Practical ReProCS for Separating Sparse and Low-dimensional Signal Sequences from their Sum – Part 2

Han Guo, Namrata Vaswani
Dept. of Electrical and Computer Engineering
Iowa State University, Ames, Iowa, USA
Email: {hanguo, namrata}@iastate.edu

Chenlu Qiu
Traffic Management Research Institute
Ministry of Public Security, Beijing, China
Email: echotosusan@gmail.com

***Abstract*—In this work, we experimentally evaluate and verify model assumptions for our recently proposed algorithm (practical ReProCS) for recovering a time sequence of sparse vectors S_t and a time sequence of dense vectors L_t from their sum, $M_t := S_t + L_t$, when L_t lies in a slowly changing low-dimensional subspace. A key application where this problem occurs is in video layering where the goal is to separate a video sequence into a slowly changing background sequence and a sparse foreground sequence that consists of one or more moving regions/objects. Practical-ReProCS is the practical analog of its theoretical counterpart that was studied in our recent work.**

***Index Terms*—robust PCA, robust matrix completion, sparse recovery**

I. Introduction

The goal is to recover a time sequence of sparse vectors S_t and a time sequence of dense vectors L_t from their sum, $M_t := S_t + L_t$, when L_t lies in a slowly changing low-dimensional subspace. This problem can be interpreted either as online / recursive sparse recovery from potentially large but structured noise or as online / recursive robust PCA. This problem occurs in separating a slowly changing background from moving foreground objects/regions (sparse image) [2], [3] for applications like video surveillance. By letting M_t be the image arranged as a 1D vector, L_t be the (mean subtracted) background image and defining S_t as the the foreground-background intensity difference on the foreground support and zero everywhere else, this becomes a problem of the above form.

Existing Work. Most existing algorithms are batch methods, e.g. [2], [3], [4], [5], [6]. Online/recursive methods include older work [7], recent works [8], [6] and partial theoretical results [9], [10], [11].

Contributions. In part 1 of this work [12], we designed a practical modification of the theoretical ReProCS algorithm studied in [9], [13] and showed comparisons for simulated videos. In this paper, (a) we show that the assumptions used by it are valid for real data, and (b) we show extensive performance comparisons for real videos. As we will show, for large sized and correlated foreground supports and similar foreground-background intensities (small S_t), ReProCS significantly outperforms batch methods like RSL [2] and PCP [3], [4] and recursive methods - GRASTA [8]. As shown in [1], [7], [6] do not work at all.

Notation. For a set T, we use $|T|$ to denote its cardinality. The notation [.] denotes an empty matrix. A matrix P is a *basis matrix* if $P'P = I$. The *notation $Q = basis(range(M))$, or $Q = basis(M)$ for short,* means that Q satisfies $Q'Q = I$ and $\text{range}(Q) = \text{range}(M)$. The $b\%$ *left singular values' set* of a matrix M is the smallest set of indices of its singular values that contains at least $b\%$ of the total singular values' energy. The corresponding matrix of left singular vectors, U_T, is referred to as the $b\%$ *left singular vectors' matrix.* The notation $Q = \text{approx-basis}(M, b\%)$ means that Q is the $b\%$ left singular vectors' matrix for M. The notation $Q = \text{approx-basis}(M, r)$ means that Q contains the left singular vectors of M corresponding to its r largest singular values and the notation $Q = \text{approx-basis}(M, , \theta)$ means that Q contains all left singular vectors of M with singular values above θ.

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

II. Problem Definition and Assumptions

The measurement vector at time t, M_t, is an n length vector satisfying $M_t := S_t + L_t$. We let T_t denote the support set of S_t.We assume that S_t and L_t satisfy the assumptions given below. Suppose that an initial training sequence which does not contain the sparse components is available, i.e. we are given $\mathcal{M}_{\text{train}} = [M_t; 1 \le t \le t_{\text{train}}]$ with $M_t = L_t$. At each $t > t_{\text{train}}$, the goal is to recursively estimate S_t and L_t and the subspace in which L_t lies.

