Università degli Studi di Padova - Department of Information Engineering

Source coding course

Final Project

Implementation of LBG algorithm

Marco Centenaro
ID number: 1043634

June 11, 2014

Abstract

The goal of source coding is that of finding efficient ways of representing information for both efficient transmission and storage of data. This can be done employing algorithms whose aim is that of compressing the original data.
In this report I propose:

1. a quick introduction about the source coding approaches,
2. a review of the LBG algorithm,
3. an evaluation of the performances of this compressing technique,
4. a discussion about the results.

1 Introduction

The compression techniques are generally divided into

1. *lossless techniques*: they identify and eliminate the statistical redundancy of the source. These kind of techniques are invertible, can be applied only to digital sources and exploit variable-length codewords. Their target is that of compressing the original information *as much as possible*.

2. *lossy technique*: they identify and eliminate unnecessary information which is present in the source. These kind of techniques are not invertible, i.e. they introduce some kind of distortion of the original information, and they are usually required for analog sources; they provide a much higher compression with respect to lossless coding. In the case of lossy coding we have two possible targets: to provide either the *maximum fidelity*, finding the minimum possible distortion for a given bitrate, or the *maximum compression*, finding the minimum possible bitrate for a given maximum distortion constraint.

In particular, lossy coding in practise consists of a *quantization* procedure. A further distinction can be done between the following two quantization processes

1. *scalar quantization*: each input sample is quantized independently on the other ones;

2. *vector quantization*: a block of input samples are processed, yielding to a joint quantization procedure.

One of the most important algorithms for what concerns the vector quantization is the *Linde-Buzo-Gray algorithm* (LBG), which was introduced in 1980. In this report the LBG algorithm will be implemented and its performances will be tested.

2 Technical approach

2.1 Objectives

The objective of this section is that of providing a brief description of vector quantization and explaining the procedure of LBG algorithm.

2.2 Reference scenario: vector quantization

In scalar quantization we compress one sample at a time; in vector quantization, instead, we quantize sets of samples, i.e. vectors. This strategy is theoretically optimal as the size of the vectors grows to infinity, but obviously it becomes unfeasible in practise if we consider vectors which are too long.

Let's call L the dimension of the vectors we are going to consider; we define the generic input vectors as

$$\mathbf{x} = [x_1, \ldots, x_L] \ , \ \mathbf{x} \in \mathbb{R}^L$$

where $x_i, \ i = 1, \ldots, L$ are input samples. The *codebook*, i.e. the set of reconstruction levels in the case of vector quantization, is defined as

$$\mathcal{B} = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$$

where $\mathbf{y}_i$ are the *codevectors*; K is the size of the codebook. Finally we define the decision cells as

$$I_i \subseteq \mathbb{R}^L, \ i = 1, \ldots, K, \text{ such that } I_i \cap I_j = \emptyset \ \forall \ i \neq j \text{ and } \bigcup_{i=1}^{K} I_i = \mathbb{R}^L$$

We define the *bitrate* $\mathcal{R}$ as the number of bits that are used to quantize each single component of the codevector, i.e.

$$\mathcal{R} \triangleq \frac{1}{L} \lceil \log_2 K \rceil$$

The target of the quantization procedure is that of minimizing the distortion D, which is defined as follows:

$$\begin{aligned} D &\triangleq \mathbb{E}\left[\|\mathbf{x} - Q(\mathbf{x})\|^2 \right] = \\ &= \int_{\mathbb{R}^L} \|\mathbf{x} - Q(\mathbf{x})\|_2^2 \ f_{\mathbf{X}}(\mathbf{x}) \ d\mathbf{x} = \\ &= \sum_{i=1}^{K} \int_{I_i} \|\mathbf{x} - \mathbf{y}_i\|_2^2 \ f_{\mathbf{X}}(\mathbf{x}) \ d\mathbf{x} \end{aligned}$$

where $Q(\mathbf{x}) = \mathbf{y}_i$ is the map that associates the input vector $\mathbf{x} \in \mathbb{R}^L$ to the reconstruction level $\mathbf{y}_i \in \mathcal{B}$ if and only if $\mathbf{x} \in I_i \subseteq \mathbb{R}^L$.

