Block Compressed Sensing of Images and Video

Sungkwang Mun

Department of Electrical & Computer Engineering
Geosystems Research Institute
Mississippi State University

Oct. 9. 2012

MISSISSIPPI STATE UNIVERSITY™

Outline

Motivation

Traditional Sampling

- Sampling theorem states the exact reconstruction is possible for any signal satisfying Nyquist rate
- Minimal prior knowledge required
- Commonly interested in more restricted class of signals (i.e. more structured signals)
- Possible to go below the rate with certain prior knowledge of the signal being sampled
  - bandpass

Motivation

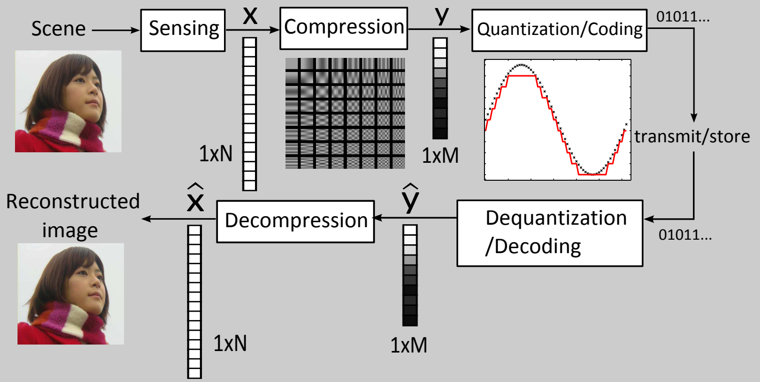
Sparsity in Natural Images

Wavelet transform

Image using 5 % coeffs

Motivation

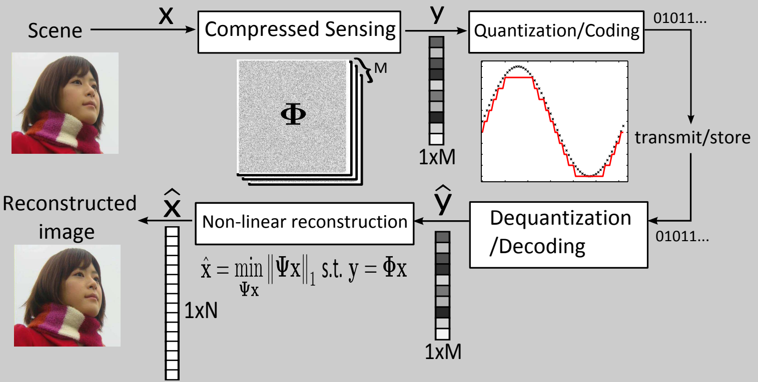
Traditional Compression



- Eventually a few percent of the meaningful data are stored/transferred
- Could we **directly** obtain the meaningful data?

Motivation

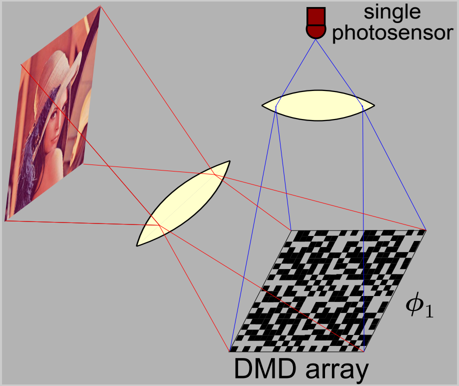
Compressed Sensing (CS)



- **Simultaneous sampling and dimension reduction**
- **Extremely low-cost encoding by reducing the size of sensor, memory, and computational unit**

Motivation

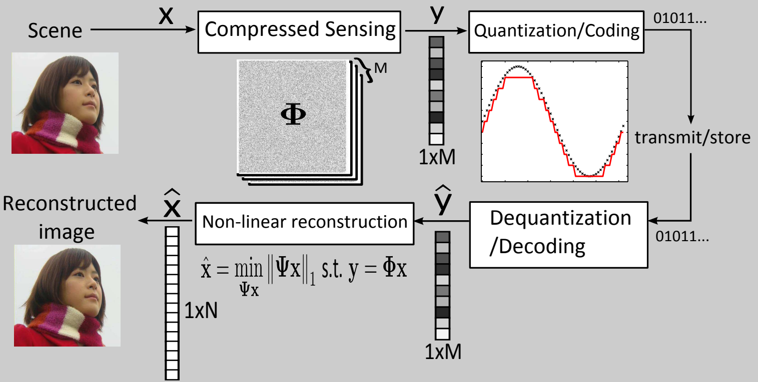
Nonadaptive Random Sampling in CS



- **Light integration can be represented as** $y_m = \langle \phi_m, \mathbf{x} \rangle, m \in \{1, 2, \dots, M\}$

Motivation

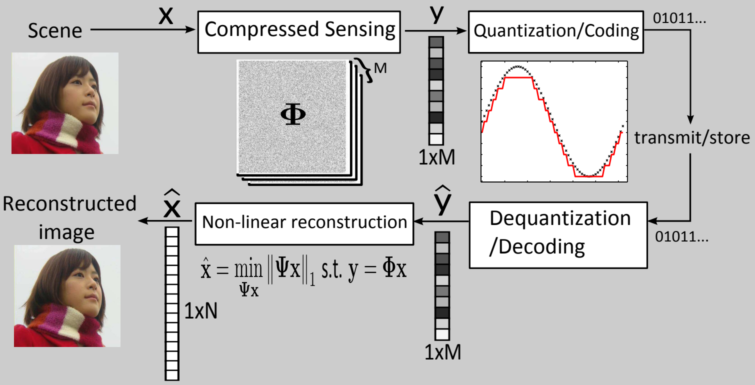
Challenges in CS



- **Huge sensing matrix, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$: $O(N^2)$**
- Complex recovery process: no explicit solution
- Quantization distortion: hinders reconstruction

Motivation

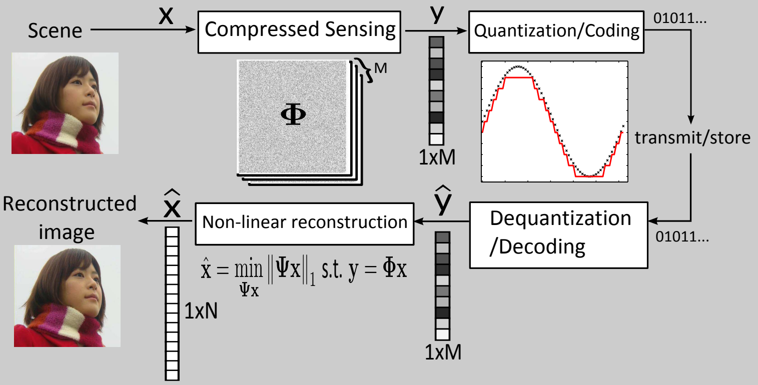
Challenges in CS



- **Huge sensing matrix, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$: $O(N^2)$**
- **Complex recovery process: no explicit solution**
- **Quantization distortion: hinders reconstruction**

Motivation

Challenges in CS



- **Huge sensing matrix, $\Phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$: $O(N^2)$**
- **Complex recovery process: no explicit solution**
- **Quantization distortion: hinders reconstruction**

Motivation

We want to solve these three challenges

1. huge sensing matrix
2. complex recovery process
3. quantization distortion

to make CS application (images and video) more realistic

Outline

CS Overview

Goal

Recover $\mathbf{x} \in \Re^N$ from

$$\mathbf{y} = \mathbf{\Phi x} \in \Re^M$$

- $\mathbf{\Phi}$: $M \times N$ measurement matrix, $M \ll N$
- Usually, $\mathbf{\Phi}$ is a random matrix (Gaussian, ± 1)

Fundamental Tenet of CS

If x is sufficiently sparse, recovery is exact from

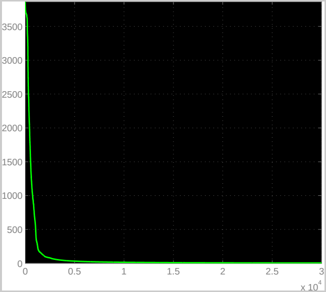
$$M \geq O(K \cdot logN)$$

measurements by solving tractable program

- K: number of nonzero coefficients in some transform Ψ
- Approximate recovery for nearly sparse

Images Nearly Sparse in Transform Domain

Wavelet transform



Sorted Coefficients

- K: **number of significant transform coefficients**

$$|\check{x}|_{(1)} \geq |\check{x}|_{(2)} \geq \cdots \geq |\check{x}|_{(K)} \geq \cdots \geq |\check{x}|_{(N)}$$

where $\check{x} = \sum_{i=1}^{N} x_i \psi_i$ **or** $\check{\mathbf{x}} = \mathbf{\Psi x}$ **in matrix form**

CS Overview

BCS of Images and Video

S. Mun

Practical Recovery: ℓ_1 optimization

$$\min_{\mathbf{x}} \|\mathbf{x}\|_{\ell_1} \ \textbf{subject to} \ \|\mathbf{y} - \Phi\mathbf{x}\|_{\ell_2} \leq \epsilon$$

- **No explicit solution due to non-differentiability of $\|\mathbf{x}\|_{\ell_1}$, but the solution can be found iteratively**
- **Many solvers exists:**
  - **Interior point method for 1D signal**
  - **Regularized total variation replacing $\|\mathbf{x}\|_{\ell_1}$ with $\|\nabla\mathbf{x}\|_{\ell_1}$ for image**
  - **Gradient projection (GPSR)**
  - **Matching pursuit replacing $\|\mathbf{y} - \Phi\mathbf{x}\|_{\ell_2}$ with $\|\mathbf{y} - \sum_{i=1}^{m} x_i\phi_i\|_{\ell_2}$**
  - **Projected Landweber (a.k.a iterative threshold)**

Outline

Solution for 1. Huge Measurement Matrix

Block Compressed Sensing (BCS)

