Sampling in image representation and compression

Dedicated to Paul L. Butzer on the occasion of his 85th birthday

John J. Benedetto and Alfredo Nava-Tudela

November 25, 2013

1 Introduction

1.1 Background

In recent years, interest has grown in the study of sparse solutions of underdetermined systems of linear equations because of their many and potential applications [11]. In particular, these types of solutions can be used to describe images in a compact form, provided one is willing to accept an imperfect representation.

We shall develop this approach in the context of sampling theory and for problems in image compression. These problems arise for a host of reasons including the ever-increasing volume of images used in multimedia that, for example, is pervasive in Internet traffic.

The basic idea is the following. Suppose that we have a full-rank matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, where $n < m$, and that we want to find solutions to the equation,

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

where $\mathbf{b}$ is a given "signal." Since the matrix $\mathbf{A}$ is full-rank and there are more unknowns than equations, there are an infinite number of solutions to Equation (1). What if from all possible solutions we could find $\mathbf{x}_0$, the "sparsest" one, in the sense of having the least number of nonzero entries? Then, if the number of nonzero entries in $\mathbf{x}_0$ happens to be less than the number of nonzero entries in $\mathbf{b}$, we could store $\mathbf{x}_0$ instead of $\mathbf{b}$, achieving a representation $\mathbf{x}_0$ of the original signal $\mathbf{b}$ in a compressed way. This notion of *compression* is our point of view, and our presentation is phenomenological and computational, as opposed to theoretical. Our goal is to set the stage for addressing rigorously the natural questions that arise given this approach.

For example, is there a unique "sparsest" solution to Equation (1)? How does one find such a solution? What are the practical implications of this approach to image compression? How does resolution of a sparsity problem fit into the context of signal transform compression techniques? For perspective, the JPEG and JPEG 2000 standards have at their core transformations that result in different representations of the original image which can be truncated to achieve compression at the expense of some acceptable error [41, 2, 38, 15].

1.2 Finding sparse solutions

The *orthogonal matching pursuit* (OMP) algorithm is one of the techniques used to find sparse solutions to systems of linear equations such as Equation (1) [29]. It is one of the greedy algorithms that attempts to solve the general problem,

$$(P_0^\epsilon): \qquad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{Ax}-\mathbf{b}\|_2 < \epsilon. \tag{2}$$

Here, $\|\mathbf{x}\|_0 = \#\{j : |x_j| > 0\}$ is the "zero-norm" of vector $\mathbf{x}$, that counts the number of nonzero entries in $\mathbf{x}$. A greedy algorithm approach is necessary for the solution of the optimization problem defined by (2), since (P_0^ϵ) is an NP-complete problem [28]. Moreover, it can be proven that under certain circumstances there is a unique sparsest solution to (P_0^ϵ); and, under those same circumstances, OMP is then guaranteed to find it [11]. Generally, OMP *will converge to a solution of* Equation (1); but our interest, and the relevance of sampling theory, is contingent on sparsity results.

1.3 Image representation

To make a practical implementation of the compression idea described in Section 1.1, we follow the approach used in the JPEG and JPEG 2000 standards [41, 2, 38]. Also, in order to test our image representation and compression technique, we select a set of four commonly used test images, and review standard image representation concepts.

All images in our database are 512×512, 8-bit depth, grayscale images. We proceed to partition each image into $64 \cdot 64 = 4096$ subsets of 8×8 non-overlapping sub-images, and to process each of these individually. Partitioning a signal to work with more manageable pieces is a common technique [44]. Then, we vectorize each 8×8 sub-image into a vector $\mathbf{b} \in \mathbb{R}^{64}$ to be used as a right hand side in Equation (1). There are many ways to do this vectorization and we investigate three of them.

To complete the setup, we need to choose a matrix $\mathbf{A}$. We begin with $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1] \in \mathbb{R}^{64\times128}$, where DCT_1 is a basis of one-dimensional discrete cosine transform waveforms and Haar_1 is a basis of Haar wavelets, respectively. That is, we concatenate two bases of $\mathbb{R}^{64}$, since $\mathbf{b} \in \mathbb{R}^{64}$. We also consider bases for $\mathbb{R}^{64}$ built from tensor products of the one-dimensional waveforms constructed analogously to the columns of DCT_1 and Haar_1. This allows us to capture the two-dimensional nature of an image.

1.4 Outline

In Section 2 we relate our approach to analyze Equation (1) with classical sampling theory. Then, in Section 3 we give the necessary background on OMP for our use in image compression.

In Section 4 we begin by defining the image database, see Section 4.1, and review elementary image representation concepts, see Section 4.2. Then, in Section 4.3, we give an overview of image compression by sparsity and define the various vectorizations that we use. Finally, in Section 4.4, we provide the details for the various matrices $\mathbf{A}$ that we implement.

We shall measure the quality of the reconstruction from compressed representations with the *peak signal-to-noise ratio* (PSNR) [38], the *structural similarity index* (SSIM), and the *mean structural similarity index* (MSSIM) [42], all as functions of the tolerance ϵ chosen when solving (P_0^ϵ). These concepts are defined in Section 5.

Sections 6, 7, and 8 contain our results in terms of the phenomenological and computational theme mentioned in Section 1.1. Section 8 is a recapitulation of all our techniques and goes back to our sampling point of view in Section 2. In particular, we frame compressed sensing with sampling, and introduce *deterministic sampling masks*, see Section 8.2. We then perform image reconstruction with them, and do error analysis on the results, see Section 8.3. It remains to interleaved our results into the structure of general transmission/storage systems in the context of the information-theoretical paradigms formulated by Shannon [36].

Section 9 deals with the aforementioned transmission/storage process in terms of quantization, rate, and distortion. We frame our approach to image compression and representation in the setting of transform encoding, and obtain upper bounds on distortion.

2 Sampling theory and $\mathbf{Ax} = \mathbf{b}$

The Classical Sampling Formula,

$$f(t) = \sum_j \; Tf(jT) \; s(t - jT), \tag{3}$$

was essentially formulated by Cauchy in the 1840s, and, even earlier, Lagrange had formulated similar ideas. The hypotheses of Equation (3) are the following: $f \in L^2(\mathbb{R})$, the support of its Fourier transform, $\hat{f}$, is contained in $[-\Omega, \Omega]$, where $0 < 2T\Omega \leq 1$, and the sampling function s has the properties that $\hat{s} \in L^2(\widehat{\mathbb{R}}) \cap L^\infty(\widehat{\mathbb{R}})$ and $\hat{s} = 1$ on $[-\Omega, \Omega]$. Here, $\widehat{\mathbb{R}} = \mathbb{R}$ is considered as the domain of the Fourier transform. One can then prove that Equation (3) is valid in L^2-norm and uniformly on $\mathbb{R}$, see [3], Section 3.10. Cauchy's statement was not quite as general, but the idea of periodization which underlies the success of uniform sampling through the years was the same.

Important generalizations, modifications, refinements, and applications of Equation (3) abound. We shall describe how the equation, $\mathbf{Ax} = \mathbf{b}$, fits into this context.

A major extension of Equation (3) is to the case,

$$f(t) = \sum_j \; f(t_j) \; s_j(t), \tag{4}$$

where $\{t_j\}$ is a non-uniformly spaced sequence of sampled values of f. Because of the non-uniform spacing, the proofs of such formulas can not use periodization methods. In fact, the theory surrounding Equation (4) uses deep ideas such as balayage, sophisticated density criteria, and the theory of frames, see, e.g., the work of Duffin and Schaefer [17], Beurling [7], Beurling and Malliavin [8, 9], H. J. Landau [25], Jaffard [24] and Seip [35]. A natural extension of the non-uniform sampling Equation (4) is to the case of Gabor expansions,

$$f(t) = \sum_j \; \langle f, \tau_{t_j} e_{\sigma_j} g \rangle \; S^{-1}(\tau_{t_j} e_{\sigma_j} g), \tag{5}$$

that are a staple in modern time-frequency analysis, see, e.g., the authoritative [20] and the new balayage dependent result, [1]. In Equation (5), g is a window function used in the definition of the Short Time Fourier Transform, τ_t denotes translation by t, e_σ is modulation by σ, and S^{-1} is an appropriate inverse frame operator.

It turns out there is a useful Gabor matrix formulation of Equation (5) in the context of $\mathbf{Ax} = \mathbf{b}$, see [21, 30].

A discretization of Equation (3), for functions $f : \mathbb{Z} \to \mathbb{C}$, could take the form,

$$f(k) = \sum_{j \in J \subseteq \mathbb{Z}} f(j) a_j(k), \tag{6}$$

for all $k \in \mathbb{Z}$, and a given sequence $\{a_j\}$ of sampling functions a_j corresponding to the sampling functions $a_j = \tau_{jT} s$ of Equation (3). A similar discretization can be formulated for Equation (4). In both cases, the set J is the domain of sampled values of f from which f is to be characterized as in Equation (6).

Equation (6) can be generalized for a given sequence of sampling functions a_j to have the form,

$$f(k) = \sum_{j \in J \subseteq \mathbb{Z}} x(j) a_j(k), \tag{7}$$

where $x(j) = Kf(j)$ for some operator $K : L(\mathbb{Z}) \to L(\mathbb{Z})$, e.g., an averaging operator ($L(\mathbb{Z})$ designates a space of functions defined on $\mathbb{Z}$).

To make the tie-in with Equation (1), we write $\mathbf{b} = f$ and consider $\mathbb{Z}/n\mathbb{Z}$ instead of $\mathbb{Z}$; in particular, we are given $\mathbf{b} = (b(0), \ldots, b(n-1))^{\mathrm{T}} \in L(\mathbb{Z}/n\mathbb{Z})$, the space of all functions defined on $\mathbb{Z}/n\mathbb{Z}$, and $K : L(\mathbb{Z}/nz) \to L(\mathbb{Z}/m\mathbb{Z})$. Consequently, for $\#(J) \leq m$, we can define the sampling formula,

$$\forall k \in \mathbb{Z}/n\mathbb{Z}, \quad b(k) = \sum_{j \in J \subseteq \mathbb{Z}/m\mathbb{Z}} x(j) a_j(k), \tag{8}$$

where, $\mathbf{x} = (x(0), \ldots, x(m-1))^{\mathrm{T}} \in L(\mathbb{Z}/m\mathbb{Z})$, and where we would like $\|\mathbf{x}\|_0 < n \leq m$. In this format, the sampling formula Equation (8), is precisely of the form $\mathbf{Ax} = \mathbf{b}$, and the condition $\|\mathbf{x}\|_0 < n$ is a desired sparsity constraint. In fact, $\mathbf{A} = (\mathbf{a}_0 | \ldots | \mathbf{a}_{m-1})$, where each column vector $\mathbf{a}_j = (a_j(0), \ldots, a_j(n-1))^{\mathrm{T}} \in \mathbb{C}^n = L(\mathbb{Z}/n\mathbb{Z})$.

3 Finding sparse solutions with orthogonal matching pursuit

We give a brief description of the orthogonal matching pursuit (OMP) algorithm in the setting of solving (P_0^ϵ). Starting from $\mathbf{x}^0 = \mathbf{0}$, a greedy strategy iteratively constructs a k-term approximation $\mathbf{x}^k$ by maintaining a set of active columns—initially empty—and, at each stage, expanding that set by one additional column. The column chosen at each stage maximally reduces the residual ℓ^2 error in approximating $\mathbf{b}$ from the current set of active columns. After constructing an approximation including the new column, the residual error ℓ^2 is evaluated. When it falls below a specified threshold $\epsilon > 0$, the algorithm terminates. Technically, we proceed as follows.

Task: Find the solution to $(P_0^\epsilon) : \min_{\mathbf{x}} \ \|\mathbf{x}\|_0$ subject to $\|\mathbf{Ax} - \mathbf{b}\|_2 < \epsilon$.

Parameters: Given $\mathbf{A}$, $\mathbf{b}$, and ϵ.

Initialization: Initialize $k = 0$, and set the following:

- The initial solution $\mathbf{x}^0 = \mathbf{0}$;
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$;
- The initial solution support $\mathcal{S}^0 = \mathit{Support}\{\mathbf{x}^0\} = \emptyset$.

Main Iteration: Increment k by 1 and perform the following steps:

- **Sweep** – Compute the errors $\epsilon(j) = \min_{z_j} \|z_j \mathbf{a}_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$;
- **Update Support** – Find a minimizer j_0 of $\epsilon(j)$, i.e., for all $j \notin \mathcal{S}^{k-1}$, $\epsilon(j_0) \leq \epsilon(j)$, and then update, i.e., set $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$;
- **Update Provisional Solution** – Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{Ax} - \mathbf{b}\|_2^2$, subject to $\mathit{Support}\{\mathbf{x}\} = \mathcal{S}^k$;
- **Update Residual** – Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$;
- **Stopping Rule** – If $\|\mathbf{r}^k\|_2 < \epsilon$, stop; otherwise, apply another iteration.

Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

This algorithm is known in the signal processing literature as *orthogonal matching pursuit* (OMP) [22, 16, 29, 27, 11], and it is the algorithm we have implemented, validated, and used throughout to find sparse solutions of (P_0^ϵ).

The notion of *mutual coherence* gives us a simple criterion by which we can test when a solution of Equation (1) is the unique sparsest solution. In what follows, we assume that $\mathbf{A} \in \mathbb{R}^{n\times m}$, $n < m$, and $\operatorname{rank}(\mathbf{A}) = n$.

Definition 1. The *mutual coherence* of a given matrix $\mathbf{A}$ is the largest absolute normalized inner product between different columns from $\mathbf{A}$. Thus, denoting the k-th column in $\mathbf{A}$ by $\mathbf{a}_k$, the mutual coherence of $\mathbf{A}$ is given by

$$\mu(\mathbf{A}) = \max_{1\le j,k\le m,\ j\ne k} \frac{|\mathbf{a}_j^T \mathbf{a}_k|}{\|\mathbf{a}_j\|_2 \cdot \|\mathbf{a}_k\|_2}. \tag{9}$$

Theorem 1. *If $\mathbf{x}$ solves the system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\|\mathbf{x}\|_0 < \frac{1}{2}\big(1 + 1/\mu(\mathbf{A})\big)$, then $\mathbf{x}$ is the sparsest solution. Hence, if $\mathbf{y} \ne \mathbf{x}$ also solves the system, then $\|\mathbf{x}\|_0 < \|\mathbf{y}\|_0$.*

This same criterion can be used to test when OMP will find the sparsest solution.

Theorem 2. *For a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, if a solution $\mathbf{x}$ exists obeying $\|\mathbf{x}\|_0 < \frac{1}{2}\big(1 + 1/\mu(\mathbf{A})\big)$, then an OMP run with threshold parameter $\epsilon = 0$ is guaranteed to find $\mathbf{x}$ exactly.*

The proofs of these theorems can be found in [11].

4 Image representation and compression

4.1 Image database

To carry out our experiments and test image compression via sparsity, as well as the properties of the matrices described in Section 4.4 for that purpose, we selected 4 natural images, 3 of them from the University of Southern California's Signal & Image Processing Institute (USC-SIPI) image database [40]. This database has been widely used for image processing benchmarking. The images are shown in Figure 1. All images are 512 × 512, 8-bit grayscale images, which means they are composed of $512^2 = 262{,}144$ pixels that can take integer values from 0 (black) to 255 (white).

4.2 Image representation concepts

For our purposes an image is a two dimensional sequence of sample values,

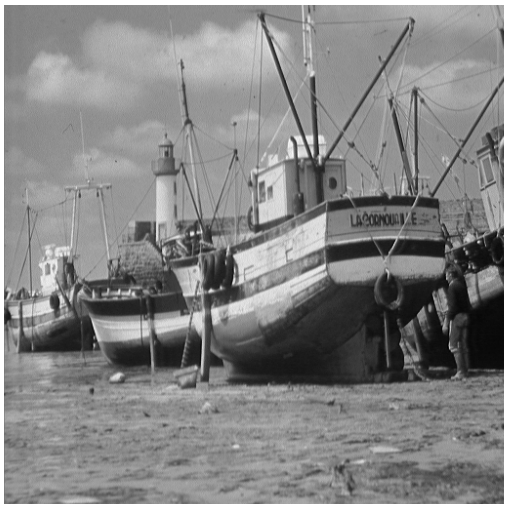
$$\mathcal{I}[n_1, n_2], \qquad 0 \le n_1 < N_1, \qquad 0 \le n_2 < N_2,$$

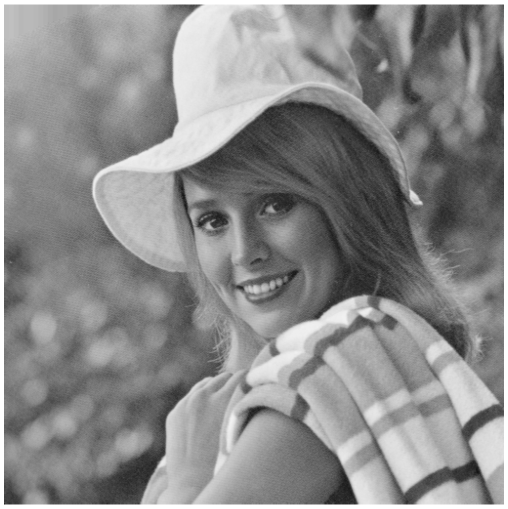
having finite extents, N_1 and N_2, in the vertical and horizontal directions, respectively. The term *pixel* is synonymous with an image sample value. The first coordinate, n_1, is the row index and the second coordinate, n_2, is the column index of the pixel. The ordering of the pixels follows the canonical ordering of a matrix's rows and columns, e.g., [38].

