FAST IMAGE VECTOR QUANTIZATION USING SPARSE OBLIQUE REGRESSION TREES

Rasul Kairgeldin and **Miguel Á. Carreira-Perpiñán**, Dept. Computer Science and Engineering, UC Merced, USA

1 Abstract

Vector quantization is a fundamental model for image coding, but large codebooks require a long training and encoding time. This can be sped up with tree-structured codes. We propose the use of sparse oblique decision trees, which have hyperplane splits with few nonzero weights in the decision nodes. Such trees can be trained on a dataset of image patches using tree alternating optimization. Experimentally with different datasets, we show these trees consistently achieve a low distortion, close to that of a flat codebook, and a much faster encoding, exceeding other tree-structured vector quantizers.

Work supported by NSF award IIS–2007147

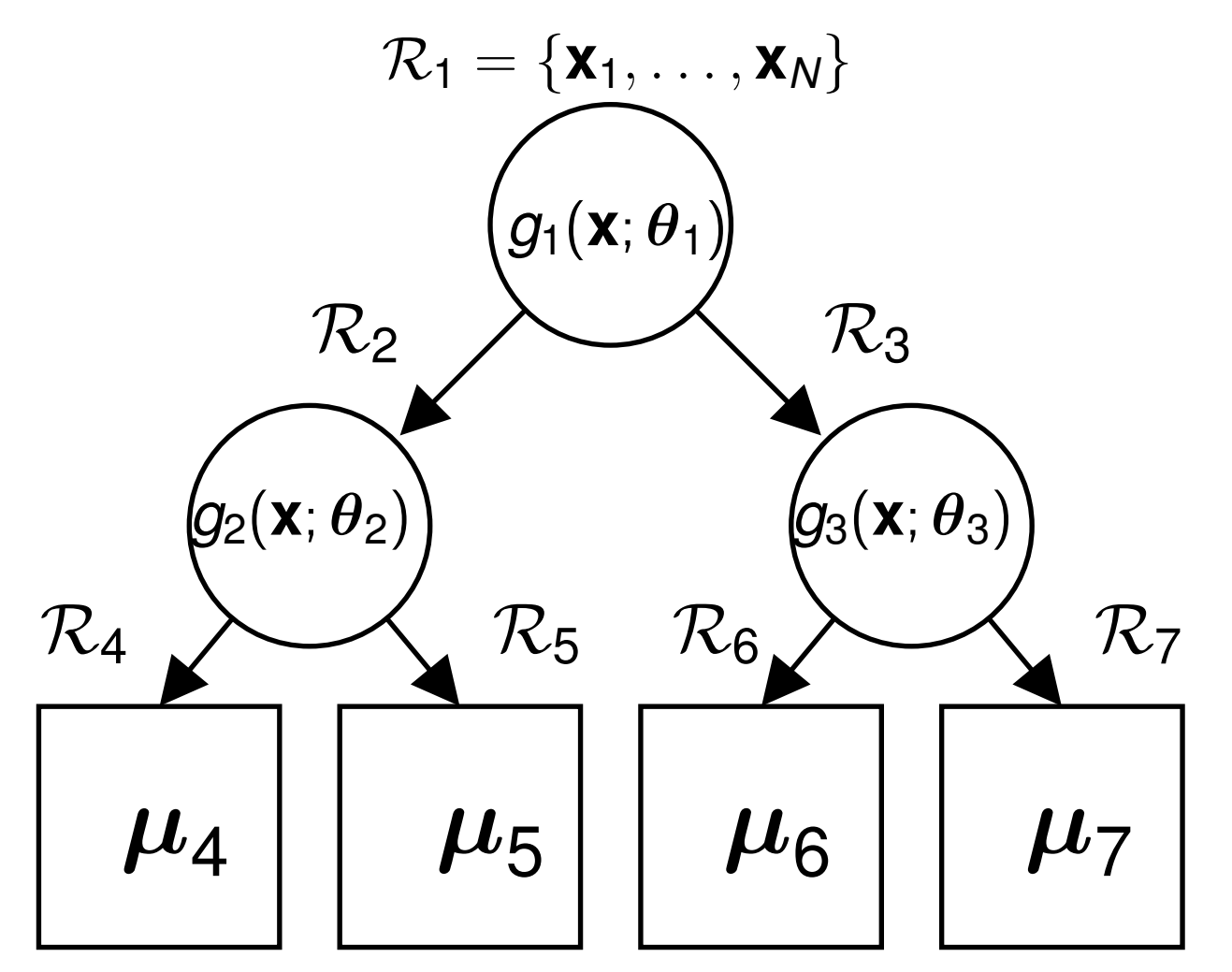
2 Vector quantization (VQ)

