Background Subtraction Using Low Rank and Group Sparsity Constraints

Xinyi Cui[1], Junzhou Huang[2], Shaoting Zhang[1], and Dimitris N. Metaxas[1]

[1] CS Dept., Rutgers University
Piscataway, NJ 08854, USA
[2] CSE Dept., Univ. of Texas at Arlington
Arlington, TX, 76019, USA
{xycui,shaoting,dnm}@cs.rutgers.edu,
jzhuang@uta.edu

Abstract. Background subtraction has been widely investigated in recent years. Most previous work has focused on stationary cameras. Recently, moving cameras have also been studied since videos from mobile devices have increased significantly. In this paper, we propose a unified and robust framework to effectively handle diverse types of videos, *e.g.*, videos from stationary or moving cameras. Our model is inspired by two observations: 1) background motion caused by orthographic cameras lies in a low rank subspace, and 2) pixels belonging to one trajectory tend to group together. Based on these two observations, we introduce a new model using both low rank and group sparsity constraints. It is able to robustly decompose a motion trajectory matrix into foreground and background ones. After obtaining foreground and background trajectories, the information gathered on them is used to build a statistical model to further label frames at the pixel level. Extensive experiments demonstrate very competitive performance on both synthetic data and real videos.

1 Introduction

Background subtraction is an important preprocessing step in video surveillance systems. It aims to find independent moving objects in a scene. Many algorithms have been proposed for background subtraction under stationary. In recent years, videos from moving cameras have also been studied [1] because the number of moving cameras (such as smart phones and digital videos) has increased significantly. However, handling diverse types of videos robustly is still a challenging problem (see the related work in Sec. 2). In this paper, we propose a unified framework for background subtraction, which can robustly deal with videos from stationary or moving cameras with various number of righd/non-rigid objects.

The proposed method for background subtraction is based on two sparsity constraints applied on foreground and background levels, *i.e.*, low rank [2] and group sparsity constraints [3]. It is inspired by recently proposed sparsity theories [4, 5]. There are two "sparsity" observations behind our method. *First*, when

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 612–625, 2012.

the scene in a video does not have any foreground moving objects, video motion has a low rank constraint for orthographic cameras [6]. Thus the motion of background points forms a low rank matrix. *Second*, foreground moving objects usually occupy a small portion of the scene. In addition, when a foreground object is projected to pixels on multiple frames, these pixels are not randomly distributed. They tend to group together as a continuous trajectory. Thus these foreground trajectories usually satisfy the group sparsity constraint.

These two observations provide important information to differentiate independent objects from the scene. Based on them, the video background subtraction problem is formulated as a matrix decomposition problem. First, the video motion is represented as a matrix on *trajectory* level (*i.e.* each row in the motion matrix is a trajectory of a point). Then it is decomposed into a background matrix and a foreground matrix, where the background matrix is low rank, and the foreground matrix is group sparse. This low rank constraint is able to automatically model background from both stationary and moving cameras, and the group sparsity constraint improves the robustness to noise.

The trajectories recognized by the above model can be further used to label a frame into foreground and background at the pixel level. Motion segments on a video sequence are generated using fairly standard techniques. Then the color and motion information gathered from the trajectories is employed to classify the motion segments as foreground or background. Our approach is validated on various types of data, *i.e.*, synthetic data, real-world video sequences recorded by stationary cameras or moving cameras and/or nonrigid foreground objects. Extensive experiments also show that our method compares favorably to the recent state-of-the-art methods.

The main contribution of the proposed approach is a new model using low rank and group sparsity constraints to differentiate foreground and background motions. This approach has three merits: 1) The low rank constraint is able to handle both static and moving cameras; 2)The group sparsity constraint leverages the information of neighboring pixels, which makes the algorithm robust to random noise; 3) It is relatively insensitive to parameter settings.

2 Related Work

Background Subtraction. A considerable amount of work has studied the problem of background subtraction in videos. Here we review a few related work, and please refer to [7] for comprehensive surveys. The mainstream in the research area of background subtraction focuses on stationary cameras. The earliest background subtraction methods use frame difference to detect foreground [8]. Many subsequent approaches have been proposed to model the uncertainty in background appearance, such as, but not limited to, W4 [9], single gaussian model [10], mixture of gaussian [11], non-parametric kernel density [12] and joint spatial-color model [13]. One important variation in stationary camera based research is the background dynamism. When the camera is stationary, the background scene may change over time due to many factors (*e.g.* illumination

