# ADAPTIVE IMAGE COMPRESSION USING SPARSE DICTIONARIES

*Inbal Horev, Ori Bryt*

Signal and Image Processing Lab
Department of Electrical Engineering
Technion, Haifa, Israel

*Ron Rubinstein**

Geometric Image Processing Lab
Department of Computer Science
Technion, Haifa, Israel

## ABSTRACT

Transform coding is a widely used image compression technique, where entropy reduction can be achieved by decomposing the image over a dictionary which provides compaction. Existing algorithms, such as JPEG and JPEG2000, utilize *fixed* dictionaries which are shared by the encoder and decoder. Recently, works utilizing *content-specific* dictionaries show promising results by focusing on specific classes of images and using highly specialized dictionaries. However, such approaches lose the ability to compress arbitrary images.

In this paper we propose an *input-adaptive* compression approach, which encodes each input image over a dictionary specifically trained for it. The scheme is based on the *sparse dictionary* structure, whose compact representation allows relatively low-cost transmission of the dictionary along with the compressed data. In this way, the process achieves both adaptivity and generality. Our results show that although this method involves transmitting the dictionary, it remains competitive with the JPEG and JPEG2000 algorithms.

***Index Terms***— Image compression, sparse representation, dictionary learning, Sparse K-SVD, JPEG.

## 1. INTRODUCTION

Compression of natural images relies on the ability to capture and exploit redundancies found in these images. The most common compression approach, known as *transform coding*, utilizes a *dictionary* of atomic signals, such as the DCT or wavelet dictionaries, over which the image is known to be compressible. The dictionary is typically arranged as a matrix $\mathbf{D} = [\mathbf{d}_1 \mathbf{d}_2 \dots \mathbf{d}_L] \in \mathbb{R}^{N \times L}$, with the columns $\mathbf{d}_i$ constituting the atoms, and $L > N$. Given a signal $\mathbf{x} \in \mathbb{R}^N$, compression is achieved by approximating it as a linear combination of the atoms,

$$\mathbf{x} \approx \mathbf{D}\boldsymbol{\gamma} \;, \tag{1}$$

where the representation vector $\boldsymbol{\gamma}$ is expected to have lower entropy than the entries of $\mathbf{x}$.

When $\mathbf{D}$ is invertible, the representation $\boldsymbol{\gamma}$ can be computed by inverting $\mathbf{D}$ and quantizing the coefficients: $\boldsymbol{\gamma} = \mathcal{Q}(\mathbf{D}^{-1}\mathbf{x})$. This is the case in the JPEG [1] and JPEG2000 [2] compression standards, where $\mathbf{D}$ is the DCT or wavelet dictionary, respectively.

When $\mathbf{D}$ is overcomplete ($L \geq N$), the null space of $\mathbf{D}$ introduces additional degrees of freedom in the choice of $\boldsymbol{\gamma}$, which can be exploited to improve its compressibility. The representation is typically selected by minimizing some penalty function $C(\boldsymbol{\gamma})$ which estimates its compressibility, such as the $\ell^0$ penalty $C(\boldsymbol{\gamma}) = \|\boldsymbol{\gamma}\|_0$ which measures the number of non-zeros in the representation:

$$\hat{\boldsymbol{\gamma}} = \underset{\boldsymbol{\gamma}}{\operatorname{argmin}} \; \|\boldsymbol{\gamma}\|_0 \quad \text{Subject To} \quad \|\mathbf{x} - \mathbf{D}\boldsymbol{\gamma}\|_2^2 \leq \epsilon^2 \;. \tag{2}$$

Here, $\epsilon$ is the error target, controlling the distortion of the compressed signal. This problem is known as the *sparse approximation* problem, and though NP-hard in general, can be approximated by a wide range of techniques [3]. We note that even though the compressibility of a representation is affected by additional factors, sparsity provides a simple and relatively reliable approximation of it.

Transform-based coding schemes generally assume the dictionary $\mathbf{D}$ to be fixed, and built into both the encoder and decoder. This is the case for the JPEG family of algorithms, for instance, which are based on predetermined fixed dictionaries and are targeted at general-purpose image compression. Recently, compression schemes aimed at specific classes of images have been developed, and show substantial gains by employing a *content-specific* dictionary, which is optimized for a specific class of images.

