Orthogonal Matching Pursuit and K-SVD for Sparse Encoding

Manny Ko
Senior Software Engineer, Imaginations Technologies
Robin Green
SSDE, Microsoft Xbox ATG

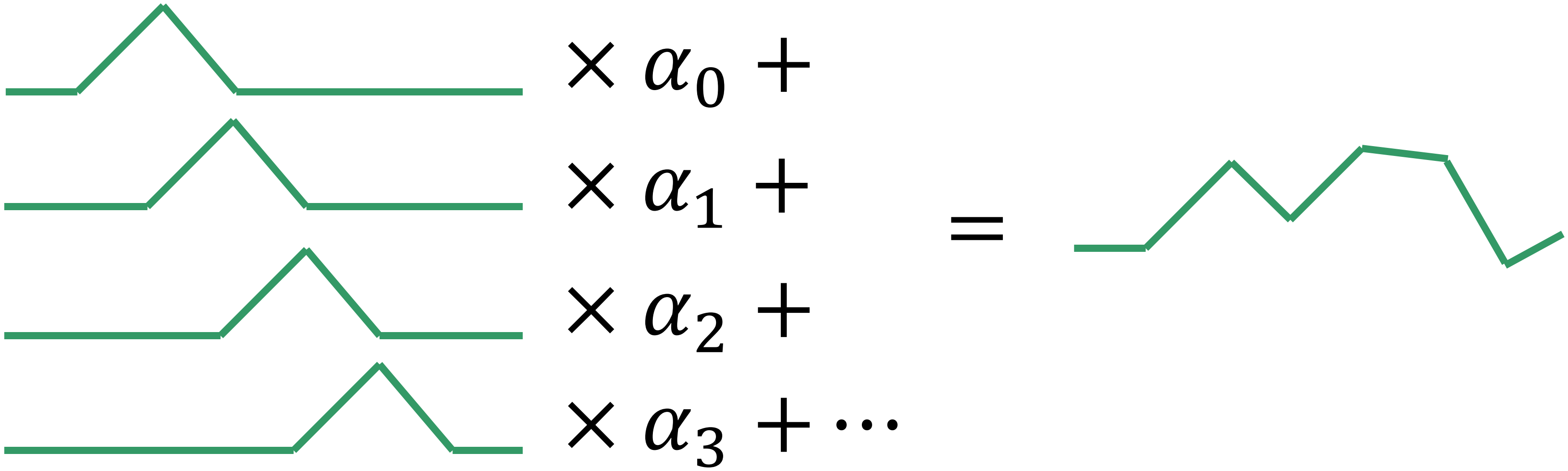
GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 25-29, 2013
EXPO DATES: MARCH 27-29
2013

Outline

- Signal Representation
- Orthonormal Bases vs. Frames
- Dictionaries
- The Sparse Signal Model
- Matching Pursuit
- Implementing Orthonomal Matching Pursuit
- Learned Dictionaries and KSVD
- Image Processing with Learned Dictionaries
- OMP for GPUs

Representing Signals

- We represent most signals as linear combinations of things we already know, called a *projection*



Representing Signals

- Each function we used is a *basis* and the scalar weights are *coefficients*

$$\hat{x}(t) = \sum_{i=0}^{N} b_i(t)\alpha_i$$

- The reconstruction is an *approximation* to the original x
  - We can measure and control the error $\|x - \hat{x}\|^2$

Orthonormal Bases (ONBs)

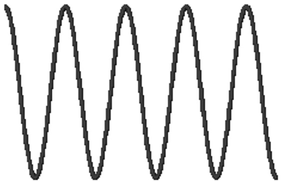
- The simplest way to represent signals is using a set of *orthonormal bases*

$$\int_{-\infty}^{+\infty} b_i(t) b_j(t)\, dt = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Example ONBs

- Fourier Basis

$b_k(t) = e^{i2pkt}$

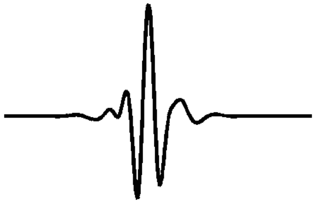


- Gabor Functions

$b_{k,n}(t) = \omega(t - bn)e^{i2pkt}$

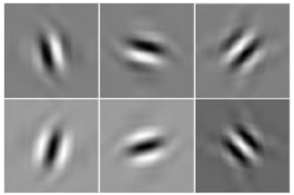
- Wavelets

$b_{m,n}(t) = a^{-m/2}x(a^{-m}t - bm)$



- Contourlet

$b_{j,k,\mathbf{n}}(t) = \lambda_{j,k}\left(t - 2^{j-1}\mathbf{S}_k\mathbf{n}\right)$



Benefits of ONB

- Analytic formulations
- Well understood mathematical properties
- Fast algorithms for projection

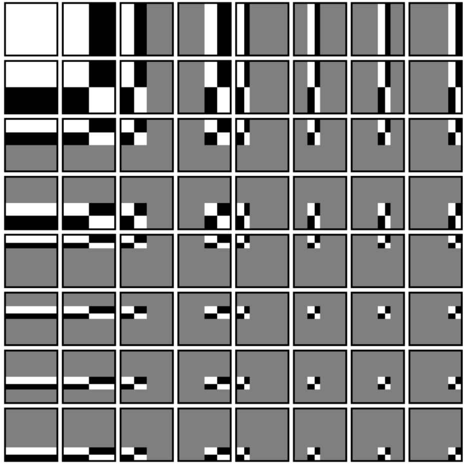
Limitations

- Orthonormal bases are optimal only for specific synthetic signals
  - If your signal looks exactly like your basis, you only need one coefficient
- Limited expressiveness, all signals behave the same
- Real world signals often take a lot of coefficients
  - Just truncate the series, which leads to *artifacts* like *aliasing*