A. Low-dimensionality and slow subspace change

We assume that for τ large enough, any τ length subsequence of the L_t's lies in a subspace of $\mathbf{R}^n$ of dimension less than $\min(\tau, n)$, and usually much less than $\min(\tau, n)$. In other words, for τ large enough, $\max_t \text{rank}([L_{t-\tau+1}, \ldots L_t]) \ll \min(\tau, n)$. Also, this subspace is either fixed or changes slowly over time. One way to model this is as follows [9]. Let $L_t = P_t a_t$ where P_t is an $n \times r_t$ *basis matrix* with $r_t \ll n$ that is piecewise constant with time, i.e. $P_t = P_{(j)}$ for all $t \in [t_j, t_{j+1})$ and $P_{(j)}$ changes as

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

where $P_{(j),\text{new}}$ and $P_{(j),\text{old}}$ are basis matrices of size $n \times c_{j,\text{new}}$ and $n \times c_{j,\text{old}}$ respectively with $P'_{(j),\text{new}} P_{(j-1)} = 0$ and R_j is a rotation matrix. Moreover, (a) $0 \le \sum_{i=1}^{j} (c_{i,\text{new}} - c_{i,\text{old}}) \le c_{\text{dif}}$; (b) $0 \le c_{j,\text{new}} \le c_{\max} < r_0$; (c) $(t_{j+1} - t_j) \gg r_0 + c_{\text{dif}}$; and (d) there are a total of J change times with $J \ll (n - r_0 - c_{\text{dif}})/c_{\max}$.

By slow subspace change, we mean that: for $t \in [t_j, t_{j+1})$, $\|(I - P_{(j-1)} P'_{(j-1)}) L_t\|_2$ is initially small and increases gradually. In particular, we assume that, for $t \in [t_j, t_j + \alpha)$,

$$\|(I - P_{(j-1)} P'_{(j-1)}) L_t\|_2 \le \gamma_{\text{new}} \ll \min(\|L_t\|_2, \|S_t\|_2)$$

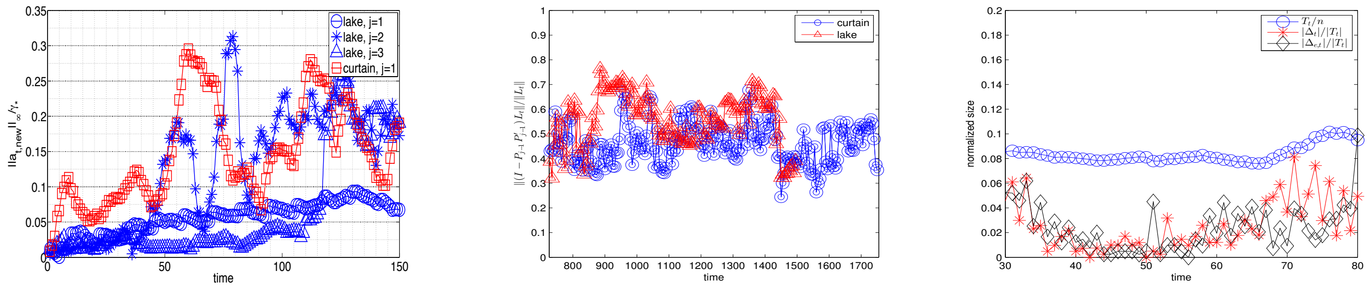
and increases gradually after $t_j + \alpha$.

The above piecewise constant subspace change model is a simplified model for what typically happens in practice. In most cases, P_t changes a little at each t in such a way that the low-dimensional assumption approximately holds. Since background images typically change only a little over time (except in case of a camera viewpoint change or a scene change), it is valid to model the mean-subtracted background image sequence as lying in a slowly changing low-dimensional subspace. We verify this assumption in Sec III.