The sample value, $\mathcal{I}[n_1, n_2]$, represents the intensity (brightness) of the image at location $[n_1, n_2]$. The sample values will be B-bit signed or unsigned integers. Thus, we have

$$\mathcal{I}[n_1, n_2] \in \{0, 1, \ldots, 2^B - 1\} \quad \text{for unsigned imagery,}$$
$$\mathcal{I}[n_1, n_2] \in \{-2^{B-1}, -2^{B-1} + 1, \ldots, 2^{B-1} - 1\} \quad \text{for signed imagery.}$$

(a) Barbara


(b) Boat


(c) Elaine

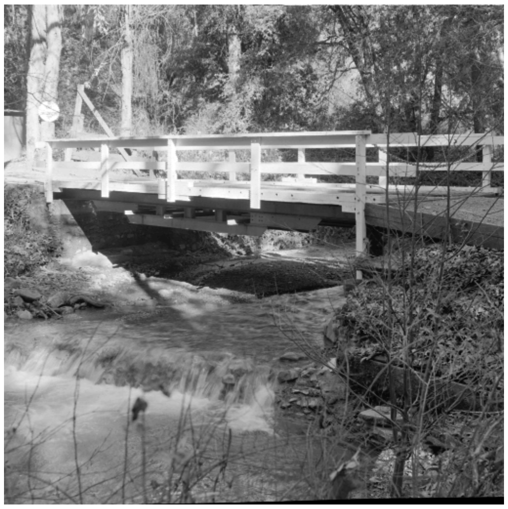

(d) Stream

Figure 1: Images used to test our compression algorithms based on sparse image representation.

In many cases, the B-bit sample values are interpreted as uniformly quantized representations of real-valued quantities, $\mathcal{I}'[n_1, n_2]$, in the range 0 to 1 (unsigned) or $-\frac{1}{2}$ to $\frac{1}{2}$ (signed). Letting $round(\cdot)$ denote rounding to the nearest integer, the relationship between the real-valued and integer sample values may be written as

$$\mathcal{I}[n_1, n_2] = round(2^B \mathcal{I}'[n_1, n_2]). \tag{10}$$

This accounts for the sampling quantization error which is introduced by rounding the physically measured brightness at location $[n_1, n_2]$ on a light sensor to one of the allowed pixel values, e.g., [38, 44].

We shall use this framework to represent grayscale images, where a pixel value of 0 will represent "black" and a value of $2^B - 1$ will represent "white". The value of B is called the *depth* of the image, and typical values for B are 8, 10, 12, and 16. Color images are represented by either three values per sample, $\mathcal{I}_R[n_1, n_2]$, $\mathcal{I}_G[n_1, n_2]$, and $\mathcal{I}_B[n_1, n_2]$ each for the red, green, and blue channels, respectively; other times by luminance $\mathcal{I}_Y[n_1, n_2]$, blue $\mathcal{I}_{C_b}[n_1, n_2]$, and red $\mathcal{I}_{C_r}[n_1, n_2]$ chrominance; or by four values per pixel, $\mathcal{I}_C[n_1, n_2]$, $\mathcal{I}_M[n_1, n_2]$, $\mathcal{I}_Y[n_1, n_2]$, and $\mathcal{I}_K[n_1, n_2]$, each for the cyan, magenta, yellow, and black channels commonly used in applications for color printing, respectively. We shall restrict ourselves to grayscale images given that it is always possible to apply a compression system separately to each component in turn, e.g., [2, 38].

4.3 Image compression via sparsity

4.3.1 Setup

In this section, we give an overview of image compression via sparsity. The basic idea is that if $\mathbf{Ax} = \mathbf{b}$, $\mathbf{b}$ is dense—that is, it has mostly nonzero entries—and $\mathbf{x}$ is sparse, then we can achieve compression by storing wisely $\mathbf{x}$ instead of $\mathbf{b}$.

Specifically, suppose we have a signal $\mathbf{b} \in \mathbb{R}^n$ that requires n numbers for its description. However, if we can solve problem (P_0^ϵ), whose solution $\mathbf{x}_0^\epsilon$ has k nonzero entries, with $k \ll n$, then we shall have obtained an approximation $\hat{\mathbf{b}} = \mathbf{A}\mathbf{x}_0^\epsilon$ to $\mathbf{b}$ using k scalars, with an approximation error of at most ϵ. Thus, by increasing ϵ we can obtain better compression at the expense of a larger approximation error. We shall characterize this relationship between error and compression, or equivalently, error and bits per sample, in Section 7.1.

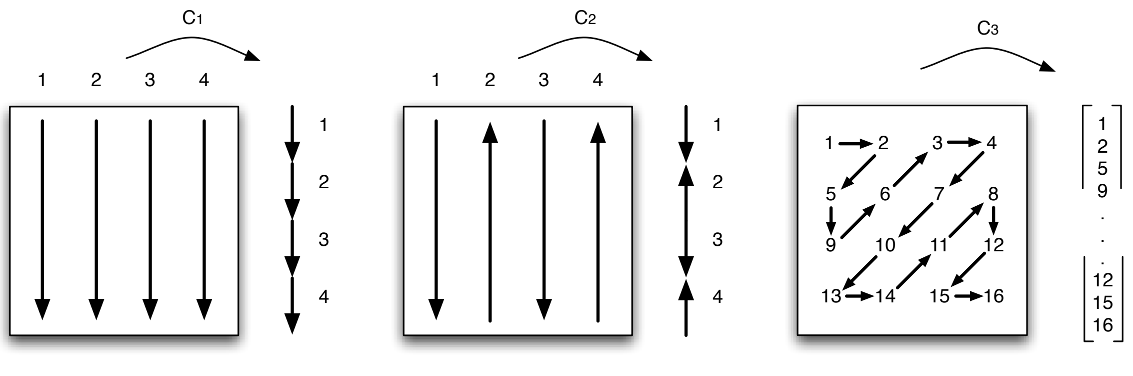
4.3.2 From an image to a vector to an image

Following the approach to image processing at the core of the JPEG image compression standard [41, 2], we subdivide each image in our database into 8×8 non-overlapping squares that will be treated individually. Since we need to generate a right hand side vector $\mathbf{b}$ to implement our compression scheme via sparsity, cf. Section 4.3.1, a sub-image $\mathbf{Y} \in \mathbb{R}^{8\times 8}$ of size 8×8 pixels needs to be vectorized into a vector $\mathbf{y} \in \mathbb{R}^{64}$ to play the role of $\mathbf{b}$. There are many ways to do this, and we tested three possible approaches.

The first consists of concatenating one after the other the columns of $\mathbf{Y}$ to form $\mathbf{y}$, and we shall call this method c_1. It can be thought of as a bijection $c_1 : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$ that maps $\mathbf{Y} \mapsto \mathbf{y}$, see Figure 2(a).

The second approach is to reverse the ordering of the entries of every even column of $\mathbf{Y}$ and then concatenate the columns of the resulting matrix. We shall call this method c_2. It is also a bijection $c_2 : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$, see Figure 2(b).

Finally, a third method, $c_3 : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$, traverses an 8×8 sub-image $\mathbf{Y}$ in a zigzag pattern from left to right, see Figure 2(c). It too is a bijection.



(a) (b) (c)

Figure 2: Three ways to vectorize a 4×4 matrix. (a) Concatenate the columns of the matrix from top to bottom one after the other, from left to right; or (b) first flip every other column, and then concatenate the columns as before; or (c) traverse the elements of the matrix in a Cantor-Peano zigzag fashion. We have shown 4×4 instead of 8×8 for ease of illustration.

We must still designate a matrix $\mathbf{A} \in \mathbb{R}^{64\times 128}$ to complete the setup. We shall address this issue in Section 4.4. For now, assume $\mathbf{A}$ is given.

Once we have chosen c_i and $\mathbf{A}$, we proceed in the following way. Given a tolerance $\epsilon > 0$, and an image $\mathcal{I}$ that has been partitioned into 8×8 non-overlapping sub-images $\mathbf{Y}_l$, where $l = 1, \ldots, M$ and M is the number of sub-images partitioning $\mathcal{I}$, we obtain the approximation to $\mathbf{y}_l = c_i(\mathbf{Y}_l)$ derived from the OMP algorithm, i.e., from the sparse vector $\mathbf{x}_l = \mathtt{OMP}(\mathbf{A}, \mathbf{y}_l, \epsilon)$, we compute $\tilde{\mathbf{y}}_l = \mathbf{A}\mathbf{x}_l$. Using $\tilde{\mathbf{y}}_l$ we can reconstruct a sub-image by setting $\widetilde{\mathbf{Y}}_l = c_i^{-1}(\tilde{\mathbf{y}}_l)$.

Finally, we rebuild and approximate the original image $\mathcal{I}$ by pasting together, in the right order and position, the set $\{\widetilde{\mathbf{Y}}_l\}$ of sub-images and form the approximate image reconstruction $\widetilde{\mathcal{I}}$ of $\mathcal{I}$. This new image $\widetilde{\mathcal{I}}$ is an approximation, and not necessarily the original image $\mathcal{I}$, because in the process we have introduced an error by setting the tolerance $\epsilon > 0$, and not $\epsilon = 0$. On the other hand, we do have $\|\tilde{\mathbf{y}} - \mathbf{y}\|_2 < \epsilon$. Since $\|\mathbf{x}_l\|_0 \leq \|\mathbf{y}_l\|_0$, and more likely $\|\mathbf{x}_l\|_0 \ll \|\mathbf{y}_l\|_0$, storing the set $\{\mathbf{x}_l\}_{l=1}^M$ wisely will provide a compressed representation of the image $\mathcal{I}$.

The means to create efficiently a compressed representation of the image $\mathcal{I}$ by using $\{\mathbf{x}_l\}_{l=1}^M$, the map $c_i : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$, and the matrix $\mathbf{A}$, and analyzing the effects of the choice of the tolerance ϵ on such a representation will be addressed in subsequent sections.

4.4 Choosing a matrix A

4.4.1 Setup

The choice of matrix $\mathbf{A}$ is clearly central to our approach to compression. Because of the JPEG and JPEG 2000 standards that use at their core the discrete cosine transform (DCT) and a wavelet transform [41, 2, 38], respectively, we shall incorporate both transforms in our choices of $\mathbf{A}$.

4.4.2 One-dimensional basis elements

Given that the signal that we are going to process comes in the form of a vector $\mathbf{b}$, an inherently one-dimensional object, a first approach is to consider the one-dimensional DCT waveforms, and any one-dimensional wavelet basis for $L^2[0,1]$. For the choice of the wavelet basis we opt for the

Haar wavelet and its scaling function, see Equations (12) and (13).

More specifically, we know that the one-dimensional DCT-II transform [10, 32],

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \qquad k=0,\ldots,N-1, \tag{11}$$

has at its core a sampling of the function $f_{k,N}(x) = \cos\left(\pi\left(x+\frac{1}{2N}\right)k\right)$ on the regularly spaced set $\mathcal{S}(N) = \left\{s_i \in [0\ 1) : s_i = \frac{i}{N},\, i=0,\ldots,N-1\right\}$ of points. We define the vector $\mathbf{w}_{k,N} = \sqrt{2}^{\operatorname{sgn}(k)} \cdot (f_{k,N}(s_0),\ldots,f_{k,N}(s_{N-1}))^{\mathrm{T}} \in \mathbb{R}^N$, which we generically call a *DCT waveform of wave number k and length N*. We shall use DCT waveforms with $N = 64$ for the one-dimensional compression approach. This is because we subdivide each image in our database into collections of 8×8 non-overlapping sub-images, which are then transformed into vectors $\mathbf{b} \in \mathbb{R}^{64}$, and subsequently compressed, as described in Section 4.3.2. This collection of DCT waveforms is a basis for $\mathbb{R}^{64}$, and we arrange its elements column-wise in matrix form as $\mathrm{DCT}_1 = (\mathbf{w}_{0,64}\ldots\mathbf{w}_{63,64}) \in \mathbb{R}^{64\times 64}$. Note that all column vectors of DCT_1 have the same ℓ^2 norm.

The corresponding basis for $\mathbb{R}^{64}$ based on the Haar wavelet is built in the following way. Consider the Haar wavelet [26, 33],

$$\psi(x) = \begin{cases} 1 & \text{if } 0 \le x < 1/2, \\ -1 & \text{if } 1/2 \le x < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

and its scaling function,

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \le x < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

For each $n \ge 0$, define the set H_n of functions $\psi_{n,k}(x) = 2^{n/2}\psi(2^n x - k)$, $0 \le k < 2^n$, and define $H_{-1} = \{\phi\}$. Note that $\#H_n = 2^n$ for $n \ge 0$ and $\#H_{-1} = 1$; and therefore $\#\bigcup_{j=-1}^{n} H_j = 1 + \sum_{j=0}^{n} 2^j = 2^{n+1}$. Since $64 = 2^6$, a value of $n = 5$ will produce 64 functions to choose from in $\mathcal{H}(n) = \bigcup_{j=-1}^{n} H_j$. For each function $h \in \mathcal{H}(5)$, define the vector $\mathbf{v}_{h,64} \in \mathbb{R}^{64}$ by sampling h on the set $\mathcal{S}(64)$, i.e., $\mathbf{v}_{h,64} = (h(s_0),\ldots,h(s_{63}))^{\mathrm{T}}$. Observe that $\|\mathbf{v}_{h,64}\|_2 = \|\mathbf{w}_{0,64}\|_2$ for all $h \in \mathcal{H}(5)$.

We can order the elements of $\mathcal{H}(5)$ in a natural way, viz., $h_0 = \phi$, $h_1 = \psi_{0,0}$, $h_2 = \psi_{1,0}$, $h_3 = \psi_{1,1}$, etc. It is easy to see that the set $\{\mathbf{v}_{h_j,64}\}_{j=0}^{63}$ of vectors is a basis for $\mathbb{R}^{64}$. As with the DCT waveforms, we arrange these vectors column-wise in matrix form as $\mathrm{Haar}_1 = (\mathbf{v}_{h_0,64}\ldots\mathbf{v}_{h_{63},64}) \in \mathbb{R}^{64\times 64}$.

Then, for the one-dimensional approach, we define $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1] \in \mathbb{R}^{64\times 128}$, the concatenation of both bases.

4.4.3 Two-dimensional basis elements

Another way to choose a basis for $\mathbb{R}^{64}$ results from taking into account the intrinsic two-dimensional nature of an image and to define basis elements that reflect this fact. Specifically, consider the two-dimesional DCT-II,

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{n1,n2} \cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\left[\frac{\pi}{N_2}\left(n_1+\frac{1}{2}\right)k_2\right], \tag{14}$$

used in the JPEG standard [41, 2]. Consider the family of functions g_{k_1,k_2,N_1,N_2} indexed by k_1 and k_2, and defined as

$$g_{k_1,k_2,N_1,N_2}(x,y) = \cos\left[\pi\left(x + \frac{1}{2N_1}\right)k_1\right]\cos\left[\pi\left(y + \frac{1}{2N_2}\right)k_2\right],$$

sampled on all points $(x, y) \in \mathcal{S}(N_1) \times \mathcal{S}(N_2)$. In our computations, we take $N_1 = N_2 = 8$ and $k_1, k_2 \in \{0, \ldots, 7\}$.

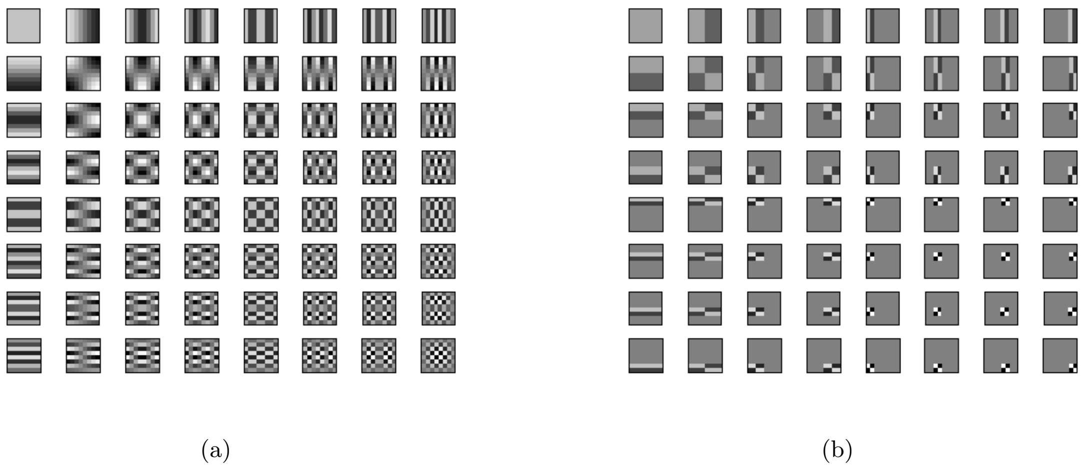


Figure 3: Natural representation for the 2-dimensional DCT (a) and Haar (b) bases for $\mathbb{R}^{64}$. White corresponds to the maximum value achieved by the basis element, black to the minimum. The intermediate shade of gray corresponds to 0.

Since $g_{k_1,k_2,N_1,N_2}(x,y) = f_{k_1,N_1}(x)f_{k_2,N_2}(y)$, the image of $\mathcal{S}(8) \times \mathcal{S}(8)$ under $g_{k_1,k_2,8,8}$ can be naturally identified with the tensor product,

$$\mathbf{w}_{k_1,8} \otimes \mathbf{w}_{k_2,8}, \qquad k_1, k_2 = 0, \ldots, 7, \tag{15}$$

modulo the constants $\sqrt{2}^{\operatorname{sgn}(k_1)}$ and $\sqrt{2}^{\operatorname{sgn}(k_2)}$, that make $\|\mathbf{w}_{k_1,8}\|_2 = \|\mathbf{w}_{k_2,8}\|_2$, respectively. See Figure 3(a) for a graphical representation of all of these tensor products.

