Chapter 1

Incremental Methods for Robust Local Subspace Estimation

Paul Rodriguez (PUCP), Brendt Wohlberg (LANL)

1.1 Introduction

As discussed in detail in the initial chapters of this book, the Robust PCA (RPCA) problem

$$\arg\min_{L,S} \|L\|_* + \lambda\|S\|_1 \;\; \text{s.t. } D = L + S \;, \tag{1.1}$$

which decomposes a matrix D into a low-rank, L, and sparse component, S, has been shown to give very good performance for video background modeling, in which context the stationary background is represented by the low-rank component, and the moving foreground is represented by the sparse component. This chapter introduces two distinct types of enhancements to the standard RPCA problem, (i) the development of more computationally efficient algorithms for solving this problem (or a variant thereof), including an incremental algorithm that is able to process a single video one frame at a time, and (ii) modifying the problem form to make it invariant to transformations such as translation and rotation so that it can be applied to video captured by a non-stationary camera. These two enhancements are combined in the final section of the chapter.

1.2 Alternating Minimization Algorithm for Robust PCA

It is useful to recall that the RPCA problem [1, Eq. 1.1] is derived as the convex relaxation of the original problem [2, Section 2]

$$\arg\min_{L,S} \ \ \text{rank}(L) + \lambda\|S\|_0 \ \ \text{s.t. } D = L + S \ , \tag{1.2}$$

based on decomposing matrix D such that $D = L + S$, with low-rank L and sparse S. While most RPCA algorithms, including the Augmented Lagrange Multiplier (ALM) and inexact ALM (iALM) algorithms [3, 4] are directly based on (1.1), this is not the only possible tractable problem that can be derived from (1.2). In particular, changing the constraint $D = L + S$ to a penalty and the rank penalty to an inequality constraint leads to the problem

$$\arg\min_{L,S} \frac{1}{2}\|L + S - D\|_F^2 + \lambda\|S\|_1 \quad \text{s.t. } \text{rank}(L) \leq r \ . \tag{1.3}$$

A computationally efficient solution can be found via an alternating optimization (AO) [5] procedure, since it seems natural to split (1.3) into a low-rank approximation followed by a shrinkage.

In what follows, it will be shown that an AO method applied to (1.3), i.e. (i) $\arg\min_L \frac{1}{2}\|L+S_0-D\|_F^2$ s.t. $\text{rank}(L) \leq r$, (ii) $\arg\min_S \frac{1}{2}\|L+S-D\|_F^2+\lambda\|S\|_1$, not only converges to the global minimum of (1.3) but it is computationally efficient and also provides a smooth transition toward an incremental solution of (1.3). We will also include a brief description of other two works that use very closely related feasible reformulations of (1.2), and use an AO method to compute its solution.

1.2.1 Related Work

To the best of our knowledge, there are three algorithms that use an AO method to solve feasible reformulation of the direct Lagrangian reformulation of the RPCA (1.2): (i) the "Fast Alternating Minimization PCP" (amFastPCP) algorithm [6], to be described in Section 1.2.2; (ii) the "Go Decomposition" (GoDec) algorithm [7]; and (iii) the "Direct Robust Matrix Factorization" (DRMF) algorithm [8].

GoDec algorithm

The main motivation behind the GoDec algorithm [7] is to estimate the low-rank (L) and sparse (S) approximations of matrix D, s.t. $D = L+S+E$, where E is noise. The original problem GoDec aims to solve is

$$\arg\min_{L,S} \frac{1}{2}\|L + S - D\|_F^2 \quad \text{s.t. } \text{rank}(L) \leq r, \ \|S\|_0 \leq c \ , \tag{1.4}$$

noting that the authors of [7] preferred the notation card$(\cdot)$ or cardinality, instead of the ℓ_0 norm of S. This problem can be naively solved via the AO

$$L^{(j+1)} = \arg\min_{L} \|L + S^{(j)} - D\|_F^2 \quad \text{s.t. } \operatorname{rank}(L) \leq r \tag{1.5}$$

$$S^{(j+1)} = \arg\min_{S} \|L^{(j+1)} + S - D\|_F^2 \quad \text{s.t. } \|S\|_0 \leq c\,. \tag{1.6}$$

Sub-problem (1.5) is the well-known low-rank approximation problem. Assuming that $\text{SVD}(D - S^{(j)}) = U\Lambda V^T$, then it can be proved that $L^{(j+1)} = U\Lambda_r V^T$, where $\Lambda_r = \{\lambda_1, \lambda_2, \ldots, \lambda_r, 0, \ldots, 0\}$. Sub-problem (1.6) is a cardinality constrained problem, which also has a closed form solution given by entry-wise hard-thresholding the matrix $D - L^{(j+1)}$. In order to reduce the computational cost of computing the full SVD of $D - S^{(j)}$ in the solution of (1.5), the GoDec algorithm makes use of bilateral random projections (BRP) along with a modified power method to solve it. This scheme needs to invert an auxiliary $r \times r$ matrix, followed by its full SVD computation. Overall in [7] it is reported that the GoDec algorithm needs $O(r^2(m + 3n + 4r) + (4q + 4)mnr)$ flops per loop, where q is a constant greater or equal to zero.

Direct Robust Matrix Factorization algorithm

The Direct Robust Matrix Factorization (DRMF) [8] algorithm also focuses on solving (1.4). As for the GoDec algorithm, DRMF also proposed to solve (1.4) via (1.5)-(1.6). However DRMF uses partial SVD, making use of the PROPACK package [9], sub-problem (1.6) being solved in the same way as for the GoDec. In [8] it is reported that the computational cost per iteration is given by $O(mn(r + \log(c)))$ flops.

1.2.2 Fast alternating minimization PCP (amFastPCP) Algorithm

While the GoDec and DRMF algorithms focus on solving the optimization problem described by (1.4), the amFastPCP algorithm [6] aims to solve (1.3), which as mentioned before, can be easily solved via the AO

$$L^{(j+1)} = \arg\min_{L} \quad \|L + S^{(j)} - D\|_F^2 \quad \text{s.t. } \operatorname{rank}(L) \leq r \tag{1.7}$$

$$S^{(j+1)} = \arg\min_{S} \quad \|L^{(j+1)} + S - D\|_F^2 + \lambda\|S\|_1, \tag{1.8}$$

which is summarized in Algorithm 1. It is worth noting that (1.7)-(1.8) converge to the global minimum of (1.3) since (i) (1.7) and (1.8) have each a unique global minimizer, and (ii) at each iteration of (1.7)-(1.8) the cost functional of (1.3) is reduced:

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

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

where $L^{(j+1)}$ is the solution to (1.7); 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 (1.8); then, in particular

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

then $f(L^{(j)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j)}) \geq f(L^{(j+1)}, S^{(j+1)})$. This proves that the AO method based on (1.7)-(1.8) converges to the global minimum of (1.3), since it complies with the necessary conditions for convergence of any given AO method (see [5, Section 6] for more details).

Clearly Algorithm 1 is a batch method due to the nature of sub-problem (1.7), which is a low-rank approximation that can be solved by computing the partial SVD of $D - S^{(j)}$ with r components. In Algorithm 1 this is related to 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 joint memory requirement for these operations is $O(2 \cdot m \cdot n)$. The solution to (1.8) is simple element-wise shrinkage (soft thresholding): $\text{shrink}(D - L^{(j+1)}, \lambda)$, where

$$\text{shrink}(x, \epsilon) = \text{sign}(x) \max\{0, |x| - \epsilon\} \ . \tag{1.9}$$

The solution obtained via the iterative solution of (1.7)-(1.8) is of comparable quality to the solution of the original PCP problem (see [6] for details), being able to deliver a useful estimate of the sparse component even after a single outer loop, while being approximately an order of magnitude faster than the inexact ALM [4] algorithm to construct a sparse component of similar quality.

Algorithm 1: Fast alternating minimization PCP [6]

Inputs : Observed video $D \in \mathbb{R}^{m \times n}$, regularization parameter λ, eigen-value 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 $S^{(j+1)} = \text{shrink}(D - L^{(j+1)}, \lambda)$

1.3 Incremental Algorithm for Robust PCA

1.3.1 Related Work

To the best of our knowledge problem, (i) Recursive Projected Compressive Sensing (ReProCS) [10], (ii) Grassmannian Robust Adaptive Subspace Tracking Algorithm GRASTA [11], (iii) ℓ_p-norm Robust Online Subspace Tracking pROST [12], and (iv) Grassmannian Online Subspace Updates with Structured-sparsity (GOSUS) [13] are the only RPCA-like methods that are considered to be incremental. All of these methods are only partially incremental in that they have a batch initialization, i.e. they need a initial background subspace estimate (represented by U in (1.10)), which is typically accomplished by analyzing a temporally sub-sampled version of the original dataset via a batch procedure.

Although there are some similarities in the formulation of the ReProCS and the GRASTA methods (see below), they are fundamentally different algorithms: ReProCS uses a known model for the moving objects in order to update the background subspace estimation, whereas GRASTA forces the background subspace to be a Grassmannian manifold. At a high level, both pROST and GOSUS are very similar to GRASTA, however the former uses a different norm than GRASTA in order to improve its tracking capabilities and the latter imposes an additional group sparsity constraint on the moving objects.

Let $\mathbf{d}_k$, $\mathbf{l}_k$ and $\mathbf{s}_k$ denote the k^{th} column of matrices D, L and S (see (1.1)). Starting from $\mathbf{d}_k = \mathbf{l}_k + \mathbf{s}_k$, ReProCS and GRASTA independently proposed (for details see [10, equation after eq. (1)], and [11, eq. (1)])

$$\mathbf{l}_k = U\mathbf{x}_k, \qquad \mathbf{d}_k = U\mathbf{x}_k + \mathbf{s}_k \tag{1.10}$$

where noise can be added to (1.10) depending on the method. In both methods U is a $m \times m$ unknown orthonormal matrix, $\mathbf{x}_k$ is a vector of weights, and letting $\mathcal{N}_k$ denote the support of $\mathbf{x}_k$, then $P_k = (U)_{\mathcal{N}_k}$ span the low rank subspace in which the current set of $\mathbf{l}_k$ lies. Assuming P_k is known (or estimated), and $P_{k,\perp}$ is its orthonormal compliment, then ReProCS solves [14, eq. (6)]

$$\min \|\mathbf{s}_k\|_1 \quad \text{s.t. } 0.5\|P_{k,\perp}\mathbf{s}_k + P_{k,\perp}^T\mathbf{d}_k\|_2^2 \leq \epsilon \,, \tag{1.11}$$

whereas GRASTA solves [11, eq. (4)]

$$\min \|\mathbf{s}_k\|_1 \quad \text{s.t. } \mathbf{d}_k = U\mathbf{x}_k + \mathbf{s}_k \,. \tag{1.12}$$

ReProCS interprets (1.11) as a compressive sensing problem, whereas GRASTA solves (1.12) via the ADMM method [15]. Due to additional constraints on (1.10), which include a known model for the trajectories of the moving objects, ReProCS becomes a non-real-time algorithm since it uses a batch method in its SVD-based initialization step. Moreover, it cannot process real videos where multiple moving objects enter and leave the field of view of the camera.

On the other hand, GRASTA [11] uses a reduced number of frames, $q \ll n$, compared to the RPCA problem (1.1), to estimate an initial low rank sub-space

representation of the background U and then processes each frame (which can be spatially sub-sampled) at a time. This initialization step is a batch procedure and can have a relatively high complexity. Once initialized, GRASTA can estimate and track non-stationary backgrounds.

pROST [12] is very similar to the GRASTA algorithm, but it uses a ℓ_p weighted version of (1.12) to track the low rank sub-space representation of the background. Experimental results in [12] show that pROST can outperform GRASTA in the case of dynamic backgrounds.

Similarly, GOSUS [13] is also closely related to GRASTA, however GOSUS enforces structured/group sparsity on the sparse component and 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. Although GOSUS is known to have better tracking properties than GRASTA, its computational cost is higher. Furthermore computational results (presented in Section 1.3.3) suggest that its complexity does not depend linearly with the number of pixel in the analyzed video frame, but it is influenced by the number of moving objects.