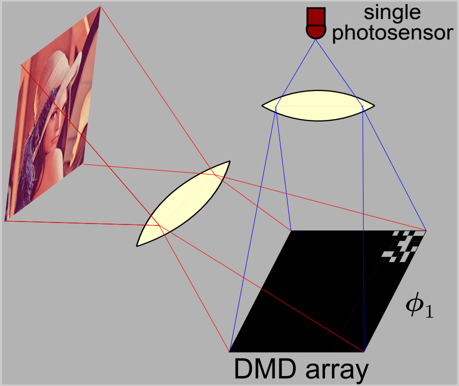
Image partitioned into small blocks ($B \times B$)

$$\mathbf{y}_j = \mathbf{\Phi}_B \mathbf{x}_j$$

$\mathbf{\Phi}_B$: $\lfloor \frac{M}{N} B^2 \rfloor \times B^2$, $\mathbf{x}_j$: block j of image

Solution for 1. Huge Measurement Matrix

Sampling Process in Block CS



- **The size of projection matrix Φ is drastically reduced by keeping only the small projection matrix**

Projected Landweber (PL)

$$\check{\check{\mathbf{x}}}^{(k)} = \begin{cases} \check{\mathbf{x}}^{(k)}, & \left|\check{\mathbf{x}}^{(k)}\right| \geq \tau^{(k)} \\ 0 & \textbf{else}, \end{cases} \quad (1)$$

$$\mathbf{x}^{(k+1)} = \bar{\mathbf{x}}^{(k)} + \frac{1}{\alpha}\boldsymbol{\Phi}^T\left(\mathbf{y} - \boldsymbol{\Phi}\bar{\mathbf{x}}^{(k)}\right) \quad (2)$$

α : **scaling factor,** $\alpha = 1$ **with orthonormal** $\boldsymbol{\Phi}$
$\boldsymbol{\Psi}, \boldsymbol{\Psi}^{-1}$: **transform and its inverse,** $\check{\mathbf{x}} = \boldsymbol{\Psi}\mathbf{x}$, $\bar{\mathbf{x}} = \boldsymbol{\Psi}^{-1}\check{\check{\mathbf{x}}}$
$\tau^{(k)}$: **threshold value for iteration** k

Smooth PL Reconstruction

Attempt to impose:

- Sparsity through (1)
- Consistency with observation space through (2)
- Smoothness through Wiener filter

Solution for 2. Complex Recovery

Projected Landweber (PL)

$$\check{\check{\mathbf{x}}}^{(k)} = \begin{cases} \check{\mathbf{x}}^{(k)}, & \left|\check{\mathbf{x}}^{(k)}\right| \geq \tau^{(k)} \\ 0 & \textbf{else}, \end{cases} \qquad (1)$$

$$\mathbf{x}^{(k+1)} = \bar{\mathbf{x}}^{(k)} + \frac{1}{\alpha}\mathbf{\Phi}^T\left(\mathbf{y} - \mathbf{\Phi}\bar{\mathbf{x}}^{(k)}\right) \qquad (2)$$

α : scaling factor, $\alpha = 1$ with orthonormal $\mathbf{\Phi}$
$\mathbf{\Psi}, \mathbf{\Psi}^{-1}$: transform and its inverse, $\check{\mathbf{x}} = \mathbf{\Psi}\mathbf{x}$, $\bar{\mathbf{x}} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}$
$\tau^{(k)}$: threshold value for iteration k

Smooth PL Reconstruction

Attempt to impose:

- Sparsity through (1)
- Consistency with observation space through (2)
- Smoothness through Wiener filter

Solution for 2. Complex Recovery

SPL Reconstruction Algorithm

function $\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$

$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$

for each block j

$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \hat{\mathbf{x}}_j^{(k)})$

$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi} \hat{\hat{\mathbf{x}}}^{(k)}$

$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$

$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1} \check{\check{\mathbf{x}}}^{(k)}$

for each block j

$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \bar{\mathbf{x}}_j^{(k)})$

$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$

endfunction

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \mathbf{\Phi}_B^T \mathbf{y}_j$

SPL Reconstruction Algorithm

function $\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$

$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$

for each block j

$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\hat{\mathbf{x}}_j^{(k)})$

$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi}\hat{\hat{\mathbf{x}}}^{(k)}$

$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$

$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}^{(k)}$

for each block j

$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\bar{\mathbf{x}}_j^{(k)})$

$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$

endfunction

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \mathbf{\Phi}_B^T\mathbf{y}_j$

Solution for 2. Complex Recovery

SPL Reconstruction Algorithm

```
function x^(k+1) = SPL(x^(k), y_j, Φ_B, Ψ, τ)
    x̂^(k) = Wiener(x^(k))
    for each block j
        x̂̂_j^(k) = x̂_j^(k) + Φ_B^T (y_j − Φ_B x̂_j^(k))
    x̌^(k) = Ψ x̂̂^(k)
    x̌̌^(k) = Threshold(x̌^(k), τ)
    x̄^(k) = Ψ^{-1} x̌̌^(k)
    for each block j
        x_j^(k+1) = x̄_j^(k) + Φ_B^T (y_j − Φ_B x̄_j^(k))
    D^(k+1) = ||x^(k+1) − x̂̂^(k)||_2
endfunction
```

$$\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$$
$$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$$
$$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \hat{\mathbf{x}}_j^{(k)})$$
$$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi}\hat{\hat{\mathbf{x}}}^{(k)}$$
$$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$$
$$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}^{(k)}$$
$$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \bar{\mathbf{x}}_j^{(k)})$$
$$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$$

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \mathbf{\Phi}_B^T \mathbf{y}_j$

Solution for 2. Complex Recovery

SPL Reconstruction Algorithm

function $\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$

$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$

for each block j

$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\hat{\mathbf{x}}_j^{(k)})$

$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi}\hat{\hat{\mathbf{x}}}^{(k)}$

$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$

$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}^{(k)}$

for each block j

$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\bar{\mathbf{x}}_j^{(k)})$

$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$

endfunction

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \mathbf{\Phi}_B^T\mathbf{y}_j$

Solution for 2. Complex Recovery

SPL Reconstruction Algorithm

function $\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$

$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$

for each block j

$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \hat{\mathbf{x}}_j^{(k)})$

$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi}\hat{\hat{\mathbf{x}}}^{(k)}$

$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$

$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}^{(k)}$

for each block j

$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B \bar{\mathbf{x}}_j^{(k)})$

$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$

endfunction

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \Phi_B^T \mathbf{y}_j$

Solution for 2. Complex Recovery

SPL Reconstruction Algorithm

function $\mathbf{x}^{(k+1)} = \mathrm{SPL}(\mathbf{x}^{(k)}, \mathbf{y}_j, \mathbf{\Phi}_B, \mathbf{\Psi}, \tau)$

$\hat{\mathbf{x}}^{(k)} = \mathrm{Wiener}(\mathbf{x}^{(k)})$

for each block j

$\hat{\hat{\mathbf{x}}}_j^{(k)} = \hat{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\hat{\mathbf{x}}_j^{(k)})$

$\check{\mathbf{x}}^{(k)} = \mathbf{\Psi}\hat{\hat{\mathbf{x}}}^{(k)}$

$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$

$\bar{\mathbf{x}}^{(k)} = \mathbf{\Psi}^{-1}\check{\check{\mathbf{x}}}^{(k)}$

for each block j

$\mathbf{x}_j^{(k+1)} = \bar{\mathbf{x}}_j^{(k)} + \mathbf{\Phi}_B^T(\mathbf{y}_j - \mathbf{\Phi}_B\bar{\mathbf{x}}_j^{(k)})$

$D^{(k+1)} = \|\mathbf{x}^{(k+1)} - \hat{\hat{\mathbf{x}}}^{(k)}\|_2$

endfunction

- Iterate until $|D^{(i)} - D^{(i-1)}| < 10^{-4}$
- Initialization: $\mathbf{x}_j^{(0)} = \mathbf{\Phi}_B^T\mathbf{y}_j$

Solution for 2. Complex Recovery

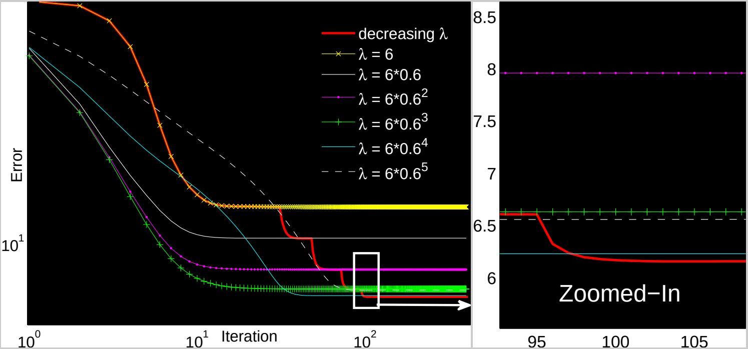
$$\check{\mathbf{x}}^{(k)} = \Psi \hat{\hat{\mathbf{x}}}^{(k)}$$
$$\check{\check{\mathbf{x}}}^{(k)} = \mathrm{Threshold}(\check{\mathbf{x}}^{(k)}, \tau)$$

Two issues of SPL

- A good transform function, Ψ
  - should prevent oscillations or shift variance due to thresholding
  - should preserve local structure (directional patterns)
- A good thresholding value, τ
  - should determine which coeff is important
  - should be adaptive to the signal at each iteration

Solution for 2. Complex Recovery

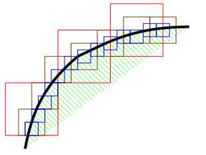
Possible Choice: Ψ = block DCT, $\tau = \lambda \sigma_n \sqrt{2 \log K}$



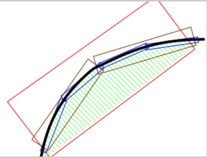
- Ψ: simple, but blocking artifacts
- τ: Convenient heuristic, but theoretical shortcomming

