Incremental Gradient on the Grassmannian for Online Foreground and Background Separation in Subsampled Video

Jun He*
Nanjing University of Information Science and Technology
hejun.zz@gmail.com

Laura Balzano†
University of Wisconsin, Madison
sunbeam@ece.wisc.edu

Arthur Szlam‡
City University of New York
aszlam@ccny.cuny.edu

Abstract

It has recently been shown that only a small number of samples from a low-rank matrix are necessary to reconstruct the entire matrix. We bring this to bear on computer vision problems that utilize low-dimensional subspaces, demonstrating that subsampling can improve computation speed while still allowing for accurate subspace learning. We present GRASTA, Grassmannian Robust Adaptive Subspace Tracking Algorithm, an online algorithm for robust subspace estimation from randomly subsampled data. We consider the specific application of background and foreground separation in video, and we assess GRASTA on separation accuracy and computation time. In one benchmark video example [16], GRASTA achieves a separation rate of 46.3 frames per second, even when run in MATLAB on a personal laptop.

1. Introduction

Low-rank subspaces have long been a powerful tool in data modeling and analysis. Subspace models are of great interest in computer vision for background subtraction [27], object tracking [9, 26], and to represent a single scene under varying illuminations [4, 21]. Applications in communications, medical imaging, and source localization and target tracking in radar and sonar [15] all leverage subspace models to recover the signal of interest and reject noise.

Recent developments at the interface of optimization and statistical signal processing have resulted in the theory of matrix completion [6, 13], which shows that a very large yet low-rank matrix can be reconstructed from a very small number of its entries. For example, a million-by-million rank-50 matrix could be reconstructed from a uniform random sample of only 3% of its entries with probability at least .96[1]. The columns of a low-rank matrix lie in a low-dimensional subspace, so matrix completion amounts to subspace estimation with incomplete data vectors, which has been studied in [2]. In the paper at hand, we turn this around: since the subspace can be identified from incomplete vectors, we can subsample in order to improve on computational efficiency, and we still retain subspace estimation accuracy.

In particular we consider the problem of background and foreground separation in video. Background subtraction is a useful technique for preprocessing image and video data, especially when there are active, moving objects of interest and a relatively more static background. It is has been used as a crucial component in human activity recognition and the analysis of video from surveillance cameras. Computational cost and real-time processing are of utmost importance here: There are an estimated minimum 10,000 surveillance cameras in the city of Chicago and an estimated 500,000 in London [1, 20], and these video datasets are almost entirely used post-hoc for analysis. The goal of preprocessing is to enable technologies that can analyze video data in real-time.

The contribution of this paper is the introduction of GRASTA, or Grassmannian Robust Adaptive Subspace Tracking Algorithm, an online robust subspace tracking algorithm that operates on highly subsampled data. If we model each frame of video as the sum of two components–the relatively static background and a dynamic foreground–then robust subspace estimation (i.e., robust PCA) is a natural approach to separating the two.

Many previous robust PCA algorithms [7] operate in batch[2]. Those batch algorithms which are based on convex optimization can estimate the low-dimensional sub-

*The authors would like to thank the Institute for Pure and Applied Mathematics for bringing them together during the Internet MRA Program.

†Laura Balzano would like to acknowledge the AFOSR and 3M for funding support.

‡ADS thanks the NSF for generous support from DMS-1201666.

[1] See [23] and ignore the constant; we see empirically that this is a good rule of thumb.

[2] For example, [7] used batch size 200 for the "Airport hall" video, see Section 3.

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

space with subsampled data, taking a cue from low-rank matrix completion theory. Online algorithms for robust PCA were presented in [19, 17, 22], but these do not support estimation from subsampled data. GRASTA is an online algorithm that operates one highly subsampled frame at a time, making it especially suitable for video preprocessing. We show that GRASTA achieves high quality separation of background and foreground in video with exceptional training speeds. We believe that GRASTA will make realizable many real-time applications which were previously constrained by computational resources.

1.1. Foreground and background separation

A common assumption for background subtraction is that certain attributes of the objects of interest change more rapidly than the background scene. It is very natural to think that objects with attributes that change rapidly should be analyzed in a different manner from objects changing slowly. However, deciding which attributes are the correct ones and how to measure change can be subtle. For example, one might wish to ignore rapid changes to objects from the sun going behind the clouds, but notice slow changes like a increasing puddle on the kitchen floor from a leaky faucet.

