J Math Imaging Vis
DOI 10.1007/s10851-015-0610-z

Incremental Principal Component Pursuit for Video Background Modeling

Paul Rodriguez · Brendt Wohlberg

Received: 15 January 2015 / Accepted: 9 October 2015

Abstract Video background modeling is an important preprocessing step in many video analysis systems. Principal Component Pursuit (PCP), which is currently considered to be the state-of-the-art method for this problem, has a high computational cost, and processes a large number of video frames at a time, resulting in high memory usage and constraining the applicability of this method to streaming video.

In this paper we propose a novel fully incremental PCP algorithm for video background modeling. It processes one frame at a time, obtaining similar results to standard batch PCP algorithms, while being able to adapt to changes in the background. It has an extremely low memory footprint, and a computational complexity that allows real-time processing.

Keywords Principal Component Pursuit · Video Background Modeling · incremental Singular Value Decomposition

Paul Rodriguez
Electrical Engineering Department
Pontificia Universidad Católica del Perú, Lima, Peru
Tel.: +511-626-2000
E-mail: prodrig@pucp.edu.pe
This research was supported by the "Fondo para la Innovación, la Ciencia y la Tecnología" (Fincyt) Program.
There is a patent application number 14/722,651 that covers the incremental PCP method described in this document.

Brendt Wohlberg
Theoretical Division
Los Alamos National Laboratory, Los Alamos, NM 87545 Tel: +1-505-667-6886
E-mail: brendt@lanl.gov
This research was supported by the U.S. Department of Energy through the LANL/LDRD Program and by UC Lab Fees Research grant 12-LR-236660.

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. In the present paper we restrict our interest to videos acquired by a static sensor, as is common in surveillance systems e.g. [6], which makes this problem more tractable.

Two recent publications [14, 7] have presented a systematic evaluation and comparative analysis of several Principal Component Pursuit (PCP) / Robust Principal Component Analysis (RPCA) [45,10] based algorithms, which are considered to be the state-of-the-art for the video background modeling problem (see [40] for a survey of alternative methods). In this context, the PCP problem [10, Eq. 1.1] is

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

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 representing the background, $S \in \mathbb{R}^{m\times n}$ is a sparse matrix representing the foreground, $\|L\|_*$ is the nuclear norm of matrix L (i.e. $\sum_k |\sigma_k(L)|$, the sum of the singular values of L), and $\|S\|_1$ is the ℓ^1 norm of S.

Numerical algorithms for solving (1) (for a complete list see [7]) are usually based on splitting methods [10, 26,3,28], such as the augmented Lagrange multiplier (ALM) method [24,23] or its variants, in which (1) is

solved via the problem

$$\arg\min_{L,S,Y} \|L\|_* + \lambda\|S\|_1 + \langle Y, D - L - S\rangle + \frac{\mu}{2}\|D - L - S\|_F \ , \quad (2)$$

the computation including a full or partial Singular Value Decomposition (SVD) depending on the ALM variant.

Although [14,7] show that PCP provides state-of-the-art performance in the video background modeling problem, [7] also acknowledges several limitations of the PCP method. Of particular relevance to the present work are its high computational cost and that the PCP method is inherently a batch method, in that a large number of frames have to be observed before starting any processing.

The computational cost of most PCP algorithms is dominated by a partial SVD computation at each outer loop, with a cost of $O(m \cdot n \cdot r)$, where $r = \text{rank}(L)$. It is also important to emphasize the significant overhead of memory transfers due to the typical size of matrix D. For instance, in the case of a 400 frame (13.3 seconds at 30 fps) 640×480 color video, the size of D is 921600×400, equivalent to 1.925 Gb in single precision floating-point representation. Likewise, in the case of a 900 frame (36 seconds at 25 fps) 1920×1088 (HD) color video, the size of D is 6266880×900, equivalent to 22.6 Gb in single precision floating-point representation.

In this paper we develop a novel incremental PCP algorithm that fundamentally differs from current state-of-the-art PCP algorithms: (i) as the name suggests, the proposed algorithm can process an input video in a fully incremental fashion, that is, one frame at a time, whereas most existing PCP algorithms are batch methods or have a batch initialization; (ii) it has a trivial initialization stage, usually needing only a handful of frames to converge to the right background, whereas other online algorithms have a batch initialization or need a larger number of frames to converge to the right background; (iii) its computational cost is $O(14 \cdot m \cdot r \cdot p)$ (assuming $r \ll m, n$) per frame, where p is the number of inner loops (usually between 1 and 3); (iv) its memory footprint is $O(3 \cdot m) + O(m \cdot r)$. Furthermore our proposed algorithm has the ability to adapt to changes in the background, as do other non-batch/online algorithms.

To the best of our knowledge, the proposed algorithm is a novel approach. In Section 2 we give an intuitive description of the proposed algorithm, where we highlight the key ideas behind our method and introduce its main properties. We then proceed to summarize, in Section 3, the related works regarding the proposed algorithm; specifically, in Section 3.1 the most closely related algorithms are briefly described, and in Section 3.2 we include a description of the batch precursor method used to develop the proposed incremental PCP algorithm. In Section 4 we give a detailed description of rank-1 modifications for thin SVD, since they are at the core of the proposed algorithm. In Section 5 we give a full description of the proposed method and in Section 6 we present our experimental results and comparisons where we provide computational evidence of the performance of the proposed method. Finally our concluding remarks are listed in Section 7.

2 Intuitive description of the proposed algorithm

It is useful to recall that the PCP problem (1) is derived as the convex relaxation of the original problem [45, Section 2]

$$\arg\min_{L,S} \ \text{rank}(L) + \lambda\|S\|_0 \ \text{ s.t. } D = L + S \ , \quad (3)$$

based on decomposing matrix D such that $D = L + S$, with low-rank L and sparse S. While most PCP algorithms, including the Augmented Lagrange Multiplier (ALM) and inexact ALM (iALM) algorithms [24,23] are directly based on (1), this is not the only possible tractable problem that can be derived from (3). In particular, changing the constraint $D = L+S$ to a penalty, the rank penalty to an inequality constraint, including the usual relaxation of the ℓ_0 norm by an ℓ_1 norm, leads to the problem

$$\arg\min_{L,S} \frac{1}{2}\|L+S-D\|_F^2 + \lambda\|S\|_1 \quad \text{s.t. } \text{rank}(L) \leq r \ . \quad (4)$$

A computationally efficient solution can be found via an alternating optimization (AO) [5] procedure, since it seems natural to split (4) into a low-rank approximation, i.e. $\arg\min_L \frac{1}{2}\|L+S-D\|_F^2$ s.t. $\text{rank}(L) \leq r$, with fixed S, followed by a shrinkage, i.e. $\arg\min_S \frac{1}{2}\|L+S-D\|_F^2 + \lambda\|S\|_1$, with fixed L computed in the previous iteration. If the low-rank approximation sub problem could be solved via a computationally efficient incremental procedure, for instance, based on rank-1 modifications for thin SVD [9,11,4,8], then (4) could be easily solved incrementally, provided that the shrinkage step could be also computed in a frame-wise, incremental fashion.

Several questions arise from this tentative algorithm based on the above statement; among others: (i) does the proposed AO procedure for solving (4) converge? (ii) what kind of initialization would be needed for the incremental procedure (fully incremental or batch)? (iii) how robust is the initialization for different setups, e.g.

how does moving objects occluding the background, in the initialization stage, affect the performance? (iv) is it possible to adapt to changes in the background, and could this adaptation be also done incrementally? (v) assuming that the answer to the previous question is positive, is it possible to always use a rank-1 subspace to model the background, and if so, would this lead to a fast implementation?

These questions are answered in the relevant parts of the present paper. In Section 3.2 we address the convergence question raised in (i); we discuss our initialization procedure, questions (ii) and (iii), in Section 5, and give computational support to our claims in Section 6. As argued in Section 5, our algorithm can quickly adapt to changes in the background (question (iv)), with related results shown in Section 6. Finally, our computational results presented in Section 6 show that it is indeed possible to use a rank-1 subspace to model the background (question (v)) and that the overall computational performance of our algorithm is faster than competing state-of-the-art algorithms.

3 Alternative algorithms and related work

In this section we briefly summarize the batch precursor method used to develop the proposed incremental PCP algorithm: the fast alternating minimization PCP (amFastPCP) algorithm [35,36]. We also include a brief description of other PCP-like methods that are considered to be incremental procedures.

3.1 Alternative incremental / on-line PCP methods

To the best of our knowledge, recursive projected compressive sensing (ReProCS) [30] (and similar ideas by the same authors [29,31,13]) along with Grassmannian robust adaptive subspace tracking algorithm (GRASTA) [17], ℓ_p-norm robust online subspace tracking (pROST) [39] and Grassmannian online subspace updates with structured sparsity (GOSUS) [48] are the only PCP-like methods for the video background modeling problem that are considered to be incremental. However all of them have a batch initialization/training stage as the default/recommended initial background estimation[1].

ReProCS [30] is not a real-time algorithm since it uses a batch method in its SVD-based initialization step. Furthermore, in its original form, it cannot process real videos where multiple moving objects enter and leave the field of view of the camera. Although [30] can take advantage of a known model for the trajectories of the moving objects to improve the performance, in [13] this optional model assumption is dropped. Interestingly, [13] makes use of incremental SVD procedures to adapt to slow changes in the background; this modified algorithm, called "practical ReProCS", also needs a batch initialization.

GRASTA [15,17] is presented as an "on-line" algorithm for low rank subspace tracking. It can select a few frames (either randomly or from the initial part of the video, [15, Section 4.5.1]), or use a time sub-sampled version of all the available frames for its background initialization [17, Section 3]; both possible initialization stages are batch methods. Although GRASTA can estimate and track non-stationary backgrounds, its batch initialization step can have a relatively high complexity.

pROST [39] is closely related to GRASTA, but instead of using an ℓ_1 norm of the singular values to estimate the low rank subspace representation of the background it uses an ℓ_p norm ($p < 1$). Experimental results [39] show that pROST can outperform GRASTA in the case of dynamic backgrounds.

At a high level, GOSUS [48] is similar to GRASTA, but GOSUS uses a small number of frames from the initial part of the video to be analyzed for its batch initialization stage, and then proceeds to update the background. GOSUS can also be initialized with a random sub-space (representing the background), but in [48, Section 5.1] it is stated that this initialization takes up to 200 frames to converge to the right background sup-space. Although GOSUS has better tracking properties than GRASTA (see Section 6), its computational cost is higher.

Finally we mention that there exist several methods for video background modeling based on the (general) Maximum-Margin Matrix Factorization (M3F) method (see [41] and derived works). While this topic is related to PCP, it exceeds the scope of the present work. Nevertheless, we point out that the Probabilistic Approach to Robust Matrix Factorization (PRMF) [43] and Robust PCA as bilinear decomposition (RPCA-BL) [25] methods, also target the video background modeling problem and include "online" extensions to their batch algorithms. Both PRMF and RPCA-BL use the key M3F observation (5) to derive their algorithms, where U and V are the factors of matrix L.

$$\min_{L} \|L\|_* = \min_{\substack{U,V \\ L=UV^T}} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) \; . \tag{5}$$

In particular, PRMF is based on the robust PCA model

$$\min_{U,V} \|D - UV^T\|_1 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \; , \tag{6}$$

[1] The GRASTA and GOSUS algorithms can perform the initial background estimation in a non-batch fashion, however the resulting performance is not as good as when the default batch procedure is used, as shown in Section 6.

which is solved using a parallelizable expectation maximization (EM) algorithm, which includes an online extension that uses a batch initialization. Likewise, RPCA-BL uses the model

$$\min_{U,V} \|D-UV^T-S\|_F+\frac{\lambda_1}{2}(\|U\|_F^2+\|V\|_F^2)+\lambda_2\|S\|_{2,r} \quad (7)$$

where $\|.\|_{2,r}$ represents the row-wise ℓ^2-norm. This model is solved via an alternating optimization procedure that also includes an online extension with a batch initialization.