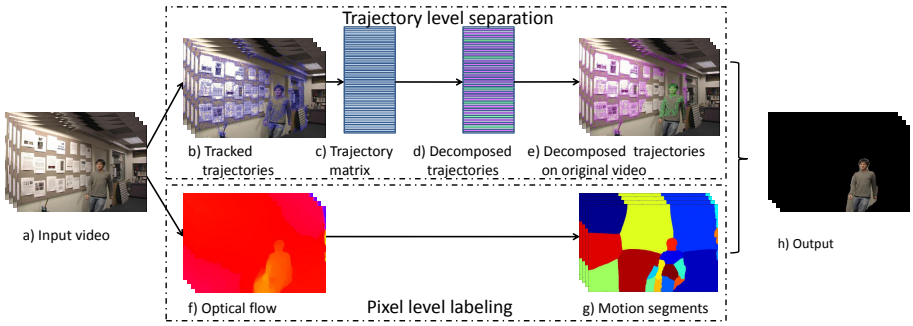

Fig. 1. The framework. Our method takes a raw video sequence as input, and produces a binary labeling as the output. Two major steps are trajectory level separation and pixel level labeling.

changes, waves in water bodies, shadows, etc). Several algorithms have been proposed to handle dynamic background [14–16].

The research for moving cameras has recently attracted people's attention. Motion segmentation approaches [17, 18] segment point trajectories based on subspace analysis. These algorithms provide interesting analysis on sparse trajectories, though do not output a binary mask as many background subtraction methods do. Another popular way to handle camera motion is to have strong priors of the scene, *e.g.*, approximating background by a 2D plane or assuming that camera center does not translate [19, 20], assuming a dominant plane [21], etc. [22] propose a method to use belief propagation and Bayesian filtering to handle moving cameras. Different from their work, long-term trajectories we use encode more information for background subtraction. Recently, [1] has been proposed to build a background model using RANSAC to estimate the background trajectory basis. This approach assumes that the background motion spans a three dimensional subspace. Then sets of three trajectories are randomly selected to construct the background motion space until a consensus set is discovered, by measuring the projection error on the subspace spanned by the trajectory set. However, RANSAC based methods are generally sensitive to parameter selection, which makes it less robust when handling different videos.

Group sparsity [23, 24] and low rank constraint [2] have also been applied to background subtraction problem. However, these methods only focus on stationary camera and their constraints are at spatial pixel level. Different from their work, our method is based on constraints in temporal domain and analyzing the trajectory properties.

3 Methodology

Our background subtraction algorithm takes a raw video sequence as input, and generates a binary labeling at the pixel level. Fig. 1 shows our framework. It has two major steps: trajectory level separation and pixel level labeling. In

the first step, a dense set of points is tracked over all frames. We use an off-the-shelf dense point tracker [25] to produce the trajectories. With the dense point trajectories, a low rank and group sparsity based model is proposed to decompose trajectories into foreground and background. In the second step, motion segments are generated using optical flow [26] and graph cuts [27]. Then the color and motion information gathered from the recognized trajectories builds statistics to label motion segments as foreground or background.

3.1 Low Rank and Group Sparsity Based Model

Notations: Given a video sequence, k points are tracked over l frames. Each trajectory is represented as $p_i = [x_{1i}, y_{1i}, x_{2i}, y_{2i}, ...x_{li}, y_{li}] \in \mathbb{R}^{1\times 2l}$, where x and y denote the 2D coordinates in each frame. The collection of k trajectories is represented as a $k \times 2l$ matrix, $\phi = [p_1^T, p_2^T, ..., p_l^T]^T, \quad \phi \in \mathbb{R}^{k\times 2l}$.

In a video with moving foreground objects, a subset of k trajectories comes from the foreground, and the rest belongs to the background. Our goal is to decompose tracked k trajectories into two parts: m background trajectories and n foreground trajectories. If we already know exactly which trajectories belong to the background, then foreground objects can be easily obtained by subtracting them from k trajectories, and vice versa. In other words, ϕ can be decomposed as:

$$\phi = B + F, \tag{1}$$

where $B \in \mathbb{R}^{k\times 2l}$ and $F \in \mathbb{R}^{k\times 2l}$ denote matrices of background and foreground trajectories, respectively. In the ideal case, the decomposed foreground matrix F consists of n rows of foreground trajectories and m rows of flat zeros, while B has m rows of background trajectories and n rows of zeros.

Eq. 1 is a severely under-constrained problem. It is difficult to find B and F without any prior information. In our method, we incorporate two effective priors to robustly solve this problem, *i.e.*, the low rank constraint for the background trajectories and the group sparsity constraint for the foreground trajectories.