- VQ is widely used for image coding, representing image patches as vectors in Euclidean space
- Codebooks are typically learned by minimizing squared Euclidean distortion
- The k-means algorithm is the standard method to optimize the codebook:
  - Minimizes squared Euclidean distance (distortion) between patches and assigned codebook vectors
  - Alternates between assigning patches to codebook vectors $\mathcal{O}(NDK)$ and updating the codebook vectors $\mathcal{O}(ND)$
  - Limitation: encoding cost is linear in K (slow for large K)
  - Used as a baseline in comparisons with advanced VQ methods

3 VQ with sparse oblique trees

A sparse oblique tree is a hierarchically-structured function that maps an image $\mathbf{x} \in \mathbb{R}^D$ to an index in the codebook with K entries.

- Binary tree of depth Δ
- Each decision node $i \in \mathcal{D}$ contains a routing function $g_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \{\mathcal{D} \cup \mathcal{L}\}$
- $g_i(\mathbf{x};\boldsymbol{\theta}_i) = \texttt{left}_i$ if $\mathbf{w}_i^T\mathbf{x} + w_{0i} < 0$, otherwise $\texttt{right}_i$
- Each leaf $j \in \mathcal{L}$ contains a codebook vector $\boldsymbol{\mu}_j \in \mathbb{R}^D$
- The tree routing function $\mathbf{T}(\mathbf{x}_n;\Theta)$ directs a patch $\mathbf{x}_n$ from a root to a single leaf and predicts a corresponding codeword $\boldsymbol{\mu}_j$


$\mathcal{R}_1 = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
$g_1(\mathbf{x};\boldsymbol{\theta}_1)$
$\mathcal{R}_2$
$\mathcal{R}_3$
$g_2(\mathbf{x};\boldsymbol{\theta}_2)$
$g_3(\mathbf{x};\boldsymbol{\theta}_3)$
$\mathcal{R}_4$
$\mathcal{R}_5$
$\mathcal{R}_6$
$\mathcal{R}_7$
$\boldsymbol{\mu}_4$
$\boldsymbol{\mu}_5$
$\boldsymbol{\mu}_6$
$\boldsymbol{\mu}_7$

4 Problem formulation for VQ with sparse oblique trees

Given a oblique tree $\mathbf{T}(\mathbf{x},\Theta)$ of a fixed structure (e.g. a complete tree of depth Δ) and initial parameters (e.g. random) with K leaves, we use Tree Alternating Optimization (**TAO**) to minimize the following objective:

$$\min_{\Theta} \sum_{n=1}^{N} \left\|\mathbf{x}_n - \mathbf{T}(\mathbf{x}_n;\Theta)\right\|^2 + \lambda \sum_{i\in\mathcal{D}} \|\mathbf{w}_i\|_1$$

- It can be seen as generalizing the regular squared distortion over a dataset of patches $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$ from a flat codebook to a hierarchical one
- Assignments are not free variables, but implicitly determined by the tree structure (not a Voronoi cell, but a trainable polytope)
- The codebook consists of the leaf node vectors
- Hyperparameters:
  - Tree depth Δ controls primary model capacity and codebook size
  - Sparsity parameter λ can prune the tree by zeroing out weights, reducing complexity (secondary control of codebook size)

