PRACTICAL REPROCS FOR SEPARATING SPARSE AND LOW-DIMENSIONAL SIGNAL SEQUENCES FROM THEIR SUM – PART 1

Han Guo, Chenlu Qiu and Namrata Vaswani

ECE dept, Iowa State University, Ames, IA, USA
Email: {hanguo,chenlu,namrata}@iastate.edu

ABSTRACT

This paper designs and evaluates a practical algorithm, called Prac-ReProCS, for recovering a time sequence of sparse vectors S_t and a time sequence of dense vectors L_t from their sum, $M_t := S_t + L_t$, when any subsequence of the L_t's lies in a slowly changing low-dimensional subspace. A key application where this problem occurs is in video layering where the goal is to separate a video sequence into a slowly changing background sequence and a sparse foreground sequence that consists of one or more moving regions/objects. Prac-ReProCS is the practical analog of its theoretical counterpart that was studied in our recent work.

Index Terms— robust PCA, robust matrix completion, sparse recovery, compressed sensing

1. INTRODUCTION

The goal of this work is to recover a time sequence of sparse vectors S_t and a time sequence of dense vectors L_t from their sum, $M_t := S_t + L_t$, when any subsequence of the L_t's lies in a slowly changing low-dimensional subspace. The magnitude of the entries of L_t could be larger, roughly equal or smaller than that of the nonzero entries of S_t.

The above problem can be interpreted as one of recursive sparse recovery from potentially large but structured noise. In this case, S_t is the quantity of interest and L_t is the potentially large but structured (low-dimensional) noise. Alternatively it can be posed as a recursive robust principal components analysis (PCA) problem. In this case L_t, or in fact, the subspace in which the last several (d) L_t's lie, range($[L_{t-d+1}, \dots L_t]$), is the quantity of interest while S_t is the outlier.

A key application where this problem occurs is in video layering where the goal is to separate a slowly changing background from moving foreground objects/regions (sparse image) [2, 3]. The foreground layer, e.g. moving people/objects, is of interest in applications such as automatic video surveillance, tracking moving objects, or video conferencing. The background sequence is of interest in applications such as background editing (video editing applications). In most static camera videos, the background images do not change much over time and hence the mean-subtracted background image sequence is well modeled as lying in a fixed or slowly-changing low-dimensional subspace of $\mathbf{R}^n$ [3]. Moreover the changes are typically global, e.g. due to lighting variations, and hence modeling it as a dense image sequence is valid too. The foreground layer usually consists of one or more moving objects/persons/regions that move in a correlated fashion, i.e. it is a sparse image sequence that often changes in a correlated fashion over time. By letting M_t be the entire image, L_t be the background image and defining S_t as the the foreground-background intensity difference on the foreground support and zero everywhere else, video layering becomes a problem of separating S_t and L_t from $M_t = S_t + L_t$.

Related Work. In the last few decades, there has been a large amount of work on robust PCA, e.g. [2, 4, 5, 6], and recursive robust PCA e.g. [7, 8, 9]. In most of these works, either the locations of the missing/corruped data points are assumed known [7] (not a practical assumption); or they first detect the corrupted data points and then replace their values using nearby values [8]; or weight each data point in proportion to its reliability (thus soft-detecting and down-weighting the likely outliers) [2, 9]; or just remove the entire outlier vector [5, 6].

In a series of recent works [3, 10], a new and provably correct solution to robust PCA called Principal Components' Pursuit (PCP) has been proposed, that does not require a two step outlier location detection/correction process and also does not throw out the entire vector. It redefines batch robust PCA as a problem of separating a low rank matrix $\mathcal{L}$ and a sparse matrix $\mathcal{S}$ from their sum $\mathcal{M}$. PCP finds $\mathcal{S}$ and $\mathcal{L}$ by solving $\min_{\mathcal{S},\mathcal{L}} \|\mathcal{S}\|_1 + \|\mathcal{L}\|_*$ s.t. $\mathcal{M} = \mathcal{S} + \mathcal{L}$ where $\|.\|_1$ denotes the vector ℓ_1 norm and $\|.\|_*$ denotes the nuclear norm. It is shown that if the low-rank matrix is dense and if the sparse matrix has support set entries that are independently selected, then solving PCP will indeed return the correct sparse and low-rank matrices. Other recent works that also study batch algorithms for recovering a sparse matrix and a low-rank matrix from their sum, or from undersampled measurements of their sum, include [11, 12, 13, 14, 15, 16, 17, 18].

