FAST COMPUTATION OF THE L_1-PRINCIPAL COMPONENT OF REAL-VALUED DATA

Sandipan Kundu, Panos P. Markopoulos, and Dimitris A. Pados†

Department of Electrical Engineering, State University of New York at Buffalo, NY 14260
E-mail: {skundu, pmarkopo, pados}@buffalo.edu

ABSTRACT

Recently, Markopoulos et al. [1], [2] presented an optimal algorithm that computes the L_1 maximum-projection principal component of any set of N real-valued data vectors of dimension D with complexity polynomial in N, $\mathcal{O}(N^D)$. Still, moderate to high values of the data dimension D and/or data record size N may render the optimal algorithm unsuitable for practical implementation due to its exponential in D complexity. In this paper, we present for the first time in the literature a fast greedy single-bit-flipping conditionally optimal iterative algorithm for the computation of the L_1 principal component with complexity $\mathcal{O}(N^3)$. Detailed numerical studies are carried out demonstrating the effectiveness of the developed algorithm with applications to the general field of data dimensionality reduction and direction-of-arrival estimation.

Index Terms— Dimensionality reduction, direction-of-arrival estimation, eigen-decomposition, L_1 and L_2 principal component, machine learning, outlier resistance, subspace signal processing.

1. INTRODUCTION

Conventional -and vastly successful over the years- subspace signal processing theory and practice relies on L_2-norm based singular-value decomposition (SVD) of the observed data or, equivalently, eigen-value decomposition (EVD) of the associated data autocovariance matrix. The SVD solution traces its origin to the fundamental problem of L_2-norm low-rank matrix approximation [3], [4].

It is well known, however, that L_2-norm subspace decomposition is sensitive to the presence of outliers in the given data set (erroneous measurements that are way out of line in value compared to correctly measured nominal data). On the other hand, absolute-value (L_1) calculation puts significantly less emphasis on extreme errors than squared-value (L_2) expressions. Motivated by this notion, recently there has been an arguably small but growing interest in pursuing L_1-norm based approaches to deal with the problem of faulty measurements in training in subspace signal processing and principal-components design [5]-[18].

Given any data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$ of N signal samples of dimension D, the computation of the L_1 principal component of the data by maximum L_1-norm projection is NP-hard jointly in $N,\ D$ as shown in [1], [2]. However, when the signal dimension D is fixed with $D \leq N$ –which is arguably of more interest in signal processing applications– the L_1 principal component can be computed in polynomial time with complexity $\mathcal{O}(N^{\mathrm{rank}(\mathbf{X})})$, $\mathrm{rank}(\mathbf{X}) \leq D$, as presented in [1], [2].

Still, when the data dimension D (and/or data record size N) is relatively high, the optimal algorithm may be unsuitable for practical implementation due to its exponential in D complexity. In this work, we present a new low-complexity sub-optimal algorithm for the computation of the L_1 principal component of any data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$. Convergence (stopping in finite number of steps) to a fixed point with complexity $\mathcal{O}(N^3)$ is established theoretically. Statistically (empirically only), globally optimal performance with frequency-of-occurrence one is observed. Numerical studies in this paper *(i)* compare the proposed algorithm against the optimal in computing the L_1 principal component of arbitrary data matrices and *(ii)* test the proposed algorithm with applications in the fields of dimensionality reduction and direction-of-arrival (DoA) estimation to demonstrate effectiveness of data representation and DoA estimation when the principal component is acquired from corrupted data.

2. EXISTING ALGORITHMS TO COMPUTE THE L_1-PRINCIPAL COMPONENT

The problem of computing the L_1 maximum-projection principal component of a real-valued data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$, $D \leq N$, can be mathematically formulated as the problem of finding

$$\mathbf{r}_{L_1} = \underset{\mathbf{r}\in\mathbb{R}^{D\times 1},\ \mathbf{r}^T\mathbf{r}=1}{\arg\max} \left\|\mathbf{r}^T\mathbf{X}\right\|_1. \tag{1}$$

Below, we review the current state of the art in optimal and sub-optimal algorithms for computing an L_1 principal component.

