Digital Image Processing

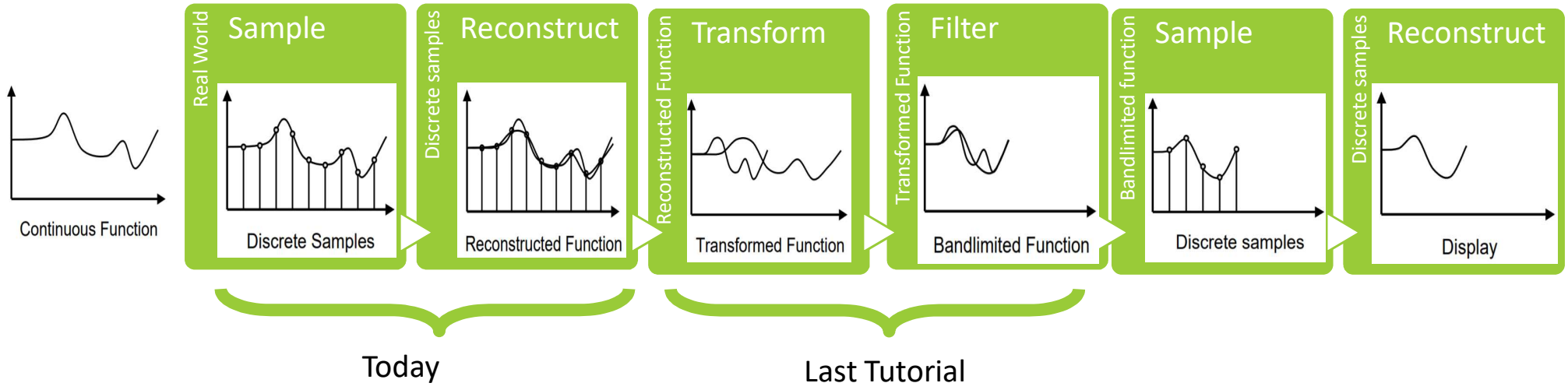
5. QUANTIZATION

Session 5

Quantization:

- Sampling and Quantization Definition
- Uniform quantizer
- Optimal uniform quantizer - Max-Lloyd
- Non-uniform quantizer
- Vector quantization
- K-means

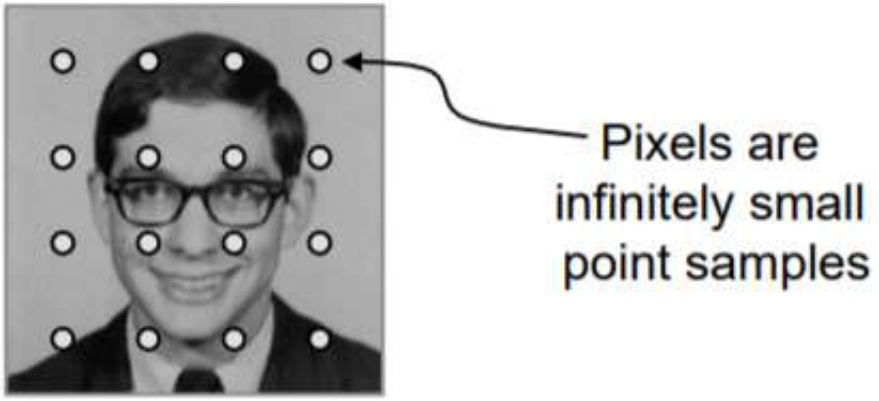
Image Processing



Is a resampling problem…

Image sampling and quantization

- Image is a 2D linear array of samples
- To create a digital image we need to convert the continuous sensed data into digital form
- The **Digital Image is represented** by:
  1. The points (x,y) on the grid on which we sample
  2. The Intensity on that specific point
- The **quality** of the image is **determined** by:
  1. The **number of samples on the grid**
  2. The **number of the possible Intensity levels**



Spatial and Intensity Resolution

- **Spatial Resolution** = The size of the grid on which we'd like to present the image

  (Number of rows and columns) / Number of samples.

  For example: 8*8 / 256*256 and etc…

  Sampling is the digitization of the spatial coordinates – the principal factor defining the resolution of an image.

  Sampling is mapping the original image into those **coordinates.**

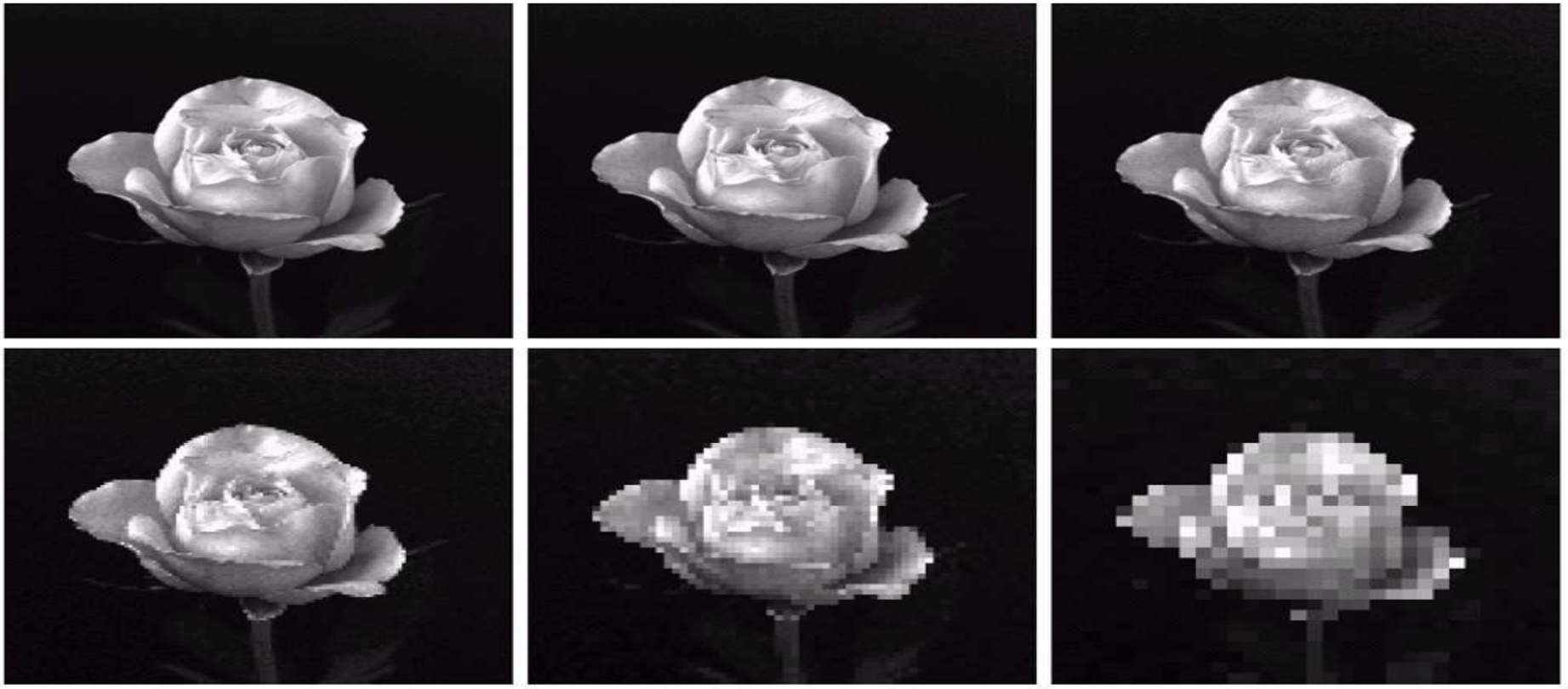
- **Intensity Resolution** = number of the bits that represent the color/grey intensity (also called the amplitude)

  Quantization is the principal factor defining the intensity resolution (8 bits, 16 bits and etc.).

  Its mapping the original image into those **intensities.**

Sampling

Reduce the file size of pixels => Save image as close as possible to the original.



