Sparse Solutions of Linear Systems of Equations and Sparse Modeling of Signals and Images: Final Presentation

Alfredo Nava-Tudela

John J. Benedetto, advisor

Problem

Let **A** be an *n* by *m* matrix with *n* < *m*, and rank(**A**)=*n.* We want to solve

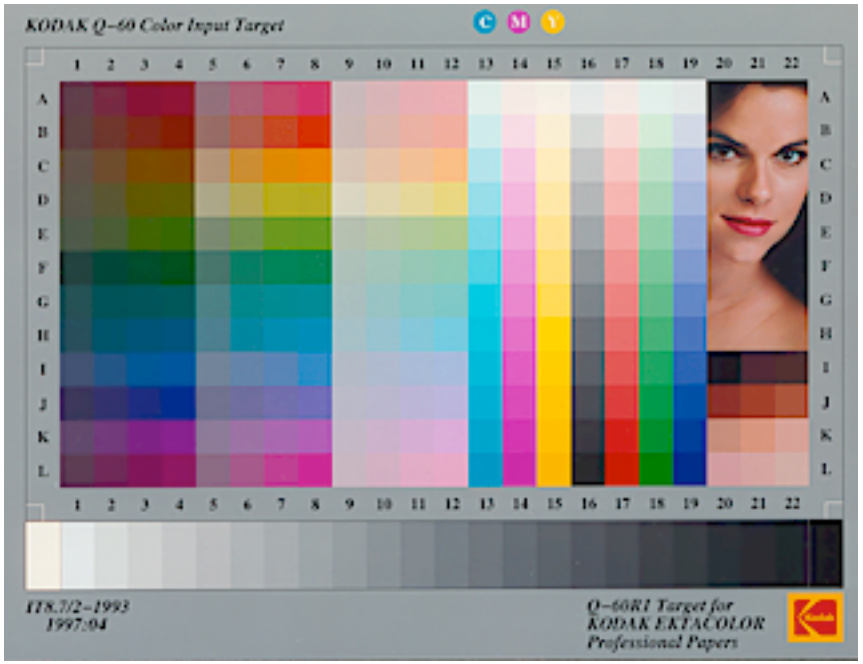
$$\mathbf{Ax} = \mathbf{b}, \qquad \text{where } \mathbf{b} \text{ is a data or signal vector,}$$

and **x** is the solution with the fewest number of non-zero entries possible, that is, the "sparsest" one.

Observations:

- **A** is underdetermined and, since rank(**A**)=*n*, there is an infinite number of solutions. Good!
- How do we find the "sparsest" solution? What does this mean in practice? Is there a unique sparsest solution?

But, why do we care?



231 kb, uncompressed,
320x240x3x8 bit

74 kb, compressed 3.24:1
JPEG

“Sparsity” equals compression

Both JPEG and JPEG2000 achieve their compression mainly because at their core one finds a linear transform (DCT and DWT, respectively) that reduces the number of non-zero entries required to represent the data, within an acceptable error.

We can then think of signal compression in terms of our problem

$$\mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} \text{ is sparse, } \mathbf{b} \text{ is dense, store } \mathbf{x}!$$

Definitions of “sparse”

- Convenient to introduce the l_0 “norm” [1]:

$$||\mathbf{x}||_0 = \#\{k : x_k \neq 0\}$$

- (P_0): $\min_{\mathbf{x}} ||\mathbf{x}||_0$ subject to $||\mathbf{Ax} - \mathbf{b}||_2 = 0$

- (P_0^{ε}): $\min_{\mathbf{x}} ||\mathbf{x}||_0$ subject to $||\mathbf{Ax} - \mathbf{b}||_2 < \varepsilon$

Observations: In practice, (P_0^{ε}) is the working definition of sparsity as it is the only one that is computationally practical. Solving (P_0^{ε}) is NP-hard [2].

Finding sparse solutions:OMP

Orthogonal Matching Pursuit algorithm:

Task: Approximate the solution of $(P_0) : \min_{\mathbf{x}} ||\mathbf{x}||_0$ subject to $\mathbf{Ax} = \mathbf{b}$.

Parameters: We are given the matrix $\mathbf{A}$, the vector $\mathbf{b}$, and the threshold ϵ_0.

Initialization: Initialize $k = 0$, and set

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

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

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} ||z_j\mathbf{a}_j - \mathbf{r}^{k-1}||_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T\mathbf{r}^{k-1}/||\mathbf{a}_j||_2^2$.
- **Update Support:** Find a minimizer j_0 of $\epsilon(j)$: $\forall j \notin \mathcal{S}^{k-1}$, $\epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $||\mathbf{Ax} - \mathbf{b}||_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.
- **Update Residual:** Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$.
- **Stopping Rule:** If $||\mathbf{r}^k||_2 < \epsilon_0$, stop. Otherwise, apply another iteration.

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

Implementation Fine Tuning

My initial OMP implementation wasn't optimized for speed. I made some improvements:

The core of the algorithm is found in the following three steps. Modifying the approach to each of them cut execution times considerably.

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} ||z_j \mathbf{a}_j - \mathbf{r}^{k-1}||_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / ||\mathbf{a}_j||_2^2$.
- **Update Support:** Find a minimizer j_0 of $\epsilon(j)$: $\forall j \notin \mathcal{S}^{k-1}$, $\epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $||\mathbf{Ax} - \mathbf{b}||_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.

Implementation Fine Tuning, Round 1: ompQRf

The first improvement came from computing norm($\mathbf{r}_{k-1}$) |cos(θ_j)|, where θ_j is the angle between $\mathbf{a}_j$ and $\mathbf{r}_{k-1}$. This number reflects how good an approximation to the residue z_j $\mathbf{a}_j$ is, and it is faster to compute than $\varepsilon(j)$.

We also kept track of the best approximant during the 'Sweep' so that 'Update Support' is done in a more efficient way compared to what we had done in ompQR.

Finally, we sweep only on the set of columns that have not been added to the support set, resulting in further time gains on the 'Sweep' step when $k > 1$.

Implementation Fine Tuning, Round 2: ompQRf2

We don't build explicitly $\mathbf{A}_k$ as was done in ompQR or ompQRf, we now update **Q** and **R** at each step *k* such that $\mathbf{A}_k = \mathbf{Q}'\mathbf{R}' = F(\mathbf{Q},\mathbf{R})$. This way we don't have to perform a complete QR decomposition of $\mathbf{A}_k$ at step *k* as was done in those algorithms. This saves time too.

$$\mathbf{Q}'\mathbf{R}' = \mathbf{Q}\mathbf{H}^T(\mathbf{R} \mid \mathbf{H}^T\mathbf{w}) = \mathbf{Q}(\mathbf{H}^T\mathbf{R} \mid \mathbf{H}^2\mathbf{w}) = (\mathbf{Q}\mathbf{R} \mid \mathbf{Q}\mathbf{w})$$
$$= (\mathbf{A}_{k-1} \mid \mathbf{Q}\mathbf{Q}^T\mathbf{a}_{jk}) = (\mathbf{A}_{k-1} \mid \mathbf{a}_{jk}) = \mathbf{A}_k$$

where $\mathbf{Q}' = \mathbf{Q}\mathbf{H}^T$, $\mathbf{R}' = (\mathbf{R} \mid \mathbf{H}\mathbf{w})$, and $\mathbf{w} = (\mathbf{a}_{jk}{}^T\mathbf{Q})^T$

and $\mathbf{H}^T = \mathbf{H}$, $\mathbf{H}^2 = \mathbf{I}$, $\mathbf{H}\mathbf{R} = \mathbf{R}$, with $\mathbf{H}\mathbf{w} = \mathbf{v}$, $\mathbf{v} = (\#,...,\#,0,...,0)^T$
k n-k

Implementation Fine Tuning, Round 3: ompQRf3

Finally, we heed the advice of Matlab to allocate some variables for speed, this change saves time too:

Runtimes for 'experiment.m' ($k = 2$)

ompQR	617.802467 seconds
ompQRf	360.192118 seconds, 1.715 speedup
ompQRf2	308.379138 seconds, 1.168 speedup
ompQRf3	298.622174 seconds, 1.032 speedup

Total speedup from ompQR to ompQRf3: 2.068
(Matlab version 2010b)

Implementation and Validation

Definition: The *mutual coherence* of a matrix **A** is the number

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

Theorem: If **x** solves **Ax** = **b**, and $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$, then **x** is the sparsest solution. That is, if **y** $\neq$ **x** also solves the equation, then $||\mathbf{x}||_0 < ||\mathbf{y}||_0$.

Theorem: For a system of linear equations **Ax** = **b** (**A** an *n* by *m* matrix, *n* < *m*, and rank(**A**) = *n*), if a solution **x** exists obeying $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$, then an OMP run with threshold parameter $\varepsilon_0 = 0$ is guaranteed to find **x** exactly.

Implementation and Validation

In light of these theoretical results, we can envision the following roadmap to validate an implementation of OMP.

- We have a simple theoretical criterion to guarantee both solution uniqueness and OMP convergence:

If $\mathbf{x}$ *is a solution to* $\mathbf{Ax} = \mathbf{b}$*, and* $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$*, then* $\mathbf{x}$ *is the unique sparsest solution to* $\mathbf{Ax} = \mathbf{b}$ *and OMP will find it.*

- Hence, given a full-rank n by m matrix $\mathbf{A}$ ($n < m$), compute $\mu(\mathbf{A})$, and find the largest integer k smaller than or equal to $(1+\mu(\mathbf{A})^{-1})/2$. That is, $k = \text{floor}((1+\mu(\mathbf{A})^{-1})/2)$.

Implementation and Validation

- Build a vector **x** with exactly *k* non-zero entries and produce a right hand side vector **b** = **Ax**. This way, you have a known sparsest solution **x** to which to compare the output of any OMP implementation.

- Pass **A**, **b**, and ε_0 to OMP to produce a solution vector $\mathbf{x}_{omp} = OMP(\mathbf{A},\mathbf{b},\varepsilon_0)$.

- If OMP terminates after *k* iterations and $||\mathbf{Ax}_{omp} - \mathbf{b}|| < \varepsilon_0$, for all possible **x** and $\varepsilon_0 > 0$, then the OMP implementation would have been validated.

Caveat: The theoretical proofs assume infinite precision.

Validation Results

We ran two experiments:

1) $\mathbf{A} \in R^{100x200}$, with entries in N(0,1) i.i.d. for which $\mu(\mathbf{A}) = 0.3713$, corresponding to $k = 1 \leq \mathrm{K}$.
2) $\mathbf{A} \in R^{200x400}$, with entries in N(0,1) i.i.d. for which $\mu(\mathbf{A}) = 0.3064$, corresponding to $k = 2 \leq \mathrm{K}$.

Observations:

- **A** will be full-rank with probability 1 [1].
- For full-rank matrices **A** of size n x m, the mutual coherence satisfies $\mu(\mathbf{A}) \geq \sqrt{\{(m - n)/(n \cdot (m - 1))\}}$ [4]. That is, the upper bound of $\mathrm{K} = (1 + \mu(\mathbf{A})^{-1})/2$ can be made as big as needed, provided n and m are big enough.

Validation Results

For each matrix **A**, we chose 100 vectors with *k* non-zero entries whose positions were chosen at random, and whose entries were in N(0,1).

