9

Sparsity Driven Background Modeling and Foreground Detection

Junzhou Huang
University of Texas, Arlington, USA

Chen Chen
University of Texas, Arlington, USA

Xinyi Cui
Facebook, USA

9.1 Introduction

Background subtraction is an important pre-processing step in video monitoring applications. It aims to extract foreground objects (*e.g.* humans, cars, text etc.) in videos from static and moving cameras for further processing (object recognition etc.).

A robust background subtraction algorithm should be able to handle lighting changes, repetitive motions from clutter and long-term scene changes [39]. The earliest background subtraction methods use frame difference to detect foreground [19]. Many subsequent approaches have been proposed to model the uncertainty in background appearance. The Mixture of Gaussians (MoG) background model assumes the color evolution of each pixel can be modeled as a MoG and are widely used on realistic scenes [35]. Elgammal et al. [12] proposed a non-parametric model for the background under similar computational constraints as the MoG. Spatial constraints are also incorporated into their model. Sheikh and Shah consider both temporal and spatial constraints in a Bayesian framework [33], which results in good foreground segmentations even when the background is dynamic. The model in [25] also uses a similar scheme. All these methods only implicitly model the background dynamics. In order to better handle dynamic scenes, some recent works [26, 52] explicitly model the background as dynamic textures. Most dynamic texture modeling methods are based on the Auto Regressive and Moving Average (ARMA) model, whose dynamics are driven by a linear dynamic system (LDS). While this linear model can handle background dynamics with certain stationarity, it will cause over-fitting for more complex scenes.

Recently, Huang et al. presented a new method for modeling foreground in dynamic scenes using dynamic group sparsity [16]. It is assumed that the pixels belong to the foreground should be sparse, and more importantly, cluster together. They extend the CS theory to efficiently handle data with both sparsity and dynamic group clustering priors. A dynamic group sparsity recovery algorithm is then proposed based on the extended CS theory. This algorithm iteratively prunes the signal estimations according to both sparsity and group clustering priors. The group clustering trend implies that, if a pixel lives in the foreground, its neighboring pixels would also tend to live in this foreground with higher probability, and vice versa. By enforcing these constraints, the degrees of freedom of the sparse solutions have been significantly reduced to a narrower union of subspaces. It leads to several advantages: 1) accelerating the result pruning process; 2) decreasing the minimal number of necessary measurements; and 3) improving robustness to noise and preventing the recovered foreground from having artifacts. These advantages enable the algorithm to efficiently obtain stable background subtraction even when the background is dynamic.

So far, all the methods mentioned assume the cameras are fixed and stationary. In recent years, videos from moving cameras have also been studied [32] because the number of moving cameras (such as smart phones and digital videos) increased significantly. The research for moving cameras has recently attracted people's attention. Motion segmentation approaches [29,47] 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 [14, 30], assuming a dominant plane [48], etc. In [22], a method is proposed to use belief propagation and Bayesian filtering to handle moving cameras. However, handling diverse types of videos robustly is still a challenging problem.

For sparsity based methods, Cui et al. propose a unified framework for background subtraction [8], which can robustly deal with videos from stationary or moving cameras with various number of rigid/non-rigid objects. This method is based on two sparsity constraints applied on foreground and background levels, *i.e.*, low rank [7] and group sparsity constraints [49]. It is inspired by recently proposed sparsity theories [6, 34]. There are two "sparsity" observations behind this method. *First*, when the scene in a video does not have any foreground moving objects, video motion has a low rank constraint for orthographic cameras [21]. 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. Cui et. al.'s 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 demonstrate this method compared favorably to the recent state-of-the-art methods.

The remainder of the chapter is organized as follows. Section 9.2 briefly reviews the sparsity theory. In Section 9.3, we review the dynamic group sparsity model for background subtraction. In Section 9.4, we review the background subtraction using low rank and group sparsity constraints, which could be used for both stationary and moving cameras. We conclude this chapter in Section 9.5.

9.2 Sparsity Techniques

Suppose I_t is a frame in a video at time t, in ideal conditions, it can be represented as the combination of background and an object in foreground:

$$I_t = b + f \tag{9.1}$$

where $b \in \mathbb{R}^{n\times 1}$ denotes the background and $f \in \mathbb{R}^{n\times 1}$ denotes the foreground. An example is shown in Figure 9.1. We could observe that the foreground f is sparse, with most of it values are zeros. In this section, we review the sparsity techniques and the benefit when apply them to background subtraction.

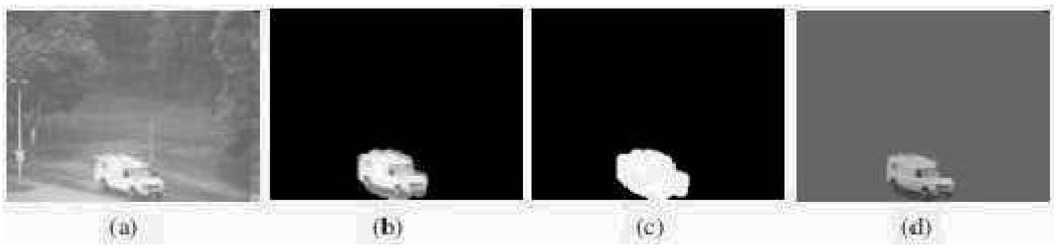

FIGURE 9.1 An example of background subtraction: (a) one video frame, (b) the foreground image, (c) the foreground mask and (d) the background subtracted image with AdaDGS [16].