The simplest methods of background subtraction keep a running mean or median of previous frames as a model for the background. While this is efficient and simple to implement, and can be effective in sufficiently regular enviornments, changes in lighting or slow moving objects can cause trouble. Using the median (or mean) gives a 0-dimensional model for the background: we measure the residual of a given frame as a distance to a point, the median. We can make this model more powerful by keeping a d dimensional subspace or affine set with $d > 0$ [21, 25, 28]. Suppose we have T frames and $V = [v_1 \; ... \; v_T]$, where each length-mp vector v_i corresponds to the $m \times p$ pixels of the ith image in the sequence. A d-dimensional subspace model is then a factorization $V \approx LR^T$, where L is an $mp \times d$ basis matrix, and R is a $T \times d$ coefficient matrix. In the following we define $n := mp$ for the vectorized image dimension. It has been noticed experimentally [11] that the vectorized images corresponding to a given object under different lighting conditions approximately lie on a low dimensional subspace. A theoretical justification for this is described in [4].

1.2. Robust PCA

If we wish to learn the background subspace dynamically from the video to be analyzed, classical least squares low-rank approximation is insufficient, precisely because we expect the "foreground" objects to not follow the low-rank model; the pixels corresponding to these objects will not be well approximated by the model. It is well known that even small numbers of gross coordinate corruptions lead to large changes in a least squares low-rank model [12]. On the other hand, there has been a large amount of recent work on low-rank models which are more robust to gross corruptions [25, 8, 7]. These methods lead to good results in the background subtraction problem, but they are too slow to be dynamically trained in real time from video data. We are therefore motivated to bridge the gap between fast/online algorithms and robust algorithms with GRASTA.

1.3. Subspace Tracking with Incomplete Data

Consider a sequence of d-dimensional subspaces $\mathcal{S}_t \subset \mathbb{R}^n$, $d < n$, and a sequence of vectors $v_t \in \mathcal{S}_t$. The object of a subspace tracking algorithm with subsampled vectors is to estimate $\mathcal{S}_t$ given v_{Ω_t}– an incomplete version of v_t, sampled only on the indices $\Omega \subset \{1, \ldots, n\}$. The GROUSE [2] algorithm addresses exactly this problem. GROUSE is an incremental gradient descent algorithm performed on the Grassmannian $\mathcal{G}(d, n)$, the space of all d-dimensional subspaces of $\mathbb{R}^n$. The algorithm minimizes an l^2-norm cost between observed incomplete vectors and their fit to the subspace variable. Each step of the algorithm is simple and requires very few operations. However, the use of the l^2 loss makes GROUSE very susceptible to outliers.

2. Grassmannian Robust Adaptive Subspace Tracking

Let the columns of an $n \times d$ matrix U_t be orthonormal and span $\mathcal{S}_t$. Tracking the evolving subspace $\mathcal{S}_t$ is equivalent to estimating U_t at each time step[3].

2.1. Model

At each time step t, we assume that a vector or video frame v_t is generated by the following model:

$$v_t = U_t w_t + s_t + \zeta_t \tag{1}$$

where w_t is the $d \times 1$ weight vector, ζ_t is the $n \times 1$ zero-mean Gaussian white noise vector with small variance, and s_t is the $n \times 1$ sparse outlier vector that models foreground pixels in the separation problem. We subsample v_t on the index set $\Omega_t \subset \{1, \ldots, n\}$. Conforming to the notation of GROUSE [2], we let U_{Ω_t} denote the submatrix of U_t consisting of the rows indexed by Ω_t; also for a vector $v_t \in \mathbb{R}^n$, let v_{Ω_t} denote a vector in $\mathbb{R}^{|\Omega_t|}$ whose entries are those of v_t indexed by Ω_t. A critical problem raised when we only partially observe v_t is how to quantify the subspace error only from the incomplete and corrupted data. GROUSE [2] uses the Euclidean distance, the l^2-norm, to measure the subspace error from the subspace spanned by the columns of U_t to the observed vector v_{Ω_t}:

$$F_{grouse}(\mathcal{S}; t) = \min_w \|U_{\Omega_t} w - v_{\Omega_t}\|_2^2 \; . \tag{2}$$

[3] We remind the reader here that U_t is not unique for a given subspace, but the projection matrix $U_t U_t^T$ is unique.

It was shown in [3] that this cost function gives an accurate estimate of the same cost function with full data ($\Omega = \{1, \ldots, n\}$), as long as $|\Omega_t|$ is large enough[4]. However, if the observed data vector is corrupted by outliers as in Equation (1), an l^2-based best-fit to the subspace can be influenced arbitrarily with just one large outlier; this in turn leads to an incorrect subspace update. In order to quantify the subspace error robustly, we use the l^1-norm as follows:

$$F_{grasta}(\mathcal{S}; t) = \min_{w} \|U_{\Omega_t} w - v_{\Omega_t}\|_1 \ . \tag{3}$$

With U_{Ω_t} known (or estimated, but fixed), this l^1 minimization problem is the classic least absolute deviations problem; Boyd [5] describes in detail a fast solver based on the technique of ADMM (Alternating Direction Method of Multipliers)[5]. According to [5], we can rewrite the right hand of Equation (3) as the equivalent constrained problem by introducing a sparse outlier vector $s \in \mathbb{R}^{|\Omega_t|}$:

$$\begin{aligned} \min \quad & \|s\|_1 \\ s.t. \quad & U_{\Omega_t} w + s - v_{\Omega_t} = 0 \end{aligned} \tag{4}$$