Solution for 2. Complex Recovery

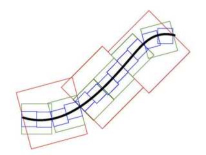
Better Choice for Ψ: Directional Transformation



DWT



Contourlet



Dual-tree DWT

- Contourlet Transform (Do, 2005)
  - Couples Laplacian-pyramid decomposition with directional filterbank
  - 2^d directional sub-bands, usually n=3,4
- Dual-tree DWT (Kingsbury, 2001)
  - keeps real & imaginary values in decomposition
  - 12 directional sub-bands, real 6 and imaginary 6

We choose Contourlet and Dual-tree DWT for Ψ

Solution for 2. Complex Recovery

Example: DDWT subbands of the real part coefficients

Solution for 2. Complex Recovery

Better Choice for τ: Wavelet coeffs' dependency

- **Bivariate Shrinkage (Şendur & Selesnick, 2002)**

  For each coefficient $\check{x}$ in $\check{\mathbf{x}}$

$$\check{\check{x}} = \text{Shrinkage}(z, \tau) \cdot \frac{\check{x}}{z}$$

where

$$\text{Shrinkage}(z, \tau) = \begin{cases} |z| - \tau, & |z| \geq \tau \\ 0 & \textbf{else}, \end{cases}$$

$$z = \sqrt{\check{x}^2 + \check{x}_p^2}, \tau = \lambda \frac{\sqrt{3}\sigma_n}{\sigma_{\check{x}}}$$

$\check{x}_p$: **coeff in parent level,** σ_n: **noise variance**

  - **Works well with multi-level decomposition**
  - **Takes into account statistical dependency**

We choose bivariate shrinkage for τ

BCS of Images and Video

S. Mun

Images
CS Overview
BCS-SPL
Results
MS-BCS-SPL
Results
Video
Quantization
Conclusion

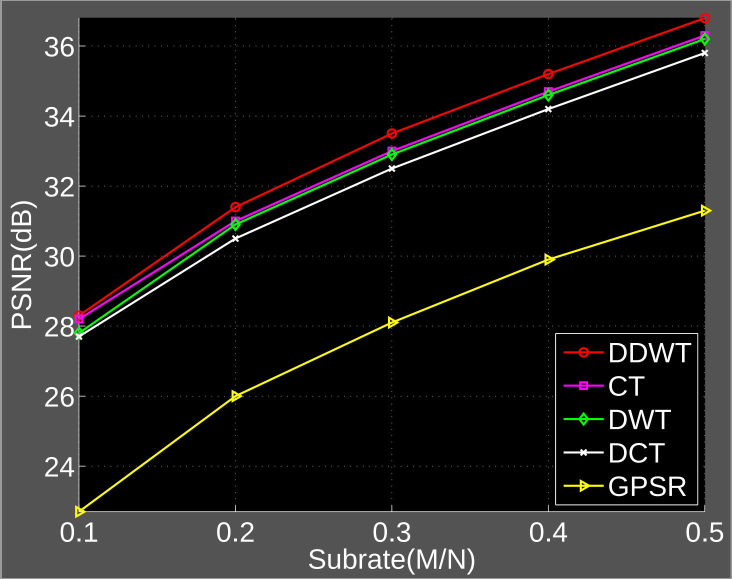
Experimental Setup

- Φ_B: orthonormalized Gaussian matrix, B = 32×32
- λ = 10(CT),25(DWT),25(DDWT),6(DCT)
- Sub-sampling ratio (subrate): $S = M/N$
- Distortion measure: Peak signal-to-noise ratio

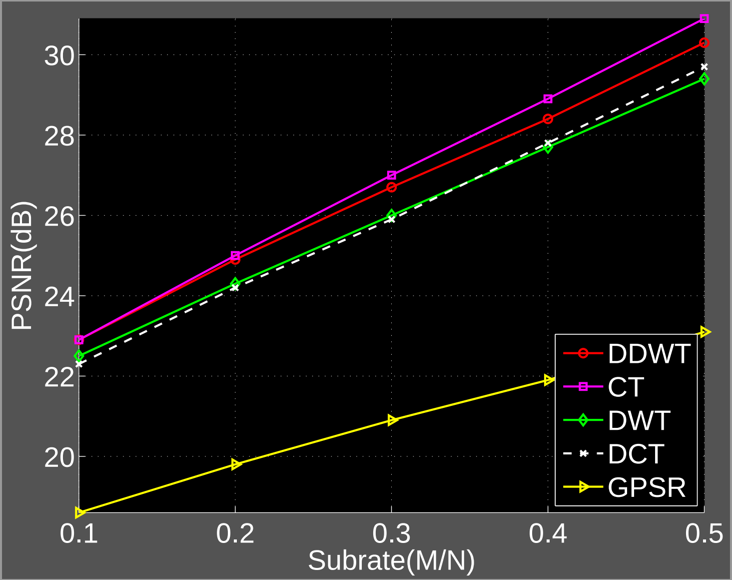
Algorithms Compared

- BCS-SPL using Directional Transform
  - Dual-tree DWT (DDWT) - bivariate shrinkage
  - Contourlet Transform (CT) - bivariate shrinkage
- BCS-SPL using DWT with bivariate shrinkage
- BCS-SPL using DCT with heuristic thresholding
- Gradient projection method in BCS as benchmark
  - GPSR(Schniter *et al*. 2008)

PSNR performance comparison, “Lenna”



PSNR performance comparison, “Mandrill”



Outline

Multiscale extention

Block-Based Compressed Sensing (BCS)

- BCS: CS sampling within small image blocks
- Advantage: very fast, low memory
- Drawback: reduced reconstruction quality comparing to global sampling
- Motivation: deploy BCS within multiscale framework

Multiscale Framework

- Random sampling in wavelet domain, not spatial domain
  - Exploiting wavelet structure enables to determine which subband is important than others
  - Mutiscale CS—Sampling rate adjusted with DWT level (baseband retained in full)

Multiscale BCS (MS-BCS)

Multiscale Sampling

Random sampling in multi-scale domain is defined

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{\Phi}'\mathbf{\Omega}\mathbf{x}$$

and multiscale coefficients are defined

$$\tilde{\mathbf{x}} = \mathbf{\Omega}\mathbf{x}$$

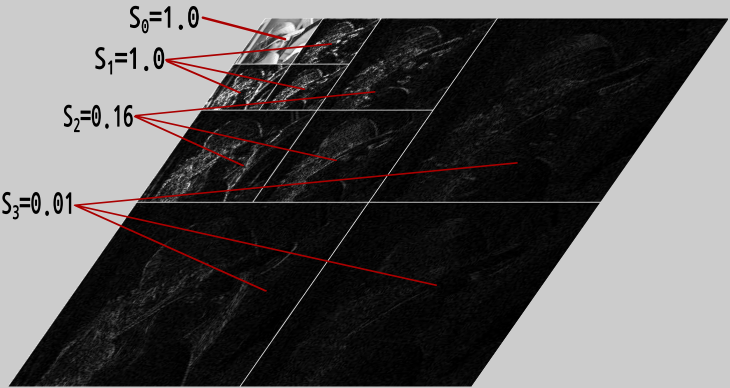
Then,

$$\mathbf{y} = \mathbf{\Phi}'\tilde{\mathbf{x}}$$

- Ω: multiscale transform (i.e., L-level DWT)
- Φ': multiscale block-based sampling operator
  - set of BCS sampling operators with subrates varying with level
  - $\Phi' = \{\Phi_l\}$, $1 \le l \le L$

Multiscale Sampling Example, $\mathbf{y} = \mathbf{\Phi}'\tilde{\mathbf{x}}$

Target subrate $S = 0.1$ with $L = 3$



$$S = \frac{1}{4^3} \times S_0 + \frac{3}{4^3} \times S_1 + \frac{3}{4^{3-2+1}} \times S_2 + \frac{3}{4^{3-3+1}} \times S_3 = 0.1$$

Multiscale BCS Reconstruction

MS-BCS-SPL algorithm

function $\tilde{\mathbf{x}}^{(i)} = \text{MS-BCS-SPL}\left(\mathbf{y}, \{\boldsymbol{\Phi}_{l,\ 1\le l\le L}\}, \boldsymbol{\Psi}, \boldsymbol{\Omega}, \tilde{\mathbf{x}}^{(0)}_{l,s,j}\right)$

do

$$
\begin{aligned}
\mathbf{x}^{(i)} &= \boldsymbol{\Omega}^{-1}\tilde{\mathbf{x}}^{(i)} \\
\hat{\mathbf{x}}^{(i)} &= \text{Wiener}(\mathbf{x}^{(i)}) \\
\hat{\hat{\mathbf{x}}}^{(i)} &= \boldsymbol{\Omega}\hat{\mathbf{x}}^{(i)}
\end{aligned}
$$

for each l, **for each** s, **for each** j

$$\hat{\hat{\hat{\mathbf{x}}}}^{(i)}_{l,s,j} = \hat{\hat{\mathbf{x}}}^{(i)}_{l,s,j} + \boldsymbol{\Phi}_l^T\left(\mathbf{y}_{l,s,j} - \boldsymbol{\Phi}_l\hat{\hat{\mathbf{x}}}^{(i)}_{l,s,j}\right)$$

$$
\begin{aligned}
\check{\check{\mathbf{x}}}^{(i)} &= \boldsymbol{\Psi}\boldsymbol{\Omega}^{-1}\hat{\hat{\hat{\mathbf{x}}}}^{(i)} \\
\check{\mathbf{x}}^{(i)} &= \text{Threshold}(\check{\check{\mathbf{x}}}^{(i)}) \\
\bar{\bar{\mathbf{x}}}^{(i)} &= \boldsymbol{\Omega}\boldsymbol{\Psi}^{-1}\check{\mathbf{x}}^{(i)}
\end{aligned}
$$