9.2.1 Standard Sparsity

Sparsity based methods have been very popular since the emerging of compressive sensing (CS) [4, 11]. In CS, the capture of a sparse signal and compression are integrated into a single process. A signal $x \in \mathbb{R}^n$ is k-sparse if only $k \ll n$ entries of x are nonzero. We call the set of indices corresponding to the nonzero entries the support of x and denote it by $supp(x)$ [1,18]. Thus, we do not capture a sparse signal x directly but rather capture $m < n$ linear measurements $y = \Phi x$ based on a measurement matrix $\Phi \in \mathbb{R}^{m\times n}$. Suppose the set of k-sparse signal x lives in the union Ω_k of k-dimensional subspaces, the union Ω_k thus includes C_n^k subspaces. To stably recover the k-sparse signal x from m measurements, the measurement matrix Φ is required to satisfy the Restricted Isometry Property (RIP) [4]. CS result shows that, if $m = \mathcal{O}(k + k\log(n/k))$, then measurement matrix Φ whose entries are i.i.d. Gaussian, Bernoulli (± 1) can satisfy the RIP with high probability.

9.2.2 Group Sparsity

In CS, the assumption cares the sparsity number k of the data, while positions of such k non-zeros entries can be random. There is no prior information utilized other than sparsity. Background subtracted images are typical structured sparsity data in static video surveillance applications. They generally correspond to the foreground objects of interest. Unlike the whole scene, these images are not only spatially sparse but also inclined to cluster into groups, which correspond to different foreground objects. The success of sparse recovery in compressive sensing motivates us to further improve background substraction by utilizing such prior information. Compared with that of standard sparse data, the locations of non-zeros entries in grouped sparse data should be more fixed. Intuitively, the measurement number bound may be further reduced or the solution will be more accurate with the number of measurements.

Similar to the definition of k-sparse data, we can define dynamic group sparse (DGS) data as follow:

Definition:($G_{k,q}$-sparse data) *A data $x \in \mathbb{R}^n$ is defined as the dynamic group sparse data ($G_{k,q}$-sparse data) if it can be well approximated using $k \ll n$ nonzero coefficients under some linear transforms and these k nonzero coefficients are clustered into $q \in \{1, \cdots, k\}$ groups.*

From this definition, we can know that $G_{k,q}$-sparse data only requires that the nonzero coefficients in the sparse data have the group clustering trend and does not require to know any information about the group size and location. Although the information may be helpful, it is not necessarily required. In the following, it will be further illustrated that the group number q is also not necessary to be known. The group structures can be dynamic and unknown. We can find that nonzero coefficients are not randomly distributed but clustered spatially in the background subtracted image (Figure 9.1 (b)) and the foreground mask (Figure 9.1 (c)). More specially for color images, the R, G and B channels of the background subtracted image share a common support set although the nonzero coefficients are spatially clustered in each channel respectively.

Due to the additional dynamic clustering prior, the union of subspaces containing $G_{k,q}$-sparse data does not span all k-dimensional subspaces of the union Ω_k as in the conventional CS [5, 9, 11, 28, 41]. The former is far narrower than the latter in most cases. The dynamic group clustering prior significantly reduces the degrees of freedom of the sparse signal since it only permits certain combinations of its support set rather than all random combinations. This will make it possible for us to decrease the minimal measurement number m for stable recovery.

Structured sparsity theories [1, 18] show that the number of measurements required for robustly recovering dynamic group sparsity data is $m = \mathcal{O}(k + q\log(n/q))$, which is a significant improvement over the $m = \mathcal{O}(k + k\log(n/k))$ that would be required by conventional CS recovery algorithms [5, 11, 28, 41]. While the group number q is smaller, more improvements can be obtained. While q is far smaller than k and k is close to $log(n)$, we can get $m = \mathcal{O}(k)$. Note that, this is a sufficient condition. If we know more priors about group settings, we can further reduce this bound.

9.3 Sparsity Driven Background Modeling and Foreground Detection for Stationary Cameras

The inspiration for the AdaDGS background subtraction came from the success in online DT video registration based on the sparse representation constancy assumption (SRCA) [15]. The SRCA states that a new coming video frame should be represented as a linear combination of as few preceding image frames as possible. As a matter of fact, the traditional brightness constancy assumption seeks that the current video frame can be best represented by a single preceding frame, while the SRCA seeks that the current frame can be best sparsely represented by all preceding image frames. Thus, the former can be thought as a special case of SRCA.

9.3.1 Group Sparsity Based Model

Suppose a video sequence consists of frames $I_1, ..., I_n \in \mathbb{R}^m$. Without loss of generality, we can assume that background subtraction has already been performed on the first t frames. Let $A = [I_1, ..., I_t] \in \mathbb{R}^{m \times t}$. Denote the background image and the background subtracted image by b and f, respectively, for I_{t+1}. From the introduction in a previous section, we know that f is dynamic group sparse data with unknown sparsity number k_f and group structure. According to SRCA, we have $b = Ax$, where $x \in \mathbb{R}^t$ should be k_x-sparse vector and $k_x << t$. Let $\Phi = [A, I] \in \mathbb{R}^{m \times (t+m)}$, where $I \in \mathbb{R}^{m \times m}$ is an identity matrix. Then, we have:

$$I_{t+1} = Ax + f = [A, I] \begin{bmatrix} x \\ f \end{bmatrix} = \Phi z \tag{9.2}$$

where $z \in \mathbb{R}^{t+m}$ is the DGS data with unknown sparsity $k_x + k_f$. Background subtraction is thus formulated as the following sparse recovery problem:

$$(x_0, f_0) = argmin \|z\|_0, \quad \|I_{t+1} - \Phi z\|^2 < \varepsilon \tag{9.3}$$

9.3.2 Algorithm

Although problem (9.3) can be solved by conventional methods for standard sparsity, the benefit of group sparsity equips us to propose a new recovery algorithm for dynamic group sparse data z, namely dynamic group sparsity (DGS) recovery algorithm. We demonstrate how to seamlessly integrate the dynamic group clustering prior into previous framework [9, 28]. The algorithm includes five main steps in each iteration: 1) pruning the residue estimation; 2) merging the support sets; 3) estimating the signal by least square; 4) pruning the signal estimation and 5) updating the signal/residue estimation and support set. One can observe that it is similar to that of SP/CoSaMP algorithms [9, 28]. The difference only exists in the pruning process in step 1 and step 4. The modification is simple. We prune the estimation in the step 1 and step 4 using DGS approximation pruning rather than k-sparse approximation, as we only need to search over subspaces of $\mathcal{A}_{k,q}$ instead of C_n^k subspaces of Ω_k. It directly leads to fewer measurement requirement for stable data recovery.

