_b(input_img, name)

    mse=[];
    for s = 1:7
        output_img = Uniform_Quantizer2(uint8(input_img), s);
        mse(end+1) = immse(output_img, uint8(input_img));
    end

    x=1:1:7;
    figure,
    plot(x,mse)
```

```
    hold on;

    mse_lloyd=[];
    for s = 1:7

        [N,M] = size(input_img);
        training_set = reshape(input_img,N*M,1);

        [partition, codebook] = lloyds(training_set, pow2(s));
        lloyd_output = zeros(N,M);

        for ii = 1:N
            for jj = 1:M
                value = input_img(ii,jj);
                idx = 1;
                while idx <= length(partition) && value>partition(idx)
                    % when it is outside the bounds
                    idx = idx + 1;
                end
                lloyd_output(ii,jj) = codebook(idx);
            end
        end

        lloyd_output = uint8(lloyd_output);
        mse_lloyd(end+1)= immse(lloyd_output, uint8(input_img));
    end

    plot(x,mse_lloyd)
    legend("Uniform","Lloyd"),
    str = sprintf('MSE Values for the image: %s', name);
    title(str);

    figure,
    str = sprintf('Uniform image of: %s ', name);
    imshow(output_img, [0,256]), title(str);

    figure,
    str = sprintf('lloyd image of:  %s', name);
    imshow(lloyd_output, [0,256]), title(str);


end
```

```
function [output_img] = Uniform_Quantizer2(input_img, s)

input_img = uint8(input_img);
% Write a function that takes as inputs an 8-bit image and a scalar
% s ∈ [1,7] and performs uniform quantization so that the output is
% quantized to a s-bit image.

output_img = idivide(input_img, pow2(8-s)).*pow2(8-s); %round off
output_img = output_img + pow2(8-s-1);

end
```

Published with MATLAB® R2022b