Notice that most applications where video layering is required, such as video surveillance, require an online solu-

Longer version of this paper is under submission to IEEE Trans. Sig. Proc [1]. This work was supported by NSF grant CCF-1117125.

tion. A batch solution would require a long delay; and would also be much slower than a recursive solution. Moreover, the assumption that the foreground support is independent over time is not usually valid. To address these issues, in [19] we introduced a novel recursive solution approach which we later called Recursive Projected Compressive Sensing (ReProCS) [20]. In recent work [21, 22], we have tried to obtain performance guarantees for ReProCS. Under mild assumptions and an assumption on an algorithm estimate (that holds in simulations as long as there is *some* support change every few frames), we showed that, with high probability, ReProCS can exactly recover the support set of S_t at all times; and the reconstruction errors of both S_t and L_t are upper bounded by a time invariant and small value.

Contributions. In this work, we design a practically usable modification of the theoretical ReProCS algorithm studied in [21, 22]. By "practically usable", we mean that (a) it requires much fewer parameters and we explain how to set these parameters without any model knowledge; and (b) it exploits practically valid assumptions such as denseness of L_t's, slow subspace change of L_t's, and gradual support change of S_t's. We show via extensive simulation experiments that ReProCS is more robust to correlated support change of S_t than PCP and other existing work. Also, it is also able to recover small magnitude sparse vectors better than other existing works. The simulation experiments are shown in this paper; the model verification and real video experiments are shown in longer version of this paper [1].

Some later work of this topic includes [23]. Its key idea is similar to that of the original ReProCS algorithm [19, 20].

Notation. For a set $T \subseteq \{1, 2, \cdots n\}$, we use $|T|$ to denote its cardinality, i.e., the number of elements in T. The symbols $\cup, \cap, \setminus$ denote set union set intersection and set difference respectively. The notation $[.]$ denotes an empty matrix. We use the notation $B \overset{SVD}{=} U\Sigma V'$ to denote the singular value decomposition (SVD) of B, and range(B) denotes the subspace spanned by the columns of B.

A matrix P is a *basis matrix* if $P'P = I$.

The *notation $Q = basis(range(M))$, or $Q = basis(M)$ for short,* means that Q is a basis matrix for range(M) i.e. Q satisfies $Q'Q = I$ and range$(Q) =$ range(M).

The *$b\%$ left singular values' set* of a matrix M is the smallest set of indices of its singular values that contains at least $b\%$ of the total singular values' energy. The corresponding matrix of left singular vectors, U_T, is referred to as the *$b\%$ left singular vectors' matrix.*

Definition 1.1 *The notation Q = approx-basis$(M, b\%)$ means that Q is the $b\%$ left singular vectors' matrix for M. The notation Q = approx-basis(M, r) means that Q contains the left singular vectors of M corresponding to its r largest singular values.*

2. PROBLEM DEFINITION AND ASSUMPTIONS

The measurement vector at time t, M_t, is an n dimensional vector which can be decomposed as

$$M_t = S_t + L_t \tag{1}$$

where S_t is a sparse vector and L_t is a dense but low-dimensional vector. We use T_t to denote the support set of S_t.

Suppose that an initial training sequence which does not contain the sparse components is available, i.e. we are given $\mathcal{M}_{\text{train}} = [M_t; 1 \le t \le t_{\text{train}}]$ with $M_t = L_t$. This is used to get an initial estimate of the subspace in which the L_t's lie [1]. At each $t > t_{\text{train}}$, the goal is to recursively estimate S_t and L_t and the subspace in which the last several L_t's lie. By "recursively" we mean: use $\hat{S}_{t-1}, \hat{L}_{t-1}$ and the previous subspace estimate to estimate S_t and L_t.