1.3.2 Incremental and rank-1 modifications for thin SVD

Given a 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 column vectors $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^l$, note that

$$D + \mathbf{a}\mathbf{b}^T = [U_0 \ \mathbf{a}] \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} [V_0 \ \mathbf{b}]^T \ , \tag{1.13}$$

where $\mathbf{0}$ is a zero column vector of the appropriate size. Based on [16, 17], as well as on [18] we briefly describe an incremental thin SVD and rank-1 modifications (update, downdate and replace) for thin SVD. Before proceeding with the description of the rank-1 modifications procedures, note that

- the computational complexity of any of the procedures described below (see [16, Section 3], [18, Section 4]) is upper bounded by $O(10 \cdot m \cdot r) + O(r^3) + O(3 \cdot r \cdot l)$. If $r \ll m, l$ and $l \ll m$ hold, then the complexity is dominated by $O(10 \cdot m \cdot r)$;
- matrices U_0 and V_0 are orthonormal, i.e. $U^T U = I_m$ and $V^T V = I_l$, where I_k is the identity matrix of size $k \times k$, as well as that the diagonal elements Σ_0, are non-negative and non-increasing;
- we assume that the Gram-Schmidt orthonormalization of $\mathbf{a}$ and $\mathbf{b}$ w.r.t. U_0 and V_0 have been computed:

$$\mathbf{x} = U_0^T \mathbf{a}, \quad \mathbf{z}_x = \mathbf{a} - U\mathbf{x}, \quad \rho_x = \|\mathbf{z}_x\|_2, \quad \mathbf{p} = \frac{1}{\rho_x}\mathbf{z}_x \tag{1.14}$$

$$\mathbf{y} = V_0^T \mathbf{b}, \quad \mathbf{z}_y = \mathbf{b} - V\mathbf{y}, \quad \rho_y = \|\mathbf{z}_y\|_2, \quad \mathbf{q} = \frac{1}{\rho_y}\mathbf{z}_y \ . \tag{1.15}$$

Incremental or update thin SVD

Given $D = U_0\Sigma_0 V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r\times r}$ and $\mathbf{d} \in \mathbb{R}^{m\times 1}$, we want to compute

$$\text{thinSVD}([D\ \mathbf{d}]) = U_1\Sigma_1 V_1^T\ , \tag{1.16}$$

with (i) $\Sigma_1 \in \mathbb{R}^{r+1\times r+1}$ or (ii) $\Sigma_1 \in \mathbb{R}^{r\times r}$. In this case $[D\ \mathbf{0}] = U_0\Sigma_0[V_0\ \mathbf{0}]^T$ and $[D\ \mathbf{d}] = [D\ \mathbf{0}] + \mathbf{d}\mathbf{e}^T$, where $\mathbf{e} \in \mathbb{R}^{l+1\times 1}$ is a unit vector and $\mathbf{e}(l+1) = 1$, so (1.13) is equivalent to (1.17)-(1.18) with $\hat{\Sigma} \in \mathbb{R}^{r+1\times r+1}$.

$$[D\ \mathbf{0}] + \mathbf{d}\mathbf{e}^T = [U_0\ \mathbf{p}]\cdot(G\hat{\Sigma}H^T)\cdot\begin{bmatrix} V_0^T & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\ , \tag{1.17}$$

$$G\hat{\Sigma}H^T = \text{SVD}\left(\begin{bmatrix} \Sigma_0 & \mathbf{x} \\ \mathbf{0}^T & \rho_x \end{bmatrix}\right)\ . \tag{1.18}$$

Using (1.19) we can compute thinSVD($[D\ \mathbf{d}]$) with (i) $\Sigma_1 \in \mathbb{R}^{r+1\times r+1}$; similarly using (1.20) we can compute thinSVD($[D\ \mathbf{d}]$) with (ii) $\Sigma_1 \in \mathbb{R}^{r\times r}$, where Matlab notation is used to indicate array slicing operations.

$$U_1 = [U_0\ \mathbf{p}]\cdot G, \quad \Sigma_1 = \hat{\Sigma}, \quad V_1 = \begin{bmatrix} V_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}\cdot H \tag{1.19}$$

$$\begin{aligned} &U_1 = U_0\cdot G(1{:}r, 1{:}r) + \mathbf{p}\cdot G(r{+}1, 1{:}r), \quad \Sigma_1 = \hat{\Sigma}(1{:}r, 1{:}r), \\ &V_1 = [V_0\cdot H(1{:}r,\ 1{:}r)H(r+1,\ 1{:}r)]\ . \end{aligned} \tag{1.20}$$

Downdate thin SVD

Given $[D\ \mathbf{d}] = U_0\Sigma_0 V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r\times r}$, we want to compute

$$\text{thinSVD}(D) = U_1\Sigma_1 V_1^T \tag{1.21}$$

with r singular values. Since $[D\ \mathbf{0}] = [D\ \mathbf{d}] + (-\mathbf{d})\mathbf{e}^T$, the rank-1 modification (1.13) is equivalent to (1.22)-(1.23)

$$[D\ \mathbf{d}] + (-\mathbf{d})\mathbf{e}^T = [U_0\ \mathbf{0}]\cdot(G\hat{\Sigma}H^T)\cdot[V_0\ \mathbf{q}]^T, \tag{1.22}$$

$$G\hat{\Sigma}H^T = \text{SVD}\left(\begin{bmatrix} \Sigma_0\text{-}\Sigma_0\mathbf{y}\mathbf{y}^T & \text{-}\rho_y\cdot\Sigma_0\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix}\right), \tag{1.23}$$

from which we can compute thinSVD(D) via (1.24).

$$\begin{aligned} &U_1 = U_0\cdot G(1{:}r, 1{:}r), \quad \Sigma_1 = \hat{\Sigma}(1{:}r, 1{:}r), \\ &V_1 = V_0\cdot H(1{:}r, 1{:}r) + \mathbf{q}\cdot H(r+1, 1{:}r)\ . \end{aligned} \tag{1.24}$$

Thin SVD replace

Given $[D\ \mathbf{d}] = U_0\Sigma_0 V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r\times r}$, we want to compute

$$\text{SVD}([D\ \hat{\mathbf{d}}]) = U_1\Sigma_1 V_1^T \tag{1.25}$$

with r singular values. Since $[D\ \hat{\mathbf{d}}] = [D\ \mathbf{d}] + \mathbf{c}\mathbf{e}^T$, where $\mathbf{c} = \hat{\mathbf{d}}\text{-}\mathbf{d}$, the rank-1 modification (1.13) is equivalent to (1.26)-(1.27)

$$[D\,\hat{\mathbf{d}}] \;=\; [U_0\ \mathbf{p}] \begin{bmatrix} \Sigma_0 + \mathbf{x}\mathbf{y}^T & \rho_y \cdot \mathbf{x} \\ \rho_x \cdot \mathbf{y}^T & \rho_x \cdot \rho_y \end{bmatrix} [V_0\ \mathbf{q}]^T \tag{1.26}$$

$$G\hat{\Sigma}H^T = \text{SVD}\left(\begin{bmatrix} \Sigma_0 + \mathbf{x}\mathbf{y}^T & \rho_y \cdot \mathbf{x} \\ \rho_x \cdot \mathbf{y}^T & \rho_x \cdot \rho_y \end{bmatrix} \right) \tag{1.27}$$

from which we can compute thinSVD($[D\ \hat{\mathbf{d}}]$) via (1.28).

$$\begin{aligned} U_1 &= U_0 \cdot G(1:r, 1:r) + \mathbf{p} \cdot G(r+1, 1:r), \;\; \Sigma_1 = \hat{\Sigma}(1:r, 1:r) \\ V_1 &= V_0 \cdot H(1:r, 1:r) + \mathbf{q} \cdot H(r+1, 1:r). \end{aligned} \tag{1.28}$$

1.3.3 Incremental RPCA based on the amFastPCP algorithm

In what follows, we assume that we have solved the RPCA up to time $k-1$, i.e.

$$\underset{Lk-1, Sk-1}{\arg\min} \; \|L_{k-1}\|_* + \lambda \|S_{k-1}\|_1 \qquad \text{s.t. } D_{k-1} = L_{k-1} + S_{k-1} \tag{1.29}$$

where $D_{k-1} = D(:, 1:k-1)$. Furthermore we also assume that we know the partial (thin) SVD of $L_{k-1} = U_r \Sigma_r V_r^T$, where $\Sigma_r \in \mathbb{R}^{r \times r}$; this result is usually a by-product of any RPCA algorithm. Although (1.29) can be solved in a fully incremental fashion (described at the end of this section), for the sake of simplicity, here we assume that (1.29) has been solved via the amFastPCP algorithm (see Section 1.2.2).

If we were to solve the RPCA problem from scratch when the next frame $\mathbf{d}_k$ is available via the amFastPCP algorithm, then we need to solve the following alternating optimizations:

$$L_k^{(j+1)} \;=\; \underset{L}{\arg\min} \, \|L_k + S_k^{(j)} - D_k\|_F^2 \qquad \text{s.t. } \text{rank}(L_k) = r \tag{1.30}$$

$$S_k^{(j+1)} \;=\; \underset{S}{\arg\min} \, \|L_k^{(j+1)} + S_k - D_k\|_F^2 + \lambda \|S_k\|_1, \tag{1.31}$$

where $L_k = [L_{k-1}\, \mathbf{l}_k]$, $S_k = [S_{k-1}\, \mathbf{s}_k]$ and $D_k = [D_{k-1}\, \mathbf{d}_k]$. The minimizer of (1.30), for $j = 0$ is given by

$$L_k^{(1)} \;=\; \text{partialSVD}(D_k - S_k^{(0)}) = \text{partialSVD}([D_{k-1}\text{-}S_{k-1}\ \mathbf{d}_k]) \;, \tag{1.32}$$

since (i) we know S_{k-1} and (ii) the amFastPCP algorithm is initialized with a zero sparse solution (see Algorithm 1). Ideally $[D_{k-1}\text{-}S_{k-1}\ \mathbf{d}_k] = [L_{k-1}\ \mathbf{d}_k]$, and therefore the solution of (1.32) can be computed via the incremental, non-increasing rank, thin SVD procedure previously described in Section 1.3.2. Alternatively, the solution of (1.32) could be computed via the rank-increasing

case, which is preferred if the smallest singular value σ_{r+1} has a significant contribution, which could be evaluated in a similar fashions as for the amFastPCP algorithm (see Section 1.2.2 and line 3 in Algorithm 1).

In the case of the sparse component, the minimizer of (1.31) for $j = 0$ is given by

$$S_k^{(1)} \quad = \quad \text{shrink}(D_k - L_k^{(1)}, \lambda) = [S_{k-1}, \text{shrink}(\mathbf{d}_k - \mathbf{l}_k^{(1)}, \lambda)] \ , \qquad (1.33)$$

which is only applied to the current frame, since we know S_{k-1}; this is computationally cheap. The solution (1.30) for $j = 1$ (the next inner loop) follows the same logic:

$$L_k^{(2)} \quad = \quad \text{partialSVD}(D_k - S_k^{(1)}) = \text{partialSVD}([D_{k-1}\text{-}S_{k-1} \ \mathbf{d}_k\text{-}\mathbf{s}_k^{(1)}]) \qquad (1.34)$$

which can be effectively computed using the thin SVD replace procedure, since in the previous step we have computed the partial SVD for $[D_{k-1}\text{-}S_{k-1} \ \mathbf{d}_k]$.

In Algorithm 2, lines $2-6$, we summarize the method described in the previous paragraphs, where incSVD$(\cdot)$, repSVD$(\cdot)$ and dwnSVD$(\cdot)$ refer respectively to the incremental, replace and downdate procedures described in Section 1.3.2.

Algorithm 2: incPCP: incremental amFastPCP.

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 $\quad ++m$

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

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

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

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

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

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

7 $\quad$ **if** $m \geq bL$

$\quad$ **then** dwnSVD("1st column", U_k, Σ_k, V_k)

8 $\quad$ **if** $\|L(:,k) - L(:,k-1)\|_2^2 / \|L(:,k-1)\|_2^2 \geq \tau$

$\quad$ **then** $m = k_0$, $\quad$ use procedure in Section 1.3.3.

For a video shot from a static camera, 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. To model these slow changes we could assume that the background is unchanged for at least bL frames. Afterwards we need to "forget" the background frames that are "too old" and always keep a low-rank estimate of a more or less constant background, which can be done via the downdate procedure, as described in Algorithm 2, line 7. The resulting algorithm is equivalent to a "sliding window" incremental PCP algorithm for which

the background estimate represents the background as observed for the last bL frames.

Finally in Algorithm 2, line 8, we check whether the current low-rank approximation is significantly different from the previous one. In real scenarios this would be related to an abrupt background change (e.g. camera is moved, sudden illumination change, etc.), and if this condition is true, then we re-initialize the background estimate using the procedure described next.

An incremental initialization for Algorithm 2

While Algorithm 2 describes an incremental procedure for RPCA, its initialization seems to hint at the need for a batch computation, which will greatly reduce its usefulness to process live streaming videos. In what follows we describe a fully incremental initialization procedure.

In the context of RPCA and video background modeling, the rank of the low-rank matrix $r \in [2, 8]$ is adequate when analyzing real videos acquired with a static camera, and thus we assume that r is known. Then we can proceed as follows

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

to obtain an estimate of the low-rank approximation of the first $k-1$ frames of the input video.

It was experimentally determined that $r = 1$, $k_0 = 1$ is sufficient, in the long run, to produce very good results. Although this initialization gives a very rough background estimate, this estimate is improved at each iteration (via the incremental SVD procedure, line 2 in Algorithm 2) for bL frames, before any downdate operation is performed. Furthermore, 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 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 that choosing $r = 1$, $k_0 = 1$ for our initialization stage gives good results.

Computational Results

All simulations, related to Algorithm 2 and presented in this section, are available at [19]. They have been carried out using single-threaded Matlab-only code running on an Intel i7-4710HQ quad-core (2.5 GHz, 6MB Cache, 32GB RAM) based laptop with a nvidia GTX980M GPU card. Furthermore, since Algorithm

2 behaves in practice as a "sliding window" incremental PCP (we use $bL = 100$ by default), using rank $r = 1$ is adequate for a good background estimate and good tracking properties.