Smooth vs. Sharp

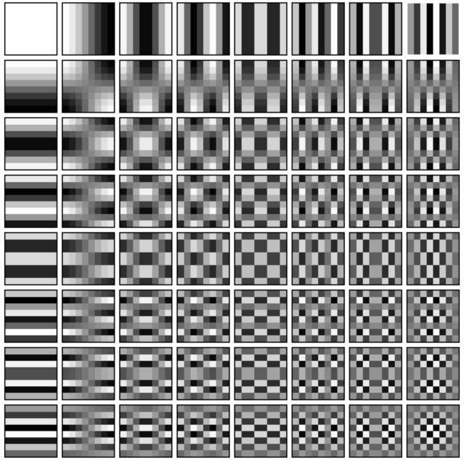
Haar Wavelet Basis

- Sharp edges
- Local support



Discrete Cosine Transform

- Smooth signals
- Global support

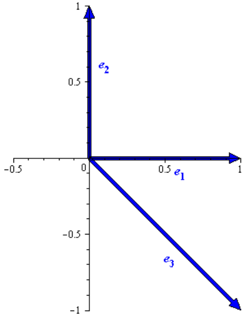


Overcomplete Bases

- Frames are *overcomplete bases*
  - There is now more than one way to represent a signal

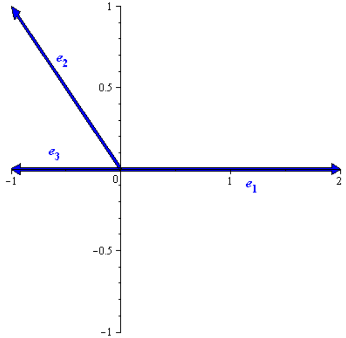
$$\Phi = [e_1 | e_2 | e_3]$$

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}$$



$$\tilde{\Phi} = [\tilde{e}_1 | \tilde{e}_2 | \tilde{e}_3]$$

$$= \begin{bmatrix} 2 & -1 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$



- By relaxing the ONB rules on *minimal span*, we can better approximate signals using more coefficients

Dictionaries

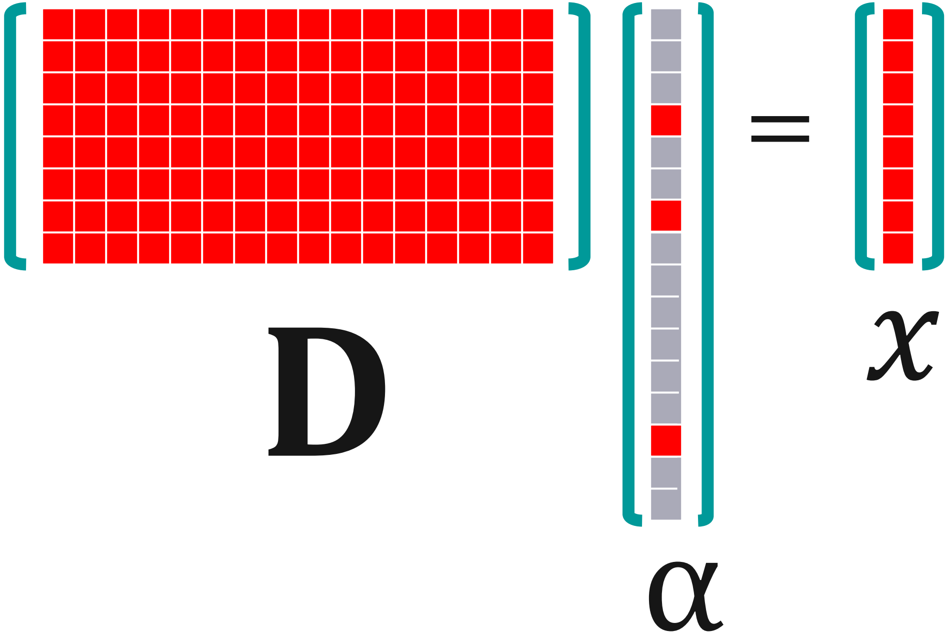
- A *dictionary* is an overcomplete basis made of *atoms*
- A signal is represented using a linear combination of only a few atoms

$$\sum_{i \in I} d_i \, \alpha_i = x$$

$$\boldsymbol{D}\alpha = x$$

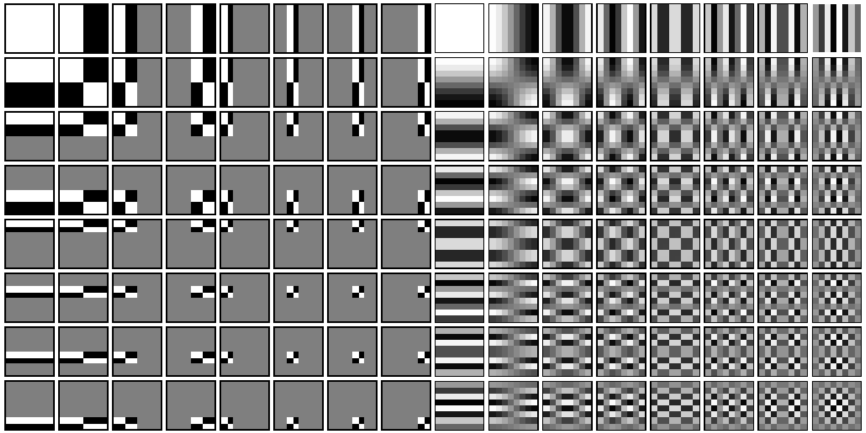
- Atoms work best when *zero-mean* and *normalized*