Our algorithm is based on three assumptions that we explain next. These assumptions are verified for real video data in [1].

2.1. Low-dimensionality and slow subspace change

One way to quantify this assumption is as follows. We let $L_t = P_{(t)} a_t$ where $P_{(t)}$ is a tall matrix that is piecewise constant with time, i.e. $P_{(t)} = P_j$ for all $t \in [t_j, t_{j+1})$ where P_j is an $n \times r_j$ basis matrix with $r_j \ll \min((t_{j+1} - t_j), n)$. A very simple model for slow subspace change is to let P_j change as

$$P_j = [(P_{j-1} R_j \setminus P_{j,\text{old}}), P_{j,\text{new}}]$$

where $P_{j,\text{new}}$ and $P_{j,\text{old}}$ are basis matrices of size $n \times c_{j,\text{new}}$ and $n \times c_{j,\text{old}}$ respectively with $P_{j,\text{new}}' P_{j-1} = 0$ and R_j is a rotation matrix. Moreover, the projection of L_t along $P_{j,\text{new}}$ is small initially for the first α frames, i.e.

$$\|(I - P_{j-1} P_{j-1}') L_t\|_2 \ll \min(\|L_t\|_2, \|S_t\|_2) \text{ if } t \in [t_j, t_j + \alpha)$$

and can increase gradually after $t_j + \alpha$.

2.2. Denseness

We assume that the subspace spanned by the L_t's is dense, i.e.

$$\kappa_{2s}(P_j) = \kappa_{2s}([L_{t_j}, \dots L_{t_{j+1}-1}]) \le \kappa_*$$

for a κ_* significantly smaller than one. Here

$$\kappa_s(B) = \kappa_s(\text{range}(B)) := \max_{|T| \le s} \|I_T{}' \text{basis}(B)\|_2 \tag{2}$$

is the denseness coefficient for any vector or matrix B [21, 22]. Moreover, a similar assumption holds for $P_{j,\text{new}}$ with a

[1]If an initial sequence without S_t's is not available, one can use a batch robust PCA algorithm to get the initial subspace estimate as long as the initial sequence satisfies its required assumptions.

tighter bound: $\kappa_{2s}(P_{j,\text{new}}) \leq \kappa_{\text{new}} < \kappa_*$. By [21, Lemma 2], a small $\kappa_{2s}(P_j)$ means that the restricted isometry constant (RIC) [24] of the matrix $(I - P_j P_j')$ is small. Using any of the RIC based sparse recovery results, e.g. [25], this ensures that for $t \in [t_j, t_{j+1})$, s-sparse vectors S_t are recoverable from $(I - P_j P_j')M_t = (I - P_j P_j')S_t$ by ℓ_1 minimization.

2.3. Support size, support change of S_t

We assume two things. First, we assume that either the support size is small or the support changes are slow or both. At the same time, we also assume that there is *some* support change during any set of α frames. Practically, this is needed to ensure that at least some of the background behind the foreground is visible so that the changes to the background subspace can be estimated. In the performance guarantees derived in [21], this ensures that the currently unestimated subspace of range$(P_{j,\text{new}})$ is dense.

3. PRACTICAL REPROCS

The complete practical Recursive Projected Compressive Sensing (ReProCS) algorithm is summarized in Algorithm 1. We explain its steps below. We use $\hat{S}_t, \hat{T}_t, \hat{L}_t$ to denote estimates of S_t, its support, T_t, and L_t respectively; and we use $\hat{P}_{(t)}$ to denote the basis matrix of the estimated subspace of the last several L_t's (sometimes we just refer to $\hat{P}_{(t)}$ as the subspace estimate). Also, let

$$\beta_t := \Phi_{(t)} L_t, \text{ where } \Phi_{(t)} := (I - \hat{P}_{(t-1)} \hat{P}_{(t-1)}') \quad (3)$$