We have used four real and synthetic video sets as a test videos:

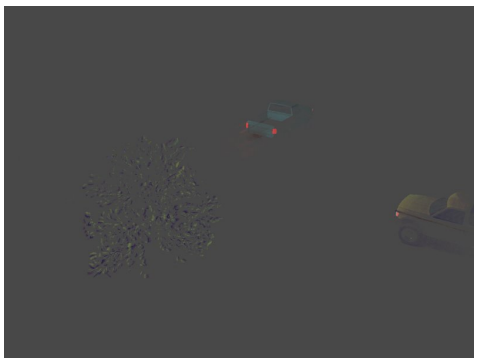
- V320: 320×256 pixel, 1286-frame color video sequence of 51.4 seconds at 25 fps, from the atrium of a mall, with lots of people walking around [20].
- V640: 640×480 pixel, 400-frame color video sequence of 26.66 seconds at 15 fps, from the Lankershim Boulevard traffic surveillance dataset [21, camera3], where a large number of cars are seen.
- V800: 800×600 pixel, 600-frame color synthetic video sequence of a street with highly variable lighting [22].
- V1920: 1920×1088 pixel, 900-frame color video sequence of 36 seconds at 25 fps, from the Neovison2 public space surveillance dataset with observably variable lighting. [23, Tower 23rd video].

Test	Initialization (sec.)		Average per frame (sec.)				
	GRASTA grayscale	GOSUS color	incPCP grayscale		incPCP color		GRASTA grayscale
Video	Standard Matlab + MEX	Standard Matlab + MEX	Standard Matlab	GPU-enabled Matlab	Standard Matlab	GPU-enabled Matlab	Standard Matlab + MEX
V320	23.0	11.0	1.0e-2	1.3e-2	2.0e-2	1.7e-2	3.4e-2
V640	72.4	39.9	2.9e-2	2.4e-2	6.4e-2	3.7e-2	1.1e-1
V800	115.5	52.6	4.6e-2	3.7e-2	9.7e-2	5.7e-2	1.7e-1
V1920	537.5	$^{(*)}$	1.9e-1	1.2e-1	3.4e-1	2.8e-2	8.1e-1

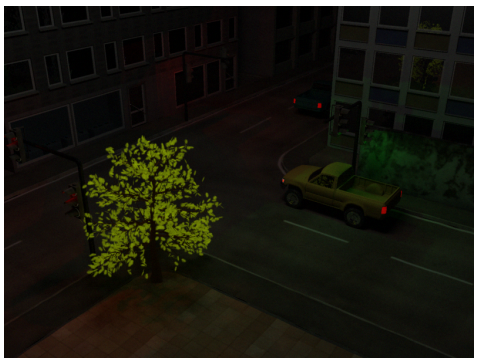
Table 1.1: Elapsed time to initialize the GRASTA [11] and GOSUS [13] algorithms as well as the average processing time per frame for all algorithms on an Intel-i7 (32 GB RAM) based laptop. Note that "e-k" $= 10^{-k}$. $^{(*)}$ Runs out of memory before completing the processing.

We compare our results with the GRASTA and GOSUS algorithms. ReProCS is not considered since the implementation available [24] is not able to deliver a consistent sparse representation for any of the test videos considered. We use [25] (a GRASTA implementation, by the authors of [11], that can only process grayscale videos) with its default parameters: 30% and 10% of the information for the batch initialization and sub-space tracking respectively; we also use [26] (a GOSUS implementation, by the authors of [13], that can only process color videos) with its default parameters, and using the initial 200 frames for the initialization stage. Furthermore, GRASTA and GOSUS implementations are Matlab based which takes advantage of MEX interfaces, and while Matlab implementation of incPCP does not use any MEX interface, it comes in two

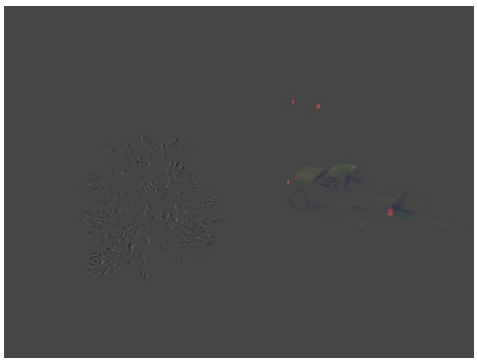
(a) Frame 99.



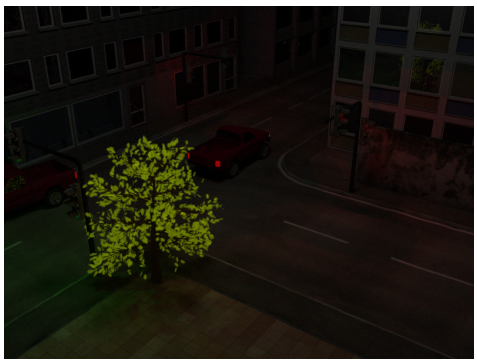
(b) incPCP sparse estimate.



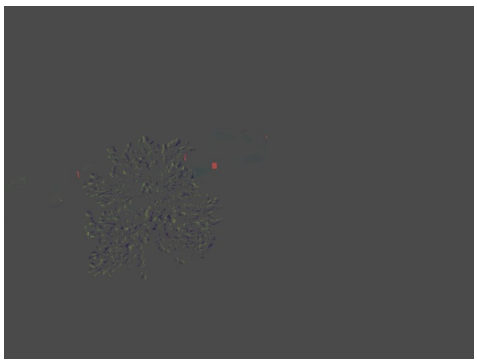
(c) Frame 120.



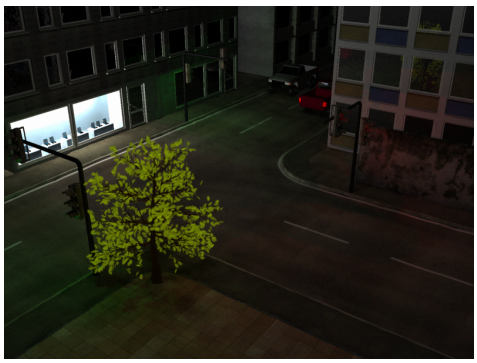
(d) incPCP sparse estimate.



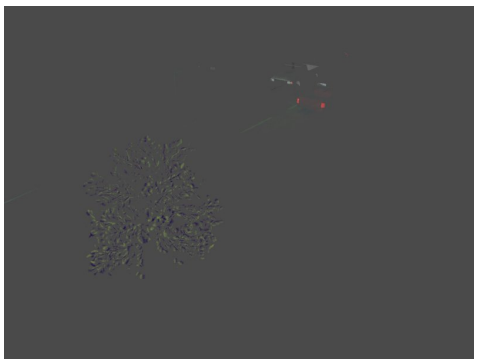
(e) Frame 299.



(f) incPCP sparse estimate.



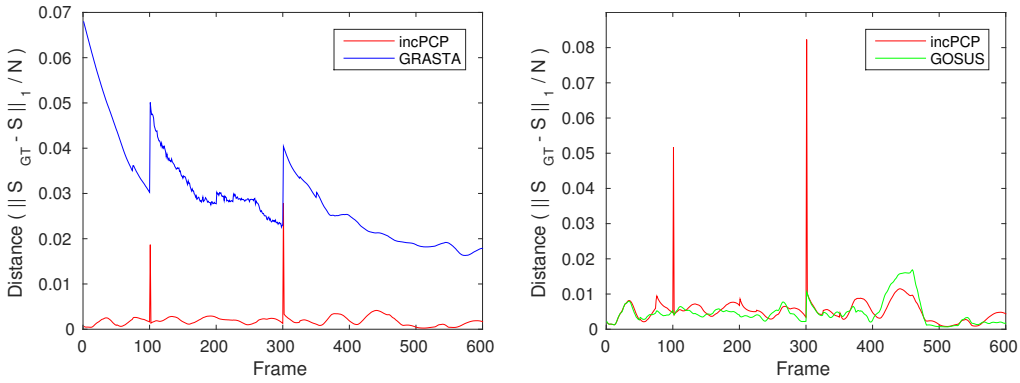
(g) Frame 340.



(h) incPCP sparse estimate.

Figure 1.1: Original and sparse estimates via the incPCP algorithm for the V800 (800×600 pixel) test video, where a variable lighting environment can be

flavors (i) one that uses standard single-thread Matlab code and (ii) one that takes advantage of (mainly linear algebra) GPU-enabled Matlab functions.



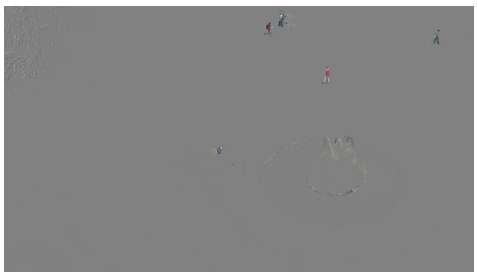
(a) Grayscale case. (b) Color case.

Figure 1.2: Sparse approximation (V800) 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 (a) grayscale (blue) GRASTA and (red) incPCP methods, or (b) color (green) GOSUS and (red) incPCP methods, and N is the number of pixels per frame (used as a normalization factor). It is worth noting that the spikes due to background adaptation to light illumination (frames 100 and 300) are less prominent for GOSUS than for incPCP, since GOSUS uses the first 200 frames (by default) for its batch initialization, and thus it has already observed the background change which is built into its initial subspace approximation.

In order to assess the performance of the incPCP algorithm we present two kinds of results: (i) computational performance measured as the time to process a given video, summarized in Table 1.1, and (ii) reconstruction quality, depicted in Figures 1.2 and 1.4, measured by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the "ground truth" sparse video approximation and S_P is the sparse video approximation of either GRASTA, GOSUS, or incPCP algorithms, and N is the number of pixels per frame, which is same value for either the grayscale or color case.

Since one of the main features of the incPCP algorithm is its ability to process large videos (e.g. full-HD or 1920×1080), results for small videos do not illustrate one of its main advantages. Unfortunately we have been unable to find manually segmented ground-truth for large videos. For example, the BMC or background models challenge dataset [27] has videos of size 320×240; some manually segmented ground-truth frames are available for test video V320 (available from [20]), but only for a very small fraction of each test video (V320: 20 ground-truth frames out of 1286 frames), so that results based on these ground truth frames are not representative of the entire test videos, and could be substantially misleading. Therefore for the V320, V640, V800 and V1920 test videos (see note below for the V800 test video case) we use the sparse video

(a) Frame 160.



(b) incPCP sparse estimate.

(c) Frame 340.

(d) incPCP sparse estimate.

Figure 1.3: Original and sparse estimate via the incPCP algorithm for the V1920 (1088×1920 pixel) test video, where a smooth change in the sun-lighting can be observed.

approximation computed via the batch iALM algorithm [4] with 20 outer loops (10 for the V1920 case due to memory constrains) as a proxy ground-truth since this result has a high level of confidence, see [28, Tables 6 and 7]. For synthetic test video V800, from the Stuttgart Artificial Background Subtraction Dataset [22], the ground-truth is given in the form of background without any moving object, however the sudden light change (see Fig. 1.1) is not included for this dataset, limiting its use.

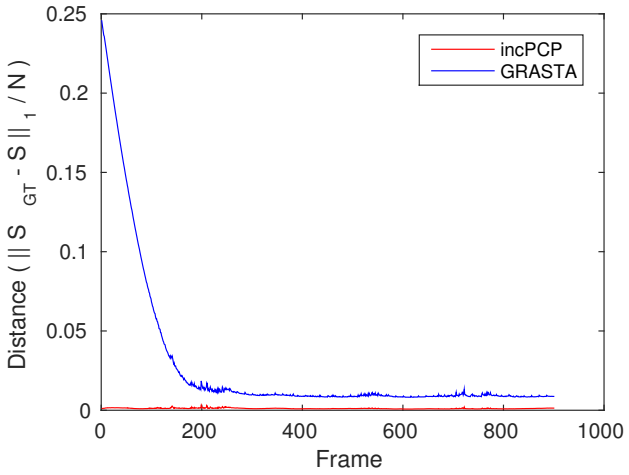


Figure 1.4: Sparse approximation (V1920, grayscale version) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (10 outer loops) and S_P is the sparse approximation computed via the (blue) GRASTA and (red) the incPCP method, and N is the number of pixels per frame (used as a normalization factor).

Without taking into account the batch initializations (for GRASTA and GOSUS, since incPCP does not need one), the average processing time per frame listed in Table 1.1 shows that the incPCP algorithm (Matlab-only version) is between 3 and 4 times faster than GRASTA, when processing grayscale videos, and between 2 and 3 orders of magnitude faster than GOSUS when processing color videos; in addition, the Matlab GPU-enabled incPCP version is about 1.2 $\sim$ 1.5 times faster than the Matlab-only one. Furthermore, results from Table 1.1 suggest that the computational performance of the GRASTA and incPCP (both versions) algorithms has a more or less linear dependence on the the frame size, whereas this is not true for the GOSUS algorithm: GOSUS takes, in average, 26.5 seconds to process each frame of video V640 (640×480 pixel) and 24.2 seconds to process each frame of video V800 (800×600 pixel). Besides the frame size difference between V640 and V800, they also differ in the number and size of moving objects: V640 shows a street with a large number of (relatively) small moving vehicles, whereas in V800 at most two large moving objects are

observed.

Furthermore, the above mentioned speed-up does not take into consideration the time spent by GRASTA and GOSUS in their batch initializations respectively: for instance, GRASTA needs between 23 and 537 seconds to complete its initialization for videos of size between 240×320 and 1088×1920 respectively; in the GOSUS case, its batch initialization takes between 11 and 52 seconds for videos of size between 240×320 and 600×800; while GOSUS is much faster than GRASTA, the memory requirements of GOSUS are much larger than those of GRASTA and can thwart any further processing (as for the V1920 case). Furthermore, such initializations procedures hamper the use of either GRASTA or GOSUS as a real-time alternative to the incPCP algorithm, particularly when the background is dynamic.

