A Report On

STUDY OF QUANTIZATION
AND
IMPLEMENTATION USING K-MEANS CLUSTERING

Submitted by:

1. KEVAL GORADIA 101060077

2. AJAY GORE 101060050

3. NIREVDH MESHRAM 101060001

4. RAHUL RAGHATATE 101060019

Digital Signal Processing Lab

Department of Electrical Engineering

VJTI

May 2013

ABSTRACT:-

Quantization is a lossy technique for data compression used in various applications. Color quantization reduces the number of colors used in an image & proves to be very useful for displaying images on devices that support only a limited amount of colors. It also reduces the file size of an image while transmitting, thus making it useful during transmission. For vector quantization to be efficient a good codebook is required. For that purpose we use the lbg-algorithm as a powerful & efficient tool to compress a color image to reduce its size. This method generates codewords. With this method, a significant time reduction can be achieved in transmitting the data.

INTRODUCTION

The fundamental goal of image compression is to reduce the bit rate for transmission or data storage while maintaining an acceptable image quality. The Vector Quantization (VQ) scheme can be divided into three parts: the codebook generation process, the encoding procedure and the decoding procedure. The codebook generated is required in both the encoding procedure and the decoding procedure in VQ scheme. In the process of vector quantization, the image to be encoded is segmented into a set of input image vectors. In the encoding procedure, the closest codeword for each input vector is chosen, and its index is transmitted to the receiver. In the decoding procedure, a simple table look-up procedure is done to reconstruct the encoded image in the receiver. Thus, the encoded image of the original input image becomes available to the receiver. The whole compression process is accomplished when the encoded image is reconstructed with the corresponding index of each input image vector.

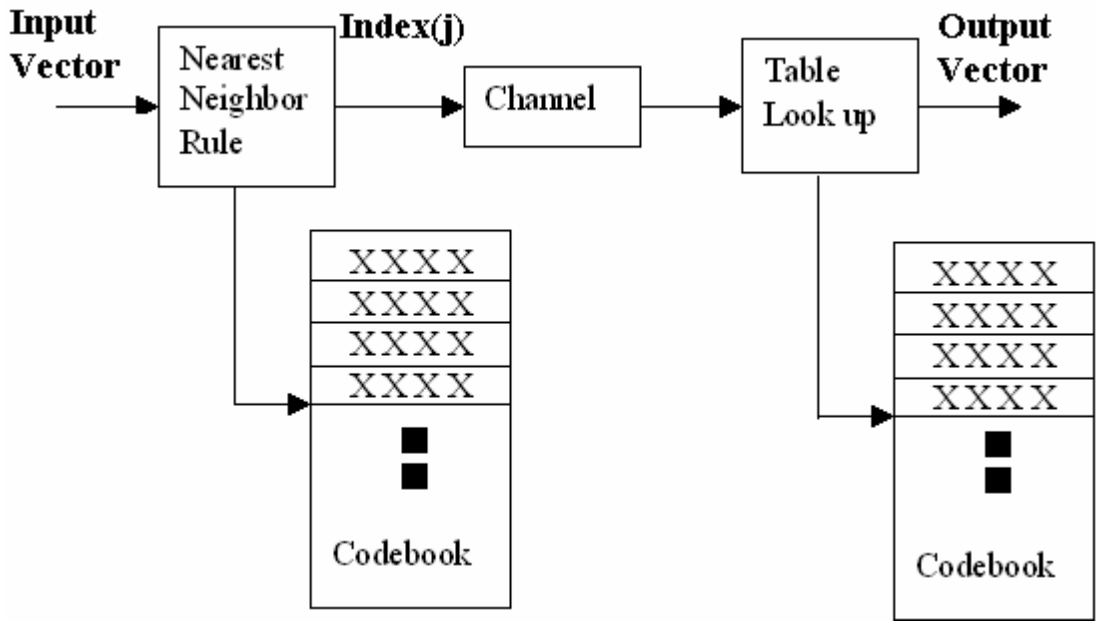
The most important task for the VQ scheme is to design a good codebook**.** A good codebook is required because the reconstructed image highly depends on the codewords in this very codebook. The generalized Lloyd clustering algorithm referred to as the LBG algorithm proposed by Linde, Buzo, and Gray is the most popular method for codebook training. The LBG algorithm is an iterative procedure. Starting with an initial codebook, the LBG algorithm performs the segmentation of the training set, where an exhaustive search procedure is conducted to classify each vector in the training set according to its corresponding closest codeword in current codebook. The closest codeword of one training vector is the codeword with the smallest squared Euclidean distance in the codebook. Then each training vector records the index of its closest codeword in the codebook. After completing the segmentation process for the whole training set, each codeword in the current codebook is updated with the centroid of those training vectors so that the current codeword is the closest codeword of them. The newly generated codebook is then used in the next iteration to minimize the overall averaged distortion in the codebook design procedure. Both the segmentation and the updating procedures are executed repeatedly until the overall averaged distortion between the last two successive iterations is not significantly noticeable or a certain number of iterations are reached.