The augmented Lagrangian of this problem is

$$\begin{aligned} \mathcal{L}(U, s, w, y) = \|s\|_1 \quad &+ \quad y^T (U_{\Omega_t} w + s - v_{\Omega_t}) \\ &+ \quad \frac{\rho}{2} \|U_{\Omega_t} w + s - v_{\Omega_t}\|_2^2 \end{aligned} \tag{5}$$

where $y \in \mathbb{R}^{|\Omega_t|}$ is the dual vector. Our unknowns are s, y, U, and w. Note that since U is constrained to a non-convex manifold ($U^T U = I$), this function is not convex (neither is Equation (2)). However, if U were estimated, we could solve for the triple (s, w, y) using ADMM; also if (s, w, y) were estimated, we could refine our estimate of U. This is the alternating approach we take with GRASTA.

2.2. Update of the sparse vector, weight vector, and dual vector

Given the current estimated subspace $\widehat{U}_t$, the partial observation v_{Ω_t}, and the observed entries' indices Ω_t, the optimal (s^*, w^*, y^*) of Equation (3) can be found by minimizing the augmented Lagrangian. Specifically, in ADMM [5] these quantities are computed as follows. In this paper we always assume that $U_{\Omega_t}^T U_{\Omega_t}$ is invertible, which is guaranteed if $|\Omega_t|$ is large enough [3]. We have:

$$w^{k+1} = \frac{1}{\rho} (\widehat{U}_{\Omega_t}^T \widehat{U}_{\Omega_t})^{-1} \widehat{U}_{\Omega_t}^T (\rho(v_{\Omega_t} - s^k) - y^k) \tag{6}$$

$$s^{k+1} = \mathsf{S}_{\frac{1}{1+\rho}} (v_{\Omega_t} - \widehat{U}_{\Omega_t} w^{k+1} - y^k) \tag{7}$$

$$y^{k+1} = y^k + \rho(\widehat{U}_{\Omega_t} w^{k+1} + s^{k+1} - v_{\Omega_t}) \tag{8}$$

where $\mathsf{S}_{\frac{1}{1+\rho}}$ is the elementwise soft thresholding operator; for this and details on implementation of ADMM, we refer the reader to [5].

[4] In [3] the authors show that $|\Omega_t|$ must be larger than $\frac{8}{3}\mu(\mathcal{S}) d \log(2d/\delta)$, where $\mu(\mathcal{S})$ is a measure of incoherence on the subspace and δ controls the probability of the result.

[5] http://www.stanford.edu/~boyd/papers/admm/

2.3. Subspace Update

The set of all subspaces of $\mathbb{R}^n$ of fixed dimension d is called *Grassmannian*, which is a compact Riemannian manifold and is denoted by $\mathcal{G}(d, n)$. The comprehensive survey [10] covers how both the Grassmannian geodesics and the gradient of a function defined on the Grassmannian can be explicitly computed.

Augmented Lagrangian as the Loss Function There is a critical limitation to using Equation 3 directly as the robust loss function: regarding U as the variable, this loss function is not differentiable everywhere. We propose to instead use the augmented Lagrangian as the subspace loss function once we have estimated (s^*, w^*, y^*) from the previous $\widehat{U}_{\Omega_t}$ and v_{Ω_t}. Simply substitute (s^*, w^*, y^*) for (s, w, y) in Equation 5; this new subspace loss function is differentiable. The use of the augmented Lagrangian for the loss function is a crucial innovation in this work. One can think of using this loss function for updating U as running an ADMM on the full, nonconvex problem. To the best of the authors' knowledge, there is no theoretical work discussing convergence of ADMM on a nonconvex problem. We are unable to remedy this in the present work, but we show empirically that ADMM efficiently solves our nonconvex subspace tracking problem.

Grassmannian Geodesic Gradient Step In order to take a gradient step along the geodesic of the Grassmannian, according to [10], we first need to derive the gradient formula of the real-valued loss function Equation (5) $\mathcal{L} : \mathcal{G}(d, n) \rightarrow \mathbb{R}$. From Equation (2.70) in [10], the gradient $\nabla\mathcal{L}$ can be determined from the derivative of $\mathcal{L}$ with respect to the components of U. Let χ_{Ω_t} be the $|\Omega_t|$ columns of an $n \times n$ identity matrix corresponding to those indices in Ω_t; that is, this matrix zero-pads a vector in $\mathbb{R}^{|\Omega_t|}$ to be length n with zeros on the complement of Ω_t. The derivative of the augmented Lagrangian loss function $\mathcal{L}$ with respect to the components of U is as follows:

$$\frac{d\mathcal{L}}{dU} = \left[\chi_{\Omega_t} \left(y^* + \rho(U_{\Omega_t} w^* + s^* - v_{\Omega_t})\right)\right] w^{*T} \ . \tag{9}$$

