DECODING REAL-FIELD CODES BY AN ITERATIVE EXPECTATION-MAXIMIZATION (EM) ALGORITHM

*H. Zayyani, M. Babaie-Zadeh**

Sharif University of Technology
Department Of Electrical Engineering
Tehran,Iran

C. Jutten

Institut National Polytechnique de Grenoble (INPG)
Labratoire des Images et de Signaux (LIS)
Grenoble,France

ABSTRACT

In this paper, a new approach for decoding real-field codes based on finding sparse solutions of underdetermined linear systems is proposed. This algorithm iteratively estimates the positions and the amplitudes of the sparse errors (or noise impulses) using an Expectation-Maximization (EM) algorithm. Iterative estimation of amplitudes is done in the Expectation step (E-step), while iterative estimation of error positions is done in the Maximization step (M-step). Simulation results show 1-2 dB improvement over Linear Programming (LP) which has been previously used for error correction.

Index Terms— Error correcting codes, sparse component analysis, sparse decomposition, impulsive noise cancelation

1. INTRODUCTION

This paper suggests a novel decoding method to recover the real-valued messages encoded by a real-valued generator matrix and corrupted by an impulsive noise. The input real-valued vector $\mathbf{s} \in \mathbb{R}^n$ is encoded with a generator matrix $\mathbf{G}$ which is a $m \times n$ matrix ($m > n$ to add redundancy to the input vector). The channel is a memoryless channel and can be considered as an impulsive noise added to the encoded signal. This noise which can be nominated as error in parallelism with the coding literature, is a $m \times 1$ vector. The mathematical model of our problem is:

$$\mathbf{y} = \mathbf{G}\mathbf{s} + \mathbf{e} \tag{1}$$

The main problem is to recover the input vector $\mathbf{s}$ from the vector $\mathbf{y}$ and with knowledge of the matrix $\mathbf{G}$. The problem has the flavor of error correcting problems in the real- field domain. We may think $\mathbf{G}$ as a linear code matrix with its columns as codewords. No special assumption is made about the generator matrix $\mathbf{G}$. This general selection of the generator matrix enables our coding and decoding scheme to have a ciphering ability, i.e. without knowing the generator matrix, the recovery of the input vector is difficult or at least may need a step to estimate the generator matrix which complicates the decoding process.

One of the most commonly used error correction techniques in the real-field is to use DFT-based codes, which the codewords is generated by padding zeros in the DFT domain as the redundancy of the linear code [1], [2]. The oversampled filter banks can also be considered as error correcting codes [3]. Moreover, real-valued BCH codes are used as joint source-channel coding specially in the image transition [4].

As will be seen in Section 2, applying the parity check matrix to (1) leads to an underdetermined system of linear equations. Solving this linear system of equations for its sparse solution, are the same as the problem of Sparse Component Analysis (SCA) which is recently focussed by many researchers in the context of Blind Source Separation (BSS) [5], [6], [7]. We use this framework as a tool for the error correction in real-field codes. In [8] a Linear Programming (LP) method has been used for decoding the real-field error correcting problem. It has been also proved that under suitable constraints on the coding matrix $\mathbf{G}$, the input $\mathbf{s}$ is the unique solution to the ℓ^1-minimization problem ($\min ||\mathbf{y} - \mathbf{G}\mathbf{s}||_1$), provided that the support (where the errors are nonzero) of the vector of errors $\mathbf{e}$ is not too large. This minimization can be implemented by a linear programming method as a convex optimization problem.

In this paper, we use a Bernoulli-Gaussian (BG) noise to consider the background noise in addition to the impulse noises (or errors). This model of noise with some changes has been also used in [3]. For the decoding step, we use an iterative estimation of position and amplitude of errors which is based on Maximum A Posteriori (MAP) estimation of these variables. This leads to an iterative Expectation-Maximization (EM) method.

2. SYSTEM MODEL

Our system model (1) shows the relationship between the input vector ($\mathbf{s}$) and the observed vector ($\mathbf{y}$). The error vector is considered as an impulsive noise plus Gaussian noise.

*This work has been partially funded by Sharif University of Technology, Advanced Communication Research Institute (ACRI), ISMO and Iran NSF (INSF).