Low Rank Constraint for the Background. In a 3D structured scene without any moving foreground object, video motion solely depends on the scene and the motion of the camera. Our background modeling is inspired from the fact that B can be factored as a $k \times 3$ structure matrix of 3D points and a $3 \times 2l$ orthogonal matrix [6]. Thus the background matrix is a low rank matrix with rank value at most 3. This leads us to build a low rank constraint model for the background matrix B:

$$rank(B) \leq 3, \tag{2}$$

Another constraint has been used in the previous research work using RANSAC based method [1]. This work assumes that the background matrix is of rank three: $rank(B) = 3$. This is a very strict constraint for the problem. We refer the above two types of constraints as the General Rank model (GR) and the Fixed Rank model (FR). Our GR model is more general and handles more situations. A rank-3 matrix models 3D scenes under moving cameras; a rank-2

matrix models a 2D scene or 3D scene under stationary cameras; a rank-1 matrix is a degenerated case when scene only has one point. The usage of GR model allows us to develop a unified framework to handle both stationary cameras and moving cameras. The experiment section (Sec. 4.2) provides more analysis on the effectiveness of the GR model when handling diverse types videos.

Group Sparsity Constraint for the Foreground. Foreground moving objects, in general, occupy a small portion of the scene. This observation motivates us to use another important prior, *i.e.*, the number of foreground trajectories should be smaller than a certain ratio of all trajectories: $m \leq \alpha k$, where α controls the sparsity of foreground trajectories.

Another important observation is that each row in ϕ represents one trajectory. Thus the entries in ϕ are not randomly distributed. They are spatially clustered within each row. If one entry of the ith row ϕ_i belongs to the foreground, the whole ϕ_i is also in the foreground. This observation makes the foreground trajectory matrix F satisfy the group sparsity constraint:

$$\|F\|_{2,0} \leq \alpha k, \tag{3}$$

where $\|\cdot\|_{2,0}$ is the mixture of both L_2 and L_0 norm. The L_2 norm constraint is applied to each group separately (*i.e.*, each row of F). It ensures that all elements in the same row are either zero or nonzero at the same time. The L_0 norm constraint is applied to count the nonzero groups/rows of F. It guarantees that only a sparse number of rows are nonzero. Thus this group sparsity constraint not only ensures that the foreground objects are spatially sparse, but also guarantees that each trajectory is treated as one unit. Group sparsity is a powerful tool in computer vision problems [28, 23]. The standard sparsity constraint, L_0 norm, (we refer this as *Std. sparse* method) has been intensively studied in recent years. However, it does not work well for this problem compared to the group sparsity one. *Std. sparse* method treats each element of F independently. It does not consider any neighborhood information. Thus it is possible that points from the same trajectory are classified into two classes. In the experiment section (Sec. 4.1), we discuss the advantage of group sparsity constraint over sparsity constraint through synthetic data analysis, and also show that this constraint improves the robustness of our model.

Based on the low rank and group sparsity constraints, we formulate our objective function as:

$$\begin{aligned} \left(\hat{B}, \hat{F}\right) &= \arg\min_{B,F} \left(\| \phi - B - F \|_F^2\right), \\ \text{s.t. } &\mathit{rank}(B) \leq 3, \ \| F \|_{2,0} < \alpha k, \end{aligned} \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm. This model leads to a good separation of foreground and background trajectories. Fig. 2 illustrates our model.

Eq. 4 only has one parameter α, which controls the sparsity of the foreground trajectories. In general, user-tuning parameter is a key issue for a good model. It is preferable that the parameters are easy to tune and not sensitive to different

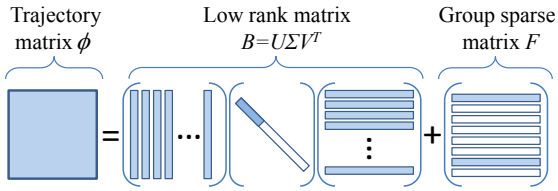

Fig. 2. Illustration of our model. Trajectory matrix ϕ is decomposed into a background matrix B and a foreground matrix F. B is a low rank matrix, which only has a few nonzero eigenvalues (*i.e.* the diagonal elements of Σ in SVD); F is a group sparse matrix. Elements in one row belongs to the same group (either foreground or background), since they lie on one trajectory. The foreground rows are sparse comparing to all rows. White color denotes zero values, while blue color denotes nonzero values.

datasets. Our method is relatively insensitive to parameter selection. Using $\alpha = 0.3$ works for all tested videos.

Low rank constraint is a powerful method in computer vision and machine learning area [29]. Low rank constraints and Robust PCA have been recently used to solve vision problems [30, 2, 31], including background subtraction at the pixel level [2]. It assumes that the stationary scenes satisfy a low rank constraint. However, this assumption does not hold when camera moves. Furthermore, that formulation does not consider any group information, which is an important constraint to make sure neighbor elements are considered together.