Dictionaries



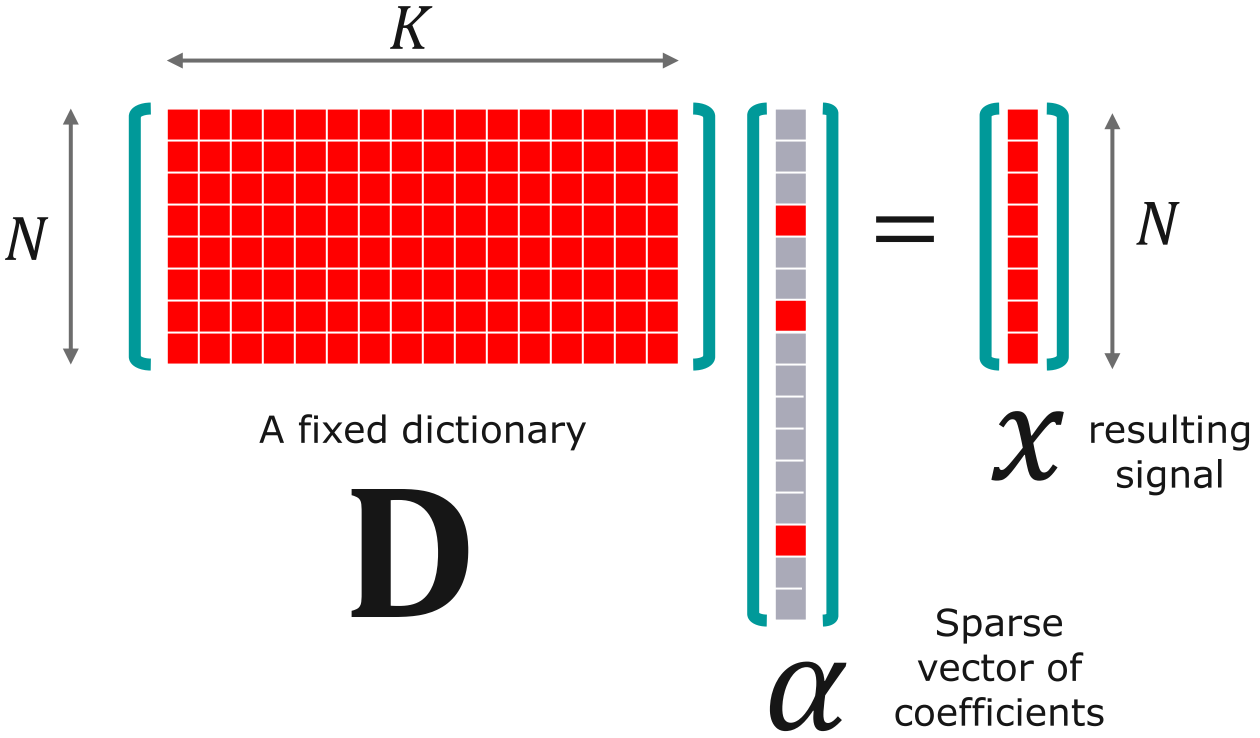
Mixed Dictionaries

- A dictionary of Haar + DCT gives the best of both worlds



But now how do we pick which coefficients to use?

The Sparse Signal Model



The Sparse Signal Model

It's Simple

- Every result is built from a combination of a few atoms

It's Rich

- It's a general model, signals are a union of many low dimensional parts

It's Used Everywhere

- The same model is used for years in Wavelets, JPEG compression, anything where we've been throwing away coefficients

Solving for Sparsity

What is the minimum number of coefficients we can use?

1. **Sparsity Constrained**
   keep adding atoms until we reach a maximum count

$$\alpha = \underset{\alpha}{\operatorname{argmin}} \ \|\mathbf{D}\alpha - x\|_2^2 \quad \text{s.t.} \quad \|\alpha\|_0 \leq K$$

2. **Error Constrained**
   Keep adding atoms until we reach a certain accuracy

$$\alpha = \underset{\alpha}{\operatorname{argmin}} \ \|\alpha\|_0 \quad \text{s.t.} \quad \|\mathbf{D}\alpha - x\|_2^2 \leq \epsilon$$

Naïve Sparse Methods

- We can directly find α using Least Squares

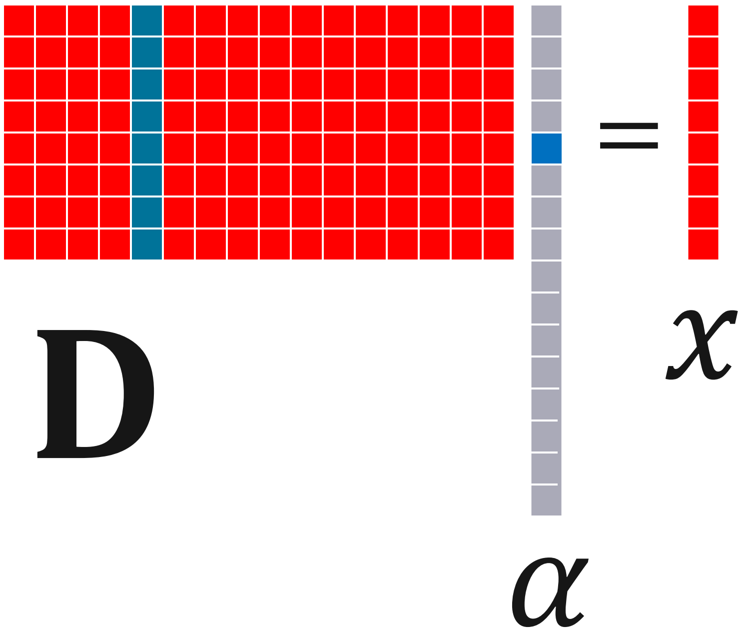
    1. set $L = 1$
    2. generate $S = \{ \mathcal{P}_L(\boldsymbol{D}) \}$
    3. for each set solve the Least Squares problem $\min_{\alpha} \|\mathbf{D}\alpha - x\|_2^2$ where $supp(\alpha) \in S_i$
    4. if LS error $\leq \epsilon$ finish!
    5. set $L = L + 1$
    6. goto 2

- Given K=1000 and L=10 at one LS per nanosecond this would complete in ~8 million years.