IMPLEMENTATION:

Platform: MATLAB R2010a

The implementation aims to achieve an image with lesser number of component levels then the original image which has 256 levels for each three components (Red, Green and Blue) of each pixel.

This is demonstrated in the block diagram below



[01]

The process consist of three steps

1. Generation of Code book
2. Encoding and transmitting across the channel
3. Decoding at the receiver end.

Each step is as follows

1. Generation of Code book

Initially a codebook is taken at random and number of code words in the codebook is equal to the number of quantized levels to be obtained.

In the implemented Algorithm these random code words are chosen equally spaced as shown below.

For having an initial codebook of Size N we chose the initial code words as

code word[i]=[(256/N)*i]-1 i= 1 to N

Now each component level of each pixel is assigned to its nearest codebook. This is done by using the distance formula. The algorithm used is called **K-means clustering.**

This algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x(i,j) - c(j)\|^2$$

,

where $\|x(i,j) - c(j)\|^2$ is a chosen distance measure between a data point $x(i,j)$ and the cluster centre $c(j)$, is an indicator of the distance of the *n* data points from their respective cluster centers'.[2]

In the implemented algorithm $x(i,j)$ is a component level of a given pixel and $c(j)$[2] The algorithm follows the following steps

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which quantized levels can be obtained.

This completes the codebook generation part.

2. Encoding of the Image

In encoding section, the implementation again associate the component levels of each pixel to its closest code word. Then we make a matrix containing the indexes of the code word associated to each component level of each matrix.

Then we transmit the encoded matrix which will require lesser bits per pixel per component to transmit than it would have taken for the original image.

3. Decoding of the image

In decoding section, the receiver receives the encoded matrix from the encoder and it also receives the codebook whi6ch it uses as a look up table.

The decoding section then assigns t6he value to each encoded level from its look up table. Thus forming the new image matrix with only as many levels as were desired.

Error Calculations:

In this section, the distortion between the reconstructed image and the actual image is calculated. For each component of each pixel of both the images, the square error is calculated and plotted.

The root mean square error for each component is calculated by taking sum of the square errors and dividing it by the total number of components and then taking the square root.

