Abdul Qayyum

Lecturer at University of Burgundy, France

- Postdoc in Electrical and Informatics Engineering
- PhD in Electrical & Electronics Engineering
- Masters in Electronics Engineering
- Bachelor in Computer Engineering

Collaborations & Expertise:

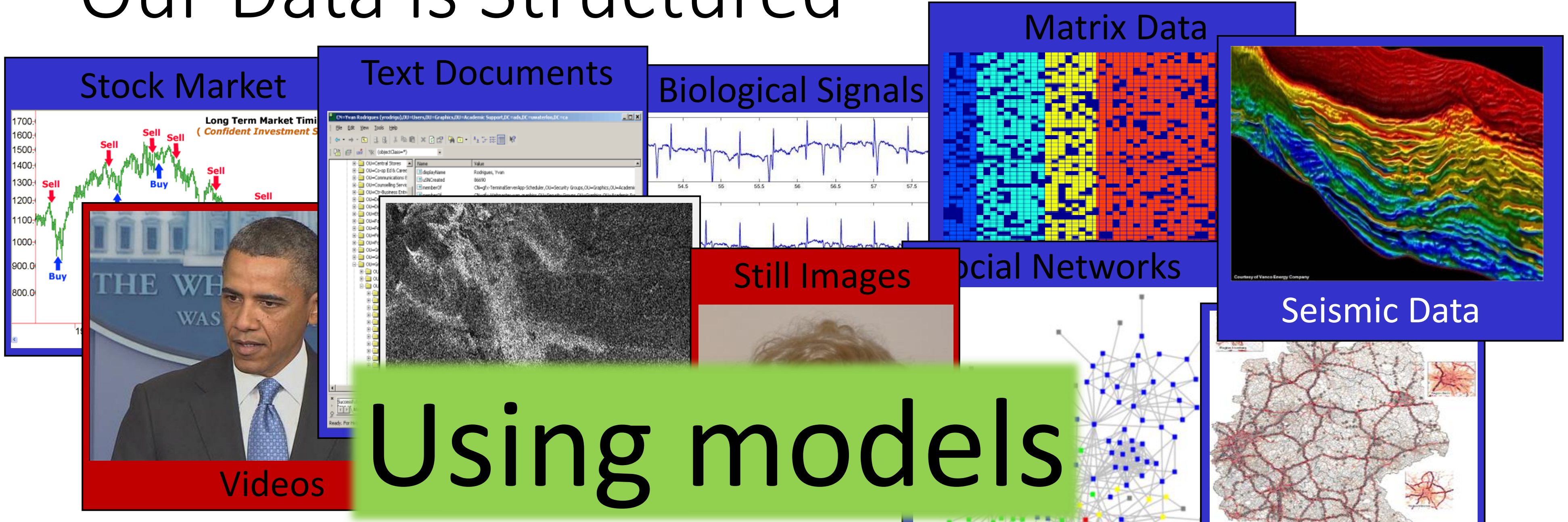
Topic: Sparse Representation and modeling

Instructor: Abdul Qayyum, PhD

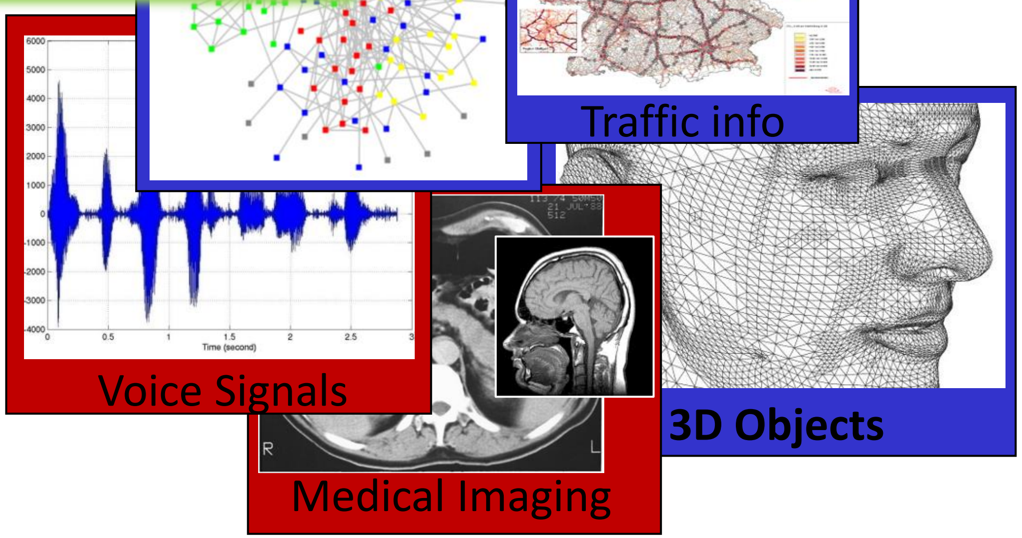
Class: MSCV

University of Burgundy, France

Our Data is Structured



- We are surrounded by various diverse sources of massive information
- Each of these sources have an internal structure, which can be exploited
- This structure, when identified, is the engine behind the ability to process data
- How to identify structure?



What is Sparseland?

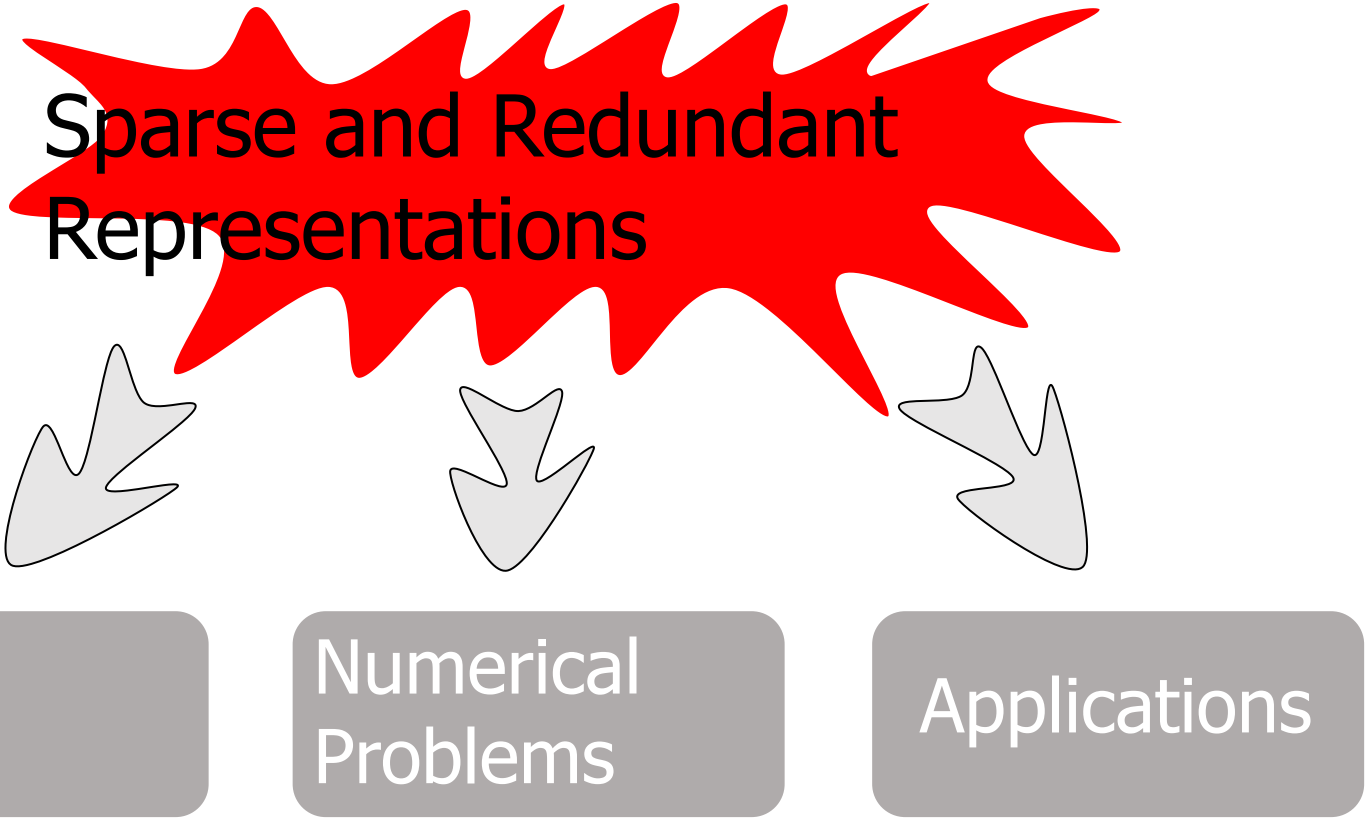
Data Models and Their Use

- Almost any task in data processing requires a model – true for denoising, deblurring, super-resolution, inpainting, compression, anomaly-detection, sampling, recognition, separation, and more
- Sparse and Redundant Representations offer a new and highly effective model – we call it

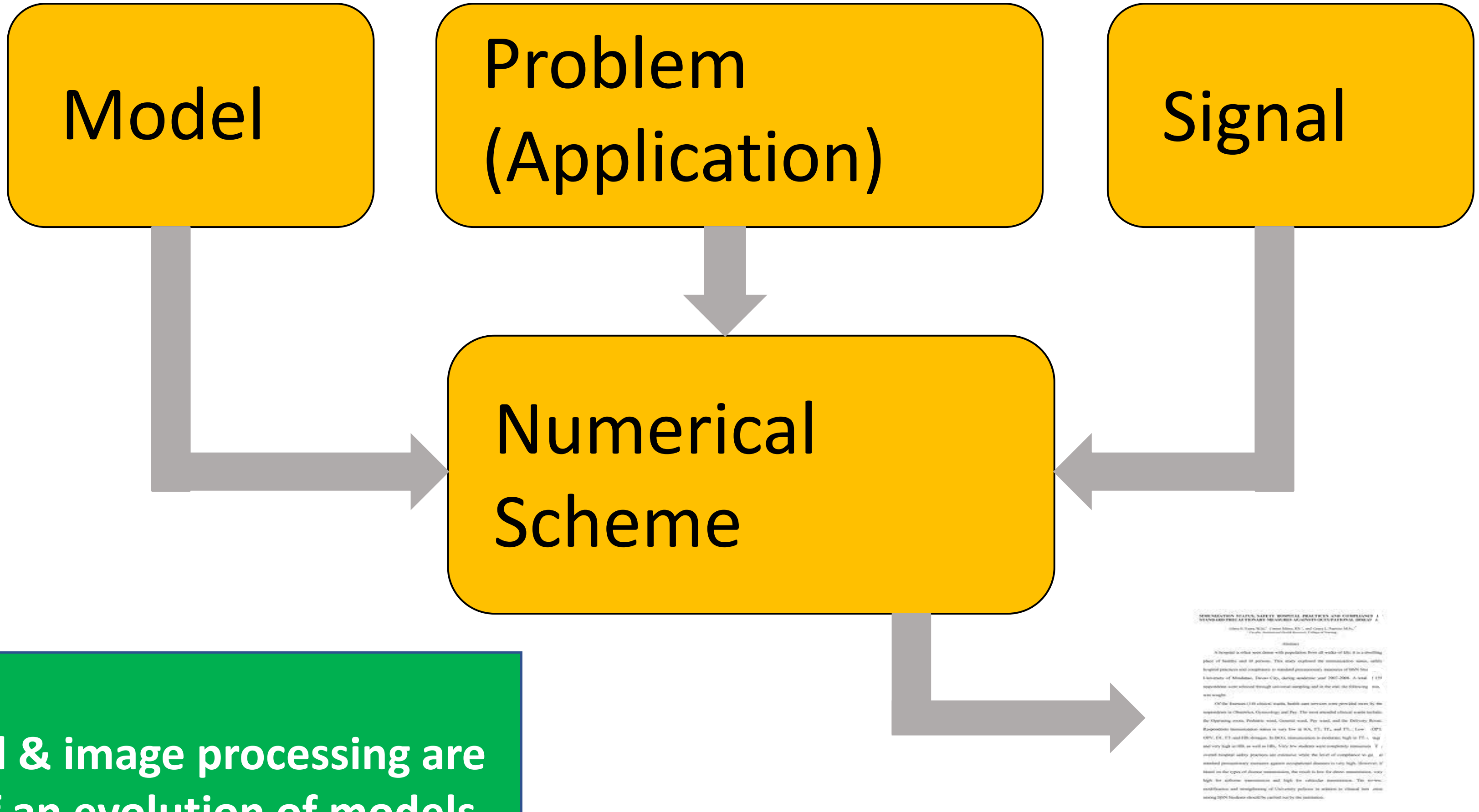
Sparseland

- We shall describe this and descendant versions of it that lead all the way to …

Overview


Sparse and Redundant Representations
Theory
Numerical Problems
Applications

Research in Signal/Image Processing



The fields of signal & image processing are essentially built of an evolution of models and ways to use them for various tasks

Sparse and Redundant Representation Modeling of Signals – Theory and Applications
By: Michael Elad

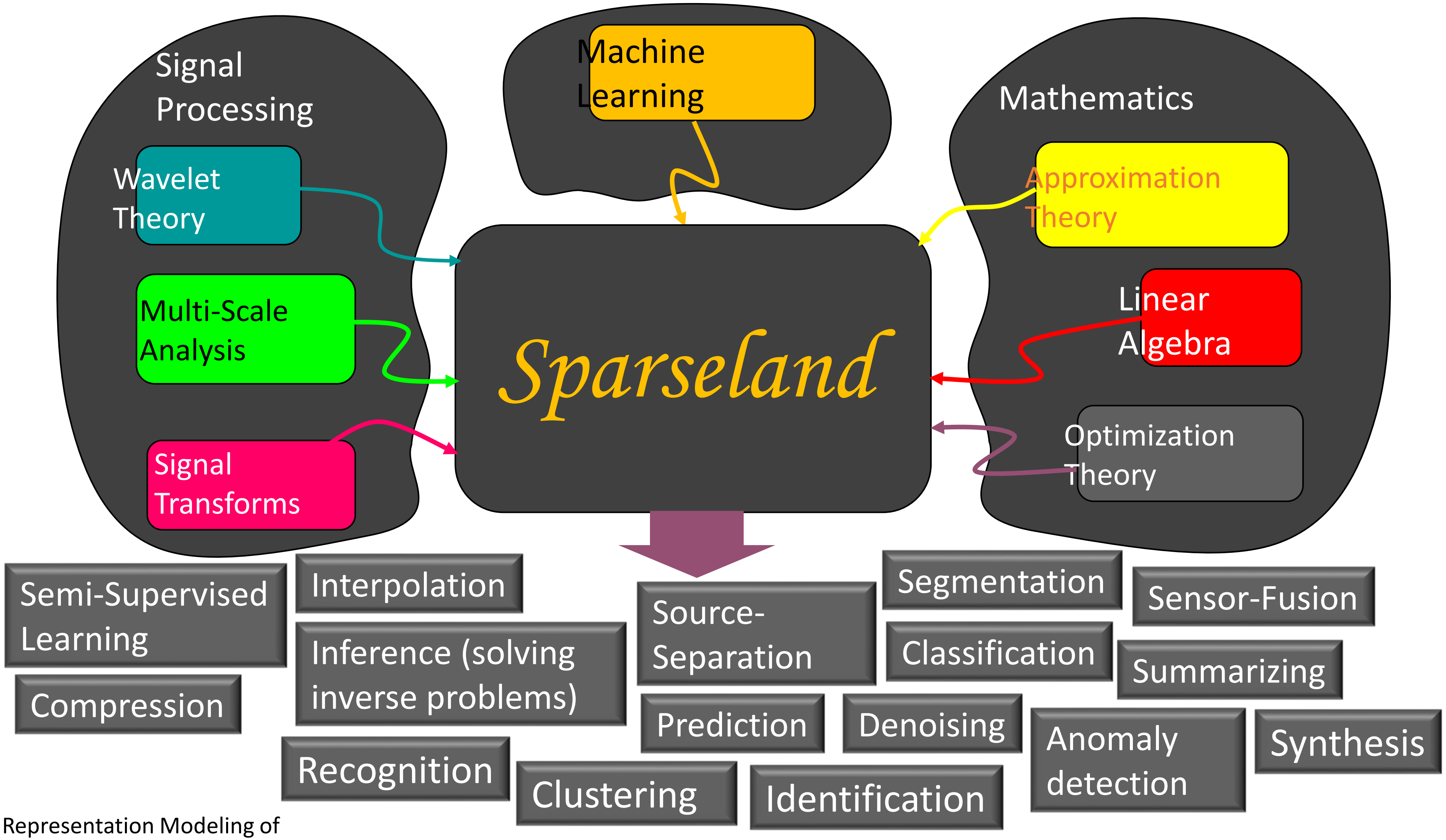
Sparse Representation and Modeling

Almost any task in data processing requires a model – true for denoising, deblurring, super-resolution, inpainting, compression, anomaly-detection, sampling, recognition, separation, and more

Sparse and Redundant Representations offer a new and highly effective model – we call it

Sparseland

New Emerging Model



Noise Removal?

Important: (i) Practical application; (ii) A convenient platform (being the simplest inverse problem) for testing basic ideas in image processing, and then generalizing to more complex problems.

Many Considered Directions: Partial differential equations, Statistical estimators, Adaptive filters, Inverse problems & regularization, Wavelets, Example-based techniques, Sparse representations, …

Denoising By Energy Minimization

Many of the proposed image denoising algorithms are related to the minimization of an energy function of the form

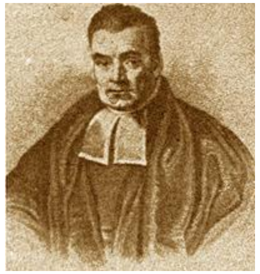
$$f(\underline{x}) = \frac{1}{2}\left\|\underline{x} - \underline{y}\right\|_2^2 + G(\underline{x})$$

Relation to measurements

Prior or regularization

$\underline{y}$: Given measurements

$\underline{x}$: Unknown to be recovered

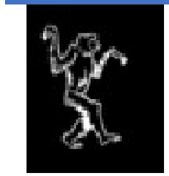


- This is in-fact a Bayesian point of view, adopting the Maximum-A-posteriori Probability (MAP) estimation.
- Clearly, the wisdom in such an approach is within the choice of the prior – **modeling the images** of interest.