B. Denseness

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

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



(a) slow subspace change - 1 (b) slow subspace change - 2 (c) support size, support change

Fig. 1: Assumptions' verification.

for a κ_* significantly smaller than one. Here

$$\kappa_s(B) = \kappa_s(\text{range}(B)) := \max_{|T| \le s} \|I_T{}' \text{basis}(B)\|_2 \qquad (1)$$

is the denseness coefficient for any vector or matrix B [9]. Moreover, a similar assumption holds for $P_{j,\text{new}}$ with a tighter bound: $\kappa_{2s}(P_{j,\text{new}}) \le \kappa_{\text{new}} < \kappa_*$. By [9, Lemma 2], a small $\kappa_{2s}(P_j)$ means that the restricted isometry constant (RIC) [14] of the matrix $(I - P_j P_j')$ is small.

Very often, the background images primarily change due to lighting changes (in case of indoor sequences) or due to moving waters or moving leaves (in case of many outdoor sequences) [3], [9]. All of these result in global changes and hence it is valid to assume that the subspace spanned by the background image sequences is dense. Denseness as defined above cannot be verified in practice because computing $\kappa_s(M)$ has exponential complexity.

C. *Support size, support change of S_t*

We assume two things. (1) We assume that

$$|T_t| + \min(|T_t|, |\Delta_t| + |\Delta_{e,t}|) \le s + s_\Delta \text{ with } s_\Delta \ll s.$$

Here $\Delta_t := T_t \setminus T_{t-1}$ and $\Delta_{e,t} := T_{t-1} \setminus T_t$ denote the additions to and removals from the support at time t. In particular, this implies that we either need $|T_t| \le s$ and $|\Delta_t| + |\Delta_{e,t}| \le s_\Delta$ (S_t is sparse with support size at most s, and its support changes slowly) or, in cases when the change $|\Delta_t| + |\Delta_{e,t}|$ is large, we need $|T_t| \le 0.5(s + s_\Delta)$ (need a tighter bound on the support size).

(2) We assume that there is *some* support change every few frames, i.e. at least once every h frames, $|\Delta_t| > s_{\Delta,\min} > 0$. Practically, this is needed to ensure that at least some of the background behind the foreground is visible so that the changes to the background subspace can be estimated.

In the video application, foreground images typically consist of one or more moving objects/people/regions and hence are sparse. Also, typically the objects are not static, i.e. there is some support change at least every few frames. On the other hand, since the objects usually do not move very fast, slow support change is also valid most of the time. The time when the support change is almost comparable to the support size is usually when the object is entering or leaving the image, but these are the exactly the times when the object's support size is itself small (being smaller than $0.5(s + s_\Delta)$ is a valid). We show some verification of these assumptions in Sec III.

III. Model Verification

We verify low-dimensionality, slow subspace change, small support size and support change assumptions here.

A. *Low-dimensionality and slow subspace change*

We used two background image sequence datasets. The first was a video of lake water motion. For this sequence, $n = 6480$ and the number of images were 1500. The second was an indoor video of window curtains moving due to the wind. There was also some lighting variation. For this sequence, the image size was $n = 5120$ and the number of background-only images were 1755. The latter part of this sequence also contains a foreground (various persons coming in, writing on the board and leaving).

First note that any given background image sequence will never be exactly low-dimensional, but only be approximately so. Secondly, in practical data, the subspace does not just change as simply as in the model of Sec II-A. Typically there are some changes to the subspace at every time t. Moreover, with just one training sequence of a given type, it is not possible to estimate the covariance matrix of L_t at each t and thus one cannot detect the subspace change times. The only thing one can do is to assume that there may be a change every τ frames, and that during these τ frames the L_t's are stationary and ergodic; estimate the covariance matrix of L_t for this period using a time average. We verified the assumption using two different methods: with and without "low-rankifing" the data.