In Figures 1.2 and 1.4 we present the reconstruction quality measure for videos V800 and V1920; the other considered test videos have similar results and thus are omitted. Test video V800 has an illumination change in frames 100 and frame 300 (see Figure 1.1); since this is a synthetic video, this sudden illumination change spans only one frame; furthermore, this video also include small changes in the background due to traffic lights, making it a very challenging environment, in which any background modeling algorithm would need fast tracking capabilities as well as adaption capabilities to (more or less) smooth or localised background changes that do not dramatically affect the main background characteristics. In Figure 1.2 we can observed the background tracking properties of the incPCP, GRASTA and GOSUS algorithms. The incPCP algorithm quickly adapts to the above mentioned sudden illumination changes, while at the same time also adapts to other non-dramatic background changes: this is observed due to the sinusoid-like variation in the incPCP restoration quality plot. A similar behavior is also observed for the GOSUS algorithm.

In Figure 1.4 we present the reconstruction quality measure for the V1920 test video. In this case, there is an observable (more or less) smooth change in the background due to changes in the sun-lighting illumination, as can be observed in Figure 1.3. Although V1920 is a color video, the plots shown in Figure 1.4 correspond to its grayscale version since (i) the GRASTA algorithm can only process grayscale videos and (ii) the GOSUS algorithm, which can process color videos, was unable to finish any processing due to memory requirements that exceeded that of the computer (RAM 32 GB) use to run the simulations. In Figure 1.4 it is observed that the incPCP restoration quality measure is almost constant to the smooth illumination change; it also is better than the GRASTA sparse approximation, even after GRASTA has adapted to the background (frame 200 onwards).

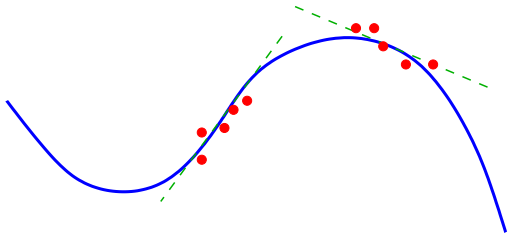


Figure 1.5: Illustration of a set of local tangent planes providing location approximations to a manifold.

1.4 Local Subspace Models and Endogenous Sparse Representations

The RPCA decomposition assumes that there is a single, low-dimensional model that describes most components of the elements of the dataset. This provides a good approximation of the stationary background component of a video sequence captured by a stationary camera, but is not a good model when the camera is non-stationary. In this section we consider generalising the model from a single low-rank subspace with a sparse set of possibly-large deviations, to a low-dimensional *manifold* [29] with the same type of deviations. As a result, the low-dimensional representation is able to vary across the dataset, modeling, for example, the distinct scenes captured by a moving camera.

Our approach is based on modeling local tangent planes of the manifold, assuming that the local sampling density is adequate to make this possible. The geometric intuition for the representation of these subspaces is that every samples in a local tangent plane may be represented as a sparse linear combination of neighboring samples in the same tangent plane, as illustrated in Fig. 1.5.

1.4.1 Endogenous Sparse Representations

This representation corresponds very closely to the union of subspaces model underlying the Sparse Subspace Clustering method, proposed by Elhamifar and Vidal [30, 31, 32, 33], for subspace clustering of data belonging to a set of disjoint subspace. The primary difference is that our goal is not clustering, and we do not assume that the subspaces are sufficiently well separated for subspace recovery results [33] to hold, since our goal is merely to estimate the subspace, not to distinguish it from its neighbours.

The core of this method is computing a sparse representation of the data with respect to itself, an unusual form of sparse representation that has been labeled an *endogenous* sparse representation [34]. The basic form of the sparse coding problem is

$$\arg\min_{X,S} \frac{1}{2} \|DX + S - D\|_F^2 + \lambda_X \|X\|_1 + \lambda_S \|S\|_1 \quad \text{s.t.} \quad \text{diag}(X) = 0 \;, \quad (1.35)$$

where D is the data matrix, X is the sparse representation of D with respect to itself, and S represents a sparse set of outliers, as in the RPCA problem. Theoretical aspects of this model, such as conditions under which subspace recovery is possible, have been considered [35, 36, 34, 33], and a number of variants have been proposed, the most significant of which is the Low Rank Representation [3], constructed by replacing the ℓ_1 norm $\|X\|_1$ with a nuclear norm $\|X\|_*$.

A variant of Eq. (1.35),

$$\begin{aligned}\arg\min_{X,S} \frac{1}{2} \|DX + S - D\|^2 + \alpha\|X\|_1 + \beta\|X\|_{2,1} \\ + \gamma\|S\|_1 + \delta \left\| \sqrt{(\nabla_x S)^2 + (\nabla_y S)^2 + (\nabla_z S)^2} \right\|_1 . \quad (1.36)\end{aligned}$$

was applied to the video background modeling problem in [37]. The main differences are:

- the replacement of the constraint $\mathrm{diag}(X) = 0$ with an $\ell_{2,1}$ norm term on X to encourage structured sparsity, which is also effective in avoiding trivial solutions, and
- the inclusion of a Total Variation (TV) penalty on S to encourage spatial contiguity of the sparse deviations, since in this problem they represent contiguous objects in the foreground.

This simple model does not perform very well in practice because the columns of the endogenous dictionary include the moving foreground objects.

An improved, although more computationally expensive, model can be derived by observing the the sparse deviations (or outliers) S of data D are also the deviations of the dictionary D, so that $(D - S)X$ should provide a better locally low-dimensional approximation than DX:

$$\begin{aligned}\arg\min_{X,S} \frac{1}{2} \|(D - S)X + S - D\|^2 + \alpha\|X\|_1 + \beta\|X\|_{2,1} \\ + \gamma\|S\|_1 + \delta \left\| \sqrt{(\nabla_x S)^2 + (\nabla_y S)^2 + (\nabla_z S)^2} \right\|_1 . \quad (1.37)\end{aligned}$$

This method performs well if there is a sufficient number of samples in each local subspace (i.e. the underlying manifold is sufficiently well sampled to give good estimates of local tangent spaces), as is the case of jitter, very slow panning, or when the camera pans back and forth across the same scene. The performance is demonstrated using a synthetic slowly-panning test video sequence (see Fig. 1.6) constructed by taking a moving 240×320 pixel cropping window within the original sequence, a 288-frame traffic video sequence from the Lankershim Boulevard Dataset [38, camera 4, 8:45–9:00 AM]. This window moves slowly to the left, and then back to the original position, at a rate of 1/4 pixel/frame. In this case the background is not very well approximated by any

single low-dimensional subspace, but since the background motion is slow with respect to the foreground motion, a locally low-dimensional model provides a substantially better approximation, as illustrated in Fig. 1.7.

Figure 1.6: Frame 180 of a synthetic slowly-panning test video sequence.

The model of Eq. (1.37) does not perform well when the camera motion is less constrained. The next section introduces a modification to the form of representation that allows this method to handle greater translations between frames, corresponding to low sampling density on the manifold of video frames.

1.4.2 Endogenous Convolutional Sparse Representations

Recall that the standard Basis Pursuit DeNoising (BPDN) sparse coding problem [39] with dictionary matrix A can be expressed as

$$\arg\min_{\mathbf{x}} \frac{1}{2} \|A\mathbf{x} - \mathbf{s}\|_2^2 + \lambda \|\mathbf{x}\|_1 \ . \tag{1.38}$$

The convolutional form [40] is obtained by replacing the linear combination of columns of A with a sum of convolutions by a set of dictionary filters $\{\mathbf{h}_m\}$

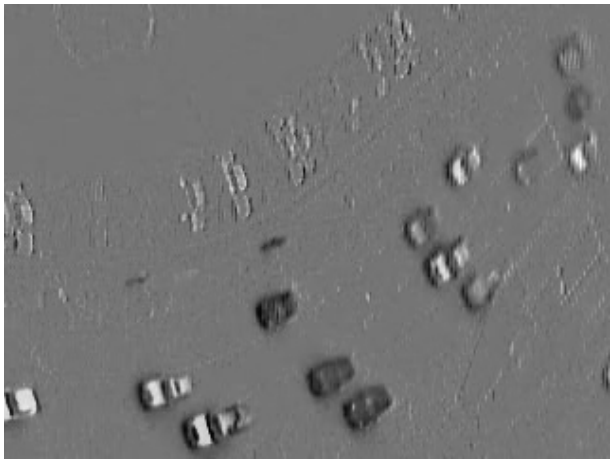
$$\arg\min_{\{\mathbf{x}_m\}} \frac{1}{2} \left\| \sum_m \mathbf{h}_m * \mathbf{x}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \ . \tag{1.39}$$

This is substantially more computationally expensive than standard sparse representations, but considerable progress has recently been made in developing faster algorithms [41].

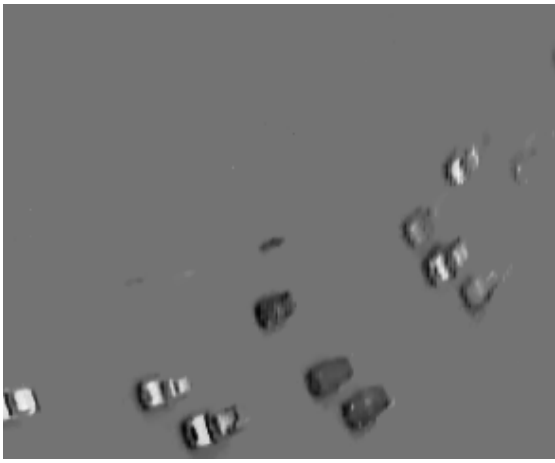
In an endogenous context, the filters $\{\mathbf{h}_m\}$ are themselves images from the data set of interest, and the corresponding $\{\mathbf{x}_m\}$ are now 2-d coefficient maps,

(a) RPCA low rank

(b) Local low rank



(c) RPCA sparse



(d) Local sparse

Figure 1.7: Results for frame 180 from the slowly-panning test video sequence. The RPCA results were obtained using RPCA with the standard choice of parameters, and the local subspace representation results were computed using Eq. (1.37) with parameters $\alpha = 4.0 \times 10^{-3}, \beta = 8.0 \times 10^{-2}, \gamma = 5.0 \times 10^{-4}$, and $\delta = 3.0 \times 10^{-4}$.

rather than simple scalars in the case of standard sparse representations. The problem to be solved can be expressed as

$$\begin{aligned} \operatorname*{arg\,min}_{\{\mathbf{x}_{m,k}\},\{\mathbf{s}_k\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{h}_m * \mathbf{x}_{m,k} + \mathbf{s}_k - \mathbf{d}_k \right\|_2^2 + \lambda \sum_k \sum_m \|\mathbf{x}_{m,k}\|_1 + \mu \sum_k \|\mathbf{s}_k\|_1 \\ \text{such that } \mathbf{x}_{m,k} = 0 \, \forall m \in \mathcal{E}_k \;, \end{aligned} \tag{1.40}$$

where $\mathcal{E}_k = \{m \mid \mathbf{h}_m = \mathbf{d}_k\}$ as the set of indices m for signal $\mathbf{d}_k$ for which dictionary element $\mathbf{h}_m$ is the same datum as that signal. Expressing this problem in the form

$$\begin{aligned} \operatorname*{arg\,min}_{\{\mathbf{x}_{m,k}\},\{\mathbf{s}_k\}} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{h}_m * \mathbf{x}_{m,k} + \mathbf{s}_k - \mathbf{d}_k \right\|_2^2 + \lambda \sum_k \sum_m \|\mathbf{x}_{m,k}\|_1 + \mu \sum_k \|\mathbf{s}_k\|_1 \\ + \sum_k \sum_{m \in \mathcal{E}_k} \iota(\mathbf{x}_{m,k}) \text{ such that } \mathbf{x}_{m,k} - \mathbf{y}_{m,k} = 0 \;, \end{aligned} \tag{1.41}$$

where $\iota(\cdot)$ is zero if its argument is a zero vector and infinite otherwise, allows this problem to be solved via relatively minor modification[1] of a recent ADMM algorithm for the standard Convolutional BPDN problem [41].

The advantage that this type of sparse representation brings to video background modeling is that, due to the properties of convolution, integer-pixel translations of the $\{\mathbf{h}_m\}$ can be compensated for by translations within the corresponding coefficient maps $\{\mathbf{x}_{m,k}\}$, allowing translation invariant subspace modeling [42]. In the simplest case, the set of filters is the same as the set of frames, so that $\mathbf{h}_k = \mathbf{d}_k$, and $\{\mathbf{x}_{m,k}\}$ represents the contribution of frame m to representing frame k. The goal of the sparsity penalty on $\{\mathbf{x}_{m,k}\}$ is that each coefficient map $\{\mathbf{x}_{m,k}\}$ should be very sparse, with a single very localised concentration of non-zero coefficients, the position of which indicates the relative alignments of frames k and m. Since periodic boundary conditions are not suitable for most video frames, boundary issues arising from the misalignment between frames k and m need to be considered so that the solution is not perturbed by mismatch between the frames in the boundary region. The simplest solution is to mask out a boundary region, the size of which is determined by the largest translation that is desired to be tolerated, in the data fidelity term of Eq. (1.40). Unfortunately such a spatial domain projection is not compatible with the efficient frequency domain solution of the problem [41], but as pointed out in [42], the same effect can instead by achieved by the use of a spatial

[1] In the algorithm described in [42], $\mathbf{s}_k$ is dealt with independently from $\mathbf{x}_{m,k}$. This approach works, but can take many iterations to converge, and convergence is somewhat sensitive to correct choice of the auxiliary parameters λ and μ. More recently, it has become clear that a more effective approach is to absorb the outliers as an additional coefficient map with an impulse filter

$$\sum_m \mathbf{h}_m * \mathbf{x}_{k,m} + \mathbf{s}_k = \sum_m \mathbf{h}_m * \mathbf{x}_{k,m} + \boldsymbol{\delta} * \mathbf{s}_k$$

and then apply the standard Convolutional BPDN algorithm.

weighting in the ℓ_1 norm of the $\mathbf{s}_k$ so that there is no penalty at all on $\mathbf{s}_k$ in the boundary region.