Evolution of $G(\underline{x})$

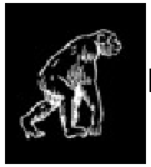
During the past several decades we have made all sort of guesses about the prior $G(\underline{x})$ for images:

$G(\underline{x}) = \lambda \|\underline{x}\|_2^2$



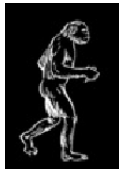
Energy

$G(\underline{x}) = \lambda \|\mathbf{L}\underline{x}\|_2^2$



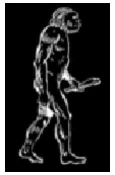
moothness

$G(\underline{x}) = \lambda \|\mathbf{L}\underline{x}\|_{\mathbf{W}}^2$



Adapt+ Smooth

$G(\underline{x}) = \lambda \rho\{\mathbf{L}\underline{x}\}$

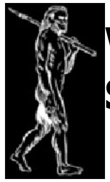


Robust Statistics

$G(\underline{x}) = \lambda \|\nabla \underline{x}\|_1$

Total-Variation

$G(\underline{x}) = \lambda \|\mathbf{W}\underline{x}\|_1$



Wavelet Sparsity

$G(\underline{x}) = \lambda \|\underline{\alpha}\|_0^0$

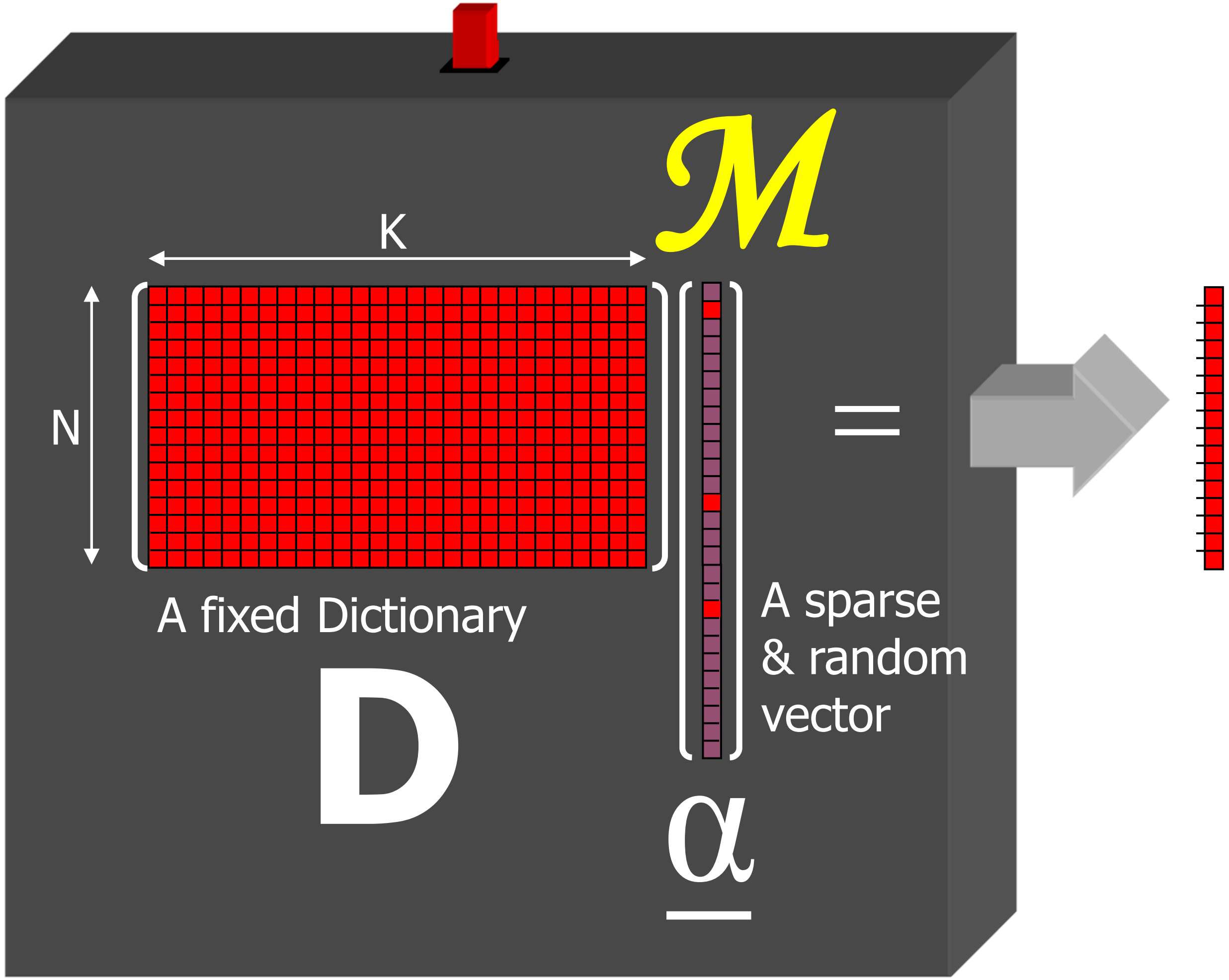
for $\underline{x} = \mathbf{D}\underline{\alpha}$

Sparse & Redundant

Hidden Markov Models,

- Compression algorithms as priors,
- Direct use of examples

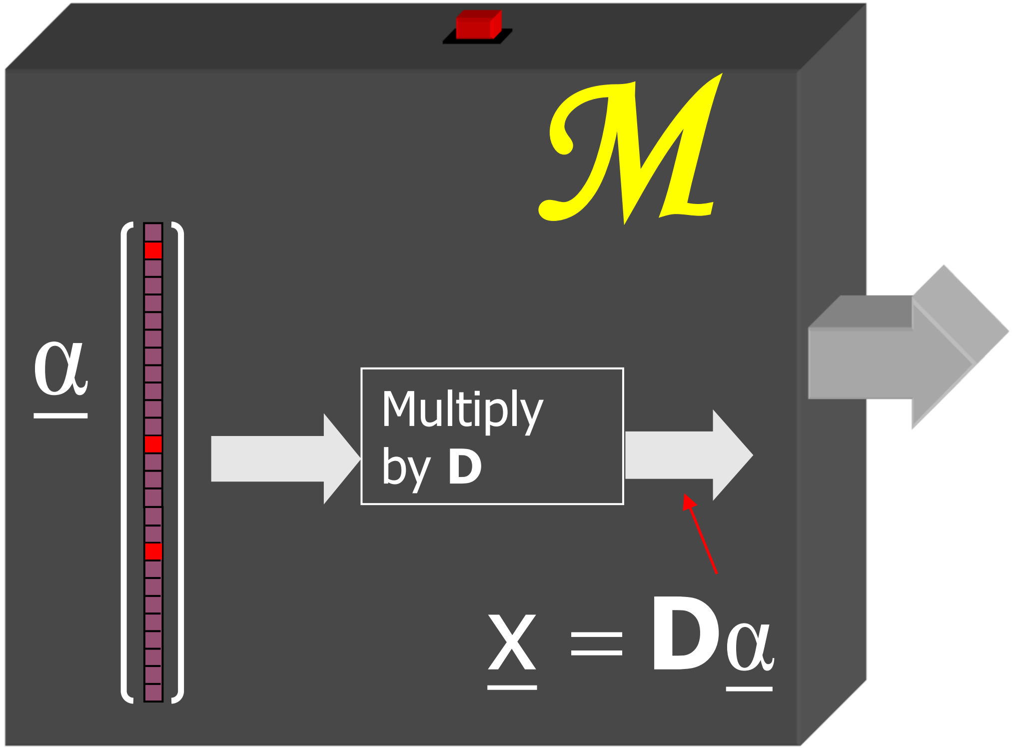
Sparse Modeling of Signals



- Every column in **D** (dictionary) is a prototype signal (atom).
- The vector $\underline{\alpha}$ is generated randomly with few (say L) non-zeros at random locations and with random values.

this model as *Sparseland*

Sparseland Signals are Special



Interesting Model:

Simple: Every generated signal is built as a linear combination of <u>**few**</u> atoms from our dictionary **D**

Rich: A general model: the obtained signals are a union of many low-dimensional Gaussians.

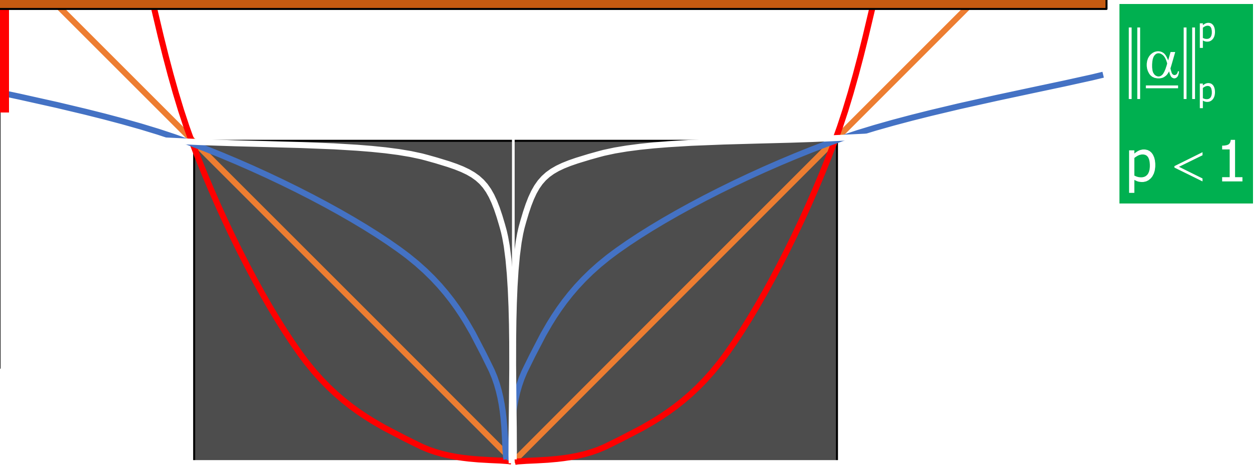
Familiar: We have been using this model in other context for a while now wavelet, JPEG, …).

Sparse & Redundant Rep. Modeling

Our signal model is thus: $\underline{x} = \mathbf{D}\underline{\alpha}$ where $\underline{\alpha}$ is sparse

As $p \to 0$ w
count
non-zeros in the
vector

$\Rightarrow \|\underline{\alpha}\|_0^0$



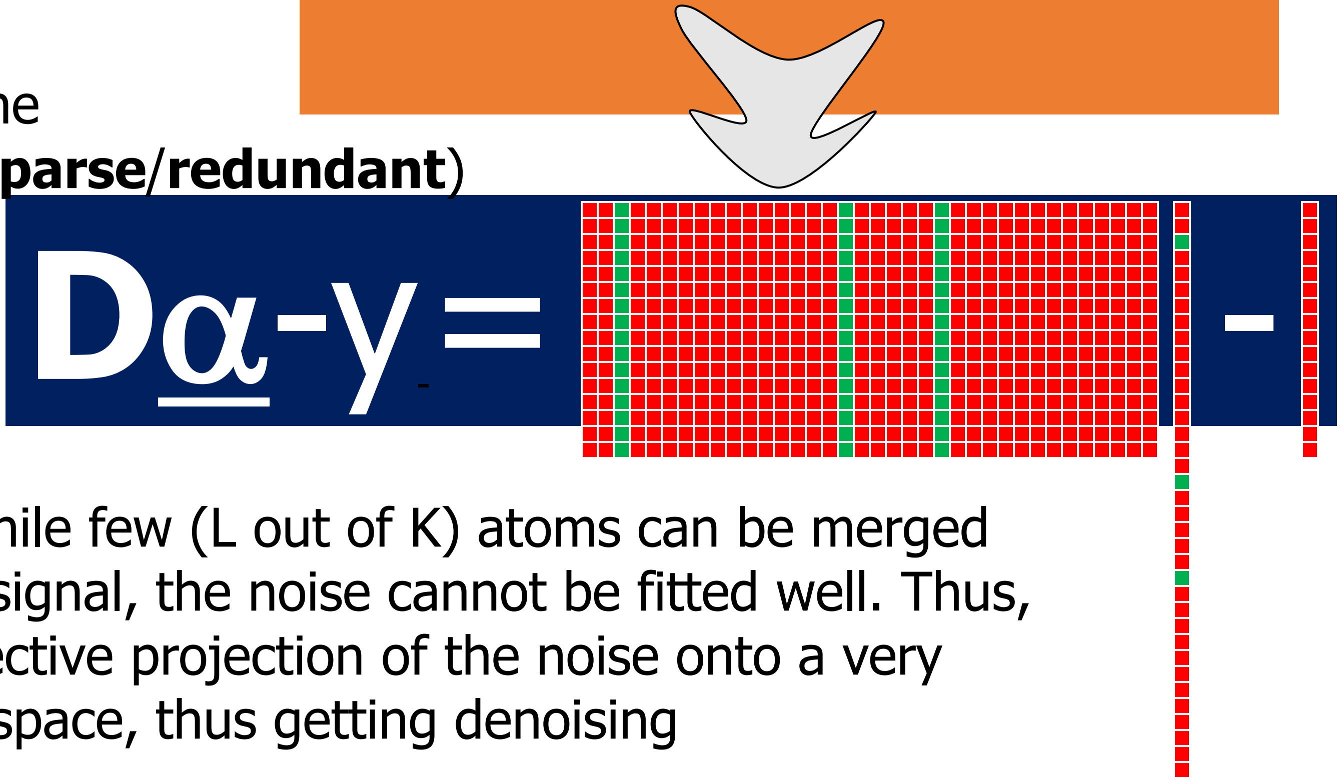
$\underline{x} = \mathbf{D}\underline{\alpha}$ where $\|\underline{\alpha}\|_0^0 \leq L$

Our MAP Energy Function

counting the number of non-zeros in $\underline{\alpha}$.

$$\frac{1}{2}\left\|\underline{x}-\underline{y}\right\|_2^2$$

- The vector $\underline{\alpha}$ is the representation (**sparse/redundant**) of the desired signal x.



- The core idea: while few (L out of K) atoms can be merged to form the true signal, the noise cannot be fitted well. Thus, we obtain an effective projection of the noise onto a very low-dimensional space, thus getting denoising

Wait! There are Some Issues

Numerical Problems: How should we solve or approximate the solution of the problem

$$\min_{\underline{\alpha}} \left\| \mathbf{D}\underline{\alpha} - \underline{y} \right\|_2^2 \text{ s.t. } \left\| \underline{\alpha} \right\|_0^0 \leq L \quad \text{or} \quad \min_{\underline{\alpha}} \left\| \underline{\alpha} \right\|_0^0 \text{ s.t. } \left\| \mathbf{D}\underline{\alpha} - \underline{y} \right\|_2^2 \leq \varepsilon^2$$

$$\text{or} \quad \min_{\underline{\alpha}} \lambda \left\| \underline{\alpha} \right\|_0^0 + \left\| \mathbf{D}\underline{\alpha} - \underline{y} \right\|_2^2 \quad ?$$

Theoretical Problems: Is there a unique sparse representation? If we are to approximate the solution somehow, how close will we get?

Practical Problems: What dictionary **D** should we use, such that all this leads to effective denoising? Will all this work in applications?

Summarize So Far …

Image denoising (and many other problems in image processing) requires a model for the desired image

What do we do?

We proposed a model for signals/images based on sparse and redundant representations

Great! No?

There are some issues:

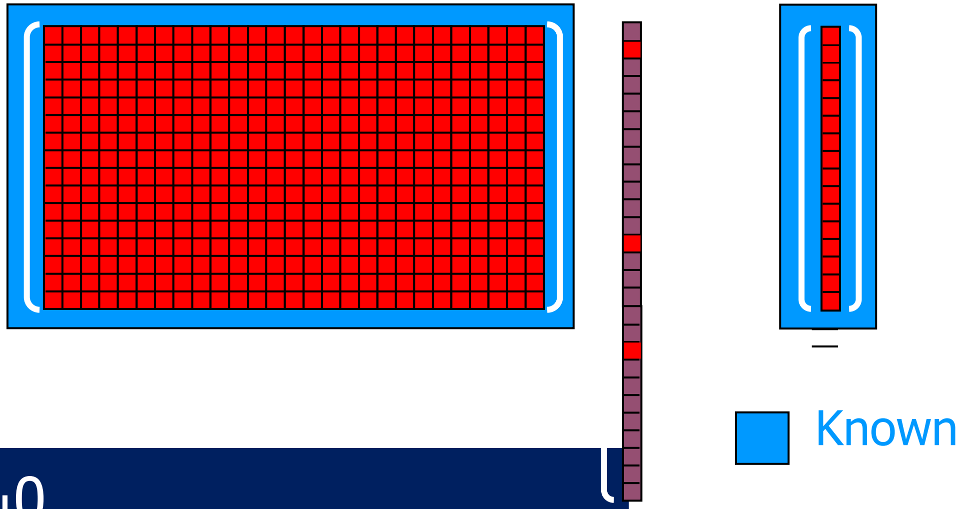
1. Theoretical
2. How to approximate?
3. What about **D**?

Theoretical & Numerical Foundations

Lets Start with the Noiseless Problem

Suppose we build a signal by the relation

$$\mathbf{D}\underline{\alpha} = \underline{x}$$



We aim to find the signal's representation:

$$\hat{\underline{\alpha}} = \underset{\underline{\alpha}}{\text{ArgMin}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \underline{x} = \mathbf{D}\underline{\alpha}$$

Uniqueness

Why should we necessarily get $\hat{\underline{\alpha}} = \underline{\alpha}$?

It might happen that eventually $\|\hat{\underline{\alpha}}\|_0^0 < \|\underline{\alpha}\|_0^0$

Our Goal

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \leq \varepsilon^2$$

This is a combinatorial problem, proven to be NP-Hard!

Here is a recipe for solving this problem:

Set L=1 → Gather all the supports $\{S_i\}_i$ of cardinality L → Solve the LS problem $\min_{\underline{\alpha}} \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \text{ s.t. } \operatorname{supp}(\underline{\alpha}) = S_i$ for each support → LS error $\leq \varepsilon^2$?

No → Set L=L+1 → Gather all the supports

Yes → Done

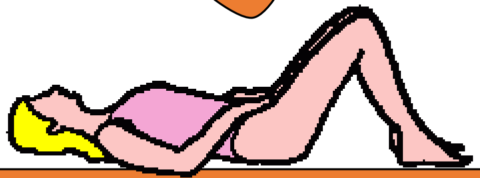
There are $\binom{K}{L}$ such supports

Assume: K=1000, L=10 (known!), 1 nano-sec per each LS

We shall need ~8e+6 years to solve this problem !!!!!

Lets Approximate

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \ \text{ s.t. } \ \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$



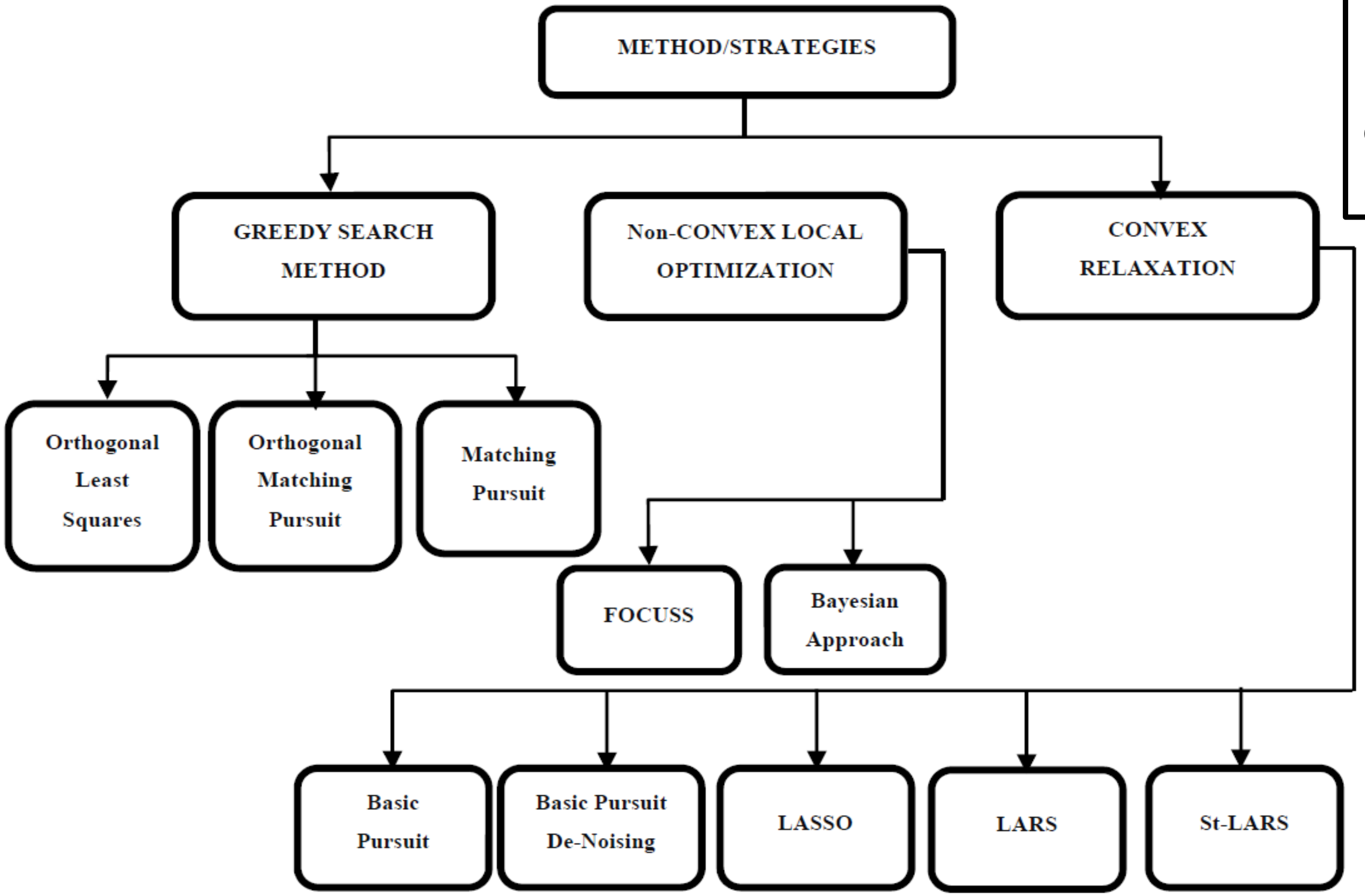
Relaxation methods

Smooth the L_0 and use continuous optimization techniques

Greedy methods

Build the solution one non-zero element at a time

Methods or techniques for sparse exact solution



LARS (least angle regression)
FOCUSS(Focal underdetermine solver system)
LASSO(least absolute shrinkage and selection operator)

Relaxation – The Basis Pursuit (BP)

Instead of solving

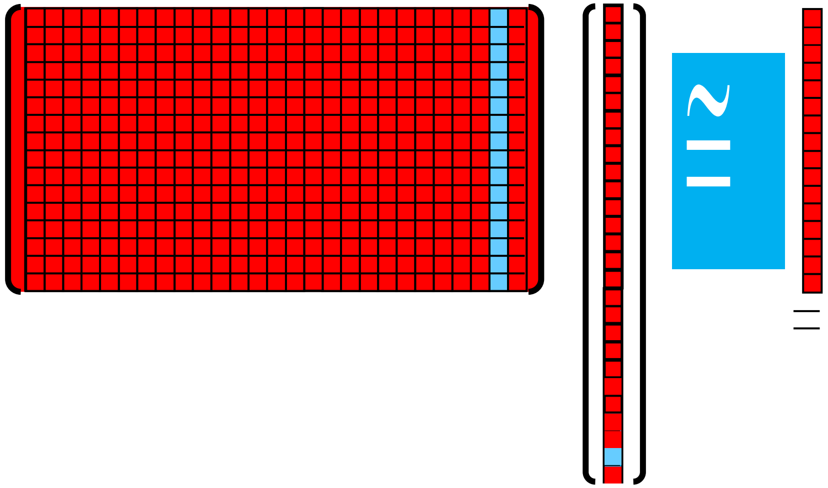
$$\underset{\underline{\alpha}}{\text{Min}} \; \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2 \le \varepsilon$$

Solve Instead

$$\underset{\underline{\alpha}}{\text{Min}} \; \|\underline{\alpha}\|_1 \quad \text{s.t.} \quad \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2 \le \varepsilon$$

- This is known as the Basis-Pursuit (BP) [Chen, Donoho & Saunders ('95)].
- The newly defined problem is convex (quad. programming).
- Very efficient solvers can be deployed:
  - **Interior point methods** [Chen, Donoho, & Saunders ('95)] [Kim, Koh, Lustig, Boyd, & D. Gorinevsky (`07)].
  - **Sequential shrinkage** for union of ortho-bases [Bruce et.al. ('98)].
  - **Iterative shrinkage** [Figuerido & Nowak ('03)] [Daubechies, Defrise, & De-Mole ('04)] [E. ('05)] [E., Matalon, & Zibulevsky ('06)] [Beck & Teboulle (`09)] …

Greedy: Matching Pursuit (MP)



- The MP is one of the greedy algorithms that finds one atom at a time [Mallat & Zhang ('93)].
- Step 1: find the one atom that best matches the signal.
- Next steps: given the previously found atoms, find the next **one** to best fit the rsidual.
- The algorithm stops when the error $\left\| \mathbf{D}\underline{\alpha} - \underline{y} \right\|_2$ is below the destination threshold.
- The Orthogonal MP (OMP) is an improved version that re-evaluates the coefficients by Least-Squares after each round.