Then the gradient $\nabla\mathcal{L}$ is $\nabla\mathcal{L} = (I - UU^T)\frac{d\mathcal{L}}{dU}$ [10]. From Step 4 of Algorithm 1, we have that $\nabla\mathcal{L} = \Gamma w^{*T}$ (see the definition of Γ in Alg 1). It is easy to verify that $\nabla\mathcal{L}$ is rank one since Γ is a $n \times 1$ vector and w^* is a $d \times 1$ weight vector. Then it is trivial to compute the singular value decomposition of $\nabla\mathcal{L}$, from which we can derive the formula for a geodesic step on the Grassmannian in the direction of the gradient. The sole non-zero singular value is $\sigma = \|\Gamma\|\|w^*\|$, and the corresponding left and right singular vectors are $\frac{\Gamma}{\|\Gamma\|}$ and $\frac{w^*}{\|w^*\|}$ respectively. Then we can write the SVD of the gradient explicitly by adding the orthonormal set $x_2, \ldots, x_d$ orthogonal to Γ as left singular vectors

and the orthonormal set $y_2, \ldots, y_d$ orthogonal to w^* as right singular vectors as follows:

$$\nabla \mathcal{L} = \left[\frac{\Gamma}{\|\Gamma\|} \;\; x_2 \;\; \ldots \;\; x_d \right] \times \mathrm{diag}(\sigma, 0, \ldots, 0) \times \left[\frac{w^*}{\|w^*\|} \;\; y_2 \;\; \ldots \;\; y_d \right]^T$$

Finally, following Equation (2.65) in [10], a gradient step of length η in the direction $-\nabla \mathcal{L}$ is given by

$$U(\eta) = U + (\cos(\eta\sigma) - 1) \frac{U w_t^*}{\|w_t^*\|} \frac{w_t^{*T}}{\|w_t^*\|} - \sin(\eta\sigma) \frac{\Gamma}{\|\Gamma\|} \frac{w_t^{*T}}{\|w_t^*\|} . \quad (10)$$

2.4. Algorithm

The discussion of Sections 2.2 to 2.3 can be summarized into our algorithm as follows. For each time step t, we observe v_{Ω_t}, a subsampled vector (i.e., sampling pixels from a video frame) which follows the low-rank + sparse model (i.e., background + foreground). Our algorithm will first estimate the optimal value (s^*, w^*, y^*) from our current estimated subspace U_t via the l^1 minimization ADMM solver; then compute the gradient of the augmented Lagrangian loss function $\mathcal{L}$; and finally do the rank one subspace update via Equation (10). There are many ways to pick the step-size; [2] shows both diminishing and constant step-sizes, and we have also tried an adaptive step-size based on [14].

We exhibit GRASTA (Grassmannian Robust Adaptive Subspace Tracking Algorithm) in Algorithm 1. Step 3 of GRASTA is ADMM with a slight modification: in addition to returning s^* we also return the weight vector w^* and the dual vector y^* for the further computation of the gradient $\nabla \mathcal{L}$ and the update step.

2.5. Computational Cost and Memory Usage

Each subspace update step in GRASTA needs only simple linear algebraic computations. The total computational cost of each step of Algorithm 1 is $O(|\Omega|d^3 + Kd|\Omega| + nd^2)$, where $|\Omega|$ is the number of samples per vector used, d is the dimension of the subspace, n is the ambient dimension, and K is the number of ADMM iterations.

Specifically, estimating (s_t^*, w_t^*, y_t^*) with ADMM costs at most $O(|\Omega|d^3 + Kd|\Omega|)$ operations; computing the gradient $\nabla \mathcal{L}$ needs simple matrix-vector multiplication which costs $O(|\Omega|d + nd)$ operations; and the final update step also costs $O(nd^2)$ oeprations. Throughout the tracking process, GRASTA only needs $O(nd)$ memory elements to maintain the estimated low-rank orthonormal basis $\widehat{U}_t$, $O(n)$ elements for s^* and y^*, $O(d)$ elements for w^*. This analysis shows that the GRASTA updates are both computation and memory efficient.

Algorithm 1 Grassmannian Robust Adaptive Subspace Tracking (GRASTA)

Require: An $n \times d$ orthogonal matrix U_0. A sequence of corrupted vectors v_t, each vector observed in entries $\Omega_t \subset \{1, \ldots, n\}$. A structure OPTS that holds parameters for ADMM.

Return: The estimated subspace U_t at time t.