Given the initial training sequence which does not contain the sparse components, $\mathcal{M}_{\text{train}} = [L_1, L_2, \dots L_{t_{\text{train}}}]$ we compute $\hat{P}_0$ as an approximate basis for $\mathcal{M}_{\text{train}}$, i.e. $\hat{P}_0 = \text{approx-basis}(\mathcal{M}_{\text{train}}, b\%)$ with $b\% = 95\%$. Also let $\hat{r} = \text{rank}(\hat{P}_0)$. We need to compute an approximate basis because for real data, the L_t's are only approximately low-dimensional. After this, at each time t, ReProCS involves 4 steps that we explain next.

Perpendicular Projection. At time t, we project the measurement vector, M_t, into the space orthogonal to $\hat{P}_{(t-1)}$ to get $y_t := \Phi_{(t)} M_t$. As we explain in the Subspace Update step, $\hat{P}_{(t)}$ is updated every α frames.

Sparse Recovery (Recover T_t and S_t). With the above projection, y_t can be rewritten as

$$y_t = \Phi_{(t)} S_t + \beta_t$$

where β_t is defined in (3). As explained in [1, 21], $\|\beta_t\|_2$ is small. Briefly, if the current subspace is accurately estimated, then this is because projecting orthogonal to range$(\hat{P}_{(t-1)})$ nullifies most of the contribution of L_t; on the other hand, if range$(P_{j,\text{new}})$ has so far not been estimated, then this is still true because of the slow subspace change assumption. As a result the problem of recovering S_t from y_t becomes a traditional sparse recovery / CS problem in small noise, β_t. Notice that, since the $n \times n$ projection matrix, $\Phi_{(t)}$, has rank $(n - \text{rank}(\hat{P}_{(t-1)}))$, therefore y_t has only this many "effective" measurements, even though its length is n.

To recover S_t from y_t, we solve

$$\min_x \|x\|_1 \text{ s.t. } \|y_t - \Phi_{(t)} x\|_2 \leq \xi \quad (4)$$

and denote its solution by $\hat{S}_{t,cs}$. By the denseness assumption, the basis matrix $P_{(t-1)}$ is dense. Since $\hat{P}_{(t-1)}$ approximates it, this is true for $\hat{P}_{(t-1)}$ as well [21, Lemma 8.3]. Thus, by [21, Lemma 2], the restricted isometry constant (RIC) of $\Phi_{(t)}$ is small. Using [25, Theorem 1], this and the fact that β_t is small ensures that S_t can be accurately recovered from y_t. By thresholding on $\hat{S}_{t,cs}$ to get an estimate of its support followed by computing a least squares (LS) estimate of S_t on the estimated support and setting it to zero everywhere else, we can get a more accurate estimate, $\hat{S}_t$. The thresholding and LS help to reduce the bias and total reconstruction error in the solution.

The constraint ξ used in the minimization should equal $\|\beta_t\|_2$ or its upper bound. Since β_t is unknown we can replace $\|\beta_t\|_2$ by $\|\hat{\beta}_t\|_2$ where $\hat{\beta}_t := \Phi_{(t)} \hat{L}_{t-1}$. This will usually be smaller than the upper bound on $\|\beta_t\|_2$. However that only means that the solution of (4) may have some extra nonzero elements. With an appropriate thresholding step, most of these should not be detected into the support.

Recover L_t. The estimate $\hat{S}_t$ is used to estimate L_t as $\hat{L}_t = M_t - \hat{S}_t$. Thus, if S_t is recovered accurately, so will L_t.