3.2 Optimization Framework

This subsection discusses how to effectively solve Eq. 4. The first challenge is that it is not a convex problem, because of the nonconvexity of the low rank constraint and the group sparsity constraint. Furthermore, we also need to simultaneously recover matrix B and F, which is generally a Chicken-and-Egg problem.

In our framework, alternating optimization and greedy methods are employed to solve this problem. We first focus on the fixed rank problem (*i.e.*, rank equals to 3), and then will discuss how to deal with the more general constraint of $rank \leq 3$.

Eq. 4 is divided into two subproblems with unknown B or F, and solved by using two steps iteratively:

Step 1: Fix B, and update F. The subproblem is:

$$\left(\hat{F}\right) = \arg\min_{F} \left(\| \phi' - F \|_F^2\right), \text{ s.t. } \|F\|_{2,0} < \alpha k, \tag{5}$$

where $\phi' = \phi - B$.

Step 2: Fix F, and update B. The subproblem is:

$$\left(\hat{B}\right) = \arg\min_{B} \left(\| \phi'' - B \|_F^2\right), \text{ s.t. } rank(B) = 3, \tag{6}$$

where $\phi'' = \phi - F$.

To initialize this optimization framework, we simply choose $B_{init} = \phi$, and $F_{init} = \mathbf{0}$. Greedy methods are used to solve both subproblems. To solve Eq. 5, we compute $\|F_i\|_2, i \in 1, 2, ..., k$, which represents the L_2 norm of each row. Then the αk rows with largest values are preserved, while the rest rows are set to zero. This is the estimated F in the first step. In the second step, ϕ'' is computed as per newly-updated F. To solve Eq. 6. Singular value decomposition (SVD) is applied on ϕ''. Then three eigenvectors with largest eigenvalues are used to reconstruct B. Two steps are alternatively employed until a stable solution of $\hat{B}$ is found. Then $\hat{F}$ is computed as $\phi - \hat{B}$. The reason of updating $\hat{F}$ after all iterations is that the greedy method of solving Eq. 5 discovers exact αk number of foreground trajectories, which may not be the real foreground number. On the contrary, B can be always well estimated, since a subset of unknown number of background trajectories is able to have a good estimation of background subspace. Thus we finalize $\hat{F}$ by $\phi - \hat{B}$. Since the whole framework is based on greedy algorithms, it does not guarantee a global minimum. In our experiments, however, it is able to generate reliable and stable results. The above-mentioned method solves the fixed rank problem, but the rank value in the background problem usually cannot be pre-determined. To handle this undetermined rank issue, we propose a multiple rank iteration method. First, B and F are initialized as $B_{init}^{(0)} = \phi$ and $F_{init}^{(0)} = \mathbf{0}^{k \times 2l}$. Then the fixed rank optimization procedure is performed on each specific rank starting from 1 to 3. The output of the current fixed rank procedure is fed to the next rank as its initialization. We obtain the final result $B^{(3)}$ and $F^{(3)}$ in the rank-3 iteration. Given a data matrix of $K \times 2L$ with K trajectories over L frames, the major calculation is $O(KL^2)$ for SVD on each iteration. Convergence of each fixed rank problem is achieved 6.7 iterations on average. The overall time complexity is $O(KL^2)$.

To explain why our framework works for the general rank problem, we discuss two examples. First, if the rank of B is 3 (*i.e.*, moving cameras), then this framework discovers an optimal solution in the third iteration, *i.e.*, using rank-3 model. The reason is that the first two iterations, *i.e.* the rank-1 and rank-2 models, cannot find the correct solution as they are using the wrong rank constraints. Second, if the rank of the matrix is 2 (*i.e.*, stationary cameras), then this framework obtains stable solution in the second iteration. This solution will not be affected in the rank-3 iteration. The reason is that the greedy method is used to solve Eq. 6. When selecting the eigenvectors with three largest eigenvalues, one of them is simply flat zero. Thus B does not change, and the solution is the same in this iteration. Note that low rank problems can also be solved using convex relaxation on the constraint problem [2]. However, our greedy method on unconstrained problem is better than convex relaxation in this application. Convex relaxation is not able to make use of the specific rank value constraint (≤ 3 in our case). The convex relaxation uses λ to implicitly constrain the rank level, which is hard to constrain a matrix to be lower than a specific rank value.