Greedy Methods

Matching Pursuit

1. Set the residual $r = x$
2. Find an unselected atom that best matches the residual $\|\mathbf{D}\alpha - r\|$
3. Re-calculate the residual from matched atoms $r = x - \mathbf{D}\alpha$
4. Repeat until $\|r\| \leq \epsilon$



Problems with Matching Pursuit (MP)

- If the dictionary contains atoms that are very similar, they tend to match the residual over and over

- Similar atoms do not help the basis *span* the space of representable values quickly, wasting coefficients in a *sparsity constrained* solution

- Similar atoms may match strongly but will not have a large effect in reducing the absolute error in an *error constrained* solution

Orthogonal Matching Pursuit (OMP)

- Add an Orthogonal Projection to the residual calculation

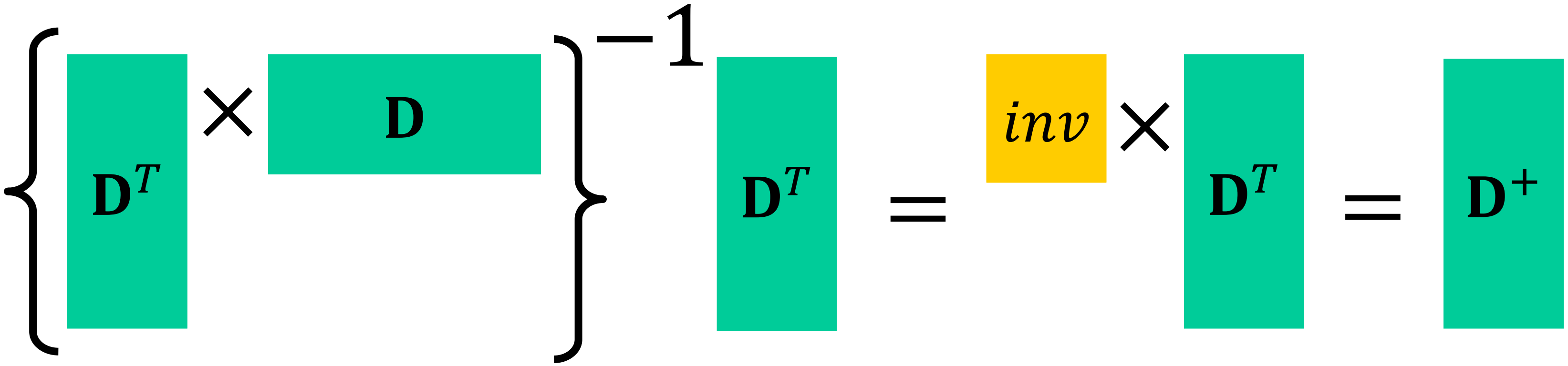
1. set $I := \{\emptyset\},\ r := x,\ \gamma := 0$
2. while ($stopping\ test\ false$) do
3. $\quad k := \underset{k}{\operatorname{argmax}} \left| d_k^T r \right|$
4. $\quad I := (I, k)$
5. $\quad \gamma_I := (\mathbf{D}_I)^+ x$
6. $\quad r := x - \mathbf{D}_I \gamma_I$
7. end while

Uniqueness and Stability

- OMP has guaranteed reconstruction (provided the dictionary is overcomplete)
- By projecting the input into the range-space of the atoms, we know that that the residual will be orthogonal to the selected atoms
- Unlike Matching Pursuit (MP) that atom, and all similar ones, will not be reselected so more of the space is spanned per iteration

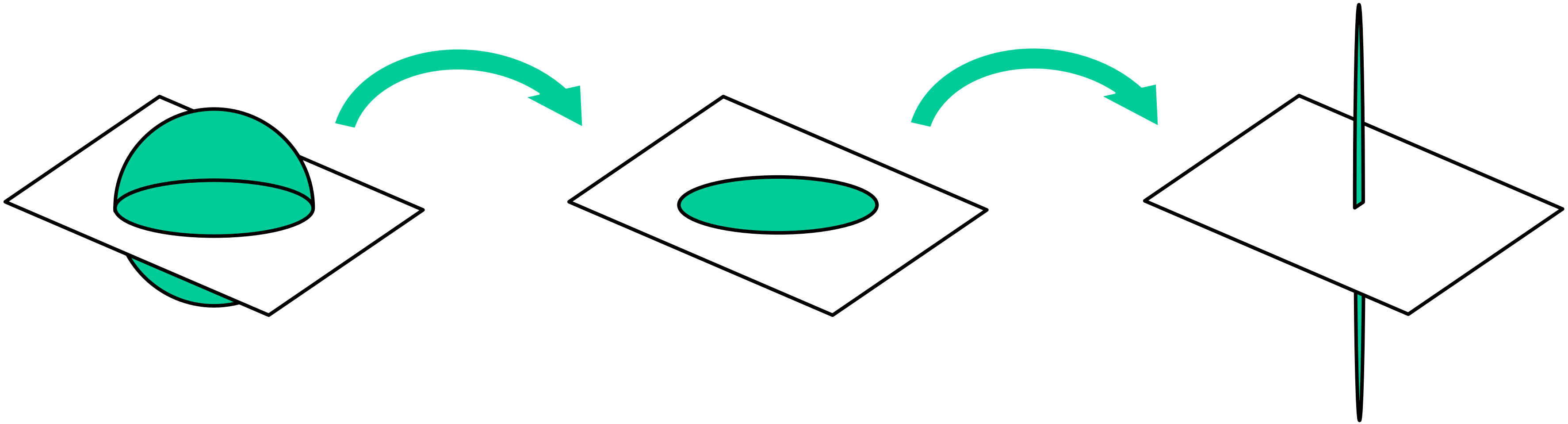
Orthogonal Projection

- If the dictionary $\mathbf{D}$ was square, we could use an inverse
- Instead we use the Pseudo-inverse $\mathbf{D}^{+} = (\mathbf{D}^{T}\mathbf{D})^{-1}\mathbf{D}^{T}$



Pseudoinverse is Fragile

- In floating point, the expression $\left(\mathbf{D}^{\mathrm{T}}\mathbf{D}\right)^{-1}$ is notoriously numerically troublesome – the classic FP example
  - Picture mapping all the points on a sphere using $\mathbf{D}^{T}\mathbf{D}$ then inverting



Implementing the Pseudoinverse

- To avoid this, and reduce the cost of inversion, we can note that $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ is always *symmetric* and *positive definite*
- We can break the matrix into two triangular matrices using *Cholesky Decomposition* $\mathbf{A} = \mathbf{L}\mathbf{L}^T$
- *Incremental Cholesky Decomp* reuses the results of the previous iteration, adding a single new row and column each time

$$\mathbf{L}_{new} = \begin{bmatrix} \mathbf{L} & \underline{0} \\ \underline{w}^T & \sqrt{1 - \underline{w}^T\underline{w}} \end{bmatrix} \quad \text{where} \quad \underline{w} = \mathbf{L}^{-1} D_I d_k$$