Quantization levels

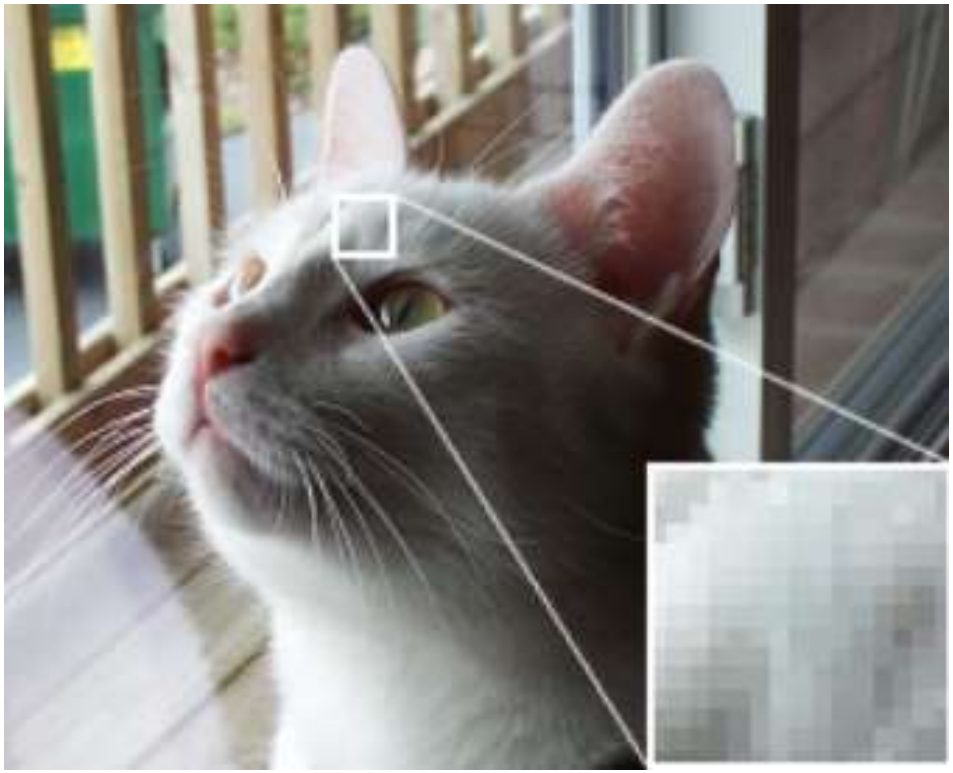
In the grey scale for example:

1 bit quantization: 2^1= 2 gray levels

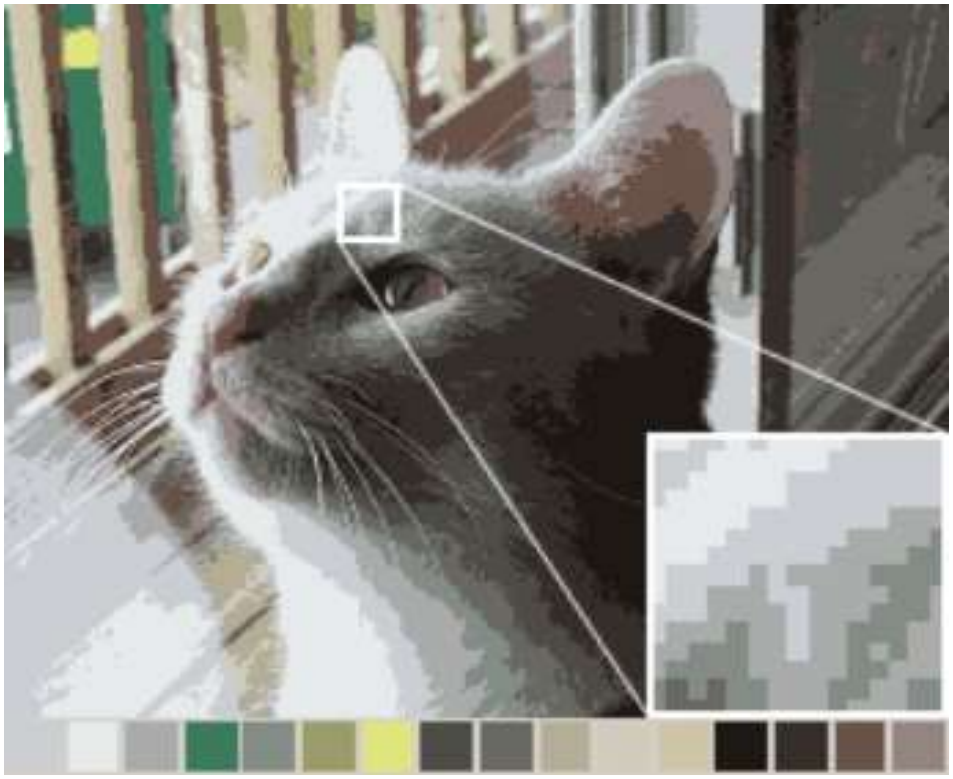
(0: black, 1: white)

8 bit quantization: 2^8=256 gray levels

(0: black, 255: white)



24 Bit Color



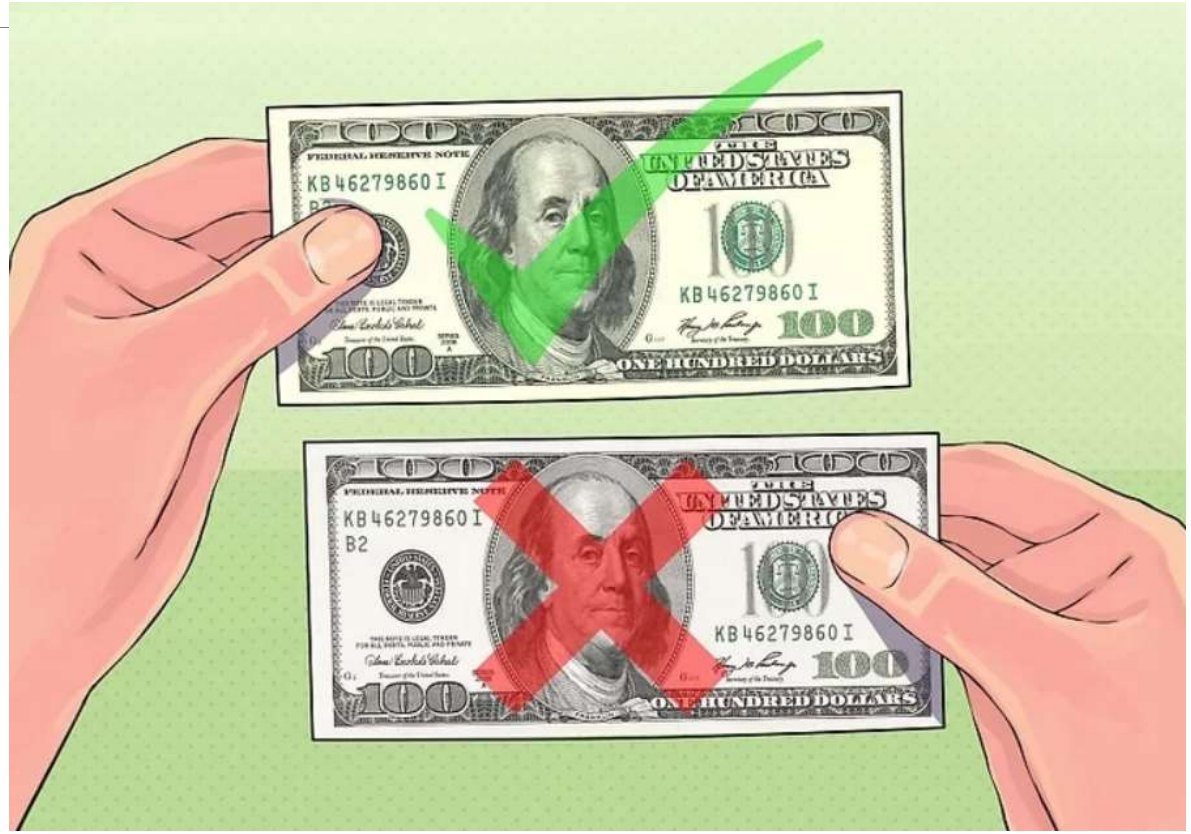
4 Bit Color

16 million colors

16 colors

Quantization

by Ron Leishman



Uniform Quantization

- Equal distances between adjacent decision levels and between adjacent reconstruction levels