3.3 Pixel Level Labeling

The labeled trajectories from the previous step are then used to label each frame at the pixel level (*i.e.* return a binary mask for a frame). In this step, each frame is treated as an individual labeling task. First, the optical flow [26] is calculated between two consecutive frames. Then motion segments are computed using graph cuts [27] on optical flow. After collecting the motion segments, we want to label each motion segment s as f or b, where f and b denotes the label of foreground and background. There are two steps to label the segments. First, segments with high confidence belonging to f and b are selected. Second, a statistical model is built based on those segments. This model is used to label segments with low confidence. The confidence of a segment is determined by the number of labeled f and b trajectories. The low confidence segments are those ambiguous areas. To predict the labels of these low confidence segments, a statistical model is built for f and b based on high confidence ones. First, 20% pixels are uniformly sampled on segments with high confidence. Each sampled pixel w is represented by color in hue-saturation space (h, s), optical flow (u, v) and position on the frame (x, y). The reason we use sampled points to build the model instead of the original trajectories is that the sparse trajectories may not cover enough information (*i.e.*, color and positions) on the motion unit. Uniform sampling covering the whole segment can build a richer model. The segments are then evaluated using a kernel density function:$P(s_i|c) = \frac{1}{N \cdot |s_i|} \sum_{i=1}^{N} \sum_{j \in s_i} \kappa(e_j - w_i), c \in \{f, b\}$, where $\kappa(\cdot)$ is the Normal kernel, N is the total number of sampled pixels, and $|s_i|$ is the pixel number of s_i. For every pixel j lying on s_i, e_j denotes the vector containing color, optical flow and position.

4 Experiments

To evaluate the performance of our algorithm, we conduct experiments on different data sources: synthetic data, real-world videos from both moving and stationary cameras. The performance is evaluated by F-Measure, the harmonic mean of recall and precision. This is a standard measurement for background subtraction [7, 16]: $F = 2 \cdot recall \cdot precision / (recall + precision)$. Since our major contribution is to separate background/foreground motions at trajectory level, thus our comparisons and analysis mainly focus on the first step: trajectory level labeling.

4.1 Evaluations on Synthetic Data

Experimental Settings. A grid of background points and four shapes are generated to simulate objects moving under camera movements. The homogeneous representation of each point is $(X, Y, Z, 1)$ as its position in the 3D world. Then the projected points (x, y) are obtained in a 2D image by $(x, y, 1)^T = C \cdot (X, Y, Z, 1)^T$, where C is a 3×4 camera projection matrix. The depth value

Z in the 3D world for foreground shapes and background grid is 10±5 and 20±10, respectively. The foreground shapes move and the background grid stays still. Changing the camera projection matrix C simulates the camera movement and generates projected images.

Group Sparsity Constraint versus Standard Sparsity Constraint. We compare the performance between the group sparsity constraint ($L_{2,0}$ norm) and *Std. sparse* constraint (L_0 norm). The sparsity constraint aims to find a sparse set of nonzero elements, which is $\|F\|_0 < \alpha k \times 2l$ in our problem. $k \times 2l$ denotes the total number of nonzero elements. It is equivalent to the total number of nonzero elements in the group sparsity constraint. Note that the formulation with this *Std. sparse* method is similar to Robust PCA method [2]. The difference is that we use it at the trajectory level instead of pixel level.

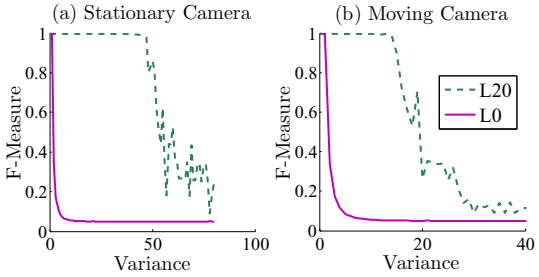

Fig. 3. Comparison between group sparsity constraint ($L_{2,0}$ norm) and *Std. sparse* constraint. The left is synthetic data simulated with a stationary camera, and the right is simulated with a moving camera.

Two sets of data are generated to evaluate the performance. One is simulated with a stationary camera, and the other is from a moving one. Foreground keeps moving in the whole video sequence. Random noise with variance v is added to both the foreground moving trajectories and camera projection matrix C. The performance is shown in Fig. 3. In the noiseless case (*i.e.* $v = 0$), the motion pattern from the foreground is distinct from the background in the whole video sequence. Thus each element on the foreground trajectories is different from the background element. Sparsity constraint produces the same perfect result as group sparsity constraint. When v goes up, the distinction of elements between foreground and background goes down. Thus some elements from the foreground may be recognized as background. On the contrary, the group sparsity constraint connects the elements in the neighboring frames. It treats the elements on one trajectory as one unit. Even some elements on this trajectories are similar to the background, the distinction along the whole trajectory is still large from the background. As shown in Fig. 3, using the group sparsity constraint is more robust than using the sparsity constraint when variance increases.