The impulsive noise models the errors due to communication modulation and demodulation in actual systems and the Gaussian noise models the background noise or quantization noise. But in our paper, we consider the Bernoulli-Gaussian (BG) noise to consider the two noises in one variable. So the probability density of the error variable (e_i) is:

$$p(e_i) = pN(0, \sigma^2_{\text{off}}) + (1-p)N(0, \sigma^2_{\text{on}}) \tag{2}$$

where the probability p is the probability that the value of noise is not impulsive (we say it is inactive) and it is modeled by a Gaussian with a small variance (σ^2_{off}) and the $(1-p)$ is the probability of the value of noise being impulsive (we say it is active) and in this case, it is modeled by a Gaussian distribution with a large variance ($\sigma^2_{\text{on}} \gg \sigma^2_{\text{off}}$). The probability $(1-p)$ can be considered as the symbol error probability (p_e) which the symbol here means a real number. This probability is assumed to be small compared to one. So the probability p that the noise be inactive can be assumed to be near one ($p \approx 1$).

To recover the input vector **s**, it is sufficient to recover the error vector **e**, because the knowledge of the vector **e** gives the value of **Gs**, and consequently, **s**, since the matrix **G** has more rows than columns and assumed to be full rank [8]. Therefore, the pseudo inverse of **G** can recover the input vector **s** from overcomplete mixtures of it ($\mathbf{x} = \mathbf{Gs}$). This can be written as:

$$\hat{\mathbf{s}} = \mathbf{G}^{\dagger}\mathbf{x} = \mathbf{G}^{\dagger}(\mathbf{y} - \mathbf{e}) \tag{3}$$

To reconstruct the error vector **e**, we can define a parity check matrix **H** (in parallel with coding theory) which annihilates **G** from left, that is $\mathbf{HG} = \mathbf{0}$. To obtain matrix **H** from **G**, we consider the projection operator over **G** which has the following form:

$$\Pi^{\perp}_{\mathbf{G}} = \mathbf{I}_{m\times m} - \mathbf{G}(\mathbf{G}'\mathbf{G})^{-1}\mathbf{G}' \tag{4}$$

This projection operator is a $m \times m$ matrix that the rows are orthogonal to each column of **G**. To omit the redundancy of the equations, we only choose the $m-n$ rows of this projection operator for the parity check matrix ($\mathbf{H} = \Pi^{\perp}_{\mathbf{G}}(1 : m-n, :)$). Then applying this parity check matrix to (1), we can obtain the syndrome $\tilde{\mathbf{y}}$ as:

$$\tilde{\mathbf{y}} = \mathbf{Hy} = \mathbf{H}(\mathbf{Gs} + \mathbf{e}) = \mathbf{HGs} + \mathbf{He} \tag{5}$$

Therefore, the parity check equations lead to:

$$\tilde{\mathbf{y}} = \mathbf{He} \tag{6}$$

where this equation is an underdetermined system of linear equations with m unknowns e_i and $(m-n)$ observed variables $\tilde{y}_i$ as the syndrome elements. This equation has a sparse solution **e** which only a few of its samples are active (impulsive) and the remaining are inactive (background small Gaussian noise).

In our approach, for recovering the sparse error vector, we should find the position of the active errors (impulses) and also the amplitudes of the errors. To do this, we define an error activity vector $\mathbf{q} = (q_1, q_2, ..., q_m)^T$ as:

$$q_i = \begin{cases} 1 & \text{if } e_i \text{ is active (impulsive)} \\ 0 & \text{if } e_i \text{ is inactive} \end{cases} \tag{7}$$

3. OUR ITERATIVE EM METHOD

In our algorithm, we should estimate the $m \times 1$ error vector **e** from the syndrome vector $\tilde{\mathbf{y}}$ in (6). The main idea in our algorithm is that we iteratively estimate the error activity vector and the error amplitudes and also the parameters. In [7], we used a similar method for SCA. Here, we generalize the spiky model to the BG-model and the parameter estimation is done iteratively.

In the first step, an error activity vector $\hat{\mathbf{q}}$ is assumed and the MAP estimation of error vector **e** is computed. Since **q** is known, the vector **e** has a Gaussian distribution and its MAP estimation is the same as its Linear Least Square (LLS) estimation given by [9]:

$$\hat{\mathbf{e}}_{\text{MAP}} = E(\mathbf{e}|\tilde{\mathbf{y}}, \hat{\mathbf{q}}) = \hat{\mathbf{e}}_{\text{LLS}} = E(\mathbf{e}\tilde{\mathbf{y}}'|\hat{\mathbf{q}})E(\tilde{\mathbf{y}}\tilde{\mathbf{y}}'|\hat{\mathbf{q}})^{-1}\tilde{\mathbf{y}} \tag{8}$$