The DGS pruning algorithm is described in algorithm 9.1. There exist two prior-dependent parameters J_y and J_b. J_y is the number of tasks if the problem can be represented as a multi-task CS problem [20]. J_b is the block size if the interested problem can be modelled as a block sparsity problem [37, 42, 43, 49]. Their default values are set as 1, which is the case of traditional sparse recovery in compressive sensing. Moreover, there are two important user-tuning parameters, the weight w of neighbors and the neighbor number τ of each element in sparse data. In practice, it is very straightforward to adjust them since they have the physical meanings. The first one controls the balance between the sparsity prior and the group clustering prior. While w is smaller/bigger, it means that the degree of dynamic group clustering is lower/higher in the sparse signal. Generally, they are set as $0.5's$ if there is not more knowledge about that in practice. The parameter τ controls the number of neighbors that can be affected by each element in sparse data. Generally, it is good enough to set it as 2, 4 and 6 for 1D, 2D and 3D data respectively.

Up to now, we assume that we know the sparsity number k of the sparse data before recovery. However, it is not always true in practical applications. For example, we do not

Algorithm 9.1 - DGS Approximation Pruning

```
Input: $x \in \mathbb{R}^n$ {estimations}; $k$ {the sparsity number}; $J_y$ {task number}; $J_b$ {block size}; $N_x \in \mathbb{R}^{n\times\tau}$ {values of x's neighbors}; $w \in \mathbb{R}^{n\times\tau}$ {weights for neighbors}; $\tau$ {neighbor number}
$J_x = J_y J_b$;$x \in \mathbb{R}^n$ is shaped to $x \in \mathbb{R}^{\frac{n}{J_x}\times J_x}$
$N_x \in \mathbb{R}^{n\times\tau}$ is shape to $N_x \in \mathbb{R}^{\frac{n}{J_x}\times J_x\times\tau}$;
for all $i = 1, ..., \frac{n}{J_x}$ do
  Combing each entry with its neighbors
  $$z(i) = \sum_{j=1}^{J_x} x^2(i,j) + \sum_{j=1}^{J_x}\sum_{t=1}^{\tau} w^2(i,t)N_x^2(i,j,t)$$
end for
$\Omega \in \mathbb{R}^{\frac{n}{J_x}\times 1}$ is set as indices corresponding to the largest $k/J_x$ entries of $z$
for all $j = 1, ..., J_x$ do
  for all $i = 1, ..., \frac{k}{J_x}$ do
    Obtain the final list
    $\Gamma((j-1)\frac{k}{J_x} + i) = (j-1)\frac{k}{J_x} + \Omega(i)$
  end for
end for
Output: $supp(x,k) \leftarrow \Gamma$
```

know the exact sparsity numbers of the background subtracted images although we know they tend to be dynamic group sparse. Motivated by the idea in [10], a new recovery algorithm called AdaDGS is developed by incorporating an adaptive sparsity scheme into the above DGS recovery algorithm.

Suppose the range of the sparsity number is known to be $[k_{min}, k_{max}]$. We can set the step size of sparsity number as $\triangle k$. The whole recovery process is divided into several stages, each of which includes several iterations. Thus, there are two loops in AdaDGS recovery algorithm. The sparsity number is initialized as k_{min} before iterations. During each stage (inner loop), we iteratively optimize sparse data with the fixed sparsity number k_{curr} until the halting condition within the stage is true (for example, the residue norm is not decreasing). We then switch to the next stage after adding $\triangle k$ into the current sparsity number k_{curr} (outer loop). The whole iterative process will stop whenever the halting condition is satisfied. For practical applications, there is a trade-off between the sparsity step size $\triangle k$ and the recovery performance. Smaller step sizes require more iterations and bigger step size may cause inaccuracy. The sparsity range depends on the applications. Generally, it can be set as $[1, n/3]$, where n is the dimension of the sparse data. Algorithm 9.2 describes the AdaDGS recovery algorithm.

With the AdaDGS recovery algorithm, problem (9.3) can be efficiently solved. Similar ideas are used for face recognition robust to occlusion [46]. It is worth mentioning that the coefficients in w corresponding to the x part are randomly sparse while those corresponding to f are dynamic group sparse. During the DGS approximation pruning, we thus can set those coefficients in weight w for the x-related part as zeros and those for f as nonzeros. Since we do not know the sparsity number k_x and k_f, we can set sparsity ranges for them respectively and run the AdaDGS recovery algorithm until the halting condition is true. Then, we can obtain the optimized background subtracted image f and background image $b = Ax$. For long video sequences, it is impractical to build a model matrix $A = [I_1, ..., I_t] \in$

Algorithm 9.2 - AdaDGS Recovery [16]

```
1: Input: $\Phi \in \mathbb{R}^{m\times n}$ {sample matrix}; $y \in \mathbb{R}^{m}$ {sample vector};$[k_{min}, k_{max}]$ {sparsity range}; $\triangle k$ {sparsity step size}
2: Initialization: residue $y_r = y$; $\Gamma = supp(x) = \varnothing$; sparse data $x = 0$; sparsity number $k = k_{min}$
3: repeat
4:    Perform DGS recovery algorithm with sparsity number $k$ to obtain $x$ and the residue
5:    if halting criterion false then
6:       Update $\Gamma$, $y_r$ and $k = k + \triangle k$
7:    end if
8: until halting criterion true
9: Output: $x = \Phi_{\Gamma}^{\dagger} y$
```

$\mathbb{R}^{m\times t}$, where t denotes the last frame number. In order to cope with this case, we can set a time window width parameter τ. We then build the model matrix, $A = [I_{t-\tau+1}, ..., I_t] \in \mathbb{R}^{m\times(t-\tau)}$, for the $(t+1)$ frame, which can avoid the memory requirement blast for a long video sequence. The complete algorithm for AdaDGS based background subtraction is summarized in Algorithm 9.3. As we know, the sparsity number must be provided in most of current recovery algorithms, which make them impractical for this problem. In contrast, the AdaDGS can apply well to this task since it not only can automatically learn the sparsity number and group structures but also is a fast enough greedy algorithm.