  $t_l - t_{l-1} = r_l - r_{l-1} = q$

- Parameters of Uniform Quantization –
  - L: levels ($L = 2^R$)
  - B: dynamic range

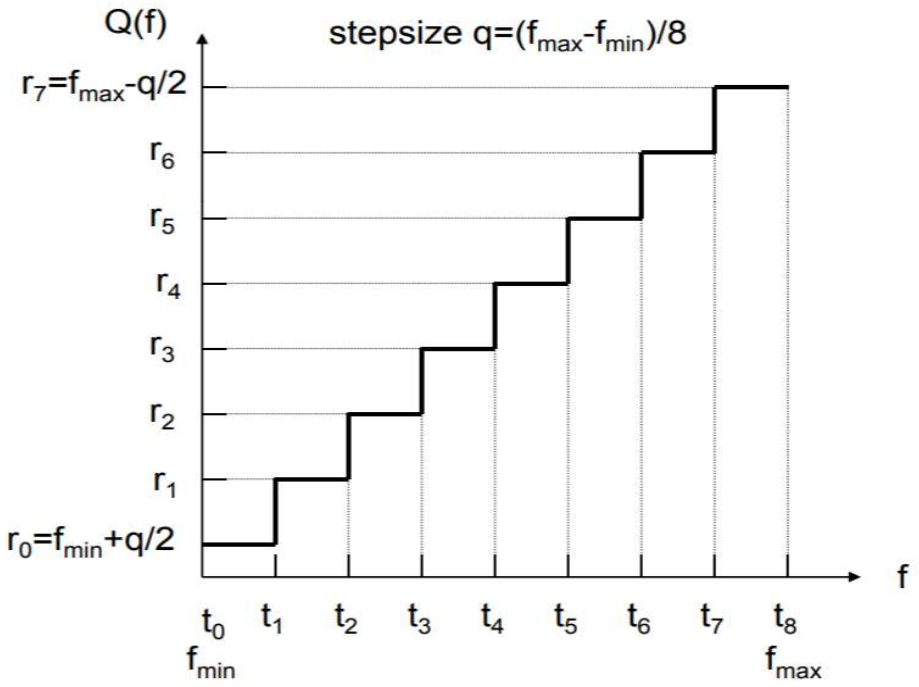
    $B = f_{max} - f_{min}$
  - q: quantization interval (step size)

    $q = B/L = B*2^{-R}$

$$Q(f) = \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min}$$

The new value on the limited screen

Uniform quantization- Functional representation



$$Q(f) = \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min}$$

Uniform quantizer- discrete

- Input signal is discrete
- Digital Image of 256 gray levels
- Quantize it into 4 levels
- fmin= 0,
- fmax = 256,
- q (step size – every 64 bit are mapped to 1 value) = 256 / 4 = 64,
- q/2 = 32

$$Q(f) = \lfloor f / 64 \rfloor * 64 + 32$$

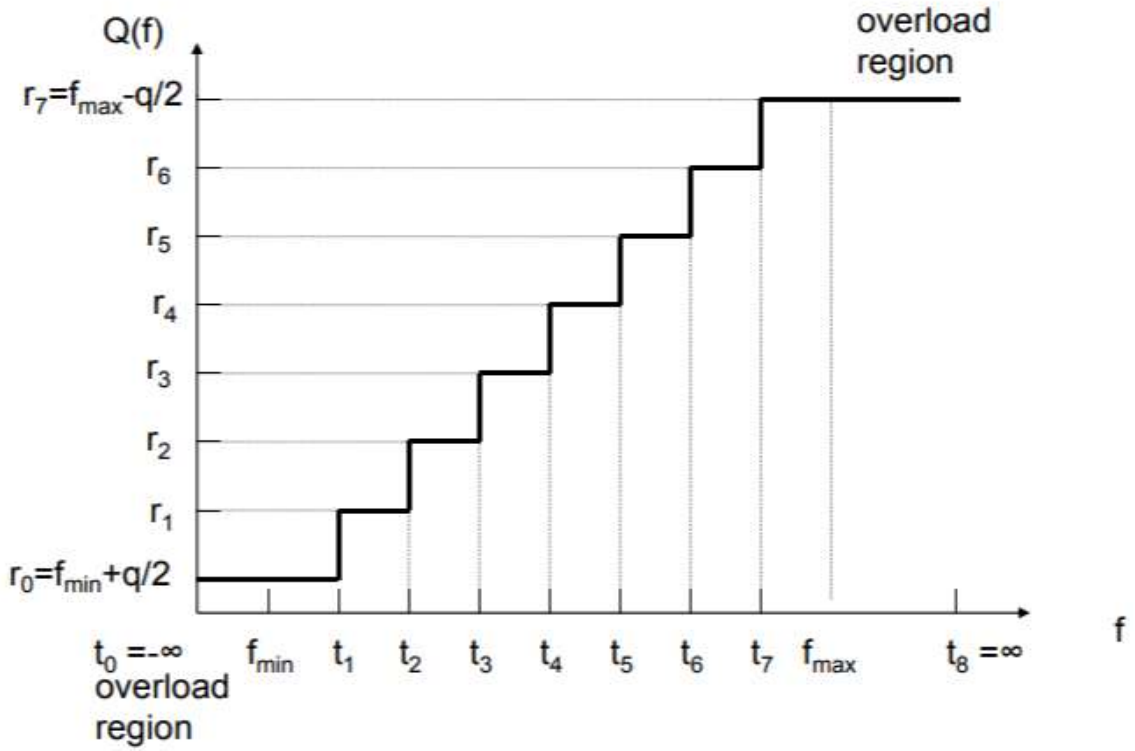
The new value on the ccd camera screen

Truncated Uniform Quantization

- Signal with infinite dynamic range
- Truncate the lower and higher values to f_{min} and f_{max}
- Parameters: f_{min} , f_{max} , L(R)

$$Q(f)=\begin{cases}\left\lfloor \dfrac{f-f_{\min}}{q} \right\rfloor * q+\dfrac{q}{2}+f_{\min} & f_{\min}<f\le f_{\max}\\ f_{\min}+q/2 & f\le f_{\min}\\ f_{\max}-q/2 & f>f_{\max}\end{cases}$$

Truncated Functional representation



Uniform Quantizer – Continuous

- Input signal is continuous
- The output of CMOS sensor is in the range of 0.0 to 5.0 volt.
- L (levels) = 256
- q (quantization intervals=step size = (dynamic range/levels)) = 5.0 / 256
- The output value in the intervals

