TRANSLATIONAL AND ROTATIONAL JITTER INVARIANT INCREMENTAL PRINCIPAL COMPONENT PURSUIT FOR VIDEO BACKGROUND MODELING

Paul Rodríguez [*]

Department of Electrical Engineering
Pontificia Universidad Católica del Perú
Lima, Peru

Brendt Wohlberg [†]

T-5 Applied Mathematics and Plasma Physics
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

ABSTRACT

While Principal Component Pursuit (PCP) is currently considered to be the state of the art method for video background modeling, it suffers from a number of limitations, including a high computational cost, a batch operating mode, and sensitivity to camera jitter. In this paper we propose a novel fully incremental PCP algorithm for video background modeling that is robust to translational and rotational jitter. It processes one frame at a time, obtaining similar results to standard batch PCP algorithms, while being able to deal with translational and rotational jitter. It also has extremely low memory footprint, and a computational complexity that allows almost real-time processing.

Index Terms— Principal Component Pursuit, Video Background Modeling, Rigid transformations

1. INTRODUCTION

Video background modeling, which consists of segmenting the moving objects or "foreground" from the static "background", is an important task in several applications. Principal Component Pursuit (PCP) is currently considered to be one of the leading algorithms for this problem [1]. The PCP optimization problem is defined by

$$\arg\min_{L,S} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad D = L + S \tag{1}$$

where $D \in \mathbb{R}^{m \times n}$ is the observed video of n frames, each of size $m = N_r \times N_c \times N_d$ (rows, columns and depth or channels respectively), $L \in \mathbb{R}^{m \times n}$ is a low rank matrix ($r \ll m, n$) representing the background, $S \in \mathbb{R}^{m \times n}$ is a sparse matrix representing the foreground, $\|L\|_*$ is the nuclear norm, $\sum_k |\sigma_k(L)|$, of matrix L, and $\|S\|_1$ is the ℓ^1 norm of S, seen as a long vector.

Although PCP provides state of the art performance [1] in the video background modeling problem, it has several limitations [1], three prominent ones being (i) its high computational cost, dominated by a partial SVD computation at each major outer loop, with a cost of $O(m \cdot n \cdot r)$ where $r = \text{rank}(L)$, (ii) the batch processing mode whereby a large number of frames have to be observed before starting any processing, resulting in high memory requirements, and (iii) its sensitivity to camera jitter, which can affect airborne and space-based sensors [2] as well as fixed ground-based cameras [1] subject to wind.

In previous work [3, 4, 5] we proposed incPCP, a fully incremental PCP algorithm for video background modeling, which is able to processes one frame at a time, obtaining similar results to batch PCP algorithms, while being able to adapt to changes in the background. Although incPCP has an extremely low memory footprint, and a computational complexity that allows real-time processing [3, 4], it shares the sensitivity to camera jitter of other PCP algorithms.

In this paper we propose to overcome the jitter sensitivity of the incPCP algorithm by incrementally solving

$$\min_{L^*, S, \mathcal{T}} \frac{1}{2} \|D - \mathcal{T}(L^*) - S\| + \lambda \|S\|_1 \quad \text{s.t.} \quad \text{rank}(L^*) \leq r \, , \tag{2}$$

where we assume that the observed frames D are misaligned due to camera jitter, the low-rank representation, L^*, is properly aligned, and that $\mathcal{T} = \{\mathcal{T}_k\}$ is a set of invertible and independent transformations such that when applied to each frame $D = \mathcal{T}(L^*) + S$ is satisfied. For the scope of this paper we focus on the case where $\mathcal{T}_k$ is a rigid transformation, i.e. the camera suffers from translational and rotational jitter. Our computational results show that the proposed method (i) delivers consistent results, similar to those obtained via standard batch PCP applied to fixed camera videos, (ii) has an extremely low memory footprint, and (iii) has a computational complexity that allows almost real-time processing.

2. PREVIOUS RELATED WORK

In this section we discuss other PCP methods that are considered to be robust to jitter, and summarize the mathematical methods used to develop the proposed translational and rotational jitter invariant incremental PCP algorithm.

[*]This research was supported by the "Fondo para la Innovación, la Ciencia y la Tecnología" (Fincyt) Program

[†]This research was supported by the NNSA's Laboratory Directed Research and Development Program.

