Master: INFORMATIQUE
Parcours: VICO Visual Computing

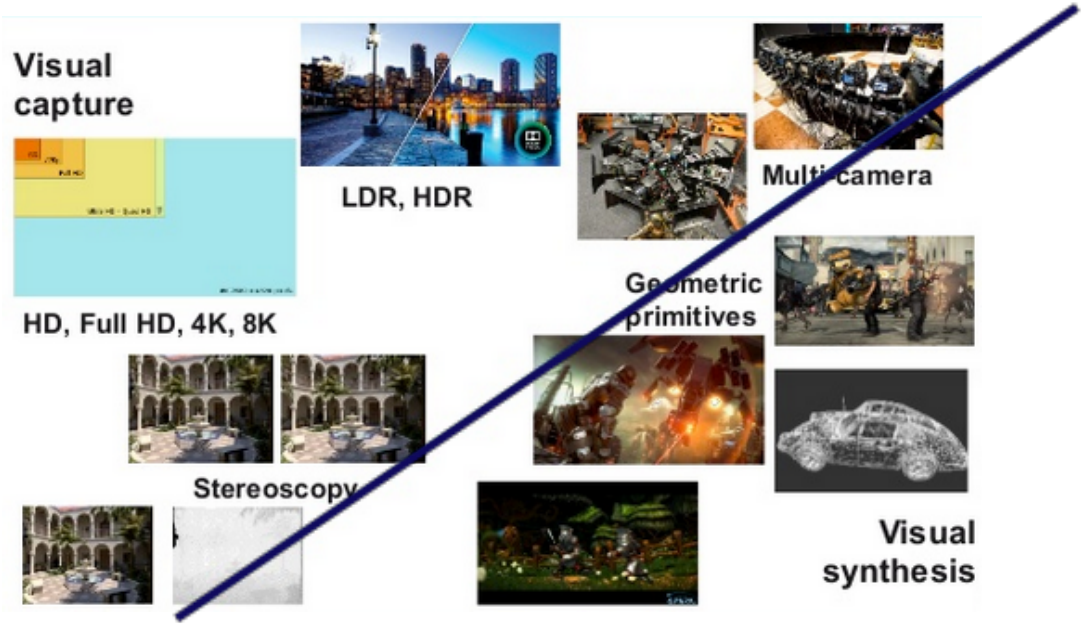
UE: Multimedia Communication

VQ - Vector Quantization
TSLVQ - Tree Structured Lattice VQ

vincent.ricordel@univ-nantes.fr

Plan

1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results
6. Conclusion

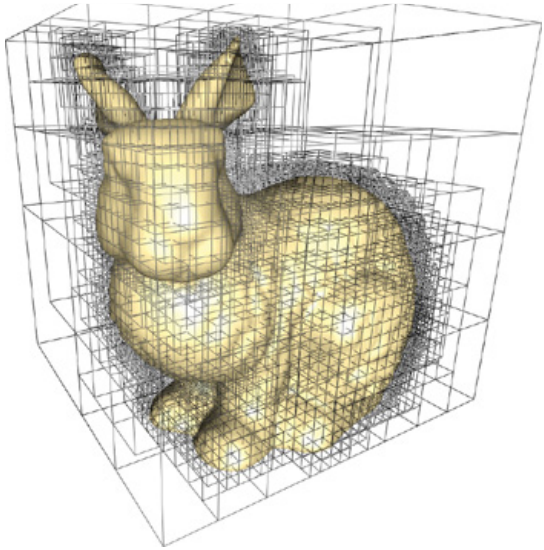


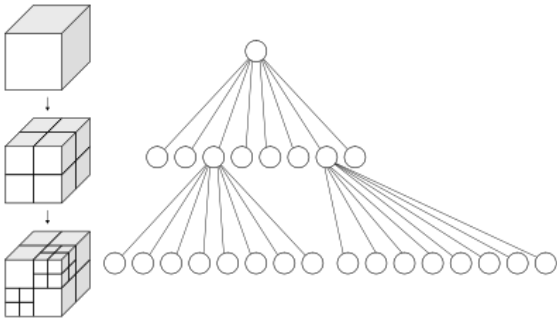
Convergence between 2 worlds

Point Cloud

- Challenge: Massive 3-D data sets
  - Unstructured millions of points
  - Costly to store/transmit/manipulate
- Goal: Find efficient techniques for representation and compression

- standards
  - April 2017: MPEG issued a Call for Proposals on PCC
    => Final standard in early 2020 with V-PCC & G-PCC
- Octree-based PCC methods
  - Point cloud compression with graph transform (Zhang2014)
    - construct graphs on small neighborhoods of occupied voxels
  - Extended Shape Adaptive DCT (Cohen2016)
    - represent points (average) on an uniform grid and apply 1-D DCT along each direction
- Tree-Structured Lattice Vector Quantization (TSLVQ) (Ricordel1998)
  - truncated lattices embedding
  - multi-stage procedure of quantization based on 3x3x3 tree splitting





Plan

1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results
6. Conclusion

Principe

- Vector Quantizer, dimension k, size L

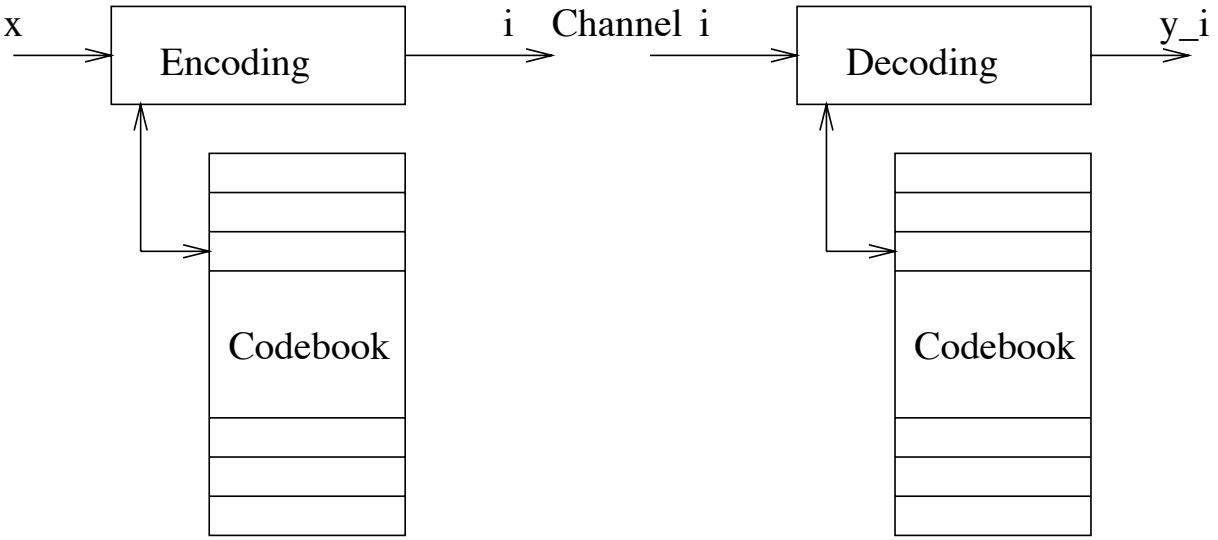
$$\begin{aligned} \mathcal{Q} : R^k &\longrightarrow \mathcal{D} \\ \mathbf{x} &\longmapsto \mathcal{Q}(\mathbf{x}) = \mathbf{y}_i \end{aligned}$$