Let the data matrix (with its empirical mean subtracted) be denoted by $\mathcal{L}_{\text{full}}$. $\mathcal{L}_{\text{full}}$ is a $n \times t_{\max}$ matrix. We first "low-rankified" this dataset by computing $P_{\text{full}} = \text{approx-basis}(\mathcal{L}_{\text{full}}, 90\%)$ and projecting $\mathcal{L}_{full}$ along P_{full} to get $\mathcal{L} := P_{\text{full}} P_{\text{full}}' \mathcal{L}_{\text{full}}$. Thus $\mathcal{L}$ is a $n \times t_{\max}$ matrix with $\text{rank}(\mathcal{L}) < \min(n, t_{\max})$. For the curtains dataset, we observed that 90% of the energy is contained in only 34 directions, i.e. $\text{rank}(\mathcal{L}) = 34$ while for the lake dataset, $\text{rank}(\mathcal{L}) = 14$. This indicates that both datasets are approximately low rank since $38 \ll \min(n, t_{\max}) = 1500$.

We then took the first set of τ frames of the low-rank matrix, $\mathcal{L}_{1:\tau} := [L_1, L_2 \dots L_\tau]$ and computed $P_0 = \text{approx-basis}(\mathcal{L}_{1:\tau}, 99.99\%)$. Also, we stored the square of the lowest retained singular value and called it σ^-. It is assumed that all directions with singular values below σ^- are due to noise. For the next set of τ frames, $\mathcal{L}_{\tau+1:2\tau} := [L_{\tau+1}, L_{\tau+2}, \dots L_{2\tau}]$, we computed $P_{1,\text{new}} := \text{approx-basis}((I - P_0 P_0')\mathcal{L}_{\tau+1:2\tau}, , \sigma^-)$. Then, $P_1 = [P_0, P_{1,\text{new}}]$. For the third set of τ frames, we repeated the above procedure, but with P_0 replaced by P_1 and obtained P_2. We used $\tau = 150$ for both the datasets. In each case, we computed $r_0 := \text{rank}(P_0)$, and $c_{\max} := \max_j \text{rank}(P_{j,\text{new}})$. For each batch of d frames, we also computed $a_{t,\text{new}} := P_{j,\text{new}}' L_t$, $a_{t,*} := P_{j-1}' L_t$ and $\gamma_* := \max_t \|a_t\|_\infty$. We got $c_{\max} = 3$ and $r_0 = 8$ for the lake sequence and $c_{\max} = 5$ and $r_0 = 29$ for the curtain sequence. Thus the ratio $c_{\max}/r_0$ is sufficiently small in both cases. In Fig. 1(a), we plot $\|a_{t,\text{new}}\|_\infty / \gamma_*$ for one 150-frame period of the curtain sequence and for three 150-frame change periods of the lake sequence. As can be seen, this quantity is initially small and increases gradually.

A second way to test subspace change is to do it without low-rankifying the data first. We proceed as follows. We let $t_0 = 0$ and $t_j = t_0 + j\tau$ with $\tau = 725$. We computed P_j as $P_j = \text{approx-basis}([L_{t_j}, \dots L_{t_{j+1}-1}], 95\%)$. We observed that $\text{rank}(P_j) \le$

38 for curtain sequence, while $\text{rank}(P_j) \leq 33$ for lake sequence. To test for slow subspace change, in Fig. 1(b), we plot $\|(I - P_{j-1}P_{j-1}')L_t\|_2/\|L_t\|_2$ when $t \in [t_j, t_{j+1})$. Notice that, after every change time ($t_j = 725, 1450$), this quantity is initially small for some time and increases gradually.

B. Support size, support change of S_t

For real video sequences, it is not possible to get the true foreground support. Thus, we do the following. For the curtain sequence, we select a part of the sequence in which the person is wearing a black shirt (against a white curtains' background). For this part, ReProCS returns a very accurate estimate of T_t, and we use this estimated support as a proxy for the true support T_t. We plot the support size normalized by the image size $|T_t|/n$, and we plot the number of additions and removals normalized by the support size, i.e. $|\Delta_t|/|T_t|$ and $|\Delta_{e,t}|/|T_t|$ in Fig 1(c). Notice from the figure that the support size is at most 10.2% of the image size. Notice also that at least at every 3 frames, there is at least a 1% support change. Thus there is some support change every few frames, thus exposing the part of the background behind the foreground. Finally notice that the maximum number of support changes is only 9.9% of the support size, i.e. slow support change holds for this piece.