The necessary conditions for the optimality are

Nearest neighbour condition

$$I_i = \{\mathbf{x} \in \mathbb{R}^L \text{ such that } \|\mathbf{x} - \mathbf{y}_i\|_2^2 \leq \|\mathbf{x} - \mathbf{y}_j\|_2^2,\ j \neq i\},\ i = 1, \ldots, K$$

Centroid condition

$$\mathbf{y}_i = \frac{\int_{I_i} \mathbf{x}\ f_{\mathbf{X}}(\mathbf{x})\ d\mathbf{x}}{\int_{I_i} f_{\mathbf{X}}(\mathbf{x})\ d\mathbf{x}} = \int_{I_i} \mathbf{x}\ f_{\mathbf{X}|\mathbf{X}\in I_i}(\mathbf{x}|\mathbf{x} \in I_i)\ d\mathbf{x},\ i = 1, \ldots, K$$

2.3 The LBG algorithm

This algorithm was designed by Linde, Buzo and Gray in the 1980. Assume that we know the probability distribution function of the input data $f_{\mathbf{X}}(\mathbf{x})$; then we should

1. start from an initial codebook
$$\left\{\mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_K^{(0)}\right\}$$
set $n \leftarrow 1$, $D^{(0)} \leftarrow \infty$ and select $\epsilon > 0$.

2. compute the optimal partition of $\mathbb{R}^L$ as
$$I_i^{(n)} \leftarrow \{\mathbf{x} \in \mathbb{R}^L \text{ such that } \|\mathbf{x} - \mathbf{y}_i^{(n-1)}\|_2^2 \leq \|\mathbf{x} - \mathbf{y}_j^{(n-1)}\|_2^2,\ j \neq i\},\ i = 1, \ldots, K$$

3. compute the new codebook
$$\mathbf{y}_i^{(n)} \leftarrow \int_{I_i^{(n)}} \mathbf{x}\ f_{\mathbf{X}|\mathbf{X}\in I_i^{(n)}}\left(\mathbf{x}|\mathbf{x} \in I_i^{(n)}\right)\ d\mathbf{x},\ i = 1, \ldots, K$$

4. evaluate the distortion
$$D^{(n)} \leftarrow \sum_{i=1}^{K} \int_{I_i^{(n)}} \|\mathbf{x} - \mathbf{y}_i^{(n)}\|_2^2\ f_{\mathbf{X}}(\mathbf{x})\ d\mathbf{x}$$

5. check the termination condition
$$\frac{D^{(n-1)} - D^{(n)}}{D^{(n)}} < \epsilon$$
If it is satisfied then stop the iteration else set $n \leftarrow n + 1$ and go to step 2.

Note that the distortion D may not be globally minimized by this procedure, however it is guaranteed that it will never increase among consecutive iterations. The drawbacks of this approach are that

- multivariate integrals need to be computed,
- the knowledge of $f_{\mathbf{X}}(\mathbf{x})$ is required.

If we have no way of knowing $f_{\mathbf{X}}(\mathbf{x})$, then we need to extract this information straight from the empirical data. We build a *training set*

$$\mathcal{T} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

which consists of $N > 1000 \cdot K$ vectors. Then we should

1. start from an initial codebook
$$\left\{\mathbf{y}_1^{(0)}, \dots, \mathbf{y}_K^{(0)}\right\}$$
set $n \leftarrow 1$, $D^{(0)} \leftarrow \infty$ and select $\epsilon > 0$.

2. compute the optimal partition of $\mathbb{R}^L$ as
$$I_i^{(n)} \leftarrow \{\mathbf{x} \in \mathcal{T} \text{ such that } \|\mathbf{x} - \mathbf{y}_i^{(n-1)}\|_2^2 \leq \|\mathbf{x} - \mathbf{y}_j^{(n-1)}\|_2^2,\ j \neq i\},\ i = 1, \dots, K$$

3. compute the new codebook
$$\mathbf{y}_i^{(n)} \leftarrow \frac{1}{|I_i^{(n)}|} \sum_{\mathbf{x} \in I_i^{(n)}} \mathbf{x},\ i = 1, \dots, K$$

4. evaluate the distortion
$$D^{(n)} \leftarrow \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x} \in \mathcal{T}} \|\mathbf{x} - Q(\mathbf{x})\|_2^2$$

5. check the termination condition
$$\frac{D^{(n-1)} - D^{(n)}}{D^{(n)}} < \epsilon$$
If it is satisfied then stop the iteration else set $n \leftarrow n + 1$ and go to step 2.