2.1. Alternative PCP methods robust to camera jitter

2.1.1. *RASL algorithm*

The Robust Alignment by Sparse and Low-rank decomposition (RASL) algorithm [6] was introduced as a batch PCP method able to handle misaligned video frames by solving

$$\min_{L,S,\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t.} \quad \tau(D) = L + S \tag{3}$$

where $\tau(\cdot) = \{\tau_k(\cdot)\}$ is a set of independent transformations (one per frame), each having a parametric representation, such that $\tau(D)$ aligns all the observed video frames. (Note that $\tau(\cdot)$ is the inverse of $\mathcal{T}(\cdot)$ in (2)).

In [6], the non-linearity in (3), is handled via:

$$\min_{L,S,\Delta\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t.} \quad \tau(D)+\sum_{k=1}^{n} J_k \Delta\tau_k \epsilon_k = L + S \tag{4}$$

where J_k is the Jacobian of frame k with respect to transformation k and ϵ_k denotes the standard basis for $\mathbb{R}^n$. Computational results in [6] mainly focus on rigid transformations, and as long as the initial misalignment is not too large, (4) effectively recovers the correct transformations.

2.1.2. *t-GRASTA algorithm*

The GRASTA [7] method is an "on-line" algorithm for low rank subspace tracking. It uses a reduced number of frames, compared to the PCP problem (1), to estimate an initial low rank sub-space representation of the background and then processes one frame at a time. It must be emphasized that this procedure is not fully incremental since it uses a time subsampled version of all the available frames for initialization.

The transformed Grassmannian robust adaptive subspace tracking algorithm (t-GRASTA) [8] is based on the GRASTA [7] and RASL algorithms. It is able to handle misaligned video frames in an online, but not fully incremental, fashion. In particular the t-GRASTA method handles the misaligned video frames by solving (4) via a modified GRASTA algorithm. It uses a batch mode to train an initial low rank subspace considering the first p frames [9], with $p = 20$ by default. The initial transformation, τ in (4), is estimated by using a similarity transformation taking a group of points manually chosen from the corresponding original and canonical frames. Computational results in [8] focused on rigid transformations.

2.2. Image rotation as 1D convolutions

In [10] a simple FFT based algorithm was proposed to rotate an image around any given point. An image $\mathbf{d}^*$ can be rotated by α radians, denoted by $\mathbf{d} = R(\mathbf{d}^*, \alpha)$, via a collection of independent translations applied to each row and column: (1) translate each row by $\Delta_x = -y \cdot \tan(\alpha/2)$; (2) translate each column by $\Delta_y = x \cdot \sin(\alpha)$; (3) translate each row by $\Delta_x = -y \cdot \tan(\alpha/2)$. The center of rotation is defined by the origin of the x and y axes, which can be chosen as necessary.

The above described translations can be implemented as 1D convolutions of each row (column) with a filter-based translation operator in the spatial domain, or by 1D FFTs applied to each row (column) and then multiplied with a complex exponential with the proper phase. The computational complexity of this FFT-based implementation for a grayscale image of N_r rows and N_c columns is given by $O(10 \cdot N_c \cdot N_r(2 \cdot \log_2(N_r) + \log_2(N_c))$ (assuming that N_r and N_c are exact powers of 2), which is equivalent to computing three 2D FFTs.

2.3. Affinely Constrained Matrix Rank Minimization

In [11] several iterative hard thresholding (IHT) based algorithms are proposed to solve the affinely constrained matrix rank minimization problem. In particular the solution to

$$\min_{L} \; : \; \frac{1}{2}\|\mathcal{A}(L) - D\|_F^2 \quad \text{s.t. } \mathrm{rank}(L) \le r \tag{5}$$

is given by $L^{(j+1)} = \mathrm{partialSVD}(L^{(j)} - \mathcal{A}^*(\mathcal{A}(L^{(j)}) - D), r)$, where $\mathrm{partialSVD}(\cdot, r)$ represents the partial (thin) SVD which only considers the r largest singular values of the input and $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$.

2.4. amFastPCP algorithm [12]

Instead of solving (1) directly, the amFastPCP algorithm solves the equivalent alternating minimization

$$L^{(j+1)} = \arg\min_{L} \quad \|L + S^{(j)} - D\|_F^2 \text{ s.t. } \mathrm{rank}(L) \le r \tag{6}$$

$$S^{(j+1)} = \arg\min_{S} \quad \|L^{(j+1)} + S - D\|_F^2 + \lambda\|S\|_1, \tag{7}$$

where sub-problem (6) can be solved by computing a partial (with r components) SVD of $D - S^{(j)}$.