IV. Practical ReProCS

The algorithm is summarized in Algorithm 1. It is essentially the same as that in [1]. We have added two new heuristics to help detect the absence of a sparse vector (e.g. when the foreground object moves out of the scene) and to deal with very static backgrounds.

V. Real Video Experiments

Code, data and videos for experiments discussed below and more experiments are available at [15]. We show comparisons on three real video sequences taken from [16]. The first is the curtain sequence described earlier. For $t > 1755$, in the foreground, a person with a black shirt walks in, writes on the board and then walk out, then a second person with a white shirt does the same and then a third person with a white shirt does the same. This video is challenging because (i) the white shirt color and the curtains' color is quite similar, making the corresponding S_t small in magnitude; and (ii) because the foreground person moves slowly (correlated support change). As can be seen from Fig. 2, ReProCS's performance is significantly better than that of the other algorithms. This is most easily seen from the recovered background images. One or more frames of the background recovered by PCP, RSL [2] and GRASTA [8] contains the person, while none of the ReProCS ones does.

The second sequence consists of a person entering an office, leaving the office (absent for most frames) and re-entering the office. The person switches the light on/off in the office, making the background suddenly changes. Initially, ReProCS and GRASTA have nearly the same performance. But after a long time span, ReProCS and RSL outperform GRASTA. As can be seen in the second row of Fig. 3, when the person is absent, ReProCS is able to determine no fg is shown due to fg detection step. In the third sequence, a person is walking around with a background of water surface. As can be seen in Fig. 4, the water surface changes globally. ReProCS and GRASTA are able to recover the foreground person for most frames, but ReProCS returns a more complete shape. Moreover, ReProCS recovers a better background than GRASTA.

Speed. GRASTA is the fastest even though its performance is much worse. ReProCS is the second fastest. PCP and RSL are slower because they jointly process the entire image sequence. For the curtain sequence, GRASTA took 45 seconds, ReProCS took 164 seconds, PCP took 1080 and RSL took 644 seconds.