This step can be nominated as Expectation step or Estimation step (E-step). We use the following equation to consider the reconstruction error:

$$\tilde{\mathbf{y}} = \mathbf{H}\hat{\mathbf{e}} + \mathbf{v} \tag{9}$$

where the variable $\mathbf{v} = \mathbf{H}(\mathbf{e} - \hat{\mathbf{e}})$ is the reconstruction error and is assumed to have a Gaussian distribution with a variance σ^2_v. To simplify (8), we first compute the $E(\mathbf{e}\tilde{\mathbf{y}}'|\hat{\mathbf{q}})$ term. This is equal to:

$$E(\mathbf{e}\tilde{\mathbf{y}}'|\hat{\mathbf{q}}) = E(\mathbf{e}\hat{\mathbf{e}}'|\hat{\mathbf{q}})\mathbf{H}' + E(\mathbf{e}\hat{\mathbf{v}}'|\hat{\mathbf{q}}) \tag{10}$$

The first term $E(\mathbf{e}\hat{\mathbf{e}}'|\hat{\mathbf{q}})$ in (10) is equal to $E((\mathbf{e} - \hat{\mathbf{e}})\hat{\mathbf{e}}'|\hat{\mathbf{q}}) + E(\hat{\mathbf{e}}\hat{\mathbf{e}}'|\hat{\mathbf{q}}) = 0 + E(\hat{\mathbf{e}}\hat{\mathbf{e}}'|\hat{\mathbf{q}})$ due to orthogonality principle of LLS estimation. The term $E(\hat{\mathbf{e}}\hat{\mathbf{e}}'|\hat{\mathbf{q}})$ is the conditional covariance of the $\hat{\mathbf{e}}$ and is a diagonal matrix with elements σ^2_{on} for the active errors and σ^2_{off} for the inactive errors. So this matrix can be simplified as:

$$\mathbf{V}_q = E(\hat{\mathbf{e}}\hat{\mathbf{e}}'|\hat{\mathbf{q}}) = \sigma^2_{\text{off}}\mathbf{I} + (\sigma^2_{\text{on}} - \sigma^2_{\text{off}})\hat{\mathbf{Q}} \tag{11}$$

where $\mathbf{Q} = \text{diag}(\mathbf{q})$ is the diagonal matrix of the error activity vector. So we can write $E(\mathbf{e}\tilde{\mathbf{y}}'|\hat{\mathbf{q}}) = \mathbf{V}_q\mathbf{H}'$. After some similar simplifications, we can compute $E(\tilde{\mathbf{y}}\tilde{\mathbf{y}}'|\hat{\mathbf{q}}) = \sigma^2_v\mathbf{I} + \mathbf{H}\mathbf{V}_q\mathbf{H}'$. So our E-step of algorithm is:

$$\text{E-step} : \hat{\mathbf{e}}_{\text{MAP}} = (\mathbf{V}_q\mathbf{H}')(\sigma^2_v\mathbf{I} + \mathbf{H}\mathbf{V}_q\mathbf{H}')^{-1}\tilde{\mathbf{y}} \tag{12}$$

In the second step, we estimate **q** based on the known $\hat{\mathbf{e}}$ and the syndrome $\tilde{\mathbf{y}}$:

$$\text{M-step} : \hat{\mathbf{q}} = \underset{\mathbf{q}}{\text{argmax}}\, L(\mathbf{q}) = \log p(\mathbf{q})p(\tilde{\mathbf{y}}|\mathbf{q}, \hat{\mathbf{e}}) \tag{13}$$

This maximization step should be done over discrete space with 2^m element which is intractable for large m's. So, an efficient method for doing this is a necessity. To use efficient optimization methods in continuous space, we convert the discrete variables q_i to a continuous variable with a mixture of two Gaussian with sufficiently small variances around 0, 1. This approximation can be shown mathematically as:

$$p(q_i) \approx pN(0, \sigma_0^2) + (1-p)N(1, \sigma_0^2) \tag{14}$$

Also, to avoid falling into the local maxima of (13), a gradually decreasing variance of this approximation is used in different iterations. After converting the function in (13), we should use an optimization method such as steepest descent. To compute the log posterior which is defined in (13), we compute the two terms, the prior probability $p(\mathbf{q})$ and the likelihood $p(\tilde{\mathbf{y}}|\mathbf{q}, \hat{\mathbf{e}})$ in (15) and (16) respectively.