There is a total of 64 such tensor products, and for each of them we can obtain a vector $\widetilde{\mathbf{w}}_{k_1,k_2,j} = c_j^{-1}(\mathbf{w}_{k_1,8} \otimes \mathbf{w}_{k_2,8})$; see Section 4.3.2. It is easy to see that the vector columns of the matrix,

$$\mathrm{DCT}_{2,j} = (\widetilde{\mathbf{w}}_{k_1,k_2,j}) \in \mathbb{R}^{64\times 64}, \qquad k_1, k_2 = 0, \ldots, 7, \tag{16}$$

form a basis for $\mathbb{R}^{64}$. The ordering of the column vectors of $\mathrm{DCT}_{2,j}$ follows the lexicographical order of the sequence of ordered pairs (k_1, k_2) for $k_1, k_2 = 0, \ldots, 7$ when k_2 moves faster than k_1, i.e., the ordering is (0,0), (0,1), …, (0,7), (1, 0), …, (7,7).

In a similar fashion, we can build $\mathrm{Haar}_{2,j}$. Specifically, first consider the set $\mathcal{H}(2)$, which contains 8 functions. Sampling $h_k \in \mathcal{H}(2)$ on $\mathcal{S}(8)$, we obtain the vector $\mathbf{v}_{h_k,8} \in \mathbb{R}^8$. Given $k_1, k_2 \in \{0, \ldots, 7\}$ we then compute the vector,

$$\widetilde{\mathbf{v}}_{k_1,k_2,j} = c_j^{-1}\left(\mathbf{v}_{h_{k_1},8} \otimes \mathbf{v}_{h_{k_2},8}\right) \in \mathbb{R}^{64},$$

with which, for all $k_1, k_2 \in \{0, \ldots, 7\}$ and following the same order for the ordered pairs (k_1, k_2) mentioned above, we can construct

$$\mathrm{Haar}_{2,j} = (\widetilde{\mathbf{v}}_{k_1,k_2,j}) \in \mathbb{R}^{64\times 64}, \qquad k_1, k_2 = 0, \ldots, 7. \tag{17}$$

For a visual representation of the tensor products $\mathbf{v}_{h_{k_1},8} \otimes \mathbf{v}_{h_{k_2},8}$, see Figure 3(b).

Finally, we define $\mathbf{A} = \mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}] \in \mathbf{R}^{64\times 128}$, the concatenation of both bases.

5 Imagery metrics

5.1 Compression and bit-rate

Following the methodology described in Section 4.3.2, suppose that we have an image $\mathcal{I}$ of size $N_1 \times N_2$ pixels ($N_1 = N_2 = 512$ in our case), and let $\{\mathbf{Y}_l\}_{l\in L}$ be a partition of $\mathcal{I}$ into $\#L$ 8×8 sub-images. Let $\mathbf{y}_l = c_i(\mathbf{Y}_l)$ for some $i = 1, 2, \text{or } 3$, where $c_i : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$, see Figure 2. Using OMP, with a full-rank matrix $\mathbf{A} \in \mathbb{R}^{64\times 128}$ and a tolerance $\epsilon > 0$, obtain $\mathbf{x}_l = \mathtt{OMP}(\mathbf{A}, \mathbf{y}_l, \epsilon)$, and compute the approximation to $\mathbf{y}_l$ given by $\tilde{\mathbf{y}}_l = \mathbf{A}\mathbf{x}_l$.

We know that $\|\tilde{\mathbf{y}}_l - \mathbf{y}_l\|_2 < \epsilon$, and we can count how many nonzero entries there are in $\mathbf{x}_l$, viz., $\|\mathbf{x}_l\|_0$. With this, we can define the *normalized sparse bit-rate.*

Definition 2 (Normalized Sparse Bit-Rate)**.** Given a matrix $\mathbf{A}$ and tolerance ϵ, the *normalized sparse bit-rate*, measured in bits per pixel (bpp), for the image $\mathcal{I}$ is the number,

$$nsbr(\mathcal{I}, \mathbf{A}, \epsilon) = \frac{\sum_l \|\mathbf{x}_l\|_0}{N_1 N_2}, \tag{18}$$

where the image $\mathcal{I}$ is of size $N_1 \times N_2$ pixels.

To interpret this definition, suppose a binary digit or "bit" represents a nonzero coordinate of the vector $\mathbf{x}_l$. Then we need at least $\|\mathbf{x}_l\|_0$ bits to store or transmit $\mathbf{x}_l$; and so the total number of bits needed to represent image $\mathcal{I}$ is at least $\sum_l \|\mathbf{x}_l\|_0$. Thus, the average number of bits per pixel (bpp) is obtained by dividing this quantity by $N_1 N_2$. This is the normalized sparse bit-rate.

Suppose $\mathcal{I}$ is of depth B, and that $\mathcal{I}$ can be represented by a string of bits (*bit-stream*) $\mathbf{c}$. An important objective of image compression is to assure that the length of $\mathbf{c}$, written *length*($\mathbf{c}$), is as small as possible. In the absence of any compression, we require $N_1 N_2 B$ bits to represent the image sample values, e.g. [38]. Thus, the notion of *compression ratio* is defined as

$$cr(\mathcal{I}, \mathbf{c}) = \frac{N_1 N_2 B}{length(\mathbf{c})}, \tag{19}$$

and the *compressed bit-rate*, expressed in bpp, is defined as

$$br(\mathcal{I}, \mathbf{c}) = \frac{length(\mathbf{c})}{N_1 N_2}. \tag{20}$$

The compression ratio is a dimensionless quantity that tells how many times we have managed to reduce in size the original representation of the image, while the compressed bit-rate has bpp units, and it tells how many bits are used on average per sample by the compressed bit-stream $\mathbf{c}$ to represent the original image $\mathcal{I}$.

We note from Equations (18) and (20) that it is likely that $nsbr(\mathcal{I}, \mathbf{A}, \epsilon) \leq br(\mathcal{I}, \mathbf{c})$ if the bit-stream $\mathbf{c}$ is derived from the sparse representation induced by $\mathbf{A}$ and ϵ. The rationale for this assertion is two-fold. First, it is unlikely that the coordinates in each of the resulting $\mathbf{x}_l$ vectors will be realistically represented by only one bit, and, second, because $\mathbf{c}$ would somehow have to include a coding for the indices l for each $\mathbf{x}_l$, necessarily increasing the bit count some more. These particular issues, i.e., the number of bits to represent the entries of vectors $\mathbf{x}_l$, and the coding of the indices l for each of those vectors into a final compressed bit-stream $\mathbf{c}$, will be addressed in Section 9.

5.2 Error estimation criteria

Let $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$ and consider the vectorization function c_2.

Given a 512×512 image $\mathcal{I}$ in our database, we can proceed to compress it using the methodology described in Section 4.3.2. If $\mathcal{I}$ is partitioned into 8×8 non-overlapping sub-images $\mathbf{Y}_l, l = 1, \ldots, 4096$, we obtain for each of them a corresponding reconstructed sub image $\widetilde{\mathbf{Y}}_l = c_2^{-1}(\tilde{\mathbf{y}}_l)$, and from those we reconstruct an approximation $\widetilde{\mathcal{I}}$ to $\mathcal{I}$. Here, $\tilde{\mathbf{y}}_l = \mathbf{A}\mathbf{x}_l$, $\mathbf{x}_l = \mathtt{OMP}(\mathbf{A}, \mathbf{y}_l, \epsilon)$, and $\mathbf{y}_l = c_2(\mathbf{Y}_l)$, as before. The compression would come from storing $\{\mathbf{x}_l\}$ efficiently. We summarize this procedure with the notation, $\widetilde{\mathcal{I}} = rec(\mathcal{I}, \mathbf{A}, \epsilon)$.

In order to assess the quality of $\widetilde{\mathcal{I}}$ when compared to $\mathcal{I}$, we introduce three error estimators.

Definition 3 (PSNR)**.** The *peak signal-to-noise ratio* between two images $\widetilde{\mathcal{I}}$ and $\mathcal{I}$ is the quantity,

$$\mathrm{PSNR}(\widetilde{\mathcal{I}}, \mathcal{I}) = 20 \log_{10} \left(\frac{\max_{\mathcal{I}}}{\sqrt{\mathrm{mse}(\widetilde{\mathcal{I}}, \mathcal{I})}} \right),$$

measured in dB, where $\max_{\mathcal{I}}$ is the maximum possible value for any given pixel in $\mathcal{I}$ (typically, $\max_{\mathcal{I}} = 2^B - 1$) and $\mathrm{mse}(\widetilde{\mathcal{I}}, \mathcal{I}) = \frac{1}{N_1 N_2} \sum_{i,j} \left(\widetilde{\mathcal{I}}[i,j] - \mathcal{I}[i,j] \right)^2$ is the mean square error between both images. Here N_1 and N_2 represent the dimensions of $\mathcal{I}$, and $\mathcal{I}[i,j]$ represents the value of the pixel at coordinates $[i,j]$ in image $\mathcal{I}$—similarly for $\widetilde{\mathcal{I}}[i,j]$. In our case $N_1 = N_2 = 512$, and $\max_{\mathcal{I}} = 255$.

PSNR [38] has the advantage that it is easy to compute and has widespread use, but it has been criticized for poorly correlating with perceived image quality [42, 43]. In recent years extensive work on other error estimators that take into account the human visual system have arisen. In particular, we define the *structural similarity* and *mean structural similarity* indices [42].

Definition 4 (SSIM)**.** Let $\widetilde{\mathcal{I}}$ and $\mathcal{I}$ be two images that have been decomposed in $L \times L$ non-overlapping sub-images, $\{\widetilde{\mathbf{Y}}_l\}$ and $\{\mathbf{Y}_l\}$, respectively. Then the *structural similarity* index for two corresponding sub-image vectorizations, say $\tilde{\mathbf{y}}_l = c_2(\widetilde{\mathbf{Y}}_l)$ and $\mathbf{y}_l = c_2(\mathbf{Y}_l)$, is defined as follows,

$$\mathrm{SSIM}(\tilde{\mathbf{y}}_l, \mathbf{y}_l) = \frac{(2\mu_{\tilde{\mathbf{y}}_l}\mu_{\mathbf{y}_l} + C_1)(2\sigma_{\tilde{\mathbf{y}}_l\mathbf{y}_l} + C_2)}{(\mu_{\tilde{\mathbf{y}}_l}^2 + \mu_{\mathbf{y}_l}^2 + C_1)(\sigma_{\tilde{\mathbf{y}}_l}^2 + \sigma_{\mathbf{y}_l}^2 + C_2)},$$

where $\mu_{\mathbf{y}_l}$ and $\sigma_{\mathbf{y}_l}$ represent the mean and standard deviation of $\mathbf{y}_l$, respectively, with a similar definition for $\tilde{\mathbf{y}}_l$. The term $\sigma_{\tilde{\mathbf{y}}_l\mathbf{y}_l}$ is the correlation between $\tilde{\mathbf{y}}_l$ and $\mathbf{y}_l$. The values C_1 and C_2 are two small constants.

For our purposes, we chose the default values of $L = 11$, $C_1 = 0.01$, and $C_2 = 0.03$ used in [42] when assessing the SSIM of an image in our database and its reconstruction. We used a value of $L = 4$ when we modified OMP to use internally the SSIM as a stopping criteria. More on this later.

From Definition 4, we can see that the SSIM index is a localized quality measure that can be represented on a plane that maps its values. It can take values from 0 to 1 and when it takes the value of 1 the two images are identical. In practice, we usually require a single overall quality of measure for the entire image. In that case we use the *mean structural similarity index* to evaluate the overall image quality, defined next.

Definition 5 (MSSIM)**.** Let $\widetilde{\mathcal{I}}$ and $\mathcal{I}$ be two images, where the former is the approximation and the later is the original. Then the *mean structural similarity index* is

$$\mathrm{MSSIM}(\widetilde{\mathcal{I}}, \mathcal{I}) = \frac{1}{M} \sum_{l=1}^{M} \mathrm{SSIM}(\tilde{\mathbf{y}}_l, \mathbf{y}_l),$$

where $\tilde{\mathbf{y}}_l$ and $\mathbf{y}_l$ are vectorizations of sub-images $\widetilde{\mathbf{Y}}_l$ and $\mathbf{Y}_l$, respectively. M is the number of sub-images.

Finally, we take a look at the relationship between the size of the sub-image and the tolerance, and how this affects the quality of the approximation. We analyze the idealized error distribution in which all pixels of the approximation are c units apart from the original. Consider an $L \times L$ sub image that has been linearized to a vector $\mathbf{y}$ of length L^2. Assume that the OMP approximation within ϵ has distributed the error evenly, that is, if $\mathbf{x} = \texttt{OMP}(\mathbf{A}, \mathbf{y}, \epsilon)$ and $\tilde{\mathbf{y}} = \mathbf{A}\mathbf{x}$, then

$$\begin{aligned} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 < \epsilon \Leftrightarrow & \|\tilde{\mathbf{y}} - \mathbf{y}\|_2^2 < \epsilon^2, \\ \Leftrightarrow & \sum_{j=1}^{L^2} \big(\tilde{\mathbf{y}}(j) - \mathbf{y}(j)\big)^2 < \epsilon^2, \\ \Leftrightarrow & L^2 c^2 < \epsilon^2, \\ \Leftrightarrow & c < \frac{\epsilon}{L}. \end{aligned} \tag{21}$$

That is, if we want to be within c units from each pixel, we have to choose a tolerance ϵ such that $c = \epsilon/L$.

We note that the least-squares approximation at the core of OMP approximates the idealized error distribution. This can be seen in Figure 4 where the black dashed line represents this idealized error approximation. For tolerances $\epsilon > 40$, we see that the PSNR for all images is greater than the idealized error distribution. This can be explained by noting that, for example, for $\epsilon = 2048$, we would have from Equation (21) that $c = 2048/8 = 256$, but the maximum pixel value is only 255. Therefore, unless the original image $\mathcal{I}$ is just a white patch, the initial value of the OMP approximation being an all black image, there are matching pixels in the original and the approximation image $\widetilde{\mathcal{I}} = rec(\mathcal{I}, \mathbf{A}, 2048)$ that are less than 256 units apart. By Definition 3, this would necessarily imply $\text{PSNR}(\widetilde{\mathcal{I}}, \mathcal{I}) > 0$, a value greater than the value of the PSNR for the idealized error distribution when $\epsilon = 2048$, which is a small negative value.

On the other hand, for small tolerances, say $\epsilon < 3$, we observe that the PSNR value for all images jumps again above the PSNR for the idealized error model. This is a happy case when roundoff error actually helps. What happens is that for such small tolerances, the roundoff to the closest integer for all entries in $\tilde{\mathbf{y}}_l = \mathbf{A}\mathbf{x}_l$ when we form the sub image approximation $\widetilde{\mathbf{Y}}_l = c_2^{-1}(\tilde{\mathbf{y}}_l)$, coincides with the true value of the pixels in the original sub image $\mathbf{Y}_l$. Again, by Definition 3, this increases the value of $\text{PSNR}(\widetilde{\mathcal{I}}, \mathcal{I})$ compared to the case where roundoff would not have taken place.

6 Effects of vectorization on image reconstruction

6.1 Setup

Given an image $\mathcal{I}$ in our database, we can follow and apply to it the methodology described in Section 4.3.2, and obtain at the end of this process a reconstructed image $\widetilde{\mathcal{I}}$ from it. In this section we explore the effects of the choice of map $c_i : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$ on the characteristics of image $\widetilde{\mathcal{I}}$ for the different choices of matrix $\mathbf{A}$ that we have selected to study.

6.2 Results for $\mathbf{A} = [\mathbf{DCT}_1\ \mathbf{Haar}_1]$

We set $\mathbf{A} = [\text{DCT}_1\ \text{Haar}_1]$, and choose a tolerance $\epsilon = 32$. Then, for each image $\mathcal{I}$ in our database and each index $i = 1, 2, 3$ we choose the map $c_i : \mathbb{R}^{8\times 8} \to \mathbb{R}^{64}$, and follow the methodology

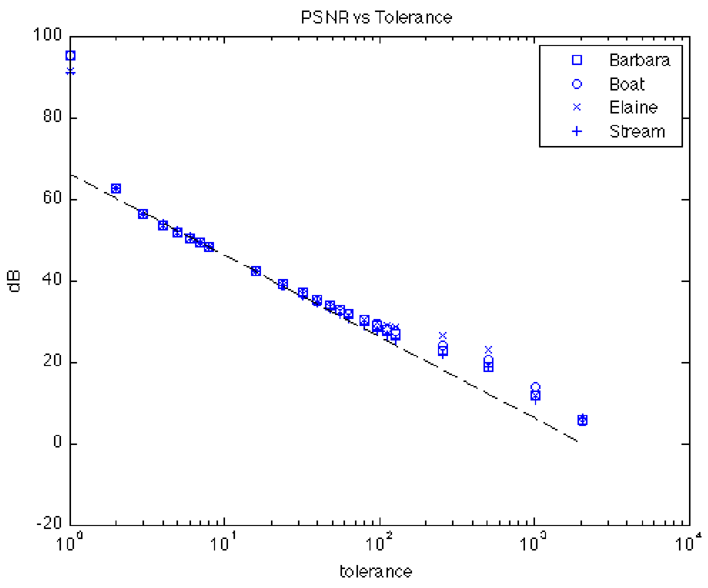


Figure 4: Peak Signal-to-Noise Ratio vs tolerance. We observe three typical behaviors for all images. For large values of the tolerance, about $\epsilon > 40$, the PSNR of all images is above the PSNR value for the idealized error distribution marked by the black dashed line. This behavior is also observed for very small values of the tolerance, about $\epsilon < 3$. For values between these two extreme behaviors, all images conform very closely to the idealized error distribution, a fact that is expected from the least-squares approximation at the core of the OMP algorithm.

described in Section 4.3.2.