2.5. Incremental PCP algorithm [3, 4, 5]

This is an incremental algorithm constructed by exploiting the particular structure of amFastPCP [12]. If we assume that we have computed low-rank and sparse components L_k and S_k respectively, where $L_k + S_k = D_k$ and $D_k = D(:, 1:k)$ and that we know the partial (thin) SVD of $L_k = U_r \Sigma_r V_r^T$, where $\Sigma_r \in \mathbb{R}^{r \times r}$, then when the next frame $\mathbf{d}_{k+1}$ becomes available the computationally demanding (and batch) solution of sub-problem (6) can be efficiently computed via rank-1 modifications for thin SVD (see [13] and the many references therein) since $L_{k+1}^{(j+1)} = \mathrm{partialSVD}(D_{k+1} - S_{k+1}^{(j)}) = \mathrm{partialSVD}([L_k \; \mathbf{d}_{k+1}\text{-}\mathbf{s}_{k+1}^{(j)}])$, as $L_k = D_k\text{-}S_k$.

A Matlab-only implementation of this algorithm [3] (code available at [14]) running on a standard laptop (Intel i7-2670QM quad-core, 6GB RAM, 2.2 GHz) can process color videos of size 640×480 and 1920×1088 at a rate of 8 and 1.5 frames per second respectively. On the same hardware, an ANSI-C implementation [4] can deliver a rate of 49.6 and

7.2 frames per second for grayscale videos of size 640×480 and 1920×1088 respectively.

3. TRANSLATIONAL AND ROTATIONAL JITTER INVARIANT INCPCP ALGORITHM

The observed video sequence that suffers from jitter is represented by the matrix $D \in \mathbb{R}^{m \times n}$, where each video frame, labeled as $\mathbf{d}_k$ $k \in \{1, 2, \ldots, n\}$, is a column of D. This notation is also used for L and S, the low-rank and sparse components. Furthermore, we will assume that jitter-free video is represented by D^*, and that the observed video sequence results from applying the set of (rigid) transformations $\mathcal{T}_k$:

$$\mathbf{d}_k = \mathcal{T}_k(\mathbf{d}_k^*) = h_k * R(\mathbf{d}_k^*, \alpha_k) \tag{8}$$

where $R(\mathbf{d}_k^*, \alpha_k)$ denotes rotation by angle α and the filter h_k is an "uncentered" Dirac delta function.

3.1. Translation estimation

Inspired by [15], we use the following optimization procedure to estimate the translation between two images $\mathbf{u}$ and $\mathbf{b}$ related by $\mathbf{b} = h^* * \mathbf{u}$

$$h = \arg\min_{h} \frac{1}{2}\|h * \mathbf{u} - \mathbf{b}\|_2^2 + \gamma\|h\|_1, \tag{9}$$

since a purely translational relationship between two images can be modeled as a sparse filter, close to an "uncentered" Dirac delta function.

This problem can be solved by a variable splitting approach, transforming it into $\arg\min_{h,g} \frac{1}{2}\|g * \mathbf{u} - \mathbf{b}\|_F^2 + \gamma\|h\|_1 + \frac{\mu}{2}\|h - g\|_2^2$ and then applying alternating minimization on h and g. The h update is solved by soft-thresholding, and the g update can be efficiently solved in the Fourier domain via a special case of the method described in [16].

3.2. Rotation estimation

In Section 2.2 we briefly described a simple FFT based algorithm to rotate a given image around any given point. Without loss of generality, here we will assume that the rotation is about the center of the image.

Although there are several methods (e.g. [17, 18]), based on [10], that target rotational alignment, such algorithms assume either that a pseudopolar transformation is applied to the observed (rotated) image, or that the Fourier domain can be sampled with arbitrary trajectories. In what follows we describe a simple procedure (not previously exploited to the best of our knowledge) to estimate the rotation of one image relative to another.

We start by assuming that two observed images $\mathbf{u}$ and $\mathbf{b}$ are related by $\mathbf{b} = R(\mathbf{u}, \alpha^*)$. Then the solution of

$$\alpha = \arg\min_{\alpha} \frac{1}{2}\|R(\mathbf{u}, \alpha) - \mathbf{b}\|_2^2 \tag{10}$$

will estimate the angle α^* that relates $\mathbf{u}$ and $\mathbf{b}$. Unfortunately (10) is not convex in α; however, empirically (10) behaves as a concave functional with a unique minimum within a region close enough to the optimum, and since the rotation operation is computationally cheap (equivalent to three 2D FFTs) a line search procedure can be used to find its global minimizer (and thus estimate α). In particular, we choose to use a Fibonacci line search method based on [19] to solve (10).