Algorithm 1 Practical ReProCS

```
Input: $M_t$; Output: $\hat{T}_t$, $\hat{S}_t$, $\hat{L}_t$; Parameters: $\alpha = 20, K_{\min} = 3, K_{\max} = 10$.
Initialization: Compute $\hat{P}_0^1 \leftarrow$ approx-basis($[M_1, \ldots M_{t_{\text{train}}}], 95\%$) and compute $\hat{P}_0^2 \leftarrow$ approx-basis($[M_1, \ldots M_{t_{\text{train}}}], t_{\text{train}}/10$) and set $\hat{P}_0$ as the matrix with smaller number of columns.
Set $\hat{r} \leftarrow \text{rank}(\hat{P}_0)$, $\hat{P}_{t_{\text{train}}} \leftarrow \hat{P}_0$, $\hat{\sigma}_{\min} \leftarrow \hat{r}$-largest singular value obtained from computing SVD of $[M_1, \ldots M_{t_{\text{train}}}]$, $\gamma \leftarrow 0.2$.
Set $\hat{T}_t \leftarrow [.]$. For $t > t_{\text{train}}$ do
  1) Perpendicular Projection
       a) $y_t \leftarrow \Phi_{(t)} M_t$, $\Phi_{(t)} \leftarrow I - \hat{P}_{t-1}\hat{P}_{(t-1)}'$
  2) Sparse Recovery (Recover $S_t$ and $T_t$)
     If $\frac{|\hat{T}_{t-2} \cap \hat{T}_{t-1}|}{|\hat{T}_{t-2}|} < 0.5$
       a) Compute $\hat{S}_{t,\text{cs}}$ by solving $\min_x \|x\|_1$ s.t. $\|y_t - \Phi_{(t)}x\|_2 \leq \xi$
          with $\xi = \|\Phi_{(t)}\hat{L}_{t-1}\|_2$; $\hat{T}_{95\%} \leftarrow \text{Prune}(\hat{S}_{t,\text{cs}}, 95\%)$.
       b) If $|\hat{T}_{t-1}| < 0.1n$ and $|\hat{T}_{95\%}| > \gamma n$, $\hat{T}_t \leftarrow [.]$;
          else $\hat{T}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{cs}}, \omega)$ with $\omega = \sqrt{\|M_t\|^2/n}$
     Else
       a) Compute $\hat{S}_{t,\text{cs}}$ by solving weighted-$\ell_1$
             $\min_x \lambda \|x_{\hat{T}_{t-1}}\|_1 + \|x_{\hat{T}_{t-1}^c}\|_1$ s.t. $\|y_t - \Phi_{(t)}x\|_2 \leq \xi$
          with $\lambda = \frac{|\hat{T}_{t-2} \setminus \hat{T}_{t-1}|}{|\hat{T}_{t-1}|}$ and $\xi = \|\Phi_{(t)}\hat{L}_{t-1}\|_2$; $\hat{T}_{95\%} \leftarrow$ Prune($\hat{S}_{t,\text{cs}}, 95\%$).
       b) If $|\hat{T}_{t-1}| < 0.1n$ and $|\hat{T}_{95\%}| > \gamma n$, $\hat{T}_t \leftarrow [.]$;
          else
          • $\hat{T}_{\text{add}} \leftarrow \text{Prune}(\hat{S}_{t,\text{cs}}, 1.4|\hat{T}_{t-1}|)$.
          • $\hat{S}_{t,\text{add}} \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{T}_{\text{add}})$
          • $\hat{T}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{add}}, \omega)$ with $\omega = \sqrt{\|M_t\|^2/n}$
     Set $\hat{S}_t \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{T}_t)$
  3) Estimate $L_t$: $\hat{L}_t \leftarrow M_t - \hat{S}_t$
  4) Update $\hat{P}_t$: projection PCA
       a) If flag = detect and $\text{mod}(t - \hat{t}_j + 1, \alpha) = 0$, (here $\text{mod}(t, \alpha)$ is the remainder when $t$ is divided by $\alpha$)
            i) compute the SVD of $\frac{1}{\sqrt{\alpha}}(I - \hat{P}_{(j-1)}\hat{P}_{(j-1)}')[\hat{L}_{t-\alpha+1}, \ldots \hat{L}_t]$ and check if any singular values are above $\hat{\sigma}_{\min}$
           ii) if the above number is more than zero then set flag $\leftarrow$ pPCA, increment $j \leftarrow j + 1$, set $\hat{t}_j \leftarrow t - \alpha + 1$, reset $k \leftarrow 1$
          Else $\hat{P}_t \leftarrow \hat{P}_{t-1}$.
       b) If flag = pPCA and $\text{mod}(t - \hat{t}_j + 1, \alpha) = 0$,
            i) compute the SVD of $\frac{1}{\sqrt{\alpha}}(I - \hat{P}_{(j-1)}\hat{P}_{(j-1)}')[\hat{L}_{t-\alpha+1}, \ldots \hat{L}_t]$,
           ii) let $\hat{P}_{j,\text{new},k}$ retain all its left singular vectors with singular values above $\hat{\sigma}_{\min}$ or all $\alpha/3$ top left singular vectors whichever is smaller,
          iii) update $\hat{P}_t \leftarrow [\hat{P}_{(j-1)}\ \hat{P}_{j,\text{new},k}]$, increment $k \leftarrow k + 1$
           iv) If $k \geq K_{\min}$ and $\frac{\|\sum_{t-\alpha+1}^{t}(\hat{P}_{j,\text{new},i-1}\hat{P}_{j,\text{new},i-1}' - \hat{P}_{j,\text{new},i}\hat{P}_{j,\text{new},i}')L_t\|_2}{\|\sum_{t-\alpha+1}^{t}\hat{P}_{j,\text{new},i-1}\hat{P}_{j,\text{new},i-1}'L_t\|_2} < 0.01$
               for $i = k-2, k-1, k$; or $k = K_{\max}$,
               then $K \leftarrow k$, $\hat{P}_{(j)} \leftarrow [\hat{P}_{(j-1)}\ \hat{P}_{j,\text{new},K}]$ and reset flag $\leftarrow$ detect.
          Else $\hat{P}_t \leftarrow \hat{P}_{t-1}$.
```