$$\mathcal{D} = \left\{ \mathbf{y}_i \in R^k / i = 1, 2, ..., L \right\}$$

- R^k partition into L Voronoï cells

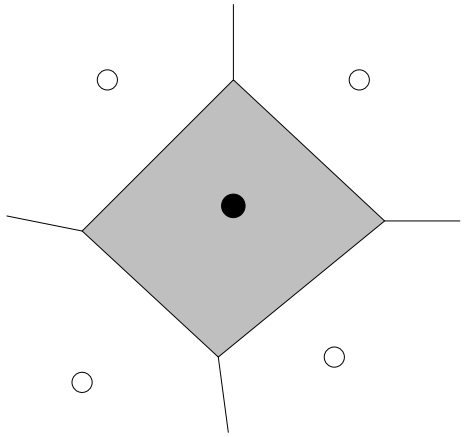
$$C_i = \left\{ \mathbf{x} \in R^k / \mathcal{Q}(\mathbf{x}) = \mathbf{y}_i \right\}$$

- Coding



- Encoding, "Nearest Neighbour" rule

$$C_i = \left\{ \mathbf{x} \in R^k / \mathcal{Q}(\mathbf{x}) = \mathbf{y}_i,\ si\ d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \forall j \neq i \right\}$$

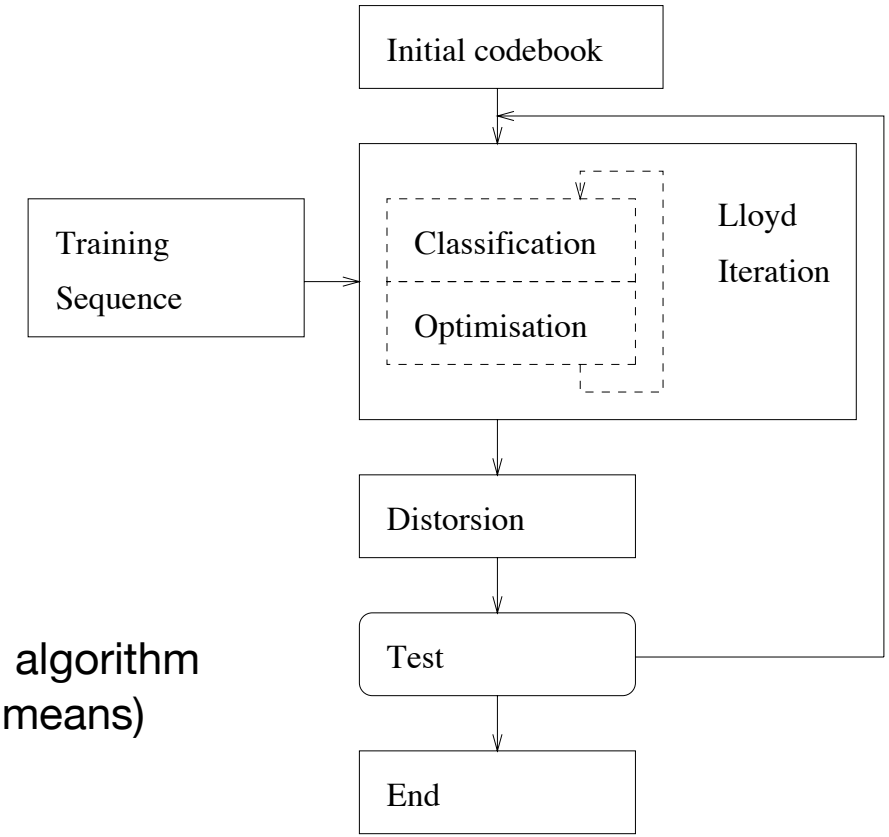


Performance evaluation

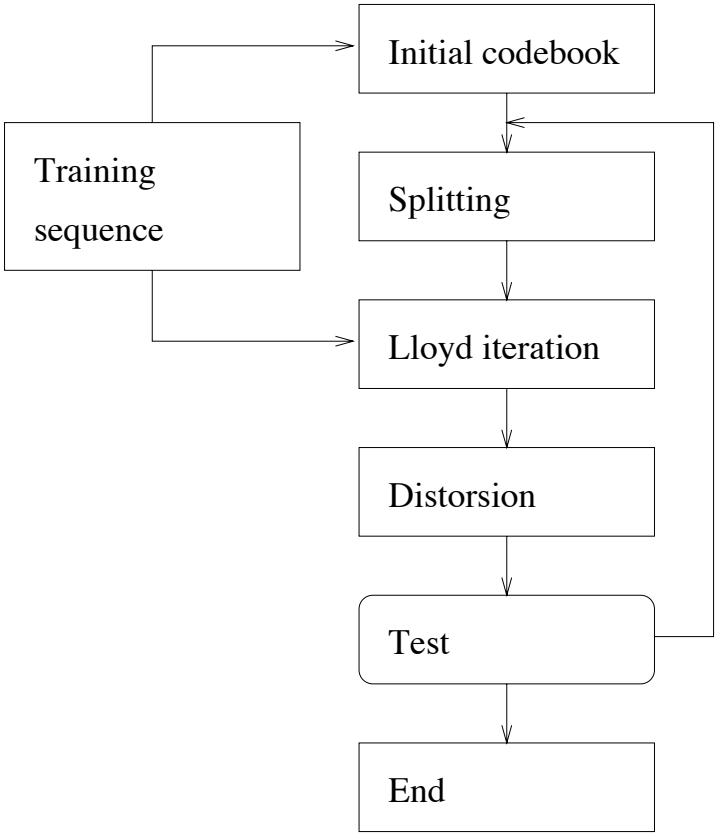
- Rate [bpp]
  - ▷ Rate constrained : $R = \frac{1}{k}.\log_2 L$
  - ▷ Entropy constrained : $R \simeq H(\mathcal{D})$
- Distorsion

$$D = \frac{1}{k} \sum_{i=1}^{L} \int_{C_i} L_2(\mathbf{x}, \mathbf{y}_i).p_{\mathbf{X}}(\mathbf{x})\ d\mathbf{x}$$

(nearly) Optimal VQ



Lloyd algorithm
(k-means)