Greedy: Orthogonal Matching Pursuit (OMP)

Algorithm. Orthogonal matching pursuit algorithm.

Task: Approximate the constraint problem:

$\alpha_1 = arg\ min_{\alpha_1} ||\alpha_1||_0 \quad \text{s.t} \ B_1 = D\alpha_1$

Input: Input sample B_1 , Dictionary matrix D , sparse coefficients vector α_1.

Initialization: $t = 1, r_0 = B_1, \alpha_1 = 0, D_0 = D$, index set $\Lambda_0 = \emptyset$. Where $\emptyset$ denotes empty an set, τ is small constant. d_j is the dictionary elements stack as column vector.

While $||r_t|| > \tau$ **do**

Step 1: Find the best matching sample, i.e. the biggest inner product between r_{t-1} and $d_j (j \notin \Lambda_{t-1})$ by exploiting

$$\lambda_t \doteq arg\ max_{j \notin \Lambda_{t-1}} |\langle r_{t-1}, d_j \rangle|.$$

Step 2:Update the index set $\Lambda_t = \Lambda_{t-1} \cup \lambda_t$ and reconstruct data set

$$\dot{D_t} = |D_{t-1}, d_{\lambda_t}|.$$

Step 3: Compute the sparse coefficients by using the least square algorithm

$$\breve{\alpha}_1 = \dot{a}rg\ min ||B_1 - D_t \breve{\alpha}_1||_2^2$$

Step 4: Update the representation residual using $r_t = B_1 - D_t \breve{\alpha}_1$

Step 5: $t = t + 1$

End

OMP(orthogonal Matching Pursuit)

```
function [sols, iters, activationHist] = SolveOMP(A, y, N, maxIters,
lambdaStop, solFreq, verbose, OptTol)
% SolveOMP: Orthogonal Matching Pursuit
% Usage
%   [sols, iters, activationHist] = SolveOMP(A, y, N, maxIters,
lambdaStop, solFreq, verbose, OptTol)
% Input
%   A           Either an explicit nxN matrix, with rank(A) = min(N,n)
%               by assumption, or a string containing the name of a
%               function implementing an implicit matrix (see below for
%               details on the format of the function).
%   y           vector of length n.
%   N           length of solution vector.
%   maxIters    maximum number of iterations to perform. If not
%               specified, runs to stopping condition (default)
%   lambdaStop  If specified, the algorithm stops when the last
coefficient
%               entered has residual correlation <= lambdaStop.
%   solFreq     if =0 returns only the final solution, if >0, returns an
%               array of solutions, one every solFreq iterations
(default 0).
%   verbose     1 to print out detailed progress at each iteration, 0
for
%               no output (default)
%   OptTol      Error tolerance, default 1e-5
% Outputs
%    sols            solution(s) of OMP
%    iters           number of iterations performed
%    activationHist  Array of indices showing elements entering
%                    the solution set
%
```



```
Description
%   SolveOMP is a greedy algorithm to estimate
the solution
%   of the sparse approximation problem
%      min ||x||_0 s.t. A*x = b
%   The implementation implicitly factors the
active set matrix A(:,I)
%   using Cholesky updates.
%   The matrix A can be either an explicit
matrix, or an implicit operator
%   implemented as an m-file. If using the
implicit form, the user should
%   provide the name of a function of the
following format:
%     y = OperatorName(mode, m, n, x, I, dim)
%   This function gets as input a vector x and an
index set I, and returns
%   y = A(:,I)*x if mode = 1, or y = A(:,I)'*x if
mode = 2.
%   A is the m by dim implicit matrix implemented
by the function. I is a
%   subset of the columns of A, i.e. a subset of
1:dim of length n. x is a
%   vector of length n is mode = 1, or a vector
of length m is mode = 2.
```

Pursuit Algorithms

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$

There are various algorithms designed for approximating the solution of

Why should they work ?

- Greedy A ... uit (OMP), L ... ing Pursuit [1
- Relaxatio ... ector & numeri
- Hybrid Al ... Hard-Thresholding [2007-today].

To Summarize So

Image denoising (and many other problems in image processing) requires a model for the desired image

What do we do?

We proposed a model for signals/images based on sparse and redundant representations

Problems?

We have seen that there are approximation methods to find the sparsest solution, and there are theoretical results that guarantee their success.

What next?

The Dictionary **D** should be found somehow !!!

Dictionary Construction

What Should **D** Be?

$$\underline{\hat{\alpha}} = \underset{\underline{\alpha}}{\operatorname{argmin}} \|\underline{\alpha}\|_0^0 \ \text{ s.t. } \frac{1}{2}\|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2 \quad \Rightarrow \quad \underline{\hat{x}} = \mathbf{D}\underline{\hat{\alpha}}$$

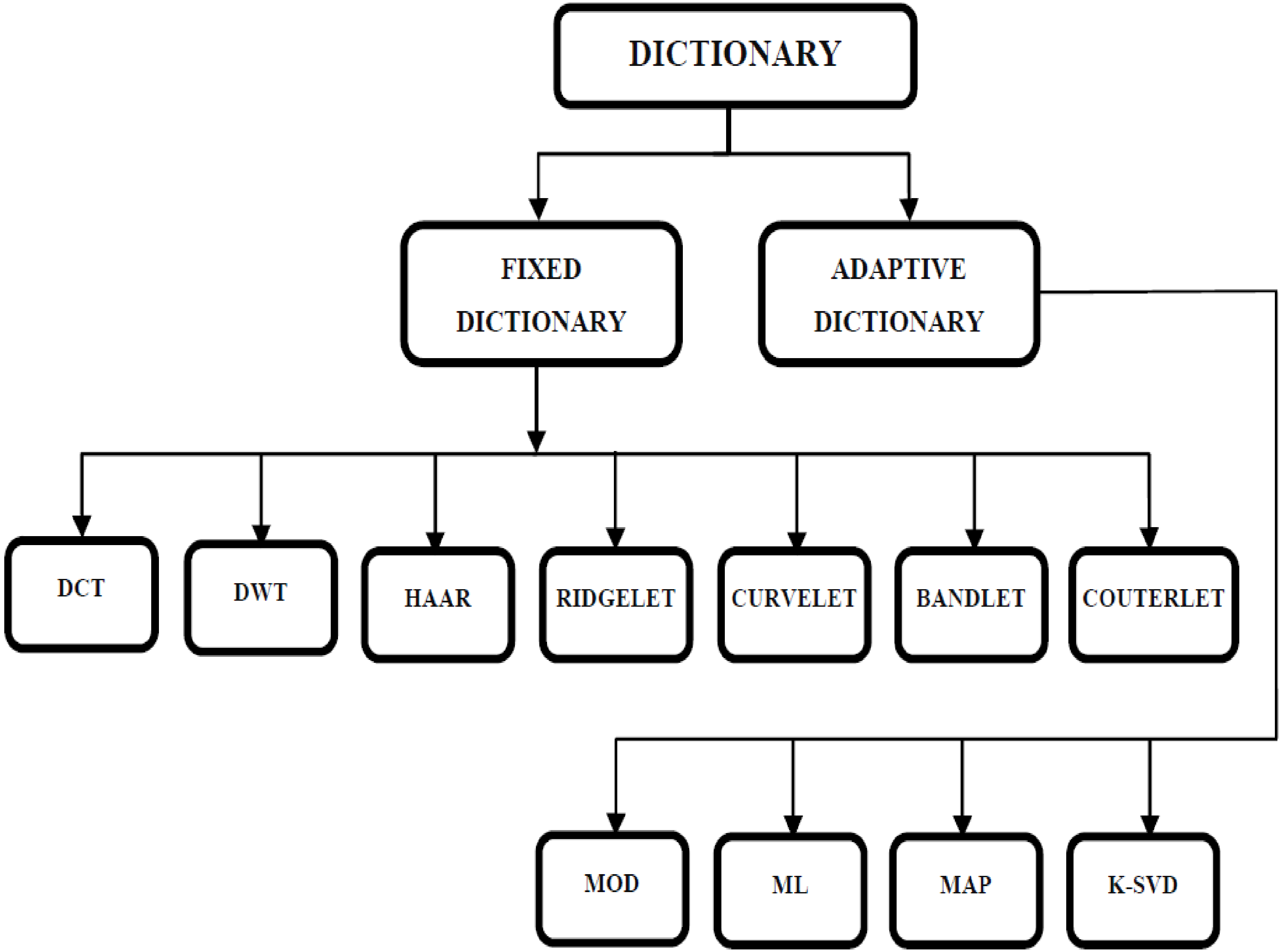
Our Assumption: Good-behaved Images have a sparse representation

D should be chosen such that it sparsifies the representations

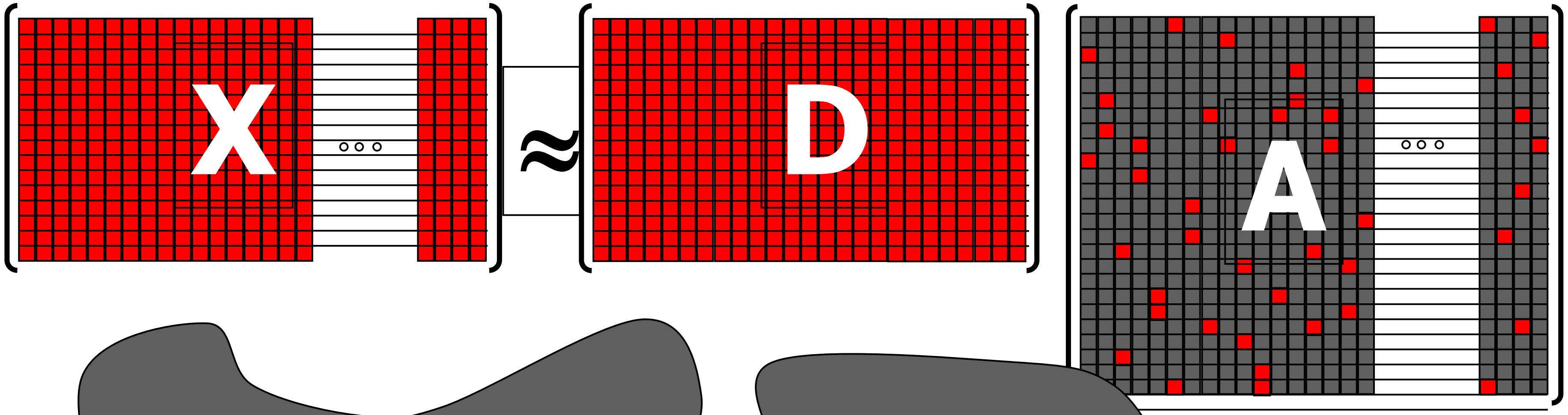
One approach to choose **D** is from a known set of transforms (Steerable wavelet, Curvelet, Contourlets, Bandlets, Shearlets …)

The approach we will take for building **D** is training it, based on **Learning** from **Image Examples**

Types of Dictionaries



Measure of Quality for



$$\underset{\mathbf{D},\mathbf{A}}{\mathrm{Min}} \sum_{j=1}^{P} \left\| \mathbf{D}\underline{\alpha}_j - \underline{x}_j \right\|_2^2 \quad \text{s.t. } \forall j, \left\| \underline{\alpha}_j \right\|_0^0 \le L$$

Each example is combination **D**

Each example has a sparse representation wit no more than L atoms

[Field & Olshausen ('96)]
[Engan et. al. ('99)]
[Lewicki & Sejnowski ('00)]
[Cotter et. al. ('03)]
[Gribonval et. al. ('04)]
[Aharon, E. & Bruckstein ('04)]
[Aharon, E. & Bruckstein ('05)]

K–Means For Clustering

Clustering: An extreme sparse representation

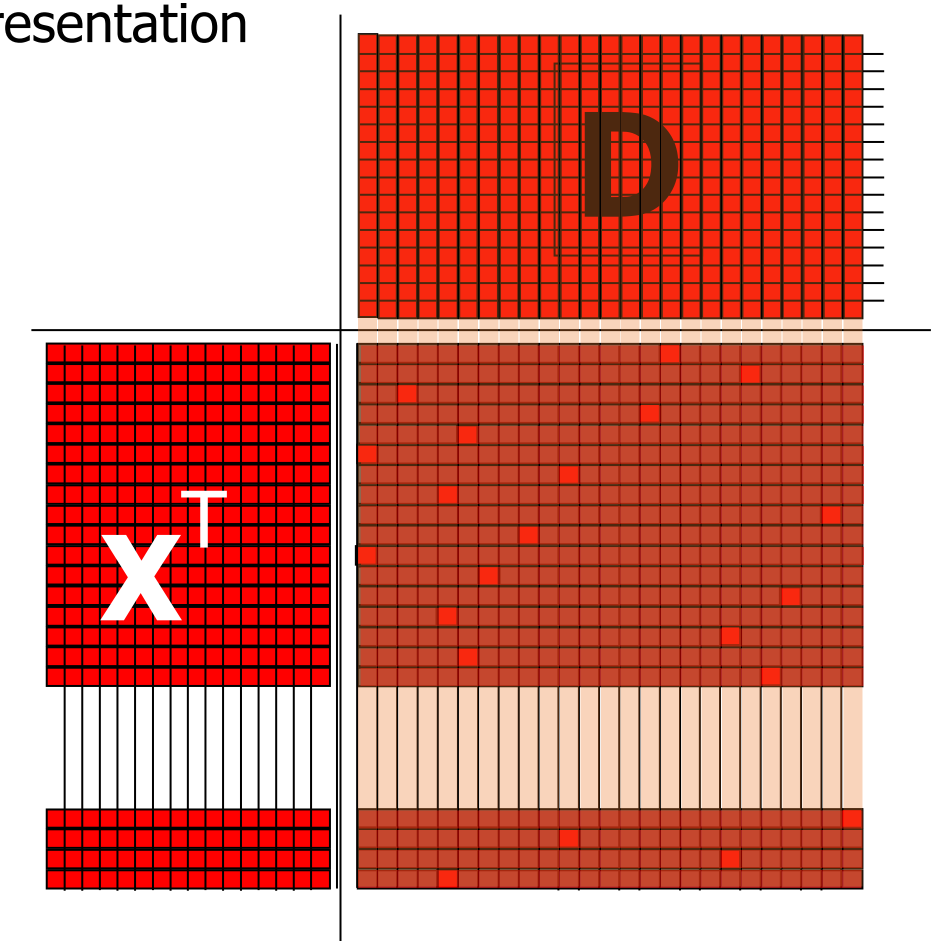
Initialize **D**

Sparse Coding

Nearest Neighbor

Dictionary Update

Column-by-Column by
Mean computation over the relevant examples



The K–SVD Algorithm – General

[Aharon, E. & Bruckstein ('04,'05)]

Initialize

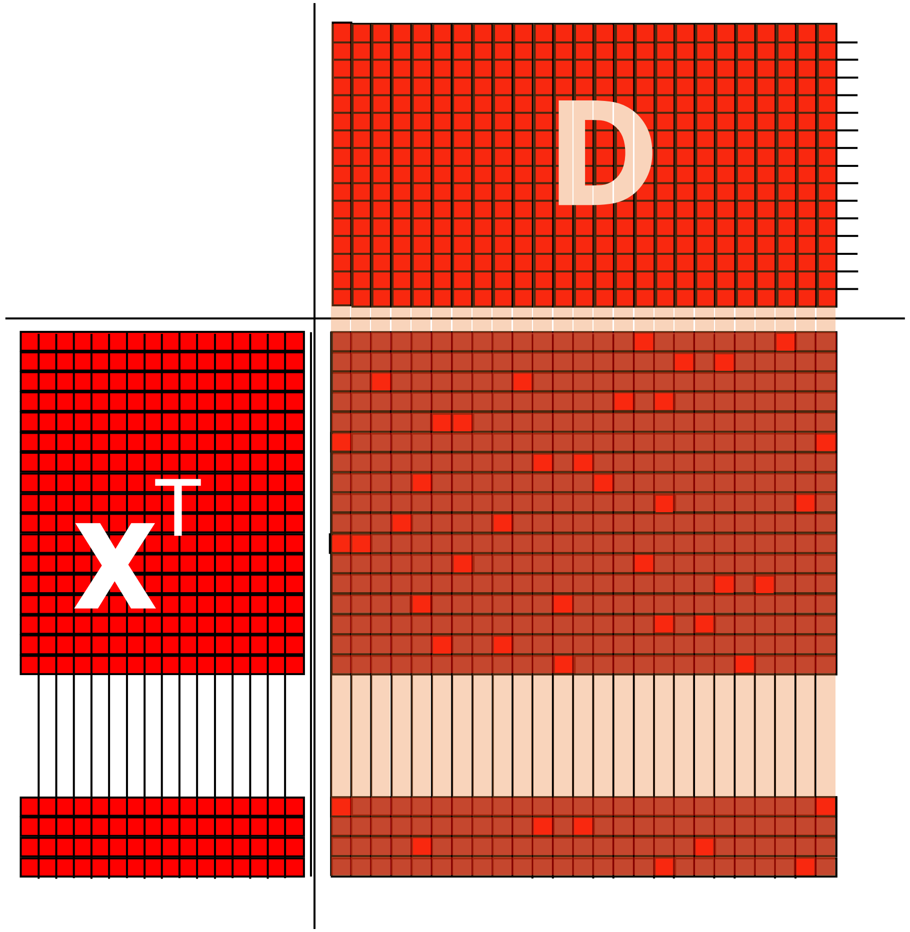
Sparse Coding

Use Matching Pursuit

Dictionary Update

Column-by-Column

SVD computation over the relevant examples



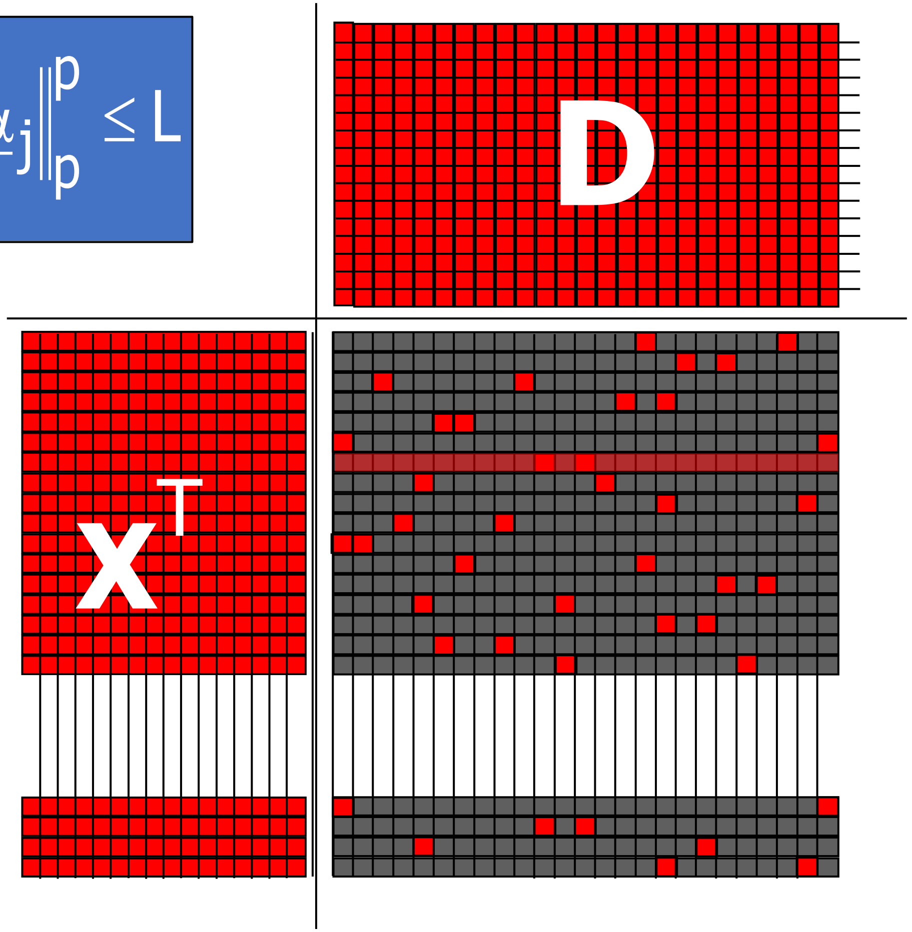
K–SVD: Sparse Coding Stage

$$\underset{\mathbf{A}}{\text{Min}} \sum_{j=1}^{P} \left\| \mathbf{D}\underline{\alpha}_j - \underline{x}_j \right\|_2^2 \quad \text{s.t.} \quad \forall j, \ \left\| \underline{\alpha}_j \right\|_p^p \le L$$