3.3. Proposed Algorithm: incPCP_TI

Here we assume that objects of interest are at such a distance from the camera that the observed frame $\mathbf{d}_k$, which suffers from jitter, can be modeled as a random rigid transformation with respect to un-observed jitter-free frame.

Under this assumptions, (2) can be cast as

$$\arg\min_{L,S,H,\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_k \|h_k * R(\mathbf{l}_k^*, \alpha_k) + \mathbf{s}_k - \mathbf{d}_k\|_F^2 + \lambda\|S\|_1$$
$$+\gamma \sum_k \|h_k\|_1 \quad \text{s.t. } \mathrm{rank}(L^*) \le r; \tag{11}$$

where H represents the set of filters $\{h_k\}$, which models the translational jitter, and $\boldsymbol{\alpha}$ represents the set of angles $\{\alpha_k\}$ that models the rotational jitter. Furthermore we propose to incrementally solve (11) via the alternating minimization

$$\arg\min_{h_k} \frac{1}{2}\|h_k * R(\mathbf{l}_k^{*(j)}, \alpha_k^{(j)}) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2 + \gamma\|h_k\|_1, \tag{12}$$

$$\arg\min_{\alpha_k} \frac{1}{2}\|h_k^{(j+1)} * R(\mathbf{l}_k^{*(j)}, \alpha_k) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2, \tag{13}$$

$$\arg\min_{\mathbf{l}_k^*} \frac{1}{2}\|h_k^{(j+1)} * R(\mathbf{l}_k^*, \alpha_k^{(j+1)}) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2$$
$$\text{s.t. } \mathrm{rank}(L_k^*) \le r, \quad (\text{note that } L_k^* = [\mathbf{l}_1^*, \mathbf{l}_2^*, \ldots, \mathbf{l}_k^*]) \tag{14}$$

$$\arg\min_{\mathbf{s}_k} \frac{1}{2}\|h_k^{(j+1)} * R(\mathbf{l}_k^{*(j+1)}, \alpha_k^{(j+1)}) + \mathbf{s}_k - \mathbf{d}_k\|_F^2 + \lambda\|\mathbf{s}_k\|_1. \tag{15}$$

Sub-problems (12), (13) and (15) are simple to handle; i.e. (12), (13) can be solved via the procedures described in Sections 3.1 and 3.2 respectively, whereas the solution to (15) is just element-wise shrinkage ($\mathrm{shrink}(x, \epsilon) = \mathrm{sign}(x)\max\{0, |x| - \epsilon\}$). Sub-problem (14) is not as straightforwards as the other sub-problems; however, by noting that (i) the adjoint operator of rotation $R(., \alpha_k)$ is $R(., -\alpha_k)$ and (ii) the adjoint operator of the translation represented by filter $h_k(x, y)$ is the filter $h_k(-x, -y)$, then (14) can be effectively solved via the IHT described in Section 2.3.

4. COMPUTATIONAL RESULTS

We have used three real video sets as a test videos:

- V352: a 352 × 224 pixel, 1200-frame color video sequence of 40 seconds at 30 fps, from a sidewalk surveillance camera with real (mainly translational) jitter, also used in [20, Ch. 16] and in [9].
- V640: a 640 × 480 pixel, 400-frame color video sequence of 26.66 seconds at 15 fps, from the Lankershim Boulevard traffic surveillance dataset [21, cam. 3].
- V1920: a 1920 × 1088 pixel, 900-frame color video sequence of 36 seconds at 25 fps, from the Neovison2 public space dataset [22, Tower 3rd video].

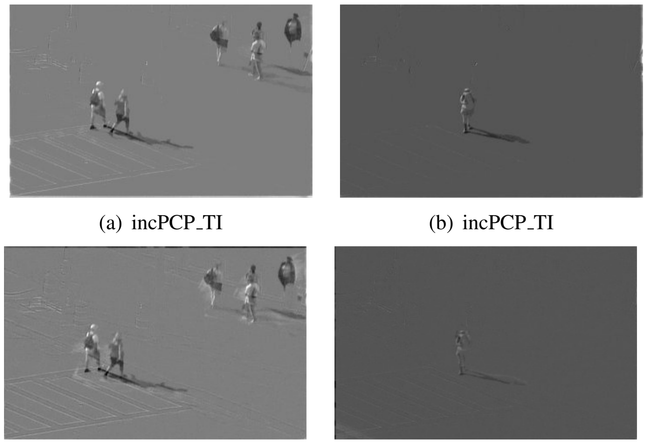
All simulations presented in this section, related to our proposed algorithm (see Section 3), have been carried out using Matlab-only code[1] running on an Intel i7-4710HQ quad-core (2.5 GHz, 6MB Cache, 32GB RAM) based laptop with a nvidia GTX980M GPU card. Videos "V640" and "V1920" were synthetically jittered with random uniformly distributed translations (±T pixel, T= $\{5, 10\}$) and rotations ($\pm\alpha$ degrees, $\alpha = \{0.5, 1\}$), and used as controlled datasets with known alignment. The jittered output has the same size as the input video frame. Our results are compared with t-GRASTA [8] implementation [9][2]. RASL [6] is not considered since it is a batch method.