Subspace Update (Update $\hat{P}_{(t)}$). Within a short delay after every subspace change time, one needs to update the subspace estimate, $\hat{P}_{(t)}$. In practice, since the subspace change model is not known, the subspace update needs to be done at regular short enough intervals. This is needed to ensure that the subspace gets updated quickly enough so that the projected noise β_t seen by the sparse recovery step never becomes too large. At the same time, to get an accurate subspace estimate using simple PCA, one needs to use d frames for a d that is large enough compared to r_j. To satisfy both requirements, we use overlapping periods for subspace estimation: every α frames, we do a subspace update using the previous d estimates $\hat{L}_t$ with a $d \gg \alpha$. To be precise at every $t = t_{\text{train}} + k\alpha$, $k = 1, 2, \dots$, we compute $\hat{P}_{(t)} = \text{approx-basis}([\hat{L}_{t-d+1}, \dots \hat{L}_t], \hat{r})$ where $\hat{r} = \text{rank}(\hat{P}_0)$. The choice of α is governed by computational complexity. In the experiments shown, we used $d = 10\hat{r}$ and $\alpha = 50$.

The subspace update step can be made recursive as explained in [1]. Alternatively, one can use projection PCA introduced in [21] (practical version explained in [1]). Experiments using these are shown in [1].

Improved Sparse Recovery. Whenever slow support change holds, one can replace ℓ_1 minimization by modified-CS [26] or its generalization called weighted ℓ_1 [27, 28].

Algorithm 1 Practical ReProCS

Input: M_t; **Output:** T_t, $\hat{S}_t$, $\hat{L}_t$; **Parameters:** b, d, α.

The algorithm uses Definition 3.1.

Initialization: Compute $\hat{P}_0 \leftarrow$ approx-basis$([M_1, \ldots M_{t_{\text{train}}}], b\%)$ with $b = 95$. Set $\hat{r} \leftarrow \text{rank}(\hat{P}_0)$, $d \leftarrow 10\hat{r}$, $\alpha \leftarrow 50$; $\hat{P}_{t_{\text{train}}} \leftarrow \hat{P}_0$ and $\hat{T}_t \leftarrow [.]$.

For $t > t_{\text{train}}$ do

1. Perpendicular Projection
   (a) $y_t \leftarrow \Phi_{(t)} M_t$, $\Phi_{(t)} \leftarrow I - \hat{P}_{t-1}\hat{P}'_{(t-1)}$
2. Sparse Recovery (Recover S_t and T_t)
   If $\frac{|\hat{T}_{t-2} \cap \hat{T}_{t-1}|}{|\hat{T}_{t-2}|} < 0.5$
   (a) Compute $\hat{S}_{t,\text{cs}}$ by solving simple ℓ_1, i.e. (4) with $\xi = \|\Phi_{(t)}\hat{L}_{t-1}\|_2$.
   (b) $\hat{T}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{cs}}, \omega)$ with $\omega = \sqrt{\|M_t\|^2/n}$

   Else

   (a) Compute $\hat{S}_{t,\text{cs}}$ by solving weighted-ℓ_1
   $$\min_x \lambda \|x_{\hat{T}_{t-1}}\|_1 + \|x_{\hat{T}^c_{t-1}}\|_1 \text{ s.t. } \|y_t - \Phi_{(t)}x\|_2 \leq \xi$$
   with $\lambda = \frac{|\hat{T}_{t-2} \setminus \hat{T}_{t-1}|}{|\hat{T}_{t-1}|}$ and $\xi = \|\Phi_{(t)}\hat{L}_{t-1}\|_2$.
   (b) $\hat{T}_{\text{add}} \leftarrow \text{Prune}(\hat{S}_{t,\text{cs}}, 1.4|\hat{T}_{t-1}|)$.
   (c) $\hat{S}_{t,\text{add}} \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{T}_{\text{add}})$
   (d) $\hat{T}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{add}}, \omega)$ with $\omega = \sqrt{\|M_t\|^2/n}$

   Set $\hat{S}_t \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{T}_t)$
3. Estimate L_t: $\hat{L}_t \leftarrow M_t - \hat{S}_t$
4. Update $\hat{P}_{(t)}$: If $t = t_{\text{train}} + k\alpha$, $k = 1, 2, \ldots$
   (a) $\hat{P}_{(t)} \leftarrow$ approx-basis$([\hat{L}_{t-d+1}, \ldots \hat{L}_t], \hat{r})$

   Else $\hat{P}_{(t)} \leftarrow \hat{P}_{(t-1)}$

These require fewer measurements for exact/accurate recovery when the previous support estimate, $\hat{T}_{t-1}$, is an accurate enough estimate of the current support, T_t. Moreover, support estimation can be improved by using an approach similar to the Add-LS-Del procedure [29].