Algorithm 9.3 - AdaDGS Background Subtraction [16]

```
1: Input: The video sequence $I_1, ..., I_n$, the number $t$ which means $1^{st} \sim t^{th}$ have been performed background subtraction, the time window width $\tau \leq t$
2: for all $j = t + 1, ..., n$ do
3:    Set $A = [I_{j-\tau}, ..., I_{j-1}]$ and form $\Phi = [A, I]$
4:    Set $y = I_j$ and the sparsity ranges/step-sizes
5:    $(x_0, f_0) = AdaDGS(\Phi, y)$
6: end for
7: Output: Background subtracted images
```

9.3.3 Evaluations

The experiment is designed to validate the advantage of the AdaDGS model. The AdaDGS algorithm is tested on Zhong's dataset [52]. The background subtracted images can be directly obtained with the AdaDGS. The corresponding binary mask of these images are obtained with the simple threshold. The Zhong's results with robust Kalman model are also shown for comparisons. Figure 9.2 shows the results. Note that all results with AdaDGS are not post-processed with morphological operations and the results are directly the solutions of the optimization problem in Equation 9.3. It is clear that our AdaDGS produces clean background subtracted images, which shows the advantages of the DGS model. Figure 9.3 and Figure 9.4 show the background subtraction results on two other videos [12, 26]. Results of AdaDGS without postprocessing can compete with others with postprocessing. The results show the AdaDGS model can handle well highly dynamic scenes by exploiting the effective sparsity optimization scheme.

All experimental results show the AdaDGS algorithm gains marked improvement over

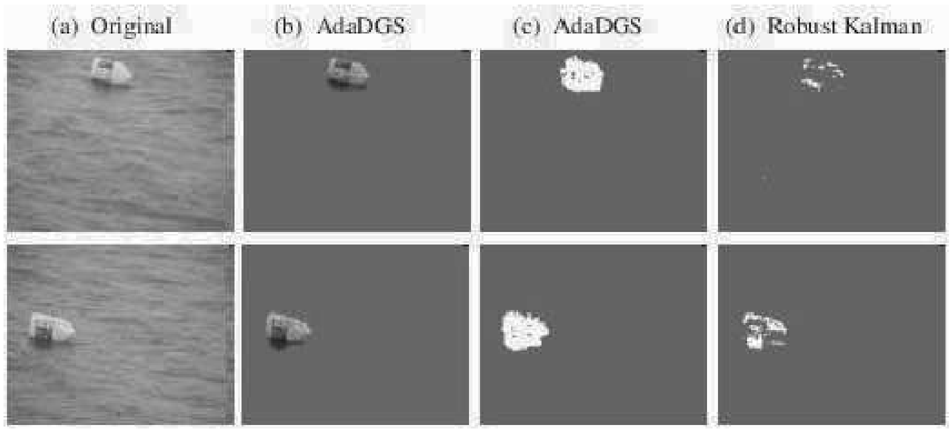

FIGURE 9.2 Results on the Zhong's dataset (a) original frame, (b) background subtracted image with AdaDGS [16], (c) the binary mask with AdaDGS [16] and d) with robust Kalman model [52].

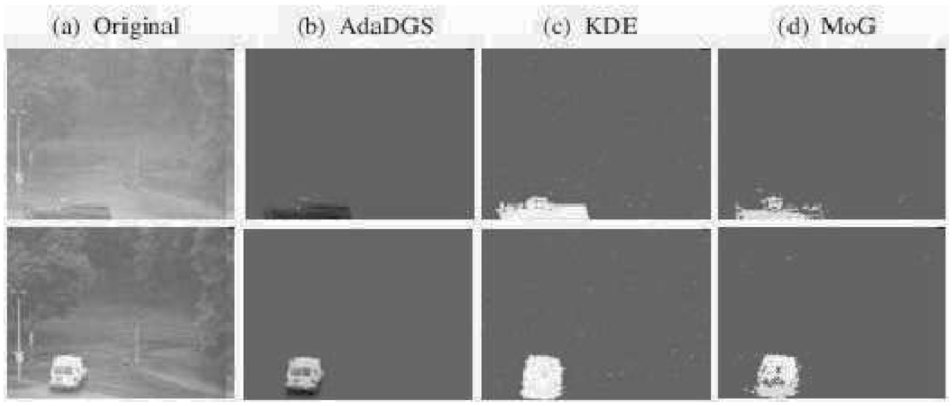

FIGURE 9.3 Results on the Elgammal's dataset. (a) original frame, (b) with AdaDGS [16], (c) with KDE model [12] (d) with MoG [35].