One of the first works to successfully employ a content-specific approach is [4], where the authors propose an algorithm for facial image compression. The algorithm employs a pre-processing geometric alignment step, followed by a sparse approximation of the image patches over a set of pre-trained dictionaries. The method is shown to achieve a dramatic improvement over JPEG2000 for facial imagery owing to the optimized dictionaries, and demonstrates the potential of content-aware compression. However, this method is not easily extendible to other classes of images.

Recently, a method based on iteration-tuned dictionaries (ITDs) has been proposed in [5]. The scheme uses a single hierarchical ITD which is pre-trained for a specific class of

*Corresponding author: ronrubin@cs.technion.ac.il

images, and used to encode the input image patches. The authors test their method with facial images, and show that it can convincingly outperform JPEG and JPEG2000 for this class of images.

As can be seen, the main drawback of the content-specific approaches is their loss of generality, limiting them to encoding images for which a suitable dictionary has been preshared. In this work we adopt a new, *input-adaptive* approach, which aims to restore this generality while preserving adaptivity. Our goal is to increase sparsity by encoding the image over a *specifically-trained* dictionary adapted for the input image. Achieving this goal is not trivial, as it requires transmitting the dictionary along with the compressed data. To control the overhead in sending the dictionary, we propose using a *parametric* dictionary structure, which can be represented by relatively few values. Several such dictionaries have been recently proposed [6].

In this work we focus on the *sparse dictionary* structure [7], which is a simple structure able to represent relatively rich dictionaries. Our compression scheme trains the dictionary specifically for the input image, and encodes it as part of the compressed stream. In this way, the compression method can accommodate a wide range of images, since it imposes fewer assumptions on their behavior. Our simulations show that even though our method must encode the dictionary as part of the stream, it consistently outperforms JPEG compression, and comes close to JPEG2000 in a few cases. We view these results as significant and encouraging, and demonstrate the feasibility of the input-adaptive approach, opening the door to further research.

This paper is organized as follows: In section 2 we review the sparse dictionary structure, which forms the core of our algorithm. The compression scheme is described in section 3, followed by results in section 4. We conclude and discuss some future directions in section 5.

## 2. SPARSE DICTIONARIES

The sparse dictionary structure is a parametric dictionary model recently proposed as a means of bridging the gap between analytic and trained dictionaries [7]. It is a simple and effective structure based on sparsity of the atoms over a known *base dictionary*. This view suggests that dictionaries describing different images have a common underlying explanation in the form of a *universal* base dictionary. This base dictionary consists of a fixed set of fundamental signals, from which all observable dictionary atoms are formed.

Formally, the sparse dictionary model represents a dictionary $\mathbf{D}$ as the product of a pre-specified base dictionary $\mathbf{\Phi}$ and a sparse representation matrix $\mathbf{A}$:

$$\mathbf{D} = \mathbf{\Phi}\mathbf{A} \ . \tag{3}$$

Thus, each atom in $\mathbf{D}$ is a sparse combination of atoms from $\mathbf{\Phi}$. For simplicity, we assume $\mathbf{A}$ has a fixed number of non-zeros per column, so $\|\mathbf{a}_i\|_0 \leq p$ for some $p$. The base dictionary $\mathbf{\Phi}$ is a *fixed* non-adaptive dictionary, such as the DCT dictionary, and is part of the model. An example sparse dictionary, trained for $6 \times 6$ natural image patches using an overcomplete DCT base dictionary [7], is shown in Fig. 1.

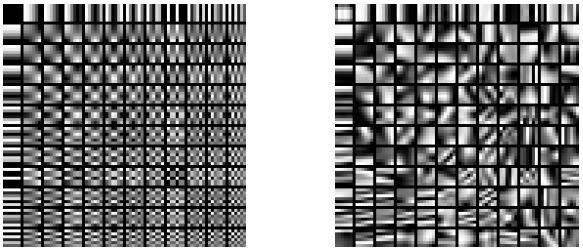

**Fig. 1**. Left: Overcomplete DCT base dictionary. Right: Sparse K-SVD dictionary trained over this base dictionary.

Benefits of this model include adaptability (via modification of $\mathbf{A}$), efficiency (assuming $\mathbf{\Phi}$ has an efficient algorithm), and compact representation (as only $\mathbf{A}$ requires specification). Training the sparse dictionary is done using the *Sparse K-SVD* algorithm [7], which efficiently adapts the matrix $\mathbf{A}$ given a set of examples. We refer the reader to [7] for a description of the algorithm.