LBG algorithm

- Training
- Encoding $\rightarrow$ exhaustive research $O(L)$

Plan

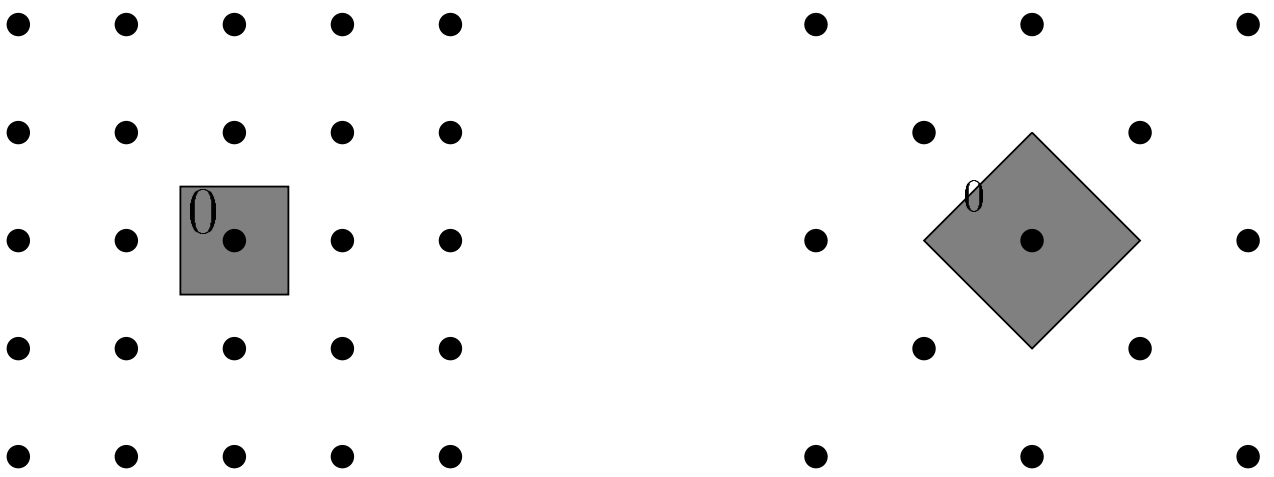
1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results
6. Conclusion

Lattices Λ

- Regular arrangement of identical spheres $\rightarrow$ spheres centers

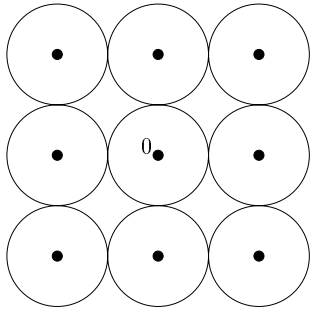
  $0 \rightarrow$ origin

- $Z^k = \left\{ \mathbf{y} = (y_1,\ y_2,\ \ldots,\ y_k)^T \ \ / \ \ y_i \in Z \right\}$
- $D_k = \left\{ \mathbf{y} = (y_1,\ y_2,\ \ldots,\ y_k)^T \ \ / \ \ \mathbf{y} \in Z^k,\ \Sigma_{i=1}^{k}\, y_i = 0 \ \mathsf{mod}(2) \right\}$

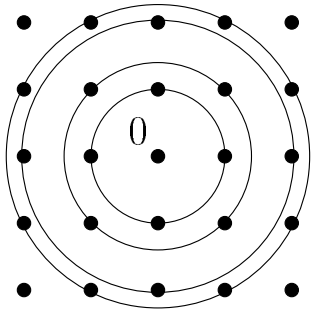


Characteristics

- Packing radius ρ



- Series Theta, Nu (Gaidon93), modified Theta (Moureaux94)



- Best quantizing lattices : A_2, D_4, E_8, Λ_{16}
- Fast quantizing lattices (Conway and Sloane83) : Z^k, D_k, E_8, Λ_{16}
  $\Longrightarrow$ complexity $O(k)$

Coding scheme

- Lattice Truncation
  - ▷ Shape
    - ▷ i.i.d Gaussian source $\rightarrow$ sphere (L_2)
    - ▷ i.i.d Laplacian source $\rightarrow$ pyramid (L_1)
    - ▷ correlated GG source $\rightarrow$ ellipse (ponderated L_2)
  - ▷ Truncation energy (Fisher86) $\mathcal{E}_t \rightarrow$ points number
- Source normalisation
  - ▷ before quantization
  - ▷ scaling factor

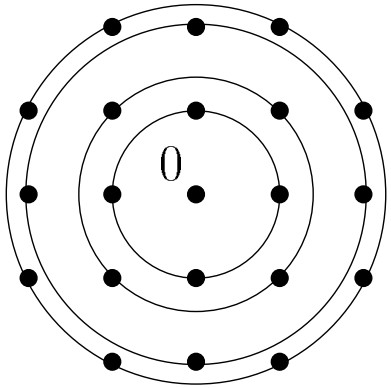
Projection within a sphere

- Sphere radius $\sqrt{\mathcal{E}_t}$
- Training sequence $\mathcal{SA} = \left\{\mathbf{x}_j = (x_1, \ldots, x_k)^T \;\; / \;\; j = 0, 1, 2, \ldots\right\}$
- Vector energy $\mathcal{E}(\mathbf{x}) = L_2(\mathbf{x})$
- $\mathcal{E}_{max} = \max_{\mathcal{E}} \left\{\mathcal{E}(\mathbf{x}) \;\; / \;\; \mathbf{x} \in \mathcal{SA}\right\}$
- Scaling factor $F = \sqrt{\mathcal{E}_t / \mathcal{E}_{max}}$

- Real source $\rightarrow$ vectors with energy greater than $\mathcal{E}_{max}$
  are processed separately

Labeling of the lattice points

- Index calculation $\rightarrow$ product code (Lamblin88, Moureaux94, Onno95)
  - ▷ sub-index for the sphere energy
  - ▷ sub-index for the point position



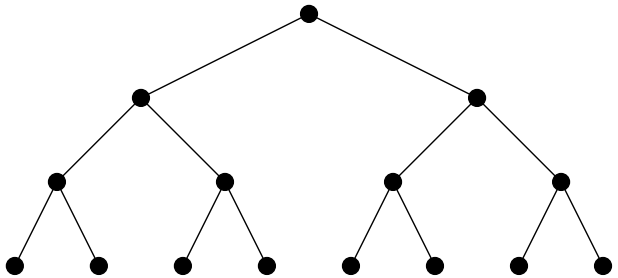
Conclusion

- Advantage
  - ▷ fast quantization
  - ▷ predefined codebook
- Drawback $\rightarrow$ method for simple sources

$\Longrightarrow$ TSLVQ : hierarchical packing of embedded truncated lattices

Plan

1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results
6. Conclusion



- Encoding
  - ▷ complexity $O(\log_B L)$
  - ▷ tree storage
- Decoding
  - ▷ leaves
  - ▷ progressive reconstruction
- Unbalanced tree $\rightarrow$ variable rate
  - ▷ pruning approach (Breiman84, Chou89)
  - ▷ greedy approach (Makhoul85, Riskin91)