for each l, **for each** s, **for each** j

$$\tilde{\mathbf{x}}^{(i+1)}_{l,s,j} = \bar{\bar{\mathbf{x}}}^{(i)}_{l,s,j} + \boldsymbol{\Phi}_l^T\left(\mathbf{y}_{l,s,j} - \boldsymbol{\Phi}_l\bar{\bar{\mathbf{x}}}^{(i)}_{l,s,j}\right)$$

$$D^{(i+1)} = \|\tilde{\mathbf{x}}^{(i+1)} - \hat{\hat{\hat{\mathbf{x}}}}^{(i)}\|_2$$

$$i = i + 1$$

until $|D^{(i)} - D^{(i-1)}| < 10^{-2}$

Experimental Setup

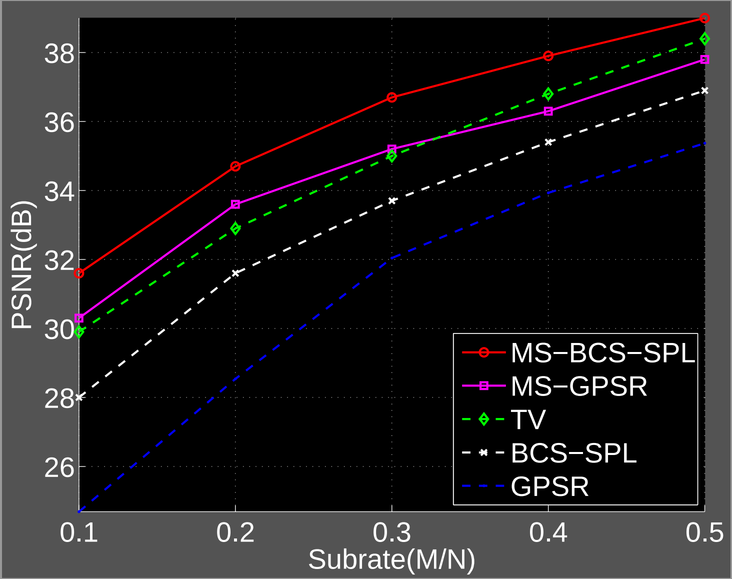
Experiment Setup

- Sparsity basis, Ψ: DDWT
- Sampling basis, Ω: 3-level 9/7 biorthogonal DWT
- $B_l \times B_l$ blocks sampled using structured random matrices (SRM)
- Block size $B_1 = 16$, $B_2 = 32$, $B_3 = 64$

Algorithms Compared

- BCS-SPL
- TV (Candès *et al*. 2006)
- GPSR (Schniter *et al*. 2008)
- MS-GPSR multiscale version of GPSR

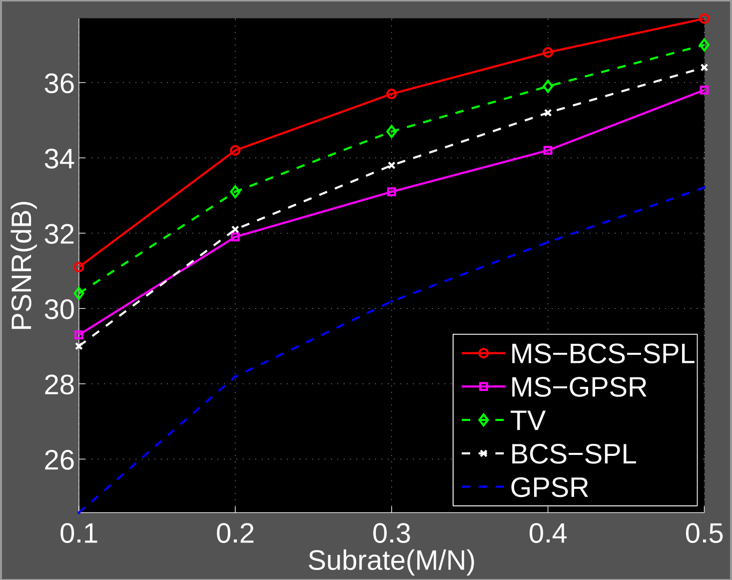
Performance Comparison, “Lenna”



Performance Comparison, “Peppers”

BCS of Images and Video

S. Mun



Simulation Results for 2D Images

Lenna (512×512) with subrate $S = 0.1$

MS-BCS-SPL
31.6 dB
50 seconds

BCS-SPL
28.0 dB
30 seconds

Simulation Results for 2D Images

Lenna (512×512) with subrate $S = 0.1$

MS-GPSR
30.3 dB
20 minutes

TV
29.9 dB
1.8 hours

BCS-SPL of images

Remarks

- BCS-SPL simplified measurement process and sped up the reconstruction process
- Multiscale random sampling significantly enhanced the reconstruction quality

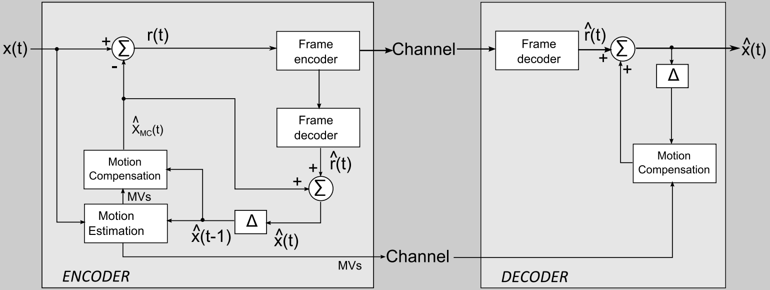
Publications

- S. Mun and J. E. Fowler, "Block Compressed Sensing of Images Using Directional Transforms," in Proceedings of the International Conference on Image Processing, Cairo, Egypt, November 2009, pp. 3021-3024.
- J. E. Fowler, S. Mun, and E. W. Tramel, "Multiscale Block Compressed Sensing with Smoothed Projected Landweber Reconstruction," in Proceedings of the European Signal Processing Conference, Barcelona, Spain, August 2011, pp. 564-568.

Outline

Traditional Video Compression

Traditional Video Compression



- Temporal motion compensation at encoding making encoder complex
- Well suited for once encode, decode many times
- Not the best solution for resource-limited encoding

Extension of Still-image CS to Video

BCS for multiple frames

- BCS can provide extremely simple video sampler.
- Unlike traditional video coder, motion cannot be tracked in sampling stage
  - Random sampling generates measurements, not images
- Instead, the motion can be tracked at receiver side

Experimental Assumption

- Same 2D BCS sampler used for images
- Adjustable sampling rate for certain frames(key frames)

We want to find a good CS video recovery while keeping the sampler simple

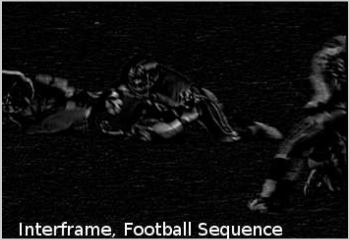
Outline

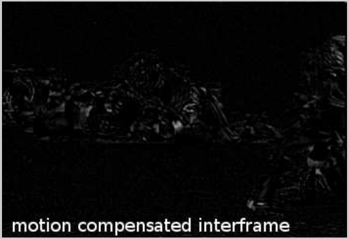
Motion Compensated Reconstruction

BCS of Images and Video

S. Mun

Residual Reconstruction





- **Using motion estimation/compensation (ME/MC) to capture objects motion in frames**
- **Obtaining correct motion vectors (MVs) are crucial**
  - **using iterative-refinement of MVs**
  - **using sparser representation in ME/MC residual**
- **Residual is *random projected* motion compensated interframe**

MC-BCS-SPL algorithm for single frame

function $\hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \Phi_B, \Psi, \mathbf{x}_{\textsf{ref}})$

$\hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \Phi_B, \Psi, \mathbf{x}_{\textsf{ref}})$

$i = 0$

while $i <$ **MAX_ITERATION**

$\hat{\mathbf{x}}_{\textsf{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\textsf{ref}})$

for each block j

$\mathbf{y}_{\textsf{mc}_j} = \Phi_B \hat{\mathbf{x}}_{\textsf{mc}_j}$

$\mathbf{y}_{\textsf{r}} = \mathbf{y} - \mathbf{y}_{\textsf{mc}}$

$\hat{\mathbf{x}}_{\textsf{r}} = \text{BCS-SPL}(\mathbf{y}_{\textsf{r}}, \Phi_B, \Psi)$

$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\textsf{mc}} + \hat{\mathbf{x}}_{\textsf{r}}$

$i = i + 1$

end while

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

```
function x̂ = MC-BCS-SPL(y, Φ_B, Ψ, x_ref)
    x̂ = Initialize(y, Φ_B, Ψ, x_ref)
    i = 0
    while i < MAX_ITERATION
        x̂_mc = MotionCompensation(x̂, x_ref)
        for each block j
            y_mc_j = Φ_B x̂_mc_j
        y_r = y − y_mc
        x̂_r = BCS-SPL(y_r, Φ_B, Ψ)
        x̂ = x̂_mc + x̂_r
        i = i + 1
    end while
```

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

function $\hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \boldsymbol{\Phi}_B, \boldsymbol{\Psi}, \mathbf{x}_{\text{ref}})$

$\hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \boldsymbol{\Phi}_B, \boldsymbol{\Psi}, \mathbf{x}_{\text{ref}})$

$i = 0$

while $i <$ **MAX_ITERATION**

$\hat{\mathbf{x}}_{\text{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\text{ref}})$

for each block j

$\mathbf{y}_{\text{mc}_j} = \boldsymbol{\Phi}_B \hat{\mathbf{x}}_{\text{mc}_j}$

$\mathbf{y}_{\text{r}} = \mathbf{y} - \mathbf{y}_{\text{mc}}$

$\hat{\mathbf{x}}_{\text{r}} = \text{BCS-SPL}(\mathbf{y}_{\text{r}}, \boldsymbol{\Phi}_B, \boldsymbol{\Psi})$

$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\text{mc}} + \hat{\mathbf{x}}_{\text{r}}$

