Rapport de TP - SY26

TP03 - Quantification scalaire

Rémi Burtin

Cyril Fougeray

22 avril 2014

Université de Technologie de Compiègne

1 Introduction

L'objectif de ce TP est de programmer l'algorithme itératif de Lloyd-Max pour la conception d'un quantificateur scalaire optimal.

2 Quantification uniforme

2.1 Distorsion normalisée et rapport signal bruit

Ce premier script nous permet de calculer la distorsion normalisée (NMSE) ainsi que le rapport signal/bruit (Signal to Noise Ratio - SNR).

Pour cela, nous utilisons la fonction *lloyds* qui permet d'encoder un vecteur (ici une image en vecteur ligne) selon Lloyd-Max avec un nombre précis de partitions (correspondant aux nombre de valeurs de reconstruction). Cette fonction *lloyds* retourne la distorsion (non normalisée). Afin de calculer la distorsion normalisée, nous devons diviser cette distorsion D par la variance du vecteur passé à la fonction :

$$NMSE = \frac{D}{\sigma_X^2}$$

2.1.1 Résultats

Après avoir exécuté la fonction, nous nous rendons compte des résultats :

```
>> [nmse snr] = disto_1_1('lena.bmp', 2)
nmse = 0.3009
snr = 5.2157

>> [nmse snr] = disto_1_1('lena.bmp', 8)
nmse = 0.0190
snr = 17.2149

>> [nmse snr] = disto_1_1('lena.bmp', 256)
nmse = 0
snr = Inf
```

Lors de la première exécution, nous choisissons d'encoder l'image avec 2 valeurs différentes (correspondant à 2 niveaux de gris dans notre cas). La distorsion est donc importante, et maximale (il n'est pas possible et inutile d'encoder l'image avec 1 seule valeur car aucune information n'est conservée dans ce cas).

Évidemment, dans le cas où nous encodons l'image avec 256 niveaux de gris (telle que l'image originale), aucune distorsion n'est présente.

2.2 Quantification uniforme

Nous allons maintenant programmer une fonction permettant la quantification uniforme d'une image à partir d'un débit, correspondant aux nombre valeurs de reconstruction. Cette fonction renvoie l'image quantifiée ainsi que la distorsion avec l'image originale.

L'image originale utilisée sera ici en niveau de gris. Si l'image passée en paramètre de la fonction est une image en couleur, elle sera alors convertie via la fonction *rgb2gray*. Le signal X obtenu est alors borné entre la valeur minimum 0 et la valeur maximum 255 (domaine de variation).

Nous allons utiliser la fonction *quantiz* afin de quantifier uniformément l'image. Cette fonction permet de choisir les seuils de décision ainsi que les valeurs de reconstruction pour un signal donné.

Pour quantifier uniformément le signal, nous allons construire N niveaux de reconstruction ce qui nous permettra de définir nos seuils de décision. N est ici fonction du débit binaire R passé en paramètre de la fonction de quantification : $N = 2^R$. Ainsi, nous devons construire $N+1$ seuils de décision. Afin de partitionner uniformément le domaine de variation, nous utilisons la fonction *linspace* qui permet de créer un vecteur ayant des valeurs linéairement réparties entre une borne inférieure et une borne supérieure. La fonction *quantiz* nous impose la suppression de ces bornes dans le vecteur des seuils de décision, comme énoncé dans la documentation de la fonction :

```
 Elements of INDX = 0, 1, 2, ..., N-1 represent SIG in the range of
[-Inf, PARTITION(1)], [PARTITION(1), PARTITION(2)], [PARTITION(2),
PARTITION(3)], ..., [PARTITION(N-1), Inf].
```

Ainsi, les deux bornes extrêmes seront automatiquement $-\infty$ et $+\infty$. La fonction *linspace* a inclus les bornes dans le vecteur, nous devons donc supprimer les valeurs 0 et 255 de notre vecteur `PARTITION`.

Nous devons maintenant choisir nos valeurs de reconstruction $r_i, i \in [1, N]$. Lors d'une quantification uniforme, ces valeurs de reconstruction sont au milieu des seuils de décisions, soit la moyenne des seuils de décisions consécutifs : $r_i = (d_i + d_{i+1})/2$. Une boucle permet de construire un vecteur contenant les valeurs de reconstruction à partir du vecteur des seuils de reconstruction (*cf* Annexe A.2). L'image quantifiée est donc retournée par la fonction *quantiz*, nous la reconstruisons ensuite afin d'obtenir une matrice, telle que l'image originale. Afin d'afficher l'image via la fonction *imshow*, il faut diviser les valeurs de la matrice par 255.

Nous exécutons la fonction dans MATLAB :

```
[image_quant dist] = quant_uni_1_2('lena.bmp', 2);
partition =
   63.7500  127.5000  191.2500
codebook =
   31.8750   95.6250  159.3750  223.1250
>> imshow(image_quant./255)
>> dist
dist = 331.5913
```

FIGURE 1 – Quantification uniforme, lena.bmp, débit = 2

Dans le cas d'une quantification avec un débit égal à 8 (soit $N = 2^8 = 256$), nous construisons 257 régions de largeur 256/257. Les valeurs de reconstruction sont au milieu de chaque intervalle. La distorsion est donc dans tous les cas non nulle.

3 Quantification scalaire optimale de Lloyd-Max

3.1 Mise en oeuvre

3.2 Tests sur les différentes images

```
>> LOG
```

4 Conclusion

A Codes source MATLAB

A.1 NMSE et SNR

```
function [nmse snr] = disto(image, nbval)
    img = imread(image);

    % Conversion image en gris
    composantes = size(img, 3);
    if (composantes == 3)
        img = rgb2gray(img);
    end

    % Conversion image en liste
    dim = size(img,1)*size(img,2);
    list=reshape(double(img), 1, dim);

    % Codage arithmetique via quantification de Lloyd-Max
    [part, codebook, dist] = lloyds(list, nbval);

    % Distorsion normalisee
    nmse = dist/var(list);
    % Rapport signal bruit
    snr = -10*log10(nmse);

    return;
end
```

A.2 Quantification uniforme

```
function [image_quant disto] = quant_uni(image, debit)
    img = imread(image);
    % Conversion en gris si couleur
    composantes = size(img, 3);
    if (composantes == 3)
        img = rgb2gray(img);
    end
    % Conversion image en vecteur ligne
    dim = size(img,1)*size(img,2);
    list=reshape(double(img), 1, dim);

    % Construction des partitions
    partition = linspace(0,255, 2^debit+1);

    % Creation des valeurs de reconstruction
    for i=1:(2^debit),
        codebook(i) = (partition(i) + partition(i+1)) / 2 ;
    end

    % Quantiz utilise -Inf et +Inf pour borner la partition.
    partition = partition(2:2^debit);

    % Quantification
    [~, image_quant] = quantiz(list, partition, codebook);

    partition
    codebook

    % Calcul de la distorsion
    disto = mean((list-image_quant).^2);

    % Reconstruction de l'image en vecteur
    image_quant = reshape(image_quant, size(img,1), size(img,2));
end
```