Codebook initialization The initialization of the codebook is of fundamental importance. Several techniques have been designed for this purpose:

Random coding

Build the codebook $\mathcal{B}$ by just taking K random samples in the training set $\mathcal{T}$.

Pruning

Draw the first sample of $\mathcal{B}$ randomly in $\mathcal{T}$, then pick the vector which is the furthest from it and iterate the procedure, considering step by step the furthest vector from those ones which have already been selected.

Splitting

Take as first member $\mathbf{y}_1$ of the codebook the average of the vectors in $\mathcal{T}$; then split it in two different vectors

$$(1 + \epsilon) \cdot \mathbf{y}_1\ ,\ (1 - \epsilon) \cdot \mathbf{y}_1$$

and apply LBG algorithm to the new codebook that consists of these two vectors. Then iterate the procedure until the desired codebook size K is reached (it will be a power of two for sure).

Product codes

Consider the components of the vectors separately and perform the scalar quantization of each single component.

Pairwise nearest neighbour

Start considering all the pairs of the nearest vectors in $\mathcal{T}$; then the pairs of the nearest pairs and so on. Proceding in this way, grouping points together, the area is divided into blocks and we take as the representative vectors the centroids of the blocks.

The empty cell problem In the second step of the algorithm based on empirical data it is assumed that the cells are all non-empty. If a cell is empty, then we just fix it, substituting the representative of that decision cell with a vector that belongs to the cell that has the highest cardinality.

2.4 Matlab implementation

The programme consists of seven scripts:

- `LBG.m`. This function contains the instructions to implement the LBG algorithm based on a training set which was presented in the previous section. The input parameters are the training set, the initial codebook and the distortion threhsold ϵ; the output consists of the optimized codebook. First of all, every vector in the training set is assigned to the appropriate decision cell through the column array `addressing`, of size N, and the cardinalities of the cells are recorded in a column vector of counters called `cells_cardinality`, of size K. Then the new codebook is computed according to the formulas shown in the previous section, updating the decision cells and the codevectors; if there exist some cells whose cardinalities are zero, their representatives are chosen between the vectors of the most populated cell. Finally the average distortion is computed and the termination condition is checked. A further termination condition is present upon the maximum number of cycles (set to 20), in order to make the algorithm always converge.

- `split.m`. This function performs the splitting of the vectors in the codebook, before passing the new codebook to the LBG routine that has to optimize it.

- `Q.m`. This routine implements the quantization of a certain input vector, assigning it to the appropriate decision cell and returning the relative representative vector of the codebook.

- `extract_vectors_from_image.m`. This is the function which performs the segmentation of the original image (a matrix) into a set of row vectors of length L.

- `insert_vectors_in_the_image.m`. This routine performs the inverse transformation of the previous one, reconstructing an image from a set of vectors.

- `encode.m`. This function just encapsulates all previous functions to provide a unique function call to perform the encoding process of a desired image.
- `main.m`. This is the script that has to be run in order to test the LBG algorithm and, of course, contains all other methods.

2.5 Complications found

Fixing all indices in the vectors was really a great battle.

3 Results

Two different scenarios were tested. In the first one it is

$$L = 2 \times 2 = 4 \ , \ \mathcal{R} = 2 \text{ bit/pixel}$$

$$\mathcal{R} = \frac{\log_2(K)}{L} = 2 \Rightarrow K = 2^{\mathcal{R}L} = 256$$

whereas in the second one

$$L = 4 \times 4 = 16 \ , \ \mathcal{R} = 0.5 \text{ bit/pixel}$$

$$\mathcal{R} = \frac{\log_2(K)}{L} = 2 \Rightarrow K = 2^{\mathcal{R}L} = 256$$

so in both these situations the size of the codebook is the same. The training set that was employed in the simulations consists of images picked from `http://www.cipr.rpi.edu/resource/stills/`; they are shown in Figure 1. The sizes of the training sets $\mathcal{T}$ are

1. $N = 688128 = 2688 \cdot K$ for $L = 4$
2. $N = 620544 = 2424 \cdot K$ for $L = 16$.

so they are almost of the same dimension.

In the next sections the following topics will be investigated:

1. what are the performances of the LBG quantizer with $L = 4$ and $L = 16$ for various values of the threshold ϵ if the input consists of an image of the training set? What if the image is not part of the training set?
2. what are the performances of the LBG when the codebook is initialized through random initialization instead of LBG splitting?