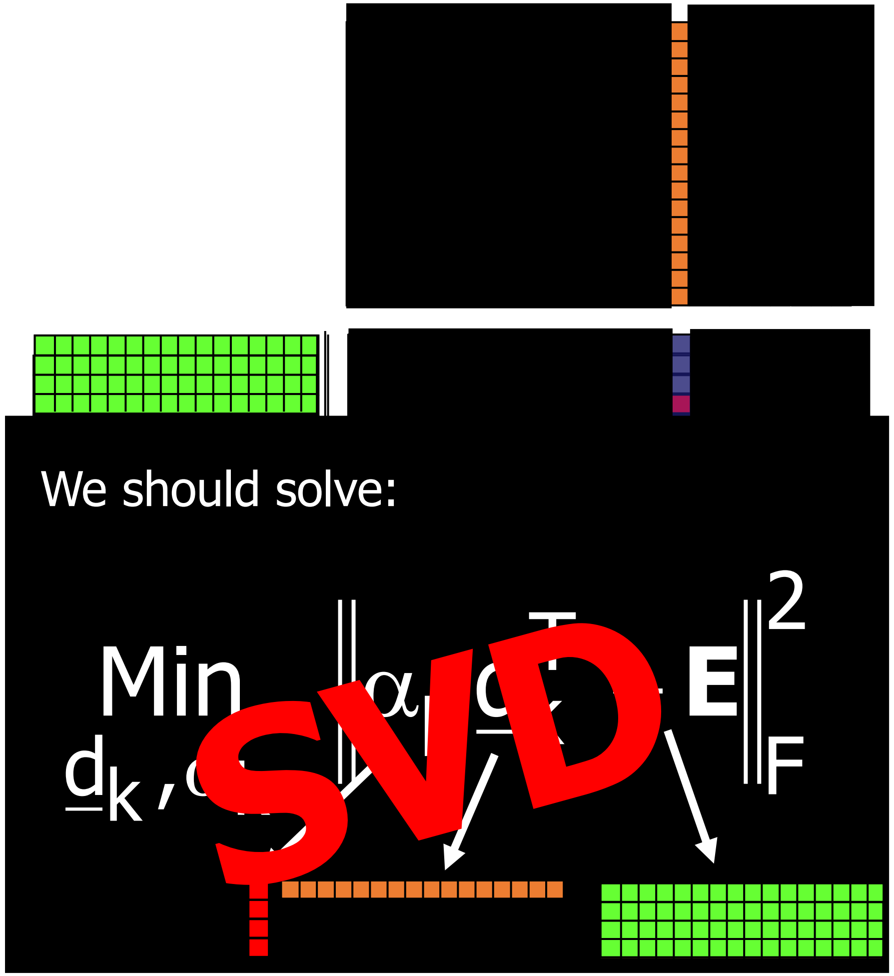
D is known! For the j^{th} item we solve

$$\underset{\underline{\alpha}}{\text{Min}} \left\| \mathbf{D}\underline{\alpha} - \underline{x}_j \right\|_2^2 \quad \text{s.t.} \quad \left\| \underline{\alpha} \right\|_p^p \le L$$

Solved by
A Pursuit Algorithm



–SVD: Dictionary Update

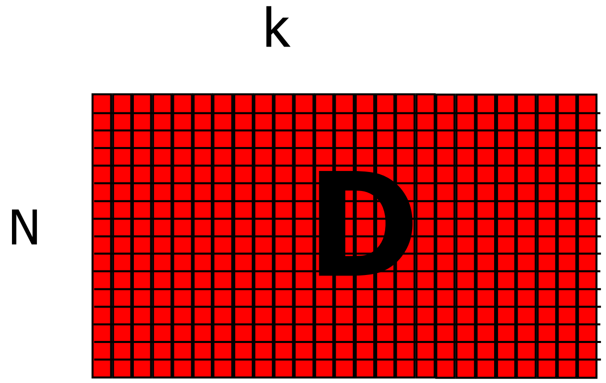


We refer only to the examples that use the column $\underline{d}_k$

Fixing all **A** and **D** apart from the k^{th} column, and seek both $\underline{d}_k$ and the k^{th} column in **A** to better fit the **residual**!

From Local to Global Treatment

- The K-SVD algorithm is reasonable for low-dimension signals (N in the range 10-400). As N grows, the complexity and the memory requirements of the K-SVD become prohibitive.



- So, how should large images be handled?

The solution: Force shift-invariant sparsity - on each patch of size N-by-N (N=8) in the image, including overlaps.

$$\hat{\underline{x}} = \underset{\underline{x},\{\underline{\alpha}_{ij}\}_{ij}}{\text{ArgMin}} \; \frac{1}{2}\left\|\underline{x}-\underline{y}\right\|_2^2 + \mu\sum_{ij}\left\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}_{ij}\right\|_2^2$$

$$\text{s.t.} \quad \left\|\underline{\alpha}_{ij}\right\|_0^0 \le L$$

Extracts a patch in the ij location

Our prior

K-SVD Image Denoising

$$\hat{\underline{x}} = \underset{\underline{x},\{\underline{\alpha}_{ij}\}_{ij},\mathbf{D}?}{\text{ArgMin}} \frac{1}{2}\|\underline{x}-\underline{y}\|_2^2 + \mu\sum_{ij}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}_{ij}\|_2^2 \quad \text{s.t.} \quad \|\underline{\alpha}_{ij}\|_0^0 \le L$$

$\underline{x}=\underline{y}$ and **D** known

$\underline{x}$ and α_{ij} known

D and α_{ij} known

Compute α_{ij} per patch

$$\underline{\alpha}_{ij} = \underset{\underline{\alpha}}{\text{Min}}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}\|_2^2$$

$$\text{s.t.} \ \|\underline{\alpha}\|_0^0 \le L$$

using the matching pursuit

Compute **D** to minimize

$$\underset{\underline{\alpha}}{\text{Min}}\sum_{ij}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}\|_2^2$$

using SVD, updating one column at a time

Compute $\underline{x}$ by

$$\underline{x} = \left[\mathrm{I} + \mu\sum_{ij}\mathbf{R}_{ij}^T\mathbf{R}_{ij}\right]^{-1}\left[\underline{y} + \mu\sum_{ij}\mathbf{R}_{ij}^T\mathbf{D}\underline{\alpha}_{ij}\right]$$

which is a simple averaging of shifted patches

K-SVD

Complexity of this algorithm: O(N^2×K×L×Iterations) per pixel. For N=8, L=1, K=256, and 10 iterations, we need 160,000 (!!) operations per pixel.

Proposed Dictionaries

Pre-determined Dictionaries

Proposed Dictionaries (DRT and DTT)

- **DTT** based on orthogonal transform basis
- High energy compaction, perfect image reconstruction and decorrelation properties.
- It captured well-defined structural atoms in regular forms and also stored well-structured image features.
- It stores prominent features of stereo images using similar coefficients vectors.

$$f_{n_1}(v) = (B_1 v + B_2) f_{n_1-1}(v) + B_3 f_{n_1-2}(v)$$

$$B_1 = \frac{2}{n_1}\sqrt{\frac{4n_1^2 - 1}{M^2 - n_1^2}}$$

$$B_2 = \frac{1-M}{n_1}\sqrt{\frac{4n_1^2 - 1}{M^2 - n_1^2}}$$

$$B_3 = \frac{n_1 - 1}{n_1}\sqrt{\frac{2n_1 + 1}{2n_1 - 3}}\sqrt{\frac{M^2 - (n_1 - 1)^2}{M^2 - n_1^2}}$$

$$F_{n_1 n_2} = \sum_{v=0}^{M-1} f_{n_1}(v) \sum_{u=0}^{M-1} f_{n_2}(u) f(v,u)$$

- **DRT** constructed by using the ricker wavelet basis function to generate finite ridgelet transform as bases elements for a hybrid dictionary.
- The hybrid ridgelet dictionary will preserve the fine details of edges, contours and texture in images.
- It used for approximation of non-linear signals and better deals with line discontinuities.
- It search the similar coefficients based on prominent features b\w images

$$\vartheta_{a,b,\theta}(x) = a^{-1/2}\,\vartheta\left(\frac{(x_1 cos\theta + x_2 sin\theta - b)}{a}\right)$$

$$\vartheta_{a_1,a_2,b_1,b_2}(x) = \vartheta_{a_1,a_2}(x_1)\vartheta_{b_1,b_2}(x_2)$$

$$\vartheta_{a,b}(t) = a^{-1/2}\vartheta_{a,b}(t - b/a)$$

$$t = x_1 cos\theta + x_2 sin\theta$$

DRT dictionary basis

	Algorithm. Dictionary based on hybrid Ricker Wavelet Basis Function
1	D is dictioanry size, S scale factor, T translation parameter , θ is the rotation parameter, M_1 number of dictionary atoms, N_1 size of dictionary
2	**FOR** $each\ Scale\ S = 1{:}\ M_1$ **do**
4	**FOR** $each\ translation\ T = 1{:}\ M$ **do**
5	**FOR** $each\ rotation\ \theta = -\pi{:}\ \pi$ **do**
6	**FOR** $each\ n_1 = 1{:}\ M_1$ **do**
7	**FOR** $each\ n_2 = 1{:}\ N_1$ **do**
8	$Temp(n_1, n_2) = \sqrt{S} \times \sin[n_1 \times cos\theta + n_2 \times sin\theta - T]e^{-t/2}$
9	$D = temp(:)$;
10	**ENDFOR END FOR**
11	$storeD(:,count) = temp(:)$

12	**END FOR**
	END FOR
	ENDFOR

13	Select bases have greater variance than a certain threshold $T_1 = M^{2\times 0.05}$ Select size of dictionary $D_1\ (:, count) = D_1\ (:, size\ of\ dictioanry)$ $D = D_1$ **END FOR**

$$\vartheta_{a,b,\theta}(x) = a^{-1/2}\,\vartheta\left(\frac{x_1 cos\theta + x_2 sin\theta - b}{a}\right)$$

$$\vartheta_{a_1,a_2,b_1,b_2}(x) = \vartheta_{a_1,a_2}(x_1)\vartheta_{b_1,b_2}(x_2)$$

$$\vartheta_{a,b}(t) = a^{-1/2}\vartheta_{a,b}(t - b/a)$$

$$t = x_1 cos\theta + x_2 sin\theta$$

DRT dictionary basis

```
basis=[];
M=8; N=M;  % size of path (M by N)
%% 2-D Ridgelet Basis
s={}; temp=zeros(M,N);
Dl=zeros(M^2,10);
count1=0;  count=1;
for s1=0.2:0.4:M-1
    s1
        count1=count1+1;
        count2=0;
        for tau1=0.1:0.2:M-1
                for theta=-pi:0.05:pi
                    count2=count2+1;
                    for n1=1:M
                        for n2=1:N
                            t=(n1*cos(theta)+n2*sin(theta)-tau1)/s1;
                            %t2=(n2-tau2)/s2;
%                             temp(n1,n2)= 2^(s1/2)*
%                             db1_wavelet(t,tau1,s1); % db1  % (2^s-tau);%(x-tau)/(2^s); %(2^j)*x-i;
%                              temp(n1,n2)= s1^(1/2)*Meyer_wavelet(t);
                            temp(n1,n2)= s1^(1/2)* sin(5*t)*exp(-t^2/2); %2^(s1/2) % Morelet wavelet
%                              temp(n1,n2) = (2/sqrt(3)*pi^(-0.25))*(1-t^2)*exp(-t^2/2); % Maxican Hat
%                             temp(n1,n2) =2^(s1/2)*DoG_wavelet(n1,s1,t); %*2^(s2/2)*DoG_wavelet(n2,s2,tau2);  %
DoG
%                             basis{count1, count2}=temp;
                        end
                    end

                    if sum(abs(temp(:)))~=0
                        Dl(:,count)=temp(:);
                        count=count+1;
                    end
                end
        end
end
```

```
xp=randperm(size(Dl,2));
Dl=Dl(:,xp);
%% Select bases having
variance greater than a
certain threshold
threshold=1.2; %M^2*0.05;
pvars = var(Dl, 0, 1);
idx = pvars > threshold;
Dl = Dl(:, idx);
dict_size    = 1024;          %
dictionary size
Dl=Dl(:,1:dict_size );
```

$$\vartheta_{a,b,\theta}(x) = a^{-1/2}\,\vartheta\left(\frac{(x_1 cos\theta + x_2 sin\theta - b)}{a}\right)$$

$$\vartheta_{a_1,a_2,b_1,b_2}(x) = \vartheta_{a_1,a_2}(x_1)\vartheta_{b_1,b_2}(x_2)$$

$$\vartheta_{a,b}(t) = a^{-1/2}\vartheta_{a,b}(t - b/a)$$

$$t = x_1 cos\theta + x_2 sin\theta$$

DTT dictionary basis

Algorithm. Dictionary based on DTT bases function	
1	D is dictioanry size, a_1 , a_2 , a_3 are coefficients vectors, n_1 , n_2 are the index parameters, $i = n = 0,1,2,\dots,N-1$, $j = m = 0,1,2,\dots,M-1$. f_{n_1} and f_{n_2} are the Tchebichef polynomials.
2	**FOR** *each* $i = 1\!:\!n$ **do**
3	**FOR** *each* $j = 1\!:\!m$ **do**
4	Construct Tchebichef polynomials. $f_{n_1}(v) = (B_1\ v + B_2\ )f_{n_1-1}(v) + B_3\ f_{n_1-2}(v)$ $f_0(v) = 1/\sqrt{M}$ $f_1(v) = (2v + 1 - M)\sqrt{3/M(M^2-1)}$
5	**FOR** *each* $n_1 = 1\!:\!n$ **do**
6	**FOR** *each* $n_2 = 1\!:\!m$ **do**
7	Temp(n_1 , n_2)= $f_{n_1}(v).\,f_{n_2}(v)$
8	$D = temp(:)$
9	**ENDFOR** **END FOR**
10	$D(:,count) = temp(:)$
11	**END FOR** **END FOR**
12	Select size of dictionary $D_1\ (:,count) = D_1\ (:,size\ of\ dictioanry)$ $D = D_1$

$$\boldsymbol{f_{n_1}(v) = (B_1 v + B_2) f_{n_1-1}(v) + B_3 f_{n_1-2}(v)}$$

$$\boldsymbol{B_1 = \frac{2}{n_1}\sqrt{\frac{4{n_1}^2 - 1}{M^2 - {n_1}^2}}}$$

$$\boldsymbol{B_2 = \frac{1-M}{n_1}\sqrt{\frac{4{n_1}^2 - 1}{M^2 - {n_1}^2}}}$$

$$\boldsymbol{B_3 = \frac{n_1 - 1}{n_1}\sqrt{\frac{2n_1 + 1}{2n_1 - 3}}\sqrt{\frac{M^2 - (n_1 - 1)^2}{M^2 - {n_1}^2}}}$$

$$\boldsymbol{F_{n_1 n_2} = \sum_{v=0}^{M-1} f_{n_1}(v) \sum_{u=0}^{M-1} f_{n_2}(u) f(v,u)}$$

DCT dictionary basis

	Dictionary based on DCT Basis Functions
1	D is dictionary size, K_1, K_2 scale factors, n_1, n_2 are the indexing parameters, M_1 number of dictionary atoms, N_1 size of dictionary, M_1, M are scaling factors.
2	**FOR** $K_1 = 1{:}M$ **do**
3	**FOR** $K_2 = 1{:}M_2$ **do**

4	**IF** $K_2 = 1; n = \sqrt{1/M_1}$
	 ELSE
	 $n = \sqrt{1/M_1}$
	 END IF

5	**FOR each** $n_1 = 1{:}M_1$ **do**
6	**FOR each** $n_2 = 1{:}N_1$ **do**
7	$Temp(n_1, n_2) = dct(n_1, n_2, K_1, K_2)$
8	$D = Temp(n_1, n_2)$
9	**END FOR End FOR**
10	$D(:, count) = Temp(:)$
11	**END FOR** **END FOR**

$$X_1 = cos\left[\frac{\pi \times (2n_1 + 1) \times T_1}{2M_1}\right]$$

$$X_2 = cos\left[\frac{\pi \times (2n_2 + 1) \times T_2}{2N_1}\right]$$

$$X_{T_1,T_2} = \sqrt{\frac{2}{M_1}} \times \sqrt{\frac{2}{N_1}} \sum_{n_1=0}^{M_2-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} X_1 X_2$$

DCT dictionary basis

```
basis={};
M=8; N=8;
M1=M^2; N1=N^2;
M2=M; N2=M;

count1=0;
for k1=1:0.5:M1
    if k1==1; nf1=sqrt(1/M); else nf1=sqrt(2/M); end
    for k2=1:0.5:N1
        count1=count1+1;
        if k2==1; nf2=sqrt(1/N); else nf2=sqrt(2/N); end
        for n1=1:M2
            for n2=1:N2
                temp(n1,n2)=nf1*nf2*cos(( (2*(n1-1)+1)*(k1-1)*pi)/(2*M)) *
cos(( (2*(n2-1)+1)*(k2-1)*pi)/(2*N));
            end
        end
        Dl(:,count1)=temp(:);
    end
end


xp=randperm(size(Dl,2));
Dl=Dl(:,xp);
DicDorm = sqrt(sum(Dl.^2));
lNorm = sqrt(sum(Dl.^2));
Idx = find(lNorm);
Dl = Dl(:, Idx);
% % Dl = Dl./repmat(sqrt(sum(Dl.^2)), size(Dl, 1), 1);
dict_size   = 1024;          % dictionary size
Dl=Dl(:,1:dict_size );
```

```
basis=Dl;
count=1;
d=zeros(128,128);
for n1=1:16
    lx=(n1-1)*N+1; hx=lx+N-1;
    for n2=1:16
        ly=(n2-1)*N+1; hy=ly+N-1;
        d(lx:hx, ly:hy)=reshape(basis(:,
count),N,N);
%          if n1==8
%              n1
%              reshape(basis(:,
count),8,8)
%          end
        count=count+1;
        if count > 256
            break
        end
    end
    if count > 256
        break
    end
end

imshow(abs(d),[])
```