2.1. Optimal Algorithm

Recently, Markopoulos et al. [1], [2] presented the only known optimal algorithm for computing the L_1 principal component of $\mathbf{X}$. In [1], [2], it was shown that if

$$\mathbf{b}_{\text{opt}} = \underset{\mathbf{b}\in\{\pm 1\}^{N\times 1}}{\arg\max}\ \mathbf{b}^T\mathbf{X}^T\mathbf{X}\mathbf{b}, \tag{2}$$

†Corresponding author.

then the optimal L_1 principal component is given by

$$\mathbf{r}_{L_1} = \frac{\mathbf{X}\mathbf{b}_{\text{opt}}}{\|\mathbf{X}\mathbf{b}_{\text{opt}}\|_2}. \tag{3}$$

A direct way to solve (2) is to exhaustively search over all 2^N antipodal binary vectors of size $N \times 1$. Clearly, even for moderate values of N, finding the optimal solution via exhaustive search becomes quickly infeasible.

Indeed, the problem is NP-hard jointly in N, D [1], [2]. However, for fixed data dimension $D \leq N$ (a case of engineering interest), [1], [2] presented an optimal algorithm that solves (2) with polynomial complexity in the data record size, $\mathcal{O}(N^{\text{rank}(\mathbf{X})})$. Still, moderate to large values of $\text{rank}(\mathbf{X}) \leq D$ (and N) can easily make the polynomial algorithm unsuitable for practical implementation.

2.2. Sub-optimal Algorithms

In [5] and [7], the L_1 principal component was sub-optimally computed via coupled optimization or alternating maximization until convergence, respectively. Both algorithms can be seen to yield the same identical iteration as follows,

$$\mathbf{b}^{(i+1)} = \text{sgn}\left(\mathbf{X}^T\mathbf{X}\mathbf{b}^{(i)}\right), \quad i = 1, 2, \dots . \tag{4}$$

The above iteration does not guarantee convergence to the optimal L_1 principal component and suffers from noticeable performance degradation, as will be seen in our numerical studies section.

Hence, there is still need for a low-complexity algorithm (suitable for practical implementation) that can compute the optimal L_1 principal component for moderate to high dimensional data matrices $\mathbf{X}$ with high accuracy.

3. BIT-FLIPPING COMPUTATION OF THE L_1 PRINCIPAL COMPONENT

In this section, we develop a low-complexity algorithm to calculate the L_1 principal component of a given data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$ $(D \leq N)$. From (2), (3), we observe that the core challenge in computing the L_1 principal component lies in the efficient calculation of the binary vector that maximizes the quadratic form in (2). In that effort, we propose a new algorithm that finds the binary vector that maximizes the quadratic form in (2) (empirically with highest probability) and then computes the L_1 principal component through (3).

The proposed algorithm for finding a near-optimal solution to the problem in (2) has two steps: *(i)* We obtain an initial binary vector; and *(ii)* perform optimal iterative single-bit-flipping (SBF) to obtain a final binary vector. The developed iterative greedy SBF procedure updates any given binary vector to the best binary vector possible in quadratic value via single bit flipping per iteration. The idea is reminiscent of bit flipping algorithms in the channel coding literature [19].

Next, we describe in detail the SBF algorithm. Given a data matrix $\mathbf{X}$ and an initial binary vector $\mathbf{b}$, the SBF algorithm iteratively produces a sequence of binary vectors where each one differs from the immediately previous and following vector only in a single bit position. That is, the kth bit vector in the sequence is obtained by a single-bit-flip to the $(k-1)$th bit vector. At each iteration, we choose to flip the bit that results in the highest increase of the quadratic value in (2).