1: **while** $t =$ current frame **do**
2: Extract U_{Ω_t} from U_t: $U_{\Omega_t} = \chi_{\Omega_t}^T U_t$
3: Estimate the sparse residual s_t^*, weight vector w_t^*, and dual vector y_t^* from the observed entries Ω_t via ADMM using OPTS:
$(s_t^*, w_t^*, y_t^*) = \arg\min_{w,s,y} \mathcal{L}(U_{\Omega_t}, w, s, y)$
4: Compute the gradient $\nabla \mathcal{L}$ as follows:
$\Gamma_1 = y_t^* + \rho(U_{\Omega_t} w_t^* + s_t^* - v_{\Omega_t})$, $\Gamma_2 = U_{\Omega_t}^T \Gamma_1$,
$\Gamma = \chi_{\Omega_t} \Gamma_1 - U\Gamma_2$, $\nabla \mathcal{L} = \Gamma w_t^{*T}$
5: Compute step-size η_t.
6: Update subspace:
$U_{t+1} = U_t + \left((\cos(\eta_t\sigma) - 1) U_t \frac{w_t^*}{\|w_t^*\|} - \sin(\eta_t\sigma) \frac{\Gamma}{\|\Gamma\|} \right) \frac{w_t^{*T}}{\|w_t^*\|}$, where $\sigma = \|\Gamma\|\|w_t^*\|$
7: **end while**

3. Performance Evaluation

In this section we discuss the application of GRASTA to realtime separation of foreground objects from the background in video surveillance. Here we consider three scenarios in the video tasks, with a spectrum of challenges for robust subspace tracking. In the first we have videos with static background and objects moving in the foreground. In the second, we have a video with a still background but with changing lighting. In the third, we simulate a panning camera to examine GRASTA's performance with a dynamic background. All the following experiments were done on a Macbook Pro laptop with 2.3GHz Intel Core i5 CPU and 4 GB RAM. Our MATLAB GRASTA code is available online[6].

Static Background We start by examining the speed-accuracy trade-off of three benchmark videos from [16], "Airport Hall", "Lobby", and "Bootstrap". We compare GRASTA with three other algorithms: Inexact Augmented Lagrange Multiplier Method (IALM) for batch or offline robust PCA [18], Recursive Projected Compressed Sensing (ReProCS) for online robust PCA [22], and the median filter. Unless otherwise noted, for all experiments we used a max-iteration value of 20 for IALM and $\lambda = 1/\sqrt{mp}$ where one video frame is $m \times p$. For ReProCS, we use the ReProCS(modCS) algorithm downloaded from the au-

[6] http://sites.google.com/site/hejunzz/grasta

thors' website[7], and use the default parameters provided in the demo code of ReProCS. Since ReProCS can be very memory consuming, in order to run it on our experimental laptop, we resized all the videos for ReProCS only: "Airport Hall" to 36×44, "Lobby" to 64×80, and "Bootstrap" to 30×40; see Table 1 for original sizes. Also since ReProCS requires a clean background for initialization, we selected 400 frames randomly and used the output of IALM to initialize ReProCS. For the median filter we used a running window of 20 frames. Finally, for GRASTA, we cycled 10 times over 100 random frames for training an initial $d = 5$ dimensional subspace, and then we subsampled at different levels to get the different points in Figure 1. We used max-iterations $K = 10$ for ADMM. We assessed accuracy by normalizing the sparse output of each algorithm to be between zero and one, thresholding at 0.1, and comparing the result to the ground truth frames[8]. The figure shows the trade-off between computation time and accuracy afforded by ReProCS, IALM, and GRASTA at two different sampling rates.

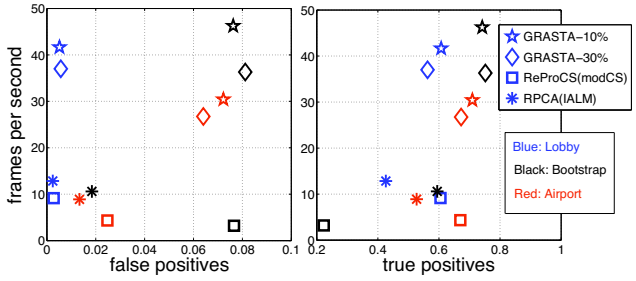


Figure 1. A comparison of GRASTA at 10% and 30% sampling to ReProCS [22] and IALM for RPCA [18] on accuracy and time. On the "Bootstrap" data set with 10% sampling, GRASTA operated at 46.3 fps.