OMP-Cholesky

1. set $I := \{\emptyset\},\ L := [1],\ r := x,\ \gamma := 0,$
   $\alpha := \mathbf{D}^T x,\ n := 1$
2. while $(stopping\ test\ false)$ do
3. $k := \underset{k}{\operatorname{argmax}} \left| d_k^T r \right|$
4. if $n > 1$ then
   $w := \text{Solve for } w \{ \mathbf{L}w = \mathbf{D}_I^T d_k \}$
   $$\mathbf{L} := \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ w^T & \sqrt{1 - w^T w} \end{bmatrix}$$
5. $I := (I, k)$
6. $\gamma_I := \text{Solve for } c \ \{ \mathbf{L}\mathbf{L}^T c = \alpha_I \}$
7. $r := x - \mathbf{D}_I \gamma_I$
8. $n := n + 1$
9. end while

OMP compression of Barbara

2 atoms

3 atoms

4 atoms

Batch OMP (BOMP)

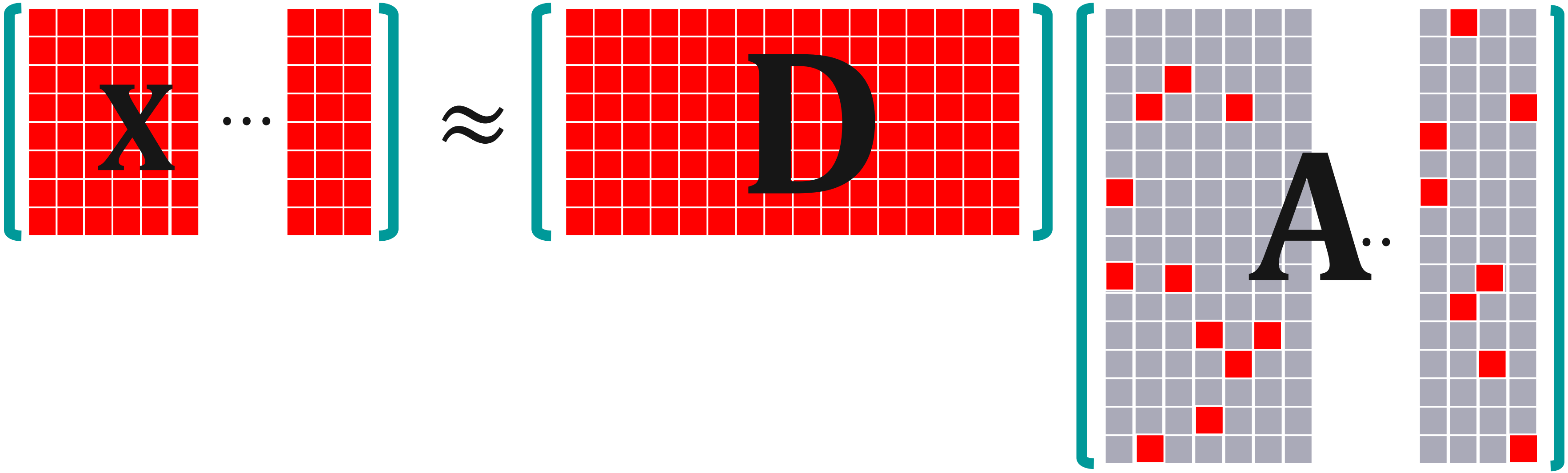
- By pre-computing matrices, Batch OMP can speed up OMP on large numbers (>1000) of inputs against one dictionary
- To avoid computing $\mathbf{D}^T\underline{r}$ at each iteration

$$\begin{aligned}\mathbf{D}^T\underline{r} &= \mathbf{D}^T(\underline{x} - \mathbf{D}_I(\mathbf{D}_\mathbf{I})^+\underline{x}) \\ &= \mathbf{D}^T\underline{x} - \mathbf{G}_I(\mathbf{D}_I)^+\underline{x} \\ &= \mathbf{D}^T\underline{x} - \mathbf{G}_I(\mathbf{D}_I^T\mathbf{D}_I)^{-1}\mathbf{D}_I^T\underline{x} \\ &= \mathbf{D}^T\underline{x} - \mathbf{G}_I(\mathbf{G}_{I,I})^{-1}\mathbf{D}_I^T\underline{x}\end{aligned}$$

- Precompute $\mathbf{D}^T\underline{x}$ and the Gram-matrix $\mathbf{G} = \mathbf{D}^T\mathbf{D}$

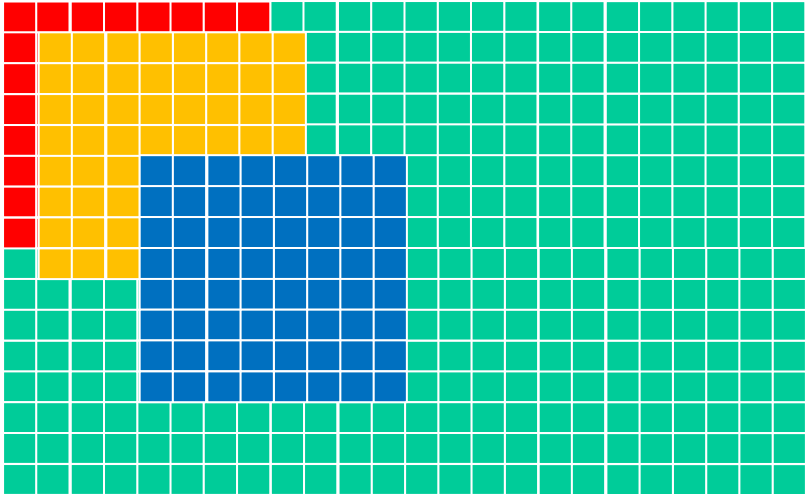
Learned Dictionaries and K-SVD

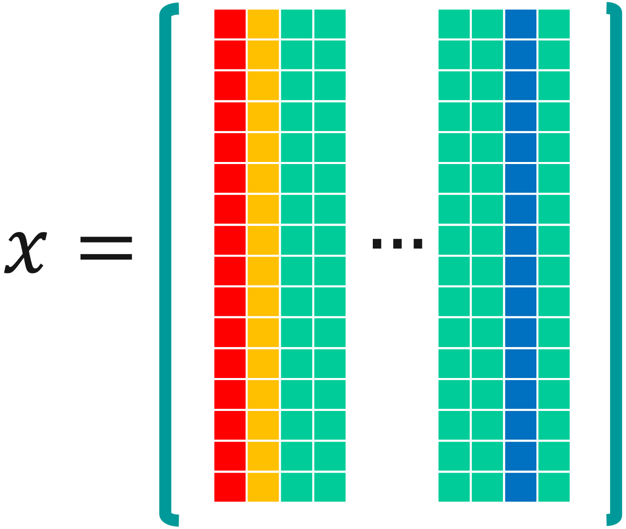
- OMP works well for a fixed dictionary, but it would work better if we could optimize the dictionary to fit the data