Algorithm 1: amFastPCP [35]

Inputs : Observed video $D \in \mathbb{R}^{m\times n}$, regularization parameter λ, eigenvalue tolerance τ.

Initialization: $S_0 = 0$, initial rank r.

for $j = 0$: *mLoops* **do**

1 $[U, \Sigma, V] = \text{partialSVD}(D - S^{(j)}, r)$

2 $L^{(j+1)} = U * \Sigma * V$

3 if $\frac{\sigma_r}{\sum_{l=1}^{r}\sigma_l} > \tau$ then $++r$

4 $S^{(j+1)} = \text{shrink}(D - L^{(j+1)}, \lambda)$

3.2 amFastPCP: Fast alternating minimization PCP

A computationally efficient algorithm, amFastPCP, was recently proposed [35] for solving the batch PCP problem. Instead of solving (1) directly, the approach is to solve (4), reproduced here for convenience

$$\arg\min_{L,S} \frac{1}{2}\|L+S-D\|_F^2+\lambda\|S\|_1 \text{ s.t. } \text{rank}(L)\leq r\ . \quad (8)$$

While (8), with some minor variations, is also the starting point for [49,46], the final algorithmic forms of these independently developed methods do not resemble that of the amFastPCP algorithm (see Algorithm 1).

In [35], (8) is solved via the alternating minimization

$$L^{(j+1)} = \arg\min_{L} \|L+S^{(j)}-D\|_F^2 \text{ s.t. } \text{rank}(L)\leq r \quad (9)$$

$$S^{(j+1)} = \arg\min_{S} \|L^{(j+1)}+S-D\|_F^2+\lambda\|S\|_1, \quad (10)$$

which is summarized in Algorithm 1. Sub-problem (9) can be solved by computing a partial SVD of $D - S^{(j)}$ with r components, which is the only computationally demanding part of the algorithm. In Algorithm 1 this is computed in lines 1 and 2, which require $O(m \cdot n \cdot r)$ and $O(2 \cdot m \cdot n \cdot r)$ flops per outer loop respectively. The solution to (10) is simple element-wise shrinkage (soft thresholding)

$$\text{shrink}(D - L^{(j+1)}, \lambda)\ , \quad (11)$$

where

$$\text{shrink}(x,\epsilon) = \text{sign}(x)\max\{0, |x| - \epsilon\}\ . \quad (12)$$

The convergence of the AO method based on (9)-(10) to the global minimizer of (8) can be proven by showing that it complies with the general convergence conditions for AO methods [5, Section 6], which in this case are that (i) (9) and (10) have each a unique global minimizer, and (ii) at each iteration of (9)-(10) the cost functional of (8) is reduced:

- let $f(L, S^{(j)})$ denote $\arg\min_L \|L + S^{(j)} - D\|_F^2$ s.t. $\text{rank}(L) \leq r$, then

  $$f(L, S^{(j)}) \geq f(L^{(j+1)}, S^{(j)})\ \ \forall L\,,$$

  where $L^{(j+1)}$ is the solution to (9); then, in particular

  $$f(L^{(j)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j)})\,;$$

- let $f(L^{(j+1)}, S)$ denote $\arg\min_S \|L^{(j+1)}+S-D\|_F^2+\lambda\|S\|_1$, then

  $$f(L^{(j+1)}, S) \geq f(L^{(j+1)}, S^{(j+1)})\ \ \forall S\,,$$

  where $S^{(j+1)}$ is the solution to (10); then, in particular

  $$f(L^{(j+1)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j+1)})\,;$$

- therefore
  $f(L^{(j)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j+1)})$.

Based on the observation that for the video background modeling application, the low-rank component L typically has very low rank, [35] proposed a simple procedure to estimate an upper bound for r in (9), described in line 3 of Algorithm 1. Since the singular values of L_{k+1} are a by-product of solving (9) at each outer loop, we can increment r and test the contribution of the new singular vector. When this contribution becomes small enough (less that 1% of the sum all other singular values) we stop incrementing r. Experimental simulations in [35,36] showed that the rule described in Algorithm 1 typically increases r up to a value of 6 or less, which is consistent with the rank estimation performed by other batch PCP algorithms, such the inexact ALM, etc.

The solution obtained via the iterative solution of (9)-(10) is of comparable quality to the solution of the original PCP problem (see [35] for details), being approximately an order of magnitude faster than the inexact ALM [23] algorithm to construct a sparse component of similar quality.

Finally we note that lines 1 and 2 (or sub-problem (9)), are the ones that impose the "batch" property of the amFastPCP algorithm, as well as its memory requirements ($O(2 \cdot m \cdot n)$).

4 Incremental and rank-1 modifications for thin SVD

In this section we give a full description of incremental and rank-1 modifications for thin SVD, originally described in [9,8,11,4]. First, we recall that the full SVD (singular value decomposition) of a matrix $D \in \Re^{m \times n}$ is given by

$$D = U \Sigma V^T \tag{13}$$

where

$$U^T U = I_m \qquad V^T V = I_n \ , \tag{14}$$

and

$$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \ , \tag{15}$$

and I_k is the identity matrix of size $k \times k$ and the diagonal elements σ_k of Σ are non-negative and non-increasing.

Before proceeding, note that the computational complexity of all of the procedures described below (see [11, Section 3], [8, Section 4]) is upper bounded by $O(10 \cdot m \cdot r) + O(r^3) + O(3 \cdot r \cdot l)$. If $r \ll m, l$ holds, then the complexity is dominated by $O(10 \cdot m \cdot r)$.

4.1 Incremental or update thin SVD

Given matrix $D \in \mathbb{R}^{m \times l}$ with thin SVD $D = U_0 \Sigma_0 V_0^T$ where $\Sigma_0 \in \mathbb{R}^{r \times r}$ and vector $\mathbf{d} \in \mathbb{R}^m$, we want to compute thin SVD($[D\ \mathbf{d}]$) $= U_1 \Sigma_1 V_1^T$, with (i) $\Sigma_1 \in \mathbb{R}^{r+1 \times r+1}$ or (ii) $\Sigma_1 \in \mathbb{R}^{r \times r}$. Noting that

$$[D\ \mathbf{0}] = U_0 \Sigma_0 [V_0\ \mathbf{0}]^T \ ,$$

then

$$\begin{aligned} [D\ \mathbf{d}] &= [D\ \mathbf{0}] + \mathbf{d}\mathbf{e}^T \\ &= U_0 \Sigma_0 [V_0\ \mathbf{0}]^T + \mathbf{d}\mathbf{e}^T \\ &= [U_0\ \mathbf{d}] \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} V_0^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \end{aligned} \tag{16}$$

where $\mathbf{0}$ is a zero column vector of the appropriate size and $\mathbf{e} \in \mathbb{R}^{l+1}$ is a unit vector with $\mathbf{e}(l+1) = 1$.

Considering the Gram-Schmidt orthonormalization of $\mathbf{d}$ w.r.t. U_0

$$\begin{aligned} &\mathbf{x} = U_0^T \mathbf{d}, \qquad \mathbf{z}_x = \mathbf{d} - U_0 \mathbf{x} = \mathbf{d} - U_0 U_0^T \mathbf{d}, \\ &\rho_x = \|\mathbf{z}_x\|_2, \qquad \mathbf{p} = \frac{1}{\rho_x} \mathbf{z}_x \end{aligned} \tag{17}$$

then (16) can be written as

$$[D\ \mathbf{d}] = [U_0\ \mathbf{p}] \begin{bmatrix} \Sigma_0 & \mathbf{x} \\ \mathbf{0}^T & \rho_x \end{bmatrix} \begin{bmatrix} V_0^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{18}$$

since

$$U_0 \mathbf{x} + \rho_x \mathbf{p} = U_0 U_0^T \mathbf{d} + (\mathbf{d} - U_0 U_0^T \mathbf{d}) = \mathbf{d} \ . \tag{19}$$

Then, after computing full SVD of the middle matrix (size $(r+1) \times (r+1)$) of the right-hand-side of (18)

$$\mathrm{SVD}\left(\begin{bmatrix} \Sigma_0 & \mathbf{x} \\ \mathbf{0}^T & \rho_x \end{bmatrix} \right) = G \hat{\Sigma} H^T, \tag{20}$$

(18) becomes

$$[D\ \mathbf{d}] = [U_0\ \mathbf{p}] \cdot (G \hat{\Sigma} H^T) \cdot \begin{bmatrix} V_0^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{21}$$

where we note that by taking

$$U_1 = [U_0\ \mathbf{p}] \cdot G, \ \ \Sigma_1 = \hat{\Sigma} \text{ and } V_1 = \begin{bmatrix} V_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot H, \tag{22}$$

we obtain the thin SVD of $[D\ \mathbf{d}]$ with $r+1$ singular values. In the case in which we are interested, the non-increasing thin SVD of $[D\ \mathbf{d}]$ (which considers only the r largest singular values), we take

$$\begin{aligned} &U_1 = U_0 \cdot G(1\!:\!r, 1\!:\!r) + \mathbf{p} \cdot G(r+1, 1\!:\!r), \\ &\Sigma_1 = \hat{\Sigma}(1\!:\!r, 1\!:\!r), \\ &V_1 = [V_0 \cdot H(1\!:\!r,\ 1\!:\!r);\ H(r+1,\ 1\!:\!r)], \end{aligned} \tag{23}$$

where Matlab notation is used to indicate array slicing operations.

4.2 Downdate thin SVD

Given $[D\ \mathbf{d}] = U_0 \Sigma_0 V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r \times r}$, $D \in \mathbb{R}^{m \times l}$ and $\mathbf{d} \in \mathbb{R}^m$, we want to compute thin SVD(D) $= U_1 \Sigma_1 V_1^T$ with r singular values.