RESULTS

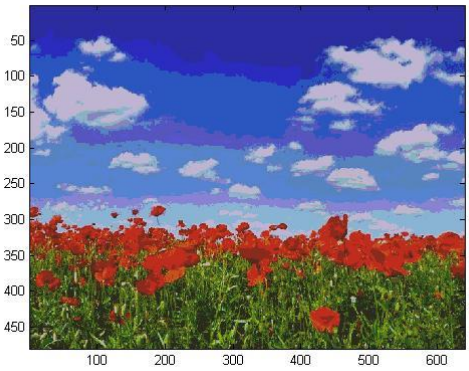
Original image:-

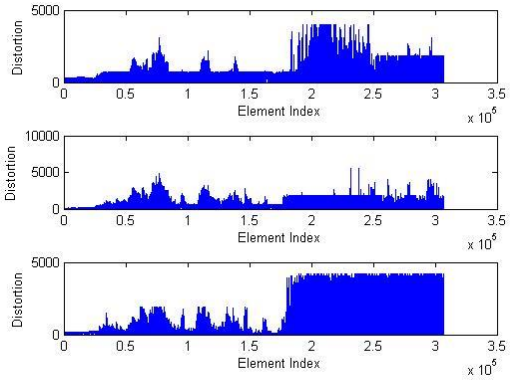
A. Effect of Quantization Levels:-

Enter quantization levels for each color 4

Enter iterations 20

0.0256 0.0270 0.0285

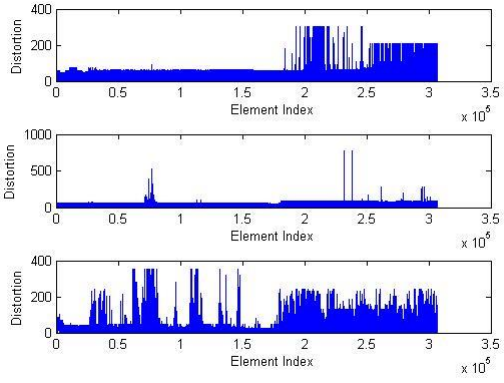




Enter quantization levels for each color 16

Enter iterations 20

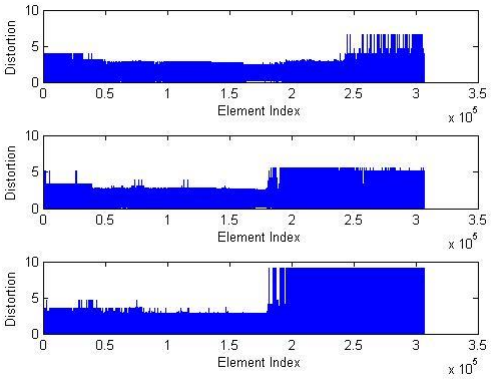
0.0079 0.0079 0.0067



Enter quantization levels for each color 64

Enter iterations 20

0.0020 0.0020 0.0021

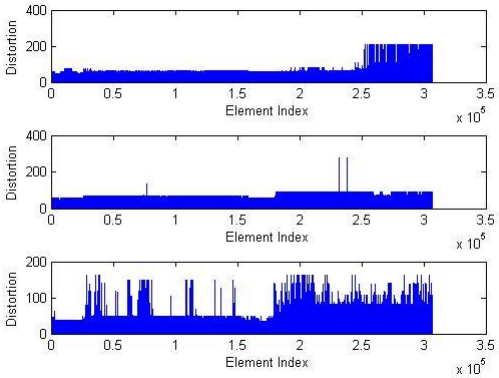


B. Effect of number of iterations

Enter quantization levels for each color 16

Enter iterations 5

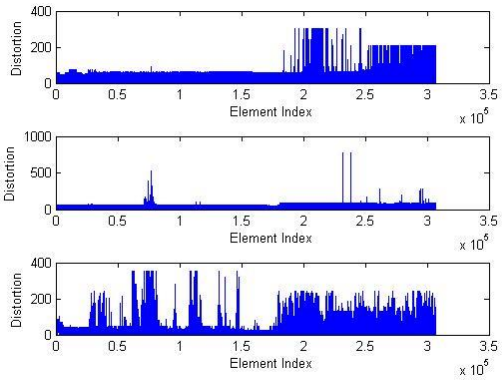
0.0080 0.0082 0.0076



Enter quantization levels for each color 16

Enter iterations 20

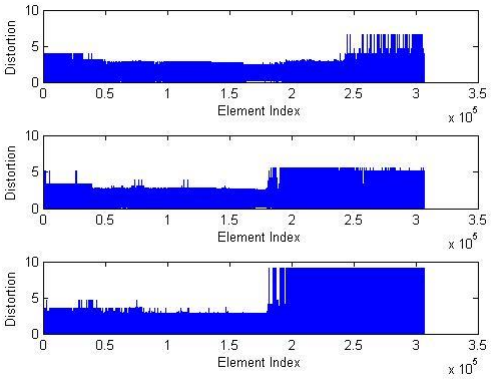
0.0079 0.0079 0.0067



Enter quantization levels for each color 16

Enter iterations 50

0.0079 0.0079 0.0066



ANALYSIS

- As quantization levels are increased, the root mean square error in all three matrices decreases considerably. Consider following example, (Number of iterations = 20)

Quantization levels	RMS error in red matrix
4	0.0256
16	0.0079
64	0.0020

- As number of iterations is increased, the root mean square error in all three matrices decreases. Consider following example, (Quantization level = 16)

Number of iterations	RMS error in blue matrix
5	0.0076
20	0.0067
50	0.0066

- The above two examples show that increasing quantization levels reduce the error. But after some value, increasing number of iterations does not reduce error much.

LIMITATIONS

- It suffers from the serious drawback that its performance depends heavily on the initial conditions.
- This method is unsuitable for real world data that usually constitute a large data set and consequently require numerous codewords.

SUMMARY

The Linde-Buzo-Gray algorithm for vector quantization is implemented on Matlab. The initial codebook is generated randomly. Each pixel is associated with a value from codebook. The new codeword is generated according to the pixels associated with previous codeword. The results are obtained and effect of quantization levels and number of iterations is studied.

REFERENCES

[01] IMAGE COMPRESSION USING VECTOR QUANTIZATION,

[02]http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html