Test video	Initialize (sec.)	Average per frame (sec.)				
	t-GRASTA grayscale	incPCP_TI grayscale		incPCP_TI color		t-GRASTA grayscale
V352	65^*	0.39^*	0.27^+	0.63^*	0.37^+	0.88^*
V640	232^*	1.49^*	0.60^+	2.45^*	0.82^+	3.40^*
V1920	1858^*	10.2^*	2.90^+	18.1^*	5.46^+	25.6^*

Table 1. Elapsed time to initialize the t-GRASTA [8] and average processing time per frame for t-GRASTA and incPCP_TI. Videos V640, V1920 have synthetic jitter: T= 5, $\alpha = 0.5$. *: standard Matlab. $^+$: GPU-enabled Matlab.

The computational performance, measured as the time to process a given video frame, including the jitter estimation, is compared with that of t-GRASTA in Table 1. The incPCP_TI algorithm is 2.5 $\sim$ 3 (standard Matlab version) and 4 $\sim$ 8.5 (GPU-enabled Matlab version) times faster than t-GRASTA even without considering the t-GRASTA initialization stage, which incPCP_TI does not have. Furthermore, the incPCP_TI sparse estimate is of better subjective quality than that of t-GRASTA (see Fig. 1).

An objective reconstruction quality measure is computed for the synthetically jittered videos as $\frac{\|S_{GT}-S_P\|_1}{N}$, where S_{GT} is the "ground truth" sparse video approximation of the unjittered video and S_P is the sparse approximation of a given method. S_{GT} is estimated via the batch iALM [24] algorithm using 20 outer iterations. The reconstruction quality for the grayscale "V640" case for two different synthetic jitter conditions (T= $\{5, 10\}$, $\alpha = 0.5$) is presented in Fig. 2. The incPCP_TI algorithm generates a sparse component similar to that generated by the baseline (computed via [3]), and it is of better quality than that of t-GRASTA, despite our best efforts to choose suitable parameters for t-GRASTA. For this case (video V640) the number of moving objects (cars) drastically increases from frame 200 onwards, which appear to severely affect the quality of the low rank subspace estimated by t-Grasta. Additional results, omitted here due to space constraints, are included in [23].


(a) incPCP_TI (b) incPCP_TI

(c) t-GRASTA (d) t-GRASTA

Fig. 1. Sparse estimates of frames 150 (a, c) and 750 (b, d) from the "V352" via incPCP_TI (a, b) and t-GRASTA (c, d).

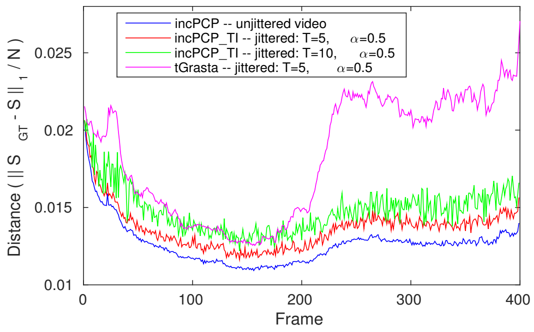


Fig. 2. Sparse approximation (V640) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the (i) sparse component computed via incPCP_TI method (red, green) and t-GRASTA (magenta) for different synthetic jitter conditions or (ii) the sparse component for the unjittered case (blue), computed via [3], provided as a baseline.

5. CONCLUSIONS

The proposed method, incPCP_TI, is able to solve the PCP problem at near real-time speed when the observed video suffers from translational and rotational jitter. Furthermore incPCP_TI has a low memory footprint (under tenfold the frame size) and computational results show that it is both faster and delivers a superior sparse representation to t-GRASTA, the sole non-batch PCP alternative for jittered videos.

