STREAMING COMPRESSIVE SENSING FOR HIGH-SPEED PERIODIC VIDEOS

M. Salman Asif,[a] Dikpal Reddy,[b] Petros T. Boufounos,[c] Ashok Veeraraghavan[c]

[a] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA

[b] Department of Electrical and Computer Engineering, University of Maryland, College Park, MD

[c] Mitsubishi Electric Research Laboratories, Cambridge, MA

ABSTRACT

The ability of Compressive Sensing (CS) to recover sparse signals from limited measurements has been recently exploited in computational imaging to acquire high-speed periodic and near-periodic videos using only a low-speed camera with coded exposure and intensive off-line processing. Each low-speed frame integrates a coded sequence of high-speed frames during its exposure time. The high-speed video can be reconstructed from the low-speed coded frames using a sparse recovery algorithm. This paper presents a new streaming CS algorithm specifically tailored to this application. Our streaming approach allows causal on-line acquisition and reconstruction of the video, with a small, controllable, and guaranteed buffer delay and low computational cost. The algorithm adapts to changes in the signal structure and, thus, outperforms the off-line algorithm in realistic signals.

1. INTRODUCTION

Modern digital acquisition and sensing technology, largely based in the pioneering work of Shannon and Nyquist, has enabled the acquisition, transmission, and storage of ever increasing amounts of physical signals. However, the increasing demand for higher acquisition rates and improved resolution is placing significant burden on existing hardware architectures, reaching their performance limits. Fortunately, recent work in various signal processing fields, mostly in the area of Compressive Sensing (CS) [1–3] but also in the field of signal models and representations has provided new powerful frameworks that enable the acquisition of signals at rates significantly below the Nyquist rate. The rapid success of CS—which uses randomized incoherent measurements to acquire the signal and exploits sparse structure of the signal for the reconstruction— demonstrates the paradigm-shifting potential and the need for the technology.

One area significantly affected by the introduction of inexpensive computation and signal models is image and video acquisition. The field of computational imaging enables us to computationally achieve de-blurring, super-resolution, large dynamic range, variable depth-of-field and other desired properties, using simple modifications to inexpensive sensors [4, 5].

A recent example of such a breakthrough technology is coded strobing and coded exposure video acquisition [6, 7], which enables the acquisition of high-speed video using a standard speed low-cost video sensor enhanced by an inexpensive coded strobe or coded shutter (also known as flutter-shutter). The coding on the exposure[1] and the reconstruction algorithms are designed using CS principles, assuming a periodic signal model. In [7] each pixel location of the video is considered an independent periodic frequency-sparse signal of unknown fundamental period, compressively acquired at the rate of the low-speed video sensor. The well-established CoSaMP [8] reconstruction algorithm is combined with several computational heuristics to recover the high-speed time waveform of each pixel and reconstruct the high-speed video from the acquired data.

In this paper we significantly enhance this work in several ways:

- **We develop a streaming reconstruction algorithm**, the Streaming Greedy Pursuit (SGP) [9], which enables on-line reconstruction of the high-speed video. The CoSaMP-based SGP is specifically designed for streaming CS scenarios, with explicit guarantees on the computational cost per sample and on the input-output delay.
- **We formulate a signal model** to incorporate the similarities in the sparsity structure of nearby pixels in the reconstruction algorithm using the principles of joint sparsity and model-based CS [10, 11].
- **We combat matrix-conditioning problems** that arise due to the non-negative nature of imaging measurements by revisiting the minimum variance (or *Capon*) beamformer from the array processing literature [12] and re-introducing it in the context of CS.

Our work significantly improves the reconstruction performance of coded strobing video and, more importantly, enables on-line reconstruction of the acquired video.

2. BACKGROUND

2.1. Compressive sensing

The recently emerged field of compressive sensing [1–3] demonstrates that a signal sparse or compressible in some basis can be efficiently sampled and reconstructed using very few linear measurements. The signal of interest, $\mathbf{x} \in \mathbb{R}^N$, is measured using the system

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{1}$$

where $\mathbf{y}$ denotes the measurement vector and $\mathbf{A}$ an $M \times N$ measurement matrix with $M \ll N$. The signal $\mathbf{x}$ is assumed K-sparse in some basis $\mathbf{B}$, i.e., $\mathbf{B}^{-1}\mathbf{x}$ contains only K non-zero coefficients.