Algorithm 1: Learning a tree-structured vector quantizer with TAO

```
input training set $\{\mathbf{x}_n\}_{n=1}^N$;
initial tree $\mathbf{T}(\cdot;\Theta)$ of depth $\Delta$;
for depth $d = 0$ to $\Delta$ do
    for $i \in$ nodes at depth $d$ do
        if $i$ is a leaf then
            $\boldsymbol{\mu}_i \leftarrow$ fit a constant regressor at a leaf with patches in $\mathcal{R}_i$ as targets:
                $\min_{\boldsymbol{\mu}_j} \sum_{n\in\mathcal{R}_j} \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\|^2$
        else
            $\boldsymbol{\theta}_i \leftarrow$ fit a weighted binary classifier:
                $\min_{\boldsymbol{\theta}_i} \sum_{n\in\mathcal{R}_i} \overline{L}(\overline{y}_n, g_i(\mathbf{x}_n;\boldsymbol{\theta}_i)) + \lambda \|\mathbf{w}_i\|_1$
                $\overline{L}_{in}(\overline{y}_{in}, y) = l_{in}(y) - l_{in}(\overline{y}_{in}) \forall y \in \{\texttt{left}, \texttt{right}\}$
                $\overline{y}_{in} = \arg\min_y l_{in}(y)$
return trained tree $\mathbf{T}$
```

4 Computational complexity

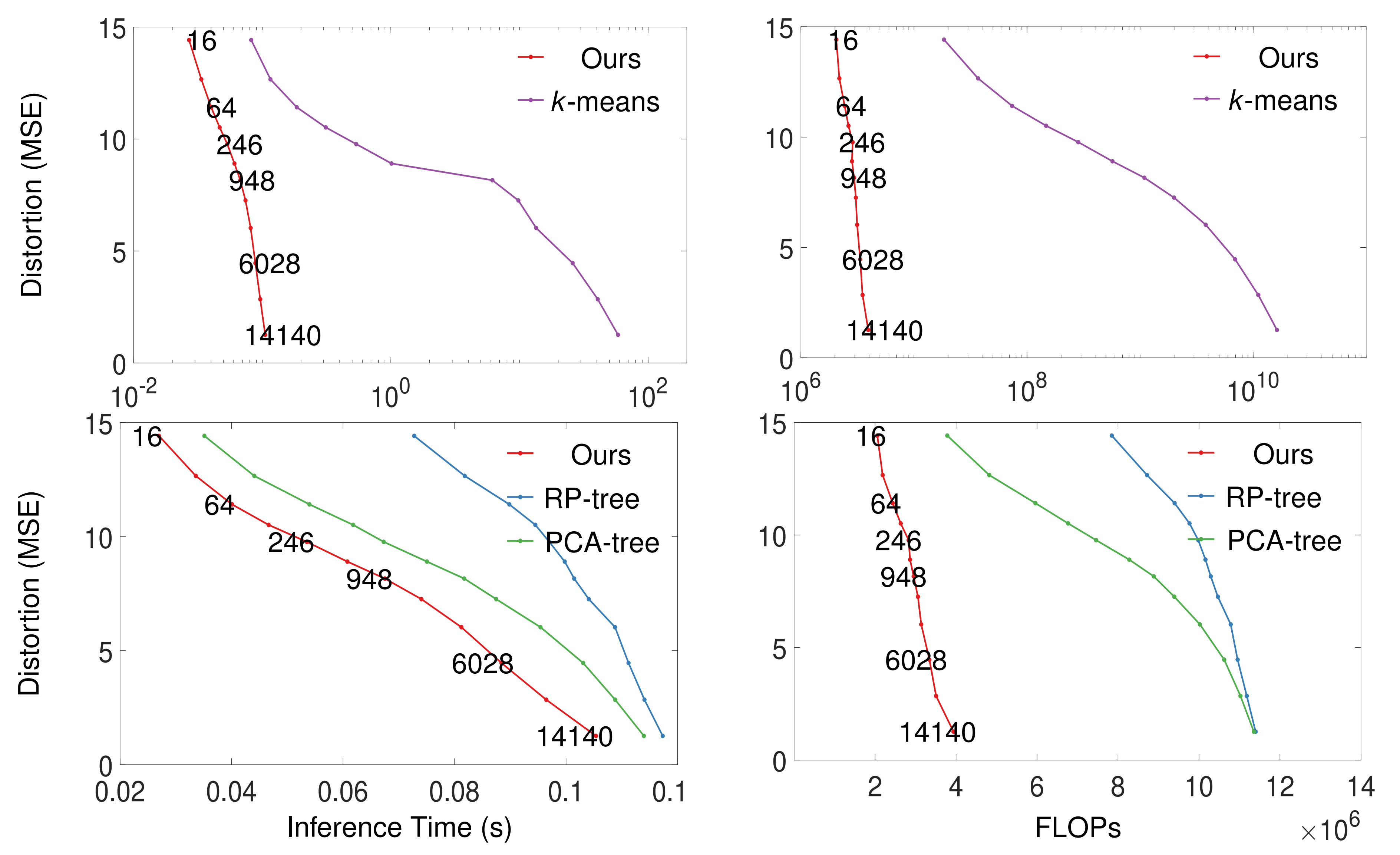
- Root node: $\mathcal{O}(\Delta DN) + \mathcal{O}(cDN)$
- Decision nodes at level Δ_i can be optimized in parallel: $\mathcal{O}(\Delta_i DN) + \mathcal{O}(cDN)$
- Total complexity at most $\mathcal{O}(\Delta^2 DN) + \mathcal{O}(c\Delta DN)$