[1]Publicly available [23]; it comes in two versions: (i) standard single-thread Matlab and (ii) another that uses GPU-enabled Matlab functions.

[2]t-GRASTA Matlab implementation for grayscale videos which takes advantage of MEX interfaces, run with its default parameters.

6. REFERENCES

[1] T. Bouwmans and E. Zahzah, "Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance," *Computer Vision and Image Understanding*, vol. 122, pp. 22–34, 2014.

[2] K. Simonson and T. Ma, "Robust real-time change detection in high jitter," Sandia Report SAND2009-5546, Sandia National Laboratories, 2009.

[3] P. Rodríguez and B. Wohlberg, "A Matlab implementation of a fast incremental principal component pursuit algorithm for video background modeling," in *IEEE Int'l. Conf. on Image Proc. (ICIP)*, Paris, France, Oct. 2014, pp. 3414–3416.

[4] P. Rodriguez, "Real-time incremental principal component pursuit for video background modeling on the TK1," Accepted, GPU Technical Conference, 2015.

[5] P. Rodriguez and B. Wohlberg, "Incremental principal component pursuit for video background modeling," Submitted, Springer Journal of Mathematical Imaging and Vision, 2015.

[6] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images," *IEEE Trans. Pattern Analysis & Machine Intelligence*, vol. 34, no. 11, pp. 2233 – 2246, 2012.

[7] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2012, pp. 1568–1575.

[8] J. He, D. Zhang, L. Balzano, and T. Tao, "Iterative Grassmannian optimization for robust image alignment," *Image and Vision Computing*, vol. 32, no. 10, pp. 800 – 813, 2014.

[9] J. He, D. Zhang, L. Balzano, and T. Tao, "t-GRASTA code," `https://sites.google.com/site/hejunzz/t-grasta`.

[10] M. Unser, P. Thevenaz, and L. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Trans. Image Process.*, vol. 4, no. 10, pp. 1371–1381, Oct. 1995.

[11] D. Goldfarb and S. Ma, "Convergence of fixed-point continuation algorithms for matrix rank minimization," *ArXiv e-prints*, Jun. 2009.

[12] P. Rodríguez and B. Wohlberg, "Fast principal component pursuit via alternating minimization," in *IEEE Int'l. Conf. on Image Proc. (ICIP)*, Melbourne, Australia, Sept. 2013, pp. 69–73.

[13] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Applications*, vol. 415, no. 1, pp. 20 – 30, 2006.

[14] P. Rodríguez and B. Wohlberg, "incremental PCP simulations," `http://sites.google.com/a/istec.net/prodrig/Home/en/pubs/incpcp`.

[15] B. Wohlberg, "Endogenous convolutional sparse representations for translation invariant image subspace models," in *IEEE Int'l. Conf. on Image Proc. (ICIP)*, Paris, France, Oct. 2014, pp. 2859–2863.

[16] B. Wohlberg, "Efficient convolutional sparse coding," in *IEEE Int'l Conf. on Acoustics, Speech, and Signal Proc. (ICASSP)*, Florence, Italy, May 2014, pp. 7173–7177.

[17] Y. Keller, A. Averbuch, and Moshe Israeli, "Pseudopolar-based estimation of large translations, rotations, and scalings in images," *IEEE Trans. Image Process.*, vol. 14, no. 1, pp. 12–22, Jan 2005.

[18] G. Vaillant, C. Prieto, C. Kolbitsch, G. Penney, and T. Schaeffter, "Retrospective rigid motion correction in k-space for segmented radial MRI," *IEEE Trans. Medical Imaging*, vol. 33, no. 1, pp. 1–10, Jan. 2014.

[19] H. Benson, "An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem," *J. of Global Optimization*, vol. 13, no. 1, pp. 1–24, 1998.

[20] T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, *Background Modeling and Foreground Detection for Video Surveillance*, Chapman and Hall/CRC, July 2014.

[21] "Lankershim Boulevard dataset," Jan. 2007, U.S. Department of Transportation Publication FHWA-HRT-07-029, data available from `http://ngsim-community.org/`.

[22] "USC Neovision2 Project," DARPA Neovision2, data available from `http://ilab.usc.edu/neo2/`.

[23] P. Rodríguez and B. Wohlberg, "Rigid transformation invariant incremental PCP simulations," `http://sites.google.com/a/istec.net/prodrig/Home/en/pubs/incpcpti`.

[24] Z. Lin, M. Chen, and Y. Ma, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," arXiv:1009.5055v2, 2011.