DWT dictionary basis

Dictionary based on Difference of Gaussian wavelet Basis Function	
1	D is dictionary size, S_1, S_2 scale factors, θ_1, θ_2 are the dilation parameters, M_1 is number of dictionary atoms, N_1 size of dictionary M, N are scaling factors.
2	**FOR each Scale** $S_1 = 1: M$ **do**
3	**FOR each Scale** $S_2 = 1: M$ **do**
4	**FOR each translation** $\theta_1 = 1: N$ **do**
5	**FOR each translation** $\theta_2 = 1: N$ **do**
6	**FOR each** $n_1 = 1: M_1$ **do**
7	**FOR** $n_2 = 1: M_1$ **do**
8	$temp(n_1, n_2) = DOG\_Wavelet(S_1, S_2, \theta_1, \theta_2)$
9	$D = temp(:)$
10	**ENDFOR END FOR**
11	$D(:, count) = temp(:)$

12	**END FOR**
	END FOR
	ENDFOR
	ENDFOR

$$\varphi(t) = \frac{1}{\pi^{1/4}} e^{j\omega_0 t} e^{-t^2/2}$$

$$F(n_1) = 2^{S_1/2} \sin(5(n_1 - \tau)) e^{-t^2/2}$$

$$F(n_2) = 2^{S_2/2} \sin(5(n_2 - \tau)) e^{-t^2/2}$$

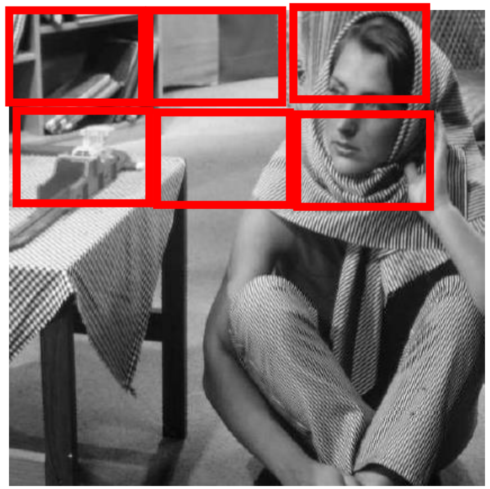
$$F(n_1, n_2) = F(n_1) \times F(n_2)$$

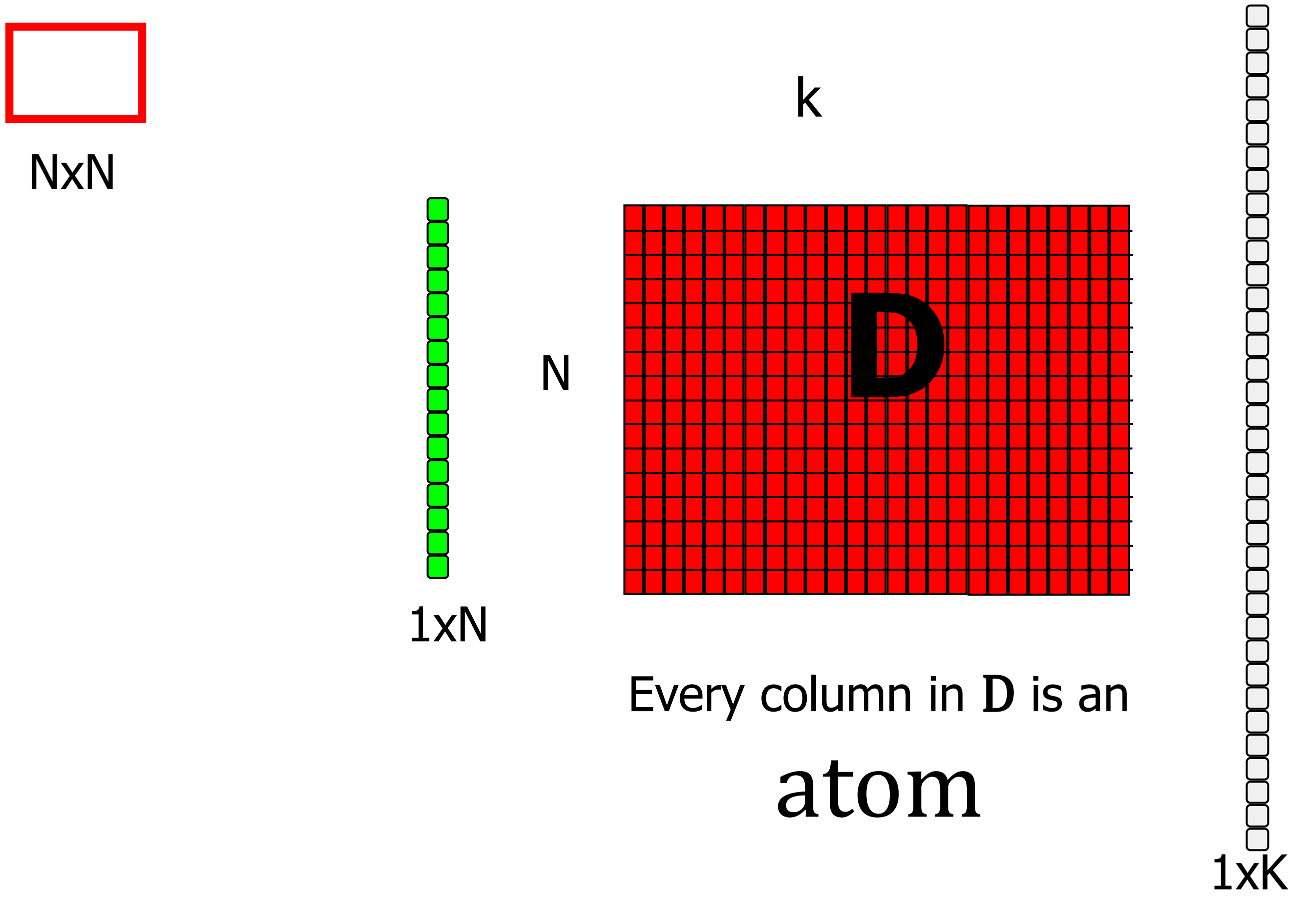
DWT dictionary basis

```
s={};
count1=0; count2=0;
for s1=1:6
    for s2=1:6
        count1=count1+1;
        count2=0;
        for tau1=0:0.1:0.2
            for tau2=0:0.1:0.2
                count2=count2+1;
                for n1=1:M
                    for n2=1:N
                        t1=(n1-tau1)/s1;
                        t2=(n2-tau2)/s2;
                       temp(n1,n2)= 2^(s1/2)* sin(5*t1)*exp(-
t1^2/2)*2^(s2/2)* sin(5*t2)*exp(-t2^2/2); %2^(s/2) % Morelet wavelet
%                          temp(n1,n2)
=2^(s1/2)*DoG_wavelet(n1,s1,tau1)*2^(s2/2)*DoG_wavelet(n2,s2,tau2);  %
DoG
%                          temp(n1,n2)=(4/(36*pi^0.5))*(1-
t1^2/sigma^2)*exp(-t1^2/(2*sigma^2))*...
%                               (1-t2^2/sigma^2)*exp(-t2^2/(2*sigma^2));
%
%                            temp(n1,n2)=Meyer_wavelet(t1)*
Meyer_wavelet(t2); % Meyer wavelet
                        basis{count1, count2}=temp;
                    end
                end
            end
        end
    end
end
```

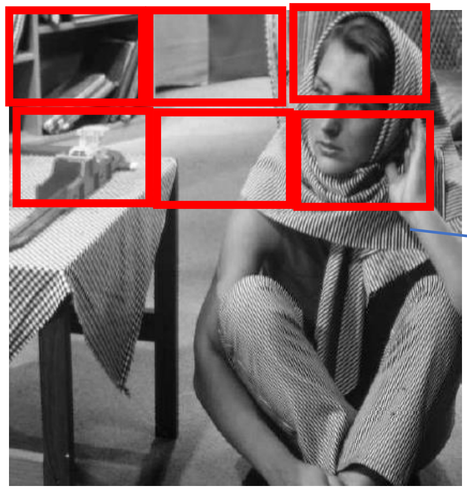
```
xp=randperm(size(Dl,2));
% Dl=Dl(:,xp);


DicDorm = sqrt(sum(Dl.^2));
lNorm = sqrt(sum(Dl.^2));
Idx = find(lNorm);
Dl = Dl(:, Idx);

Dl = Dl./repmat(sqrt(sum(Dl.^2)),
size(Dl, 1), 1);

DicWavelet=Dl;
```

Sparse Representation Dimension





Sparse Representation Coefficients



NxN

```
%% compute sparse cofficents using overcmplete dictionary
D=randn(64,1024); % compute over-complete dictionary with size
200*1024.
% load Dic_Dl_1024_8_woSbtMean.mat
% D=Dl;
patch_size = sqrt(size(D, 1));
I=double(imread('TL.jpg')); % input image
I=I(1:100,1:100);
[m n]=size(I);
Mean_I1=mean(mean(I))
for ii=1:m-patch_size+1%:m-patch_size+1,
    count=1;
    if (ii>1 && ii<m-patch_size+1)
    for jj = 1:n-patch_size+1%:n-patch_size+1,
        if (jj>1 && jj<m-patch_size+1)
        patch=I(ii:ii+patch_size-1, jj: jj+patch_size-1);
        Mean_patch = mean(patch(:));
        patch1=single(patch-Mean_patch);
        sparse_coeff = SolveOMP(D2, patch1(:), size(D2,2),10);
        sparse_coeff1(jj,count)=max(sparse_coeff); % store spares
coefficients of first image. Similarly you can store all spare
coefficients based on different number of images
        end
    end
    count=count+1;
    end
end
```

OMP(orthogonal Matching Pursuit)

```
function [sols, iters, activationHist] = SolveOMP(A, y, N, maxIters,
lambdaStop, solFreq, verbose, OptTol)
% SolveOMP: Orthogonal Matching Pursuit
% Usage
%   [sols, iters, activationHist] = SolveOMP(A, y, N, maxIters,
lambdaStop, solFreq, verbose, OptTol)
% Input
%   A           Either an explicit nxN matrix, with rank(A) = min(N,n)
%               by assumption, or a string containing the name of a
%               function implementing an implicit matrix (see below for
%               details on the format of the function).
%   y           vector of length n.
%   N           length of solution vector.
%   maxIters    maximum number of iterations to perform. If not
%               specified, runs to stopping condition (default)
%   lambdaStop  If specified, the algorithm stops when the last
coefficient
%               entered has residual correlation <= lambdaStop.
%   solFreq     if =0 returns only the final solution, if >0, returns an
%               array of solutions, one every solFreq iterations
(default 0).
%   verbose     1 to print out detailed progress at each iteration, 0
for
%               no output (default)
%   OptTol      Error tolerance, default 1e-5
% Outputs
%    sols           solution(s) of OMP
%    iters          number of iterations performed
%    activationHist Array of indices showing elements entering
%                   the solution set
%
```

```
Description
%   SolveOMP is a greedy algorithm to estimate
the solution
%   of the sparse approximation problem
%      min ||x||_0 s.t. A*x = b
%   The implementation implicitly factors the
active set matrix A(:,I)
%   using Cholesky updates.
%   The matrix A can be either an explicit
matrix, or an implicit operator
%   implemented as an m-file. If using the
implicit form, the user should
%   provide the name of a function of the
following format:
%     y = OperatorName(mode, m, n, x, I, dim)
%   This function gets as input a vector x and an
index set I, and returns
%   y = A(:,I)*x if mode = 1, or y = A(:,I)'*x if
mode = 2.
%   A is the m by dim implicit matrix implemented
by the function. I is a
%   subset of the columns of A, i.e. a subset of
1:dim of length n. x is a
%   vector of length n is mode = 1, or a vector
of length m is mode = 2.
```

To Summarize So Far …

Image denoising (and many other problems in image processing) requires a model for the desired image

What do we do?

We proposed a model for signals/images based on sparse and redundant representations

Problems?

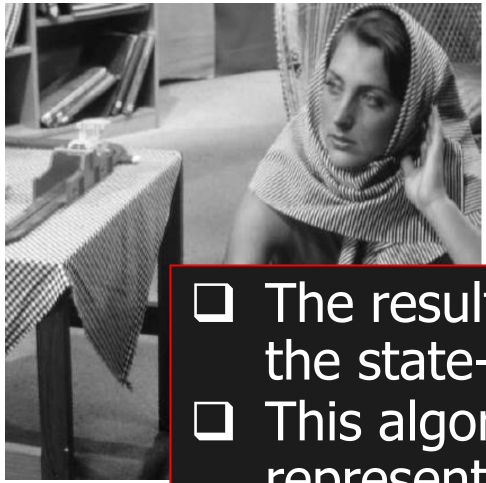
We have seen approximation methods that find the sparsest solution, and theoretical results that guarantee their success. We also saw a way to learn **D**

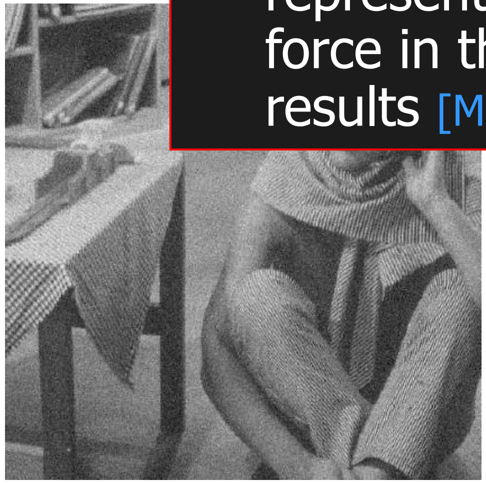
What next?

Will it all work in applications?

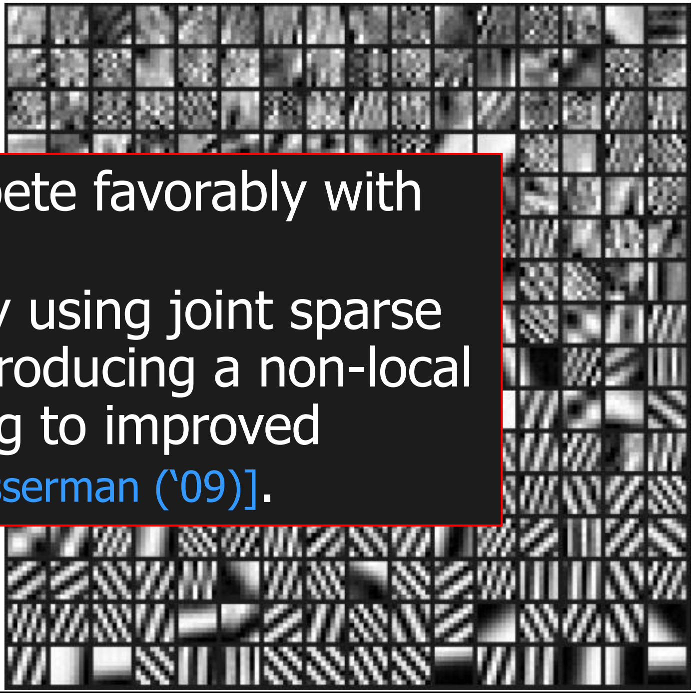
Applications in image processing

Image Denoising (Gray) [E. & Aharon ('06)]





- The results of this algorithm compete favorably with the state-of-the-art.
- This algorithm can be extended by using joint sparse representation on the patches, introducing a non-local force in the denoising, thus leading to improved results [Mairal, Bach, Ponce, Sapiro & Zisserman ('09)].



The obtained dictionary after 10 iterations

Inpainting [Mairal, E. & Sapiro ('08)]

Original 80% missing Result

Ori

Spare Representation in Stereo Vision

Sparse Representation(SR) Algorithm

Disparity map estimation using sparse representation

$$(T_0)\ \ min_D \|\alpha_1\|_0 \ \text{ subject to } B_1\text{=D}\alpha_1$$
$$(T_0,\varepsilon)\ \ min_D \|\alpha_1\|_0 \ \text{ subject to } \|B_1 - \text{D}\alpha_1\| \le \varepsilon$$

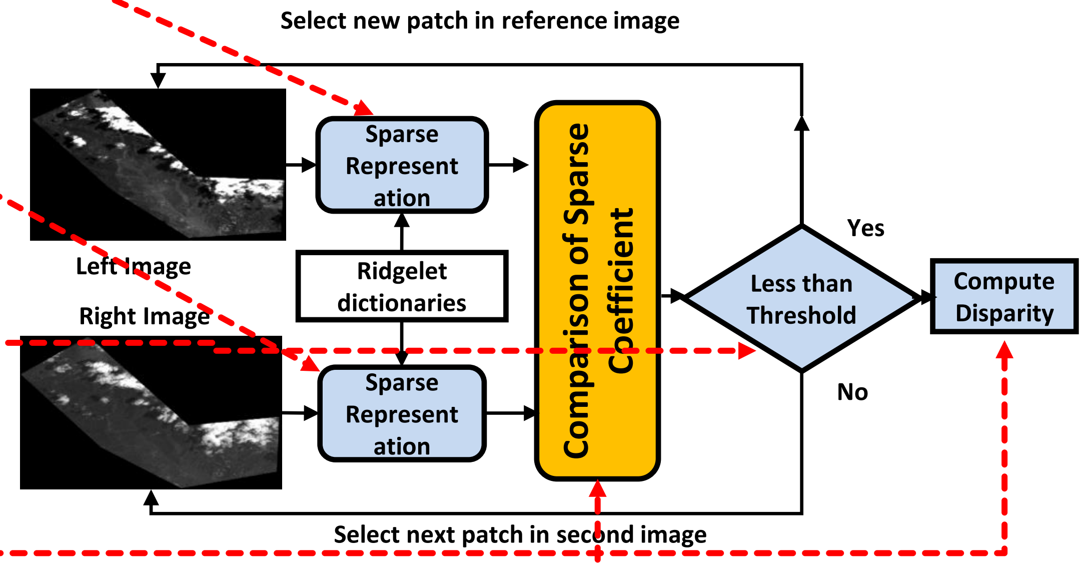
$$(T_0)\ \ min_D \|\alpha_2\|_0 \ \text{ subject to } B_2\text{=D}\alpha_2$$
$$(T_0,\varepsilon)\ \ min_D \|\alpha_2\|_0 \ \text{ subject to } \|B_2 - \text{D}\alpha_2\| \le \varepsilon$$

$$(f_1(\text{d})\text{=}\ min_D \textstyle\sum_k |\alpha_1 - \alpha_{2k}|$$

$$\text{d} \ge d_{min}$$
$$\text{d=}(y_r\text{-}y_l) \text{ and } x_l\text{=}x_r$$

$$\text{H(i,j)=d}$$

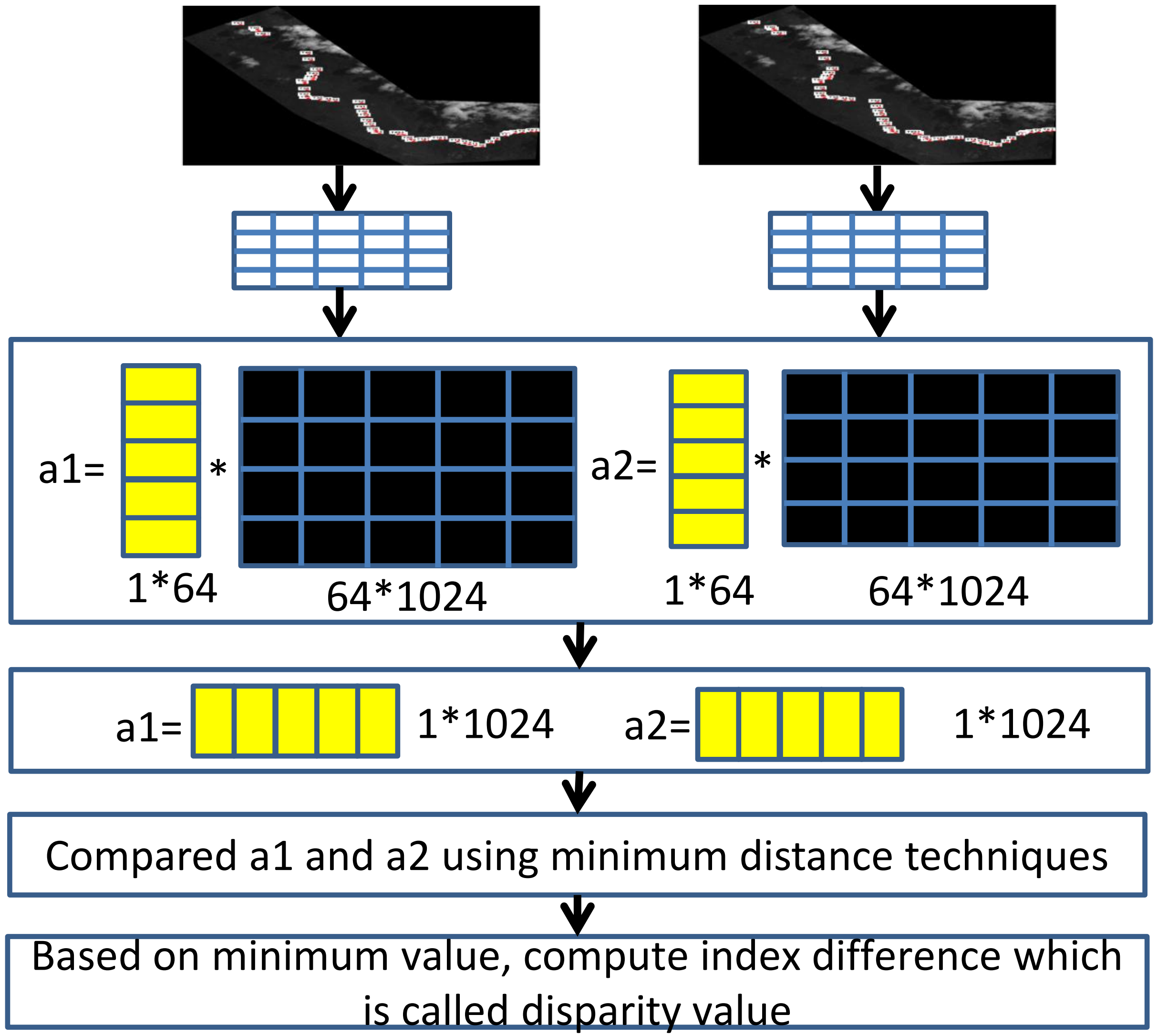
- **Spares representation** is a very active area in the field of computer vision and image processing of present era.
- **Dictionary Construction**
  - The different type of dictionaries are used to represent the sparse coefficients.

