

# PROJET TI: Quantification Vectorielle des Images

Bellili Fouad, Sourdrille Nathan  
ITI4

December 8, 2025

# Plan de la Présentation

- ▶ Rappels sur la **Quantification Scalaire (QS)**
- ▶ Présentation de la **Quantification Vectorielle (QV)**
- ▶ Algorithmes de Construction du Codebook : **K-means** et **LBG**
- ▶ **Méthodes Avancées** (TSVQ, PQ)
- ▶ Exemples de cas d'utilisation et **Applications**

## Rappels sur la Quantification Scalaire (QS)

- ▶ La QS est un processus de traitement **pixel par pixel**.
- ▶ Chaque pixel  $x$  est remplacé par un niveau d'intensité  $\hat{x}$  parmi  $L$  niveaux possibles.
- ▶ Formule de quantification uniforme (pour une image 8 bits) :

$$\hat{x} = \text{round} \left( \frac{x}{255} (L - 1) \right) \frac{255}{L - 1}$$

- ▶ **Limite de la QS** : Ce procédé ne tient **pas compte des structures locales** de l'image (corrélations spatiales).

# QS : Illustration de l'effet des faibles niveaux

Quantification scalaire,  $L = 2$



Quantification scalaire,  $L = 4$



Quantification scalaire,  $L = 8$



Quantification scalaire,  $L = 16$



Figure: QS pour différentes valeurs de niveau  $L$

## Indices de Comparaison (Qualité)

- **MSE** (*Mean Squared Error*) : Erreur quadratique moyenne entre l'image originale  $x_i$  et l'image reconstruite  $\hat{x}_i$ .

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2$$

- **PSNR** (*Peak Signal-to-Noise Ratio*) : Mesure de qualité (plus le PSNR est élevé, meilleure est la qualité).

$$\text{PSNR} = 10 \log_{10} \left( \frac{255^2}{\text{MSE}} \right)$$

# Principe de la Quantification Vectorielle

- ▶ Contrairement à la QS, la QV opère sur des **blocs de pixels** (vecteurs de  $\mathbb{R}^n$ ).
- ▶ Cette approche permet une **meilleure exploitation** des corrélations spatiales.
- ▶ L'objectif est de réduire la quantité de données en conservant une bonne qualité (compression).
- ▶ **Les 4 grandes étapes de la QV :**
  1. Découpage en vecteur blocs.
  2. Construction d'un Codebook.
  3. Codage (recherche du vecteur le plus proche).
  4. Décodage (reconstruction).

## QV : Illustration de la compression par blocs

Image originale



Image quantifiée (QV)



# Étapes détaillées de la Quantification Vectorielle

1. **Découpage en vecteurs** : L'image est divisée en blocs  $n \times n$  (vecteurs  $v \in \mathbb{R}^n$ ).



## Étapes détaillées de la Quantification Vectorielle

1. **Découpage en vecteurs** : L'image est divisée en blocs  $n \times n$  (vecteurs  $v \in \mathbb{R}^n$ ).
2. **Construction du Codebook ( $\mathcal{C}$ )** : Création d'un dictionnaire de  $K$  vecteurs représentatifs (codewords).

$$\mathcal{C} = \{c_1, c_2, \dots, c_K\}, \quad c_k \in \mathbb{R}^n$$

## Étapes détaillées de la Quantification Vectorielle

1. **Découpage en vecteurs** : L'image est divisée en blocs  $n \times n$  (vecteurs  $v \in \mathbb{R}^n$ ).
2. **Construction du Codebook ( $\mathcal{C}$ )** : Création d'un dictionnaire de  $K$  vecteurs représentatifs (codewords).

$$\mathcal{C} = \{c_1, c_2, \dots, c_K\}, \quad c_k \in \mathbb{R}^n$$

3. **Codage** : Chaque bloc  $v$  est remplacé par l'indice  $k^*$  du codeword le plus proche (minimisation de la distance).

$$k^* = \arg \min_{k \in \{1, \dots, K\}} \|v - c_k\|^2$$

# Étapes détaillées de la Quantification Vectorielle

1. **Découpage en vecteurs** : L'image est divisée en blocs  $n \times n$  (vecteurs  $v \in \mathbb{R}^n$ ).
2. **Construction du Codebook ( $\mathcal{C}$ )** : Création d'un dictionnaire de  $K$  vecteurs représentatifs (codewords).

$$\mathcal{C} = \{c_1, c_2, \dots, c_K\}, \quad c_k \in \mathbb{R}^n$$

3. **Codage** : Chaque bloc  $v$  est remplacé par l'indice  $k^*$  du codeword le plus proche (minimisation de la distance).

$$k^* = \arg \min_{k \in \{1, \dots, K\}} \|v - c_k\|^2$$

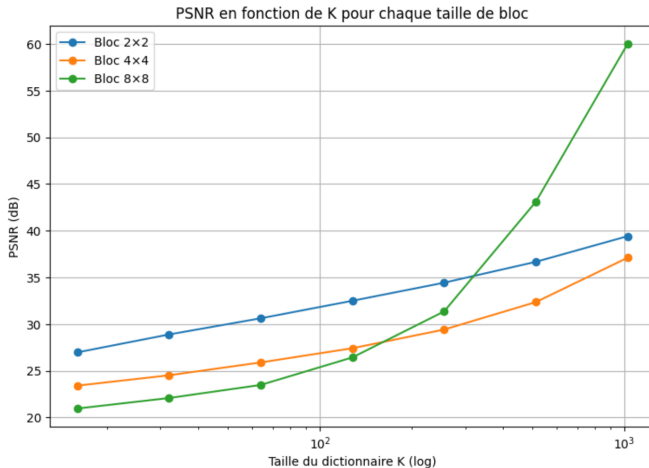
4. **Décodage** : L'image est reconstruite en remplaçant chaque indice  $k^*$  par son codeword associé  $\hat{v}$ .