$$p(\mathbf{q}) = \prod_{i=1}^{m} p(q_i) \propto \prod_{i=1}^{m} [p \exp(\frac{-q_i^2}{2\sigma_0^2}) + (1-p)\exp(\frac{-(q_i-1)^2}{2\sigma_0^2})] \tag{15}$$

$$p(\tilde{\mathbf{y}}|\mathbf{q}, \hat{\mathbf{e}}) = p_{\mathbf{v}}(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}}) = (2\pi\sigma_v^2)^{\frac{-m}{2}} \exp(\frac{-1}{2\sigma_v^2}(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})'(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})) \tag{16}$$

So the log posterior is simplified as:

$$L(\mathbf{q}) \propto \sum_{i=1}^{m} \log(p(q_i)) + (\frac{-1}{2\sigma_v^2})(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})'(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}}) \tag{17}$$

In the appendix, we show that the steepest-ascent algorithm which is $\mathbf{q}_{k+1} = \mathbf{q}_k + \mu \frac{\partial L(\mathbf{q})}{\partial \mathbf{q}}$ and it is used for the M-step is:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \frac{\mu}{\sigma_0^2}\mathbf{g}(\mathbf{q}_k) + \frac{\mu}{\sigma_v^2}\mathrm{diag}(\mathbf{H}')(\mathbf{H}\hat{\mathbf{e}} - \tilde{\mathbf{y}}).\hat{\mathbf{e}} \tag{18}$$

where $\mathbf{g}(\mathbf{q})$ is defined in the appendix. In the successive iterations, we gradually decrease the variance of σ_0 in the form $\sigma_0^{(i)} = \alpha\sigma_0^{(i-1)}$ where α is selected between 0.6 and 1. This parameter affects the speed of convergence of our algorithm. Also, the step-size μ should be decreasing, i.e. for smaller σ's, smaller μ's should be applied. This is because for smaller variances, our function under maximization is more fluctuating. In fact, for having equal (i.e.proportional) steps of the steepest-ascent algorithm in (18), the step-size should be in the form of $\mu = \mu^{(0)}(\sigma_0^{(i)})^2$.

The initialization of our algorithm is the minimum ℓ^2norm solution. For the initialization of our parameters, we assume that the elements of the generator matrix has a uniform distribution between [-1,1] and also the columns of the matrix is scaled to have unit norm. We also assume that $\sum_{j=1}^{m-n} h_{ji}^2 \approx c = (m-n)E(h_{ji}^2)$. From $\tilde{y}_j = \sum_{i=1}^{m} h_{ji}e_i$ we can write:

$$E(\tilde{y}_j^2) = mE(h_{ji}^2)E(e_i^2) \tag{19}$$

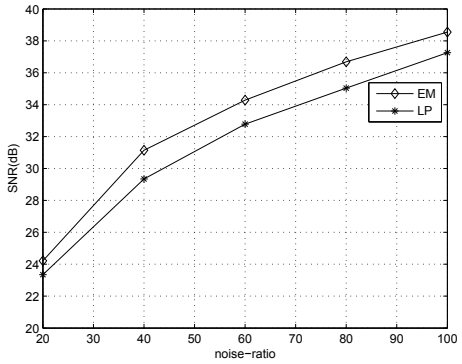

Fig. 1. The simulation results for various values of noise ratios, where $m = 200$, $n = 100$ and $p = .9$.

Since $E(e_i^2) = (1-p)\sigma_{\text{on}}^2 + p\sigma_{\text{off}}^2 \approx (1-p)\sigma_{\text{on}}^2$ and $E(h_{ji}^2) = \frac{c}{m-n}$, and from (19) we can estimate σ_{on} as:

$$\hat{\sigma}_{\text{on}} = \sqrt{\frac{(m-n)E(\tilde{y}_j^2)}{mc(1-\hat{p})}} \tag{20}$$

$$\hat{p} = \frac{||\mathbf{q}||_0}{m} \tag{21}$$

$$\hat{\sigma}_{\text{off}} = \sqrt{\text{var}(\mathbf{e}_{\text{inactive}})} \tag{22}$$

$$\hat{\sigma}_v = \sqrt{||\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}}||_2} \tag{23}$$

4. EXPERIMENTS

This section investigates the result of our EM algorithm to reconstruct the original signal $\mathbf{s}$ from corrupted encoded data by an impulsive noise (with BG distribution) in comparison to Linear Programming (LP). The LP decoding algorithm was simulated by ℓ^1-magic package [10].