Another approach if the video background is known to be static or near static is to use GRASTA first to identify the background, and then use only ADMM to separate the foreground from the background. Once the video background has been trained as a subspace U, separating the foreground in each frame can be simply done by first estimating the weight vector w_t via ADMM from a small subsample of each frame's pixels, and then subtracting Uw_t from the full video, as in Equation 11. We call this "GRASTA-0" from here on.

$$\begin{aligned} BG &= Uw_t \\ FG &= video(t) - BG \end{aligned} \tag{11}$$

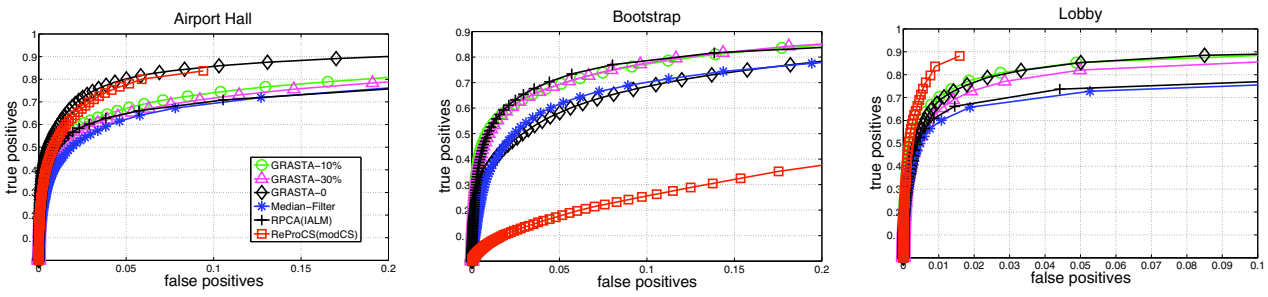
[7] http://home.engineering.iastate.edu/~chenlu/ReProCS/ReProCS_main.htm

[8] Find these along with the videos at http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.

Now we look at the ROC (receiver operating curve) behavior as in [24], comparing against ReProCS, IALM, and the median filter. The tradeoffs in performance between the algorithms is clear from Figure 2. On "Bootstrap," GRASTA and IALM both perform very well. Also note that for GRASTA, the performance is stable even as the sampling percentage decreases; since the background is truly low-rank we do not need to fully sample and we can still perform separation successfully. We believe that ReProCS performs poorly here because the movements of the foreground objects are much more complex than other benchmark videos, and thus ReProCS cannot predict the sparse supports correctly. On the other hand, ReProCS does almost as well as GRASTA-0 on the "Hall" video where the movements of foreground objects are more regular; GRASTA performs as well as IALM.

GRASTA performance. Here we focus on simulations testing the speed performance of GRASTA. Table 1 shows real-time video separation results. For the first experiment, we use the "Airport Hall" dataset. We let GRASTA cycle 10 times over 50 training frames just from 30% random entries of each frame to get the stationary subspace U. Training the subspace costs 11.3 seconds. Then we perform background and foreground separation for all frames in a streaming fashion, and when dealing with each frame we only randomly observe 1% of the entries. The separation task is performed as in Equation 11, and the separating time is 20.9 seconds, which means we achieve 171.5 FPS (frames per second) real-time performance. In order to show GRASTA can handle higher resolution video effectively, we use the "Shopping Mall" [16] video as the second experiment. The subspace training stage uses the same parameter settings as "Airport Hall". We do the background and foreground separation only from 1% entries of each frame. For "Shopping Mall" the separating time is 27.5 seconds for total 1286 frames. Thus we achieve 46.8 FPS real-time performance. Figure 3 shows the separation quality at $t = 1, 600, 1200$.

Dynamic Background: Changing Lighting Here we consider a problem where the lighting in the video is changing throughout. We use the "Lobby" dataset from [16]. In order to adjust to the lighting changes, we run the full GRASTA Algorithm 1 for every frame. We use 30% of the pixels of every frame to do this update and 100% of the pixels to do the separation. In all of these dynamic background video experiments we used a maximum of $K = 20$ iterations of the ADMM algorithm per subspace update. Again, see the numerical results in Table 1. The results are illustrated in Figure 4. Figure 2 (c) shows the ROC comparison to other algorithms. GRASTA-0 performs well because its initial background subspace is trained from frames throughout the different lighting conditions. ReProCS does well here because people move into the scene and remain still;



(a) (b) (c)

Figure 2. ROC curves for benchmark videos. Grasta-0 does not perform subspace updates after the background is trained.

Dataset	Resolution	Total Frames	Training SS (Sub-Sampling)	Tracking SS	Separation SS	Tracking/Separation Algorithm	Training Time	Tracking and Separating Time	FPS
Airport Hall	144×176	3584	30%	-	1%	GRASTA-0	11.3 sec	20.9 sec	171.5
Bootstrap	120×160	3055	30%	-	1%	GRASTA-0	13.8 sec	15.5 sec	197.1
Shopping Mall	320×256	1286	30%	-	1%	GRASTA-0	33.9 sec	27.5 sec	46.8
Lobby	144×176	1546	30%	30%	100%	Full GRASTA Alg 1	3.9 sec	71.3 sec	21.7
Hall with Virtual Pan (1)	144×88	3584	100%	100%	100%	Full GRASTA Alg 1	3.8 sec	191.3 sec	18.7
Hall with Virtual Pan (2)	144×88	3584	50%	50%	100%	Full GRASTA Alg 1	3.7 sec	144.8 sec	24.8

Table 1. Real-time video background and foreground separation by GRASTA. Here we use four different resolution video datasets, the first three with static background and the last three with dynamic background. We train from 50 frames; in the first three experiments they are chosen randomly from throughout, and in the last three they are the first 50 frames. The subspace dimension is 5 for all.