The metrics that we use to measure the impact of the choice of map c_i are the *normalized sparse bit-rate*, and the *peak signal-to-noise ratio* (PSNR), see Definitions 2 and 3, respectively. A smaller value for the normalized sparse bit-rate is better than a larger one given that this implies fewer bits are necessary to represent the image. A larger value for the PSNR is better than a smaller one as this means the fidelity of the representation is higher. Table 1 summarizes the results of the experiments. From these, we conclude that for $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$, the choice of c_2 over c_1 or c_3 produces better results. This could be because, if $\mathbf{Y} = (Y_{i,j})_{i,j=1,\ldots,8}$ is a natural image, then on average $|Y_{8,j} - Y_{8,j+1}| < |Y_{8,j} - Y_{1,j+1}|$ for $j = 1, \ldots, 7$, which makes $\mathbf{y}_{c_2} = c_2(\mathbf{Y})$ change more slowly than $\mathbf{y}_{c_1} = c_1(\mathbf{Y})$ or $\mathbf{y}_{c_3} = c_3(\mathbf{Y})$. By analogy to the behavior of the DFT, this must translate into fewer column vectors from $\mathbf{A}$ to describe, within a certain error ϵ, the signal $\mathbf{y}_{c_2}$ compared to the number of columns needed to approximate the signals $\mathbf{y}_{c_1}$ or $\mathbf{y}_{c_3}$ to the same error tolerance.

6.3 Results for $\mathbf{A}_j = [\mathbf{DCT}_{2,j}\ \mathbf{Haar}_{2,j}]$

Proceeding similarly as in Section 6.2, we set $\mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}]$ for $j = 1, 2, \text{and } 3$, and we perform the following experiment.

Define the vectorization function to match the same ordering that was used to form $\mathbf{A}_j$. This means we pick the vectorization function to be c_j. We compute $\mathbf{y}_l = c_j(\mathbf{Y}_l)$, where, as before, the sub-image $\mathbf{Y}_l$ comes from the partition $\{\mathbf{Y}_l\}_{l \in L}$ of an image $\mathcal{I}$ in our image database. Then, continuing with the compression methodology described in Section 4.3.2, we obtain $\mathbf{x}_l = \mathtt{OMP}(\mathbf{A}, \mathbf{y}_l, \epsilon)$, setting $\epsilon = 32$ for this experiment. Finally, from the set of vectors $\{\mathbf{x}_l\}_{l \in L}$ we obtain $\tilde{\mathbf{y}}_l = \mathbf{A}\mathbf{x}_l$, and use $\{\tilde{\mathbf{y}}_l\}_{l \in L}$ to obtain the reconstructed image $\widetilde{\mathcal{I}}$ of our original image $\mathcal{I}$. Again, as in Section

Image/Function	PSNR (dB)	Normalized sparse bit-rate (bpp)
Barbara		
c_1:	36.8996	0.1833
c_2:	36.9952	0.1863
c_3:	36.8470	0.2338
Boat		
c_1:	36.5791	0.1812
c_2:	36.6020	0.1608
c_3:	36.5615	0.2205
Elaine		
c_1:	36.5003	0.1763
c_2:	36.5155	0.1682
c_3:	36.4877	0.1885
Stream		
c_1:	36.4423	0.3161
c_2:	36.4686	0.3050
c_3:	36.4400	0.3504

Table 1: Performance results for $\mathbf{A} = [\text{DCT}_1\ \text{Haar}_1]$ for c_1, c_2, and c_3. For each image in our test database, we vectorized each 8×8 sub-image using c_1, c_2, or c_3. In all cases, the PSNR value was larger using c_2; and in all cases, except for image *Barbara*, the normalized sparse bit-rate was smaller. Both of these measures make c_2 a better choice than c_1 or c_3. The results correspond to runs of OMP with an ℓ^2 stopping rule, and a tolerance $\epsilon = 32$.

6.2, we assess the effects of the choice of the vectorization function c_i by the values of PSNR and normalized sparse bit-rate resulting from this representation of $\mathcal{I}$ by $\widetilde{\mathcal{I}}$. We give a summary of the results of this experiment in Table 2.

We point out that choosing c_i, with $i \neq j$, when $\mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}]$, results in worse values of both PSNR and normalized sparse bit-rate than when $i = j$. We record only the results where $i = j$.

Moreover, any choice of $\mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}]$ with a matching vectorization function c_j performs better than when $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$ for the normalized sparse bit-rate metric, and better for the PSNR metric except for the image *Stream*. Also, on average, the vectorization order imposed by c_3 is slightly better than those by either c_1 or c_2, although the difference is practically imperceptible to the human eye. The normalized sparse bit-rate figures all coincide.

Image/Function	PSNR (dB)	Normalized sparse bit-rate (bpp)	Matrix
Barbara			
c_1 :	37.0442	0.1634	$[\mathrm{DCT}_{2,1}\ \mathrm{Haar}_{2,1}]$
c_2 :	37.0443	0.1634	$[\mathrm{DCT}_{2,2}\ \mathrm{Haar}_{2,2}]$
c_3 :	37.0443	0.1634	$[\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$
Boat			
c_1 :	36.6122	0.1541	$[\mathrm{DCT}_{2,1}\ \mathrm{Haar}_{2,1}]$
c_2 :	36.6120	0.1541	$[\mathrm{DCT}_{2,2}\ \mathrm{Haar}_{2,2}]$
c_3 :	36.6120	0.1541	$[\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$
Elaine			
c_1 :	36.5219	0.1609	$[\mathrm{DCT}_{2,1}\ \mathrm{Haar}_{2,1}]$
c_2 :	36.5219	0.1609	$[\mathrm{DCT}_{2,2}\ \mathrm{Haar}_{2,2}]$
c_3 :	36.5220	0.1609	$[\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$
Stream			
c_1 :	36.4678	0.2957	$[\mathrm{DCT}_{2,1}\ \mathrm{Haar}_{2,1}]$
c_2 :	36.4676	0.2957	$[\mathrm{DCT}_{2,2}\ \mathrm{Haar}_{2,2}]$
c_3 :	36.4677	0.2957	$[\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$

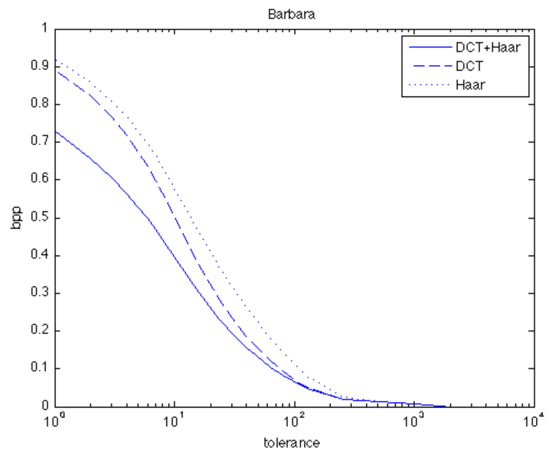
Table 2: Performance results for $\mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}], j = 1, 2, 3$, with corresponding vectorization functions c_1, c_2, and c_3. In all cases, the PSNR and normalized sparse bit-rate values were almost identical. Matrix $\mathbf{A}_3$ performs slightly better on average. Mismatching function c_i with matrix $\mathbf{A}_j = [\mathrm{DCT}_{2,j}\ \mathrm{Haar}_{2,j}]$, when $i \neq j$, results in degraded performance. The values correspond to runs of OMP with an ℓ^2 stopping rule, and a tolerance $\epsilon = 32$.

7 Comparisons between imagery metrics

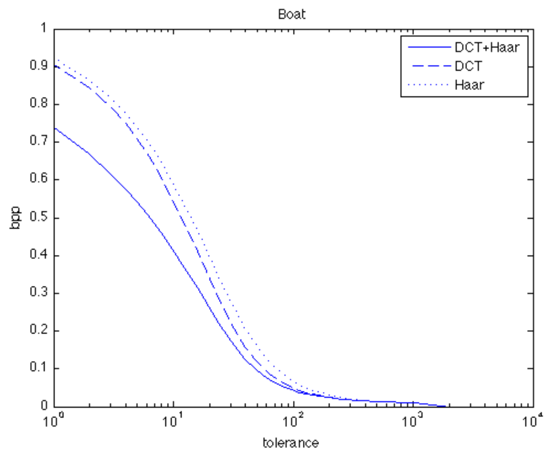
7.1 Normalized sparse bit-rate vs tolerance

Notwithstanding the remarks in Section 5.1, there is still value in using the normalized bit-stream measure to quantify and plot normalized sparse bit-rate vs tolerance graphs to gauge the compression properties of various compression matrices.

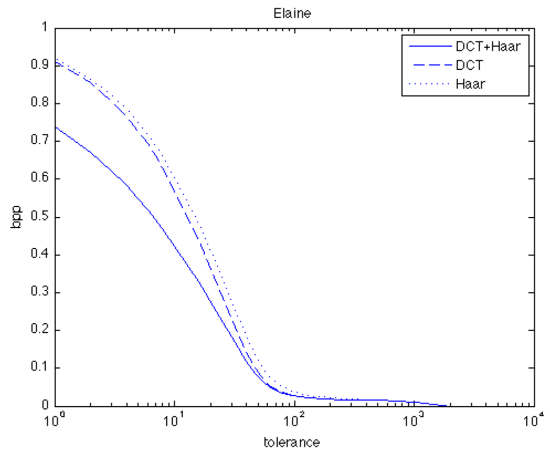
Given the results in Table 2, we study the compression properties of matrices $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$,



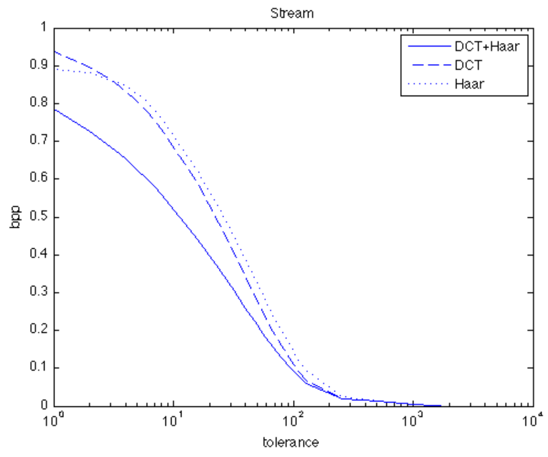
(a) Barbara



(b) Boat



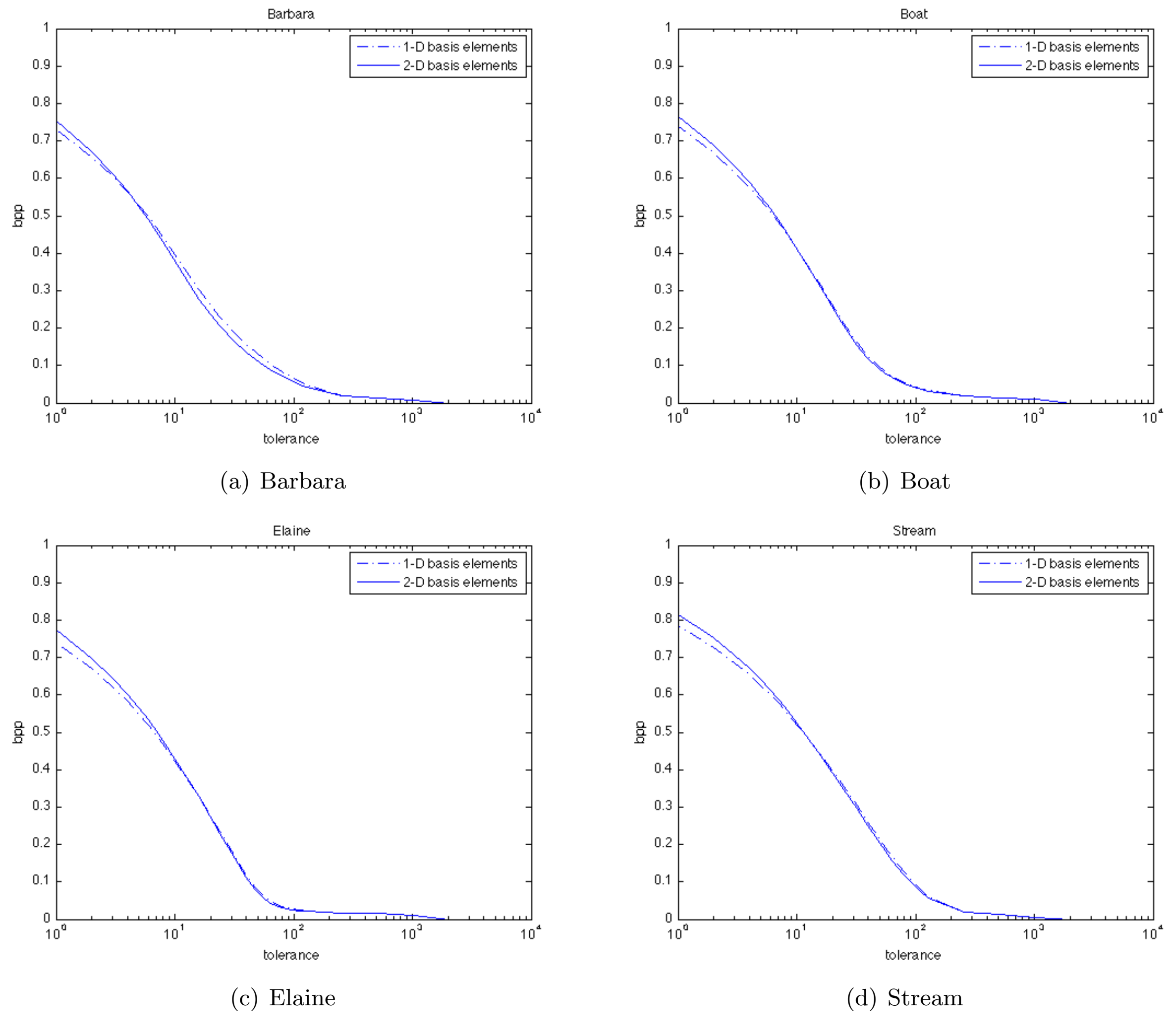
(c) Elaine



(d) Stream

Figure 5: Normalized sparse bit-rate vs tolerance: One-dimensional basis elements. We observe that for all images the best normalized sparse bit-rate for a given tolerance is obtained for matrix $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$ which combines both the DCT_1 and Haar_1 bases for $\mathbb{R}^{64}$.

and $\mathbf{A}_3 = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$. We compare these properties for both matrices relative to each other, and to the compression properties of $\mathbf{B}$ and $\mathbf{C}$, which are formed from the DCT_1 or the Haar_1 submatrices of matrix $\mathbf{A}$, respectively. We plot for all images in our database their respective normalized sparse bit-rate vs tolerance graphs. We let $\epsilon \in T = \{2^k\}_{k=0}^{11} \cup \{3, 5, 6, 7, 24, 40, 48, 56, 80, 96, 112\}$ and for each image $\mathcal{I}$ in our image database we obtained the corresponding normalized sparse bit-rates $nsbr(\mathcal{I}, \mathbf{A}, \epsilon)$, $nsbr(\mathcal{I}, \mathbf{B}, \epsilon)$, and $nsbr(\mathcal{I}, \mathbf{C}, \epsilon)$ to obtain the plots in Figure 5. In Figure 6 we compare $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$ with $\mathbf{A}_3 = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$. We observe that up to a tolerance $\epsilon_{\mathcal{I}}$, dependent on image $\mathcal{I}$, $\mathbf{A}_3$ performs better for tolerance values $\epsilon \geq \epsilon_{\mathcal{I}}$. That is, the value of the normalized sparse bit-rate is smaller when performing compression utilizing $\mathbf{A}_3$. For values of $\epsilon \leq \epsilon_{\mathcal{I}}$, compression with $\mathbf{A}$ results in better normalized sparse bit-rate values. We shall see in Section 7.2, with the aid of Figure 4, that for values of $\epsilon = 32$, and smaller, the quality of the image reconstruction is satisfactory. We note from Figure 6 that, for all images in our database, $\epsilon_{\mathcal{I}} < 32$. This means that, for most practical cases, the use of $[\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$ results in slightly smaller normalized sparse bit-rate values than when using $[\mathrm{DCT}_1\ \mathrm{Haar}_1]$.



(a) Barbara (b) Boat

(c) Elaine (d) Stream

Figure 6: Normalized sparse bit-rate vs tolerance: Comparison between $\mathbf{A} = [\mathrm{DCT}_1\ \mathrm{Haar}_1]$ and $\mathbf{A}_3 = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$.

From the results shown in Figure 5, we can see that the DCT_1 basis elements perform better compression for any given tolerance than when using the Haar_1 basis elements, except for image *Stream*. In fact, when the tolerance ϵ is close to but less than 3, the Haar_1 basis elements result in a smaller normalized sparse bit-rate value. Moreover, and more importantly, combining both the DCT_1 and Haar_1 bases results in better compression than if either basis is used alone. The same is true for $\text{DCT}_{2,3}$ and $\text{Haar}_{2,3}$.