Now, we discuss how to efficiently identify the bit to flip in a binary vector $\mathbf{b}^k$ at the kth iteration step of the algorithm. The associated quadratic value can be algebraically written as

$${\mathbf{b}^k}^T\mathbf{X}^T\mathbf{X}\mathbf{b}^k = \text{Tr}(\mathbf{X}^T\mathbf{X}) + \sum_i 2b_i^k \left\{\sum_{j>i} b_j^k(\mathbf{X}^T\mathbf{X})_{i,j}\right\}. \tag{5}$$

In view of (5), changing the ith bit b_i^k in $\mathbf{b}^k$ will change the quadratic value by

$$\alpha_i^{(k)} \triangleq \pm 4\, b_i^k \left\{\sum_{j\neq i} b_j^k(\mathbf{X}^T\mathbf{X})_{i,j}\right\}. \tag{6}$$

Therefore, if (6) is negative, changing b_i^k to $-b_i^k$ will increase the quadratic value in (5) by $|\alpha_i^k|$. It is apparent that flipping the bit with the most negative contribution will offer the biggest increase in the quadratic value in (5). On the other hand, flipping a bit that corresponds to positive contribution term will decrease the objective value.

In the context of formal convergence analysis, from (6) we observe that an ith bit flip corresponding to $\alpha_i^k > 0$ will decrease the quadratic value in (2). Hence, if $\alpha_i^k \geq 0 \,\forall i \in \{1, \cdots, N\}$, there is no single bit flip that will increase the quadratic value and the SBF algorithm terminates. Knowing the termination condition for the SBF algorithm, the obvious question at this point is whether the termination condition will ever be met. We can answer the question in the affirmative by considering that *(i)* the binary quadratic form maximization in (2) has a finite upper bound and *(ii)* at every iteration of the SBF algorithm the quadratic value increases ensuring convergence of the SBF algorithm in a finite number of iterations.

Having described the SBF algorithm and established convergence in finite steps for any given initial binary vector $\mathbf{b}$, we focus now on initialization. We begin by looking at the following motivating example. Consider the case where the matrix $\mathbf{X}^T\mathbf{X}$ can be well approximated by a rank-1 matrix, i.e. by the L_2 principal component of $\mathbf{X}^T\mathbf{X}$. Mathematically, under rank-1 approximation we write

$$\mathbf{X}^T\mathbf{X} \simeq \lambda \mathbf{r}_{L_2}{\mathbf{r}_{L_2}}^T \triangleq \tilde{\mathbf{X}}$$

where $\mathbf{r}_{L_2} \in \mathbb{R}^N$ is the L_2 principal component of $\mathbf{X}^T\mathbf{X}$. Under rank-1 approximation, the optimization problem in (2) is solved by

$$\mathbf{b}_{\text{opt}} = \underset{\mathbf{b}\in\{\pm 1\}^N}{\arg\max}\ \mathbf{b}^T\tilde{\mathbf{X}}\mathbf{b} = \text{sgn}\Big(\tilde{\mathbf{X}}(:,i)\Big)\ \forall i = 1, \cdots, N.$$

Accordingly, we decide to initialize the SBF algorithm -in parallel or serially- to each of the $\bar{N} \leq N$ distinct binary vectors given by the sign of each column of matrix $\mathbf{X}^T\mathbf{X}$.

Once, upon convergence of the $\bar{N}$ parallel or serial searches, the final decision for the binary vector that solves (2) is obtained, we compute the L_1 principal component of $\mathbf{X}$ via the linear operation in (3). The complete details of the proposed algorithm in direct implementation form are given in Fig. 1.

Algorithm: L_1-Principal Calculation

Input: $\mathbf{X}_{D\times N}$ data matrix

$[\mathbf{B}]_{:,n} \leftarrow \text{single\_bit\_flipping}\left(\mathbf{X}, \text{sgn}([\mathbf{X}^T\mathbf{X}]_{:,n})\right) \forall n$

$c \leftarrow \text{argmax}_i \, [\mathbf{B}^T\mathbf{X}^T\mathbf{X}\mathbf{B}]_{i,i}$

Return: $\hat{\mathbf{r}}_{L_1} \leftarrow \mathbf{X}[\mathbf{B}]_{:,c}/\|\mathbf{X}[\mathbf{B}]_{:,c}\|_2$

Function: single_bit_flipping

Input: $\mathbf{X}_{D\times N}$ and $\mathbf{b} \in \{\pm 1\}^N$

Repeat:

$\alpha_i \leftarrow b_i \left(\sum_{j\neq i} b_j [\mathbf{X}^T\mathbf{X}]_{i,j}\right) \; \forall i \in \{1,\ldots,N\}$

$(v,p) \leftarrow \text{minimum}(\alpha)$

if $v < 0$, *then* $b_p \leftarrow -b_p$; *else*, EXIT

Return: $\mathbf{b}$

Fig. 1: Fast computation of the L_1 principal component via greedy single-bit-flipping (SBF).