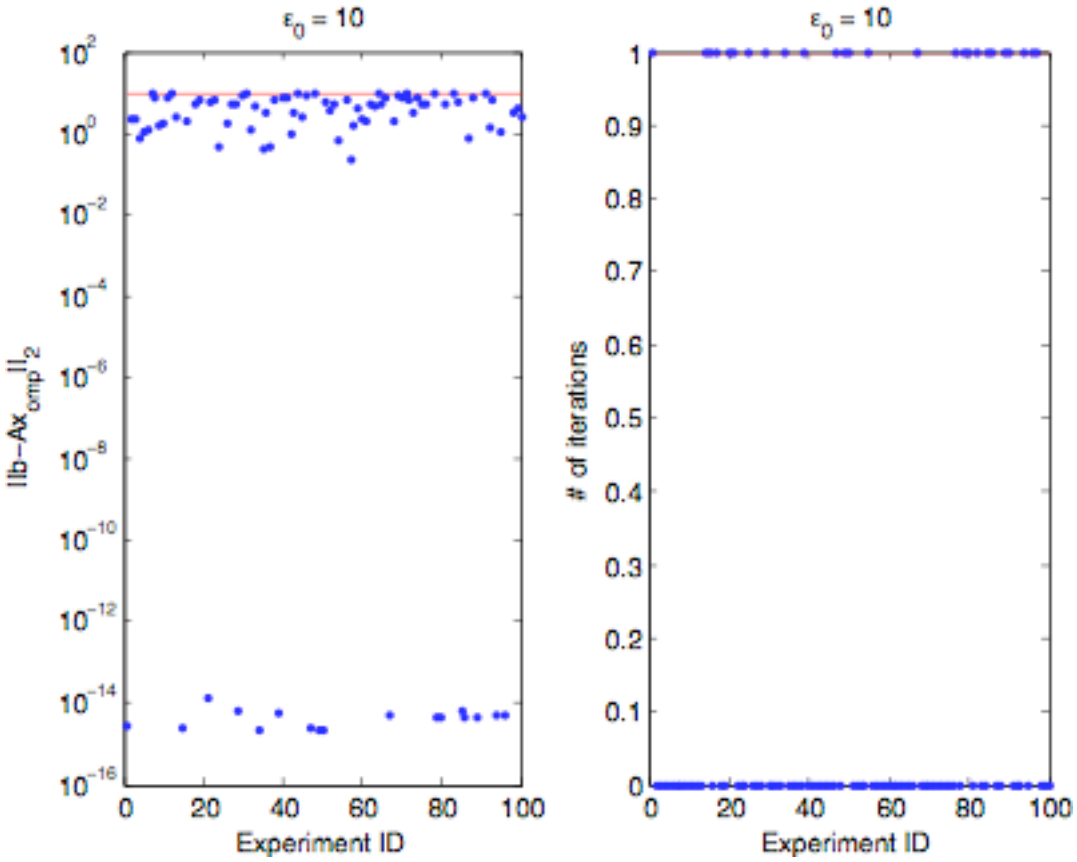
Then, for each such vector **x**, we built a corresponding right hand side vector **b** = **Ax**.

Each of these vectors would then be the unique sparsest solution to **Ax** = **b**, and OMP should be able to find them.

Finally, given $\varepsilon_0 > 0$, if our implementation of OMP were correct, it should stop after *k* steps (or less), and if $\mathbf{x}_{OMP} = OMP(\mathbf{A},\mathbf{b},\varepsilon_0)$, then $||\mathbf{b} - \mathbf{A}\mathbf{x}_{OMP}|| < \varepsilon_0$.