Sourcing Enough Data

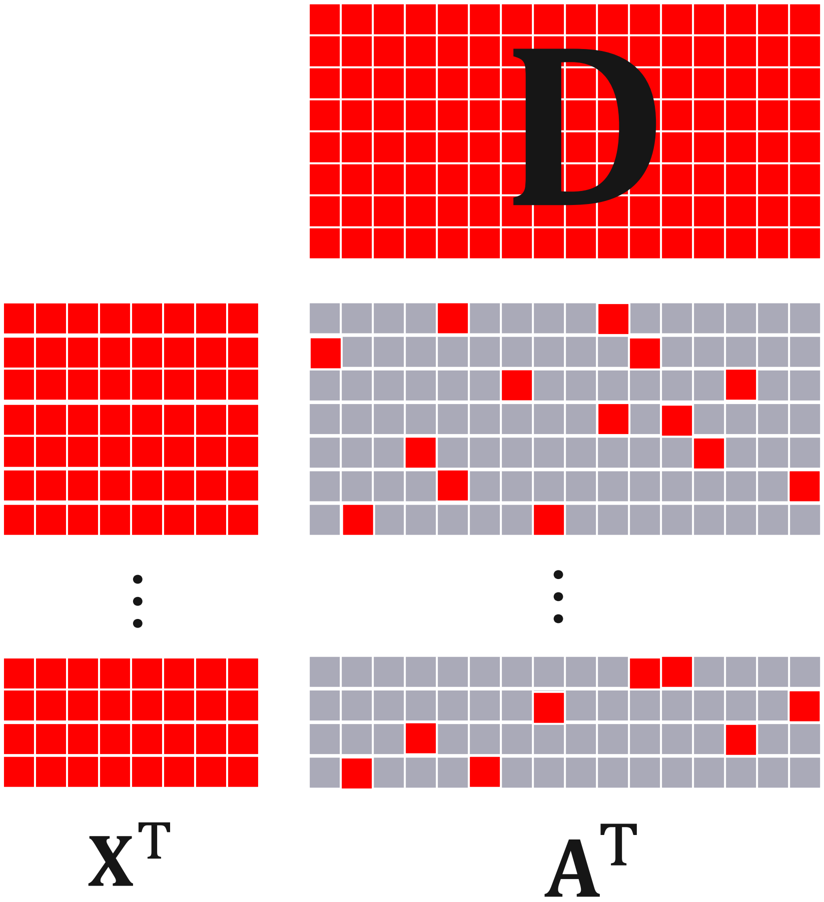
- For training you will need a large number of samples compared to the size of the dictionary.
- Take blocks from all integer offsets on the pixel grid





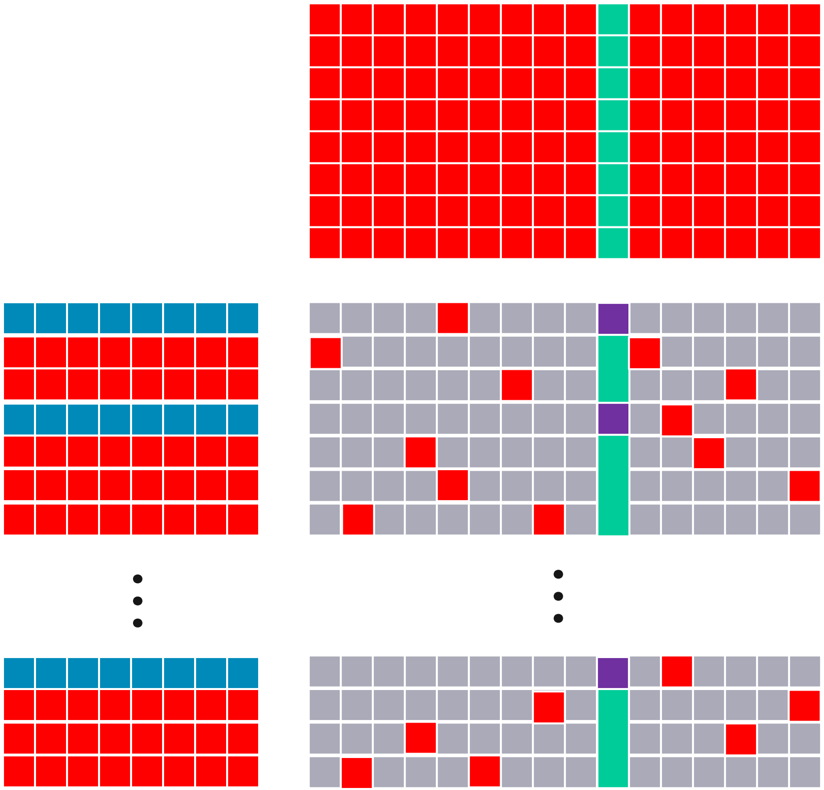
1. Sparse Encode

- Sparse encode all entries in $\mathbf{x}$. Collect these sparse vectors into a square array $\mathbf{A}$