Figure 3. Real-time video background and foreground separation from partial information. The first row is the original video frame at each time; the middle row is the recovered background at each time only from 1% information; and bottom row is the foreground estimated by GRASTA-0.

ReProCS predicts the sparse part at time t using that at time $t-1$, whereas IALM and GRASTA will adapt the background to incorporate the unchanging "foreground."

Dynamic Background: Virtual Pan In the last experiment, we demonstrate that GRASTA can effectively track the right subspace in video with a dynamic background. We consider panning a "virtual camera" periodically by 20 pixels from left to right and back through the video to simulate a dynamic background. The idea of the virtual camera is illustrated with Figure 5.

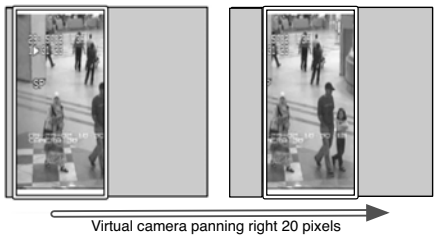


Figure 5. Demonstration of panning the "virtual camera" right 20 pixels.

We choose "Airport Hall" as the dataset. We set the scope of the virtual camera to be half the width, so the resolution of the virtual camera is 144×88. We set the subspace dimension to 5. Figure 6 shows how GRASTA can quickly adapt to the changed background in just 25 frames when the virtual camera pans 20 pixels to the right at $t = 101$. We also let GRASTA track and do the separation task for all frames. When we use 100% of the pixels for the tracking and separation, the total computation time is 191.3 seconds, or 18.7 FPS, and adjusting to a new camera position after the camera pans takes 25 frames as can be seen in Figure 6. When we use 50% of the pixels for tracking and 100% of the pixels for separation, the total computation time is 144.8 seconds or 24.8 FPS, and the adjustment to the new camera position takes around 50 frames.

4. Discussion and Future Work

In this paper we have presented a robust online subspace tracking algorithm, GRASTA. The algorithm estimates a low-rank model from noisy, corrupted, and incomplete data, even when the best low-rank model may be time-varying.

Though this work presents a successful algorithm, many questions remain. First and foremost, because the cost function in Equation (3) has the subspace variable U which is constrained to a non-convex manifold, the resulting optimization is non-convex. A proof of convergence to the global minimum of this algorithm is of great interest.

We have shown that one of the very promising applications of GRASTA is that of separating background and foreground in video surveillance. We are very interested to apply GRASTA to more videos with dynamic backgrounds: for example, natural background scenery which may blow in the wind. In doing this we will study the resulting tradeoff between the kinds of movement that would be captured as part of the background and the movement that would be identified as foreground.

References

[1] D. Babwin. Cameras make chicago most closely watched U.S. city. *Huffington Post*, April 6 2010. 1

[2] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *Proceedings of Allerton*, September 2010. Available at http://arxiv.org/abs/1006.4046. 1, 2, 4

[3] L. Balzano, B. Recht, and R. Nowak. High-dimensional matched subspace detection when data are missing. In *Proceedings of ISIT*, 2010. 3

[4] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE TPAMI*, 25(2):218–233, February 2003. 1, 2

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–123, 2011. 3

[6] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. 1

[7] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(1):1–37, 2009. 1, 2

[8] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21:572, 2011. 2

[9] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29, 1998. 1

[10] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. 3, 4

[11] R. Epstein, P. Hallinan, and A. Yuille. 5+/-2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *PBMCV95*, page SESSION 4, 1995. 2

[12] P. J. Huber and E. M. Ronchetti. *Robust Statistics*. Wiley, 2009. 2

[13] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–2078, July 2010. 1

[14] S. Klein, J. Pluim, M. Staring, and M. Viergever. Adaptive stochastic gradient descent optimisation for image registration. *International journal of computer vision*, 81(3):227–239, 2009. 4

[15] H. Krim and M. Viberg. Two decades of array signal processing research: the parametric approach. *Signal Processing Magazine, IEEE*, 13(4):67–94, July 1996. 1

[16] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. Statistical modeling of complex background for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, November 2004. 1, 4, 5

[17] Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004. 2

[18] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report UILU-ENG-09-2215, UIUC, October 2009. Arxiv preprint arXiv:1009.5055. 4, 5