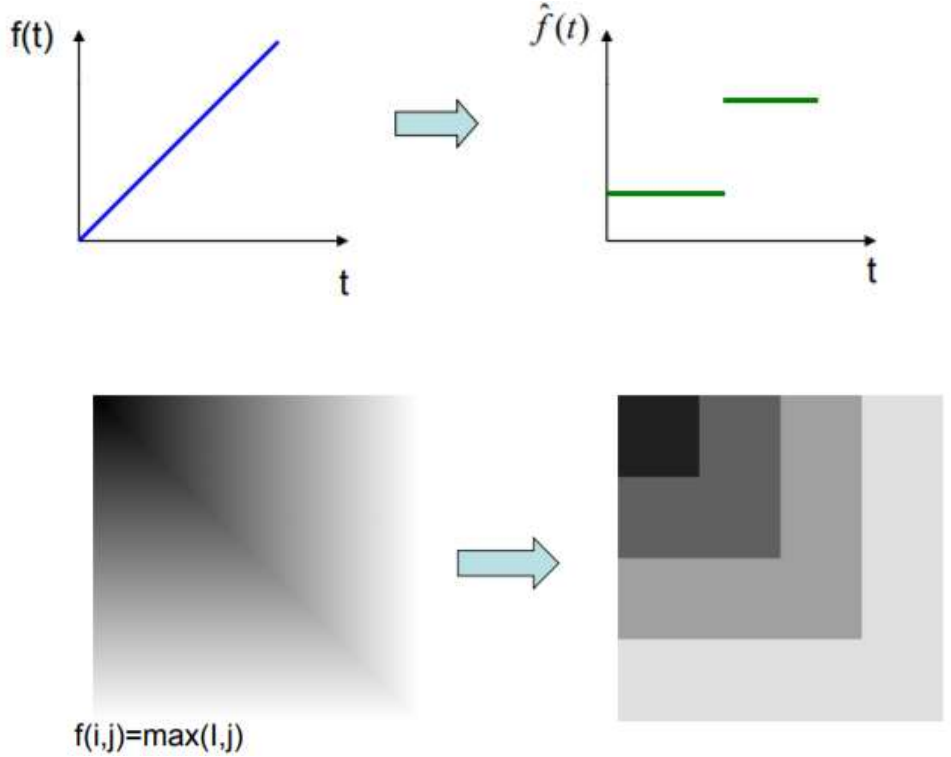
  [l * q - (l + 1) * q] is represented by index l = 0, …, 255.
- The reconstruction level rl = l * q + q/2, l = 0,…, 255.

Uniform Quantization on Images

Quantization Effect – False Contour

1-d signal

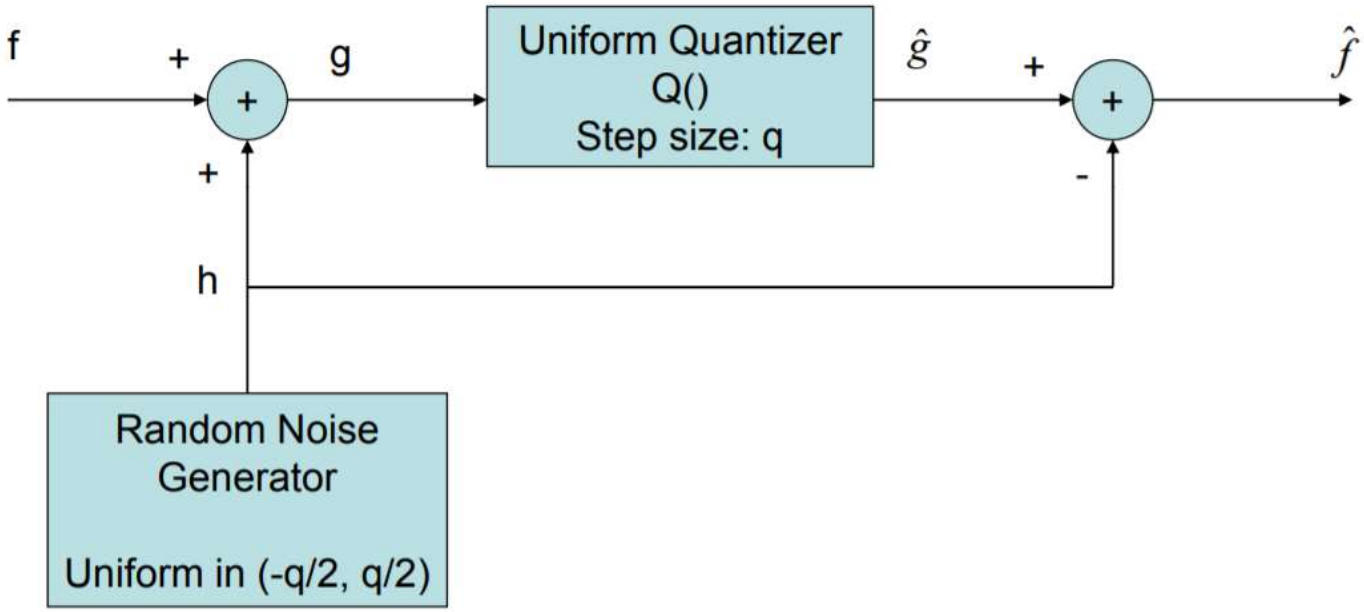
2-d image



Fixing the uniform quantization

- Human eyes are very sensitive to contours (edges)=>
- Uniform quantization does not look so good
- For better visual perception we add random noise.

Pseudo Random Noise Quantizer



Quantization Quality

Ideally we want to measure the performance by how close is the quantized image to the original image

But it is very hard to come up with an objective measure that correlates very well with the perceptual quality that correlates very well with the perceptual quality

Frequently used objective measures:

1. mean square error (MSE) between original and quantized samples

2. signal to noise ratio (SNR)

3. Peak SNR (PSNR)

MSE

- Mean square error (MSE) of a quantizer for a continuous valued signal (Where p(f) is the probability density function)

$$MSE = \sigma_q^2 = E\{(Q(f) - f)^2\} =$$

$$\int_{t_0}^{t_L} (f - Q(f))^2 \, p(f) df = \sum_{l=0}^{L-1} \int_{t_l}^{t_{l+1}} (f - r_l)^2 \, p(f) df$$

- MSE for a specific image

$$MSE = \sigma_q^2 = \frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (f(i,j) - Q(f(i,j)))^2$$

SNR and PSNR

- Signal to Noise Ratio (SNR)

$$\text{Signal Variance} = \text{Signal Energy} = \sigma_f^2 = \frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(f(i,j) - mean)^2$$

$$SNR(dB) = 10\log\frac{\sigma_f^2}{\sigma_q^2}$$

- PSNR-For the error measure to be independent of the signal energy, use the dynamic range square of the image as the signal energy – For 8 bit image, peak=255

$$PSNR(dB) = 10\log\frac{255^2}{\sigma_q^2}$$

Bigger MSE but Better to your eye!

MSE=83.7

8 Level Uniform Quantizer

MSE=173.45

8 Level Pseudo Random Noise Quantizer

Optimal Max-Lloyd Quantizer

Problem:

For a signal x a with given uniform distribution find a quantizer with the constraint that there should be only M classification levels such that:

$$d = MSE = E\left[\left(X - \hat{X}\right)^2\right] \rightarrow \min.$$

$$D = E[(x - Q(x))^2] = \int_{-\infty}^{\infty} (x - Q(x))^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x - y_k)^2 f(x)dx = \sum_{k=1}^{M} d_k.$$

Finding an optimal solution to the above problem results in a quantizer sometimes called a MMSQE (minimum mean-square quantization error) solution

Find the decision boundaries and the reconstruction levels to minimize the resulting distortion.

To solve that the partial derivatives should be “0” .

$$\frac{\partial D}{\partial b_k} = 0 \Rightarrow b_k = \frac{y_k + y_{k+1}}{2} \qquad \frac{\partial D}{\partial y_k} = 0 \Rightarrow y_k = \frac{\int_{b_{k-1}}^{b_k} x f(x) dx}{\int_{b_{k-1}}^{b_k} f(x) dx} = \frac{1}{p_k} \int_{b_{k-1}}^{b_k} x f(x) dx$$

Solution:

Lloyd-Max quantizer [Lloyd,1957] [Max,1960]

- M-1 decision thresholds exactly half-way between representative levels.
- M representative levels in the centroid of the PDF between two successive decision thresholds.