Noting that $[D\ \mathbf{0}] = [D\ \mathbf{d}] + (-\mathbf{d})\mathbf{e}^T$, where $\mathbf{e} \in \mathbb{R}^{l+1}$ is a unit vector and $\mathbf{e}(l+1) = 1$, then

$$[D\ \mathbf{0}] = U_0 \Sigma_0 V_0 - \mathbf{d}\mathbf{e}^T = [U_0\ \text{-}\mathbf{d}] \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} V_0^T \\ \mathbf{e}^T \end{bmatrix}. \tag{24}$$

Considering the Gram-Schmidt orthonormalization of $\mathbf{e}$ w.r.t. V_0

$$\begin{aligned} &\mathbf{y} = V_0^T \mathbf{e} = V_0(l+1, :)^T, \qquad \mathbf{z}_y = \mathbf{e} - V_0 \mathbf{y} \\ &\rho_y = \|\mathbf{z}_y\|_2 = \sqrt{1 - \mathbf{y}^T \mathbf{y}}, \qquad \mathbf{q} = \frac{1}{\rho_y} \mathbf{z}_y, \end{aligned} \tag{25}$$

as well as the Gram-Schmidt orthonormalization of -$\mathbf{d}$ w.r.t. U_0 while noting that $\mathbf{d} = U_0 \Sigma_0 \mathbf{y}$

$$\begin{aligned} &\mathbf{x} = -U_0^T \mathbf{d} = -\Sigma_0 \mathbf{y}, \qquad \rho_x = 0, \qquad \mathbf{p} = \mathbf{0}, \\ &\mathbf{z}_x = -\mathbf{d} - U_0 \mathbf{x} = -\mathbf{d} + U_0 \Sigma_0 \mathbf{y} = \mathbf{0}, \end{aligned} \tag{26}$$

then (24) can be written as

$$[D\ \mathbf{0}] = [U_0\ \mathbf{0}] \begin{bmatrix} \Sigma_0\text{-}\Sigma_0\mathbf{y}\mathbf{y}^T & \text{-}\rho_y \cdot \Sigma_0\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix} [V_0\ \mathbf{q}]^T \tag{27}$$

where it is straightforward to verify that the right-hand-side of (27) is equal to $[D\ \mathbf{d}] + (-\mathbf{d})\mathbf{e}^T$. Then, after computing full SVD of the middle matrix (size $r+1 \times r+1$) of the right-hand-side of (27)

$$\text{SVD}\left(\begin{bmatrix} \Sigma_0\text{-}\Sigma_0\mathbf{y}\mathbf{y}^T & \text{-}\rho_y \cdot \Sigma_0\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix} \right) = G\hat{\Sigma}H^T, \tag{28}$$

(27) becomes

$$[D\ \mathbf{0}] = [U_0\ \mathbf{0}] \cdot (G\hat{\Sigma}H^T) \cdot [V_0\ \mathbf{q}]^T \tag{29}$$

To obtain the thin SVD of D with r singular values, we take

$$\begin{aligned} U_1 &= U_0 \cdot G(1:r, 1:r) \\ \Sigma_1 &= \hat{\Sigma}(1:r, 1:r) \\ V_1 &= V_0 \cdot H(1:r, 1:r) + \mathbf{q} \cdot H(r+1, 1:r) \end{aligned} \tag{30}$$

While this procedure has been described for downdating the contribution of the last column of a matrix when computing its thin SVD, with minor variations this procedure can downdate the contribution of any given column.

4.3 Thin SVD replace

Given $[D\ \mathbf{d}] = U_0\Sigma_0V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r\times r}$ we want to compute thin SVD$([D\ \hat{\mathbf{d}}]) = U_1\Sigma_1V_1^T$ with r singular values.

Noting that $[D\ \hat{\mathbf{d}}] = [D\ \mathbf{d}] + \mathbf{c}\mathbf{e}^T$, where $\mathbf{c} = \hat{\mathbf{d}}\text{-}\mathbf{d}$, then

$$[D\,\hat{\mathbf{d}}] = [U_0\,\mathbf{c}] \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} V_0^T \\ \mathbf{e}^T \end{bmatrix}. \tag{31}$$

In this case the Gram-Schmidt orthonormalization of $\mathbf{e}$ w.r.t. V_0 is given by (25). Recalling that $\mathbf{d} = U_0\Sigma_0\mathbf{y}$ and noting that $U_0U_0^T\mathbf{c} = U_0U_0^T\hat{\mathbf{d}} - \mathbf{d}$ (since $U_0U_0^T\mathbf{d} = U_0U_0^TU_0\Sigma_0\mathbf{y} = U_0\Sigma_0\mathbf{y}$) the Gram-Schmidt orthonormalization of $\mathbf{c}$ w.r.t. U_0 is given by

$$\begin{aligned} &\mathbf{x} = U_0^T\mathbf{c} = U_0^T\hat{\mathbf{d}} - \Sigma_0\mathbf{y}, \quad \rho_x = \|\mathbf{z}_x\|_2, \quad \mathbf{p} = \frac{1}{\rho_x}\mathbf{z}_x, \\ &\mathbf{z}_x = \mathbf{c} - U_0\mathbf{x} = \hat{\mathbf{d}} - \mathbf{d} - U_0U_0^T\mathbf{c} = \hat{\mathbf{d}} - U_0U_0^T\hat{\mathbf{d}}, \end{aligned} \tag{32}$$

then (31) can be written as

$$[D\,\hat{\mathbf{d}}] = [U_0\ \mathbf{p}] \begin{bmatrix} \Sigma_0 + \mathbf{x}\mathbf{y}^T & \rho_y \cdot \mathbf{x} \\ \rho_x \cdot \mathbf{y}^T & \rho_x \cdot \rho_y \end{bmatrix} [V_0\ \mathbf{q}]^T \tag{33}$$

where it is straightforward to verify that the right-hand-side of (33) is equal to $[D\ \mathbf{d}] + \mathbf{c}\mathbf{e}^T$. Then, after computing full SVD of the middle matrix (size $r+1\times r+1$) of the right-hand-side of (33)

$$\text{SVD}\left(\begin{bmatrix} \Sigma_0 + \mathbf{x}\mathbf{y}^T & \rho_y \cdot \mathbf{x} \\ \rho_x \cdot \mathbf{y}^T & \rho_x \cdot \rho_y \end{bmatrix} \right) = G\hat{\Sigma}H^T, \tag{34}$$

(33) becomes

$$[D\ \hat{\mathbf{d}}] = [U_0\ \mathbf{p}] \cdot (G\hat{\Sigma}H^T) \cdot [V_0\ \mathbf{q}]^T. \tag{35}$$

To obtain the thin SVD of $[D\ \hat{\mathbf{d}}]$ with r singular values, we take

$$\begin{aligned} U_1 &= U_0 \cdot G(1:r, 1:r) + \mathbf{p} \cdot G(r+1, 1:r), \\ \Sigma_1 &= \hat{\Sigma}(1:r, 1:r) \\ V_1 &= V_0 \cdot H(1:r, 1:r) + \mathbf{q} \cdot H(r+1, 1:r). \end{aligned} \tag{36}$$

5 Incremental PCP algorithm

5.1 Proposed algorithm: incPCP

We start our description assuming that we have computed L_{k-1} (low-rank) and S_{k-1} (sparse), where $L_{k-1}+S_{k-1} = D_{k-1}$ and $D_{k-1} = D(:, 1:k-1)$ and that we know the partial (thin) SVD of $L_{k-1} = U_r\Sigma_rV_r^T$, where $\Sigma_r \in \mathbb{R}^{r\times r}$. This initialization can also be performed incrementally, as described in Section 5.2.

If we were to solve the PCP problem from scratch when the next frame $\mathbf{d}_k$ is available via the amFastPCP algorithm, then we need to minimize

$$\frac{1}{2}\|L_k + S_k - D_k\|_F^2 + \lambda\|S_k\|_1 \text{ s.t. } \text{rank}(L_k) \le r, \tag{37}$$

via (9) and (10), reproduced here for convenience:

$$\begin{aligned} L_k^{(j+1)} &= \arg\min_L \|L_k + S_k^{(j)} - D_k\|_F^2 \\ &\qquad \text{s.t. } \text{rank}(L_k) \le r \end{aligned} \tag{38}$$

$$S_k^{(j+1)} = \arg\min_S \|L_k^{(j+1)} + S_k - D_k\|_F^2 + \lambda\|S_k\|_1, \tag{39}$$

where $L_k = [L_{k-1}\ \mathbf{l}_k]$, $S_k = [S_{k-1}\ \mathbf{s}_k]$ and $D_k = [D_{k-1}\ \mathbf{d}_k]$.

When $j = 0$, the minimizer of (38) is given by

$$L_k^{(1)} = \text{partialSVD}(D_k - S_k^{(0)})\ . \tag{40}$$

Since $D_k - S_k^{(0)} = [D_{k-1}\text{-}S_{k-1}\ \mathbf{d}_k] = [L_{k-1}\ \mathbf{d}_k]$, and we know that $L_{k-1} = U_r\Sigma_rV_r^T$, (40) can be computed via the incremental thin SVD procedure (non-increasing rank) described in Section 4.1.

Another option is to perform the previous computation considering the rank-increasing case if the smallest singular value σ_{r+1} has a significant contribution, which can be estimated by computing the quotient of σ_{r+1} and the sum of all other singular values, in a similar fashion

as for the amFastPCP algorithm (see Section 3.2 and line 3 in Algorithm 1).

Continuing with the solution of (37), the minimizer of (10) or shrinkage step is only applied to the current estimate (since we know S_{k-1}) i.e.

$$\mathbf{s}_k^{(1)} = \text{shrink}(\mathbf{d}_k - \mathbf{l}_k^{(1)})\,,$$

where $\mathbf{l}_k^{(1)}$ is the last column of the current estimate $L_k^{(1)}$.

In the next inner loop ($j = 1$) for solving (37) we have a similar situation, i.e.

$$\begin{aligned} L_k^{(2)} &= \text{partialSVD}(D_k - S_k^{(1)}) \\ &= \text{partialSVD}([D_{k-1} - S_{k-1} \;\; \mathbf{d}_k - \mathbf{s}_k^{(1)}])\,, \end{aligned}$$

which can be effectively computed using the thin SVD replace procedure (non-increasing rank, see Section 4.3), since in the previous step we have computed the partial SVD for $[D_{k-1}\text{-}S_{k-1}\ \mathbf{d}_k]$. Each additional inner loop has a complexity dominated by $O(14 \cdot m \cdot r)$ due to the computation of the current low-rank frame and thin SVD replace procedure (lines 3 and 6 in Algorithm 2). We have determined experimentally that 2 or 3 inner loops (parameter iL in Algorithm 2) suffice to accomplish good results. In our experiments reported in Section 6, we use 2 inner loops for our simulations.

In some particular cases it could be assumed that the background does not change, or changes very slowly, but in practice this condition will usually not hold for long. If rapid changes do occur, we can use the downdate procedure described in Section 4.2 to "forget" the background frames that are "too old" and always keep a low-rank estimate of a more or less constant background, resulting in a "sliding window" incremental PCP algorithm for which the background estimate represents the background as observed for the last bL frames. We handle this condition in Algorithm 2, line 7, with procedure dwnSVD$(\cdot)$, which is explained in Section 4.2.

In real application scenarios an abrupt background change could also occur (e.g. camera is moved, sudden illumination change, etc.). This situation can easily be detected since the current background estimate will be significantly different from all previous backgrounds estimates. In Algorithm 2, line 8, we check for this condition and, if necessary, re-initialize the background estimate using the procedure described in Section 5.2.

5.2 An incremental initialization for Algorithm 2

In this section we describe a fully incremental initialization procedure, which replaces the batch computation described in the "Initialization" of Algorithm 2.

Algorithm 2: incPCP: incremental PCP.

Inputs : observed video $D \in \mathbb{R}^{m \times n}$, regularization parameter λ, number of inner loops iL, background frames bL, $m = k_0$.