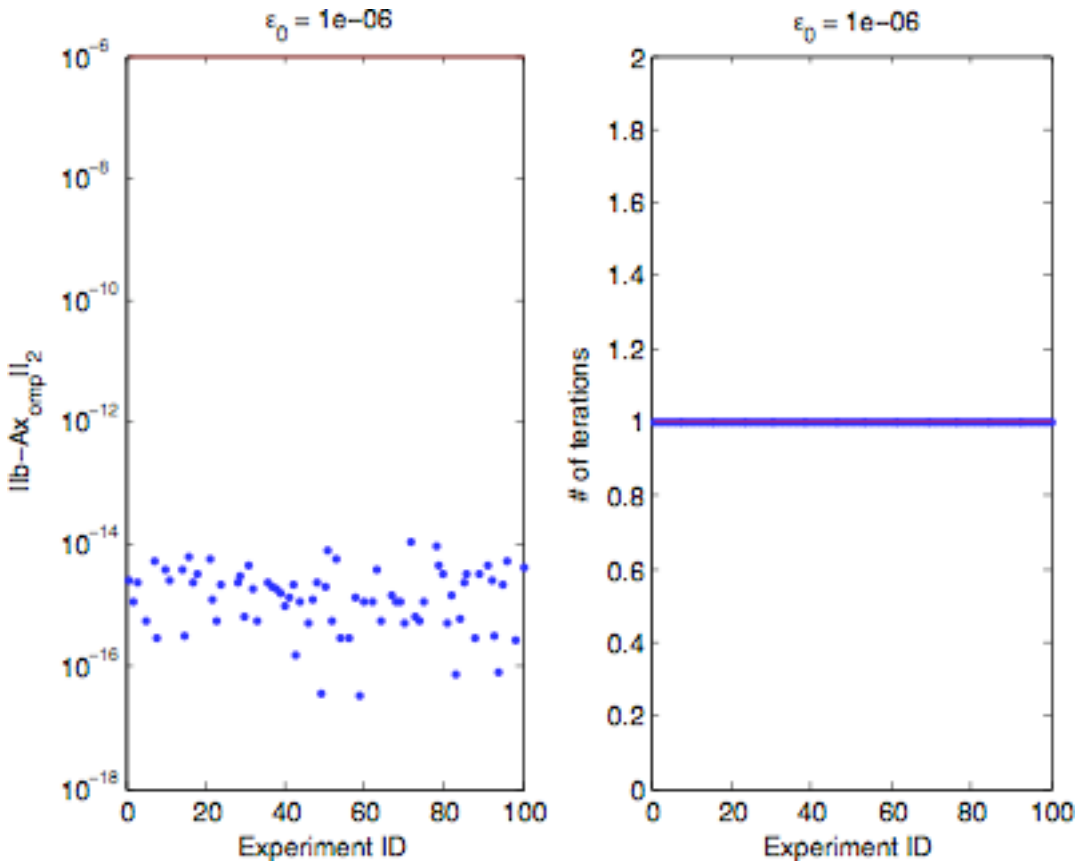
Validation Results

$k = 1$



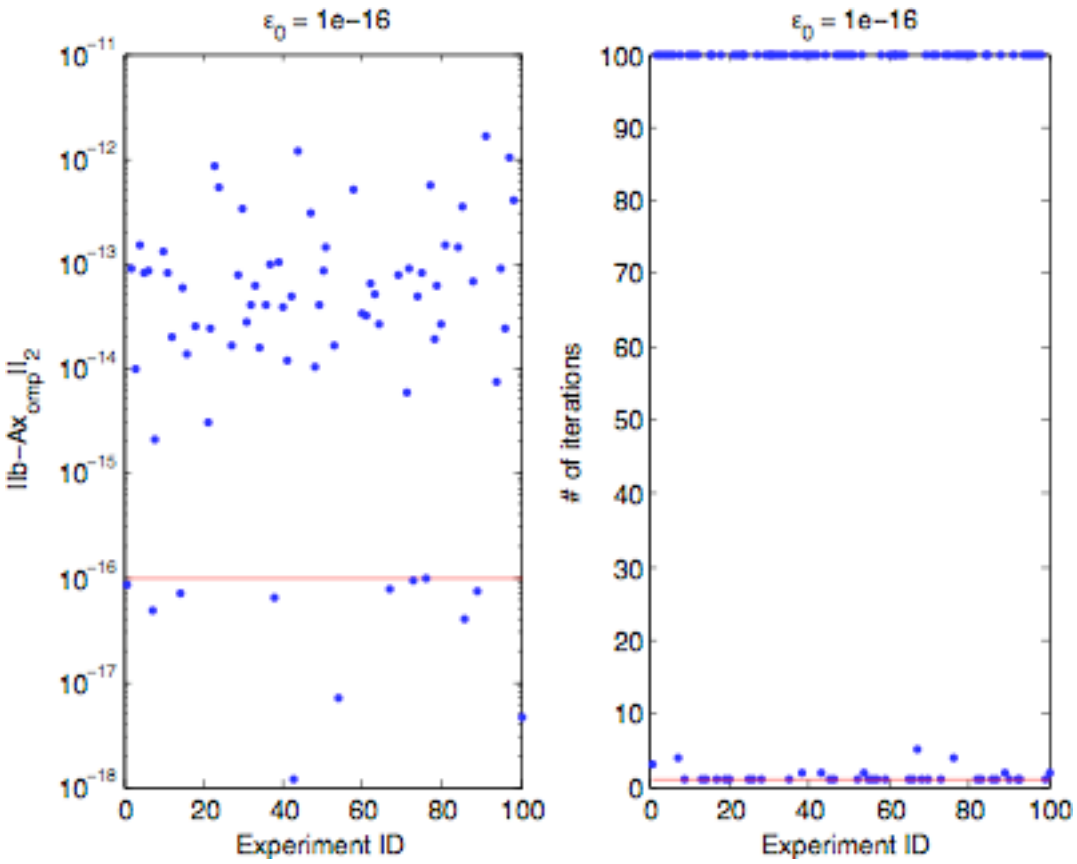
Validation Results

$k = 1$



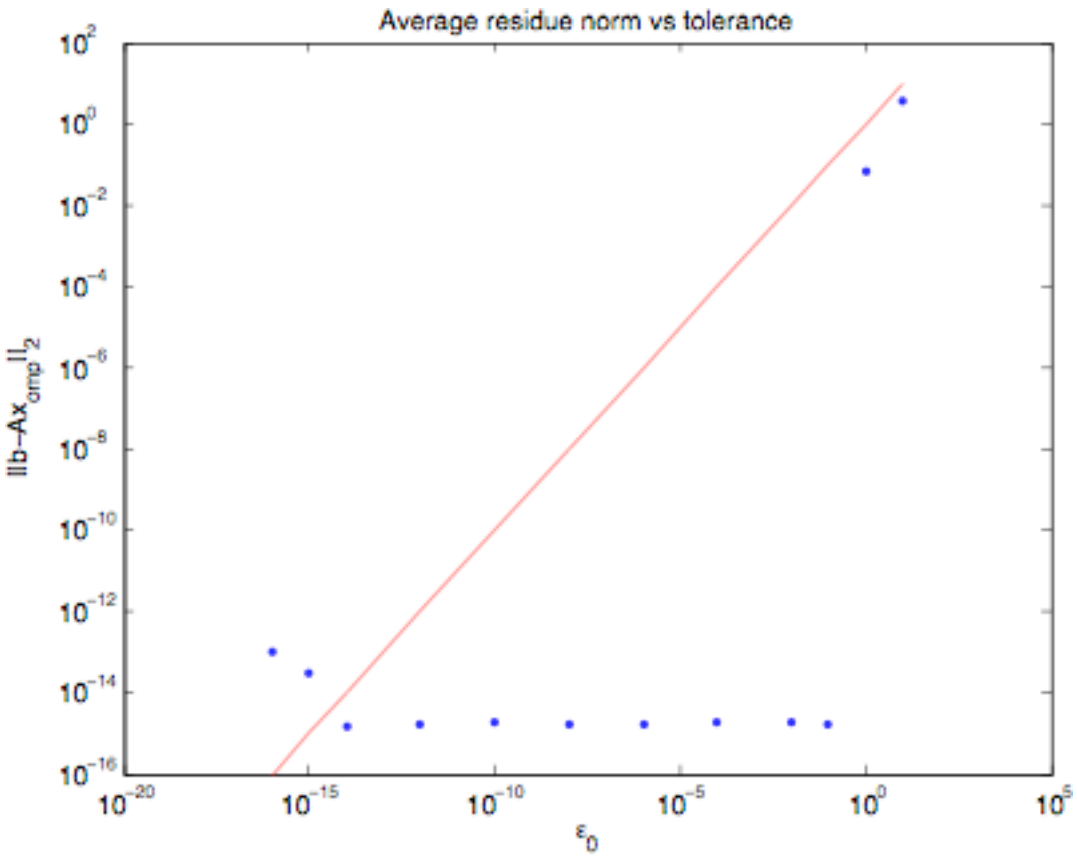
Validation Results

$k = 1$



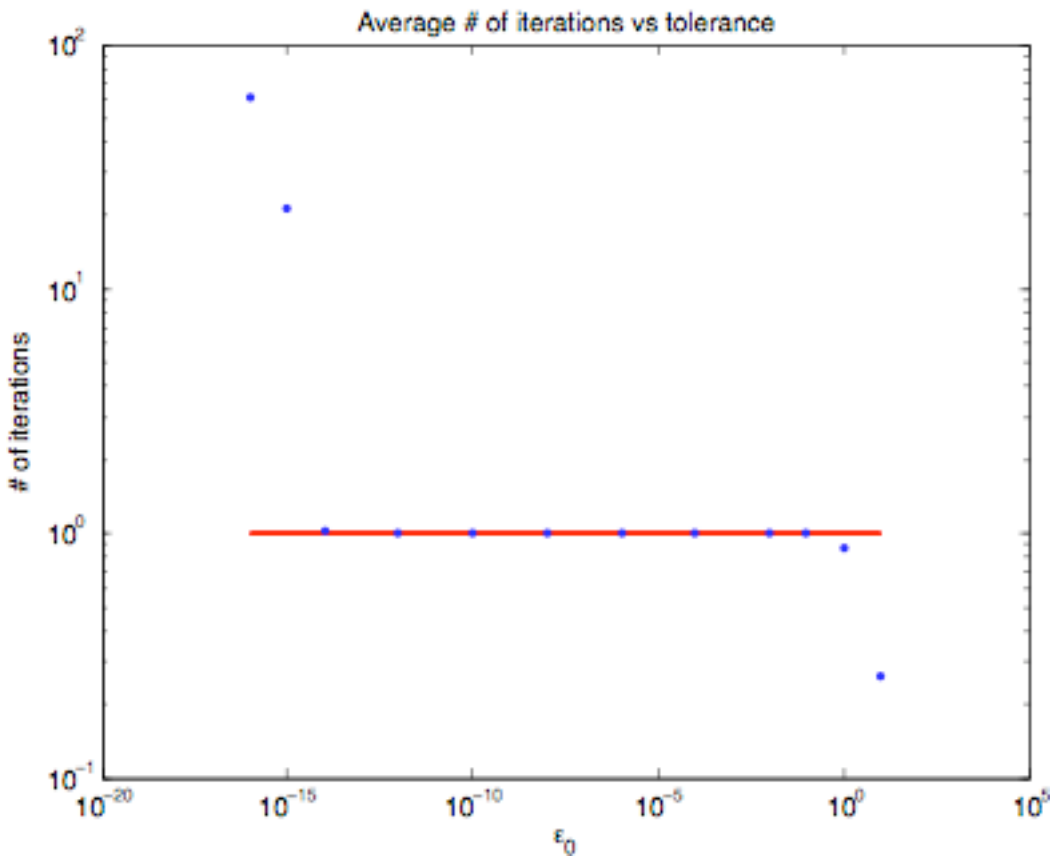
Validation Results

$k = 1$



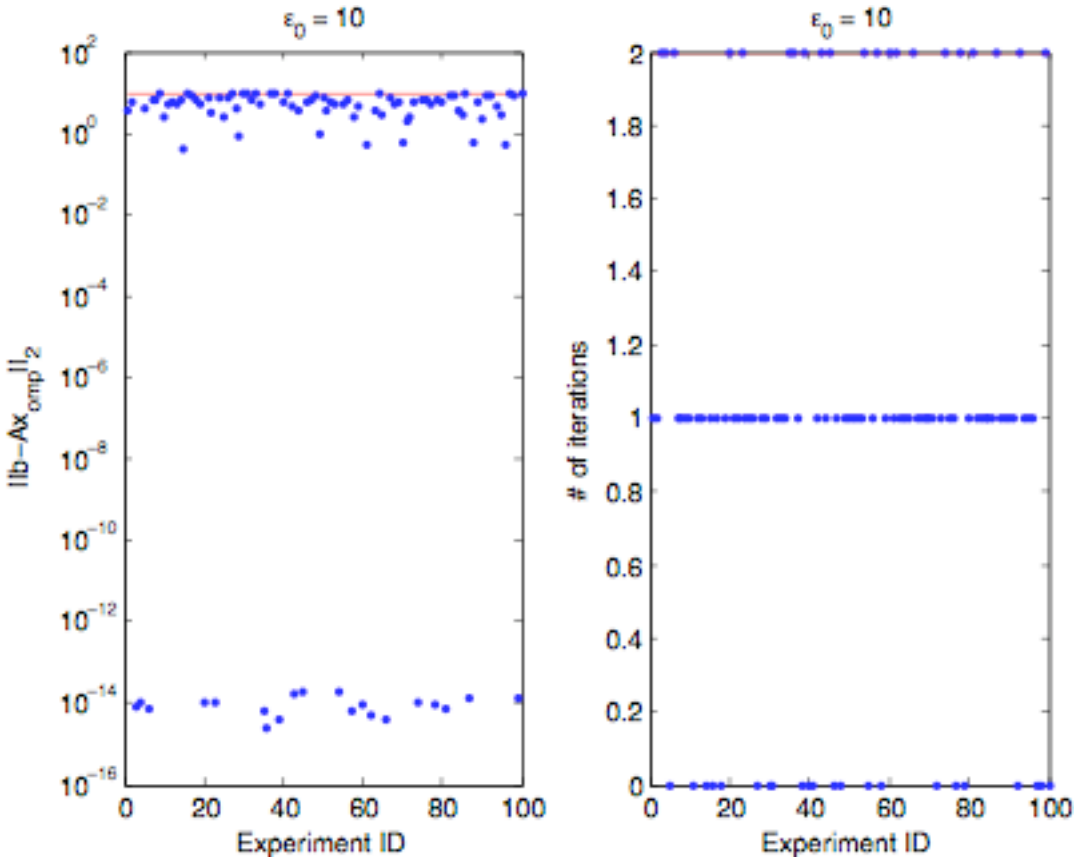
Validation Results

$k = 1$



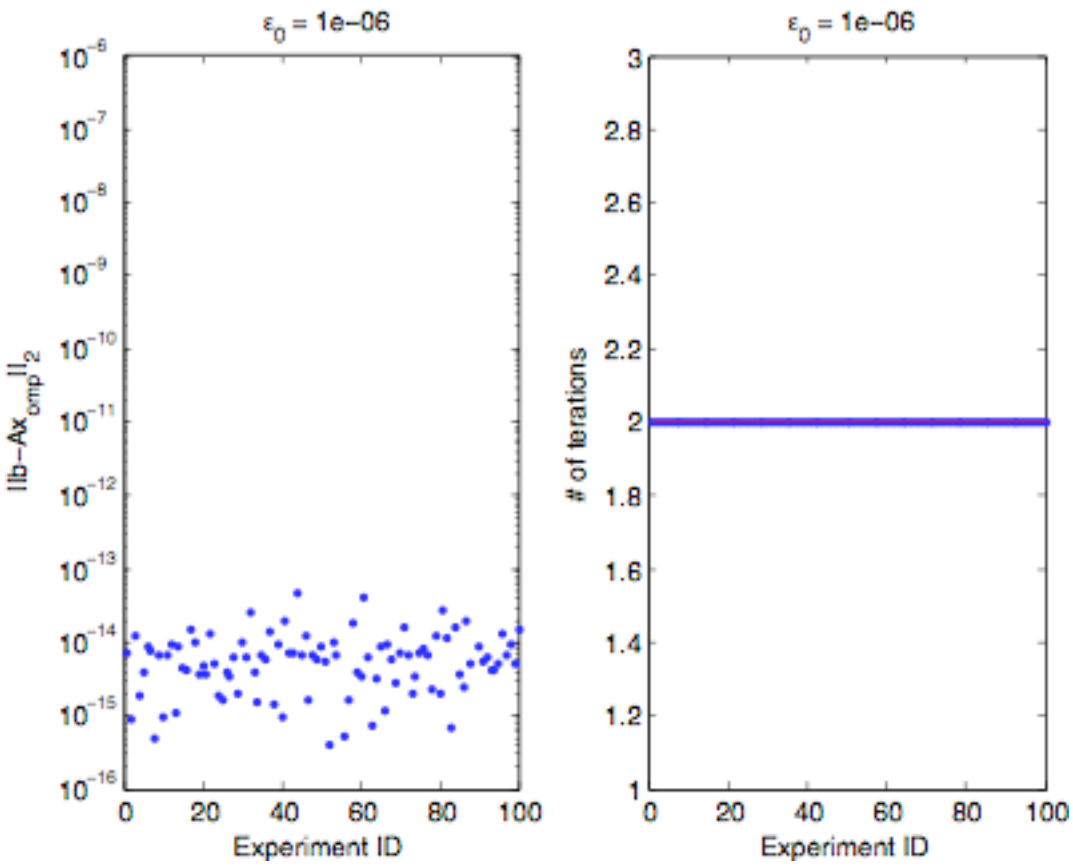
Validation Results

$k = 2$



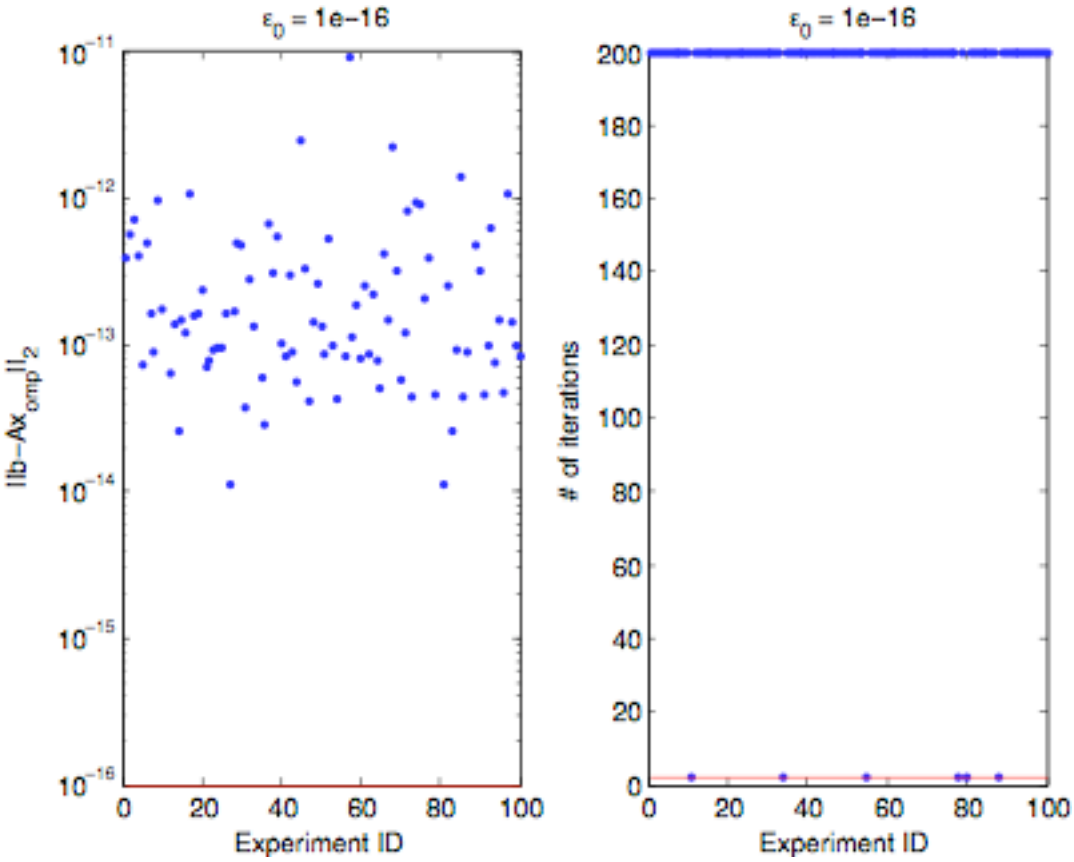
Validation Results

$k = 2$



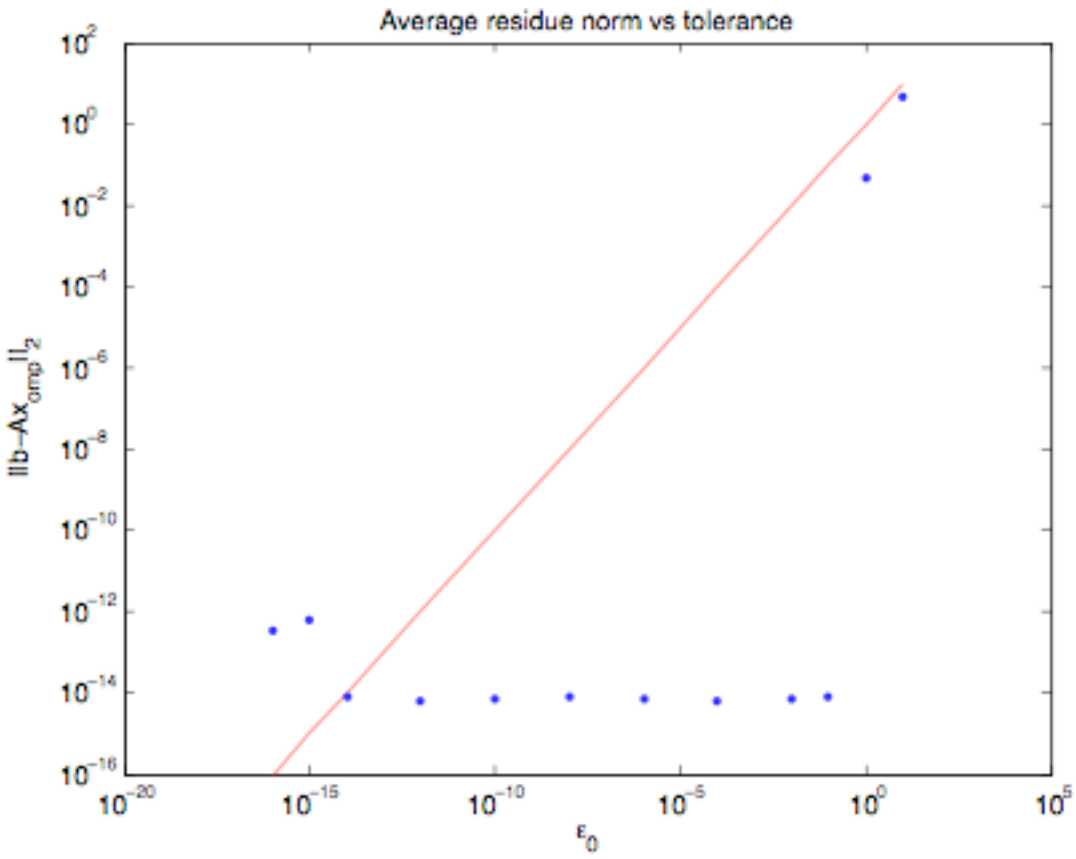
Validation Results

$k = 2$



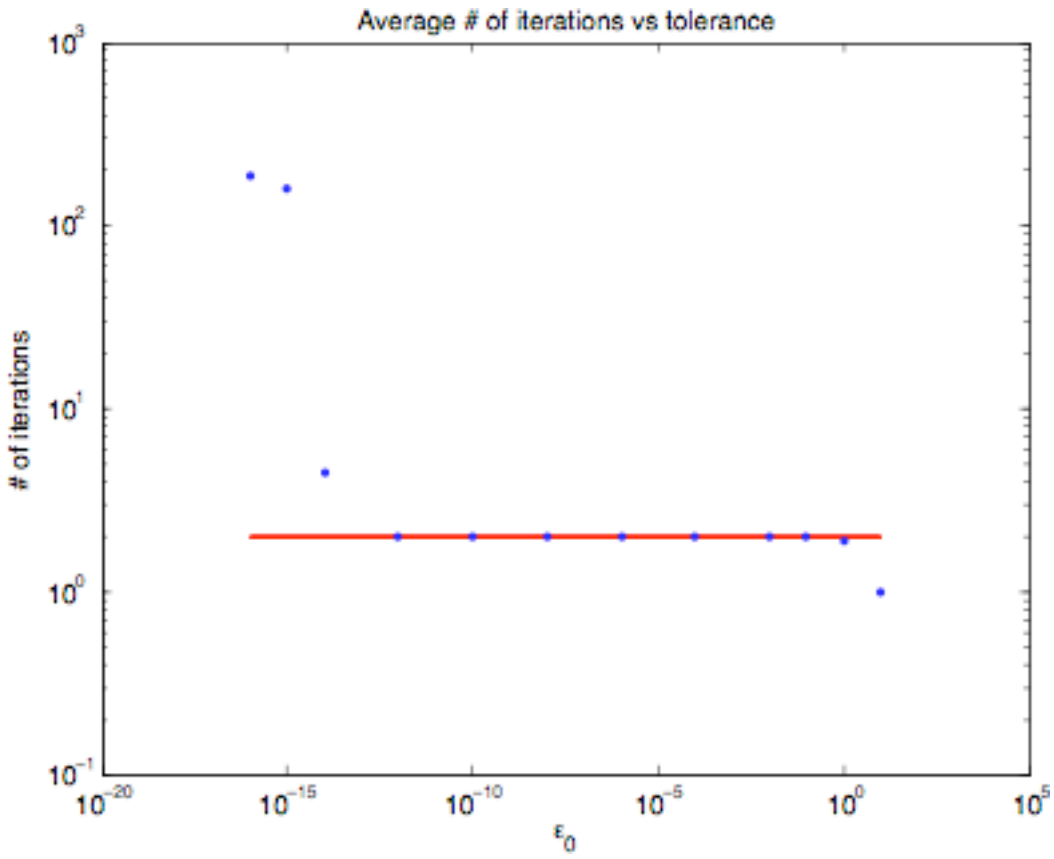
Validation Results

$k = 2$



Validation Results

$k = 2$



Reproducing Paper Results

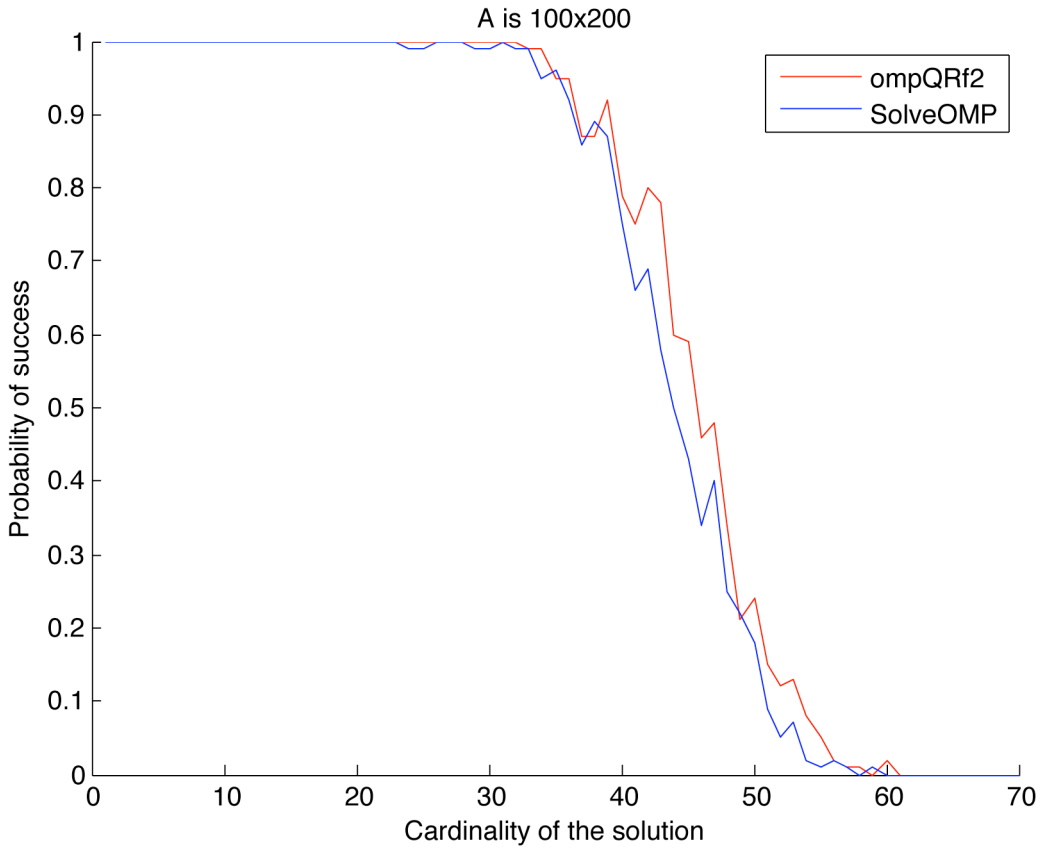
For the first portion of our testing protocol, we set to reproduce the experiment described in section (3.3.1) of [1], limited to the results obtained for OMP.