In this light, there are natural questions dealing with and relating image reconstruction quality, range of effective tolerances, and error estimators.

7.2 PSNR vs MSSIM

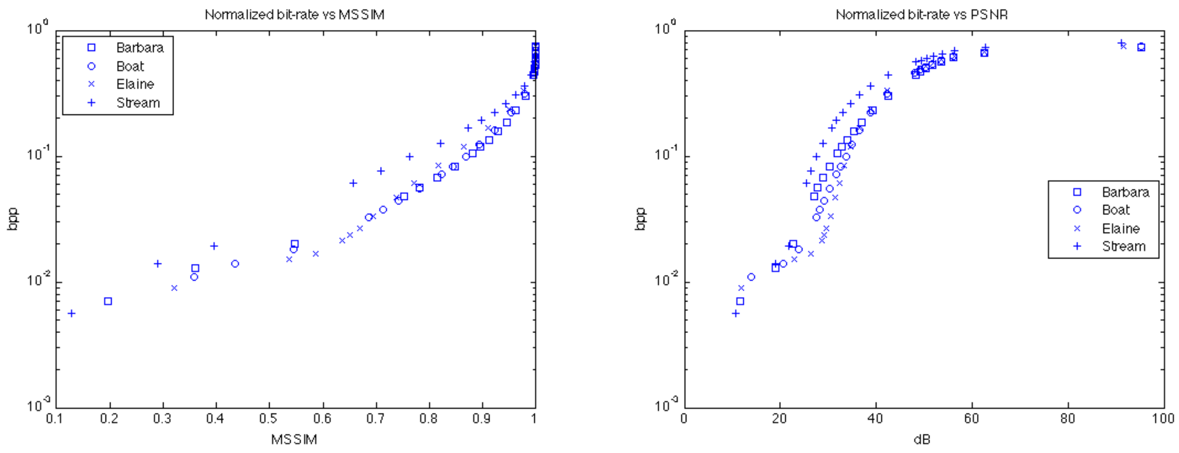


Figure 7: Normalized sparse bit-rate vs MSSIM, PSNR

In Figure 7 we have plotted the normalized sparse bit-rate versus both error indices MSSIM and PSNR. The first thing that we observe is that the sensitivity for PSNR varies more than the sensitivity for MSSIM over the range of tolerances chosen.

From Figure 8 we observe that, for the range of 20 to 40 dB in PSNR, the MSSIM index ranges from about 0.33 to 0.98. Since a value of 1 in MSSIM corresponds to two identical images, we can focus on values of PSNR no greater than 40 dB in our analysis. Also, in Figure 8, we corroborate the criticism that has been addressed to PSNR as a measure of image quality. For example, at 20 dB, the image *Stream* has an MSSIM value of 0.33, whereas the image *Elaine* has an MSSIM value of 0.48. Similarly, at 30 dB, the image *Elaine* has an MSSIM value of 0.69, whereas the image *Stream* has an MSSIM value of 0.86. It is not until 40 dB that we have a much smaller range of MSSIM values, viz., 0.96 (*Elaine*) to 0.98 (*Stream*). Therefore, if SSIM and MSSIM capture more accurately the human visual system's perception of image quality, then the PSNR index is not a reliable estimator until values larger than or equal to 35 dB.

Because of this observation about the PSNR index, we shall focus on the SSIM and MSSIM indices. We address the questions at the end of Section 7.1 and answer them with Figure 9. From this figure, if we were to consider desirable values of MSSIM to be greater than or equal to 0.9, we would see that this corresponds to a tolerance $\epsilon \leq 32$ for the image *Elaine* and $\epsilon \leq 48$ for the image *Stream*. All other tolerances for the other two images fall between these two values.

This means that if we wanted all images to have an MSSIM index of 0.9 or larger, we would have to pick a tolerance no larger than $\epsilon = 32$. According to Equation (21), this tolerance corresponds to a distance on average of no more than $32/8 = 4$ units per pixel between the reconstructed image and the original. Under these circumstances we would achieve a normalized sparse bit-rate of 0.160

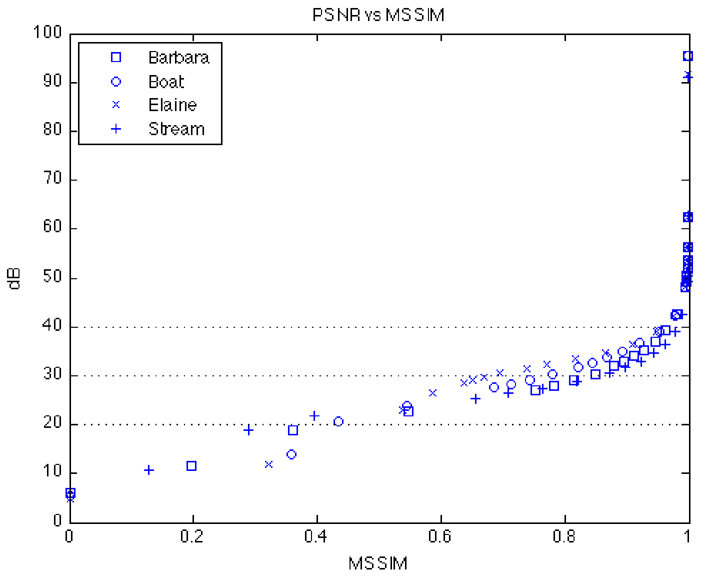


Figure 8: Peak Signal-to-Noise Ratio vs Mean Structural Similarity

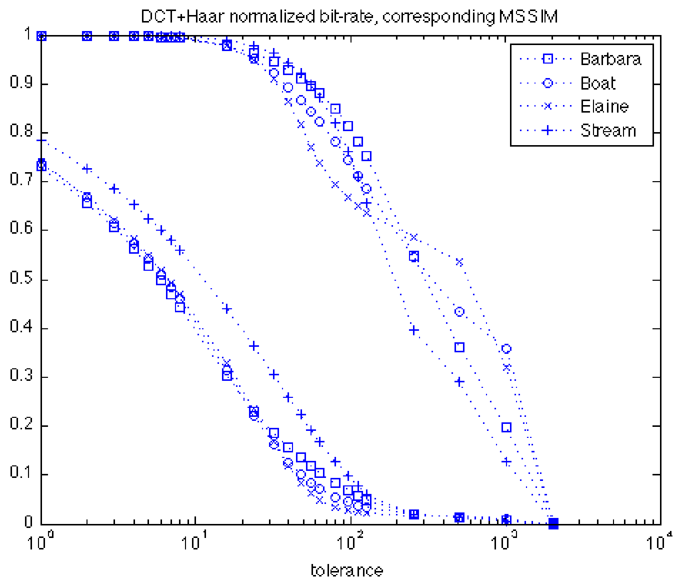


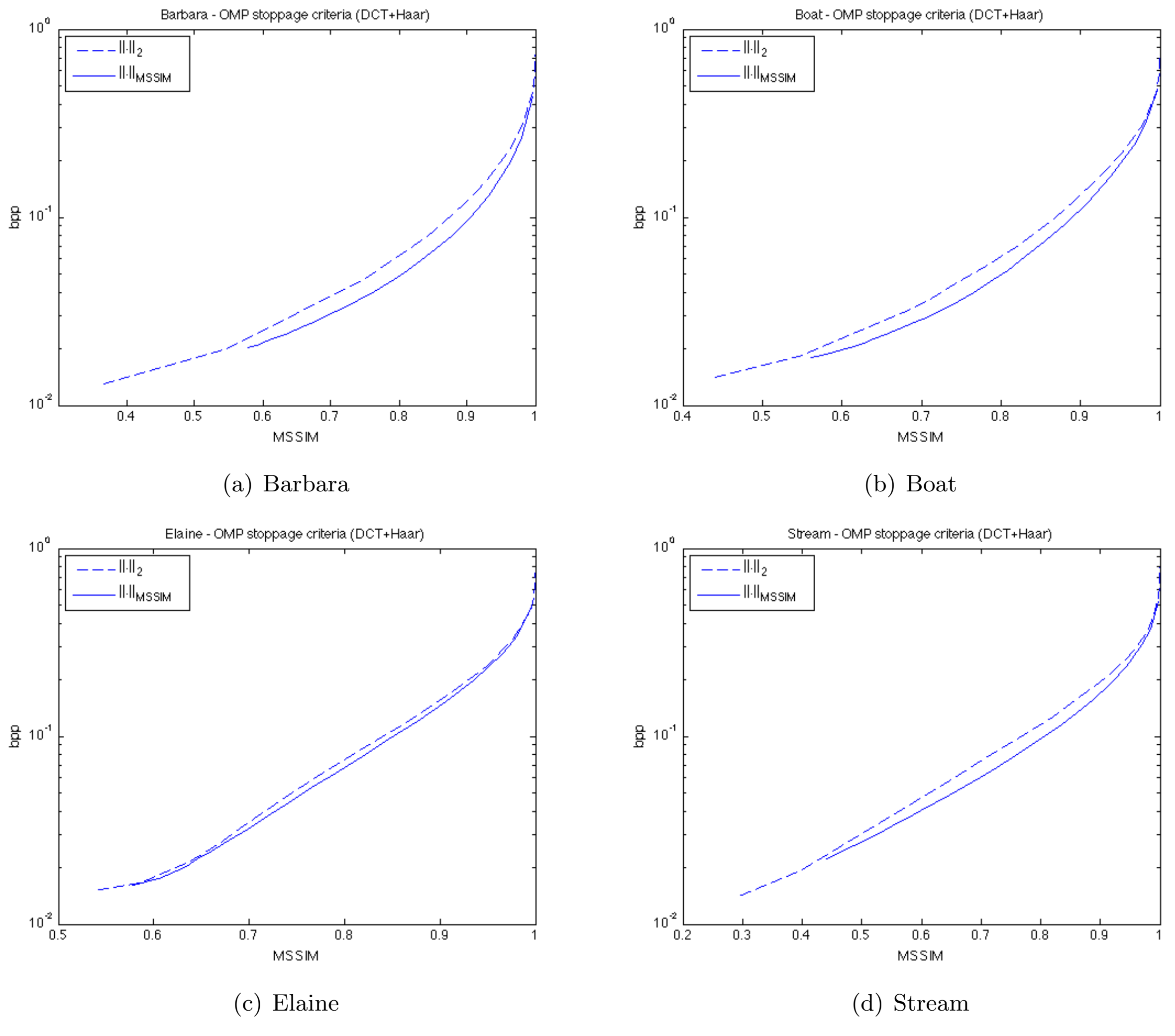
Figure 9: Normalized sparse bit-rate and corresponding MSSIM vs tolerance. In this graph we have plotted together the best normalized sparse bit-rate obtained by combining the DCT and Haar bases, and the corresponding value of the MSSIM index for a given tolerance. The normalized sparse bit-rate graphs are on the bottom left, and the MSSIM index values are above these.

to 0.305 bits per pixel. It is natural to ask if there is a modification of OMP which guarantees a certain minimum MSSIM quality level. It turns out that such a modification is possible.

Consider the following change to the stopping rule, $\|\mathbf{Ax}-\mathbf{b}\|_2 < \epsilon$, for the OMP algorithm:

$$\|\mathbf{Ax}-\mathbf{b}\|_{MSSIM} \equiv \mathrm{MSSIM}(c_2^{-1}(\mathbf{Ax}), c_2^{-1}(\mathbf{b})) > \delta_0,$$

where δ_0 is a desired minimum MSSIM index value to be achieved in each individual sub-image of the reconstruction of $\mathcal{I}$. When we make this change, and recompute the normalized sparse bit-rate vs MSSIM graphs, we obtain the plots shown in Figure 10. In this figure, we observe that changing the stopping rule for OMP leads to an improvement in the normalized sparse bit-rate without sacrificing image quality. To see this from the opposite perspective, given a normalized sparse bit-rate, we can achieve a better MSSIM image quality index when we use the new stopping criterion. In fact, this change redistributes the work that OMP performs more evenly across the image.



(a) Barbara

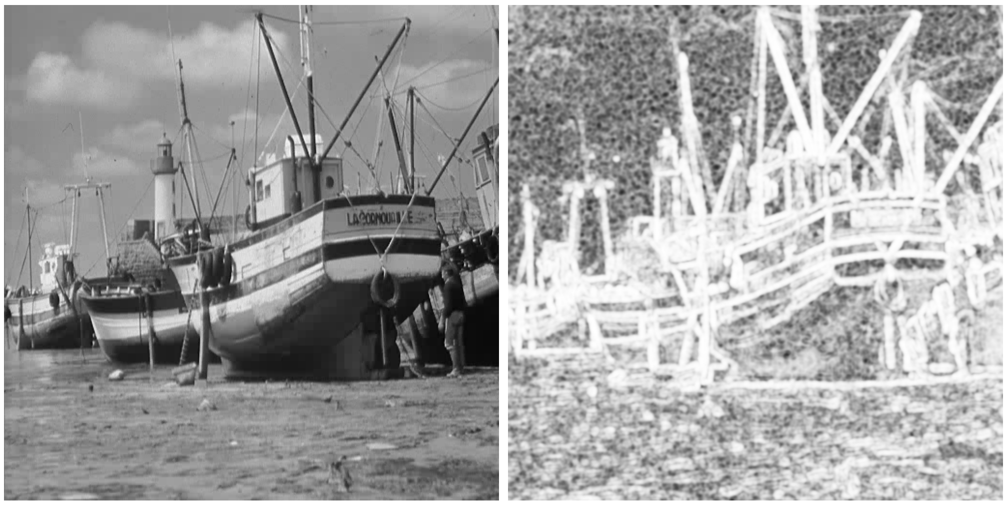
(b) Boat

(c) Elaine

(d) Stream

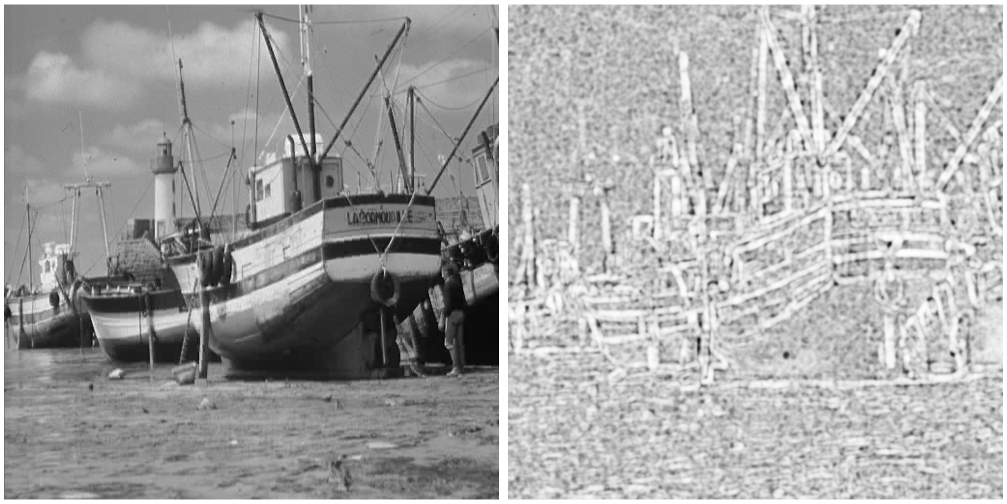
Figure 10: Normalized sparse bit-rate vs MSSIM. Comparison of the two different stopping rules for OMP.

Figures 11 and 12 consist of two images each, the reconstruction from the original (left) and the corresponding SSIM index map (right). The SSIM map represents the localized quality of the image reconstruction. Lighter values are values closer to 1 (“white” = 1), whereas darker values are values closer to 0 (“black” = 0). Using the image of the *Boat* as our image $\mathcal{I}$, we obtained a reconstruction $\widetilde{\mathcal{I}}_1$ with $\epsilon = 32$ for the ℓ^2 stopping criterion, and a reconstruction $\widetilde{\mathcal{I}}_2$ for the MSSIM stopping criterion choosing $\delta_0 < \text{MSSIM}(\widetilde{\mathcal{I}}_1, \mathcal{I})$ in such a way that $\text{MSSIM}(\widetilde{\mathcal{I}}_2, \mathcal{I}) \simeq \text{MSSIM}(\widetilde{\mathcal{I}}_1, \mathcal{I})$.



(a) Boat (b) SSIM

Figure 11: Boat: $\epsilon = 32$, PSNR = 36.6020 dB, MSSIM = 0.9210, normalized sparse bit-rate = 0.1608 bpp, stopping rule: $\|\cdot\|_2$.



(a) Boat (b) SSIM

Figure 12: Boat: $\delta_0 = 0.92$, PSNR = 34.1405 dB, MSSIM = 0.9351, normalized sparse bit-rate = 0.1595 bpp, stopping rule: $\|\cdot\|_{MSSIM}$.

8 Sampling in image representation and compression

8.1 Compressed sensing and sampling

Our approach to image representation and our treatment of images for compression lends itself to experimentation in the area of compressed sensing. With a slight modification of the classical compressed sensing technique, we shall show how to construct *deterministic sampling masks* in order to sample an original image and recover an approximation having a controllable signal-to-noise ratio. This technique can be interpreted in two ways, either as a classical compressed sensing problem or as a non-uniform sampling reconstruction problem, see Section 8.3.