Training

Characterisation of each node n_i

- Probability of reaching n_i

$$P(n_i) = \frac{\text{card}(C_{n_i})}{\text{card}(SA)}$$

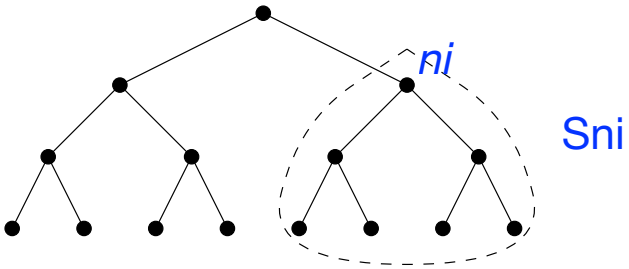
- Average distorsion

$$d(n_i) = \frac{1}{\text{card}(C_{n_i})} \cdot \sum_{\mathbf{x} \in C_{n_i}, \mathbf{x} \in SA} \|\mathbf{x} - \mathbf{y}_{n_i}\|^2$$

- Entropy code length

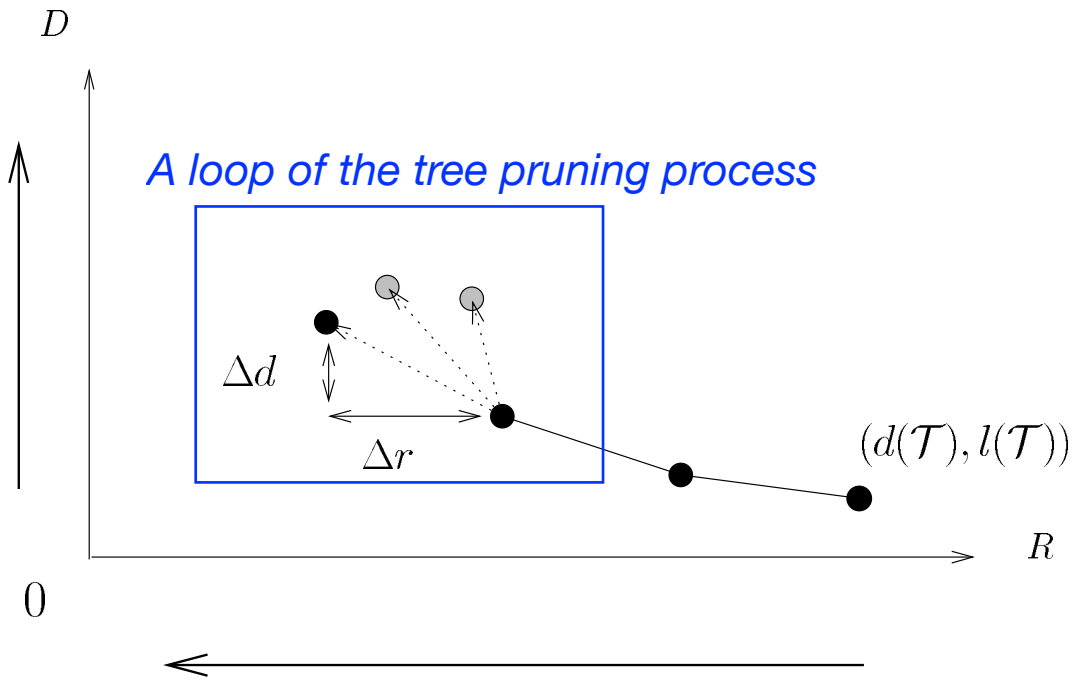
$$r(n_i) = -\log_2 P(n_i)$$

Tree pruning



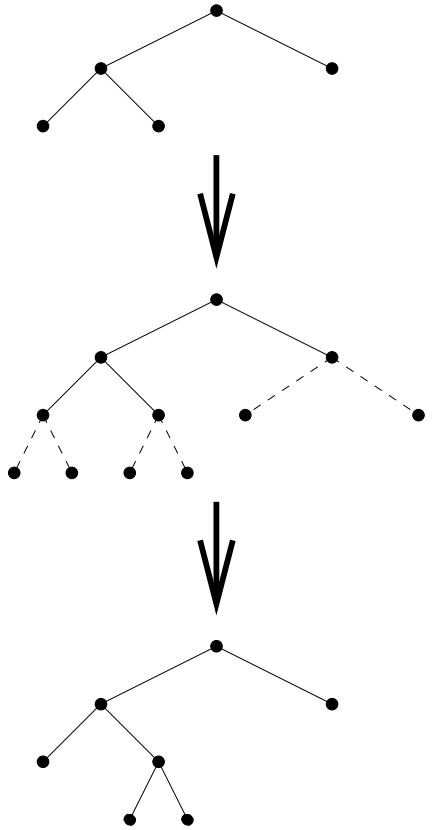
- BFOS algorithm (Breiman84)
  1. complet tree $\mathcal{T}$
  2. successive pruning
- Characterisation of each branch $\mathcal{S}_{n_i}$
  - ▷ increase in distorsion if $\mathcal{S}_{n_i}$ is removed $\Delta d(\mathcal{S}_{n_i})$
  - ▷ decrease in rate if $\mathcal{S}_{n_i}$ is removed $\Delta r(\mathcal{S}_{n_i})$
  - ▷ **BFOS criterion** $\lambda(n_i) = \Delta d(\mathcal{S}_{n_i})/\Delta r(\mathcal{S}_{n_i})$