4.2 Evaluations on Videos

Experimental Settings. We test our algorithm on publicly available videos from various sourcese. One video source is provided by Sand and Teller [32] (refer this as ST sequences). ST sequences are recorded with hand held cameras, both indoors and outdoors, containing a variety of non-rigidly deforming objects

(hands, faces and bodies). They are high resolution images with large frame-to-frame motion and significant parallax. Another source of videos is provided from Hopkins 155 dataset [33], witch has two or three motions from indoor and outdoor scenes. These sequences contain degenerate and non-degenerate motions, independent and partially dependent motions, articulated and non-rigid motions. We also test our algorithm on a typical video for stationary camera: "*truck*". The trajectories in these sequences were created using an off-the-shelf dense particle tracker [25]. We manually label the pixels into foreground/background (binary maps). If a trajectory falls into the foreground area, it is considered as foreground, and verse versa.

Handling Stationary and Moving Cameras. We first demonstrate that our approach handles both stationary cameras and moving cameras automatically in a unified framework, by using the General Rank constraint (GR) instead of the Fixed Rank constraint (FR). We use two videos to show the difference. One is "*VHand*" from a moving camera ($rank(B) = 3$), and the other is "*truck*" captured by stationary camera ($rank(B) = 2$). We use the distribution of L_2 norms of estimated foreground trajectories (*i.e.*, $\|\hat{F}_i\|_2, i \in 1, 2, ..., k$) to show how well background and foreground is separated in our model. For a good separation result, F should be well estimated. Thus $\|\hat{F}_i\|_2$ is large for foreground trajectories and small for background ones. In other words, its distribution has an obvious difference between the foreground region and the background region (see examples in Fig. 4).

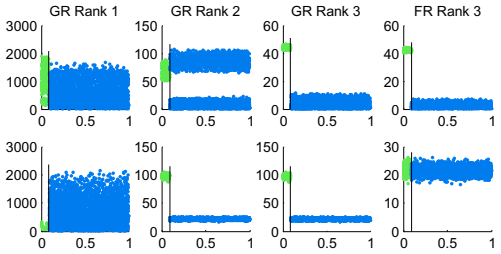

Fig. 4. $\|\hat{F}_i\|_2$ distribution of the GR and the FR model. Green means foreground and blue means background. Separation means good result. Top row is from moving camera, and bottom row is from stationary camera. The four columns are GR−1, GR−2, GR−3 and FR.

We use GR-$i, i \in 1, 2, 3$ to denote the optimization iteration on each rank value. $\|\hat{F}_i\|_2$ of each specific rank iteration is plotted in Fig. 4. The GR method works for both cases. When the rank of B is 3 (the first row of Fig. 4), the FR model also finds a good solution, since rank-3 perfectly fits the FR model. However, the FR constraint fails when the rank of B is 2, where the distribution of $\|\hat{F}_i\|_2$ between B and F are mixed together. On the other hand, GR−2 can handle this well, since the data perfectly fits the constraint. On GR−3 stage, it uses the result from GR−2 as the initialization, thus the result on GR−3 still holds. The figure shows that the distribution of $\|\hat{F}_i\|_2$ from the two parts has been clearly separated in the third column of the bottom row. This experiment demonstrates that the GR model can handle more situations than the FR model. Since in real applications it is hard to know the specific rank value in advance, the GR model provides a more flexible way to find the right solution.

Table 1. Quantitative evaluation at trajectory level labeling

	RANSAC-b	*GPCA*	*LSA*	*RANSAC-m*	*Std. Sparse*	*Ours*
VPerson	0.786	0.648	0.912	0.656	0.616	**0.981**
VHand	0.952	0.932	0.909	0.930	0.132	**0.987**
VCars	0.867	0.316	0.145	0.276	0.706	**0.993**
cars2	0.750	0.773	0.568	0.958	0.625	**0.976**
cars5	0.985	0.376	0.054	0.637	0.779	**0.990**
people1	0.932	0.564	0.087	0.743	0.662	**0.955**
truck	0.351	0.368	0.140	0.363	0.794	**0.975**