The sources are assumed to be uniform in [-1,1]. The sizes are assumed to be $m = 200$ and $n = 100$. The elements of the generator matrix $\mathbf{G}$ is generated by uniform distribution in [-1,1]. Then the columns of this matrix are normalized to have unit norm. The error vector $\mathbf{e}$ is generated by a BG distribution described in Section 2 with parameters $\sigma_{\text{on}} = E(|\mathbf{Gs}|)$, $\sigma_{\text{off}} = \frac{\sigma_{\text{on}}}{40}$ and $p = 0.9$. The experiments was repeated 100 times for new random input signals, generator matrices and error vectors. The measure of performance is the SNR between the input signal vector $\mathbf{s}$ and recovered input signal vector $\hat{\mathbf{s}}$ defined as $\text{SNR} = 10\log\frac{||\mathbf{s}||_2}{||\mathbf{s}-\hat{\mathbf{s}}||_2}$. In our experiment, various strengths of background noise were examined. We define the noise ratio as $R_e = \frac{\sigma_{\text{on}}}{\sigma_{\text{off}}}$. By this definition, and for a fixed σ_{on}, we can select $\sigma_{\text{off}} = \frac{\sigma_{\text{on}}}{R_e}$. The results for the various noise ratios are depicted in Fig.1. The results show 1-2 dB improvement of our algorithm in comparison to LP method.

Although, the CPU time is not an exact measure of complexity, it can give us a rough estimation of it, and we compare our algorithm with LP using this measure. Our simulations were performed in MATLAB 7.0 environment using an Intel 2.40 GHz processor with 512 MB of RAM and under Microsoft Windows XP operating system. The simulation time of 100 experiments is computed for each algorithm. This is 57.7 seconds for the LP decoding algorithm, while it is 36.8 seconds for our EM algorithm.

5. CONCLUSIONS

In this paper, we introduced a new EM algorithm for decoding real-field codes in presence of impulsive noise (or errors). This algorithm iteratively estimates the positions and amplitudes of the errors. Our simulations show that our EM algorithm is slightly (1-2 dB) better than the LP method when the background noise is present. Moreover, it is slightly faster than the LP method.

6. APPENDIX

From (17), we have:

$$\frac{\partial L(\mathbf{q})}{\partial \mathbf{q}} \propto \frac{\partial}{\partial \mathbf{q}} \sum_{i=1}^{m} \log(p(q_i)) - \frac{1}{2\sigma_v^2} \frac{\partial}{\partial \mathbf{q}} (\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})'(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}}) \tag{24}$$

we define $\mathbf{g}(\mathbf{q}) = -\sigma_0^2 \frac{\partial}{\partial \mathbf{q}} \sum_{i=1}^{m} \log(p(q_i))$ and $\mathbf{n}(\mathbf{q}) = (\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})'(\tilde{\mathbf{y}} - \mathbf{H}\hat{\mathbf{e}})$. With these definitions the scalar function $g(q_i)$ and the $\mathbf{n}(\mathbf{q})$ (with omitting the constant terms) can be computed as:

$$g(q_i) = \frac{pq_i \exp(\frac{-q_i^2}{2\sigma_0^2}) + (1-p)(q_i - 1)\exp(\frac{-(q_i-1)^2}{2\sigma_0^2})}{p\exp(\frac{-q_i^2}{2\sigma_0^2}) + (1-p)\exp(\frac{-(q_i-1)^2}{2\sigma_0^2})} \tag{25}$$

$$\mathbf{n}(\mathbf{q}) = -2\tilde{\mathbf{y}}'\mathbf{H}\hat{\mathbf{e}} + \hat{\mathbf{e}}'\mathbf{H}'\mathbf{H}\hat{\mathbf{e}} \tag{26}$$

with definitions $\mathbf{C} \triangleq \mathbf{H}'\mathbf{H}$ and $\mathbf{n_1}(\mathbf{q}) \triangleq -2\tilde{\mathbf{y}}'\mathbf{H}\hat{\mathbf{e}}$ and $\mathbf{n_2}(\mathbf{q}) \triangleq \hat{\mathbf{e}}'\mathbf{H}'\mathbf{H}\hat{\mathbf{e}}$ and with approximation $\hat{\mathbf{e}} \simeq \mathbf{Q}\hat{\mathbf{r}}$ we can write:

$$\frac{\partial n_1(\mathbf{q})}{\partial \mathbf{q}} = \mathrm{diag}(-2\tilde{\mathbf{y}}'\mathbf{H}).\hat{\mathbf{r}} \tag{27}$$