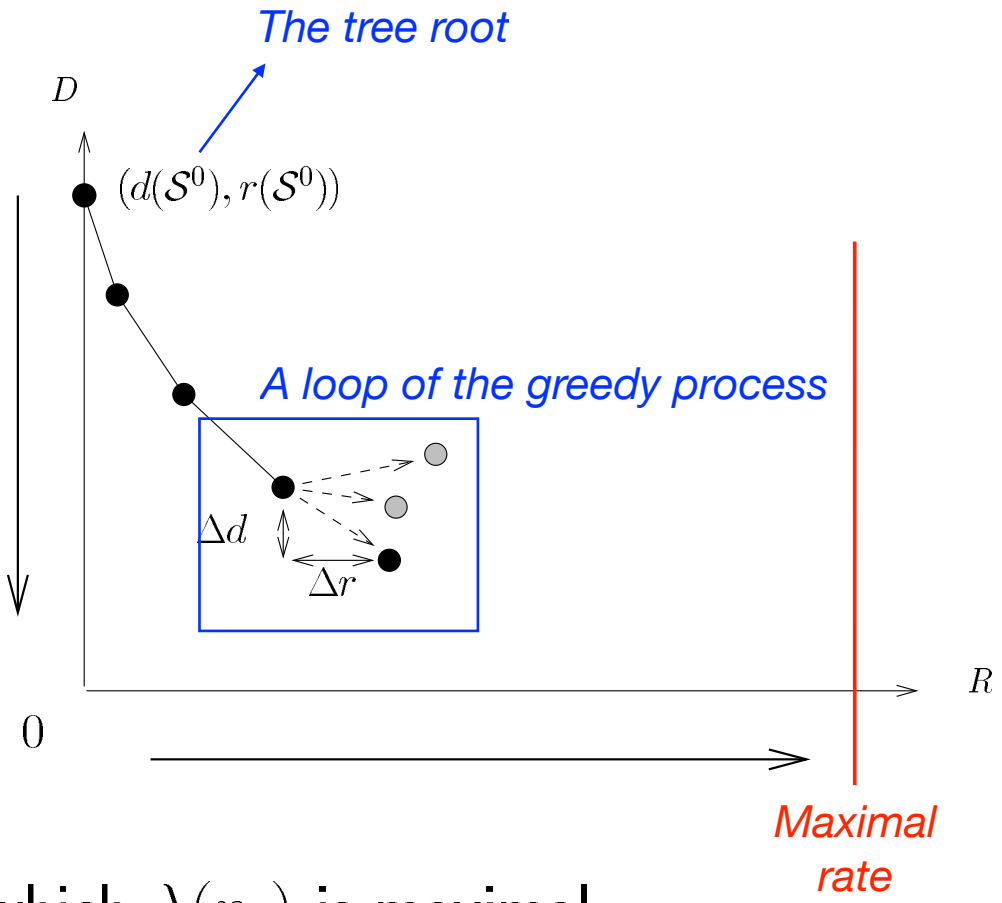
Pruning principle



$\Longrightarrow$ Pruning of the branch for which $\lambda(n_i)$ is minimal

Greedy approach





$\Longrightarrow$ Splitting of the leaf for which $\lambda(n_i)$ is maximal

Conclusion

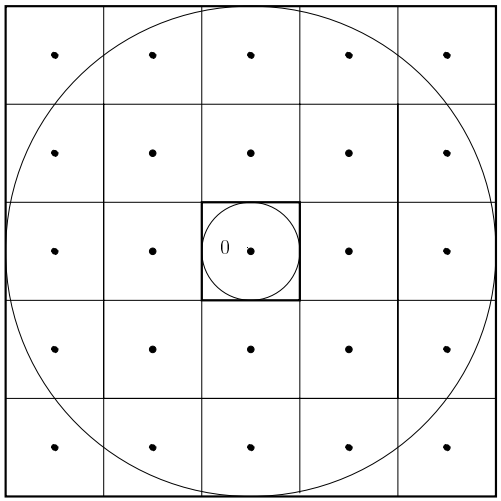
- Pruning algorithm
  - ▷ global approach
  - ▷ storage of the complete tree
- Greedy algorithm
  - ▷ local approach
  - ▷ limited storage

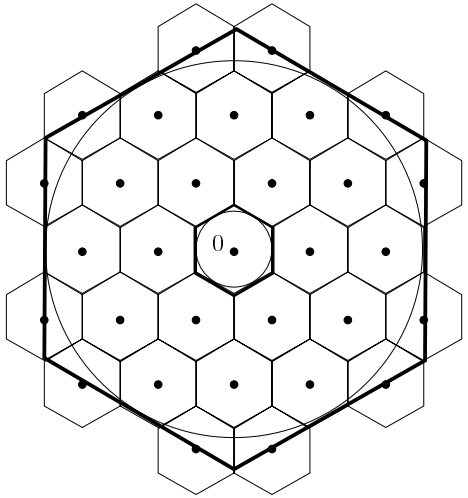
Plan

1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results
6. Conclusion

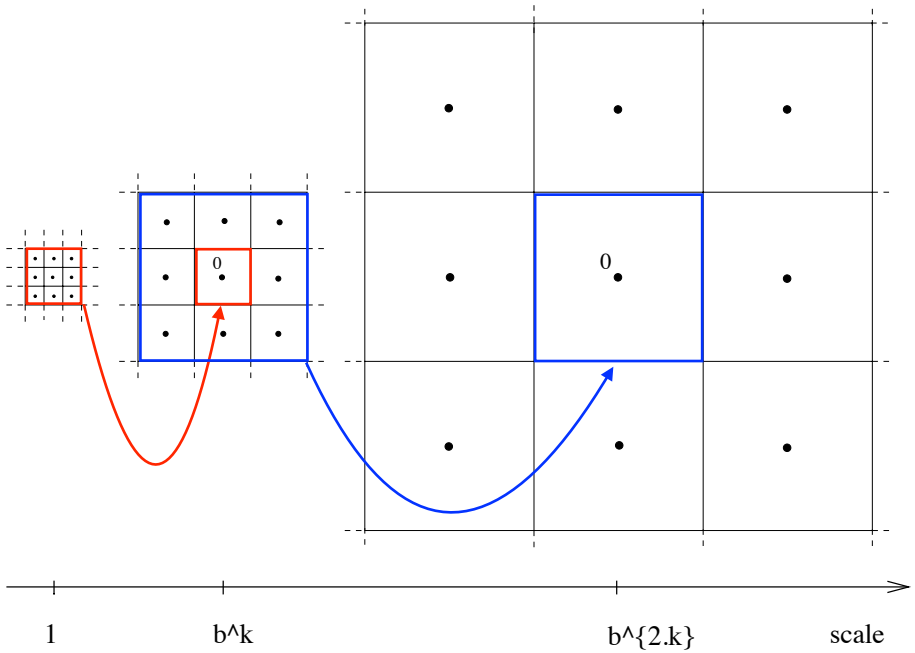
Embedded Lattices

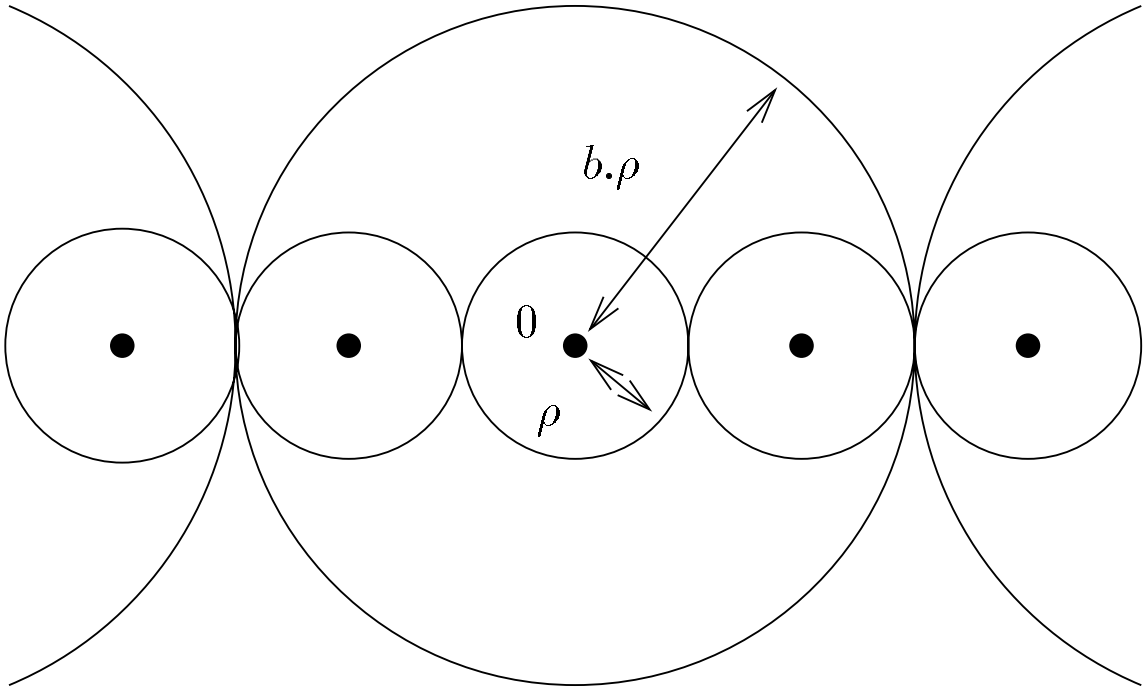
- Support Lattices Z^k, D_k, E_8, $\Lambda_{16} \rightarrow$ fastest quantizing algorithms
- **Embedding** :
  by contracting it, embed a truncated lattice in its Voronoï cell
- **Optimal embedding** :
  the rescaled truncated lattice covers exactly the Voronoï cell
- **Sub-optimal embedding** :
  the rescaled truncated lattice covers maximaly the Voronoï cell