4. COMPUTATIONAL COMPLEXITY

In the sequel, we find the computational complexity of the developed algorithm for the calculation of the L_1-principal component of an $\mathbf{X}_{D\times N}$ data matrix. Let $\bar{M}$ be the maximum number of steps required for the SBF algorithm to converge among all $\bar{N} \leq N$ initializations.

Given any one of the initial binary vectors $\mathbf{b}$ (sign of one column of $\mathbf{X}^T\mathbf{X}$), the computational cost for calculating $\alpha_j^{(1)}$ has complexity $\mathcal{O}(N^2)$. Finding the minimum among $\alpha_j^{(1)}$ is of complexity $\mathcal{O}(N-1)$. Hence, the dominant complexity of the first step of the SBF algorithm is $\mathcal{O}(N^2)$.

Assume that the i'th bit has been flipped in the first iteration of the SBF algorithm. The computational complexity to calculate $\alpha_j^{(2)} \, \forall j \neq i'$ (because $\alpha_{i'}^{(2)} > 0$) in the second step of the SBF algorithm is $\mathcal{O}(N-1)$. Calculating $\alpha_j^{(2)}$ and finding the minimum is $\mathcal{O}(N-1)$. Hence, the dominant complexity in the second step of the SBF algorithm is $\mathcal{O}(N-1)$.

Assume that the i''th bit has been flipped in the second iteration of the SBF algorithm. We can prove (the proof has to be omitted due to lack of space) that, if the ith bit in $\mathbf{b}^k$ is flipped in the $(k+1)$th iteration to obtain $\mathbf{b}^{k+1}$, then the ith bit may be flipped again only after the $(k+3)$th iteration. We conclude that the i' and i'' bit will not change at the third iteration of the SBF algorithm. Then, the computational complexity to calculate $\alpha_j^{(3)} \, \forall j \neq i', i''$ in the third step of the SBF algorithm is $\mathcal{O}(N-2)$. $\mathcal{O}(N-2)$ is also the cost to find the minimum among $\alpha_j^{(3)}$. Hence, the dominant complexity of the third step of the SBF algorithm is $\mathcal{O}(N-2)$.

Going one more step ahead, assume that the i'''th bit has been flipped in the third iteration of the SBF algorithm. The computational complexity to calculate $\alpha_j^{(4)} \, \forall j \neq i', i'', i'''$ in the fourth step of the SBF algorithm is $\mathcal{O}(N-3)$, $\mathcal{O}(N-2)$ to calculate α_j for $j = i'$, and $\mathcal{O}(N-2)$ to find the minimum. Hence, the dominant complexity of the fourth step of the SBF algorithm is $\mathcal{O}(N-2)$. The complexity of all the rest of the iterations until convergence remains $\mathcal{O}(N-2)$.

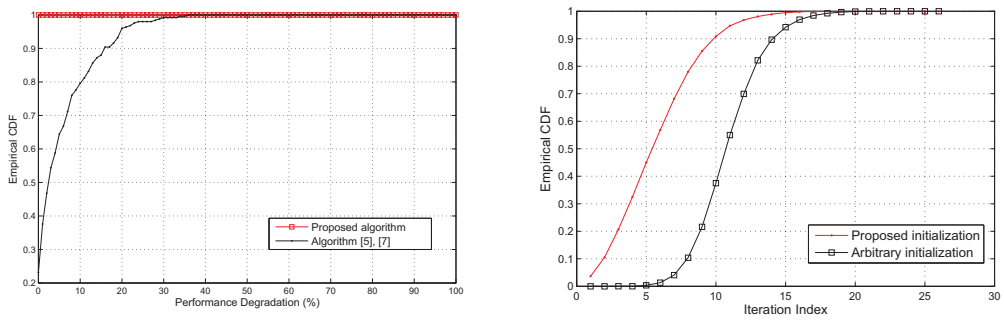
In light of the above analysis, the overall complexity of the complete SBF-based procedure is given by $\mathcal{O}(N^2) + \mathcal{O}(N-1) + \mathcal{O}((\bar{M}-2)(N-2))$. Per Section V, empirically always $\bar{M} \leq N$ or we simply bound the SBF algorithm at $\bar{M} = N$. Then, the total computational complexity for the Algorithm in Fig. 1 is $\mathcal{O}(\bar{N}N^2) < \mathcal{O}(N^3)$.