If we define $\mathbf{W} \triangleq \mathbf{Q}\hat{\mathbf{r}}$ ($m \times 1$ vector) then $\mathbf{n}_2(\mathbf{q}) = \mathbf{W}'\mathbf{C}\mathbf{W}$ and so we have:

$$\frac{\partial \mathbf{n}_2(\mathbf{q})}{\partial q_i} = \sum_{j=1}^{m} \frac{\partial \mathbf{n}_2(\mathbf{q})}{\partial W_j} \frac{\partial W_j}{\partial q_i} \tag{28}$$

From the vector derivatives, we have $\frac{\partial \mathbf{n}_2(\mathbf{q})}{\partial \mathbf{W}} = 2\mathbf{C}\mathbf{W} = \mathbf{d}$. Also from the definition of $\mathbf{W}$ we have $\frac{\partial W_j}{\partial q_i} = \hat{r_i}\delta_{ij}$. So (28) is converted to $\frac{\partial \mathbf{n}_2(\mathbf{q})}{\partial q_i} = \sum_{j=1}^{m} d_j \hat{r_i}\delta_{ij} = \hat{r_i} d_i$. Consequently, the vector form of (28) is equal to:

$$\frac{\partial \mathbf{n}_2(\mathbf{q})}{\partial \mathbf{q}} = \mathrm{diag}(\mathbf{d}).\hat{\mathbf{r}} \tag{29}$$

From (27) and (29) and $\mathbf{n}(\mathbf{q}) = \mathbf{n}_1(\mathbf{q}) + \mathbf{n}_2(\mathbf{q})$ and definitions of vectors $\mathbf{d}$ and $\mathbf{C}$, we can write:

$$\frac{\partial \mathbf{n}(\mathbf{q})}{\partial \mathbf{q}} = 2\mathrm{diag}(\mathbf{H}'\mathbf{H}\mathbf{Q}\hat{\mathbf{r}} - \mathbf{H}'\tilde{\mathbf{y}}).\hat{\mathbf{r}} \tag{30}$$

Finally, replacing $\hat{\mathbf{e}} \simeq \mathbf{Q}\hat{\mathbf{r}}$ in (30) and then in (24) with some manipulation leads to the steepest-ascent iteration in (18).

7. REFERENCES

[1] J.K. Wolf, "Redundancy, the discrete fourier transform, and impulse noise cancellation," *IEEE Trans. Comm.*, vol. COM 31, pp. 458–461, March 1983.

[2] F. Marvasti, M. Hassan, M. Echhart, and S. Talebi, "Efficient algorithms for burst error recovery using fft and other transform kernels," *IEEE Trans. Signal. Processing*, vol. 47, pp. 1065–1075, April 1999.

[3] F. Labeau, J.C. Chiang, M. Kieffer, P. Duhamel, L. Vandendorpe, and B. Macq, "Oversampled filter banks as error correcting codes: Theory and impulse noise cancellation," *IEEE Trans. Signal. Processing*, vol. 53, pp. 4619–4630, December 2005.

[4] A. Gabay, O. Rioul, and P. Duhamel, "Joint source-channel coding using structured oversampled filter banks applied to image transmission," *in proceeding ICASSP01*, vol. 4, pp. 2581–2584, 2001.

[5] M. Zibulevsky and B.A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computation*, vol. 13, pp. 863–882, April 2001.

[6] R. Gribonval and S. Lesage, "A survey of sparse component analysis for blind source separation: principles, perspectives, and new challanges," *in proceeding ESSAN'06*, pp. 323–330, 2006.

[7] H. Zayyani, M. Babaie-Zadeh, G.H. Mohimani, and C. Jutten, "Sparse component analysis in presense of noise using an iterative EM-MAP algorithm," *in proceeding ICA2007*, September 2007.

[8] E.J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, pp. 4203–4215, December 2005.

[9] B.D. Anderson and J. B. Moor, *Optimal filtering*, Prentice Hall, 1979.

[10] E.J. Candes and J. Romberg, "ℓ^1magic: recovery of sparse signals via convex programming," 2005, URL:`http://www.acm.caltech.edu/l1magic/downloads/l1-magic.pdf`.