Definition 4.1: In the algorithm above,

1) $T \leftarrow \text{Thresh}(x, \omega)$ means $T = \{i : |(x)_i| \geq \omega\}$
2) $T \leftarrow \text{Prune}(x, s)$ means T contains the s largest magnitude elements of x
3) $\hat{x} \leftarrow \text{LS}(y, A, T)$ means

$$\hat{x}_T = (A_T)^\dagger y = (A_T{}' A_T)^{-1} A_T{}' y, \ \hat{x}_{T^c} = 0.$$

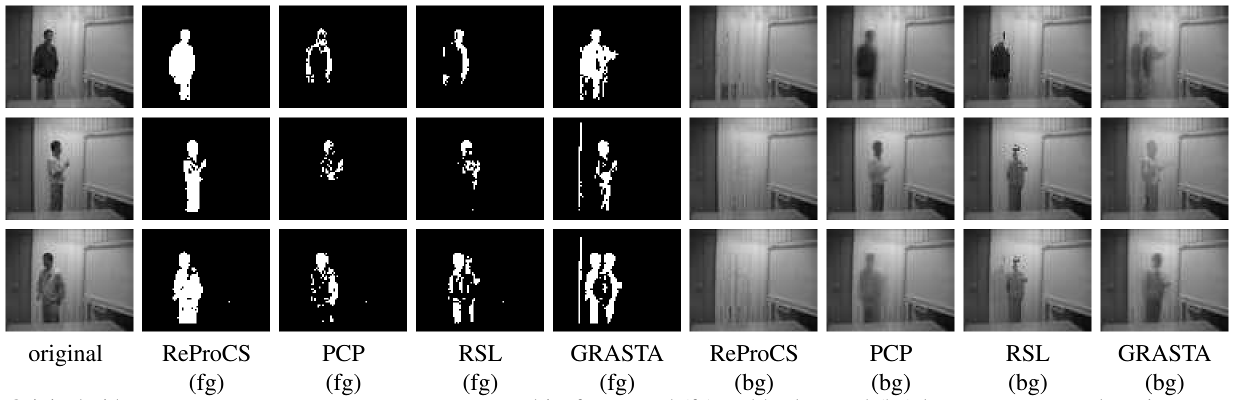


Fig. 2: Original video sequence at $t = t_{\text{train}} + 120, 475, 1148$ and its foreground (fg) and background (bg) layer recovery results using ReProCS and other algorithms. For fg, we only show the fg support in white for ease of display.