[19] G. Mateos and G. B. Giannakis. Sparsity control for robust principal component analysis. In *Proc. of Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2010. 2

[20] M. McCahill and C. Norris. Cctv in london. Working Paper 6, Centre for Criminology and Criminal Justice, University of Hull, United Kingdom, June 2002. See http://www.urbaneye.net/results/ue_wp6.pdf. 1

[21] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000. 1, 2

[22] C. Qiu and N. Vaswani. Reprocs: A missing link between recursive robust pca and recursive sparse recovery in large but correlated noise. Available at http://arxiv.org/abs/1106.3286, 2011. 2, 4, 5

[23] B. Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12:3413–3430, 2011. 1

[24] R. Sivalingam, A. D'Souza, V. Morellas, N. Papanikolopoulos, M. Bazakos, and R. Miezianko. Dictionary learning for robust background modeling. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2011. 5

[25] F. Torre and M. Black. A framework for robust subspace learning. *Intl. J. of Computer Vision*, 54:117142, 2003. 2

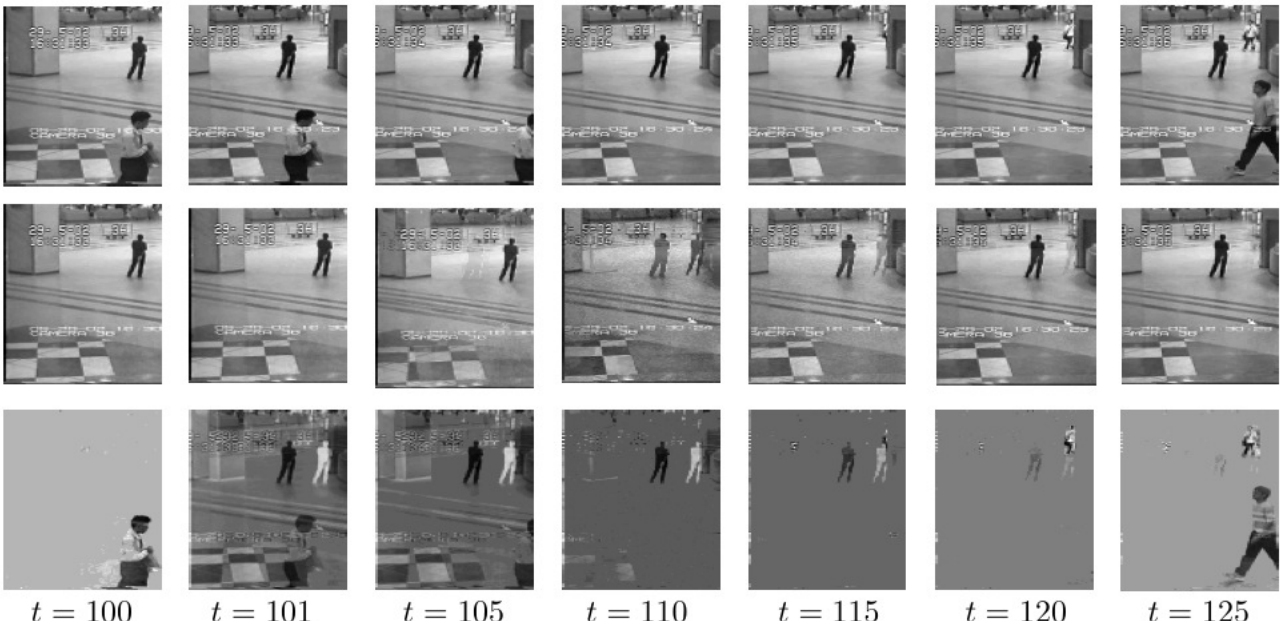
[26] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 2010. 1

[27] L. Wang, L. Wang, M. Wen, Q. Zhuo, and W. Wang. Background subtraction using incremental subspace learning. In *Proceedings of ICIP*, 2007. 1

[28] J. Wright, Y. Ma, A. Ganesh, and S. Rao. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Proceedings of NIPS*, 2009. 2

$t = 180$ $t = 366$ $t = 650$ $t = 1000$ $t = 1360$

Figure 4. Real-time video background and foreground separation from partial information. The first row is the original video frame; the middle row is the recovered background from only 30% of the pixels; and bottom row is the foreground estimated using all pixels. The differing background colors of the bottom row is simply an artifact of colormap in Matlab.



$t = 100$ $t = 101$ $t = 105$ $t = 110$ $t = 115$ $t = 120$ $t = 125$

Figure 6. Real-time dynamic background tracking and foreground separation. At time $t = 101$, the virtual camera slightly pans to right 20 pixels. We show how GRASTA quickly adapts to the new subspace by $t = 125$. The first row is the original video frame; the middle row is the tracked background; the bottom row is the separated foreground.