For signals that are sparsely generated, the classical compressed sensing paradigm can be summarized as follows. Consider a random matrix $\mathbf{P} \in \mathbb{R}^{k\times n}$ with Gaussian i.i.d. entries, and suppose that it is possible to directly measure $\mathbf{c} = \mathbf{P}\mathbf{b}$, which has k entries, rather than $\mathbf{b}$, which has n. Then we solve

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to } \|\mathbf{P}\mathbf{A}\mathbf{x} - \mathbf{c}\|_2 \leq \epsilon, \tag{22}$$

to obtain a sparse representation $\mathbf{x}_0^\epsilon$, and synthesize an approximate reconstruction $\mathbf{b} \approx \mathbf{A}\mathbf{x}_0^\epsilon$, where $\mathbf{x}_0^\epsilon$ is a solution to Problem (22) [11].

We make the following observations about this technique. If one literally generates a random matrix $\mathbf{P} \in \mathbb{R}^{k\times n}$, we would expect with probability 1 that $\mathbf{P}\mathbf{A} \in \mathbb{R}^{k\times m}$ would be full-rank, given that we have assumed all along that $\mathbf{A} \in \mathbb{R}^{n\times m}$ is such, and that $\|\mathbf{x}_0^\epsilon\|_0 \leq k \leq n$. A desirable feature from this approach would be that if $k = n$, we should obtain the same representation $\mathbf{x}_0^\epsilon$ as if we had solved (P_0^ϵ) directly, see Problem (2), with the dictionary set to $\mathbf{P}\mathbf{A}$ instead of $\mathbf{A}$, and the signal $\mathbf{b}$ set to $\mathbf{P}\mathbf{b}$. In this case, if $\mathbf{P}$ were an isometry, then $\|\mathbf{A}\mathbf{x}_0^\epsilon - \mathbf{b}\|_2 = \|\mathbf{P}\mathbf{A}\mathbf{x}_0^\epsilon - \mathbf{P}\mathbf{b}\|_2 \leq \epsilon$ and we would have the same signal-to-noise ratio, or equivalently, normalized sparse bit-rate vs tolerance, image reconstruction/representation characteristics as if we had performed a direct measurement of $\mathbf{b}$. But in general, $\|\mathbf{A}\mathbf{x}_0^\epsilon - \mathbf{b}\|_2 \neq \|\mathbf{P}\mathbf{A}\mathbf{x}_0^\epsilon - \mathbf{P}\mathbf{b}\|_2$. Hence, if we want to have the desirable property of recovering the solution to (P_0^ϵ) when $k = n$, then we must pay particular attention to the construction of $\mathbf{P}$. Moreover, having a particular matrix $\mathbf{P}$ does not tell us anything on how to actually sample $\mathbf{b}$, but we assume that somehow, we can obtain the signal $\mathbf{c}$ to treat with our algorithms.

8.2 Deterministic sampling masks

To overcome the shortcomings inherent to the classical compressed sensing approach mentioned above, we propose the use of deterministic sampling masks, which we define next.

Consider an 8×8 sub-image $\mathbf{Y}$ of an image $\mathcal{I}$, and an 8×8 matrix $\mathbf{M}$ whose entries are either 0 or 1. We can choose the values of $\mathbf{M}$ at random or in a deterministic way. We choose a hybrid, that is, we preset the number of entries in $\mathbf{M}$ that will be zero, but choose their location at random. In this case, we shall call $\mathbf{M}$ a deterministic sampling mask. Let k be the number of entries in $\mathbf{M}$ that are equal to 1, then the ratio of k to the the total number of entries in $\mathbf{M}$ is called the density of $\mathbf{M}$, $k/64$ in this case. We denote it by $\rho(\mathbf{M})$. Reciprocally, we say that $\frac{64-k}{64}$ is the sparsity of $\mathbf{M}$.

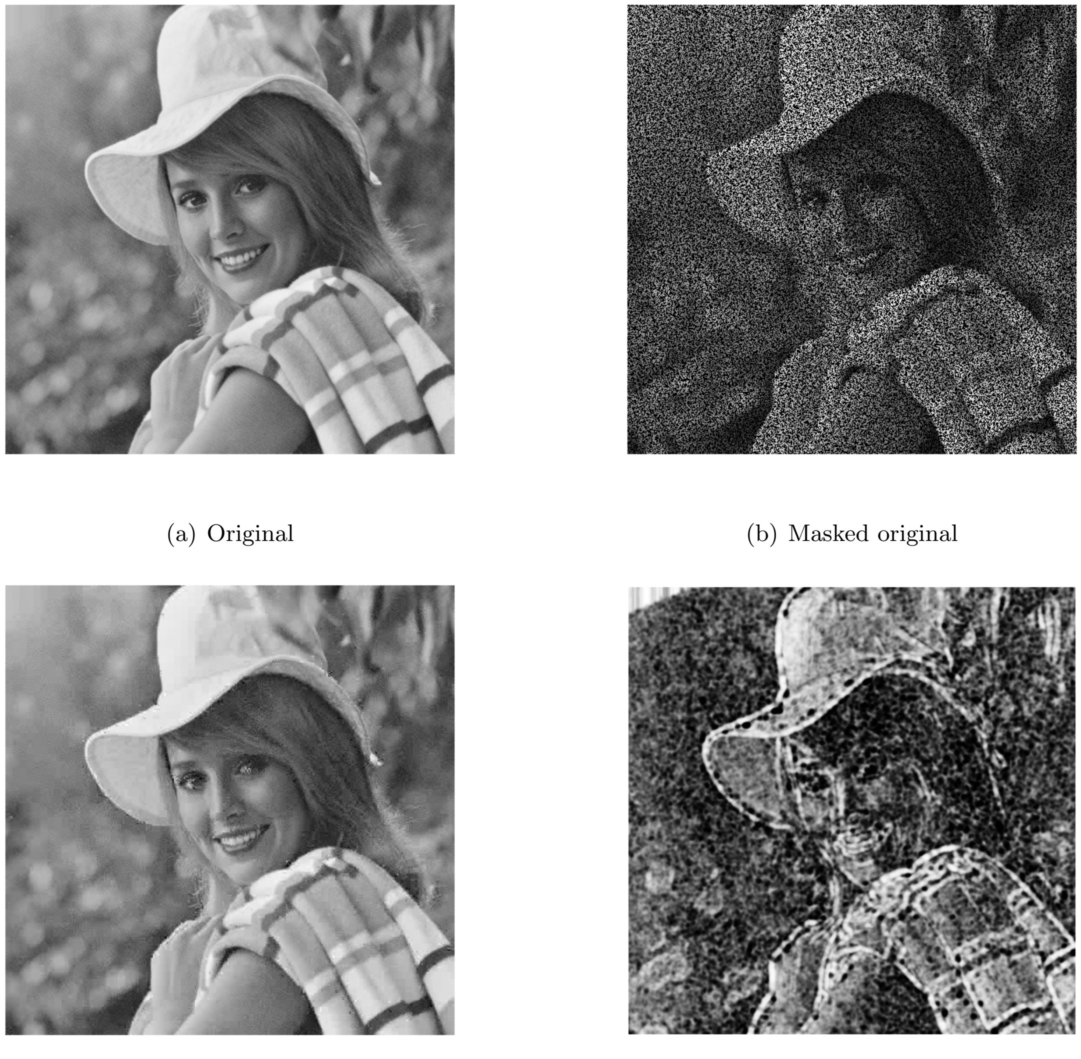
Now we choose a vectorization function, say c_3, see Section 4.3.2, and apply it both to the sub-image $\mathbf{Y}$ and mask $\mathbf{M}$, and obtain a vector $\mathbf{w} = (w_1, \ldots, w_{64})^{\mathrm{T}} \in \mathbb{R}^{64}$ equal to the entry-wise product $c_3(\mathbf{Y}) \otimes c_3(\mathbf{M})$. Finally, given both $\mathbf{w}$ and $c_3(\mathbf{M})$, we perform a dimension reduction h on $\mathbf{w}$ by collapsing its entries where $c_3(\mathbf{M})$ vanishes, obtaining $\mathbf{c} \in \mathbb{R}^k$. In function form, we have

$$\begin{aligned} h &: \mathbb{R}^{64} \times \mathbb{R}^{64} \to \mathbb{R}^k \\ h(\mathbf{w}, c_3(\mathbf{M})) &= (w_{j_1}, \ldots, w_{j_k})^{\mathrm{T}}, \end{aligned} \tag{23}$$

with $j_1 < \cdots < j_k$ and $supp(c_3(\mathbf{M})) = \{j_i\}$. We call $\mathbf{c} = h(c_3(\mathbf{Y}), c_3(\mathbf{M}))$ the masking of $\mathbf{Y}$ by $\mathbf{M}$. If $\mathbf{b} = c_3(\mathbf{Y})$ and $\mathbf{c} = h(\mathbf{b}, c_3(\mathbf{M}))$, we will also say that $\mathbf{c}$ is the masking of $\mathbf{b}$ by $\mathbf{M}$.

It is easy to see that if the density $\rho(\mathbf{M}) = 1$, then $h(\mathbf{b}, c_3(\mathbf{M})) = \mathbf{b}$ for any $\mathbf{b} \in \mathbb{R}^{64}$. This fact will help us achieve the goal of obtaining identical solutions to Problems (2) and (22) when $\rho(\mathbf{M}) = 1$, overcoming one of the main shortcomings we had mentioned in Section 8.1, as seen in the next section.

8.3 Image reconstruction from deterministic sampling masks and error analysis



(a) Original (b) Masked original

(c) Reconstruction from (b) (d) SSIM between (a) and (c)

Figure 13: Reconstruction of image *Elaine* from a deterministic sampling mask with 50% density and $\epsilon = 4\sqrt{32}$. The reconstruction is achieved at 29.8081 dB and has an MSSIM equal to 0.7461.

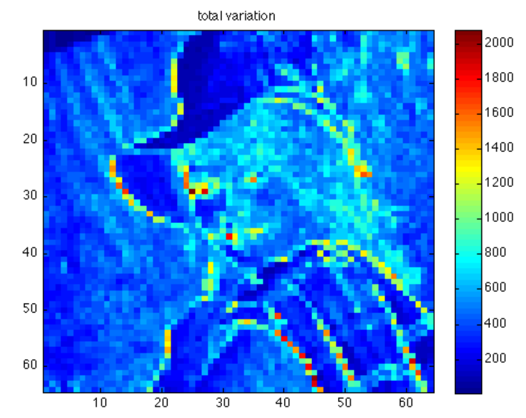
Let $\mathcal{I}$ be an image, and assume that it can be divided in N disjoint sub-images $\{\mathbf{Y}_i\}$ of size $L \times L$. Assume also that we generate an equal number of deterministic sampling masks $\{\mathbf{M}_i\}$, all with the same density $\rho(\mathbf{M}_i) = k/L^2$. That is, each deterministic sampling mask $\mathbf{M}_i$ will have exactly k entries equal to 1. Then choose a vectorization function, say c_3, cf. Section 4.3.2, and form the masking $\mathbf{c}_i$ of $\mathbf{b}_i = c_3(\mathbf{Y}_i)$ by $\mathbf{M}_i$ for all N sub-images with their respective mask.

With this setup, we are ready to propose a way to recover image $\mathcal{I}$ from $\{\mathbf{c}_i\}$. For each $i \in \{1, \dots, N\}$, let $\mathbf{x}_{0,i}^{\epsilon}$ be the solution to the problem

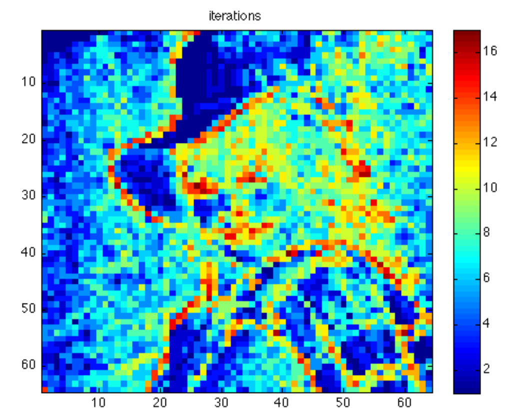
$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to } \|h(\mathbf{Ax}, c_3(\mathbf{M}_i)) - \mathbf{c}_i\|_2 \leq \epsilon, \tag{24}$$

and recover a reconstruction $\widetilde{\mathcal{I}}$ of $\mathcal{I}$ by assembling in correct order the sub-images generated by the sparse representations $\{\mathbf{x}_{0,i}^{\epsilon}\}$, that is, $\widetilde{\mathcal{I}}$ is formed by putting together in proper sequence the set of sub-image reconstructions $\{c_3^{-1}(\mathbf{Ax}_{0,i}^{\epsilon})\}$. See Figure 13 for an example of this technique in action.

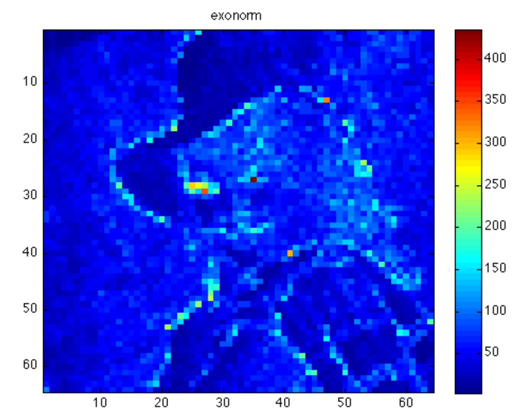
A few observations are in order. First, note that we can identify Problem (22) with Problem (24), if we identify $\mathbf{PAx}_{0,i}^{\epsilon} \equiv h(\mathbf{Ax}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i))$. This way, if the density $\rho(\mathbf{M}_i) = 1$ for all i, given that $h(\mathbf{Ax}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i)) = \mathbf{Ax}_{0,i}^{\epsilon}$—from the observation made at the end of the previous section—Problem (24), and therefore Problem (22), would be identical to Problem (2), as desired. Hence, Problem (24) is a generalization of Problem (2).



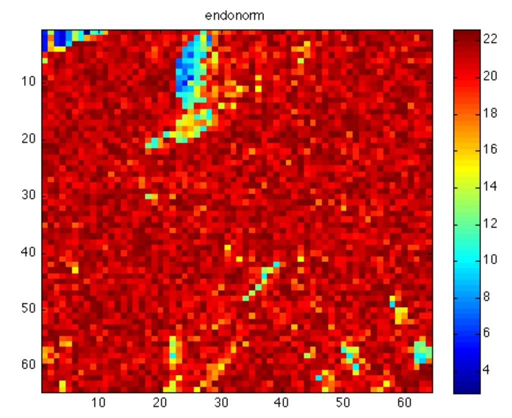
(a) Total variation



(b) Number of iterations



(c) Exonorm



(d) Endonorm

Figure 14: Total variation, performance, and error metrics for the reconstruction of *Elaine* by way of deterministic sampling masks. Density is set to 50%, $\epsilon = 4\sqrt{32}$, and dictionary $\mathbf{A} = [\mathrm{DCT}_{2,3}]$.

Second, we can say a few words regarding the choice of ϵ in Problem (24), since this parameter controls the compression and, to a degree, the error characteristics of the reconstruction image $\widetilde{\mathcal{I}}$. From the choice of deterministic sampling matrices that we made, the effective dimension of the vectors $\{\mathbf{c}_i\}$ that we are trying to approximate is k, that is, $\mathbf{c}_i \in \mathbb{R}^k$ for all i. Bearing in mind that $\mathbf{c}_i$ can be thought of as the representation of a sub-image of size $\sqrt{k} \times \sqrt{k}$ via a similar vectorization function in the spirit of, say c_3, as was done here, the value of ϵ determines the error of the resulting approximation $h(\mathbf{A}\mathbf{x}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i))$ to $\mathbf{c}_i$ constructed here. From the error analysis and interpretation done for Equation (21), we can then compute the value of ϵ given a target idealized uniform error distribution. For example, if we want on average to be at no more than 4 units away from the actual value of each pixel in the image represented by $\mathbf{c}_i$, we must choose $\epsilon = 4\sqrt{k}$. If, as in our example in Figure 13, we have partitioned our image in 8×8 sub-images, and chosen a density of 50%, this would give a value of $\epsilon = 4\sqrt{32}$. Then, the error between the points in $\mathbf{Y}_i$ represented by $\mathbf{c}_i$ and $h(\mathbf{A}\mathbf{x}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i))$ would be on average no more than 4 units per pixels. In fact, the ℓ^2 norm of $h(\mathbf{A}\mathbf{x}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i)) - \mathbf{c}_i$, will be less than or equal to $4\sqrt{32}$. We call this error the *endonorm* for sub-image $\mathbf{Y}_i$.

However, we cannot yet say much about the error outside the pixels not represented by $\mathbf{c}_i$ in the original sub-image. This error will be given by $\|\mathbf{A}\mathbf{x}_{0,i}^{\epsilon} - \mathbf{b}_i\|_2 - \|h(\mathbf{A}\mathbf{x}_{0,i}^{\epsilon}, c_3(\mathbf{M}_i)) - \mathbf{c}_i\|_2$, which we call the *exonorm* for sub-image $\mathbf{Y}_i$. It is clear from these definitions that the total error $\|\mathbf{A}\mathbf{x}_{0,i}^{\epsilon} - \mathbf{b}_i\|_2$ for the approximation of sub-image $\mathbf{Y}_i$ by $c_3^{-1}(\mathbf{A}\mathbf{x}_{0,i}^{\epsilon})$ is the sum of its endo- and exonorms. As we established above, we have exact control of the endonorm by the choice of ϵ, but non whatsoever of the exonorm. We speculate that the magnitude of the exonorm of any given sub-image $\mathbf{Y}_i$ is linked to the choice of dictionary $\mathbf{A}$, and the total variation $V(\mathbf{b}_i) \doteq \sum_{j=1}^{n-1} |b_{j+1} - b_j|$, of $\mathbf{b}_i = (b_1, \ldots, b_n)^{\mathrm{T}}$. In Figure 14 we show four maps, each showing the value of a given metric that each sub-image that makes up image *Elaine* has, for each of the four metrics we consider. The metrics are the total variation, the number of iterations that it took OMP to converge to the desired error tolerance of $\epsilon = 4\sqrt{32} \approx 22.63$ in Equation (24), the exonorm, and the endonorm.