We summarize the complete algorithm including the above step and heuristics to set its parameters in Algorithm 1.

Definition 3.1 *In the algorithm,*

1. $T \leftarrow$ *Thresh*(x, ω) *means* $T = \{i : |(x)_i| \geq \omega\}$
2. $T \leftarrow$ *Prune*(x, s) *means* T *contains the* s *largest magnitude elements of* x
3. $\hat{x} \leftarrow$ *LS*(y, A, T) *means*
$$\hat{x}_T = (A_T)^\dagger y = (A_T{}'A_T)^{-1}A_T{}'y, \ \hat{x}_{T^c} = 0.$$

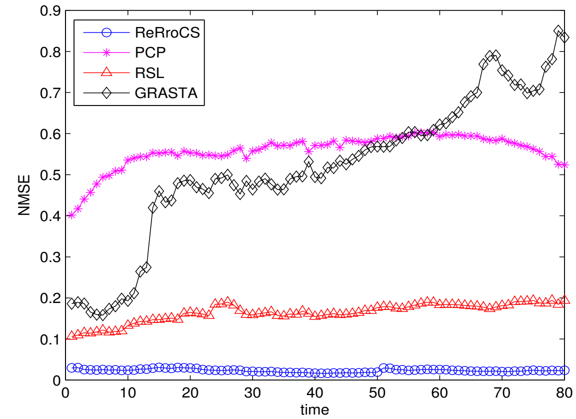

Fig. 1: NMSE for recovering S_t ($t = 0$ refers to $t = t_{\text{train}}$)

4. PARTLY SIMULATED EXPERIMENTS

We used a real slowly changing background sequence and overlaid a simulated foreground sequence consisting of a moving rectangular object on it. The use of a real background sequence allows us to evaluate performance for data that only approximately satisfies the low-dimensional and slow subspace change assumptions. The use of the simulated foreground allows us to control its intensity so that the resulting S_t is small or of the same order as L_t (making it a difficult sequence). We controlled the foreground intensity so that $\|S_t\|_2$ was roughly equal or smaller than $\|L_t\|_2$ making it a difficult sequence. Moreover it provides ground truth data so that the recovery performance can be quantitatively compared.

The background was a video of moving waters in a lake (see [1]). The moving object was simulated as explained in [1]. We generated 50 realizations of the video sequence and compared all the algorithms to estimate S_t, L_t and then the foreground and the background sequences. We show comparisons of the normalized mean squared error (NMSE) in recovering S_t in Fig. 1. As can be seen, the ReProCS error is the smallest and stable. PCP gives very large error for this sequence since the object moves in a highly correlated fashion and occupies a large part of the image. GRASTA [23] also does not work and we think it is because the GRASTA code does not update the background subspace. Robust Subspace Learning (RSL) [2] is able to recover a large part of the object correctly, however it also recovers many more extras than ReProCS. The reason is that the magnitude of the nonzero entries of S_t is equal or small compared to those of L_t.

Real video experiments are available in [1] and `http://www.ece.iastate.edu/~hanguo/PracReProCS.html`.

5. REFERENCES

[1] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *arXiv:1310.4261v2, submitted to IEEE Trans. Signal Processing*.

[2] F. De La Torre and M. J. Black, "A framework for robust subspace learning," *International Journal of Computer Vision*, vol. 54, pp. 117–142, 2003.

[3] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of ACM*, vol. 58, no. 3, 2011.

[4] T. Zhang and G. Lerman, "A novel m-estimator for robust pca," *arXiv:1112.4863v3, to appear in Journal of Machine Learning Research*, 2013.

[5] H. Xu, C. Caramanis, and S. Sanghavi, "Robust pca via outlier pursuit," *IEEE Trans. Info. Th.*, vol. 58, no. 5, 2012.