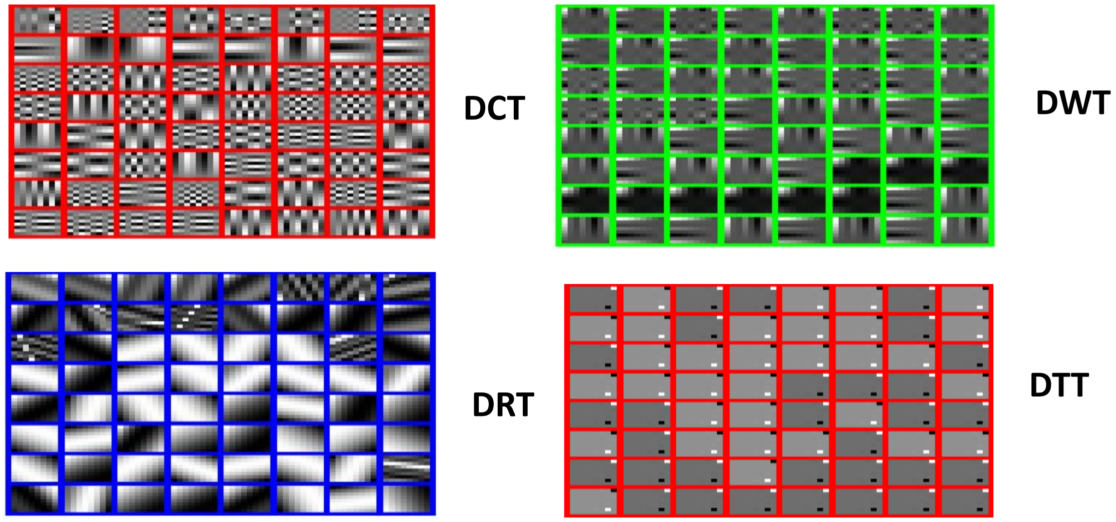


Flowchart of proposed Method based on sparse representation.

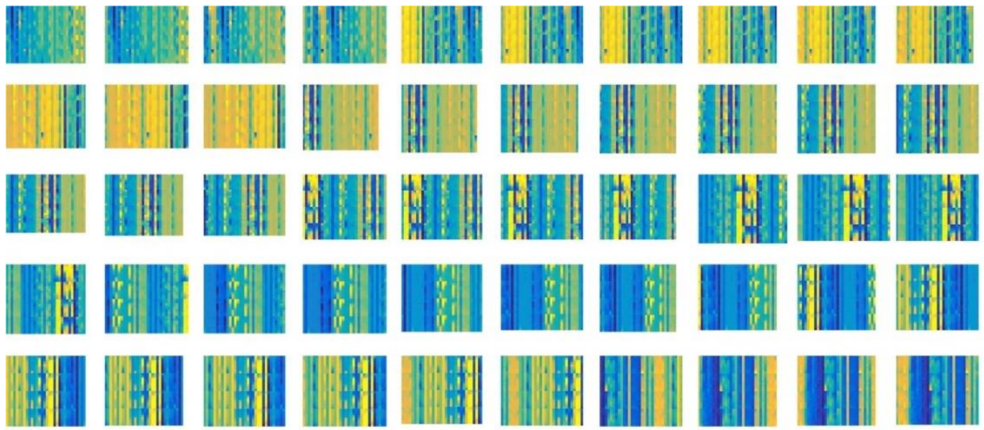
Pre-determined Dictionaries

Flow Diagram based on Sparse Representation



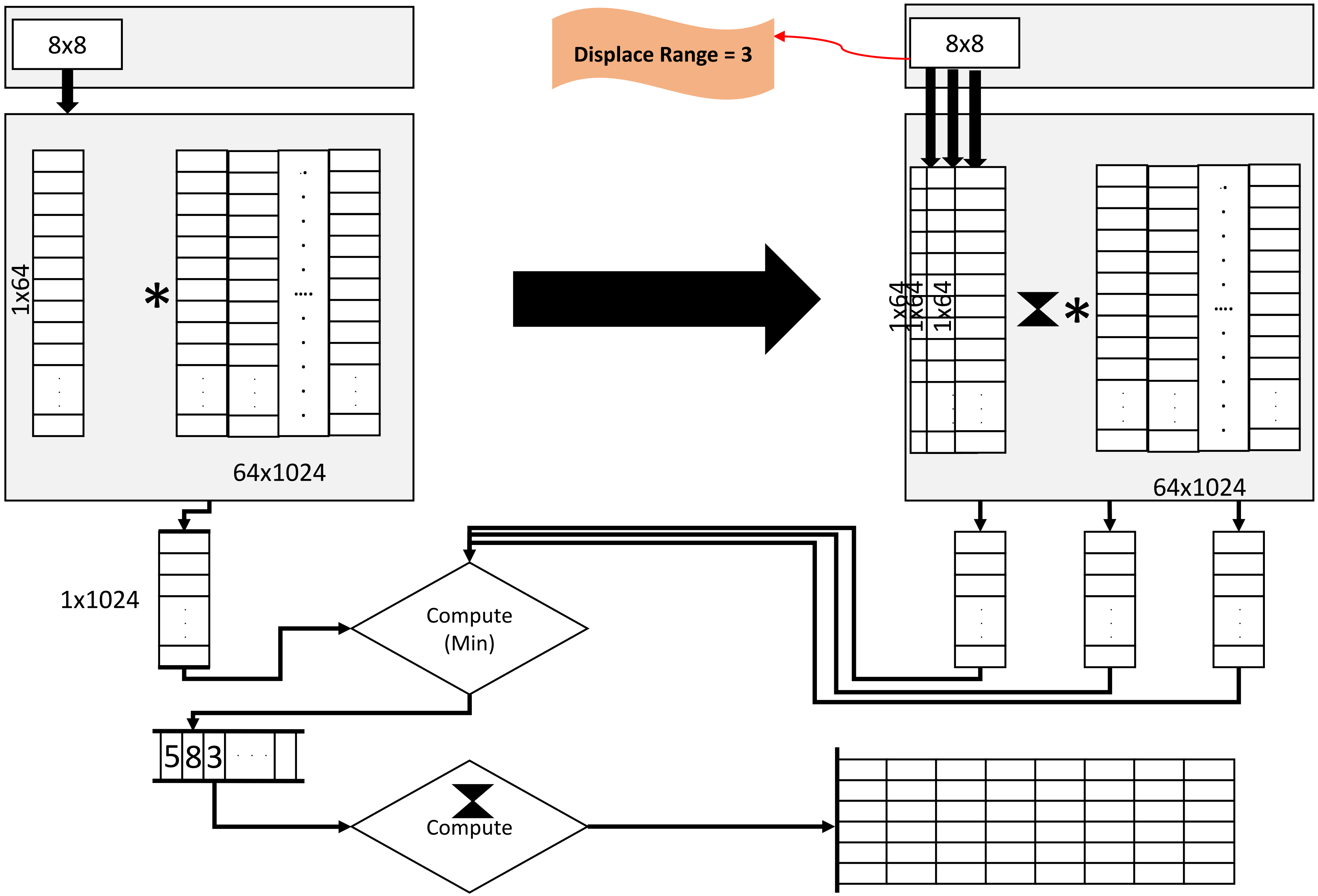


Pre-specified Dictionaries used in sparse representation reconstruction.



Dictionary contraction Atoms using dictionary basis function.

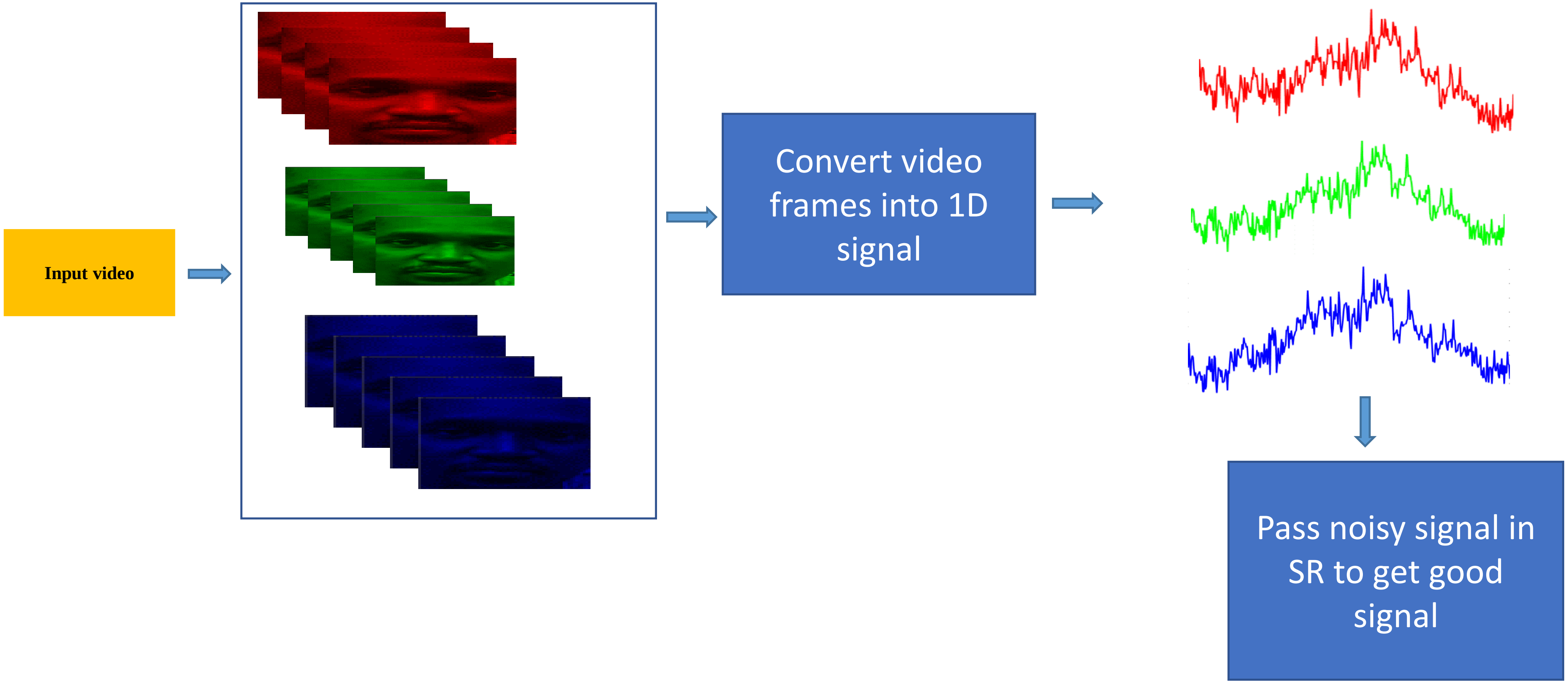
Pre-determined Dictionaries



Disparity Map estimation using SR algorithm

- Extract patches from left and right images and apply SR Algorithm.
- Compare number of patches in a certain disparity range (d=10:30) from right image with left image patch.
- Compute distances based on similar vectors Using Euclidian distance approach

Video Frames and sparse Representation method

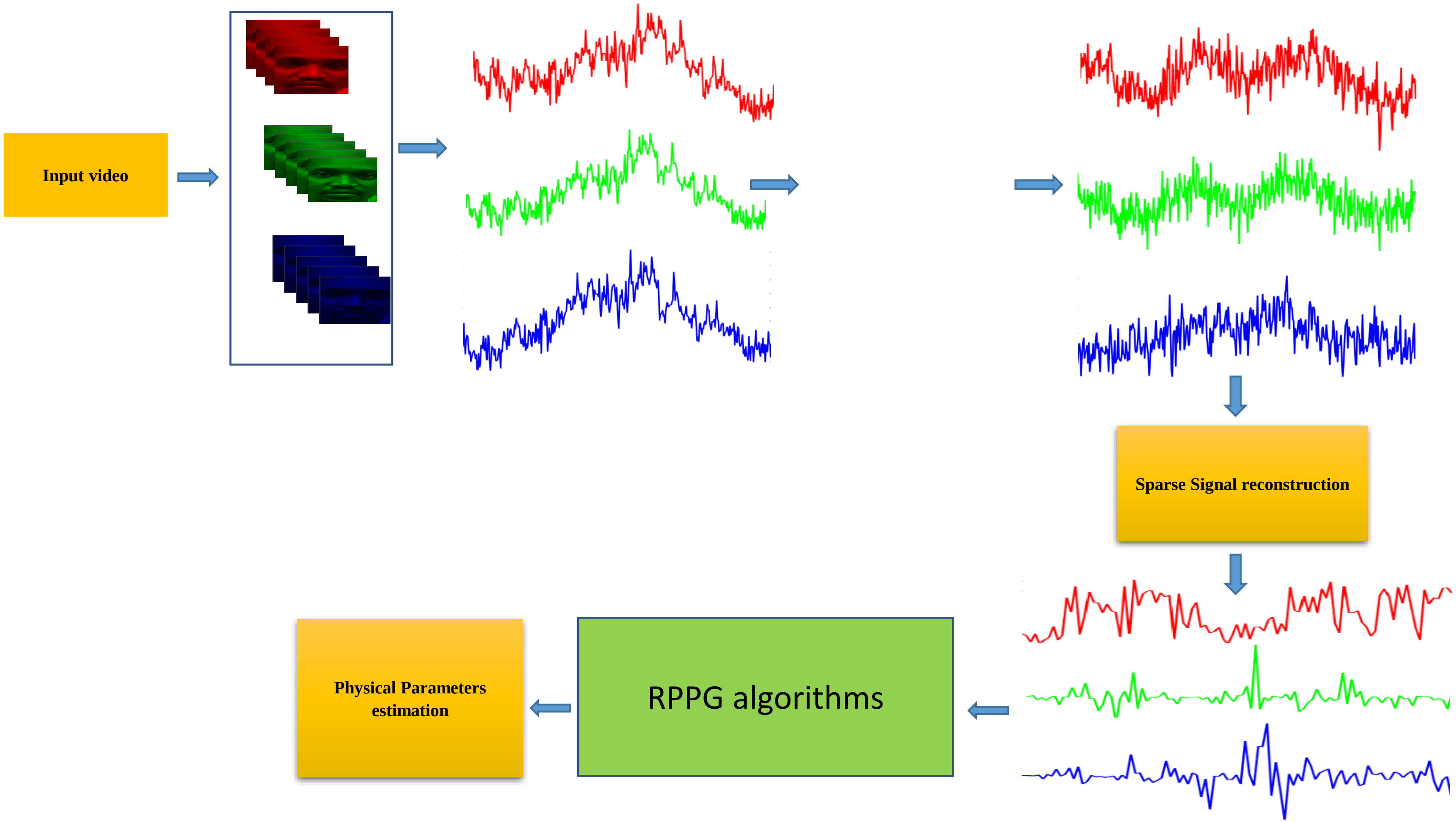


Video Frames Extraction

```
%%% Data Acuistion %%%%%%%%%%%%%%%
%srcFiles=dir('C:\datasetnew\MAHNOB-HCI\Video6\*.avi');
    srcFiles=dir('C:\datasetnew\MAHNOB-HCI\Subjectvideo\*.avi');
    filename = srcFiles.name;
        obj = VideoReader(filename);
        get(obj);
        vid1 = read(obj,1);
         a=imresize(vid1,2);
        vid=a;
        frames = obj.NumberOfFrames;
        S_time=2;
        E_time=12;
        Frame_rate=60;
        S_frame_number=S_time*Frame_rate;
        E_frame_number=S_frame_number+abs(E_time-
S_time+1)*Frame_rate;
%    imshow(a)
%    imcrop
        faceDetector = vision.CascadeObjectDetector;
        faceDetector.MergeThreshold=1;
        bbox = step(faceDetector, vid);
% Draw the returned bounding box around the detected face.
 videoFrame = insertShape(vid, 'Rectangle', bbox);
 figure; imshow(videoFrame); title('Detected face');

%Processing loop
for x= 1 : abs(E_frame_number-S_frame_number)
    F_ind=S_frame_number+x;
    videoFrame = (read(obj,F_ind));
     yFace = faceroi(videoFrame,0.2,0,0.6,0.3);
%          figure(2);
         imshow(yFace);
    R1(x) =mean2(yFace(:,:,1));
    G1(x) =mean2(yFace(:,:,2));
    B1(x) =mean2(yFace(:,:,3));
```

```
R_m=mean(R1);
G_m=mean(G1);
B_m=mean(B1); % Extracting the mean
value for each spectrum

R_sd=std(R1);
G_sd=std(G1);
B_sd=std(B1); % extracting the standard
deviation for each spectrum

R_n = (R1-R_m)/R_sd;
G_n = (G1-G_m)/G_sd;
B_n = (B1-B_m)/B_sd; % Normalized R,G,B
raw traces from the ROI
H_d=detrend(H1);
S_d=detrend(S1);
V_d=detrend(V1);
%***************************************
*************************************
H_n = (H_d-H_m)/H_sd;
S_n = (S_d-S_m)/S_sd;
V_n = (V_d-V_m)/V_sd; % Normalized R,G,B
raw traces from the ROI
```

Assessment of Vital Signs from Smartphone Face Video Analytical Approach Based on Sparse Signal Reconstruction



Assessment of Vital Signs from Smartphone Face Video Analytical Approach Based on Sparse Signal Reconstruction