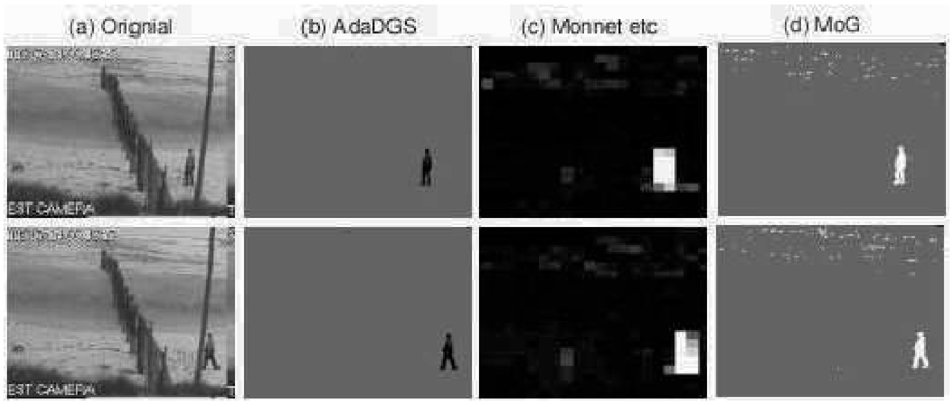

FIGURE 9.4 Results on Monnet's dataset. (a) original frame, (b) with AdaDGS [16], (c) with Monnet's method [26] and (d) with MoG [35].

previous algorithms when DGS priors are available. From a practical perspective, the DGS/AdaDGS can recover DGS data with higher accuracy and lower computational complexity from fewer measurements. From a theoretical point of view, structured sparsity theories [1, 18] offer a stronger guarantee for DGS/AdaDGS to achieve stable recovery. Moreover, we review a generalized framework for priors-driven background substraction algorithm. Group structure and sparsity number are not must-knows for this algorithm, which make it flexible and applicable in many practical applications.

9.4 Sparsity Driven Background Modeling and Foreground Detection for Moving Cameras

Besides the topic of background substraction for stationary cameras in the previous section, the research for moving cameras has recently attracted people's attention. Motion segmentation approaches [29, 47] 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 [14,30], assuming a dominant plane [48], etc. Kwak et al. [22] proposed a method to use belief propagation and Bayesian filtering to handle moving cameras. Different from their work, long-term trajectories we introduce here encode more information for background subtraction. Recently, [32] has 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 [16,17] and low rank constraint [7] 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 these works, we review a LRGS (low rank and group sparsity) method [8] which is based on constraints in temporal domain and analyzing the trajectory properties.

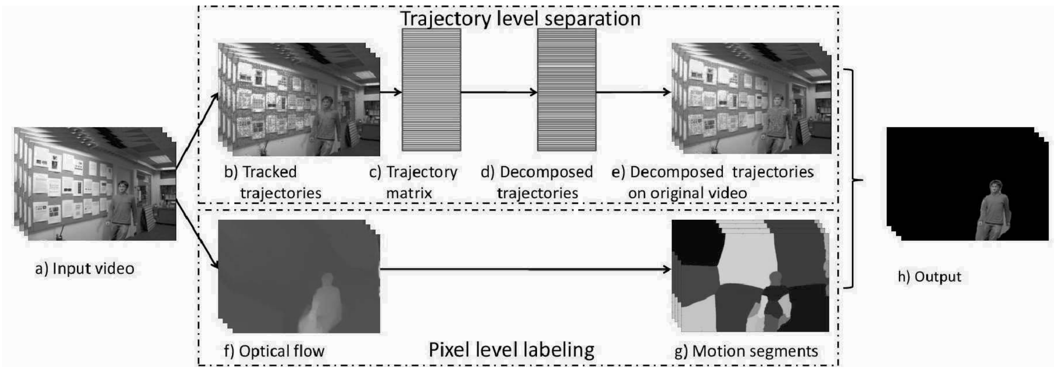

FIGURE 9.5 (See color insert.) The framework. LRGS method takes a raw video sequence as input, and produces a binary labeling as the output. Two major steps are trajectory level separation and pixel level labeling.

This background subtraction algorithm takes a raw video sequence as input, and generates a binary labeling at the pixel level. Fig. 9.5 shows the 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. An off-the-shelf dense point tracker [38] is used 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 [13] and graph cuts [2]. Then the color and motion information gathered from the recognized trajectories builds statistics to label motion segments as foreground or background.

9.4.1 Low Rank and Group Sparsity Based Model

Different from the dynamic group sparsity method in the previous section, the LRGS method for moving cameras does not process the image pixels directly. Background substraction are performed on the decomposition of the tracked trajectories, and then pixel level labeling is based on the separated trajectories. Tracking algorithms used in this framework are not of discussion scope in this chapter. Interested readers may refer to [36,38,45] for more details.

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. The 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:

$$\phi = B + F, \tag{9.4}$$

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. 9.4 is a severely under-constrained problem. It is difficult to find B and F without any prior information. In the LRGS method [8], two effective priors are incorporated 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 objects, video motion solely depends on the scene and the motion of the camera. The LRGS [8] 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 [21]. 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:

$$rank(B) \leq 3, \tag{9.5}$$

Another constraint has been used in the previous research work using RANSAC based method [32]. 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). The GR model in LRGS [8] method 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 a 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 (Section 9.4.5) provides more analysis on the effectiveness of the GR model when handling diverse types of 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:

$$\|F\|_{2,0} \leq \alpha k, \tag{9.6}$$

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 [17, 50]. 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 (Section 9.4.4), 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 LRGS model [8].

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

$$\begin{aligned}\left(\hat{B}, \hat{F}\right) &= \arg\min_{B,F} \left(\| \phi - B - F \|_F^2\right), \\ \text{s.t. } rank(B) &\leq 3, \ \| F \|_{2,0} < \alpha k,\end{aligned} \tag{9.7}$$

where $\|\cdot\|_F$ is the Frobenius norm. This model leads to a good separation of foreground and background trajectories. Fig. 9.6 illustrates the model.

Eq. 9.7 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 datasets. LRGS 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 [24]. Low rank constraints and Robust PCA have been recently used to solve vision problems [7, 27, 51], including background subtraction at the pixel level [7]. It assumes that the stationary scenes satisfy a low rank constraint. However, this assumption does not hold when the camera moves. Furthermore, that formulation does not consider any group information, which is an important constraint to make sure neighbor elements are considered together.