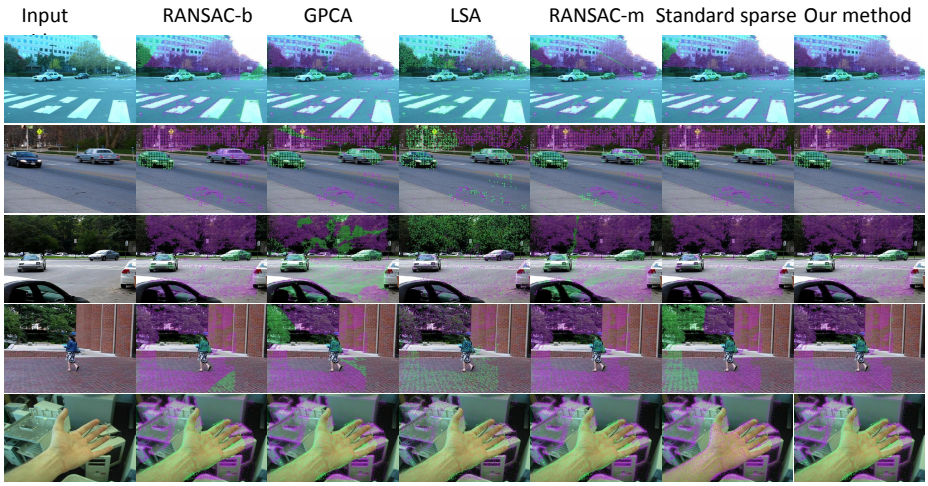

Fig. 5. Results at trajectory level labeling. Green: foreground; purple: background. From top to bottom, five videos are "*VCars*", "*cars2*", "*cars5*", "*people1*" and "*VHand*".

Performance Evaluation on Trajectory Labeling. We compare our method with four state-of-art algorithms: RANSAC-based background subtraction (referred as *RANSAC-b* here) [1], Generalized GPCA (*GPCA*) [34], Local Subspace Affinity (*LSA*) [18] and motion segmentation using RANSAC (*RANSAC-m*) [33]. *GPCA*, *LSA* and *RANSAC-m* are motion segmentation algorithms using subspace analysis for trajectories. The code of these algorithms are available online. When testing these methods, we use the same trajectories as for our own method. Since *LSA* method runs very slow when using trajectories more than 5000, we randomly sample 5000 trajectories for each test video. The three motion

segmentation algorithms ask for the number of regions to be given in advance. We provide the correct number of segments n, whereas our method does not need that. Motion segmentation methods separate trajectories into n segments. Here we treat the segment with the largest trajectory number as the background and rest as the foreground. For *RANSAC-b* method, two major parameters influence the performance: projection error threshold th and consensus percentage p. Inappropriate selection of parameters may result in failure of finding the correct result. In addition, as *RANSAC-b* randomly selects three trajectories in each round, it may end up with finding a subspace spanned by part of foreground and background. The result it generates is not stable. Running the algorithm multiple times may give different separation of background and foreground, which is undesirable. In order to have a fair comparison with it, we grid search the best parameter set over all teste videos and report the performance under the optimal parameters.

The quantitative and qualitative results on the trajectory level separation is shown in Fig. 5 and Tab. 1, respectively. Our method works well for the test videos. Take "*cars5*" for example. *GPCA* and *LSA* misclassify some trajectories. *RANSAC-b* randomly selects three trajectories to build the scene motion. On this frame, the three random trajectories all lie in the middle region. The background model built from these 3 trajectories do not cover the left and right region of the scene, thus the left and right regions are misclassified as foreground. *RANSAC-m* produces similar behavior to *RANSAC-b*. *Std. sparse* method does not have any group constraint in the consecutive frames, thus some trajectories are classified as foreground in one frame, and classified as background in the next frame. Note that the quantitative results are obtained by averaging on all frames over 50 iterations. Fig. 5 only shows performance on one frame, which may not reflect the overall performance shown in Tab. 1.

Performance Evaluation at Pixel Level Labeling. We also evaluate the performance at the pixel level using four methods: *RANSAC-b* [1], *MoG* [11], *Std. sparse* method and the proposed method. *GPCA*, *LSA*, *RANSAC-m* are not evaluated in this part, since these three algorithms do not provide pixel level labeling. Due to space limitations, one typical video "*VPerson*" is shown here in Fig. 6.

MoG does not perform well, because a statistical model of *MoG* is built on pixels of fixed positions over multiple frames, but background objects do not stay in the fixed positions under moving cameras. *RANSAC-b* can accurately label

Fig. 6. Quantitative results at pixel level labeling on *"VPerson"* sequence

pixels if the trajectories are well classified. However, it is also possible to build a wrong background subspace because *RANSAC-b* is not stable and sensitive to parameter selection. Our method can robustly handle these diverse types of data, due to our generalized low rank constraint and group sparsity constraint. One limitation of our method is that it classifies the shadow as part of the foreground (*e.g.*, on the left side of the person in "*VPerson*" video). This could be further refined by using shadow detection/removal techniques [16].