Important note The caption under each picture in the following sections contains the threshold value ϵ which was used to compute the codebook that was employed in the encoding procedure and the distortion D which was introduced by the quantization.

3.1 Parameters of the codebooks

The parameters of each codebook that has been computed are shown in Table 1.

Figure 1: These are all the 35 images that were used to build the training set for the LBG algorithm.

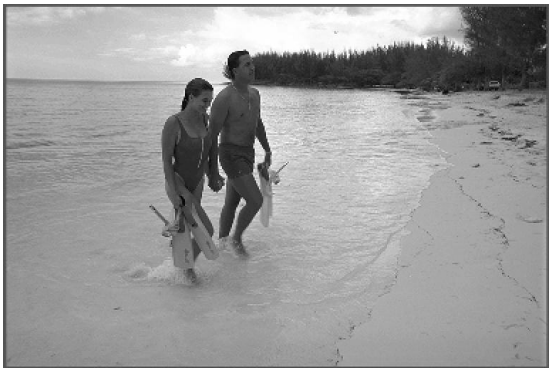
Table 1: Codebooks parameters.

L	ϵ	K	N	N/K	Initialization	D	Initialization	D
4	0.1	256	688128	2688	split	168.5	random	179.3
4	0.05	256	688128	2688	split	171.1	random	177.1
4	0.01	256	688128	2688	split	165.8	random	162.8
4	0.005	256	688128	2688	split	162.7	random	159.8
4	0.001	256	688128	2688	split	157.8	random	156.9
16	0.1	256	620544	2424	split	1783.8	random	1907.3
16	0.05	256	620544	2424	split	1826.3	random	1862.7
16	0.01	256	620544	2424	split	1800.0	random	1746.7
16	0.005	256	620544	2424	split	1774.0	random	1734.1
16	0.001	256	620544	2424	split	1722.4	random	1702.4

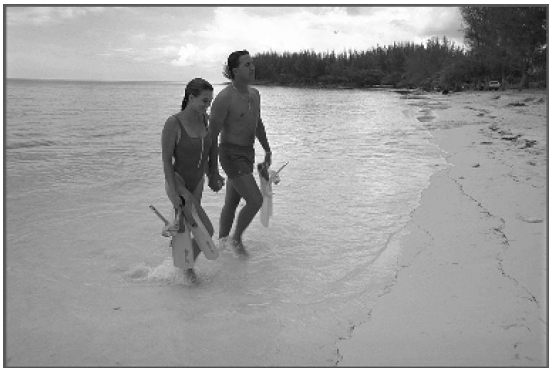
3.2 Encoding an image of the training set

(a) Original image, $D = 0$

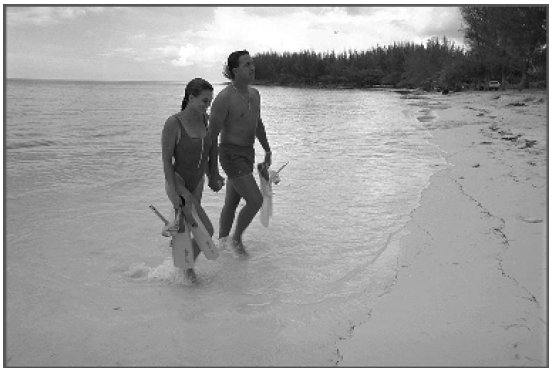
(b) $\epsilon = 0.001$, $D \simeq 406.2$



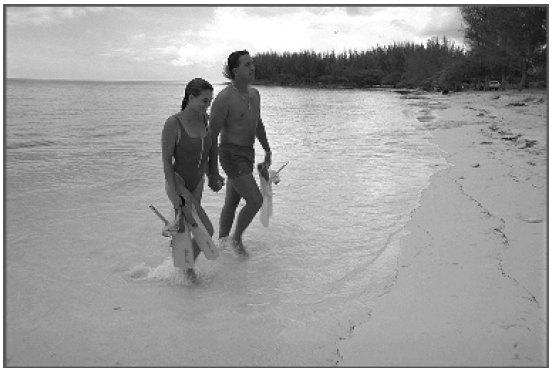
(c) $\epsilon = 0.005$, $D \simeq 404.4$