Hierarchy of embedded lattices





(lower scale lattice)

- Packing radius of the support lattice : ρ

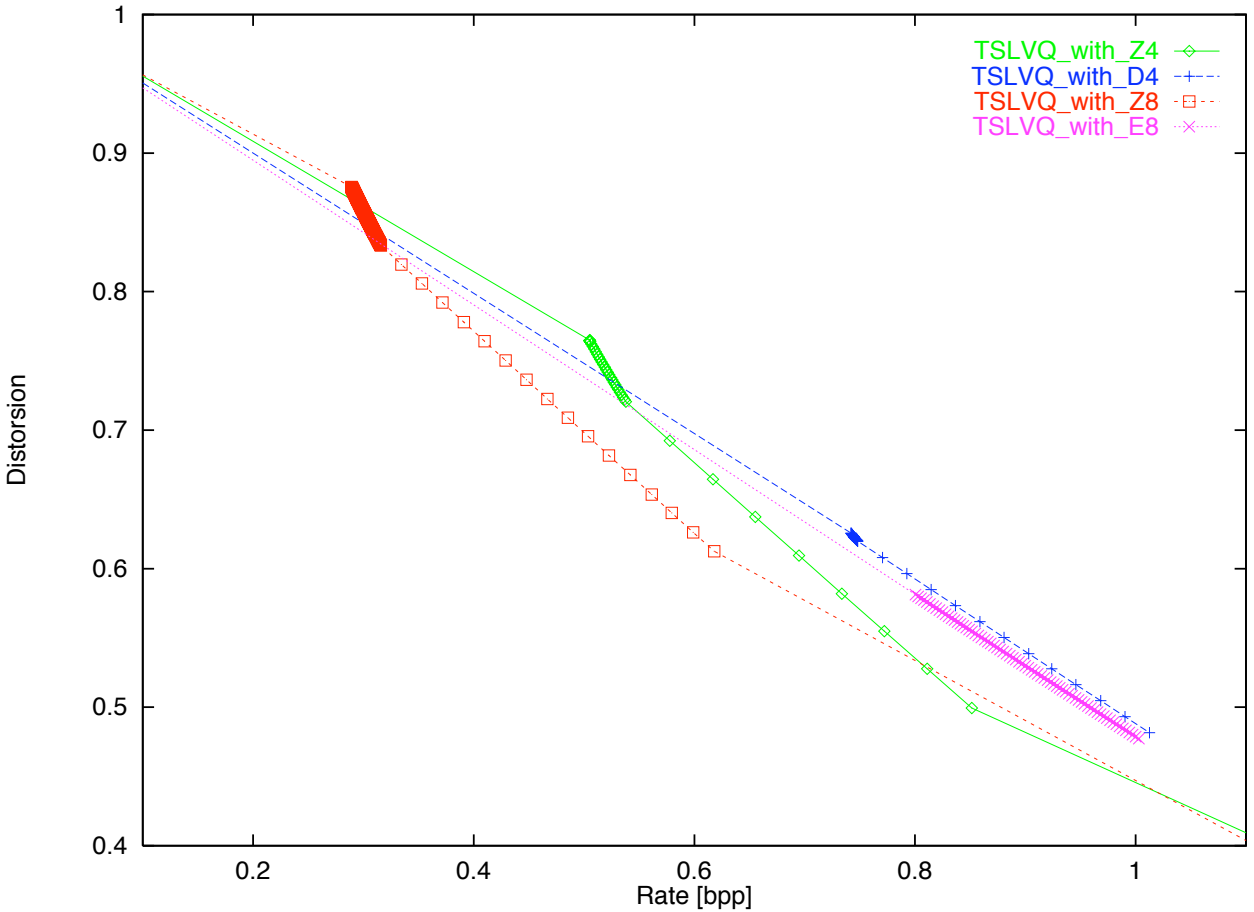
(next higher scale lattice)

- Packing radius of the dilated lattice :