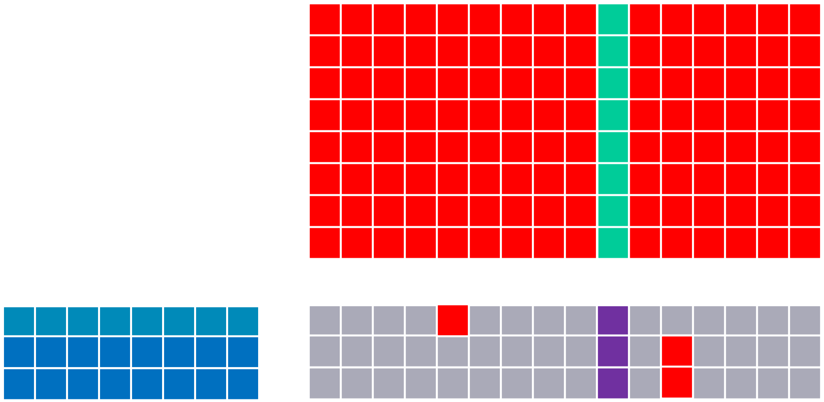


2. Dictionary Update

- Find all $\mathbf{X}$ that use atom in column $\boldsymbol{d}_k$



2. Dictionary Update



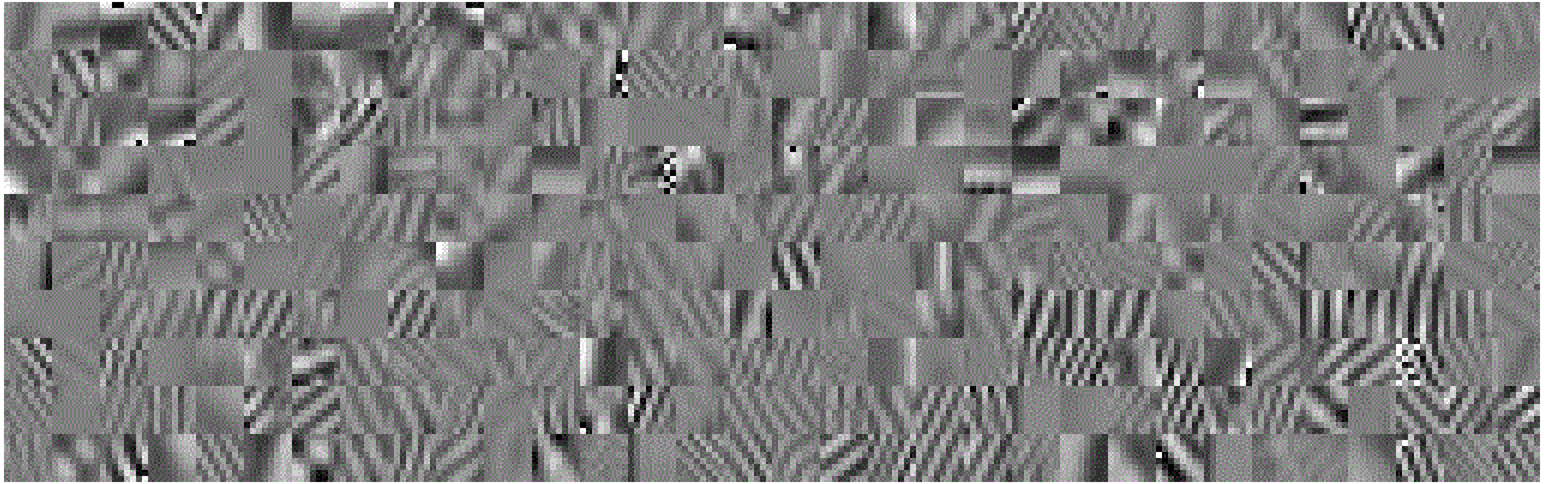
a

- Find all $\mathbf{X}$ that use atom in column d_k
- Calculate the error without d_k by $\mathbf{E} = \mathbf{X}_I - \sum_{i \neq k} d_i \mathbf{A}_\mathbf{i}$
- Solve the LS problem:

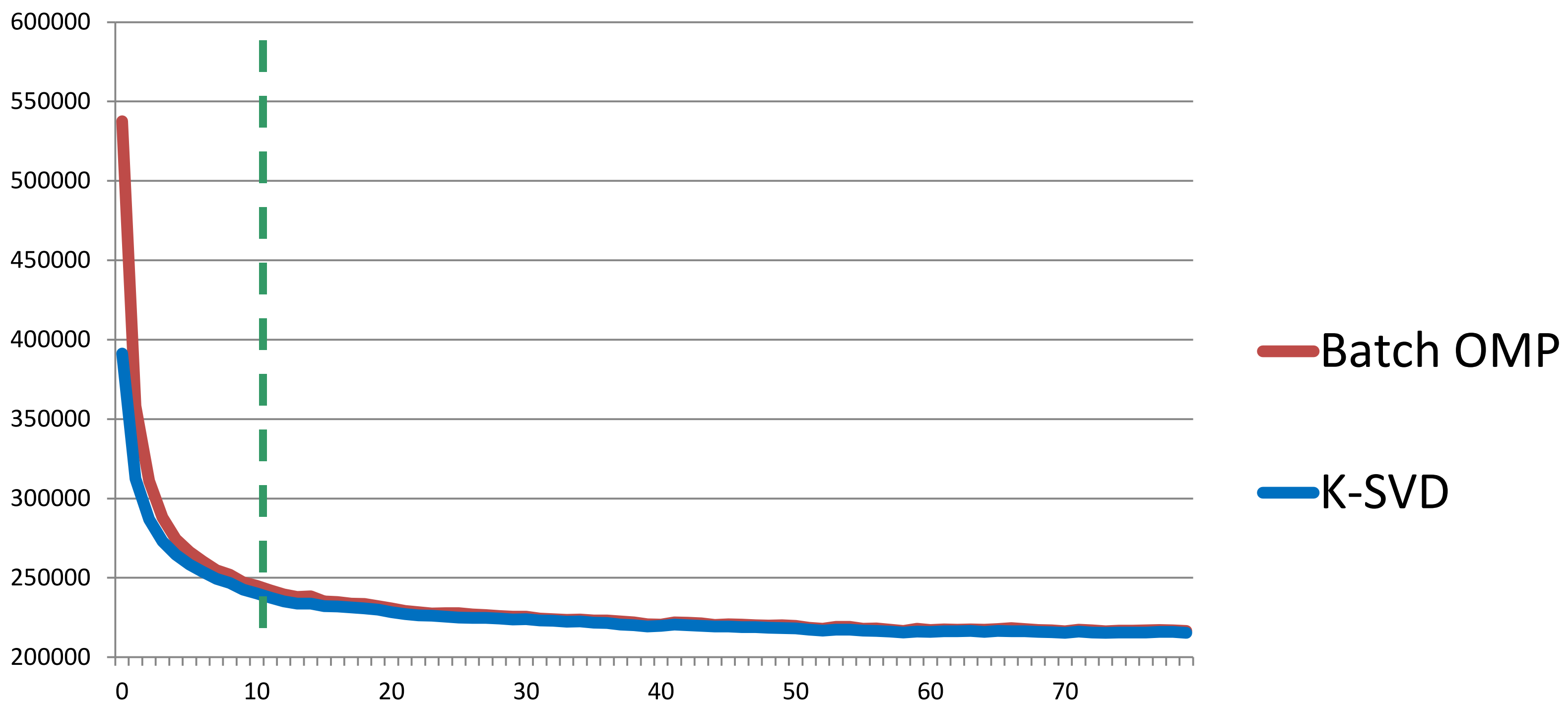
$$\{d, a\} = \underset{d,a}{\text{Argmin}} \|\mathbf{E} - da^T\|_F^2 \quad s.t. \quad \|d\|_2 = 1$$