Ax = **b**, where **A** is 100 x 200, each column i.i.d. N(0,1), and **x** has *k* non-zero entries chosen at random and i.i.d. N(0,1).

Repeat 100 times, for each *k* = 1 to 70, the following experiment and count the number of successes:
With **b** having been set to **Ax**, does $\mathbf{x}_{omp}$ = omp(**A**,**b**,1e-5) converge to **x** within the given tolerance?

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Review, 51 (2009), pp. 34–81.

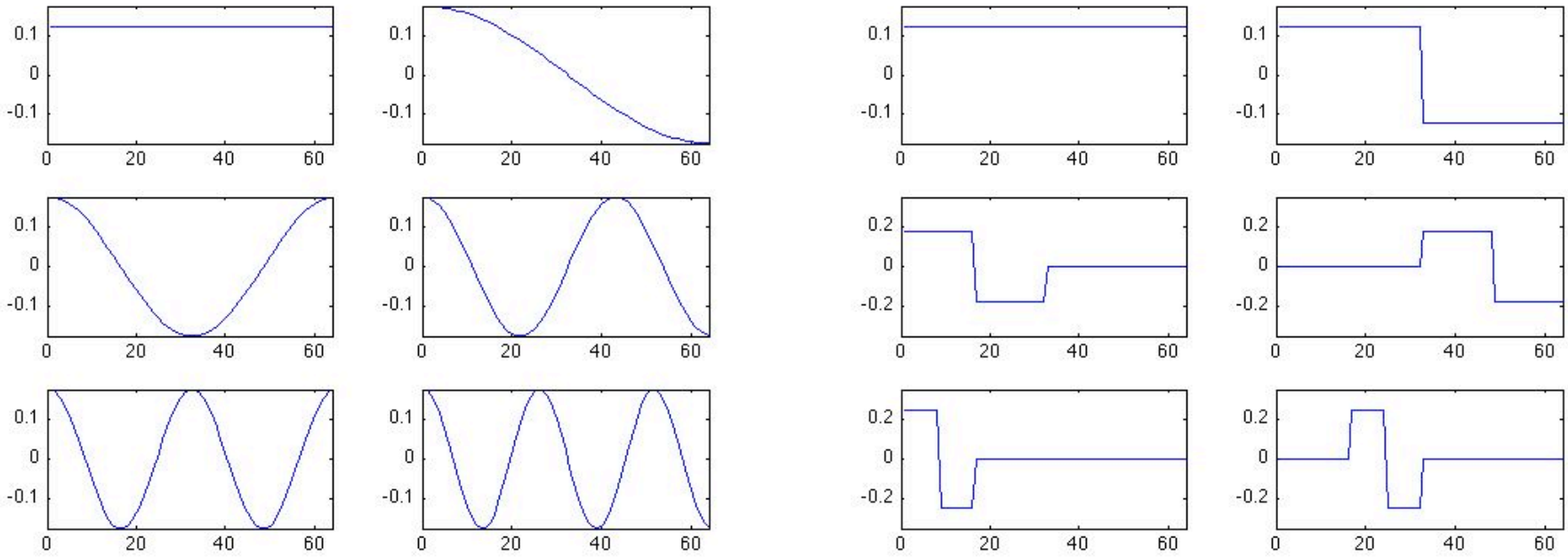
Reproducing Paper Results



SolveOMP is SparseLab's implementation of OMP (http://sparselab.stanford.edu/)

Signal Compression: setup

Consider the matrix **A** = [DCT Haar], where DCT is the basis of Discrete Cosine Transform waveforms, and Haar is the basis generated by the Haar wavelet.



DCT Haar

Signal Compression: images

We selected 5 natural images to test the compression properties of **A**, and compare to compression via DCT or Haar alone, i.e. **B** = [DCT], or **C** = [Haar]