The measurement matrix $\mathbf{A}$ satisfies the *Restricted Isometry Property* (RIP) of order $2K$ if there exists a constant $\delta_{2K} < 1$ such that

$$(1 - \delta_{2K})\|\mathbf{x}\|_2^2 \le \|\mathbf{A}\mathbf{x}\|_2^2 \le (1 + \delta_{2K})\|\mathbf{x}\|_2^2, \tag{2}$$

for all $\mathbf{x}$ that are $2K$-sparse in the basis $\mathbf{B}$. If the RIP constant δ_{2K} is sufficiently small, then the signal can be exactly reconstructed using the convex optimization [3]

$$\widehat{\mathbf{x}} = \arg\min \|\mathbf{B}^{-1}\mathbf{x}\|_1 \ \text{ s.t. } \ \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{3}$$

or a greedy algorithm such as CoSaMP [8]. Furthermore, a small RIP constant provides robustness guarantees for recovery in the presence of measurement noise and for sampling signals that are not exactly sparse but can be well-approximated by a sparse signal [13].

2.2. Coded strobing camera

Coded exposure or flutter-shutter photography is a popular technique in computational imaging where the light intensity is coded during

[1] We will use coded exposure and coded strobing terms interchangeably.

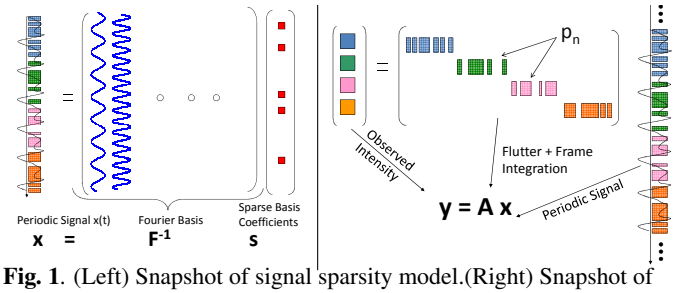

Fig. 1. (Left) Snapshot of signal sparsity model.(Right) Snapshot of coded strobing system.

the exposure duration of the camera [6, 7]. Active strobing illuminates the scene with a rapid sequence of flashes within a frame interval. Passive strobing is performed using sensors or a flutter-shutter assembly, with multiple exposures within a frame interval.

To observe a high-speed video at a rate of N frames per second (fps) a coded strobing camera working at $M \ll N$ fps is used instead of a high-frame-rate camera. The exposure time is coded by a pseudorandom binary sequence. Each coded frame integrates the light from a few randomly selected high-speed frames, as described in [7] and the next section.

3. SIGNAL AND SYSTEM MODEL

3.1. Signal model

In contrast to standard finite-length CS signal models, streaming videos do not have a pre-determined length. While it is possible to acquire the whole stream and post-process it, the storage and computational requirements for such an endeavor are often prohibitive. Furthermore, several applications often require on-line processing of such signals. Standard CS approaches perform on-line computation by processing the stream in finite-length blocks which are considered independent of each other. However, this approach ignores the continuity of the signal and introduces significant blocking artifacts. Instead, the algorithm we propose assumes a streaming signal model.

Under our model the signal intensity of each pixel is an infinite length streaming signal x_n which is compressible in the short-time Fourier transform basis of length N. Specifically, we assume the discrete Fourier transform of any length-N window of the signal, $\mathbf{s} = \mathbf{F}\mathbf{x}$, has only $K \ll N$ significant coefficients, as shown in the left hand side of Fig. 1.

We further assume that the support set (i.e., the location of the significant coefficients in the frequency domain) changes slowly in time, so the reconstruction algorithm has time to adjust. Thus, a time shift of the windowed signal should be similar to a circular shift of $\mathbf{x}$ by the same amount or an equivalent frequency-domain phase shift.

The sparsity structure of nearby pixels is captured using joint sparsity models, such as [10, 11]. We assume that signal snapshots of nearby pixels have similar frequency-domain support. The vector

$$\bar{\mathbf{s}}_j = \sum_l (\mathbf{s}_{j,l})^2, \tag{4}$$

where $\mathbf{s}_{j,l}$ denotes the j^{th} frequency component of the l^{th} pixel in the block, captures the support structure and, thus, is also compressible.