$i = i + 1$

end while

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

function $\hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}})$

$\hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}})$

$i = 0$

while $i <$ **MAX_ITERATION**

$\hat{\mathbf{x}}_{\mathsf{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\mathsf{ref}})$

for each block j

$\mathbf{y}_{\mathsf{mc}_j} = \mathbf{\Phi}_B \hat{\mathbf{x}}_{\mathsf{mc}_j}$

$\mathbf{y}_{\mathsf{r}} = \mathbf{y} - \mathbf{y}_{\mathsf{mc}}$

$\hat{\mathbf{x}}_{\mathsf{r}} = \text{BCS-SPL}(\mathbf{y}_{\mathsf{r}}, \mathbf{\Phi}_B, \mathbf{\Psi})$

$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\mathsf{mc}} + \hat{\mathbf{x}}_{\mathsf{r}}$

$i = i + 1$

end while

MC-BCS-SPL algorithm for single frame

function $\hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \Phi_B, \Psi, \mathbf{x}_{\mathbf{ref}})$

$\hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \Phi_B, \Psi, \mathbf{x}_{\mathbf{ref}})$

$i = 0$

while $i <$ **MAX_ITERATION**

$\hat{\mathbf{x}}_{\mathbf{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\mathbf{ref}})$

for each block j

$\mathbf{y}_{\mathbf{mc}_j} = \Phi_B \hat{\mathbf{x}}_{\mathbf{mc}_j}$

$\mathbf{y}_{\mathbf{r}} = \mathbf{y} - \mathbf{y}_{\mathbf{mc}}$

$\hat{\mathbf{x}}_{\mathbf{r}} = \text{BCS-SPL}(\mathbf{y}_{\mathbf{r}}, \Phi_B, \Psi)$

$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\mathbf{mc}} + \hat{\mathbf{x}}_{\mathbf{r}}$

$i = i + 1$

end while

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

```
function x̂ = MC-BCS-SPL(y, Φ_B, Ψ, x_ref)
    x̂ = Initialize(y, Φ_B, Ψ, x_ref)
    i = 0
    while i < MAX_ITERATION
        x̂_mc = MotionCompensation(x̂, x_ref)
        for each block j
            y_mc_j = Φ_B x̂_mc_j
        y_r = y − y_mc
        x̂_r = BCS-SPL(y_r, Φ_B, Ψ)
        x̂ = x̂_mc + x̂_r
        i = i + 1
    end while
```

$\texttt{function}\ \hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\textsf{ref}})$

$\quad \hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\textsf{ref}})$

$\quad i = 0$

$\quad$ **while** $i <$ **MAX_ITERATION**

$\qquad \hat{\mathbf{x}}_{\textsf{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\textsf{ref}})$

$\qquad \texttt{for each block}\ j$

$\qquad\quad \mathbf{y}_{\textsf{mc}_j} = \mathbf{\Phi}_B \hat{\mathbf{x}}_{\textsf{mc}_j}$

$\qquad \mathbf{y}_{\textsf{r}} = \mathbf{y} - \mathbf{y}_{\textsf{mc}}$

$\qquad \hat{\mathbf{x}}_{\textsf{r}} = \text{BCS-SPL}(\mathbf{y}_{\textsf{r}}, \mathbf{\Phi}_B, \mathbf{\Psi})$

$\qquad \hat{\mathbf{x}} = \hat{\mathbf{x}}_{\textsf{mc}} + \hat{\mathbf{x}}_{\textsf{r}}$

$\qquad i = i + 1$

$\quad$ **end while**

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

```
function x̂ = MC-BCS-SPL(y, Φ_B, Ψ, x_ref)
```

$$
\begin{aligned}
&\textbf{function}\ \hat{\mathbf{x}} = \text{MC-BCS-SPL}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}}) \\
&\quad \hat{\mathbf{x}} = \text{Initialize}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}}) \\
&\quad i = 0 \\
&\quad \textbf{while}\ i < \textbf{MAX\_ITERATION} \\
&\qquad \hat{\mathbf{x}}_{\mathsf{mc}} = \text{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\mathsf{ref}}) \\
&\qquad \textbf{for each block}\ j \\
&\qquad\quad \mathbf{y}_{\mathsf{mc}_j} = \mathbf{\Phi}_B \hat{\mathbf{x}}_{\mathsf{mc}_j} \\
&\qquad \mathbf{y}_{\mathsf{r}} = \mathbf{y} - \mathbf{y}_{\mathsf{mc}} \\
&\qquad \hat{\mathbf{x}}_{\mathsf{r}} = \text{BCS-SPL}(\mathbf{y}_{\mathsf{r}}, \mathbf{\Phi}_B, \mathbf{\Psi}) \\
&\qquad \hat{\mathbf{x}} = \hat{\mathbf{x}}_{\mathsf{mc}} + \hat{\mathbf{x}}_{\mathsf{r}} \\
&\qquad i = i + 1 \\
&\quad \textbf{end while}
\end{aligned}
$$

Motion Compensated Reconstruction

MC-BCS-SPL algorithm for single frame

function $\hat{\mathbf{x}} = \mathrm{MC\text{-}BCS\text{-}SPL}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}})$

$\hat{\mathbf{x}} = \mathrm{Initialize}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathsf{ref}})$

$i = 0$

while $i <$ **MAX_ITERATION**

$\hat{\mathbf{x}}_{\mathsf{mc}} = \mathrm{MotionCompensation}(\hat{\mathbf{x}}, \mathbf{x}_{\mathsf{ref}})$

for each block j

$\mathbf{y}_{\mathsf{mc}_j} = \mathbf{\Phi}_B \hat{\mathbf{x}}_{\mathsf{mc}_j}$

$\mathbf{y}_{\mathsf{r}} = \mathbf{y} - \mathbf{y}_{\mathsf{mc}}$

$\hat{\mathbf{x}}_{\mathsf{r}} = \mathrm{BCS\text{-}SPL}(\mathbf{y}_{\mathsf{r}}, \mathbf{\Phi}_B, \mathbf{\Psi})$

$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\mathsf{mc}} + \hat{\mathbf{x}}_{\mathsf{r}}$

$i = i + 1$

end while

Residual $\mathbf{y}_{\mathbf{r}}$ can be sparser in some transform than $\mathbf{y}$

Multi-hypothesis Initialization

Obtaining Correct (MVs) are Crucial

- Better Initialization, more correct MVs possible
- To have better initial reconstruction, averages two possible reconstructions
  - 2D-BCS-SPL: good for dynamic sequence
  - Residual reconstruction with no motion vector: good for stationary sequence

Algorithm

function $\hat{\mathbf{x}} = \mathrm{Initialize}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi}, \mathbf{x}_{\mathbf{ref}})$

$\hat{\mathbf{x}}' = \text{BCS-SPL}(\mathbf{y}, \mathbf{\Phi}_B, \mathbf{\Psi})$

for each block j

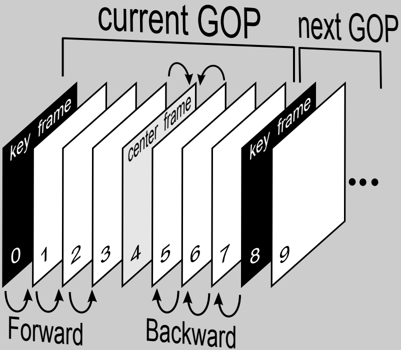
$\mathbf{y}_{\mathbf{r}_j} = \mathbf{y}_j - \mathbf{\Phi}_B \mathbf{x}_{\mathbf{ref}_j}$

$\hat{\mathbf{x}}'' = \text{BCS-SPL}(\mathbf{y}_{\mathbf{r}}, \mathbf{\Phi}_B, \mathbf{\Psi}) + \mathbf{x}_{\mathbf{ref}}$

$\hat{\mathbf{x}} = \frac{1}{2}\left[\hat{\mathbf{x}}' + \hat{\mathbf{x}}''\right]$

Multiple-Frame Processing

Forward/Backward Processing



- Key frames could have better quality than non-key frames by sampling more
- Forward processing for first half of the GOP; backward processing for last half of the GOP

Outline

Comparison to non-MC Algorithms

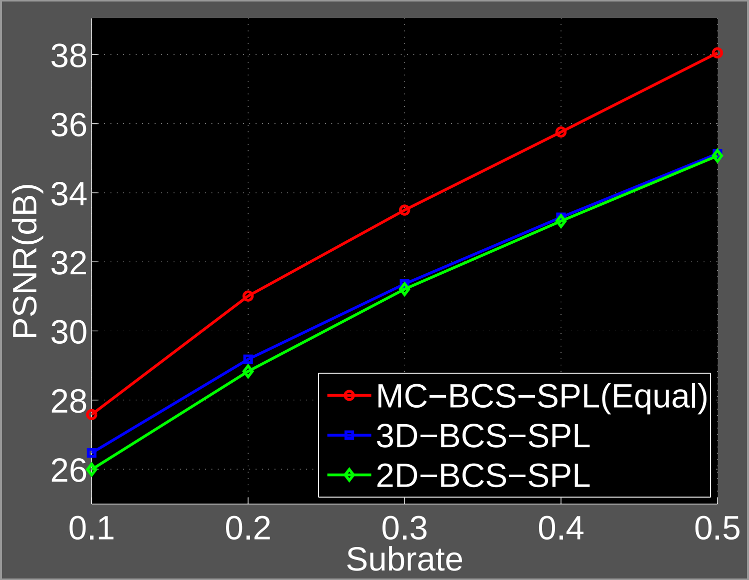
Experiment Setup

- Using DCT as sparsity basis for simplicity
- Subrate-distortion performance is observed
- GOP size: 8 frames (1st: key frame, Remainder: non-key)
- Test 1: All frames are equally subsampled
- Test 2: Key frames are sampled with higher subrate
  - Subrates are adjusted for non-key frame by the amount of measurements taken by key frames

Algorithms Compared

- Frame-by-frame reconstruction: 2D-BCS-SPL
- 3D joint reconstruction (Wakin et al. 2006): 3D-BCS-SPL