Lena

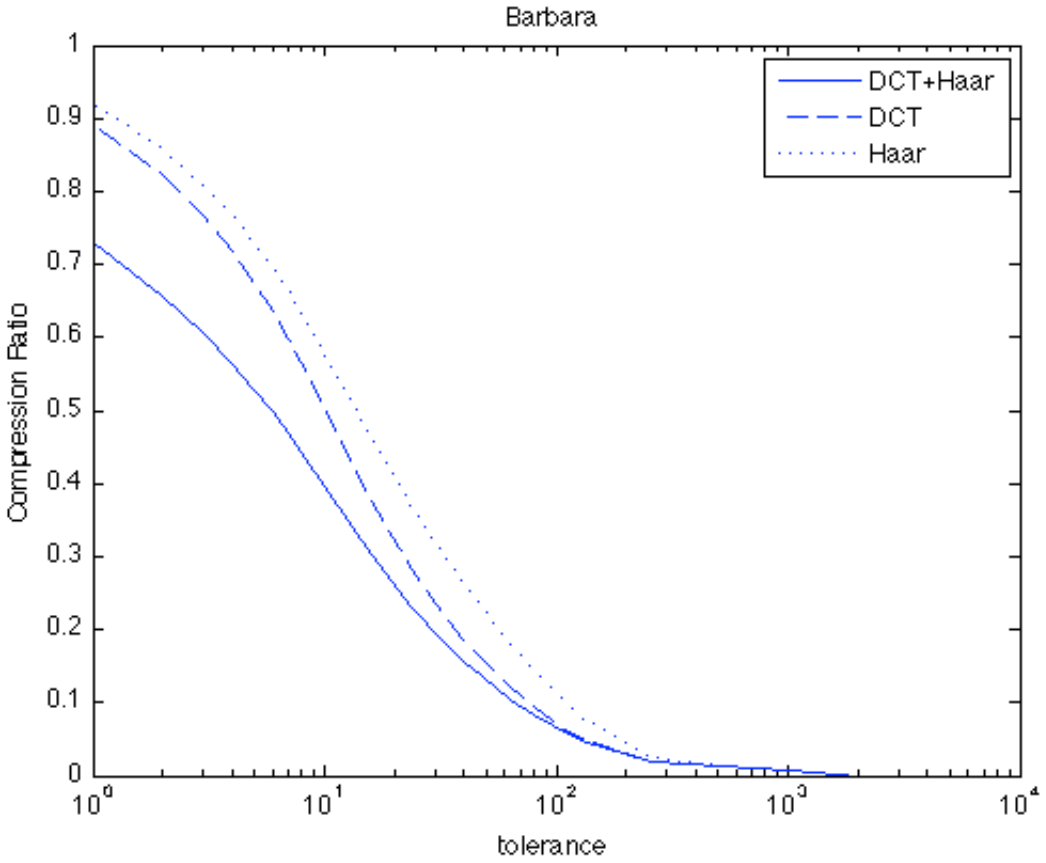
Peppers

Barbara

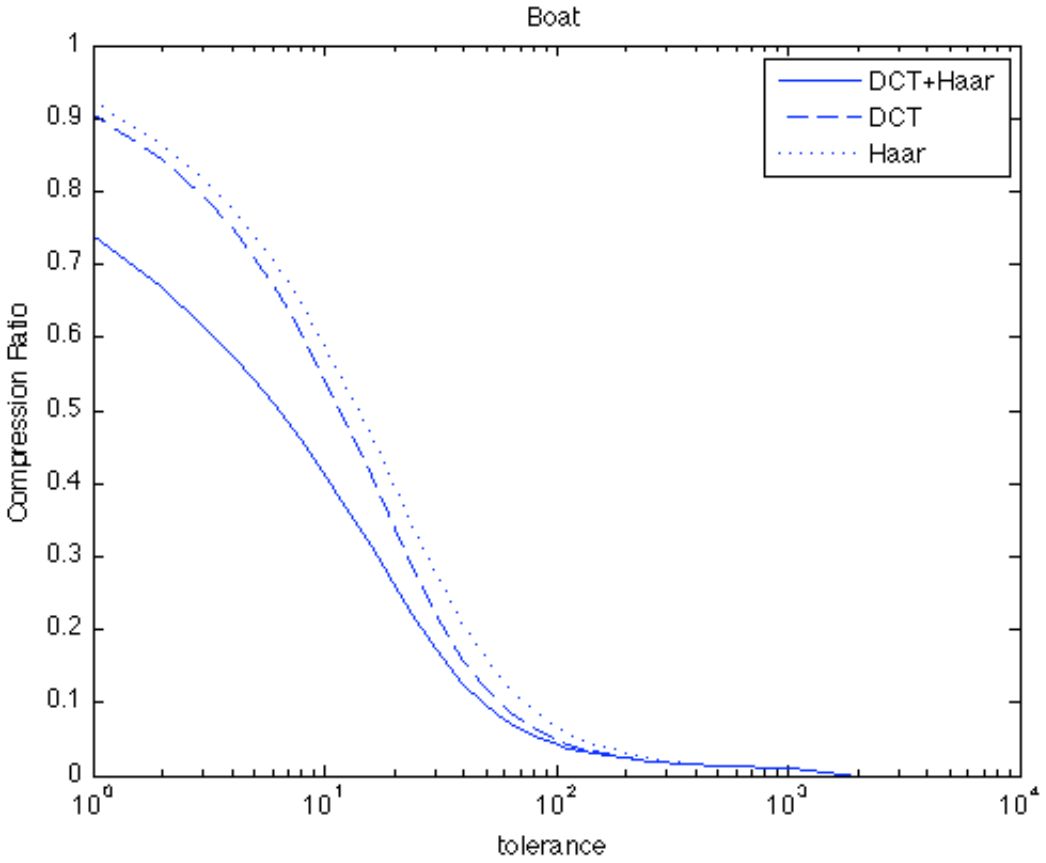
Boat

Elaine

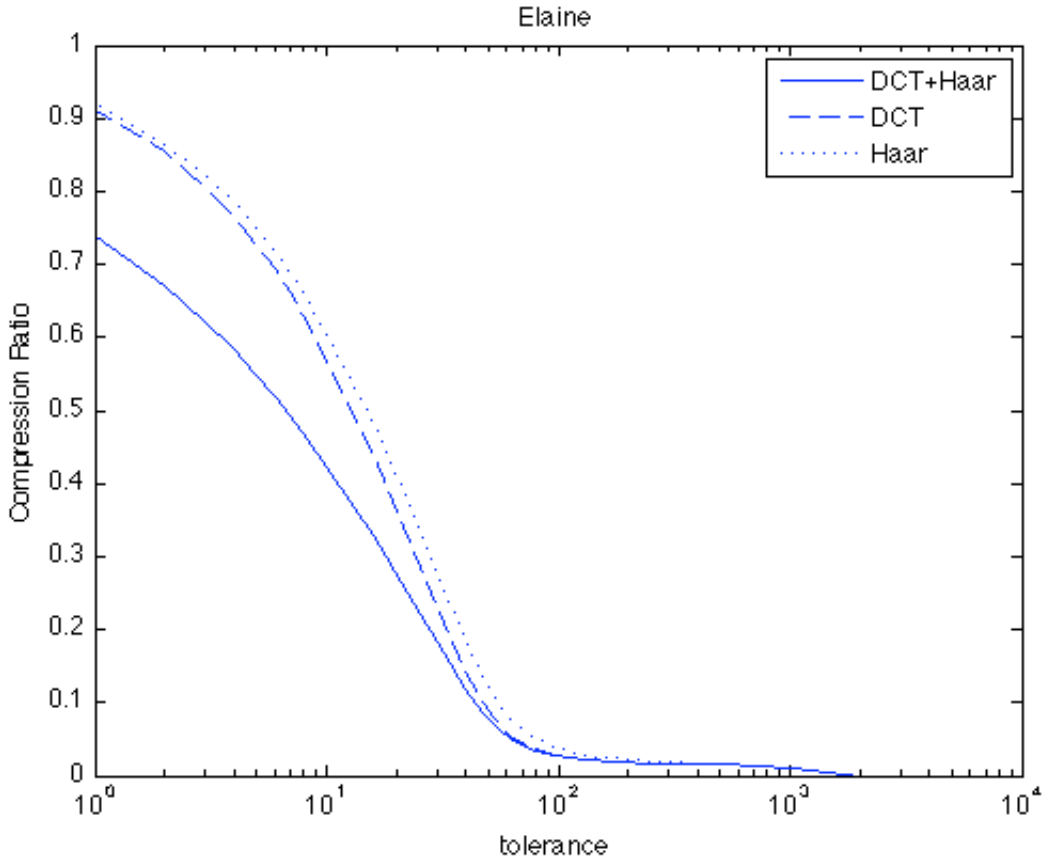
Signal Compression: Barbara



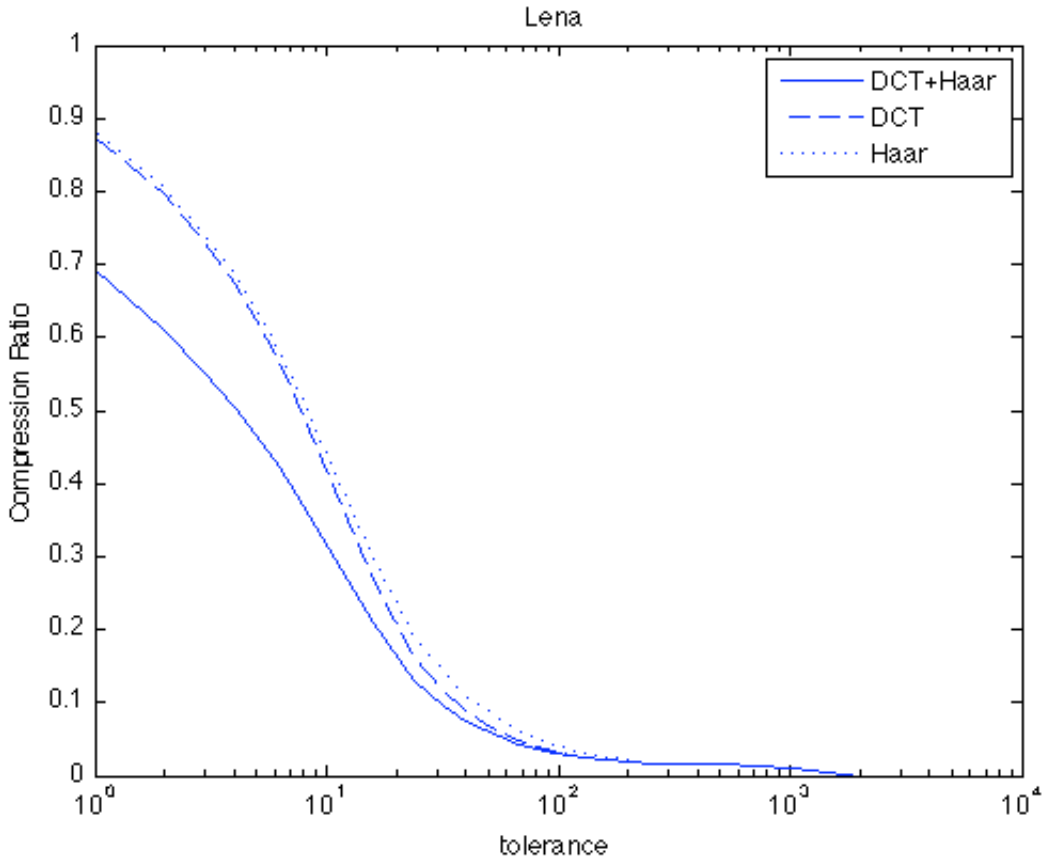
Signal Compression: Boat



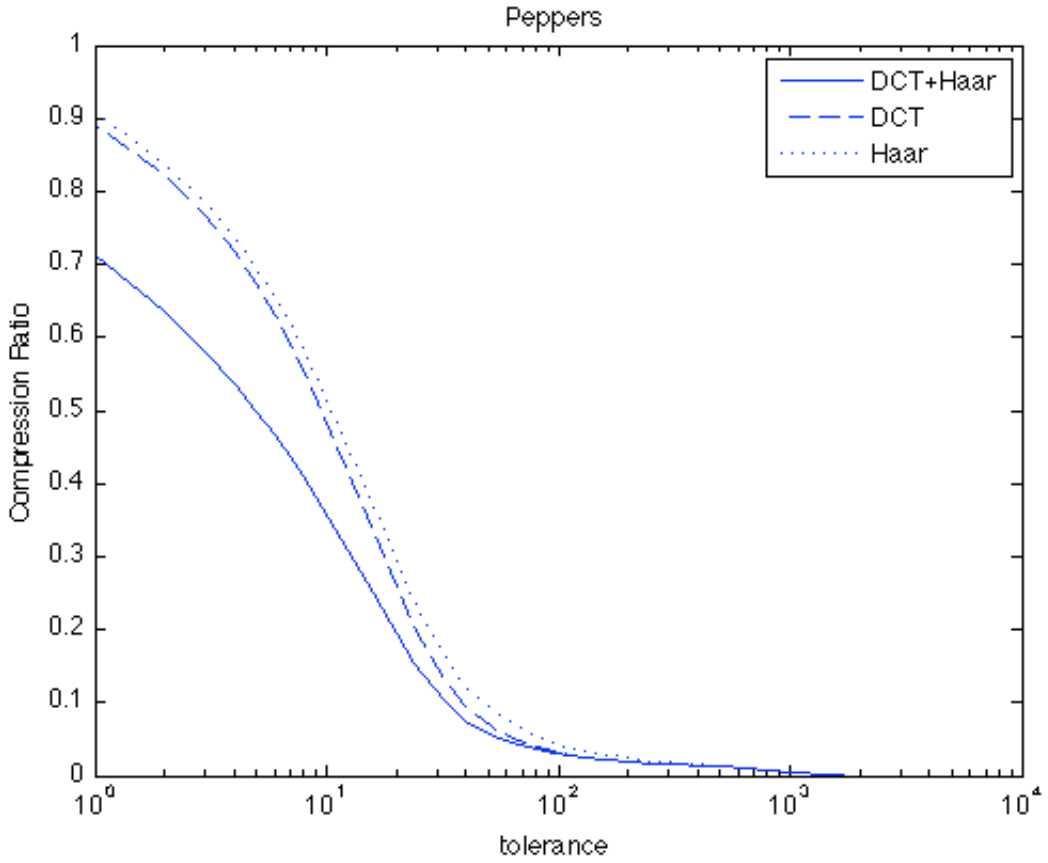
Signal Compression: Elaine



Signal Compression: Lena



Signal Compression: Peppers



Error Estimation

Peak Signal-to-Noise Ratio (PSNR):

PSNR = $20 \log_{10}(MAX_{\mathbf{X}} / \sqrt{MSE})$, (units in dB)

with $MAX_{\mathbf{X}} = 255$, and $MSE = \Sigma_{i,j} (\mathbf{X}(i,j) - \mathbf{Y}(i,j))^2 / nm$.