$$\hat{v} = c_{k^*}$$

# Algorithme K-means

- ▶ Algorithme de **clustering** fondamental.
- ▶ Il partitionne les vecteurs de l'image en K ensembles, chaque ensemble étant représenté par son centre (le **codeword**).
- ▶ L'initialisation est **aléatoire**.
- ▶ **Compromis** : Peut converger vers des minima locaux.
- ▶ paramètre K: nombre de codewords
- ▶ paramètre L: taille des blocs vecteurs

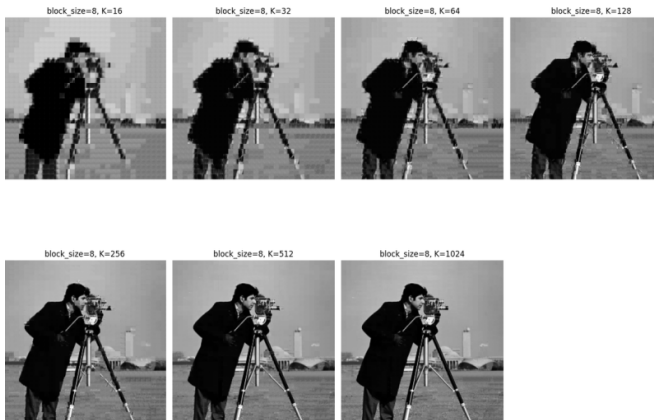
# Choix des paramètres K et L



# Algorithme LBG (*Linde–Buzo–Gray*)

- ▶ Extension du K-means utilisant une stratégie d'apprentissage **progressive** et **plus stable**.
- ▶ Utilise un mécanisme de **Split** :
  1. Commence par un unique codeword (la moyenne globale).
  2. Double la taille du codebook à chaque itération en dupliquant les codewords ( $\pm\epsilon$ ).
  3. Applique K-means localement sur les nouveaux centres.

## QS : Illustration de l'effet des faibles niveaux



## K-means vs LBG : Analyse comparative

- ▶ **PSNR** : K-means obtient souvent un PSNR **légèrement meilleur** car il est plus flexible dans le regroupement des blocs.
- ▶ **Fiabilité** : LBG construit un codebook **plus solide** et est moins dépendant du hasard de l'initialisation que K-means.
- ▶ **Conclusion opérationnelle** :
  - ▶ Pour la rapidité, on utilise **K-means**.
  - ▶ Pour la fiabilité, on utilise **LBG**.



# Quantification Vectorielle par Arbre (TSVQ)

- ▶ Utilise une **structure arborescente hiérarchique** (*Complete Binary Tree*).
- ▶ **Avantage majeur** : Permet un **encodage très rapide** (recherche du codeword en  $\mathcal{O}(\log K)$ ).
- ▶ **Compromis** : Le codebook est généralement **moins optimal** que celui généré par K-means ou LBG.

# Quantification par Produit (PQ)

- ▶ Méthode utilisée lorsque les vecteurs sont de **très grande dimension**.
- ▶ Le vecteur est découpé en **sous-vecteurs**, chacun quantifié indépendamment.
- ▶ Permet une **compression extrême** de la mémoire et des temps d'accès.
- ▶ **Application moderne** : Utilisation massive dans la **recherche de similarité** à grande échelle (ex: FAISS).

# Domaines d'Utilisation de la QV

- ▶ **Compression d'images** : Historiquement, avant le JPEG.
- ▶ **Codage audio bas débit** : Pour compresser les paramètres acoustiques (ex: systèmes GSM).
- ▶ **Compression de textures 3D** : Optimisation de l'espace mémoire dans les moteurs graphiques (ex: Nintendo 64).
- ▶ **Compression des modèles de Deep Learning** : Réduction de la taille mémoire des réseaux (un enjeu majeur aujourd'hui).

# Conclusion

- ▶ Le projet a permis de lier les notions théoriques (QS, QV, clustering) à une implémentation pratique (MSE/PSNR, K-means/LBG).
- ▶ La QV est une technique fondamentale offrant une **compression efficace** en exploitant les corrélations spatiales.
- ▶ L'efficacité repose sur le **compromis** entre qualité, complexité de calcul et taille du dictionnaire.
- ▶ La QV reste très pertinente pour l'optimisation des ressources dans les **applications modernes** (Deep Learning, recherche de similarité).

# Bibliographie

- ▶ <https://ichi.pro/quantification-vectorielle-a-l-aide-de-l-algorithme-k-means-109414082708262>  
[theses.fr/1992METZ008S](https://theses.fr/1992METZ008S)
- ▶ <https://en.wikipedia.org/wiki/Vectorquantization>
- ▶ <https://www.geeksforgeeks.org/dsa/linde-buzo-gray-lbg-algorithm/>
- ▶ <https://en.wikipedia.org/wiki/K-meansclustering>