Note that in Figure 14(d) the endonorm is less than or equal to the desired error tolerance. Observe the correlation that the number of iterations, Figure 14(b), and the exonorm, Figure 14(c), seem to have with the total variation, Figure 14(a), of each sub-image. In this case, we set $\mathbf{A} = [\mathrm{DCT}_{2,3}]$. When we try $\mathbf{A} = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$ the results show an increase in the exonorm, which impacts negatively both PSNR and MSSIM error measures, although the normalized sparse bit-rate is slightly smaller, as expected. This confirms that the choice of dictionary $\mathbf{A}$ has an impact on the error of the reconstruction from deterministic sampling masks.

Finally, we mention that since all masks have the same density, the overall sampling density for the original image is k/L^2, which means that a total of kN points are sampled, as opposed to L^2N, had we sampled all points in the image to reconstruct it. Moreover, the sampling is done at random. Hence, what starts as a compressed sensing problem can be seen as a non-uniform sampling reconstruction problem.

9 Quantization

9.1 Background

As a precursor to Shannon, Hartley wrote the following equation to quantify "information" in a discrete setting:

$$H = n \log s,$$

where H is the amount of information, n is the number of symbols transmitted, and s is the size of a given alphabet from which the symbols are drawn [18]. Shannon extended this notion by identifying the amount of information with entropy, see [36]. Specifically, in the case of a discrete information source, Shannon represented it as a Markov process, and asked if one could "define a quantity which will measure, in some sense, how much information is 'produced' by such a process, or better, at what rate information is produced". In fact, he defined this quantity H in terms of entropy as

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{25}$$

where we suppose that we have a set of n possible events whose probabilities of occurrence are $p_1, p_2, \ldots, p_n$.

To interpret Equation (25) we assume that we are given a random variable X on the finite set $\{1, 2, \ldots, n\}$ with probability distribution p. The elements $X(1) = x_1, X(2) = x_2, \ldots, X(n) = x_n$ are distinct and $p(x_1), p(x_2), \ldots, p(x_n)$ are nonnegative real numbers with $p(x_1) + p(x_2) + \ldots + p(x_n) = 1$. We write $p_i = p(x_i)$ as a shorthand for $\mathrm{prob}(X = x_i)$. The smaller the probability $p(x_i)$, the more uncertain we are that an observation of X will result in x_i. Thus, we can regard $1/p(x_i)$ as a measure of the uncertainty of x_i. The smaller the probability, the larger the uncertainty, see [36, 31, 23]. Shannon thought of uncertainty as information. In fact, if an event has probability 1, there is no information gained in asking the outcome of such an event given that the answer will always be the same.

Consequently, if we define the *uncertainty* of x_i to be $-\log_2 p(x_i)$, measured in *bits*, the *entropy* of the random variable X is defined to be the expected value,

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i),$$

of the uncertainty of X, i.e., the entropy of X will measure the information gained from observing X.

Further, Shannon defined the capacity C of a discrete noiseless channel as

$$C = \lim_{T\to\infty} \frac{\log N(T)}{T},$$

where $N(T)$ is the number of allowed signals of duration T.

Theorem 3 (Shannon, Fundamental Theorem for a Noiseless Channel [36])**.** *Let a source have entropy H (bits per symbol) and let a channel have a capacity C (bits per second). Then it is possible to encode the output of the source in such a way as to transmit at the average rate $\frac{C}{H} - \epsilon$ symbols per second over the channel where ϵ is arbitrarily small. It is not possible to transmit at an average rate greater than $\frac{C}{H}$.*

This existential result has been a driving force for developing constructive quantization and coding theory through the years, e.g., [19] for an extraordinary engineering perspective and highlighting the role of redundancy in source signals, cf. the *theory of frames* [4, Chap. 3 and 7] and [6, 14, 13].

9.2 Quantization (coding), rate, and distortion

A *scalar quantizer* is a set $\mathcal{S}$ of intervals or *cells* $S_i \subset \mathbb{R}, i \in I$, that forms a partition of the real line, where the index set I is ordinarily a collection of consecutive integers beginning with 0 or 1, together with a set $\mathcal{C}$ of *reproduction values* or *levels* $y_i \in \mathbb{R}, i \in I$, so that the overall quantizer q is defined by $q(x) = y_i$ for $x \in S_i$, expressed concisely as

$$q(x) = \sum_{i \in I} y_i \mathbb{1}_{S_i}(x), \tag{26}$$

where the indicator function $\mathbb{1}_S(x)$ is 1 if $x \in S$ and 0 otherwise [19, 12].

More generally, a class of *memoryless quantizers* can be described as follows. A quantizer of dimension $k \in \mathbb{N}$ takes as input a vector $\mathbf{x} = (x_1, \ldots, x_k)^{\mathrm{T}} \in \mathcal{A} \subseteq \mathbb{R}^k$. Memoryless refers to a quantizer which operates independently on successive vectors. The set $\mathcal{A}$ is called the *alphabet* or *support* of the source distribution. If $k = 1$ the quantizer is *scalar*, and, otherwise, it is *vector*. The quantizer then consists of three components: a *lossy encoder* $\alpha : \mathcal{A} \to I$, where the index set I is an arbitrary countable set; a *reproduction decoder* $\beta : I \to \hat{\mathcal{A}}$, where $\hat{\mathcal{A}} \subset \mathbb{R}^k$ is the *reproduction alphabet*; and a *lossless encoder* $\gamma : I \to J$, an invertible mapping (with probability 1) into a collection J of variable-lenght binary vectors that satisfies the *prefix condition*, that is, no vector in J can be the prefix of any other vector in the collection [19].

Alternatively, a lossy encoder is specified by a scalar quantizer $\mathcal{S} = \{S_i \subset \mathbb{R} : i \in I\}$ of $\mathcal{A}$; a reproduction decoder is specified by a *codebook* $\mathcal{C} = \{\beta(i) \in \hat{\mathcal{A}} : i \in I\}$ of *points*, *codevectors*, or *reproduction codewords*, also known as the *reproduction codebook*; and the lossless encoder γ can be described by its *binary codebook* $J = \{\gamma(i) : i \in I\}$ containing *binary* or *channel codewords*. The *quantizer rule* is the function $q(\mathbf{x}) = \beta(\alpha(\mathbf{x}))$ or, equivalently, $q(\mathbf{x}) = \beta(i)$ whenever $\mathbf{x} \in S_i$ [19].

The instantaneous rate of a quantizer applied to a particular input is the normalized length $r(\mathbf{x}) = \frac{1}{k} l(\gamma(\alpha(\mathbf{x})))$ of the channel codeword, the number of bits per source symbol that must be sent to describe the reproduction. If all binary codewords have the same length, it is referred to as a *fixed-length* or *fixed-rate* quantizer.

To measure the quality of the reproduction, we assume the existence of a nonnegative distortion measure $d(\mathbf{x}, \hat{\mathbf{x}})$ which assigns a distortion or cost to the reproduction of input $\mathbf{x}$ by $\hat{\mathbf{x}}$. Ideally, one would like a distortion measure that is easy to compute, useful in analysis, and perceptually meaningful in the sense that small (large) distortion means good (poor) perceived quality. No single distortion measure accomplishes all three goals [19]. However, $d(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ satisfies the first two.

We also assume that $d(\mathbf{x}, \hat{\mathbf{x}}) = 0$ if and only if $\mathbf{x} = \hat{\mathbf{x}}$. In this light we say that a code is *lossless* if $d(\mathbf{x}, \beta(\alpha(\mathbf{x}))) = 0$ for all inputs $\mathbf{x}$, and *lossy* otherwise.

Finally, the overall performance of a quantizer applied to a source is characterized by the *normalized rate*,

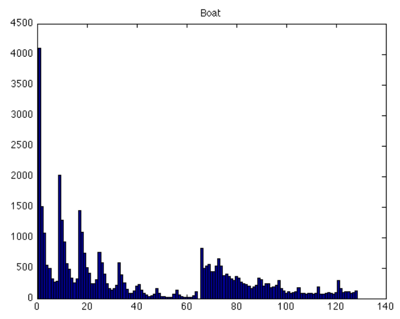
$$\begin{aligned} R(\alpha, \gamma) = E[r(X)] &= \frac{1}{k} E[l(\gamma(\alpha(X)))] \\ &= \frac{1}{k} \sum_i l(\gamma(i)) \int_{S_i} f(\mathbf{x}) \, d\mathbf{x}, \end{aligned} \tag{27}$$

and the *normalized average distortion*,

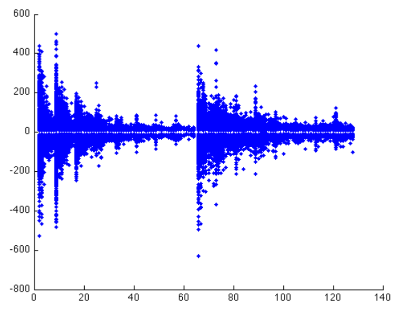
$$\begin{aligned} D(\alpha, \beta) &= \frac{1}{k} E[d(X, \beta(\alpha(X)))] \\ &= \frac{1}{k} \sum_i \int_{S_i} d(\mathbf{x}, \mathbf{y}_i) f(\mathbf{x}) \, d\mathbf{x}. \end{aligned} \tag{28}$$

Here, we assume that the quantizer operates on a k-dimensional random vector $X = (X_1, \ldots, X_k)$ that is described by a probability density function f. Every quantizer (α, γ, β) is thus described by a rate-distortion pair $(R(\alpha,\gamma), D(\alpha,\beta))$. The goal of a compression system design is to optimize the rate-distortion trade-off [19].

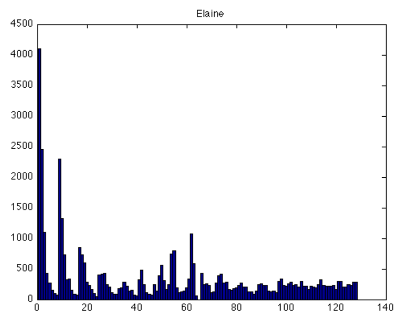
In light of the results by Shannon in Section 9.1, compression system design will also have to take into account the characteristics of the communication channel in managing the rate-distortion trade-off. Also, from the definitions above, it is clear that knowledge of the probability density function of the source messages is relevant, see Figure 15.



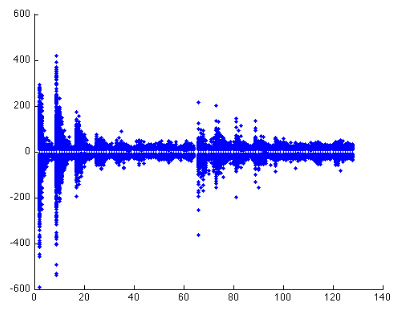
(a)



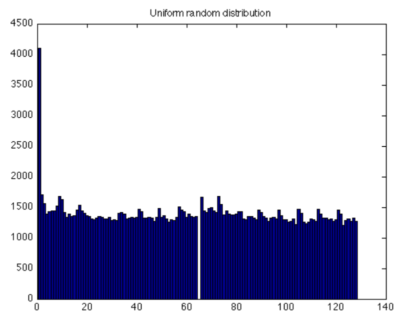
(b)



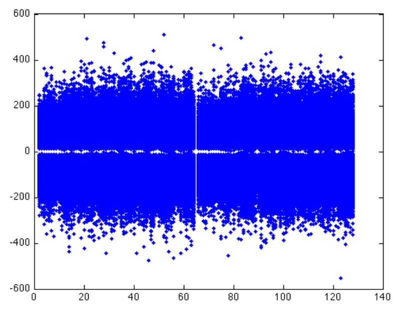
(c)



(d)



(e)



(f)

Figure 15: Histograms for images *Boat*, *Elaine* and a uniform random input. Figures (a), (c), and (e) are the histograms for the choice of columns of $\mathbf{A}$, labeled from left to right, 1 to 128, and Figures (b), (d), and (f) show the set of coefficient values chosen for each column, labeled in a similar way. We set the tolerance $\epsilon = 32$, and used $\mathbf{A}_3 = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}]$.

9.3 Image quantization and encoding

With the perspective from Sections 9.1 and 9.2, we return to the topic of image quantization and encoding with regard to compression. The image standards, JPEG and JPEG 2000, are framed in the transform coding paradigm, and contain two steps beyond their respective discrete cosine transform (DCT) and the Cohen-Daubechies-Feauveau 5/3 (lossless) and 9/7 (lossy) biorthogonal wavelet transforms. Both have *quantization* (α) and *encoding* (γ) steps, with their respective "inverses", to complete their definitions [41, 2, 38, 15].

The general schematic for transform coding is described in Figure 16.

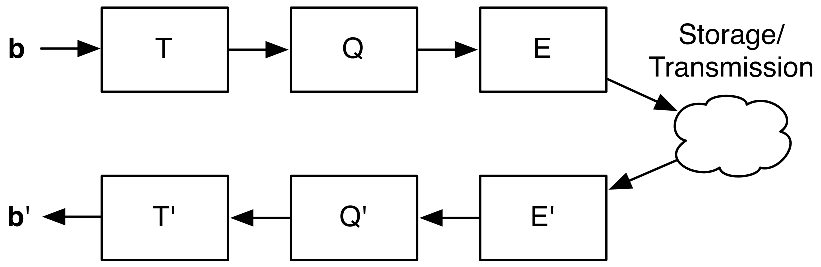


Figure 16: Schematic diagram of a general transform coding system.

Example 1. The transform T is meant to exploit the redundancy in the source signal $\mathbf{b}$ and decorrelate it. It has an inverse T^{-1} or, minimally, a left inverse T' such that $T'T\mathbf{b} = \mathbf{b}$. In our approach we have $\mathbf{A} = T'$, and T is defined via OMP by $T\mathbf{b} = \texttt{OMP}(\mathbf{A}, \mathbf{b}, \epsilon_0)$. Thus, we have $\|T'T\mathbf{b} - \mathbf{b}\|_2 < \epsilon$. Therefore, our compression scheme is lossy. Q is a non-invertible scalar quantizer that will be applied to the coefficients of the vector $T\mathbf{b}$, and Q' is its reproduction function. Finally, we have an invertible lossless encoder E, defined as γ in the previous section, with $E' = E^{-1}$. The composition QT is equivalent to the lossy encoder α, and the composition $T'Q'$ corresponds to the reproduction decoder β from Section 9.2.

In order to describe the overall performance of our quantizer, $(\alpha, \gamma, \beta) = (QT, E, T'Q')$, we must characterize the rate-distortion pair $(R(QT, E), D(QT, T'Q'))$.

Proposition 4. *Let $n < m$ and let $\mathbf{A} = (\mathbf{a}_j) \in \mathbb{R}^{n \times m}$ be a full-rank matrix with each $\|\mathbf{a}_j\|_2 = c$. Given $a > 0$ and $\mathbf{y} \in \mathbb{R}^n$. Suppose that $\mathbf{x} \in \mathbb{R}^m$ has the property that $a\mathbf{A}\mathbf{x} = \mathbf{y}$ and that $\epsilon \in \mathbb{R}^m$ satisfies $\|\epsilon\|_0 \le \|\mathbf{x}\|_0$. If $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$ and $\tilde{\mathbf{y}} = a\mathbf{A}\tilde{\mathbf{x}}$, then*

$$\|\tilde{\mathbf{y}} - \mathbf{y}\|_2 \le ac\|\epsilon\|_\infty \|\mathbf{x}\|_0. \tag{29}$$

Proof. Let $\epsilon = (\epsilon_1, \ldots, \epsilon_m)^{\mathrm{T}} \in \mathbb{R}^m$ with $\|\epsilon\|_0 \le \|\mathbf{x}\|_0$ and let $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$. Then, we compute

$$\begin{aligned}
\|\tilde{\mathbf{y}} - \mathbf{y}\|_2 &= \|a\mathbf{A}\tilde{\mathbf{x}} - a\mathbf{A}\mathbf{x}\|_2 = a\|\mathbf{A}(\mathbf{x} + \epsilon) - \mathbf{A}\mathbf{x}\|_2 \\
&= a\|\mathbf{A}\epsilon\|_2 = a\Big\|\sum_{j=1}^{m} \mathbf{a}_j\epsilon_j\Big\|_2 = a\Big\|\sum_{\epsilon_j \neq 0} \mathbf{a}_j\epsilon_j\Big\|_2 \\
&\le a\sum_{\epsilon_j \neq 0} \|\mathbf{a}_j\epsilon_j\|_2 = a\sum_{\epsilon_j \neq 0} \|\mathbf{a}_j\|_2|\epsilon_j| = a\sum_{\epsilon_j \neq 0} c|\epsilon_j| \\
&\le a\sum_{\epsilon_j \neq 0} c\|\epsilon\|_\infty = ac\|\epsilon\|_\infty\|\epsilon\|_0 \le ac\|\epsilon\|_\infty\|\mathbf{x}\|_0.
\end{aligned}$$

□

Remark 1. a. In Proposition 4, the value of $\|\epsilon\|_0$ is linked to the sparsity of $\mathbf{x}$, because in our case the error ϵ comes from scalar quantizing the entries of $\mathbf{x}$. That is, if $\tilde{\mathbf{x}} = round(\mathbf{x})$, where $\tilde{\mathbf{x}}$ is the vector whose entries are exactly those of $\mathbf{x}$ but rounded to the closest integer, then, necessarily,

$$\|\epsilon\|_0 = \|\tilde{\mathbf{x}} - \mathbf{x}\|_0 \leq \|\mathbf{x}\|_0. \tag{30}$$

Hence, in the case where a scalar quantization scheme satisfies Inequality (30), Proposition 4 gives

$$\|\tilde{\mathbf{y}} - \mathbf{y}\|_2 \leq ac\|\epsilon\|_\infty \|\mathbf{x}\|_0, \tag{31}$$

with $\mathbf{A} = (\mathbf{a}_j)$, $\tilde{\mathbf{y}} = a\mathbf{A}\tilde{\mathbf{x}}$, $\mathbf{y} = a\mathbf{A}\mathbf{x}$, and $c = \|\mathbf{a}_j\|_2$ for all j. Observe that, in particular, when $\tilde{\mathbf{x}} = round(\mathbf{x})$, we have $\|\epsilon\|_\infty = 1/2$.