Structural Similarity (SSIM), and Mean Structural Similarity(MSSIM) indices [8]:

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\,\mu_x\,\mu_y + C_1)\,(2\,\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)\,(\sigma_x^2 + \sigma_y^2 + C_2)}$$

$$\mathrm{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}(\mathbf{x}_j, \mathbf{y}_j)$$

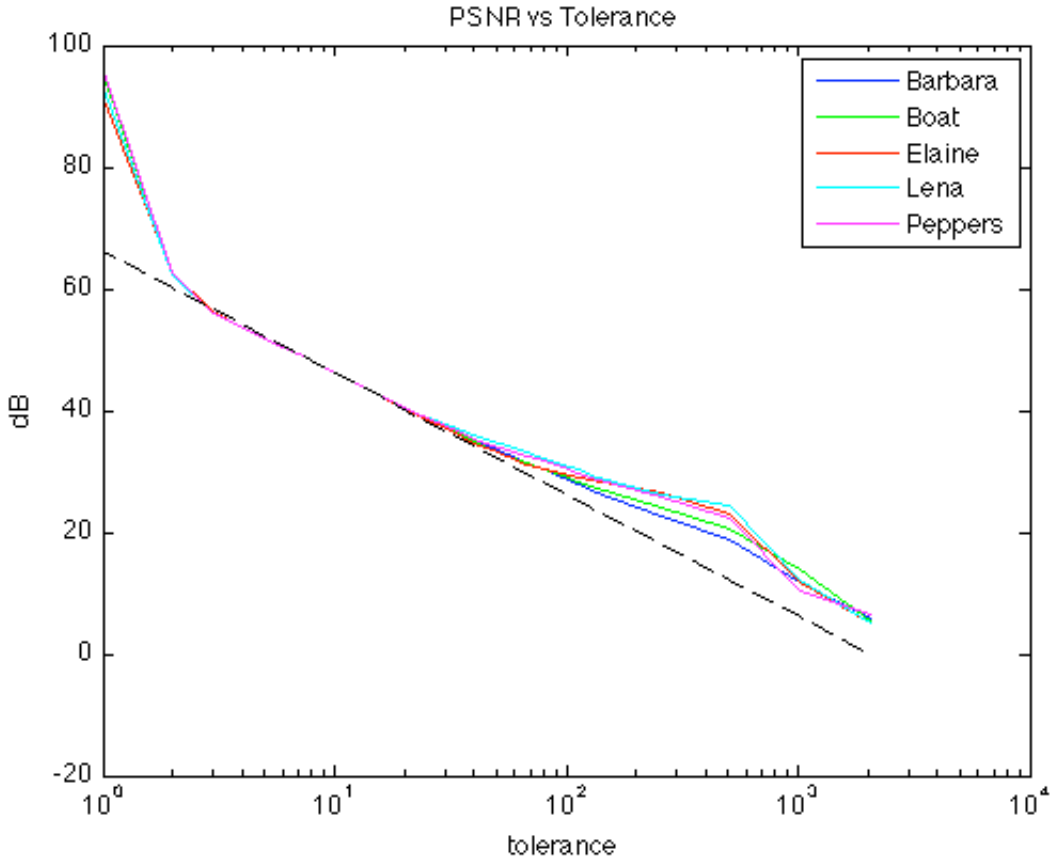
Error Estimation

Ideal error distribution. Consider an LxL image that has been linearized to a vector $\mathbf{b}$ of length L^2. Assume that the OMP approximation within ε has distributed the error evenly, that is, if $\mathbf{y} = \mathbf{A}\mathbf{x}_{omp}$

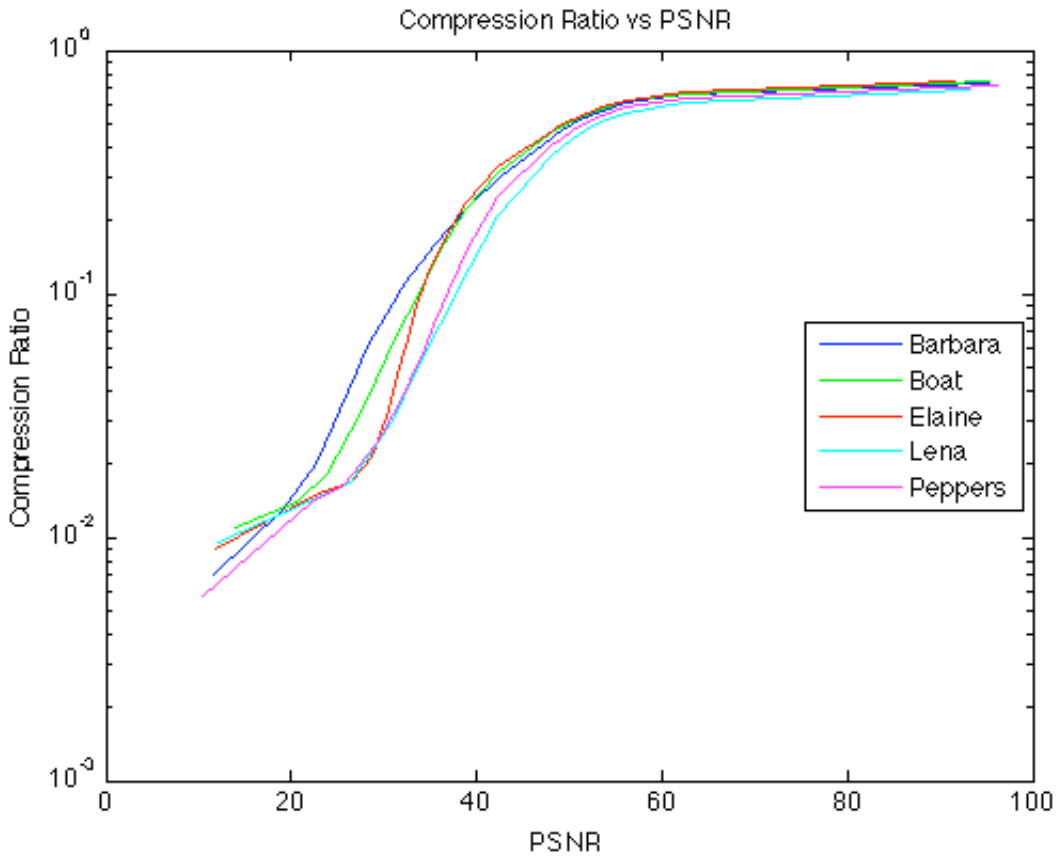
$$\begin{aligned} \| \mathbf{A}\mathbf{x}_{omp} - \mathbf{b} \|_2 < \varepsilon &\Leftrightarrow \| \mathbf{y} - \mathbf{b} \|_2^2 < \varepsilon^2 \\ &\Leftrightarrow \sum_{j=1,\ldots,L^2} (\mathbf{y}_j - \mathbf{b}_j)^2 < \varepsilon^2 \\ &\Leftrightarrow L^2 c^2 < \varepsilon^2 \\ &\Leftrightarrow c < \varepsilon/L \end{aligned}$$