(d) $\epsilon = 0.01$, $D \simeq 407.6$



(e) $\epsilon = 0.05$, $D \simeq 405.4$



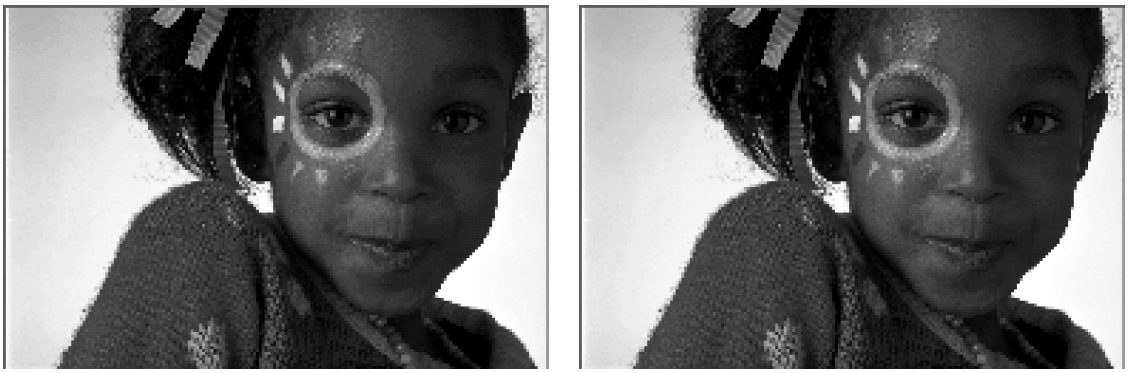
(f) $\epsilon = 0.1$, $D \simeq 406.6$

Figure 2: LBG encoding process using LBG splitting to initialize the codebook. $L = 4$.

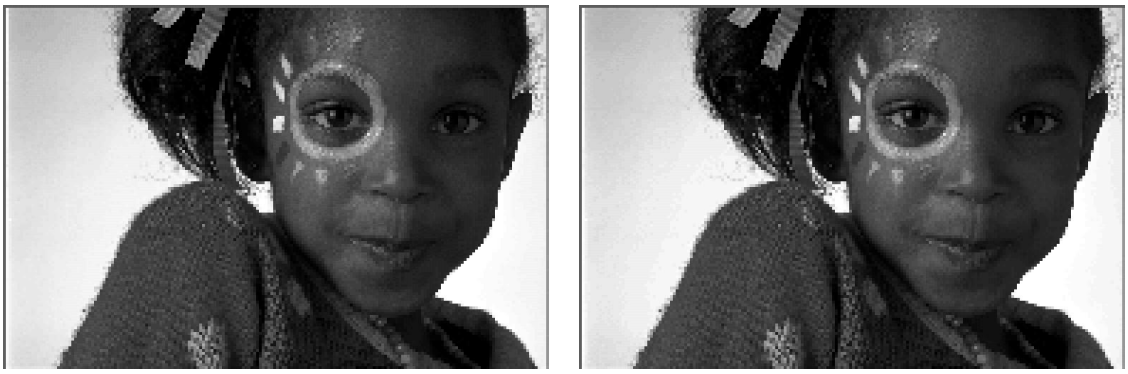
(a) Original image, $D = 0$ (b) $\epsilon = 0.01$, $D \simeq 407.2$

Figure 3: Details, $L = 4$.

(a) Original image, $D = 0$ (b) $\epsilon = 0.001$, $D \simeq 5078.9$



(c) $\epsilon = 0.005$, $D \simeq 5074.9$ (d) $\epsilon = 0.01$, $D \simeq 5078.8$