Minima will occur only if all partial derivatives are equal to 0,

$$d = MSE = E\left[\left(X - \hat{X}\right)^2\right] \rightarrow \min.$$

$$t_q = \frac{1}{2}\left(\hat{x}_{q-1} + \hat{x}_q\right) \quad q = 1, 2, \ldots, M\text{-}1$$

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x \cdot f_X(x)dx}{\int_{t_q}^{t_{q+1}} f_X(x)dx} \quad q = 0, 1, \ldots, M\text{-}1$$

*These equations can usually not be solved explicitly; instead a two-step procedure is used repeatedly until a fixed point has been reached:

Iterative Lloyd-Max quantizer design

1. Guess initial set of representative levels (2^B when B is the number of the **representative bytes uniformly 1/(2^B**))

$$\hat{x}_q \quad q = 0, 1, 2, \ldots, M-1$$

2. Calculate decision thresholds (interval boundaries)

$$t_q = \frac{1}{2}\left(\hat{x}_{q-1} + \hat{x}_q\right) \quad q = 1, 2, \ldots, M-1$$

3.

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x \cdot f_X(x)dx}{\int_{t_q}^{t_{q+1}} f_X(x)dx} \quad q = 0, 1, \ldots, M-1$$

4. Repeat 2. and 3. until no further distortion reduction

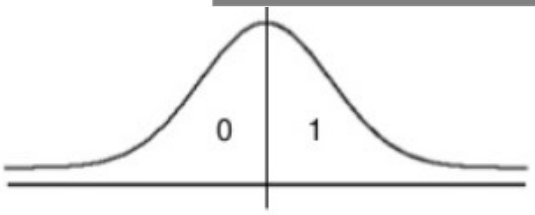
(Iteratively repeat until convergence).

Example – Max-Lloyd Quantizer for Gaussian function

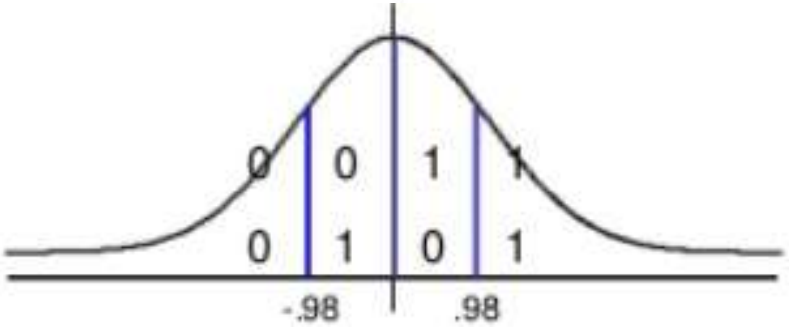
$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x \cdot f_X(x)dx}{\int_{t_q}^{t_{q+1}} f_X(x)dx} \quad q = 0,1,\ldots,M-1$$

$$t_q = \frac{1}{2}\left(\hat{x}_{q-1} + \hat{x}_q\right) \quad q = 1,2,\ldots,M-1$$

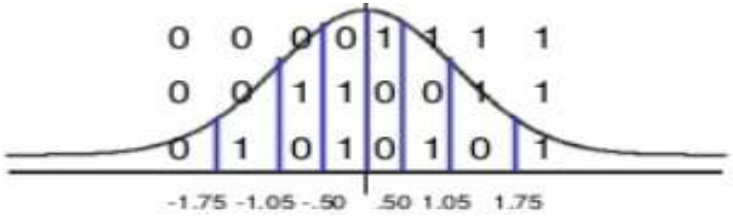
- 1 bit quantizer



- 2 bits quantizer – divides to 4 parts (t0=-0.98, t1=0, t2=0.98)



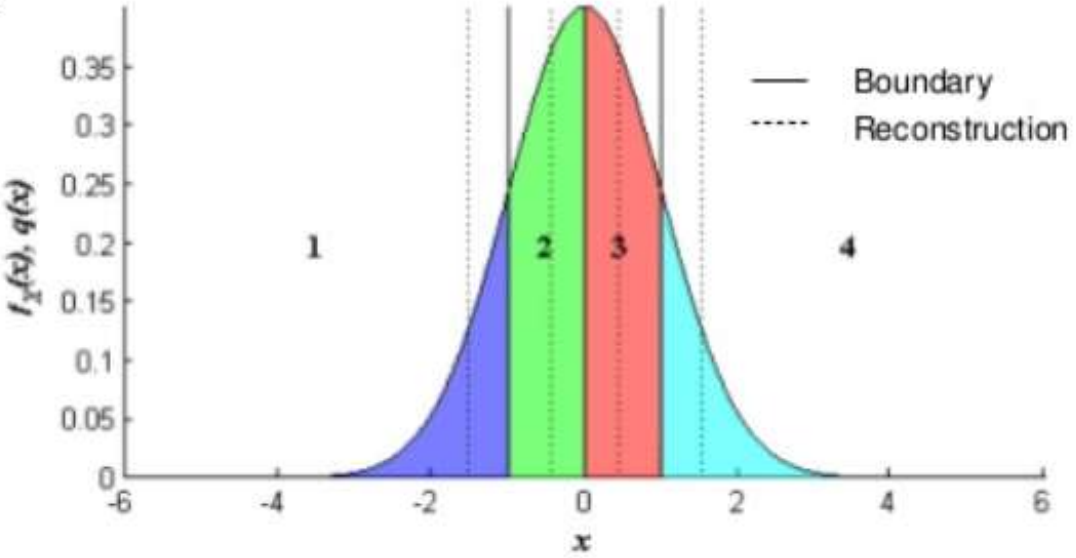
- 3 bits quantizer – divides to 8 parts (t0=-1.75….t7=1.75)…



Gaussian Function – 2 Bit.

Optimum quantizer, obtained with the Lloyd algorithm

- Decision thresholds -0.98, 0, 0.98
- Representative levels –1.51, -0.45, 0.45, 1.51
- D^*=0.12=9.30 dB



In the end we get a quantization “mapping”

Index(3bits/bin)	Range	Representative value
0	0.0000000~0.0343910	0.010867
1	0.0343910~0.0787205	0.057915
2	0.0787205~0.1221875	0.099526
3	0.1221875~0.1702110	0.144849
4	0.1702110~0.2280385	0.195573
5	0.2280385~0.3092675	0.260504
6	0.3092675~0.4440795	0.358031
7	0.4440795~1.0000000	0.530128

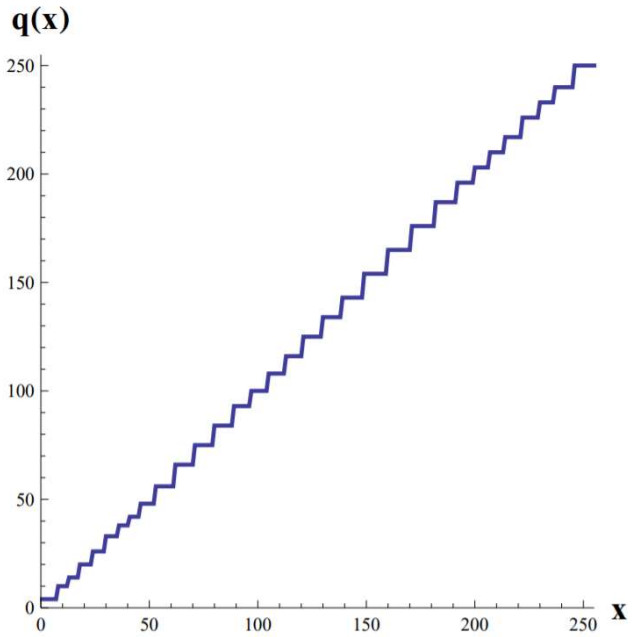
Example

Original image
n = 32

After 21 Iterations…

Quantized image
n = 32

The quantization function in the 2nd run



$$\hat{x}_q \quad q = 0,1,2,\ldots,M-1$$

$$t_q = \frac{1}{2}\left(\hat{x}_{q-1} + \hat{x}_q\right) \quad q = 1,2,\ldots,M-1$$

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x \cdot f_X(x)dx}{\int_{t_q}^{t_{q+1}} f_X(x)dx} \quad q = 0,1,\ldots,M-1$$