Training complexity comparison to k-means for large K:

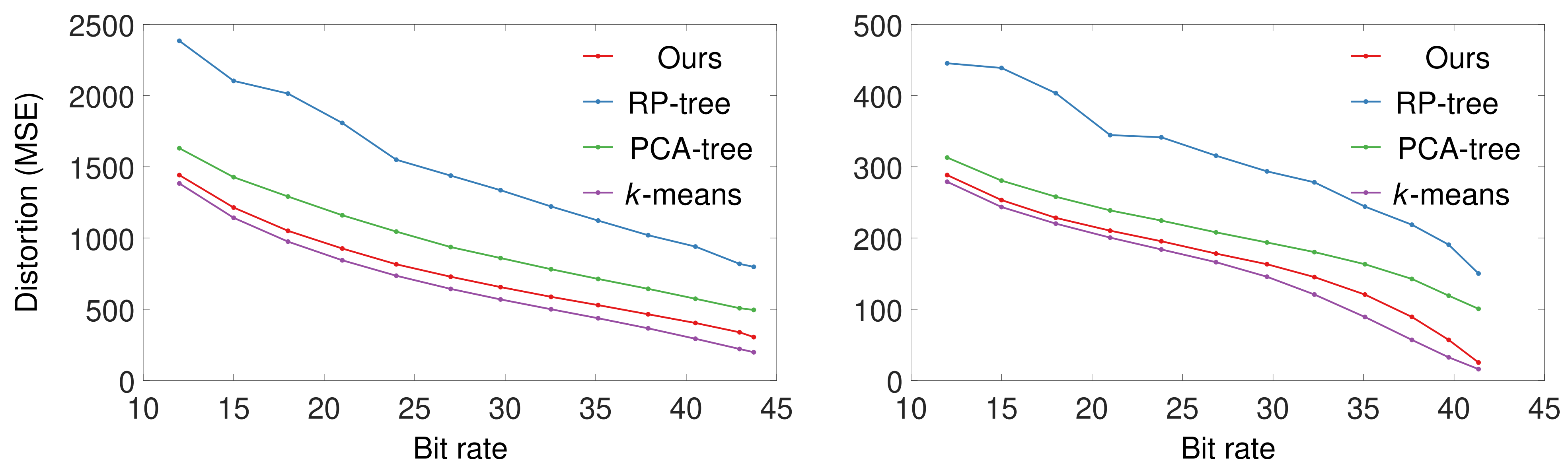
- Decision node training is $\approx \mathcal{O}(DN\log^2 K)$, much faster than k-means' $\mathcal{O}(DNK)$
- Leaf optimization matches k-means' centroid step at $\mathcal{O}(ND)$

Test patch encoding is $\mathcal{O}(D\log K)$ for a tree (root-to-leaf path) versus $\mathcal{O}(DK)$ for k-means