Initialization: $L + S = D(:, 1 : k_0)$, initial rank r, $[U_r,\, \Sigma_r,\, V_r] = \text{partialSVD}(L,\, r)$

for $k = k_0 + 1 : n$ **do**

1 $++m$

2 $[U_k, \Sigma_k, V_k] = \text{incSVD}(D(:,k), U_{k\text{-}1}, \Sigma_{k\text{-}1}, V_{k\text{-}1})$

  for $j = 1 : iL$ **do**

3   $L(:,k) = U_k(:, 1:r) * \Sigma_k * (V_k(end, :)')$

4   $S(:,k) = \text{shrink}(D(:,k) - L(:,k), \lambda)$

5   **if** $j == iL$ **then** break

6   $[U_k, \Sigma_k, V_k] = \text{repSVD}(D(:,k),\, S(:,k),\, \ldots$ $U_k,\, \Sigma_k,\, V_k)$

7 **if** $m \geq bL$ **then** dwnSVD("1st column", U_k, Σ_k, V_k)

8 **if** $\|L(:,k) - L(:,k-1)\|_2^2 / \|L(:,k-1)\|_2^2 \geq \tau$ **then** $m = k_0$, use procedure in Section 5.2.

Assuming that r, the rank of the low-rank matrix is known (usually $r \in [2,\, 8]$ is adequate for batch PCP algorithms when analyzing real videos acquired with a static camera) then we can proceed as follows

- Compute $[U,\, \Sigma] = \text{thinQR}(D(:,1))$, set $V = I_1$, where thinQR(.) represents the thin QR decomposition.
- Compute $[U, \Sigma, V] = \text{incSVD}(D(:,k), U, \Sigma, V)$ for $k \in [2,\, r]$, the rank increasing thin SVD as described in Section 4.1.
- Compute $[U, \Sigma, V] = \text{incSVD}(D(:,k), U, \Sigma, V)$ for $k \in [r+1,\, k_0]$, the non-increasing rank thin SVD as described in Section 4.1.

to obtain an estimate of the low-rank approximation of the first k_0 frames of the input video. If this initial estimate needs to be improved, we can take a second pass (or several passes) where we use the thin SVD replace operation (Section 4.3). This would be similar to performing the operations listed in lines 3-6 in Algorithm 2 but applied to the first k_0 frames.

For our incPCP algorithm, we have experimentally determined that $r = 1$, $k_0 = 1$ is usually sufficient to produce very good results. Although this initialization gives a very rough background estimate, it is improved at each iteration (via the incremental SVD procedure, line 2 in Algorithm 2) for bL frames, before any downdate operation is performed. At this point the background estimate is equivalent to that that would be computed via a batch PCP algorithm applied to frames $1 : bL$. Once frame $bL + 1$ is processed, and after the

downdate is called, the background estimate is equivalent to that that would be computed via a batch PCP algorithm applied to frames $2 : bL + 1$. This updating process is the key reason why choosing $r = 1$, $k_0 = 1$ for our initialization stage gives good results; we give computational support for this claim in Section 6.2.

5.3 Limitations of the proposed method

Although the PCP method is considered the state-of-the-art for video background modeling, it does have several limitations [7]. The incPCP algorithm addresses two of these limitations, namely the high computational cost and that PCP is inherently a batch method. Other limitations of the original PCP method, including the assumption that the the observed video is noise-free, and sensitivity to camera motion or jitter, are clearly shared by the proposed incPCP method. While not the focus of the present paper, methods exist for addressing some of these remaining limitations. With respect to noise, methods for making PCP robust to Gaussian [51] and impulse noise [36] could be incorporated into the incPCP algorithm. With respect to camera motion, while there are other works [27,18] that deal with this problem, the incPCP algorithm has recently been extended to provide robustness to translational and rotational jitter [37,32].

In addition to these general limitations, we acknowledge that the proposed incremental initialization for Algorithm 2 (see Section 5.2) could fail for some particular video sequences, particularly if large portions of the background exhibit motion, or the moving objects are large and occlude the background for long periods of time. Empirically, however, the incPCP algorithm making use of this initialization has accurately estimated the backgrounds of all test videos to which it has been applied (see Sections 6.2 and 6.3), including video from a highway surveillance camera in which the background (highway and trees) is constantly occluded by moving cars and the branches and leaves of the trees are constantly moving due to wind.

6 Computational Results

We evaluate the computational performance and accuracy of our proposed incremental method ("incPCP algorithm") and compare it with some of the leading state-of-the-art incremental / online PCP algorithms. The computational performance is measured as the time to generate the sparse component from a given video, where we also include the elapsed time for the initialization stage of the considered algorithms. To evaluate the accuracy of the incPCP algorithm we consider two types of measures: (i) the F-measure which makes use of manually segmented ground-truth and (ii) a normalized ℓ_1 norm restoration measurement of the sparse component which makes use of a proxy ground-truth.

The F-measure, which makes use of a binary ground-truth, is defined as

$$F = \frac{2 \cdot P \cdot R}{P + R}, \quad P = \frac{TP}{TP + FN}, \quad R = \frac{TP}{TP + FP} \tag{41}$$

where P and R stands for precision and recall respectively, and TP, FN and FP are the number of true positive, false negative and false positive pixels, respectively.

The F-measure is one of the most widely used metrics for video background modeling assessment, but most of the available test videos (see below) with binary and manually segmented ground-truth are of small size (less than 320×240 pixel). Since one of the key features of our proposed algorithm is its capability for close to real-time processing of large videos (color full-HD videos, i.e. 1920×1080 pixel), and given the lack of ground-truth for such large video sets, we also use a normalized ℓ_1 norm restoration measurement

$$M(S_P) = \frac{\|S_{GT} - S_P\|_1}{N} \tag{42}$$

to assess the accuracy of the incPCP method, where S_{GT} is the ground truth sparse video approximation, S_P is the sparse video approximation to be evaluated, and N is the number of pixels per frame, for which we take the same value for either the grayscale or the color case. For this case we use the sparse video approximation computed via the batch iALM algorithm [23] (with 20 outer loops) as a proxy ground-truth since, as reported in [7, Tables 6 and 7], this result is considered to be reliable.

We use several test videos to show different computational aspects of the proposed algorithm as well as the leading competing methods. In particular the I2R dataset [22], available at [21], is mainly used as a benchmark for F-measure comparisons. This dataset includes 9 videos (see Table 2) whose sizes varies from 120×160 up to 256×320 pixel, most with more than 1500 frames. Although manually segmented ground-truth is provided for all of the videos, it consists of only 20 frames for each case. We also use two videos from the baseline videos of the 2014 CDnet dataset (described in the next paragraph) [44], available at [20], each with more than 1200 frames, where manually segmented ground-truth is provided for more than the 72% of the frames of each video.

We also select six real video sets as test videos for computational performance measurements as well as for accuracy measurements based on the proposed normalized ℓ_1 norm restoration metric (see (42)):

- V160: 160 × 128 pixel, 1546-frame color video sequence, also labeled as "lobby", of 61.8 seconds at 25 fps, from a lobby with highly variable lighting, from the I2R dataset.
- V320-1: 320 × 256 pixel, 1286-frame color video sequence, also labeled as "ShoppingMall", of 51.4 seconds at 25 fps, from a mall's atrium with lots of people walking around, from the I2R dataset.
- V320-2: 320 × 240 pixel, 1700-frame color video sequence from a highway camera with lots of cars passing by, from the 2014 CDnet dataset.
- V640: 640 × 480 pixel, 400-frame color video sequence of 26.66 seconds at 15 fps, from the Lankershim Boulevard traffic surveillance dataset [2, camera3].
- V720: 720 × 576 pixel, 1200-frame color video sequence of a train station with lots of people walking around, from the 2014 CDnet dataset.
- V1920: 1920 × 1088 pixel, 900-frame color video sequence of 36 seconds at 25 fps, from the Neovison2 public space surveillance dataset [1, Tower 3rd video].

We compare our results (see following subsections) with the GRASTA and GOSUS algorithms. ReProCS, PRMF and RPCA-BL are not considered since (i) the practical ReProCS [13] implementation, available at [12], has a batch initialization stage that considers all the available frames, (ii) similarly, the PRMF implementation [42] has a batch normalization step, and there is no obvious way of removing, in which all the available data is used, and (iii) RPCA-BL there is no publicly available implementation for the online variant of RPCA-BL.

We use a GRASTA implementation [16] by the authors of [17], with its default parameters (10% and 10% of the frames for the batch initialization and subspace tracking respectively) unless explicitly stated otherwise, and a GOSUS implementation [47] by the authors of [48], with its default parameters (using the initial 200 frames for the initialization stage) unless explicitly stated otherwise. While the GRASTA and GOSUS algorithms are able to process both grayscale and color video, the GRASTA software implementation [16] can only process grayscale video, and the GOSUS implementation [47] can only process color video, which places some restrictions on our comparisons.

For GRASTA and GOSUS, such initializations are recommended since they improve their accuracy, as verified in Section 6.2. The GRASTA and GOSUS implementations are Matlab based with MEX interfaces, and while the Matlab implementation of incPCP does not use any MEX interface, there are two flavors (i) one that uses standard single-thread Matlab code and (ii) one that takes advantage of (mainly linear algebra) GPU-enabled (CUDA) Matlab functions; we test this variant in a nvidia GTX980M GPU card.

6.1 Computational performance

All simulations presented in this section, related to Algorithm 2, have been carried out using single-threaded Matlab-only code[2] running on an Intel i7-4710HQ quad-core (2.5 GHz, 6MB Cache, 32GB RAM) based laptop.

Since our algorithm behaves in practice as a "sliding window" incremental PCP (we use $bL = 150$ by default, see Algorithm 2), using rank equal to one suffices to give a good background estimate and good tracking properties (see Sections 6.2 and 6.3). Furthermore, in this case, our initialization step is simplified to a QR factorization of a matrix with just one column (see Section 5.2).

On the other hand, as mentioned in Section 3.1, the GRASTA and GOSUS algorithms can use two different strategies for their background initialization: (i) a recommended but computational costly (see Table 1) and batch background initialization procedure or (ii) an initialization that either considers only a few frames from the initial part of the video to model the background (GRASTA) or a random sub-space to represent the background (GOSUS).

For case (i), GRASTA uses a time sub-sampled (10%) version of all the available frames, whereas GOSUS uses the first 200 frames of the video; the times listed in Table 1 (see "Initialization") correspond to these strategies. In both cases such procedures, in fact, hamper the use of either GRASTA or GOSUS as a real-time alternative to our proposed incPCP algorithm. In subsequent tables and figures, depending on the available space, we will label this strategy as "dflt" for either GRASTA or GOSUS, or "sub-sampled" and "sampled" to specifically refer to the GRASTA and GOSUS cases respectively.

For case (ii), GRASTA uses a rank-5 subspace (first 5 frames of the video) for its background initialization, whereas GOSUS uses a random rank-5 subspace for the same purpose[3]. Although these initializations are as fast as the incPCP rank-1 initialization (their computational costs are not reported since they are negligible) they do have a negative impact on the accuracy of the GRASTA and GOSUS algorithms, as shown in Sections 6.2 and 6.3. In subsequent tables and figures, we

[2] The code is publicly available [34].