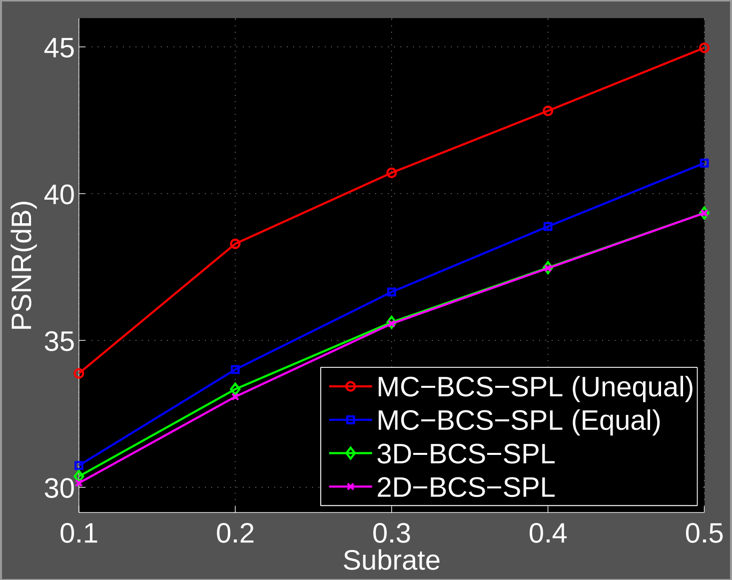
Test 1: $S_K = S_{NK}$, Foreman, 296 frames



Test 2: $S_K \neq S_K$, Mother-Daughter,296frames

BCS of Images and Video

S. Mun



Comparison to Other MC algorithms

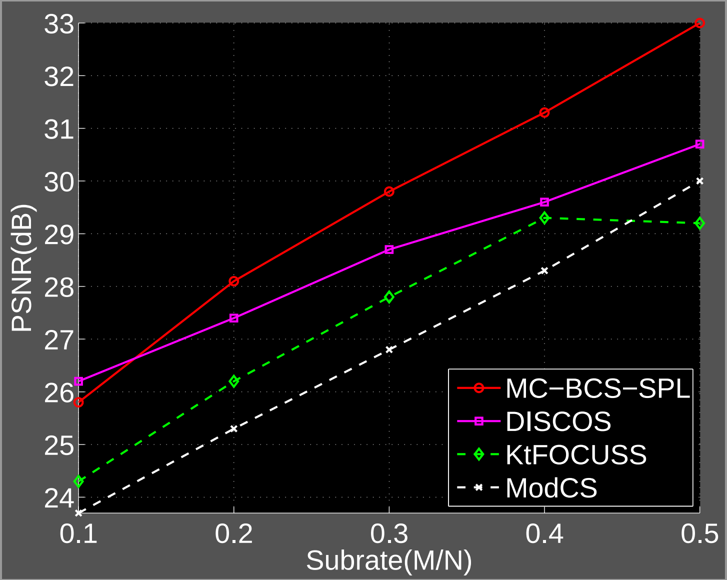
Experimental Setup

- Φ for MC-BCS-SPL: DDWT
- GOP size: 8 frames (1st: Key, 2-8th: non-key)
- Uneqaul subrate, $S_K = 0.7$ fixed, $S_{NK} = 0.1$ to 0.5

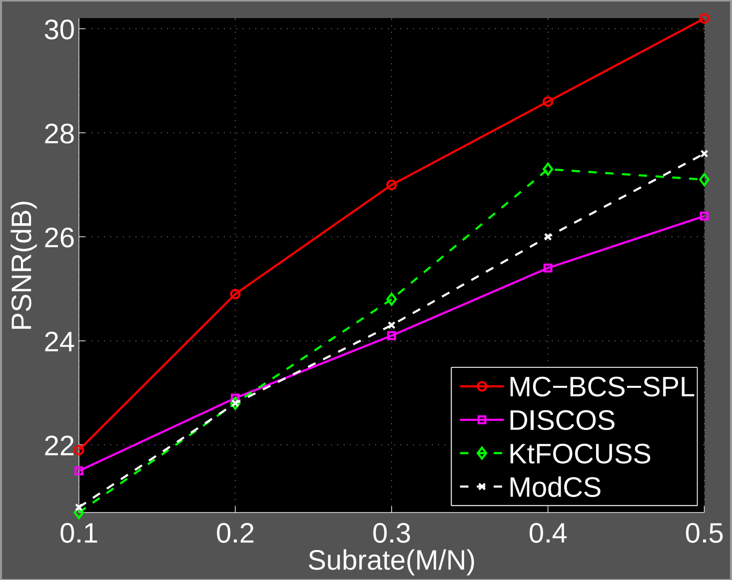
Algorithms Compared

- DISCOS (Do et al, 2009): Find the best linear combination of multiple matching blocks from the two key frames nearby
- k-t FOCUSS (Jung et al, 2009, 2010): Similar residual reconstruction with FOCUSS or IRLS. Reference frames: average of all initial reconstructions
- ModCS (Vaswani & Lu, 2010): Using previous frame's wavelet coefficients support info, estimate successive frames

Comparison to Other MC algorithms, Coastguard, 88 frames



Comparison to Other MC algorithms, Football, 88 frames



Visual Comparison of Reconstructions

4th frame of 'Foreman', $S_K = 0.7$, $S_{NK} = 0.3$

MC-BCS-SPL
36.7 dB
159 seconds

DISCOS
34.0 dB
41 seconds

Visual Comparison of Reconstructions

4th frame of 'Foreman', $S_K = 0.7$, $S_{NK} = 0.3$

k-t FOCUSS
32.7 dB
46 seconds

ModCS
29.6 dB
699 seconds

Remarks

Motion Compensated BCS-SPL

- ME/MC in residual reconstruction helps track the motion along the sequences at at receiver side
- Multi-hypothesis initialization provides a better estimation of the reference frame
- Forward/backward processing for multiple frames

Publications

- S. Mun and J. E. Fowler, "Residual Reconstruction for Block-Based Compressed Sensing of Video," in Proceedings of the IEEE Data Compression Conference, J. A. Storer and M. W. Marcellin, Eds., Snowbird, UT, March 2011, pp. 183-192.
- J. E. Fowler, S. Mun and E. W. Tramel "Block-Based Compressed Sensing of Images and Video", Foundations and Trends in Signal Processing: Vol. 4: No 4, pp 297-416, 2012.

Outline

Existing Approaches

Straightforward Approach

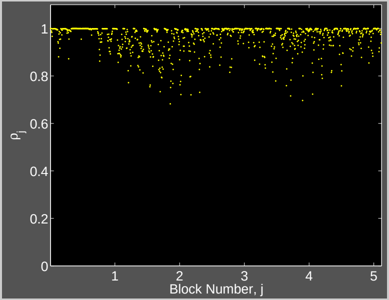
- Scalar Quantization (SQ): Simple, but inefficient

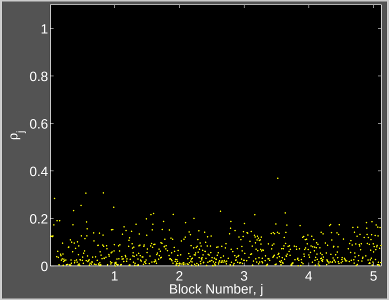
Alternative Approaches

- BPDQ(L. Jacques, 2011): seeks a sparse solution from scalar quantized measurements by adding particular data-fidelity constraint to enhance the reconstruction quality
- Progressive Quantization(L. Wang et al, 2011,2012): reconstructs a small set of fine quantized measurements to progressively estimate a large set of course quantized measurements

We want to use some statistical correlation in block CS

Correlation in CS and BCS, Test Image - "Lenna"





block correlation in BCS — $\bar{\rho}$ **= 0.971**

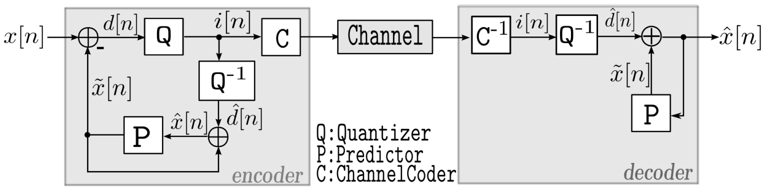
in CS — $\bar{\rho}$ **= 0.026**

- **measurements are divided into several groups to obtain the correlation using** $\rho_j = \dfrac{\mathbf{y}^{(j)\,T}\mathbf{y}^{(j-1)}}{\left\|\mathbf{y}^{(j)}\right\|\left\|\mathbf{y}^{(j-1)}\right\|}$
- $\bar{\rho}$**: average correlation**

Outline

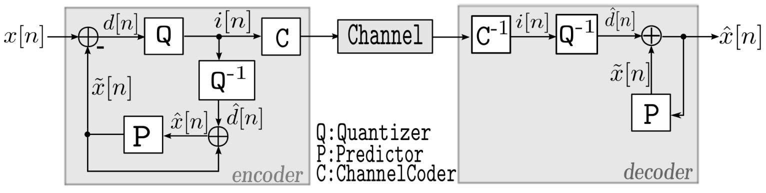
Differential Pulse Code Modulation(DPCM)

DPCM in Image Coding



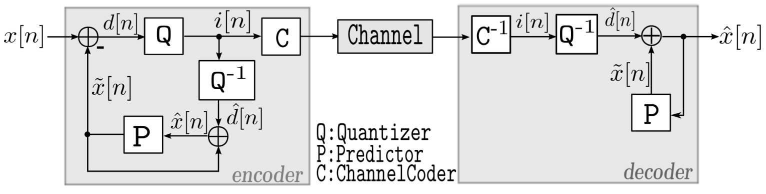
- Transmitting the pixel difference rather than pixel
- Works when signals possess a significant degree of correlation, and image does
- Predicted pixel is obtained $\tilde{x}[n] = \sum_{i=1}^{k} a_i x[n-i]$
- If 1st order prediction ($k = 1$ and $a_1 = 1.0$), the predicted pixel is simply the previously pixel