## 3. ADAPTIVE IMAGE COMPRESSION

The adaptive encoding process is summarized in Fig. 2. The process begins by partitioning the image into non-overlapping patches and subtracting the mean (DC) value from each. The DC values are subsequently quantized, and their running differences are entropy coded. The DC-free patches, which contain the bulk of the image information, are used to train a sparse dictionary using Sparse K-SVD. As the base dictionary, we use the overcomplete DCT, which is known to be relatively efficient for representing small image patches.

The outcome of the training is a matrix $\mathbf{A}$ describing an *image-specific* dictionary for representing the image patches. This matrix undergoes quantization and is then used to encode the DC-free patches. We perform sparse coding over the quantized dictionary $\mathbf{D}_q = \mathbf{\Phi}\mathbf{A}_q$ to allow inversion of the process at the decoder. For the sparse coding, we use a variant of Orthogonal Matching Pursuit (OMP) [8] which we name *Global OMP*. The sparse coding step produces a sparse matrix $\mathbf{\Gamma}$ with the sparse representations of the patches as its columns, and it is subsequently quantized to form $\mathbf{\Gamma}_q$. Finally, both $\mathbf{A}_q$ and $\mathbf{\Gamma}_q$ are fed to a sparse matrix encoder which generates the compressed stream of the DC-free content. The complete compressed stream consists of the encoded DC values and the two compressed sparse matrices. Note that the base dictionary itself is fixed in our system and is not transmitted. Decoding the stream is straightforward and includes reversing the sparse matrix encoding, computing the DC-free patches $\mathbf{X} = \mathbf{\Phi}\mathbf{A}_q\mathbf{\Gamma}_q$, and restoring the encoded DC values.

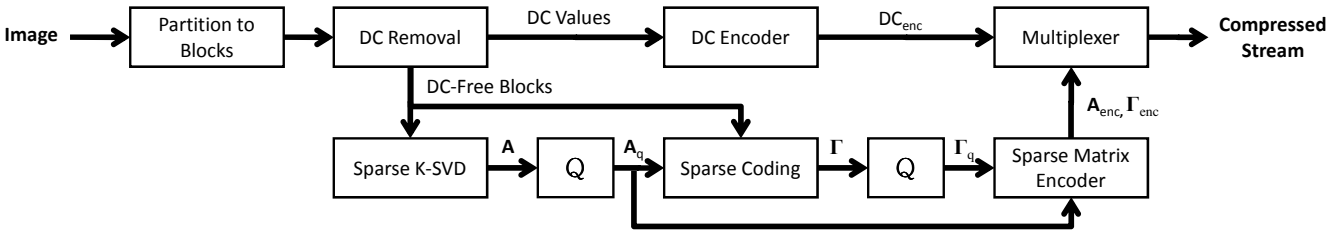


**Fig. 2**. The encoding scheme.

### 3.1. Global OMP

Our implementation of the encoder accepts a target PSNR as the control of the output rate. In the sparse coding stage, this target can be achieved by sparse coding each image patch independently, setting the error target in (2) to

$$\epsilon^2 = b \cdot 10^{\frac{-PSNR}{10}} \cdot I_{max}^2, \tag{4}$$

where $b$ denotes the number of pixels in a patch, and $I_{max}$ is the maximum intensity value. Alternatively, we can lift the uniform-distortion constraint and allow the error to be distributed *arbitrarily* among the patches. This results in a more flexible sparse coding scheme which potentially achieves higher sparsity. In this way, we solve a *global* sparse coding problem for all image patches simultaneously, given by

$$\underset{\mathbf{\Gamma}}{\text{Min}} \|\mathbf{\Gamma}\|_0 \quad \text{Subject To} \quad \|\mathbf{Y} - \mathbf{D}_q\mathbf{\Gamma}\|_F^2 \leq \epsilon_g^2 \,. \tag{5}$$

Here, $\mathbf{Y}$ is a matrix with the image patches as its columns, and $\epsilon_g$ is the global error target for the image.