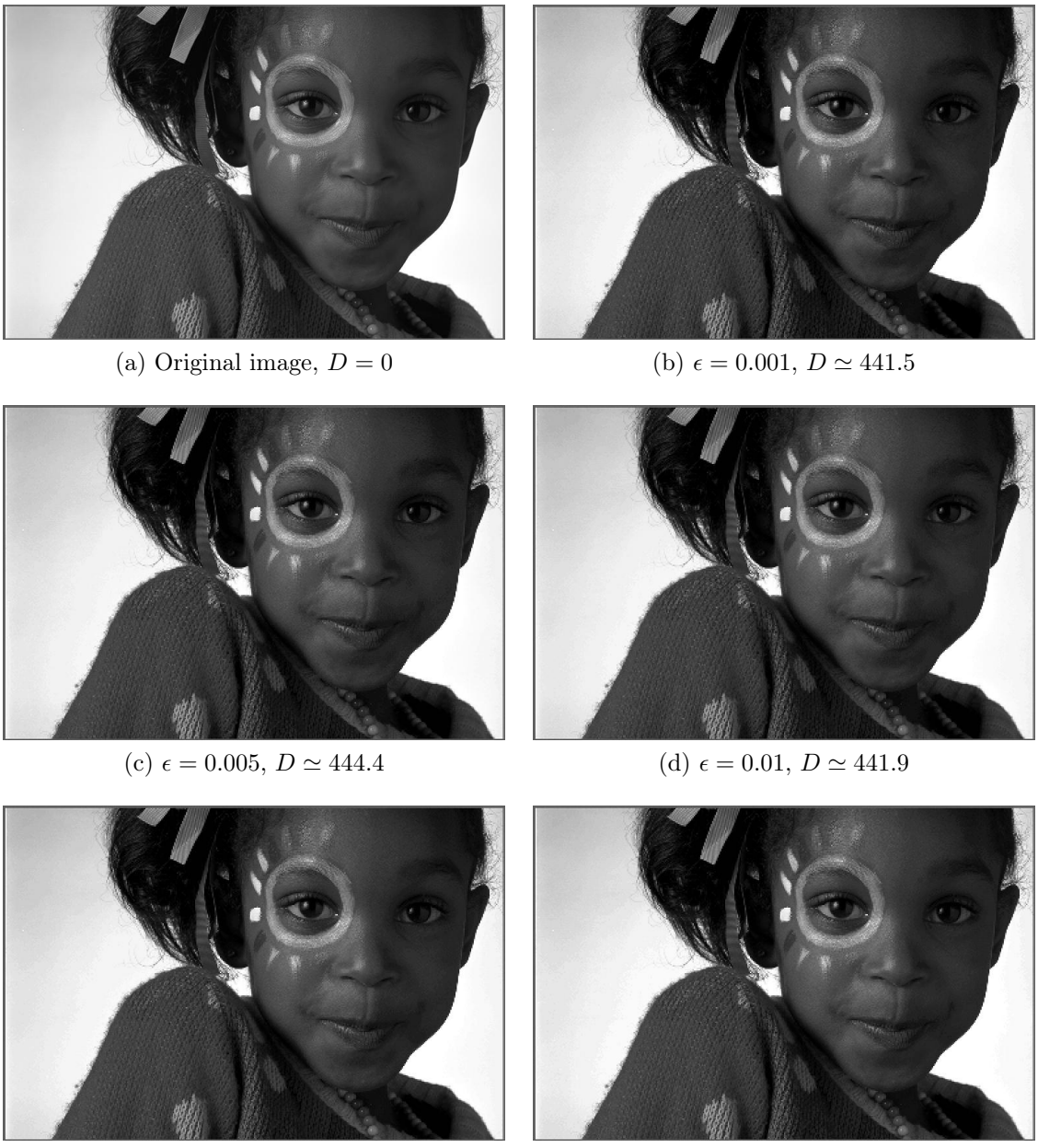
(e) $\epsilon = 0.05$, $D \simeq 5083.4$ (f) $\epsilon = 0.1$, $D \simeq 5083.4$

Figure 4: LBG encoding process using LBG splitting to initialize the codebook. $L = 16$.

(a) Original image, $D = 0$ (b) $\epsilon = 0.1$, $D \simeq 5083.4$

Figure 5: Details, $L = 16$.

3.3 Encoding an image which is not present in the training set



(a) Original image, $D = 0$

(b) $\epsilon = 0.001$, $D \simeq 441.5$

(c) $\epsilon = 0.005$, $D \simeq 444.4$

(d) $\epsilon = 0.01$, $D \simeq 441.9$

(e) $\epsilon = 0.05$, $D \simeq 440.5$

(f) $\epsilon = 0.1$, $D \simeq 442.2$

Figure 6: LBG encoding process using LBG splitting to initialize the codebook. $L = 4$.

(a) Original image, $D = 0$ (b) $\epsilon = 0.005$, $D \simeq 444.4$

Figure 7: Details, $L = 4$.