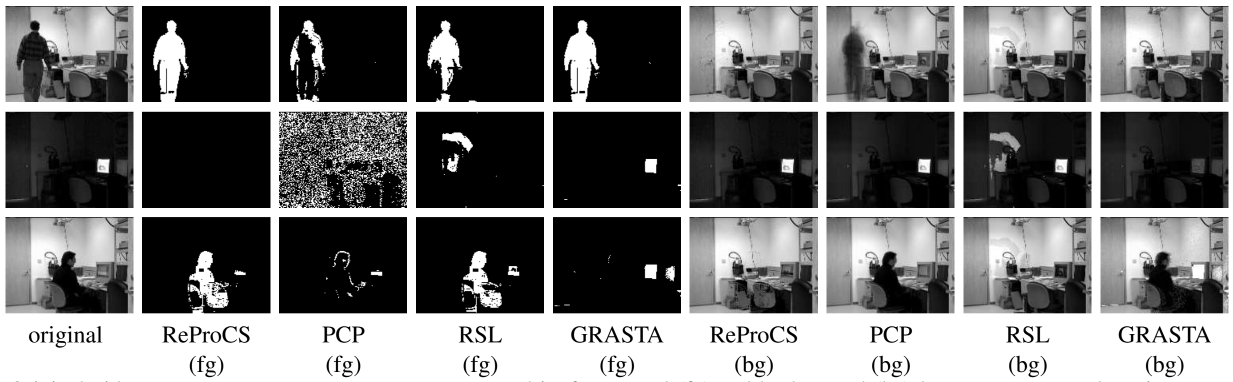


Fig. 3: Original video sequence at $t = t_{\text{train}} + 35, 500, 1300$ and its foreground (fg) and background (bg) layer recovery results using ReProCS and other algorithms. For fg, we only show the fg support in white for ease of display.

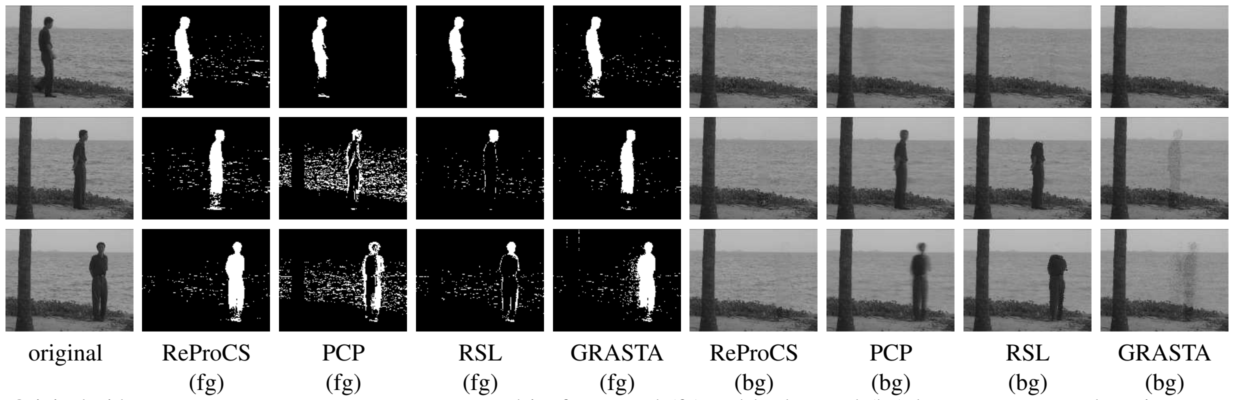


Fig. 4: Original video sequence at $t = t_{\text{train}} + 30, 80, 140$ and its foreground (fg) and background (bg) layer recovery results using ReProCS and other algorithms. For fg, we only show the fg support in white for ease of display.

References

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

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

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

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

[5] H. Xu, C. Caramanis, and S. Sanghavi, "Robust pca via outlier pursuit," *IEEE Tran. on Information Theorey*, vol. 58, no. 5, 2012.

[6] G. Mateos and G. Giannakis, "Robust pca as bilinear decomposition with outlier-sparsity regularization," *IEEE Trans. Sig. Proc.*, Oct 2012.

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

[8] Jun He, Laura Balzano, and Arthur Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. on Comp. Vis. Pat. Rec. (CVPR)*, 2012.

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

[10] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization," in *Adv. Neural Info. Proc. Sys. (NIPS)*, 2013.

[11] J. Feng, H. Xu, S. Mannor, and S. Yan, "Online pca for contaminated data," in *Adv. Neural Info. Proc. Sys. (NIPS)*, 2013.

[12] H. Guo, C. Qiu, and N. Vaswani, "Practical reprocs for separating sparse and low-dimensional signal sequences from their sum - part 1," in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2014.

[13] "Blinded title," in *Double blind conference submission*, 2014.

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

[15] H. Guo, C. Qiu, and N. Vaswani, "http://www.ece.iastate.edu/~hanguo/PracReProCS.html," .

[16] "http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html," .