5 Conclusions and Future Work

In this paper, we propose an effective approach to do background subtraction for complex videos by decomposing the motion trajectory matrix into a low rank one and a group sparsity one. Then the information from these trajectories is used to further label foreground at the pixel level. Extensive experiments are conducted on both synthetic data and real videos to show the benefits of our model. The low rank and group sparsity constraints make the model robust to noise and handle diverse types of videos.

Our method depends on trajectory-tracking technique, which is also an active research area in computer vision. When the tracking technique fails, our method may not work well. A robust way is to build the tracking errors into the optimization formulation. We will investigate it in our future work.

References

1. Sheikh, Y., Javed, O., Kanade, T.: Background subtraction for freely moving cameras. In: ICCV (2009)
2. Candes, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of ACM (2011)
3. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journals of the Royal Statistical Society (2006)
4. Candes, E., Tao, T.: Near-optimal signal recovery from random projections: Universal encoding strategies? TIT (2006)
5. Starck, J.L., Elad, M., Donoho, D.: Image decomposition via the combination of sparse representations and a variational approach. TIP (2005)
6. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. IJCV (1992)
7. Brutzer, S., Hoferlin, B., Heidemann, G.: Evaluation of background subtraction techniques for video surveillance. In: CVPR (2011)
8. Jain, R., Nagel, H.: On the analysis of accumulative difference pictures from image sequences of real world scenes. TPAMI (2009)
9. Haritaoglu, I., Harwood, D., Davis, L.: W4: real-time surveillance of people and their activities. TPAMI (2000)
10. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: Real-time tracking of the human body. TPAMI (2002)
11. Stauffer, C., Grimson, W.: Learning patterns of activity using real-time tracking. TPAMI (2000)
12. Elgammal, A., Duraiswami, R., Harwood, D., Davis, L.: Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proceedings of the IEEE (2002)

13. Sheikh, Y., Shah, M.: Bayesian object detection in dynamic scenes. In: CVPR (2005)
14. Monnet, A., Mittal, A., Paragios, N., Ramesh, V.: Background modeling and subtraction of dynamic scenes. In: ICCV (2003)
15. Zhong, J., Sclaroff, S.: Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In: ICCV (2003)
16. Liao, V.S., Zhao, G., Kellokumpu, Pietikainen, M., Li, S.Z.: Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In: CVPR (2010)
17. Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. TPAMI (2010)
18. Yan, J., Pollefeys, M.: A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part IV. LNCS, vol. 3954, pp. 94–106. Springer, Heidelberg (2006)
19. Hayman, E., Eklundh, J.-O.: Statistical background subtraction for a mobile observer. In: ICCV (2003)
20. Ren, Y., Chua, C., Ho, Y.: Statistical background modeling for non-stationary camera. PR Letters (2003)
21. Yuan, C., Medioni, G., Kang, J., Cohen, I.: Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. TPAMI (2007)
22. Kwak, S., Lim, T., Nam, W., Han, B., Han, J.H.: Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering. In: ICCV (2011)
23. Huang, J., Zhang, T.: The benefit of group sparsity. The Annals of Statistics (2010)
24. Huang, J., Huang, X., Metaxas, D.: Learning with dynamic group sparsity. In: ICCV, pp. 64–71 (2009)
25. Sundaram, N., Brox, T., Keutzer, K.: Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 438–451. Springer, Heidelberg (2010)
26. Liu, C.: Beyond pixels: Exploring new representations and applications for motion analysis. Doctoral thesis, Massachusetts Institute of Technology (2009)
27. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. TPAMI (2001)
28. Zhang, S., Huang, J., Huang, Y., Yu, Y., Li, H., Metaxas, D.: Automatic image annotation using group sparsity. In: CVPR (2010)
29. Liu, G., Lin, Z., Yu, Y.: Robust subspace segmentation by low-rank representation. In: ICML (2010)
30. Zhang, Z., Liang, X., Ma, Y.: Unwrapping low-rank textures on generalized cylindrical surfaces. In: ICCV (2011)
31. Mu, Y., Dong, J., Yuan, X., Yan, S.: Accelerated low-rank visual recovery by random projection. In: CVPR (2011)
32. Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. In: CVPR (2006)
33. Tron, R., Vidal, R.: A benchmark for the comparison of 3-D motion segmentation algorithms. In: CVPR (2007)
34. R., Vidal, Y.M., Sastry, S.: Generalized principal component analysis (GPCA). In: CVPR (2003)