Solving problem (5) is equivalent to sparse coding the column-stack vector representation of $\mathbf{Y}$ over the dictionary $\mathbf{I} \otimes \mathbf{D}$, which can be solved using OMP. The greedy process can be implemented efficiently as it amounts to adding an atom to a single patch each iteration, and hence all computations are performed locally at the patch level. We name this process *Global OMP*, and use it for both the dictionary training and the sparse coding steps of the scheme.

### 3.2. Quantization

We quantize the non-zeros in $\mathbf{A}$ and $\mathbf{\Gamma}$ using a uniform quantizer. While the distribution of these values is highly non-uniform, it is known that using a uniform quantizer followed by entropy coding generally outperforms a non-uniform quantizer. A side effect of the quantization of $\mathbf{\Gamma}$ is that the PSNR target achieved in the sparse coding step is lost. To restore the desired PSNR target, we employ a simple iterative refinement process, in which coefficients are added to $\mathbf{\Gamma}$ to compensate for the quality loss. This is done by raising the PSNR target for the Global OMP by the amount lost due to quantization. We then continue the greedy selection process from the point it terminated until the updated PSNR target is reached. The resulting coefficients are quantized, and if necessary the PSNR target is raised again and the process repeats until reaching the user-specified PSNR target. Since the sparse coding is continued rather than restarted, the overhead of these repetitions is small. For more details see [9].

### 3.3. Sparse Matrix Encoding

Our sparse matrix encoder represents the matrices $\mathbf{A}_q$ and $\mathbf{\Gamma}_q$ in column-compressed form. It encodes the non-zero coefficients via entropy coding, and their locations via difference coding of the row indices followed by entropy coding.

A useful observation is that the order of the columns of $\mathbf{A}$ is a degree of freedom of the representation. Indeed, we can apply any permutation to the columns of $\mathbf{A}$, along with the same permutation to the rows of $\mathbf{\Gamma}$, without altering the product $\mathbf{A\Gamma}$. This freedom can be used to improve the compressibility of the row indices in $\mathbf{\Gamma}$. In this work we order the columns of $\mathbf{A}$ in a descending order of usage, which results in a concentration of the non-zeros in $\mathbf{\Gamma}$ near the top of the matrix. Thus, the entropy of the index differences is reduced.

### 3.4. Entropy Coding

The entropy coding in this work is implemented using an arithmetic coder. We recall that given a set of symbols, the arithmetic coder and decoder require the symbol probabilities $\{p_i\}$ as side information. To avoid sending floating-point numbers, we quantize and transmit the log-probabilities $\log_2(1/p_i)$. These values represent the optimal codeword lengths of the symbols, and thus have a relatively small range which can be uniformly quantized. We have found that using very few bits (5-6) for the quantized values results in practically no increase to the code length, while reducing the overhead of the side information.

## 4. RESULTS

We have tested the proposed scheme on images from the CVG-Granada dataset[1]. Rate-Distortion graphs for eight images are presented in Fig. 3, and compare our method to JPEG and JPEG2000. As can be seen, our scheme consistently outperforms JPEG, and comes close to JPEG2000 in a few cases.