Image Color Quantization

- A process that reduces the number of **distinct colors used in an image**, usually with the intention that the new image should be as visually similar as possible to the original image.
- Color quantization is critical for displaying images with **many colors** on devices that **can only display a limited number of colors**, usually due to **memory limitations**, and enables efficient compression of certain types of images.
- Aim: **Reduce the file size of color representation** => Save image **as close as possible to the original.**

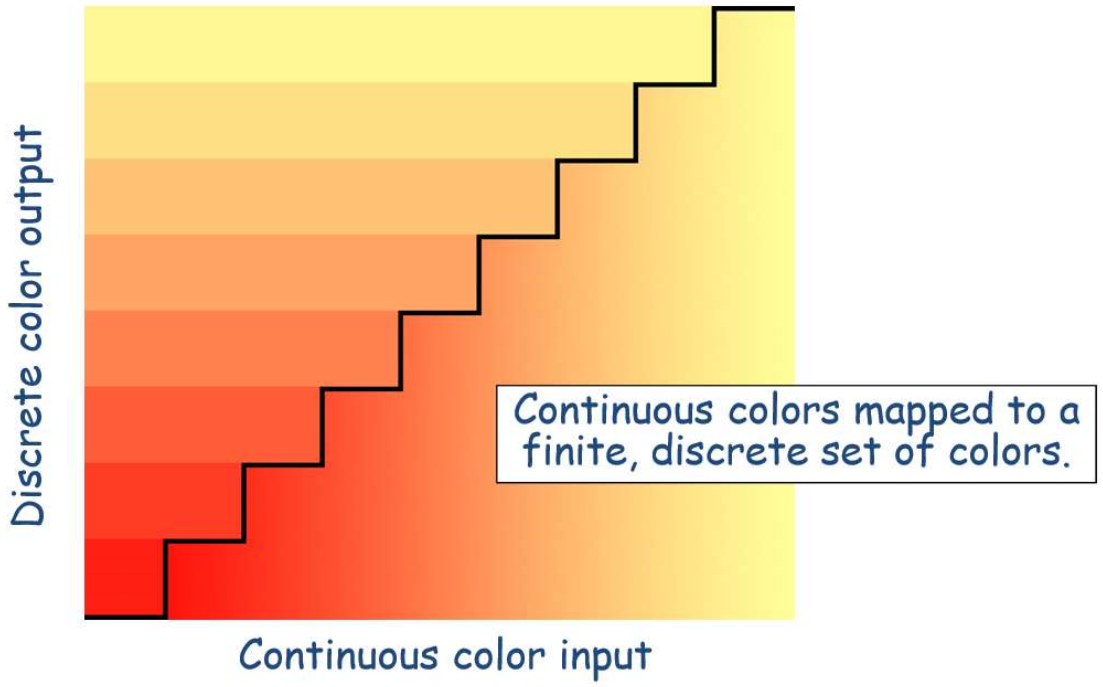
Why do we need it?

- The infinite number of colors available through the lens of a camera is impossible to display on a computer screen.
- Thus converting any photograph to a digital representation necessarily involves some quantization.
- And.. It's not so bad… because 24-bit color is sufficiently rich to represent almost all colors perceivable by humans with sufficiently small error as to be visually identical (if presented faithfully).
- However, there are colors that may be impossible to reproduce, regardless of how many bits are used to represent the color. For example - the full range of green colors that the human eye is capable of perceiving.

Why do we need it?

- Memory limitations=> **limited colors support** (save in pixels or colors)
- Sometimes the **color palette is fixed,** as is often the case in **real-time color quantization systems such as those used in operating systems.**
- Nowadays, color quantization is **mainly used in GIF and PNG images.**
- **GIF**, for a long time the **most popular lossless and animated bitmap** format on the **World Wide Web only supports up to 256 colors.**
- **PNG** images support **24-bit color**, but can often be made much smaller in file size without much visual degradation by application of color quantization, **since PNG files use fewer bits per pixel**

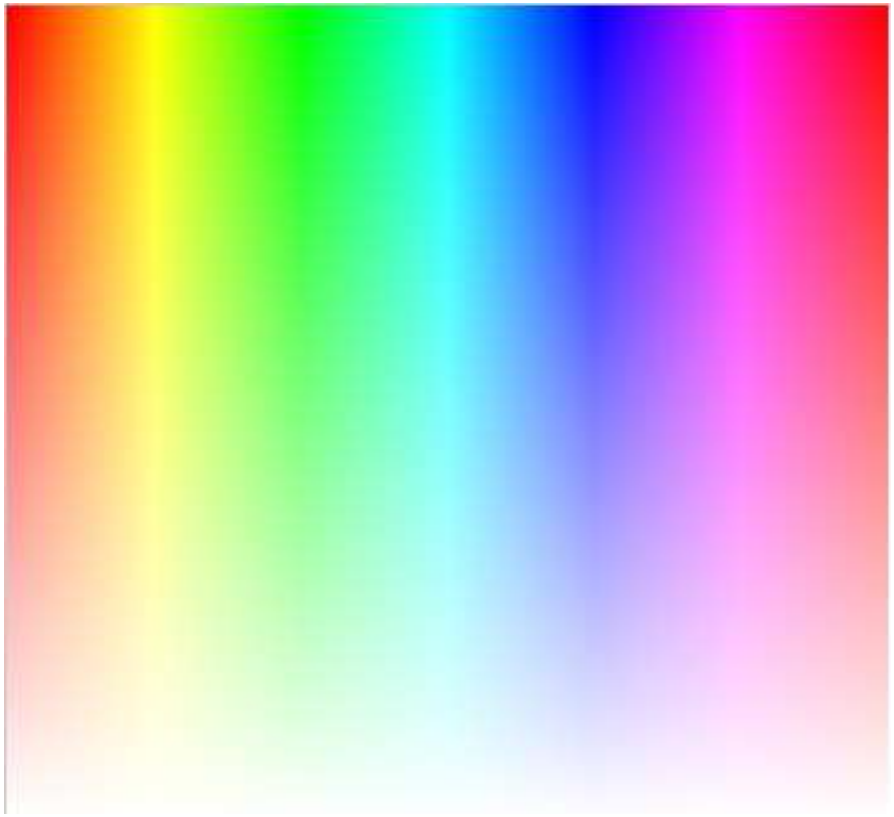
Quantization

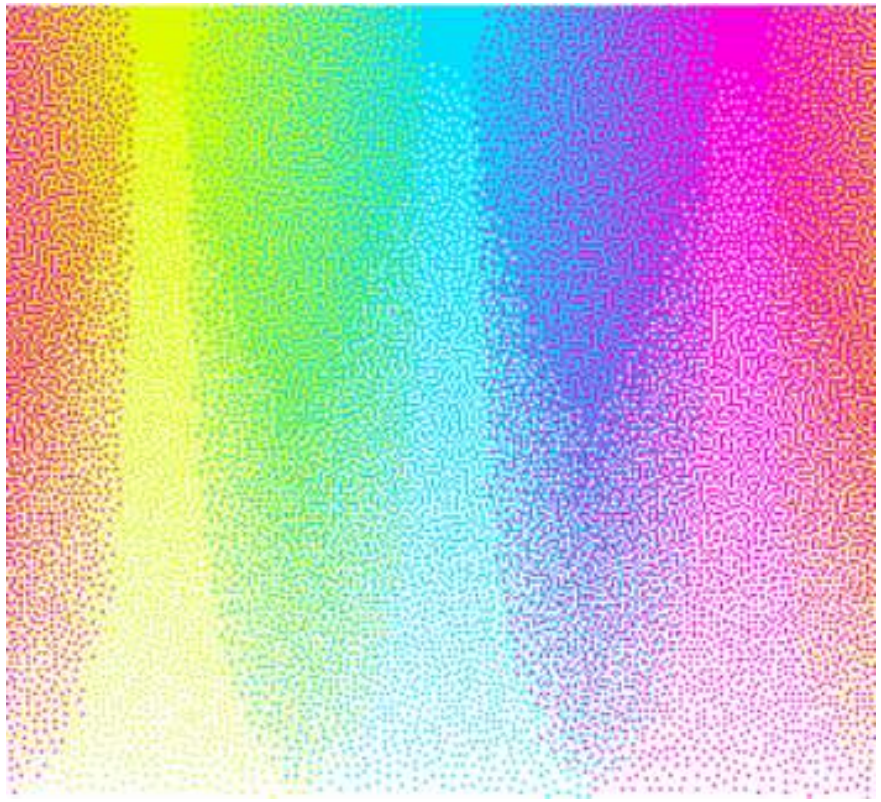


by Ron Leishman

Khawar Khurshid, 2012

Spatial Color Quantization





Uniform Color Quantization (“Adaptive”)