From Inequality (31) we see that the error in the reconstruction due to scalar quantization is linked to the size of c, $\|\epsilon\|_\infty$, and the sparsity of $\mathbf{x}$. We are tempted to make c as small as possible, or modify the quantization scheme to make $\|\epsilon\|_\infty$ smaller to reduce the reconstruction error due to scalar quantization.

b. If $\mathbf{x}_0$ is the sparsest vector that solves $\mathbf{A}\mathbf{x} = \mathbf{y}$, then $a^{-1}\mathbf{x}_0$ is the sparsest vector that solves $a\mathbf{A}\mathbf{x} = \mathbf{y}$, and $\|\mathbf{x}_0\|_0 = \|a\mathbf{x}_0\|_0$. We conclude that the norm of $a^{-1}\mathbf{x}_0$ has an inversely proportional relationship with the size of a. Therefore, if we are to use a finite predetermined number of bits to represent $a^{-1}\mathbf{x}_0$, the solution of $a\mathbf{A}\mathbf{x} = \mathbf{y}$, we necessarily have a constraint on a.

c. We know that the magnitude of the coordinates of $\mathbf{x}$ are bounded by a multiple of $\|\mathbf{y}\|_2$, see [5]. This has an impact on how many bits are needed to represent $\mathbf{x}$. Therefore, when choosing a in Proposition 4 we have to take into consideration the maximum value that the value of $\|\mathbf{y}\|_2$ will impose on the magnitude of the coordinates of $\mathbf{x}$, see Examples 2 and 3.

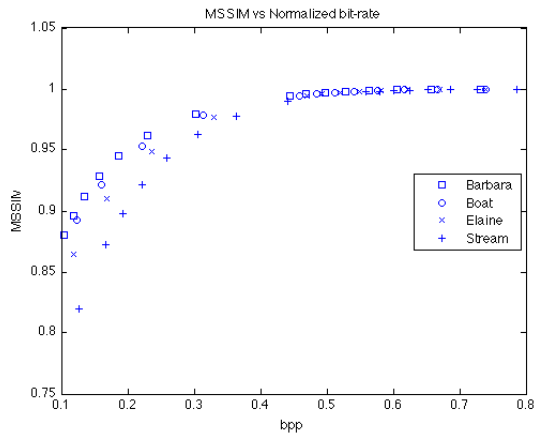
Finally, recall that our image compression approach comes from the OMP solution $\mathbf{x}_0$ to problem $(P_0^{\epsilon_0})$ for a given matrix $\mathbf{A} = (\mathbf{a}_j)$ whose column vectors satisfy $\|\mathbf{a}_j\|_2 = c$, where $\mathbf{b}$ is a given vector, and for which $\|\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2 < \epsilon_0$ for the given tolerance ϵ_0. Then, choosing $a > 0$, and following the description of T at the beginning of this section, if we set $\mathbf{x}_0 = T\mathbf{b} = \mathtt{OMP}(a\mathbf{A}, \mathbf{b}, \epsilon_0)$ for a given signal $\mathbf{b}$ and a tolerance $\epsilon_0 > 0$, $a\mathbf{A} = T'$, Q a scalar quantizer that satisfies Inequality $\|\epsilon\|_0 = \|Q(\mathbf{x}) - \mathbf{x}\|_0 \leq \|\mathbf{x}\|_0$, and Q' its corresponding reproduction function, the triangle inequality and Inequality (29) give

$$\begin{aligned}
d(\beta(\alpha(\mathbf{b})), \mathbf{b}) &= \|T'Q'QT\mathbf{b} - \mathbf{b}\|_2 \\
&= \|T'Q'QT\mathbf{b} - T'T\mathbf{b} + T'T\mathbf{b} - \mathbf{b}\|_2 \\
&= \|a\mathbf{A}\tilde{\mathbf{x}}_0 - a\mathbf{A}\mathbf{x}_0 + a\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2 \\
&\leq \|a\mathbf{A}\tilde{\mathbf{x}}_0 - a\mathbf{A}\mathbf{x}_0\|_2 + \|a\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2 \\
&= ac\|\delta\|_\infty \|\mathbf{x}_0\|_0 + \epsilon_0,
\end{aligned}$$

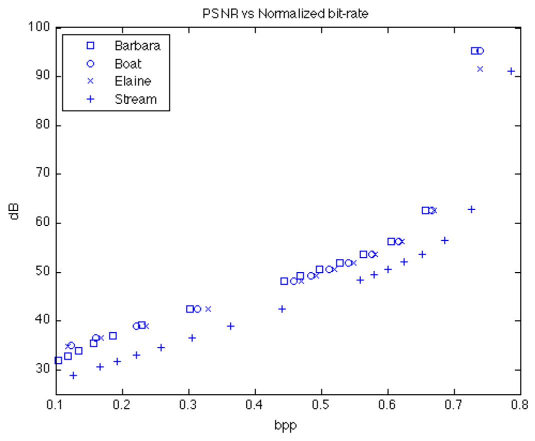
where $\delta = \tilde{\mathbf{x}}_0 - \mathbf{x}_0$. This inequality would give us a footing in the computation of the normalized average distortion $D(\alpha, \beta)$, see Equation (28).

Example 2. From the definition of $D(\alpha, \beta)$, it is clear that we need to know something about the probability density function of the input sources, i.e., the statistics of the 8×8 vectorized sub-images into which each image is partitioned, if we are to compute D. In place of such knowledge, we can observe the distribution of the coefficients for each of the vectors resulting from the analysis of the images in our database and their corresponding histograms. This is what Figure 15 shows. For each such image $\mathcal{I}$, we used the matrix $\mathbf{A}_3 = [\mathrm{DCT}_{2,3}\ \mathrm{Haar}_{2,3}] = (\mathbf{a}_i)$ with a tolerance of $\epsilon = 32$ to compute its statistics. On the x-axis of each subfigure in Figure 15 we have matched column $\mathbf{a}_i$ at position i to the integer i. Hence, positions 1 to 64 correspond to the DCT waveforms, and positions 65 to 128 to the Haar waveforms. All subfigures on the left are the histograms for the frequency with which each column vector of $\mathbf{A}_3$ is chosen. For example, since there are 4096 sub-images of size 8×8 in a 512×512 image, the column vector $\mathbf{a}_1$ will be chosen 4096 times since it corresponds to the constant vector, which computes the mean or DC component for each sub-image. All subfigures to the right correspond to partial representations of the distribution of the coefficients that multiply each and every column vector whenever such a vector is chosen in the representation/approximation of an input $\mathbf{b}$. For example, suppose that column $\mathbf{a}_{74}$ was multiplied by a coefficient $a_{74} = 3.2310$ to obtain the representation of some input $\mathbf{b} = a_{74}\mathbf{a}_{74} + \mathbf{r}$ within a tolerance of $\epsilon = 32$. Then we would have plotted point $(74, 3.2310)$ in its corresponding subfigure to the right. We have not plotted the coefficients for $\mathbf{a}_1$ since they correspond to the DC components of the sub-images of $\mathcal{I}$, which vary between 0 and 255. We note that all images in our database have a similar structure.

Example 3. For comparison purposes, we obtained the histogram and the distribution of coefficients for a randomly generated image with a uniform distribution on [0 255], see Figures 15(e) and 15(f). The first thing we note is that unlike the natural images in our database, all column vectors, except $\mathbf{a}_1$ and $\mathbf{a}_{65}$, which correspond to the constant vectors (one for the DCT and one for the Haar waveforms), are chosen about the same number of times regardless of their position. Further, the distribution of the values of the coefficients is uniform. It is also clear that, in order to be reconstructed, this is the image that requires the most nonzero coefficients within the tolerance $\epsilon = 32$. This is consistent with the definition of information by Shannon: the more the uncertainty in a source, the more information it carries.



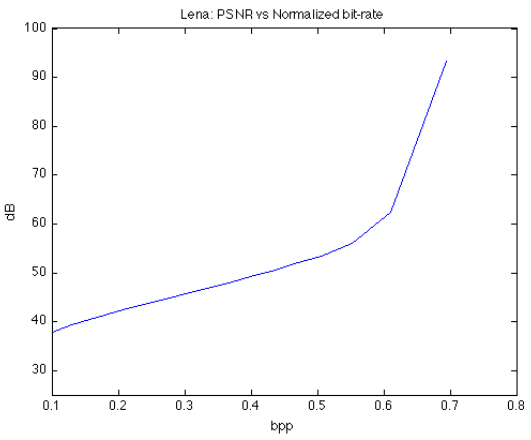
(a)



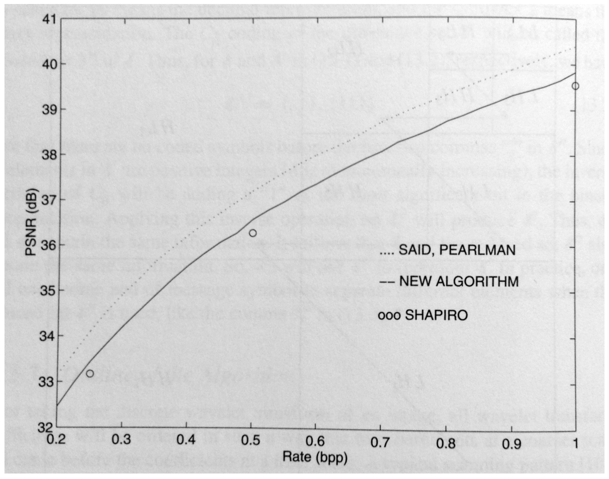
(b)

Figure 17: (a) MSSIM vs Normalized sparse bit-rate, (b) PSNR vs Normalized sparse bit-rate.

Example 4. In Figure 17, we have plotted the distortion as measured by both the MSSIM index and the PSNR versus the idealized normalized sparse bit-rate. This bit-rate is unattainable in practice but nonetheless gives us an idea of an upper bound in the rate-distortion trade-off. It also allows us to compare how much room we have to select a lossless encoder γ to complete the implementation of a quantizer using our sparse image representation approach. Figure 18(a) shows the PSNR versus normalized sparse bit-rate trade-off for image *Lena*, that we computed to compare with Figure 18(b), which shows results for that image for three published fully implemented



(a)



(b)

Figure 18: PSNR vs bit-rate: (a) Normalized sparse bit-rate results for $\mathbf{A} = [\text{DCT}_1\ \text{Haar}_1]$ prior to any γ coding, and (b) bit-rate coding performances published in [33] for image *Lena*: Said and Pearlman's SPIHT algorithm [34], Embeded Coding and the Wavelet-Difference-Reduction compression algorithm ("new algorithm") [39], and Shapiro's EZW algorithm [37].

quantizers. We observe that there is enough room to pick an encoder γ that could compete with these implementations.

Regarding the computation of the rate $R(\alpha, \gamma)$ for our image quantizer, we would have to choose a lossless encoder γ, which we have not done here.

10 Acknowledgement

The first named author gratefully acknowledges the support of MURI-ARO Grant W911NF-09-1-0383, NGA Grant 1582-08-1-0009, and DTRA Grant HDTRA1-13-1-0015. The second named author gratefully acknowledges the support of the Institute for Physical Science and Technology at the University of Maryland, College Park. We are both appreciative of expert observations by Professor Radu V. Balan, Department of Mathematics and Center for Scientific Computation and Mathematical Modeling (CSCAMM), Professor Ramani Duraiswami, Department of Computer Science and University of Maryland Institute of Advanced Computer Studies (UMIACS), and Professor Wojciech Czaja, Department of Mathematics, all at the University of Maryland, College Park.

References

[1] Enrico Au-Yeung and John J. Benedetto. Balayage and short time Fourier transform frames. *Proceedings of SampTA*, 2013.

[2] David Austin. What is... JPEG? *Notices of the AMS*, 55(2):226–229, February 2008.

[3] John J. Benedetto. *Harmonic Analysis and Applications.* CRC Press, Boca Raton, FL, 1997.

[4] John J. Benedetto and M. W. Frazier. *Wavelets: Mathematics and Applications.* CRC Press, Boca Raton, FL, 1994.

[5] John J. Benedetto and Alfredo Nava-Tudela. Frame estimates for OMP. Preprint, 2014.

[6] John J. Benedetto, Alex M. Powell, and O. Yilmaz. Sigma-delta ($\Sigma\Delta$) quantization and finite frames. *IEEE Transactions on Information Theory*, 52(5):1990–2005, May 2006.

[7] Arne Beurling. *The Collected Works of Arne Beurling. Vol. 2. Harmonic Analysis.* Birkhäuser, Boston, 1989.

[8] Arne Beurling and Paul Malliavin. On Fourier transforms of measures with compact support. *Acta Mathematica*, 107:291–309, 1962.

[9] Arne Beurling and Paul Malliavin. On the closure of characters and the zeros of entire functions. *Acta Mathematica*, 118:79–93, 1967.

[10] William L. Briggs and Van Emden Henson. *The DFT, an Owner's Manual for the Discrete Fourier Transform.* SIAM, Philadelphia, PA, 1995.

[11] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[12] James C. Candy and Gabor C. Temes, editors. *Oversampling Delta-Sigma Data Converters.* IEEE Press, NY, 1992.

[13] Peter G. Casazza and J. Kovačević. Uniform tight frames with erasures. *Advances in Computational Mathematics*, 18(2-4):387–430, February 2003.

[14] Ole Christensen. *An Introduction to Frames and Riesz Bases.* Springer-Birkhäuser, NY, 2003.

[15] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The JPEG 2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, November 2000.

[16] David Donoho, Iain Johnstone, Peter Rousseeuw, and Werner Stahel. Discussion: Projection pursuit. *Annals of Statistics*, 13(2):496–500, June 1985.

[17] Richard J. Duffin and A. C. Schaeffer. A class of nonharmonic Fourier series. *Trans. Amer. Math. Soc.*, 72:341–366, 1952.

[18] James Gleick. *The Information: a History, a Theory, a Flood.* Pantheon Books, New York, NY, 2011.

[19] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, October 1998.

[20] Karlheinz Gröchenig. *Foundations of Time-Frequency Analysis.* Applied and Numerical Harmonic Analysis. Birkhäuser Boston Inc., Boston, MA, 2001.

[21] Matthew A. Herman and Thomas Strohmer. High-resolution radar via compressed sensing. *IEEE Transactions on Signal Processing*, 57(6):2275–2284, June 2009.

[22] Peter J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, June 1985.

[23] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes.* Cambridge University Press, New York, NY, 2010.

[24] Stéphane Jaffard. A density criterion for frames of complex exponentials. *Michigan Math. J.*, 38:339–348, 1991.

[25] Henry J. Landau. Necessary density conditions for sampling and interpolation of certain entire functions. *Acta Mathematica*, 117:37–52, 1967.

[26] Stéphane G. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, CA, 1998.

[27] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[28] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[29] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with application to wavelet decomposition. In *27th Asilomar Conference on Signals, Systems and Computers, 1993*, pages 40–44, 1993.

[30] Götz E. Pfander. Gabor frames in finite dimensions. In Peter G. Casazza and Gitta Kutyniok, editors, *Finite Frames: Theory and Applications*, pages 193–239. Birkhäuser, 2013.

[31] Vera S. Pless and W. Cary Huffman, editors. *Handbook of Coding Theory*, volume 1. Elsevier Science B. V., Amsterdam, The Netherlands, 1998.

[32] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[33] Howard L. Resnikoff and Raymond O. Wells, Jr. *Wavelet Analysis. The Scalable Structure of Information.* Springer-Verlag, New York, NY, 1998. Corrected 2nd printing.

[34] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[35] K. Seip. On the connection between exponential bases and certain related sequences in $L^2(-\pi, \pi)$. *J. Funct. Anal.*, 130:131–160, 1995.

[36] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[37] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[38] D. S. Taubman and Michael W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice.* Kluwer Academic Publishers, Norwell, MA, second edition, 2002.

[39] J. Tian and R. O. Wells, Jr. A lossy image codec based on index coding. In *IEEE Data Compression Conference, DCC '96*, page 456, 1996.

[40] Viterbi School of Engineering, University of Southern California. The USC-SIPI Image Database. http://sipi.usc.edu/database/, 2012.

[41] Gregory K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[42] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error measurement to structural similarity. *IEEE Transactions on Image Processing*, 13(1):1–14, 2004.

[43] Andrew B. Watson, editor. *Digital Images and Human Vision.* MIT Press, Cambridge, MA, 1993.

[44] Mladen V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* A K Peters, Ltd., 1996.