$$b.\rho \text{ avec } b \in \mathbf{R} \;\; / \;\; b > 1$$

$$\Longrightarrow \qquad b = 2.n + 1 \;\; / \;\; n \in N^*$$

Optimal lattice

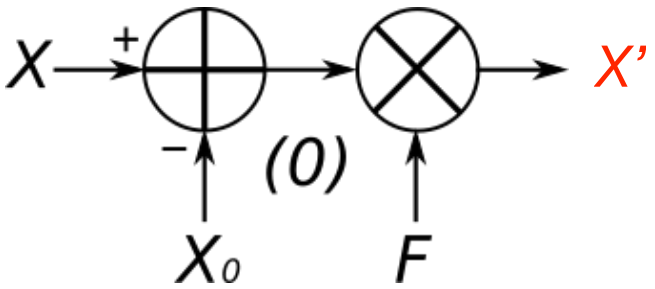


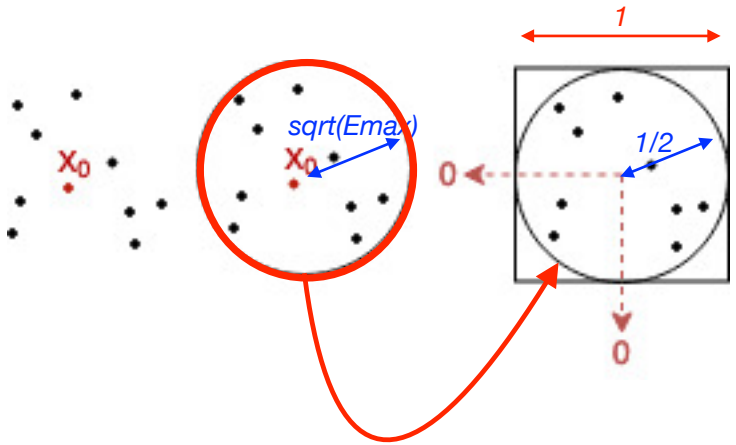
$\Longrightarrow Z^k$ is optimal

TSLVQ design

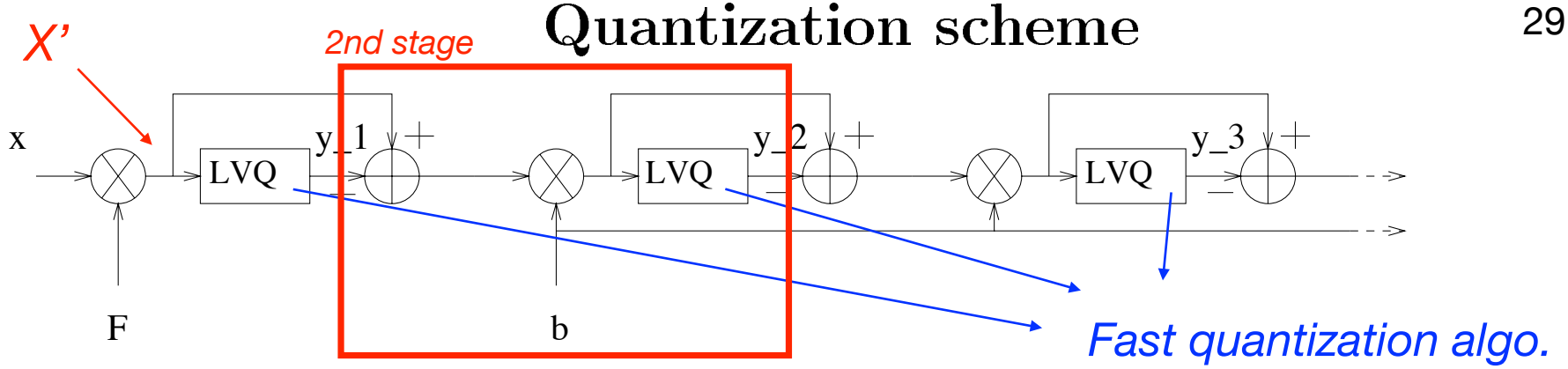
- Step (0): Initialization
  - The points set is normalized:
    centering the points ($\boldsymbol{X_0}$: points mean)
    Scaling data with the factor $F = \frac{1}{2\sqrt{E_{max}}}$

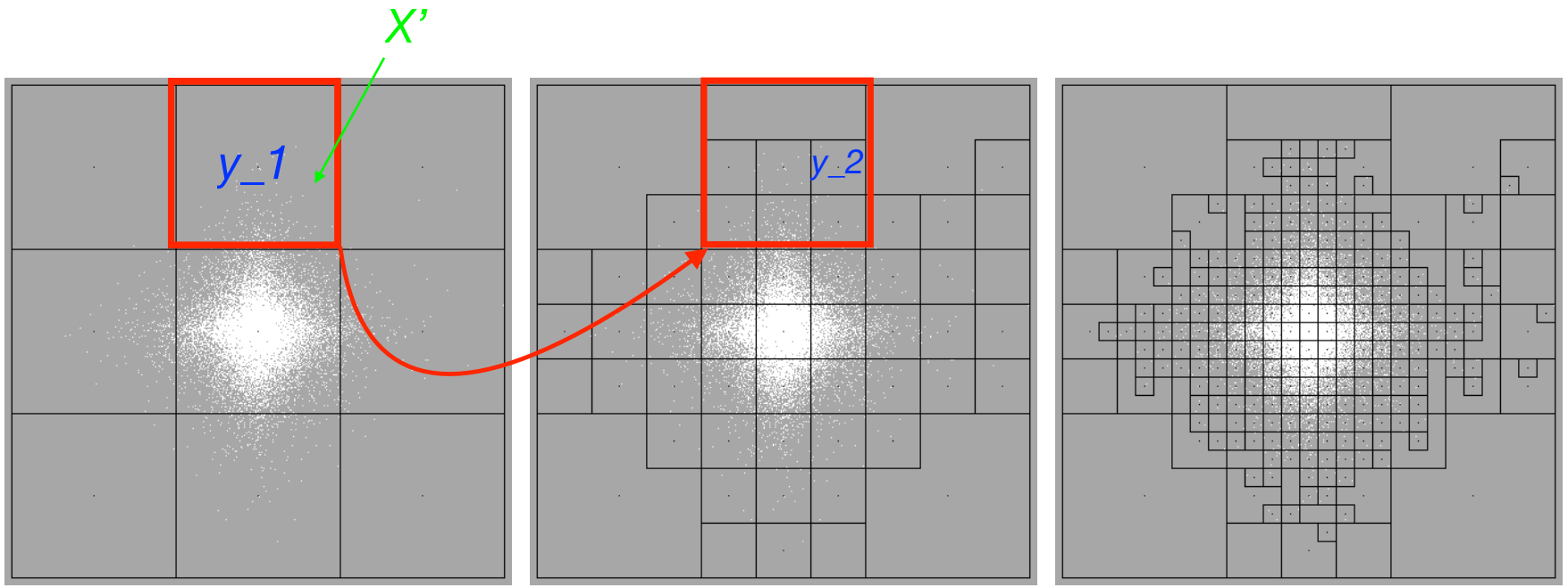
F = (1/2) / sqrt(Emax)





Quantization scheme





- A tree-structured codebook
- Progressive splitting $\rightarrow b = b_{min} = 3$

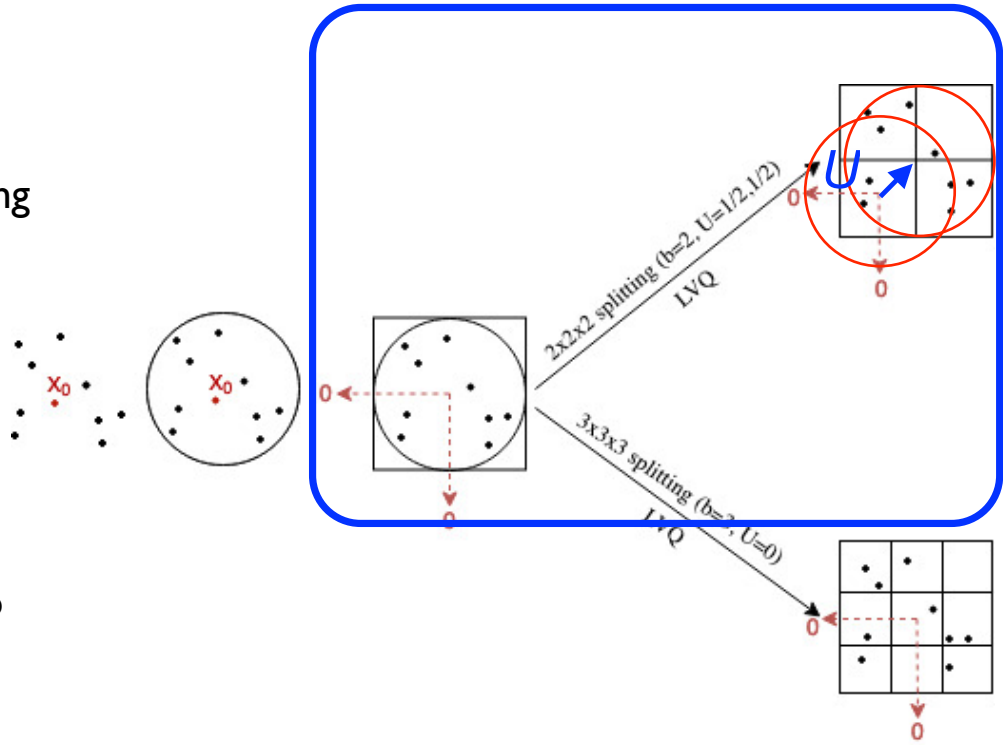
Conclusion

- Partition of the space according to :
  - ▷ the source distribution
  - ▷ the rate vs. distorsion tradeoff
- Simple labeling method
- $Z^k \rightarrow$ simple processing for the outlying source vectors
- Fast quantizing $\rightarrow$ complexity $O(h.k)$
- Bit allocation