3.2. Measurement model

For each pixel, we measure the signal x_n by modulating it with a pulsing sequence p_n taking the values of 0 or 1, depending on whether the strobe is on or off (or, equivalently, whether the flutter-shutter is open or closed), and accumulating every R coefficients to produce a measurement sequence y_m:

$$y_m = \sum_{n=mR}^{(m+1)R-1} x_n p_n, \tag{5}$$

The system has input rate of N coefficients per unit time (the implied frame rate of the high-speed video) for x_n and an output (measurement) rate of $M = N/R$ measurements per unit time (the frame rate of the low-speed sensor), where R denotes the down-sampling rate.

A finite-length snapshot of the streaming system (5) can be captured, as shown in the left hand side of Fig. 1, using

$$\mathbf{y} = \mathbf{A}\mathbf{x} \equiv \mathbf{A}\mathbf{F}^{-1}\mathbf{s}, \tag{6}$$

where $\mathbf{y}$ denotes a finite-length window of the measurements, $\mathbf{A}$ denotes an $M \times N$ measurement matrix, $\mathbf{F}^{-1}$ is N-point inverse DFT matrix, and $\mathbf{s}$ denotes Fourier transform of $\mathbf{x}$.

The mathematical formulation of coded strobing is similar to the random demodulator, introduced in [14] to compressively sample temporal one-dimensional signals that are sparse in the frequency domain. The measurement matrix implemented in hardware has the same structure, representing a multiplication of the signal by the coded pulsing sequence p_n, followed by integration and low-rate sampling. However, a major difference of the two approaches is that while the random demodulator uses $p_n = \pm 1$ for the pulsing sequence, coded strobing is restricted to $p_n = 0$ or 1, due to non-negativity of light.

The positivity constraint on the pulsing sequence has the effect of significantly amplifying the DC component of the signal, which is already strong since the signal itself is positive. Thus, while the matrix implied by the random demodulator satisfies the RIP [14], this is not the case in the coded strobing camera and the matrix is not well conditioned around the zero frequency. A high-speed video reconstruction algorithm should accommodate this issue.

4. STREAMING GREEDY PURSUIT

In contrast to the standard CoSaMP reconstruction algorithm used in [7] for high-speed video reconstruction, the SGP (described in detail in [9]) is an on-line algorithm that receives one frame per iteration from the sensor and outputs R frames of the high-speed video. As with the off-line version, streaming recovery algorithm estimates a time-series for each pixel (or a block of neighboring pixels), and can be parallelized in a straightforward manner.

Section 4.1 describes the high-level streaming iterations of SGP, presented in Algorithm 1, and Sec. 4.2 discusses in detail the signal refinement part, presented in Algorithm 2.

4.1. Streaming iteration

The recovery algorithm continuously re-estimates the video sequence on a sliding window of N high-speed frames using M low-speed captured frames (measurement frames). After each iteration, a new measurement frame is incorporated at the end of the measurement window and the oldest measurement frame is removed from the beginning. Similarly, R new high-speed frames (to be estimated) are included in the signal window and the oldest R are removed from the window and committed to the output. Almost all the steps of the algorithm (except where noted) operate on each pixel independently, so we present the algorithm as it operates on the time signals x_n and y_m generated by the value of one pixel.

In the i^{th} iteration, the recovery algorithm maintains a working signal estimate of length N, denoted $\widehat{\mathbf{x}}^i$, a working measurement matrix of dimension $M \times N$, denoted $\widehat{\mathbf{A}}^i$, and a measurement vector of length M, denoted $\widehat{\mathbf{y}}^i$. The iteration is presented in Algorithm 1,

Algorithm 1 Streaming iteration for recovery algorithm

1: *Increase iteration count:* $i \leftarrow i+1$

2: *Refine working estimate:*

$$\widetilde{\mathbf{x}}^i \leftarrow \text{Refine}(\widehat{\mathbf{x}}^{i-1}, \widehat{\mathbf{y}}^{i-1}, \widehat{\mathbf{A}}^{i-1}),$$

where Refine$(\cdot)$ is described in Algorithm 2.

3: *Update the weighted average of all the samples in the window and commit the estimate for the oldest samples:*

$$\bar{\mathbf{x}}^i \leftarrow \begin{bmatrix} \bar{\mathbf{x}}^{i-1}_{\{R+1,\ldots,N\}} \\ \mathbf{0}_R \end{bmatrix} + \mathbf{W}\widetilde{\mathbf{x}}^i,$$

$\mathbf{0}_R$ is a vector with R zeros, $\mathbf{W}$ is the diagonal weight matrix.

$$\widehat{x}_{Ri+j} = \bar{\mathbf{x}}^i_j, \quad j = 1, \ldots, R$$

where $\widehat{x}_n$ is the streaming signal estimate at the output.