	RPCA	ESR	ECSR	ECSR-R	ECSR-M
Back	4.6dB	7.6dB	10.9dB	14.7dB	19.3dB
Fore	-0.5dB	1.9dB	3.6dB	9.6dB	14.2dB

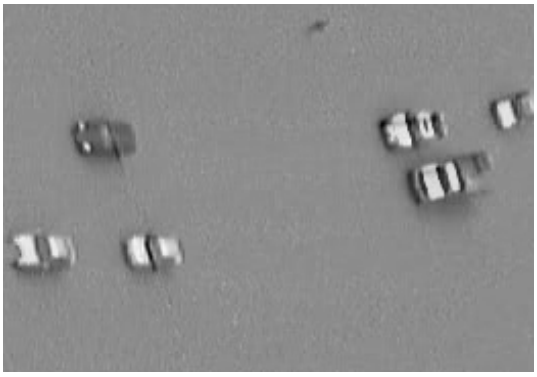
Table 1.2: Comparison of background/foreground separation performance, measured as SNR against ground truth, of RPCA, the endogenous sparse representation method of [37] (ESR), the proposed method without post-processing (ECSR), and the proposed method with RPCA and median filtering post-processing (ECSR-R and ECSR-M respectively).

This method was tested using a video sequence with synthetic translational panning constructed as in Eq. (1.40), but with with the 240×320 pixel cropping window moving within the original sequence at a rate of 3 pixels/frame. Due to the increased computational and memory cost of the convolutional form of the problem, the test sequence consisted of only 30 frames. Since the panning view was simulated, approximate ground truth background and foreground could be constructed from the RPCA result for the original uncropped video (see Figs. 1.8(a) and 1.8(b)). Direct application of Eq. (1.40) to this test sequence (with $\lambda = 50$ and $\mu = 0.06$) gave substantially better performance than the non-convolutional sparse models described in Sec. 1.4.1 (see Figs. 1.8(e) and 1.8(f)), and much better performance than standard RPCA (see Figs. 1.8(c) and 1.8(d)), but performance was still somewhat disappointing (see Figs. 1.9(a) and 1.9(b)). Inspection of the set of $\mathbf{d}_m * \mathbf{x}_{m,k}$ contributing to the background estimate for frame k revealed that the actual frame alignments produced by the positioning of the non-zero coefficients in the $\mathbf{x}_{m,k}$ were quite accurate, but their sum contained artifacts due to the different positions of the foreground objects in the aligned frames $\mathbf{d}_m$; the same phenomenon that prompted the change from Eq. (1.36) to Eq. (1.37) in Sec. 1.4.1 [2]

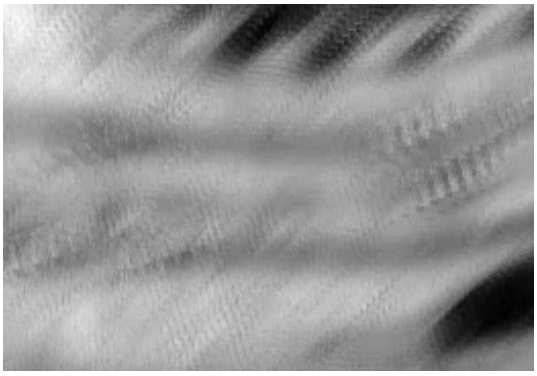
Since the method was able to produce frame alignment robust to boundary effects and the motion of foreground objects, two alternative methods were considered. Both of these methods use the $\mathbf{x}_{m,k}$ obtained by solving Eq. (1.40) to construct, for each frame k, a set of additional frames $\mathbf{d}_m * \mathbf{x}_{m,k}$ with their backgrounds aligned to that of frame k. The first of these methods simply normalizes each member of the set and applies median filtering in the temporal direction, and the second applies RPCA independently to the set of estimates for each frame k. The resulting estimate of the background is then subtracted from the original sequence to obtain an estimate of the foreground that improves on the $\mathbf{s}_k$ from Eq. (1.40) (see Figs. 1.9(c) to 1.9(f)). Objective performance as measured by SNR against ground truth of the different methods described here

[2] The obvious path is to use filters with the sparse component subtracted as the dictionary, in analogy with Eq. (1.37). This direction was not pursued, however, since it became apparent that the alternative incremental approach described in Sec. 1.5 gave good performance at vastly reduced computational cost.

(a) Ground truth



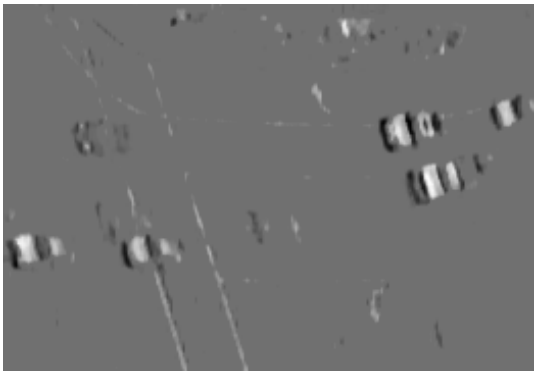
(b) Ground truth



(c) RPCA

(d) RPCA

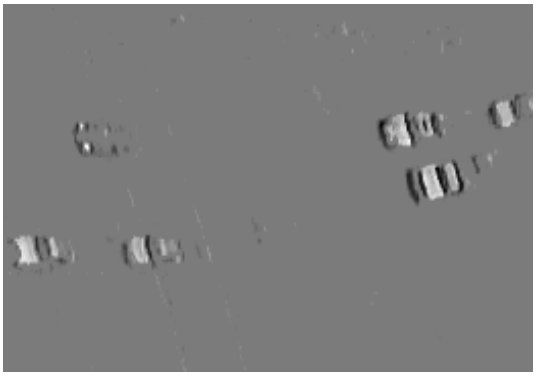
(e) ESR



(f) ESR

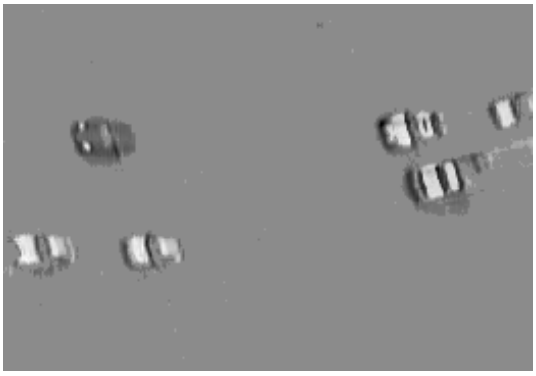
Figure 1.8: Background (first row) and foreground (second row) estimates for frame 15 from the fast-panning test video sequence, computed via the endogenous sparse representation method of [37] (ESR) and the proposed method with median filtering post-processing (ECSR-Median).

(a) ECSR



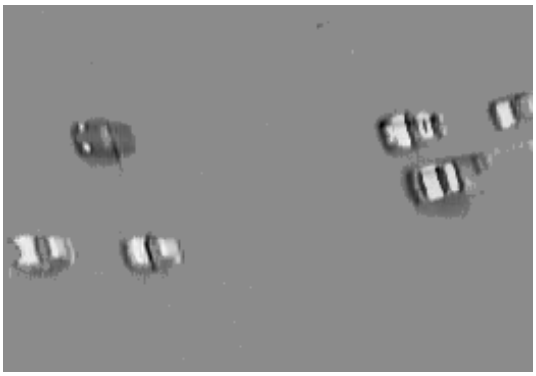
(b) ECSR

(c) ECSR-RPCA



(d) ECSR-RPCA

(e) ECSR-Median



(f) ECSR-Median

Figure 1.9: Background (first row) and foreground (second row) estimates for frame 15 from the fast-panning test video sequence, computed via the endogenous sparse representation method of [37] (ESR) and the proposed method with median filtering post-processing (ECSR-Median).

is compared in Table 1.2.

1.5 Transform Invariant Incremental RPCA

The main motivation for the development of an incremental RPCA algorithm, such as the one described in Section 1.3.3, is to have a procedure with an extremely low memory footprint, and a computational complexity that allows real-time processing. Algorithm 2 does the job, eliminating the batch processing mode, typically needed for RPCA algorithms whereby a large number of frames have to be observed before starting any processing. However, Algorithm 2, as well as most RPCA algorithms, has a high sensitivity to camera jitter, which can affect airborne and space-based sensors [43] as well as fixed ground-based cameras [28] subject to wind.

Here we overcome the jitter sensitivity of Algorithm 2 by incrementally solving

$$\min_{L,S,\mathcal{T}} \frac{1}{2}\|D - \mathcal{T}(L^*) - S\| + \lambda\|S\|_1 \quad \text{s.t.} \quad \text{rank}(L^*) \leq r \;, \tag{1.42}$$

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. To make the solution of (1.42) tractable, we focus on the case where $\mathcal{T}_k$ is a rigid transformation, i.e. the camera used to acquire the video frames suffers from translational and rotational jitter. Furthermore, inspired by convolutional sparse representation (see Section 1.4.2) the solution to (1.42) includes a set of filters that describes a translation / rotation invariant sparse representation (see [44, 45, 46, 47] for other applications on this topic).

1.5.1 Related Work

Transform Invariant Sparse Representation

Transform invariant sparse representation is closely related to convolutional sparse representations, although there have been independent streams of research [48]. For this section we specifically focus on [44, 45, 46, 47] because these works are relevant to the transform invariant RPCA algorithm described in Section 1.5.2.

In the traditional sparse representation model, given the overcomplete dictionary Φ we seek to express observed data $\mathbf{s}$ via

$$\mathbf{s} = \Phi\mathbf{x} + \boldsymbol{\epsilon} \;, \tag{1.43}$$

where $\mathbf{x}$ is the desired sparse representation and $\boldsymbol{\epsilon}$ is noise. There are several algorithms to solve (1.43), among them Basis Pursuit and Matching Pursuit are arguably the most well-known ones.

The sparse representation model can be extended to handle transformation invariance via:

$$\mathbf{s} = \sum_k \tau_k(\Phi)\mathbf{x}^{(k)} + \boldsymbol{\epsilon}\ , \tag{1.44}$$

where $\tau_k(\cdot)$ is the operator that accounts for the desired transformation; we note that this model could also be applied to patches in the observed data, giving in this case $\mathbf{s}^{(n)} = \sum_k \tau_k(\Phi)\mathbf{x}^{(n,\,k)} + \boldsymbol{\epsilon}$ where index n describes a particular sub-divison of the input data. The model described in (1.44) has been used in (i) [44], along with a particular sub-divison of the input data, for shift-invariant sparse coding; (ii) [45] adapts (1.44) for the case of digital images sparse representation via

$$\mathbf{s} = \sum_{k,\,n} T_{k,\,n} R_{\theta_{k,\,n}} G_{k,\,n} \mathbf{x}^{(k,\,n)} + \boldsymbol{\epsilon}, \tag{1.45}$$

where $T_{k,\,n}$ and $R_{\theta_{k,\,n}}$ represent translation and rotation (angles $\theta_{k,\,n}$ are manually chosen) operators, and $G_{k,\,n}$ represents local structures of the image; (iii) in [46] a transform invariant sparse coding approach is taken, adapting (1.44) to

$$\mathbf{s} = \sum_k \sum_n \boldsymbol{\alpha}^{(k,\,n)} * \tau_n(G_k), \tag{1.46}$$

where $\tau_n(\cdot)$ represents a set of pre-specified transformations, $*$ represents convolution, and G_k is a set of desired feature images; (iv) [47] also focus on a transform invariant sparse coding, with emphasis on shift and rotation invariance for multivariate signals, where using a representation akin to (1.45) along with a multivariate orthogonal matching pursuit approach is used to proposed a shift and rotation invariant new dictionary learning algorithm able to provide a robust decomposition.

Transform Invariant Algorithms in the RPCA context

- **RASL algorithm**: The Robust Alignment by Sparse and Low-rank decomposition (RASL) algorithm [49] was introduced as a batch PCP method able to handle misaligned video frames by solving

$$\min_{L,S,\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t.} \quad \tau(D) = L + S \tag{1.47}$$

  where $\tau(\cdot) = \{\tau_k(\cdot)\}$ is a set of independent transformations[3] (one per frame), each having a parametric representation, such that $\tau(D)$ aligns all the observed video frames.

  In [49], the non-linearity in (1.47), is handled via:

$$\min_{L,S,\Delta\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t.} \quad \tau(D) + \sum_{k=1}^{n} J_k \Delta\tau_k \epsilon_k = L + S \tag{1.48}$$

[3]Note that $\tau(\cdot)$ is the inverse of $\mathcal{T}(\cdot)$ in (1.42).

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 [49] mainly focus on rigid transformations; it is also stated that as long as the initial misalignment is not too large, (1.48) effectively recovers the correct transformations.