[3] These are the default values found the in GRASTA [16] and GOSUS [47] implementations

Test	Initialization (sec.)		Average per frame (sec.)					
	GRASTA grayscale	GOSUS color	incPCP grayscale		incPCP color		GRASTA grayscale	GOSUS color
Video	Standard Matlab + MEX	Standard Matlab + MEX	Standard Matlab	GPU-enabled Matlab	Standard Matlab	GPU-enabled Matlab	Standard Matlab + MEX	Standard Matlab + MEX
V160	7.3	3.5	5.2e-3	1.0e-2	6.9e-3	1.2e-2	8.9e-3	5.1e-1
V320-1	23.0	11.0	1.0e-2	1.3e-2	2.0e-2	1.7e-2	3.4e-2	3.0e+0
V640	72.4	39.9	2.9e-2	2.4e-2	5.7e-2	2.7e-2	1.1e-1	2.6e+1
V720	113.7	67.8	4.3e-2	3.2e-2	6.2e-2	3.3e-2	2.5e-1	5.1e+1
V1920	537.5	(*)	1.9e-1	1.2e-1	3.4e-1	2.8e-2	8.1e-1	(*)

Table 1 Elapsed time to initialize the GRASTA [17] and GOSUS [48] algorithms as well as the average processing time per frame for all algorithms on an Intel-i7 (32 GB RAM) based laptop. The incPCP variant, labeled as "GPU enabled", is run in a nvidia GTX980M GPU card. Note that "e±k" = $10^{\pm k}$. (*) Runs out of memory before completing the initialization stage.

will label this strategy as "i.fr." / "initial frames", or "rand" / "random", depending on the available space, for GRASTA and GOSUS respectively.

Finally, we mention that the results listed in Table 1 show that the proposed incPCP algorithm is substantially faster than GRASTA (e.g. 5 times faster for grayscale V1920, resulting in a 4 fps rate) and that GOSUS (e.g. 110 times faster for color V320-1). This speed-up does not take into consideration the time spent by GRASTA and GOSUS in their batch initializations respectively.

6.2 F-measure based accuracy

In this sub-section we first present F-measure based accuracy results for the I2R dataset (see Table 2), to then establish a link between the F-measure and ℓ_1 norm restoration measurement based (42) using the V320-2 and V720 (from the CDnet dataset) since they have a large number of ground-truth frames.

In order to compute the F-measure for incPCP as well as for GRASTA and GOSUS, a threshold is needed to compute the binary foreground mask. For the results presented in Table 2, as well as for other results presented in this sub-section, this threshold has been computed via an automatic unimodal segmentation [38] since the absolute value of the sparse representation has an unimodal histogram. This approach, although simple, adapts its threshold for each sparse representation and ensures that all algorithms are fairly treated. We also mention, however, that there exists several more effective methods to estimate a binary foreground mask based on the sparse representation of the analyzed video (for instance, see [50] for a batch PCP based algorithm and [19] for an incremental PCP based algorithm).

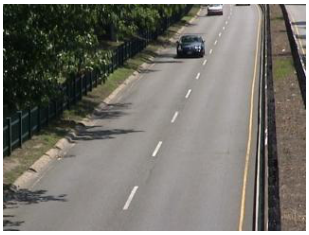
F-measures listed in Table 2, which indicates whether the F-measure is computed for a grayscale or color version of a given video, includes results for the incPCP algorithm and the GRASTA and GOSUS algorithms considering their two strategies for their background initialization, labeled as "dflt" and "rand" (see comments in Section 6.1 on this regard).

Video (20 GT)	F-measure					
	grayscale			color		
	incPCP	GRASTA		incPCP	GOSUS	
		dflt	i.fr.		dflt	rand
Bootstrap (120×160×3057, crowd scene)	0.587	**0.608**	0.607	0.636	**0.659**	0.303
Campus (120×160×1439, waving trees)	**0.244**	0.215	0.210	**0.281**	0.166	0.049
Curtain (120×160×2964, waving curtain)	0.741	**0.787**	0.737	0.758	**0.870**	0.191
Escalator (130×160×3417, moving escalator)	0.481	**0.539**	0.536	**0.472**	0.405	0.260
Fountain (128×160×523, fountain water)	**0.627**	0.662	0.256	0.632	**0.677**	0.066
Hall (144 × 176 × 3548, crowd scene)	0.570	**0.625**	0.624	**0.609**	0.464	0.196
Lobby (128×160×1546, switching light)	0.550	**0.567**	0.491	**0.713**	0.185	0.027
ShoppingMall (256×320×1286, crowd scene)	**0.693**	0.692	0.684	**0.746**	0.715	0.086
WaterSurface (128×160×633, water surface)	0.636	**0.772**	0.743	0.632	**0.787**	0.145

Table 2 Accuracy performance via the F-measure on the I2R dataset [21] for the incPCP, GRASTA and GOSUS algorithms; for the latter two, we include results for their default batch initializations ("dflt") as well as for their alternative fast initializations ("i.fr." or "rand"). Below each the name of each video we include the size and total number of frames (rows × columns × frames) along with a short description. For all cases, the number of available ground-truth (GT) frames is 20. The bold values are the maximum F-measure values (grayscale and color cases are treated independently)

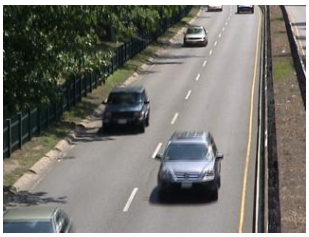
When the "dflt" background initialization for GRASTA and GOSUS are considered, the incPCP, GRASTA and GOSUS algorithms have very similar performance, although it could be argued that GRASTA has a small edge over incPCP in the case of grayscale videos, and that incPCP has a small edge over GOSUS in the case for color videos. However, when the "rand" background initialization for GRASTA and GOSUS are considered, we observe a significant difference: (i) GRASTA's accuracy noticeable drops for some test videos ("Fountain" and "Lobby") whereas for others the accuracy performance, although it is less, remains in the same level[4], and (ii) GOSUS's accuracy dramatically drops in all cases. However, given the small number of ground-truth frames (only 20) provided for the videos listed in Table 2, our view is that care should be taken in drawing conclusions from these results.

(a) V320-2 test video: Frame 470.



(b) V320-2 test video: Frame 600.

(c) V320-2 test video: Frame 700.



(d) V320-2 test video: Frame 900.

Fig. 1 The original color frames 470, 600, 700 and 900 of the V320-2 test video. This is a challenging video since the background is constantly occluded by moving cars and part of the background is moving (trees).

In order to overcome the shortcomings (reduced number of ground-truth frames) of the I2R dataset used for Table 2, we also present the F-measure for the test videos V320-2 and V720, which have 1230 ground-truth frames out of 1700 frames and 900 ground-truth frames out of 1200 frames respectively. The test videos V320-2 and V720 were chosen because their backgrounds are constantly occluded by moving objects: cars in the case of V320-2 and people in the case of V720. Furthermore, in the case of V320-2, the trees, which are part of the background, are waiving (see Figure 1). In the case of V720, where several people walk with different speeds, the highly reflective surface (see Figure 2) that is part of the background, creates a challenging environment.

[4] As will be explained next, when the GRASTA's "rand" variant is used, its background estimate is not stable for several frames (usually about 100, but varying with each case). The ground-truth frames for "Fountain" start at frame 158 and for "Lobby" at 350; also, for the latter case, see Figure 11.

(a) V720 test video: Frame 300.

(b) V720 test video: Frame 540.

(c) V720 test video: Frame 700.

(d) V720 test video: Frame 950.

Fig. 2 The original color frames 300, 540, 700 and 950 of the V720 test video. In this video the moving objects are people walking on a highly reflective surface; moreover the person with the backpack is slowly wandering around. Due to these two facts, the background can be considered as not constant, resulting in a challenging environment.

The F-measure is usually computed as a metric for an entire video sequence[5], however we also consider the F-measure in a frame by frame basis since such metric gives insights about the temporal evolution of the accuracy performance of a given algorithm. We also compare the frame by frame F-measure with ℓ_1-norm restoration measurements (also frame by frame) to establish a baseline for these two types of measurements. The batch iALM algorithm, with 20 outer loops, was used to compute the sparse component of videos V320-2 and V720. These estimates are used as the proxy ground-truth for the ℓ_1-norm restoration measurements.

Since manually segmented binary ground-truth is available from frame 470 onward in the 1700 frame V320-2 test video, we use this subset for our experiments. For the same reason, we analyze the 1200 frame

[5] TP, FN and FP variables in (41) are accumulated for all frames before computing the precision and recall measures that lead to the F-measure.

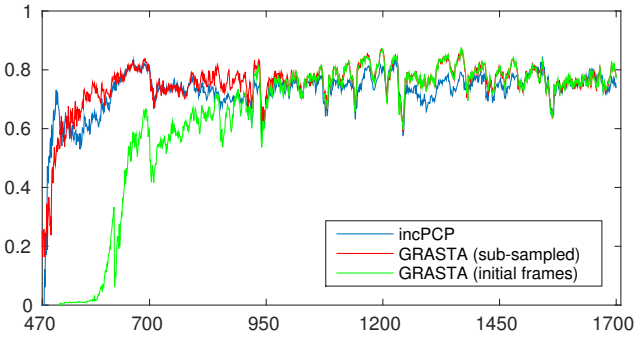


Fig. 3 Frame by frame F-measure for the grayscale version of the V320-2 test video, computed for the incPCP (blue line), and the GRASTA algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GRASTA uses two different background initializations: (i) 10% random sampled from all available frames (red line) and (ii) initial 5 frames (green line). F-measures for the entire video sequence are 0.7455 for the incPC algorithm and 0.7731 and 0.6359 for the GRASTA algorithm with initializations (i) and (ii) respectively.

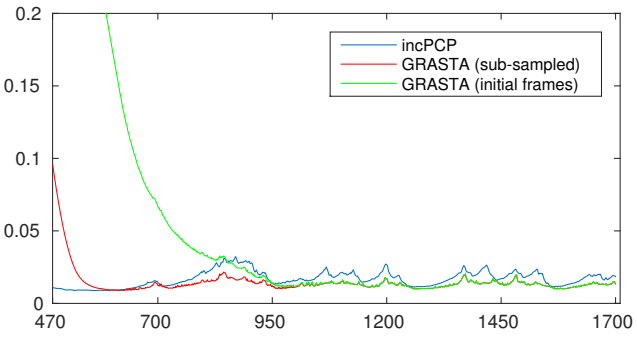


Fig. 4 Frame by frame ℓ_1 norm restoration measurement for the grayscale version of the V320-2 test video, computed for the incPCP (blue line), and the GRASTA algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GRASTA uses two different background initializations: (i) 10% random sampled from all available frames (red line) and (ii) initial 5 frames (green line). There exists a clear (inverse) correspondence with with Fig. 3.

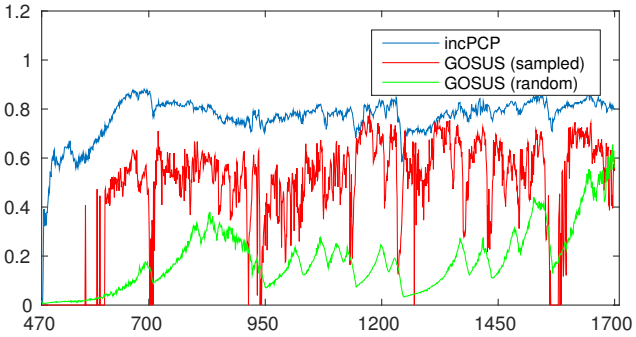


Fig. 5 Frame by frame F-measure for the color version of the V320-2 test video, computed for the incPCP (blue line), and the GOSUS algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GOSUS uses two different background initializations: (i) a reduced sampled of initial frames (red line) and (ii) random sub-space (background) initialization (green line). F-measures for the entire video sequence are 0.7943 for the incPC algorithm and 0.5492 and 0.1760 for the GOSUS algorithm with initializations (i) and (ii) respectively.