(a) Original image, $D = 0$ (b) $\epsilon = 0.001$, $D \simeq 3212.9$

(c) $\epsilon = 0.005$, $D \simeq 3219.1$ (d) $\epsilon = 0.01$, $D \simeq 3228.3$

(e) $\epsilon = 0.05$, $D \simeq 3226.0$ (f) $\epsilon = 0.1$, $D \simeq 3218.6$

Figure 8: LBG encoding process using LBG splitting to initialize the codebook. $L = 16$.

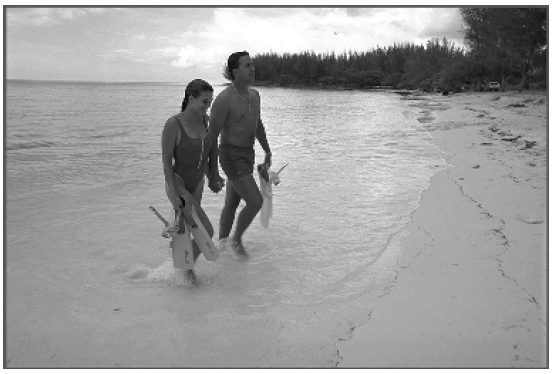
(a) Original image, $D = 0$ (b) $\epsilon = 0.01$, $D \simeq 3228.3$

Figure 9: Details, $L = 16$.

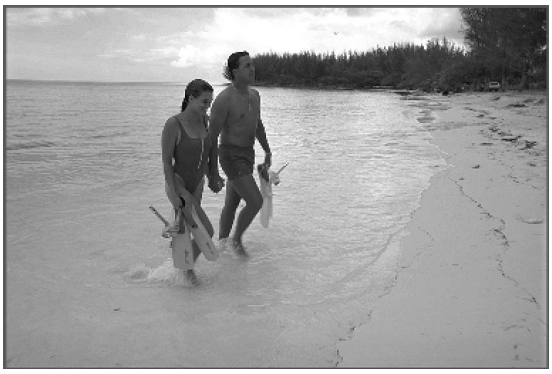
3.4 Encoding an image of the training set using a randomly initialized codebook

(a) Original image, $D = 0$

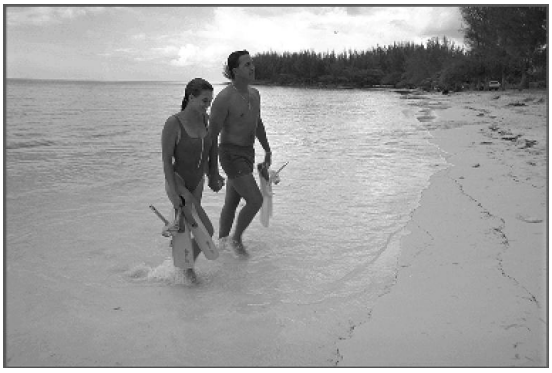
(b) $\epsilon = 0.001$, $D \simeq 404.4$



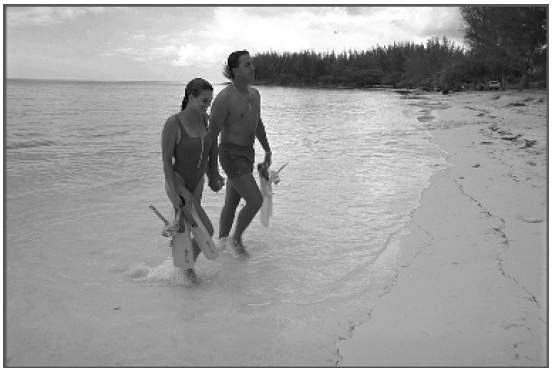
(c) $\epsilon = 0.005$, $D \simeq 402.6$



(d) $\epsilon = 0.01$, $D \simeq 404.8$



(e) $\epsilon = 0.05$, $D \simeq 406.1$



(f) $\epsilon = 0.1$, $D \simeq 405.3$

Figure 10: LBG encoding process using a random initialized codebook. $L = 4$.