[6] Michael McCoy and Joel A Tropp, "Two proposals for robust pca using semidefinite programming," *Electronic Journal of Statistics*, vol. 5, pp. 1123–1160, 2011.

[7] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *European Conference on Computer Vision*, 2002, pp. 707–720.

[8] D. Skocaj and A. Leonardis, "Weighted and robust incremental method for subspace learning," in *IEEE Intl. Conf. on Computer Vision (ICCV)*, Oct 2003, vol. 2, pp. 1494 –1501.

[9] Y. Li, L. Xu, J. Morphett, and R. Jacobs, "An integrated algorithm of incremental and robust pca," in *IEEE Intl. Conf. Image Proc. (ICIP)*, 2003, pp. 245–248.

[10] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, 2011.

[11] Michael B McCoy and Joel A Tropp, "Sharp recovery bounds for convex deconvolution, with applications," *arXiv:1205.1580, accepted to J. Found. Comput. Math*, 2012.

[12] Venkat Chandrasekaran, Benjamin Recht, Pablo A. Parrilo, and Alan S. Willsky, "The convex geometry of linear inverse problems," *Foundations of Computational Mathematics*, , no. 6, 2012.

[13] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk, "Sparcs: Recovering low-rank and sparse matrices from compressive measurements," in *Proc. of Neural Information Processing Systems(NIPS)*, 2011.

[14] Daniel Hsu, Sham M Kakade, and Tong Zhang, "Robust matrix decomposition with sparse corruptions," *Information Theory, IEEE Transactions on*, vol. 57, no. 11, pp. 7221–7234, 2011.

[15] Morteza Mardani, Gonzalo Mateos, and Georgios B. Giannakis, "Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies," *IEEE Trans. Info. Th.*, 2013.

[16] John Wright, Arvind Ganesh, Kerui Min, and Yi Ma, "Compressive principal component pursuit," *Information and Inference*, vol. 2, no. 1, pp. 32–68, 2013.

[17] Arvind Ganesh, Kerui Min, John Wright, and Yi Ma, "Principal component pursuit with reduced linear measurements," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1281–1285.

[18] Min Tao and Xiaoming Yuan, "Recovering low-rank and sparse components of matrices from incomplete and noisy observations," *SIAM Journal on Optimization*, vol. 21, no. 1, pp. 57–81, 2011.

[19] C. Qiu and N. Vaswani, "Real-time robust principal components' pursuit," in *Allerton Conf. on Communications, Control and Computing*, 2010.

[20] C. Qiu and N. Vaswani, "Recursive sparse recovery in large but correlated noise," in *Allerton Conf. on Communication, Control, and Computing*, 2011.

[21] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," *under revision for IEEE Trans. Info. Th., also at arXiv: 1211.3754[cs.IT], shorter versions in ICASSP 2013 and ISIT 2013*.

[22] C. Qiu and N. Vaswani, "Recursive sparse recovery in large but structured noisepart 2," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 864–868.

[23] Jun He, Laura Balzano, and Arthur Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1568–1575.

[24] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Info. Th.*, vol. 51(12), pp. 4203 – 4215, Dec. 2005.

[25] E. Candes, "The restricted isometry property and its implications for compressed sensing," *Compte Rendus de l'Academie des Sciences, Paris, Serie I*, pp. 589–592, 2008.

[26] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Signal Processing*, September 2010.

[27] M Amin Khajehnejad, Weiyu Xu, Amir Salman Avestimehr, and Babak Hassibi, "Weighted ℓ_1 minimization for sparse recovery with prior information," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2009, pp. 483–487.

[28] M.P. Friedlander, H. Mansour, R. Saab, and O. Yilmaz, "Recovering compressively sampled signals using partial support information," *IEEE Trans. Info. Th.*, vol. 58, no. 2, pp. 1122–1134, 2012.

[29] N. Vaswani, "Stability (over time) of Modified-CS for Recursive Causal Sparse Reconstruction," in *Allerton Conf. Communication, Control, and Computing*, 2010.