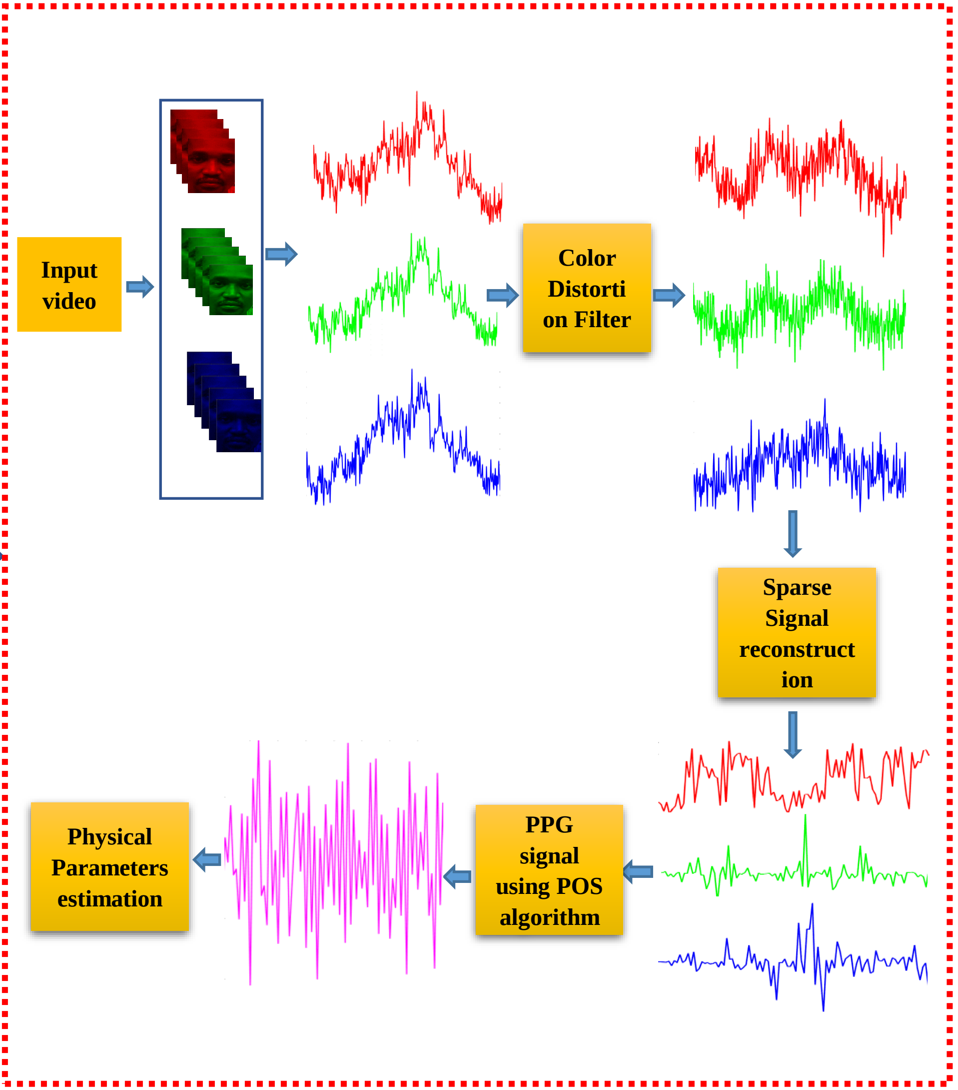
Number of Subject(18):
(8 female and 10 males)

Duration :1 min duration smartphone camera at 30 frames per second (Res:1080 × 1920)

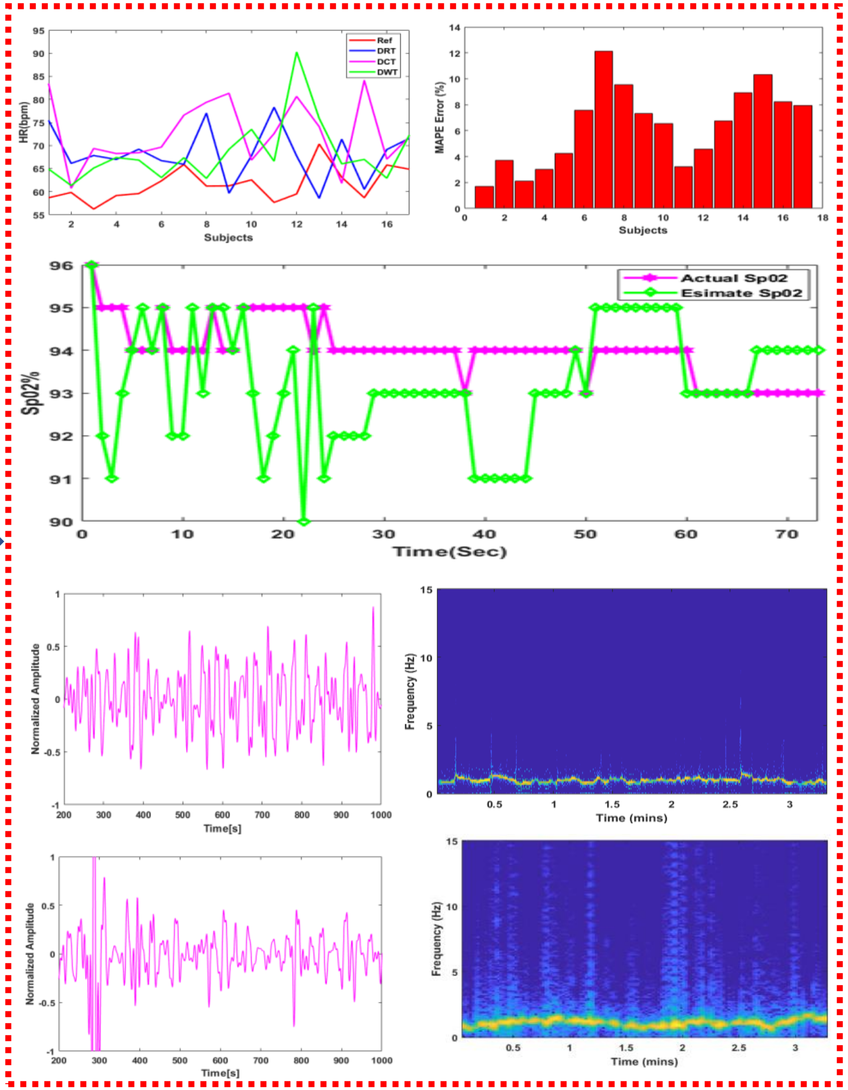
Sessions: Morning(9 am) Evening (9-10 pm)

Ground Truth:
Device: WristOx2 model 3150 wrist-worn

Dataset
(Dataset Collected at UTP, Malaysia)

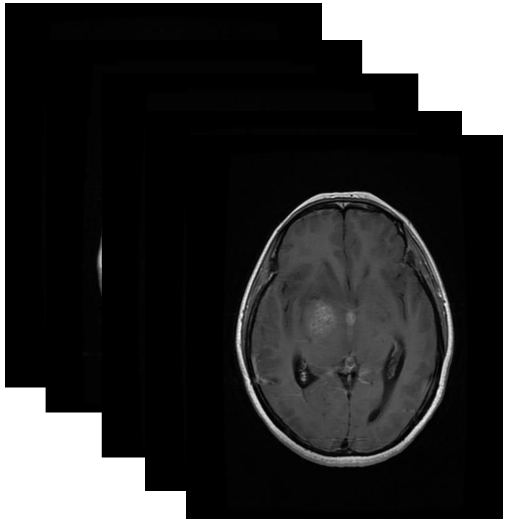


Proposed Model

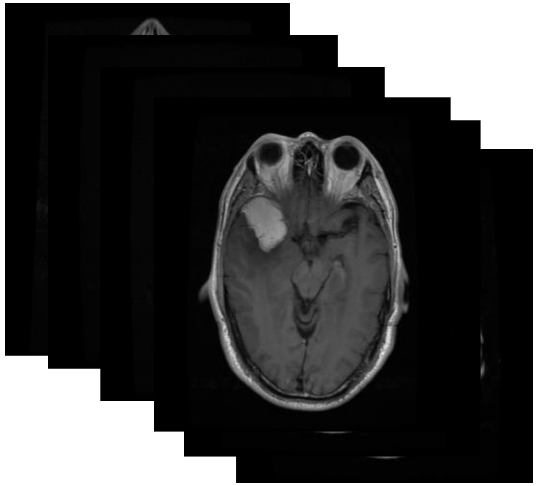


Results

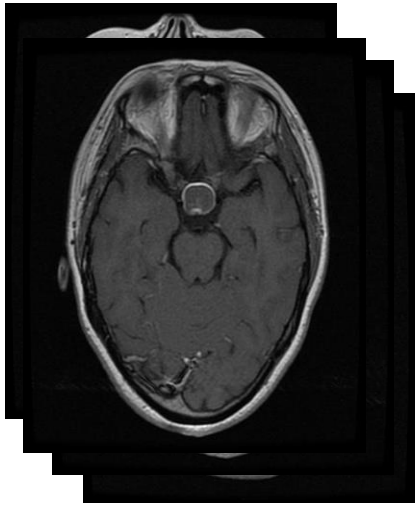
Image Classification (Feature Extraction for Classification)



Glioma(1426)



Meningioma(708)



pituitary(930)

Goal : Extract Features from each samples and classify these tumor types.

Class 1: Glioma(number of samples or images=1426)

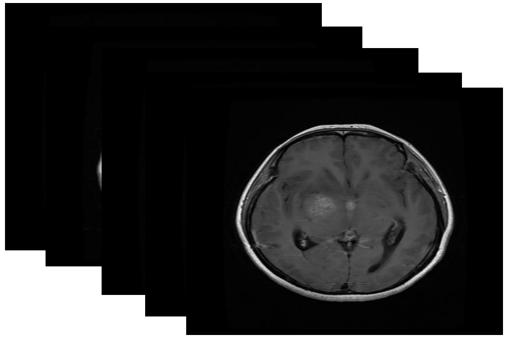
Class2: Meningioma(number of samples or images=708)

Class3: Pituitary(number of samples or images=930)

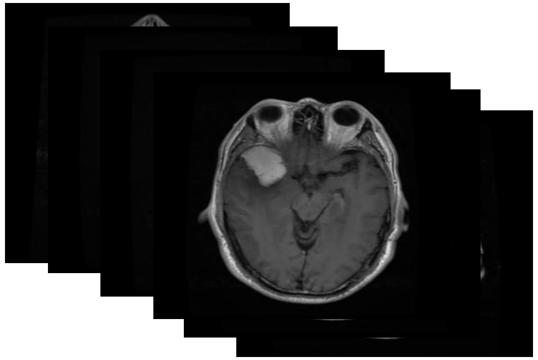
Input size of each sample=512x512

https://figshare.com/articles/brain_tumor_dataset/1512427/5

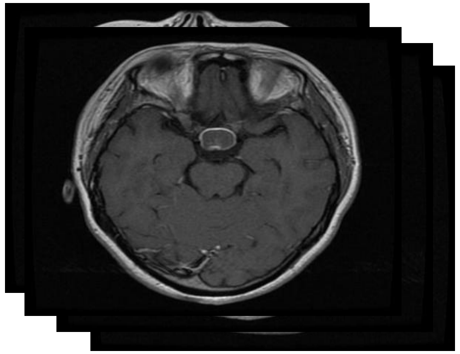
Image Classification (Feature Extraction for Classification)



Glioma(1426)

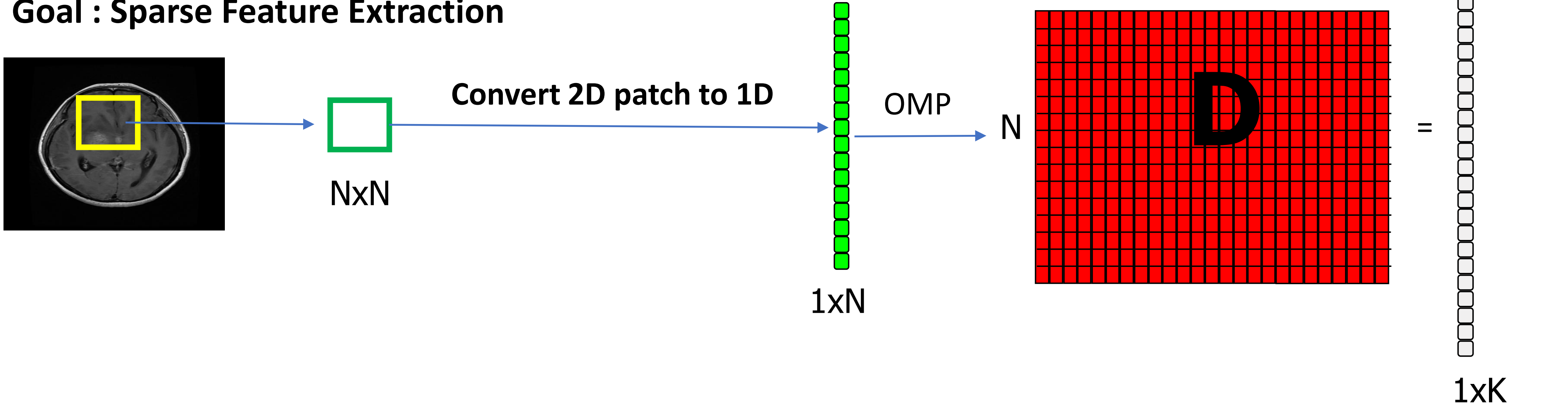


Meningioma(708)



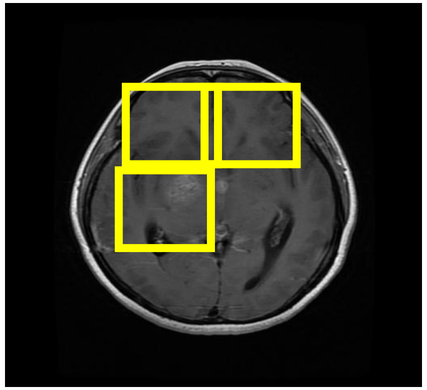
pituitary(930)

Goal : Sparse Feature Extraction



https://figshare.com/articles/brain_tumor_dataset/1512427/5

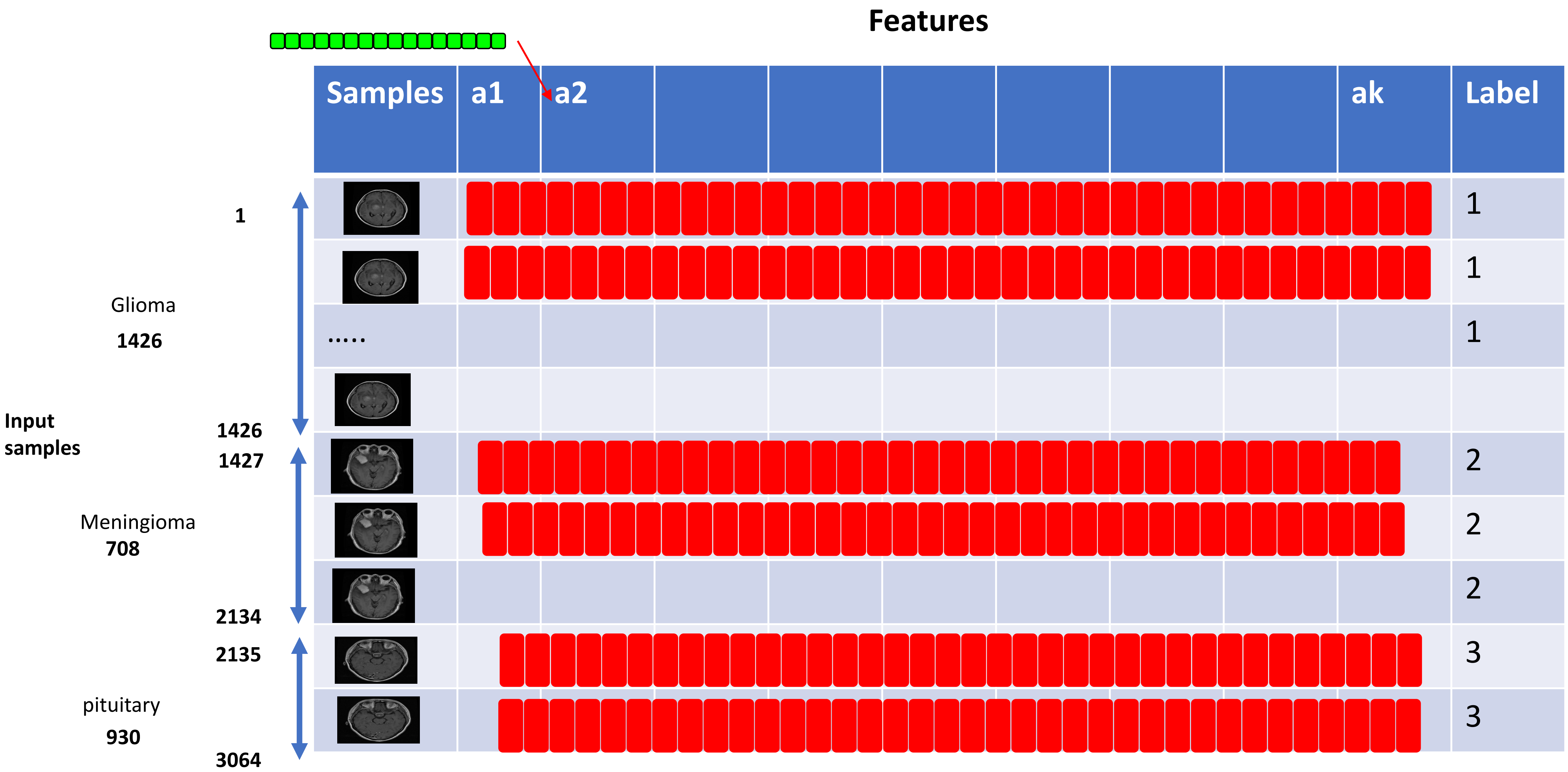
Sparse Representation Coefficients



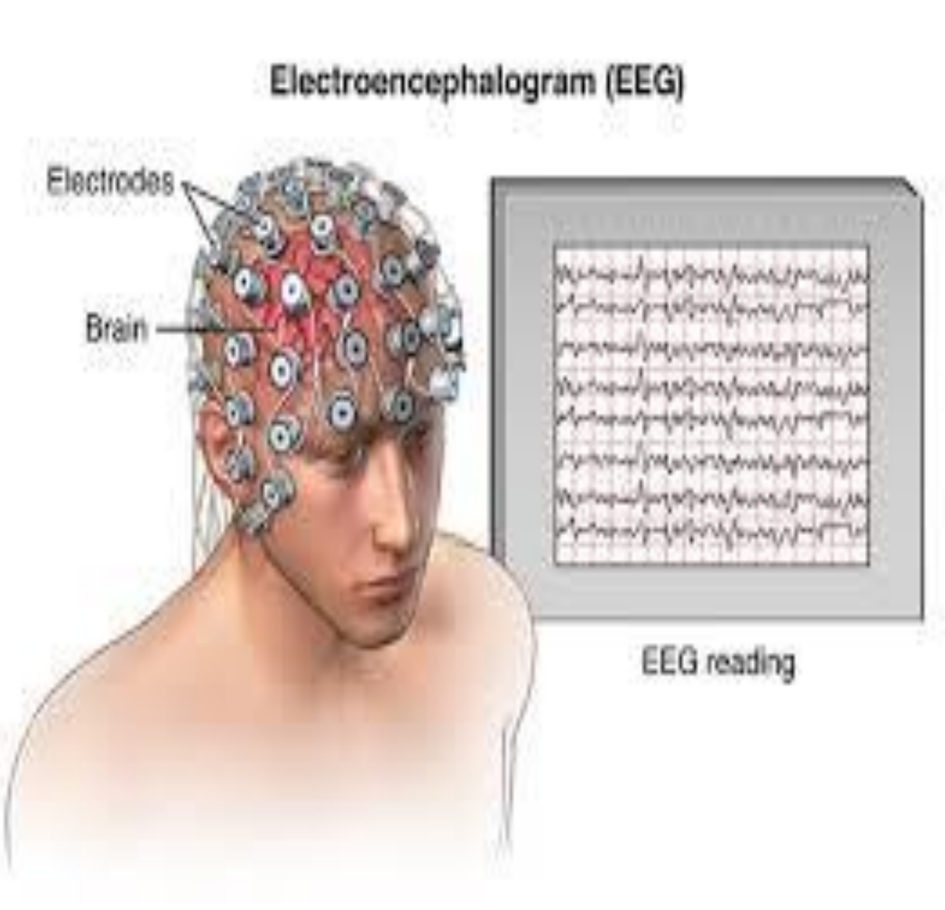
NxN

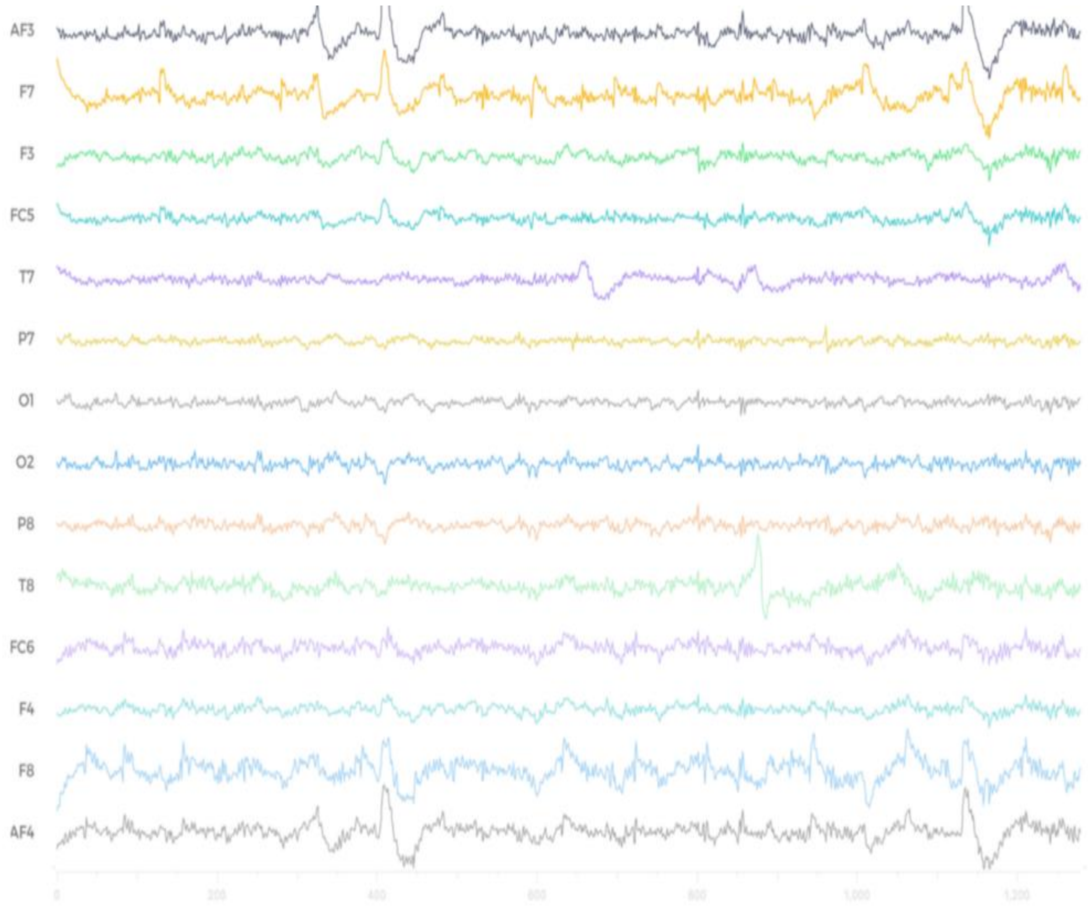
```
%% compute sparse cofficents using overcmplete dictionary
D=randn(64,1024); % compute over-complete dictionary with size
200*1024.
% load Dic_Dl_1024_8_woSbtMean.mat
% D=Dl;
patch_size = sqrt(size(D, 1));
I=double(imread('TL.jpg')); % input image
I=I(1:100,1:100);
[m n]=size(I);
Mean_I1=mean(mean(I))
for ii=1:m-patch_size+1%:m-patch_size+1,
    count=1;
    if (ii>1 && ii<m-patch_size+1)
    for jj = 1:n-patch_size+1%:n-patch_size+1,
        if (jj>1 && jj<m-patch_size+1)
        patch=I(ii:ii+patch_size-1, jj: jj+patch_size-1);
        Mean_patch = mean(patch(:));
        patch1=single(patch-Mean_patch);
        sparse_coeff = SolveOMP(D2, patch1(:), size(D2,2),10);
        sparse_coeff1(jj,count)=max(sparse_coeff); % store spares
coefficients of first image. Similarly you can store all spare
coefficients based on different number of images
        end
    end
    count=count+1;
    end
end
```