4: *Slide working coefficients window (circular shift in time):*

$$\widehat{\mathbf{x}}^i \leftarrow \begin{bmatrix} \widetilde{\mathbf{x}}^i_{\{R+1,\ldots,N\}} \\ \widetilde{\mathbf{x}}^i_{\{1,\ldots,R\}} \end{bmatrix}$$

5: *Slide working measurement window:*

$$\widehat{\mathbf{y}}^i \leftarrow \begin{bmatrix} \widehat{\mathbf{y}}^i_{\{2,\ldots,M\}} \\ y_{i+M} \end{bmatrix}$$

6: *Slide working measurement matrix:*

$$\widehat{\mathbf{A}}^i \leftarrow \begin{bmatrix} \widehat{\mathbf{A}}^{i-1}_{\{2,\ldots,M\},\{R+1,\ldots,N\}} & \mathbf{0}_{[M-1\times R]} \\ \mathbf{0}_{[1\times N-R]} & p_{\{(i+M)R,\ldots,(i+M+1)R-1\}} \end{bmatrix},$$

where $\mathbf{0}_{[m\times n]}$ denotes $m \times n$ zero matrix.

Algorithm 2 Signal Refinement function: Refine$(\widehat{\mathbf{x}},\ \widehat{\mathbf{y}},\ \widehat{\mathbf{A}})$

1: *Calculate residual:*

$$\mathbf{r} = \widehat{\mathbf{y}} - \widehat{\mathbf{A}}\widehat{\mathbf{x}},$$

2: *Compute proxy:*

$$\mathbf{p} = \mathbf{W}_S \left(\widehat{\mathbf{A}}\mathbf{F}^{-1}\right)^* \mathbf{r},$$

where $\mathbf{W}_S$ is the diagonal strobing weight matrix, $\mathbf{F}$ denotes the DFT matrix and $(\cdot)^*$ denotes the conjugate transpose.

3: *Identify and merge support:*

$$\Omega = \text{supp}\left(\mathbf{F}\widehat{\mathbf{x}}\right) \cup \text{supp}\left(\left.\mathbf{p}\right|_{T,\text{block}}\right),$$

where $\text{supp}(\cdot)$ denotes the support index set of a vector, and $\mathbf{p}|_{T,block}$ denotes truncation of the vector $\mathbf{p}$ to the T dominant coefficients in the surrounding block.

4: *Estimate DFT coefficients over the merged support:*

$$\mathbf{b} = \left(\left.\widehat{\mathbf{A}}\mathbf{F}^{-1}\right|_{\Omega}\right)^{\dagger} \widehat{\mathbf{y}}$$

5: *Truncate DFT coefficients and compute the new estimate:*

$$\widetilde{\mathbf{s}} \leftarrow \left.\mathbf{b}\right|_{K,\text{block}}$$

6: *Output:*

$$\widetilde{\mathbf{x}} \leftarrow \mathbf{F}^{-1}\widetilde{\mathbf{s}}.$$

where the subscript $(\cdot)_m$ notation denotes the m^{th} element of a vector and $(\cdot)_{\{m_1,\ldots,m_2\}}$ denotes a range of elements. Non-boldface letters denote the underlying infinite-length streaming signals.

It takes M streaming iterations for any sample of x_n to pass through the signal window and the SGP computes its estimate at every iteration. The final estimate of each sample is a weighted average over all the iterations, maintained and stored in $\overline{x}^i$. The diagonal matrix $\mathbf{W}$ contains the positive weights for the averaging, and has trace equal to unity: $\text{tr}(\mathbf{W}) = \sum_{i=1}^{M} w_{ii} = 1$.

Also note that since we use the old estimate of the support of the DFT coefficients during the refinement procedure, the circular shift of the old estimate in step 4 of Algorithm 1 only introduces a phase shift and does not disturb the support.