Uniform (scalar quantization)

- **Quantize each color component uniformly**
- E.g. 24 bit-> 8 bit can be realized by using 3 bits (8 levels) for red, 3 bits (8 levels) for green, 2 bits (4 levels) for blue
- **Does not produce good result!**

The Uniform Method

- **Select a set of colors**, save them in **a look-up table** (also known as color map or color palette)
- **Any color is quantized to one of the indexed colors**
- Only needs to save the index as the image pixel value in the display buffer
- Typically: k=8, m=8 (selecting 256 out of 16 million)

Input index (k bits)	Red color (m bits)	Green color (m bits)	Blue color (m bits)
Index 1	…	…	…
…….	…	…	…
Index 2^k	…	…	…

Vector Quantization ("Adaptive")

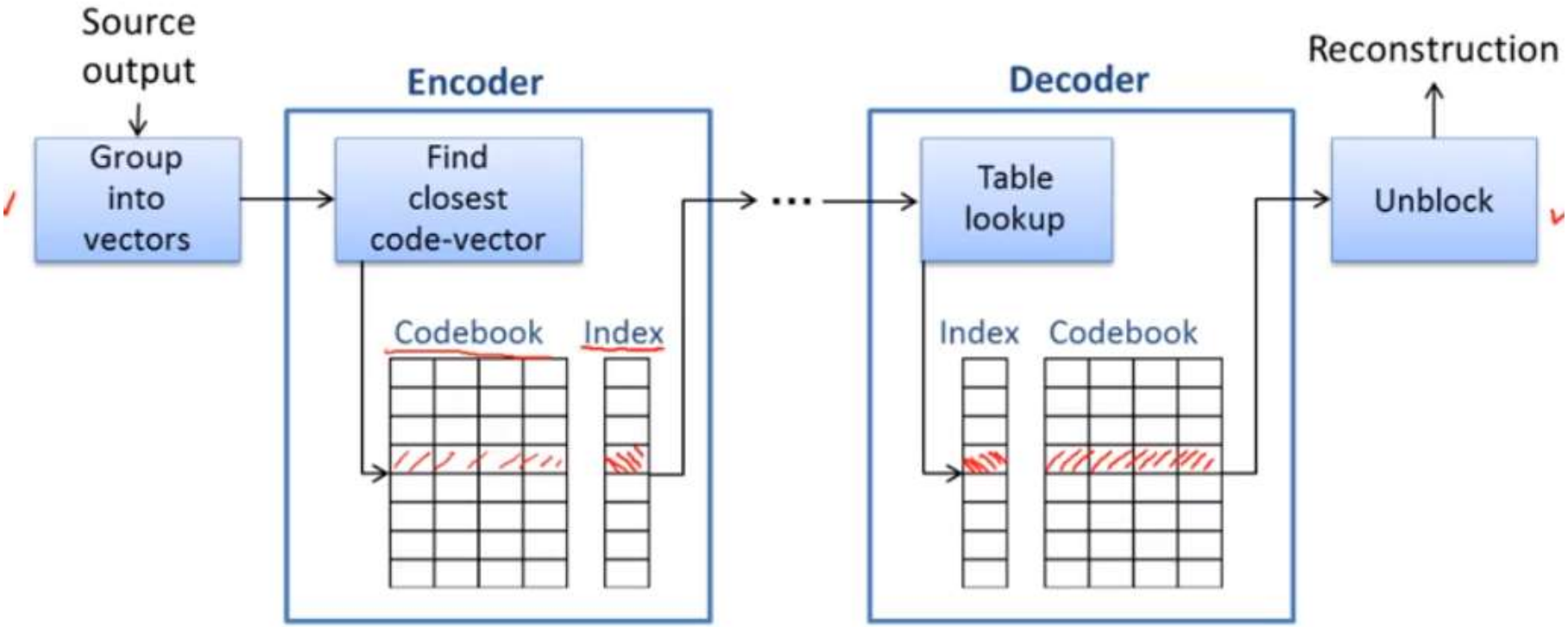
Uniform (scalar quantization)

- **Quantize each color component uniformly**
- **Does not produce good result!**

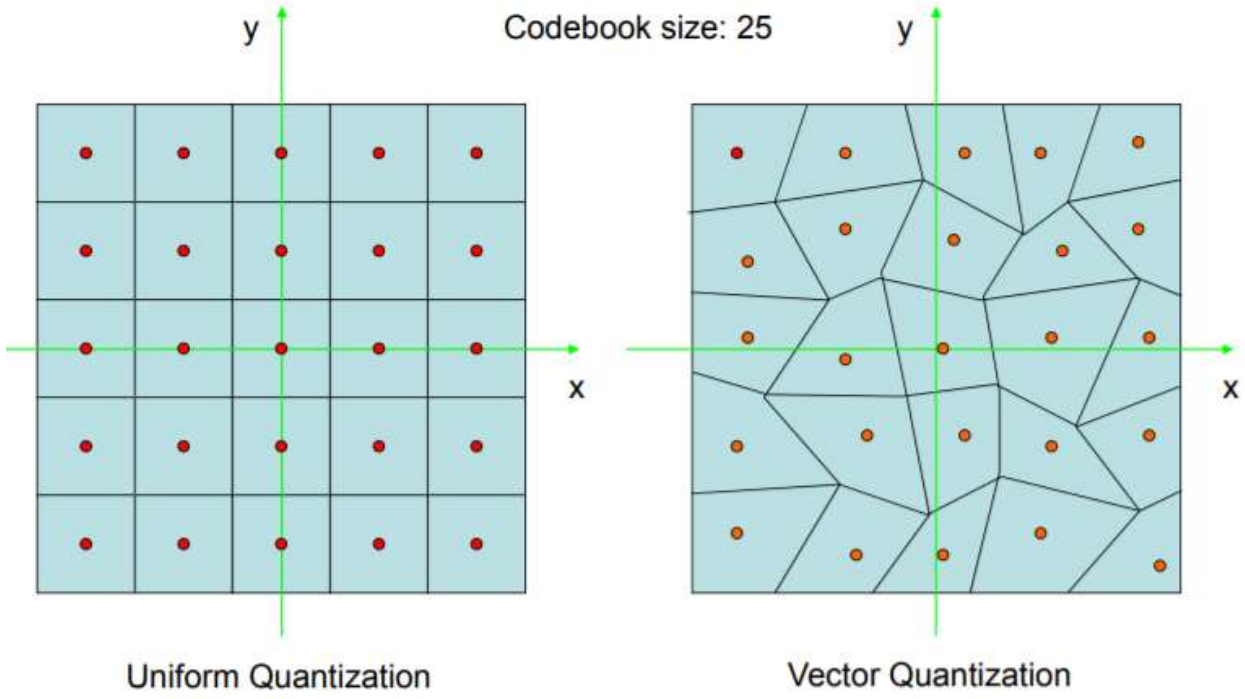
Adaptive (vector quantization)

- **Treat each color (a 3-D vector) as one entity**.
- Find the **N colors (vectors) that appear most often** in a given image, save them in the color palette (codebook).
- **Replace the color at each pixel by the closest color in the codebook** (I.e. color palette)
- **The codebook (I.e. color palette) varies from image to image** -> adaptive

Vector quantization - Scheme



Illustration



24 bits -> 8 bits

Uniform quantization
(3 bits for R,G, 2 bits for B)

Adaptive (non-uniform) quantization
(vector quantization)

How Do We Define the “Distance”?