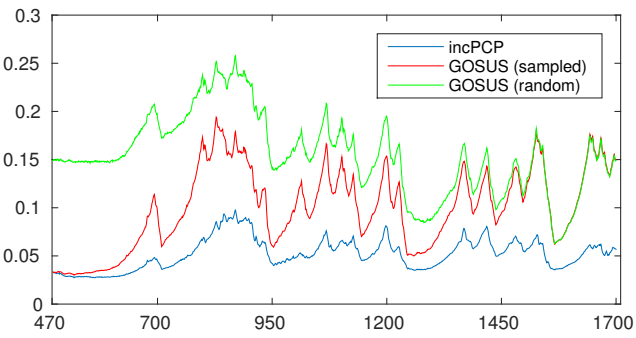


Fig. 6 Frame by frame ℓ_1 norm restoration measurement for the color version of the V320-2 test video, computed for the incPCP (blue line), and the GRASTA algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GOSUS uses two different background initializations: (i) a reduced sampled of initial frames (red line) and (ii) random sub-space (background) initialization (green line). There exists a clear (inverse) correspondence with with Fig. 5.

V720 video from frame 300 onwards. To assess the robustness of their different background initializations, we present results for the two initialization strategies of the GRASTA and GOSUS algorithms in Table 2.

In Figures 3 and 4 we present the F-measure and ℓ_1-norm restoration results for the grayscale version of V320-2, involving the incPCP and GRASTA algorithms. Overall the GRASTA variant that considers a time-subsampled background initialization has a better performance than the other GRASTA variant and a slightly better performance than the incPCP. Although this result is expected, since the partial observation of the video frames should provide a reliable background estimate, the incPCP performs extremely well considering that it uses a rank-1 approximation of the background. Furthermore, incPCP is faster to adapt to the background at the beginning of the video, as shown in Figures 3 and 4. We also note that GRASTA's background initialization that considers only a few frames from the beginning is slow, taking about 500 frames to converge to an acceptable background estimation. If we consider the entire video sequence, the F-measures are 0.7731, 0.6359 and 0.7455 for the GRASTA algorithm (time-subsampled and initial frames initialization variants) and the incPCP algorithm respectively.

In Figures 5 and 6 we present the F-measure and ℓ_1-norm restoration measurements for the color version of V320-2, involving the incPCP and GOSUS algorithms.

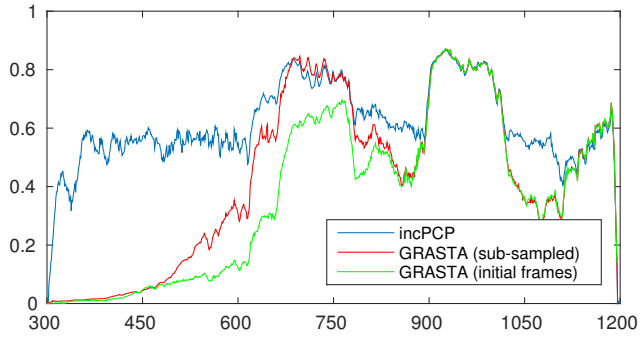


Fig. 7 Frame by frame F-measure for the grayscale version of the V720 test video, computed for the incPCP (blue line), and the GRASTA algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GRASTA uses two different background initializations: (i) time-subsampled all available frames (red line) and (ii) randomly select 100 frames (green line). F-measures for the entire video sequence are 0.6878 for the incPC algorithm and 0.1691 and 0.1585 for the GRASTA algorithm with initializations (i) and (ii) respectively.

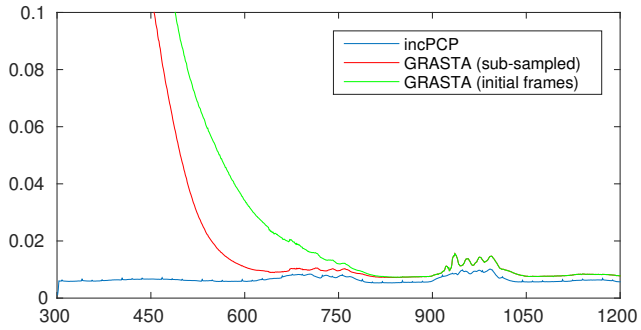


Fig. 8 Frame by frame ℓ_1 norm restoration measurement for the grayscale version of the V720 test video, computed for the incPCP (blue line), and the GRASTA algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GRASTA uses two different background initializations: (i) time-subsampled all available frames (red line) and (ii) randomly select 100 frames (green line). There exists a clear (inverse) correspondence with with Fig. 7.

Overall the incPCP algorithm has a better performance than both GOSUS variants. This result is somewhat unexpected since the batch GOSUS initialization takes into consideration the first 200 frames. However we argue that since (i) cars are constantly occluding the actual background, (ii) the waiving trees perturb the background and (iii) the absence of a any procedure to "forget" the learned background in the past (as for the incPCP algorithm via its downdate operation), GOSUS fails to accurately estimate the background. Moreover, GOSUS random initialization is extremely slow for this (V320-2) challenging video. If we consider the entire video sequence, the F-measure are 0.5492, 0.1760 and 0.7943 for the GOSUS algorithm (sampled and random initialization variants) and the incPCP algorithm respectively.

For the test video V720 we have a similar situation, although the incPCP algorithm has in this case a better performance than both the GOSUS and GRASTA algorithms. In Figures 7 and 8 we present the F-measure and ℓ_1-norm restoration measurements for the grayscale version of V720, involving the incPCP and GRASTA algorithms. In both Figures (7 and 8) we observe that the incPCP algorithm rapidly converges to the right background, whereas both variants of GRASTA are comparatively slow, taking approximately 300 sub-sampled frames or 600 initial frames to reach a reliable background estimate. We argue that (i) the highly reflective surface and (ii) the wandering person in V720 hampers GRASTA's ability to quickly estimate a reliable background. This drawback is also clearly observed for test videos V640 and V1920 (see Section 6.3, Figures 16 and 18) and in a lesser extent for video V160 (see Section 6.3, Figure 11). If we consider the entire video sequence, the F-measures are 0.1691, 0.1585 and 0.6878 for the time-subsampled and initial frames initialization variants of the GRASTA algorithm and the incPCP algorithm respectively.

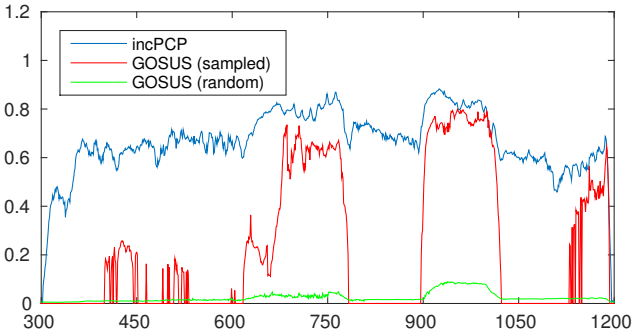


Fig. 9 Frame by frame F-measure for the color version of the V720 test video, computed for the incPCP (blue line), and the GOSUS algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GOSUS uses two different background initializations: (i) a reduced sampled of initial frames (red line) and (ii) random sub-space (background) initialization (green line). F-measures for the entire video sequence are 0.7282 for the incPCP algorithm and 0.4261 and 0.0.0257 for the GOSUS algorithm with initializations (i) and (ii) respectively.

Likewise, in Figures 9 and 10 we present the F-measure and ℓ_1-norm restoration measurements for the color version of V720, involving the incPCP and GOSUS algorithms. The incPCP algorithm has a better performance than any of the GOSUS variants. Moreover, in this case, the performance of the GOSUS random initialization variant is very poor. If we consider

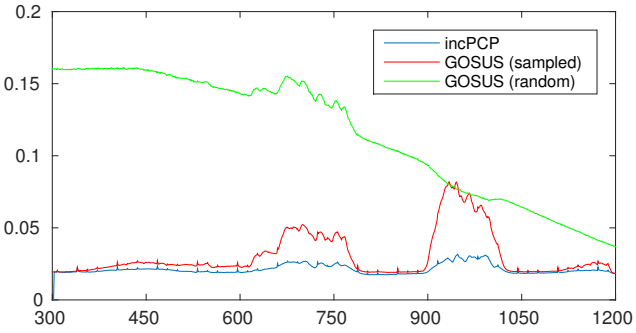


Fig. 10 Frame by frame ℓ_1 norm restoration measurement for the color version of the V720 test video, computed for the incPCP (blue line), and the GOSUS algorithms. incPCP uses a rank-1 initialization to estimate the background, whereas GOSUS uses two different background initializations: (i) a reduced sampled of initial frames (red line) and (ii) random sub-space (background) initialization (green line). There exists a clear (inverse) correspondence with with Fig. 9.

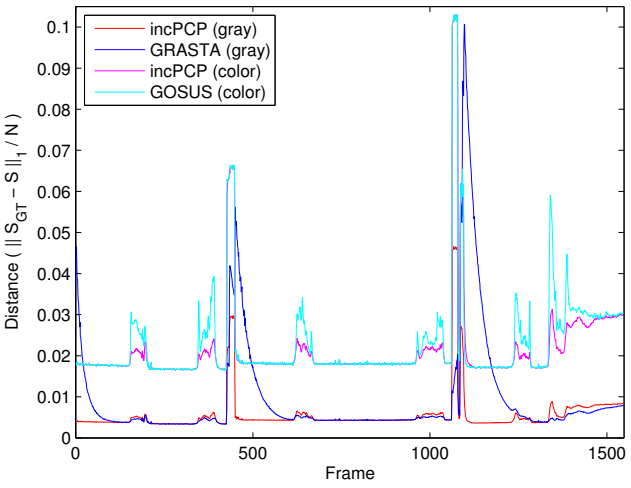


Fig. 11 Frame by frame ℓ_1 norm restoration measure (42) $\frac{\|S_{GT}-S_P\|_1}{N}$ for test video V160 Sparse approximation, where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the sparse approximation computed via the (i) grayscale (blue) GRASTA and (red) the incPCP method, or (ii) color (cyan) GOSUS and (magenta) the incPCP method, and N is the number of pixels per frame (used as a normalization factor; N's value is the same for the grayscale and color cases). For other computational results use [34].

the entire video sequence, the F-measure are 0.4261, 0.0256 and 0.7282 for the sampled and random initialization variants of the GOSUS algorithm and the incPCP algorithm respectively.

6.3 ℓ_1 norm restoration measurement based accuracy

In this section we consider only the batch initialization variants of GRASTA and GOSUS since the results of the previous section indicate that they give better results than their fast counterparts.

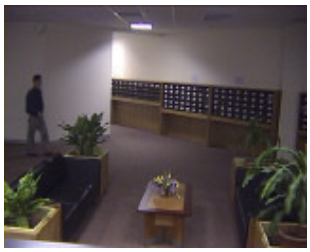
In Fig. 11 we present the ℓ_1 norm restoration measurement based accuracy for video V160, in which the lights go off in frame 430, and then on again in frame 1095, each transition in these sudden illumination changes spanning from 4 to 8 frames. We observed that GRASTA, GOSUS, and our proposed incremental algorithm can adapt to these sudden illumination changes in the background, although GOSUS and ours are faster to adapt with very similar sparse approximation. GRASTA's adaptation rate (see also Figs. 16 and 18) could be improved if more information is used to track the low-rank subspace, at the cost of increased computation time [15].