- **t-GRASTA algorithm**: The GRASTA [11] method is an "on-line" algorithm for low rank subspace tracking. It uses a reduced number of frames, $q \ll n$, compared to the RPCA problem (1.1), 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 sub-sampled version of all the available frames for initialization.

  The Transformed Grassmannian robust adaptive subspace tracking algorithm (t-GRASTA) [50] is based on the GRASTA [11] 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 (1.48) via a modified GRASTA algorithm. It uses a batch mode to train an initial low rank subspace considering the first p frames (the default is $p = 20$ [51]). The initial transformation (τ in (1.48)), is estimated by using a similarity transformation taking a group of points manually chosen from the corresponding original and canonical frames. Computational results in [50] focused on rigid transformations.

1.5.2 Rigid Transformation Invariant Incremental RPCA

We assume that 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 $\mathcal{T}_k$ (see (1.42)):

$$\mathbf{d}_k = \mathcal{T}_k(\mathbf{d}_k^*) = \mathbf{h}_k * R(\mathbf{d}_k^*, \alpha_k) \tag{1.49}$$

where $R(\mathbf{d}_k^*, \alpha_k)$ denotes rotation by angle α_k and the filter $\mathbf{h}_k$ is (ideally) an "uncentered" Dirac delta function that accounts for the translation; furthermore it is important to notice that the center of rotation could be any given point, since such rotation can always be expressed as the rotation around a fixed point followed by a translation. It is also interesting to note that in this observation model, the transformation $\mathcal{T}_k(\cdot)$ is the inverse of the transformation $\tau_k(\cdot)$, used for RASL and t-GRASTA (see previous section) algorithms.

Since $\mathbf{d}_k = \mathbf{l}_k + \mathbf{s}_k$, then (1.49) implies $\mathbf{l}_k = \mathcal{T}_k(\mathbf{l}_k^*) = \mathbf{h}_k * R(\mathbf{l}_k^*, \alpha_k)$. In general, $\mathbf{h}_k$ and α_k are unknown, however if we are able to compute the rigid

transform invariant sparse representation given by

$$\mathbf{l}_k \quad = \quad \sum_n \mathbf{g}_{k,\,n} * R(\mathbf{l}_k^*, \beta_{k,\,n}), \tag{1.50}$$

where $*$ represents convolution, then (1.42) can be incrementally solved via

$$\begin{aligned} \underset{L,S,G}{\arg\min} \quad & \frac{1}{2}\sum_k \left\| \sum_n \mathbf{g}_{k,\,n} * R(\mathbf{l}_k^*, \beta_{k,\,n}) + \mathbf{s}_k - \mathbf{d}_k \right\|_2^2 + \lambda \|S\|_1 \\ & + \sum_k \gamma_k \sum_n \|\mathbf{g}_{k,\,n}\|_1 \quad \text{s.t. } \operatorname{rank}(L^*) \leq r, \end{aligned} \tag{1.51}$$

where G represents the set of filters $\{\mathbf{g}_{k,\,n}\}$. Before delving into details of (1.51), we first elaborate on the chosen sparse representation (1.50) as well as in its relationship with other transform invariant sparse representations, such [44, 45, 46, 47].

While the sparse representation (1.50) seems to consists of both variables $\mathbf{g}_{k,n}$ and $\beta_{k,n}$, we stress that rotation angles $\beta_{k,n}$ are manually sampled, thus making (1.50) an endogenous convolutional sparse representation, akin to that described in Section 1.4.1. Using this direct approach (which has also been used in [45, 46, 47]) we first sample $\beta_{k,n}$, then we compute the rotations $R(\mathbf{l}_k^*, \beta_{k,\,n})$, which will be equivalent to the desired feature images, composed of rotated versions of previous low-rank approximations, giving as a result an endogenous representation, when we assume that the background has not changed. Since the sought rigid transformation can always be expressed as just one rotation followed by a translation, then the sparse variable in (1.50) will be $\mathbf{g}_{k,n}$ since (ideally) only one filter will be active. Clearly this angle sampling is a very naive approach, which increases the complexity as the sample size increases; in order to reduce such unnecessary increased complexity, several options have been proposed: (i) although a linear sampling of 2π is used in [45], the rotations are applied to small images patches, and the update of the coefficients is reduced to the neighbor search, (ii) [46] used a predefined set of rotation operators (using linear interpolation between the image pixels) which along the SignSearch algorithm [52] allows to seek the solution using a gradient descent like approach, (iii) [47] used complex kernels (filters) and coding coefficient resulting in kernels the are no longer learned through a particular orientation, but in kernels that are shift and rotation invariant.

In its algorithmic form, the incremental solution of (1.51) uses a simple and computationally efficient FFT based algorithm [53] to rotate a given image around any given point, along with a line search to reduce complexity of the naive approach describe above. We will elaborate on this on the following subsections, however we first note that an incremental solution of (1.51) is possible by modifying the incremental amFastPCP algorithm (see Section 1.3.3). If we assume that we have solved problem (1.51) up to frame $k-1$ then we can first solve

$$\hat{\mathbf{h}}_k,\, \hat{\alpha}_k \quad = \quad \underset{G}{\arg\min}\, \frac{1}{2}\| \sum_n \mathbf{g}_{k,\,n} R(\mathbf{l}_k^*, \beta_{k,\,n}) + \mathbf{s}_k - \mathbf{d}_k\|_2^2 + \gamma_k \sum_n \|\mathbf{g}_{k,\,n}\|_1, \tag{1.52}$$

via the algorithm described in Section 1.4.2. In order to simplify the notation we assume that only one filter is activated by the solution of (1.52), thus estimating the pair $\hat{\mathbf{h}}_k$ (translation) and $\hat{\alpha}_k$ (rotation).

With the solution to (1.52), problems (1.53)- (1.54)

$$\arg\min_{\mathbf{l}_k^*} \frac{1}{2}\|\hat{\mathbf{h}}_k * R(\mathbf{l}_k, \hat{\alpha}_k) + \mathbf{s}_k - \mathbf{d}_k\|_2^2 \quad \text{s.t. } \operatorname{rank}(L_k^*) \leq r, \quad (L_k^* = [\mathbf{l}_1^*, \mathbf{l}_2^*, \ldots, \mathbf{l}_k^*]) \tag{1.53}$$

$$\arg\min_{\mathbf{s}_k} \frac{1}{2}\|\hat{\mathbf{h}}_k * R(\mathbf{l}_k^*, \hat{\alpha}_k) + \mathbf{s}_k - \mathbf{d}_k\|_2^2 + \lambda\|\mathbf{s}_k\|_1 \tag{1.54}$$

resemble problems (1.30)-(1.31). Furthermore, although sub-problem (1.53) is no longer a low-rank approximation problem, it can also be solved incrementally since it represents an Affinely Constrained Matrix Rank Minimization [54, 55] problem, whereas (1.54) is just element-wise shrinkage.

In the next sub-sections, we delve into the mathematical details of solving (1.51); this encompasses the following topics: (i) image rotation as 1D convolutions [53], (ii) affinely constrained matrix rank minimization [54, 55] and (iii) a practical solution to (1.52). Finally the Rigid Transformation Invariant Incremental RPCA algorithm is fully listed.

Image rotation as 1D convolutions

In [53] a simple FFT based algorithm was proposed to rotate an image around any given point. An image $\mathbf{d}^*$ can be rotated by α degrees, 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 rotation point is defined by the origin of the x and y axes, which easily could be changed to be any point.

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 such 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 Fourier transforms.

Although there are several methods (e.g. [56, 57]), based on [53], that target rotational alignment, these algorithms assume that either a pseudopolar transformation is applied to the observed (rotated) image, or that the Fourier domain can be sampled with arbitrary trajectories. Next we describe a simple procedure (to the best of our knowledge not previously exploited) to estimate the rotation of two images around a known point.

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

$$\alpha \quad = \quad \arg\min_{\alpha} \frac{1}{2}\|R(\mathbf{u}, \alpha) - \mathbf{b}\|_2^2 \tag{1.55}$$

will estimate the angle α^* that relates $\mathbf{u}$ and $\mathbf{b}$. Unfortunately (1.55) is not convex in α; however, it can be experimentally determined that (1.55) is a concave functional with a unique minimum, within a region close enough to the optimum. Moreover, 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 [58] to solve (1.55).

Affinely Constrained Matrix Rank Minimization

The affinely constrained matrix rank minimization problem, defined by (1.56)

$$\min_{L} \quad : \quad \frac{1}{2}\|\mathcal{A}(L) - D\|_F^2 \text{ s.t. } \text{rank}(L) \leq r \tag{1.56}$$

arises in many practical applications such as collaborative filtering and matrix completion [59], system identification and optimal control [60], low-dimensional Euclidean embedding [61], etc.

Problem (1.56) is, in general, difficult to solve. However in recent years several iterative hard thresholding (IHT) based algorithms have been propose [54], as well as iterative reweighted methods [55] that are both simple and computational appealing. In what follows we briefly described a simple IHT based method that suits our needs of an incremental solution (see (1.60) in the next sub-section).

Based on [54], we choose to compute the minimizer to (1.56) via the IHT procedure given by $L^{(j+1)} = \text{partialSVD}(L^{(j)} - \mathcal{A}^*(\mathcal{A}(L^{(j)}) - D), r)$, where $\text{partialSVD}(\cdot, r)$ represents the partial (thin) SVD which only considers the r largest singular values of the input and $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$.

Rigid Transformation Invariant Incremental RPCA Algorithm

As mentioned before, (1.51) includes an endogenous convolutional sparse representation which would allow to solve the rigid transformation invariant problem (1.42) incrementally. However the computational cost of such solution becomes prohibitive as the as the number of sample angles increases: in our experiments (see Section 1.5.3) a resolution of 0.01 degrees is needed to avoid artifacts in the sparse approximation, and without any a-priori information a linear sampling with such resolution is computationally infeasible.

Instead of solving (1.51), here we directly solve for the ideal case where just one filter is activated; this gives the simplified problem

$$\begin{aligned} \underset{L,S,H,\boldsymbol{\alpha}}{\arg\min} \quad & \frac{1}{2}\sum_k \|h_k * R(\mathbf{l}_k^*, \alpha_k) + \mathbf{s}_k - \mathbf{d}_k\|_F^2 + \lambda\|S\|_1 \\ & + \gamma \sum_k \|h_k\|_1 \quad \text{s.t. } \text{rank}(L^*) \leq r \end{aligned} \tag{1.57}$$

where H represents the set of filters $\{h_k\}$ (just one per frame), which models the translational jitter, and $\boldsymbol{\alpha}$ represents the set of angles $\{\alpha_k\}$ that models the rotational jitter.

Using the ideas mentioned before, (1.57) can be solved incrementally via the alternating minimization, recalling that $L_k^* = [\mathbf{l}_1^*, \mathbf{l}_2^*, \ldots, \mathbf{l}_k^*]$ is the set of the first k unobserved and aligned background (low-rank representation):

$$\mathbf{h}_k^{(j+1)} = \arg\min_{\mathbf{h}_k} \frac{1}{2}\|\mathbf{h}_k * R(\mathbf{l}_k^{*(j)}, \alpha_k^{(j)}) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2 + \gamma\|\mathbf{h}_k\|_1 \,, \tag{1.58}$$

$$\alpha_k^{(j+1)} = \arg\min_{\alpha_k} \frac{1}{2}\|\mathbf{h}_k^{(j+1)} * R(\mathbf{l}_k^{*(j)}, \alpha_k) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2 \,, \tag{1.59}$$

$$\mathbf{l}_k^{*(j+1)} = \arg\min_{\mathbf{l}_k^*} \frac{1}{2}\|\mathbf{h}_k^{(j+1)} * R(\mathbf{l}_k^*, \alpha_k^{(j+1)}) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2 \quad \text{s.t. } \operatorname{rank}(L_k^*) \leq 60 \tag{1.60}$$

$$\mathbf{s}_k^{(j+1)} = \arg\min_{\mathbf{s}_k} \frac{1}{2}\|\mathbf{h}_k^{(j+1)} * R(\mathbf{l}_k^{*(j+1)}, \alpha_k^{(j+1)}) + \mathbf{s}_k^{(j)} - \mathbf{d}_k\|_F^2 + \lambda\|\mathbf{s}_k\|_1 \,. \tag{1.61}$$

Sub-problems (1.58), (1.59) and (1.61) are simple to handle:

- Sub-problem (1.58) can be solved by a variable splitting approach, transforming it into $\arg\min_{\mathbf{h},\mathbf{g}} \frac{1}{2}\|\mathbf{g} * \mathbf{u} - \mathbf{b}\|_F^2 + \gamma\|\mathbf{h}\|_1 + \frac{\mu}{2}\|\mathbf{h} - \mathbf{g}\|_2^2$ and then applying alternating minimization on $\mathbf{h}$ and $\mathbf{g}$. The $\mathbf{h}$ update is solved by soft-thresholding, and the $\mathbf{g}$ update can be efficiently solved in the Fourier domain via a special case of the method described in Section 1.4.2 (see also [41]).
- Sub-problem (1.59) is equivalent to (1.55), and can be effectively solved via the same procedure.
- the solution to sub-problem (1.61) is just element-wise shrinkage.

Sub-problem (1.60) is not as straightforward as all other sub-problems, but 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 (1.60) can be effectively solved via the IHT algorithm used to solve (1.56).

1.5.3 Computational Results for the Rigid Transformation Invariant Incremental RPCA (incPCP_TI) algorithm

We have used four 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, from the Change Detection 2014 dataset [62] also used in [63, Ch. 16] and in [51].

- 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].