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

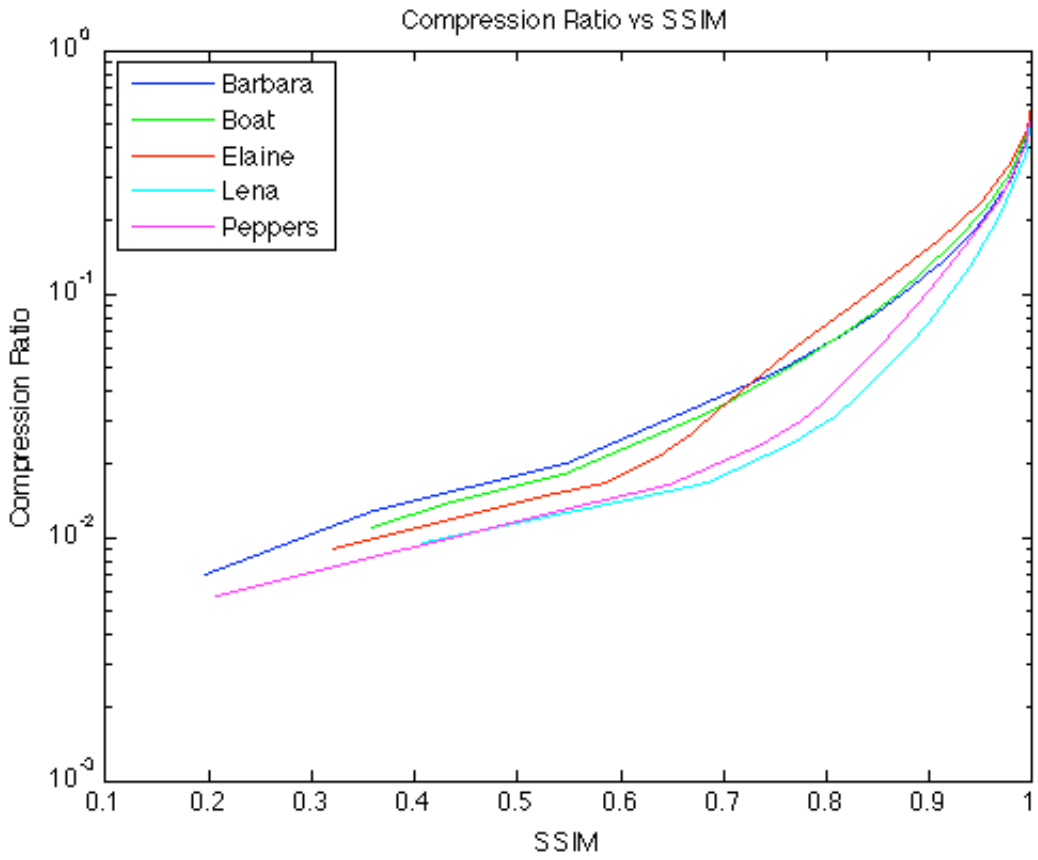
Signal Compression: PSNR



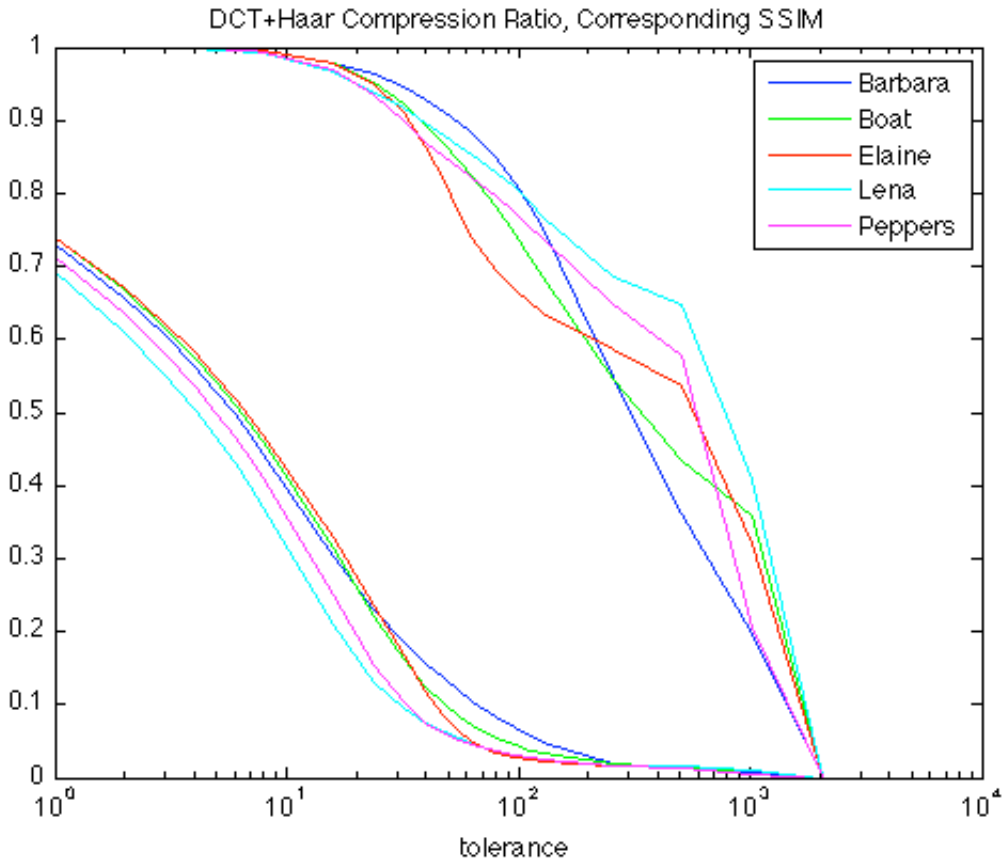
Signal Compression: PSNR



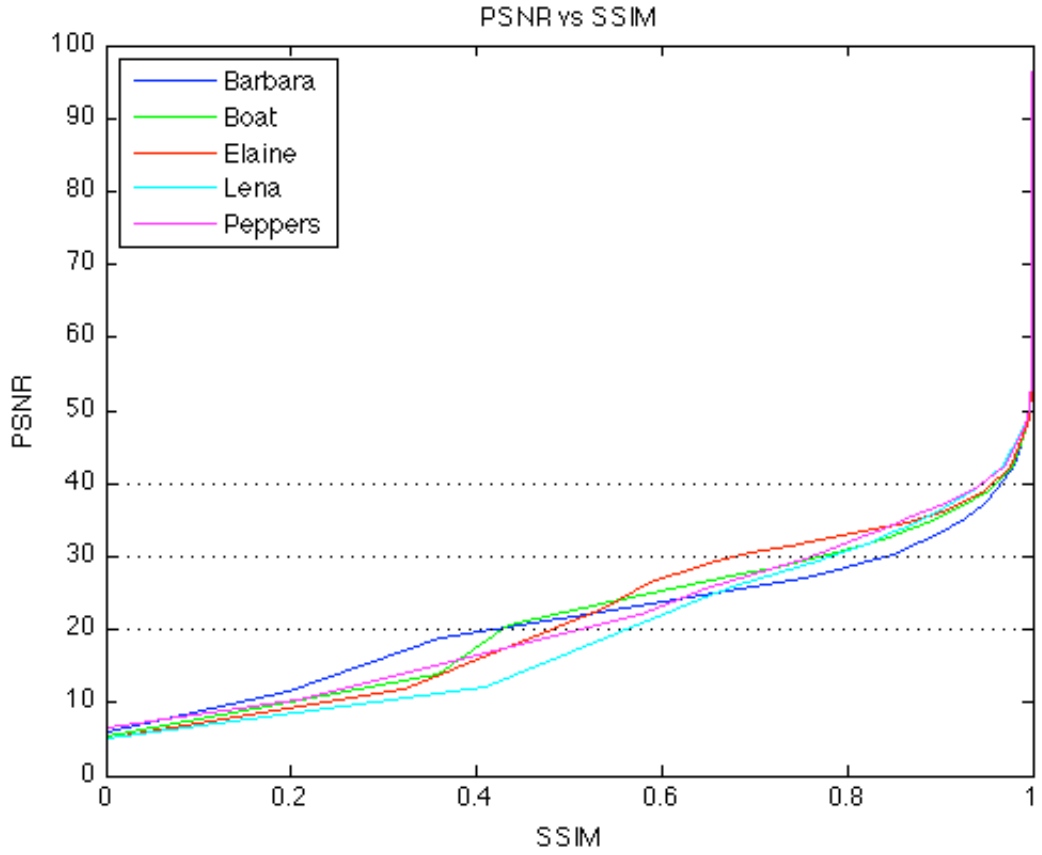
Signal Compression: SSIM



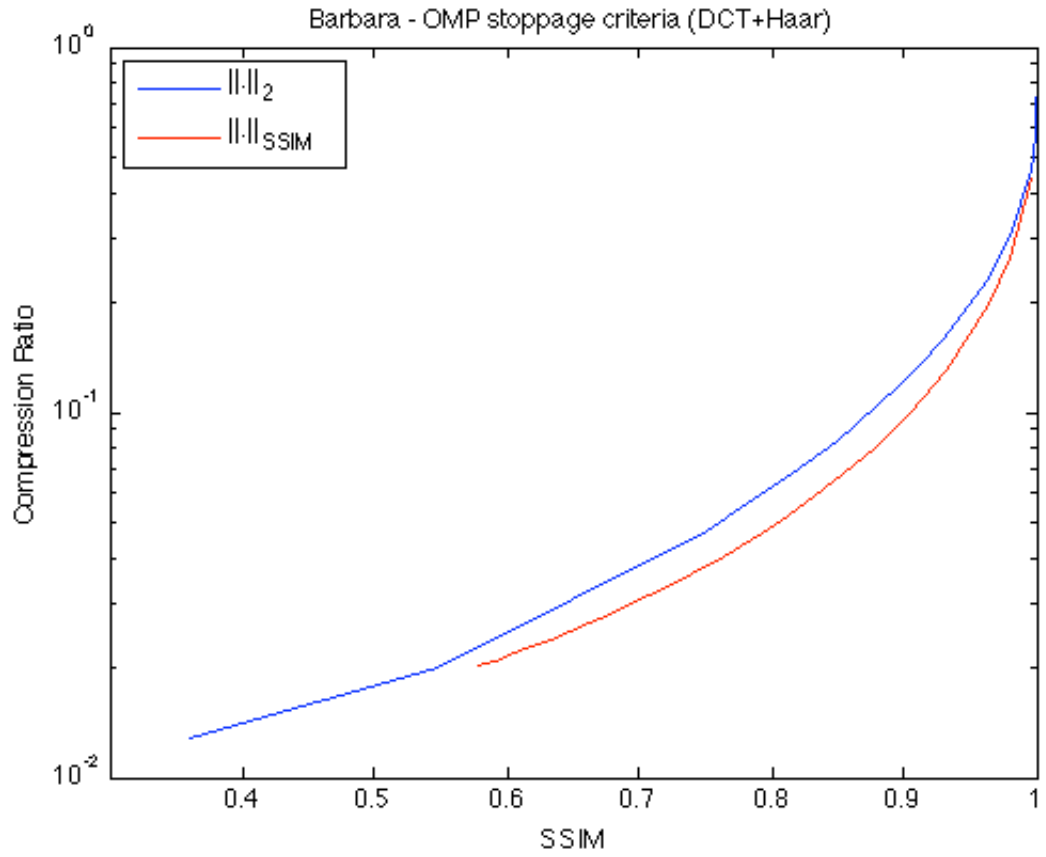
Signal Compression: SSIM



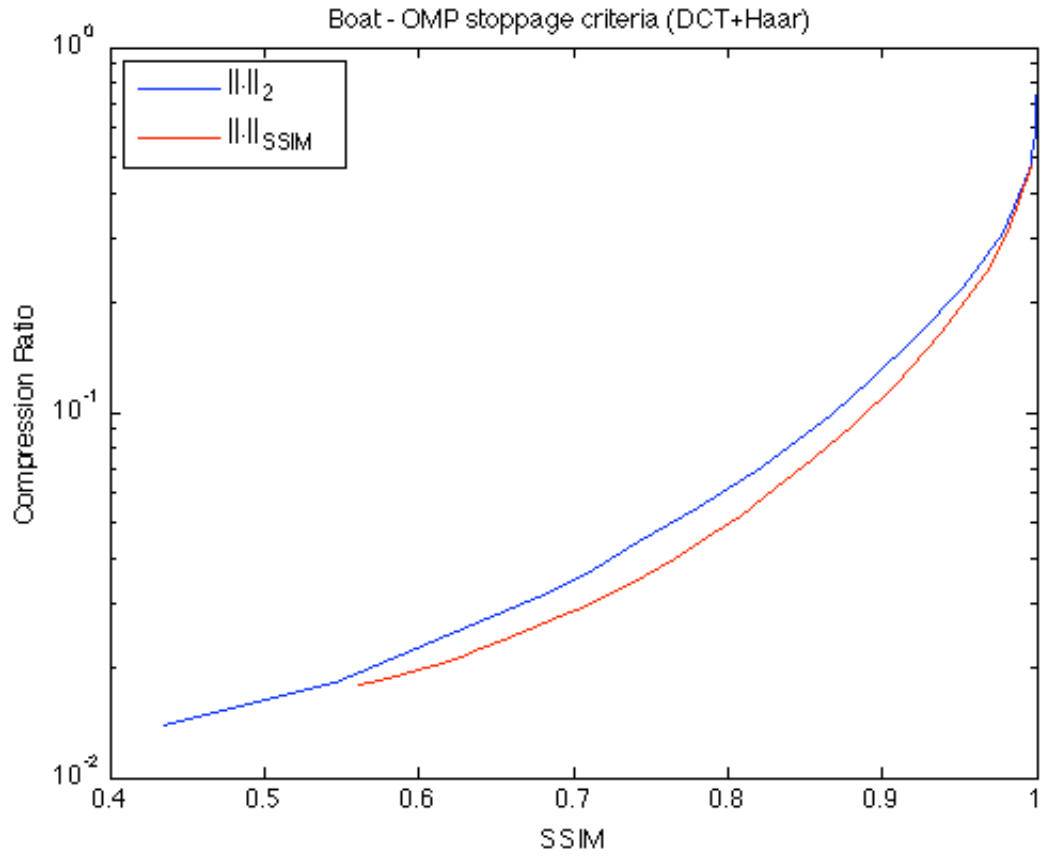
Error Comparison



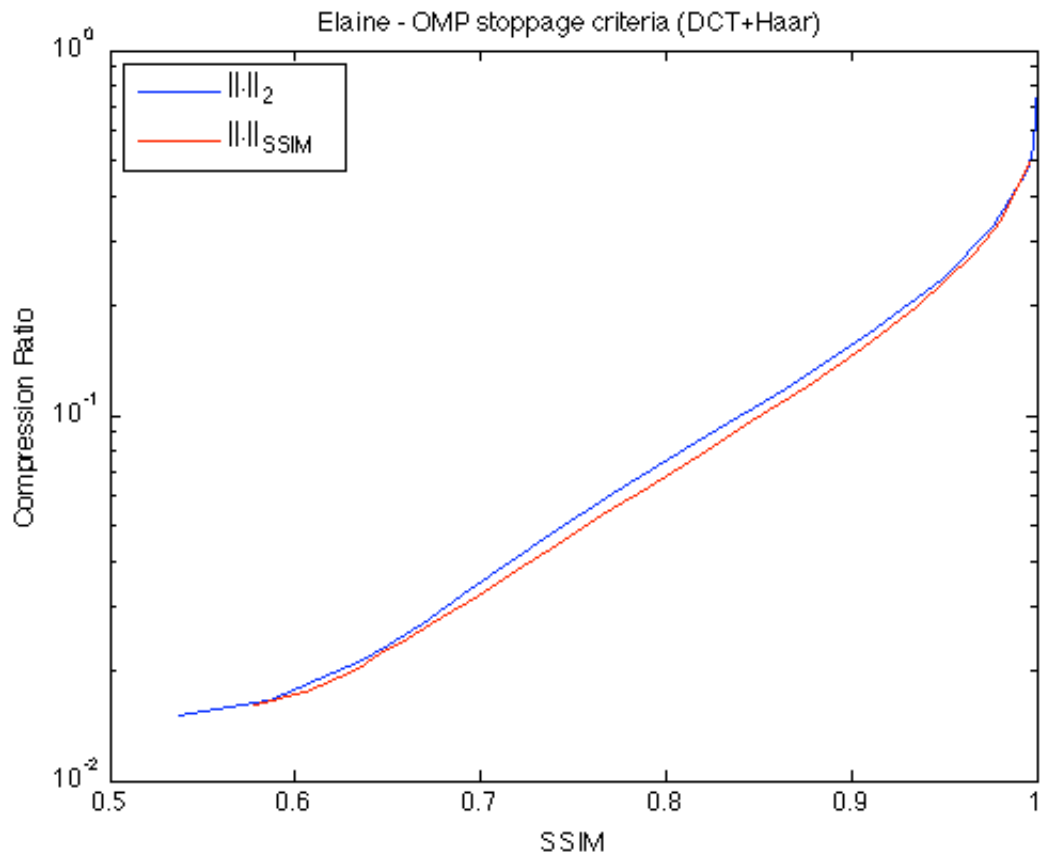
Error Comparison: Barbara



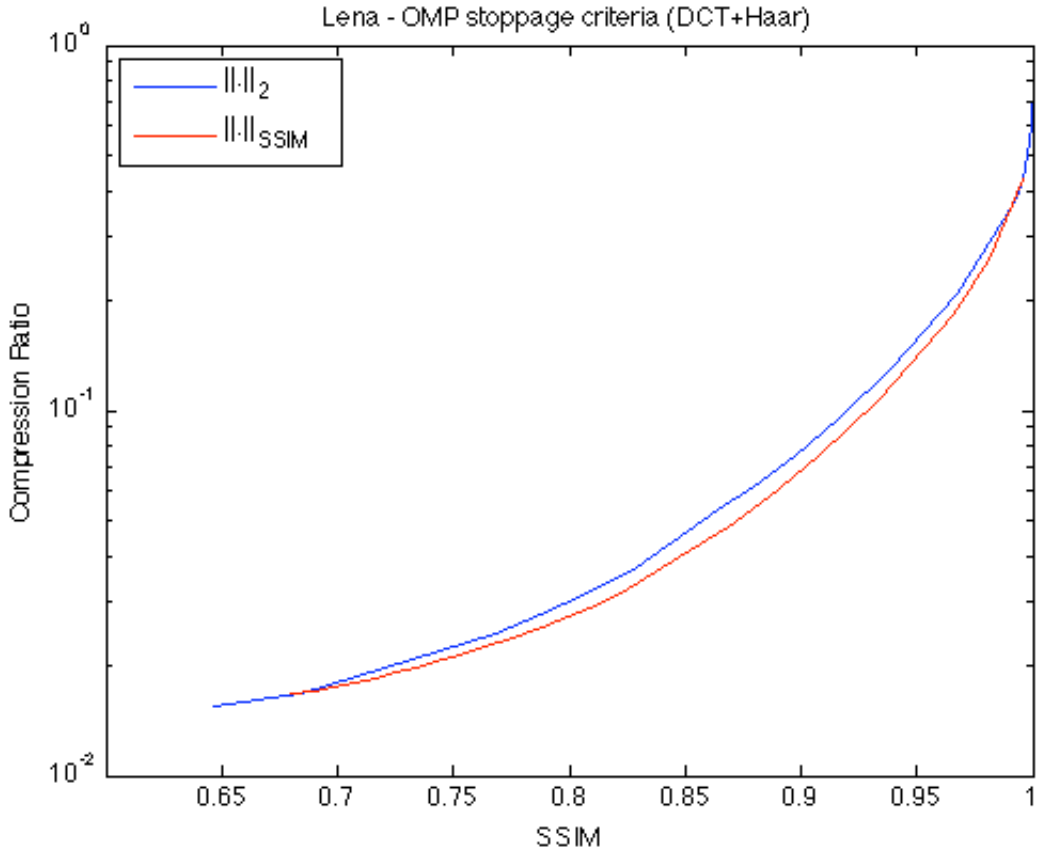
Error Comparison: Boat



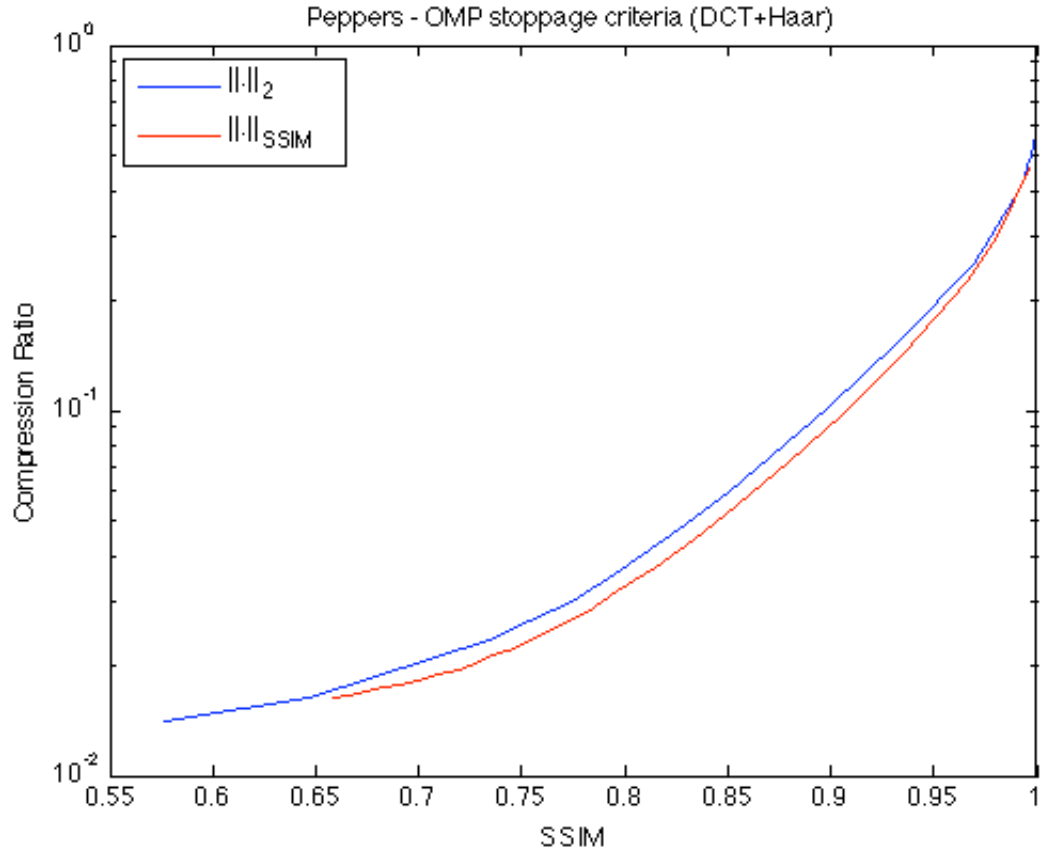
Error Comparison: Elaine



Error Comparison: Lena

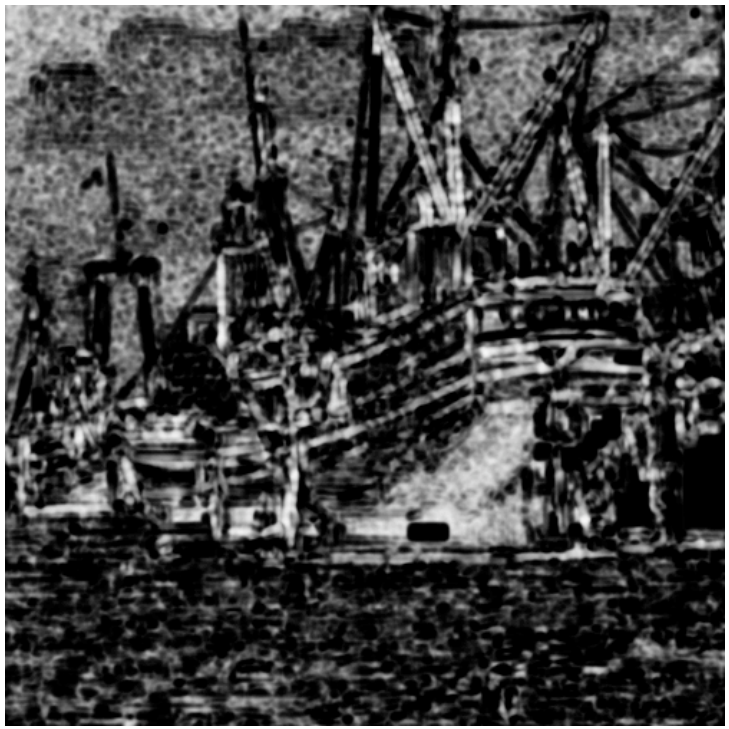


Error Comparison: Peppers



Visual overview: Boat

ε = 200, c = 25
PSNR = 25.2711
MSSIM = 0.6006
Comp. Ratio = 0.0217
Termination: $||.||_2$

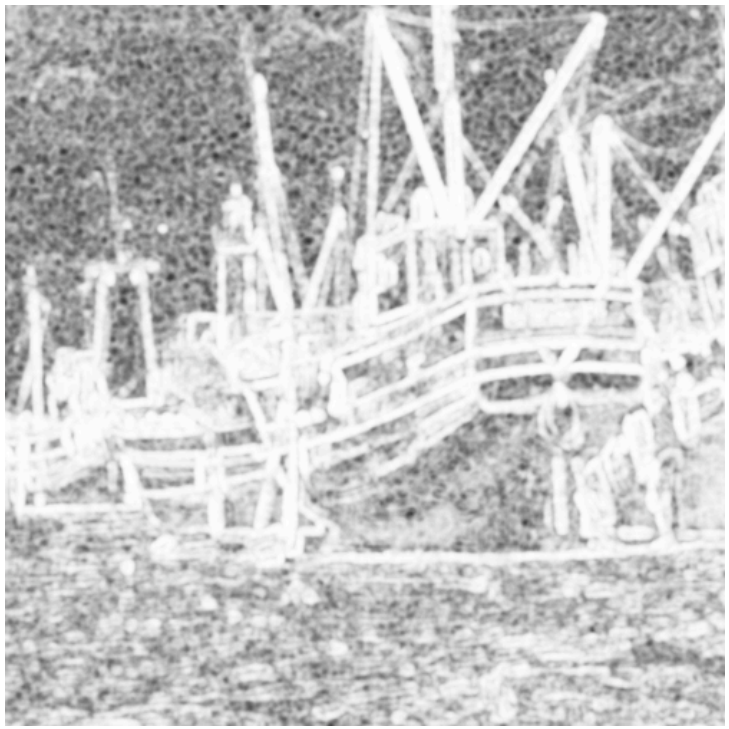


Visual overview: Boat

ε = 64, c = 8
PSNR = 31.7332
MSSIM = 0.8222
Comp. Ratio = 0.0710
Termination: $||.||_2$

Visual overview: Boat

ε = 32, c = 4
PSNR = 36.6020
MSSIM = 0.9214
Comp. Ratio = 0.1608
Termination: $||.||_2$

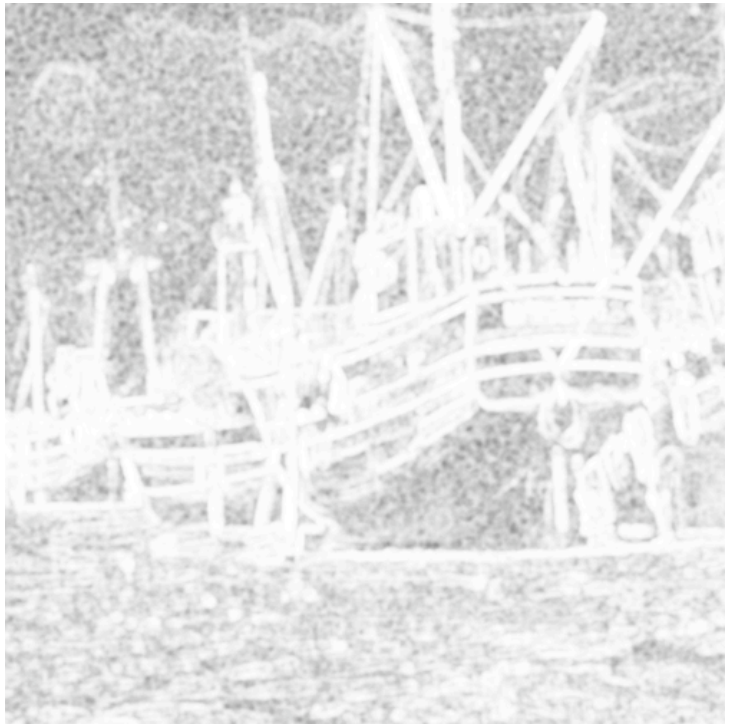


Visual overview: Boat

ε = 0.92
PSNR = 34.1405
MSSIM = 0.9355
Comp. Ratio = 0.1595
Termination: $||.||_{ssim}$

Visual overview: Boat

ε = 20, c = 2.5
PSNR = 40.4636
MSSIM = 0.9668
Comp. Ratio = 0.2628
Termination: $||.||_2$