4.2. Signal estimation and refinement

The signal estimation and refinement step 2 of the recovery algorithm, summarized in Algorithm 2, is inspired by CoSaMP [8]. Both CoSaMP and SGP refine the signal estimate using the unexplained residual to identify a candidate support for the signal estimate (steps 1–3 in Alg. 2), inverting the measurements in that support to obtain a new signal estimate (step 4), and truncating the signal estimate to have the required sparsity (step 5).

However, three major modifications are required to adapt the refinement steps to the particular application. The first modifies the signal proxy formation to accommodate for the bad conditioning of the acquisition system around the DC component of the signal and the strong presence of such DC components. The second modifies the support identification to incorporate the signal model that dictates that nearby pixels have common support in the frequency domain. The third reduces the number of support coefficients added to the candidate support in step 3, in order to reduce the computation burden for on-line performance.

The refinement algorithm comprises the following steps:

Residual and proxy computation: This step uses the current signal estimate to compute the unexplained residual in the measurements. It then computes a proxy for the change in the signal that can explain the residual. The proxy is determined by correlating the residual with all the columns of $\widehat{\mathbf{A}}\mathbf{F}^{-1}$ and normalizing the resulting correlations by the weights in the diagonal matrix $\mathbf{W}_S$, computed using

$$(\mathbf{W}_S)_{i,i} = 1 \Big/ (\mathbf{F}\widehat{\mathbf{A}}^*\widehat{\mathbf{A}}\mathbf{F}^{-1})_{i,i}\ . \tag{7}$$

This weighting scheme is inspired by the minimum variance beamformer from in the array processing literature [12]. It aims to penalize the signal directions according to how much they are amplified by the proxy formation, and undo the bad conditioning of the measurement system around the DC components. This weighting can also be interpreted as the normalization of the columns of $\mathbf{A}\mathbf{F}^{-1}$.

Support identification and merger: This step combines the proxies determined from all the nearby pixels together with the existing signal support to determine a candidate support for the signal frequency components that explain the proxy. To determine the support, the magnitude of each frequency component in each of the proxies is squared and added to the squares of the magnitudes of the same frequency components from the nearby pixels in the block, as in (4). The T largest frequency components are identified from these sums and added to the existing support set for the signal. This step exploits the joint sparsity model of the signal, as described in [15].

Signal estimation: This step computes the least squares inverse of the measurements and estimates the signal coefficients over the candidate support, using conjugate gradient method.

Truncation: The final refinement step truncates the signal to have support size K, as the model dictates. As with the support identification step 3, the support is jointly updated over all the pixels in the block, using the joint sparsity model (4) on the signal estimates.

5. EXPERIMENTS

In this section, we present experimental results to demonstrate the performance of the SGP algorithm for reconstructing near-periodic signals from coded strobing measurements.

One Pixel reconstruction: The first experiment measures the reconstruction performance for one pixel of a video sequence. Each measurement integrates $R/2$ randomly chosen time samples of the

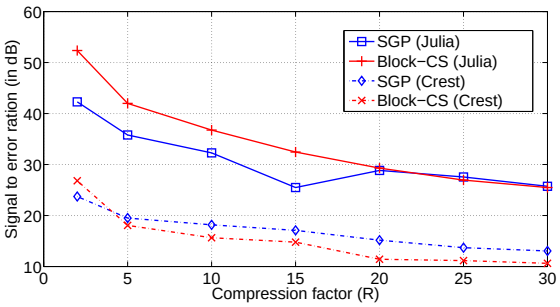

Fig. 2. Comparison between performance of SGP and Block-CS.

same pixel from R consecutive input samples, where R is the compression factor. To simulate the camera and electronics noise, we add Gaussian noise to the measurements to make the SNR equal to 35dB. We compare the reconstruction performance of SGP algorithm with the off-line processing described in [7], i.e., CoSaMP applied to the entire set of measurements, which we term Block-CS. We compare SGP and Block-CS with different values of R for two different signals.

The first signal is a *near-periodic* signal taken from the pixel time-series of a high-speed video of an oscillating Crest toothbrush, shown in Fig. 3. The period is roughly 16 samples and fluctuates over time due to variations in the movement of the brush. We used $M = 160, K = 30, T = 10$ and executed the SGP on 4800 samples. Similarly, we used the Block-CS to estimate the 4800 samples at once using the same $4800/R$ measurements, with $K = 4800/5R$. The results for different compression ratios are shown in Fig. 2, where SGP outperforms Block-CS.