- V720: a 720×480 pixel, 1150-frame color video sequence, from a badminton court, where several players can be seen in action, from the Change Detection 2014 dataset [62].

- V1920: a 1920 × 1088 pixel, 900-frame color video sequence of 36 seconds at 25 fps, from the Neovison2 public space dataset [23, Tower 3rd video].

Test video	Initialize (sec.)	Average per frame (sec.)				
	t-GRASTA grayscale	incPCP_TI grayscale		incPCP_TI color		t-GRASTA grayscale
	Standard Matlab	Standard Matlab	GPU-enabled Matlab	Standard Matlab	GPU-enabled Matlab	Standard Matlab
V352	65.2	0.39	0.27	0.63	0.37	0.88
V640	232.6	1.49	0.60	2.45	0.82	3.40
V720	339.8	1.67	0.55	2.95	0.96	4.28
V1920	1858.7	10.20	2.9	18.04	5.46	25.6

Table 1.3: Elapsed time to initialize the t-GRASTA [11] and average processing time per frame for t-GRASTA and incPCP_TI. Video sets V352 and V720 have real jitter, whereas sets V640, V1920 have synthetic jitter: random uniformly distributed translations (±T pixel, T= 5) and rotations (±α degrees, $\alpha = 0.5$).

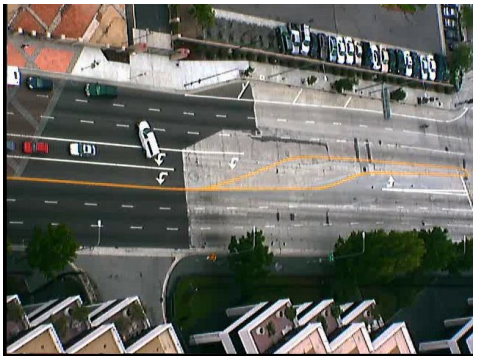
All simulations presented in this section, related to the above described algorithm, have been carried out using Matlab-only code[4] running on an Intel i7-4710HQ quad-core (2.5 GHz, 6MB Cache, 32GB RAM) based laptop with a nvidia GTX980M GPU card. Videos "V640" and "V1920" were synthetically jittered with random uniformly distributed translations (±T pixel, T= {5, 10}) and rotations (±α degrees, $\alpha = \{0.5, 1\}$), and used as controlled datasets with known alignment, whereas videos "V352" and "V720" do have real jitter. Moreover video "V720" has manually segmented ground-truth (moving objects) from frame 800 onwards, and it is a very challenging because, besides its mainly translational jitter, blurred frames are observable; this generates the side effect of an ever changing background.

We compare our results with t-GRASTA [50] using a t-GRASTA implementation [51] for grayscale videos by the authors of [50] (using its default parameters). We do not compare with RASL [49], since it is a batch method. Furthermore, t-GRASTA implementation is Matlab based and takes advantage of MEX interfaces, and while Matlab implementation of incPCP_TI does not

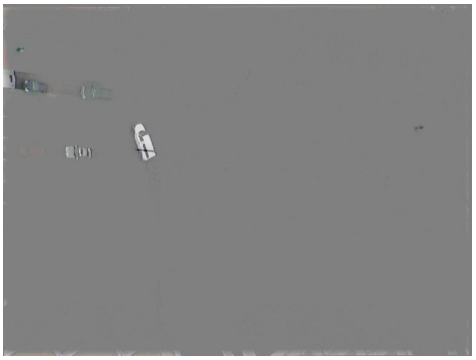
[4]This code is publicly available [64].

(a) Frame 30.

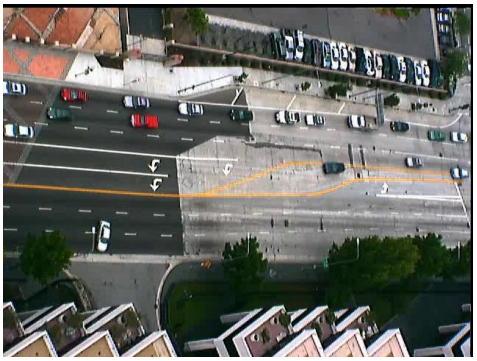
(b) incPCP sparse estimate.



(c) Frame 100.

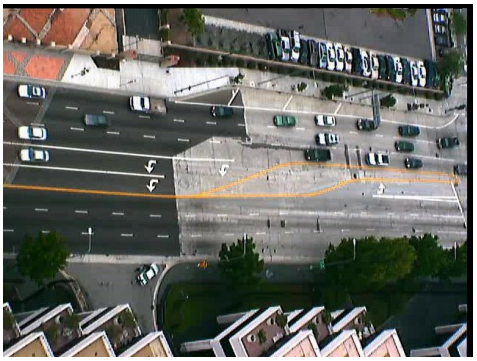


(d) incPCP sparse estimate.



(e) Frame 300.

(f) incPCP sparse estimate.



(g) Frame 366.

(h) incPCP sparse estimate.

Figure 1.10: Synthetically jittered and sparse estimates via the incPCP_TI algorithm for the V640 (640×480 pixel) test video.

use any MEX interface, it comes in two flavors (i) one that uses standard single-thread Matlab code and (ii) one that takes advantage of (mainly linear algebra, FFT and convolution) GPU-enabled Matlab functions.

In order to assess the performance of the incPCP_TI algorithm we present two kinds of results: (i) computational performance measured as the time to process a given video, summarized in Table 1.3, and (ii) reconstruction quality (see Figures 1.11 and 1.14) measured by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the "ground truth" sparse video approximation and S_P is the sparse video approximation of either t-GRASTA or incPCP_TI algorithms, and N is the number of pixels per frame, which is same value for either the grayscale or color case.

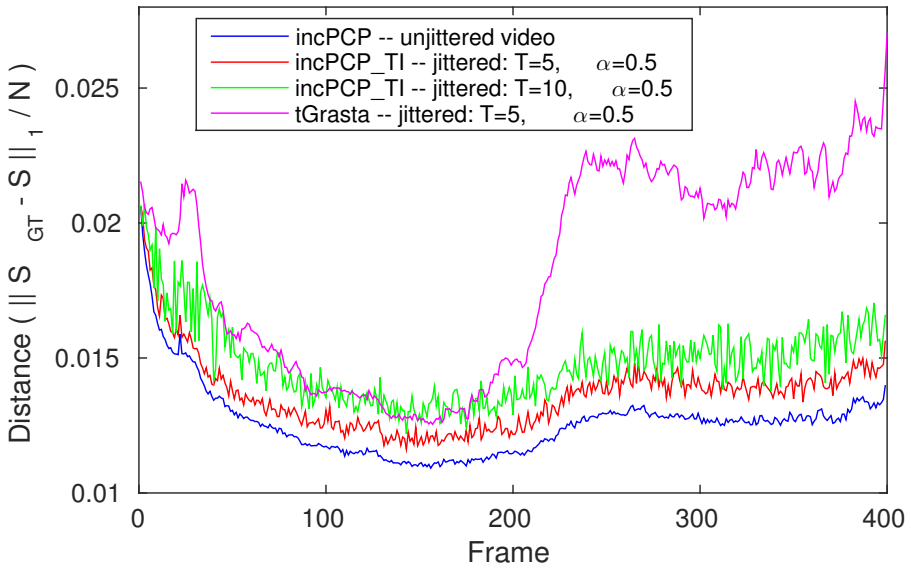
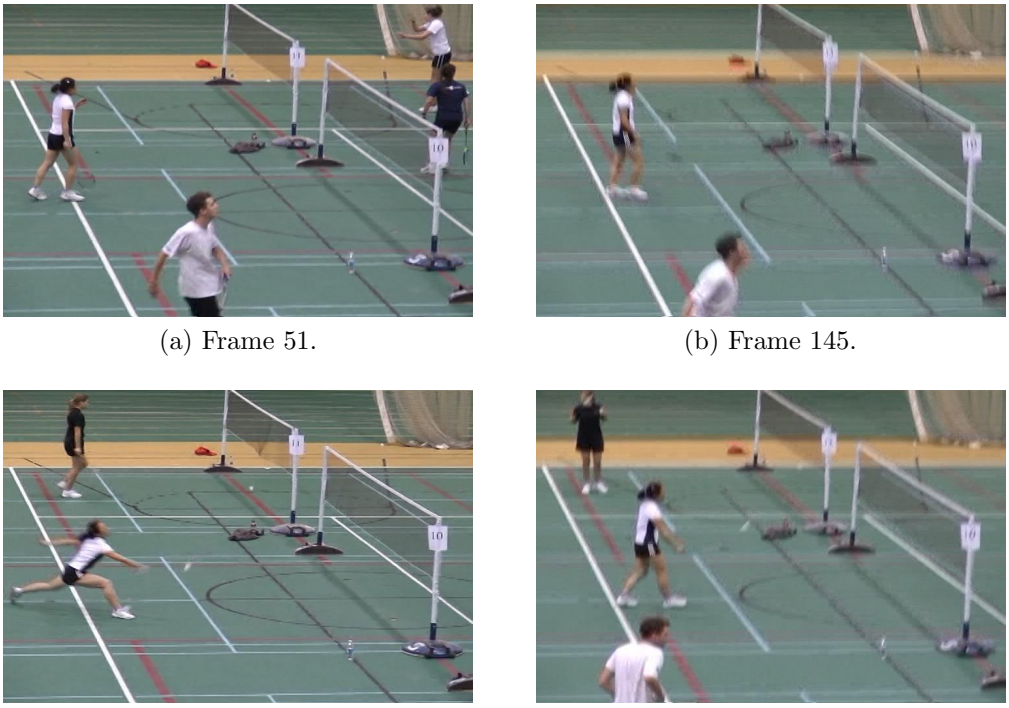


Figure 1.11: Sparse approximation (V640) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the ground-truth computed via the (batch) iALM algorithm (20 outer loops) and S_P is the (i) sparse component computed via incPCP_TI method (red, green) and t-Grasta (magenta) for different synthetic jitter conditions or (ii) the sparse component for the unjittered case (blue), computed via [65], provided as a baseline.

Without taking into account the batch initializations for t-GRASTA, the average processing time per frame listed in Table 1.3 shows that the incPCP_TI algorithm (Matlab-only version) is 2.5 times faster than t-GRASTA, when processing grayscale videos. Furthermore, when we consider the initialization time needed by t-GRASTA (about 100 times the time needed to process one frame), the incPCP_TI algorithm looks even more computational appealing. Furthermore, the GPU-enabled incPCP_TI implementation is between 1.3 $\sim$ 3.3 times faster than the incPCP_TI Matlab-only version; this the performance improve-

ment is specially noticeable for the V640, V720 and V1920 test videos.

Reconstruction quality results are presented for the V640 (synthetically jittered) and V729 (real jitter); for the other two video sets, results are similar and thus omitted. For test video V640, a proxy ground-truth was used: the sparse video approximation computed via the (batch) iALM algorithm [4] with 20 outer loops, whereas for the test video V720, the ground-truth is given (in the form of manually segmented moving objects) from frame 800 onwards.



(a) Frame 51. (b) Frame 145.

(c) Frame 855. (d) Frame 1002.

Figure 1.12: Selected frames from video V720, where real jitter can be observed. Moreover, several frames are blurred, as shown in sub-figure (b), which severely affects the quality of sparse estimates (see Figure 1.13).

In figure 1.11 the V640 reconstruction quality results show that the incPCP_TI algorithm has better properties than t-GRASTA: the red and magenta lines represent the reconstruction quality for the incPCP_TI and t-GRASTA algorithms respectively, applied to the same (synthetically jittered) video with random uniformly distributed translations (T= 5 pixel) and rotations ($\alpha = 0.5$ degrees). incPCP_TI has slightly better quantitative results than t-GRASTA and both results follow more or less the same tendency; however, a clear difference is observed from frame 200 onwards; this is due to the characteristics of V640 and t-GRASTA's background adaptation capabilities (which are similar to

GRASTA's, see Sections 1.3.1 as well as computational results in Section 1.3.3): (i) there are only a few moving objects in V640 from frame 1 up to (more or less) 200, afterwards a large number of moving objects starts to appear and thus this can be interpreted as a change in the background properties, as can be seen in Figure 1.10; (ii) t-GRASTA tries to adapt to this new environment, however it is known that t-GRASTA has a delay for this type of situation, resulting in a large gap between the reconstruction quality for incPCP_TI and t-GRASTA from frame 200 onwards.

We also note that in Figure 1.11 the incPCP_TI reconstruction quality for jitter T= 10 pixel, $\alpha = 0.5$ degrees is also included, along with the reconstruction quality of the incPCP algorithm (see Section 1.3.3) applied to the unjittered video, which is included as baseline.

Likewise, in Figure 1.14 the V720 reconstruction quality results also show that the incPCP_TI has an edge over t-GRASTA: this is specifically shown in Figure 1.14(a) for the grayscale case; moreover in Figures 1.12 and 1.13 we can observed four selected frames from V720 as well as their corresponding sparse estimates for t-GRASTA and incPCP_TI respectively. It is worth noting that in Figure 1.12 the real jitter that affects V720 is observable and that 1.12(b) is clearly blurred; such blurred frames are not uncommon throughout V720, and does affect the sparse estimates correspondingly, as shown in Figures 1.13(c) and 1.13(d). Finally, for the sake of completeness, in Figure1.14(b) the reconstruction quality metric is depicted for color case. Finally, this section is closed by stating that the subjective quality of incPCP_TI's sparse approximation is good at capturing the moving objects (large and small), however some artifacts are also observable, specially when real jitter is involved.