[1] http://decsai.ugr.es/cvg/dbimagenes/

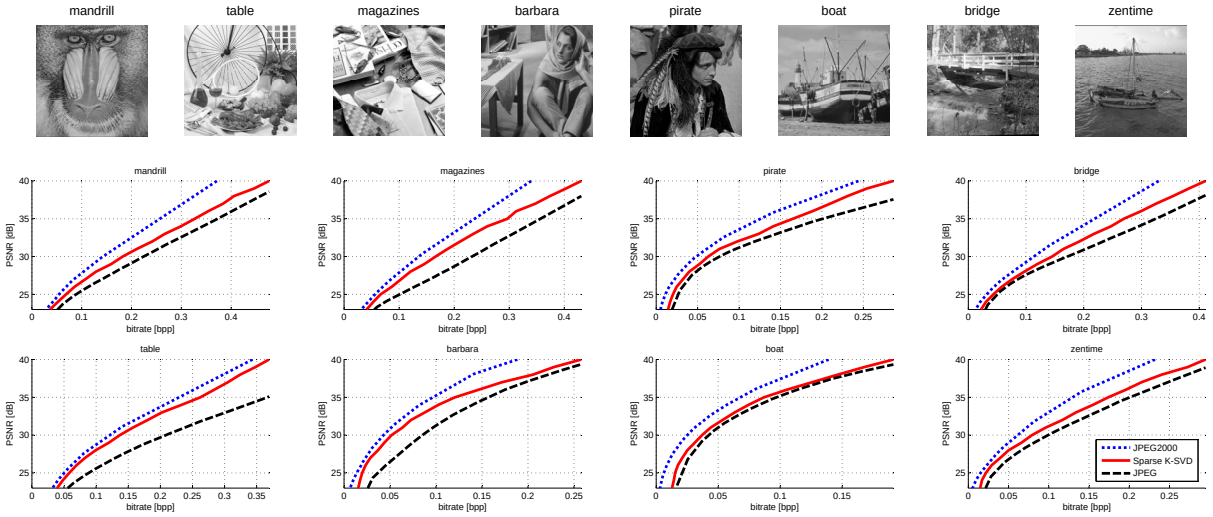


**Fig. 3**. Rate-Distortion graphs of the sparse K-SVD method compared to JPEG and JPEG2000.

We note that the compression process involves the choice of a few parameters, including patch size, dictionary size, atom sparsity, and quantization step sizes. Our system implements a semi-automatic parameter tuning process which dynamically determines these values for the given image, with no user intervention. For additional details and results see [9].

## 5. CONCLUSION AND FUTURE DIRECTIONS

This work has presented a new image compression scheme based on *input-adaptive* dictionaries. The system is unique in that it encodes the image over a dictionary *specifically trained* for it. This approach, which requires transmission of the dictionary as part of the compressed stream, is made possible owing to the compact representation of the sparse dictionary. We have shown that despite the overhead in sending the dictionary, our system consistently outperforms JPEG, which is a similar patch-based scheme, but utilizes a pre-shared fixed dictionary. Indeed, while our current implementation does not reach JPEG2000 performance, our results remain significant in that they demonstrate the feasibility and potential of the adaptive approach. Such an approach, as far as the authors are aware of, has so far been considered impractical.

Many enhancements to the scheme could be introduced. Most notably, working with several image scales could more efficiently represent differently-sized features, as well as eliminate the need to select a patch size for each input individually. Another way to achieve such behavior is to apply the scheme using a wavelet (or other multi-scale) base dictionary, which would also reduce blockiness. Finally, we have observed that the encoded indices occupy a significant part of the compressed stream. Thus, discovering or creating regular patterns in $\mathbf{\Gamma}$ could improve compression efficiency.

## 6. ACKNOWLEDGEMENTS

The authors would like to thank Prof. Michael Elad, Prof. David Malah, and the lab staff for their guidance and helpful advice throughout this work.

## 7. REFERENCES

[1] W B Pennebaker and J L Mitchell, *JPEG still image data compression standard*, Springer, New York, 1993.

[2] D S Taubman and M W Marcellin, *JPEG2000: Image compression fundamentals, standards and practice*, Kluwer Academic Publishers Norwell, MA, 2001.

[3] M Elad, *Sparse and redundant representations - From theory to applications in signal and image processing*, Springer, 2010.

[4] O Bryt and M Elad, "Compression of facial images using the K-SVD algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–283, 2008.

[5] J Zepeda, C Guillemot, and E Kijak, "Image Compression using the Iteration-Tuned and Aligned Dictionary," in *Proceedings of IEEE ICASSP'11*, 2011, pp. 793–796.

[6] R Rubinstein, A M Bruckstein, and M Elad, "Dictionaries for sparse representation modeling," *IEEE Proceedings*, vol. 98, no. 6, pp. 1045–1057, 2010.

[7] R Rubinstein, M Zibulevsky, and M Elad, "Double sparsity: learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[8] Y C Pati, R Rezaiifar, and P S Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *1993 Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

[9] I Horev, O Bryt, and R Rubinstein, "Adaptive Image Compression Using Sparse Dictionaries," *Technical Report CS-2011-10, Computer Science, Technion*, 2011.