Visual overview: Barbara

ε = 32, c = 4
PSNR = 36.9952
MSSIM = 0.9447
Comp. Ratio = 0.1863
Termination: $||.||_2$

Visual overview: Barbara

ε = 0.94
PSNR = 32.1482
MSSIM = 0.9466
Comp. Ratio = 0.1539
Termination: $||.||_{ssim}$

Future Work

-Compression encoding, how to?

-From frame theory perspective, what can we say?

-Can we do better than Haar?

-Uncertainty Principle, what is its role?

References

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Review, 51 (2009), pp. 34–81.

[2] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing, 24 (1995), pp. 227-234.

[3] G. W. Stewart, **Introduction to Matrix Computations**, Academic Press, 1973.

[4] T. Strohmer and R. W. Heath, *Grassmanian frames with applications to coding and communication*, Appl. Comput. Harmon. Anal., 14 (2004), pp. 257-275.

[5] D. S. Taubman and M. W. Mercellin, **JPEG 2000: Image Compression Fundamentals, Standards and Practice**, Kluwer Academic Publishers, 2001.

[6] G. K. Wallace, *The JPEG still picture compression standard*, Communications of the ACM, 34 (1991), pp. 30-44.

[7] S. Mallat, **A Wavelet Tour of Signal Processing**, Academic Press, 1998.

[8] Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Transactions on Image Processing , vol.13, no.4 pp. 600- 612, April 2004.
https://ece.uwaterloo.ca/~z70wang/research/ssim/index.html