In Fig. 12 we present the sparse approximation of frame 628 of the V160 test video for iALM, GRASTA and incPCP algorithms. We show grayscale images due to limitations of the GRASTA code, which, unlike our implementation, cannot handle color video; the sparse approximation of GOSUS for this example is similar to that obtained via the incPCP algorithm.

Likewise, in Figures 13, 15 and 17 we show the sparse approximation (iALM, GRASTA, GOSUS and proposed algorithm) of selected frames for the V320-1, V640 and V1920 test videos respectively, along with the original frames, while in Figures 14, 16 and 18 we present the ℓ_1 norm restoration measure for the V320-1, V640 and V1920 test videos. Common features of all of these videos are the large number of objects (walking people or cyclists or cars) constantly going in or out of the field of view, and the absence of any apparent change due to illumination in the background. Furthermore, the moving objects in V320-1 and V640 fill up a large percentage ($> 50\%$) of the observed field of view.

Although, from a visual perception point of view, the sparse approximation computed by our proposed algorithm (Figures 13(d), 13(h), 15(d), 17(d) and 17(h)) are similar to those computed by the iALM algorithms, they are not of the same quality as shown by Figs. 14, 16 and 18, in which there exist a small but clear quantitative differences in reconstruction quality.

From a quantitative point of view the grayscale sparse approximation computed via the GRASTA algorithm, shown in Figures 13(c), 13(g), 17(c) and 17(g), has a slightly better ℓ_1 norm restoration measure than that of the incPCP, except for the V1920 case We argue that this is because in videos V320-1 and V640 several moving objects stop for noticeable periods of time. Due to the background tracking properties of our proposed algorithm, it tries to learn the "new" background, al-



(a) V160 test video: Frame 628.

(b) GRASTA sparse estimation

(c) iALM sparse estimation

(d) incPCP Sparse estimation

Fig. 12 The original color frame 628 of the V160 test video (a) as well as the grayscale sparse approximation computed via (c) the (batch) iALM (20 outer loops), (b) GRASTA and (d) the proposed incremental PCP method. For other computational results see [34].

though this represents a relative small change in the background, resulting in slightly worse quantitative reconstruction quality; these artifacts can be dealt with by using a larger number of frames to estimate the background: in Algorithm 2, use a larger value for bL (experimentally, we have increased the bL value up to 300 without any loss of performance for these videos). The above mentioned artifacts – due to moving objects stopping for some time – do not occur in the V1920 case, and thus the better quantitative reconstruction quality of the incPCP algorithm.

In our experiments with color videos, the GOSUS ℓ_1 norm restoration measure is clearly worse than that of the incPCP, as can be seen in Figures 11, 14 and 15b. We hypothesize that although GOSUS and incPCP tracking properties are similar for strong background changes (e.g. V160), when the change is due to the entrance of several slow moving objects (e.g. V320-1), incPCP is faster to discern those objects as foreground.

7 Conclusions

We have presented a novel incremental PCP algorithm that fundamentally differs from current state-of-the-art PCP algorithms: (i) the proposed algorithm can process an input video in a fully incremental fashion, that is, one frame at a time, whereas existing state-of-the-art PCP algorithms are batch methods or have batch initialization; (ii) it has a trivial initialization stage, usually needing only a handful of frames to converge to the right background, whereas other existing online algorithms do have a batch initialization or need a larger number of frames to converge to the right background (as shown in Sections 6.2 and 6.3); (iii) its computational cost is $O(14 \cdot m \cdot r \cdot p)$ (assuming $r \ll m, n$) per frame, where p (we use $p = 2$ in our experiments) is the number of inner loops ; (iv) its memory footprint is $O(3 \cdot m) + O(m \cdot r)$.

The proposed method can process full HD color videos at a rate of 0.34 and 0.28 seconds per frame, using a Matlab-only single-threaded and a GPU-Matlab enabled implementation respectively. Like other online algorithms, our proposed algorithm can adapt to changes in the background since the core operations of the incPCP algorithm are rank-1 modifications (update, replace or downdate). Since the incPCP algorithm has the ability to forget a previously learn background, it behaves in practice as a "sliding window" algorithm, which is a very useful property, particularly when the background changes slowly.

Ongoing work focuses on two aspects: (i) a parallel (CUDA) implementation (initial results in [33]), that we expect will make it possible to analyze full HD videos in real-time, and (ii) on an extension to the incPCP algorithm that is robust to translational and rotational jitter, with initial results reported in [37,32].

References

1. USC Neovision2 Project. DARPA Neovision2, data available from `http://ilab.usc.edu/neo2/`
2. Lankershim boulevard dataset (2007). U.S. Department of Transportation Publication FHWA-HRT-07-029, data available from `http://ngsim-community.org/`
3. Aybat, N., Goldfarb, G., Iyengar, G.: Fast first-order methods for stable principal component pursuit (2011). URL `http://arxiv.org/abs/1105.2126`. ArXiv:1105.2126
4. Baker, C., Gallivan, K., Dooren, P.V.: Low-rank incremental methods for computing dominant singular subspaces. Linear Algebra and its Applications **436**(8), 2866 – 2888 (2012)
5. Bezdek, J., Hathaway, R.: Some notes on alternating optimization. In: Advances in Soft Computing – AFSS 2002, *Lecture Notes in Computer Science*, vol. 2275, pp. 288–300. Springer Berlin Heidelberg (2002)
6. Bouwmans, T., Baf, F., Vachon, B.: Handbook of Pattern Recognition and Computer Vision, chap. Statistical Background Modeling for Foreground Detection: A Survey, pp. 181–199. World Sci. Pub. (2010)
7. Bouwmans, T., Zahzah, E.: Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance. Computer Vision and Image Understanding **122**, 22–34 (2014)

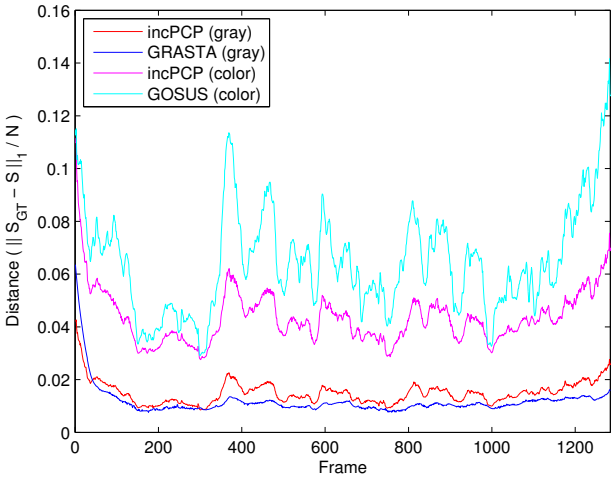


Fig. 14 Frame by frame ℓ_1 norm restoration measure (42) $\frac{\|S_{GT}-S_P\|_1}{N}$ for test video V320-1 Sparse approximation, where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the sparse approximation computed via the (i) grayscale (blue) GRASTA and (red) the proposed incremental PCP method, or (ii) color (cyan) GOSUS and (magenta) the proposed incremental PCP method, and N is the number of pixels per frame (used as a normalization factor).

(a) V320-1 test video: Frame 371.

(b) GOSUS sparse estimation

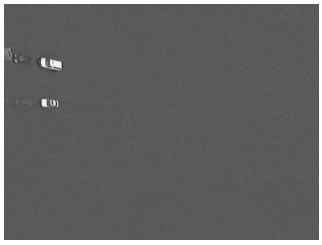
(c) iALM sparse estimation

(d) incPCP Sparse estimation

Fig. 15 The original color frame 371 of the V320-1 test video (a) as well as the sparse approximation computed via (c) the (batch) iALM (20 outer loops), (b) GOSUS and (d) the proposed incremental PCP method. For other computational results use [34].

8. Brand, M.: Fast low-rank modifications of the thin singular value decomposition. Linear Algebra and its Applications **415**(1), 20 – 30 (2006)
9. Bunch, J., Nielsen, C.: Updating the singular value decomposition. Numerische Mathematik **31**(2), 111–129 (1978). DOI: 10.1007/BF01397471. URL `http://dx.doi.org/10.1007/BF01397471`
10. Candès, E., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of the ACM **58**(3) (2011)
11. Chahlaoui, Y., Gallivan, K., Van Dooren, P.: Computational information retrieval. In: In Computational Information Retrieval, chap. An Incremental Method for Computing Dominant Singular Spaces, pp. 53–62. SIAM

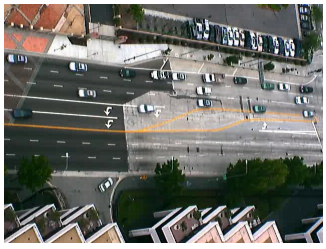
(a) V640 test video: frame 155.



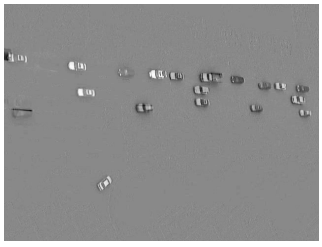
(b) iALM sparse estimation.

(c) GRASTA sparse estimation.

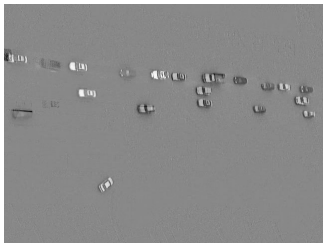
(d) incPCP sparse estimation.



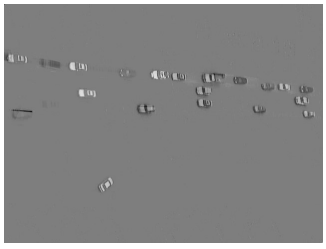
(e) V640 test video: frame 324.



(f) iALM sparse estimation.



(g) GRASTA sparse estimation.



(h) incPCP sparse estimation.

Fig. 13 The original color frames 155 (a) and 324 (e) of the "640" test video, along with the corresponding (grayscale) sparse approximations computes via the iALM (b, f), GRASTA (c, g) and proposed algorithm (d, h) respectively.