9.4.2 Optimization Framework

This subsection discusses how to effectively solve Eq. 9.7. The first challenge is that it is not a convex problem, because of the nonconvexity of the low rank constraint and the group

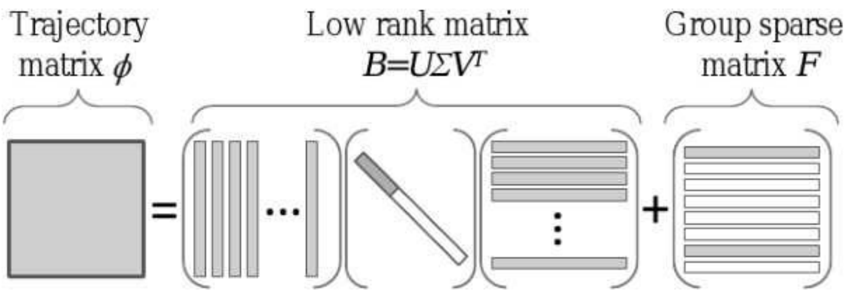

FIGURE 9.6 Illustration of our model. Trajectory matrix ϕ is decomposed into a background matrix B and a foreground matrix F. B is a low rank matrix, which only has a few nonzero eigenvalues (*i.e.* the diagonal elements of Σ in SVD); F is a group sparse matrix. Elements in one row belongs to the same group (either foreground or background), since they lie on one trajectory. The foreground rows are sparse compared to all rows. White color denotes zero values, while blue color denotes nonzero values.

sparsity constraint. Furthermore, we also need to simultaneously recover matrix B and F, which is generally a Chicken-and-Egg problem.

Alternating optimization and greedy methods are employed to solve this problem. We can 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. 9.7 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:

$$\left(\hat{F}\right) = \arg\min_{F} \left(\| \phi' - F \|_F^2\right), \text{ s.t. } \|F\|_{2,0} < \alpha k, \tag{9.8}$$

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

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

$$\left(\hat{B}\right) = \arg\min_{B} \left(\| \phi'' - B \|_F^2\right), \text{ s.t. } rank(B) = 3, \tag{9.9}$$

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

To initialize this optimization framework, we can simply choose $B_{init} = \phi$, and $F_{init} = \mathbf{0}$. Greedy methods are used to solve both subproblems. To solve Eq. 9.8, 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. 9.9. 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. 9.8 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 the 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, a multiple rank iteration method is used. 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 this 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. 9.9. 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 [7]. However, the 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.

9.4.3 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 [13] is calculated between two consecutive frames. Then motion segments are computed using graph cuts [2] 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.

To evaluate the performance of the LRGS algorithm, experiments on different data sources are conducted: 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 [3, 23]: $F = 2 \cdot recall \cdot precision/(recall + precision)$. The comparisons and analysis mainly focus on the first step: trajectory level labeling.

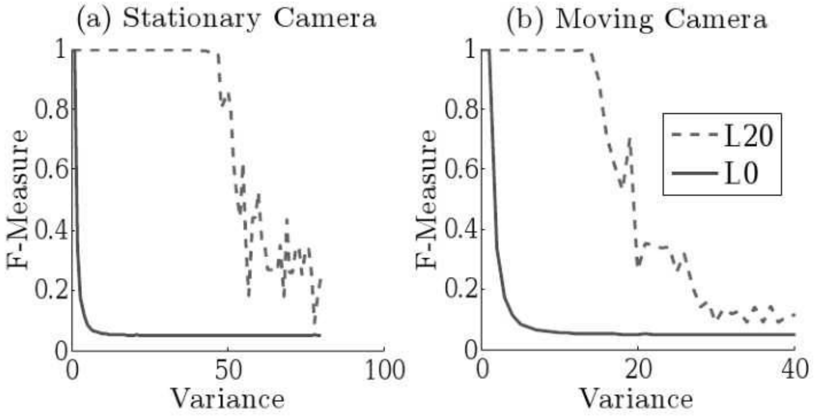

FIGURE 9.7 Comparison between group sparsity constraint ($L_{2,0}$ norm) and *Std. sparse* constraint. The left is synthetic data simulated with a stationary camera, and the right is simulated with a moving camera.

9.4.4 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. The performance between the group sparsity constraint ($L_{2,0}$ norm) and *Std. sparse* constraint (L_0 norm) is compared. The sparsity constraint aims to find a sparse set of nonzero elements, which is $\|F\|_0 < \alpha K \times 2l$ in this 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 [7]. The difference is that LRGS use it at the trajectory level instead of pixel level.

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. 9.7. 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 these trajectories are similar to the background, the distinction along the whole trajectory is still large from the background. As shown in Fig. 9.7, using the group sparsity constraint is more robust than using the sparsity constraint when variance increases.

9.4.5 Evaluations on Videos

Experimental settings. The LRGS algorithm is then tested on publicly available videos from various sources. One video source is provided by Sand and Teller [31] (refer to 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 [40], which 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. The algorithm is also tested on a typical video for stationary camera: "*truck*". The trajectories in these sequences were created using an off-the-shelf dense particle tracker [38]. The pixels are manually labeled into foreground/background (binary maps). If a trajectory falls into the foreground area, it is considered as foreground, and vice versa.

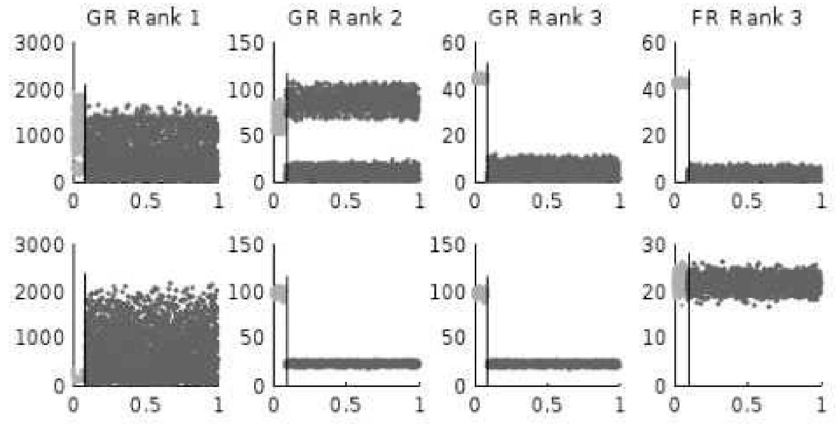

FIGURE 9.8 (See color insert.) $\|\hat{F}_i\|_2$ distribution of the GR and the FR model. Green means foreground and blue means background. Separation means good result. Top row is from moving camera, and bottom row is from stationary camera. The four columns are GR-1, GR-2, GR-3 and FR.