The second signal is a perfectly periodic signal with period of 25 samples, constructed from the fractal image dataset Julia. The SGP is used with parameters $M = 150, K = 30, T = 10$, over a set of 4500 samples. Similarly, Block-CS estimates 4500 samples of the signal using the same $4500/R$ strobing measurements. Since the signal is perfectly periodic and has sparse DFT coefficients, Block-CS is able to exploit the stationarity within the larger observation time, and unsurprisingly outperforms the SGP. Notably, at higher compression factors the SGP is able to perform as well as Block-CS (Fig. 2 solid lines).

Note that the advantage of Block-CS, namely that it can observe the whole signal off-line and thus outperform the SGP for perfectly periodic signals, becomes a disadvantage in realistic scenarios, such as the Crest dataset, in which the signal is approximately periodic. In this case the SGP is able to adapt to the time variation and outperform Block-CS.

Video reconstruction: In this experiment we reconstruct the whole high-speed video of the pulsating Crest toothbrush with near-periodic linear and oscillatory motions, from coded strobing measurements. Three snapshots of an image section from Crest video sequence are shown in Fig 3. The strobing measurements are taken at different compression factors (R) and the original frame sequence is subsequently reconstructed using the SGP with parameters $M = 200$, $K = 30$, and $T = 10$.

In this experiment we use the joint-sparsity model in Algorithm 2 by jointly processing blocks of 4 adjacent pixels, and estimating their jointly sparse coefficients. Figure 3 shows snapshots of reconstructed videos at different compression rates and reports the corresponding signal to reconstruction error ratio (SER). Note that, as expected, quality of the reconstructed video degrades with increasing compression rate[2]. Quality can be improved by increasing the size of the sliding window (by increasing M) but this also leads to increased input-output delay and computational complexity.

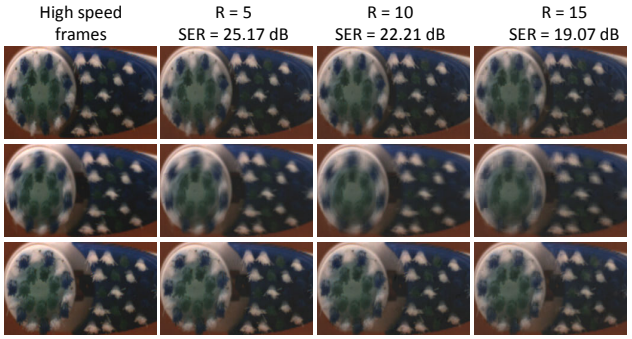

Fig. 3. Snapshots of Crest toothbrush video, with respective reconstructed frames and SER at compression ratio R.

6. REFERENCES

[1] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[2] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[3] E. Candès, "Compressive sampling," *Proceedings of the International Congress of Mathematicians, Madrid, Spain*, vol. 3, pp. 1433–1452, 2006.

[4] B. Bascle, A. Blake, and A. Zisserman, "Motion deblurring and super-resolution from an image sequence," *ECCV '96*, pp. 573–582, 1996.

[5] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," pp. 249–256, ACM SIGGRAPH, 2002.

[6] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: motion deblurring using fluttered shutter," pp. 795–804, ACM SIGGRAPH, 2006.

[7] A. Veeraraghavan, D. Reddy, and R. Raskar, "Coded strobing photography: Compressive sensing of high-speed periodic events," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. to appear.

[8] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples.," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, June 2008.

[9] P. T. Boufounos and M. S. Asif, "Compressive sampling for streaming signals with sparse frequency content," *CISS*, March 2010.

[10] J. A. Tropp, "Algorithms for simultaneous sparse approximation: part II: Convex relaxation," *Signal Processing*, vol. 86, no. 3, pp. 589–602, 2006.

[11] P. T. Boufounos, G. Kutyniok, and H. Rauhut, "Sparse recovery from combined fusion frame measurements." Submitted, http://arxiv.org/pdf/0912.4988, 2009.

[12] D. H. Johnson and D. E. Dudgeon, *Array Signal Processing: Concepts and Techniques*. Signal Processing Series, New Jersey, USA: Prentice-Hall, 1993.

[13] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[14] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk, "Beyond Nyquist: Efficient sampling of sparse, bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, pp. 520–544, Jan. 2010.

[15] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 56, pp. 1982 –2001, April 2010.

[2]Complete videos available at http://users.ece.gatech.edu/∼sasif/icip10