DPCM in Image Coding



- **Transmitting the pixel difference rather than pixel**
- **Works when signals possess a significant degree of correlation, and image does**
- **Predicted pixel is obtained** $\tilde{x}[n] = \sum_{i=1}^{k} a_i x[n-i]$
- **If 1st order prediction (**$k = 1$ **and** $a_1 = 1.0$**), the predicted pixel is simply the previously pixel**

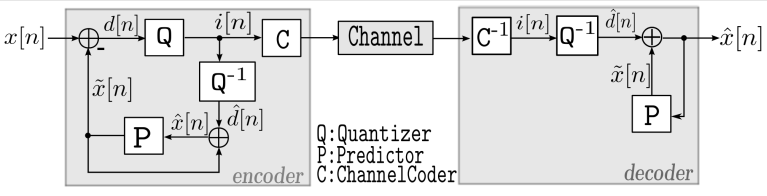
DPCM in Image Coding



- **Goal is to minimize the difference**
  $\sigma_d^2 = E\left[d[n]^2\right] = E\left[(x[n] - \tilde{x}[n])^2\right]$
- If 1st order predictor, it becomes $\sigma_d^2 = \sigma_x^2(1 - \rho_1^2)$
- Meaning the variance of the difference is reduced by the factor of $(1 - \rho_1^2)$
- Typical grayscale images having $\rho_1 \approx 0.95$ shows the rate gain 1-3 bpp over SQ

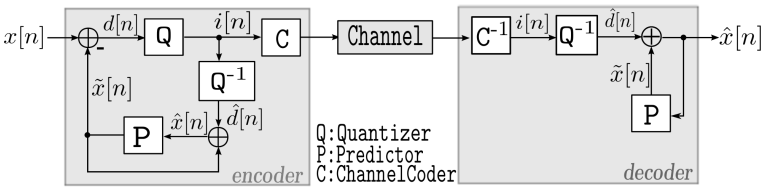
Differential Pulse Code Modulation(DPCM)

DPCM in Image Coding



- **Goal is to minimize the difference**
  $\sigma_d^2 = E\left[d[n]^2\right] = E\left[(x[n] - \tilde{x}[n])^2\right]$
- **If 1st order predictor, it becomes** $\sigma_d^2 = \sigma_x^2(1 - \rho_1^2)$
- **Meaning the variance of the difference is reduced by the factor of** $(1 - \rho_1^2)$
- **Typical grayscale images having** $\rho_1 \approx 0.95$ **shows the rate gain 1-3 bpp over SQ**

DPCM in Image Coding



- **Goal is to minimize the difference** $\sigma_d^2 = E\left[d[n]^2\right] = E\left[(x[n] - \tilde{x}[n])^2\right]$
- **If 1st order predictor, it becomes** $\sigma_d^2 = \sigma_x^2(1 - \rho_1^2)$
- **Meaning the variance of the difference is reduced by the factor of** $(1 - \rho_1^2)$
- **Typical grayscale images having** $\rho_1 \approx 0.95$ **shows the rate gain 1-3 bpp over SQ**

Outline

DPCM procedure in BCS

Given M-dimensional BCS measurement vector

$$\mathbf{y}^{(j)} = \left[y_1^{(j)} \quad \cdots \quad y_m^{(j)} \quad \cdots \quad y_{M_B}^{(j)} \right]^T = \mathbf{\Phi}_B \mathbf{x}^{(j)}$$

j: **index of block,** m: **measurement vector component**

To predict $y_m^{(j)}$, m^{th} **measurement of the previous vector is used.**

$$d_m^{(j)} = y_m^{(j)} - \hat{y}_m^{(j-1)}$$

Next, the residual, $d_m^{(j)}$, **is scalar quantized.**

$$i_m^{(j)} = Q\left[d_m^{(j)} \right]$$

Index $i_m^{(j)}$ **is then entropy coded.**

DPCM procedure in BCS

Given M-dimensional BCS measurement vector

$$\mathbf{y}^{(j)} = \left[y_1^{(j)} \quad \cdots \quad y_m^{(j)} \quad \cdots \quad y_{M_B}^{(j)} \right]^T = \mathbf{\Phi}_B \mathbf{x}^{(j)}$$

j: index of block, m: measurement vector component

To predict $y_m^{(j)}$, m^{th} measurement of the previous vector is used.

$$d_m^{(j)} = y_m^{(j)} - \hat{y}_m^{(j-1)}$$

Next, the residual, $d_m^{(j)}$, is scalar quantized.

$$i_m^{(j)} = Q\left[d_m^{(j)} \right]$$

Index $i_m^{(j)}$ is then entropy coded.

DPCM procedure in BCS

Given M-dimensional BCS measurement vector

$$\mathbf{y}^{(j)} = \left[y_1^{(j)} \quad \cdots \quad y_m^{(j)} \quad \cdots \quad y_{M_B}^{(j)}\right]^T = \mathbf{\Phi}_B \mathbf{x}^{(j)}$$

j**: index of block,** m**: measurement vector component**

To predict $y_m^{(j)}$**,** m^{th} **measurement of the previous vector is used.**

$$d_m^{(j)} = y_m^{(j)} - \hat{y}_m^{(j-1)}$$

Next, the residual, $d_m^{(j)}$**, is scalar quantized.**

$$i_m^{(j)} = Q\left[d_m^{(j)}\right]$$

Index $i_m^{(j)}$ **is then entropy coded.**

DPCM procedure in BCS

Feedback loop consists of dequantization of $i_m^{(j)}$, producing the quantized residual $\hat{d}_m^{(j)}$,

$$\hat{d}_m^{(j)} = Q^{-1}\left[i_m^{(j)}\right]$$

such that

$$\hat{y}_m^{(j)} = \hat{d}_m^{(j)} + \hat{y}_m^{(j-1)}.$$

Finally, the prediction is implemented with a one-block delay buffer.

- **The set of measurements in the first block is processed in the same manner by initializing $\hat{y}^{(0)}$ to be the zero vector.**

DPCM for quantized BCS

DPCM procedure in BCS

Feedback loop consists of dequantization of $i_m^{(j)}$, producing the quantized residual $\hat{d}_m^{(j)}$,

$$\hat{d}_m^{(j)} = Q^{-1}\left[i_m^{(j)}\right]$$

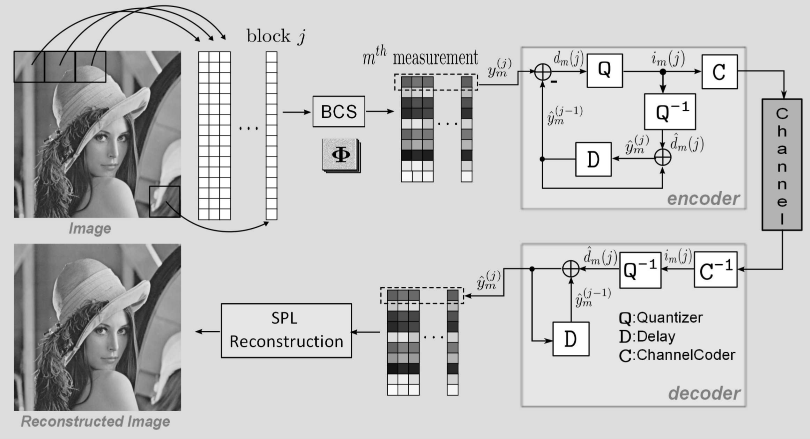
such that

$$\hat{y}_m^{(j)} = \hat{d}_m^{(j)} + \hat{y}_m^{(j-1)}.$$

Finally, the prediction is implemented with a one-block delay buffer.

- **The set of measurements in the first block is processed in the same manner by initializing $\hat{y}^{(0)}$ to be the zero vector.**

Application of DPCM to BCS-SPL

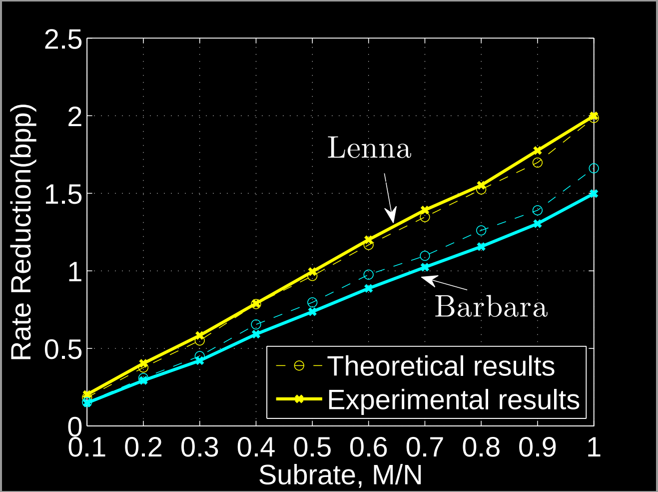


- **Proposed Algorithm: DPCM + SQ to BCS-SPL**

Rate Reduction using DPCM+SQ

BCS of Images and Video

S. Mun



$$\Delta R = \frac{1}{2}\frac{M}{N}\log_2\frac{1}{(1-\rho_1^2)}$$

Performance Comparison to SQ-alone

Experimental Setup

- RD performance measured by PSNR and bitrate
- Optimal stepsize and subrate are chosen through exhaustive search
- Φ_B: Orthonormal Gaussian, Ψ: 5-level DDWT
- $B = 8$ for BCS-SPL and MH-BCS-SPL, $B = 2$ for MS-BCS-SPL
- all SQ is uniform quantization

Algorithms Compared

- BCS-SPL
- MS-BCS-SPL
- MH-BCS-SPL(Chen et al, 2011) - extension using multi-hypothesis predictions

Experimental Results (BCS-SPL)