Plan

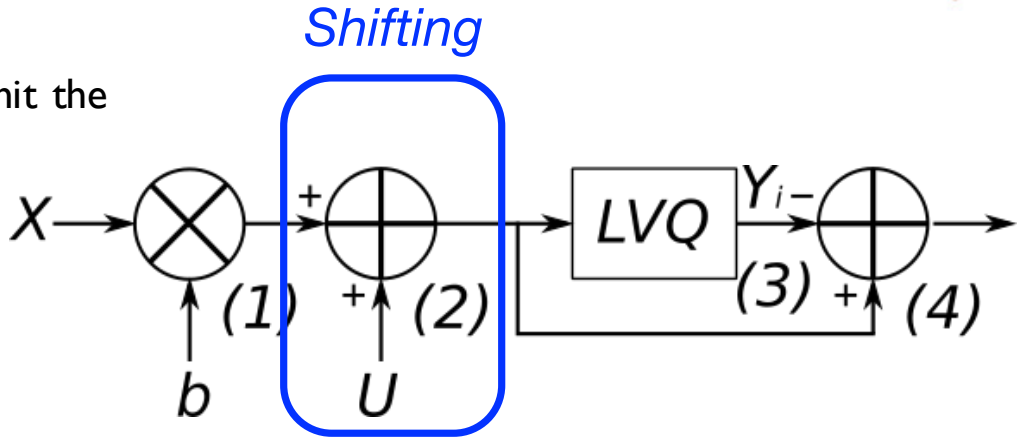
1. Context of the study
2. Vector Quantization (VQ)
3. Lattice VQ and Tree-Structured VQ
4. Tree-Structured Lattice VQ (TSLVQ)
5. Experimental results *(context: G-PCC)*

- Steps (1) and (2): Splitting
  - Scaling the input vector ***X*** by ***b*** then shifting by ***U***
  - Splitting the cube in 2x2x2 (**b=2, U=(1/2,1/2,1/2)**)
    or in 3x3x3 (**b=3, U=(0,0,0)**)



- Step (3): Projection into truncated lattices
  - Fast quantization algorithm is then used to produce by rounding the corresponding reproduction vector Y_i

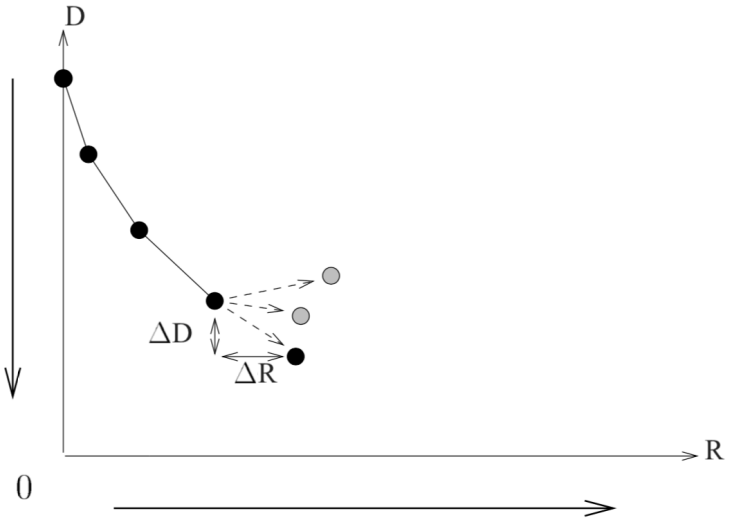
- Step (4): Recentering
  - The output vector is centered to permit the next quantization level



Iterative Process

- Tree structure design : Greedy approach
  - Cube $\mathbf{C_i}$ to split
  - Adapted splitting method (2x2x2 vs. 3x3x3)

- Lagrangian optimization technique
  - Multiplier associated to each node $\lambda = \frac{\Delta D}{\Delta R}$



Globally:

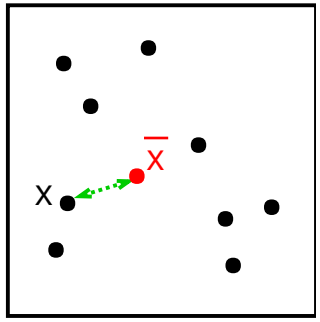
- 2 possible splittings by node for all nodes ➔ The best λ

  ➔ Maximal decrease in distortion, minimal increase in rate

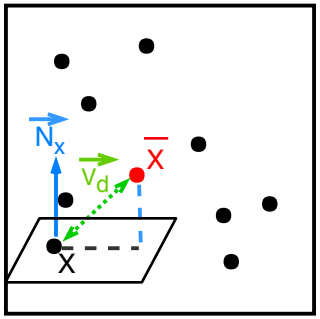
Locally:

- Increase in rate ΔR: tree encoding cost
- Decrease in distortion ΔD: 2 local metrics

  - Point-to-Point metric $D_{P2Point}(X, \overline{X}) = \sum_{X \in C_i} d(X, \overline{X})$



  - Point-to-Plane metric $D_{P2Plane}(X, \overline{X}) = \sum_{X \in C_i} || \overrightarrow{V_d}(X, \overline{X}) \cdot \overrightarrow{N}_{X_i} ||$



Iterative Process Stop: number of leaves / depth is reached

- Each occupied leaf: Output point

- Objective Comparison: *TSPLVQ method*
  - 3 splitting schemes
    - Only 3x3x3
    - Only 2x2x2
    - 2x2x2 vs. 3x3x3
  - 2 local metrics
    - Point2Point
    - Point2Plane (in **Bold**)

PC Images	MPEG G-PCC	Our approach		
		3x3x3	2x2x2	hybrid
Soldier	16.48	18.72 **19.02**	11.2816 **14.32**	18.70 **14.45**
Long Dress	16.53	18.74 **19.89**	9.28 **61.08**	15.89 **17.74**
Loot	16.54	15.33 **17.65**	18.77 **10.49**	15.36 **19.13**

Comparison of symmetric **MSE** metric

source: ICIP'2019

- Subjective Comparison: *Visual rendering result of Soldier point cloud*

(a) original point cloud 1 089 091 points

(b) MPEG lossy PCC ≈ 20 000 points

(c) only 2x2x2 splitting Point-to-point ≈ 20 000 points

(d) Only 3x3x3 splitting Point-to-point ≈ 20 000 points

(e) hybrid splitting Point-to-point ≈ 20 000 points

(f) hybrid splitting Point-to-plane ≈ 20 000 points

(g) hybrid splitting Point-to-point ≈ 30 000 points

source: ICIP'2019