Image Classification (Feature Extraction for Classification)



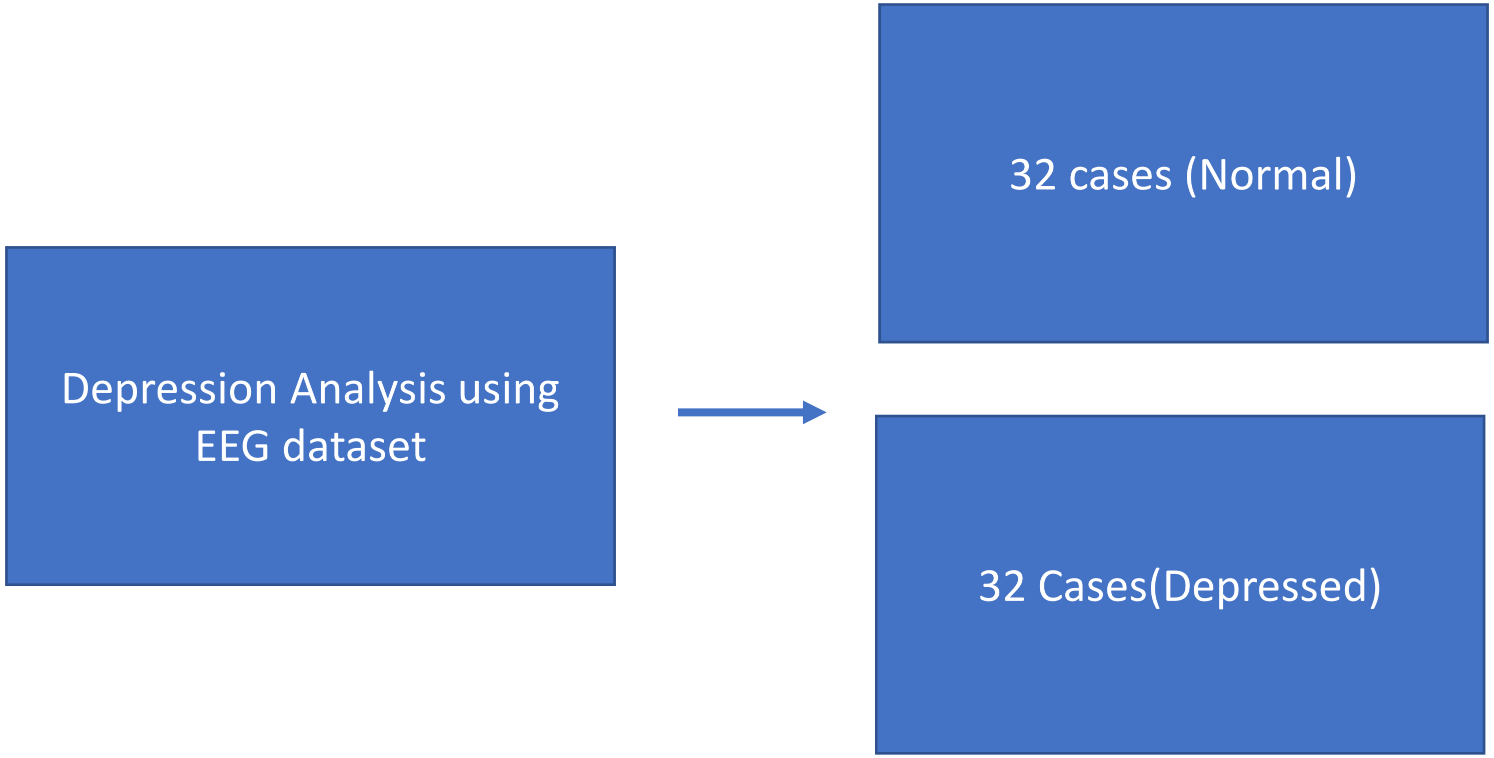
Signal Classification (Feature Extraction for Classification)





https://www.google.ca/search?q=EEG&tbm=isch&ved=2ahUKEwj9zNLKtNHoAhUFNxoKHT-hA5gQ2-cCegQIABAA&oq=EEG&gs_lcp=CgNpbWcQA1AAWABg5L0HaABwAHgAgAEAiAEAkgEAmAEAqgELZ3dzLXdpei1pbWc&sclient=img&ei=rd-JXr3CPIXuaL_CjsAJ&bih=674&biw=1536

Signal Classification (Feature Extraction for Classification)



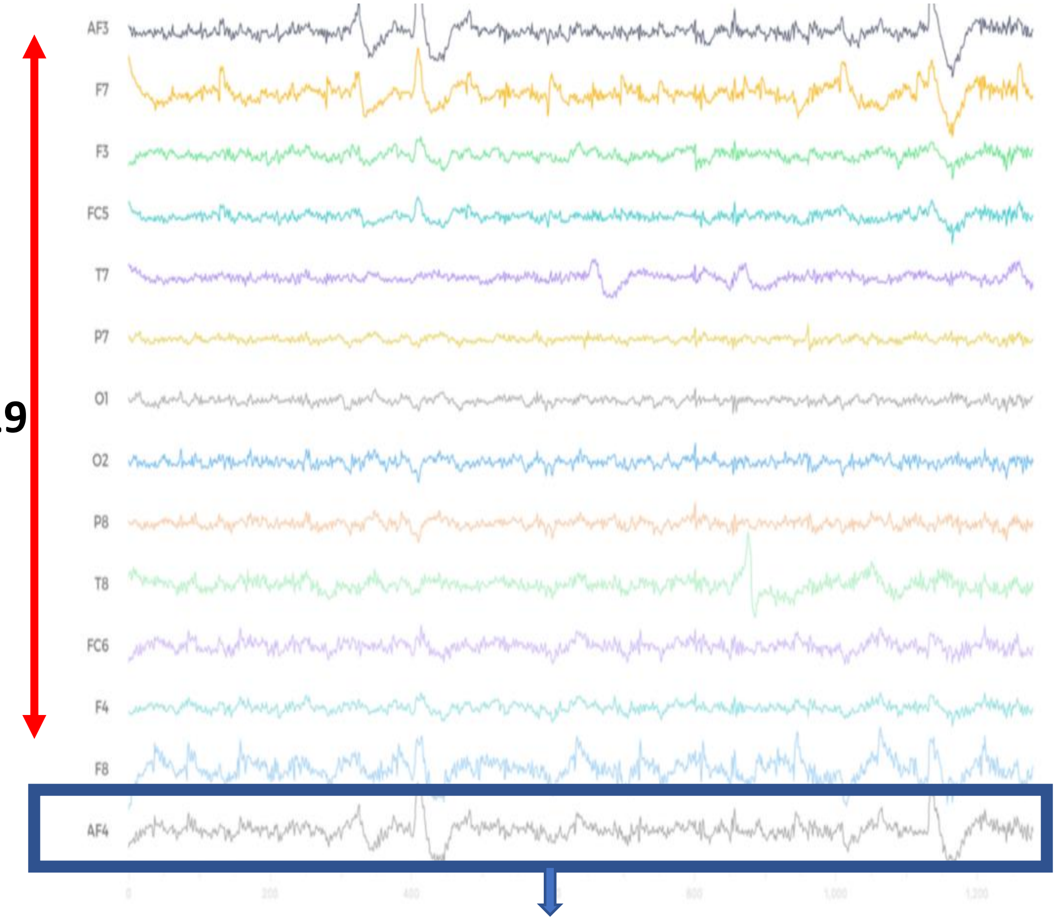
Data acquisition used two scenario: (Eye open and eye close)
For Eye open (32 cases for normal(normal) and 32 cases(depressed))
For Eye close (32 cases for normal(normal) and 32 cases(depressed))
Acquisition time is 5 minutes
Number of channels: 19
Number of samples point (256 sample point per second(256x60x5))

Signal Classification (Feature Extraction for Classification)

Data acquisition used two scenario: (Eye open and eye close)
For Eye open (32 cases for normal(normal) and 32 cases(depressed))
For Eye close (32 cases for normal(normal) and 32 cases(depressed))
Acquisition time is 5 minutes
Number of channels: 19
Number of samples point (256 sample point per second(256x60x5))

Number of samples in class1 (normal)=32
Number of samples in class2 (Depressed)=32

Number of channels: 19

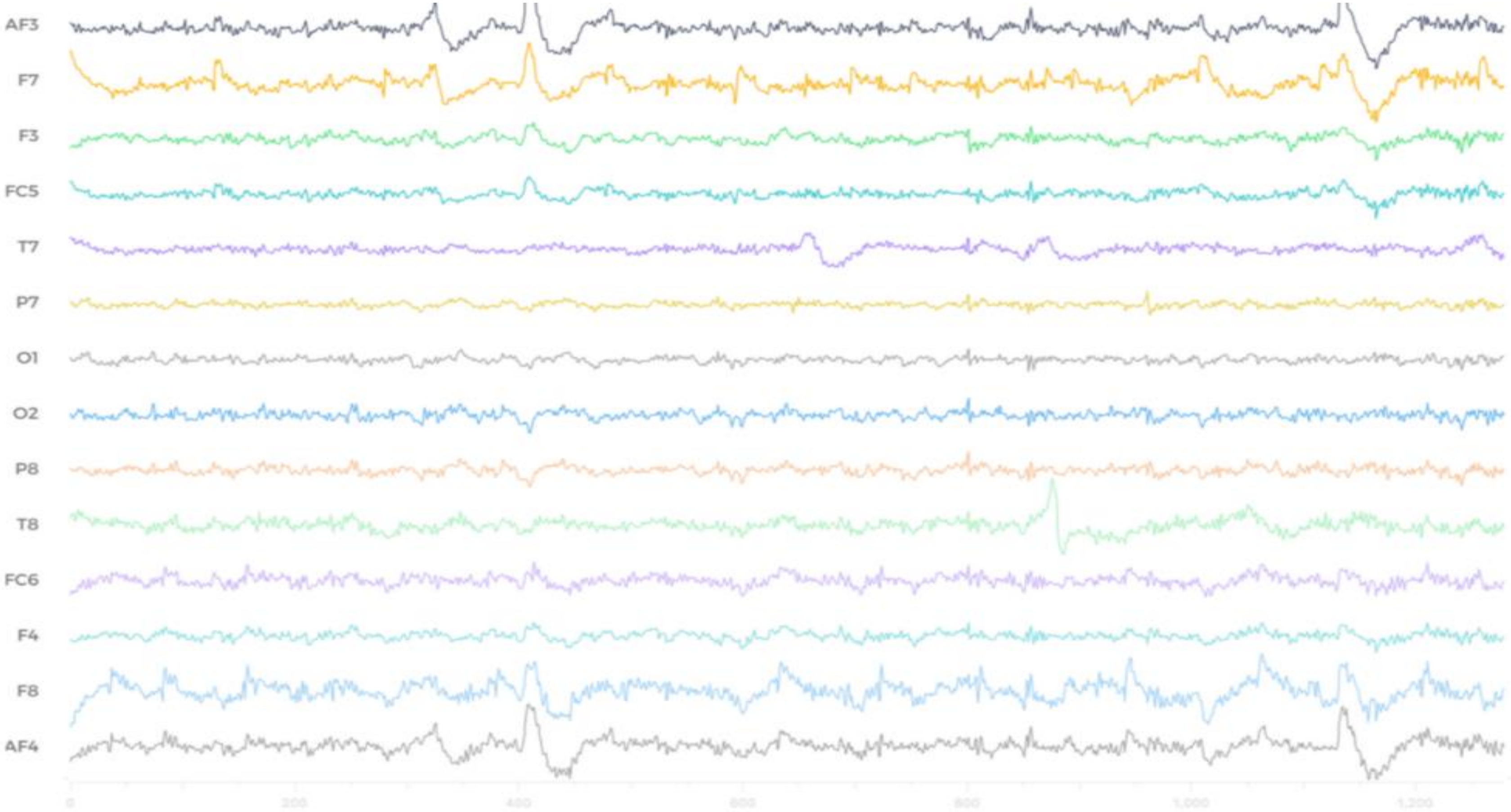


(256x60x5))

Data length for sample 1=19x256x60x5=19x75000(each signal length 1x75000)
Data length for class 1 samples=32x19x75000
Data length for class 2 samples=32x19x75000

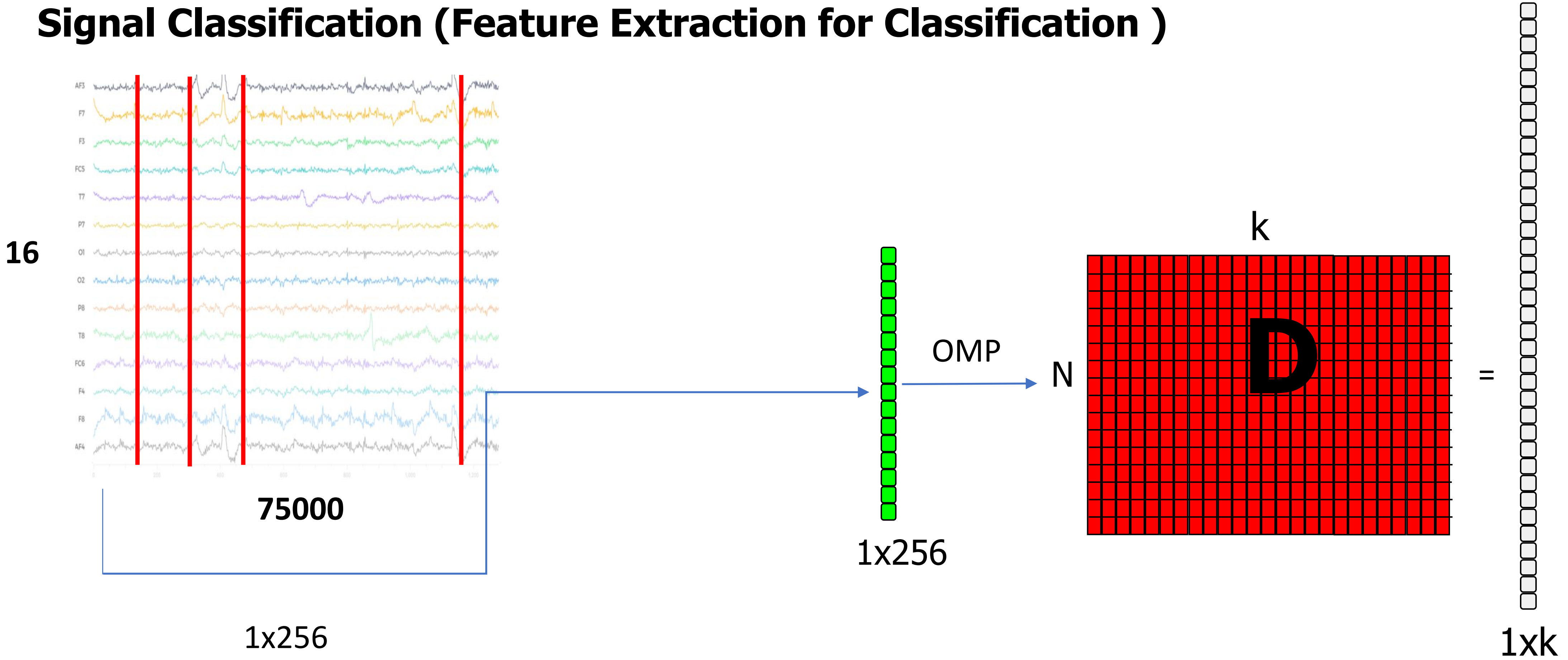
Signal Classification (Feature Extraction for Classification)

Number of channels: 16



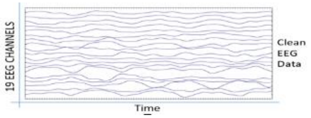
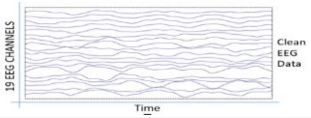
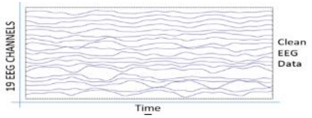
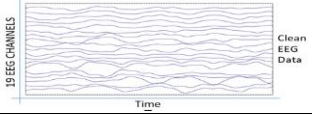
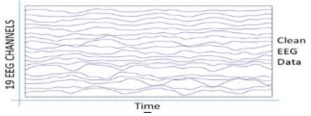
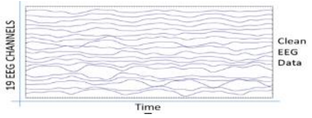
(1x75000 (each signal length))

Signal Classification (Feature Extraction for Classification)



Take spare vector as a feature for one second data=1x256
Spare vector gives 10 non zeros value signal(take mean or maximum for 1 second data)
If we have one minute (60 second=60 sparse point)
5 minute signal=300 spares samples or features

Feature extraction based on EEG dataset

No of samples	Classes	Dimension	Labels
1	Patient1	19x256 	0
2	Patient1	19x256 	0
	Patient1	----	0
	Patient1	---	0
32	Patient1	19x256 	0
33	Patient2	19x256 	1
	Patient2	19x256 	1
	Patient2	----	1
63	Patient2	----	1
64	Patient2	19x256 	1

Feature extraction based on SR

No of samples	Classes	Feature vector	Labels
1	Patient1	1 300 1 16	0
2	Patient1	1 300 1 16	0
	Patient1	----	0
	Patient1	---	0
32	Patient1	1 300 1 16	0
33	Patient2	1 300 1 16	1
	Patient2		1
	Patient2	----	1
63	Patient2	--- 1 300	1
64	Patient2	1 16	1

16 vector features

16x32=512 samples for class1

16x32=512 samples for class2


Q&A