- Update d_i with the new d and $\mathbf{A}$ with the new a

Atoms after K-SVD Update



How many iterations of update?



Sparse Image Compression

- As we have seen, we can control the number of atoms used per block
- We can also specify the exact size of the dictionary and optimize it for each data source
- The resulting coefficient stream can be coded using a Predictive Coder like Huffman or Arithmetic coding

Domain Specific Compression

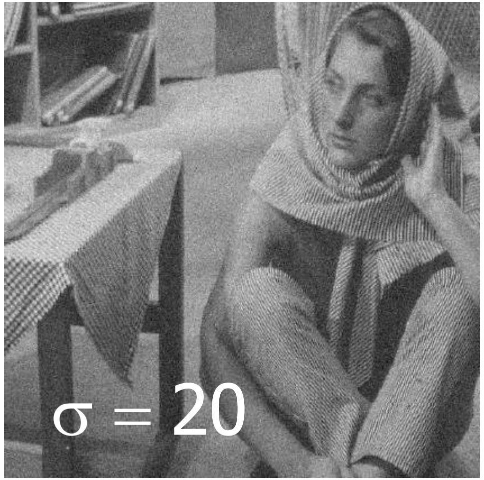
- Using just 550 bytes per image

1. Original
2. JPEG
3. JPEG2000
4. PCA
5. KSVD per block

Sparse Denoising

- Uniform noise is incompressible and OMP will reject it
- KSVD can train a denoising dictionary from noisy image blocks

Source

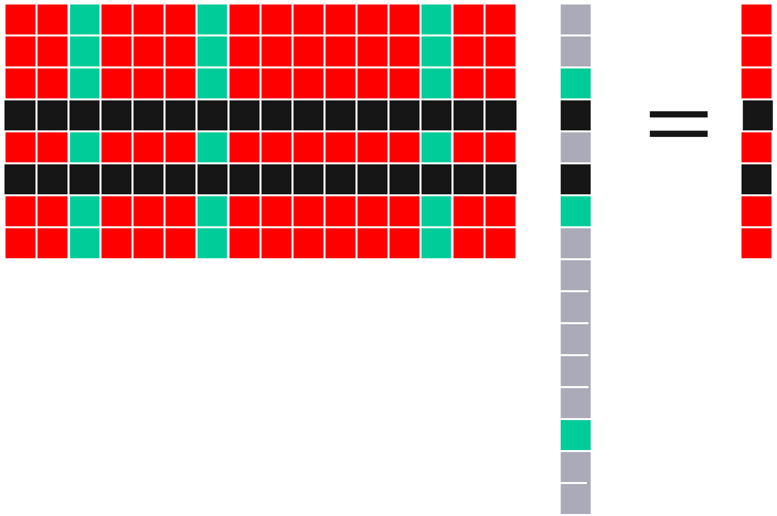


Noisy image

Result 30.829dB

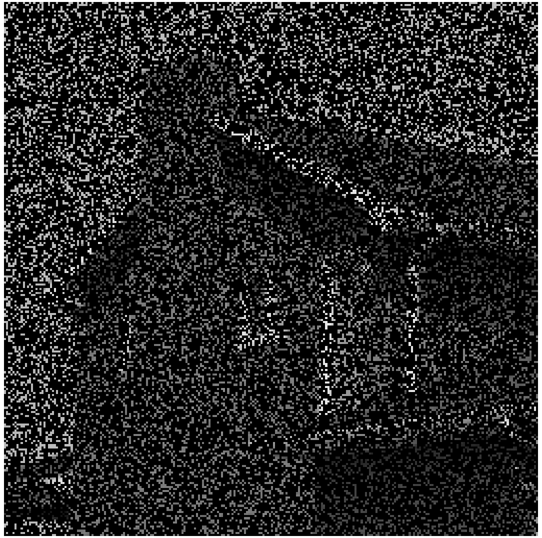
Sparse Inpainting

- Missing values in $\mathbf{x}$ means missing rows in $\mathbf{D}$
- Remove these rows and refit α to recover $\mathbf{x}$
  - If α was sparse enough, the recovery will be perfect



Sparse Inpainting

Original



80% missing

Result

Super Resolution

Super Resolution

The Original

Bicubic Interpolation

SR result

Block compression of Voxel grids

- "*A Compression Domain output-sensitive volume rendering architecture based on sparse representation of voxel blocks*" Gobbetti, Guitian and Marton [2012]

- COVRA sparsely represents each voxel block as a dictionary of 8x8x8 blocks and three coefficients

- The voxel patch is reconstructed only inside the GPU shader so voxels are decompressed just-in-time

- Huge bandwidth improvements, larger models and faster rendering

Thank you to:

- Ron Rubstein & Michael Elad
- Marc LeBrun
- Enrico Gobbetti