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 the LRGS 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. 9.8).

GR-$i, i \in 1, 2, 3$ is used to denote the optimization iteration on each rank value. $\|\hat{F}_i\|_2$ of each specific rank iteration is plotted in Fig. 9.8. The GR method works for both cases. When the rank of B is 3 (the first row of Fig. 9.8), 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 9.1 Quantitative evaluation at trajectory level labeling.

Sequence	RANSAC-b	GPCA	LSA	RANSAC-m	Std.sparse	LRGS
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**
VPerson	0.786	0.648	0.912	0.656	0.616	**0.981**
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**

Performance evaluation on trajectory labeling. We compare LRGS method [8] with four state-of-the-art algorithms: RANSAC-based background subtraction (referred as *RANSAC-b* here) [32], Generalized GPCA (*GPCA*) [44], Local Subspace Affinity (*LSA*) [47] and motion segmentation using RANSAC (*RANSAC-m*) [40]. *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, the same trajectories are used. Since *LSA* method runs very slow when using trajectories of 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. Motion segmentation methods separate trajectories into n segments. Here the LRGS method treats 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 test videos and report the performance under the optimal parameters.

The quantitative and qualitative results on the trajectory level separation is shown in Fig. 9.9 and Table 9.1, respectively. LRGS method [8] 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. 9.9 only shows performance on one frame, which may not reflect the overall performance shown in Table 9.1.

Performance evaluation at pixel level labeling. We also evaluate the performance at the pixel level using four methods: *RANSAC-b* [32], *MoG* [36], *Std. sparse* method and the LRGS method [8]. *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. 9.10 and the quantitative evaluation using F-measure is shown in Table 9.2.

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

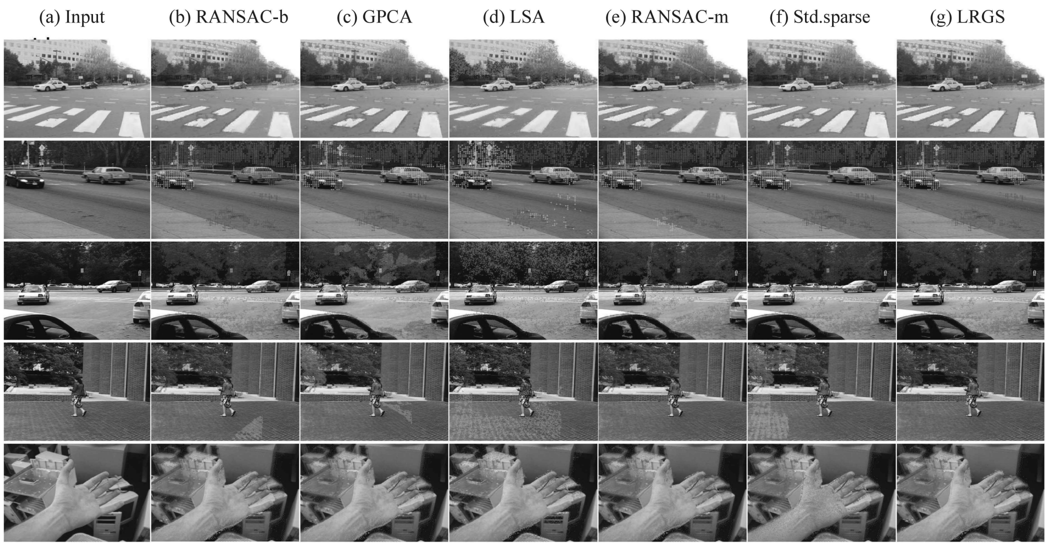

FIGURE 9.9 (See color insert.) Results at trajectory level labeling. Green: foreground; purple: background. From top to bottom, five videos are "*VCars*", "*cars2*", "*cars5*", "*people1*" and "*VHand*".

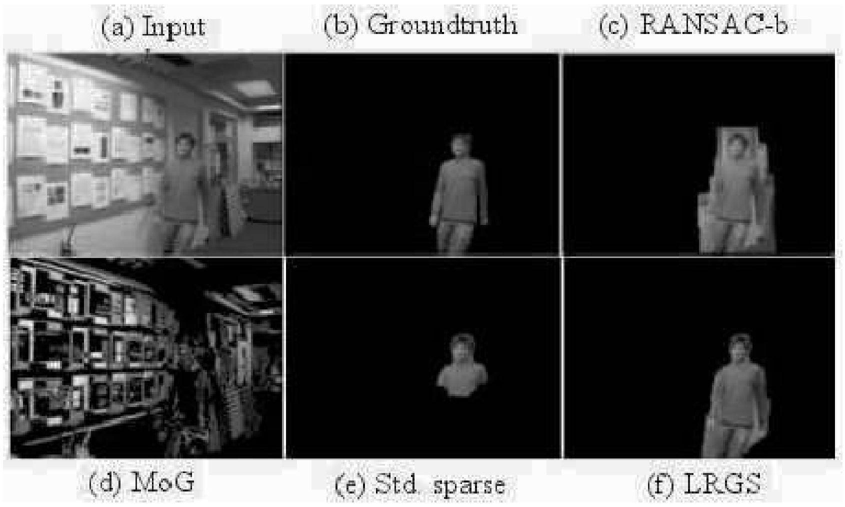

FIGURE 9.10 Quantitative results at pixel level labeling on "*VPerson*" sequence.

TABLE 9.2 Quantitative evaluation at pixel level labeling.