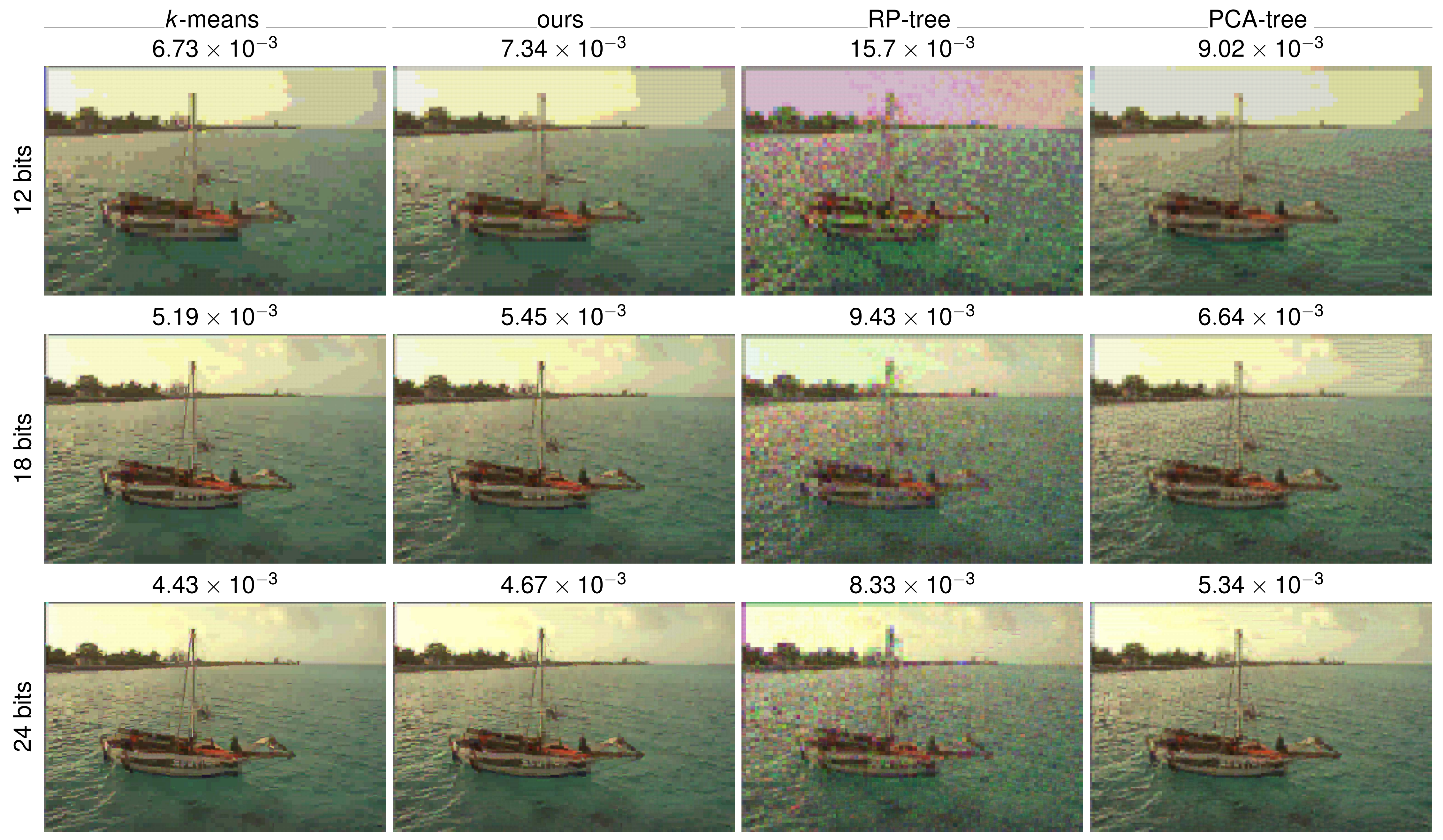
5 Experiment results



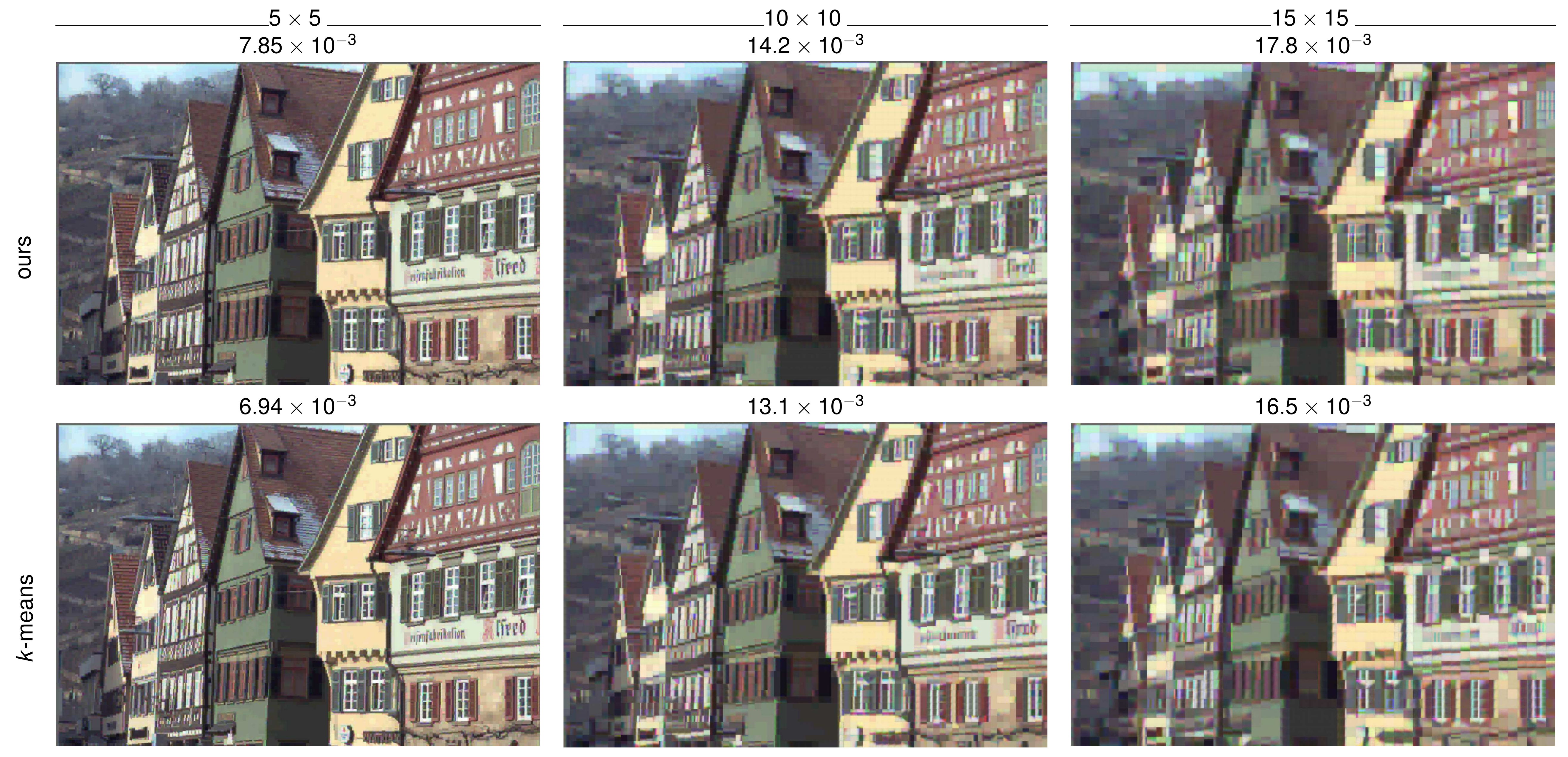
Encoding time vs distortion. Dependence between average inference time (encoding time) and floating-point operations of each method (ours, k-means, PCA-tree and RP-tree) and distortion for 1 image from Kodak dataset for different patch size. We show average codebook size per channel of our trees. Patch size 15×15. For given distortion proposed approach is much faster compared to both k-means and other tree-based VQ methods.



Rate-Distortion curves. Distortion (MSE) vs bit rate for different patch size (from left to right 5×5 and 15×15) of each method (ours, k-means, PCA-tree and RP-tree) on Kodak dataset. Our trees have much better distortion only slightly worse than one of k-means.



Quantization quality comparison of different methods for different bit rate. Distortion (MSE) and bit rate is on top of each image. The proposed method, on par with k-means, displays the best image quality.



Quantization quality produced by k-means and our method for 21 bit encoding. This shows the impact of patch size on image quality. As the patch size increases, the image becomes more susceptible to blocking artifacts. Larger patch sizes require significantly more bits to capture the details of various patches because the algorithms encode each patch as a single codeword. Quantization quality is very close to one of k-means.