(a) Original image, $D = 0$

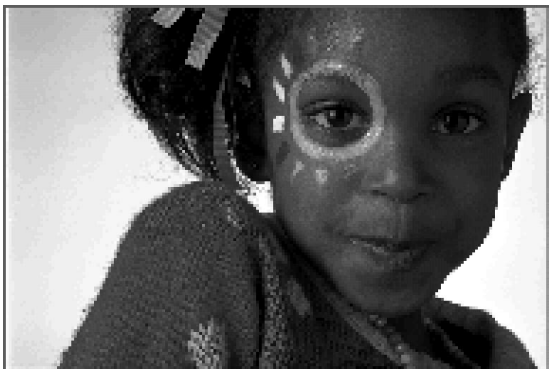
(b) $\epsilon = 0.05$, $D \simeq 406.1$

Figure 11: Details, $L = 4$.

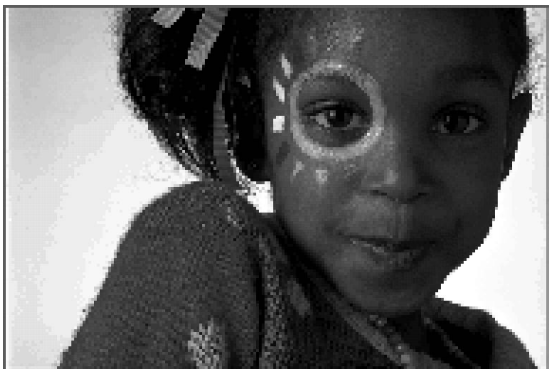
(a) Original image, $D = 0$

(b) $\epsilon = 0.001$, $D \simeq 5078.1$

(c) $\epsilon = 0.005$, $D \simeq 5073.8$



(d) $\epsilon = 0.01$, $D \simeq 5075.2$



(e) $\epsilon = 0.05$, $D \simeq 5104.3$

(f) $\epsilon = 0.1$, $D \simeq 5088.8$

Figure 12: LBG encoding process using a random initializated codebook. $L = 16$.

(a) Original image, $D = 0$ (b) $\epsilon = 0.05$, $D \simeq 5104.3$

Figure 13: Details, $L = 16$.

3.5 Observations

The following observations can be made:

- the quantization process is much noisier when $L = 16$ (see Figures 4 and 5) than when $L = 4$ (see Figures 2 and 3). We can state this either watching the output of the encoding process or having a look at the distorsion values. This is reasonable, since the codebook size K is the same in both cases even though the vector sizes are different: it would be necessary to build a much larger codebook for the case with $L = 16$ to obtain performances that are similar to the case with $L = 4$.

- in the case of $L = 4$ the quantization process is applied to two pictures of exactly the same size (768×512), one inside the training set (see Figures 2 and 3) and the other outside it (see Figures 6 and 7): the distortion of the latter one is higher. However, performing the same test in the case of $L = 16$ (see Figures 4, 5 and 8, 9) the distortion of the image which is present in the training set is higher than the distortion introduced in the image which is outside it. Therefore, one can conclude that the distortion which is introduced by the LBG quantization procedure does depend only on the input picture and not on whether the input picture is in the training set or not.

- there is not much difference in terms of distortion between the random initialization technique and the LBG splitting technique (compare Figures 2 and 10 for $L = 4$ and Figures 4 and 12 for $L = 16$): the performances of these two methods are approximately the same. The reason for this is that the codebook size K is small (only 256 reconstruction levels, when the number of possible sets of L input samples is $(2^8)^L = 256^L \gg 256$, provided that every pixel is coded using 8 bits = 256 greyscale levels).

- even though one decreases the values of the threshold ϵ, the average distortion of the various codebooks (see Table 1) and of the encoded pictures does not decrease much. This happens for both $L = 4$ and $L = 16$, using either LBG splitting initialized codebooks or randomly initialized ones; again, it is due to the fact that the codebook size K is small.

4 Conclusions

In this report a short introduction to the theory of compression is given and then the Linde-Buzo-Gray (LBG) algorithm is presented and its performances are analyzed. This procedure is a particular kind of vector quantization, since it quantizes a set of input samples jointly, at the same time. The algorithm is implemented using Maltab: different training sets and different initialization techniques are employed; images belonging to the training set and images outside the training set are quantized in order to evaluate the performances of this compressing strategy. One can observe that, as long as we consider a very small codebook size K, the performances don't vary too much if we change the values of threshold ϵ or the initialization technique (random initialization of splitting). The main difference that one can notice is between the size of the input vectors L, indeed the quantization process with $L = 16$ is much noisier than the one with $L = 4$.

4.1 Lessons learned

This experience helped me in acquiring new skills in the image processing and image compression field and in the Matlab framework.