1.6 Conclusions

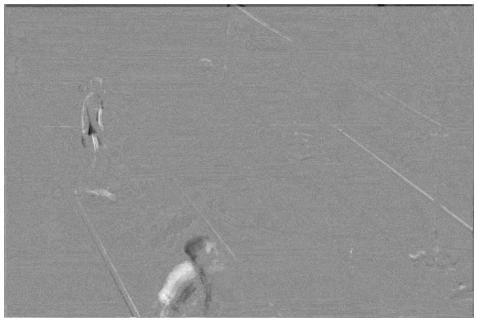
The initial contribution of this chapter is the development of a fully-incremental[5] algorithm for solving a variant of the RPCA problem. It has an extremely low memory footprint, and a computational complexity that allows real-time processing, even when implemented purely in Matlab. Computational experiments indicate that this algorithm provides similar performance to the partially-incremental algorithms while being substantially faster.

The second topic considered is the development of variants of the RPCA problem that are capable of dealing with video sequences with non-stationary backgrounds due to the motion of the camera capturing the scene. A subspace model closely related to that of the Sparse Subspace clustering method is shown to be effective when the background motion is suitable constrained, e.g. moving very slowly, or panning back and forth across the same scene, but does not perform well for more realistic types of motion. An extension of this model using convolutional sparse representations is able to model a much more rapidly

[5]We describe it as fully-incremental to distinguish it from other algorithms that are described as incremental, but that have a batch initialization procedure that requires processing a large block of frames, making them substantially less suitable for real-time application.

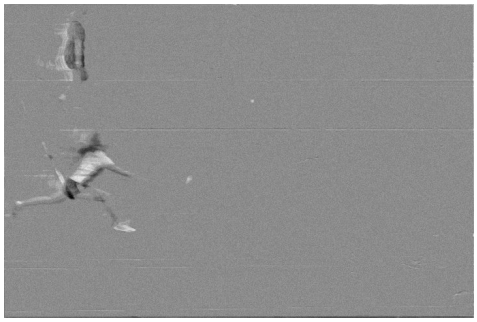
(a) t-GRASTA sparse estimate of frame 51.

(b) incPCP_TI sparse estimate of frame 51.



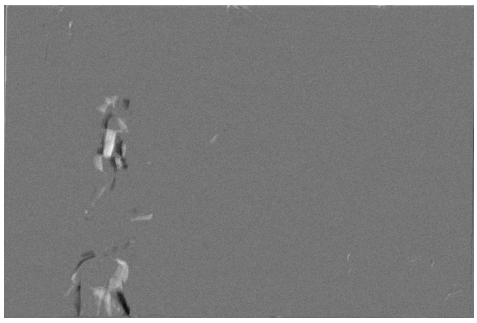
(c) t-GRASTA sparse estimate of frame 145.

(d) incPCP_TI sparse estimate of frame 145.



(e) t-GRASTA sparse estimate of frame 855.

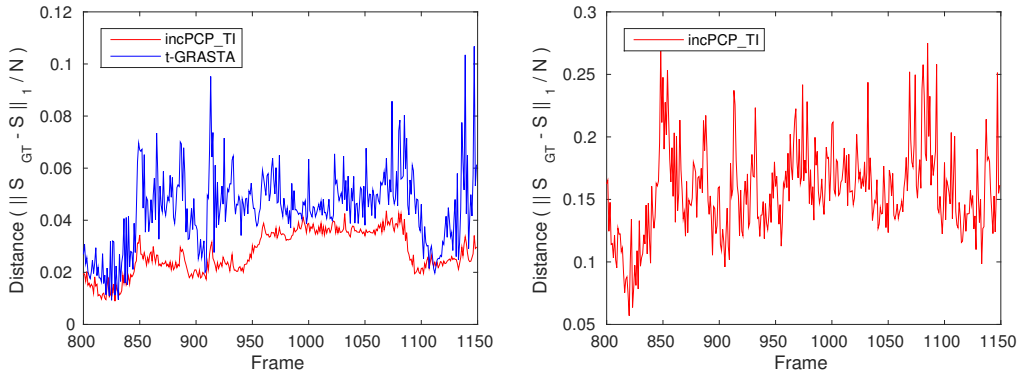
(f) incPCP_TI sparse estimate of frame 855.



(g) t-GRASTA sparse estimate of frame 1002.

(h) incPCP_TI sparse estimate of frame 1002.

Figure 1.13: Sparse estimates via the t-GRASTA and incPCP_TI algorithms for the V720 (720×480 pixel) test video. Note that t-GRASTA can only process grayscale videos.



(a) Grayscale case. (b) Color case.

Figure 1.14: Sparse approximation (V720) frame distance measure by $\frac{\|S_{GT}-S_P\|_1}{N}$ where S_{GT} is the manually segmented ground-truth and S_P is the sparse approximation computed via (a) grayscale (blue) t-GRASTA and (red) incPCP_TI methods, or (b) color (red) incPCP_TI method, and N is the number of pixels per frame (used as a normalization factor).

translating background, but is computationally expensive and unable to directly model transformations other than translation.

The final contribution is a synthesis of the incremental algorithm with a method that is inspired by the convolutional sparse representation approach, but is vastly less computationally expensive. This algorithm is able to process video with both translating and rotating background while operating at near real-time speed. Computational experiments indicate that it is both faster and delivers a superior sparse representation to t-GRASTA, the only competing algorithm of which we are aware.

Of the possible topics for future work, two stand out has having potential for significant impact. First, the rank-1 update and FFT-based image rotation procedures, which are heavily used by the final form of the presented rigid transformation invariant RPCA algorithm, are easily parallelizable and thus a C/C++ CUDA implementation of the proposed algorithm should be explored. Due to the low memory footprint and overall computational complexity properties of our algorithm, such a parallel implementation could be deployed in mobile CUDA-aware processors such the Tegra K1 or X1 for in-situ processing. Second, it is desirable to model more realistic types of motions, such those observed when a video is acquired by cameras mounted in UAVs or self-driving cars. The framework proposed by (1.42) does allow more general transformations, but substantial work is required to develop computationally efficient methods that would support them.

Bibliography

[1] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, pp. 11:1–11:37, June 2011.

[2] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "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.

[3] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation.," in *Intl. Conf. Mach. Learn. (ICML)*, pp. 663–670, 2010.

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

[5] J. Bezdek and R. Hathaway, "Some notes on alternating optimization," in *Advances in Soft Computing – AFSS 2002*, vol. 2275 of *Lecture Notes in Computer Science*, pp. 288–300, Springer Berlin Heidelberg, 2002.

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

[7] T. Zhou and D. Tao, "Godec: Randomized low-rank & sparse matrix decomposition in noisy case," in *ACM Int'l. Conf. on Machine Learning*, ICML '11, pp. 33–40, June 2011.

[8] L. Xiong, X. Chen, and J. Schneider, "Direct robust matrix factorization for anomaly detection," in *IEEE Int'l. Conf. on Data Mining*, pp. 844–853, Dec. 2011.

[9] R. M. Larsen, "PROPACK." Functions for computing the singular value decomposition of large and sparse or structured matrices, available from `http://sun.stanford.edu/~rmunk/PROPACK`.

[10] C. Qiu and N. Vaswani, "Support predicted modified-cs for recursive robust principal components pursuit," in *IEEE International Symposium on Information Theory (ISIT)*, 2011.

[11] 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)*, pp. 1568–1575, June 2012.

[12] F. Seidel, C. Hage, and M. Kleinsteuber, "pROST: a smoothed lp-norm robust online subspace tracking method for background subtraction in video," *Machine Vision and Applications*, vol. 25, no. 5, pp. 1227–1240, 2014.

[13] J. Xu, V. K. Ithapu, L. Mukherjee, J. M. Rehg, and V. Singh, "Gosus: Grassmannian online subspace updates with structured-sparsity," in *IEEE Int'l, Conf. on Computer Vision (ICCV)*, pp. 3376–3383, Dec. 2013.

[14] C. Qiu and N. Vaswani, "Automated recursive projected cs (ReProCS) for real-time video layering," in *Int'l Recognition (CVPR)*, 2012.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, jan 2011.

[16] Y. Chahlaoui, K. Gallivan, and P. Van Dooren, "Computational information retrieval," in *In Computational Information Retrieval*, ch. An Incremental Method for Computing Dominant Singular Spaces, pp. 53–62, SIAM, 2001.

[17] C. Baker, K. Gallivan, and P. V. Dooren, "Low-rank incremental methods for computing dominant singular subspaces," *Linear Algebra and its Applications*, vol. 436, no. 8, pp. 2866 – 2888, 2012.

[18] 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.

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

[20] "Video sequences for foreground object detection from complex background." available from `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`.

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

[22] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Computer Vision and Pattern Recognition (CVPR)*, pp. 1937–1944, IEEE, 2011.

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

[24] C. Qiu and N. Vaswani, "ReProCS code." `http://home.engineering.iastate.edu/~chenlu/ReProCS/ReProCS_main.html`.

[25] J. He, L. Balzano, and A. Szlam, "GRASTA code." `https://sites.google.com/site/hejunzz/grasta`.

[26] J. Xu, V. K. Ithapu, L. Mukherjee, J. M. Rehg, and V. Singh, "GOSUS code." `http://pages.cs.wisc.edu/~jiaxu/projects/gosus/`.

[27] A. Vacavant, T. Chateau, and A. W. L. Lequièvre, "A benchmark dataset for outdoor foreground/background extraction," in *Computer Vision - ACCV 2012 Workshops*, vol. 7728 of *Lecture Notes in Computer Science*, pp. 291–300, Springer Berlin Heidelberg, 2013.

[28] 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.

[29] G. Peyré, "Manifold models for signals and images," *Comp. Vis. Image Understand.*, vol. 113, no. 2, pp. 249–260, 2009.

[30] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pp. 2790–2797, June 2009.

[31] E. Elhamifar and R. Vidal, "Sparsity in unions of subspaces for classification and clustering of high-dimensional data," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1085–1089, 2011.

[32] E. Elhamifar and R. Vidal, "Structured sparse recovery via convex optimization," Tech. Rep. arXiv:1104.0654v2, arXiv, 2011.

[33] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[34] E. L. Dyer, A. C. Sankaranarayanan, and R. G. Baraniuk, "Greedy feature selection for subspace clustering," *Journal of Machine Learning Research*, April 2012. Submitted.

[35] M. Soltanolkotabi and E. J. Candès, "A geometric analysis of subspace clustering with outliers," *CoRR*, vol. abs/1112.4258, 2011.

[36] B. Nasihatkon and R. Hartley, "Graph connectivity in sparse subspace clustering," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2137–2144, June 2011.

[37] B. Wohlberg, R. Chartrand, and J. Theiler, "Local principal component pursuit for nonlinear datasets," in *Proc. IEEE Int. Conf. Acoust. Speech Sig. Proc. (ICASSP)*, pp. 3925–3928, Mar. 2012.

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

[39] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[40] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. and Pat. Recog. (CVPR)*, pp. 2528–2535, June 2010.

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

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

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

[44] T. Blumensath and M. E. Davies, "On shift-invariant sparse coding," in *Independent Component Analysis and Blind Signal Separation*, vol. 3195 of *Lecture Notes in Computer Science*, pp. 1205–1212, 2004.

[45] Y.-k. Tomokusa, M. Nakashizuka, and Y. Iiguni, "Sparse image representations with shift and rotation invariance constraints," in *Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, pp. 256–259, Jan. 2009.

[46] M. Mørup and M. N. Schmidt, "Transformation invariant sparse coding," in *IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, pp. 1–6, 2011.

[47] Q. Barthélemy, A. Larue, A. Mayoue, D. Mercier, and J. I. Mars, "Shift & 2d rotation invariant sparse coding for multivariate signals," *IEEE Trans. Signal Process.*, vol. 60, pp. 1597–1611, Apr. 2012.

[48] B. Wohlberg, "Efficient algorithms for convolutional sparse representations." Manuscript currently under review, 2015.

[49] 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 Transactions on Pattern Analysis & Machine Intelligence*, vol. 34, no. 11, pp. 2233 – 2246, 2012.

[50] 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.

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

[52] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Proceedings of the Neural Information Processing Systems (NIPS)*, pp. 801–808, 2007.

[53] M. Unser, P. Thevenaz, and L. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Transactions on Image Proc.*, vol. 4, pp. 1371–1381, Oct. 1995.

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

[55] K. Mohan and M. Fazel, "Iterative reweighted algorithms for matrix rank minimization," *Journal of Machine Learning Research*, vol. 13, pp. 3441–3473, Nov. 2012.

[56] Y. Keller, A. Averbuch, and M. Israeli, "Pseudopolar-based estimation of large translations, rotations, and scalings in images," *IEEE Transactions on Image Proc.*, vol. 14, pp. 12–22, Jan 2005.

[57] G. Vaillant, C. Prieto, C. Kolbitsch, G. Penney, and T. Schaeffter, "Retrospective rigid motion correction in k-space for segmented radial MRI," *IEEE Transactions on Medical Imaging*, vol. 33, pp. 1–10, Jan. 2014.

[58] 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.

[59] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[60] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, pp. 1235–1256, Nov. 2009.

[61] M. Fazel, H. Hindi, and S. Boyd, "Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2156–2162 vol.3, June 2003.

[62] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "Cdnet 2014: An expanded change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pp. 393–400, June 2014.

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

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

[65] 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), pp. 3414–3416, Oct. 2014.