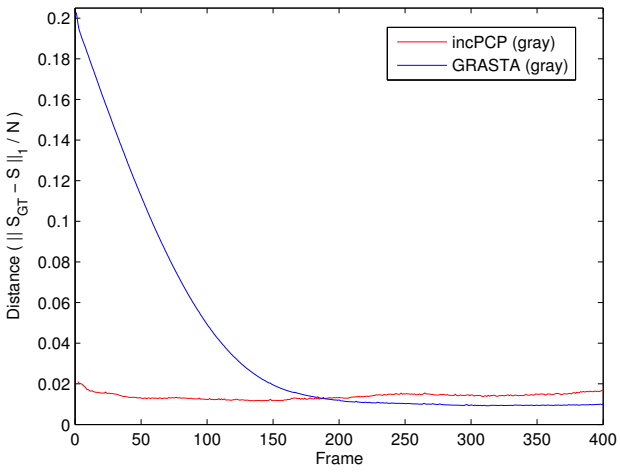
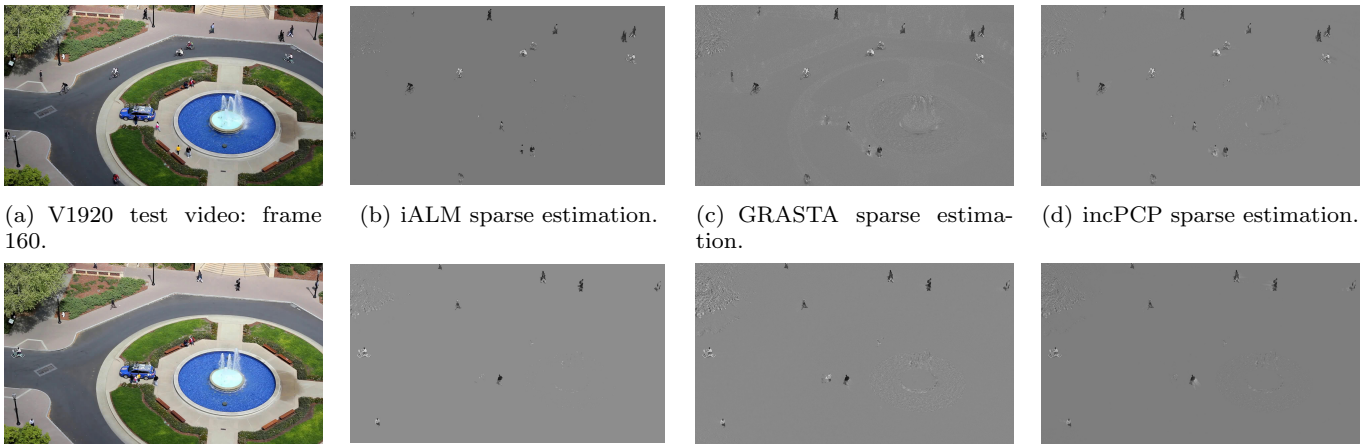


Fig. 16 Sparse approximation (V640) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the sparse approximation computed via the (blue) GRASTA and (red) the proposed incremental PCP method, and N is the number of pixels per frame (used as a normalization factor).

(2001)
12. Guo, H.: Practical ReProCS code. `http://www.ece.iastate.edu/~hanguo/PracReProCS.html`
13. Guo, H., Qiu, C., Vaswani, N.: An online algorithm for separating sparse and low-dimensional signal sequences from their sum. Signal Processing, IEEE Transactions on **62**(16), 4284–4297 (2014)
14. Guyon, C., Bouwmans, T., Zahzah, E.: Robust principal component analysis for background subtraction: Systematic evaluation and comparative analysis. In: Principal Component Analysis. ch. 12. InTech (2012). DOI: 10.5772/38267
15. He, J., Balzano, L., Lui, J.: Online robust subspace tracking from partial information. CoRR **abs/1109.3827** (2011). Submitted
16. He, J., Balzano, L., Szlam, A.: GRASTA code. `https://sites.google.com/site/hejunzz/grasta`
17. He, J., Balzano, L., Szlam, A.: Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1568–1575 (2012). DOI: 10.1109/CVPR.2012.6247848
18. He, J., Zhang, D., Balzano, L., Tao, T.: Iterative Grassmannian optimization for robust image alignment. Image and Vision Computing **32**(10), 800 – 813 (2014)
19. Hu, Y., Sirlantzis, K., Howells, G., Ragot, N., Rodriguez, P.: An online background subtraction algorithm using a contiguously weighted linear regression model. In: 23rd European Signal Processing Conference (EUSIPCO). Nice, France (2015)
20. Jodoin, P., Konrad, J.: Cdnet 2014 dataset. `http://wordpress-jodoin.dmi.usherb.ca/dataset2014/`
21. Li, L., Huang, W., Gu, I.Y.H., Tian, Q.: Background model test data. `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`
22. Li, L., Huang, W., Gu, I.Y.H., Tian, Q.: Foreground object detection from videos containing complex background. In: Proceedings of the Eleventh ACM International Conference on Multimedia, MULTIMEDIA '03, pp. 2–10. ACM, New York, NY, USA (2003). DOI: 10.1145/957013.957017. URL `http://doi.acm.org/10.1145/957013.957017`
23. Lin, Z., Chen, M., Ma, Y.: The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices (2011). URL `http://arxiv.org/abs/1009.5055v2`. ArXiv:1009.5055v2
24. Liu, G., Lin, Z., Yu, Y.: Robust subspace segmentation by low-rank representation. In: Intl. Conf. Mach. Learn. (ICML), pp. 663–670 (2010)
25. Mateos, G., Giannakis, G.: Robust pca as bilinear decomposition with outlier-sparsity regularization. Signal Processing, IEEE Transactions on **60**(10), 5176–5190 (2012)
26. Min, K., Zhang, Z., Wright, J., Ma, Y.: Decomposing background topics from keywords by principal component pursuit. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM), pp. 269–278. ACM, New York, NY, USA (2010)
27. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. IEEE Trans. Pattern Analysis & Machine Intelligence **34**(11), 2233 – 2246 (2012)
28. Pope, G., Baumann, M., Studer, C., Durisi, G.: Real-time principal component pursuit. In: 45th Asilomar Conference on Signals, Systems and Computers, pp. 1433 –1437 (2011). DOI: 10.1109/ACSSC.2011.6190254
29. Qiu, C., Vaswani, N.: Real-time robust principal components' pursuit. CoRR **abs/1010.0608** (2010). URL `http://arxiv.org/abs/1010.0608`
30. Qiu, C., Vaswani, N.: Reprocs: A missing link between recursive robust PCA and recursive sparse recovery in large but correlated noise. CoRR **abs/1106.3286** (2011)
31. Qiu, C., Vaswani, N.: Support predicted modified-cs for recursive robust principal components pursuit. In: IEEE International Symposium on Information Theory (ISIT), pp. 668–672 (2011)
32. Rodríguez, P.: Jitter invariant incremental principal component pursuit for video background modeling on the tk1 (2015). Accepted, IEEE Asilomar Conference on Signal, Systems and Computers (ACSSC)
33. Rodriguez, P.: Real-time incremental principal component pursuit for video background modeling on the TK1 (2015). GPU Technical Conference
34. Rodríguez, P., Wohlberg, B.: incremental PCP simulations. `http://sites.google.com/a/istec.net/prodrig/Home/en/pubs/incpcp`
35. Rodríguez, P., Wohlberg, B.: Fast principal component pursuit via alternating minimization. In: Proc. of IEEE Int'l. Conf. on Image Proc. (ICIP), pp. 69–73. Melbourne, Australia (2013)
36. Rodríguez, P., Wohlberg, B.: Video background modeling under impulse noise. In: Proc. of IEEE Int'l. Conf. on Image Proc. (ICIP). Paris, France (2014)
37. Rodríguez, P., Wohlberg, B.: Translational and rotational jitter invariant incremental principal component pursuit for video background modeling (2015). Accepted, IEEE International Conference on Image Processing (ICIP)
38. Rosin, P.: Unimodal thresholding. Pattern Recognition **34**, 2083–2096 (2001)
39. Seidel, F., Hage, C., Kleinsteuber, M.: pROST: a smoothed lp-norm robust online subspace tracking method for background subtraction in video. Machine Vision and Applications **25**(5), 1227–1240 (2014)
40. Shah, M., Deng, J., Woodford, B.: Video background modeling: recent approaches, issues and our proposed techniques. Machine Vision and Applications pp. 1–15



(a) V1920 test video: frame 160.
(b) iALM sparse estimation.
(c) GRASTA sparse estimation.
(d) incPCP sparse estimation.
(e) V1920 test video: frame 480.
(f) iALM sparse estimation.
(g) GRASTA sparse estimation.
(h) incPCP sparse estimation.

Fig. 17 The original color frames 160 (a) and 480 (e) of the V1920 test video, along with the corresponding (grayscale) sparse approximations computed via the iALM (b, f), GRASTA (c, g) and proposed algorithm (d, h) respectively.

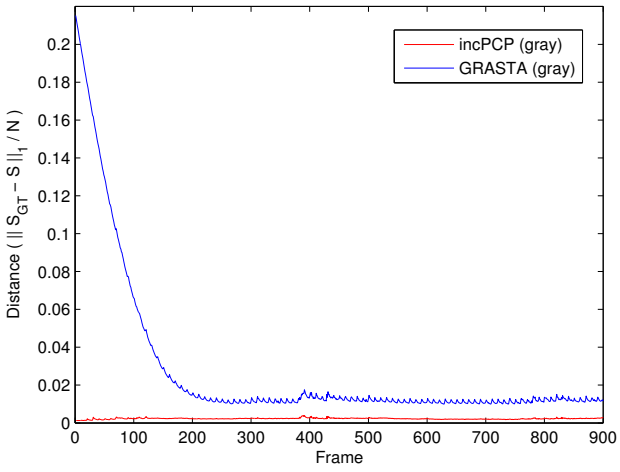


Fig. 18 Sparse approximation (V1920) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the sparse approximation computed via the (blue) GRASTA and (red) the proposed incremental PCP method, and N is the number of pixels per frame (used as a normalization factor).

(2013). DOI: 10.1007/s00138-013-0552-7. URL http://dx.doi.org/10.1007/s00138-013-0552-7
41. Srebro, N., Rennie, J., Jaakola, T.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems 17, pp. 1329–1336. MIT Press (2005)
42. Wang, N.: PRMF code. http://winsty.net/prmf.html
43. Wang, N., Yao, T., Wang, J., Yeung, D.Y.: A probabilistic approach to robust matrix factorization. In: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid (eds.) Computer Vision – ECCV 2012, *Lecture Notes in Computer Science*, vol. 7578, pp. 126–139. Springer Berlin Heidelberg (2012). URL http://dx.doi.org/10.1007/978-3-642-33786-4_10
44. Wang, Y., Jodoin, P., Porikli, F., Konrad, J., Benezeth, Y., Ishwar, P.: Cdnet 2014: An expanded change detection benchmark dataset. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 393–400 (2014). DOI: 10.1109/CVPRW.2014.126
45. Wright, J., Ganesh, A., Rao, S., Peng, Y., Ma, Y.: Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In: Adv. in Neural Inf. Proc. Sys. (NIPS) 22, pp. 2080–2088 (2009)
46. Xiong, L., Chen, X., Schneider, J.: Direct robust matrix factorization for anomaly detection. In: IEEE Int. Conf. on Data Mining, pp. 844–853 (2011)
47. Xu, J., Ithapu, V.K., Mukherjee, L., Rehg, J.M., Singh, V.: GOSUS code. http://pages.cs.wisc.edu/~jiaxu/projects/gosus/
48. Xu, J., Ithapu, V.K., Mukherjee, L., Rehg, J.M., Singh, V.: GOSUS: Grassmannian online subspace updates with structured-sparsity. In: IEEE Int, Conf. on Computer Vision (ICCV), pp. 3376–3383 (2013)
49. Zhou, T., Tao, D.: GoDec: Randomized low-rank & sparse matrix decomposition in noisy case. In: ACM Int. Conf. on Machine Learning, ICML '11, pp. 33–40 (2011)
50. Zhou, X., Yang, C., Yu, W.: Moving object detection by detecting contiguous outliers in the low-rank representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(3), 597–610 (2013). DOI: 10.1109/TPAMI.2012.132
51. Zhou, Z., Li, X., Wright, J., Candes, E., Ma, Y.: Stable principal component pursuit. In: Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on, pp. 1518–1522 (2010)