	RANSAC-b	MoG	Std.sparse	LRGS
VPerson	0.868	0.132	0.504	**0.892**

positions under moving cameras. *RANSAC-b* can accurately label 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. The LRGS method [8] can robustly handle these diverse types of data, due to the generalized low rank constraint and group sparsity constraint. One limitation of LRGS method [8] 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 [23].

9.5 Conclusion

In this chapter, we have reviewed sparsity based methods AdaDGS [16] and LRGS [8] for background substraction. The AdaDGS method is motivated by the group sparsity property of the pixel values on foreground images, that they are not only sparse but also clustered into groups. The LRGS method is developed by decomposing the motion trajectory matrix into a low rank one and a group sparsity one. The group sparsity constraint for AdaDGS is at spatial pixel level while LRGS is at trajectory level. In addition, the group setting in LRGS is known and has been utilized, which is unknown but not required in AdaDGS. Compared with AdaDGS, LRGS can be applied on videos from both stationary cameras and moving cameras. However, this method depends on trajectory-tracking technique, which is also an active research area in computer vision. When the tracking technique fails, the LRGS method may not work well. Benefit from group sparsity property, these two algorithms outperform all the other methods that are compared in this chapter.

References

1. R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.
2. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.
3. S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*, pages 1937–1944, 2011.
4. E. Candès. Compressive sampling. In *International Congress of Mathematicians*, pages 1433–1452, 2006.
5. E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.
6. E. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.
7. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *arXiv preprint arXiv:0912.3599*, 2009.
8. X. Cui, J. Huang, S. Zhang, and D. Metaxas. Background subtraction using low rank and group sparsity constraints. *European Conference on Computer Vision, ECCV 2012*, pages 612–625, 2012.
9. W. Dai, O. Milenkovic, and O. Olgica. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. Technical report, DTIC Document, 2008.
10. T. Do, L. Gan, N. Nguyen, and T. Tran. Sparsity adaptive matching pursuit algorithm

for practical compressed sensing. In *Proceedings of Asilomar Conference on Signals, Systems and Computers*, pages 581–587, 2008.

11. D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
12. A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *European Conference on Computer Vision, ECCV 2000*, pages 751–767, 2000.
13. C. Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis.* PhD thesis, Massachusetts Institute of Technology, 2009.
14. E. Hayman and J. Eklundh. Statistical background subtraction for a mobile observer. In *IEEE International Conference on Computer Vision, ICCV 2003*, pages 67–74, 2003.
15. J. Huang, X. Huang, and D. Metaxas. Simultaneous image transformation and sparse representation recovery. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008*, pages 1–8, 2008.
16. J. Huang, X. Huang, and D. Metaxas. Learning with dynamic group sparsity. In *IEEE International Conference on Computer Vision, ICCV 2009*, pages 64–71, 2009.
17. J. Huang and T. Zhang. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004, 2010.
18. J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12:3371–3412, 2011.
19. R. Jain and H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):206–214, 1979.
20. S. Ji, D. Dunson, and L. Carin. Multitask compressive sensing. *IEEE Transactions on Signal Processing*, 57(1):92–106, 2009.
21. T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.
22. S. Kwak, T. Lim, W. Nam, B. Han, and J. Han. Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering. In *IEEE International Conference on Computer Vision, ICCV 2011*, pages 2174–2181, 2011.
23. S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, and S. Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pages 1301–1306, 2010.
24. G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning, ICML 2010*, volume 3, 2010.
25. A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2004*, 2:II–302, 2004.
26. A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In *International Conference on Computer Vision, ICCV 2003*, pages 1305–1312, 2003.
27. Y. Mu, J. Dong, X. Yuan, and S. Yan. Accelerated low-rank visual recovery by random projection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*, pages 2609–2616, 2011.
28. D. Needell and J. Tropp. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.
29. S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010.
30. Y. Ren, C. Chua, and Y. Ho. Statistical background modeling for non-stationary camera. *Pattern Recognition Letters*, 24(1):183–196, 2003.

31. P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2006*, volume 2, pages 2195–2202, 2006.
32. Y. Sheikh, O. Javed, and T. Kanade. Background subtraction for freely moving cameras. In *IEEE International Conference on Computer Vision, ICCV 2009*, pages 1219–1225, 2009.
33. Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1778–1792, 2005.
34. J. Starck, M. Elad, and D. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE Transactions on Image Processing*, 14(10):1570–1582, 2005.
35. C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 1999*, volume 2, 1999.
36. C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.
37. M. Stojnic, F. Parvaresh, and B. Hassibi. On the reconstruction of block-sparse signals with an optimal number of measurements. *IEEE Transactions on Signal Processing*, 57(8):3075–3085, 2009.
38. N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. *European Conference on Computer Vision, ECCV 2010*, pages 438–451, 2010.
39. B. Tamersoy. Background subtraction- lecture notes. Technical report, University of Texas at Austin, 2009.
40. R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007*, pages 1–8, 2007.
41. J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.
42. J. Tropp, A. Gilbert, and M. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.
43. E. van den Berg, M. Schmidt, M. Friedlander, and K. Murphy. Group sparsity via linear-time projection. *Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada*, 2008.
44. R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.
45. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.
46. J. Wright, A. Yang, A. Ganesh, S. Sastry, S. Shankar, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.
47. J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. *European Conference on Computer Vision, ECCV 2006*, pages 94–106, 2006.
48. C. Yuan, G. Medioni, J. Kang, and I. Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1627–1641, 2007.
49. M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2005.

50. S. Zhang, J. Huang, Y. Huang, Y. Yu, H. Li, and D. Metaxas. Automatic image annotation using group sparsity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pages 3312–3319, 2010.
51. Z. Zhang, X. Liang, and Y. Ma. Unwrapping low-rank textures on generalized cylindrical surfaces. In *IEEE International Conference on Computer Vision, ICCV 2011*, pages 1347–1354, 2011.
52. J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust Kalman filter. In *International Conference on Computer Vision, ICCV 2003*, pages 44–50, 2003.