Quantization is usually done using the "straight-line distance" or "nearest color" algorithm.

That simply takes each color in the original image and finds the closest palette entry.

The Distance is determined by the distance between the two corresponding points in x-dimensional space.

Most standard techniques treat quantization as a problem of **clustering points** in one/three-dimensional space (depends on greyscale/color).

Almost any **three-dimensional clustering algorithm** can be applied to color quantization, and vice versa.

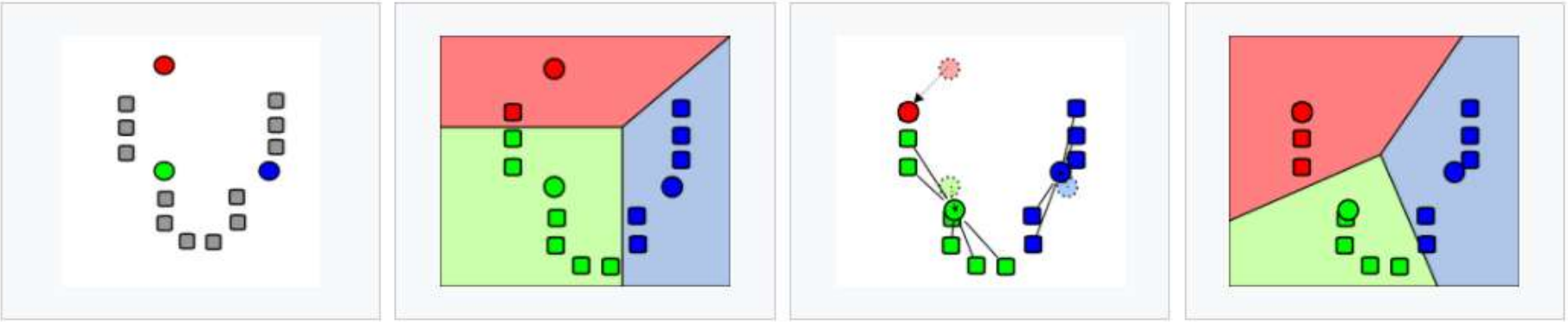
After the clusters are located, typically the points in each cluster are averaged to obtain the representative color that all colors in that cluster are mapped to (For example K-Means)

K-means

***k*-means clustering** is actually a method for vector quantization.

- Originally taken from signal processing, that is popular very popular for cluster analysis and data mining.
- *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
- This results in a partitioning of the data space into cells (like the cells of the Max-Llloyd just this time with a non-uniform distribution)
- This is a private case of the MAX Lloyd Algorithm for non-uniform distribution…

K-means – Reminder



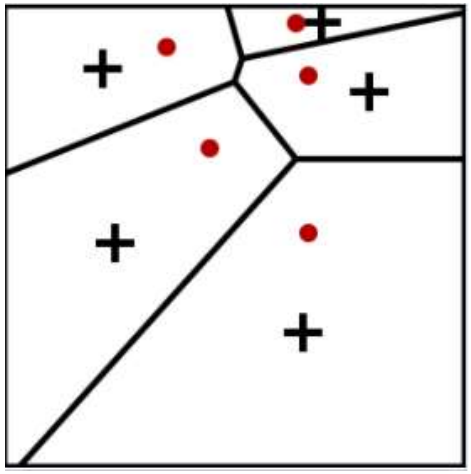
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

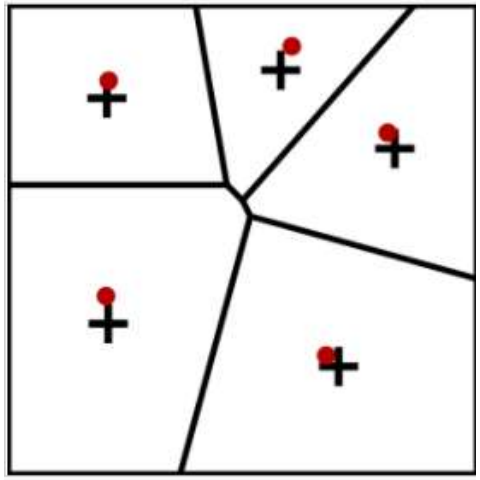
3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

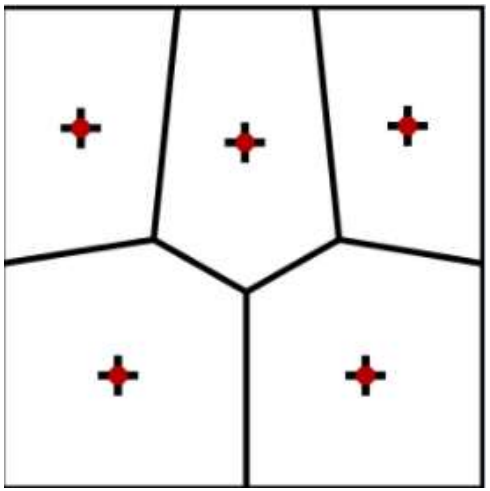
The final result is a color “classifier” in the compressed domain



Iteration 1



Iteration 3



Iteration 15…

In the last iteration the points are very near the centroids of the cells…If the pixel is in the cell it gets the centroid value.

And in other words

Initiation:

1. Choose a number (**K)** of centers (clusters) **that you would like to have**.

2. Choose randomly K points that can be initial **K-means**.

Step:

3. For every point in the image, find the distance from every one of the "**Means**" , and add the point to the "cluster" with the mean that is the closest to it (in that way you will have k "clusters")

4. After all points are in their initial clusters, for every cluster, recalculate the center. Find a new mean of all the points that are currently in that cluster (Add up each pixel in each cluster and divide by the number of pixels in the cluster and we get the new center) => those will be our **new "Means"**

5. Repeat steps 3 and 4 until you get to a convergence (and points does not change their places between clusters anymore / the centroids no longer move).

6. When you get to the convergence, your new points are your image "Codebook" and all points in the "Clusters" will get the value of the "Means" (the final clusters centroids)

So how do we do that with colors?

- So, how do we know how many K-s should there be in our codebook?

  And how do we choose the initial centroids for the K-means?

- One of the proposed way (as we spoken the vector quantization slide) is to find the **K colors (vectors) that appear most often** in a given image , and those will be the initial "centroids". So we can do a histogram of the colors and find the histogram picks (or the number of peaks above some threshold and no neighbors based on some span) and their number will be our K in the K-means and their values will be our initial centroids.
- Another way to do it is choose the K as the number of levels available in the digital format. So if for example the picture is in a Gif format which means - 256 bits, 256 K's , we do a histogram and choose K to be 256, and the most 256 values to be the K initial mean values.
- Another way is just choosing randomly numbers of k-s and make a bunch of runs on all those different k-s and then choose the best run.
- Maybe someday you'll find another way…

The Difference between K-means and K-NN

K-Means – Clustering problem , unsupervised learning (interpret and discover hidden structure in unlabeled data)

Meaning: Having a full set of unlabeled data, try to find the structures in the data by identifying and grouping similar objects together.

Method: For a data of N observations, partition those N observations into K clusters in which each observation belongs to the cluster with the nearest mean.

KNN- Classification problem, supervised learning (the label for the training vector acts as supervision for the classifier).

Meaning: Having a known set of data (training data) determine the classification of a new point that was not in the training set.

Method: For a new point of data for which classification is missing , the k-NN approach looks at the K points that their classification is known in the training set that are **closest** to the test point; the **test point** is then classified according to the class to which the **majority of the K-nearest neighbors belong.** (tell me who are your closest K friends and I'll tall you who you are).

Thank you!