PSNR Performance in dB for a bitrate of 0.5 bpp

	BCS-SPL		
Image	SQ	DPCM	*Gain*
Lenna	27.7	29.4	+1.7
Barbara	22.9	23.6	+0.7
Peppers	28.6	29.5	+0.9
Goldhill	26.7	27.4	+0.7
Man	26.2	26.9	+0.7
Clown	26.7	27.6	+0.9
Average	26.5	27.4	+0.9

Experimental Results (MH-BCS-SPL)

PSNR Performance in dB for a bitrate of 0.5 bpp

	MH-BCS-SPL		
Image	SQ	DPCM	*Gain*
Lenna	29.2	31.4	+2.3
Barbara	24.4	27.9	+3.5
Peppers	29.2	31.2	+2.1
Goldhill	26.8	28.8	+2
Man	26.5	27.9	+1.4
Clown	28.4	30.8	+2.4
Average	27.4	29.7	+2.3

Experimental Results (MS-BCS-SPL)

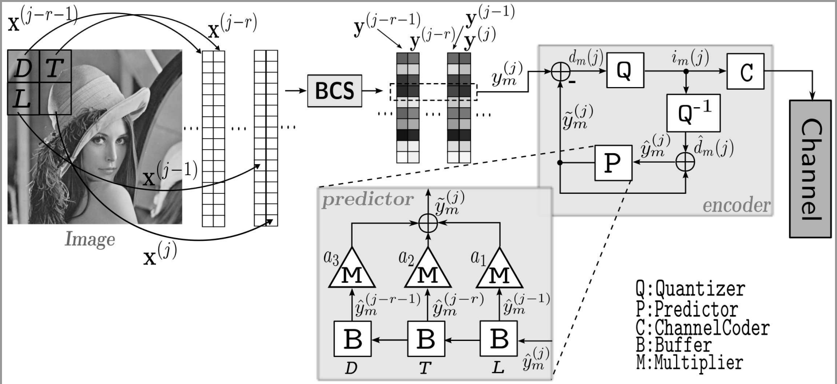
PSNR Performance in dB for a bitrate of 0.5 bpp

	MS-BCS-SPL		
Image	SQ	DPCM	*Gain*
Lenna	33.8	34.7	+0.9
Barbara	26.6	27.4	+0.8
Peppers	33.8	34	+0.2
Goldhill	30.5	31	+0.5
Man	30.5	30.7	+0.2
Clown	32.7	33.2	+0.5
Average	31.3	31.8	+0.5

- Gain for MS is not as large as others
- Because DPCM is only applied to the baseband which only shows high correlation

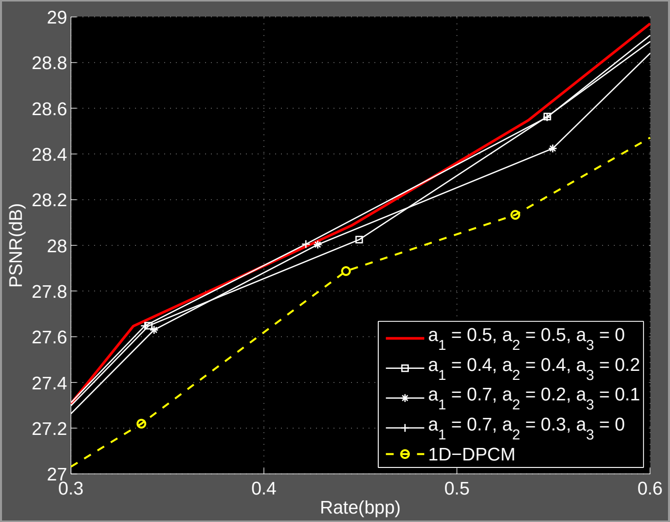
Outline

2D Extension of DPCM for BCS



- For better coding gain, more prediction coefficients used.
- Prediction coefficients—a_1, a_2, and a_3—that sum 1.

Prediction Coefficients for 2D DPCM



- **Empirically found** $a_1 = 0.5$, $a_2 = 0.5$, **and** $a_3 = 0$

Comparison to Other Approaches

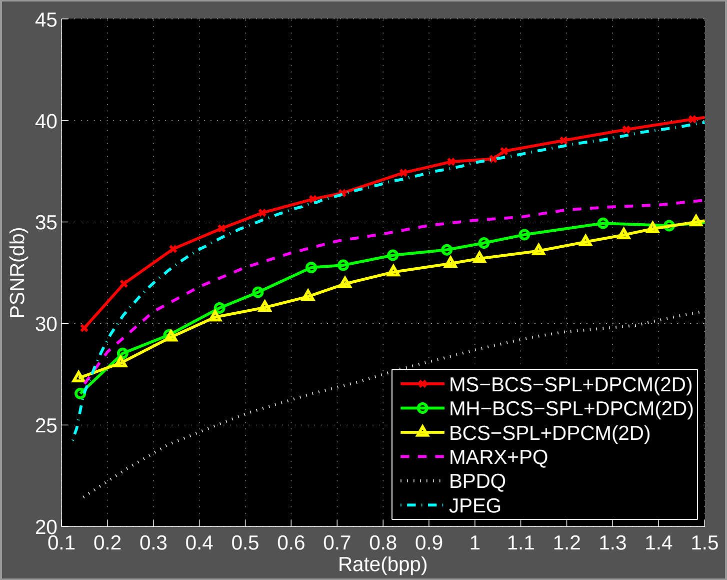
Experimental Setup

- Only 2D-DPCM, not 1D-DPCM and SQ
- Prediction Coefficients, $a_1 = 0.5$, $a_2 = 0.5$, and $a_3 = 0$
- BCS-SPL and MS and MH version

Algorithms Compared

- BPDQ (L. Jacques, 2011)
- MARX + PQ (L. Wang et al, 2011, 2012)
  - MARX (L. Wang et al,2009): extension of TV regularizing more directional gradients
  - Gain using PQ over SQ is averagely 1-2 dB
- Tranditional JPEG as benchmark

Performance Comparison, 'Lenna'

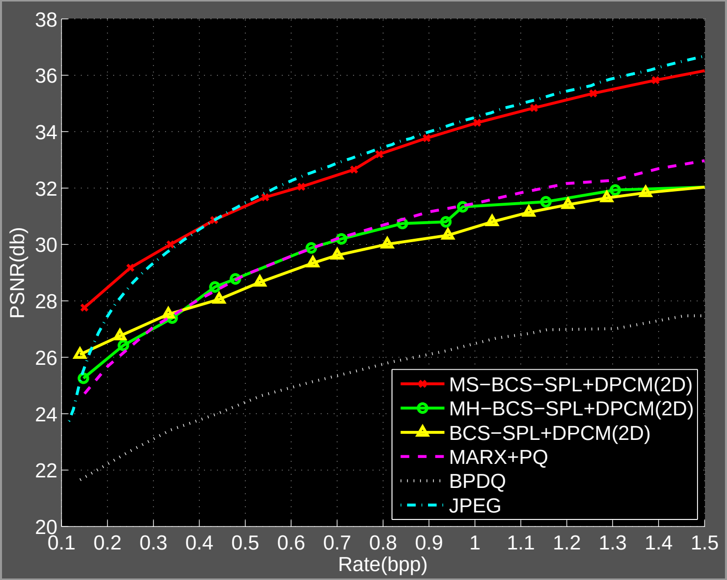


Performance Comparison,‘Goldhill’

BCS of Images and Video

S. Mun

- Images
- Video
- Quantization
  - Quantization
  - DPCM for Images
  - 1D-DPCM-BCS
  - **2D-DPCM-BCS**
- Conclusion



Simulation Results for 2D Images

Lenna (512×512) at 0.25bpp

MS-BCS-SPL
31.6 dB

JPEG
30.9 dB

Simulation Results for 2D Images

Lenna (512×512) at 0.25bpp

MARX+PQ
29.8 dB

BPDQ
23.1 dB

Remarks

Remarks

DPCM is applied on block CS framework

- increasing compression ratio, resulting in better reconstruction quality
- keeping the simple sender part of CS by adding only a small subset of the memory and arithmetic operator
- MH rivals other CS coding technique, MS rivals traditional JPEG

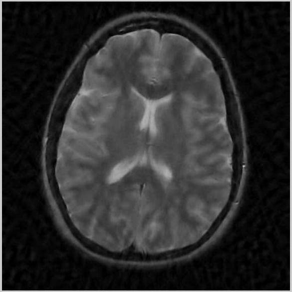
Publication

- S. Mun and J. E. Fowler "DPCM for Quantized Block-Based Compressed Sensing of Images," in Proceedings of the European Signal Processing Conference, Bucharest, Romania, August 2012.

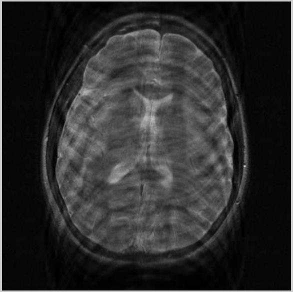
Future Work

Medical Imaging Application

- Medical Imaging Reconstruction by modification of BCS-SPL



CS-SPL

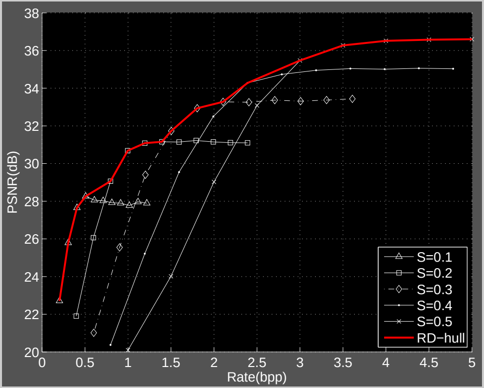


ktFOCUSS

- Radial sensing required (Gaussain, ± 1 not suitable)
- Expected to work in dynamic MRI as well

Future Work

Optimal Subrate & Stepsize without exhaustive search



- Currently needs to test all possible combinations of subrates and stepsize
- Line fitting might be used to find closed equation

Conclusion

URL for all source codes (MATLAB)

- `http://www.ece.msstate.edu/~fowler/BCSSPL/`