5. EXPERIMENTAL STUDIES

In this section, we first carry out a comparative experimental study on the computation of the L_1 principal component of arbitrary matrices by the proposed algorithm. We continue on with applications to dimensionality reduction and direction-of-arrival (DoA) estimation problems in the presence of erroneous outlier values/sporadic jammers.

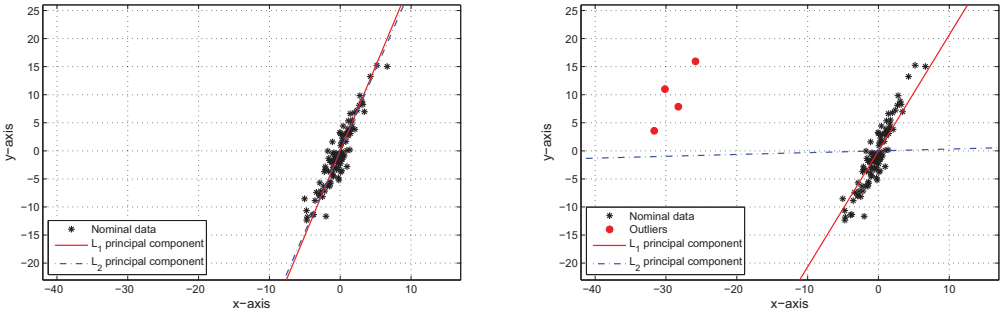
Experiment 1 - Numerical Evaluation. We generate arbitrary data matrices $\mathbf{X} \in \mathbb{R}^{5\times 25}$ filled in with independent Gaussian realizations of mean zero and variance ten. In Fig. 2(a), we evaluate and compare the proposed algorithm and the algorithm in [5] (equivalent to the algorithm in [7]) in terms of objective value degradation per the L_1 metric in (1). In particular, we plot the empirical cumulative distribution function (CDF) of the performance loss event in $\%$-scale (probability that degradation of less than $x\%$ is experienced). We conclude that the proposed algorithm finds the optimal L_1 principal component with empirical frequency of occurrence one. For the same set of generated data matrices, in Fig. 2(b) we evaluate empirically the CDF of the iteration termination index of the developed algorithm under the proposed and arbitrary initialization. As expected, the proposed algorithm on average converges quite faster when initialized with $\text{sgn}([\mathbf{X}^T\mathbf{X}]_{:,n})$ as opposed to arbitrary initialization.

Experiment 2 - Data Dimensionality Reduction. We generate a data-set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ of $N = 100$ two-dimensional observation points drawn from the nominal Gaussian distribution $\mathcal{N}\left(\mathbf{0}_2, \begin{bmatrix} 4 & 10 \\ 10 & 29 \end{bmatrix}\right)$. In Fig. 3(a), we plot the generated data points on the 2-D plain and calculate and plot the L_2 (by standard SVD) and L_1 (by the proposed algorithm) principal component of the data. We note the great similarity of the two principal components, L_1 and L_2. Then, we "contaminate" our nominal data with four outlier measurements depicted by red dots in Fig. 3(b) and recalculate the L_2 and L_1 principal component of the contaminated data set. It is striking, in this illustration, how poorly now the L_2 principal component represents the nominal data and how resistant to the outliers the L_1 principal component calculation is.



(a) (b)

Fig. 2: Empirical CDF of *(a)* performance degradation of L_1 principal component calculation algorithms and *(b)* SBF iteration termination index under proposed and arbitrary initialization.



(a) (b)

Fig. 3: L_2 and L_1 principal components calculated over *(a)* clean and *(b)* corrupted data.

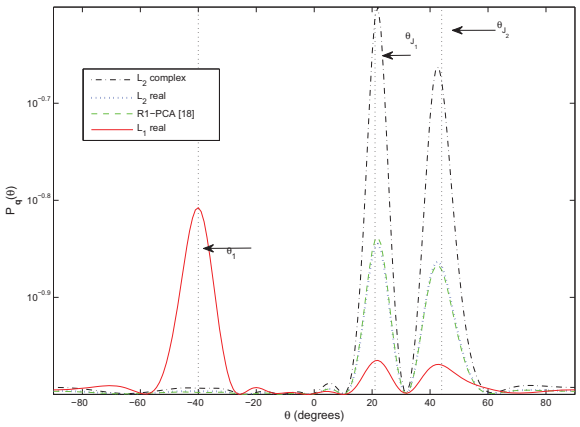


Fig. 4: MUSIC principal-component direction-of-arrival estimation.

Experiment 3 - Direction-of-Arrival Estimation. We consider a uniform linear antenna array of $D = 10$ elements. The array collects $N = 20$ snapshots of a single binary Bernoulli-equiprobable signal $m \in \{\pm 1\}$ with amplitude A, coming at an angle of arrival $\theta_1 = -40^\circ$ in the presence of white complex Gaussian noise $\mathbf{n}$,

$$\mathbf{x}_n = A\, m_n\, \mathbf{s}_{\theta_1} + \mathbf{n}_n,\ n = 1, \cdots, 20, \tag{7}$$

where $\mathbf{s}_{\theta_1}$ is the array response vector. The signal-to-noise ratio is set at $\mathrm{SNR} = 6\mathrm{dB}$. We assume that two arbitrary measurements (among the twenty available) are corrupted by two jammers operating at angles $\theta_{J_1} = -1^\circ$ and $\theta_{J_2} = 44^\circ$ with $\mathrm{SNR}_{\mathrm{J}_{1,2}} = 15$ dB. The resulting corrupted observation set is called $\mathbf{X}^{\mathrm{COR}} \in \mathbb{C}^{10\times 20}$ and transformed in real-domain to $\tilde{\mathbf{X}}^{\mathrm{COR}} = \begin{bmatrix} \mathrm{Re}\{\mathbf{X}^{\mathrm{COR}}\}, & -\mathrm{Im}\{\mathbf{X}^{\mathrm{COR}}\} \\ \mathrm{Im}\{\mathbf{X}^{\mathrm{COR}}\}, & \mathrm{Re}\{\mathbf{X}^{\mathrm{COR}}\} \end{bmatrix} \in \mathbb{R}^{20\times 40}$. Then, we compute: *(i)* The real L_2 principal component of $\tilde{\mathbf{X}}^{\mathrm{COR}}$ denoted by $\mathbf{r}_{L_2}$; *(ii)* the complex L_2 principal component of $\mathbf{X}^{\mathrm{COR}}$ denoted by $\mathbf{c}_{L_2}$; *(iii)* the robust R_1 principal component of $\tilde{\mathbf{X}}^{\mathrm{COR}}$ per [18], denoted by $\mathbf{r}_{R_1}$; and *(iv)* the proposed real L_1 principal component of $\tilde{\mathbf{X}}^{\mathrm{COR}}$ denoted by $\mathbf{r}_{L_1}$.

We then perform in Fig. 4 MUSIC-type [20] DoA estimation using the spectrum function

$$P_{\mathbf{q}}(\theta) = \left(\mathrm{Trace}(\tilde{\mathbf{s}}_\theta^T (\mathrm{I} - \mathbf{q}\mathbf{q}^T)\tilde{\mathbf{s}}_\theta)\right)^{-1}, \mathbf{q} \in \{\mathbf{r}_{L_2}, \mathbf{r}_{R_1}, \hat{\mathbf{r}}_{L_1}, \hat{\mathbf{c}}_{L_2}\}$$

where $\tilde{\mathbf{s}}_\theta$ is the concatenated real array response matrix [21] for the real-valued principal-component cases *(i), (iii)* and *(iv)* and the standard complex array response vector for case *(ii)*. It is, again, most interesting to observe how much less reactive to the presence of the outlier jammers in the data set is the calculated L_1 principal component compared to the conventional L_2 approaches or the robust rotational-invariant principal component of [18].

6. REFERENCES

[1] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, "Some options for L1-subspace singal processing," in *Proc. Intern. Symp. on Wireless Commun. Systems (ISWCS)*, Ilmenau, Germany, Aug. 2013, pp. 622-626.

[2] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, "Optimal algorithms for L_1-subspace signal processing," *IEEE Trans. Signal Proc.*, submitted June 2013.

[3] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211-218, Sept. 1936.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd Ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

[5] N. Kwak "Principal component analysis based on L1-norm maximization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 1672-1680, Sept. 2008.

[6] Q. Ke and T. Kanade, "Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, June 2005, pp. 739–746.

[7] F. Nie, H. Huang, C. Ding, D. Luo, and H. Wang, "Robust principal component analysis with non-greedy l_1-norm maximization," in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Barcelona, Spain, July 2011, pp. 1433–1438.

[8] J. S. Galpin and D. M. Hawkins, "Methods of L_1 estimation of a covariance matrix," *J. Comput. Stat. Data Anal.*, vol. 5, pp. 305–319, 1987.

[9] X. Li, Y. Pang, and Y. Yuan, "L1-norm-based 2DPCA," *IEEE Trans. Syst., Man. Cybern. B, Cybern.*, vol. 40, pp. 1170–1175, Aug. 2009.

[10] Y. Pang, X. Li, and Y. Yuan, "Robust tensor analysis with L1-norm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, pp. 172–178, Feb. 2010.

[11] N. Funatsu and Y. Kuroki, "Fast parallel processing using GPU in computing L1-PCA bases," in *Proc. IEEE TENCON 2010*, Fukuoka, Japan, Nov. 2010, pp. 2087–2090.

[12] M. McCoy and J. A. Tropp, "Two proposals for robust PCA using semidefinite programming," *Electron. J. Stat.*, vol. 5, pp. 1123–1160, June 2011.

[13] D. Meng, Q. Zhao, and Z. Xu, "Improve robustness of sparse PCA by L_1-norm maximization," *Pattern Recogn.*, vol. 45, pp. 487–497, Jan. 2012.

[14] H. Wang, Q. Tang, and W. Zheng, "L1-norm-based common spatial patterns," *IEEE Trans. Biomed. Eng.*, vol. 59, pp. 653–662, Mar. 2012.

[15] H. Wang, "Block principal component analysis with L1-norm for image analysis," *Pattern Recogn. Lett.*, vol. 33, pp. 537–542, Apr. 2012.

[16] H. Q. Luong, B. Goossens, J. Aelterman, A. Pižurica, and W. Philips, "A primal-dual algorithm for joint demosaicking and deconvolution," in *Proc. IEEE Intern. Conf. Image Proc. (ICIP)*, Oct. 2012, pp. 2801–2804.

[17] Z. Gu, W. Lin, B.-S. Lee, and C. T. Lau, "Rotated orthogonal transform (ROT) for motion-compensation residual coding," *IEEE Trans. Image Proc.*, vol. 21, pp. 4770–4781, Dec. 2012.

[18] C. Ding, D. Zhou, X. He, and H. Zha, "R_1-PCA: Rotational invariant L_1-norm principal component analysis for robust subspace factorization," in *Proc. Int. Conf. Machine Learning*, Pittsburgh, PA, 2006, pp. 281–288.

[19] Z. Liu and D. A. Pados, "A decoding algorithm for finite geometry LDPC codes," *IEEE Trans. Commun.*, vol. 53, pp. 415–421, Mar. 2005.

[20] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Prop.*, vol. 34, pp. 276-280, Mar. 1986.

[21] P. P. Markopoulos, N. Tsagkarakis, D. A. Pados, and G. N. Karystinos, "Direction finding with L1-norm subspaces," in *Proc. Compressive Sensing Conf., SPIE Sensing Tech. + Appl.*, Baltimore, MD, May 2014.